Solución de las ecuaciones de Maxwell en la materia en medios homogéneos e isotrópicos, en una, dos y tres dimensiones, implementando el método de diferencias finitas en el dominio del tiempo

Autor
Stiven Orlando Melo Vargas

Tutor
César Aurelio Herreño Fierro

Modalidad monografía presentada en cumplimiento para obtener el título de Licenciado en Física

Universidad Distrital Francisco José de Caldas
Facultad de ciencias y educación
Proyecto curricular de Licenciatura en física
02 de junio de 2023

Agradecimientos

Quiero expresar mis más sinceros agradecimientos al profesor Cesar Aurelio Herreño Fierro, quien aporto de manera significativa en mi formación como licenciado en física, asimismo, por la orientación brindada durante el desarrollo de este trabajo para obtener mi título de licenciado en física.

Índice

Índice de figuras

Índice de tablas

Resumen

Al estudiar la propagación de campos electromagnéticos en un punto arbitrario del espacio, se recurre a las ecuaciones de Maxwell que relacionan los campos eléctricos y magnéticos considerando variaciones temporales y espaciales. La dificultad radica en la geometría del problema, haciendo que la solución de este conjunto de ecuaciones presente dificultad y no sea posible obtener una solución numérica, ni lograr representar visualmente la solución de los campos electromagnéticos. Es por eso, que se implementa la simulación computacional como una alternativa que permita realizar una descripción del campo electromagnético en diversas geometrías y medios.

El método de diferencias finitas en el dominio del tiempo (FDTD) es una técnica de modelado electrodinámico computacional; al ser un método que depende del tiempo, las soluciones cubren un amplio rango de frecuencias al ejecutar una sola simulación. Las ecuaciones de Maxwell en la materia dependientes del tiempo, para el caso de la propagación de ondas electromagnéticas en medios homogéneos e isotrópicos, se discretizan a través del método (FDTD), usando aproximaciones de diferencia central en las derivadas temporales y espaciales. Las ecuaciones resultantes se resuelven mediante la implementación del lenguaje de programación Python, desarrollado un algoritmo que permita obtener una solución numérica del problema, logrando visualizaciones animadas del comportamiento físico de las ondas electromagnéticas en una, dos y tres dimensiones. El conjunto de ecuaciones discretizadas tendrá condiciones de frontera de un conductor perfectamente eléctrico, y condiciones de frontera absorbente en el límite computacional, con el fin de encontrar la aplicabilidad en diferentes tipos de problemas, y obtener una solución numérica. Las aplicaciones que ofrecen este método son diversas, permitiendo resolver diferentes tipos de problemas en múltiples disciplinas de la ciencia con la cual sea posible estudiar sistemas y encontrar una solución cuantificable de los fenómenos relacionados con los campos electromagnéticos.

Palabras Clave

Electromagnético, campos, FDTD, Python, animadas, frontera, dimensiones, computacional, Maxwell.

Abstract

When studying the propagation of electromagnetic fields at an arbitrary point in space, Maxwell's equations are used to relate the electric and magnetic fields, considering both temporal and spatial variations. The difficulty lies in the geometry of the problem, making it challenging to find a numerical solution and visually represent the electromagnetic field solution. That's why computational simulation is implemented as an alternative to describe the electromagnetic field in various geometries and media.

The Finite Difference Time Domain (FDTD) method is a computational electromagnetics modeling technique. Being a time-dependent method, the solutions cover a wide range of frequencies in a single simulation. The time-dependent Maxwell's equations for the propagation of electromagnetic waves in homogeneous and isotropic media are discretized using the FDTD method, employing central difference approximations for temporal and spatial derivatives. The resulting equations are solved by implementing the Python programming language and developing an algorithm to obtain a numerical solution to the problem, enabling animated visualizations of the physical behavior of electromagnetic waves in one, two, and three dimensions. The set of discretized equations will have boundary conditions of a perfectly electrically conducting conductor and absorbing boundary conditions at the computational limit, aiming to achieve applicability in different types of problems and obtain a numerical solution. This method offers various applications, allowing the resolution of different types of problems in multiple scientific disciplines, where it is possible to study systems and obtain quantifiable solutions for phenomena related to electromagnetic fields.

Keywords

Electromagnetic, fields, FDTD, Python, animated, boundary, dimensions, computational, Maxwell.

1. Introducción

Debido a la dificultad de obtener y representar la solución de las ecuaciones de Maxwell en diferentes geometrías y entornos, es necesario recurrir a una alternativa que permita la solución de este conjunto de ecuaciones. La alternativa que logra facilitar la solución de este conjunto de ecuaciones son los métodos numéricos. Uno de ellos es el método de diferencias finitas en el dominio del tiempo (FDTD). Es una técnica de discretización, que permite realizar diversas descripciones del campo electromagnético en diversas geometrías y medios, considerando condiciones de frontera en una, dos y tres dimensiones, para el caso de un medio homogéneo e isotrópico.

El método consiste en discretizar las ecuaciones de Maxwell en la parte temporal y la parte espacial por medio del método de diferencias finitas en el dominio de tiempo mediante el uso de las celdas de Yee [1]. Las ecuaciones de diferencias finitas en el dominio del tiempo resultantes se solucionan implementando el lenguaje de programación Python de una manera a gran escala: las componentes del vector de campo eléctrico en un determinado espacio se resuelven en un instante dado de tiempo; luego, los componentes del vector del campo magnético en el mismo espacio se resuelven en el siguiente instante en el tiempo, el proceso se repite una y otra vez hasta que el comportamiento deseado del campo electromagnético ha evolucionado por completo obteniendo así una solución numérica, y brinde, además, una representación gráfica cuadro por cuadro del comportamiento físico de las ondas electromagnéticas, aplicando condiciones de frontera de un conductor perfectamente eléctrico y condiciones de frontera absorbente en el límite computacional. Posterior a esto, con los resultados obtenidos en la simulación numérica en una dimensión, se calculan las frecuencias de corte de una guía de onda unidimensional.

2. Justificación

Existen fenómenos electromagnéticos que presentan un desafío en la obtención de soluciones analíticas, siendo en algunos casos prácticamente imposible. La dificultad de encontrar soluciones a las ecuaciones de Maxwell en medios homogéneos e isotrópicos, en diferentes medios y geometrías, ha llevado al desarrollo de nuevas tecnologías para simular estos fenómenos. Sin embargo, algunas de las herramientas de simulación electromagnética no son de acceso libre o no permiten la personalización de las características de los campos electromagnéticos en consideración. En este sentido, es importante destacar el valor del desarrollo de recursos con licencias libres y documentación adecuada, lo que no solo permite personalizar, mejorar y complementar los recursos existentes, sino también impactar los procesos de aprendizaje de los estudiantes y profesionales en formación a nivel de pregrado. De esta manera, se fomenta la democratización del conocimiento y se incentiva la investigación en el campo de la simulación electromagnética.

Este trabajo tiene como objetivo desarrollar simulaciones electromagnéticas mediante el método de diferencias finitas en el dominio del tiempo, lo que permite obtener una solución numérica y una representación gráfica de la evolución temporal de los campos electromagnéticos en una, dos y tres dimensiones, teniendo en cuenta las condiciones de frontera en el límite computacional. Los resultados obtenidos se presentarán como una guía para futuras investigaciones en el campo de la electrodinámica computacional. Además, se busca promover el libre acceso y distribución de los códigos desarrollados, lo que permitirá que cualquier investigador pueda utilizarlos y adaptarlos a sus necesidades específicas. De esta manera, se pretende fomentar la colaboración y el intercambio de conocimientos en el campo de la simulación electromagnética, lo que puede resultar en nuevas y valiosas contribuciones a la comunidad científica en general.

3. Objetivos

Obtener simulaciones numéricas para el estudio de la propagación de ondas electromagnéticas en medios homogéneos e isotrópicos, por medio del método de diferencias finitas en el dominio del tiempo.

3.1. Objetivos específicos

1. Implementar el método de diferencias finitas en el dominio del tiempo para discretizar las ecuaciones de Maxwell en la materia en una, dos y tres dimensiones.

2. Resolver las ecuaciones discretizadas mediante la implementación del lenguaje de programación Python.

3. Obtener simulaciones numéricas para la representación gráfica de las ecuaciones de Maxwell en una, dos y tres dimensiones, aplicando condiciones de frontera de un conductor perfectamente eléctrico, y condiciones de frontera absorbente.

4. Cómo aplicación de las simulaciones numéricas obtenidas en una dimensión, se calcula las frecuencias de corte de una guía de onda en una dimensión.

4. Teoría electromagnética

El presente capítulo tiene como fin introducir los conceptos básicos de la teoría electromagnética, enfocándose principalmente en las ecuaciones de Maxwell.

4.1. Ecuaciones de Maxwell

Las ecuaciones de Maxwell son el resultado de un análisis minucioso de los fenómenos observables referente a la electricidad y magnetismo. Estas describen las propiedades fundamentales de los campos electromagnéticos [2].

En el caso de medios materiales homogéneos, las ecuaciones de Maxwell lucen así:

$$\nabla \cdot \mathbf{D} = \rho, \tag{4.1}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{4.2}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{4.3}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t}. \tag{4.4}$$

Donde ρ es la densidad de carga, $\mathbf{D}$ es el vector desplazamiento eléctrico, $\mathbf{B}$ es el campo magnético, $\mathbf{E}$ es el campo eléctrico y $\mathbf{H}$ es la intensidad del campo magnético. Las relaciones entre estos campos se describen a continuación.

4.2. Efectos macroscópicos de la interacción electromagnética

Los efectos macroscópicos de la incidencia de campo electromagnético sobre un material induce una polarización $\mathbf{P}$ y una magnetización $\mathbf{M}$. Un cuerpo material en presencia de un campo eléctrico externo induce un desplazamiento de las cargas internas, en consecuencia, se induce un dipolo eléctrico en el material que se encuentra distribuido sobre el volumen de todo el cuerpo. A partir de esto, se define la densidad volumétrica de dipolos eléctricos por medio del vector polarización dieléctrica $\mathbf{P}$. Análogo a esto, la presencia de un campo magnético externo sobre un material induce una magnetización $\mathbf{M}$, definida como la densidad de dipolos magnéticos en el material. [3]

La relación entre $\mathbf{D}$ y $\mathbf{H}$ con la polarización y magnetización, respetivamente, se define así:

- Vector desplazamiento:

$$\mathbf{D} \equiv \epsilon \mathbf{E} + \mathbf{P},$$

$$\mathbf{D} = \epsilon_0 \left[1 + \chi_E\right] \mathbf{E},$$

$$\mathbf{D} = \epsilon \mathbf{E}. \tag{4.5}$$

- Campo magnético:

$$\mathbf{B} = \mu_0 \left[1 + \chi_B\right] \mathbf{H},$$

$$\mathbf{H} = \frac{1}{\mu} \mathbf{B}. \tag{4.6}$$

Donde ϵ y μ son la permitividad eléctrica y la permeabilidad magnética del medio, respectivamente. Además, se define χ_E como la susceptibilidad eléctrica, y χ_B la susceptibilidad magnética.

4.3. Clasificación de los medios materiales

Al considerar un medio material como un continuo, es necesario clasificar los medios en torno a sus propiedades eléctricas y/o ópticas.

1. **Medios conductores:** Su conductividad eléctrica es muy cercana a cero.
2. **Medios isotrópicos:** La respuesta del medio no depende de la dirección de incidencia de la onda electromagnética.
3. **Medio homogéneo:** Las propiedades del medio son uniformes, por lo tanto, se mantiene invariante cualquier punto que se mida del medio.
4. **Medio absorbente:** El índice de refracción del medio se considera una cantidad compleja.

4.4. Condiciones de frontera

Existen algunas condiciones que se deben cumplir en un medio o un entorno, vinculado con las funciones dieléctricas de los materiales, de modo que satisfagan las ecuaciones de Maxwell en los dos medios. Al aplicar las ecuaciones de Maxwell a una frontera separada por dos medios continuos de diferentes propiedades eléctricas y/o magnéticas, se encuentra la continuidad o discontinuidad de las componentes tangenciales y normales a la superficie de la frontera para el campo eléctrico y el campo magnético.

- **Condición de frontera para la componente normal del campo eléctrico y campo magnético:** Se toma un cilindro de base circular como se representa en la Fig. 4.1, se busca calcular el flujo que atraviesa la superficie del cilindro, tomando el infinitesimal cuando $dL \to 0$.

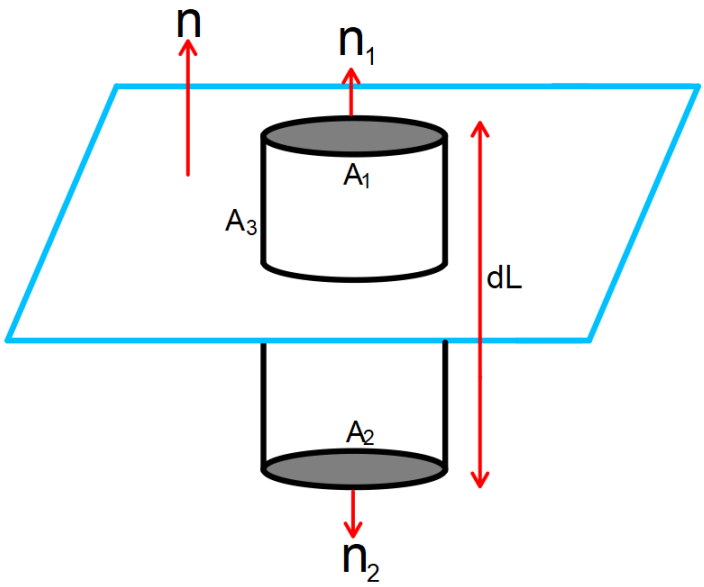

Figura 4.1: Condición de frontera para la componente normal.

Para la ley de Gauss en la materia en su forma integral, donde la carga encerrada hace referencia a la densidad de carga superficial σ, teniendo así la siguiente ecuación:

$$\oint_s \mathbf{D} \cdot d\mathbf{A} = Q_{enc} = \int\int \sigma dA.$$

Las áreas del cilindro A_1 y A_2 corresponde a las tapas del cilindro y A_3 el área lateral, reescribiendo la integral de flujo para las tres contribuciones:

$$\int\int_{A_1} \mathbf{D}_1 \cdot d\mathbf{A}_1 + \int\int_{A_2} \mathbf{D}_2 \cdot d\mathbf{A}_2 + \int\int_{A_3} \mathbf{D}_3 \cdot d\mathbf{A}_3 = \int\int \sigma dA.$$

Debido a la geometría del cilindro $A_1 = A_2 = A$ y para el área lateral $dA_3 = 2\pi r dL$, pero dado que se busca la condición de frontera cuando $dL \to 0$, el área lateral tiende a cero. Además, la orientación de los vectores normales a la superficie cumplen la relación $\mathbf{n}_1 = -\mathbf{n}_2 = \mathbf{n}$, teniendo así:

$$\int\int_A \mathbf{D}_1 \cdot \mathbf{n} dA - \int\int_A \mathbf{D}_2 \cdot \mathbf{n} dA = \int\int \sigma dA,$$

sumando sobre el mismo elemento de área, queda finalmente:

$$[\mathbf{D}_1 - \mathbf{D}_2] \cdot \mathbf{n} = \sigma. \tag{4.7}$$

Por lo tanto, la componente normal del vector desplazamiento eléctrico sufre una discontinuidad al cruzar la superficie que viene siendo igual a la densidad de carga superficial en dicho punto de la superficie.

De manera equivalente para la segunda ecuación de Maxwell, se obtiene una continuidad para la componente normal de la inducción magnética, es decir, no cambia al cruzar la frontera entre los medios:

$$[\mathbf{B}_1 - \mathbf{B}_2] \cdot \mathbf{n} = 0. \tag{4.8}$$

- **Condición de frontera para la componente tangencial del campo eléctrico y campo magnético:** Se toma una curva cerrada, representada en la Fig. 4.2, se busca calcular la circulación bajo la condición $dL \to 0$.

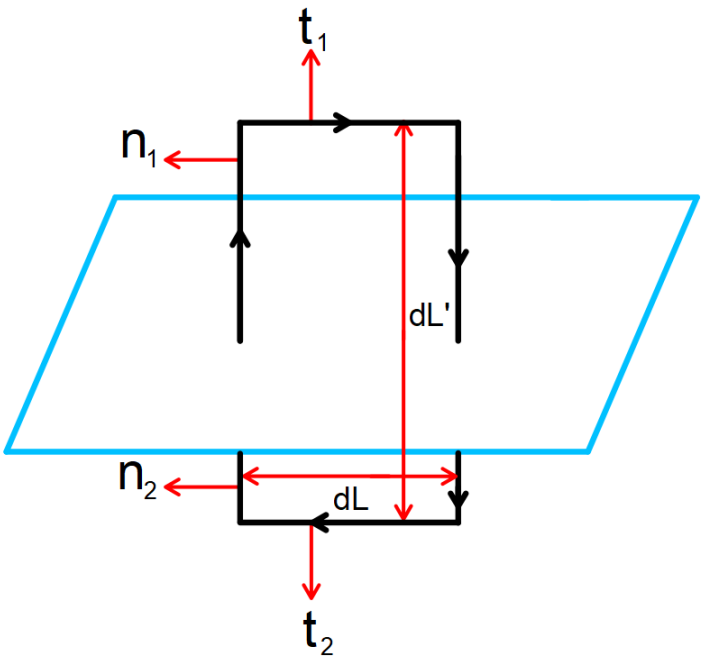

Figura 4.2: Condición de frontera para la circulación.

Para la ley de Ampere-Maxwell en su forma integral, tomando la densidad de corriente superficial $\mathbf{K}$; dado que se considera únicamente la corriente que pasa por la superficie, está dada por:

$$\oint_L \mathbf{H} \cdot d\mathbf{L} = \int \mathbf{K} \cdot d\mathbf{L}' + \frac{\partial}{\partial t} \int \int \mathbf{D} \cdot d\mathbf{A}.$$

La curva cerrada se compone por lados paralelos dL' y lados perpendiculares dL a la superficie. Se toman los vectores $\mathbf{n}_i$ como vectores normales o perpendiculares y t_i como vectores tangenciales o paralelos. Los vectores normales están relacionados bajo la condición geométrica $\mathbf{n}_1 = -\mathbf{n}_2$ y las tangenciales $\mathbf{t}_1 = -\mathbf{t}_2$. El área de la curva $dA = dLdL'$. A partir de esto, se calculan las cuatro contribuciones para la circulación:

$$\oint \mathbf{H}_1 \cdot dl' \mathbf{n}_1 + \oint \mathbf{H}_2 \cdot dl' \mathbf{n}_3 + \oint \mathbf{H}_3 \cdot dl' \mathbf{t}_1 + \oint \mathbf{H}_4 \cdot dl' \mathbf{t}_2 = \int \mathbf{K} \cdot d\mathbf{L'} + \frac{\partial}{\partial t} \int \int \mathbf{D} \cdot d\mathbf{A}.$$

Aplicando la condición infinitesimal $dL \to 0$:

$$\oint [\mathbf{H}_1 - \mathbf{H}_2] \cdot t dl' = \int K dl',$$

sumando sobre el mismo elemento, queda finalmente la ecuación:

$$[\mathbf{H}_1 - \mathbf{H}_2] \cdot \mathbf{t} = K. \tag{4.9}$$

Por lo tanto, la componente tangencial del campo magnético sufre una discontinuidad al cruzar la superficie que viene siendo igual a la densidad de corriente superficial en dicho punto de la superficie.

De manera equivalente para la Ley de Faraday-Lenz, se obtiene la continuidad de la componente tangencial del vector campo eléctrico cuando pasa de una frontera a otra:

$$[\mathbf{E}_1 - \mathbf{E}_2] \cdot \mathbf{t} = 0. \tag{4.10}$$

4.5. Ondas electromagnéticas

Al desacoplar las cantidades vectoriales del campo eléctrico y campo magnético presentes en las ecuaciones de Maxwell, se llega a una ecuación diferencial homogénea de segundo grado tanto para el campo eléctrico como para el campo magnético. Para llegar a esta ecuación diferencial, se considerará la propagación de los campos electromagnéticos en el vacío.

4.5.1. Ondas electromagnéticas en el vacío

Al considerar el vacío, la polarización y la magnetización son cero. Además, la permitividad y permeabilidad, ahora son ϵ_0 y μ_0, respectivamente. Las ecuaciones de Maxwell para el vacío vienen dadas por:

$$\left.\begin{array}{ll} \text{(i)} \quad \nabla \cdot \mathbf{E} = 0, & \text{(iii)} \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \\ \text{(ii)} \quad \nabla \cdot \mathbf{B} = 0, & \text{(iv)} \quad \nabla \times \mathbf{B} = \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t}. \end{array}\right\} \tag{4.11}$$

Para desacoplar las ecuaciones, se hace uso de la identidad vectorial para el rotacional de un rotacional [2]. Se aplica el rotacional a la ecuación (iii) y (iv), además, se tiene en cuenta el resultado (i) y (ii) del conjunto de ecuaciones 4.11:

$$\nabla \times (\nabla \times \mathbf{E}) = \nabla (\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E} = \nabla \times \left(-\frac{\partial \mathbf{B}}{\partial t}\right)$$

$$= -\frac{\partial}{\partial t} (\nabla \times \mathbf{B}) = \epsilon_0 \mu_0 \frac{\partial^2 \mathbf{E}}{\partial t^2},$$

$$\nabla \times (\nabla \times \mathbf{B}) = \nabla (\nabla \cdot \mathbf{B}) - \nabla^2 \mathbf{B} = \nabla \times \left(\epsilon_0 \mu_0 \frac{\partial \mathbf{E}}{\partial t}\right)$$

$$= \epsilon_0 \mu_0 \frac{\partial}{\partial t} (\nabla \times \mathbf{E}) = -\epsilon_0 \mu_0 \frac{\partial^2 \mathbf{B}}{\partial t^2}.$$

Finalmente, se obtiene una ecuación diferencial de segundo orden, tanto para el campo eléctrico como para el magnético:

$$\nabla^2 \mathbf{E} = \epsilon_0 \mu_0 \frac{\partial^2 \mathbf{E}}{\partial t^2}, \tag{4.12}$$

$$\nabla^2 \mathbf{B} = \epsilon_0 \mu_0 \frac{\partial^2 \mathbf{B}}{\partial t^2}. \tag{4.13}$$

Los resultados anteriores son similares a la ecuación clásica de una onda, cuya rapidez de propagación viene dada por la relación entre la permitividad y permeabilidad en el vacío; valor que corresponde con la rapidez de la luz:

$$c = \frac{1}{\sqrt{\epsilon_0 \mu_0}}. \tag{4.14}$$

4.5.2. Solución a la ecuación de onda

El siguiente paso es encontrar una solución para las ecuaciones de Maxwell y además, que satisfaga las condiciones de frontera. Para eso, se propone una de las soluciones más simples para las ecuaciones 4.12 y 4.13, que vienen dadas por ondas planas expresadas en funciones complejas:

$$\tilde{\mathbf{E}} = \tilde{\mathbf{E}}_0 e^{i(\mathbf{k}\cdot\mathbf{r} - \omega t)}, \tag{4.15}$$

$$\tilde{\mathbf{B}} = \tilde{\mathbf{B}}_0 e^{i(\mathbf{k}\cdot\mathbf{r} - \omega t)}. \tag{4.16}$$

Donde $\tilde{\mathbf{E}}_0$ y $\tilde{\mathbf{B}}_0$ son las amplitudes complejas, $\tilde{\mathbf{E}}$ y $\tilde{\mathbf{B}}$ son las componentes reales de los campos eléctricos y magnéticos. Además, la solución propuesta tiene como argumento

k que es el vector de onda y es quien da la dirección de propagación de la onda, ω es la frecuencia angular de la onda y t el tiempo.

Al tomar esta solución e introducirla en las ecuaciones de Maxwell, se llega a dos resultados:

1. No hay componente del campo eléctrico **E** y campo magnético **B** en dirección de propagación de la onda, por lo tanto, son ondas transversales.

2. El campo eléctrico **E** y el campo magnético **B** son mutuamente perpendiculares y se encuentran en fase:
$$\tilde{\mathbf{B}} = \frac{\mathbf{k} \times \mathbf{E}}{\omega}. \tag{4.17}$$

4.6. Guía de ondas

En la sección 4.5.2 se consideró la propagación de ondas planas de extensión infinita. Ahora, se consideran ondas electromagnéticas confinadas en el interior de una guía de ondas. A partir del supuesto de guías de ondas sin perdidas, se aplicarán las ecuaciones de Maxwell con las condiciones de frontera adecuadas para obtener diferentes modos de propagación de ondas y los campos **E** y **H**. Las paredes de esta guía de onda están constituidas por un conductor perfecto, de modo que dentro del material el campo eléctrico es cero. Las condiciones de contorno para una guía de onda vienen dadas por las ecuaciones 4.8 y 4.10.

El interés de este problema son las ondas monocromáticas que se propagan en la guía de onda. Se toma de nuevo la solución para ondas planas expresadas en funciones complejas:

$$\tilde{\mathbf{E}}(x, y, z, t) = \tilde{\mathbf{E}}_0(x, y)\, e^{i(kz - \omega t)}, \tag{4.18}$$

$$\tilde{\mathbf{B}}(x, y, z, t) = \tilde{\mathbf{B}}_0(x, y)\, e^{i(kz - \omega t)}. \tag{4.19}$$

A partir de esto, se debe encontrar las funciones $\tilde{\mathbf{E}}_0$ y $\tilde{\mathbf{B}}_0$ tales que los campos 4.18 y 4.19 satisfagan las ecuaciones 4.11. Para la ecuación (iii) de 4.11, se tiene el siguiente sistema de ecuaciones:

$$\left(\tfrac{\partial E_z}{\partial y} - \tfrac{\partial E_y}{\partial z}\right)\hat{\mathbf{x}} = -\tfrac{\partial B_x}{\partial t} \Rightarrow \tfrac{\partial E_z}{\partial y} - ikE_y = i\omega B_x,$$

$$\left(\tfrac{\partial E_x}{\partial z} - \tfrac{\partial E_z}{\partial x}\right)\hat{\mathbf{y}} = -\tfrac{\partial B_y}{\partial t} \Rightarrow ikE_x - \tfrac{\partial E_z}{\partial x} = i\omega B_y,$$

$$\left(\tfrac{\partial E_y}{\partial x} - \tfrac{\partial E_x}{\partial y}\right)\hat{\mathbf{z}} = \tfrac{\partial B_z}{\partial t} \Rightarrow \tfrac{\partial E_y}{\partial x} - \tfrac{\partial E_x}{\partial y} = i\omega B_z.$$

Haciendo el mismo análisis para la ecuación (iv), se obtiene el siguiente sistema de ecuaciones:

$$\left(\frac{\partial B_z}{\partial y} - \frac{\partial B_y}{\partial z}\right)\hat{\mathbf{x}} = \epsilon_0\mu_0\frac{\partial E_x}{\partial t} \Rightarrow \frac{\partial B_z}{\partial y} - ikB_y = -\frac{i\omega}{c^2}E_x,$$

$$\left(\frac{\partial B_x}{\partial z} - \frac{\partial B_z}{\partial x}\right)\hat{\mathbf{y}} = \epsilon_0\mu_0\frac{\partial E_y}{\partial t} \Rightarrow ikB_x - \frac{\partial B_z}{\partial x} = -\frac{i\omega}{c^2}E_y,$$

$$\left(\frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y}\right)\hat{\mathbf{z}} = \epsilon_0\mu_0\frac{\partial E_z}{\partial t} \Rightarrow \frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} = -\frac{i\omega}{c^2}E_z.$$

Ya con esto, se obtiene siguiente conjunto de ecuaciones acopladas para cada dirección de propagación de la onda plana tanto para el campo eléctrico como para el campo magnético:

$$\begin{aligned}
&\text{(i)} \quad \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = i\omega B_z, \quad \text{(iv)} \quad \frac{\partial B_y}{\partial x} - \frac{\partial B_x}{\partial y} = -\frac{i\omega}{c^2}E_z,\\
&\text{(ii)} \quad \frac{\partial E_z}{\partial y} - ikE_y = i\omega B_x, \quad \text{(v)} \quad \frac{\partial B_z}{\partial y} - ikB_y = -\frac{i\omega}{c^2}E_x,\\
&\text{(iii)} \quad ikE_x - \frac{\partial E_z}{\partial x} = i\omega B_y, \quad \text{(vi)} \quad ikB_x - \frac{\partial B_z}{\partial x} = -\frac{i\omega}{c^2}E_y.
\end{aligned} \tag{4.20}$$

El siguiente paso a seguir, es combinar las ecuaciones 4.20 para despejar las componentes x y y del campo eléctrico y campo magnético:

$$E_x = \frac{i}{\left(\frac{\omega^2}{c^2} - k^2\right)}\left(\omega\frac{\partial B_z}{\partial y} + k\frac{\partial E_z}{\partial x}\right), \tag{4.21}$$

$$E_y = \frac{i}{\frac{\omega^2}{c^2} - k^2}\left(k\frac{\partial E_z}{\partial y} - \omega\frac{\partial B_z}{\partial x}\right), \tag{4.22}$$

$$B_x = \frac{i}{\frac{\omega^2}{c^2} - k^2}\left(k\frac{\partial B_z}{\partial x} - \frac{\omega}{c^2}\frac{\partial E_z}{\partial y}\right), \tag{4.23}$$

$$B_y = \frac{i}{\frac{\omega^2}{c^2} - k^2}\left(k\frac{\partial B_z}{\partial y} + \frac{\omega}{c^2}\frac{\partial E_z}{\partial x}\right). \tag{4.24}$$

Del conjunto anterior de ecuaciones, se deducen los diferentes tipos de configuración de los campos, llamados modos:

- **Modos TE**: $E_z = 0$, $H_z \neq 0$. En este modo las componentes del campo eléctrico E_x y E_y son transversales a la dirección de propagación. Los campos se encuentran en modos eléctricos transversales.
- **Modos TM**: $E_z \neq 0$, $H_z = 0$. En este caso, el campo magnético $\mathbf{H}$ es transversal a la dirección de propagación de la onda, resultando en modos magnéticos transversales.

Ya se encontró un conjunto de ecuaciones que satisface las ecuaciones (iii) y (iv) de 4.11. Queda por demostrar lo mismo, pero esta vez para (i) y (ii) nuevamente de 4.11, con el fin de encontrar las componentes longitudinales E_z y B_z. Para (i) se obtiene la ecuación:

$$\frac{\partial^2 E_z}{\partial x^2} + \frac{\partial^2 E_z}{\partial y^2} + E_z\left(\frac{\omega^2}{c^2} - k^2\right) = 0. \tag{4.25}$$

Y para el caso de (ii):

$$\frac{\partial^2 B_z}{\partial x^2} + \frac{\partial^2 B_z}{\partial y^2} + B_z\left(\frac{\omega^2}{c^2} - k^2\right) = 0. \tag{4.26}$$

Cuando $E_z = 0$ de la ecuación 4.25 son llamadas ondas transversales eléctricas (TE) y cuando $B_z = 0$ de 4.26 son llamadas ondas transversales magnéticas (TM).

4.6.1. Ondas transversales magnéticas en una guía de ondas rectangulares

Se considera una guía de onda rectangular con altura a y ancho b Fig. 4.3, donde se propaga una onda transversal magnética en dirección z positiva. El interior de esta guía de onda está constituida por un material perfectamente conductor, de modo que el problema consiste en resolver la ecuación 4.25 sujeta a la condición de contorno 4.10.

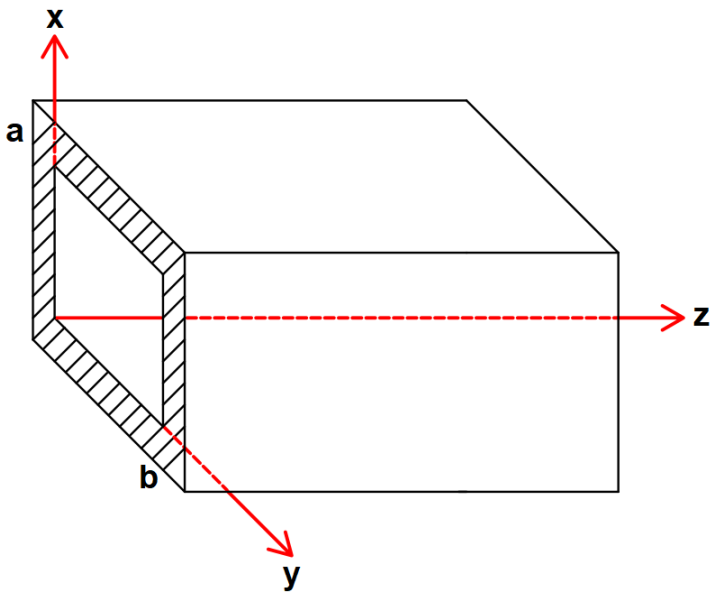

Figura 4.3: Guía de onda rectangular.

El método para resolver este problema, es por medio del método de separación de variables para la solución de una ecuación diferencial de segundo orden, efectuando el siguiente cambio de variables [3]:

$$E_z(x,y) = X(x)\,Y(y). \tag{4.27}$$

Reemplazando este cambio de variable en la ecuación 4.25 se obtiene:

$$\frac{1}{X(x)}\frac{\partial^2 X(x)}{\partial x^2} + \frac{1}{Y(y)}\frac{\partial^2 Y(y)}{\partial y^2} + \left(\frac{\omega^2}{c^2} - k^2\right) = 0. \tag{4.28}$$

A partir de esta ecuación, se hace una separación de variable para los términos que solo dependen de las variables x e y, para desacoplar el sistema de ecuaciones y expresar una solución para cada componente:

$$\begin{aligned} \tfrac{1}{X(x)}\tfrac{\partial^2 X(x)}{\partial x^2} &= -K_x^2, \\ \tfrac{1}{Y(y)}\tfrac{\partial^2 Y(y)}{\partial y^2} &= -K_y^2. \end{aligned} \tag{4.29}$$

Resolviendo esta ecuación diferencial homogénea de segundo orden:

$$X(x) = Asen(K_x x) + Bcos(K_x x),$$

$$Y(y) = Csen(K_y y) + Dcos(K_y y).$$

Dado que el propósito es resolver las ondas transversales magnéticas dentro de una guía de onda constituida por un material perfectamente eléctrico (PEC), se deben aplicar las condiciones de contorno para una onda transversal magnética, sabiendo que en este caso $E_z \neq 0$ y $B_z = 0$:

$$\text{cuando} \quad E_x = 0 \quad para \quad x = 0 \quad y \quad x = a,$$

$$\text{cuando} \quad E_y = 0 \quad para \quad y = 0 \quad y \quad y = b.$$

Obteniendo una solución del tipo:

$$E_z = E_0 sen\left(\tfrac{m\pi x}{a}\right) sen\left(\tfrac{n\pi y}{b}\right),$$

$$K_x = \tfrac{m\pi}{a}\,(m = 0, 1, 2, ..), \tag{4.30}$$

$$K_y = \tfrac{n\pi}{b}\,(n = 0, 1, 2, ..).$$

Reemplazando los valores de K_x y K_y de 4.30 en 4.28, y despejando el valor de la frecuencia, sabiendo la relación entre rapidez angular y la frecuencia $\omega = 2\pi f$:

$$f = \frac{c}{2}\sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2}. \tag{4.31}$$

Con esta expresión 4.31, es posible calcular la frecuencia de corte de una guía de ondas para diferentes modos de propagación.

5. Método de diferencias finitas en el dominio del tiempo

El presente capítulo tiene como fin abordar los conceptos relacionados con el método de diferencias finitas en el dominio del tiempo, la celda de Yee, los pulsos para la excitación temporal y las condiciones que debe satisfacer una simulación computacional para evitar errores numéricos.

5.1. Técnica numérica

Existen fenómenos en los que no es posible obtener una solución analítica, esto principalmente se debe a la geometría o medio que se esté analizando y/o el conjunto de ecuaciones que represente el fenómeno, que por lo general está constituido por ecuaciones diferenciales. La alternativa que permite la solución de algunos de estos fenómenos son mediante técnicas numéricas.

Una de estas técnicas numéricas es el método de diferenciales finitas en el dominio del tiempo. Esta técnica numérica consiste en tomar inicialmente la ecuación diferencial de tipo continúa y resolverla de forma aproximada mediante la aproximación por derivadas centradas, cuya solución es un conjunto finito de puntos localizados en el dominio de la solución, pasando de un problema continuo a un problema discreto.

5.2. Aproximación de los operadores diferenciales mediante diferencias finitas

Los métodos de diferencias finitas se fundamentan en el desarrollo de una función en serie de potencias, permitiendo predecir el comportamiento de una función cualquiera conociendo únicamente el valor de la función y de todas sus derivadas en un punto. Se desarrolla una función con incrementos a través de las series de Taylor [4], para una función monoevaluada, finita y continua, $f(x)$:

$$F(x+\alpha\Delta x) = F(x) + \frac{\alpha\Delta x}{1!}\frac{\mathrm{d}F(x)}{\mathrm{d}x} + \frac{(\alpha\Delta x)^2}{2!}\frac{\mathrm{d}^2F(x)}{\mathrm{d}x^2} + ... + \frac{(\alpha\Delta x)^n}{n!}\frac{\mathrm{d}^nF(x)}{\mathrm{d}x^n}. \quad (5.1)$$

Esta función evalúa la función en el punto $(x+\alpha\Delta x)$ a partir de los valores de la función. Sí se aproxima la primera derivada mediante diferencia finitas, tomando $\alpha = \frac{1}{2}$:

$$F\left(x+\frac{\Delta x}{2}\right) - F(x) = \frac{\Delta x}{2}\left(\frac{\mathrm{d}F(x)}{\mathrm{d}x} + \left(\frac{\Delta x}{2}\right)\frac{1}{2!}\frac{\mathrm{d}^2F(x)}{\mathrm{d}x^2} + \left(\frac{\Delta x}{2}\right)^2\frac{\mathrm{d}^3F(x)}{\mathrm{d}x^3} + ...\right).$$

Reduciendo la expresión anterior se llega a:

$$\frac{F\left(x+\frac{\Delta x}{2}\right)-F\left(x\right)}{\frac{\Delta x}{2}}=\frac{\mathrm{d}F\left(x\right)}{\mathrm{d}x}+\left(\frac{\Delta x}{2}\right)\frac{1}{2!}\frac{\mathrm{d}^2F\left(x\right)}{\mathrm{d}x^2}+\left(\frac{\Delta x}{2}\right)^2\frac{\mathrm{d}^3F\left(x\right)}{\mathrm{d}x^3}+...$$

Sí llamamos $G\left(\Delta x\right)$ a los elementos después de la segunda derivada:

$$G\left(\Delta x\right)=\left(\frac{\Delta x}{2}\right)\frac{1}{2!}\frac{\mathrm{d}^2F\left(x\right)}{\mathrm{d}x^2}+\left(\frac{\Delta x}{2}\right)^2\frac{\mathrm{d}^3F\left(x\right)}{\mathrm{d}x^3}+...$$

Se obtiene una expresión reducida:

$$\frac{\mathrm{d}F\left(x\right)}{\mathrm{d}x}=\frac{F\left(x+\frac{\Delta x}{2}\right)-F\left(x\right)}{\frac{\Delta x}{2}}+G\left(\Delta x\right).$$

Si Δx es muy pequeño para que $G\left(\Delta x\right)$ se anule, se encuentra la primera derivada de una función. Llegando finalmente a la siguiente expresión:

$$\frac{\mathrm{d}F\left(x\right)}{\mathrm{d}x}\approx\frac{F\left(x+\frac{\Delta x}{2}\right)-F\left(x\right)}{\frac{\Delta x}{2}}. \tag{5.2}$$

Haciendo el mismo procedimiento anterior, pero en este caso tomando $\alpha=-\frac{1}{2}$, se obtiene un resultado similar:

$$\frac{\mathrm{d}F\left(x\right)}{\mathrm{d}x}\approx\frac{-F\left(x-\frac{\Delta x}{2}\right)+F\left(x\right)}{\frac{\Delta x}{2}}. \tag{5.3}$$

Si se juntan las dos soluciones por derecha, y por izquierda, se llega a una ecuación de aproximación de la primera derivada mediante diferencias finitas. Permitiendo así obtener una aproximación de valores por izquierda y por derecha en el punto a analizar, como se muestra en la ecuación 5.4.

$$\frac{\mathrm{d}F\left(x\right)}{\mathrm{d}x}=\frac{F\left(x+\Delta x\right)-F\left(x-\Delta x\right)}{\Delta x}. \tag{5.4}$$

5.3. Celda de Yee

En 1966 Kane Yee [1] propuso un conjunto de ecuaciones en diferencias finitas para resolver las ecuaciones de Maxwell. Para ello, propone un cubo en donde en el centro de sus aristas se evalúa el campo eléctrico y en el centro de sus caras se evalúa el campo magnético. Esto resulta en dos redes cúbicas desplazadas una con respecto a la otra que garantizan que cada nodo donde se evalúa una componente del campo eléctrico o campo magnético, este rodeado por cuadro componente de los mismos campos Fig. 5.1. De esta forma, partiendo de las condiciones iniciales impuestas, el método permite calcular la evolución temporal del campo electromagnético en el punto de interés [5].

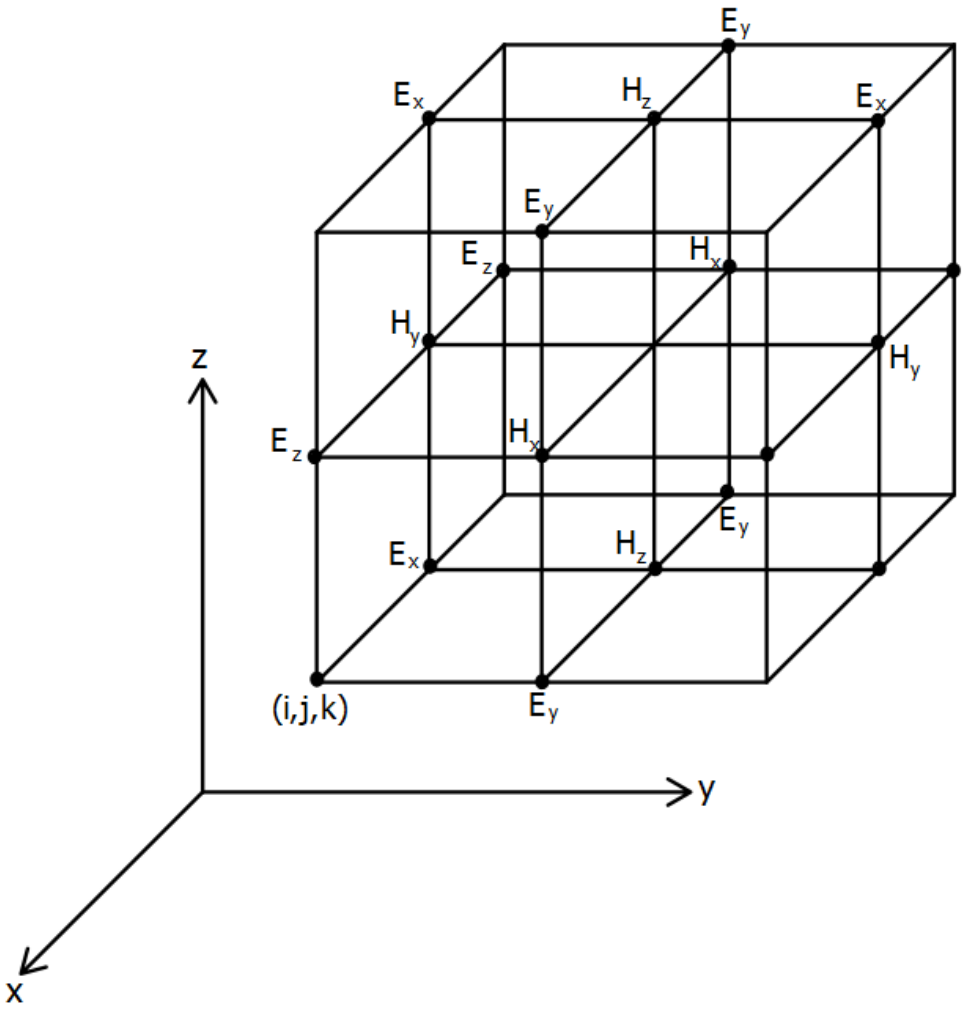

Figura 5.1: Celda de Yee en tres dimensiones.

La región tridimensional tiene origen en las coordenadas $(i, j, k) = (i\Delta x, j\Delta y, k\Delta z,)$, donde Δx, Δy y Δz son los incrementos espaciales. Para la variación del tiempo $t = n\Delta t$, donde n representa un instante de tiempo dado y Δt el incremento temporal. Cada función de tiempo y espacio se escribirá cómo:

$$F\left(i\Delta x, j\Delta y, k\Delta z, n\Delta t\right) = F^n\left(i, j, k\right). \tag{5.5}$$

Teniendo esta función, se pueden expresar las derivas temporales y espaciales, utilizando una aproximación en diferencias finitas centradas, aplicando el resultado obtenido en la ecuación 5.4. Para la parte espacial, se tiene el siguiente sistema de ecuaciones, suponiendo que la variación espacial por derecha y por izquierda se da en la componente x de la celda de Yee:

$$\frac{\partial F^n(i,j,k)}{\partial x} = \frac{F^n\left(i+\frac{1}{2},j,k\right) - F^n\left(i-\frac{1}{2},j,k\right)}{\Delta x}, \tag{5.6}$$

Para el caso temporal, se analiza el salto temporal tanto por izquierda como por derecha en cualquier punto del espacio, obteniendo así la siguiente ecuación:

$$\frac{\partial F^n(i,j,k)}{\partial t} = \frac{F^{n+\frac{1}{2}}(i,j,k) - F^{n-\frac{1}{2}}(i,j,k)}{\Delta t}. \tag{5.7}$$

Al utilizar este conjunto de ecuaciones, es posible actualizar tanto el campo eléctrico como el campo magnético en diferentes instantes de tiempo. En particular, se actualiza el campo eléctrico en el instante n y el campo magnético en el instante $n+\frac{1}{2}$.

Con el uso de la primera derivada mediante diferencias finitas, se calcula el valor de cada nueva componente del campo en cada punto de la celda de Yee, de modo que depende únicamente de su valor anterior y de los valores anteriores de las otras componentes del campo en puntos adyacentes. De esa forma, en un tiempo dado, el cálculo de una componente del campo puede calcularse en un punto a la vez o en p puntos a la vez. El algoritmo tiene la siguiente secuencia:

1. Inicialmente, el campo eléctrico y campo magnético en todo el dominio es cero.
2. Inicializar las condiciones de frontera y todas las matrices de los campos eléctricos y magnéticos.
3. Se calcula el campo eléctrico y se imponen las condiciones de contorno.
4. Se impone la excitación.
5. Se calcula el vector campo magnético y se imponen las condiciones de contorno.
6. Se incrementa el tiempo en un $n = n + \Delta n$.

En la Fig. 5.2 se representa esta secuencia en una diagrama de flujo.

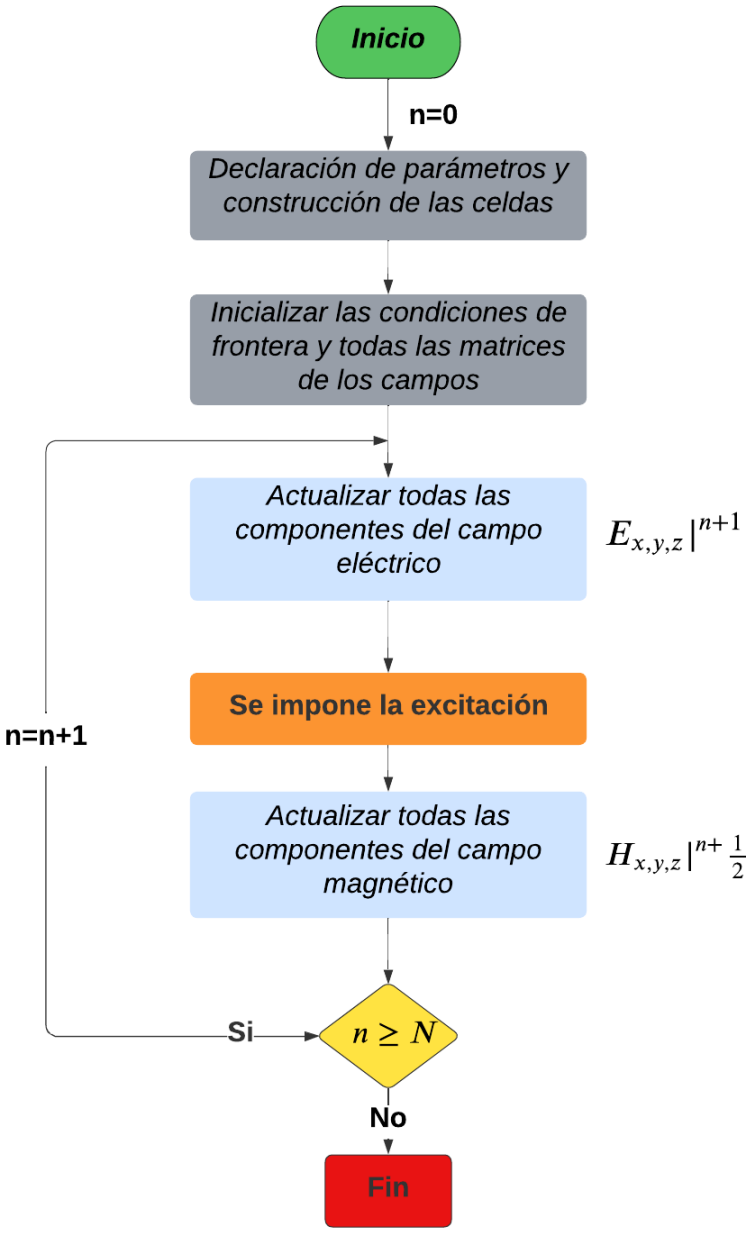

Figura 5.2: Diagrama de flujo para el proceso de pasos de tiempo estándar del método de diferencias finitas en el dominio del tiempo.

5.4. Pulsos para la excitación temporal

Se modela el proceso de excitación en el espacio computacional haciendo uso de pulsos temporales. Este tipo de excitación permite experimentar con gran variedad de pulsos finitos a través de diferentes funciones matemáticas. El proceso de iteración del

método de diferencias finitas en el dominio del tiempo se mantiene hasta que todo el campo introducido en el pulso se ha dispersado, saliendo por los contornos hasta el límite computacional que tiene diferentes parámetros. El uso de pulsos tiene la ventaja obtener resultados en un amplio rango de frecuencias mediante una sola simulación, lo que permite calcular diferentes valores para la frecuencia de corte de una guía de ondas, tomando diferentes modos de propagación [6].

Para el desarrollo e implementación del método de diferencias finitas en el dominio del tiempo a la solución en la solución de las ecuaciones de Maxwell, se hace uso de las siguientes fuentes que simulan una onda electromagnética:

5.4.1. Pulso Gaussiano

Un pulso gaussiano tiene la forma de una función gaussiana, la cual puede ser ubicada en un punto cualquiera dentro del espacio computacional. El valor que toma el pulso gaussiano viene determinado por el vector campo eléctrico definido anteriormente por medio de la discretización de las ecuaciones de Maxwell. La ecuación característica de un pulso gaussiano es la siguiente:

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-b)^2}{2\sigma^2}}, \tag{5.8}$$

donde b determina la ubicación de la campana y σ es la desviación estándar que da el valor del ancho de la campana. En la Fig. 5.3 se representa un pulso gaussiano ubicado en el origen ($b = 0$) y cuya desviación estándar es ($\sigma = 0{,}1$).

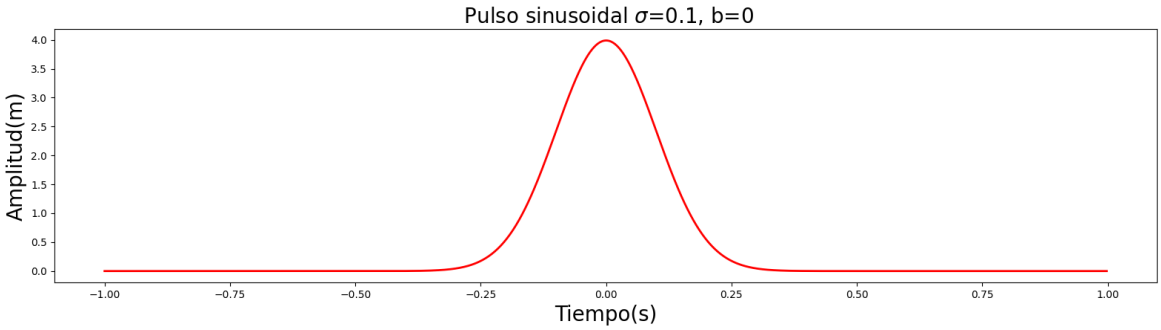

Figura 5.3: Pulso gaussiano en el dominio del tiempo con $\sigma = 0{,}1$ y $b = 0$.

5.4.2. Pulso sinusoidal

El pulso sinusoidal viene representado por una función trigonométrica sinusoidal. Matemáticamente se representa con la siguiente ecuación:

$$f(t) = A\sin(2\pi f t + \varphi), \tag{5.9}$$

donde A es la amplitud, f es la frecuencia y φ es la fase. En la Fig. 5.4 se representa un pulso sinusoidal con frecuencia $f = 1$ y amplitud $A = 1$.

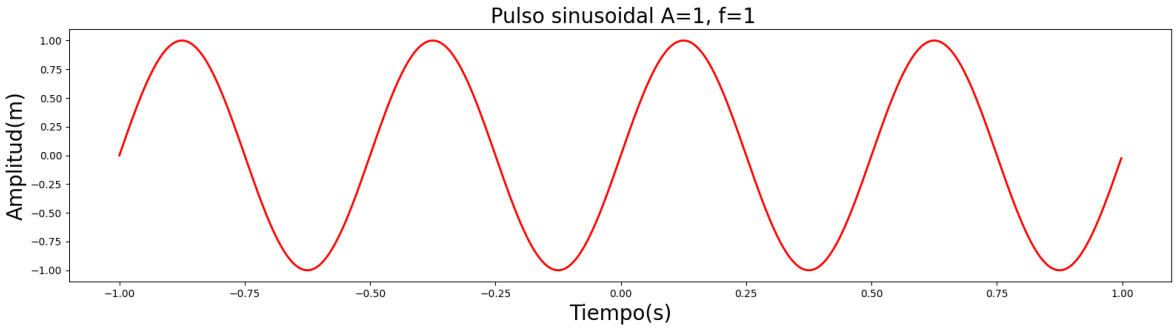

Figura 5.4: Pulso sinusoidal en el dominio del tiempo con $f = 1$ y $A = 1$.

5.4.3. Blackman–Harris window

Una señal Blackman–Harris window (BHW) es una función matemática cuyo comportamiento es cero fuera de algún intervalo elegido, normalmente simétrica alrededor de la mitad del intervalo, generalmente cerca de un máximo en el medio, y generalmente disminuyendo lejos del medio. Matemáticamente, se representa con la siguiente ecuación [7]:

$$S\left(n\right) = a_0 - a_1 cos\left(\frac{2\pi n}{N}\right) + a_2 cos\left(\frac{4\pi n}{N}\right) + a_3 cos\left(\frac{6\pi n}{N}\right) \tag{5.10}$$

Cuyos parámetros a_0, a_1, a_2 y a_3 tienen los siguientes valores:

$$a_0 = 0{,}35875$$
$$a_1 = 0{,}48829$$
$$a_2 = 0{,}14128$$
$$a_3 = 0{,}01168$$

La función Blackman–Harris window proporciona una señal periódica en el dominio del tiempo, dado que está conformada por funciones sinusoidales o cosenoidales, de modo que al aplicar una transformación rápida de Fourier se puede obtener un amplio rango de frecuencias de esta señal, desarrollando valores no nulos de la función. En la Fig. 5.5 y Fig. 5.6 se representa la señal Blackman–Harris window y su correspondiente transformación de Fourier, respectivamente.

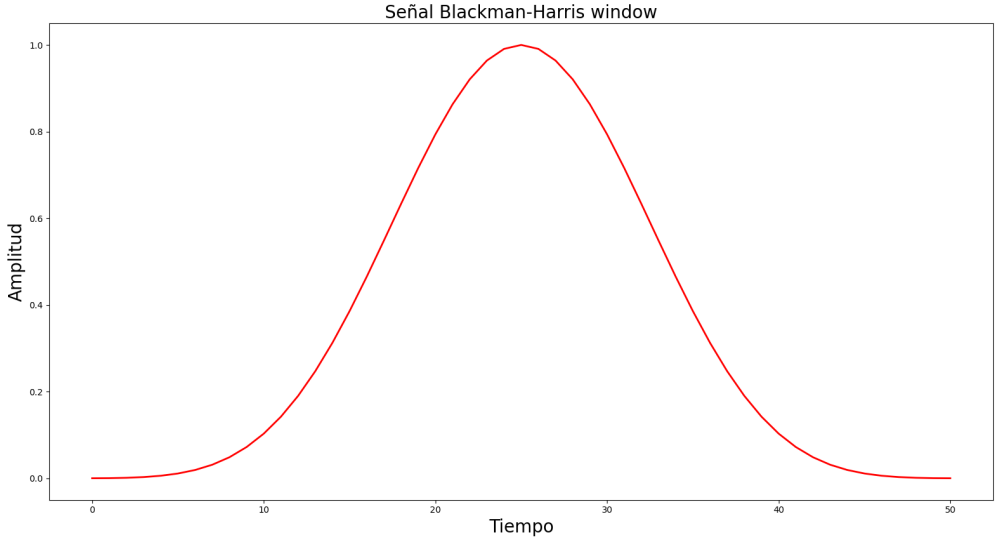

Figura 5.5: Señal Blackman-Harris window en el dominio del tiempo.

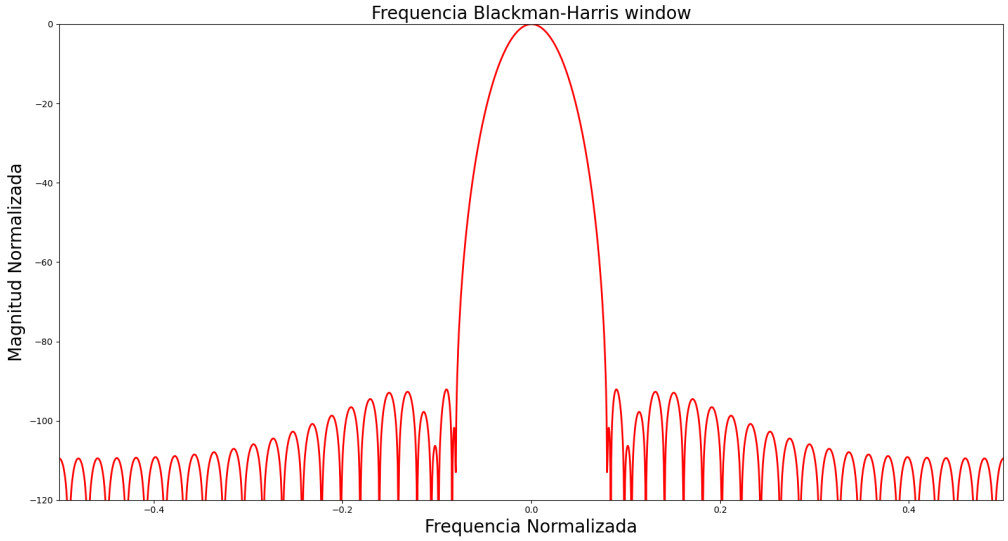

Figura 5.6: Frecuencia característica de la señal Blackman-Harris window.

5.5. Condiciones de simulación computacional

El algoritmo numérico propuesto en la sección 5.3 para resolver las ecuaciones de Maxwell, requiere ciertas condiciones de estabilidad. Una de estas debe cumplir que el paso temporal guarde una relación de proporcionalidad con los incrementos espaciales. Esta relación es necesaria para evitar la inestabilidad numérica, ya que esta lleva a errores de cálculo numéricos por cada iteración [8].
Si una onda se mueve en una cuadrícula discreta y se quiere calcular la amplitud en pasos discretos de igual duración, entonces la duración debe ser menor o igual que el tiempo para que la onda viaje en los puntos de la cuadrícula adyacente. Se debe cumplir un tiempo mínimo para que el pulso se propague a la rapidez de la luz c_0 en la malla discreta Δx, si el paso temporal es mayor, la onda ya habría sobrepasado la distancia Δx y ya no estaría en la malla de la simulación computacional. El criterio de estabilidad establece lo siguiente:

$$\Delta t \leq \frac{1}{c\sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}} \tag{5.11}$$

Para una simulación en dos dimensiones se debe cumplir $\Delta t = \frac{\Delta x}{\sqrt{2}c_0}$ y caso similar para tres dimensiones $\Delta t = \frac{\Delta x}{\sqrt{3}c}$. Este principio es conocido como condición de Courant [9], que establece la siguiente relación:

$$\Delta t = \frac{\Delta x}{\sqrt{d}c_0}$$

Donde d es el tamaño de la simulación. En este trabajo se toma el valor del tamaño de la simulación como $\sqrt{d} = 2$ de acuerdo con [8].

$$\Delta t = \frac{\Delta x}{2c_0} \tag{5.12}$$

Este resultado permite establecer que la onda que se propaga en el entorno computacional debe avanzar en el ordenador de modo que a lo sumo un paso temporal recorra una distancia infinitesimal, donde esta distancia está determinada por los valores que tome las discretización en x, y, z. Si esto no ocurre así, llevaría a múltiples errores numéricos y a la inestabilidad del mismo.

6. Condiciones de frontera

El presente capítulo tiene como fin introducir las condiciones de frontera del entorno computacional, diseñando las condiciones de frontera de un conductor perfectamente eléctrico y las condiciones de frontera absorbente mediante el diseño de capas perfectamente emparejadas.

6.1. Condiciones de frontera del entorno computacional

Es necesario considerar ahora las condiciones de frontera del entorno computacional para así modelar de forma correcta las ecuaciones de Maxwell por medio del método de diferencias finitas y así imponer diferentes características al límite computacional. Una vez que se propaga un pulso de diferente naturaleza dentro de la guía de onda, se hace una división de la red computacional por zonas Fig. 6.1:

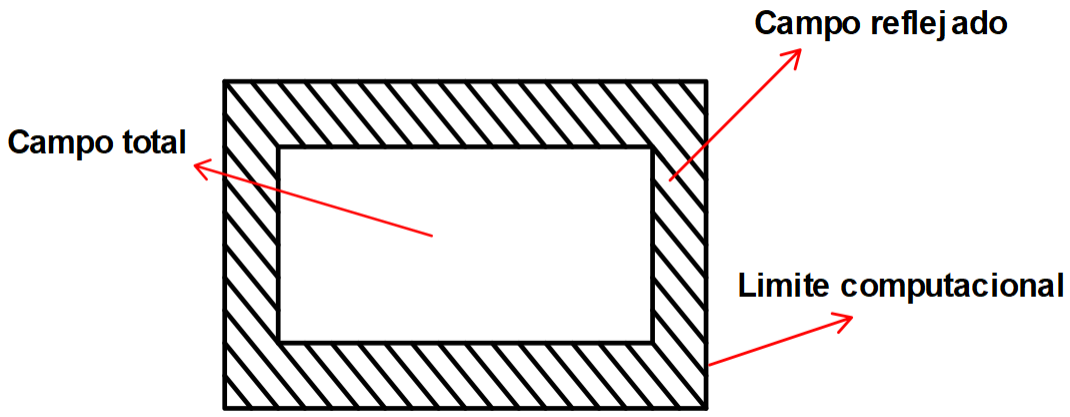

Figura 6.1: División de la red computacional.

- **Campo total:** En esta zona se encuentra la estructura sobre la que se analiza la dispersión por medio de la discretización de las ecuaciones de Maxwell en una, dos y tres dimensiones.
- **Campo reflejado:** Una vez que el pulso generado llega a la parte exterior de la estructura, este puede ser reflejado o absorbido parcial o totalmente, según sean las condiciones de frontera impuestas.
- **Límite computacional:** En esta zona no hay propagación de los campos electromagnéticos.

Teniendo en cuenta esto se plantean las condiciones de fronteras de un conductor perfecto y condiciones de frontera absorbente.

6.2. Condiciones de frontera PEC

Las condiciones de frontera PEC (perfect electric conductor) hacen alusión a un conductor perfectamente eléctrico. Los campos eléctricos en una superficie metálica y/o conductora se pueden descomponer en campos eléctricos tangenciales y campos eléctricos normales, como se puede ver en la Fig. 6.2, impuestas por las condiciones de frontera vistas en la sección 4.4.

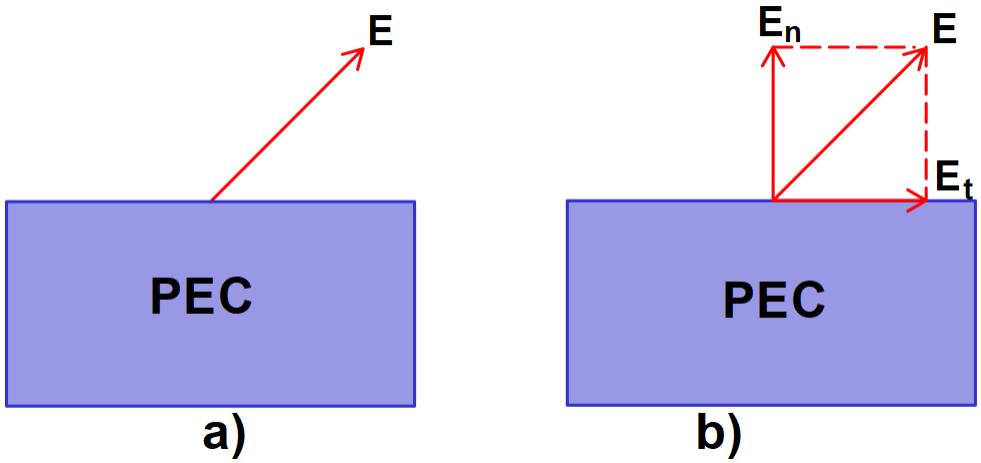

Figura 6.2: a) Campo eléctrico en la superficie de un PEC, b) Componente tangencial y normal del campo eléctrico.

Las características principales que se obtienen al descomponer las componentes del campo eléctrico son las siguientes:

- La componente tangencial del campo eléctrico siempre será cero en una superficie metálica ($E_t = 0$).
- La componente normal del campo eléctrico es diferente de cero en la superficie de un metal ($E_t \neq 0$).

De modo que, si en el límite computacional se imponen condiciones de frontera PEC, una vez que las ondas alcanzan la frontera, estas se reflejarán de nuevo hacia el interior del entorno computacional en su totalidad.

6.3. Condiciones de frontera PML

Las condiciones de frontera absorbentes o diseño de capa perfectamente combinada, PML (Perfectly matches layer) se basa en un diseño de un material con pérdidas artificiales en el límite computacional, llevando a cabo pérdidas de la onda electromagnética de manera independiente en cada una de las componentes del campo, permitiendo así, que no existan ondas reflejadas hacia el interior del entorno computacional [10].

Para generar las capas perfectamente absorbentes se tiene en cuenta que una discontinuidad entre dos medios produce una reflexión relacionada con las impedancias intrínsecas de cada uno, y este coeficiente de reflexión debe anularse para impedir que al llegar las ondas a la PML, tenga reflexiones hacia el interior del dominio computacional [11]. El coeficiente de reflexión entre dos medios, A y B se define cómo:

$$\Gamma = \frac{\eta_B - \eta_A}{\eta_B + \eta_A}. \tag{6.1}$$

Donde las impedancias intrínsecas de cada uno de los medios se definen a partir de la permitividad y la permeabilidad $\eta = \sqrt{\frac{\mu}{\epsilon}}$.
Al introducir estas capas adicionales en los límites del entorno computacional con el fin de simular un medio con pérdidas, se introduce tanto la permitividad y la permeabilidad en términos complejos para simular la pérdida. Para esto, se incorporarán constantes ficticias para la permitividad y permeabilidad en la dirección x y en la dirección y, asumiendo que la onda se propaga en dirección z positiva, aclarando que estas variables agregadas carecen de significado físico y no se relaciona con las características propias del medio [12].
Para hacer la construcción de las condiciones absorbente se tienen en cuenta las siguientes condiciones, aclarando que los términos que llevan el superíndice $(*)$ hacen referencia a la parte imaginaria de la permeabilidad y permitividad:

1. La impedancia vista desde la región de propagación al límite computacional donde se encuentra la PML debe mantenerse constante:

$$\eta_0 = \eta_m = \sqrt{\frac{\mu^*_{Fx}}{\epsilon^*_{Fx}}} = 1. \tag{6.2}$$

2. En la dirección perpendicular a la frontera, por ejemplo, en dirección x, la constante dieléctrica relativa y la permeabilidad magnética deben ser inversas a las mismas en otras direcciones, esto se puede expresar de la siguiente manera:

$$\epsilon^*_{Fx} = \frac{1}{\epsilon^*_{Fy}} \tag{6.3}$$

$$\mu^*_{Fx} = \frac{1}{\mu^*_{Fy}} \tag{6.4}$$

   Se define la constante dieléctrica y la permeabilidad magnética imaginaria, esto con el fin de considerar medios con perdidas, ya que la parte imaginaria de las constantes dieléctricas hacen referencia a medios con perdidas, donde σ es la densidad superficial de carga y $m = x, y$:

$$\epsilon^*_{Fm} = \epsilon_{Fm} + \frac{\sigma_{Dm}}{i\omega\epsilon_0}, \tag{6.5}$$

$$\mu^*_{Fm} = \mu_{Fm} + \frac{\sigma_{Hm}}{i\omega\epsilon_0}. \tag{6.6}$$

Aplicando las condiciones impuestas, se obtiene:

$$\epsilon_{Fm} = \mu_{Fm} = 1,$$

$$\frac{\sigma_{Dm}}{\epsilon_0} = \frac{\sigma_{Hm}}{\mu_0} = \frac{\sigma_D}{\epsilon_0},$$

$$\eta_0 = \eta_m = \sqrt{\frac{\mu^*_{Fx}}{\epsilon^*_{Fx}}} = \sqrt{\frac{1 + \frac{\sigma_D(x)}{j\omega\epsilon_0}}{1 + \frac{\sigma_D(x)}{j\omega\epsilon_0}}} = 1. \tag{6.7}$$

Teniendo así una relación que permite generar capas ficticias en el límite computacional para que la onda sea absorbida y no regrese hacia el entorno computacional [13].

7. Discretización de las ecuaciones de Maxwell en medios materiales

El presente capítulo tiene como fin presentar la discretización de las ecuaciones de Maxwell en la materia en una, dos y tres dimensiones, empleando el método de diferencias finitas en el dominio del tiempo, aplicando condiciones de frontera de un conductor perfectamente eléctrico y condiciones de frontera absorbente.

7.1. Normalización gaussiana

Antes de discretizar las ecuaciones de Maxwell se introduce una normalización gaussiana para el vector desplazamiento eléctrico y el campo eléctrico, esto con el propósito de que los valores del campo eléctrico $\mathbf{E}$ y la intensidad del campo magnético $\mathbf{H}$ no difieran en su valor debido a la diferencia en el orden de magnitud de las constantes ϵ_0 y μ_0:

$$\mathbf{E} = \sqrt{\frac{\mu_0}{\epsilon_0}}\ \tilde{\mathbf{E}}, \tag{7.1}$$

$$\mathbf{D} = \sqrt{\epsilon_0 \mu_0}\ \tilde{\mathbf{D}}. \tag{7.2}$$

Se reescriben las ecuaciones de Maxwell 4.3 y 4.4:

$$\frac{1}{\sqrt{\epsilon_0 \mu_0}} \nabla \times \tilde{\mathbf{E}} = -\frac{\partial \mathbf{H}}{\partial t}, \tag{7.3}$$

$$\frac{1}{\sqrt{\epsilon_0 \mu_0}} \nabla \times \mathbf{H} = \frac{\partial \tilde{\mathbf{D}}}{\partial t}. \tag{7.4}$$

Donde $\tilde{\mathbf{E}}$ y $\tilde{\mathbf{D}}$ es el campo eléctrico y vector desplazamiento eléctrico normalizado, respectivamente.

7.2. Ecuaciones onda unidimensional: condiciones PEC

Se inicia considerando una onda electromagnética que se propaga en dirección z positiva con el campo eléctrico orientado en dirección x y el campo magnético orientado en dirección y. El límite computacional estará constituido por un conductor perfecto, de modo que cuando llegue la onda electromagnética esta se refleja con la misma magnitud pero con fase opuesta a la inicial. Las ecuaciones de Maxwell normalizadas en una dimensión son las siguientes:

$$\frac{\partial \tilde{E}_x}{\partial t} = -\frac{1}{\sqrt{\epsilon_0 \mu_0}} \frac{\partial H_y}{\partial z}, \tag{7.5}$$

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\sqrt{\epsilon_0 \mu_0}} \frac{\partial \tilde{E}_x}{\partial z}. \tag{7.6}$$

La ecuación 7.5 y 7.6 describen la ley de Ampere-Maxwell y la ley de Faraday-Lenz, respetivamente. Además, debe estar presente que las ecuaciones 4.1 y 4.2 satisfacen este sistema de ecuaciones de forma trivial.

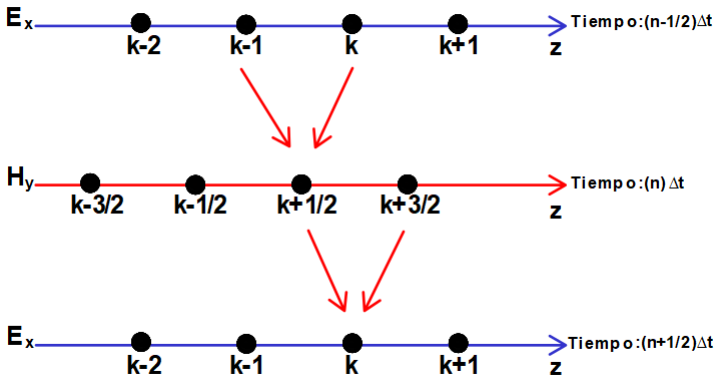

Figura 7.1: Celda de Yee en una dimensión para una onda que se propaga en dirección z.

Para discretizar las ecuaciones de Maxwell en una dimensión se toma como referencia la celda de Yee en una dimensión Fig. 7.1. Por ejemplo, si se quiere calcular el valor de H_y se necesitan los valores de E_x en $k-1$ y k. De igual forma, si se quiere encontrar el valor de E_x se hace a partir de los valores de H_y en $k+\frac{1}{2}$ y $k+\frac{3}{2}$. Este esquema es conocido como "algoritmo de salto de rana" [5]. Discretizando la ecuación 7.5 y 7.6 se obtiene respectivamente:

$$\tilde{E}_x^{n+\frac{1}{2}}(k) = \tilde{E}_x^{n-\frac{1}{2}}(k) - \frac{\Delta t}{\sqrt{\epsilon_0\mu_0}\Delta x}\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right],$$

$$H_y^{n+1}\left(k+\frac{1}{2}\right) = H_y^n\left(k+\frac{1}{2}\right) - \frac{\Delta t}{\sqrt{\epsilon_0\mu_0}\Delta x}\left[\tilde{E}_x^{n+\frac{1}{2}}(k+1) - \tilde{E}_x^{n+\frac{1}{2}}(k)\right].$$

Donde cabe aclarar que el tamaño de la celda es igual para las tres direcciones, es decir, $\Delta x = \Delta y = \Delta z$. Del criterio de estabilidad descrito en la sección 5.5, que describe la relación entre el paso temporal, el tamaño de la celda y la rapidez de la luz, pueden relacionarse de la siguiente manera para simplificar las expresiones anteriores:

$$\frac{\Delta t}{\sqrt{\epsilon_0\mu_0}\Delta x} = \frac{\Delta x}{2c_0}\frac{1}{\sqrt{\epsilon_0\mu_0}\Delta x} = \frac{1}{2}. \tag{7.7}$$

De modo que las ecuaciones discretizadas en una dimensión se expresan de la siguiente manera:

$$\tilde{E}_x^{n+\frac{1}{2}}(k) = \tilde{E}_x^{n-\frac{1}{2}}(k) - 0{,}5\left[H_y^n\left(k+\frac{1}{2}\right) - H_y^n\left(k-\frac{1}{2}\right)\right], \tag{7.8}$$

$$H_y^{n+1}\left(k+\frac{1}{2}\right) = H_y^n\left(k+\frac{1}{2}\right) - 0{,}5\left[\tilde{E}_x^{n+\frac{1}{2}}(k+1) - \tilde{E}_x^{n+\frac{1}{2}}(k)\right]. \tag{7.9}$$

La ecuación 7.8 y 7.9 describen una onda electromagnética en una dimensión que se propaga en dirección z y cuyas condiciones de frontera son las de un conductor perfecto.

7.3. Ecuaciones de onda unidimensional: condiciones de frontera absorbente PML

El siguiente paso es implementar las condiciones de frontera absorbente al límite computacional. Las ecuaciones discretizadas en este caso son iguales a las ecuaciones 7.8 y 7.9, lo único que debe modificar son las condiciones en el límite computacional. Como se vio en la Fig. 7.1 para conocer el valor del campo en un punto es necesario conocer los valores de los puntos vecinos tanto por izquierda como por derecha. Para hacer este arreglo consideremos un espacio computacional compuesto por 50 celdas de tamaño Δx. Este espacio inicia en el punto 0 y termina en el punto 50. Cuando el pulso alcanza el punto 50, este debe actualizarse con el punto 49 y 51, pero como no se conoce el valor de ese punto 51 el algoritmo tendrá múltiples errores. Este problema puede solucionarse fácilmente asignando valores al campo en los puntos extremos. A partir del criterio de estabilidad de la ecuación 5.12 es posible establecer que se necesitan dos pasos de tiempo para que el campo cruce una celda, es decir:

$$Distancia = \frac{\Delta x}{2}.$$

Con esto, se puede plantear la siguiente relación:

$$E_x^n(0) = E_x^{n-2}(1). \tag{7.10}$$

Con la ecuación 7.10 se asigna un valor fijo del campo para los dos extremos computacionales (en el ejemplo anterior eran los puntos 0 y 50), almacenando un valor de $E_x(1)$ para dos pasos de tiempo y luego se le asigna esos valores a $E_x(0)$. Esta condición se implementa en ambos extremos de modo que la onda unidimensional sea absorbida en su totalidad [14].

7.4. Ecuaciones TM en dos dimensiones: condiciones PEC

Al considerar la propagación de ondas electromagnéticas en dos dimensiones, se puede elegir la orientación de propagación de la onda; esto con el fin, de considerar el campo transversal eléctrico TE y el campo transversal magnético TM. Si la onda se propaga en dirección z positiva, el modo transversal eléctrico se compone de los vectores $\tilde{E}_x$, $\tilde{E}_y$ y H_z y para el modo transversal magnético $\tilde{E}_z$, H_x y H_y. En este trabajo, se hace el análisis para el modo transversal magnético. Aplicando las condiciones TM para las ecuaciones de Maxwell en dos dimensiones, se obtiene:

$$\frac{\partial \tilde{D}_z}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right), \tag{7.11}$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\sqrt{\epsilon_0 \mu_0}} \frac{\partial \tilde{E}_z}{\partial y}, \tag{7.12}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \frac{\partial \tilde{E}_z}{\partial y}. \tag{7.13}$$

La ecuación 7.11 describe la ley de Ampere-Maxwell, mientras que las ecuaciones 7.12 y 7.13 describen la ley de Faraday-Lenz. Además, debe estar presente que la primera 4.1 y segunda 4.2 ecuación de Maxwell satisface este sistema de ecuaciones de forma trivial.

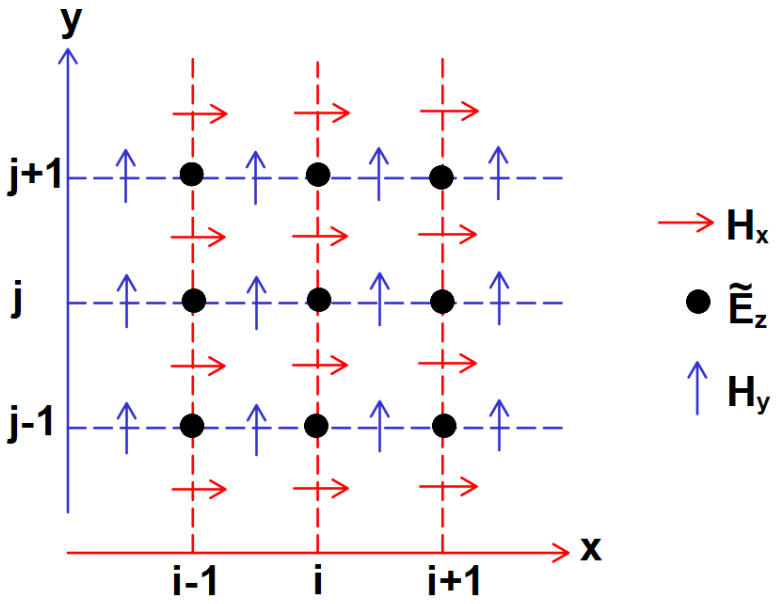

Figura 7.2: Celda de Yee en dos dimensiones para modo transversal magnético TM.

Para discretizar las ecuaciones de Maxwell en dos dimensiones mediante el método de diferencias finitas en el dominio del tiempo se toma como referencias la celda de Yee en dos dimensiones para el modo transversal magnético Fig. 7.2, donde se encuentran discretizados los campos eléctricos y campos magnéticos.
Para discretizar la ecuación 7.11 se toma el valor del desplazamiento eléctrico en el punto (i, j) y se procede a aproximar mediante la derivada central tanto por arriba como por los lados, obteniendo la ecuación:

$$\begin{gathered} \tilde{D}_z^{n+\frac{1}{2}}(i,j) = \tilde{D}_z^{n-\frac{1}{2}}(i,j) + 0{,}5 \\ \left[H_y^n\left(i+\tfrac{1}{2},j\right) - H_y^n\left(i-\tfrac{1}{2},j\right) - H_x^n\left(i,j+\tfrac{1}{2}\right) + H_x^n\left(i,j-\tfrac{1}{2}\right) \right]. \end{gathered} \tag{7.14}$$

De modo similar que la ecuación anterior, se aproxima la ecuación 7.12 y 7.13 analizando el valor de la componente del campo magnético en dirección x en el punto $\left(i, j+\frac{1}{2}\right)$ y para la componente del campo magnético en dirección y en el punto $\left(i+\frac{1}{2}, j\right)$, obteniendo las siguientes ecuaciones:

$$H_x^{n+1}\left(i, j+\frac{1}{2}\right) = H_x^n\left(i, j+\frac{1}{2}\right) + 0{,}5\left[\tilde{E}_z^{n+\frac{1}{2}}(i,j) - \tilde{E}_z^{n+\frac{1}{2}}(i,j+1)\right], \tag{7.15}$$

$$H_y^{n+1}\left(i+\frac{1}{2}, j\right) = H_y^n\left(i+\frac{1}{2}, j\right) + 0{,}5\left[\tilde{E}_z^{n+\frac{1}{2}}(i+1,j) - \tilde{E}_z^{n+\frac{1}{2}}(i,j)\right]. \tag{7.16}$$

El conjunto de ecuaciones 7.14 a 7.16 describe el modo transversal magnético de las ecuaciones de Maxwell para las condiciones de frontera de un conductor perfecto PEC.

7.5. Ecuaciones TM en dos dimensiones: condiciones de frontera absorbente PML

En la sección 7.4 se describieron las ecuaciones de Maxwell en dos dimensiones para el modo tranversal magnético aplicando condiciones de frontera de un conductor perfectamente eléctrico. El siguiente paso es tomar las ecuaciones 7.11 a 7.13, pero ahora considerando la permitividad y permeabilidad ficticia, de modo que se logre aplicar las condiciones de frontera absorbente PML descritas en la sección 6.3. El conjunto de ecuaciones se expresa de la siguiente manera:

$$i\omega\tilde{D}_z\epsilon_{Fz}^*(x)\,\epsilon_{Fz}^*(y) = c_0\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right), \tag{7.17}$$

$$i\omega H_x\mu_{Fx}^*(x)\,\mu_{Fx}^*(y) = -c_0\frac{\partial\tilde{E}_z}{\partial y}, \tag{7.18}$$

$$i\omega H_y\mu_{Fy}^*(x)\,\mu_{Fy}^*(y) = c_0\frac{\partial\tilde{E}_z}{\partial x}. \tag{7.19}$$

Reemplazando las condiciones impuestas en la ecuación 6.5 y 6.6 se llega al conjunto de ecuaciones expresadas en términos de la permitividad y permeabilidad compleja:

$$\begin{aligned}
i\omega\tilde{D}_z\left[1+\tfrac{\sigma_D(i)}{i\omega\epsilon_0}\right]\left[1+\tfrac{\sigma_D(j)}{i\omega\epsilon_0}\right] &= c_0\left(\tfrac{\partial H_y}{\partial x} - \tfrac{\partial H_x}{\partial y}\right), && (i)\\
i\omega H_x\left[1+\tfrac{\sigma_D(i)}{i\omega\epsilon_0}\right]^{-1}\left[1+\tfrac{\sigma_D(k)}{i\omega\epsilon_0}\right] &= -c_0\tfrac{\partial\tilde{E}_z}{\partial y}, && (ii)\\
i\omega H_y\left[1+\tfrac{\sigma_D(i)}{i\omega\epsilon_0}\right]\left[1+\tfrac{\sigma_D(j)}{i\omega\epsilon_0}\right]^{-1} &= c_0\tfrac{\partial\tilde{E}_z}{\partial x}. && (iii)
\end{aligned} \tag{7.20}$$

Para discretizar este conjunto de ecuaciones es necesario considerar el problema por direcciones, es decir, implementar la PML en la dirección x y luego en la dirección y, esto con el fin de facilitar el desarrollo matemático. La discretización del conjunto de

ecuaciones 7.20 se da a partir de la Fig. 7.2, donde se muestra la celda de Yee en dos dimensiones para el modo transversal magnético TM. Discretizando la ecuación (i) de 7.20 en dirección x y en dirección y haciendo unos pasos algebraicos para simplificar la expresión, se obtiene:

$$\begin{aligned}
\tilde{D}_z^{n+\frac{1}{2}}(i,j) &= \tilde{D}_z^{n-\frac{1}{2}}(i,j)\frac{\left[1-\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]} + \left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]^{-1} 0{,}5 \\
&\left[H_y^n\left(i+\tfrac{1}{2},j\right) - H_y^n\left(i-\tfrac{1}{2},j\right) + H_x^n\left(i,j-\tfrac{1}{2}\right) - H_x^n\left(i,j+\tfrac{1}{2}\right)\right], \qquad (i) \\
\tilde{D}_z^{n+\frac{1}{2}}(i,j) &= \tilde{D}_z^{n-\frac{1}{2}}(i,j)\frac{\left[1-\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]} + \left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]^{-1} 0{,}5 \\
&\left[H_y^n\left(i+\tfrac{1}{2},j\right) - H_y^n\left(i-\tfrac{1}{2},j\right) + H_x^n\left(i,j-\tfrac{1}{2}\right) - H_x^n\left(i,j+\tfrac{1}{2}\right)\right]. \qquad (ii)
\end{aligned} \tag{7.21}$$

Para la ecuación (ii) y (iii) de 7.20 en dirección x y en dirección y se hace el siguiente cambio de variable para simplificar la expresión de modo que sea más sencillo a la hora de implementar el código en Python:

$$\frac{\partial \tilde{E}_z}{\partial y} = \frac{\tilde{E}_z^{n+\frac{1}{2}}(i,j+1) - \tilde{E}_z^{n+\frac{1}{2}}(i,j)}{\Delta x} = -\frac{camp_e}{\Delta x}. \tag{7.22}$$

Para que la expresión 7.22 pueda aproximarse por derecha y por izquierda por medio del método de diferencias finitas en el dominio del tiempo, se hace el siguiente cambio de variable para la componente del campo H_x y H_y, respectivamente:

$$\begin{aligned}
I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right) &= I_{Hy}^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right) + camp_e, \\
I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j\right) &= I_{Hy}^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j\right) + camp_e.
\end{aligned} \tag{7.23}$$

Discretizando la ecuación (ii) de 7.20 tanto en dirección x como en dirección y se obtiene:

$$\begin{aligned}
H_x^{n+1}\left(i,j+\tfrac{1}{2}\right) &= H_x^n\left(i,j+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]} + \\
&\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1} 0{,}5\left[\tilde{E}_z^{n+\frac{1}{2}}(i,j+1) - \tilde{E}_z^{n+\frac{1}{2}}(i,j)\right], \qquad (i)
\end{aligned} \tag{7.24}$$

$$\begin{aligned}
H_x^{n+1}\left(i,j+\tfrac{1}{2}\right) &= H_x^{n+1}\left(i,j+\tfrac{1}{2}\right) + 0{,}5\cdot camp_e + \\
&\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\cdot I_{H_x}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right). \qquad (ii)
\end{aligned}$$

Discretizando la ecuación (iii) de 7.20 tanto en dirección x como en dirección y se obtiene:

$$H_y^{n+1}\left(i+\tfrac{1}{2},j\right)=H_y^n\left(i+\tfrac{1}{2},j\right)\frac{\left[1-\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+$$
$$\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}0{,}5\left[\tilde{E}_z^{n+\frac{1}{2}}\left(i+1,j\right)-\tilde{E}_z^{n+\frac{1}{2}}\left(i,j\right)\right],\qquad (i)\qquad (7.25)$$

$$H_y^{n+1}\left(i+\tfrac{1}{2},j\right)=H_y^{n+1}\left(i+\tfrac{1}{2},j\right)+0{,}5\cdot camp_e+$$
$$\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\cdot I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j\right).\qquad (ii)$$

Con el fin de simplificar el conjunto de ecuaciones discretizadas anteriormente, se toman los siguientes cambios de variable presentados en la Tabla 7.1 y Tabla 7.2 para la componente x y componente y de la PML.

Término	Expresión
$\alpha_1\left(i\right)$	$\frac{\left[1-\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]}$
$\alpha_2\left(i\right)$	$\left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]^{-1}$
$\beta_1\left(i+\frac{1}{2}\right)$	$\frac{\left[1-\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}$
$\beta_2\left(i+\frac{1}{2}\right)$	$\left[1+\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}$
$A\left(i\right)$	$\frac{\sigma_D(i)\Delta t}{2\epsilon_0}$

Tabla 7.1: Cambios de variable para reducir las expresiones discretizadas en la coordenada x de la PML.

Término	Expresión
$\alpha_1(j)$	$\frac{\left[1-\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]}$
$\alpha_2(j)$	$\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]^{-1}$
$\beta_1\left(j+\frac{1}{2}\right)$	$\frac{\left[1-\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}$
$\beta_2\left(j+\frac{1}{2}\right)$	$\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}$
$A(j)$	$\frac{\sigma_D(j)\Delta t}{2\epsilon_0}$

Tabla 7.2: Cambios de variable para reducir las expresiones discretizadas en la coordenada y de la PML.

Introduciendo estos cambios de variables y combinando la solución en dirección x y en dirección y, esto con el fin para construir una sola ecuación para los campos $\tilde{D}_z$, H_x y H_y. Para el conjunto de ecuaciones (i) y (ii) de 7.21, 7.24 y 7.25, respectivamente:

$$\begin{gathered}\tilde{D}_z^{n+\frac{1}{2}}(i,j)=\tilde{D}_z^{n-\frac{1}{2}}(i,j)\,\alpha_1(i)\,\alpha_1(j)+\alpha_2(i)\,\alpha_2(j)\,0{,}5\\ \left[H_y^n\left(i+\tfrac{1}{2},j\right)-H_y^n\left(i-\tfrac{1}{2},j\right)+H_x^n\left(i,j-\tfrac{1}{2}\right)-H_x^n\left(i,j+\tfrac{1}{2}\right)\right],\end{gathered} \tag{7.26}$$

$$camp_e=\left(\tilde{E}_z^{n+\frac{1}{2}}(i,j)-\tilde{E}_z^{n+\frac{1}{2}}(i,j+1)\right),$$

$$\begin{gathered}I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right)=I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right)+camp_e,\\ H_x^{n+1}\left(i,j+\tfrac{1}{2}\right)=\beta_1\left(j+\tfrac{1}{2}\right)H_x^{n+1}\left(i+\tfrac{1}{2},j\right)+\\ \beta_2\left(j+\tfrac{1}{2}\right)\left[0{,}5\cdot camp_e+A(i)\cdot I_{Hy}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2}\right)\right].\end{gathered} \tag{7.27}$$

$$
\begin{aligned}
camp_e &= \left(\tilde{E}_z^{n+\frac{1}{2}}(i,j) - \tilde{E}_z^{n+\frac{1}{2}}(i+1,j)\right), \\
I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j\right) &= I_{Hy}^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j\right) + camp_e, \\
H_y^{n+1}\left(i+\tfrac{1}{2},j\right) &= \beta_1\left(i+\tfrac{1}{2}\right) H_y^{n+1}\left(i+\tfrac{1}{2},j\right) \\
&-\beta_2\left(i+\tfrac{1}{2}\right)\left[0{,}5\cdot camp_e + A(j)\cdot I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j\right)\right].
\end{aligned}
\tag{7.28}
$$

Las ecuaciones 7.26 a 7.28 permiten tener conjuntamente las capas absorbentes en direcciones x y y para las componentes $\tilde{D}_z$, H_x y H_y respectivamente, de modo que cuando se propague una onda tranversal magnética dentro de una guía de onda y esta llegue al límite computacional no se produzcan reflexiones hacia el interior del entorno computacional.
Para el cálculo de los parámetros β y α se puede evidenciar que estos términos están en función de los parámetros $A(i)$ y $A(j)$. Para este trabajo no se consideran variaciones de la conductividad. Para calcular estos términos, se hace uso de un parámetro auxiliar construido por **Bérenger** [10], dado por la siguiente expresión:

$$x_n = \frac{\sigma_D \Delta x}{2\epsilon_0}, \tag{7.29}$$

este término se encontró empíricamente y puede ser calculado a través de la siguiente expresión:

$$x_n(i) = 0{,}333\left(\frac{i}{\text{Longitud de la PML}}\right)^3, \tag{7.30}$$

donde cabe destacar lo siguiente:

- El factor (0.333) de la ecuación 7.30 fue encontrado empíricamente por ser el número más grande que se mantenía estable en la aproximación numérica.
- Longitud de la PML: Es el número de capas que se disponen en el diseño de capas absorbentes.
- El factor cúbico de la expresión 7.30 fue encontrado de forma similar por ser la variación más efectiva encontrada.

En la Tabla 7.3 se presentan los cambios que sufren los parámetros α y β para terminar de constituir las condiciones de frontera absorbente, a partir de los resultados encontrados por **Bérenger** [10].

Término	Expresión
$\alpha_1(i)$	$\frac{1-xn(i)}{1+xn(i)}$
$\alpha_2(i)$	$\frac{1}{1+xn(i)}$
$\alpha_1(j)$	$\frac{1-xn(j)}{1+xn(j)}$
$\alpha_2(j)$	$\frac{1}{1+xn(j)}$
$A(i)$	$xn(i)$
$A(j)$	$xn(j)$

Tabla 7.3: Cambios de variable de los parámetros α y β a partir de los resultados obtenidos por Berenger.

Con el conjunto de ecuaciones encontradas para las ondas transversales magnéticas TM y garantizando las condiciones de frontera PML se puede garantizar que una vez que el pulso electromagnético llegue al límite computacional este no tenga ninguna reflexión hacia el interior, de modo que el pulso sea absorbido en su totalidad por las capas PML.

7.6. Ecuaciones en tres dimensiones: condiciones de frontera PEC

La solución a las ecuaciones de Maxwell en tres dimensiones con condiciones de frontera de un conductor perfecto es similar a la solución en dos dimensiones. Para construir la solución se tiene en cuenta la celda de Yee [1] presentada en la sección 5.3, donde se debe tener presente que en el centro de las aristas del cubo se evalúa el campo eléctrico y en el centro de las caras se evalúa el campo magnético. Las ecuaciones de Maxwell en tres dimensiones son las siguientes:

$$\frac{\partial \tilde{D}_x}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \right), \tag{7.31}$$

$$\frac{\partial \tilde{D}_y}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \right), \tag{7.32}$$

$$\frac{\partial \tilde{D}_z}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \right), \tag{7.33}$$

$$\frac{\partial H_x}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_y}{\partial z} - \frac{\partial \tilde{E}_z}{\partial y} \right), \tag{7.34}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_z}{\partial x} - \frac{\partial \tilde{E}_x}{\partial z} \right), \tag{7.35}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_x}{\partial y} - \frac{\partial \tilde{E}_y}{\partial x} \right). \tag{7.36}$$

Para facilitar la discretización de los campos en tres dimensiones se representan las celdas de Yee para cada componente del campo eléctrico y del campo magnético. La componente x del desplazamiento eléctrico viene dado por la ecuación 7.31, y se representa como se muestra en la Fig. 7.3.

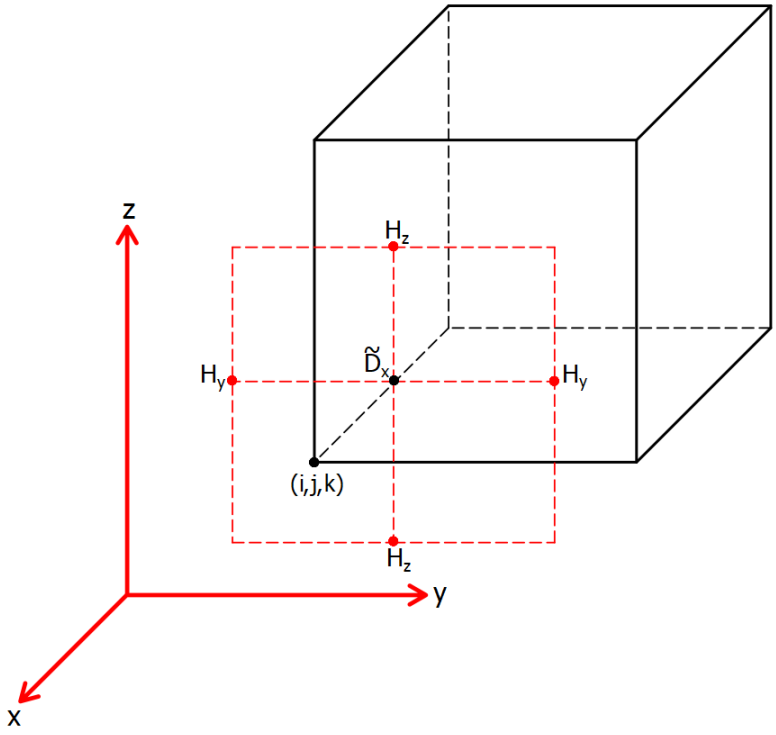

Figura 7.3: Celda de Yee para la componente $\tilde{D}_x$.

Discretizando la componente $\tilde{D}_x$ aproximando los valores H_z y H_y mediante el método de diferencias finitas en el dominio del tiempo se obtiene la siguiente ecuación:

$$\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)=\tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)+0{,}5\left[H_z^n\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)-H_z^n\left(i+\tfrac{1}{2},j-\tfrac{1}{2},k\right)-H_y^n\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)+H_y^n\left(i+\tfrac{1}{2},j,k-\tfrac{1}{2}\right)\right]. \tag{7.37}$$

La componente y del desplazamiento eléctrico viene dada por la ecuación 7.32, y se representa como se muestra en la Fig. 7.4.

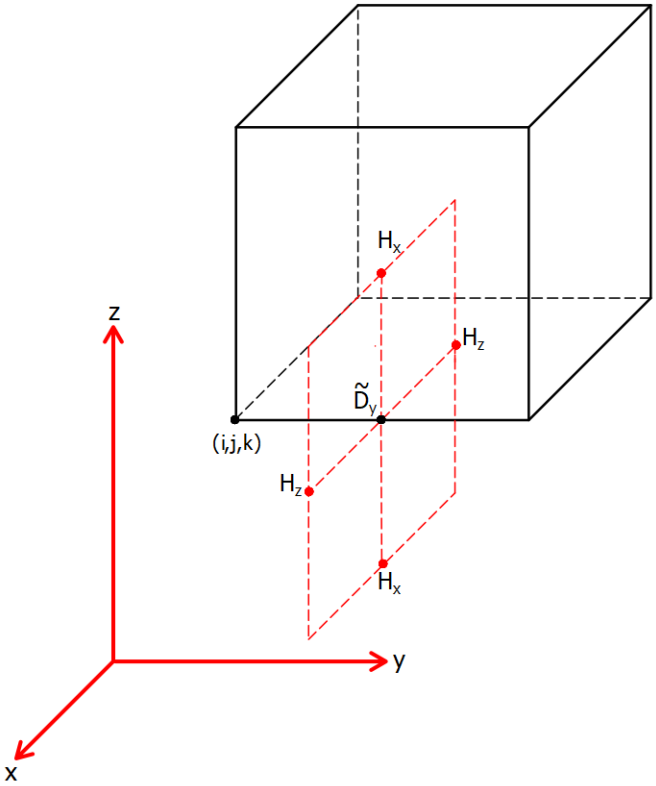

Figura 7.4: Celda de Yee para la componente $\tilde{D}_y$.

Discretizando la componente $\tilde{D}_y$ aproximando los valores de H_x y H_z se obtiene la siguiente ecuación:

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\tilde{D}_y^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)+0{,}5\left[H_x^n\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)-H_x^n\left(i,j+\tfrac{1}{2},k\right)-H_z^n\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)+H_z^n\left(i-\tfrac{1}{2},j+\tfrac{1}{2},k\right)\right]. \tag{7.38}$$

La componente z del desplazamiento eléctrico viene dada por la ecuación 7.33, y se representa como se muestra en la Fig. 7.5.

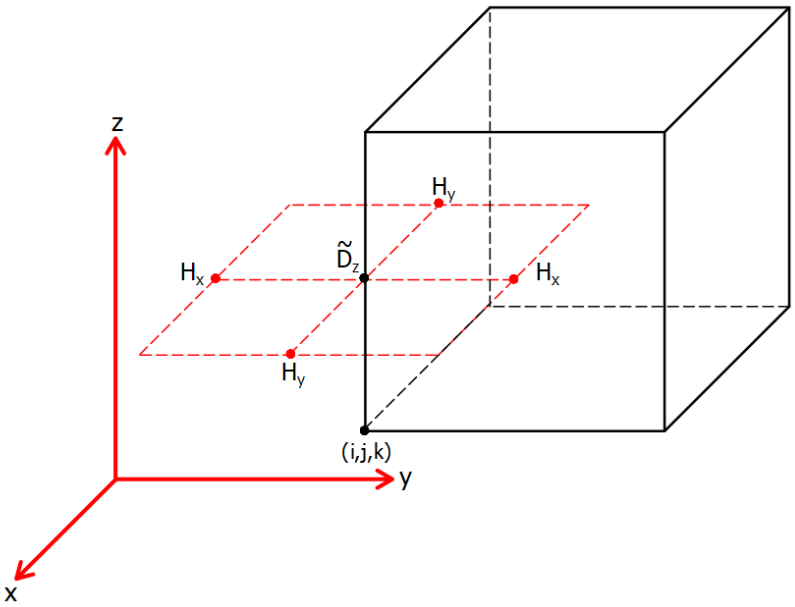

Figura 7.5: Celda de Yee para la componente $\tilde{D}_z$.

Discretizando la componente $\tilde{D}_z$ aproximando los valores de H_x y H_y se obtiene la siguiente ecuación:

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\tilde{D}_x^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)+0{,}5\left[H_y^n\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)-H_y^n\left(i-\tfrac{1}{2},j,k+\tfrac{1}{2}\right)-H_x^n\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)+H_x^n\left(i,j-\tfrac{1}{2},k+\tfrac{1}{2}\right)\right]. \tag{7.39}$$

Las ecuaciones discretizadas 7.37 a 7.39 representan las componentes $\tilde{D}_x$, $\tilde{D}_y$ y $\tilde{D}_z$ del vector desplazamiento eléctrico. El siguiente paso es discretizar las componentes del campo magnético. La componente x del campo magnético viene dada por la ecuación 7.34, y se representa como se muestra en la Fig. 7.6.

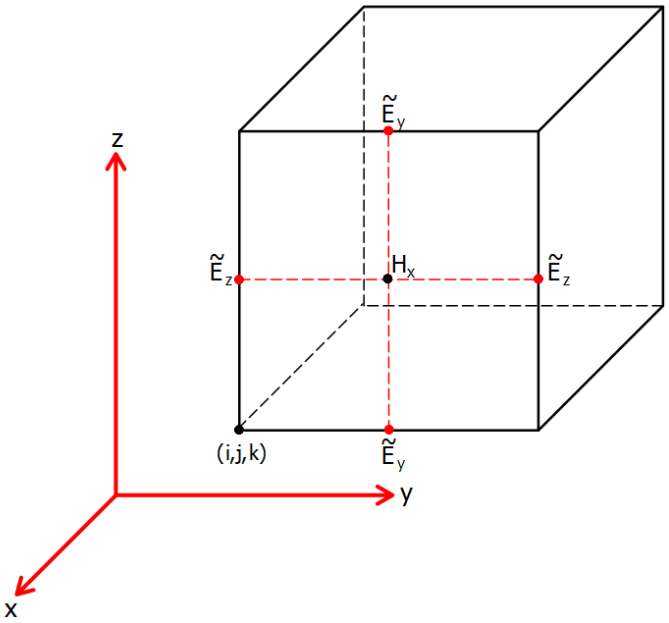

Figura 7.6: Celda de Yee para la componente H_x.

Discretizando la componente H_x aproximando los valores de $\tilde{E}_z$ y $\tilde{E}_y$ se obtiene la siguiente ecuación:

$$H_x^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)=H_x^{n-\frac{1}{2}}\left(i+,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)+0{,}5\left[\tilde{E}_y^n\left(i,j+\tfrac{1}{2},k+1\right)-\tilde{E}_y^n\left(i,j+\tfrac{1}{2},k\right)-\tilde{E}_z^n\left(i,j+1,k+\tfrac{1}{2}\right)+\tilde{E}_z^n\left(i,j,k+\tfrac{1}{2}\right)\right]. \tag{7.40}$$

La componente y del campo magnético viene dado por la ecuación 7.35, y se representa como se muestra en la Fig. 7.7:

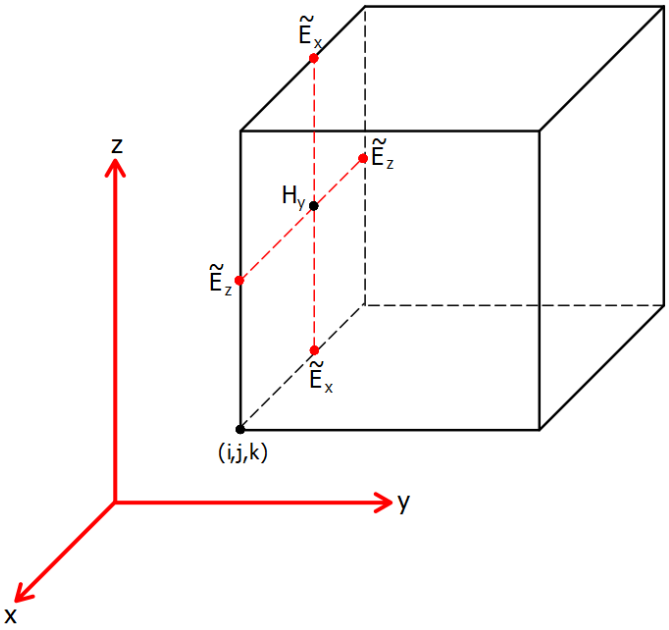

Figura 7.7: Celda de Yee para la componente H_y.

Discretizando la componente H_y aproximando los valores de $\tilde{E}_z$ y $\tilde{E}_x$ se obtiene la siguiente ecuación:

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) = H_y^{n}\left(i,j,k+\tfrac{1}{2}\right) + 0{,}5\left[\tilde{E}_z^{n+\frac{1}{2}}\left(i+1,j,k+\tfrac{1}{2}\right) - \tilde{E}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right) - \tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+1\right) + \tilde{E}_z^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\right]. \tag{7.41}$$

La componente z del campo magnético viene dado por la ecuación 7.36, y se representa como se muestra en la Fig. 7.8:

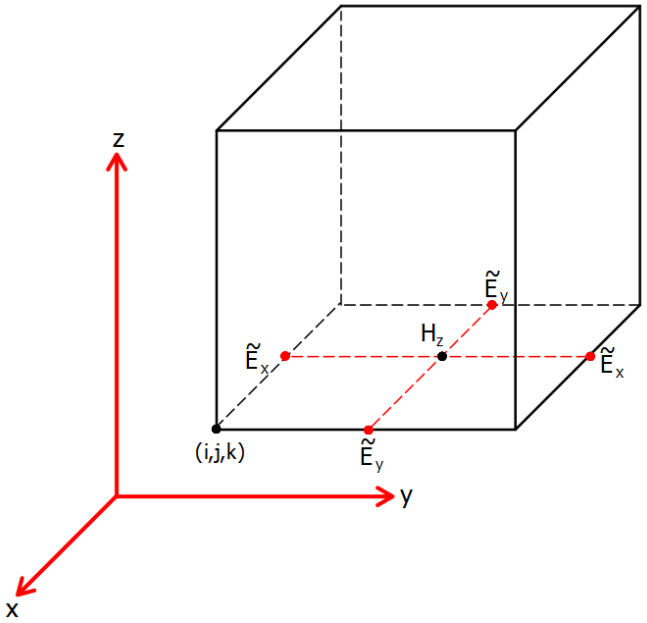

Figura 7.8: Celda de Yee para la componente H_z.

Discretizando la componente H_z aproximando los valores de $\tilde{E}_x$ y $\tilde{E}_y$ se obtiene la siguiente ecuación:

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right) = H_z^{n}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right) + 0{,}5\left[\tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right) - \tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) - \tilde{E}_y^{n+\frac{1}{2}}\left(i+1,j+\tfrac{1}{2},k\right) + \tilde{E}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)\right]. \tag{7.42}$$

Las ecuaciones discretizadas 7.40 a 7.42 representan las componentes H_x, H_y y H_z del campo magnético.

7.7. Ecuaciones en tres dimensiones: condiciones de frontera PML

El diseño de capas absorbentes en tres dimensiones debe hacerse para las tres direcciones (x, y, z) cumpliendo las condiciones de frontera absorbente PML descritas en la sección 6.3, por lo tanto, el sistema de ecuaciones que se deben discretizar son las siguientes:

$$j\omega\tilde{D}_x\epsilon_{fx}^{x}\epsilon_{fx}^{y}\epsilon_{fx}^{z} = \frac{1}{\sqrt{\epsilon_0\mu_0}}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right), \tag{7.43}$$

$$j\omega\tilde{D}_y\epsilon_{fy}^{x}\epsilon_{fy}^{y}\epsilon_{fy}^{z} = \frac{1}{\sqrt{\epsilon_0\mu_0}}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right), \tag{7.44}$$

$$j\omega\tilde{D}_z\epsilon_{fz}^{x}\epsilon_{fz}^{y}\epsilon_{fz}^{z} = \frac{1}{\sqrt{\epsilon_0\mu_0}}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right), \tag{7.45}$$

$$j\omega H_x \epsilon_{fx}^x \epsilon_{fx}^y \epsilon_{fx}^z = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_y}{\partial z} - \frac{\partial \tilde{E}_z}{\partial y} \right), \tag{7.46}$$

$$j\omega H_y \epsilon_{fy}^x \epsilon_{fy}^y \epsilon_{fy}^z = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_z}{\partial x} - \frac{\partial \tilde{E}_x}{\partial z} \right), \tag{7.47}$$

$$j\omega H_z \epsilon_{fz}^x \epsilon_{fz}^y \epsilon_{fz}^z = \frac{1}{\sqrt{\epsilon_0 \mu_0}} \left(\frac{\partial \tilde{E}_x}{\partial y} - \frac{\partial \tilde{E}_y}{\partial x} \right). \tag{7.48}$$

Para discretizar este conjunto de ecuaciones se procede de forma similar que en dos dimensiones: se inicia discretizando una componente a la vez para luego unir las soluciones en una sola ecuación que de cuenta la dirección x, y, z con respecto a las componentes del vector desplazamiento eléctrico y el campo magnético construyendo las condiciones de frontera PML en cada dirección.
Se inicia resolviendo las ecuaciones para la ecuación 7.43, componente $\tilde{D}_x$. La discretización de los campos se hace a partir de la Fig. 7.3, obteniendo el siguiente sistema de ecuaciones:

$$\begin{aligned}
\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) &= \frac{\left[1-\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]} \tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) + \\
&\quad 0{,}5\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]^{-1} camp_h, \qquad (i) \\
\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) &= \frac{\left[1-\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]} \tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) + \\
&\quad 0{,}5\left[1+\frac{\sigma(y)\Delta t}{2\epsilon_0}\right]^{-1} camp_h, \qquad (i) \\
\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) &= \tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) + \\
&\quad 0{,}5 camp_h + \frac{\sigma(i)}{2\epsilon_0} I_{Dx}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right). \qquad (iii)
\end{aligned} \tag{7.49}$$

Por simplicidad, a la hora de introducir el código en Python se simplifican las expresiones 7.49 de la siguiente manera:

$$\begin{aligned}
camp_h = H_z^n\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right) - H_z^n\left(i+\tfrac{1}{2},j-\tfrac{1}{2},k\right) - H_y^n\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) + \\
H_y^n\left(i+\tfrac{1}{2},j,k-\tfrac{1}{2}\right),
\end{aligned}$$

$$I_{Dx}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) = I_{Dx}^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right) + camp_h.$$

Combinando las ecuaciones (i), (ii) y (iii) de 7.49 para obtener una ecuación general para la componente $\tilde{D}_x$ del desplazamiento eléctrico se obtiene:

$$\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)=\tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)\frac{\left[1-\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}+ \left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]^{-1}\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\,camp_h+\frac{\sigma(i)}{2\epsilon_0}I_{Dx}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)\right]. \tag{7.50}$$

Ahora se resuelve la ecuación 7.44 para la componente $\tilde{D}_y$ tomando como referencia la Fig. 7.4 para obtener la discretización de los campos, obteniendo el siguiente sistema de ecuaciones:

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\frac{\left[1-\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}\tilde{D}_x^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)+0{,}5 \left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]^{-1}camp_h, \qquad (i)$$

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\frac{\left[1-\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}\tilde{D}_y^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)+ 0{,}5\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]^{-1}camp_h, \qquad (ii) \tag{7.51}$$

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\tilde{D}_y^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)+ 0{,}5\,camp_h+\frac{\sigma(j)}{2\epsilon_0}I_{Dy}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right). \qquad (iii)$$

Para simplificar las expresiones 7.51 se hacen los siguientes cambios de variable:

$$camp_h=H_x^n\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)-H_x^n\left(i,j+\tfrac{1}{2},k\right)-H_z^n\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)+ H_z^n\left(i-\tfrac{1}{2},j+\tfrac{1}{2},k\right),$$

$$I_{Dy}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=I_{Dy}^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)+camp_h.$$

Combinando las ecuaciones (i), (ii) y (iii) de 7.51 para obtener una ecuación general para la componente $\tilde{D}_y$ del desplazamiento eléctrico se obtiene:

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\tilde{D}_y^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)\frac{\left[1-\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]}+ \left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]^{-1}\left[1+\frac{\sigma(k)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\,camp_h+\frac{\sigma(j)}{2\epsilon_0}I_{Dy}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)\right]. \tag{7.52}$$

Finalmente, se resuelve la componente $\tilde{D}_z$ ecuación 7.45 tomando como referencia la Fig. 7.5 para obtener la discretización de los campos, obteniendo el siguiente sistema de ecuaciones:

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\frac{\left[1-\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}\tilde{D}_z^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)+ 0{,}5\left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]^{-1} camp_h, \qquad (i)$$

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\frac{\left[1-\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}\tilde{D}_z^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)+ 0{,}5\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]^{-1} camp_h, \qquad (ii) \qquad (7.53)$$

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\tilde{D}_z^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)+ 0{,}5\, camp_h+\frac{\sigma(k)}{2\epsilon_0}I_{Dz}^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right). \qquad (iii)$$

Para simplificar las expresiones 7.53 se hacen los siguientes cambios de variable:

$$camp_h=H_y^n\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)-H_y^n\left(i-\tfrac{1}{2},j,k+\tfrac{1}{2}\right)-H_x^n\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)+ H_x^n\left(i,j-\tfrac{1}{2},k+\tfrac{1}{2}\right),$$

$$I_{Dz}^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=I_{Dz}^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)+camp_h.$$

Combinando las ecuaciones (i), (ii) y (iii) de 7.53 para obtener una ecuación general para la componente $\tilde{D}_z$ del desplazamiento eléctrico se obtiene:

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\tilde{D}_z^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]}+ \left[1+\frac{\sigma(i)\Delta t}{2\epsilon_0}\right]^{-1}\left[1+\frac{\sigma(j)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\, camp_h+\frac{\sigma(k)}{2\epsilon_0}I_{Dz}^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)\right]. \qquad (7.54)$$

Al igual que en dos dimensiones se hacen los cambios de variable para simplificar las expresiones y posterior a esto se tiene en cuenta las aproximaciones hechas por **Bérenger** [10] para calcular los parámetros auxiliares, considerando que no es necesario variar la conductividad, ecuación 7.30. En la Tabla 7.4 se presentan los cambios que sufren los parámetros α, β y γ en dirección (x,y,z) respectivamente, para construir las condiciones de frontera absorbente por medio de capas ficticias en los límites computacionales.

Término	Expresión	Expresión con aproximación
$\alpha_1(i)$	$\frac{\left[1-\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn(i)}{1+xn(i)}$
$\alpha_2(i)$	$\left[1+\frac{\sigma_D(i)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn(i)}$
$\beta_1(j)$	$\frac{\left[1-\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn(j)}{1+xn(j)}$
$\beta_2(j)$	$\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn(j)}$
$\gamma_1(k)$	$\frac{\left[1-\frac{\sigma_D(k)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D(k)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn(k)}{1+xn(k)}$
$\gamma_2(k)$	$\left[1+\frac{\sigma_D(k)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn(k)}$
$A(i)$	$\frac{\sigma_D(i)\Delta t}{2\epsilon_0}$	$xn(i)$
$B(j)$	$\frac{\sigma_D(j)\Delta t}{2\epsilon_0}$	$xn(j)$
$C(k)$	$\frac{\sigma_D(k)\Delta t}{2\epsilon_0}$	$xn(k)$

Tabla 7.4: Cambios de variable para reducir las expresiones discretizadas para las componentes $\tilde{D}_x$, $\tilde{D}_y$ y $\tilde{D}_z$.

Aplicando estos cambios de variables para las ecuaciones 7.50, 7.52 y 7.54 se obtienen las componentes $\tilde{D}_x$, $\tilde{D}_y$ y $\tilde{D}_z$ del vector desplazamiento eléctrico discretizados con condiciones de frontera absorbente, de modo que cuando llegue un pulso electromagnético

hacia una de estas componentes en el límite computacional, el pulso sea absorbido en su totalidad. El conjunto de ecuaciones son las siguientes:

$$\tilde{D}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)=\tilde{D}_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)\beta_1\left(j\right)\gamma_1\left(k\right)+ \beta_2\left(j\right)\gamma_2\left(k\right)\left[0{,}5\,camp_h+B\left(j\right)I_{Dx}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k\right)\right], \tag{7.55}$$

$$\tilde{D}_y^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)=\tilde{D}_y^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)\alpha_1\left(i\right)\gamma_1\left(k\right)+ \alpha_2\left(i\right)\gamma_2\left(k\right)\left[0{,}5\,camp_h+A\left(i\right)I_{Dy}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k\right)\right], \tag{7.56}$$

$$\tilde{D}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)=\tilde{D}_z^{n-\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)\alpha_1\left(i\right)\beta_1\left(j\right)+ \alpha_2\left(i\right)\beta_2\left(j\right)\left[0{,}5\,camp_h+C\left(k\right)I_{Dz}^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)\right]. \tag{7.57}$$

Procedimiento similar se hace para encontrar las componentes discretizadas del vector campo magnético. Se inicia resolviendo la ecuación 7.46 para la componente H_x, tomando como referencia la Fig. 7.6, obteniendo el siguiente sistema de ecuaciones:

$$H_x^{n+1}\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)=\frac{\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}H_x^n\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)+ 0{,}5\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}camp_e, \qquad (i)$$

$$H_x^{n+1}\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)=\frac{\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}H_x^n\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)+ 0{,}5\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}camp_e, \qquad (ii) \tag{7.58}$$

$$H_x^{n+1}\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)=H_x^n\left(i,j+\frac{1}{2},k+\frac{1}{2}\right)+0{,}5curl_h+ \frac{\sigma\left(i\right)}{2\epsilon_0}I_{Hx}^{n+\frac{1}{2}}\left(i+\frac{1}{2},j,k+\frac{1}{2}\right). \qquad (iii)$$

De forma similar que en el caso anterior, se hacen los siguientes cambios de variable en la ecuación 7.58 para simplificar la expresión a la hora de implementar el código en Python:

$$camp_e = \tilde{E}_z^{n+\frac{1}{2}}\left(i+1,j,k+\tfrac{1}{2}\right) - \tilde{E}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right) - \tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+1\right) + \tilde{E}_z^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right),$$

$$I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right) = I_{Hx}^{n-\frac{1}{2}}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right) + camp_e.$$

Combinando las ecuaciones (i), (ii) y (iii) de 7.58 para obtener una ecuación general para la componente H_x del campo magnético, obteniendo:

$$H_x^{n+1}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right) = H_x^{n}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+ \left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\,camp_e + \frac{\sigma(i)}{2\epsilon_0}I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)\right]. \tag{7.59}$$

Ahora se resuelve la ecuación 7.47 para la componente H_y, tomando como referencia la Fig. 7.7, obteniendo el siguiente sistema de ecuaciones:

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) = H_y^{n}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+ 0{,}5\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1} camp_e, \qquad (i)$$

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) = H_y^{n}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+ 0{,}5\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1} camp_e, \qquad (ii) \tag{7.60}$$

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) = H_y^{n}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) + 0{,}5\,camp_e + \frac{\sigma(j)}{2\epsilon_0}I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right). \qquad (iii)$$

Para simplificar las expresiones 7.60 se hacen los siguientes cambios de variable:

$$camp_e = \tilde{E}_z^{n+\frac{1}{2}}\left(i+1,j,k+\tfrac{1}{2}\right) - \tilde{E}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right) - \tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+1\right) + \tilde{E}_z^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right),$$

$$I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) = I_{Hy}^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right) + camp_e.$$

Combinando las ecuaciones (i), (ii) y (iii) de 7.60 para obtener una ecuación general para la componente H_y del campo magnético, obteniendo:

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)=H_y^{n}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\frac{\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+$$
$$\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[1-\frac{\sigma\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\,camp_e+\frac{\sigma(j)}{2\epsilon_0}I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\right]. \tag{7.61}$$

Finalmente, se resuelve la ecuación 7.48 para la componente H_z, tomando como referencia la Fig. 7.8, obteniendo el siguiente sistema de ecuaciones:

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=H_z^{n}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\frac{\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+$$
$$0{,}5\left[1+\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}camp_e, \qquad (i)$$

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=H_z^{n}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\frac{\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+ \tag{7.62}$$
$$0{,}5\left[1+\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}camp_e, \qquad (ii)$$

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=H_z^{n}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)+0{,}5\,camp_e+$$
$$\frac{\sigma(k)}{2\epsilon_0}I_{Hz}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right). \qquad (iii)$$

Para simplificar la expresión 7.66 se hacen los siguientes cambios de variable:

$$camp_e=\tilde{E}_z^{n+\frac{1}{2}}\left(i+1,j,k+\tfrac{1}{2}\right)-\tilde{E}_z^{n+\frac{1}{2}}\left(i,j,k+\tfrac{1}{2}\right)-\tilde{E}_x^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+1\right)+$$
$$\tilde{E}_z^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right),$$

$$I_{Hz}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=I_{Hy}^{n-\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)+camp_e.$$

Combinando la ecuación (i), (ii) y (iii) de 7.66 para obtener una ecuación general para la componente H_z del campo magnético, se obtiene:

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=H_z^{n}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\frac{\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}\frac{\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}+$$
$$\left[1-\frac{\sigma\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[1-\frac{\sigma\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}\left[0{,}5\,camp_e+\frac{\sigma(k)}{2\epsilon_0}I_{Hz}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\right]. \tag{7.63}$$

Se considera nuevamente que para construir las condiciones de frontera absorbente no es necesario varias la conductividad, por lo que se consideran nuevamente las aproximaciones hechas por **Berenger** [10]. Dado que en este caso se están encontrando las ecuaciones para el campo magnético las constantes están desplazadas en la mitad de las celdas de Yee como se vio en la sección 5.3. En la Tabla 7.5 se muestran los cambios que sufren los parámetros $\alpha + \frac{1}{2}$, $\beta + \frac{1}{2}$ y $\gamma + \frac{1}{2}$ en dirección (x, y, z) respectivamente.

Término	Expresión	Expresión con aproximación
$\alpha_1\left(i+\frac{1}{2}\right)$	$\frac{\left[1-\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn\left(i+\frac{1}{2}\right)}{1+xn\left(i+\frac{1}{2}\right)}$
$\alpha_2\left(i+\frac{1}{2}\right)$	$\left[1+\frac{\sigma_D\left(i+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn\left(i+\frac{1}{2}\right)}$
$\beta_1\left(j+\frac{1}{2}\right)$	$\frac{\left[1-\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}{\left[1+\frac{\sigma_D(j)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn\left(j+\frac{1}{2}\right)}{1+xn\left(j+\frac{1}{2}\right)}$
$\beta_2\left(j+\frac{1}{2}\right)$	$\left[1+\frac{\sigma_D\left(j+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn\left(j+\frac{1}{2}\right)}$
$\gamma_1\left(k+\frac{1}{2}\right)$	$\frac{\left[1-\frac{\sigma_D\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right)}{\left[1+\frac{\sigma_D\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]}$	$\frac{1-xn\left(k+\frac{1}{2}\right)}{1+xn\left(k+\frac{1}{2}\right)}$
$\gamma_2\left(k+\frac{1}{2}\right)$	$\left[1+\frac{\sigma_D\left(k+\frac{1}{2}\right)\Delta t}{2\epsilon_0}\right]^{-1}$	$\frac{1}{1+xn\left(k+\frac{1}{2}\right)}$

Tabla 7.5: Cambios de variable para reducir las expresiones discretizadas para las componentes H_x, H_y y H_z.

Aplicando estos cambios de variables para las ecuaciones 7.59, 7.65 y 7.63 se obtienen las componentes H_x, H_y y H_z discretizadas con condiciones de frontera absorbente para el campo magnético, de modo que cuando llegue un pulso electromagnético al límite computacional hacia las componentes del campo magnético este sea absorbido en su totalidad. El conjunto de ecuaciones resultantes son las siguientes:

$$H_x^{n+1}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)=H_x^n\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)\beta_1\left(j+\tfrac{1}{2}\right)\gamma_2\left(k+\tfrac{1}{2}\right)+\beta_2\left(j+\tfrac{1}{2}\right)\gamma_2\left(k+\tfrac{1}{2}\right)\left[0{,}5\,camp_e+A_1\left(i\right)I_{Hx}^{n+\frac{1}{2}}\left(i,j+\tfrac{1}{2},k+\tfrac{1}{2}\right)\right], \tag{7.64}$$

$$H_y^{n+1}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)=H_y^n\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\alpha_1\left(i+\tfrac{1}{2}\right)\gamma_1\left(k+\tfrac{1}{2}\right)+\alpha_2\left(i+\tfrac{1}{2}\right)\gamma_2\left(k+\tfrac{1}{2}\right)\left[0{,}5\,camp_e+B_1\left(j\right)I_{Hy}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j,k+\tfrac{1}{2}\right)\right], \tag{7.65}$$

$$H_z^{n+1}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)=H_z^n\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\alpha_1\left(i+\tfrac{1}{2}\right)\beta_1\left(j+\tfrac{1}{2}\right)+\alpha_2\left(i+\tfrac{1}{2}\right)\beta_2\left(j+\tfrac{1}{2}\right)\left[0{,}5\,camp_e+C_1\left(k\right)I_{Hz}^{n+\frac{1}{2}}\left(i+\tfrac{1}{2},j+\tfrac{1}{2},k\right)\right]. \tag{7.66}$$

En esta sección se discretizaron las ecuaciones de Maxwell en tres dimensiones aplicando las condiciones de frontera absorbente PML empleando el método que consiste en diseñar capas absorbentes en los límites computacionales. Se encontró las capas absorbentes en cada componente del vector desplazamiento eléctrico y del campo magnético. El siguiente paso es introducir estos código en el lenguaje de programación Python para obtener resultados gráficos de los campos eléctricos magnéticos aplicando las condiciones de frontera de un conductor perfectamente eléctrico PEC y condiciones de frontera absorbente PML.

8. Resultados

El presente capítulo tiene como fin obtener la representación gráfica de los campos electromagnéticos en una, dos y tres dimensiones, aplicando en el límite computacional condiciones de frontera de un conductor perfecto y condiciones de frontera absorbente, con el método de diferencias finitas en el dominio del tiempo, descrito en la sección 7.

8.1. Onda unidimensional: condiciones PEC

Al implementar las ecuaciones 7.8 y 7.9 en una dimensión con condiciones de frontera de un conductor perfecto en el lenguaje de programación Python, se obtiene la distribución de los campos eléctrico E_x y magnético H_y a lo largo de cien celdas del espacio computacional $(0 < z < 100)$, después de cien pasos temporales $T = 100$, de un pulso electromagnético, con origen en $z = 50$, que se propaga en dirección z, como se muestra en la Fig. 8.1. Ahora bien, obsérvese que los campos se encuentran en fase y, además, el campo eléctrico y magnético son perpendiculares a la dirección de propagación z, correspondiendo con ondas transversales, propias de la naturaleza de las ondas electromagnéticas.

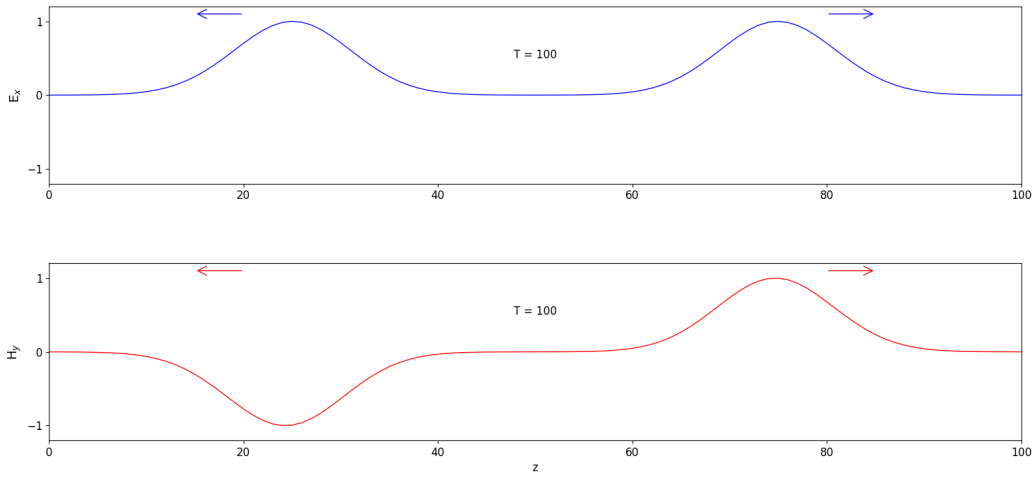

Figura 8.1: Distribución de los campos eléctrico E_x y magnético H_y con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional $(0 < z < 100)$, después de cien pasos temporales $T = 100$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo A.1.

Dado que las condiciones de frontera impuestas son las de un conductor perfecto, es

de interés observar qué sucede con la onda electromagnética cuando llega al límite computacional. Para esto, se toman diferentes pasos temporales para las componentes E_x y H_y, empleando un pulso gaussiano, descrito en la sección 5.4.1, como se observa en la Fig. 8.2 y Fig. 8.3, respectivamente.

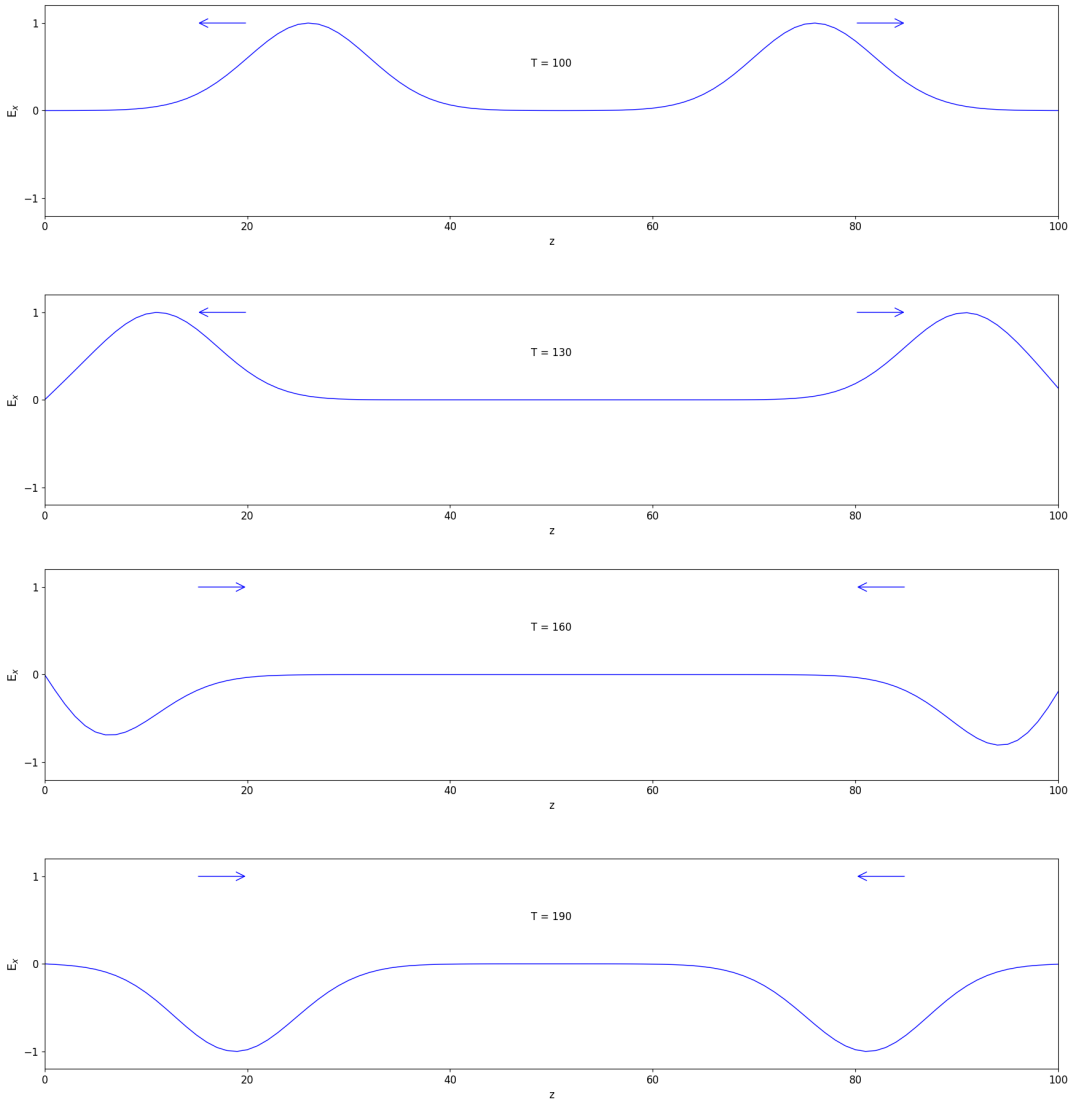

Figura 8.2: Distribución del campo eléctrico E_x con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo A.2.

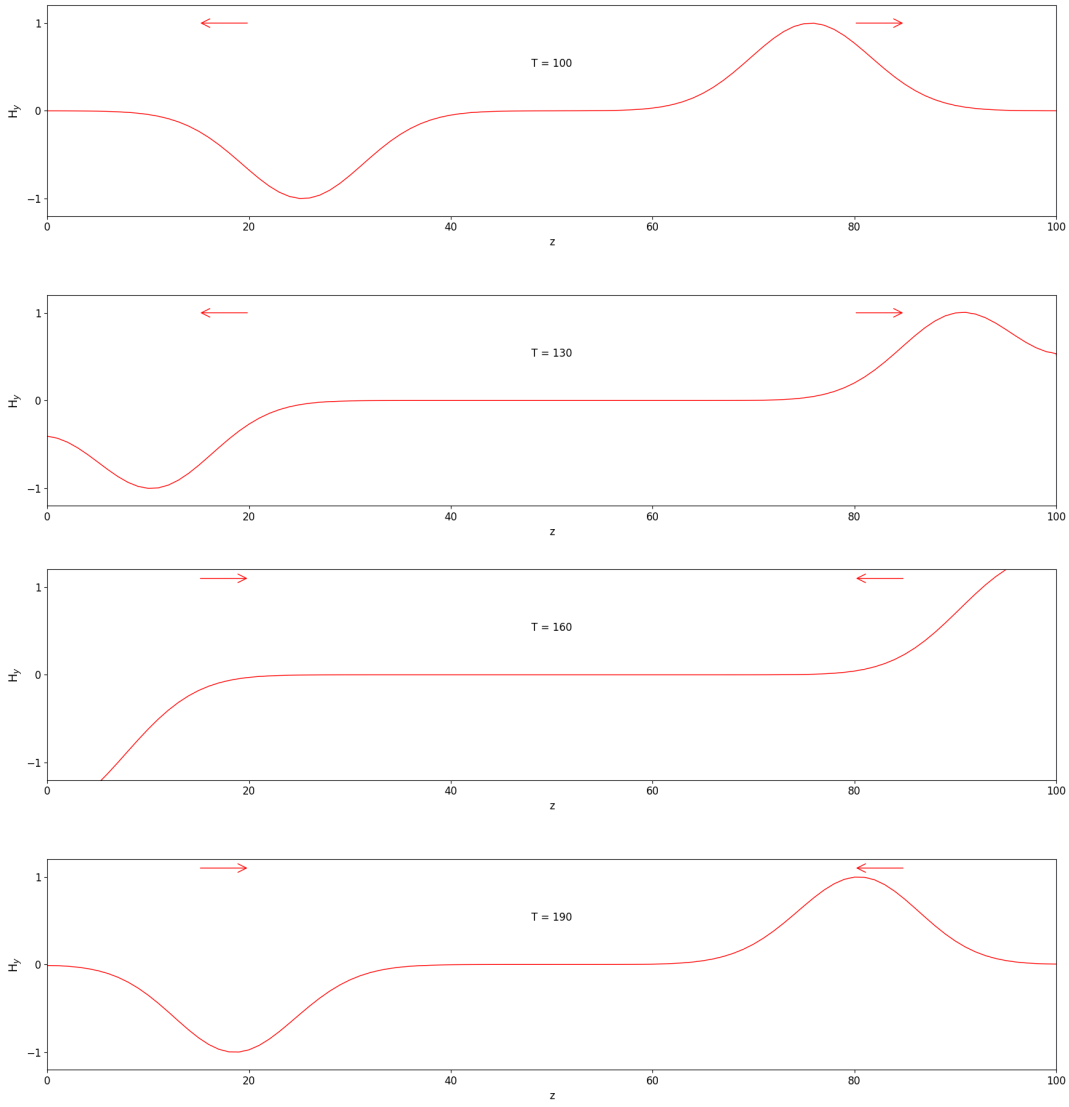

Figura 8.3: Distribución del campo magnético H_y con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo A.3.

Cuando la onda electromagnética llega al límite computacional se refleja en su totalidad. De la teoría electromagnética descrita en la sección 4.4, es de esperar que el campo eléctrico tangencial sea cero en el límite computacional constituido por un PEC, por lo tanto, el campo eléctrico reflejado debe estar desfasado 180 grados en comparación con el campo incidente. Este resultado se muestra en la Fig. 8.2 y Fig. 8.3, al comparar

el paso temporal $T = 100$ antes de chocar con el PEC con el paso temporal $T = 190$ después de chocar con el PEC. Concluyendo así, que el límite computacional actúa como un operador que cambia la fase, pero conserva la amplitud de la onda, características que pertenecen a las condiciones de frontera de un conductor perfecto PEC, descritas en la sección 6.2.

8.2. Onda unidimensional: condiciones PML

El siguiente paso es obtener la solución de una onda electromagnética en una dimensión con condiciones de frontera absorbente en el límite computacional. Las ecuaciones en una dimensión para las condiciones de frontera absorbente son similares a las ecuaciones con condiciones de frontera PEC, lo único que cambia son las condiciones impuestas en el límite computacional, como se vio en la sección 7.3. Por lo tanto, el campo eléctrico y campo magnético se encuentran en fase y son perpendiculares entre sí, características propias de las ondas transversales. Es de interés observar qué sucede cuando la onda electromagnética llega al límite computacional. Para esto, se toman diferentes pasos temporales para la componente E_x del campo eléctrico y la componente H_y del campo magnético, como se observa en la Fig. 8.4 y Fig. 8.5, respectivamente.

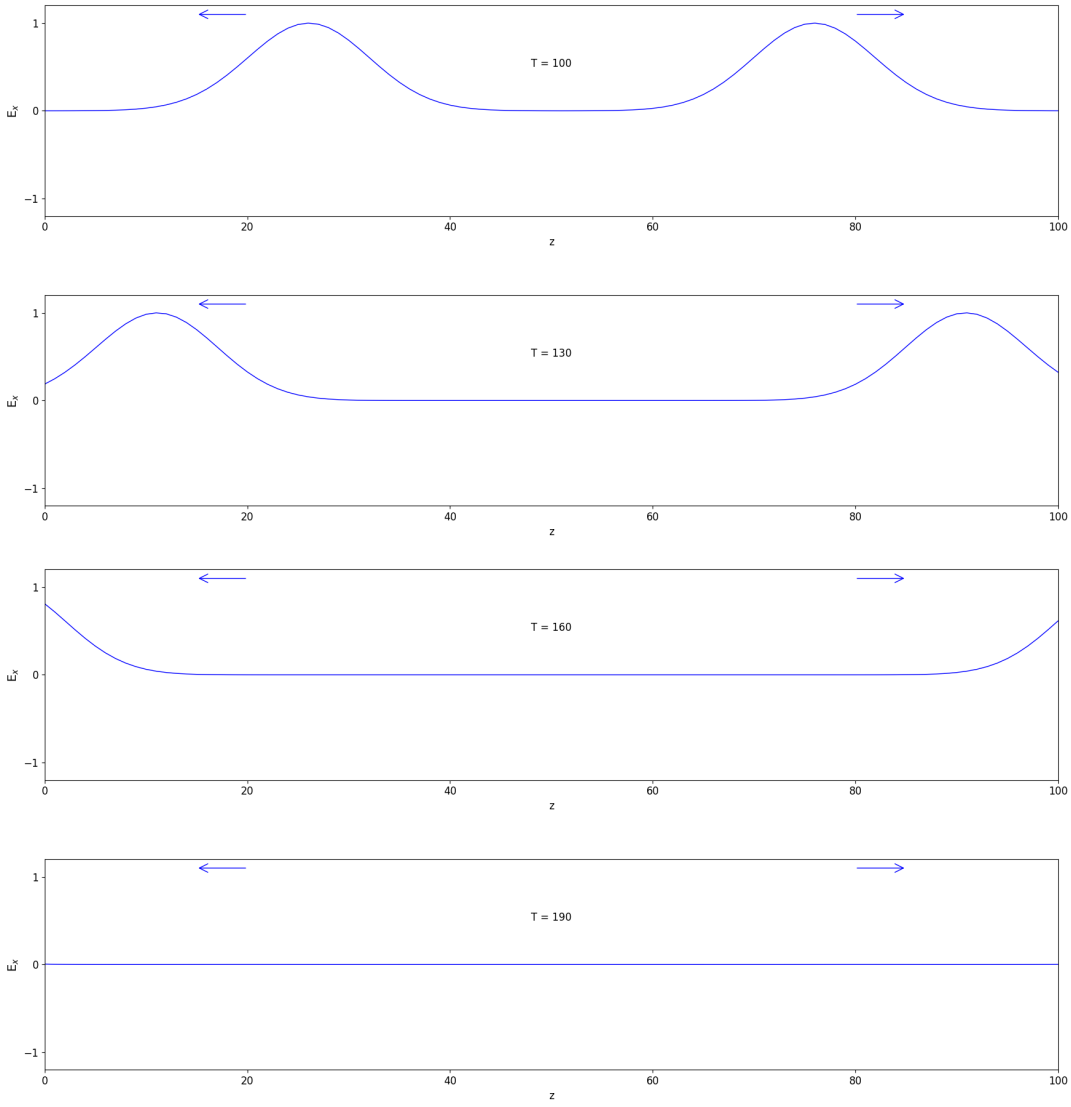

Figura 8.4: Distribución del campo eléctrico E_x con condiciones de frontera PML a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo B.1.

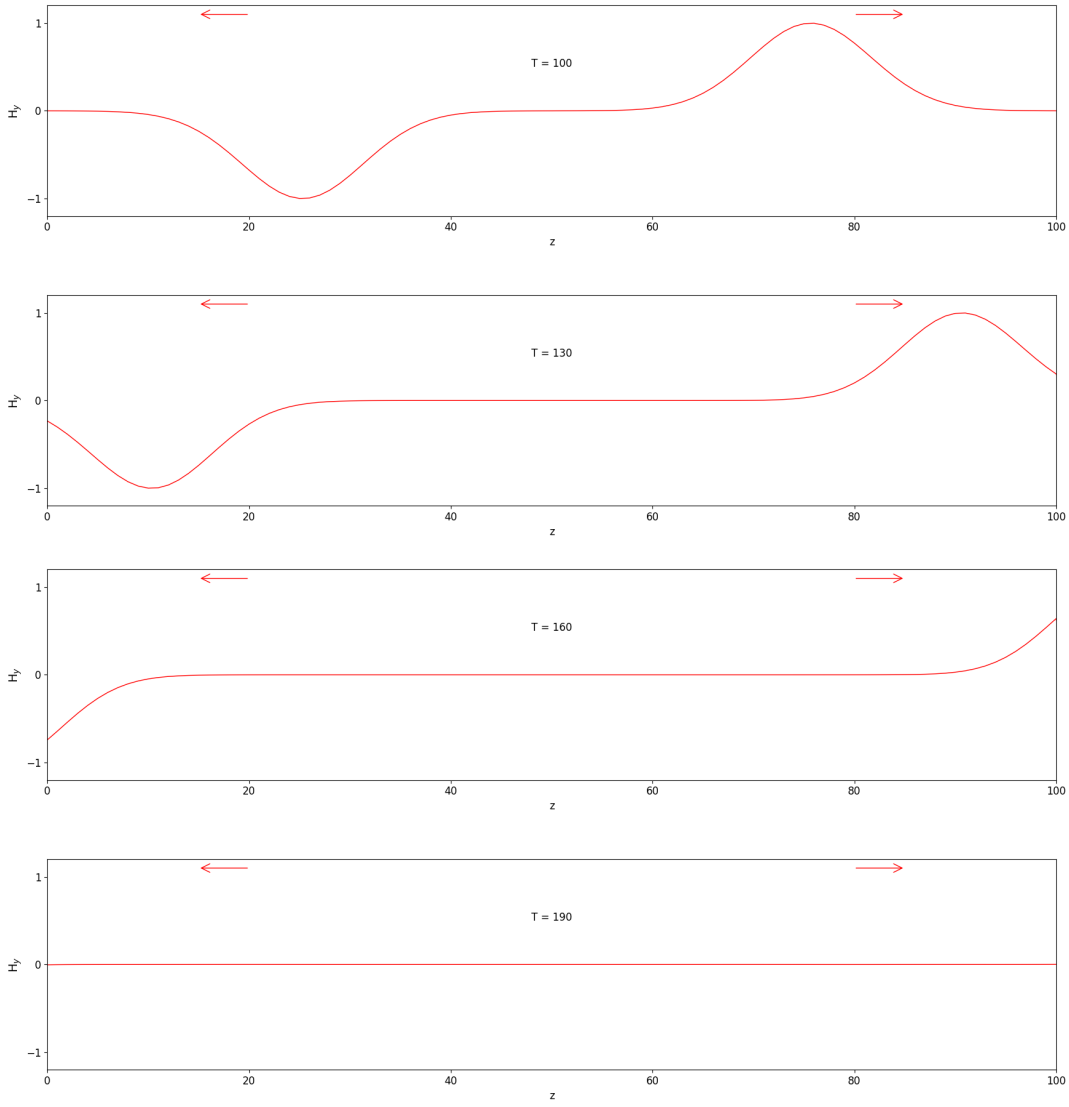

Figura 8.5: Distribución del campo magnético H_y con condiciones de frontera PML a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo B.2.

Al observar la representación gráfica de los campos electromagnéticos en una dimensión, se evidencia que en el paso temporal $T = 160$ la onda es absorbida parcialmente por los límites computacionales, y en el paso temporal $T = 190$ la onda electromagnética ha sido absorbida en su totalidad. Es decir, no se observa ninguna distorsión o reflexión alguna hacia el interior del espacio computacional. Esto se cumple tanto para la com-

ponente E_x del campo eléctrico como para la componente H_y del campo magnético. Estas características son propias de las condiciones de frontera absorbente diseñadas por medio de las capas perfectamente combinadas PML, donde se satisface que cuando la onda electromagnética llegue al límite computacional sea absorbida en su totalidad, de tal manera que las ondas no sufran distorsión o reflexión alguna hacia el interior del espacio computacional.

8.3. Onda en dos dimensiones: condiciones PEC

Una onda electromagnética en dos dimensiones puede descomponerse en ondas transversales eléctricas TE y ondas tranversales magnéticas TM. Los siguientes resultados obtenidos son para el caso de ondas transversales magnéticas TM. Teniendo presente esto, el campo magnético se propaga en el plano xy y el campo eléctrico está polarizado en la dirección E_z.
Al implementar las ecuaciones 7.14 a 7.16 en dos dimensiones con condiciones de frontera de un conductor perfecto en el lenguaje de programación Python, se obtiene la distribución del modo transversal magnético a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, de un pulso gaussiano con origen en $(40, 40)$, con el campo eléctrico polarizado en la componente E_z. Es de interés conocer qué sucede con la onda transversal magnética cuando llega al límite computacional constituido por un conductor perfecto. De tal manera que, se toman diferentes pasos temporales, iniciando en $T = 10$ y finalizando en $T = 120$, obteniendo los siguientes resultados:

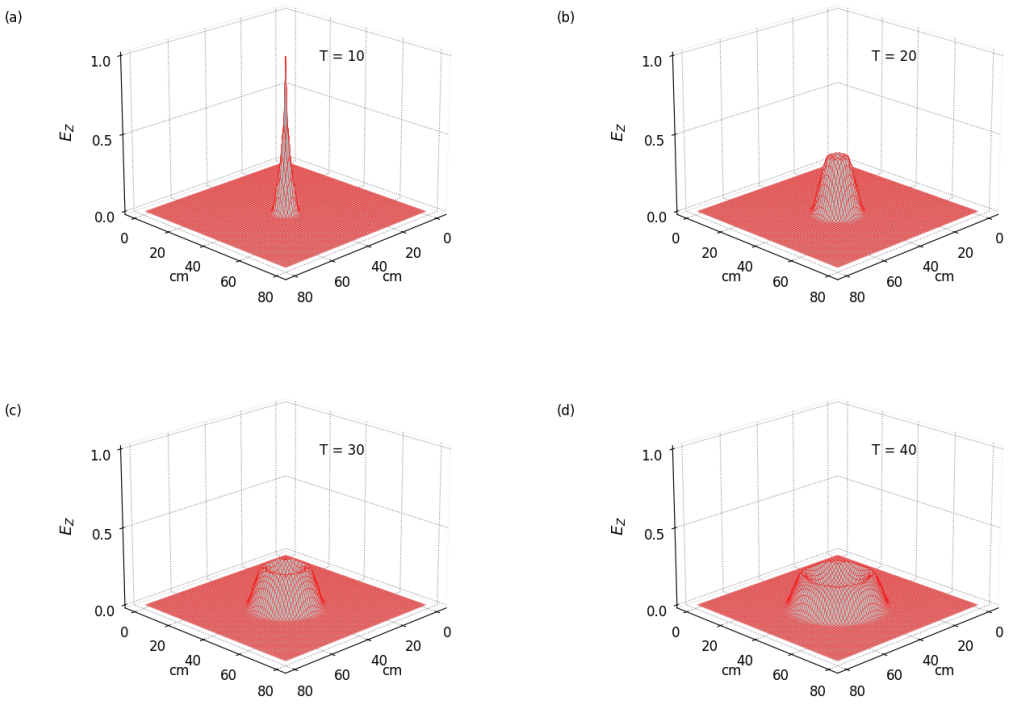

Figura 8.6: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (a) T=10, (b) T=20, (c) T=30 y (d) T=40, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo C.1.

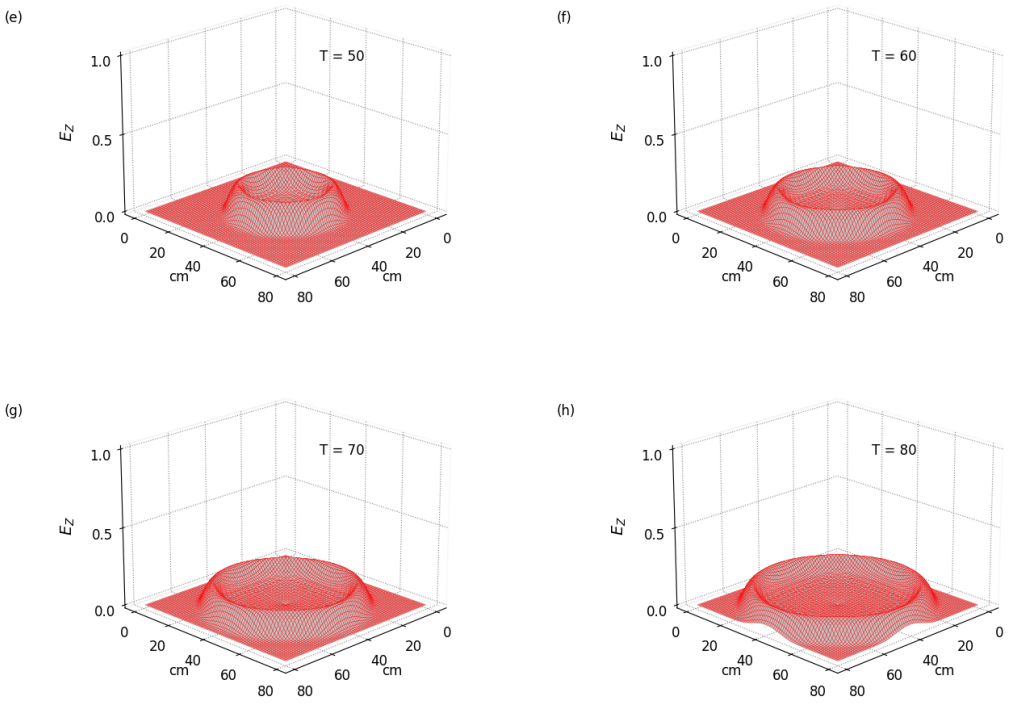

Figura 8.7: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (e) T=50, (f) T=60, (g) T=70 y (h) T=80, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo C.1.

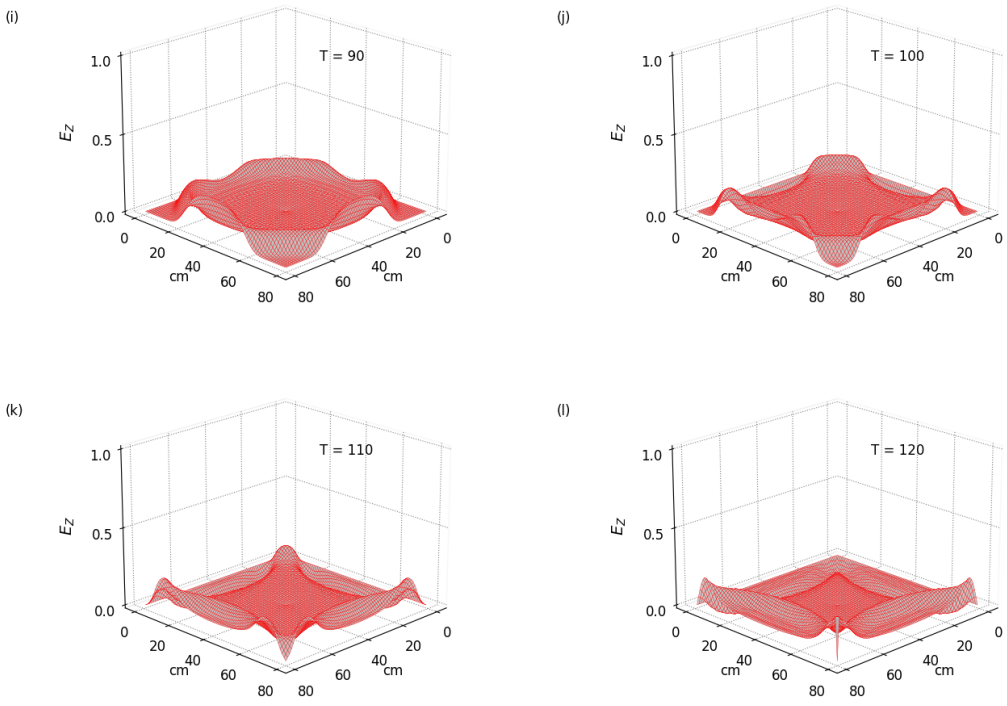

Figura 8.8: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (i) T=90, (j) T=100, (k) T=110 y (l) T=120, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo C.1.

En la Fig. 8.6, Fig. 8.7 y Fig. 8.8 se muestra la propagación de una onda transversal magnética implementando condiciones de frontera de un conductor perfecto en el límite computacional. La onda inicia en la coordenada $(40, 40)$, a medida que aumentan los pasos temporales la onda se extiende por el espacio computacional hasta alcanzar el límite. Dado que el campo eléctrico tangencial es cero en el límite computacional compuesto por un PEC, se espera que el campo eléctrico reflejado esté desfasado 180 grados en comparación con el campo incidente. Desde el paso temporal $T = 10$ hasta el paso temporal $T = 70$ la onda transversal magnética se propaga sin ninguna perturbación. En el paso temporal $T = 80$, la onda llega al límite computacional y en los siguientes pasos se evidencia que la onda es reflejada, pero con fase opuesta, como se puede observar en los pasos temporales $T = 90$, $T = 100$, $T = 110$ y $T = 120$. Además, cabe destacar que no es posible discernir la onda incidente y la onda reflejada después del paso temporal $T = 80$, lo que dificulta la visualización de la onda electromagnética.
Con el fin de obtener una mejor representación visual de la onda transversal magnética cuando llega al límite computacional, se observa la onda desde la parte superior, de modo que se logre representar un mapa de contorno. Para esto se hace uso de una

fuente sinusoidal descrita en la sección 5.4.2. El pulso sinusoidal es práctico, ya que es una onda continua, permitiendo observar si la onda es reflejada o absorbida por el límite computacional.

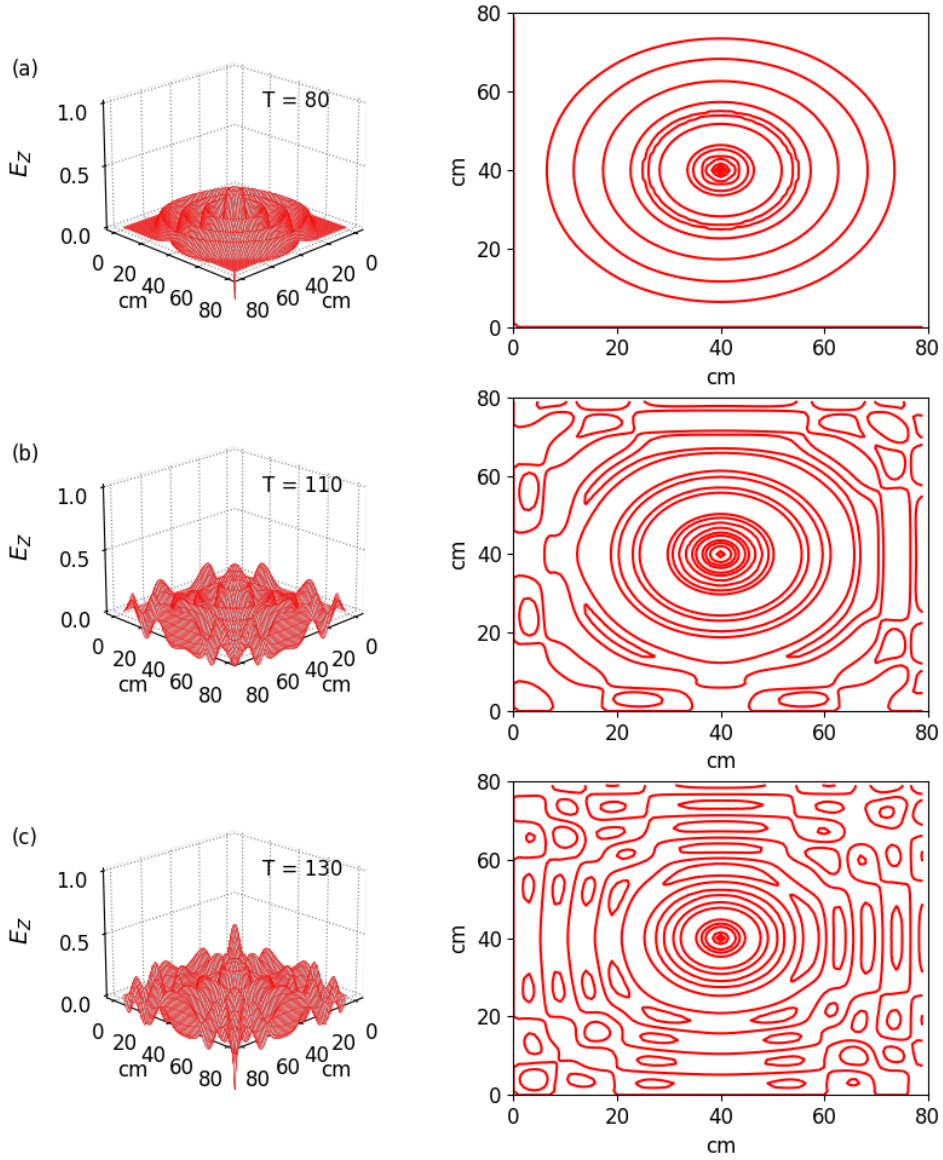

Figura 8.9: Máximos locales del campo eléctrico con condiciones de frontera PEC con mapa de contorno, a lo largo de ochenta celdas del espacio computacional ($0 < x < 80$), ($0 < y < 80$), para los pasos temporales (a) T=80, (b) T=110 y (c) T=130, de un pulso sinusoidal con origen en $(40, 40)$. Ver el código fuente en el Anexo C.2.

En la Fig. 8.9 se observa la propagación de una onda transversal magnética para diferentes pasos temporales junto con un mapa de contorno. El mapa de contorno tiene

como objetivo mostrar que la onda incidente inicial no puede diferenciarse de la onda reflejada luego del paso temporal $T = 80$. Esto debido al límite computacional constituido por un conductor perfecto. En el paso temporal $T = 110$ y $T = 130$ es evidente que no es posible lograr discrepar la onda incidente y la onda reflejada. Con el fin de medir la relación entre la onda incidente y la onda reflejada en amplitud, se mide la amplitud relativa de los máximos en la componente E_z para un paso temporal luego de que la onda transversal magnética llega al límite computacional.

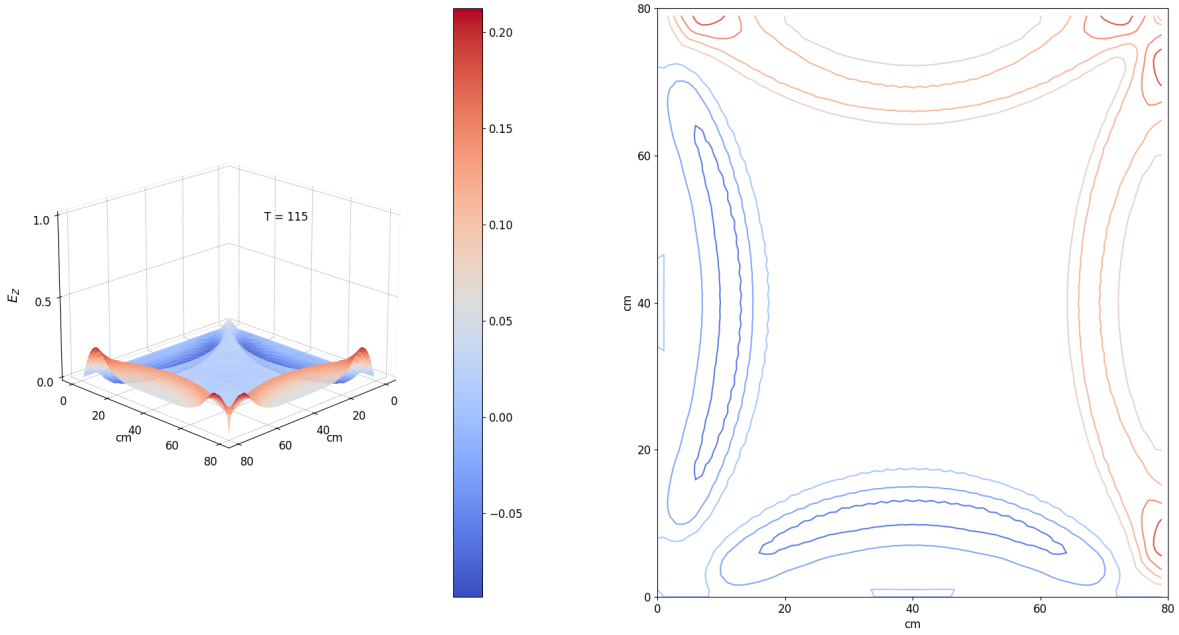

Figura 8.10: Distribución de los máximos del campo eléctrico con condiciones de frontera PEC con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, después de ciento quince pasos temporales $T = 115$, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo C.3.

En la Fig. 8.10 se muestra una onda transversal magnética en el paso temporal $T = 115$, junto a una barra de alturas y un mapa de contorno. Se observa que la onda reflejada tiene una amplitud relativa de 0,20 como máximo en algunas zonas del entorno computacional. En este paso temporal, como se vio en la Fig. 8.8, la onda incidente aumenta en amplitud debido a la reflexión con las paredes del límite computacional.
En esta sección se obtuvo por medio de la implementación del método de diferencias finitas en el dominio del tiempo, la representación gráfica de las ecuaciones de Maxwell en dos dimensiones con condiciones de frontera de un conductor perfecto, logrando una visualización de los campos eléctricos y magnéticos por medio de simulaciones computacionales, satisfaciendo las condiciones de frontera impuestas.

8.4. Onda en dos dimensiones: condiciones PML

Nuevamente se considera un onda transversal magnética TM. Teniendo presente esto, el campo magnético se propaga en el plano xy y el campo eléctrico está polarizado en la dirección E_z. Al implementar las ecuaciones 7.26 a 7.28 en dos dimensiones con condiciones de frontera absorbente PML en el lenguaje de programación Python, se obtiene la distribución del modo transversal magnético a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, de un pulso gaussiano con origen en $(40, 40)$, con el campo eléctrico polarizado en la componente E_z. Es de interés conocer que sucede con la onda transversal magnética cuando llega al límite computacional en las condiciones de frontera absorbente. De tal manera que, se toman diferentes pasos temporales, iniciando en $T = 10$ y finalizando en $T = 120$, obteniendo los siguientes resultados:

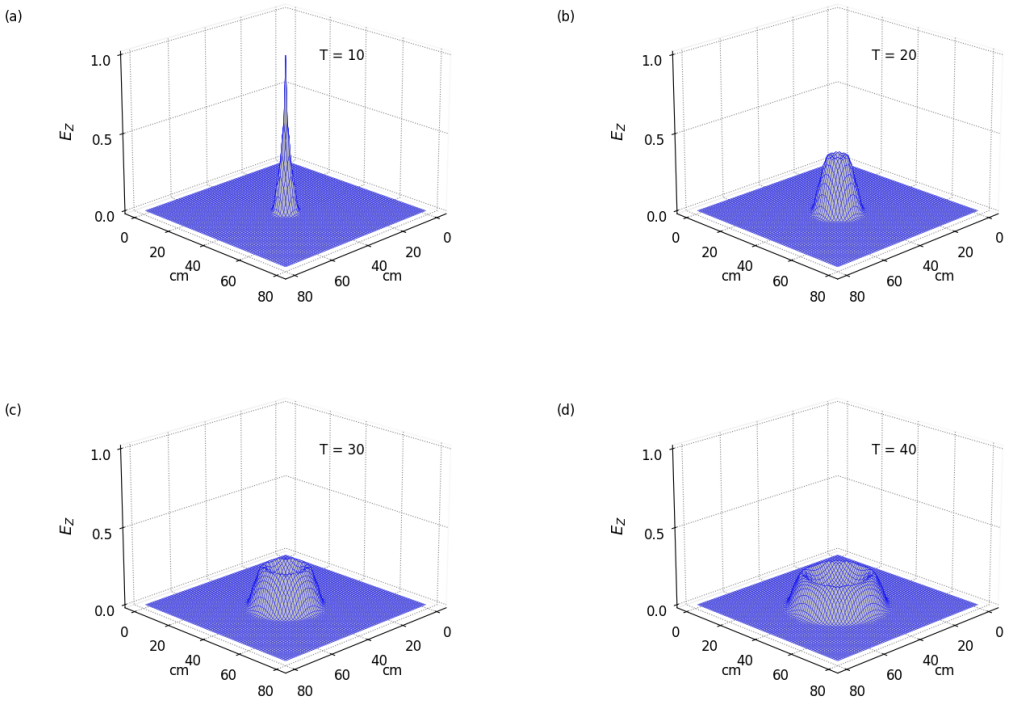

Figura 8.11: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (a) T=10, (b) T=20, (c) T=30 y (d) T=40, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo D.1.

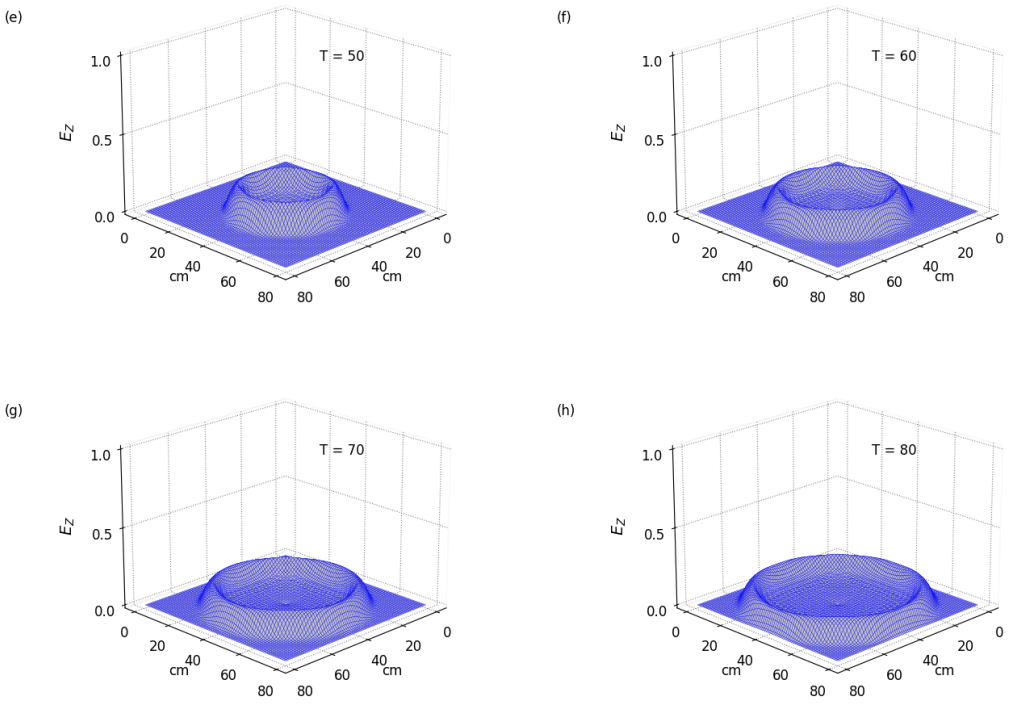

Figura 8.12: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (e) T=50, (f) T=60, (g) T=70 y (h) T=80, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo D.1.

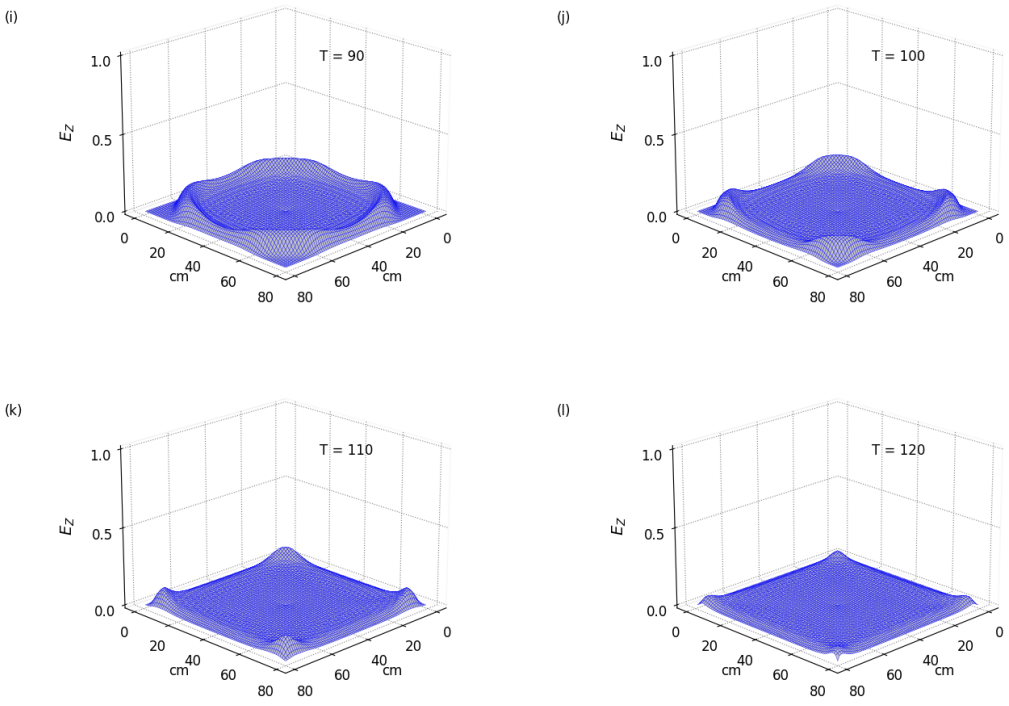

Figura 8.13: Distribución del campo eléctrico en la componente E_z con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (i) T=90, (j) T=100, (k) T=110 y (l) T=120, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo D.1.

En la Fig. 8.11, Fig. 8.12 y Fig. 8.13 se muestra la propagación de una onda transversal magnética implementando condiciones de frontera absorbente. La onda inicia en la coordenada $(40, 40)$ y se propaga hasta el límite computacional a medida que aumentan los pasos temporales. Donde cabe resaltar que en el paso temporal $T = 70$ la visualización de los campos con condiciones de frontera PEC y PML son los mismos, resultado que debe ser esperado dado qué las condiciones de frontera son diseñas únicamente en el límite computacional.
Después del paso temporal $T = 80$ hasta el paso temporal $T = 120$ la componente E_z del campo eléctrico disminuye su amplitud, donde es notorio que la onda transversal magnética es absorbida por los límites computacionales sin reflejar la onda incidente. Al compararlo con el caso con condiciones PEC se evidencia una notable disminución en la amplitud de la onda una vez que llega la onda al límite computacional. Para evidenciar la diferencia entre las condiciones PEC y PML, se representa la onda transversal magnética con condiciones de frontera absorbente junto con un mapa de contorno para analizar los máximos locales, tomando los mismos pasos temporales que en las condiciones PEC. Para esto, nuevamente se hace uso de una fuente sinusoidal.

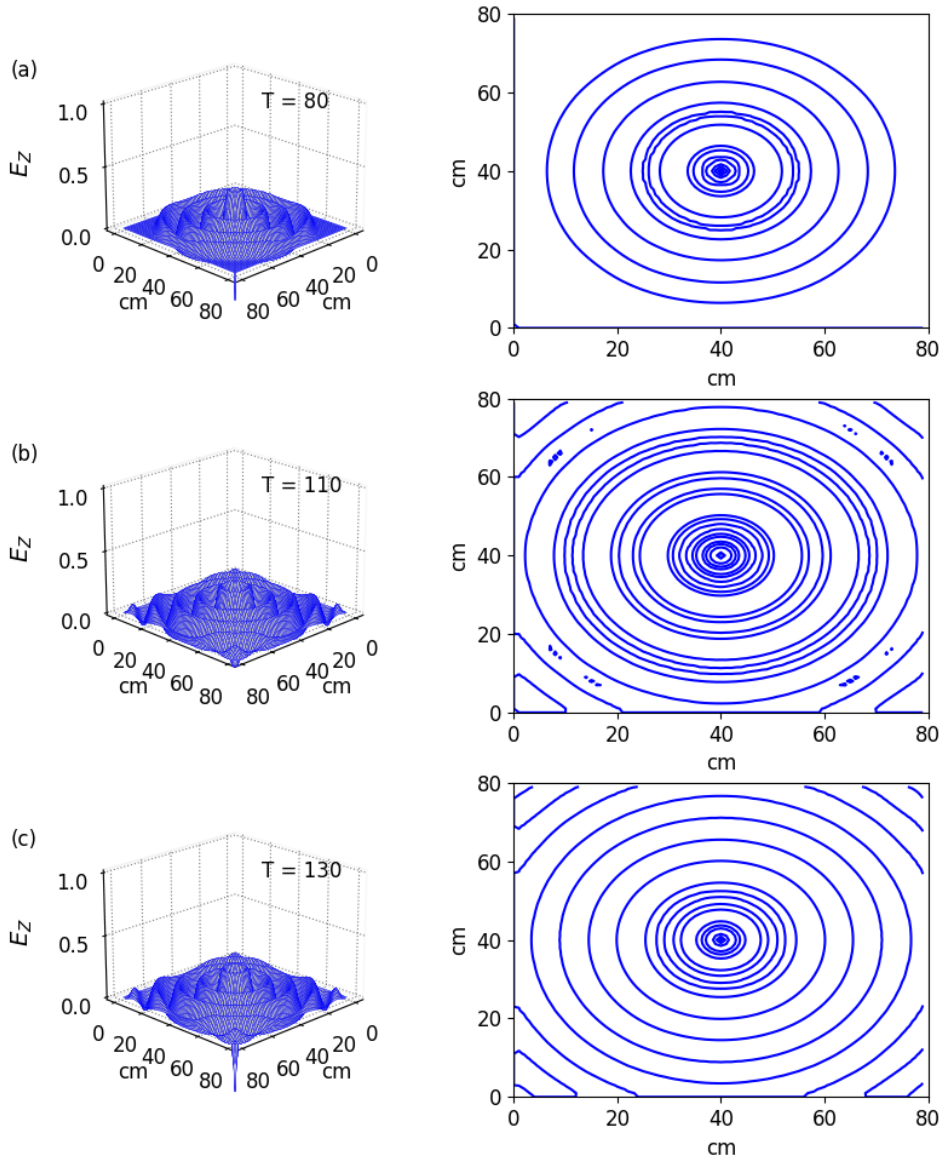

Figura 8.14: Máximos locales del campo eléctrico con condiciones de frontera PML con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (a) T=80, (b) T=110 y (c) T=130, de un pulso sinusoidal con origen en $(40, 40)$. Ver el código fuente en el Anexo D.2.

En la Fig. 8.14 se observa la propagación de una onda transversal magnética para diferentes pasos temporales con condiciones de frontera absorbente junto con un mapa de contorno. En este caso, se puede constatar que en el paso temporal $T = 110$ y $T = 130$ se puede discernir la onda incidente en cada punto del espacio computacional, en donde no se observa ninguna reflexión por parte de la onda transversal magnética incidente. Nuevamente, se mide la amplitud relativa de los máximos en la componente E_z para

un paso temporal cuando llegue la onda transversal magnética al límite computacional.

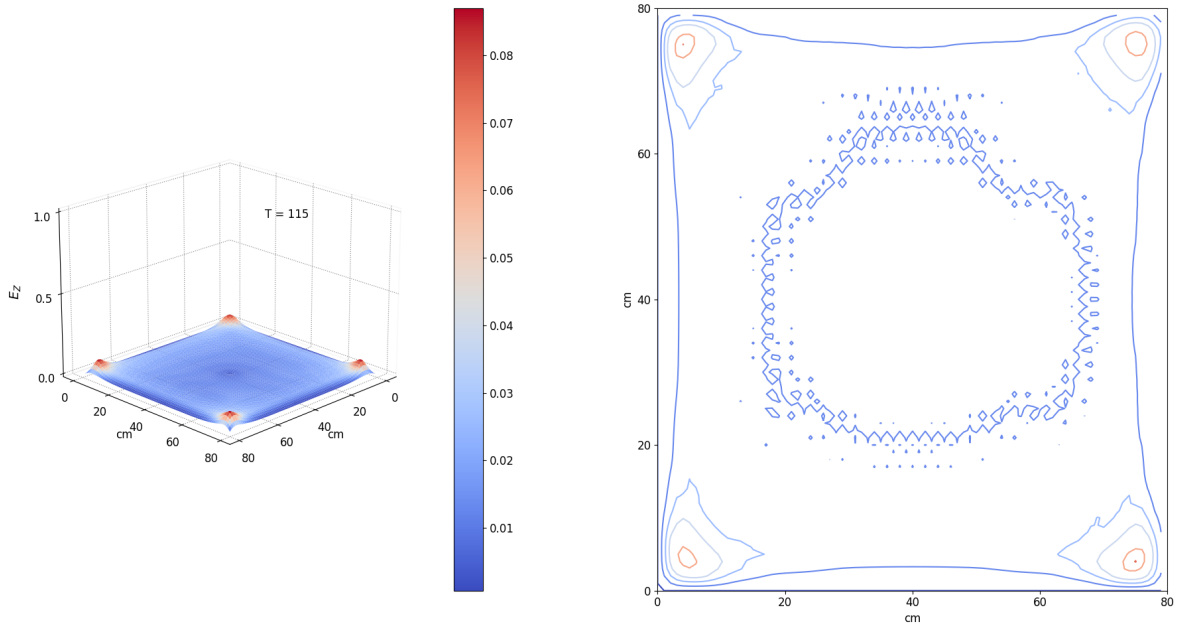

Figura 8.15: Distribución de los máximos del campo eléctrico en la componente E_z con condiciones de frontera PML con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, después de ciento quince pasos temporales $T = 115$, de un pulso gaussiano con origen en $(40, 40)$. Ver el código fuente en el Anexo D.3.

En la Fig. 8.15 se muestra una onda transversal magnética con condiciones de frontera PML en el paso temporal $T = 115$ junto a una barra de alturas y un mapa de contorno. Se observa que la onda reflejada tiene una amplitud relativa de 0,08 como máximo en algunas zonas del entorno computacional, mucho menor que en el caso con condiciones de frontera PEC, donde alcanzó una amplitud relativa de 0,20, como se observa en la Fig. 8.10, evidenciando una notable reducción de la onda reflejada gracias a las capas perfectamente combinadas diseñadas en el límite computacional.
En esta sección, se obtuvo la representación gráfica de las ecuaciones de Maxwell en dos dimensiones con condiciones de frontera absorbente, implementando el diseño de capas perfectamente emparejadas en el límite computacional, implementando el método de diferencias finitas en el dominio del tiempo, logrando la visualización de los campos magnéticos y eléctricos por medio de simulaciones computacionales.

8.5. Onda en tres dimensiones: consideraciones

Simular una onda electromagnética en tres dimensiones es similar a la solución en dos dimensiones, excepto por la fuente de propagación. Cuando una fuente puntual irradia igual cantidad de energía en todas las direcciones, se dice que obedece la ley del inverso al cuadrado, esto proviene de consideraciones geométricas. Considere una fuente de campo eléctrico constituida por una carga puntual Q, como se muestra en la Fig. 8.16, a medida que se propaga desde la fuente, la intensidad del campo disminuye con el inverso al cuadrado.

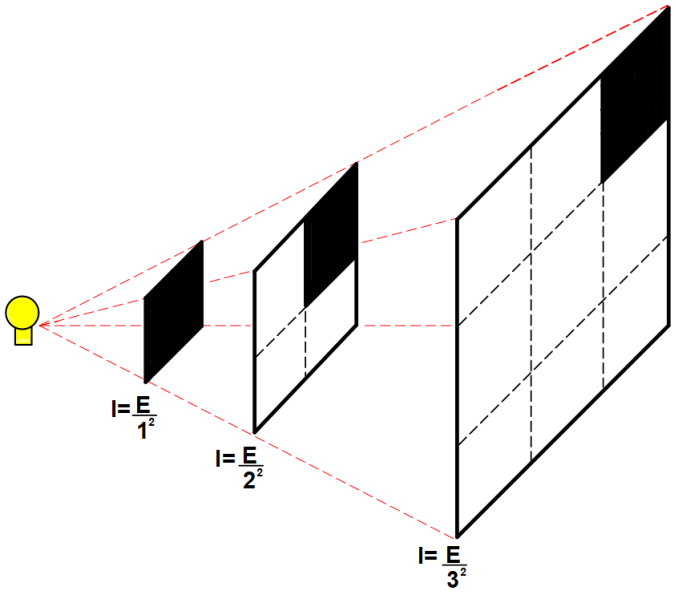

Figura 8.16: Representación de la intensidad de una onda electromagnética.

A la hora de simular una onda electromagnética en tres dimensiones, el pulso obedecerá esta ley a medida que se propaga por el entorno computacional, por lo tanto, se diseña una antena dipolo como fuente de propagación dentro del espacio computacional. La antena dipolo que se diseña consta de dos polos conductores. La densidad de corriente fluye entre los dos polos y la diferencia de potencial asociada hacen que una onda electromagnética se irradie hacia afuera de la antena. La corriente estará determinada por la ley de Ampere. A medida que se actualiza el campo magnético, se actualiza la corriente y en consecuencia se actualiza el campo eléctrico. Cabe aclarar que la dirección de propagación corresponde con la componente radial, si la onda se propaga en dirección z, la antena dipolo también tendrá radiación en esa componente. [3]

8.6. Onda en tres dimensiones: condiciones PEC

Teniendo presente el diseño de la antena dipolo visto en la sección 8.5, se considera la distribución de la componente z desde el dipolo en el plano xy. Al implementar las

ecuaciones 7.37 a 7.42 en tres dimensiones con condiciones de frontera de un conductor perfecto en el lenguaje de programación Python, se obtiene la distribución de los campos eléctricos y magnéticos a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$, con distribución de la componente z desde el dipolo en el plano xy. Es de interés conocer que sucede con la onda electromagnética en tres dimensiones cuando llega al límite computacional constituido por un conductor perfecto. De tal manera que, se toman diferentes pasos temporales, iniciando en $T = 20$ y finalizando en $T = 240$, obteniendo los siguientes resultados:

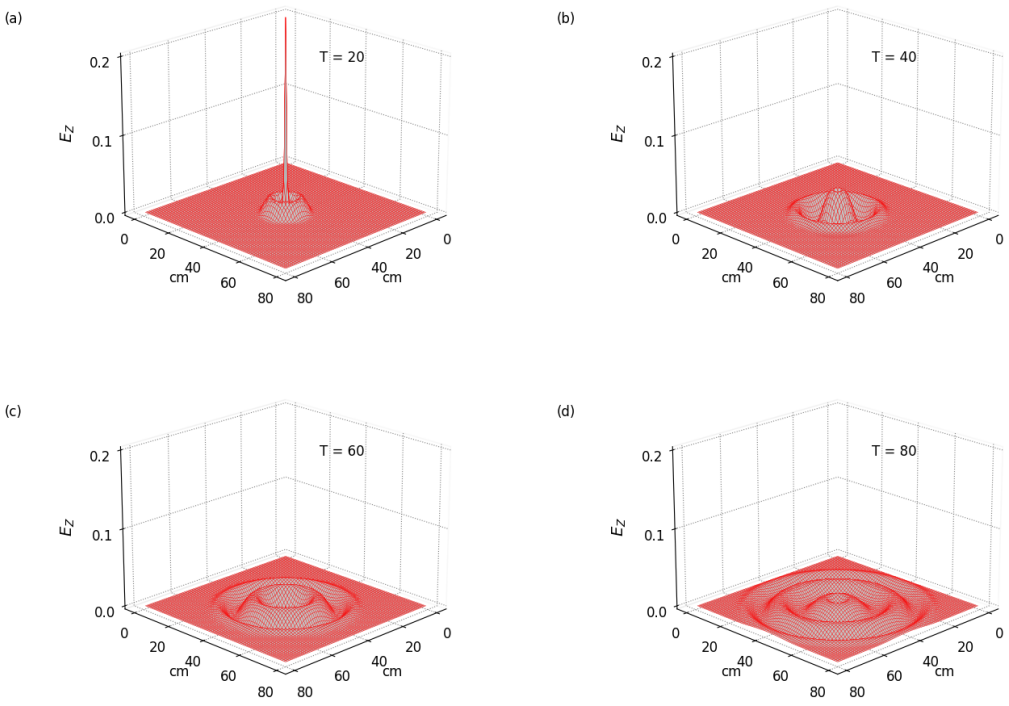

Figura 8.17: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=20, (b) T=40, (c) T=60 y (d) T=80, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo E.1.

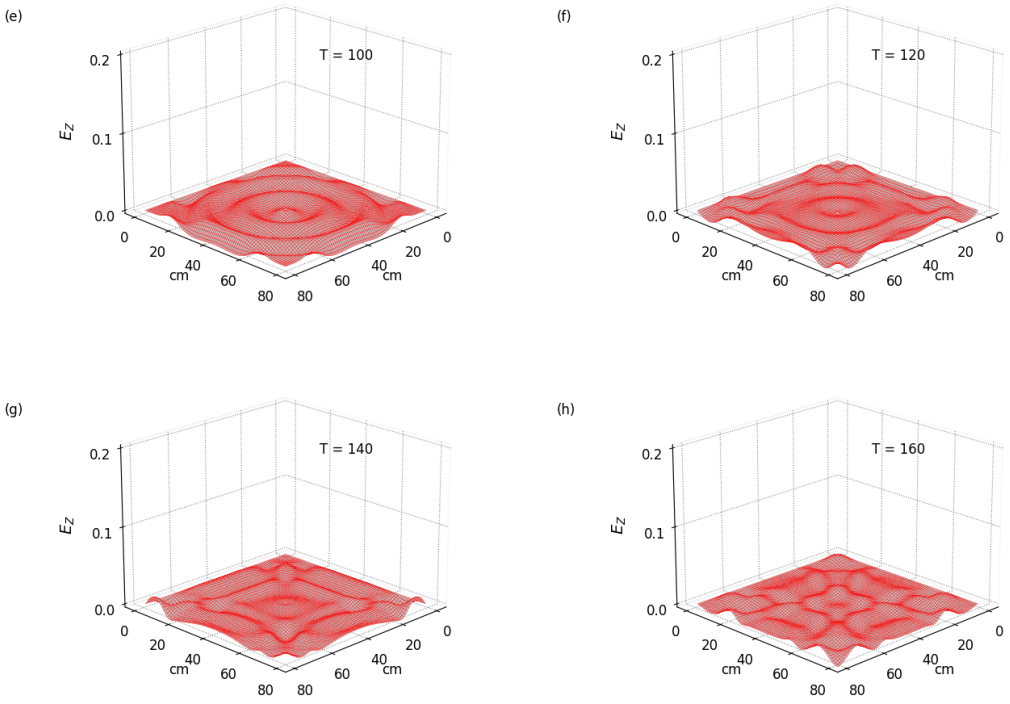

Figura 8.18: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (e) T=100, (f) T=120, (g) T=140 y (h) T=160, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo E.1.

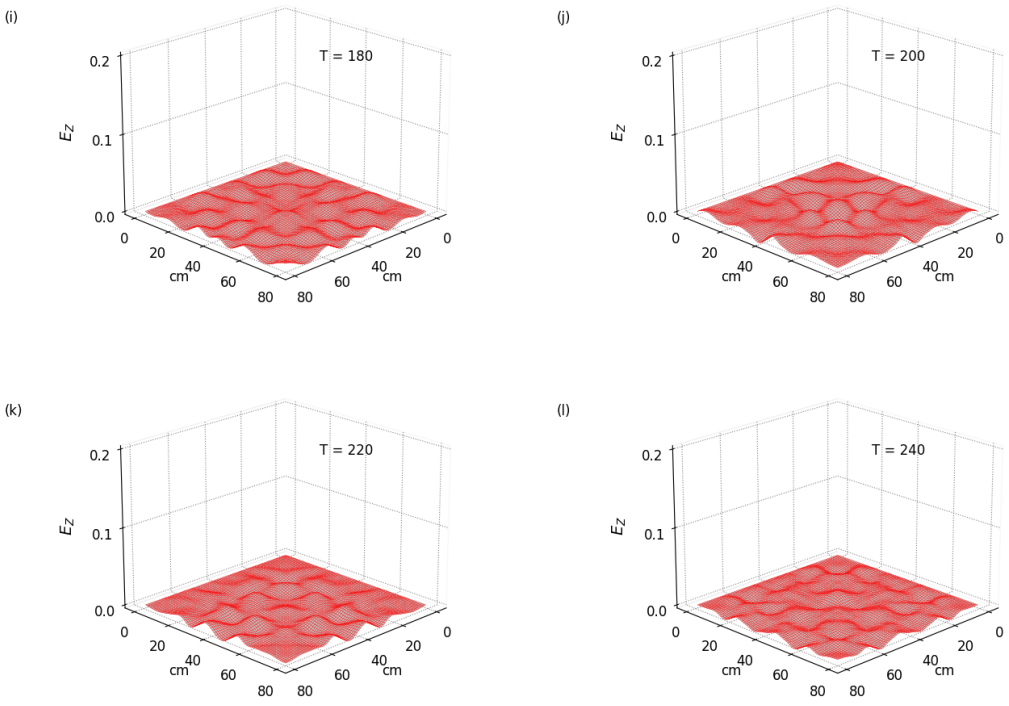

Figura 8.19: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (i) T=180, (j) T=200, (k) T=220 y (l) T=240, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo E.1.

En la Fig. 8.17, Fig. 8.18 y Fig. 8.19 se muestra la propagación de una onda electromagnética en tres dimensiones, la onda inicia en la coordenada $(40, 40, 0)$, a medida que aumentan los pasos temporales la onda se extiende por el espacio computacional hasta alcanzar el límite constituido por un conductor perfecto. Desde el paso temporal $T = 20$ hasta el paso temporal $T = 80$, la onda electromagnética en tres dimensiones se propaga de manera uniforme por el entorno computacional. Al considerar que la componente tangencial del campo eléctrico es cero en el límite computacional compuesto por un conductor perfecto, es de esperar que la onda se refleje en su totalidad, pero que esté desfasada 180 grados en comparación con la onda incidente, resultado que se observa luego del paso temporal $T = 80$ y es evidente en los siguientes pasos temporales.
Con el fin de obtener una mejor representación visual de la onda electromagnética en tres dimensiones cuando llega al límite computacional, se observa la onda desde la parte superior, de modo que se logre ilustrar en un mapa de contorno para analizar los máximos locales de la componente z. Para esto se una antena dipolo de onda continua

(pulso sinusoidal) en el plano $z = 0$ con origen en $(40, 40, 0)$.

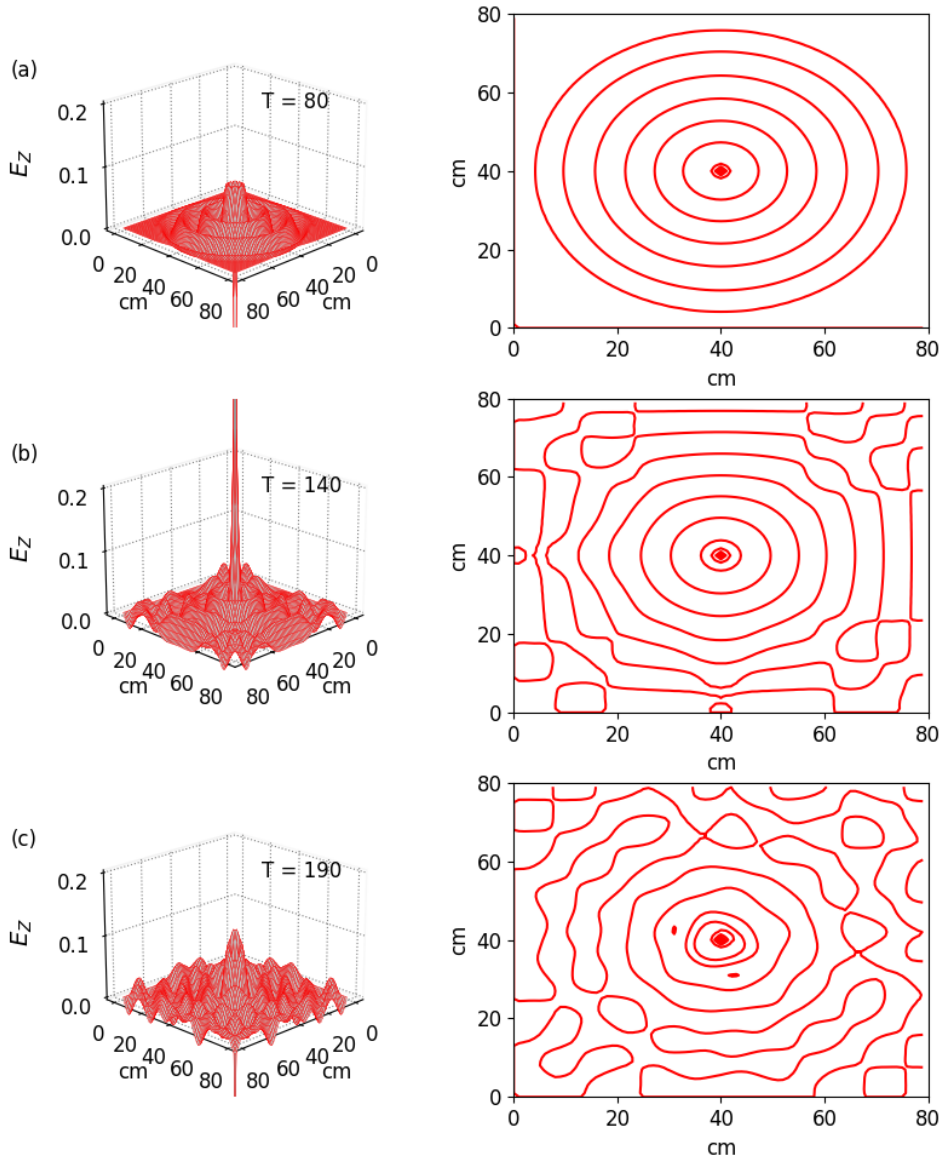

Figura 8.20: Distribución de los máximos de la componente z con condiciones de frontera PEC con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=80, (b) T=140 y (c) T=190, de una antena dipolo de onda continua (pulso sinusoidal) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo E.2.

En la Fig. 8.20 se representa una onda electromagnética en tres dimensiones junto a un mapa de contorno para tres pasos temporales imponiendo las condiciones de frontera PEC. Es evidente que justo después del paso temporal $T = 80$ la onda llega al límite

computacional y es reflejada hacia el interior. En el paso temporal $T = 190$ no es posible discernir la onda incidente y la onda reflejada, como se muestra en el mapa de contorno, condiciones propias de un conductor perfecto.
Ahora, se diseña un mapa de contorno junto a una barra de altura que permita medir la relación entre la intensidad de la onda incidente y la onda reflejada en amplitud, midiendo la amplitud relativa de los máximos en la componente z para un paso temporal luego de que la onda electromagnética en tres dimensiones llegue al límite computacional.

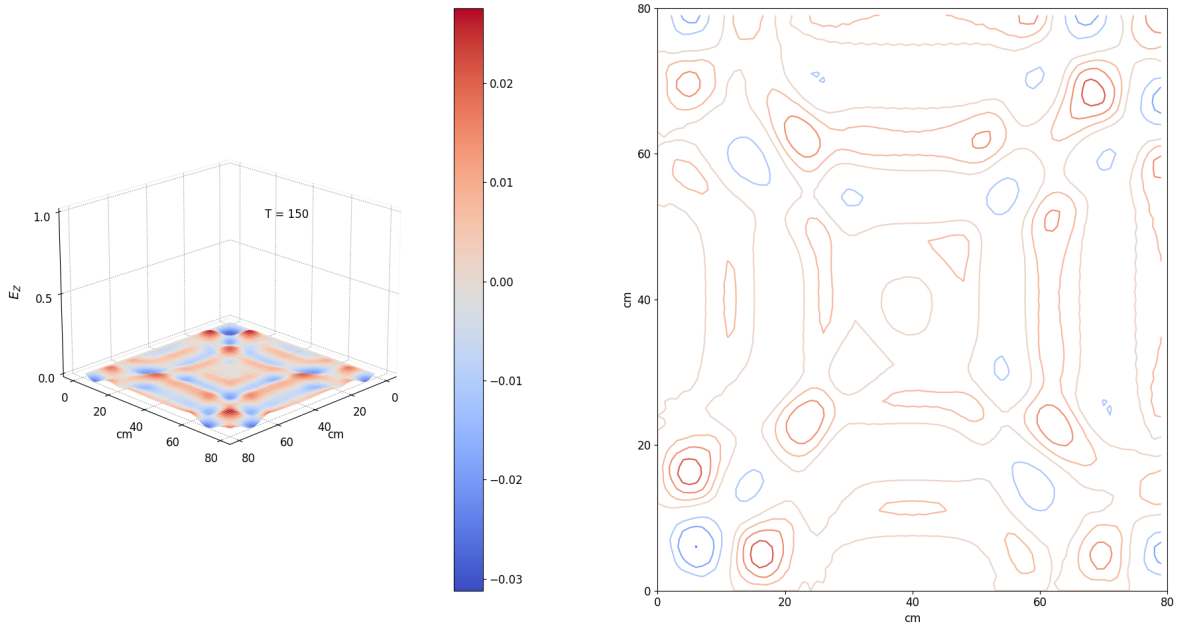

Figura 8.21: Distribución de los máximos del campo eléctrico en la componente z con condiciones de frontera PEC con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, después de ciento cincuenta pasos temporales $T = 150$, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo E.3.

En la Fig. 8.21 se muestra una onda electromagnética en tres dimensiones con condiciones de frontera PEC en el paso temporal $T = 150$, junto a una barra de alturas y un mapa de contorno. La onda reflejada tiene una amplitud relativa de 0,02, mucho menor que en el caso en dos dimensiones, donde alcanzó una altura de 0,20 como se observa en la Fig. 8.10. La disminución en el máximo de la altura (comparándolo con el caso de la propagación de una onda electromagnética en dos dimensiones) se debe principalmente a la naturaleza de la fuente.
En esta sección, se obtuvo la representación gráfica de las ecuaciones de Maxwell en dos dimensiones con condiciones de frontera de un conductor perfecto en el límite computacional, implementando el método de diferencias finitas en el dominio del tiempo, lo-

grando la visualización de los campos magnéticos y eléctricos por medio de simulaciones computacionales.

8.7. Onda en tres dimensiones: condiciones PML

Teniendo presente el diseño de la antena dipolo visto en la sección 8.5, se considera la distribución de la componente z desde el dipolo en el plano xy. Al implementar las ecuaciones 7.55 a 7.57 y 7.64 a 7.66 en tres dimensiones con condiciones de frontera absorbente en el lenguaje de programación Python, se obtiene la distribución de los campos eléctricos y magnéticos a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$, con distribución de la componente z desde el dipolo en el plano xy. Es de interés conocer que sucede con la onda electromagnética en tres dimensiones cuando llega al límite computacional en las condiciones de frontera absorbente. De tal manera que, se toman diferentes pasos temporales, iniciando en $T = 20$ y finalizando en $T = 240$, obteniendo los siguientes resultados:

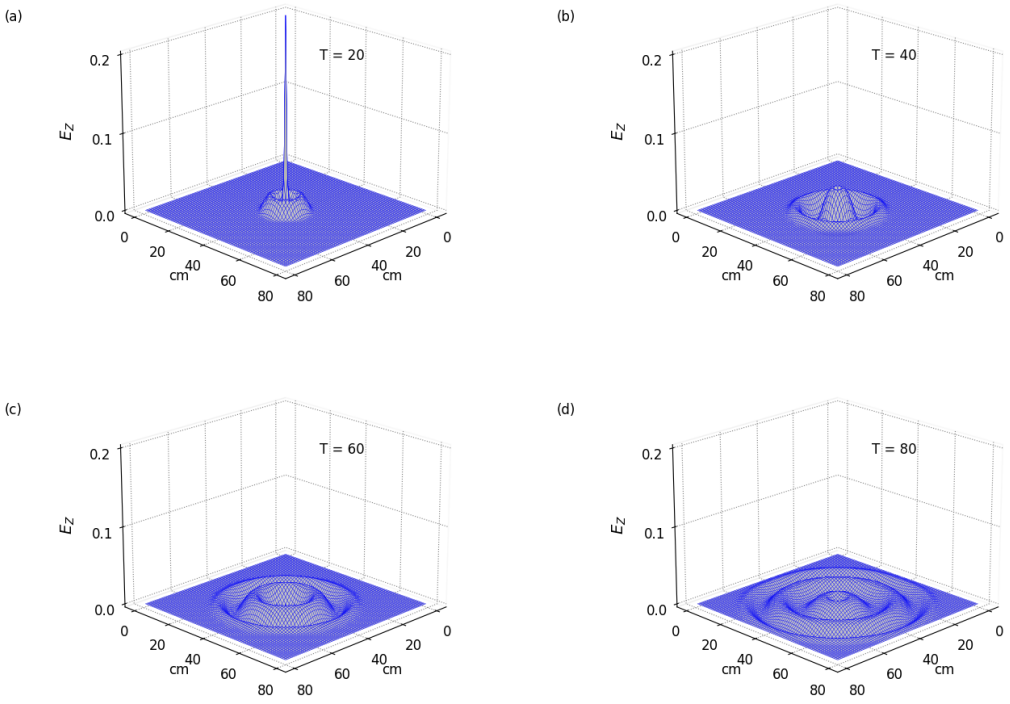

Figura 8.22: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=20, (b) T=40, (c) T=60 y (d) T=80, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo F.1.

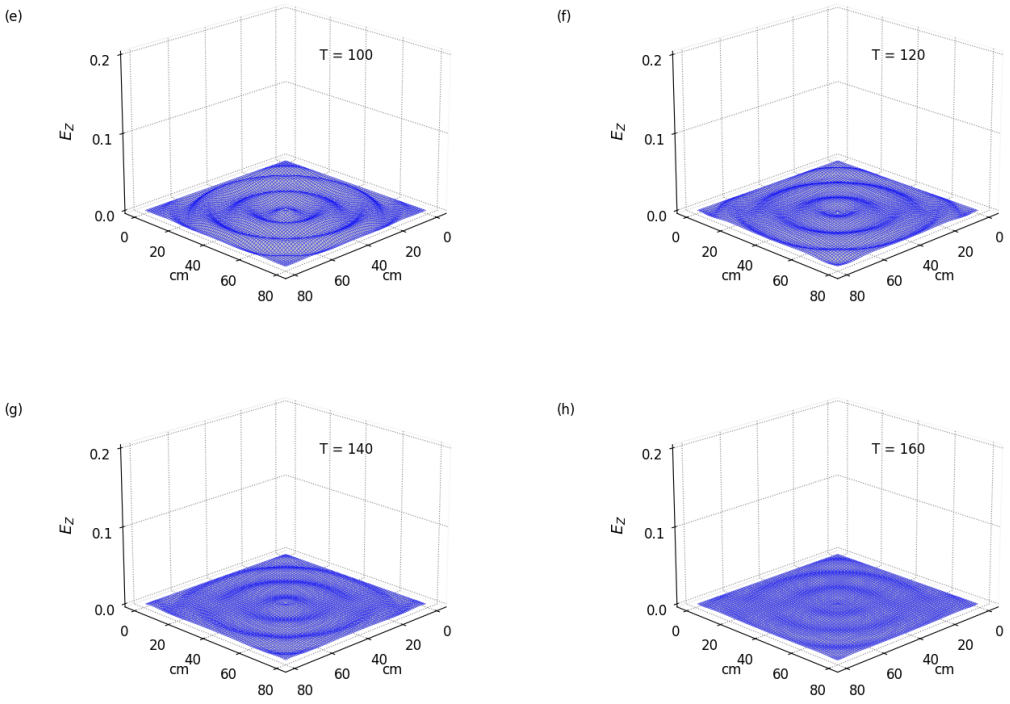

Figura 8.23: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (e) T=100, (f) T=120, (g) T=140 y (h) T=160, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo F.2.

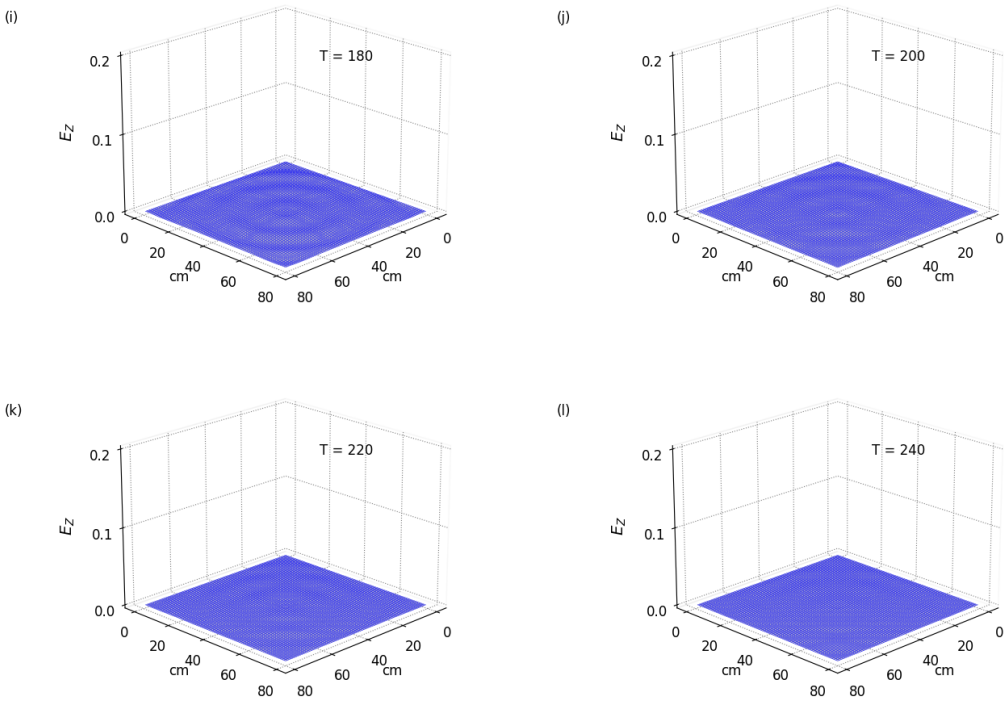

Figura 8.24: Campo eléctrico de una onda electromagnética en tres dimensiones con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (i) T=180, (j) T=200, (k) T=220 y (l) T=240, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo F.1.

En la Fig. 8.22, Fig. 8.23 y Fig. 8.24 se muestra la propagación de una onda electromagnética en tres dimensiones implementando condiciones de frontera absorbente por medio de las capas perfectamente combinadas PML. La onda inicia en la coordenada $(40, 40, 0)$, a medida que aumentan los pasos temporales la onda se extiende por el espacio computacional hasta alcanzar el límite computacional. Desde el paso temporal inicial $T = 20$ hasta el paso temporal $T = 80$, la onda electromagnética en tres dimensiones se propaga sin ninguna perturbación por el entorno computacional, obteniendo el mismo resultado que en el caso con condiciones de frontera PEC descritos en la sección 8.6, resultado que debe ser esperado dado qué las condiciones de frontera son diseñadas únicamente en el límite computacional. Después del paso temporal $T = 80$ hasta el paso temporal final $T = 240$, la onda electromagnética en tres dimensiones disminuye su amplitud en dirección z, donde es notorio que la onda es absorbida por el límite computacional sin reflejar la onda incidente. Al compararlo con el caso con condiciones de frontera PEC descrito en la sección 8.6, se evidencia una notable disminución en la

intensidad de la onda incidente una vez que llega al límite computacional. Para constatar la efectividad del diseño de las capas perfectamente combinadas PML y compararlas con las condiciones de frontera PEC, se representa la onda electromagnética en tres dimensiones junto a un mapa de contorno para analizar los máximos locales de la componente z. Para esto se una antena dipolo de onda continua (pulso sinusoidal) en el plano $z = 0$ con origen en $(40, 40, 0)$.

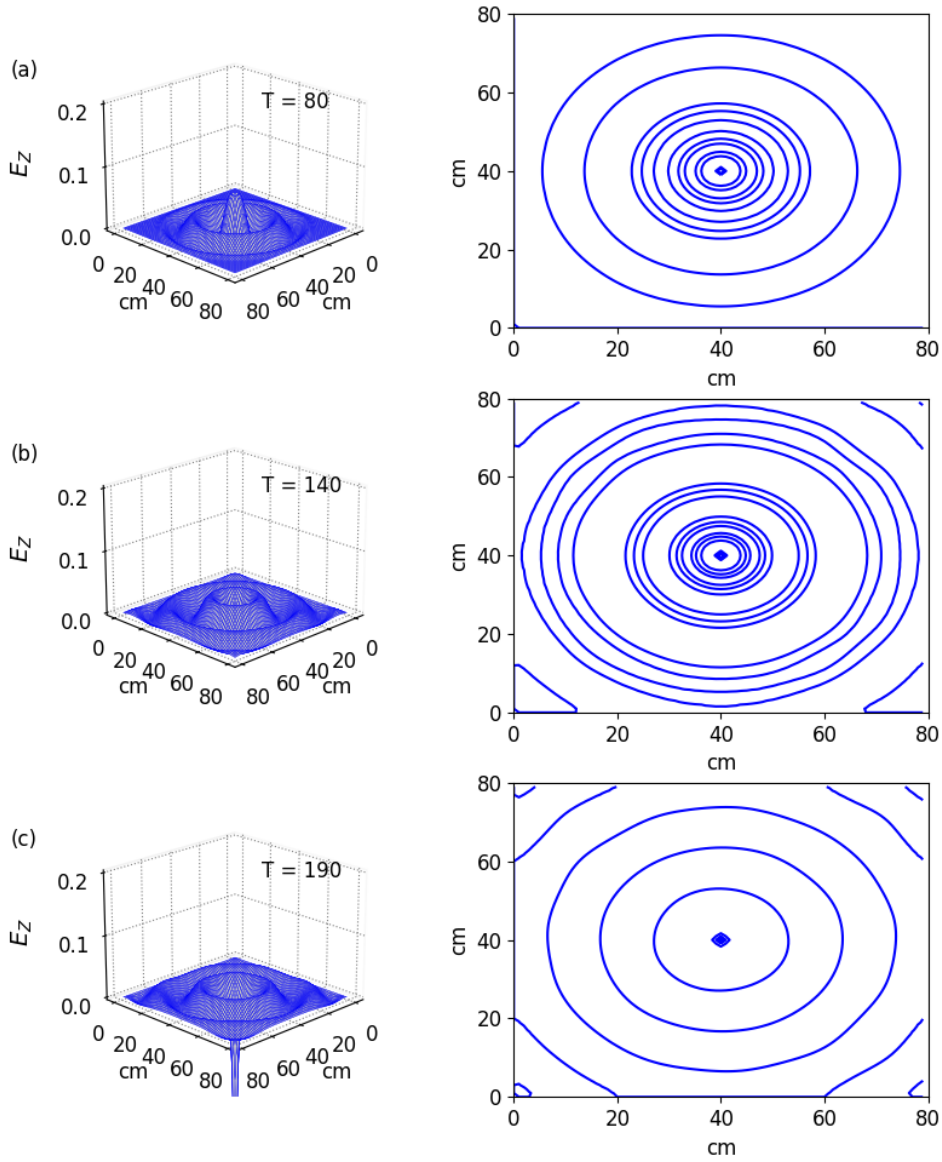

Figura 8.25: Distribución de onda electromagnética en tres dimensiones con condiciones de frontera PML con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=80, (b) T=140 y (c) T=190, de una antena dipolo de onda continua (pulso sinusoidal) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo F.2.

En la Fig. 8.25 se observa una onda electromagnética en tres dimensiones junto a un mapa de contorno para tres pasos temporales imponiendo condiciones de frontera absorbente. Después del paso temporal $T = 80$ la onda electromagnética llega al límite computacional, luego de estos pasos, se puede discernir la onda incidente en cada punto del espacio, como se evidencia en el paso temporal $T = 140$ y $T = 190$.

Ahora, se diseña un mapa de contorno junto a una barra de altura que permita medir la intensidad de la onda incidente y la onda reflejada en amplitud, midiendo la amplitud relativa de los máximo en la componente z para un paso temporal cuando llegue la onda en tres dimensiones al límite computacional constituido por capas perfectamente combinadas PML.

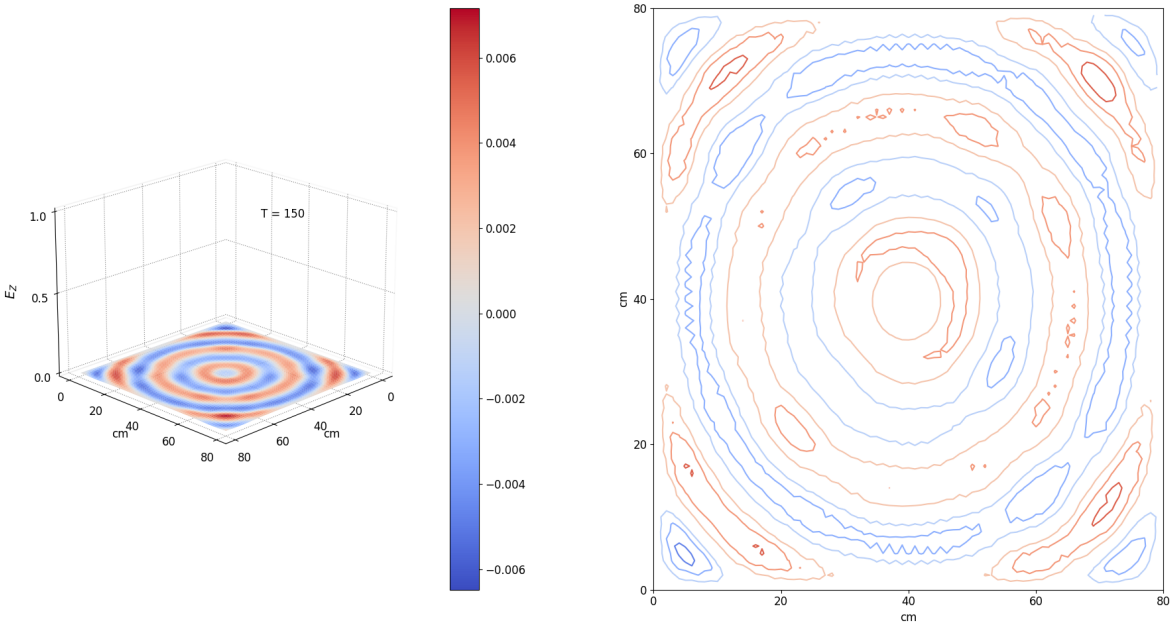

Figura 8.26: Distribución de los máximos del campo eléctrico en la componente z con condiciones de frontera PML con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, después de ciento cincuenta pasos temporales $T = 150$, de una antena dipolo de onda discontinua (pulso gaussiano) en el plano $z = 0$ con origen en $(40, 40, 0)$. Ver el código fuente en el Anexo F.3.

En la Fig. 8.26 se muestra una onda electromagnética en tres dimensiones con condiciones de frontera PML en el paso temporal $T = 150$, junto a una barra de alturas y un mapa de contorno. La onda reflejada tiene una amplitud relativa de 0,006, mucho menor que en el caso con condiciones de frontera PEC, donde alcanzó una amplitud relativa de 0,02 en el mismo paso temporal, como se observa en la Fig. 8.21, evidenciando una notable reducción de la amplitud de la onda reflejada gracias a las capas perfectamente combinadas diseñadas en el límite computacional.
En esta sección, se obtuvo la representación gráfica de las ecuaciones de Maxwell en tres dimensiones con condiciones de frontera absorbente, implementando el diseño de capas perfectamente emparejadas en el límite computacional, implementando el método de diferencias finitas en el dominio del tiempo, logrando la visualización de los campos magnéticos y eléctricos por medio de simulaciones computacionales.

9. Frecuencia de corte

El presente capítulo tiene como fin obtener las frecuencias de corte para una guía de onda unidimensional, desarrollado en la sección 4.6, implementando condiciones de frontera de un conductor perfecto en las paredes de la guía de onda.

9.1. Guía de onda unidimensional

Se considera una guía de onda unidimensional con longitud L. Las paredes de la guía de onda están conformadas por un conductor perfecto PEC, y, por lo tanto, los campos electromagnéticos que puede haber en el interior de la guía de onda no tienen el carácter de una onda viajera, puesto que están confinadas, como se vio en la sección 4.6. Las ondas electromagnéticas que se encuentran dentro de la guía de onda experimentan reflexiones continuas sobre las superficies del sistema y tienden a adoptar la forma de ondas estacionarias, que corresponden con la geometría de la cavidad.

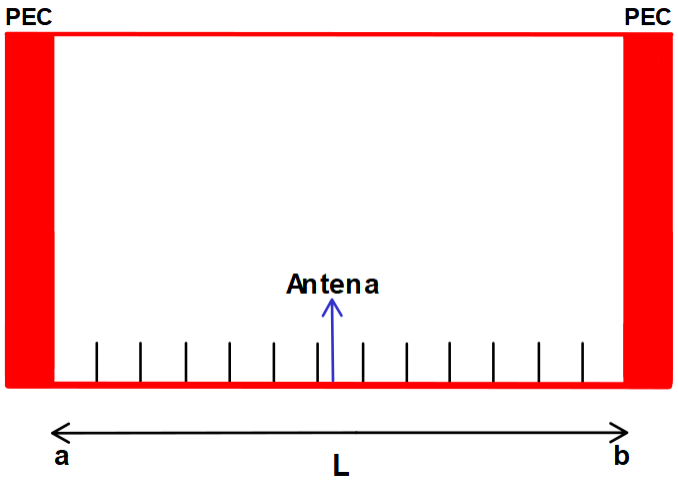

Figura 9.1: Representación de una guía de onda unidimensional de longitud L con condición de frontera PEC con una antena en el interior.

El diseño de la guía de ondas tiene como fin encontrar los modos de propagación. Cada modo de propagación de la guía de onda tiene su propia frecuencia característica, que en este caso, son las frecuencias características de una onda electromagnética unidimensional. Para esto, se diseña una guía de onda como se muestra en la Fig. 9.1. La onda electromagnética confinada en la guía de onda es una onda unidimensional con condiciones de frontera de un conductor perfecto PEC. La evolución temporal de los campos está determinada por las ecuaciones 7.8 y 7.9, que corresponden con las ecuaciones de Maxwell discretrizadas en el caso unidimensional con condiciones de frontera PEC.
Dentro de la guía de onda se generará una señal Blackman-Harris Window, descrita en la sección 5.4.3. Esta función proporciona una señal periódica en el dominio del

tiempo, de modo que al aplicar una transformación rápida de Fourier se obtiene un amplio rango de frecuencias de esta señal, que, en este caso, corresponderían con las frecuencias características de una onda electromagnética unidimensional. Este pulso tendrá una frecuencia mínima de $f_{\text{min}} = 0$, y una frecuencia máxima de $f_{\text{máx}} = 3 \times 10^9$ Hz, además, esta señal tiene una frecuencia característica dada por la siguiente relación:

$$f_c = \frac{f_{\text{máx}}}{3{,}351}.$$

En el interior de la guía de onda estará ubicada una antena, como se observa en la Fig. 9.1, cuyo fin es recibir una señal en el dominio del tiempo, y, al aplicar una transformación rápida de Fourier, se obtendrán las frecuencias características de la guía de onda unidimensional. La guía de onda tendrá una longitud de un metro $L = 1$ m. La posición de la fuente estará ubicada en $\frac{\sqrt{2}}{2}$ m y la posición de la antena estará ubicada en $\frac{\sqrt{2}}{4}$ m (estás posiciones se ponen en raíz para excitar más modos en la guía de onda).

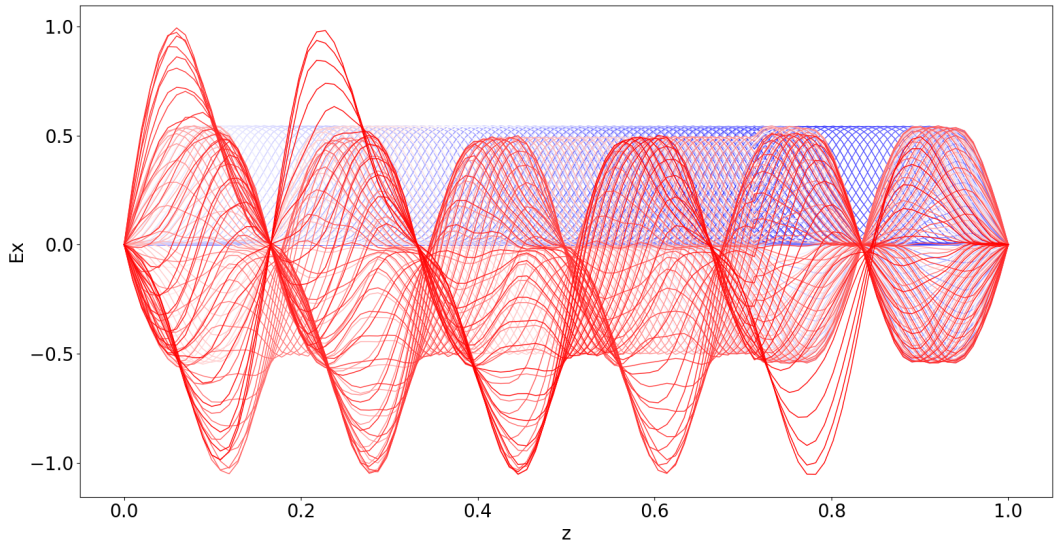

Figura 9.2: Distribución del campo eléctrico E_x en una guía de onda unidimensional de un metro de longitud $L = 1$ m, $(0 < z < 1)$, con condiciones de frontera PEC, después de cinco periodos $T = 5$, de un pulso Blackman-Harris Window con origen en $\frac{\sqrt{2}}{2}$ m, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo G.

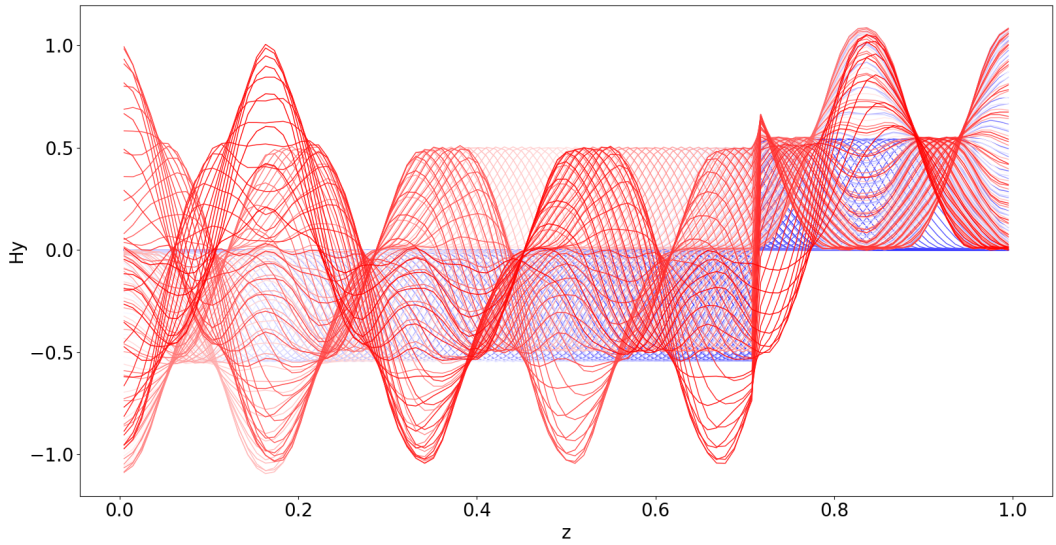

Figura 9.3: Distribución del campo magnético H_y en una guía de onda unidimensional de un metro de longitud $L = 1$ m, $(0 < z < 1)$, con condiciones de frontera PEC, después de cinco periodos $T = 5$, de un pulso Blackman-Harris Window con origen en $\frac{\sqrt{2}}{2}$ m, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo G.

Al implementar las ecuaciones 7.8 y 7.9 en una dimensión con condiciones de frontera de un conductor perfecto en el lenguaje de programación Python para $T = 5$ periodos de la señal Blackman-Harris Window se obtiene el campo eléctrico en la componente E_x y el campo magnético en la componente H_y, propagándose en dirección z como se muestra en la Fig. 9.2 y Fig. 9.3, respectivamente. Se puede observar que los campos se encuentran en fase. En los gráficos se observa la evolución de los campos a medida que aumenta el periodo, teniendo como azul los primeros periodos, y como rojo los últimos periodos.
Tanto el campo eléctrico como el campo magnético experimentan reflexiones continuas sobre las paredes de la guía de onda, resultados que deben ser esperados dado que las paredes están compuestas por un conductor perfecto. Con esto, es posible conseguir que los campos electromagnéticos adopten la forma de ondas estacionarias, logrando calcular una transformación rápida de Fourier para así calcular las frecuencias características de una guía de onda unidimensional.

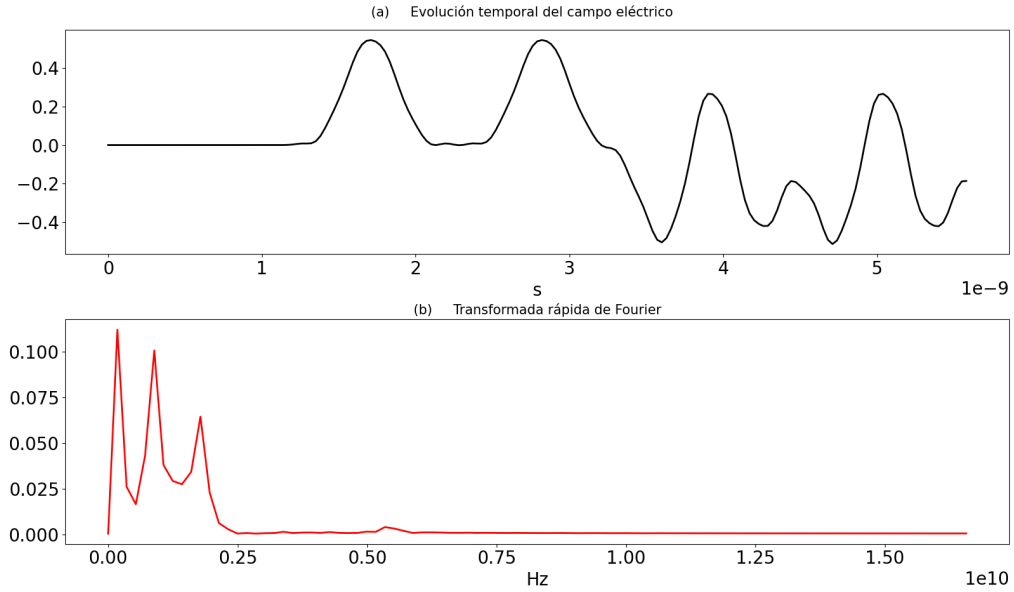

Figura 9.4: (a) Evolución temporal del campo eléctrico E_x, (b) Transformada rápida de Fourier para el campo eléctrico después de cinco periodos $T = 5$ de un pulso Blackman-Harris Window. Ver el código fuente en el Anexo G.

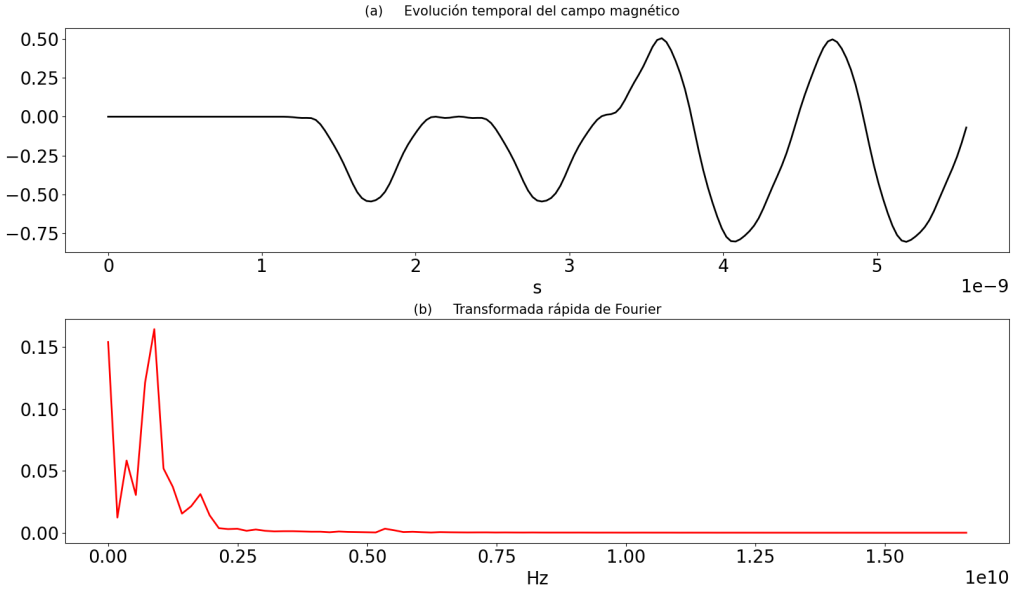

Figura 9.5: (a) Evolución temporal del campo magnético H_y, (b) transformada rápida de Fourier para el campo magnético después de cinco periodos $T = 5$ de un pulso Blackman-Harris Window. Ver el código fuente en el Anexo G.

Las Fig. 9.4 y Fig. 9.5 muestran las gráficas correspondientes a la evolución temporal (a) y la transformación rápida de Fourier (b) para el campo eléctrico y campo magnético, respectivamente. La transformación rápida de Fourier da unos valores con máximos que corresponden con las frecuencias de corte de los modos de la guía de onda unidimensional. Se pueden observar cuatro máximos tanto para el campo eléctrico como para el campo magnético. Cabe resaltar que el orden de los máximos no necesariamente corresponden con los cuatro primeros modos de la guía de onda unidimensional.
A partir de la ecuación 4.30 se obtiene los valores analíticos para los primeros modos de una guía de onda para el caso en que $a = 0$ y $b = 1$, y se hace una comparación con los valores obtenidos en los máximos de la Fig. 9.4 y Fig. 9.5, valores que se encontraron al discretizar las ecuaciones de Maxwell en una dimensión con condiciones de frontera PEC a través del método de diferencias finitas en el dominio del tiempo.

Modo	Frecuencia teórica (GHz)	Frecuencia numérica (GHz)	Desviación (%)
1	0.150	0.160	6.250
6	0.900	0.900	0.001

Tabla 9.1: Valores analíticos y numéricos de las frecuencias características de los primeros modos de una guía de onda unidimensional para el campo eléctrico después de cinco periodos $T = 5$ de la señal Blackman-Harris Window.

Modo	Frecuencia teórica (GHz)	Frecuencia numérica (GHz)	Desviación (%)
2	0.300	0.333	9.910
6	0.900	0.900	0.001

Tabla 9.2: Valores analíticos y numéricos de las frecuencias características de los primeros modos de una guía de onda unidimensional para el campo magnético después de cinco periodos $T = 5$ de la señal Blackman-Harris Window.

En la Tabla 9.3 y Tabla 9.4 se muestran los valores de los modos analíticos y los modos encontrados por medio de la simulación numérica. Para el campo eléctrico, los dos primeros máximos de la Fig. 9.4 equivalen con el primer modo y sexto modo de la guía de onda, mientras que los siguientes dos máximos no corresponden con ningún modo de la guía de onda unidimensional. Al comparar el valor analítico con el valor numérico para el primer modo se evidencia una desviación relativamente grande, mientras que para el sexto modo la desviación no difiere significativamente.

Para el campo magnético, los dos primeros máximos de la Fig. 9.5 equivalen con el segundo modo y sexto modo, mientras que los siguientes dos máximos no corresponden con ningún modo de la guía de onda unidimensional. Al comparar el valor analítico con el valor numérico para el segundo modo se evidencia un desviación relativamente grande, mientras que para el sexto modo la desviación no difiere significativamente. Los resultados para los dos casos muestran que para periodos bajos de la señal dentro de la guía de onda no es posible encontrar más de dos modos. Una nota importante es que tanto para el campo eléctrico y campo magnético, se encontró el valor del sexto modo con bastante precisión y corresponden con el segundo máximo. Además, el tercer y cuarto máximo en ambos casos no tienen asociado un modo característico, ya que toma valores muy grandes en comparación con los seis primeros modos.
Con el fin de obtener varios modos en la guía de onda unidimensional es necesario tomar varios periodos de la señal Blackman-Harris Window, para este caso se simula la evolución de los campos para 87 periodos.

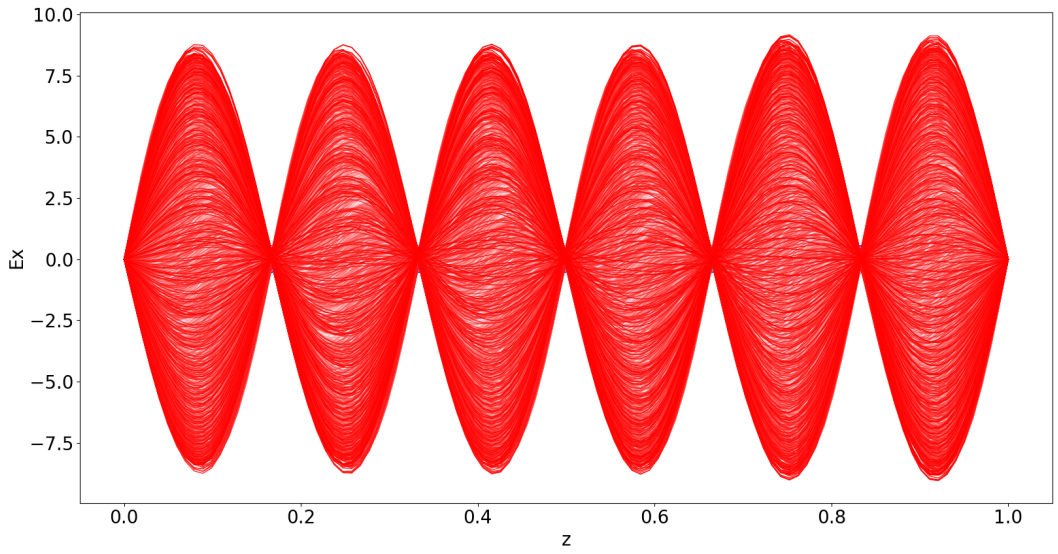

Figura 9.6: Distribución del campo eléctrico E_x en una guía de onda unidimensional de un metro de longitud $L = 1$ m, $(0 < z < 1)$, con condiciones de frontera PEC, después de ochenta y siete periodos $T = 87$, de un pulso Blackman-Harris Window con origen en $\frac{\sqrt{2}}{2}$ m, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo G.

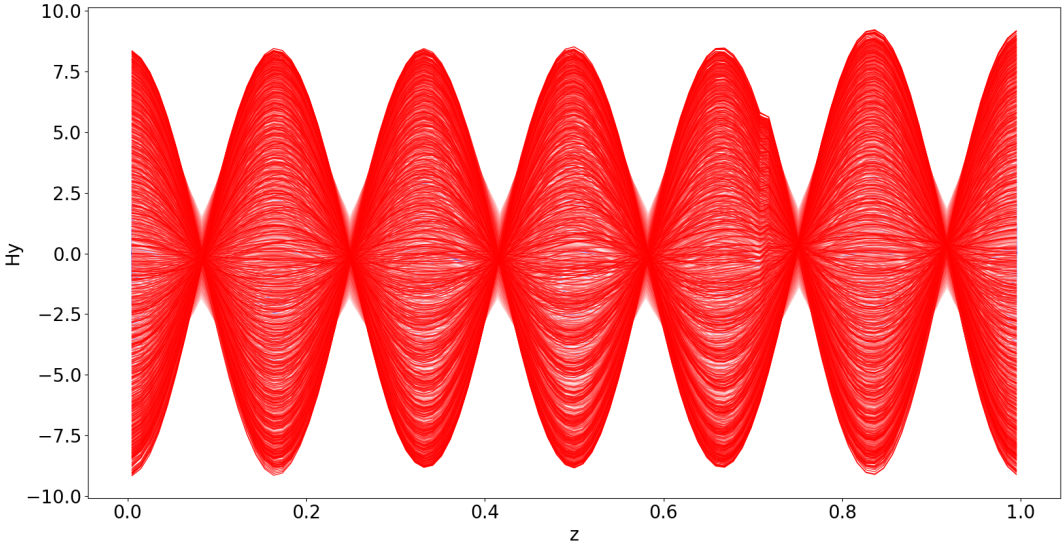

Figura 9.7: Distribución del campo magnético H_y en una guía de onda unidimensional de un metro de longitud $L = 1$ m, $(0 < z < 1)$, con condiciones de frontera PEC, después de ochenta y siete periodos $T = 87$, de un pulso Blackman-Harris Window con origen en $\frac{\sqrt{2}}{2}$ m, que se propaga a lo largo del eje z. Ver el código fuente en el Anexo G.

Al implementar nuevamente las ecuaciones 7.8 y 7.9 en una dimensión con condiciones de frontera de un conductor perfecto se obtiene el campo eléctrico en la componente E_x y el campo magnético en la componente H_y, propagándose en dirección z como se muestra en la Fig. 9.7 y Fig. 9.7, respectivamente. Dado que el periodo es mucho mayor en este caso, es evidente que tanto el campo eléctrico como el campo magnético experimentan múltiples reflexiones continuas sobre las paredes de la guía de onda, consiguiendo aumentar el paso temporal y así obtener más modos dentro de la guía de onda mediante la transformación rápida de Fourier.
Las Fig. 9.8 y Fig. 9.9 muestran las gráficas correspondientes a la evolución temporal (a) y la transformación rápida de Fourier (b) para campo eléctrico y campo magnético, respectivamente. Lo primero que se puede evidenciar es un aumento significativo en la evolución temporal de los campos, resultado que debe ser esperado debido al aumento en el periodo de la señal dentro de la guía de onda, quedando así por evidente que el aumento en el tiempo de la simulación influye de manera positiva en los resultados. Además de esto, hay un aumento en los máximos tanto para el campo eléctrico como para el campo magnético, donde se aprecian 6 máximos para los dos casos.

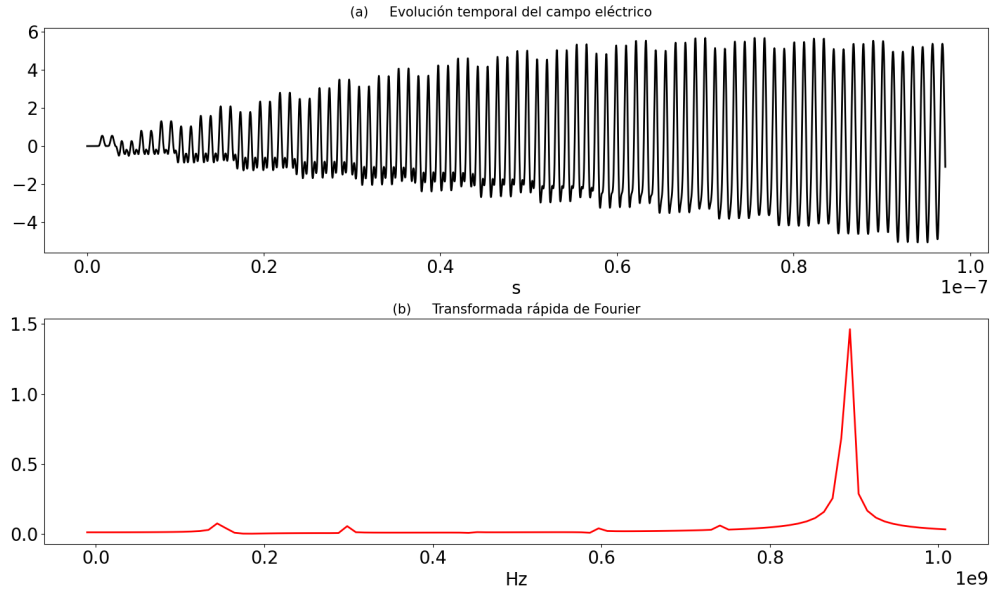

Figura 9.8: (a) Evolución temporal del campo eléctrico E_x, (b) Transformada rápida de Fourier para el campo eléctrico después de ochenta y siete periodos $T = 87$ de un pulso Blackman-Harris Window. Ver el código fuente en el Anexo G.

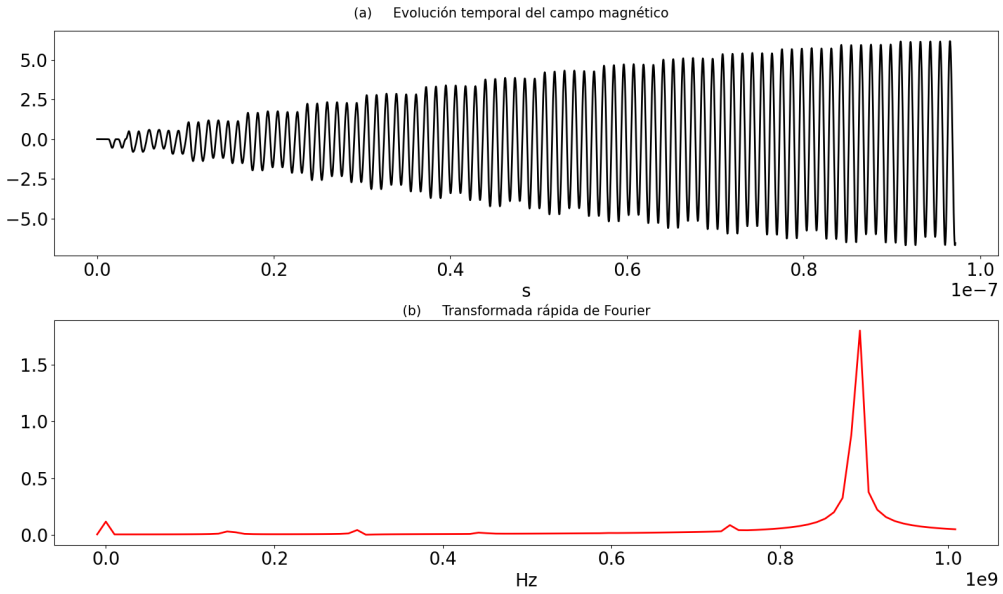

Figura 9.9: (a) Evolución temporal del campo magnético H_y, (b) transformada rápida de Fourier para el campo magnético después de ochenta y siete periodos $T = 87$ de un pulso Blackman-Harris Window. Ver el código fuente en el Anexo G.

Con el fin de evidenciar la precisión del método de diferencias finitas en el dominio del tiempo para el diseño de una guía de onda unidimensional, se comparan nuevamente los valores analíticos con los valores obtenidos a través de la simulación numérica de la Fig. 9.8 y Fig. 9.9. En la Tabla 9.3 y Tabla 9.4 se muestran los valores de los modos teóricos y los modos encontrados por medio de la simulación numérica.

Modo	Frecuencia teórica (GHz)	Frecuencia numérica (GHz)	Desviación (%)
1	0.150	0.144	2.740
2	0.300	0.298	0.671
3	0.450	0.451	0.222
4	0.600	0.599	0.167
5	0.750	0.742	1.078
6	0.900	0.897	0.334

Tabla 9.3: Valores analíticos y numéricos de las frecuencias características de los primeros seis modos de una guía de onda unidimensional para el campo eléctrico después de ochenta y siete periodos $T = 87$ de la señal Blackman-Harris Window.

Modo	Frecuencia teórica (GHz)	Frecuencia numérica (GHz)	Desviación (%)
1	0.150	0.146	2.740
2	0.300	0.297	1.010
3	0.450	0.442	1.810
4	0.600	0.590	1.695
5	0.750	0.743	0.942
6	0.900	0.894	0.671

Tabla 9.4: Valores analíticos y numéricos de las frecuencias características de los primeros seis modos de una guía de onda unidimensional para el campo magnético después de ochenta y siete periodos $T = 87$ de la señal Blackman-Harris Window.

Al comparar los valores analíticos y con los valores numéricos de las frecuencias características para una guía de onda unidimensional se evidencia que la desviación no difiere significativamente, esto se cumple tanto para el campo eléctrico como para el campo magnético. Estos resultados nos permite corroborar que un aumento en el número de periodos de la señal dentro de la guía de onda logra excitar un número mayor de modos y con esto aumentar la precisión del método numérico en los resultados.
Por medio del método de diferencias finitas en el dominio del tiempo, fue posible encontrar una solución numérica a las ecuaciones de Maxwell en una dimensión para medios homogéneos e isotrópicos, obteniendo valores con una precisión bastante alta para las frecuencias de corte de una guía de onda unidimensional.

10. Conclusiones

El objetivo de este trabajo de grado ha sido obtener simulaciones numéricas para el estudio de la propagación de ondas electromagnéticas en medios homogéneos e isotrópicos, por medio del método de diferencias finitas en el dominio del tiempo. A continuación y como resumen del trabajo realizado, se enumeran las principales aportaciones y conclusiones del mismo:

1. Se ha realizado un análisis de las características principales de los campos electromagnéticos a través de las ecuaciones de Maxwell en el vacío y la materia, analizando la continuidad y/o discontinuidad de las componentes tangenciales y normales del campo eléctrico y campo magnético.

2. Se consideró la propagación de ondas electromagnéticas confinadas en el interior de una guía de onda, con el fin de encontrar un expresión que permita calcular las frecuencias de corte de una guía de onda unidimensional.

3. Se han desarrollado los elementos necesarios para abordar la discretización de las componentes espaciales y temporales por medio del método de diferencias finitas en el dominio del tiempo, en el que por medio de las celdas de Yee es posible discretizar las ecuaciones de Maxwell, permitiendo calcular la evolución temporal del campo electromagnético en un punto de interés.

4. Se consideraron las condiciones necesarias para velar por la estabilidad del método numérico, de modo que la simulación numérica no presente errores numéricos al ejecutarla.

5. Se ha realizado un análisis de las condiciones de frontera para el entorno computacional, donde se destacan las condiciones de frontera de un conductor perfectamente eléctrico PEC, y condiciones de frontera absorbente PML.

6. Se ha realizado una profunda revisión de las bases matemáticas que permiten discretizar las ecuaciones de Maxwell por medio del método de diferencias finitas en el dominio del tiempo. Se discretizaron las ecuaciones de Maxwell en una, dos y tres dimensiones, tomando como referencia las celdas de Yee, en el que se aplicaron condiciones de frontera de un conductor perfectamente eléctrico y condiciones de frontera absorbente en el límite computacional.

7. Se han desarrollado códigos en el lenguaje de programación Python que permiten obtener la solución numérica de los campos electromagnéticos en una, dos y tres dimensiones, aplicando condiciones de frontera de un conductor perfectamente eléctrico y condiciones de frontera absorbente.

8. Con el fin de desarrollar y verificar la utilidad del método de diferencias finitas en el dominio del tiempo, se obtuvo la solución numérica de las ecuaciones de Maxwell en una, dos y tres dimensiones, permitiendo obtener la representación gráfica de

la evolución temporal de los campos electromagnéticos, aplicando condiciones de frontera de un conductor perfectamente eléctrico y condiciones de frontera absorbente.

9. Como aplicación de las simulaciones numéricas obtenidas, se obtuvo las frecuencias de corte para una guía de onda conformada por un conductor perfecto PEC en una dimensión.

10. Se obtuvieron frecuencias numéricas con una desviación bastante favorable con respecto a las frecuencias analíticas. Donde se discutió la relación existen entre el número de periodos generados dentro de la guía de onda y el número de modos obtenidos.

11. En los anexos de este trabajo se presentan los códigos diseñados en el lenguaje de programación Python. Además, se ha creado una carpeta en Drive con un enlace para que sean accesibles. Estos códigos estarán disponibles de manera gratuita para el público en general, con el objetivo de contribuir a futuras investigaciones en el campo de la electrodinámica computacional.

Anexos

A continuación se presentan todos los códigos utilizados en este trabajo. Están disponibles en el siguiente enlace de acceso libre a una carpeta de Google Drive: **Códigos Python Solución de las ecuaciones de Maxwell a través del método de diferencias finitas en el dominio del tiempo en una, dos y tres dimensiones con condiciones de frontera PEC y PML.**

Código QR para acceder a los códigos.

A. Onda unidimensional con condiciones de frontera PEC

A.1. Distribución de los campos eléctrico E_x y magnético H_y con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional $(0 < z < 100)$, después de cien pasos temporales $T = 100$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z.

```
# Librerias
import numpy as np
from math import exp
from matplotlib import pyplot as plt
```

```
# Parametros de las matrices
ke = 101
ex = np.zeros(ke)
hy = np.zeros(ke)
# Parametros del pulso
kc = int(ke / 2)
t0 = 50
ancho = 12
npasos = 100
# Ecuaciones discretizadas, programa principal
for tiempo_paso in range(1, npasos  + 1):
     # Calculo del campo eléctrico Ex
     for k in range(1, ke-1):
         ex[k] = ex[k] + 0.5 * (hy[k - 1] - hy[k])
     # Pulso Gaussiano
     pulso = exp(-0.5 * ((t0 - tiempo_paso) / ancho ) ** 2)
     ex[kc] = pulso
     # Calculo del campo magnético Hy
     for k in range(0, ke-1):
         hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])



# Diseño de las graficas para el campo eléctrico y campo magnético
plt.rcParams['font.size'] = 12
plt.figure(figsize=(8, 3.5))
plt.subplot(211)
plt.plot(ex, color='blue', linewidth=1)
plt.ylabel('E$_x$', fontsize='14')
plt.xticks(np.arange(0, 101, step=20))
plt.xlim(0, 100)
plt.yticks(np.arange(-1, 1.2, step=1))
plt.ylim(-1.2, 1.2)
plt.annotate('', xy=(20,1.1),fontsize=25, xytext=(15,1.1),
arrowprops=dict(color='blue', arrowstyle='<-'))
plt.annotate('', xy=(85,1.1),fontsize=25,xytext=(80,1.1),
arrowprops=dict(color='blue', arrowstyle='->'))
plt.text(50, 0.5, 'T = {}'.format(tiempo_paso),
horizontalalignment='center')
plt.subplot(212)
plt.plot(hy, color='red', linewidth=1)
plt.ylabel('H$_y$', fontsize='14')
plt.xlabel('z')
```

```
plt.xticks(np.arange(0, 101, step=20))
plt.xlim(0, 100)
plt.yticks(np.arange(-1, 1.2, step=1))
plt.ylim(-1.2, 1.2)
plt.annotate('', xy=(20,1.1),fontsize=25,xytext=(15,1.1),
arrowprops=dict(color='red',arrowstyle='<-'))
plt.annotate('', xy=(85,1.1),fontsize=25, xytext=(80,1.1),
arrowprops=dict(color='red',arrowstyle='->'))
plt.subplots_adjust(bottom=0.2, hspace=0.45)
plt.text(50, 0.5, 'T = {}'.format(tiempo_paso),
horizontalalignment='center')
plt.show()
```

A.2. Distribución del campo eléctrico E_x con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional $(0 < z < 100)$, para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z.

```
# Librerias
import numpy as np
from math import exp
from matplotlib import pyplot as plt
# Parametros de las matrices
ke = 102
ex = np.zeros(ke)
hy = np.zeros(ke)
# Parametros del pulso
kc = int(ke / 2)
t0 = 50
ancho = 12
npasos = 191

# Guardar puntos para luego graficarlos
trazar_puntos = [
{'numero_pasos': 100, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 130, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 160, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 190, 'datos_para_trazar': None, 'etiqueta': 'z'},
```

```
]
# Ecuaciones discretizadas, programa principal
for tiempo_paso in range(1, npasos + 1):
     # Calculo del campo eléctrico Ex
     for k in range(1, ke-1):
         ex[k] = ex[k] + 0.5 * (hy[k - 1] - hy[k])
     # Pulso Gaussiano
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / ancho) ** 2)
     ex[kc] = pulse
      # Calculo del campo magnético Hy
     for k in range(0, ke - 1):
         hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])
     # Guardar datos para graficarlos
     for trazar_punto in trazar_puntos:
         if tiempo_paso == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(ex)
# Diseño de las graficas para el campo magnético
plt.rcParams['font.size'] = 12
fig = plt.figure(figsize=(8, 5.25))
def plot_e_field(data, timestep, etiqueta):
     plt.plot(data, color='blue', linewidth=1)
     plt.ylabel('E$_x$', fontsize='14')
     plt.xticks(np.arange(0, 101, step=20))
     plt.xlim(0, 100)
     plt.yticks(np.arange(-1, 1.2, step=1))
     plt.ylim(-1.2, 1.2)
     plt.annotate('', xy=(20,1),fontsize=25, xytext=(15,1),
     arrowprops=dict(color='blue', arrowstyle='<-'))
     plt.annotate('', xy=(85,1),fontsize=25,xytext=(80,1),
     arrowprops=dict(color='blue', arrowstyle='->'))
     plt.text(50, 0.5, 'T = {}'.format(timestep),
     horizontalalignment='center')
     plt.xlabel('{}'.format(etiqueta))
# Grafica el campo eléctrico en los puntos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(4, 1, subplot_num + 1)
     plot_e_field(trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.tight_layout()
plt.show()
```

A.3. Distribución del campo magnético H_y con condiciones de frontera PEC a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z.

```
 # Librerias
import numpy as np
from math import exp, sin, pi
from matplotlib import pyplot as plt
# Parametros de las matrices
ke = 102
ex = np.zeros(ke)
hy = np.zeros(ke)
# Parametros del pulso
kc = int(ke / 2)
t0 = 50
ancho = 12
npasos = 191

# Guardar puntos para luego graficarlos
trazar_puntos = [
{'numero_pasos': 160, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 190, 'datos_para_trazar': None, 'etiqueta': 'z'},

]
# Ecuaciones discretizadas, programa principal
for tiempo_paso in range(1, npasos + 1):
     # Calculo del campo eléctrico Ex
     for k in range(1, ke - 1):
         ex[k] = ex[k] + 0.5 * (hy[k - 1] - hy[k])
     # Pulso Gaussiano
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / ancho) ** 2)
     ex[kc] = pulse
      # Calculo del campo magnético Hy
     for k in range(0, ke - 1):
         hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])
     # Guardar datos para graficarlos
     for trazar_punto in trazar_puntos:
         if tiempo_paso == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(hy)
# Diseño de las graficas para el campo magnético
```

```
plt.rcParams['font.size'] = 12
fig = plt.figure(figsize=(8, 5.25))
def plot_e_field(data, timestep, etiqueta):
     plt.plot(data, color='red', linewidth=1)
     plt.ylabel('H$_y$', fontsize='14')
     plt.xticks(np.arange(0, 101, step=20))
     plt.xlim(0, 100)
     plt.yticks(np.arange(-1, 1.2, step=1))
     plt.ylim(-1.2, 1.2)
     plt.annotate('', xy=(20,1.1),fontsize=25, xytext=(15,1.1),
     arrowprops=dict(color='red', arrowstyle='->'))
     plt.annotate('', xy=(85,1.1),fontsize=25,xytext=(80,1.1),
     arrowprops=dict(color='red', arrowstyle='<-'))
     plt.text(50, 0.5, 'T = {}'.format(timestep),
     horizontalalignment='center')
     plt.xlabel('{}'.format(etiqueta))
# Grafica el campo eléctrico en los puntos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 1, subplot_num + 1)
     plot_e_field(trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.tight_layout()
plt.show()
```

B. Onda unidimensional con condiciones de frontera PML

B.1. Distribución del campo eléctrico E_x con condiciones de frontera PML a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z.

```
# Librerias
import numpy as np
from math import exp
from matplotlib import pyplot as plt
#Parametros de las matrices
ke = 102
ex = np.zeros(ke)
```

```
hy = np.zeros(ke)
# Pulse parameters
kc = int(ke / 2)
t0 = 50
ancho = 12
npasos = 300

#Condiciones de frontera

condicion_baja = [0, 0]
condicion_alta = [0, 0]

# Guardar puntos para luego graficarlos
trazar_puntos = [
{'numero_pasos': 100, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 130, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 160, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 190, 'datos_para_trazar': None, 'etiqueta': 'z'},

]
# Ecuaciones discretizadas, programa principal
for tiempo_paso  in range(1, npasos + 1):
      # Calculo del campo eléctrico Ex
      for k in range(1, ke - 1):
           ex[k] = ex[k] + 0.5 * (hy[k - 1] - hy[k])
      #  Pulso Gaussiano
      pulse = exp(-0.5 * ((t0 - tiempo_paso ) / ancho) ** 2)
      ex[kc] = pulse
      # Condiciones de frontera absorbente
      ex[0] = condicion_baja.pop(0)
      condicion_baja.append(ex[1])
      ex[ke - 1] = condicion_alta.pop(0)
      condicion_alta.append(ex[ke - 2])
      # Calculo del campo magnético Hy
      for k in range(0, ke - 1):
           hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])
      # Guardar datos para graficarlos
      for trazar_punto in trazar_puntos:
         if tiempo_paso  == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(ex)
# Diseño de las graficas para el campo eléctrico
plt.rcParams['font.size'] = 12
fig = plt.figure(figsize=(8, 5.25))
```

```
def plot_e_field(data, timestep, etiqueta):
     plt.plot(data, color='blue', linewidth=1)
     plt.ylabel('E$_x$', fontsize='14')
     plt.xticks(np.arange(0, 101, step=20))
     plt.xlim(0, 100)
     plt.yticks(np.arange(-1, 1.2, step=1))
     plt.ylim(-1.2, 1.2)
     plt.annotate('', xy=(20,1.1),fontsize=25, xytext=(15,1.1),
     arrowprops=dict(color='blue', arrowstyle='<-'))
     plt.annotate('', xy=(85,1.1),fontsize=25,xytext=(80,1.1),
     arrowprops=dict(color='blue', arrowstyle='->'))
     plt.text(50, 0.5, 'T = {}'.format(timestep),
     horizontalalignment='center')
     plt.xlabel('{}'.format(etiqueta))
# Grafica el campo eléctrico en los puntos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 1, subplot_num + 1)
     plot_e_field(trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.tight_layout()
plt.show()
```

B.2. Distribución del campo magnético H_y con condiciones de frontera PML a lo largo de cien celdas del espacio computacional ($0 < z < 100$), para los pasos temporales $T = 100$, $T = 130$, $T = 160$ y $T = 190$, de un pulso gaussiano con origen en $z = 50$, que se propaga a lo largo del eje z.

```
# Librerias
import numpy as np
from math import exp
from matplotlib import pyplot as plt
#Parametros de las matrices
ke = 102
ex = np.zeros(ke)
hy = np.zeros(ke)
# Pulse parameters
kc = int(ke / 2)
t0 = 50
ancho = 12
```

```
npasos = 300

#Condiciones de frontera

condicion_baja = [0, 0]
condicion_alta = [0, 0]

# Guardar puntos para luego graficarlos
trazar_puntos = [
{'numero_pasos': 160, 'datos_para_trazar': None, 'etiqueta': 'z'},
{'numero_pasos': 190, 'datos_para_trazar': None, 'etiqueta': 'z'},


]
# Ecuaciones discretizadas, programa principal
for tiempo_paso  in range(1, npasos + 1):
      # Calculo del campo eléctrico Ex
      for k in range(1, ke - 1):
           ex[k] = ex[k] + 0.5 * (hy[k - 1] - hy[k])
      #  Pulso Gaussiano
      pulse = exp(-0.5 * ((t0 - tiempo_paso ) / ancho) ** 2)
      ex[kc] = pulse
      # Condiciones de frontera absorbente
      ex[0] = condicion_baja.pop(0)
      condicion_baja.append(ex[1])
      ex[ke - 1] = condicion_alta.pop(0)
      condicion_alta.append(ex[ke - 2])
      # Calculo del campo magnético Hy
      for k in range(0, ke - 1):
           hy[k] = hy[k] + 0.5 * (ex[k] - ex[k + 1])
      # Guardar datos para graficarlos
      for trazar_punto in trazar_puntos:
         if tiempo_paso  == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(hy)
# Diseño de las graficas para el campo eléctrico
plt.rcParams['font.size'] = 12
fig = plt.figure(figsize=(8, 5.25))
def plot_e_field(data, timestep, etiqueta):
     plt.plot(data, color='red', linewidth=1)
     plt.ylabel('H$_y$', fontsize='14')
     plt.xticks(np.arange(0, 101, step=20))
     plt.xlim(0, 100)
     plt.yticks(np.arange(-1, 1.2, step=1))
```

```
     plt.ylim(-1.2, 1.2)
     plt.annotate('', xy=(20,1.1),fontsize=25, xytext=(15,1.1),
     arrowprops=dict(color='red', arrowstyle='<-'))
     plt.annotate('', xy=(85,1.1),fontsize=25,xytext=(80,1.1),
     arrowprops=dict(color='red', arrowstyle='->'))
     plt.text(50, 0.5, 'T = {}'.format(timestep), horizontalalignment='center')
     plt.xlabel('{}'.format(etiqueta))
# Grafica el campo eléctrico en los puntos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 1, subplot_num + 1)
     plot_e_field(trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.tight_layout()
plt.show()
```

C. Onda en dos dimensiones con condiciones de frontera PEC

C.1. Distribución del campo eléctrico en la componente E_z con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales. a) T=10, (b) T=20, (c) T=30, (d) T=40, (e) T=50, (f) T=60, (g) T=70, (h) T=80, (i) T=90, (j) T=100, (k) T=110, (l) T=120

```
#Programa 2D con condiciones PEC
import numpy as np
from math import exp
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d



#Parametros de la malla
ie = 80
je = 80
#Parametros del pulso
ic = int(ie / 2)
jc = int(je / 2)
# Discretización de los campos y parámetros
```

```
Ez = np.zeros((ie, je))
Dz = np.zeros((ie, je))
Hx = np.zeros((ie, je))
Hy = np.zeros((ie, je))
PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy

epsz=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*m0) #Velocidad de la luz

#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 # condición de estabilidad

# Parametros del pulso
# Parametros del pulso
t0 = 10
ancho = 6
npasos=130

# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'numero_pasos': 50, 'datos_para_trazar': None, 'etiqueta': 'e'},
{'numero_pasos': 60, 'datos_para_trazar': None, 'etiqueta': 'f'},
{'numero_pasos': 70, 'datos_para_trazar': None, 'etiqueta': 'g'},
{'numero_pasos': 80, 'datos_para_trazar': None, 'etiqueta': 'h'},

]
#Nucleo del programa FDTD 2D

for tiempo_paso in range(1, npasos + 1):
     # Calcular el campo electrico Dz
     for j in range(1, je):
         for i in range(1, ie):
             Dz[i, j] = Dz[i, j] + 0.5 * (Hy[i, j] - Hy[i - 1, j] -
             Hx[i, j] + Hx[i, j - 1])
     # Pulso gaussiano
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / ancho) ** 2)
     Dz[ic, jc] = pulse
     # Calcular el campo Ez a partir de Dz
     for j in range(1, je):
         for i in range(1, ie):
```

```
            Ez[i, j] = Dz[i, j]
    # Calcular el campo Hx
    for j in range(je - 1):
        for i in range(ie - 1):
            Hx[i, j] = Hx[i, j] + 0.5 * (Ez[i, j] - Ez[i, j + 1])
    # Calcular el campo Hy
    for j in range(je - 1):
        for i in range(ie - 1):
            Hy[i, j] = Hy[i, j] + 0.5 * (Ez[i + 1, j] - Ez[i, j])
    # Guardar los puntos para graficarlos
    for trazar_punto in trazar_puntos:
        if tiempo_paso == trazar_punto['numero_pasos']:
            trazar_punto['datos_para_trazar'] = np.copy(Ez)


# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(12, 12))
X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, etiqueta):
    ax.set_zlim(0, 1)
    ax.view_init(elev=20., azim=45)
    ax.plot_surface(X, Y, data[:, :], rstride=1, cstride=1,
    color='white', edgecolor='red', linewidth=.25)
    ax.zaxis.set_rotate_label(False)
    ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10,fontsize=14)
    ax.set_zticks([0, 0.5, 1])
    ax.set_xlabel('cm')
    ax.set_ylabel('cm')
    ax.set_xticks(np.arange(0, 81, step=20))
    ax.set_yticks(np.arange(0, 81, step=20))
    ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
    ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill= False
    plt.gca().patch.set_facecolor('white')
    ax.text2D(-0.2, 0.8, "({})".format(etiqueta), transform=ax.transAxes)
    ax.dist = 11

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
    ax = fig.add_subplot(2, 2, subplot_num +1,projection='3d')
```

```
    plot_e_field(ax, trazar_punto['datos_para_trazar'],
    trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.subplots_adjust(bottom=0, left=0.114,
hspace=0, wspace=0.376, top=1, right=0.729)
plt.show()
```

C.2. Máximos locales del campo eléctrico con condiciones de frontera PEC con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (a) T=80, (b) T=110 y (c) T=130, de un pulso sinusoidal con origen en $(40, 40)$.

```
    #Programa 2D  sin con condiciones PEC, pulso sin
import numpy as np
from math import sin, pi
from math import exp
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d



#Parametros de la malla
ie = 80
je = 80

#Parametros del pulso
ic = int(ie / 2)
jc = int(je / 2)

# Discretización de los campos y parámetros
Ez = np.zeros((ie, je))
Dz = np.zeros((ie, je))
Hx = np.zeros((ie, je))
Hy = np.zeros((ie, je))


PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
epsz=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*m0) #Velocidad de la luz
```

```
#Tamaño de la celda
ddx = 0.01


dt = CFL * ddx/c0 # condición de estabilidad

# Parametros del pulso
npasos = 130

# Segumiento de los puntos para guardarlos y graficarlos

trazar_puntos= [
{'etiqueta': 'a', 'numero_pasos': 80, 'datos_para_trazar': None},
{'etiqueta': 'b', 'numero_pasos':110, 'datos_para_trazar': None},
{'etiqueta': 'c', 'numero_pasos':130, 'datos_para_trazar': None} ]

#Nucleo del programa FDTD 2D
for tiempo_paso in range(1, npasos + 1):
     # Calcular el campo electrico Dz
     for j in range(1, je):
         for i in range(1, ie):
             Dz[i, j] = Dz[i, j] + 0.5 * (Hy[i, j] - Hy[i - 1, j]
             -Hx[i, j] + Hx[i, j - 1])
     # Pulso gaussiano
     pulse = sin(2 * pi * 1500 * 1e6 * dt * tiempo_paso)
     Dz[ic, jc] = pulse
     # Calcular el campo Ez a partir de Dz
     for j in range(1, je):
         for i in range(1, ie):
             Ez[i, j] = Dz[i, j]
     # Calcular el campo Hx
     for j in range(je - 1):
         for i in range(ie - 1):
             Hx[i, j] = Hx[i, j] + 0.5 * (Ez[i, j] - Ez[i, j + 1])
     # Calcular el campo Hy
     for j in range(je - 1):
         for i in range(ie - 1):
             Hy[i, j] = Hy[i, j] + 0.5 * (Ez[i + 1, j] - Ez[i, j])
     # Guardar los puntos para graficarlos
     for trazar_punto in trazar_puntos:
         if tiempo_paso == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(Ez)
```

```
# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(12, 12))
X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, label):
     ax.set_zlim(0, 1)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :], rstride=1,
     cstride=1, color='white', edgecolor='red', linewidth=.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10,fontsize=14)
     ax.set_zticks([0, 0.5, 1])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill= False
     plt.gca().patch.set_facecolor('white')
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.dist = 11
     #3, 3, subplot_num + 1
# Se grafica el campo para cada uno de los tiempos guardados
def plot_e_field_contour(ax, data): #grafica de nivel
     CP = plt.contour(X, Y, data, colors='red', linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(3, 3,subplot_num*3+1, projection='3d')
     plot_e_field(ax, trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
     ax = fig.add_subplot(3, 3, subplot_num * 3+2 )
     plot_e_field_contour(ax, trazar_punto['datos_para_trazar'])
plt.tight_layout()
```

```
plt.subplots_adjust(bottom=0.05, left=0.07, hspace=0.224,
wspace=0.183, top=0.967, right=0.67)
plt.show()
```

C.3. Distribución de los máximos del campo eléctrico con condiciones de frontera PEC con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, después de ciento quince pasos temporales $T = 115$, de un pulso gaussiano con origen en $(40, 40)$.

```
#Programa 2D con condiciones PEC
import numpy as np
from math import sin, pi
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d
from mpl_toolkits.mplot3d.axes3d import Axes3D
from matplotlib import cm
from math import exp

#Parametros de la malla
ie = 80
je = 80

#Parametros del pulso
ic = int(ie / 2)
jc = int(je / 2)

# Discretización de los campos y parámetros
Ez = np.zeros((ie, je))
Dz = np.zeros((ie, je))
Hx = np.zeros((ie, je))
Hy = np.zeros((ie, je))
gaz = np.ones((ie, je))

PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
epsz=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
```

```
c0=1/np.sqrt(epsz*m0) #Velocidad de la luz

#Tamaño de la celda
ddx = 0.01


dt = CFL * ddx/c0 #condición de estabilidad

# Parametros del pulso
t0 = 10
propagar = 6
npasos = 300


# Segumiento de los puntos para guardarlos y graficarlos

trazar_puntos= [
{'etiqueta': '', 'numero_pasos': 115, 'datos_para_trazar': None}]
#Nucleo del programa FDTD 2D

for tiempo_paso in range(1, npasos + 1):
     # Calcular el campo electrico Dz
     for j in range(1, je):
         for i in range(1, ie):
             Dz[i, j] = Dz[i, j] + 0.5 * (Hy[i, j] - Hy[i - 1, j] -
             Hx[i, j] + Hx[i, j - 1])
     # Pulso gaussiano
     pulse = exp(-0.5 * ((t0 -tiempo_paso) / propagar) ** 2)
     Dz[ic, jc] = pulse
     # Calcular el campo Ez a partir de Dz
     for j in range(1, je):
         for i in range(1, ie):
             Ez[i, j] = Dz[i, j]
     # Calcular el campo Hx
     for j in range(je - 1):
         for i in range(ie - 1):
             Hx[i, j] = Hx[i, j] + 0.5 * (Ez[i, j] - Ez[i, j + 1])
     # Calcular el campo Hy
     for j in range(je - 1):
         for i in range(ie - 1):
             Hy[i, j] = Hy[i, j] + 0.5 * (Ez[i + 1, j] - Ez[i, j])
     # Guardar los puntos para graficarlos
     for trazar_punto in trazar_puntos:
```

```
        if tiempo_paso == trazar_punto['numero_pasos']:
            trazar_punto['datos_para_trazar'] = np.copy(Ez)

# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))
X, Y = np.meshgrid(range(je), range(ie))
def plot_e_field(ax, data, timestep):
     ax.set_zlim(0, 1)
     ax.view_init(elev=20., azim=45)
     b=ax.plot_surface(X, Y, data, rstride=1, cstride=1,
     cmap=cm.coolwarm, linewidth=.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.set_zticks([0, 0.5, 1])
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     plt.colorbar(b)
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11
def plot_e_field_contour(ax, data):
     CP = plt.contour(X, Y, data, cmap=cm.coolwarm, linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')
# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(1, 2, subplot_num * 2 +1,projection='3d')
     plot_e_field(ax, trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'])
     ax = fig.add_subplot(1, 2, subplot_num * 1 + 2)
     plot_e_field_contour(ax, trazar_punto['datos_para_trazar'])
plt.tight_layout()
```

```
plt.subplots_adjust(bottom=0.067, left=0.02, hspace=0, wspace=0.25,
top=0.9676, right=0.921)
plt.show()
```

D. Onda en dos dimensiones con condiciones de frontera PML

D.1. Distribución del campo eléctrico en la componente E_z con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales a) T=10, (b) T=20, (c) T=30, (d) T=40, (e) T=50, (f) T=60, (g) T=70, (h) T=80, (i) T=90, (j) T=100, (k) T=110, (l) T=120

```
#2D con condiciones PML
import numpy as np
from math import sin, pi
from math import exp
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d


#Parametros de la malla
ie = 80
je = 80
#Parametros del pulso
ic = int(ie / 2 )
jc = int(je / 2)

#Discretización de los campos
ez = np.zeros((ie, je))
dz = np.zeros((ie, je))
hx = np.zeros((ie, je))
hy = np.zeros((ie, je))
ihx = np.zeros((ie, je))
ihy = np.zeros((ie, je))
gaz = np.ones((ie, je))
PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
```

```
epsz=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*m0) #Velocidad de la luz


#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 #condición de estabilidad

# Parametros del pulso
t0 = 10
ancho = 6
npasos=130

# Calcular parametros de la capa PML
alphai2= np.ones(ie)
alphai1 = np.ones(ie)
Ai = np.zeros(ie)
bethai2 = np.ones(ie)
bethai1 = np.ones(ie)
alphaj2 = np.ones(ie)
alphaj1 = np.ones(ie)
Aj = np.zeros(ie)
bethaj2 = np.ones(ie)
bethaj1= np.ones(ie)
# Crear los parametros de la capa PML
npml = 8
for n in range(npml):
     xnum = npml - n
     xd = npml
     xxn = xnum / xd
     xn = 0.33 * xxn ** 3

     alphai2[n] = 1 / (1 + xn)
     alphai2[ie - 1 - n] = 1 / (1 + xn)
     alphai1[n] = (1 - xn) / (1 + xn)
     alphai1[ie - 1 - n] = (1 - xn) / (1 + xn)
     alphaj2[n] = 1 / (1 + xn)
     alphaj2[je - 1 - n] = 1 / (1 + xn)
     alphaj1[n] = (1 - xn) / (1 + xn)
     alphaj1[je - 1 - n] = (1 - xn) / (1 + xn)

     xxn = (xnum - 0.5) / xd
```

```
    xn = 0.33 * xxn ** 3
    Ai[n] = xn
    Ai[ie - 2 - n] = xn
    bethai2[n] = 1 / (1 + xn)
    bethai2[ie - 2 - n] = 1 / (1 + xn)
    bethai1[n] = (1 - xn) / (1 + xn)
    bethai1[ie - 2 - n] = (1 - xn) / (1 + xn)
    Aj[n] = xn
    Aj[je - 2 - n] = xn
    bethai2[n] = 1 / (1 + xn)
    bethai2[je - 2 - n] = 1 / (1 + xn)
    bethaj1[n] = (1 - xn) / (1 + xn)
    bethaj1[je - 2 - n] = (1 - xn) / (1 + xn)


# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'numero_pasos': 90, 'datos_para_trazar': None, 'etiqueta': 'i'},
{'numero_pasos': 100, 'datos_para_trazar': None, 'etiqueta': 'j'},
{'numero_pasos': 110, 'datos_para_trazar': None, 'etiqueta': 'k'},
{'numero_pasos': 120, 'datos_para_trazar': None, 'etiqueta': 'l'},

]
# Nucleo principal FDTD bidimensional
for tiempo_paso in range(1, npasos+ 1):
# Calculate Dz
    for j in range(1, je):
        for i in range(1, ie):
            dz[i, j] = alphai1[i] * alphaj1[j] * dz[i, j] +
            alphai2[i] * alphaj2[j] * 0.5 *  (hy[i, j] -
            hy[i - 1, j] - hx[i, j] + hx[i, j - 1])
# Pulso gaussiano
    pulse = exp(-0.5 * ((t0 - tiempo_paso) / ancho) ** 2)
    dz[ic, jc] = pulse
    ez = gaz * dz # Calcular el campo Ez con el campo Dz
# Calcular el campo Hx
    for j in range(je - 1):
        for i in range(ie - 1):
            curl_e = ez[i, j] - ez[i, j + 1]
            ihx[i, j] = ihx[i, j] + curl_e
            hx[i, j] = bethaj1[j] * hx[i, j] +
            bethaj2[j] * (0.5 * curl_e + Ai[i] * ihx[i, j])
# Calcular el campo Hy
```

```
     for j in range(0, je - 1):
         for i in range(0, ie - 1):
             curl_e = ez[i, j] - ez[i + 1, j]
             ihy[i, j] = ihy[i, j] + curl_e
             hy[i, j] = bethai1[i] * hy[i, j] -
             bethai2[i] * (0.5 * curl_e + Aj[j] * ihy[i, j])
# Guarda los datos en ciertos puntos para su posterior trazado
     for trazar_punto in trazar_puntos:
         if tiempo_paso == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(ez)


# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(12, 12))
X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, etiqueta):
     ax.set_zlim(0, 1)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :], rstride=1,
     cstride=1, color='white', edgecolor='blue', linewidth=.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10,fontsize=14)
     ax.set_zticks([0, 0.5, 1])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill= False
     plt.gca().patch.set_facecolor('white')
     ax.text2D(-0.2, 0.8, "({})".format(etiqueta), transform=ax.transAxes)
     ax.dist = 11

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 2, subplot_num +1,projection='3d')
     plot_e_field(ax, trazar_punto['datos_para_trazar'],
     trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
plt.subplots_adjust(bottom=0, left=0.114, hspace=0,
```

```
wspace=0.376, top=1, right=0.729)
plt.show()
```

D.2. Máximos locales del campo eléctrico con condiciones de frontera PML con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, para los pasos temporales (a) T=80, (b) T=110 y (c) T=130, de un pulso sinusoidal con origen en $(40, 40)$.

```
#2D con condiciones PML, pulso sin
import numpy as np
from math import sin, pi
from math import exp
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d



#Parametros de la malla
ie = 80
je = 80
#Parametros del pulso
ic = int(ie / 2 )
jc = int(je / 2)


#Discretización de los campos
ez = np.zeros((ie, je))
dz = np.zeros((ie, je))
hx = np.zeros((ie, je))
hy = np.zeros((ie, je))
ihx = np.zeros((ie, je))
ihy = np.zeros((ie, je))


PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
e0=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(e0*m0) #Velocidad de la luz
```

```
# Create Dielectric Profile
epsz = 8.854e-12

#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 #condición de estabilidad

# Parametros del pulso
npasos = 130



# Calcular parametros de la capa PML
alphai2= np.ones(ie)
alphai1 = np.ones(ie)
Ai = np.zeros(ie)
bethai2 = np.ones(ie)
bethai1 = np.ones(ie)
alphaj2 = np.ones(ie)
alphaj1 = np.ones(ie)
Aj = np.zeros(ie)
bethaj2 = np.ones(ie)
bethaj1= np.ones(ie)
# Crear los parametros de la capa PML
npml = 8
for n in range(npml):
     xnum = npml - n
     xd = npml
     xxn = xnum / xd
     xn = 0.33 * xxn ** 3

     alphai2[n] = 1 / (1 + xn)
     alphai2[ie - 1 - n] = 1 / (1 + xn)
     alphai1[n] = (1 - xn) / (1 + xn)
     alphai1[ie - 1 - n] = (1 - xn) / (1 + xn)
     alphaj2[n] = 1 / (1 + xn)
     alphaj2[je - 1 - n] = 1 / (1 + xn)
     alphaj1[n] = (1 - xn) / (1 + xn)
     alphaj1[je - 1 - n] = (1 - xn) / (1 + xn)

     xxn = (xnum - 0.5) / xd
     xn = 0.33 * xxn ** 3
     Ai[n] = xn
```

```
    Ai[ie - 2 - n] = xn
    bethai2[n] = 1 / (1 + xn)
    bethai2[ie - 2 - n] = 1 / (1 + xn)
    bethai1[n] = (1 - xn) / (1 + xn)
    bethai1[ie - 2 - n] = (1 - xn) / (1 + xn)
    Aj[n] = xn
    Aj[je - 2 - n] = xn
    bethai2[n] = 1 / (1 + xn)
    bethai2[je - 2 - n] = 1 / (1 + xn)
    bethaj1[n] = (1 - xn) / (1 + xn)
    bethaj1[je - 2 - n] = (1 - xn) / (1 + xn)

# Segumiento de los puntos para guardarlos y graficarlos

trazar_puntos= [
{'etiqueta': 'a', 'numero_pasos': 80, 'datos_para_trazar': None},
{'etiqueta': 'b', 'numero_pasos':110, 'datos_para_trazar': None},
{'etiqueta': 'c', 'numero_pasos':130, 'datos_para_trazar': None} ]
# Nucleo principal FDTD bidimensional
for tiempo_paso in range(1, npasos + 1):
# Calculate Dz
    for j in range(1, je):
        for i in range(1, ie):
            dz[i, j] = alphai1[i] * alphaj1[j] * dz[i, j] +
            alphai2[i] * alphaj2[j] * 0.5 *  (hy[i, j] - hy[i - 1, j] -
            hx[i, j] + hx[i, j - 1])
# Pulso gaussiano
    pulse = sin(2 * pi * 1500 * 1e6 * dt * tiempo_paso)
    dz[ic, jc] = pulse
    ez= dz # Calcular el campo Ez con el campo Dz
# Calcular el campo Hx
    for j in range(je - 1):
        for i in range(ie - 1):
            curl_e = ez[i, j] - ez[i, j + 1]
            ihx[i, j] = ihx[i, j] + curl_e
            hx[i, j] = bethaj1[j] * hx[i, j] +
            bethaj2[j] * (0.5 * curl_e + Ai[i] * ihx[i, j])
# Calcular el campo Hy
    for j in range(0, je - 1):
        for i in range(0, ie - 1):
            curl_e = ez[i, j] - ez[i + 1, j]
            ihy[i, j] = ihy[i, j] + curl_e
            hy[i, j] = bethai1[i] * hy[i, j] -
```

```
            bethai2[i] * (0.5 * curl_e + Aj[j] * ihy[i, j])
# Guarda los datos en ciertos puntos para su posterior trazado
     # Guardar los puntos para graficarlos
     for trazar_punto in trazar_puntos:
         if tiempo_paso == trazar_punto['numero_pasos']:
             trazar_punto['datos_para_trazar'] = np.copy(ez)

# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(12, 12))
X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, label):
     ax.set_zlim(0, 1)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :], rstride=1, cstride=1,
     color='white', edgecolor='blue', linewidth=.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10,fontsize=14)
     ax.set_zticks([0, 0.5, 1])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill= False
     plt.gca().patch.set_facecolor('white')
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.dist = 11
     #3, 3, subplot_num + 1
# Se grafica el campo para cada uno de los tiempos guardados
def plot_e_field_contour(ax, data): #grafica de nivel
     CP = plt.contour(X, Y, data, colors='blue', linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(3, 3,subplot_num*3+1, projection='3d')
```

```
    plot_e_field(ax, trazar_punto['datos_para_trazar'],
    trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
    ax = fig.add_subplot(3, 3, subplot_num * 3+2 )
    plot_e_field_contour(ax, trazar_punto['datos_para_trazar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.05, left=0.07,
hspace=0.224, wspace=0.183, top=0.967, right=0.67)
plt.show()
```

D.3. Distribución de los máximos del campo eléctrico en la componente E_z con condiciones de frontera PML con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, después de ciento quince pasos temporales $T = 115$, de un pulso gaussiano con origen en $(40, 40)$.

```
    #Programa 2D con con condiciones de frontera PML, fuente sinosoidal
import numpy as np
from math import sin, pi
from matplotlib import pyplot as plt
import mpl_toolkits.mplot3d.axes3d
from matplotlib import cm
from math import exp
#Paramaetros de la malla
ie = 80
je = 80
ic = int(ie / 2 )
jc = int(je / 2 )



#Discretización de los campos

ez = np.zeros((ie, je))
dz = np.zeros((ie, je))
hx = np.zeros((ie, je))
hy = np.zeros((ie, je))
ihx = np.zeros((ie, je))
ihy = np.zeros((ie, je))
gaz = np.ones((ie, je))
```

```
PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
e0=8.8541e-12 #permitividad del vacio
m0 = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(e0*m0) #Velocidad de la luz

# Parametros del pulso
t0 = 10
ancho = 6
npasos = 300


#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 # condición de estabilidad


# Calcular parametros de la capa PML
alphai2= np.ones(ie)
alphai1 = np.ones(ie)
Ai = np.zeros(ie)
bethai2 = np.ones(ie)
bethai1 = np.ones(ie)
alphaj2 = np.ones(ie)
alphaj1 = np.ones(ie)
Aj = np.zeros(ie)
bethaj2 = np.ones(ie)
bethaj1= np.ones(ie)
# Crear los parametros de la capa PML
npml = 8
for n in range(npml):
     xnum = npml - n
     xd = npml
     xxn = xnum / xd
     xn = 0.33 * xxn ** 3

     alphai2[n] = 1 / (1 + xn)
     alphai2[ie -1- n] = 1 / (1 + xn)
     alphai1[n] = (1 - xn) / (1 + xn)
     alphai1[ie - 1 - n] = (1 - xn) / (1 + xn)
     alphaj2[n] = 1 / (1 + xn)
     alphaj2[je - 1 - n] = 1 / (1 + xn)
     alphaj1[n] = (1 - xn) / (1 + xn)
```

```
      alphaj1[je - 1 - n] = (1 - xn) / (1 + xn)

      xxn = (xnum - 0.5) / xd
      xn = 0.33 * xxn ** 3
      Ai[n] = xn
      Ai[ie - 2 - n] = xn
      bethai2[n] = 1 / (1 + xn)
      bethai2[ie - 2 - n] = 1 / (1 + xn)
      bethai1[n] = (1 - xn) / (1 + xn)
      bethai1[ie - 2 - n] = (1 - xn) / (1 + xn)
      Aj[n] = xn
      Aj[je - 2 - n] = xn
      bethai2[n] = 1 / (1 + xn)
      bethai2[je - 2 - n] = 1 / (1 + xn)
      bethaj1[n] = (1 - xn) / (1 + xn)
      bethaj1[je - 2 - n] = (1 - xn) / (1 + xn)

# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos= [
{'etiqueta': 'a', 'numero_pasos': 115, 'datos_para_trazar': None}]
# Nucleo principal FDTD bidimensional
for tiempo_paso  in range(1, npasos + 1):
# Calculate Dz
      for j in range(1, je):
          for i in range(1, ie):
              dz[i, j] = alphai1[i] * alphaj1[j] * dz[i, j] +
              alphai2[i] * alphaj2[j] * 0.5 *  (hy[i, j] - hy[i - 1, j] -
              hx[i, j] + hx[i, j - 1])
# Pulso gaussiano
      pulse = exp(-0.5 * ((t0 -tiempo_paso) / ancho) ** 2)
      dz[ic, jc] = pulse
      ez =  dz # Calcular el campo Ez con el campo Dz
# Calcular el campo Hx
      for j in range(je - 1):
          for i in range(ie - 1):
              curl_e = ez[i, j] - ez[i, j + 1]
              ihx[i, j] = ihx[i, j] + curl_e
              hx[i, j] = bethaj1[j] * hx[i, j] +
              bethaj2[j] * (0.5 * curl_e + Ai[i] * ihx[i, j])
# Calcular el campo Hy
      for j in range(0, je - 1):
          for i in range(0, ie - 1):
              curl_e = ez[i, j] - ez[i + 1, j]
```

```
            ihy[i, j] = ihy[i, j] + curl_e
            hy[i, j] = bethai1[i] * hy[i, j] -
            bethai2[i] * (0.5 * curl_e + Aj[j] * ihy[i, j])
# Guardar los puntos para graficarlos
    for trazar_punto in trazar_puntos:
        if tiempo_paso == trazar_punto['numero_pasos']:
            trazar_punto['datos_para_trazar'] = np.copy(ez)

# Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))
X, Y = np.meshgrid(range(je), range(ie))
def plot_e_field(ax, data, timestep, label):
    ax.set_zlim(0, 1)
    ax.view_init(elev=20., azim=45)
    b=ax.plot_surface(X, Y, data, rstride=1, cstride=1,
    cmap=cm.coolwarm, linewidth=.25)
    ax.zaxis.set_rotate_label(False)
    ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
    ax.set_xlabel('cm')
    ax.set_ylabel('cm')
    ax.set_xticks(np.arange(0, 81, step=20))
    ax.set_yticks(np.arange(0, 81, step=20))
    ax.set_zticks([0, 0.5, 1])
    ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
    ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
    plt.gca().patch.set_facecolor('white')
    ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
    plt.colorbar(b)
    plt.gca().patch.set_facecolor('white')
    ax.dist = 11
def plot_e_field_contour(ax, data):
    CP = plt.contour(X, Y, data, cmap=cm.coolwarm, linestyles='solid')
    CP.collections[4].remove()
# Elimina la visualización del contorno exterior
    ax.set_xticks(np.arange(0, 81, step=20))
    ax.set_yticks(np.arange(0, 81, step=20))
    plt.xlabel('cm')
    plt.ylabel('cm')
# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
```

```
    ax = fig.add_subplot(1, 2, subplot_num * 2 +1,projection='3d')
    plot_e_field(ax, trazar_punto['datos_para_trazar'],
    trazar_punto['numero_pasos'],trazar_punto['etiqueta'])
    ax = fig.add_subplot(1, 2, subplot_num * 1 + 2)
    plot_e_field_contour(ax, trazar_punto['datos_para_trazar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.067, left=0.02,
hspace=0, wspace=0.25, top=0.9676, right=0.921)
plt.show()
```

E. Onda en tres dimensiones con condiciones de frontera PEC

E.1. Distribución de onda electromagnética en tres dimensiones con condiciones de frontera PEC a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=20, (b) T=40, (c) T=60, (d) T=80, (e) T=100, (f) T=120, (g) T=140, (h) T=160, (i) T=180, (j) T=200, (k) T=220, (l) T=240.

```
#Programa 3D con condiciones PEC
import numpy as np
from math import exp
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
import numba

#Parametros
ie = 80
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
kc = int(ke / 2)
#Discretización de los campos
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
```

```
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))

PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
e0=8.8541e-12 #permitividad del vacio
epsz = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*e0) #Velocidad de la luz
#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 # condición de estabilidad


#  creación de dipolo
gaz[ic, jc, kc - 3: kc + 3] = 0
gaz[ic, jc, kc] = 1

# Parametros del pulso
t0 = 10
propagar = 6
npasos = 240


# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'e','numero_pasos': 100, 'datos_para_graficar':
None, 'z_scale': 0.20},{'etiqueta': 'f','numero_pasos': 120,
'datos_para_graficar': None, 'z_scale': 0.20},
{'etiqueta': 'g','numero_pasos': 140, 'datos_para_graficar':
None, 'z_scale': 0.20}, {'etiqueta': 'h','numero_pasos': 160,
'datos_para_graficar': None, 'z_scale': 0.20}]

# Functions for Main FDTD Loop0.1

def calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz):
# Calcular los campos Dx, Dy, y Dz
```

```
    for i in range(1, ie):
        for j in range(1, je):
            for k in range(1, ke):
                dx[i, j, k] = dx[i, j, k] + 0.5 *
                ( hz[i, j, k] - hz[i, j - 1, k] -
                hy[i, j, k] + hy[i, j, k - 1])

    for i in range(1, ie):
        for j in range(1, je):
            for k in range(1, ke):
                dy[i, j, k] = dy[i, j, k] + 0.5 *
                (hx[i, j, k] - hx[i, j, k - 1] -
                hz[i, j, k] + hz[i - 1, j, k])

    for i in range(1, ie):
        for j in range(1, je):
            for k in range(1, ke):
                dz[i, j, k] = dz[i, j, k] + 0.5 *
                (hy[i, j, k] - hy[i - 1, j, k] -
                hx[i, j, k] + hx[i, j - 1, k])

    return dx, dy, dz

def calcular_campos_e(ie, je, ke, dx, dy, dz,gax, gay, gaz, ex, ey, ez):
#Calcular el campo E con el campo D
    for i in range(0, ie):
        for j in range(0, je):
            for k in range(0, ke):
                ex[i, j, k] = gax[i, j, k] * dx[i, j, k]
                ey[i, j, k] = gay[i, j, k] * dy[i, j, k]
                ez[i, j, k] = gaz[i, j, k] * dz[i, j, k]
    return ex, ey, ez

def calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez):
#Calcular el campo Hx, Hy, y Hz
    for i in range(0, ie):
        for j in range(0, je - 1):
            for k in range(0, ke - 1):
                hx[i, j, k] = hx[i, j, k] + 0.5 *
                (ey[i, j, k + 1] - ey[i, j, k] -
                ez[i, j + 1, k] + ez[i, j, k])

    for i in range(0, ie - 1):
```

```
            for j in range(0, je):
                for k in range(0, ke - 1):
                    hy[i, j, k] = hy[i, j, k] + 0.5 *
                    (ez[i + 1, j, k] - ez[i, j, k] -
                    ex[i, j, k + 1] + ex[i, j, k])
        for i in range(0, ie - 1):
            for j in range(0, je - 1):
                for k in range(0, ke):
                    hz[i, j, k] = hz[i, j, k] + 0.5 *
                    (ex[i, j + 1, k] - ex[i, j, k] -
                    ey[i + 1, j, k] + ey[i, j, k])
        return hx, hy, hz

# Nucleo principal del programa
for tiempo_paso in range(1, npasos + 1):
        # Calcular los campos D
        dx, dy, dz = calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz)
        # La fuente del pulso gaussiano
        pulse = exp(-0.5 * ((t0 - tiempo_paso) / propagar) ** 2)
        dz[ic, jc, kc] = pulse
        # Calcular los campos E apartir de D
        ex, ey, ez = calcular_campos_e(ie, je, ke, dx, dy, dz, gax,
        gay, gaz, ex, ey, ez)
        # Calcular los campos H
        hx, hy, hz = calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez)
        # Guarda los datos en ciertos puntos para su posterior trazado
        for trazar_punto in trazar_puntos:
            if tiempo_paso== trazar_punto['numero_pasos']:
                trazar_punto['datos_para_graficar'] = np.copy(ez)
#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(8, 6))

X, Y = np.meshgrid(range(je), range(ie))
def plot_e_field(ax, data, timestep, scale, label):
        ax.set_zlim(0, scale)
        ax.view_init(elev=20., azim=45)
        ax.plot_surface(X, Y, data[:, :, kc],
        rstride=1, cstride=1, color='white', edgecolor='red', linewidth=0.25)
        ax.zaxis.set_rotate_label(False)
        ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
```

```
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 2, subplot_num + 1, projection='3d')
     plot_e_field(ax, trazar_punto['datos_para_graficar'],
    trazar_punto['numero_pasos'],trazar_punto['z_scale'],
    trazar_punto['etiqueta'])
plt.subplots_adjust(bottom=0, left=0.114,
hspace=0, wspace=0.376, top=1, right=0.729)
plt.show()
```

E.2. Distribución de los máximos del campo eléctrico con condiciones de frontera PEC con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=80, (b) T=100 y (c) T=120, de un pulso sinusoidal con origen en $(40, 40, 0)$.

```
import numpy as np
from math import exp
from matplotlib import pyplot as plt
from math import sin, pi
from mpl_toolkits.mplot3d.axes3d import Axes3D
import numba

#Parametros
ie = 80
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
```

```
kc = int(ke / 2)
#Discretización de los campos
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))

PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
e0=8.8541e-12 #permitividad del vacio
epsz = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*e0) #Velocidad de la luz
#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 # condición de estabilidad


#  dipolo
gaz[ic, jc, kc - 3:kc + 3] = 0
gaz[ic, jc, kc] = 1

# Parametros del pulso
t0 = 10
propagar = 6
npasos = 120


# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'a','numero_pasos': 80, 'datos_para_graficar'
:None, 'z_scale': 0.20},
{'etiqueta': 'b','numero_pasos': 100,'datos_para_graficar':
None, 'z_scale':
0.20},{'etiqueta': 'c','numero_pasos': 120, 'datos_para_graficar':
None, 'z_scale': 0.20}]
```

```
# Functions for Main FDTD Loop0.1

def calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz):
# Calcular los campos Dx, Dy, y Dz
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 dx[i, j, k] = dx[i, j, k] + 0.5 *
                 ( hz[i, j, k] - hz[i, j - 1, k] -
                 hy[i, j, k] + hy[i, j, k - 1])

     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 dy[i, j, k] = dy[i, j, k] + 0.5 *
                 (hx[i, j, k] - hx[i, j, k - 1] -
                 hz[i, j, k] + hz[i - 1, j, k])

     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 dz[i, j, k] = dz[i, j, k] + 0.5 *
                 (hy[i, j, k] - hy[i - 1, j, k] -
                 hx[i, j, k] + hx[i, j - 1, k])

     return dx, dy, dz

def calcular_campos_e(ie, je, ke, dx, dy, dz,gax, gay, gaz, ex, ey, ez):
#Calcular el campo E con el campo D
     for i in range(0, ie):
         for j in range(0, je):
             for k in range(0, ke):
                 ex[i, j, k] = gax[i, j, k] * dx[i, j, k]
                 ey[i, j, k] = gay[i, j, k] * dy[i, j, k]
                 ez[i, j, k] = gaz[i, j, k] * dz[i, j, k]
     return ex, ey, ez

def calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez):
#Calcular el campo Hx, Hy, y Hz
     for i in range(0, ie):
         for j in range(0, je - 1):
```

```
            for k in range(0, ke - 1):
                hx[i, j, k] = hx[i, j, k] + 0.5 *
                (ey[i, j, k + 1] - ey[i, j, k] -
                ez[i, j + 1, k] + ez[i, j, k])

    for i in range(0, ie - 1):
        for j in range(0, je):
            for k in range(0, ke - 1):
                hy[i, j, k] = hy[i, j, k] + 0.5 *
                (ez[i + 1, j, k] - ez[i, j, k] -
                ex[i, j, k + 1] + ex[i, j, k])
    for i in range(0, ie - 1):
        for j in range(0, je - 1):
            for k in range(0, ke):
                hz[i, j, k] = hz[i, j, k] + 0.5 *
                (ex[i, j + 1, k] - ex[i, j, k] -
                ey[i + 1, j, k] + ey[i, j, k])
    return hx, hy, hz

# Nucleo principal del programa
for tiempo_paso in range(1, npasos + 1):
    # Calcular los campos D
    dx, dy, dz = calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz)
    # La fuente del pulso gaussiano
    pulse = exp(-0.5 * ((t0 - tiempo_paso) / propagar) ** 2)
    dz[ic, jc, kc] = pulse
    # Calcular los campos E apartir de D
    ex, ey, ez = calcular_campos_e(ie, je, ke, dx, dy,
    dz, gax, gay, gaz, ex, ey, ez)
    # Calcular los campos H
    hx, hy, hz = calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez)
    # Guarda los datos en ciertos puntos para su posterior trazado
    for trazar_punto in trazar_puntos:
        if tiempo_paso== trazar_punto['numero_pasos']:
            trazar_punto['datos_para_graficar'] = np.copy(ez)



#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))
```

```
X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, scale, label):
     ax.set_zlim(0, scale)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :, kc], rstride=1,
     cstride=1, color='white', edgecolor='red', linewidth=0.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11
def plot_e_field_contour(ax, data): #grafica de nivel
     CP = plt.contour(X, Y, data[:, :, kc], colors='red',
     linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(3, 3,subplot_num*3+1, projection='3d')
     plot_e_field(ax,
     trazar_punto['datos_para_graficar'],trazar_punto['numero_pasos'],
     trazar_punto['z_scale'],trazar_punto['etiqueta'])
     ax = fig.add_subplot(3, 3, subplot_num * 3+2 )
     plot_e_field_contour(ax, trazar_punto['datos_para_graficar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.05, left=0.07,
hspace=0.224, wspace=0.183, top=0.967, right=0.67)
plt.show()
```

E.3. Distribución de los máximos del campo eléctrico en la componente E_z con condiciones de frontera PEC con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, después de ciento cincuenta pasos temporales $T = 150$, de un pulso gaussiano con origen en $(40, 40, 0)$.

```
import numpy as np
from math import exp
from matplotlib import pyplot as plt
from math import sin, pi
from mpl_toolkits.mplot3d.axes3d import Axes3D
import numba
from matplotlib import cm

#Parametros
ie = 80
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
kc = int(ke / 2)
#Discretización de los campos
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))

PPW= 10 #Puntos por longitud de onda
CFL= 0.9 #Condicioon de courant-Friedrisch-Lewy
e0=8.8541e-12 #permitividad del vacio
```

```
epsz = (4e-7)*np.pi #Permeabilidad del vacio
c0=1/np.sqrt(epsz*e0) #Velocidad de la luz
#Tamaño de la celda
ddx = 0.01
dt = CFL * ddx/c0 #condición de estabilidad


#  dipolo
gaz[ic, jc, kc - 3:kc + 3] = 0
gaz[ic, jc, kc] = 3

# Parametros del pulso
t0 = 10
propagar = 6
npasos =150


# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'a','numero_pasos': 150, 'datos_para_graficar':
None, 'z_scale': 0.20}]


# Functions for Main FDTD Loop0.1

def calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz):
# Calcular los campos Dx, Dy, y Dz
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 dx[i, j, k] = dx[i, j, k] + 0.5 *
                 ( hz[i, j, k] - hz[i, j - 1, k] -
                 hy[i, j, k] + hy[i, j, k - 1])

     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 dy[i, j, k] = dy[i, j, k] + 0.5 *
                 (hx[i, j, k] - hx[i, j, k - 1] -
                 hz[i, j, k] + hz[i - 1, j, k])

     for i in range(1, ie):
         for j in range(1, je):
```

```
            for k in range(1, ke):
                dz[i, j, k] = dz[i, j, k] + 0.5 *
                (hy[i, j, k] - hy[i - 1, j, k] -
                hx[i, j, k] + hx[i, j - 1, k])

    return dx, dy, dz

def calcular_campos_e(ie, je, ke, dx, dy, dz,gax, gay, gaz, ex, ey, ez):
#Calcular el campo E con el campo D
    for i in range(0, ie):
        for j in range(0, je):
            for k in range(0, ke):
                ex[i, j, k] = gax[i, j, k] * dx[i, j, k]
                ey[i, j, k] = gay[i, j, k] * dy[i, j, k]
                ez[i, j, k] = gaz[i, j, k] * dz[i, j, k]
    return ex, ey, ez

def calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez):
#Calcular el campo Hx, Hy, y Hz
    for i in range(0, ie):
        for j in range(0, je - 1):
            for k in range(0, ke - 1):
                hx[i, j, k] = hx[i, j, k] + 0.5 *
                (ey[i, j, k + 1] - ey[i, j, k] -
                ez[i, j + 1, k] + ez[i, j, k])

    for i in range(0, ie - 1):
        for j in range(0, je):
            for k in range(0, ke - 1):
                hy[i, j, k] = hy[i, j, k] + 0.5 *
                (ez[i + 1, j, k] - ez[i, j, k] -
                ex[i, j, k + 1] + ex[i, j, k])
    for i in range(0, ie - 1):
        for j in range(0, je - 1):
            for k in range(0, ke):
                hz[i, j, k] = hz[i, j, k] + 0.5 *
                (ex[i, j + 1, k] - ex[i, j, k] -
                ey[i + 1, j, k] + ey[i, j, k])
    return hx, hy, hz

# Nucleo principal del programa
for tiempo_paso in range(1, npasos + 1):
    # Calcular los campos D
```

```
     dx, dy, dz = calcular_campos_d(ie, je, ke, dx, dy, dz, hx, hy, hz)
     # La fuente del pulso gaussiano
     pulse = exp(-0.5 * ((t0 -tiempo_paso) / propagar) ** 2)
     dz[ic, jc, kc] = pulse
     # Calcular los campos E apartir de D
     ex, ey, ez = calcular_campos_e(ie, je, ke, dx, dy,
     dz, gax, gay, gaz, ex, ey, ez)
     # Calcular los campos H
     hx, hy, hz = calcular_campos_h(ie, je, ke, hx, hy, hz, ex, ey, ez)
     # Guarda los datos en ciertos puntos para su posterior trazado
     for trazar_punto in trazar_puntos:
         if tiempo_paso== trazar_punto['numero_pasos']:
             trazar_punto['datos_para_graficar'] = np.copy(ez)

#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))

X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, scale, label):
     ax.set_zlim(0, scale)
     ax.view_init(elev=20., azim=45)
     b=ax.plot_surface(X, Y, data[:, :, kc], rstride=1,
     cstride=1, cmap=cm.coolwarm, linewidth=0.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.set_zticks([0, 0.5, 1])
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     plt.colorbar(b)
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11
def plot_e_field_contour(ax, data): #grafica de nivel
```

```
     CP = plt.contour(X, Y, data[:, :, kc], cmap=cm.coolwarm,
     linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(1, 2,subplot_num*2+1, projection='3d')
     plot_e_field(ax,
     trazar_punto['datos_para_graficar'],trazar_punto['numero_pasos'],
     trazar_punto['z_scale'],trazar_punto['etiqueta'])
     ax = fig.add_subplot(1, 2, subplot_num * 1+2 )
     plot_e_field_contour(ax, trazar_punto['datos_para_graficar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.067, left=0.02, hspace=0,
wspace=0.25, top=0.9676, right=0.921)
plt.show()
```

F. Onda en tres dimensiones con condiciones de frontera PEC

F.1. Distribución de onda electromagnética en tres dimensiones con condiciones de frontera PML a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales (a) T=20, (b) T=40, (c) T=60, (d) T=80, (e) T=100, (f) T=120, (g) T=140, (h) T=160, (i) T=180, (j) T=200, (k) T=220, (l) T=240.

```
from math import exp, sqrt, cos, sin
import numpy as np
from matplotlib import pyplot as plt

#Parametros
ie = 80
```

```
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
kc = int(ke / 2)
#Discretización de los campos
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
ix = np.zeros((ie, je, ke))
iy = np.zeros((ie, je, ke))
iz = np.zeros((ie, je, ke))
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
idx = np.zeros((ie, je, ke))
idy = np.zeros((ie, je, ke))
idz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
ihx = np.zeros((ie, je, ke))
ihy = np.zeros((ie, je, ke))
ihz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))
gbx = np.zeros((ie, je, ke))
gby = np.zeros((ie, je, ke))
gbz = np.zeros((ie, je, ke))


# dipolo
gaz[ic, jc, kc - 3:kc + 3] = 0
gaz[ic, jc, kc] = 1

#Parametros de discretizacion
ddx = 0.01
dt = ddx / 6e8
epsz = 8.854e-12

# Parametros del pulso
t0 = 10
```

```
propagar = 6
npasos = 240

# Claular parametro PML
npml = 8

def calculate_pml_parameters(npml, ie, je, ke):
#Calcular y retornar los parametros PML
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     for n in range(npml):
         xxn = (npml - n) / npml
         xn = 0.33 * (xxn ** 3)
         alphai1[n] = xn
         alphai1[ie - n - 1] = xn
         alphai2[n] = 1 / (1 + xn)
         alphai2[ie - 1 - n] = 1 / (1 + xn)
         alphai3[n] = (1 - xn) / (1 + xn)
         alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
         bethaj1[n] = xn
         bethaj1[je - n - 1] = xn
         bethaj2[n] = 1 / (1 + xn)
         bethaj2[je - 1 - n] = 1 / (1 + xn)
         bethaj3[n] = (1 - xn) / (1 + xn)
         bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
         gammak1[n] = xn
         gammak1[ke - n - 1] = xn
```

```
            gammak2[n] = 1 / (1 + xn)
            gammak2[ke - 1 - n] = 1 / (1 + xn)
            gammak3[n] = (1 - xn) / (1 + xn)
            gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)
            xxn = (npml - n - 0.5) / npml
            xn = 0.33 * (xxn ** 3)
            alphai1[n] = xn
            alphai1[ie - 1 - n] = xn
            alphai2[n] = 1 / (1 + xn)
            alphai2[ie - 1 - n] = 1 / (1 + xn)
            alphai3[n] = (1 - xn) / (1 + xn)
            alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
            bethaj1[n] = xn
            bethaj1[je - 1 - n] = xn
            bethaj2[n] = 1 / (1 + xn)
            bethaj2[je - 1 - n] = 1 / (1 + xn)
            bethaj3[n] = (1 - xn) / (1 + xn)
            bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
            gammak1[n] = xn
            gammak1[ke - 1 - n] = xn
            gammak2[n] = 1 / (1 + xn)
            gammak2[ke - 1 - n] = 1 / (1 + xn)
            gammak3[n] = (1 - xn) / (1 + xn)
            gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)

        return alphai1, alphai2, alphai3, alphai1, alphai2,
        alphai3, bethaj1, bethaj2, bethaj3, bethaj1, bethaj2,
        bethaj3, gammak1, gammak2, gammak3,gammak1, gammak2, gammak3

# calcular los parametros PML
npml = 8
alphai1, alphai2, alphai3, alphai1, alphai2, alphai3, bethaj1,
bethaj2, bethaj3, bethaj1, bethaj2, bethaj3, gammak1, gammak2,
gammak3,gammak1, gammak2, gammak3 = calculate_pml_parameters(npml, ie, je, ke)

# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'i','numero_pasos': 180, 'datos_para_graficar':
None, 'z_scale': 0.20},
{'etiqueta': 'j','numero_pasos': 200, 'datos_para_graficar':
None, 'z_scale': 0.20},
{'etiqueta': 'k','numero_pasos': 220, 'datos_para_graficar':
None, 'z_scale': 0.20},
```

```
{'etiqueta': 'l','numero_pasos': 240, 'datos_para_graficar':
None, 'z_scale': 0.20}]
def calculate_dx_field(ie, je, ke, dx, idx, hy, hz,bethaj3,
gammak3, bethaj2, gammak2, alphai1):
# Calcular el campo Dx
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hz[i, j, k] - hz[i, j - 1, k] -
                 hy[i, j, k] + hy[i, j, k - 1])
                 idx[i, j, k] = idx[i, j, k] + curl_h
                 dx[i, j, k] = bethaj3[j] * gammak3[k] *
                 dx[i, j, k] + bethaj2[j] * gammak2[k] *
                 (0.5 * curl_h + alphai1[i] * idx[i, j, k])

     return dx, idx
def calculate_dy_field(ie, je, ke, dy, idy, hx, hz,
alphai3, gammak3, alphai2, gammak2, bethaj1):
# Calcular el campo Dy
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hx[i, j, k] - hx[i, j, k - 1] -
                 hz[i, j, k] + hz[i - 1, j, k])
                 idy[i, j, k] = idy[i, j, k] + curl_h
                 dy[i, j, k] = alphai3[i] * gammak3[k] *
                 dy[i, j, k] + alphai2[i] * gammak2[k] *
                 (0.5 * curl_h + bethaj1[j] * idy[i, j, k])
     return dy, idy

def calculate_dz_field(ie, je, ke, dz, idz, hx, hy,alphai3, bethaj3,
alphai2, bethaj2, gammak1):
#Calcular el campo Dz
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hy[i, j, k] - hy[i - 1, j, k] -hx[i, j, k] +
                 hx[i, j - 1, k])
                 idz[i, j, k] = idz[i, j, k] + curl_h
                 dz[i, j, k] = alphai3[i] * bethaj3[j] *
                 dz[i, j, k] + alphai2[i] * bethaj2[j] *
                 (0.5 * curl_h + gammak1[k] * idz[i, j, k])
     return dz, idz
```

```
def calculate_e_fields(ie, je, ke, dx, dy, dz, gax, gay,
gaz,gbx, gby, gbz, ex, ey, ez, ix, iy, iz):
#Calcular el campo E a partir de D
     for i in range(0, ie):
         for j in range(0, je):
             for k in range(0, ke):
                 ex[i, j, k] = gax[i, j, k] * (dx[i, j, k] - ix[i, j, k])
                 ix[i, j, k] = ix[i, j, k] + gbx[i, j, k] * ex[i, j, k]
                 ey[i, j, k] = gay[i, j, k] * (dy[i, j, k] - iy[i, j, k])
                 iy[i, j, k] = iy[i, j, k] + gby[i, j, k] * ey[i, j, k]
                 ez[i, j, k] = gaz[i, j, k] * (dz[i, j, k] - iz[i, j, k])
                 iz[i, j, k] = iz[i, j, k] + gbz[i, j, k] * ez[i, j, k]
     return ex, ey, ez, ix, iy, iz

def calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1, bethaj2,
gammak2, bethaj3, gammak3):
#Calcular el campo Hx
     for i in range(0, ie):
         for j in range(0, je - 1):
             for k in range(0, ke - 1):
                 curl_e = (ey[i, j, k + 1] - ey[i, j, k] -ez[i, j + 1, k] +
                 ez[i, j, k])
                 ihx[i, j, k] = ihx[i, j, k] + curl_e
                 hx[i, j, k] = bethaj3[j] * gammak3[k] *
                 hx[i, j, k] + bethaj2[j] * gammak2[k] *
                 0.5 * (curl_e + alphai1[i] * ihx[i, j, k])
     return hx, ihx

def calculate_hy_field(ie, je, ke, hy, ihy, ex, ez, bethaj1, alphai2,
gammak2, alphai3, gammak3):
# Calcular el campo Hy
     for i in range(0, ie - 1):
         for j in range(0, je):
             for k in range(0, ke - 1):
                 curl_e = (ez[i + 1, j, k] - ez[i, j, k] -ex[i, j, k + 1] +
                 ex[i, j, k])
                 ihy[i, j, k] = ihy[i, j, k] + curl_e
                 hy[i, j, k] = alphai3[i] * gammak3[k] *
                 hy[i, j, k] + alphai2[i] * gammak2[k] *
                 0.5 * (curl_e + bethaj1[j] * ihy[i, j, k])
     return hy, ihy
```

```
def calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1,
alphai2, bethaj2, alphai3, bethaj3):
# Calcular el campo Hz
     for i in range(0, ie - 1):
         for j in range(0, je - 1):
             for k in range(0, ke):
                 curl_e = (ex[i, j + 1, k] - ex[i, j, k] -
                 ey[i + 1, j, k] + ey[i, j, k])
                 ihz[i, j, k] = ihz[i, j, k] + curl_e
                 hz[i, j, k] = alphai3[i] * bethaj3[j] *
                 hz[i, j, k] + alphai2[i] * bethaj2[j] *
                 0.5 * (curl_e + gammak1[k] * ihz[i, j, k])
     return hz, ihz


for tiempo_paso in range(1, npasos + 1):

    # Calcular el campo D
     dx, idx = calculate_dx_field(ie, je, ke, dx, idx, hy, hz,
     bethaj3, gammak3, bethaj2, gammak2, alphai1)
     dy, idy = calculate_dy_field(ie, je, ke, dy, idy, hx, hz,
     alphai3, gammak3, alphai2, gammak2, bethaj1)
     dz, idz = calculate_dz_field(ie, je, ke, dz, idz, hx, hy,
     alphai3, bethaj3, alphai2, bethaj2, gammak1)
# Parametros del pulso
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / propagar) ** 2)
     dz[ic, jc, kc] = pulse
# Calcular el campo E con el campo D
     ex, ey, ez, ix, iy, iz = calculate_e_fields(ie, je, ke, dx, dy,
     dz,gax, gay, gaz,gbx, gby, gbz,ex, ey, ez, ix, iy, iz)

     # Calcular el campo H
     hx, ihx = calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1,
     bethaj2, gammak2, bethaj3, gammak3)
     hy, ihy = calculate_hy_field(ie, je, ke, hy, ihy, ex, ez,bethaj1,
     alphai2, gammak2, alphai3, gammak3)
     hz, ihz = calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1,
     alphai2, bethaj2, alphai3, bethaj3)
      # Guarda los datos en ciertos puntos para su posterior trazado
     for trazar_punto in trazar_puntos:
         if tiempo_paso== trazar_punto['numero_pasos']:
             trazar_punto['datos_para_graficar'] = np.copy(ez)
```

```
#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(8, 6))

X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, scale, label):
     ax.set_zlim(0, scale)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :, kc], rstride=1, cstride=1,
     color='white', edgecolor='blue', linewidth=0.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(2, 2, subplot_num + 1, projection='3d')
     plot_e_field(ax,
     trazar_punto['datos_para_graficar'],trazar_punto['numero_pasos'],
     trazar_punto['z_scale'],trazar_punto['etiqueta'])
plt.subplots_adjust(bottom=0, left=0.114, hspace=0, wspace=0.376,
top=1, right=0.729)
plt.show()
```

F.2. Distribución de onda electromagnética en tres dimensiones con condiciones de frontera PML con mapa de contorno, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, para los pasos temporales ((a) T=80), ((b) T=100) y ((c) T=120), de un pulso gaussiano con origen en $(40, 40, 0)$.

```
from math import exp, sqrt, cos, sin
import numba
import numpy as np
from matplotlib import pyplot as plt

#Parametros
ie = 80
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
kc = int(ke / 2)
#Discretización de los campos
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
ix = np.zeros((ie, je, ke))
iy = np.zeros((ie, je, ke))
iz = np.zeros((ie, je, ke))
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
idx = np.zeros((ie, je, ke))
idy = np.zeros((ie, je, ke))
idz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
ihx = np.zeros((ie, je, ke))
ihy = np.zeros((ie, je, ke))
ihz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))
gbx = np.zeros((ie, je, ke))
```

```
gby = np.zeros((ie, je, ke))
gbz = np.zeros((ie, je, ke))


# dipolo
gaz[ic, jc, kc - 3:kc + 3] = 0
gaz[ic, jc, kc] = 1

#Parametros de discretizacion
ddx = 0.01
dt = ddx / 6e8
epsz = 8.854e-12

# Parametros del pulso
t0 = 10
propagar = 6
npasos = 120

# Claular parametro PML
npml = 8

def calculate_pml_parameters(npml, ie, je, ke):
#Calcular y retornar los parametros PML
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     for n in range(npml):
         xxn = (npml - n) / npml
```

```
    xn = 0.33 * (xxn ** 3)
    alphai1[n] = xn
    alphai1[ie - n - 1] = xn
    alphai2[n] = 1 / (1 + xn)
    alphai2[ie - 1 - n] = 1 / (1 + xn)
    alphai3[n] = (1 - xn) / (1 + xn)
    alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
    bethaj1[n] = xn
    bethaj1[je - n - 1] = xn
    bethaj2[n] = 1 / (1 + xn)
    bethaj2[je - 1 - n] = 1 / (1 + xn)
    bethaj3[n] = (1 - xn) / (1 + xn)
    bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
    gammak1[n] = xn
    gammak1[ke - n - 1] = xn
    gammak2[n] = 1 / (1 + xn)
    gammak2[ke - 1 - n] = 1 / (1 + xn)
    gammak3[n] = (1 - xn) / (1 + xn)
    gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)
    xxn = (npml - n - 0.5) / npml
    xn = 0.33 * (xxn ** 3)
    alphai1[n] = xn
    alphai1[ie - 1 - n] = xn
    alphai2[n] = 1 / (1 + xn)
    alphai2[ie - 1 - n] = 1 / (1 + xn)
    alphai3[n] = (1 - xn) / (1 + xn)
    alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
    bethaj1[n] = xn
    bethaj1[je - 1 - n] = xn
    bethaj2[n] = 1 / (1 + xn)
    bethaj2[je - 1 - n] = 1 / (1 + xn)
    bethaj3[n] = (1 - xn) / (1 + xn)
    bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
    gammak1[n] = xn
    gammak1[ke - 1 - n] = xn
    gammak2[n] = 1 / (1 + xn)
    gammak2[ke - 1 - n] = 1 / (1 + xn)
    gammak3[n] = (1 - xn) / (1 + xn)
    gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)

return alphai1, alphai2, alphai3, alphai1, alphai2, alphai3,
bethaj1, bethaj2, bethaj3, bethaj1, bethaj2, bethaj3, gammak1,
gammak2, gammak3,gammak1, gammak2, gammak3
```

```
# calcular los parametros PML
npml = 8
alphai1, alphai2, alphai3, alphai1, alphai2, alphai3, bethaj1,
bethaj2, bethaj3, bethaj1, bethaj2, bethaj3, gammak1, gammak2,
gammak3,gammak1, gammak2, gammak3 = calculate_pml_parameters(npml, ie, je, ke)

# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'a','numero_pasos': 80, 'datos_para_graficar': None,
'z_scale': 0.20},{'etiqueta': 'b','numero_pasos': 100, 'datos_para_graficar':
None, 'z_scale': 0.20},
{'etiqueta': 'c','numero_pasos': 120, 'datos_para_graficar':
None, 'z_scale': 0.20}]

def calculate_dx_field(ie, je, ke, dx, idx, hy, hz,bethaj3, gammak3,
bethaj2, gammak2, alphai1):
# Calcular el campo Dx
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hz[i, j, k] - hz[i, j - 1, k] -
                 hy[i, j, k] + hy[i, j, k - 1])
                 idx[i, j, k] = idx[i, j, k] + curl_h
                 dx[i, j, k] = bethaj3[j] * gammak3[k] *
                 dx[i, j, k] + bethaj2[j] * gammak2[k] *
                 (0.5 * curl_h + alphai1[i] * idx[i, j, k])

     return dx, idx

def calculate_dy_field(ie, je, ke, dy, idy, hx, hz, alphai3, gammak3,
alphai2, gammak2, bethaj1):
# Calcular el campo Dy
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hx[i, j, k] - hx[i, j, k - 1] -hz[i, j, k] +
                 hz[i - 1, j, k])
                 idy[i, j, k] = idy[i, j, k] + curl_h
                 dy[i, j, k] = alphai3[i] * gammak3[k] *
                 dy[i, j, k] + alphai2[i] * gammak2[k] *
                 (0.5 * curl_h + bethaj1[j] * idy[i, j, k])
     return dy, idy
```

```
def calculate_dz_field(ie, je, ke, dz, idz, hx, hy,alphai3,
bethaj3, alphai2, bethaj2, gammak1):
#Calcular el campo Dz
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hy[i, j, k] - hy[i - 1, j, k] -hx[i, j, k]
                 + hx[i, j - 1, k])
                 idz[i, j, k] = idz[i, j, k] + curl_h
                 dz[i, j, k] = alphai3[i] * bethaj3[j] *
                 dz[i, j, k] + alphai2[i] * bethaj2[j] *
                 (0.5 * curl_h + gammak1[k] * idz[i, j, k])
     return dz, idz


def calculate_e_fields(ie, je, ke, dx, dy, dz, gax, gay, gaz,gbx,
gby, gbz, ex, ey, ez, ix, iy, iz):
#Calcular el campo E a partir de D
     for i in range(0, ie):
         for j in range(0, je):
             for k in range(0, ke):
                 ex[i, j, k] = gax[i, j, k] * (dx[i, j, k] - ix[i, j, k])
                 ix[i, j, k] = ix[i, j, k] + gbx[i, j, k] * ex[i, j, k]
                 ey[i, j, k] = gay[i, j, k] * (dy[i, j, k] - iy[i, j, k])
                 iy[i, j, k] = iy[i, j, k] + gby[i, j, k] * ey[i, j, k]
                 ez[i, j, k] = gaz[i, j, k] * (dz[i, j, k] - iz[i, j, k])
                 iz[i, j, k] = iz[i, j, k] + gbz[i, j, k] * ez[i, j, k]
     return ex, ey, ez, ix, iy, iz

def calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1, bethaj2,
gammak2, bethaj3, gammak3):
#Calcular el campo Hx
     for i in range(0, ie):
         for j in range(0, je - 1):
             for k in range(0, ke - 1):
                 curl_e = (ey[i, j, k + 1] - ey[i, j, k] -ez[i, j + 1, k] +
                 ez[i, j, k])
                 ihx[i, j, k] = ihx[i, j, k] + curl_e
                 hx[i, j, k] = bethaj3[j] * gammak3[k] *
                 hx[i, j, k] + bethaj2[j] * gammak2[k] *
                 0.5 * (curl_e + alphai1[i] * ihx[i, j, k])
     return hx, ihx
```

```
def calculate_hy_field(ie, je, ke, hy, ihy, ex, ez, bethaj1, alphai2,
gammak2, alphai3, gammak3):
# Calcular el campo Hy
     for i in range(0, ie - 1):
         for j in range(0, je):
             for k in range(0, ke - 1):
                 curl_e = (ez[i + 1, j, k] - ez[i, j, k] -
                 ex[i, j, k + 1] + ex[i, j, k])
                 ihy[i, j, k] = ihy[i, j, k] + curl_e
                 hy[i, j, k] = alphai3[i] * gammak3[k] *
                 hy[i, j, k] + alphai2[i] * gammak2[k] *
                 0.5 * (curl_e + bethaj1[j] * ihy[i, j, k])
     return hy, ihy

def calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1, alphai2,
bethaj2, alphai3, bethaj3):
# Calcular el campo Hz
     for i in range(0, ie - 1):
         for j in range(0, je - 1):
             for k in range(0, ke):
                 curl_e = (ex[i, j + 1, k] - ex[i, j, k] -
                 ey[i + 1, j, k] + ey[i, j, k])
                 ihz[i, j, k] = ihz[i, j, k] + curl_e
                 hz[i, j, k] = alphai3[i] * bethaj3[j] *
                 hz[i, j, k] + alphai2[i] * bethaj2[j] *
                 0.5 * (curl_e + gammak1[k] * ihz[i, j, k])
     return hz, ihz


for tiempo_paso in range(1, npasos + 1):

    # Calcular el campo D
     dx, idx = calculate_dx_field(ie, je, ke, dx, idx, hy, hz, bethaj3,
     gammak3, bethaj2, gammak2, alphai1)
     dy, idy = calculate_dy_field(ie, je, ke, dy, idy, hx, hz, alphai3,
     gammak3, alphai2, gammak2, bethaj1)
     dz, idz = calculate_dz_field(ie, je, ke, dz, idz, hx, hy, alphai3,
     bethaj3, alphai2, bethaj2, gammak1)
# Parametros del pulso
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / propagar) ** 2)
     dz[ic, jc, kc] = pulse
# Calcular el campo E con el campo D
```

```
     ex, ey, ez, ix, iy, iz = calculate_e_fields(ie, je, ke, dx,
     dy, dz,gax, gay, gaz,gbx, gby, gbz,ex, ey, ez, ix, iy, iz)

     # Calcular el campo H
     hx, ihx = calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1,
     bethaj2, gammak2, bethaj3, gammak3)
     hy, ihy = calculate_hy_field(ie, je, ke, hy, ihy, ex, ez,bethaj1,
     alphai2, gammak2, alphai3, gammak3)
     hz, ihz = calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1,
     alphai2, bethaj2, alphai3, bethaj3)
      # Guarda los datos en ciertos puntos para su posterior trazado
     for trazar_punto in trazar_puntos:
         if tiempo_paso== trazar_punto['numero_pasos']:
             trazar_punto['datos_para_graficar'] = np.copy(ez)
#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))

X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, scale, label):
     ax.set_zlim(0, scale)
     ax.view_init(elev=20., azim=45)
     ax.plot_surface(X, Y, data[:, :, kc], rstride=1, cstride=1,
     color='white', edgecolor='blue', linewidth=0.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11
def plot_e_field_contour(ax, data): #grafica de nivel
     CP = plt.contour(X, Y, data[:, :, kc], colors='blue', linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
```

```
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(3, 3,subplot_num*3+1, projection='3d')
     plot_e_field(ax,
     trazar_punto['datos_para_graficar'],
     trazar_punto['numero_pasos'],
     trazar_punto['z_scale'],trazar_punto['etiqueta'])
     ax = fig.add_subplot(3, 3, subplot_num * 3+2 )
     plot_e_field_contour(ax, trazar_punto['datos_para_graficar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.05, left=0.07, hspace=0.224,
wspace=0.183, top=0.967, right=0.67)
plt.show()
```

F.3. Distribución de onda electromagnética en tres dimensiones con condiciones de frontera PML con mapa de contorno y barra de altura, a lo largo de ochenta celdas del espacio computacional $(0 < x < 80)$, $(0 < y < 80)$, $(0 < z < 80)$, después de ciento cincuenta pasos temporales (T=150), de un pulso gaussiano con origen en $(40, 40, 0)$.

```
from math import exp, sqrt, cos, sin
import numba
from mpl_toolkits.mplot3d.axes3d import Axes3D
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import cm

#Parametros
ie = 80
je = 80
ke = 80
ic = int(ie / 2)
jc = int(je / 2)
kc = int(ke / 2)
#Discretización de los campos
```

```
ex = np.zeros((ie, je, ke))
ey = np.zeros((ie, je, ke))
ez = np.zeros((ie, je, ke))
ix = np.zeros((ie, je, ke))
iy = np.zeros((ie, je, ke))
iz = np.zeros((ie, je, ke))
dx = np.zeros((ie, je, ke))
dy = np.zeros((ie, je, ke))
dz = np.zeros((ie, je, ke))
idx = np.zeros((ie, je, ke))
idy = np.zeros((ie, je, ke))
idz = np.zeros((ie, je, ke))
hx = np.zeros((ie, je, ke))
hy = np.zeros((ie, je, ke))
hz = np.zeros((ie, je, ke))
ihx = np.zeros((ie, je, ke))
ihy = np.zeros((ie, je, ke))
ihz = np.zeros((ie, je, ke))
gax = np.ones((ie, je, ke))
gay = np.ones((ie, je, ke))
gaz = np.ones((ie, je, ke))
gbx = np.zeros((ie, je, ke))
gby = np.zeros((ie, je, ke))
gbz = np.zeros((ie, je, ke))


# dipolo
gaz[ic, jc, kc - 3:kc + 3] = 0
gaz[ic, jc, kc] = 3

#Parametros de discretizacion
ddx = 0.01
dt = ddx / 6e8
epsz = 8.854e-12

# Parametros del pulso
t0 = 10
propagar = 6
npasos = 150

# Claular parametro PML
npml = 8
```

```
def calculate_pml_parameters(npml, ie, je, ke):
#Calcular y retornar los parametros PML
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     alphai1 = np.zeros(ie)
     alphai2 = np.ones(ie)
     alphai3 = np.ones(ie)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     bethaj1 = np.zeros(je)
     bethaj2 = np.ones(je)
     bethaj3 = np.ones(je)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     gammak1 = np.zeros(ke)
     gammak2 = np.ones(ke)
     gammak3 = np.ones(ke)
     for n in range(npml):
         xxn = (npml - n) / npml
         xn = 0.33 * (xxn ** 3)
         alphai1[n] = xn
         alphai1[ie - n - 1] = xn
         alphai2[n] = 1 / (1 + xn)
         alphai2[ie - 1 - n] = 1 / (1 + xn)
         alphai3[n] = (1 - xn) / (1 + xn)
         alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
         bethaj1[n] = xn
         bethaj1[je - n - 1] = xn
         bethaj2[n] = 1 / (1 + xn)
         bethaj2[je - 1 - n] = 1 / (1 + xn)
         bethaj3[n] = (1 - xn) / (1 + xn)
         bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
         gammak1[n] = xn
         gammak1[ke - n - 1] = xn
         gammak2[n] = 1 / (1 + xn)
         gammak2[ke - 1 - n] = 1 / (1 + xn)
         gammak3[n] = (1 - xn) / (1 + xn)
         gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)
         xxn = (npml - n - 0.5) / npml
         xn = 0.33 * (xxn ** 3)
```

```
            alphai1[n] = xn
            alphai1[ie - 1 - n] = xn
            alphai2[n] = 1 / (1 + xn)
            alphai2[ie - 1 - n] = 1 / (1 + xn)
            alphai3[n] = (1 - xn) / (1 + xn)
            alphai3[ie - 1 - n] = (1 - xn) / (1 + xn)
            bethaj1[n] = xn
            bethaj1[je - 1 - n] = xn
            bethaj2[n] = 1 / (1 + xn)
            bethaj2[je - 1 - n] = 1 / (1 + xn)
            bethaj3[n] = (1 - xn) / (1 + xn)
            bethaj3[je - 1 - n] = (1 - xn) / (1 + xn)
            gammak1[n] = xn
            gammak1[ke - 1 - n] = xn
            gammak2[n] = 1 / (1 + xn)
            gammak2[ke - 1 - n] = 1 / (1 + xn)
            gammak3[n] = (1 - xn) / (1 + xn)
            gammak3[ke - 1 - n] = (1 - xn) / (1 + xn)

        return alphai1, alphai2, alphai3, alphai1, alphai2, alphai3,
        bethaj1, bethaj2, bethaj3, bethaj1, bethaj2, bethaj3, gammak1,
        gammak2, gammak3,gammak1, gammak2, gammak3

# calcular los parametros PML
npml = 8
alphai1, alphai2, alphai3, alphai1, alphai2, alphai3, bethaj1,
bethaj2, bethaj3, bethaj1, bethaj2, bethaj3, gammak1, gammak2,
gammak3,gammak1, gammak2,
gammak3 = calculate_pml_parameters(npml, ie, je, ke)

# Segumiento de los puntos para guardarlos y graficarlos
trazar_puntos = [
{'etiqueta': 'a','numero_pasos': 150, 'datos_para_graficar':
None, 'z_scale': 0.20}]


def calculate_dx_field(ie, je, ke, dx, idx, hy, hz,bethaj3, gammak3,
bethaj2, gammak2, alphai1):
# Calcular el campo Dx
    for i in range(1, ie):
        for j in range(1, je):
            for k in range(1, ke):
                curl_h = (hz[i, j, k] - hz[i, j - 1, k] -hy[i, j, k] +
```

```
                    hy[i, j, k - 1])
                    idx[i, j, k] = idx[i, j, k] + curl_h
                    dx[i, j, k] = bethaj3[j] * gammak3[k] *
                    dx[i, j, k] + bethaj2[j] * gammak2[k] *
                    (0.5 * curl_h + alphai1[i] * idx[i, j, k])

     return dx, idx

def calculate_dy_field(ie, je, ke, dy, idy, hx, hz, alphai3,
gammak3, alphai2, gammak2, bethaj1):
# Calcular el campo Dy
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hx[i, j, k] - hx[i, j, k - 1] -hz[i, j, k] +
                 hz[i - 1, j, k])
                 idy[i, j, k] = idy[i, j, k] + curl_h
                 dy[i, j, k] = alphai3[i] * gammak3[k] *
                 dy[i, j, k] + alphai2[i] * gammak2[k] *
                 (0.5 * curl_h + bethaj1[j] * idy[i, j, k])
     return dy, idy

def calculate_dz_field(ie, je, ke, dz, idz, hx, hy,alphai3,
bethaj3, alphai2, bethaj2, gammak1):
#Calcular el campo Dz
     for i in range(1, ie):
         for j in range(1, je):
             for k in range(1, ke):
                 curl_h = (hy[i, j, k] - hy[i - 1, j, k] -hx[i, j, k] +
                 hx[i, j - 1, k])
                 idz[i, j, k] = idz[i, j, k] + curl_h
                 dz[i, j, k] = alphai3[i] * bethaj3[j] *
                 dz[i, j, k] + alphai2[i] * bethaj2[j] *
                 (0.5 * curl_h + gammak1[k] * idz[i, j, k])
     return dz, idz


def calculate_e_fields(ie, je, ke, dx, dy, dz, gax, gay, gaz,gbx,
gby, gbz, ex, ey, ez, ix, iy, iz):
#Calcular el campo E a partir de D
     for i in range(0, ie):
         for j in range(0, je):
             for k in range(0, ke):
```

```
                ex[i, j, k] = gax[i, j, k] * (dx[i, j, k] - ix[i, j, k])
                ix[i, j, k] = ix[i, j, k] + gbx[i, j, k] * ex[i, j, k]
                ey[i, j, k] = gay[i, j, k] * (dy[i, j, k] - iy[i, j, k])
                iy[i, j, k] = iy[i, j, k] + gby[i, j, k] * ey[i, j, k]
                ez[i, j, k] = gaz[i, j, k] * (dz[i, j, k] - iz[i, j, k])
                iz[i, j, k] = iz[i, j, k] + gbz[i, j, k] * ez[i, j, k]
    return ex, ey, ez, ix, iy, iz

def calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1,
bethaj2, gammak2, bethaj3, gammak3):
#Calcular el campo Hx
    for i in range(0, ie):
        for j in range(0, je - 1):
            for k in range(0, ke - 1):
                curl_e = (ey[i, j, k + 1] - ey[i, j, k] -ez[i, j + 1, k] +
                ez[i, j, k])
                ihx[i, j, k] = ihx[i, j, k] + curl_e
                hx[i, j, k] = bethaj3[j] * gammak3[k] *
                hx[i, j, k] + bethaj2[j] * gammak2[k] *
                0.5 * (curl_e + alphai1[i] * ihx[i, j, k])
    return hx, ihx

def calculate_hy_field(ie, je, ke, hy, ihy, ex, ez, bethaj1,
alphai2, gammak2, alphai3, gammak3):
# Calcular el campo Hy
    for i in range(0, ie - 1):
        for j in range(0, je):
            for k in range(0, ke - 1):
                curl_e = (ez[i + 1, j, k] - ez[i, j, k] -ex[i, j, k + 1] +
                ex[i, j, k])
                ihy[i, j, k] = ihy[i, j, k] + curl_e
                hy[i, j, k] = alphai3[i] * gammak3[k] *
                hy[i, j, k] + alphai2[i] * gammak2[k] *
                0.5 * (curl_e + bethaj1[j] * ihy[i, j, k])
    return hy, ihy

def calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1,
alphai2, bethaj2, alphai3, bethaj3):
# Calcular el campo Hz
    for i in range(0, ie - 1):
        for j in range(0, je - 1):
            for k in range(0, ke):
                curl_e = (ex[i, j + 1, k] - ex[i, j, k] -ey[i + 1, j, k] +
```

```
                    ey[i, j, k])
                    ihz[i, j, k] = ihz[i, j, k] + curl_e
                    hz[i, j, k] = alphai3[i] * bethaj3[j] *
                    hz[i, j, k] + alphai2[i] * bethaj2[j] *
                    0.5 * (curl_e + gammak1[k] * ihz[i, j, k])
     return hz, ihz

for tiempo_paso in range(1, npasos + 1):

    # Calcular el campo D
     dx, idx = calculate_dx_field(ie, je, ke, dx, idx, hy, hz, bethaj3,
     gammak3, bethaj2, gammak2, alphai1)
     dy, idy = calculate_dy_field(ie, je, ke, dy, idy, hx, hz, alphai3,
     gammak3, alphai2, gammak2, bethaj1)
     dz, idz = calculate_dz_field(ie, je, ke, dz, idz, hx, hy, alphai3,
     bethaj3, alphai2, bethaj2, gammak1)
# Parametros del pulso
     pulse = exp(-0.5 * ((t0 - tiempo_paso) / propagar) ** 2)
     dz[ic, jc, kc] = pulse
# Calcular el campo E con el campo D
     ex, ey, ez, ix, iy, iz = calculate_e_fields(ie, je, ke, dx, dy, dz,gax, gay, gaz,

     # Calcular el campo H
     hx, ihx = calculate_hx_field(ie, je, ke, hx, ihx, ey, ez,alphai1,
     bethaj2, gammak2, bethaj3, gammak3)
     hy, ihy = calculate_hy_field(ie, je, ke, hy, ihy, ex, ez,bethaj1,
     alphai2, gammak2, alphai3, gammak3)
     hz, ihz = calculate_hz_field(ie, je, ke, hz, ihz, ex, ey,gammak1,
     alphai2, bethaj2, alphai3, bethaj3)
      # Guarda los datos en ciertos puntos para su posterior trazado
     for trazar_punto in trazar_puntos:
         if tiempo_paso== trazar_punto['numero_pasos']:
             trazar_punto['datos_para_graficar'] = np.copy(ez)
#Plot
plt.rcParams['font.size'] = 12
plt.rcParams['grid.color'] = 'gray'
plt.rcParams['grid.linestyle'] = 'dotted'
fig = plt.figure(figsize=(10, 10))

X, Y = np.meshgrid(range(je), range(ie))

def plot_e_field(ax, data, timestep, scale, label):
     ax.set_zlim(0, scale)
```

```
     ax.view_init(elev=20., azim=45)
     b=ax.plot_surface(X, Y, data[:, :, kc], rstride=1, cstride=1,
     cmap=cm.coolwarm, linewidth=0.25)
     ax.zaxis.set_rotate_label(False)
     ax.set_zlabel(r' $E_{Z}$', rotation=90, labelpad=10, fontsize=14)
     ax.set_zticks([0, scale / 2, scale])
     ax.set_xlabel('cm')
     ax.set_ylabel('cm')
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     ax.set_zticks([0, 0.5, 1])
     ax.text2D(0.6, 0.7, "T = {}".format(timestep), transform=ax.transAxes)
     ax.text2D(-0.2, 0.8, "({})".format(label), transform=ax.transAxes)
     ax.xaxis.pane.fill = ax.yaxis.pane.fill = ax.zaxis.pane.fill = False
     plt.gca().patch.set_facecolor('white')
     plt.colorbar(b)
     plt.gca().patch.set_facecolor('white')
     ax.dist = 11
def plot_e_field_contour(ax, data): #grafica de nivel
     CP = plt.contour(X, Y, data[:, :, kc], cmap=cm.coolwarm,
     linestyles='solid')
     CP.collections[4].remove()
# Elimina la visualización del contorno exterior
     ax.set_xticks(np.arange(0, 81, step=20))
     ax.set_yticks(np.arange(0, 81, step=20))
     plt.xlabel('cm')
     plt.ylabel('cm')

# Se grafica el campo para cada uno de los tiempos guardados
for subplot_num, trazar_punto in enumerate(trazar_puntos):
     ax = fig.add_subplot(1, 2,subplot_num*2+1, projection='3d')
     plot_e_field(ax,
     trazar_punto['datos_para_graficar'],trazar_punto['numero_pasos'],
     trazar_punto['z_scale'],trazar_punto['etiqueta'])
     ax = fig.add_subplot(1, 2, subplot_num * 1+2 )
     plot_e_field_contour(ax, trazar_punto['datos_para_graficar'])
plt.tight_layout()
plt.subplots_adjust(bottom=0.067, left=0.02, hspace=0,
wspace=0.25, top=0.9676, right=0.921)
plt.show()
```

G. Guía de onda unidimensional con condiciones de frontera PEC.

```
#Librerias
import numpy as np
import matplotlib as mpl
import scipy.fftpack as fourier
import matplotlib.pyplot as plt
import time
from matplotlib.animation import FuncAnimation
from matplotlib.animation import FuncAnimation, PillowWriter


#Nucleo principal FDTD
def FDTD(npasos):
    #Calculo campo magnetico hy
    for k in range(0,ke-1):
        hy[k]=hy[k]+alpha[1][k]*(ex[k]-ex[k+1])

    #Fuente
    ex[Ij]=ex[Ij] +Onda(fc,t[npasos])

    #calculo campo electrico ex
    for k in range(1,ke-1):
        ex[k]=ex[k] + alpha[1][k]*(hy[k-1]-hy[k])

# Fuente Blackman Harris window
def Onda(fc, t):
    a0= 0.3532222
    a1= -0.488
    a2= 0.145
    a3= -0.010222
    T= 1/fc
    return a0+a1*np.cos(2*np.pi*t/T)+a2*np.cos(2*2*np.pi*t/T)+
    a3*np.cos(6*2*np.pi*t/T)

#parametros metodo de diferencias finitas
PPW= 10
CFL= 0.9

#Parametros para la cavidad resonante
#Dimensiones SI
```

```
a=0
b=1
L=b-a
fmin= 0
fmax = 3e9
fc=fmax / 3.351
T=1/fc
e0=8.8541e-12
m0 = (4e-7)*np.pi
c0=1/np.sqrt(e0*m0)
lam_min = c0/fmax
zj = np.sqrt(2)/2
zo= np.sqrt(2)/4

#parametros de discretización del espacio y el tiempo
dz = lam_min/PPW
ke = np.ceil(L/dz)+1
ke = ke.astype(int)
z = np.linspace(a, b, ke)
z2 = np.linspace(a+0.5*dz, b-0.5* dz, ke-1)
Ij = np.ceil(zj/dz)+1
Ij = Ij.astype(int)
keo = np.ceil(zo/dz)+1
keo = keo.astype(int)
dt = CFL * dz/c0
NT =5 #Número de periodos de la señal BHW
N = np.ceil(T * NT / dt)
N = N.astype(int)
t = np.linspace(0,(N-1)*dt,N)

# Discretizació de los campos y parámetros
ex = np.zeros(ke)
hy = np.zeros(ke-1)
exo = np.zeros(N)
hyo = np.zeros(N)
alpha = np.ones((2, ke-1))
alpha[1] = alpha[1] * dt/((np.sqrt(e0*m0))*dz)


# crear figuras y ejes
fig = plt.figure(figsize=(8, 8))
plt.rcParams.update({'font.size':20})
aE= fig.add_subplot(211)
```

```
aH = fig.add_subplot(212)
colores = plt.get_cmap('bwr', N)
#plot
for n in range(N):
    FDTD(n)
    aE.plot(z, ex, color=colores(n), linewidth=1)
    aH.plot(z2, hy, color=colores(n), linewidth=1)
    hyo[n]=hy[keo]
    exo[n]=ex[keo]
aE.set_xlabel('z',fontsize='20')
aH.set_xlabel('z',fontsize='20')
aE.set_xlabel('Ex',fontsize='20')
aH.set_ylabel('Hy',fontsize='20')
plt.show()



# Función para la animación
def animación(n):
    FDTD(n)
    f_E.set_data(z, ex)
    f_H.set_data(z2, hy)
    aE2.set(xlim=(min(z), max(z)), ylim=(-5, 5))
    aH2.set(xlim=(min(z), max(z)), ylim=(-5, 5))

# Se crea una nueva figura para la animación
fig2= plt.figure(figsize=(6, 4))
aE2= fig2.add_subplot(211)
aH2= fig2.add_subplot(212)

# Variables del plot
f_E, = aE2.plot([], [], linewidth=2.5, color= 'blue')
f_H, = aH2.plot([], [], linewidth=2.5, color= 'red')
# Se crea la animación
ani= FuncAnimation(fig=fig2, func=animación, frames=range(len(t)),
interval=100, repeat=1)
ani.save("me.gif", writer='imagemagick', fps=60)
plt.xlabel('z')
plt.suptitle('Animación para el campo eléctrico y campo magnético')
plt.show()


#Graficas de Fourier para el campo eléctrico
```

```
Ef = fourier.fft(exo)
Ef = fourier.fftshift(Ef)
absEf = (1/N)*np.abs(Ef)
angEf= (1/N)*np.unwrap(np.angle(Ef))
freq = fourier.fftfreq(N, dt)
freq = fourier.fftshift(freq)
N1=np.ceil(freq.size/2)
N1=N1.astype(int)-1
N2=N1+100

# Observardor que está dentro de la cavidad
fig3=plt.figure(figsize=(10,4))
at=fig3.add_subplot(211)
af=fig3.add_subplot(212)
at.plot(t,exo,lw=2,color='black')
af.plot(freq[N1:N2],absEf[N1:N2],lw=2,color='red')
at.set_xlabel('s',fontsize='20')
af.set_xlabel('Hz',fontsize='20')
plt.suptitle('(a)     Evolución temporal del campo eléctrico',fontsize='15')
plt.title('(b)     Transformada rápida de Fourier',fontsize='15')
plt.show()



#Graficas de Fourier para el campo magnético
Hf = fourier.fft(hyo)
Hf = fourier.fftshift(Hf)
absEf = (1/N)*np.abs(Hf)
angEf= (1/N)*np.unwrap(np.angle(Hf))
freq = fourier.fftfreq(N, dt)
freq = fourier.fftshift(freq)
N1=np.ceil(freq.size/2)
N1=N1.astype(int)-1
N2=N1+100
# Observardor que está dentro de la cavidad
fig4=plt.figure(figsize=(10,4))
at=fig4.add_subplot(211)
af=fig4.add_subplot(212)
at.plot(t,hyo,lw=2,color='black')
af.plot(freq[N1:N2],absEf[N1:N2],lw=2,color='red')
at.set_xlabel('s',fontsize='20')
af.set_xlabel('Hz',fontsize='20')
plt.suptitle('(a)     Evolución temporal del campo magnético',fontsize='15')
```

```
plt.title('(b)     Transformada rápida de Fourier',fontsize='15')
plt.show()
```

Referencias

[1] Y. KS, «Numerical solution of initial boundary value problems involving Maxwell"s equations in isotropic media,» *IEEE Trans. Antennas and Propagation*, vol. 14, págs. 302-307, 1966.

[2] J. D. Jackson, *Classical electrodynamics*, 1999.

[3] D. J. Griffiths, *Introduction to electrodynamics*, 2005.

[4] G. F. SIMMONS, «ECUACIONES DIFERENCIALES/George F. Simmons.,» inf. téc.

[5] O. González Rodrıguez et al., *Extensión del método de las diferencias finitas en el dominio del tiempo para el estudio de estructuras hıbridas de microondas incluyendo circuitos concentrados activos y pasivos*. Universidad de Cantabria, 2009.

[6] E. A. N. Camba, *Diferencias finitas en el dominio del tiempo (DFDT) para el análisis de dispositivos pasivos de microondas*. Universitat de Valencia (Spain), 1992.

[7] F. J. Harris, «On the use of windows for harmonic analysis with the discrete Fourier transform,» *Proceedings of the IEEE*, vol. 66, n.º 1, págs. 51-83, 1978.

[8] A. Taflove y M. E. Brodwin, «Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations,» *IEEE transactions on microwave theory and techniques*, vol. 23, n.º 8, págs. 623-630, 1975.

[9] A. Taflove, S. C. Hagness y M. Piket-May, «Computational electromagnetics: the finite-difference time-domain method,» *The Electrical Engineering Handbook*, vol. 3, págs. 629-670, 2005.

[10] J.-P. Berenger, «A perfectly matched layer for free-space simulation in finite-difference computer codes,» en *Annales des télécommunications*, vol. 51, 1996, págs. 39-46.

[11] J.-P. Bérenger, «An effective PML for the absorption of evanescent waves in waveguides,» *IEEE Microwave and guided wave letters*, vol. 8, n.º 5, págs. 188-190, 1998.

[12] T. SHANMUGANANTHAM y S. RAGHAVAN, «Modeling and Analysis of Printed Antenna Using Finite Difference Time Domain Algorithm,»

[13] J.-P. Bérenger, «Evanescent waves in PML's: Origin of the numerical reflection in wave-structure interaction problems,» *IEEE Transactions on Antennas and Propagation*, vol. 47, n.º 10, págs. 1497-1503, 1999.

[14] A. S. Putra, I. S. Alam, W. Srigutomo y A. T. Bon, «Numerical Simulation of Finite Difference Time Domain (FDTD) for Solving the Boundary Value Problem (BVP) in Earth Layer,»

[15] N. Matthew y O. Sadiku, «Elementos de electromagnetismo,» *España, Crıtica*, 2003.

[16] M. N. Sadiku y S. Nelatury, *Elements of electromagnetics*. Oxford university press New York, 2001, vol. 428.

[17] H. P. Langtangen, *Computational partial differential equations: numerical methods and diffpack programming*. Springer Berlin, 2003, vol. 2.

[18] G. Singh, D. K. Misra, C. Tanwar y P. Narang, «Visualization and Analysis of Populations,» en *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, IEEE, vol. 1, 2022, págs. 364-368.

[19] J. B. Schneider, «Understanding the finite-difference time-domain method,» *School of electrical engineering and computer science Washington State University*, vol. 28, 2010.

[20] G. Ohner, *Two dimensional finite difference time domain computation of electromagnetic fields in python*, 2018.

[21] H. Loui, «1D-FDTD using MATLAB,» *ECEN-6006 Numerical Methods in Photonics Project*, vol. 1, 2004.

[22] F. Monsefi, L. Carlsson, M. Rančić, M. Otterskog y S. Silvestrov, «Solution of two-dimensional electromagnetic scattering problem by FDTD with optimal step size, based on a semi-norm analysis,» en *AIP Conference Proceedings*, American Institute of Physics, vol. 1637, 2014, págs. 683-690.

[23] M. Benavides-Cruz, C. Calderón-Ramón, J. Gomez-Aguilar et al., «Numerical simulation of metallic nanostructures interacting with electromagnetic fields using the Lorentz–Drude model and FDTD method,» *International Journal of Modern Physics C*, vol. 27, n.º 04, pág. 1 650 043, 2016.

[24] A. I. Hadarig et al., «Desarrollo de dispositivos pasivos de guia de onda y microstrip en la banda de 220 GHz–325 GHz,» 2013.

[25] J. E. Houle y D. M. Sullivan, *Electromagnetic Simulation Using the FDTD Method with Python*. John Wiley & Sons, 2020.