Propagation d'Ondes Électromagnétiques en Dimension 1

Université Côte d'Azur

Département Master Ingénierie Mathématique, campus Valrose

Encadrante de mémoire : Claire SCHEID

Etudiant : Qinyan YANG

05/05/2024

Sommaire

I. Introduction

Les équations de Maxwell décrivent comment les ondes électromagnétiques se propagent à travers différents milieux. Ces équations, initialement formulées en trois dimensions, peuvent être simplifiées en une seule dimension sous certaines conditions de symétrie. Cette simplification est cruciale pour réduire la complexité des calculs tout en conservant les caractéristiques essentielles du phénomène étudié.

L'objectif de ce projet est d'explorer numériquement la propagation des ondes électromagnétiques en une dimension et en deux dimensions en utilisant les équations de Maxwell simplifiées. L'approche principale pour cette étude sera la méthode des différences finies, en particulier le schéma de Yee (FDTD), réputé pour son efficacité dans le traitement numérique des équations de Maxwell.

Nous validerons notre modèle en le testant sur diverses configurations. Nous envisagerons également l'extension de notre étude à des situations plus complexes. Cette recherche vise non seulement à confirmer la validité de l'approche numérique, mais aussi à mieux comprendre la dynamique des ondes électromagnétiques en dimension réduite.

Dans le cadre de ce projet, quatre fichiers Python ont été développés qui sont '1D_Maxwell.py', '1D_generalisation.py', '1D_milieux_diff.py', '2D_Maxwell.py'. Tous les fichiers Python sont également inclus en annexe.

II. Modélisation mathématique

II.1 Les équations de Maxwell

Les équations de Maxwell sont un ensemble de quatre équations fondamentales qui décrivent le comportement des champs électriques et magnétiques, ainsi que leurs interactions avec la matière. Elles constituent la base de l'électromagnétisme classique, de la physique des ondes et de la technologie électrique. Ces quatre équations sont respectivement appelées loi d'Ampère, loi de Faraday, loi de Gauss et loi d'absence de monopôles magnétiques. Les équations de Maxwell sont données sous forme différentielle et intégrale par

$$\int_{\partial S} H\, dl = \int_S J \cdot dS + \frac{d}{dt}\left(\int_S D \cdot dS\right)$$

$$\int_{\partial S'} E \cdot dl = -\frac{d}{dt}\left(\int_{S'} B \cdot ds\right)$$

$$\int_{\partial V} D \cdot ds = \int_V \rho\, dv$$

$$\int_{\partial v'} B \cdot ds = 0$$

où E est le champ électrique, B est l'induction magnétique, H est le champ magnétique et D est le déplacement électrique. S, S' sont des surfaces quelconques de $\mathbb{R}^3$, et V, V ' sont des volumes quelconques de $\mathbb{R}^3$. ρ est une fonction à valeurs réelles, ou scalaire, appelée densité de charge électrostatique. J est une fonction à valeurs dans $\mathbb{R}^3$, appelée densité de courant.

Sous la forme d'équations dérivées partielles, dans le cas d'un milieu homogène, non dispersif, linéaire, les équations de Maxwell sont données respectivement par :

$$\nabla \times H = \varepsilon_0\left(\frac{\partial E}{\partial t} + J_e\right) \quad (1)$$

$$\nabla \times E = -\mu_0 \frac{\partial H}{\partial t} \quad (2)$$

$$\nabla \cdot E = \frac{\rho}{\varepsilon_0}$$

$$\nabla \cdot B = 0$$

Si la propagation dans le vide, μ_0 est la permittivité magnétique du vide et ε_0 est la permittivité électrique du vide.

Dans le cadre de ce projet, nous utilisons principalement les équations de (1) et (2), car les deux dernières équations sont obtenues par la divergence de l'équation (1) et l'équation (2) respectivement. Afin de simplifier les calculs lors de la réduction des équations à deux dimensions puis à une dimension, nous faisons les hypothèses suivantes : $J_e = 0$, $\mu_0 = 1$ et $\varepsilon_0 = 1$. Ces simplifications nous permettent de reformuler les équations de Maxwell sur les rotations, et on obtient :

$$\nabla \times H = \frac{\partial E}{\partial t} \qquad (3)$$

$$\nabla \times E = -\frac{\partial H}{\partial t} \qquad (4)$$

II.2 Réduction à deux dimensions

Supposons que le champ électrique $E : (t,x,y,z) \mapsto E(t,x,y,z)$, le champ magnétique $H(t,x,y,z) \mapsto E(t,x,y,z)$ pour $t\epsilon[0,T], (x,y,z)\epsilon\mathbb{R}^3$ et l'onde électromagnétique se propagent dans le plan de x et y, elle peut être décrite par E et H qui varient spatialement et temporellement dans ce plan. Nous supposons que le domaine soit invariant dans la direction z tout comme les conditions initiales (et de bords), et nous cherchons des solutions invariantes dans la direction z. Donc, E et H sont indépendants de z (i.e. $\frac{\partial E}{\partial z} = 0$ et $\frac{\partial H}{\partial z} = 0$), Nous avons donc,

Pour le champ magnétique :

$$\frac{\partial H}{\partial t} = -\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \wedge \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix}$$

$$= -\begin{pmatrix} \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} \\ \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} \\ \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \end{pmatrix}$$

$$= -\begin{pmatrix} \dfrac{\partial E_z}{\partial y} \\ -\dfrac{\partial E_z}{\partial x} \\ \dfrac{\partial E_y}{\partial x} - \dfrac{\partial E_x}{\partial y} \end{pmatrix}$$

Pour le champ électrique :

$$\frac{\partial E}{\partial t} = \begin{pmatrix} \dfrac{\partial}{\partial x} \\ \dfrac{\partial}{\partial y} \\ \dfrac{\partial}{\partial z} \end{pmatrix} \wedge \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix}$$

$$= \begin{pmatrix} \dfrac{\partial H_z}{\partial y} - \dfrac{\partial H_y}{\partial z} \\ \dfrac{\partial H_x}{\partial z} - \dfrac{\partial H_z}{\partial x} \\ \dfrac{\partial H_y}{\partial x} - \dfrac{\partial H_x}{\partial y} \end{pmatrix}$$

$$= \begin{pmatrix} \dfrac{\partial H_z}{\partial y} \\ -\dfrac{\partial H_z}{\partial x} \\ \dfrac{\partial H_y}{\partial x} - \dfrac{\partial H_x}{\partial y} \end{pmatrix}$$

Ex, Ey et *Ez* représentent respectivement les composantes de *E* sur les axes x, y et z. De même, *Hx, Hy* et *Hz* désignent les composantes de H sur les axes x, y et z, respectivement.

Nous pouvons découpler les équations en deux systèmes :

- Pour (E_z, H_x, H_y), nous avons :

$$\frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\partial y} \qquad (5)$$

$$\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} \qquad (6)$$

$$\frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \qquad (7)$$

- Pour (E_x, E_y, H_z), nous avons :

$$\frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \quad (8)$$

$$\frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} \quad (9)$$

$$\frac{\partial E_y}{\partial t} = -\frac{\partial H_z}{\partial x} \quad (10)$$

II.3 Réduction à une dimension

Suite aux étapes précédentes, nous avons dérivé les équations de Maxwell en deux dimensions. De manière similaire, il est également possible d'obtenir les équations en une dimension.

Supposons maintenant que l'onde électromagnétique se propage dans la direction x, la description de la dynamique des champs électrique E et magnétique H peut être simplifiée en prenant en compte que la variation des champs se fait principalement le long de cet axe. Nous cherchons, E et H indépendants de y et de z (i.e. $\frac{\partial E}{\partial z} = 0$, $\frac{\partial E}{\partial y} = 0, \frac{\partial H}{\partial y} = 0$ et $\frac{\partial H}{\partial z} = 0$), nous avons donc,

$$\frac{\partial H}{\partial t} = -\begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \wedge \begin{pmatrix} E_x \\ E_y \\ E_z \end{pmatrix}$$

$$= -\begin{pmatrix} \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} \\ \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} \\ \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial E_z}{\partial x} \\ -\frac{\partial E_y}{\partial x} \end{pmatrix}$$

$$\frac{\partial E}{\partial t} = \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} \wedge \begin{pmatrix} H_x \\ H_y \\ H_z \end{pmatrix}$$

$$= \begin{pmatrix} \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \\ \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \\ \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \end{pmatrix}$$

$$= \begin{pmatrix} -\frac{\partial H_z}{\partial x} \\ \frac{\partial H_y}{\partial x} \end{pmatrix}$$

Nous pouvons découpler les équations en deux systèmes :

- Pour (E_z, H_y), nous avons :

$$\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} \quad (11)$$

$$\frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} \quad (12)$$

- Pour (E_y, H_z), nous avons :

$$\frac{\partial E_y}{\partial t} = -\frac{\partial H_z}{\partial x} \quad (13)$$

$$\frac{\partial H_z}{\partial t} = -\frac{\partial E_y}{\partial x} \quad (14)$$

Nous obtenons deux ensembles d'équations ((11), (12) et (13), (14)) décrivant deux cas différents, nous pouvons donc choisir l'un des deux ensembles d'équations pour construire un schéma de Yee. Dans la suite de ce projet, nous pourrons utiliser les équations (11) et (12).

III. Modélisation numérique

III.1 Schéma de Yee

Pour résoudre numériquement les équations de Maxwell, nous introduisons ici le schéma de Yee, qui est couramment utilisé en électromagnétisme. C'est une méthode de calcul de différences finies dans le domaine temporel, qui permet de résoudre des équations Maxwell dépendantes du temps.

Dans cette section, notre but est de construire un schéma de Yee pour les équations de Maxwell (équations (11) et (12)) en une dimension.

Pour commencer, définissons les domaines de propagation d'onde électromagnétique, comprenant le domaine temporel et les domaines spatiaux. Nous définissons le domaine temporel comme étant $[t_{ini}, t_{fin}]$ et prenons les axes x, y et z comme trois axes spatiaux orthogonaux. Sur ces axes, nous fixons x dans l'intervalle $[x_{ini}, x_{fin}]$, y dans l'intervalle $[y_{ini}, y_{fin}]$, et z dans l'intervalle $[z_{ini}, z_{fin}]$,

Pour simplifier, nous utilise pour la suite, $[t_{ini}, t_{fin}] = [0, T]$, où T est un réel positif, et $[x_{ini}, x_{fin}] = [y_{ini}, y_{fin}] = [z_{ini}, z_{fin}] = [0, L]$, avec L étant un réel positif. Ensuite, nous procédons à la discrétisation temporelle et spatiale sur chaque axe.

Pour la discrétisation temporelle, nous faisons mailler $[0, T]$, en se donnant $N \epsilon \mathbb{N}^*$, et $N+1$ points de discrétisation, forme une subdivision de l'intervalle, notons les $(t_n)_{n\epsilon\{0,\dots,N\}}$. Ils vérifient $t_0 = 0$, $t_{I+1} = T$ et $t_n < t_{n+1}$, $\forall n \epsilon \{0, \dots, N\}$. Pour simplifier, nous choisissons une subdivision uniforme de pas $\Delta t > 0$, i.e. $\forall n \epsilon \{0, \dots, N\}$ $t_{n+1} - t_n = \Delta t$. En particulier : $\Delta t = \frac{T}{N}$; $t_n = n\Delta t$.

Pour la discrétisation spatiale sur l'axe x (et respectivement des axes y et z), nous faisons le maillage sur $[0, L]$, en se donnant $I \epsilon \mathbb{N}^*$ et $I+1$ points de discrétisation, formant une subdivision de l'intervalle. Notons les $(x_i)_{i\epsilon\{0,\dots,I\}}$ (respectivement, $(y_j)_{j\epsilon\{0,\dots,I\}}$ et $(z_k)_{k\epsilon\{0,\dots,I\}}$). Ils vérifient $x_0 = y_0 = z_0 = 0$, $x_{I+1} = y_{I+1} = z_{I+1} = L$ et $x_i < x_{i+1}$, $\forall i \epsilon \{0, \dots, I\}$ (respectivement, $y_j < y_{j+1}$, $\forall j \epsilon \{0, \dots, I\}$ et $z_k < z_{k+1}$, $\forall k \epsilon \{0, \dots, I\}$). Prenons $\Delta x > 0$ i.e. $\forall i \epsilon \{0, \dots, I\}$ $x_{i+1} - x_i = \Delta x$, alors $\Delta x = \Delta y = \Delta z$. En particulier : $\Delta x = \Delta y = \Delta z = \frac{L}{I}$.

La méthode de Yee a utilisé des expressions de différences finies centrées pour les dérivées spatiales et temporelles qui sont à la fois simples à programmer et précises au second ordre dans les incréments spatiaux et temporels. Son expression pour la première dérivée partielle spatiale de E dans la direction x évaluée au temps fixe t_n, $\forall n\epsilon\{0, \dots, N\}$ est :

$$\frac{\partial E}{\partial x}(\, i\Delta x, j\Delta y, k\Delta x, n\Delta t) = \frac{E^n_{i+1/2,j,k} - E^n_{i-1/2,j,k}}{\Delta x} + o(\Delta x)^2 \qquad (15)$$

Pour la première dérivée partielle temporelle de E évaluée au point spatial fixe (i, j, k), $\forall i, j, k\epsilon\{0, \dots, I\}$, par analogie :

$$\frac{\partial E}{\partial t}(\, i\Delta x, j\Delta y, k\Delta x, n\Delta t) = \frac{E^{n+1/2}_{i,j,k} - E^{n-1/2}_{i,j,k}}{\Delta t} + o(\Delta t)^2 \qquad (16)$$

Nous procédons en partant du schéma de Yee en trois dimensions, que nous réduisons ensuite en deux dimensions, puis finalement en une seule dimension.

Comme illustré dans la figure_1 que nous avons dans [1], tridimensionnellement l'algorithme de Yee centre ses composantes E et H dans l'espace tridimensionnel de telle sorte que chaque composante E soit entourée de quatre composantes H circulantes et que chaque composante H soit entourée de quatre composantes E circulantes.

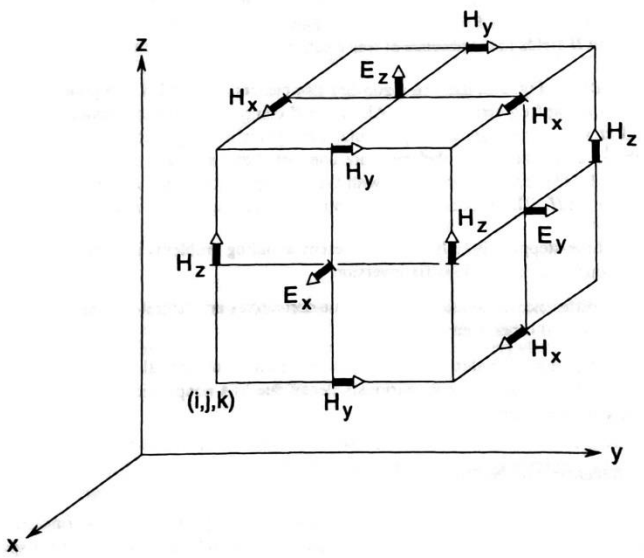

Figure_1

Dans la cellule, le centre de la surface x=i est de coordonnées (i, j+$\frac{1}{2}$, k+$\frac{1}{2}$). Comme dans la figure_1, en 3D $Ex|_{i,j+\frac{1}{2},k+\frac{1}{2}}$ est de coordonnées (i, j+$\frac{1}{2}$, k+$\frac{1}{2}$) qui est décrit par $Hz|_{i,j+1,k+\frac{1}{2}}$, $Hz|_{i,j,k+\frac{1}{2}}$, $Hy|_{i,j+\frac{1}{2},k+1}$ et $Hy|_{i,j+\frac{1}{2},k}$ qui sont autour de $Ex|_{i,j+\frac{1}{2},k+\frac{1}{2}}$.

Et les champs Ey et Ez sont décrits par des champs magnétiques qui les entourent. Pour décrire les champs Hx, Hy et Hz, nous pouvons les centrer, et de façon similaire Hx, Hy et Hz sont décrits par des champs électriques qui les entourent.

En deux dimensions, le schéma de Yee en deux dimensions pour les équations (5), (6) et (7) est donné par [1], i.e.

$$Ez\Big|_{i-\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} = Ez\Big|_{i-\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{\Delta x}\left(Hy^{n}_{i,j+\frac{1}{2}} - Hy^{n}_{i-1,j+\frac{1}{2}}\right) - \frac{\Delta t}{\Delta y}\left(Hx^{n}_{i-\frac{1}{2},j+1} - Hx^{n}_{i-\frac{1}{2},j}\right) \qquad (17)$$

$$Hx^{n+1}_{i-\frac{1}{2},j+1} = Hx^{n}_{i-\frac{1}{2},j+1} - \frac{\Delta t}{\Delta y}\left(Ez^{n+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{3}{2}} - Ez^{n+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2}}\right) \qquad (18)$$

$$Hy^{n+1}_{i,j+\frac{1}{2}} = Hy^{n}_{i,j+\frac{1}{2}} + \frac{\Delta t}{\Delta x}\left(Ez^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}} - Ez^{n+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2}}\right) \qquad (19)$$

A l'aide de la figure_2 nous pouvons comprendre les relations d'indice dans l'espace en deux dimensions (dans le plan xy) de la même manière qu'en trois dimensions.

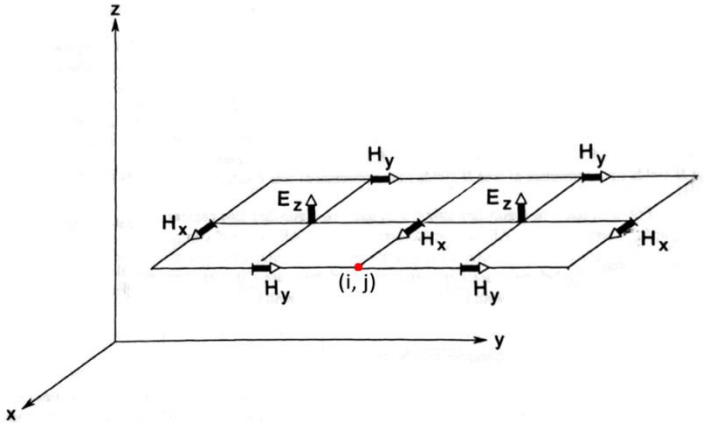

Figure_2

Dans la figure_2, nous avons deux carrés unités représentant les positions des champs Ez, Hx et Hy. Le point rouge est de coordonnées (i, j), donc $Ez|_{i-\frac{1}{2},j+\frac{1}{2}}$ est de coordonnées (i-$\frac{1}{2}$, j+$\frac{1}{2}$) qui est décrit par $Hx|_{i-\frac{1}{2},j+1}$, $Hx|_{i-\frac{1}{2},j}$, $Hy|_{i,j+\frac{1}{2}}$ et $Hy|_{i-1,j+\frac{1}{2}}$; $Hx|_{i-\frac{1}{2},j+1}$ est décrit par $Ez|_{i-\frac{1}{2},j+\frac{1}{2}}$ et $Ez|_{i-\frac{1}{2},j+\frac{3}{2}}$; $Hy|_{i,j+\frac{1}{2}}$ est décrit par $Ez|_{i+\frac{1}{2},j+\frac{1}{2}}$ et $Ez|_{i-\frac{1}{2},j+\frac{1}{2}}$.

Maintenant, nous comprenons le schéma de Yee en multi-dimensions. A partir de la figure_3, nous pouvons d'abord créer une figure en ne conservant que l'axe x pour construire le schéma de Yee en une dimension. Et ci-dessous est la figure pour une dimension.

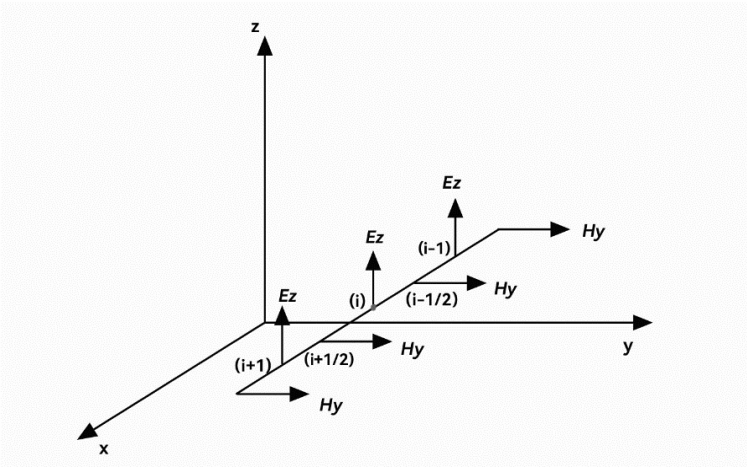

Figure_3

Par la figure_3, nous observons que Ez est décrit par les deux Hy adjacents, de même que Hy est décrit par les deux Ez adjacents. En particulier, pour $Ez|_i$ de coordonnées (i) qui est décrit par $Hy|_{i-\frac{1}{2}}$ et $Hy|_{i+\frac{1}{2}}$, et $Hy|_{i+\frac{1}{2}}$ est décrit par $Ez|_i$ et $Ez|_{i+1}$.

Selon les équations de Maxwell, la variation des champs électrique et magnétique au même endroit dépend des champs électrique et magnétique à cet endroit au temps précédent. Pour cela, la méthode de Yee applique également une méthode de différences finies dans le temps. Par la figure_4 que nous avons dans [4], nous pouvons déterminer la relation des indices temporels. En particulier, pour un instant t donné, $Ez|^{n+1}$est décrit par $Hy|^{n+\frac{1}{2}}$ et $Hy|^{n+\frac{3}{2}}$ de même que $Hy|^{n+\frac{1}{2}}$ est décrit par $Ez|^{n}$ et $Ez|^{n+1}$.

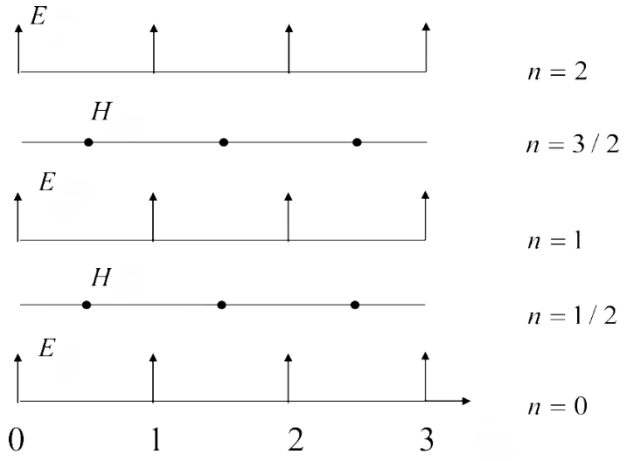

Figure_4

Nous pouvons maintenant utiliser la relation des indices spatiaux et la relation des indices temporels que nous avons obtenu et en appliquant (15) et (16) dans les équations (11) et (12) pour construire le schéma de Yee en une dimension. Nous avons donc :

$$Hy|_{i+\frac{1}{2}}^{n+\frac{1}{2}} = Hy|_{i+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{\Delta x}(Ez|_{i+1}^{n} - Ez|_{i}^{n}) \qquad (20)$$

$$Ez|_{i}^{n+1} = Ez|_{i}^{n} + \frac{\Delta t}{\Delta x}\left(Hy|_{i+\frac{1}{2}}^{n+\frac{1}{2}} - Hy|_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right) \qquad (21)$$

Pour assure la stabilité de la solution numérique par le schéma de Yee, nous devons satisfaire la condition de CFL : quand $\Delta t \to 0$ et $\Delta x \to 0$, $C = \frac{\Delta t}{\Delta x} \leq C_{max}$, où $C_{max} = 1$ que nous obtenons empiriquement par la simulation numérique avec différents Δt et Δx.

III.2 Modélisation en 1D

Pour vérifier la précision du modèle numérique, évaluer la convergence de la méthode numérique et identifier les sources d'erreur numérique, nous allons utiliser une solution exacte pour comparer avec la solution numérique.

III.2.1 Comparaison avec une solution exacte

Pour calculer une solution exacte, nous allons d'abord calculer la dérivée partielle en espace pour l'équation (11) et la dérivée partielle en temps pour l'équation (12), nous avons :

$$\frac{\partial^2 H_y}{\partial x \partial t} = \frac{\partial^2 E_z}{\partial x^2} \qquad (24)$$

$$\frac{\partial^2 E_z}{\partial t^2} = \frac{\partial^2 H_y}{\partial x \partial t} \qquad (25)$$

Puis, nous injectons l'équation (24) dans l'équation (25), nous obtenons donc une équivalence entre l'équation d'onde et les équations de Maxwell en 1D :

$$\frac{\partial^2 E_z}{\partial t^2} = c^2 \frac{\partial^2 E_z}{\partial x^2} \quad \text{où } c=1$$

Nous voulons une solution du système (24) et (25) qui vérifie les conditions initiales $E_z(0,x) = 0$ et $H_y(0,x) = -\cos(\omega x), \forall x$; et les conditions de bords $E_z(t,0) = E_z(t,L) = 0$ et $H_y(t,0) = H_y(t,L) = -\cos(\omega t), \forall t$. Par [1], nous connaissons que la solution de l'équation d'onde est sous la forme suivante :

$$E_z(t,x) = f(x+ct) + g(x-ct) \quad \text{où } f \text{ et } g \text{ sont de classe } \mathbb{C}^2, c=1.$$

Nous posons $f(x+t) = -\frac{1}{2}\cos(\omega(x+t))$, et $g(x-t) = \frac{1}{2}\cos(\omega(x-t))$, donc :

$$E_z(t,x) = \sin(\omega t)\sin(\omega x) \qquad (26)$$

Pour vérifier les conditions de bords de E_z, prenons $\omega = \frac{k\pi}{L}$, $k\epsilon\mathbb{Z}$.

Et par l'équation (11) : $\frac{\partial H_y}{\partial t} = \omega \sin(\omega t)\cos(\omega x)$

Nous l'intégrons $H_y(t,x) = \int \omega \sin(\omega t)\cos(\omega x)\, dt$ et nous obtenons :

$$H_y(t,x) = -cos(\omega t)\cos(\omega x) + h(x) \qquad (27)$$

Pour vérifier la condition initiale de H_y, nous prenons $h(x) = 0$, $\forall x$.et pour les conditions de bords de H_y, prenons $\omega = \frac{k\pi}{L}$, $k\epsilon\mathbb{Z}$ et k est pair.

Nous pouvons maintenant utiliser les équations (26) et (27) comme la solution exacte du système (E_z, H_y) dans le code pour comparer avec la solution obtenue par le schéma de Yee pour les équations (11) et (12).

Dans le code (cf le fichier '1D_Maxwell.py'), nous avons utilisé 1501 points de discrétisation temporelle, pour le dernier indice temporelle t égale à 1500, nous avons obtenu la figure_5.

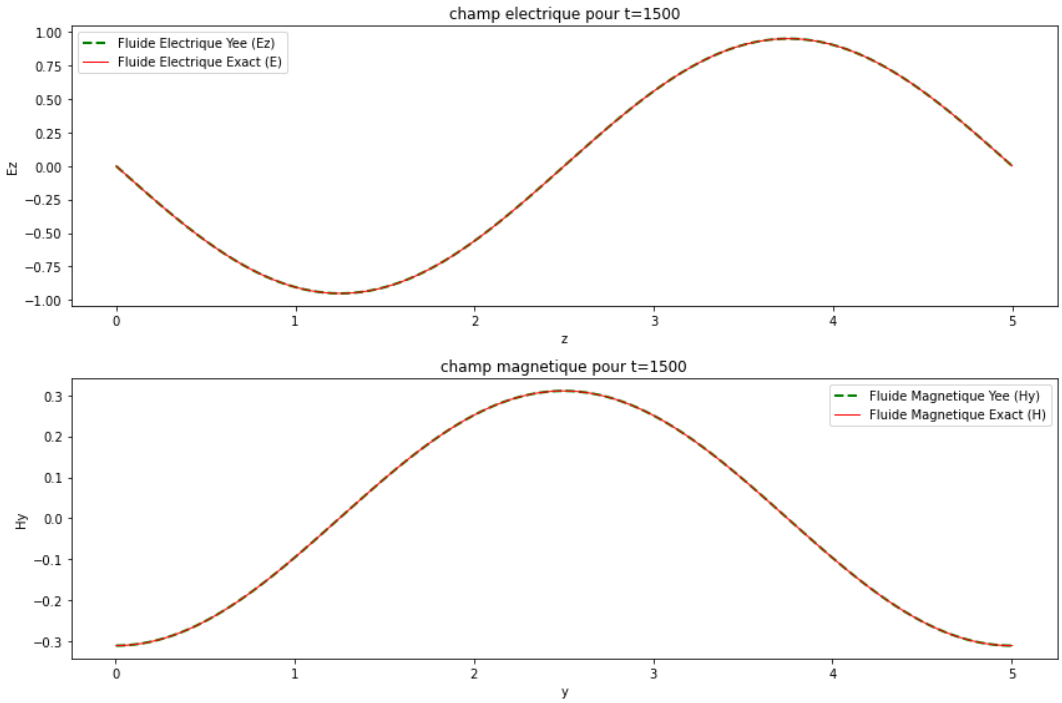

Figure_5

Dans la figure_5, les courbes rouges représentent la solution exacte du champ électrique E_z et du champ magnétique H_y. Les courbes vertes représentent la solution approchée par le schéma de Yee. Nous voyons dans cette figure que la solution approchée converge bien vers la solution exacte. Mais, en zoomant sur la figure, il y a toujours un petit écart représenté par la figure_6 entre les courbes rouges et les courbes vertes, ce qui constitue l'erreur numérique.

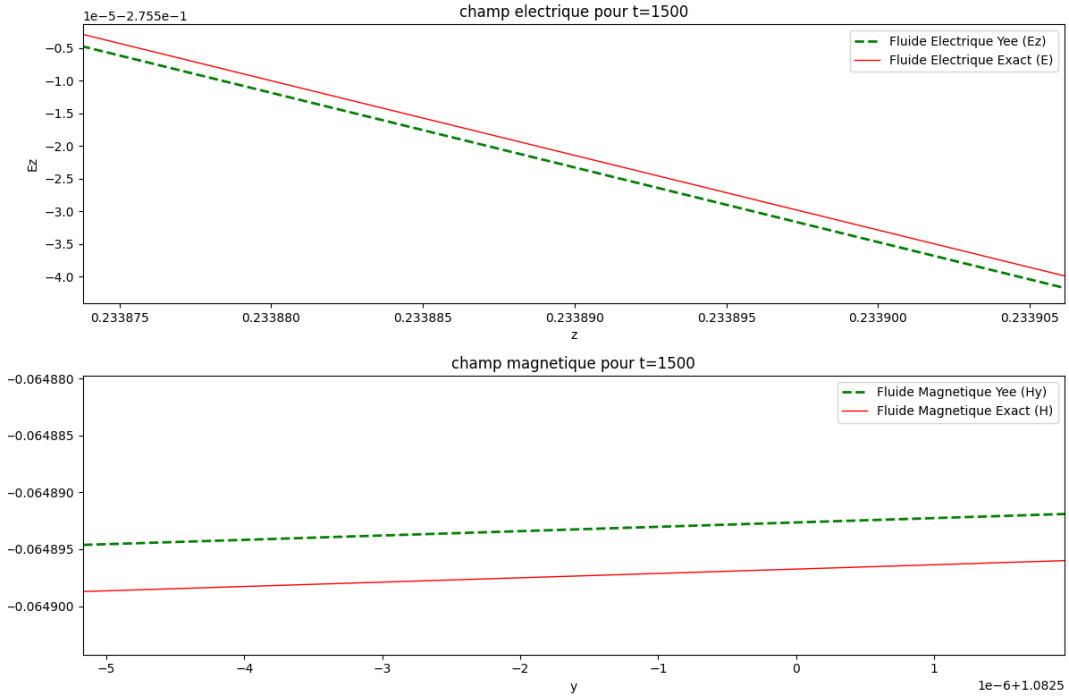

Figure_6

III.2.2 L'erreur et l'ordre de convergence

Pour savoir comment se passe la convergence du schéma, nous allons regarder l'erreur et l'ordre de convergence.

D'abord, nous définissons un ensemble de points de discrétisation en temps et en espace. Nous utilisons $\{I_1, I_2, \dots, I_Z\}$ comme un ensemble de points de discrétisation en espace, $Z \in \mathbb{N}$ et $\forall a \in \{1,2,\cdots Z\}, I_a \in \mathbb{N}^*$. Soit $\alpha = \frac{\Delta t}{\Delta x} = \frac{TI}{NL}$, alors pour satisfaire la condition de CFL, nous posons $\forall a \in \{1,2,\cdots Z\}$, $N_a = \frac{TI_a}{\alpha L}$, puis nous arrondissons N_a à l'entier le plus proche. Donc nous avons un ensemble de points de discrétisation en temps $\{N_1, N_2, \dots, N_Z\}$. Et pour chaque paire de I_a et N_a nous allons étudier l'influence des facteurs spatiaux et temporels sur l'erreur du schéma.

Nous commençons par calculer $\varepsilon_y \epsilon \mathbb{R}^Z$ et $\varepsilon_z \epsilon \mathbb{R}^Z$ la norme 2 des erreurs spatiales de H_y et de E_z respectivement à l'instant fixé pour chaque valeur de N_a, $\forall a \in$

$\{1,2,\cdots Z\}$. Donc pour $t_a \in \{1,2,\dots N_a\}$ fixé, nous avons :

$$\varepsilon_y|_a = \left\|H|^{t_a} - H_y|^{t_a}\right\|_2,\ \forall a \in \{1,2,\cdots Z\}$$

où $\varepsilon_y|_a$ est le $a^{\text{ème}}$ élément de ε_y, H est la solution exacte, et H_y est la solution approchée.

$$\varepsilon_z|_a = \|E|^{t_a} - E_z|^{t_a}\|_2,\ \forall a \in \{1,2,\cdots Z\}$$

où $\varepsilon_z|_a$ est le $a^{\text{ème}}$ élément de ε_z, E est la solution exacte, et E_z est la solution approchée.

Enfin, en calculant la norme deε_y et ε_z, nous avons ε_{H_y} et ε_{E_z} la norme 2 de l'erreur, sous l'influence spatiale et temporelle :

$$\varepsilon_{H_y} = \left\|\varepsilon_y\right\|_2$$
$$\varepsilon_{E_z} = \|\varepsilon_z\|_2$$

Grâce au code (cf le fichier '1D_Maxwell.py'), nous avons le graphe de l'erreur avec différents I_j et N_j , $\forall j \in \{1,2,\cdots 7\}$ représenté par la figure_7 :

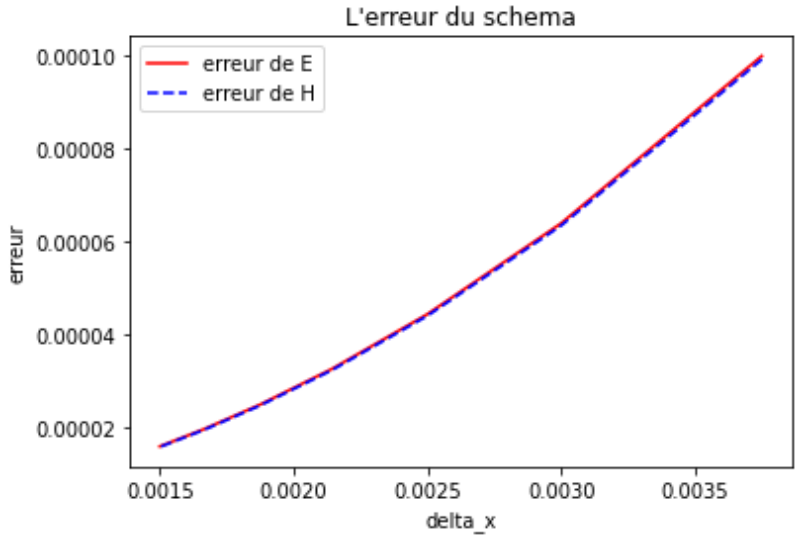

Figure_7

Si le schéma de Yee converge à l'ordre p en temps et à l'ordre q en espace, nous avons :$\|\varepsilon\| \leq C(\Delta t^p + \Delta x^q)$ où C est une constante. Sous la condition de CFL, nous avons $\alpha = \frac{\Delta t}{\Delta x} < 1$, donc $\|\varepsilon\| \leq C(\alpha^p \Delta x^p + \Delta x^q)$. Nous pouvons encore l'écrire : $\|\varepsilon\| \leq C\Delta x^{\min(p,q)}$. Par [1], nous connaissons qu'ici $p = q = 2$, donc $\|\varepsilon\| \leq C\Delta x^2$.

Preuve :

- Soit $p < q$, $C(\alpha^p \Delta x^p + \Delta x^q) \leq C(\alpha^p + 1)\Delta x^p = C\Delta x^p,\ \Delta x \to 0$
- Soit $p > q$, $C(\alpha^p \Delta x^p + \Delta x^q) \leq C(\alpha^p + 1)\Delta x^q = C\Delta x^q,\ \Delta x \to 0$ □

Enfin pour calculer l'ordre de convergence du schéma, nous calculons la pente de la courbe de $log(\Delta x)$ et $log(\varepsilon)$. Par 1D_Maxwell.py, nous avons obtenu numériquement l'ordre de convergence de H_y et de E_z est égale à 2, ce qui correspond à la valeur théorique que nous avons dans [1], et le graphe de $log - log$ représenté par la figure_8 :

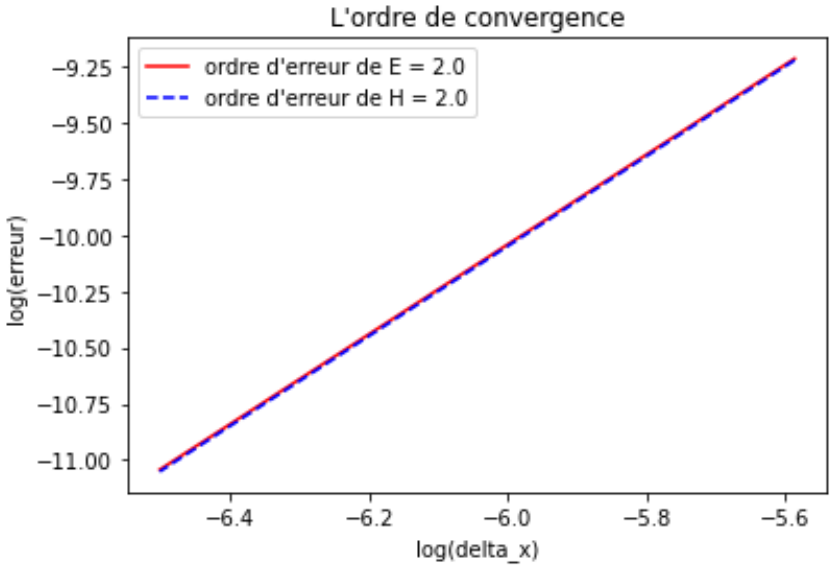

Figure_8

III.3 Généralisation en 1D

Dans le cas général, si l'onde électromagnétique se propage dans un milieu où μ_0 et ε_0 sont différent que 1 et nous mettons une source J_e, donc pour obtenir les équations de Maxwell en 1D, nous devons les recalculer. En fait, nous faisons toujours les mêmes calculs que dans le chapitre 'Modélisation mathématique-Réduction à une dimension', mais en ajoutant les paramètres J_e, μ_0 et ε_0. Nous obtenons donc les équations (28) et (29) correspondent les équations (11) et (12) :

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu_0}\frac{\partial E_z}{\partial x} \qquad (28)$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon_0}\frac{\partial H_y}{\partial x} - \frac{J_e}{\varepsilon_0} \qquad (29)$$

Puis, en ajoutant les paramètres J_e, μ_0 et ε_0 dans le schéma (20) et (21), nous obtenons :

$$Hy|_{i+\frac{1}{2}}^{n+\frac{1}{2}} = Hy|_{i+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{\mu_0 \Delta x}(Ez|_{i+1}^{n} - Ez|_{i}^{n})$$

$$Ez|_{i}^{n+1} = Ez|_{i}^{n} + \frac{\Delta t}{\varepsilon_0 \Delta x}\left(Hy|_{i+\frac{1}{2}}^{n+\frac{1}{2}} - Hy|_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right) - \Delta t \frac{J_e}{\varepsilon_0}$$

Le respect de la condition CFL assure la stabilité du schéma, pour satisfaire la condition de CFL, nous avons ici : quand $\Delta t \to 0$ et $\Delta x \to 0$, $C = c^2 \frac{\Delta t}{\Delta x} = \frac{\Delta t}{\mu_0 \varepsilon_0 \Delta x} \leq C_{max}$, et nous obtenons empiriquement $C_{max} = 1$.

Pour faire une comparaison entre la solution numérique et la solution exacte, nous allons calculer une solution exacte dans le cas général.

Comme précédemment, nous allons calculer la dérivée partielle en espace pour l'équation (28) et la dérivée partielle en temps pour l'équation (29), et nous avons :

$$\frac{\partial^2 H_y}{\partial x \partial t} = \frac{1}{\mu_0}\frac{\partial^2 E_z}{\partial x^2} \qquad (32)$$

$$\frac{\partial^2 E_z}{\partial t^2} = \frac{1}{\varepsilon_0}\frac{\partial^2 H_y}{\partial x \partial t} \qquad (33)$$

Donc sous la forme l'équation d'onde, nous avons :

$$\frac{\partial^2 E_z}{\partial t^2} = c^2 \frac{\partial^2 E_z}{\partial x^2} \quad \text{où } c = \frac{1}{\sqrt{\mu_0 \varepsilon_0}}$$

Comme précédemment, pour la solution du système (32) et (33), prenons les condition initiales $E_z(0,x) = 0$ et $H_y(0,x) = -\frac{1}{\mu_0 \boldsymbol{c}}\cos(\omega x)$, $\forall x$; et les conditions de bords $E_z(t,0) = E_z(t,L) = 0$ et $H_y(t,0) = H_y(t,L) = -\frac{1}{\mu_0 \boldsymbol{c}}\cos(\omega t), \forall t$. Dons posons :

$$E_z(t,x) = \sin(c\omega t)\sin(\omega x)$$

Pour vérifier les conditions de bords de E_z, prenons $\omega = \frac{k\pi}{L}$, $k \epsilon \mathbb{Z}$. Puis par le même calcul que précédemment, nous obtenons que

$$H_y(t,x) = -\frac{1}{\mu_0 \boldsymbol{c}} cos(\omega t)\cos(\omega x) + h(x)$$

Pour vérifier la condition initiale de H_y prenons $h(x) = 0$, $\forall x$ et pour les conditions de bords de H_y, prenons $\omega = \frac{k\pi}{L}$, $k \epsilon \mathbb{Z}$ et k est pair.

Nous utilisons toujours 1501 points de discrétisation temporelle, pour l'indice temporel t égale à 1500, nous avons obtenu la figure_9. Dans cette figure, les courbes rouges représentent la solution exacte du champ électrique E_z et du champ magnétique H_y, les courbes vertes représentent la solution approchée par le schéma

de Yee, comme précédemment. Nous voyons que la solution approchée converge bien vers la solution exacte.

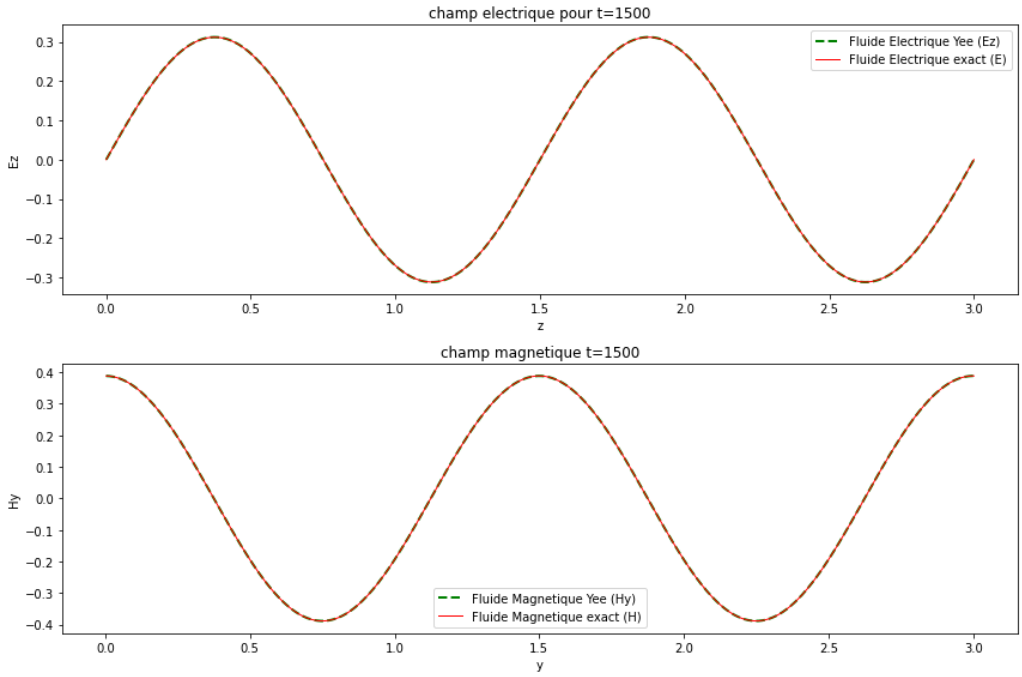

Figure_9

En zoomant sur la figure, nous voyons toujours une petite erreur numérique, représentée dans la figure_10.

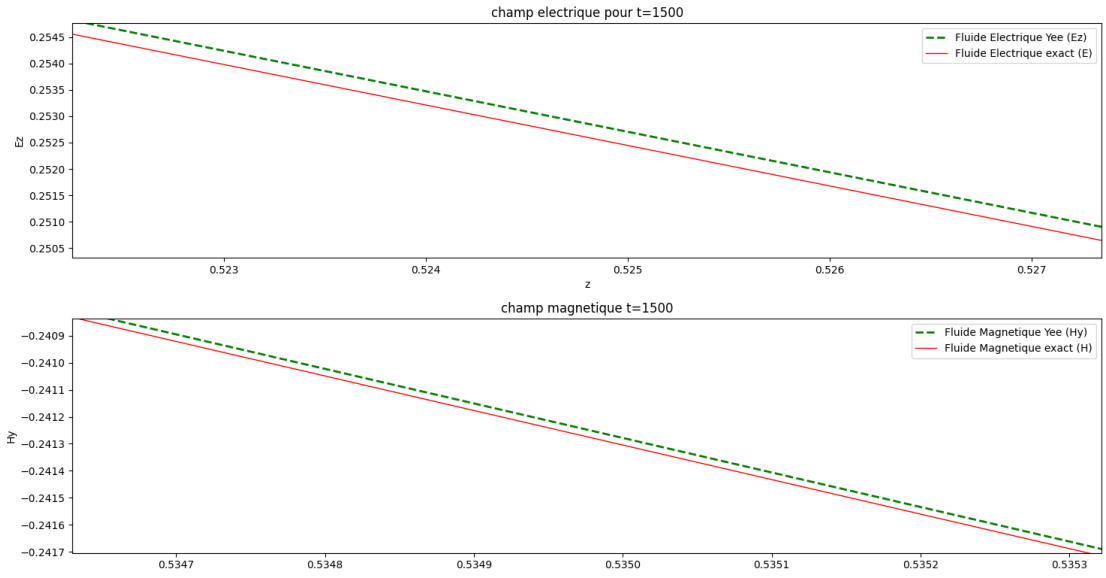

Figure_10

Comme dans la section précédente, nous voulons savoir comment se passe la convergence du schéma dans le code, nous faisons les mêmes calculs que précédemment pour l'erreur et l'ordre de convergence.

Grâce au code (cf le fichier '1D_generalisation.py'), nous avons obtenu le graphe de

l’erreur représenté par la figure_11 :

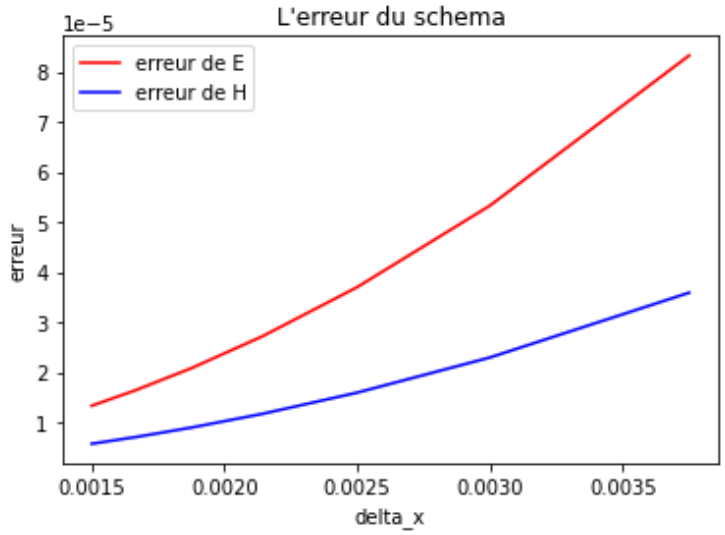

Figure_11

Nous avons obtenu l’ordre de convergence du schéma égale à 2, ce qui est bien vérifié la valeur théorique dans [1], et le graphe de l’ordre de convergence est représenté par la figure_12 :

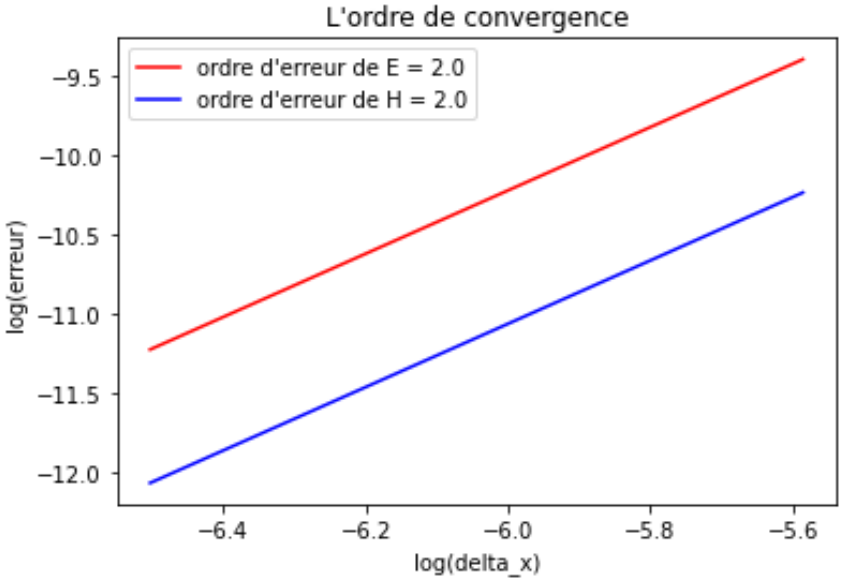

Figure_12

III.4 Propagation dans des milieux différents

Dans cette section, nous nous intéressons la propagation des ondes électromagnétiques dans des milieux unidimensionnels variés. Les propriétés des matériaux influencent fortement la vitesse, l'atténuation et la réflexion des ondes. En

explorant différents types de milieux avec des permittivités électriques et magnétiques variées, nous pouvons mieux comprendre les phénomènes de transmission et de réflexion des ondes électromagnétiques.

Dans le domaine $[0, L]$, nous réglons les permittivités électriques et magnétiques de la région $[\frac{L}{3}, \frac{2L}{3}]$ différemment de celle des autres régions de $[0, L]$ pour simuler la présence d'un autre milieu dans cette zone. Cela ressemble à la situation où la lumière se propage dans l'air, traverse une plaque de verre, puis retourne dans l'air.

Nous choisissons μ_1 comme la permittivité magnétique du milieu de la région $[\frac{L}{3}, \frac{2L}{3}]$ et ε_1 la permittivité électrique du milieu de la région $[\frac{L}{3}, \frac{2L}{3}]$; μ_0 la permittivité magnétique et ε_0 la permittivité électrique pour le reste du domaine $[0, L]$. Grâce au code (cf le fichier '1D_milieux_diff'), nous obtenons la figure_13 illustrant le processus par lequel une onde électromagnétique se propage dans un milieu, traverse un autre milieu, puis retourne dans le milieu d'origine.

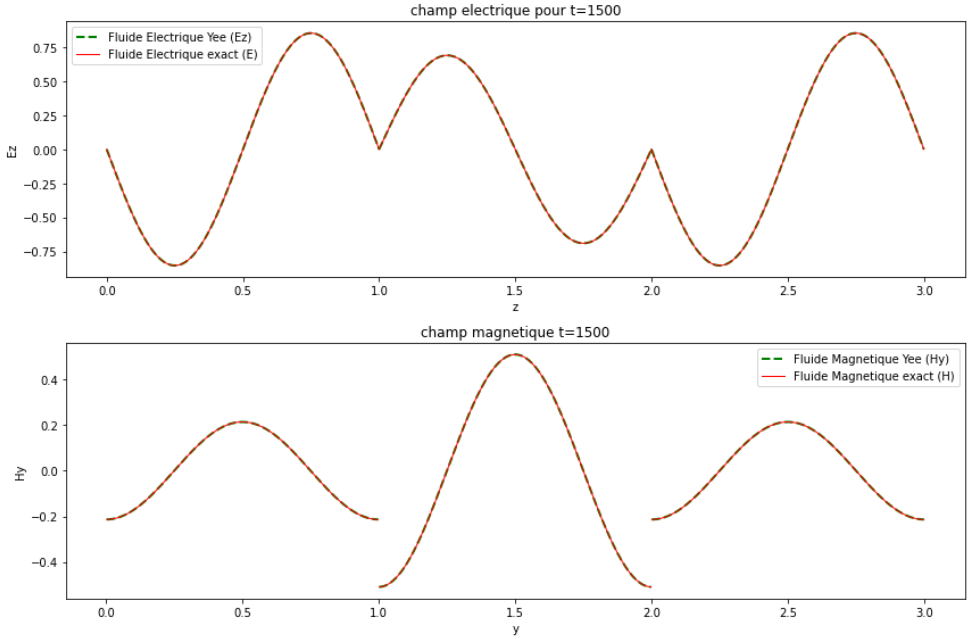

Figure_13

La figure_13 illustre la propagation d'ondes électromagnétiques en une dimension dans l'intervalle [0,3]. Pour indice temporel t=1500, les résultats numériques obtenus par le schéma de Yee sont représentés par la courbe verts, comparés à la solution exacte représentée par la courbe rouge. Nous constatons que les résultats numériques concordent étroitement avec la solution exacte.

Dans l'intervalle [0,3], les formes d'ondes du champ électrique et magnétique sont presque identiques, indiquant que ces deux régions partagent le même type de

milieu. En revanche, à l'intérieur de l'intervalle [1,2], les formes d'ondes du champ électrique et magnétique montrent des variations significatives, suggérant que ce milieu diffère de ceux des segments adjacent et postérieur.

Aux points 1 et 2, les formes d'ondes du champ électrique et magnétique perdent leur caractère régulier, indiquant la présence de frontières entre différents milieux. Lorsque les ondes électromagnétiques atteignent ces frontières, elles subissent partiellement des réflexions et des réfractions. De plus, la discontinuité observée dans le champ magnétique à ces deux points suggère que les phénomènes de réflexion et de réfraction affectent l'amplitude et la phase du champ magnétique sur des interfaces matérielles.

III.5 Modélisation en 2D

III.5.1 Comparaison avec une solution exacte

Comme dans la section précédente, nous allons calculer la dérivée partielle en espace pour l'équation (5) et (6), la dérivée partielle en temps pour l'équation (7), et nous avons :

$$\frac{\partial^2 H_x}{\partial y \partial t} = -\frac{\partial^2 E_z}{\partial^2 y} \qquad (37)$$

$$\frac{\partial^2 H_y}{\partial x \partial t} = \frac{\partial E_z}{\partial x} \qquad (38)$$

$$\frac{\partial^2 E_z}{\partial t^2} = \frac{\partial^2 H_y}{\partial t \partial x} - \frac{\partial^2 H_x}{\partial t \partial y} \qquad (39)$$

Pour une solution du système (37), (38) et (39), nous voulons les conditions initiales :

$$E_z(0, x, y) = 0$$

$$H_x(0, x, y) = \frac{1}{\sqrt{2}} \sin(\omega x) \cos(\omega y)$$

$$H_y(0, x, y) = -\frac{1}{\sqrt{2}} \cos(\omega x) \sin(\omega y)$$

Et les conditions de bords :

$$E_z(t, 0, y) = E_z(t, L, y) = 0, \forall t, y$$
$$E_z(t, x, 0) = E_z(t, x, L) = 0, \forall t, x$$

$$H_x(t, 0, y) = H_x(t, L, y) = 0, \forall t, y$$

$$H_x(t,x,0) = H_x(t,x,L) = \frac{1}{\sqrt{2}}\cos\left(\sqrt{2}\omega t\right)\sin(\omega \mathrm{x}), \forall t,x$$

$$H_y(t,0,y) = H_y(t,L,y) = -\frac{1}{\sqrt{2}}\cos\left(\sqrt{2}\omega t\right)\sin(\omega \mathrm{y}), \forall t,y$$

$$H_y(t,x,0) = H_y(t,x,L) = 0, \forall t,x$$

Puis, nous injectons les équations (37) et (38) dans l'équation (39), nous obtenons donc une équation d'onde en 2D :

$$\frac{\partial^2 E_z}{\partial t^2} = c^2\left(\frac{\partial^2 E_z}{\partial x^2} + \frac{\partial^2 E_z}{\partial^2 y}\right) \quad \text{avec } c\text{=1}$$

Nous utilisons la méthode de séparation des variables pour résoudre cette équation d'onde, nous posons :

$$E_z(t,x,y) = f(t)g(x)h(y) \quad \text{où } f, g \text{ et } h \text{ sont de classe } \mathbb{C}^2$$

Nous avons donc :

$$f''(t)g(x)h(y) = f(t)g''(x)h(y) + f(t)g(x)h''(y) \qquad (40)$$

Pour satisfaire la condition initiale de E_z, nous posons : $f(t) = \sin(at)$. Puis pour les conditions de bords de E_z, pour $x = 0$, posons : $g(x) = \sin(bx)$ et pour $y = 0$, posons : $h(y) = \sin(cy)$, $\forall (a,b,c) \epsilon \mathbb{R}^3$. Et, pour satisfaire l'équation (40), nous choisissons $b = c = \omega$, donc $a = \sqrt{2}\omega$. Nous avons donc :

$$E_z(t,x,y) = \sin\left(\sqrt{2}\omega t\right)\sin(\omega x)\sin(\omega y) \qquad (41)$$

Pour satisfaire les conditions de bords de E_z, prenons $\omega = \frac{k\pi}{l}$, $k \epsilon \mathbb{Z}$

Par l'équation (5) : $\frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\partial y} = -\omega \sin\left(\sqrt{2}\omega t\right)\sin(\omega x)\cos(\omega y)$

Et par l'équation (6) : $\frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} = \omega \sin(\sqrt{2}\omega t)\cos(\omega x)\sin(\omega y)$

Calculons les deux intégrales $H_x(t,x,y) = \int -\omega \sin\left(\sqrt{2}\omega t\right)\sin(\omega x)\cos(\omega y)dt$ et $H_y(t,x,y) = \int \omega \sin(\sqrt{2}\omega t)\cos(\omega x)\sin(\omega y)dt$, pour obtenir H_x et H_y :

$$H_x = \frac{1}{\sqrt{2}}cos\left(\sqrt{2}\omega t\right)\sin(\omega \mathrm{x})\cos(\omega y) + h_x(x,y) \qquad (42)$$

Prenons $h_x(x,y) = 0$, $\forall x,y$ pour satisfaire la condition initiale de H_x et pour les

conditions de bords de H_x, prenons $\omega = \frac{k\pi}{L}$, $k\epsilon\mathbb{Z}$, et k est pair.

$$H_y = -\frac{1}{\sqrt{2}} cos\left(\sqrt{2}\omega t\right) \cos(\omega x)\sin(\omega y) + h_y(x,y) \qquad (43)$$

Prenons $h_y(x,y) = 0, \forall x, y$ pour satisfaire la condition initiale de H_y et pour les conditions de bords de H_y, prenons $\omega = \frac{k\pi}{L}$, $k\epsilon\mathbb{Z}$, et k est pair.

Pour assure la stabilité de la solution numérique par le schéma de Yee, nous devons satisfaire la condition de CFL : quand $\Delta t \to 0$, $\Delta x \to 0$ et $\Delta y \to 0$, $C = \frac{\Delta t}{\Delta x} + \frac{\Delta t}{\Delta y} \leq C_{max}$, où $C_{max} = 1$ que nous obtenons empiriquement par la simulation numérique avec différents Δt, Δx et Δy.

Nous pouvons maintenant utiliser les équations (41), (42) et (43) comme la solution exacte pour approcher par le schéma de Yee. Grace au code (cf le fichier '2D-Maxwell.py'), nous avons la figure_14 :

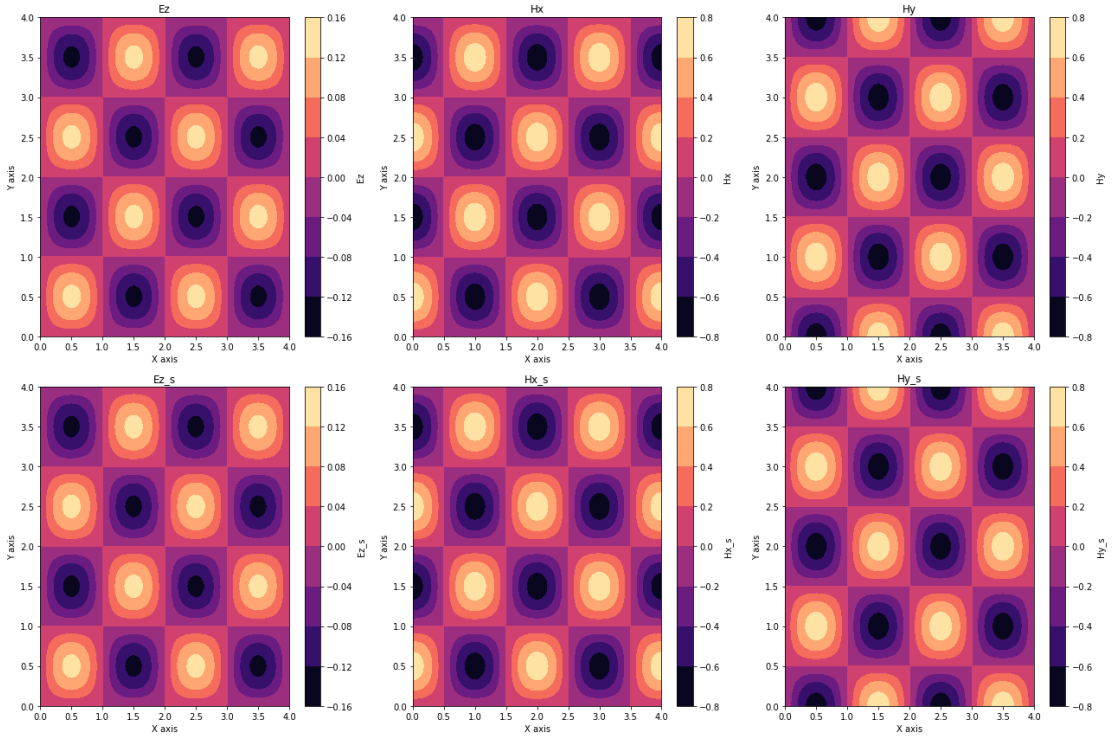

Figure_14

Dans la figure_14, H_x, H_y et E_z représentent les champs magnétiques et du champ électrique approchés par le schéma de Yee, et H_{x_s}, H_{y_s} et E_{z_s} représentent les champs magnétiques et du champ électrique calculé par la solution exacte.

III.5.2 L'erreur et l'ordre de convergence

Pour calculer l'erreur et l'ordre de convergence. Nous utilisons $\{I_1, I_2, \dots, I_Z\}$ comme dans le chapitre précédent, pour vérifier la condition de CFL, nous posons $\forall a \in \{1,2,\cdots Z\}$, $N_a = \frac{TI_a}{\alpha L}$, puis nous arrondissons N_a à l'entier le plus proche. Donc nous avons un ensemble de nombre de discrétisation en temps $\{N_1, N_2, \dots, N_Z\}$. Puis nous allons étudier l'influence des facteurs spatiaux et temporels sur l'erreur du schéma.

Pour l'influence spatiale dans la direction x, nous commençons par calculer la norme deux des erreurs spatiales de H_x, H_y et E_z, notée respectivement $\varepsilon_{x\_x}|_a \epsilon \mathbb{R}^{I_a} \times \mathbb{R}^z$, $\varepsilon_{y\_x}|_a \epsilon \mathbb{R}^{I_a} \times \mathbb{R}^z$ et $\varepsilon_{z\_x}|_a \epsilon \mathbb{R}^{I_a} \times \mathbb{R}^z$, pour $a \in \{1,2,\cdots Z\}$ fixé. Donc pour $t_a \in \{1,2,\dots N_a\}$ fixé et $j_a \in \{1,2,\dots I_a\}$ fixé :

$$\varepsilon_{x\_x}|_a = \left\| H_{x\_s}|_{j_a}^{t_a} - H_x|_{j_a}^{t_a} \right\|_2$$

$$\varepsilon_{y\_x}|_a = \left\| H_{y\_s}|_{j_a}^{t_a} - H_y|_{j_a}^{t_a} \right\|_2$$

$$\varepsilon_{z\_x}|_a = \left\| E_{z\_s}|_{j_a}^{t_a} - E_z|_{j_a}^{t_a} \right\|_2$$

où (H_{x_s}, H_{y_s} , E_{z_s}) est le solution exacte du système (H_x, H_y , E_z) ; (H_x, H_y, E_z) est la solution approchée.

Sous l'influence spatiale dans la direction x, étudions l'influence dans la direction y, nous allons calculer $\varepsilon_{x\_xy}|_a \epsilon \mathbb{R}^z$, $\varepsilon_{y\_xy}|_a \epsilon \mathbb{R}^z$ et $\varepsilon_{z\_xy}|_a \epsilon \mathbb{R}^z$ pour $a \in \{1,2,\cdots Z\}$ fixé :

$$\varepsilon_{x\_xy}|_a = \left\| H_{x\_s}|^{t_a} - |^{t_a} \right\|_2$$

$$\varepsilon_{y\_xy}|_a = \left\| H_{y\_s}|^{t_a} - H_y|^{t_a} \right\|_2$$

$$\varepsilon_{z\_xy}|_a = \left\| E_{z\_s}|^{t_a} - E_z|^{t_a} \right\|_2$$

où (H_{x_s}, H_{y_s} , E_{z_s}) est le solution exacte du système (H_x, H_y , E_z) ; (H_x, H_y, E_z) est la solution approchée.

Finalement, sous l'influence spatiale et temporelle, nous avons $\varepsilon_{H_x} \epsilon \mathbb{R}^z$, $\varepsilon_{H_y} \epsilon \mathbb{R}^z$ et $\varepsilon_{E_z} \epsilon \mathbb{R}^z$:

$$\varepsilon_{H_x}|_a = \left\| \varepsilon_{x\_xy}|_a \right\|_2$$

$$\varepsilon_{H_y}|_a = \left\| \varepsilon_{y\_xy}|_a \right\|_2$$

$$\varepsilon_{E_z}|_a = \left\|\varepsilon_{z\_xy}|_a\right\|_2$$

où $\varepsilon_{H_x}|_a$ est le $a^{\text{ème}}$ élément du vecteur ε_{H_x}, $\varepsilon_{H_y}|_a$ est le $a^{\text{ème}}$ élément du vecteur ε_{H_y} et $\varepsilon_{E_z}|_a$ est le $a^{\text{ème}}$ élément du vecteur ε_{E_z}.

Nous avons obtenu la figure_15 grâce au code (cf le fichier '2D-Maxwell.py') pour le graphe des erreurs :

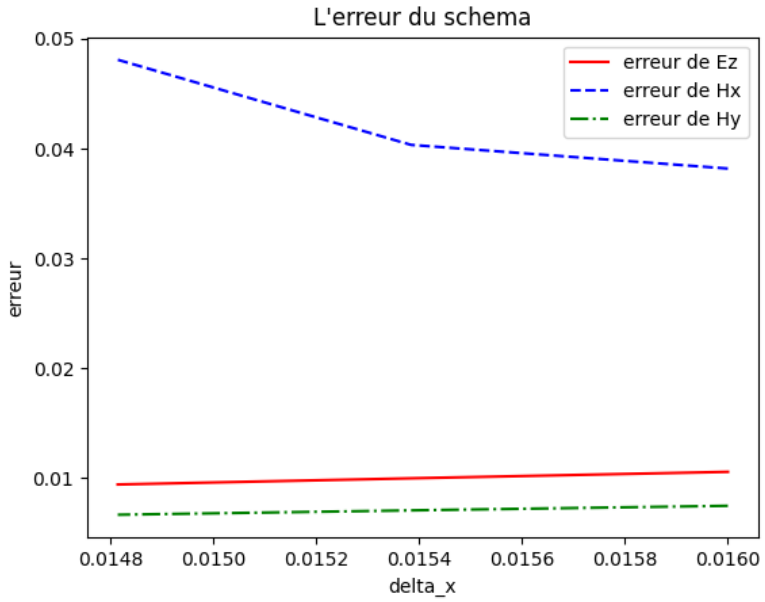

Figure_15

Si le schéma de Yee converge à l'ordre p en temps et en espace le schéma de Yee converge à l'ordre q, nous avons : $\|\varepsilon\| \leq C(\Delta t^p + \Delta x^q)$ où C est une constante. Comme $\alpha = \frac{\Delta t}{\Delta x}$, nous avons : $\|\varepsilon\| \leq C(\alpha^p \Delta x^p + \Delta x^q)$. Comme en 1D, nous avons : $\|\varepsilon\| \leq C\Delta x^{\min(p,q)}$. Par [1], nous connaissons que $p = q = r = 2$, donc $\|\varepsilon\| \leq C\Delta x^2$. Grâce au code (le fichier '2D-Maxwell.py'), nous obtenons le graphe de l'ordre de convergence représenté par la figure_16 :

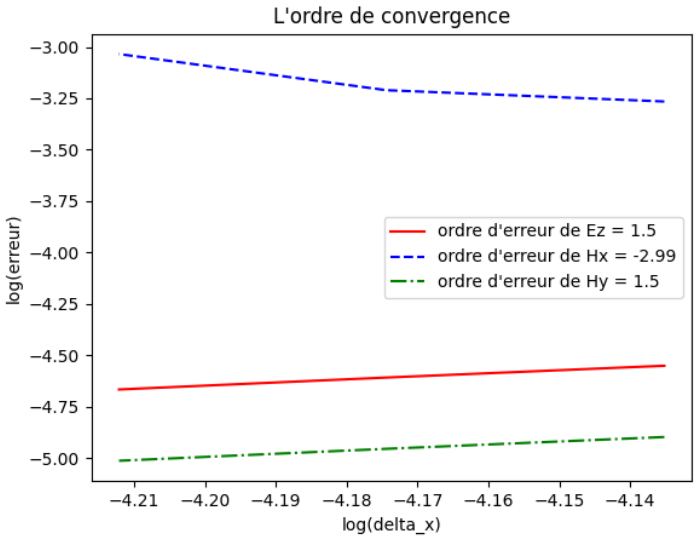

Figure_16

Nous avons obtenu numériquement que l'ordre de convergence pour H_x est d'environ 3, pour H_y est d'environ 1,5, et pour E_z est d'environ 1,5. Ces valeurs ne correspondent pas aux valeurs théoriques, ce qui pourrait indiquer un bug dans le code ou un nombre insuffisant de points de discrétisation.

IV. Conclusion

En conclusion, cette étude a permis de démontrer l'efficacité de l'approche numérique basée sur la méthode des différences finies, et plus particulièrement le schéma de Yee (FDTD), pour simuler la propagation des ondes électromagnétiques en une et deux dimensions. En utilisant les équations de Maxwell simplifiées, nous avons pu réduire la complexité des calculs tout en préservant les caractéristiques essentielles du phénomène.

Les résultats obtenus à partir des diverses configurations testées ont validé notre modèle numérique, confirmant ainsi sa capacité à reproduire fidèlement la dynamique des ondes électromagnétiques. Nos simulations ont montré des résultats satisfaisants pour la propagation en une dimension dans des milieux homogènes, avec des ordres de convergence numérique conformes aux valeurs théoriques. Cependant, pour la propagation en deux dimensions, bien que la simulation dans un milieu homogène ait été réussie, les ordres de convergence numérique obtenus n'ont pas correspondus aux valeurs théoriques attendues, probablement en raison d'un bug dans le code ou d'un nombre insuffisant de points de discrétisation.

Cette étude a également souligné la flexibilité et la robustesse du schéma de Yee (FDTD) dans le traitement numérique des équations de Maxwell. Cependant, des recherches futures pourraient explorer des situations plus complexes comme la propagation des ondes dans des milieux non homogènes ou anisotropes, ainsi que l'intégration de conditions aux limites plus sophistiquées et la prise en compte des effets non linéaires.

Je tiens à exprimer ma profonde gratitude à ma tutrice de mémoire, Madame Claire Scheid, dont la contribution a été essentielle pour guider et encadrer ce projet de manière efficace. Merci, Madame Scheid, pour votre soutien constant et vos conseils inestimables tout au long de ce travail. Votre engagement dans la réussite de ce projet a été d'une aide précieuse.

Je souhaite également remercier le département de Master en Ingénierie Mathématique pour leur soutien continu et leur engagement envers l'éducation. Grâce à ce projet de mémoire, j'ai pu approfondir mes connaissances et explorer des concepts mathématiques passionnants, enrichissant ainsi mon parcours académique de manière significative.

V. Bibliographies

[1] Taflove, A., & Hagness, S. C. Computational Electrodynamics: The Finite-Difference Time-Domain Method (3rd ed.). ARTECH HOUSE. (2005).

[2] https://en.wikipedia.org/wiki/Maxwell%27s_equations

[3] Courant–Friedrichs–Lewy condition - Wikipedia

[4] https://www.docin.com/p-307155559.html

[5] https://zhuanlan.zhihu.com/p/423756965

[6] https://zhuanlan.zhihu.com/p/554198234

[7] https://zhuanlan.zhihu.com/p/396275312

[8] https://zhuanlan.zhihu.com/p/604040543

[9] https://zhuanlan.zhihu.com/p/676475895

[10] https://zhuanlan.zhihu.com/p/176192376

[11] https://zhuanlan.zhihu.com/p/542896007

[12] https://zhuanlan.zhihu.com/p/426033773

VI. Annexe

1D_Maxwell.py

```
import numpy as np
import matplotlib.pyplot as plt

def sol_exacte(t, x, y, omega, mu, e):

    c = 1/np.sqrt(mu*e)

    E = np.zeros((len(t), len(x)))
    H = np.zeros((len(t), len(x)))

    for n in range(len(t)):
        E[n, :] = np.sin(omega * x[:]) * np.sin(c * omega * t[n])

        H[n, :] = - 1/(mu*c) * np.cos(c * omega * (delta_t/2+t[n])) * np.cos(omega * y[:])
    return H, E


def sol_exacte(t, x, y, omega):

    delta_t = t[1] - t[0]

    E = np.zeros((len(t), len(x)))
    H = np.zeros((len(t), len(x)))

    for n in range(len(t)):
        E[n, :] = np.sin(omega * x[:]) * np.sin(omega * t[n])
        # pour H on a un decalage en temps de delta_t/2
        H[n, :] = - np.cos(omega * (delta_t/2+t[n])) * np.cos(omega * y[:])
    return H, E



def Maxwell1d(N, I, T, L, omega):

    delta_t = T/N  #le pas de discritisation temporelle
    delta_x = L/I  #le pas de discritisation spatiale
    coeff = delta_t/delta_x
```

```
    z = np.arange(0, L + 0.1 * delta_x, delta_x) # I+1 points de discritisation sur pour Ez sur [0,L]
    y = np.arange(delta_x/2, L+delta_x/2 + 0.1 * delta_x, delta_x) # I+1 points de discritisation pour Hy sur [delta_x/2,L+delta_x/2]
    t = np.arange(0, T + 0.1 * delta_t, delta_t) # N+1 points de discritisation temporelle sur [0,L]

    # forme des champs magnetique et eletrique

    Hy = np.zeros((N+1,I+1))
    Ez = np.zeros((N+1, I+1))

    # conditions initiales

    Hy[0, :] = - np.cos(omega * (delta_t/2+t[0])) * np.cos(omega * y[:])
    Ez[0, :] = np.sin(omega * z[:]) * np.sin(omega * t[0])

    # le schema: Ez[1, i] = Ez[0, i] + coeff * (Hy[0, i] - Hy[0, i-1]), pour i dans {0,...I-1}
    Ez[1, 1:-1] = Ez[0, 1:-1] + coeff * (Hy[0, 1:-1] - Hy[0, :-2])

    # condtions de bords pour temps indice=1
    Ez[1, 0] = 0
    Ez[1, -1] = 0

    # mise a jour des champs electrique et magnetique
    for n in range(1,N):

        # le schema: Hy[n, i] = Hy[n-1, i] + coeff * (Ez[n, i+1] - Ez[n, i]), pour i dans {0,...I-1}
        Hy[n, :-1] = Hy[n-1, :-1] + coeff * (Ez[n, 1:] - Ez[n, :-1])
        Hy[n, -1] = Hy[n-1, -1] + coeff * (Ez[n, 1] - Ez[n, -1])

        # le schema: Ez[n + 1, i] = Ez[n, i] + coeff * (Hy[n, i] - Hy[n, i-1]), pour i dans {0,...I-1}
        Ez[n + 1, 1:-1] = Ez[n, 1:-1] + coeff * (Hy[n, 1:-1] - Hy[n, :-2])

        # condtions de bords
        Ez[n + 1, 0] = 0
        Ez[n + 1, -1] = 0

    # mise a jour de Hy pour le dernier indice temporelle
    Hy[N, :-1] = Hy[N-1, :-1] + coeff * (Ez[N, 1:] - Ez[N, :-1])
    Hy[N, -1] = Hy[N-1, -1] + coeff * (Ez[N, 1] - Ez[N, -1])


    return Hy, Ez, t, z, y
```

```
k = 2
L = 5
T = 4

omega = k * np.pi / L

N = 1500 # on a N+1=1501 points de discritisation temporelle
I = 600 ## on a I+1=601 points de discritisation spatiale

delta_t = T /N
delta_x = L / I
# dans ce cas pour verifier la condition CFL, il suffit de prendre que delta_t / delta_x < 1
CFL = delta_t / delta_x
print(f"CFL = {delta_t/delta_x}")

Hy, Ez, t, x, y = Maxwell1d(N, I, T, L, omega)

H, E = sol_exacte(t, x, y, omega)



#graphe du champ electrique et du champ magnetique
plt.figure(num=1, figsize=(12,8))
t1 = 600
plt.subplot(211)
plt.title(f"champ electrique pour t={t1}")
plt.plot(x, Ez[t1,:], label='Fluide Electrique Yee (Ez)', color='g', linestyle = '--', lw = 2)
plt.plot(x, E[t1,:], label='Fluide Electrique Exact (E)', color='r', lw = 1)
plt.xlabel('z')
plt.ylabel('Ez')
plt.legend()


plt.subplot(212)
plt.title(f"champ magnetique t={t1}")
plt.plot(y, Hy[t1,:], label='Fluide Magnetique Yee (Hy)', color='g', linestyle = '--', lw =2)
plt.plot(y, H[t1,:], label='Fluide Magnetique Exact (H)', color='r', lw=1.5)
plt.xlabel('y')
plt.ylabel('Hy')
plt.legend()
```

```
plt.tight_layout()
plt.show()


plt.figure(num=2, figsize=(12,8))
t2 = 1500
plt.subplot(211)
plt.title(f"champ electrique pour t={t2}")
plt.plot(x, Ez[t2,:], label='Fluide Electrique Yee (Ez)', color='g', linestyle = '--', lw = 2)
plt.plot(x, E[t2,:], label='Fluide Electrique Exact (E)', color='r', lw = 1)
plt.xlabel('z')
plt.ylabel('Ez')
plt.legend()


plt.subplot(212)
plt.title(f"champ magnetique pour t={t2}")
plt.plot(y, Hy[t2,:], label='Fluide Magnetique Yee (Hy)', color='g', lw = 2, linestyle = '--')
plt.plot(y, H[t2,:], label='Fluide Magnetique Exact (H)', color='r', lw = 1)
plt.xlabel('y')
plt.ylabel('Hy')
plt.legend()


plt.tight_layout()
plt.show()

#%%  Animation de la variation du champ electrique et du champ magnetique

from matplotlib.animation import FuncAnimation

fig, ((ax1, ax3), (ax2, ax4)) = plt.subplots(2, 2, figsize=(12, 8))

def animate(n):
    ax1.clear()
    ax2.clear()
    ax3.clear()
    ax4.clear()

    ax1.set_title('Fluide Electrique par Yee (Ez) at t = {}*dt'.format(n))
    ax1.set_xlabel('En axe Z')
    ax1.set_ylabel('Ez')
    ax1.plot(Ez[n, :], color='g', label='Ez par Yee', lw=2.5, ls='--')
    ax1.plot(E[n, :], color='r', label='E exacte', lw =1)
```

```
    ax1.set_ylim([-1.1, 1.1])
    ax1.legend()

    ax2.set_title('Fluide Magnetique par Yee (Hy) at t = {}*dt'.format(n))
    ax2.set_xlabel('En axe Y')
    ax2.set_ylabel('Hy')
    ax2.plot(Hy[n, :], color='g', label='Hy par Yee', lw=2.5, ls='--')
    ax2.plot(H[n, :], color='r', label='H exacte', lw=1)
    ax2.set_ylim([-1.2, 1.2])
    #ax2.set_ylim([-0.1, 0.1])
    ax2.legend()

    ax3.set_title('Solution Electrique (E) at t = {}*dt'.format(n))
    ax3.set_xlabel('En axe Z')
    ax3.set_ylabel('E')
    ax3.plot(E[n, :], color='r', label='E')
    ax3.set_ylim([-1.1, 1.1])
    ax3.legend()


    ax4.set_title('Solution Magnetique (H) at t = {}*dt'.format(n))
    ax4.set_xlabel('En axe Y')
    ax4.set_ylabel('H')
    ax4.plot(H[n, :], color='c', label='H')
    ax4.set_ylim([-1.2, 1.2])
    #ax4.set_ylim([-0.1, 0.1])
    ax4.legend()

    plt.tight_layout()

ani = FuncAnimation(fig, animate, frames = range(0, N, 25), interval=50)

plt.show()


#%%  Calcul de l'ordre de convergence
k = 4
T = 3
L = 3
omega = k * np.pi / L

I_list  = np.arange(800, 2002, 200) # un ensembre de nombre de discritisation spatiale
N_list = 3 * (T * I_list // L) # # un ensembre de nombre de discritisation temporelle, pour la
condition de CFL: delta_t/delta_x= 1/3
```

```
err_H = np.zeros(len(l_list))
err_E = np.zeros(len(l_list))

for i, N in enumerate(N_list):

    err_HH = np.zeros(N)
    err_EE = np.zeros(N)

    # mise a jour du pas spatial et du pas temporel
    delta_x = L / l_list[i]
    delta_t = T / N

    # mise a jour de la solution numerique et la solution exacte
    Hy, Ez, t, z, y = Maxwell1d(N, l_list[i], T, L, omega)
    H, E = sol_exacte(t, z, y, omega)

    for j in range(N):

        # L'effet du temps sur l'erreur
        err_HH[j] = np.linalg.norm(H[j, :] - Hy[j, :], ord=2) *delta_x**(1/2)
        #err_HH[j] = np.linalg.norm(H[j, :] - Hy[j, :], ord=np.inf)
        err_EE[j] = np.linalg.norm(E[j, :] - Ez[j, :], ord=2) *delta_x**(1/2)
        #err_EE[j] = np.linalg.norm(E[j, :] - Ez[j, :], ord=np.inf)

    #L'effet de l'axe x du temps et de l'espace sur l'erreur
    err_H[i] = np.linalg.norm(err_HH, ord=2) *delta_t**(1/2)
    err_E[i] = np.linalg.norm(err_EE, ord=2) *delta_t**(1/2)
    #err_H[i] = np.linalg.norm(err_HH, ord=np.inf)
    #err_E[i] = np.linalg.norm(err_EE, ord=np.inf)

#la norme 2 marche mieux que la norme infinie

#print(err_E)
#print(err_H)

h = L / l_list  # une liste du pas de discritisation spatialle
print(h)
H_ordre = np.polyfit(np.log(h), np.log(err_H), 1)[0] # calculer la pente de la courbe de log(h)
et log(err_H)
E_ordre = np.polyfit(np.log(h), np.log(err_E), 1)[0] # calculer la pente de la courbe de log(h)
et log(err_E)
print(f"Ordre de convergence spatial pour H: {H_ordre}")
```

```
print(f"Ordre de convergence spatial pour E: {E_ordre}")

print(f'max err_H = {max(err_H)}, max err_E = {max(err_E)}')

# figure d'erreur
plt.figure()
plt.plot(h, err_E, label="erreur de E", color='r')
plt.plot(h, err_H, label="erreur de H", color='b', ls='--')
plt.title("L'erreur du schema")
plt.xlabel('delta_x')
plt.ylabel('erreur')
plt.legend()
plt.show()


# figure d'ordre de convergence
plt.figure()
plt.plot(np.log(h), np.log(err_E), label=f"ordre d'erreur de E = {round(E_ordre,2)}", color='r')
plt.plot(np.log(h), np.log(err_H), label=f"ordre d'erreur de H = {round(H_ordre,2)}", color='b',
ls='--')
plt.title("L'ordre de convergence")
plt.xlabel('log(delta_x)')
plt.ylabel('log(erreur)')
plt.legend()
plt.show()

cfl = (T/N_list[:]) / (L/l_list[:])
print("cfl = " + str(cfl))
```

1D_generalisation.py

```
import numpy as np
import matplotlib.pyplot as plt


def sol_exacte(t, x, y, omega, mu, e):

    c = 1/np.sqrt(mu*e)

    E = np.zeros((len(t), len(x)))
    H = np.zeros((len(t), len(x)))

    for n in range(len(t)):
        E[n, :] = np.sin(omega * x[:]) * np.sin(c * omega * t[n])

        H[n, :] = - 1/(mu*c) * np.cos(c * omega * (delta_t/2+t[n])) * np.cos(omega * y[:])
    return H, E



def Maxwell1d(N, I, T, L, omega, mu, e):

    c = 1/np.sqrt(mu*e)

    delta_t = T/N  #le pas de discritisation temporelle
    delta_x = L/I  #le pas de discritisation spatiale
    coeff = delta_t/delta_x

    z = np.arange(0, L + 0.1 * delta_x, delta_x) # I+1 points de discritisation sur pour Ez sur [0,L]
    y = np.arange(delta_x/2, L+delta_x/2 + 0.1 * delta_x, delta_x) # I+1 points de discritisation pour Hy sur [delta_x/2,L+delta_x/2]
    t = np.arange(0, T + 0.1 * delta_t, delta_t) # N+1 points de discritisation temporelle sur [0,L]

    # forme des champs magnetique et eletrique

    Hy = np.zeros((N+1,I+1))
    Ez = np.zeros((N+1, I+1))

    # conditions initiales
```

```
    Hy[0, :] = - 1/(mu*c) * np.cos(c * omega * (delta_t/2+t[0])) * np.cos(omega * y[:])
    Ez[0, :] = np.sin(omega * z[:]) * np.sin(c * omega * t[0])

    Ez[1, 1:-1] = Ez[0, 1:-1] + (coeff/e) * (Hy[0, 1:-1] - Hy[0, :-2])
    # conditions de bords
    Ez[1, 0] = 0
    Ez[1, -1] = 0

    for n in range(1,N):

        Hy[n, :-1] = Hy[n-1, :-1] + (coeff/mu) * (Ez[n, 1:] - Ez[n, :-1])
        Hy[n, -1] = Hy[n-1, -1] + (coeff/mu) * (Ez[n, 1] - Ez[n, -1])

        Ez[n + 1, 1:-1] = Ez[n, 1:-1] + (coeff/e) * (Hy[n, 1:-1] - Hy[n, :-2])

        # conditions de bords
        Ez[n + 1, 0] = 0
        Ez[n + 1, -1] = 0

    Hy[N, :-1] = Hy[N-1, :-1] + (coeff/mu) * (Ez[N, 1:] - Ez[N, :-1])
    Hy[N, -1] = Hy[N-1, -1] + (coeff/mu) * (Ez[N, 1] - Ez[N, -1])
    return Hy, Ez, t, z, y

k = 4
L = 3
T = 3
mu = 2
e = 1/3

omega = k * np.pi / L

N = 1500
I = 500
delta_t = T /N
delta_x = L / I
CFL = delta_t / (mu * e * delta_x )
print(f"CFL = {CFL}")


Hy, Ez, t, z, y = Maxwell1d(N, I, T, L, omega, mu, e)
H, E = sol_exacte(t, z, y, omega, mu, e)


#graphe du champ electrique et du champ magnetique
```

```
plt.figure(num=1, figsize=(12,8))
t1 = 600
plt.subplot(211)
plt.title(f"champ electrique pour t={t1}")
plt.plot(z, Ez[t1,:], label='Fluide Electrique Yee (Ez)', color='g', lw = 2, linestyle = '--')#, marker='+')
plt.plot(z, E[t1,:], label='Fluide Electrique exact (E)', color='r', lw = 1)
plt.xlabel('z')
plt.ylabel('Ez')
plt.legend()


plt.subplot(212)
plt.title(f"champ magnetique t={t1}")
plt.plot(y, Hy[t1,:], label='Fluide Magnetique Yee (Hy)', color='g', linestyle = '--', lw =2)#, marker='+')
plt.plot(y, H[t1,:], label='Fluide Magnetique exact (H)', color='r', lw=1)
plt.xlabel('y')
plt.ylabel('Hy')
plt.legend()

plt.tight_layout()
plt.show()


plt.figure(num=2, figsize=(12,8))
t2 = 1500
plt.subplot(211)
plt.title(f"champ electrique pour t={t2}")
plt.plot(z, Ez[t2,:], label='Fluide Electrique Yee (Ez)', color='g', lw = 2, linestyle = '--')#, marker='+')
plt.plot(z, E[t2,:], label='Fluide Electrique exact (E)', color='r', lw = 1)
plt.xlabel('z')
plt.ylabel('Ez')
plt.legend()


plt.subplot(212)
plt.title(f"champ magnetique t={t2}")
plt.plot(y, Hy[t2,:], label='Fluide Magnetique Yee (Hy)', color='g', lw = 2, linestyle = '--')#, marker='+')
plt.plot(y, H[t2,:], label='Fluide Magnetique exact (H)', color='r', lw = 1)
plt.xlabel('y')
plt.ylabel('Hy')
```

```
plt.legend()


plt.tight_layout()
plt.show()


#%%  Animation de la variation du champ electrique et du champ magnetique

from matplotlib.animation import FuncAnimation

fig, ((ax1, ax3), (ax2, ax4)) = plt.subplots(2, 2, figsize=(12, 8))

def animate(n):
    ax1.clear()
    ax2.clear()
    ax3.clear()
    ax4.clear()

    ax1.set_title('Fluide Electrique par Yee (Ez) at t = {}*dt'.format(n))
    ax1.set_xlabel('En axe Z')
    ax1.set_ylabel('Ez')
    ax1.plot(Ez[n, :], color='g', label='Ez par Yee', lw=2.5, ls='--')
    ax1.plot(E[n, :], color='r', label='E exacte', lw =1)
    ax1.set_ylim([-1.1, 1.1])
    ax1.legend()

    ax2.set_title('Fluide Magnetique par Yee (Hy) at t = {}*dt'.format(n))
    ax2.set_xlabel('En axe Y')
    ax2.set_ylabel('Hy')
    ax2.plot(Hy[n, :], color='b', label='Hy par Yee', lw=2.5, ls='--')
    ax2.plot(H[n, :], color='r', label='H exacte', lw=1)
    ax2.set_ylim([-1.2, 1.2])
    #ax2.set_ylim([-0.1, 0.1])
    ax2.legend()

    ax3.set_title('Solution Electrique (E) at t = {}*dt'.format(n))
    ax3.set_xlabel('En axe Z')
    ax3.set_ylabel('E')
    ax3.plot(E[n, :], color='r', label='E')
    ax3.set_ylim([-1.1, 1.1])
    ax3.legend()
```

```
    ax4.set_title('Solution Magnetique (H) at t = {}*dt'.format(n))
    ax4.set_xlabel('En axe Y')
    ax4.set_ylabel('H')
    ax4.plot(H[n, :], color='c', label='H')
    ax4.set_ylim([-1.2, 1.2])
    #ax4.set_ylim([-0.1, 0.1])
    ax4.legend()

    plt.tight_layout()

ani = FuncAnimation(fig, animate, frames = range(0, N, 25), interval=50)

plt.show()


#%%  Calcul de l'ordre de convergence
# sous CFL = 1/2
k = 4
T = 3
L = 3
omega = k * np.pi / L

l_list  = np.arange(800, 2002, 200)
N_list = 2 * (T * l_list // L)


err_H = np.zeros(len(l_list))
err_E = np.zeros(len(l_list))

for i, N in enumerate(N_list):

    err_HH = np.zeros(N)
    err_EE = np.zeros(N)

    delta_x = L / l_list[i]
    delta_t = T / N
    #print(delta_t/delta_x)
    Hy, Ez, t, z, y = Maxwell1d(N, l_list[i], T, L, omega, mu, e)
    H, E = sol_exacte(t, z, y, omega, mu, e)

    for j in range(N):
        #L'effet du temps sur l'erreur
        err_HH[j] = np.linalg.norm(H[j, :] - Hy[j, :], ord=2) *delta_x**(1/2) #norme 2
        err_EE[j] = np.linalg.norm(E[j, :] - Ez[j, :], ord=2) *delta_x**(1/2) #norme infinie
```

```
        #err_HH[j] = np.linalg.norm(H[j, :] - Hy[j, :], ord=np.inf)
        #err_EE[j] = np.linalg.norm(E[j, :] - Ez[j, :], ord=np.inf)

    #L'effet de l'axe x du temps et de l'espace sur l'erreur
    err_H[i] = np.linalg.norm(err_HH, ord=2) *delta_t**(1/2)
    err_E[i] = np.linalg.norm(err_EE, ord=2) *delta_t**(1/2)
    #err_H[i] = np.linalg.norm(err_HH, ord=np.inf)
    #err_E[i] = np.linalg.norm(err_EE, ord=np.inf)
#la norme 2 marche mieux que la norme infinie

h = L / l_list
print(h)
H_ordre = np.polyfit(np.log(h), np.log(err_H), 1)[0]
E_ordre = np.polyfit(np.log(h), np.log(err_E), 1)[0]
print(f"Ordre de convergence spatial pour H: {H_ordre}")
print(f"Ordre de convergence spatial pour E: {E_ordre}")

print(f'max err_H = {max(err_H)}, max err_E = {max(err_E)}')

plt.figure()
plt.plot(h, err_E, label="erreur de E", color='r')
plt.plot(h, err_H, label="erreur de H", color='b')
plt.title("L'erreur du schema")
plt.xlabel('delta_x')
plt.ylabel('erreur')
plt.legend()
plt.show()

plt.figure()
plt.plot(np.log(h), np.log(err_E), label=f"ordre d'erreur de E = {round(E_ordre,2)}", color='r')
plt.plot(np.log(h), np.log(err_H), label=f"ordre d'erreur de H = {round(H_ordre,2)}", color='b')
plt.title("L'ordre de convergence")
plt.xlabel('log(delta_x)')
plt.ylabel('log(erreur)')
plt.legend()
plt.show()

cfl = (T/N_list[:]) / (L/l_list[:])
print("cfl = " + str(cfl))
```

1D_milieux_diff.py

```
import numpy as np
import matplotlib.pyplot as plt


def sol_exacte(t, x, y, omega, mu, e):
# mu est la permittivite magnetique, e est la permitivite electrique

    c = 1/np.sqrt(mu*e)

    E = np.zeros((len(t), len(x)))
    H = np.zeros((len(t), len(y)))

    for n in range(len(t)):
        E[n, :] = np.sin(omega * x[:]) * np.sin(c * omega * t[n])

        H[n, :] = - 1/(mu*c) * np.cos(c * omega * (delta_t/2+t[n])) * np.cos(omega * y[:])
    return H, E



def Maxwell1d(N, l, T, l, L, omega, mu, e):

    c = 1/np.sqrt(mu*e)

    delta_t = T/N  #le pas de discritisation temporelle
    delta_x = (L-l)/l  #le pas de discritisation spatiale
    coeff = delta_t/delta_x

    # l+1 points de discritisation sur pour Ez sur [0,L]
    z = np.arange(l, L + 0.1 * delta_x, delta_x)
    # l+1 points de discritisation pour Hy sur [delta_x/2,L+delta_x/2]
    y = np.arange(l + delta_x/2, L + 0.1 * delta_x, delta_x)
    # l+1 points de discritisation pour Hy sur [delta_x/2,L+delta_x/2]
    t = np.arange(0, T + 0.1 * delta_t, delta_t)

    # forme des champs magnetique et eletrique

    Hy = np.zeros((N+1,l))
    Ez = np.zeros((N+1, l+1))

    # conditions initiales
```

```
    Hy[0, :] = - 1/(mu*c) * np.cos(c * omega * (delta_t/2+t[0])) * np.cos(omega * y[:])
    Ez[0, :] = np.sin(omega * z[:]) * np.sin(c * omega * t[0])

    Ez[1, 1:-1] = Ez[0, 1:-1] + (coeff/e) * (Hy[0, 1:] - Hy[0, :-1])
    # conditions de bords
    Ez[1, 0] = 0
    Ez[1, -1] = 0

    for n in range(1,N):

        Hy[n, :-1] = Hy[n-1, :-1] + (coeff/mu) * (Ez[n, 1:-1] - Ez[n, :-2])
        Hy[n, -1] = Hy[n-1, -1] + (coeff/mu) * (Ez[n, -1] - Ez[n, -2])

        Ez[n + 1, 1:-1] = Ez[n, 1:-1] + (coeff/e) * (Hy[n, 1:] - Hy[n, :-1])

        # conditions de bords
        Ez[n + 1, 0] = 0
        Ez[n + 1, -1] = 0

    Hy[N, :-1] = Hy[N-1, :-1] + (coeff/mu) * (Ez[N, 1:-1] - Ez[N, :-2])
    Hy[N, -1] = Hy[N-1, -1] + (coeff/mu) * (Ez[N, 1] - Ez[N, -1])
    return Hy, Ez, t, z, y

N = 1500
k = 6
omega = k * np.pi / 3  # pour assurer que E et H sont 1-periodique
T = 1.5

delta_t = T /N

# entre 0 et L/3
I1 = 200
I1= 0
L1 = 1

mu1 = 2
e1 = 1/3

Hy1, Ez1, t1, z1, y1 = Maxwell1d(N, I1, T, I1, L1, omega, mu1, e1)
H1, E1 = sol_exacte(t1, z1, y1, omega, mu1, e1)

delta_x1 = L1 / I1
```

```
CFL1 = delta_t / (mu1* e1 * delta_x1)
print(f"CFL1 = {CFL1}")

# entre L/3 et 2L/3
I2 = 300
I2 =1
L2 = 2

mu2 = 1
e2 = 1/2

Hy2, Ez2, t2, z2, y2 = Maxwell1d(N, I2, T, I2, L2, omega, mu2, e2)
H2, E2 = sol_exacte(t2, z2, y2, omega, mu2, e2)

delta_x2 = L2 / I2
CFL2 = delta_t / (mu2 * e2 * delta_x2)
print(f"CFL2 = {CFL2}")

# entre 2L/3 et L
I3 = 400
I3 =2
L3 = 3

mu3 = 2
e3 = 1/3

Hy3, Ez3, t3, z3, y3 = Maxwell1d(N, I3, T, I3, L3, omega, mu3, e3)
H3, E3 = sol_exacte(t3, z3, y3, omega, mu3, e3)

delta_x3 = L3 / I3
CFL3 = delta_t / (mu3 * e3 * delta_x3)
print(f"CFL3 = {CFL3}")

#graphe du champ electrique et du champ magnetique
plt.figure(num=1, figsize=(12,8))
t1 = 1500
plt.subplot(211)
plt.title(f"champ electrique pour t={t1}")

# entre 0 et L/3
plt.plot(z1, Ez1[t1,:], label='Fluide Electrique Yee (Ez)', color='g', lw = 2, linestyle = '--')#,
marker='+')
plt.plot(z1, E1[t1,:], label='Fluide Electrique exact (E)', color='r', lw = 1)
# entre L/3 et 2L/3
```

```
plt.plot(z2, Ez2[t1,:], color='g', lw = 2, linestyle = '--')
plt.plot(z2, E2[t1,:], color='r', lw = 1)
# entre 2L/3 et L
plt.plot(z3, Ez3[t1,:], color='g', lw = 2, linestyle = '--')
plt.plot(z3, E3[t1,:], color='r', lw = 1)

plt.xlabel('z')
plt.ylabel('Ez')
plt.legend()


plt.subplot(212)
plt.title(f"champ magnetique t={t1}")

# entre 0 et L/3
plt.plot(y1, Hy1[t1,:], label='Fluide Magnetique Yee (Hy)', color='g', linestyle = '--', lw =2)#,
marker='+')
plt.plot(y1, H1[t1,:], label='Fluide Magnetique exact (H)', color='r', lw=1)
# entre L/3 et 2L/3
plt.plot(y2, Hy2[t1,:], color='g', linestyle = '--', lw =2)
plt.plot(y2, H2[t1,:], color='r', lw=1)
# entre 2L/3 et L
plt.plot(y3, Hy3[t1,:], color='g', linestyle = '--', lw =2)
plt.plot(y3, H3[t1,:], color='r', lw=1)

plt.xlabel('y')
plt.ylabel('Hy')
plt.legend()

plt.tight_layout()
plt.show()

#%%  Animation de la variation du champ electrique et du champ magnetique
from matplotlib.animation import FuncAnimation
fig, ((ax1, ax3), (ax2, ax4)) = plt.subplots(2, 2, figsize=(12, 8))

def animate(n):
    ax1.clear()
    ax2.clear()
    ax3.clear()
    ax4.clear()

    ax1.set_title('Fluide Electrique par Yee (Ez) at t = {}*dt'.format(n))
    ax1.set_xlabel('En axe Z')
```

```
    ax1.set_ylabel('Ez')
    ax1.plot(z1, Ez1[n, :], color='g', label='Ez par Yee', lw=2.5, ls='--')
    ax1.plot(z1, E1[n, :], color='r', label='E exacte', lw =1)
    ax1.plot(z2, Ez2[n, :], color='g', lw=2.5, ls='--')
    ax1.plot(z2, E2[n, :], color='r', lw =1)
    ax1.plot(z3, Ez3[n, :], color='g', lw=2.5, ls='--')
    ax1.plot(z3, E3[n, :], color='r', lw =1)
    ax1.set_ylim(-1.1, 1.1)
    ax1.legend()

    ax2.set_title('Fluide Magnetique par Yee (Hy) at t = {}*dt'.format(n))
    ax2.set_xlabel('En axe Y')
    ax2.set_ylabel('Hy')
    ax2.plot(y1, Hy1[n, :], color='b', label='Hy par Yee', lw=2.5, ls='--')
    ax2.plot(y1, H1[n, :], color='r', label='H exacte', lw=1)
    ax2.plot(y2, Hy2[n, :], color='b', lw=2.5, ls='--')
    ax2.plot(y2, H2[n, :], color='r', lw=1)
    ax2.plot(y3, Hy3[n, :], color='b', lw=2.5, ls='--')
    ax2.plot(y3, H3[n, :], color='r', lw=1)
    ax2.set_ylim(-1.1, 1.1)
    ax2.legend()

    ax3.set_title('Solution Electrique (E) at t = {}*dt'.format(n))
    ax3.set_xlabel('En axe Z')
    ax3.set_ylabel('E')
    ax3.plot(z1, E1[n, :], color='r', label='E')
    ax3.plot(z2, E2[n, :], color='r')
    ax3.plot(z3, E3[n, :], color='r')
    ax3.set_ylim(-1.1, 1.1)
    ax3.legend()

    ax4.set_title('Solution Magnetique (H) at t = {}*dt'.format(n))
    ax4.set_xlabel('En axe Y')
    ax4.set_ylabel('H')
    ax4.plot(y1, H1[n, :], color='c', label='H')
    ax4.plot(y2, H2[n, :], color='c')
    ax4.plot(y3, H3[n, :], color='c')
    ax4.set_ylim(-1.1, 1.1)
    ax4.legend()
    plt.tight_layout()

ani = FuncAnimation(fig, animate, frames = range(0, N, 5), interval=50)
plt.show()
```

2D_Maxwell.py

```
import numpy as np

from numpy import cos, sin, pi

import matplotlib.pyplot as plt


def Maxwell_2d(N, I, T, L, omega):


    delta_t = T/N

    delta_x = L/I

    delta_y = L/I

    coeff_x = delta_t / delta_x

    coeff_y = delta_t / delta_y


    x = np.arange(0, L + 0.1 * delta_x, delta_x)

    y = np.arange(0, L + 0.1 * delta_x, delta_x)

    t = np.arange(0, T + 0.1 * delta_t, delta_t)


    # forme des champs magnetique et eletrique

    Hx = np.zeros((N+1, I+1, I+1))

    Hy = np.zeros((N+1, I+1, I+1))

    Ez = np.zeros((N+1, I+1, I+1))


    # conditons initiales

    for i in range(I+1):

        Ez[0, i, :] = sin(omega*x[i]) * sin(omega*y[:]) * sin(np.sqrt(2)*omega* t[0])
```

```
Hx[0, i, :] = 1/np.sqrt(2) * sin(omega* x[i]) * cos(omega* (y[:] + delta_y/2)) * cos(np.sqrt(2)*omega*(t[0]+ delta_t/2))

Hy[0, i, :] = -1/np.sqrt(2) * cos(omega* (x[i] + delta_x/2)) * sin(omega* y[:]) * cos(np.sqrt(2)*omega*(t[0]+ delta_t/2))



for n in range(N):



Ez[n+1, 1:-1, 1:-1] = Ez[n, 1:-1, 1:-1] + coeff_x * (Hy[n, 1:-1 , 1:-1] - Hy[n, 0:-2, 1:-1]) - coeff_y * (Hx[n, 1:-1, 1:-1] - Hx[n, 1:-1, 0:-2])

Hx[n+1, 0:-1, 0:-1] = Hx[n, 0:-1, 0:-1] - coeff_y * (Ez[n+1, 0:-1, 1: ] - Ez[n+1, 0:-1, 0:-1])

Hy[n+1, 0:-1, 0:-1] = Hy[n, 0:-1, 0:-1] + coeff_x * (Ez[n+1, 1: , 0:-1] - Ez[n+1, 0:-1, 0:-1])



Ez[n+1, 0, :] = 0

Ez[n+1, -1, :] = 0

Ez[n+1, :, 0] = 0

Ez[n+1, :, -1] = 0



Hx[n+1, -1, :] = 0

Hx[n+1, :, -1] = 1/np.sqrt(2) * sin(omega* (x[:] +delta_x/2)) * cos(omega*(y[-1] + delta_y/2)) * cos(np.sqrt(2)*omega* (t[n+1]+ delta_t/2))



Hy[n+1, :, -1] = 0

Hy[n+1, -1, :] = -1/np.sqrt(2) * cos(omega* (x[-1] +delta_x/2)) * sin(omega*(y[:] + delta_y/2)) * cos(np.sqrt(2)*omega* (t[n+1]+ delta_t/2))
```

```
    return t, x, y, Ez, Hx, Hy



def solution(t, x, y, omega):
    delta_t = t[1]-t[0]
    Ez_s = np.zeros((len(t), len(x), len(y)))
    Hx_s = np.zeros((len(t), len(x), len(y)))
    Hy_s = np.zeros((len(t), len(x), len(y)))

    for n in range(len(t)):
        for i in range(len(x)):
            Ez_s[n,i,:] = sin(omega*x[i]) * sin(omega*y[:]) * sin(np.sqrt(2)*omega* t[n])
            Hx_s[n,i,:] = 1/np.sqrt(2) * sin(omega* x[i]) * cos(omega* (y[:] + delta_y/2)) * cos(np.sqrt(2)*omega* (t[n]+ delta_t/2))
            Hy_s[n,i,:] = -1/np.sqrt(2) * cos(omega* (x[i] +delta_x/2)) * sin(omega* y[:]) * cos(np.sqrt(2)*omega* (t[n]+ delta_t/2))

    return Ez_s, Hx_s, Hy_s

k = 4  # k pair pour que la solution soit periodique
L = 4
T = 3

N = 700
I = 250
delta_t = T/N
```

```
delta_x = L/I

delta_y = L/I

omega = k * pi / L


CFL = delta_t/delta_x + delta_t/delta_y


t, x, y, Ez, Hx, Hy = Maxwell_2d(N, I, T, L, omega)

Ez_s, Hx_s, Hy_s = solution(t, x, y, omega)



print(f"condition CFL = {CFL}")


#%%


temps_indice = 700


# variation du champ electrique et du champ magnetique sur l'axe y

x_indice = 100


plt.figure(figsize=(18, 12))


plt.subplot(2, 3, 1)

plt.plot(y, Ez[temps_indice, x_indice, :], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')

plt.plot(y, Ez_s[temps_indice, x_indice, :], label='sol', color = 'r', lw =0.8)

plt.title('Ez')
```

```
plt.xlabel('Y axis')

plt.ylabel('fluide electrique')

plt.grid(True)

plt.legend()

#plt.ylim([-0.3, 0.3])


plt.subplot(2, 3, 2)

plt.plot(y, Hx[temps_indice, x_indice, :], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')

plt.plot(y, Hx_s[temps_indice, x_indice, :], label='sol', color = 'r', lw =0.8)

plt.title('Hx')

plt.xlabel('Y axis')

plt.ylabel('fuide magnetique')

plt.grid(True)

plt.legend()

#plt.ylim([-0.45, 0.45])


plt.subplot(2, 3, 3)

plt.plot(y, Hy[temps_indice, x_indice, :], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')

plt.plot(y, Hy_s[temps_indice, x_indice, :], label='sol', color = 'r', lw =0.8)

plt.title('Hy')

plt.xlabel('Y axis')

plt.ylabel('fuide magnetique')

plt.grid(True)

plt.legend()
```

```
plt.subplot(2, 3, 4)
plt.plot(y, Ez_s[temps_indice, x_indice, :], color = 'r', label='sol')
plt.title('Ez_s')
plt.xlabel('Y axis')
plt.ylabel('fuide electrique')
plt.grid(True)
plt.legend()

plt.subplot(2, 3, 5)
plt.plot(y, Hx_s[temps_indice, x_indice, :], color = 'r', label='sol')
plt.title('Hx_s')
plt.xlabel('Y axis')
plt.ylabel('fuide magnetique')
plt.grid(True)
plt.legend()

plt.subplot(2, 3, 6)
plt.plot(y, Hy_s[temps_indice, x_indice, :], color = 'r', label='sol')
plt.title('Hy_s')
plt.xlabel('Y axis')
plt.ylabel('fuide magnetique')
plt.grid(True)
plt.legend()

plt.tight_layout()
```

```
plt.show()



# variation du champ electrique et du champ magnetique sur l'axe x

y_indice = 100


plt.figure(figsize=(18, 12))


plt.subplot(2, 3, 1)

plt.plot(x, Ez[temps_indice, :, y_indice], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')

plt.plot(x, Ez_s[temps_indice, :, y_indice], label='sol', color = 'r', lw =0.8)

plt.title('Ez')

plt.xlabel('X axis')

plt.ylabel('fuide electrique')

plt.grid(True)

plt.legend()

#plt.ylim([-0.3, 0.3])


plt.subplot(2, 3, 2)

plt.plot(x, Hx[temps_indice, :, y_indice], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')

plt.plot(x, Hx_s[temps_indice, :, y_indice], label='sol', color = 'r', lw =0.8)

plt.title('Hx')

plt.xlabel('X axis')

plt.ylabel('fuide magnetique')

plt.grid(True)
```

```
plt.legend()
#plt.ylim([-0.45, 0.45])

plt.subplot(2, 3, 3)
plt.plot(x, Hy[temps_indice, :, y_indice], label='Yee', color = 'g', lw =2, ls= '--')#, marker='+')
plt.plot(x, Hy_s[temps_indice, :, y_indice], label='sol', color = 'r', lw =0.8)
plt.title('Hy')
plt.xlabel('X axis')
plt.ylabel('fuide magnetique')
plt.grid(True)
plt.legend()

plt.subplot(2, 3, 4)
plt.plot(x, Ez_s[temps_indice, :, y_indice], color = 'r', label='sol')
plt.title('Ez_s')
plt.xlabel('X axis')
plt.ylabel('fuide electrique')
plt.grid(True)
plt.legend()

plt.subplot(2, 3, 5)
plt.plot(x, Hx_s[temps_indice, :, y_indice], color = 'r', label='sol')
plt.title('Hx_s')
plt.xlabel('X axis')
plt.ylabel('fuide magnetique')
```

```
plt.grid(True)

plt.legend()


plt.subplot(2, 3, 6)

plt.plot(x, Hy_s[temps_indice, :, y_indice], color = 'r', label='sol')

plt.title('Hy_s')

plt.xlabel('X axis')

plt.ylabel('fuide magnetique')

plt.grid(True)

plt.legend()


plt.tight_layout()

plt.show()


#%%  graphes du champ electrique et du champ magnetique


temps_indice = 700


plt.figure(figsize=(18, 12))


plt.subplot(2, 3, 1)

plt.contourf(x, y, Ez[temps_indice, :, :], cmap='magma')

plt.title('Ez')

plt.colorbar(label='Ez')

plt.xlabel('X axis')
```

```
plt.ylabel('Y axis')


plt.subplot(2, 3, 2)
plt.contourf(x, y, Hx[temps_indice, :, :], cmap='magma')
plt.title('Hx')
plt.colorbar(label='Hx')
plt.xlabel('X axis')
plt.ylabel('Y axis')


plt.subplot(2, 3, 3)
plt.contourf(x, y, Hy[temps_indice, :, :], cmap='magma')
plt.title('Hy')
plt.colorbar(label='Hy')
plt.xlabel('X axis')
plt.ylabel('Y axis')


plt.subplot(2, 3, 4)
plt.contourf(x, y, Ez_s[temps_indice, :, :], cmap='magma')
plt.title('Ez_s')
plt.colorbar(label='Ez_s')
plt.xlabel('X axis')
plt.ylabel('Y axis')


plt.subplot(2, 3, 5)
plt.contourf(x, y, Hx_s[temps_indice, :, :], cmap='magma')
```

```
plt.title('Hx_s')
plt.colorbar(label='Hx_s')
plt.xlabel('X axis')
plt.ylabel('Y axis')

plt.subplot(2, 3, 6)
plt.contourf(x, y, Hy_s[temps_indice, :, :], cmap='magma')
plt.title('Hy_s')
plt.colorbar(label='Hy_s')
plt.xlabel('X axis')
plt.ylabel('Y axis')

plt.tight_layout()
plt.show()

#%%  Animation de la variation du champ electrique et du champ magnetique
from matplotlib.animation import FuncAnimation

fig, axs = plt.subplots(2, 3, figsize=(18, 12))

def update(frame):
    axs[0, 0].clear()
    axs[0, 0].contourf(x, y, Ez[frame, :, :], cmap='magma')
    axs[0, 0].set_title('Ez at time = {:.2f}'.format(t[frame]))
    axs[0, 0].set_xlabel('X axis')
```

```
axs[0, 0].set_ylabel('Y axis')


axs[0, 1].clear()

axs[0, 1].contourf(x, y, Hx[frame, :, :], cmap='magma')

axs[0, 1].set_title('Hx at time = {:.2f}'.format(t[frame]))

axs[0, 1].set_xlabel('X axis')

axs[0, 1].set_ylabel('Y axis')


axs[0, 2].clear()

axs[0, 2].contourf(x, y, Hy[frame, :, :], cmap='magma')

axs[0, 2].set_title('Hy at time = {:.2f}'.format(t[frame]))

axs[0, 2].set_xlabel('X axis')

axs[0, 2].set_ylabel('Y axis')


axs[1, 0].clear()

axs[1, 0].contourf(x, y, Ez_s[frame, :, :], cmap='magma')

axs[1, 0].set_title('Ez_s at time = {:.2f}'.format(t[frame]))

axs[1, 0].set_xlabel('X axis')

axs[1, 0].set_ylabel('Y axis')


axs[1, 1].clear()

axs[1, 1].contourf(x, y, Hx_s[frame, :, :], cmap='magma')

axs[1, 1].set_title('Hx_s at time = {:.2f}'.format(t[frame]))

axs[1, 1].set_xlabel('X axis')

axs[1, 1].set_ylabel('Y axis')
```

```
    axs[1, 2].clear()
    axs[1, 2].contourf(x, y, Hy_s[frame, :, :], cmap='magma')
    axs[1, 2].set_title('Hy_s at time = {:.2f}'.format(t[frame]))
    axs[1, 2].set_xlabel('X axis')
    axs[1, 2].set_ylabel('Y axis')
print(len(t))

ani = FuncAnimation(fig, update, frames=50, interval=100)
plt.show()



#%%  Calcul de l'ordre de convergence


# un ensembre de nombre de discritisation spatiale
I_list  = np.arange(250, 280, 10)
# un ensembre de nombre de discritisation temporelle, pour la condition de CFL: delta_t/delta_x <= 1
N_list = 3 * (T * I_list // L)



err_Hx = np.zeros(len(I_list))
err_Hy = np.zeros(len(I_list))
err_Ez = np.zeros(len(I_list))
```

```
for i, I in enumerate(I_list):

    err_xxx = np.zeros((N, I)) # erreur de Hx sur l'axe y
    err_yyy = np.zeros((N, I)) # erreur de Hy sur l'axe y
    err_zzz = np.zeros((N, I)) # erreur de Ez sur l'axe y

    err_xx = np.zeros(N) # erreur de Hx sur l'axe x
    err_yy = np.zeros(N) # erreur de Hy sur l'axe x
    err_zz = np.zeros(N) # erreur de Ez sur l'axe x

    delta_x = L / I
    delta_t = T / N_list[i]
    print("CFL =",delta_t/delta_x + delta_t/delta_y)

    t, x, y, Ez, Hx, Hy = Maxwell_2d(N_list[i], I, T, L, omega)
    Ez_s, Hx_s, Hy_s = solution(t, x, y, omega)

    for n in range(N_list[i]):
        for j in range(I):
            # L'effet de l'axe y sur l'erreur
            err_xxx[n, j] = np.linalg.norm(Hx_s[n, j, :] - Hx[n, j, :], ord=2) *delta_x**(1/2)
            err_yyy[n, j] = np.linalg.norm(Hy_s[n, j, :] - Hy[n, j, :], ord=2) *delta_x**(1/2)
            err_zzz[n, j] = np.linalg.norm(Ez_s[n, j, :] - Ez[n, j, :], ord=2) *delta_x**(1/2)

        # L'effet de l'axe y du temps et de l'espace sur l'erreur
```

```
        err_xx[n] = np.linalg.norm(err_xxx, ord=2) *delta_t**(1/2)
        err_yy[n] = np.linalg.norm(err_yyy, ord=2) *delta_t**(1/2)
        err_zz[n] = np.linalg.norm(err_zzz, ord=2) *delta_t**(1/2)

    #L'effet de l'axe y et l'axe x du temps et de l'espace sur l'erreur
    err_Hx[i] = np.linalg.norm(err_xx, ord=2) *delta_t**(1/2)
    err_Hy[i] = np.linalg.norm(err_yy, ord=2) *delta_t**(1/2)
    err_Ez[i] = np.linalg.norm(err_zz, ord=2) *delta_t**(1/2)
h = L / I_list

Hx_ordre = np.polyfit(np.log(h), np.log(err_Hx), 1)[0]
Hy_ordre = np.polyfit(np.log(h), np.log(err_Hy), 1)[0]
Ez_ordre = np.polyfit(np.log(h), np.log(err_Ez), 1)[0]
print(Hx_ordre)
print(Hy_ordre)
print(Ez_ordre)

#%%
# figure d'erreur
plt.figure()
plt.plot(h, err_Ez, label="erreur de Ez", color='r')
plt.plot(h, err_Hx, label="erreur de Hx", color='b', ls='--')
plt.plot(h, err_Hy, label="erreur de Hy", color='g', ls='-.')
plt.title("L'erreur du schema")plt.xlabel('delta_x')
plt.ylabel('erreur')
```

```
plt.legend()

plt.show()


# figure d'ordre de convergence

plt.figure()

plt.plot(np.log(h), np.log(err_Ez), label=f"ordre d'erreur de Ez = {round(Ez_ordre,2)}", color='r')

plt.plot(np.log(h), np.log(err_Hx), label=f"ordre d'erreur de Hx = {round(Hx_ordre,2)}", color='b', ls='--')

plt.plot(np.log(h), np.log(err_Hy), label=f"ordre d'erreur de Hy = {round(Hy_ordre,2)}", color='g', ls='-.')

plt.title("L'ordre de convergence")

plt.xlabel('log(delta_x)')

plt.ylabel('log(erreur)')

plt.legend()

plt.show()
```