Numerical Simulation of 2D Wave Propagation in a PEC Cavity using the FDTD Method

Mina Shiri (40011023)

July 16, 2025

Abstract

This report details the numerical solution of the two-dimensional Maxwell's equations for a Transverse Magnetic (TMz) wave using the Finite-Difference Time-Domain (FDTD) method. The simulation is implemented in Python and visualizes the electromagnetic fields inside a rectangular cavity with Perfect Electric Conductor (PEC) boundaries. The problem considers a 2D domain with a centrally located dielectric square and an external excitation source in the form of a Gaussian pulse. We discuss the mathematical formulation, the discretization using the Yee algorithm, the stability conditions for the numerical scheme, and the implementation details. The results show the successful propagation of the wave, including reflection from the PEC walls and interaction with the dielectric material.

1 Introduction

The study of electromagnetic wave propagation in bounded structures is fundamental to many areas of electrical engineering and physics, including microwave engineering, antenna design, and optics. The Finite-Difference Time-Domain (FDTD) method is a powerful and widely used numerical technique for solving Maxwell's equations in the time domain. This report focuses on a specific problem: a two-dimensional rectangular cavity of dimensions $L = 1\,\text{m}$ and $W = 0.5\,\text{m}$ with PEC walls. A square dielectric block with a side length of 20 cm and a dielectric constant of $\epsilon_r = 13.1$ is placed at the center of this cavity. The system is excited by a linear current source located 10 cm from one of the corners. We will use the 2D FDTD method to simulate the evolution of the electromagnetic fields and present the field distributions at $t = 100\,\text{ns}$.

2 Problem Statement

We aim to solve the 2D Maxwell's equations for a TMz wave on a rectangular domain $x \in [0, 1]$, $y \in [0, 0.5]$ for a time interval up to $t = 100\,\text{ns}$. The specific conditions for the problem are as follows:

- **Domain:** Length $L = 1$ meter, Width $W = 0.5$ meter.
- **Boundary Conditions:** Perfect Electric Conductor (PEC) boundaries are applied at all four edges of the domain, meaning the tangential electric field (E_z) is fixed at zero.
- **Material Properties:** The domain contains a dielectric square with $\epsilon_r = 13.1$ and side length 0.2 m at its center. The rest of the domain is free space.
- **Excitation Source:** The system is excited by an external current source J_z with a Gaussian pulse waveform, located at position $(x, y) = (0.1, 0.1)$.

3 Numerical Method

To solve Maxwell's equations numerically, we employ the FDTD method, which involves discretizing the continuous domain into a grid of points (the Yee grid) and approximating the derivatives with finite differences.

3.1 Discretization

For a 2D TMz problem (where the electric field has only a z-component, and the magnetic field has x and y components), Maxwell's equations in a lossless medium are:

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu_0}\frac{\partial E_z}{\partial y}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu_0}\frac{\partial E_z}{\partial x}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon_0 \epsilon_r}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - J_z\right)$$

3.2 Central Difference Approximation

Using second-order central difference approximations for the derivatives leads to the explicit FDTD update equations. The magnetic fields are updated at half-time steps, and the electric field is updated at full-time steps. The vectorized update equations implemented in the code are:

$$H_x^{n+1/2}|_{i,j} = H_x^{n-1/2}|_{i,j} - \frac{\Delta t}{\mu_0 \Delta y}\left(E_z^n|_{i,j+1} - E_z^n|_{i,j}\right)$$

$$H_y^{n+1/2}|_{i,j} = H_y^{n-1/2}|_{i,j} + \frac{\Delta t}{\mu_0 \Delta x}\left(E_z^n|_{i+1,j} - E_z^n|_{i,j}\right)$$

$$E_z^{n+1}|_{i,j} = E_z^n|_{i,j} + \frac{\Delta t}{\epsilon_{i,j}}\left[\frac{H_y^{n+1/2}|_{i,j} - H_y^{n+1/2}|_{i-1,j}}{\Delta x} - \frac{H_x^{n+1/2}|_{i,j} - H_x^{n+1/2}|_{i,j-1}}{\Delta y}\right]$$

The source term J_z is added to the E_z update equation.

3.3 Stability Condition (CFL)

For the explicit scheme to be numerically stable, the time step Δt must satisfy the Courant-Friedrichs-Lewy (CFL) condition:

$$\Delta t \leq \frac{1}{c_0 \sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}}$$

where c_0 is the speed of light in vacuum. The implementation uses a Courant number of 0.99 for stability.

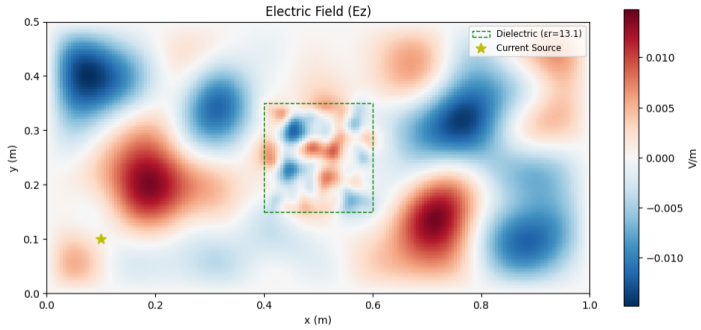
4 Implementation Details

The simulation is built using Python with the NumPy and Matplotlib libraries.

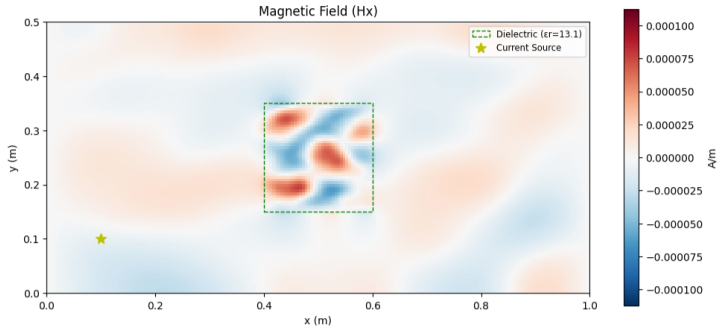
- **Parameters:** Key simulation parameters such as L, W, t_{final}, dx, dy, and the source details are defined as constants.
- **Data Structures:** Three 2D NumPy arrays (`Ez`, `Hx`, `Hy`) are used to store the field amplitudes at each grid point. A permittivity map `epsilon_r_map` stores the material properties across the grid.
- **Initialization:** The script initializes all field arrays to zero and sets up the `epsilon_r_map` array with the dielectric constant in the specified region.
- **Main Loop:** A `for` loop iterates through the time steps. In each iteration, it calculates the `Hx` and `Hy` fields, followed by the `Ez` field, using vectorized NumPy operations for efficiency.
- **Boundary Conditions:** The PEC boundary conditions are enforced implicitly. The update for `Ez` is only performed for the interior grid points (from index 1 to $N-1$), leaving the boundary points at their initial value of zero throughout the simulation.
- **Visualization:** After the simulation loop completes, `matplotlib.pyplot` is used to create image plots (`imshow`) of the `Ez`, `Hx`, and `Hy` fields at the final time step.

5 Simulation and Results

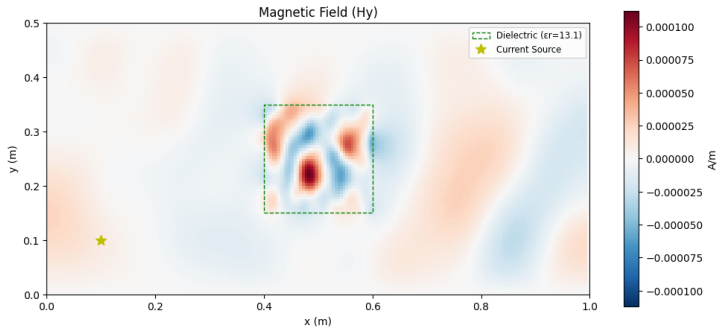
The FDTD simulation was executed for a total time of 100 ns. The resulting distributions for the electric field component E_z and the magnetic field components H_x and H_y at the final time step are presented in Figure 1. These plots provide a snapshot of the complex wave phenomena occurring within the cavity.



(a) Electric field (E_z) distribution.



(b) Magnetic field (H_x) distribution.



(c) Magnetic field (H_y) distribution.

Figure 1: Field distributions at $t = 100$ ns. The plots show complex interference patterns resulting from reflections off the PEC walls and interaction with the central dielectric block. The source location and dielectric are marked.

5.1 Discussion of E_z Field

The E_z field plot (Figure 1a) clearly shows the cylindrical wave fronts emanating from the source location. As these waves propagate, they interact with the boundaries and the dielectric material. The reflections from the four PEC walls result in a complex interference pattern, leading to the formation of standing waves within the cavity. The interaction with the central dielectric block, which has a higher permittivity, is also visible. The wavelength of the wave is shorter within the dielectric, indicating a lower speed of propagation ($c = c_0/\sqrt{\epsilon_r}$), which is consistent with electromagnetic theory.

5.2 Discussion of H_x and H_y Fields

The magnetic field components, H_x and H_y (Figures 1b and 1c), are orthogonal to the electric field and to each other. Their distribution is directly related to the spatial derivatives of the E_z field, as dictated by Maxwell's equations. For instance, strong H_x fields are observed where there is a strong gradient of E_z in the y-direction, and strong H_y fields correspond to strong gradients of E_z in the x-direction. These fields also exhibit similar reflection and interference phenomena, contributing to the overall energy distribution within the cavity.

6 Conclusion

This project successfully demonstrates the numerical solution of the 2D Maxwell's equations in a PEC cavity with an inhomogeneous medium using the FDTD method. The Python implementation provides an effective way to model and visualize complex electromagnetic wave dynamics. The simulation accurately captures fundamental wave behaviors, including propagation, reflection, and interaction with dielectric materials, offering valuable insight into the underlying physics.

A Python Simulation Code

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

# Define Physical Constants
c0 = 2.99792458e8  # Speed of light in vacuum (m/s)
eps0 = 8.854187817e-12  # Permittivity of free space (F/m)
mu0 = np.pi * 4e-7  # Permeability of free space (H/m)

# Setup Simulation Domain and Grid
L = 1.0  # Length of the rectangle (x-direction) in meters
W = 0.5  # Width of the rectangle (y-direction) in meters
dx = 0.005  # Spatial step in x (m)
dy = 0.005  # Spatial step in y (m)
Nx = int(L / dx)  # Number of grid points in x
Ny = int(W / dy)  # Number of grid points in y

# Setup Time Discretization
courant_number = 0.99
dt = courant_number / (c0 * np.sqrt((1/dx**2) + (1/dy**2)))
t_final = 100e-9  # Total simulation time in seconds (100 ns)
n_steps = int(t_final / dt)

# Define Material Properties
epsilon_r_map = np.ones((Nx, Ny))
diel_const = 13.1
diel_size = 0.20  # 20 cm side length
diel_x_start = int((L / 2 - diel_size / 2) / dx)
diel_x_end = int((L / 2 + diel_size / 2) / dx)
diel_y_start = int((W / 2 - diel_size / 2) / dy)
diel_y_end = int((W / 2 + diel_size / 2) / dy)
epsilon_r_map[diel_x_start:diel_x_end, diel_y_start:diel_y_end] = diel_const

# Initialize Field Arrays
Ez = np.zeros((Nx, Ny))
Hx = np.zeros((Nx, Ny))
Hy = np.zeros((Nx, Ny))

# Define the Current Source
source_x = int(0.1 / dx)
source_y = int(0.1 / dy)
tau = 20 * dt
t0 = 6 * tau

# Pre-calculate Update Coefficients
C_ez = dt / eps0
C_hx = dt / mu0
C_hy = dt / mu0

# Main FDTD Time-Stepping Loop
for n in range(n_steps):
    Hx[:, :-1] -= C_hx * (Ez[:, 1:] - Ez[:, :-1]) / dy
    Hy[:-1, :] += C_hy * (Ez[1:, :] - Ez[:-1, :]) / dx
    Ez[1:-1, 1:-1] += (C_ez / epsilon_r_map[1:-1, 1:-1]) * \
        ((Hy[1:-1, 1:-1] - Hy[:-2, 1:-1]) / dx - \
         (Hx[1:-1, 1:-1] - Hx[1:-1, :-2]) / dy)

    current_t = np.exp(-((n * dt - t0) / tau)**2)
    Ez[source_x, source_y] -= (C_ez / epsilon_r_map[source_x, source_y]) * current_t

# Plotting code would follow here...
```

Listing 1: Core Python code for the 2D FDTD simulation.