A Finite Difference Time Domain Method for Two Dimensional Electromagnetic Wave Propagation In Heterogeneous Lossy Dielectric Media With a Perfectly Matched Layer Boundary Condition

Patrick D. Cook

Department of Physics, Fort Hays State University,
Hays, Kansas 67601, USA

Fall 2019

Abstract

In this work, the foundations behind the one- and two-dimensional finite-difference time-domain (FDTD) method in the context of electromagnetic wave propagation are presented. Heterogeneous propagation media with arbitrary dielectric constants and arbitrary electrical conductivities are considered. Different boundary conditions, hard boundaries, cyclic boundaries, and the perfectly matched layer (PML), are presented and discussed. This work include an implementation of the method using the PML boundary condition, written in standard Python 3. This implementations is used to explore a variety of simple and complex geometries.

Keywords: Electromagnetic wave simulation, finite-difference time-domain, perfectly matched layer

1 Introduction

The goal of the finite-difference time-domain (FDTD) method is to simulate the propagation of electromagnetic waves in complex geometry. For instance, propagation through several different materials with different dielectric constants, or electrical conductivities. Problems such as these are often impossible to solve analytically, and so numerical approaches, such as this one, are required.

This work is an overview of the derivation of the necessary expressions used in the FDTD method as well as two types of boundary conditions. Implementations of the method for both boundary conditions with results are presented and discussed.

2 Governing Equations for Electromagnetic Waves

The differiential form of Maxwell's equations for the electric field, **E**, and magnetic field, **B**, in vacuum, are [1]

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0}, \tag{1a}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{1b}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t}, \tag{1c}$$

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \frac{1}{c^2}\frac{\partial \mathbf{E}}{\partial t}. \tag{1d}$$

Where

- ρ is the free electric charge density
- **J** is the free current density
- ε_0 is the permittivity of free space
- μ_0 is the permeability of free space
- c is the speed of light, $c \equiv \frac{1}{\sqrt{\varepsilon_0 \mu_0}}$

In the context of electromagnetic wave propagation, only Eqs. 1c and 1d are of interest. Assuming there is no free current, rewriting these two equations in terms of the magnetic field strength, $\mathbf{H} = \mathbf{B}/\mu_0$, and subsituting $c \equiv \frac{1}{\sqrt{\varepsilon_0 \mu_0}}$ yields

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0}\nabla \times \mathbf{E}, \tag{2a}$$

$$\varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H}. \tag{2b}$$

In a dielectric medium, we introduce the relative permittivity, ε_r, which is material-specific. This allows us to write the permittivity as $\varepsilon = \varepsilon_r \varepsilon_0$. We could also introduce the relative permeability, but in this work we will assume all materials are non-magnetic, meaning $\mu_r = 1$ and so $\mu = \mu_0$. Eqs. 2a and 2b become

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0}\nabla \times \mathbf{E}, \tag{3a}$$

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon}\nabla \times \mathbf{H}. \tag{3b}$$

A more general form of these equations uses the electric displacement, **D**, which can be written as $\mathbf{D}(\omega) = \varepsilon_0 \varepsilon_r^*(\omega)\mathbf{E}(\omega)$, where ω is the frequency of the wave and ε_r^* is the frequency dependent dielectric constant of the medium [2]. If we assume, for a lossy dielectric medium with relative permittivity ε_r and conductivity σ, that ε_r^* is of the form

$$\varepsilon_r^*(\omega) = \varepsilon_r + \frac{\sigma}{i\omega\varepsilon_0}, \tag{4}$$

then we see that the electric displacement is

$$\mathbf{D}(\omega) = \varepsilon_0 \varepsilon_r \mathbf{E}(\omega) + \frac{\sigma}{i\omega}\mathbf{E}(\omega). \tag{5}$$

Transforming this from the frequency domain to the time domain requires an identity which is derived in Appendix A. In the time domain, the above equation is

$$\mathbf{D}(t) = \varepsilon_0 \varepsilon_r \mathbf{E}(t) + \sigma \int_0^t \mathbf{E}(\tau) d\tau. \tag{6}$$

So that the relevant equations for electromagnetic waves are then

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0} \boldsymbol{\nabla} \times \mathbf{E}, \tag{7a}$$

$$\mathbf{D}(t) = \varepsilon_0 \varepsilon_r \mathbf{E}(t) + \sigma \int_0^t \mathbf{E}(\tau) d\tau, \tag{7b}$$

$$\frac{\partial \mathbf{D}}{\partial t} = \boldsymbol{\nabla} \times \mathbf{H}. \tag{7c}$$

Finally, we express everything in Gaussian units instead of SI. The purpose for this is so that E, D, and H are around the same order of magnitude [2]. In Gaussian units,

$$\tilde{\mathbf{E}} = \sqrt{\frac{\varepsilon_0}{\mu_0}} \mathbf{E}, \tag{8a}$$

$$\tilde{\mathbf{D}} = \sqrt{\frac{1}{\varepsilon_0 \mu_0}} \mathbf{D}. \tag{8b}$$

This normalization yields the final set of equations that will be used to govern the propagation of electromagnetic waves in a lossy dielectric medium,

$$\boxed{\begin{aligned} \frac{\partial \mathbf{H}}{\partial t} &= -\frac{1}{\sqrt{\varepsilon_0 \mu_0}} \boldsymbol{\nabla} \times \tilde{\mathbf{E}}, && (9a) \\ \tilde{\mathbf{D}}(t) &= \varepsilon_r \tilde{\mathbf{E}}(t) + \frac{\sigma}{\varepsilon_0} \int_0^t \tilde{\mathbf{E}}(\tau) d\tau, && (9b) \\ \frac{\partial \tilde{\mathbf{D}}}{\partial t} &= \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \boldsymbol{\nabla} \times \mathbf{H}. && (9c) \end{aligned}}$$

In these equations, the two parameters that are material-specific appear only in Eq. 9b. Larger relative permittivities, ε_r, decrease the strength of the electric field per unit charge, which has the effect of slowing electromagnetic waves. The dielectric conductivity, σ, represents the total dissipative effect of the material, accounting for any energy loss of the waves in the material.

3 Finite Difference Approximation of the Derivative

Let $f(x)$ be any continuous and thrice differientiable function with bounded derivatives. If we want to obtain an approximation of $f'(x)$, we start by considering $f(x+h)$ and $f(x-h)$, where h is a small displacement. Theorem 1 in Appendix C allows us to write $f(x+h)$ and $f(x-h)$ as shown in Eq. 10. Subtracting both expressions in Eq. 10 and rearranging yields Eq. 11a.

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(\eta_1) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(\eta_2) \end{aligned} \tag{10}$$

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{h^2}{6}\frac{f'''(\eta_1)+f'''(\eta_2)}{2} \tag{11a}$$

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{h^2}{6}f'''(\eta) \tag{11b}$$

Here, η_1 and η_2 are two unknown points in the domain of f. By applying the intermediate value theorem to the last term in Eq. 11a, the two unknown points can be reduced to a single unknown point, η. This leads to Eq. 11b. This equation suggests an approximation for the first derivative,

$$f'(x) \approx \frac{f(x+h)-f(x-h)}{2h}. \tag{12}$$

With an error of

$$|E| = \left|\frac{h^2}{6}f'''(\eta)\right|. \tag{13}$$

Since f and all derivatives of f are bounded, we may introduce M, the absolute maxmimum value of $f'''(x)$, meaning $|f'''(x)| \leq M \; \forall \; x$. Thus, the error term is a constant multiplied by h^2. We say then that this error is an order of h^2, or

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2). \tag{14}$$

This is known as the central difference approximation of the derivative. Due to the error being in h^2, this is a second order approximation. There are many other approximations of the first derivative as well as higher order derivatives. However, this is the only derivative approximation necessary in FDTD.

4 Formulation of FDTD in One Dimension

For example purposes, a formulation of the FDTD method in one dimension is presented here. First, we must choose an axis of propagation. Arbitrarily, suppose the wave travels only in the $\hat{z}$ direction. Eqs. 9a and 9c become

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\partial \tilde{E}_x}{\partial z}, \tag{15a}$$

$$\frac{\partial \tilde{D}_x}{\partial t} = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\partial H_y}{\partial z}. \tag{15b}$$

We will interleave H_y and $\tilde{D}_x$ both spatially and temporally. This is one of the fundamental features of the FDTD method [2]. To interleave the two fields, we evaluate them—as well as their derivatives—offset from one another by $\Delta z/2$ in space and $\Delta t/2$ in time,

$$\frac{\partial \tilde{D}_x}{\partial t}(z,t) = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\partial H_y}{\partial z}(z,t) \tag{16a}$$

$$\frac{\partial H_y}{\partial t}\left(z+\frac{\Delta z}{2}, t+\frac{\Delta t}{2}\right) = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\partial \tilde{E}_x}{\partial z}\left(z+\frac{\Delta z}{2}, t+\frac{\Delta t}{2}\right) \tag{16b}$$

$$\frac{\tilde{D}_x\left(z,t+\frac{\Delta t}{2}\right) - \tilde{D}_x\left(z,t-\frac{\Delta t}{2}\right)}{\Delta t} = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{H_y\left(z+\frac{\Delta z}{2},t\right) - H_y\left(z-\frac{\Delta z}{2},t\right)}{\Delta z} \tag{17a}$$

$$\frac{H_y\left(z+\frac{\Delta z}{2},t+\Delta t\right) - H_y\left(z+\frac{\Delta z}{2},t\right)}{\Delta t} = -\frac{1}{\sqrt{\varepsilon_0\mu_0}}\frac{\tilde{E}_x\left(z+\Delta z,t+\frac{\Delta t}{2}\right) - \tilde{E}_x\left(z,t+\frac{\Delta t}{2}\right)}{\Delta z} \tag{17b}$$

Using the central difference approximation of the derivative, Eq. 12, on each of the derivatives—using the same spacing as the offsets from the interleaving process, $\Delta z/2$ for the spatial stepsize and $\Delta t/2$ for the temporal stepsize—leads us to Eqs. 17a and 17b.

Now, we impose restrictions on Δt and Δz to ensure stability. A discussion including the reasoning for this is presented in the next section. By forcing

$$\Delta t = \frac{\Delta z}{2c_0}, \tag{18}$$

where $c_0 = 1/\sqrt{\varepsilon_0\mu_0}$ just as before, we guarantee the stability of the numerical method. Using this condition and rearranging Eqs. 17a and 17b into an iterative form leads to

$$\begin{aligned}\tilde{D}_x\left(z,t+\frac{\Delta t}{2}\right) = \tilde{D}_x&\left(z,t-\frac{\Delta t}{2}\right) \\ &-\frac{1}{2}\left[H_y\left(z+\frac{\Delta z}{2},t\right)\right. \\ &\left.-H_y\left(z-\frac{\Delta z}{2},t\right)\right],\end{aligned} \tag{19a}$$

$$\begin{aligned}H_y\left(z+\frac{\Delta z}{2},t+\Delta t\right) = H_y&\left(z+\frac{\Delta z}{2},t\right) \\ &-\frac{1}{2}\left[\tilde{E}_x\left(z+\Delta z,t+\frac{\Delta t}{2}\right)\right. \\ &\left.-\tilde{E}_x\left(z,t+\frac{\Delta t}{2}\right)\right].\end{aligned} \tag{19b}$$

Equation 19a tells us the entire $\tilde{D}_x$ field at the next time step if both $\tilde{D}_x$ and H_y are known at the previous time steps. Likewise, Eq. 19b tells us the entire H_y field at the next time step if $\tilde{E}_x$ and H_y are known at the previous time steps. We can discretize these expressions by defining

$$\begin{cases} z_i = i\Delta z \\ t_j = \left(j + \frac{1}{2}\right)\Delta t \end{cases} \quad \text{for } \tilde{E}_x \text{ and } \tilde{D}_x, \qquad \begin{cases} z_i = \left(i + \frac{1}{2}\right)\Delta z \\ t_j = j\Delta t \end{cases} \quad \text{for } H_y. \tag{20}$$

Which results in much more succinct expressions,

$$\begin{aligned} \tilde{D}_x(z_i, t_j) &= \tilde{D}_x(z_i, t_{j-1}) \\ &- \frac{1}{2}\left[H_y(z_i, t_j) - H_y(z_{i-1}, t_j)\right], \end{aligned} \tag{21a}$$

$$\begin{aligned} H_y(z_i, t_{j+1}) &= H_y(z_i, t_j) \\ &- \frac{1}{2}\left[\tilde{E}_x(z_{i+1}, t_j) - \tilde{E}_x(z_i, t_j)\right]. \end{aligned} \tag{21b}$$

However, we still need a way to find $\tilde{E}_x$ at the next timestep. For this, we return to Eq. 9b. The discrete form of the integral is just a summation, so the discrete form of Eq. 9b is

$$\tilde{D}_x(z_i, t_j) = \varepsilon_r \tilde{E}_x(z_i, t_j) + \frac{\sigma}{\varepsilon_0}\sum_{n=0}^{j} \tilde{E}_x(z_i, t_n)\Delta t. \tag{22}$$

By separating the j^{th} term from the rest of the summation, we see that

$$\begin{aligned} \tilde{D}_x(z_i, t_j) = \varepsilon_r \tilde{E}_x(z_i, t_j) &+ \frac{\sigma\Delta t}{\varepsilon_0}\tilde{E}_x(z_i, t_j) \\ &+ \frac{\sigma\Delta t}{\varepsilon_0}\sum_{n=0}^{j-1} \tilde{E}_x(z_i, t_n). \end{aligned} \tag{23}$$

This allows us to find $\tilde{E}_x(z_i, t_j)$, which is $\tilde{E}$ at the current time, from $\tilde{D}_x$ at the current time as well as $\tilde{E}_x$ at all past times,

$$\tilde{E}_x(z_i, t_j) = \frac{\tilde{D}_x(z_i, t_j) - \frac{\sigma\Delta t}{\varepsilon_0}\sum_{n=0}^{j-1} \tilde{E}_x(z_i, t_j)}{\varepsilon_r + \frac{\sigma\Delta t}{\varepsilon_0}}. \tag{24}$$

For conciseness as well as efficiency in implementation, we will introduce

$$I(z_i, t_j) = \frac{\sigma\Delta t}{\varepsilon_0}\sum_{n=0}^{j} \tilde{E}_x(z_i, t_n). \tag{25}$$

This allows us to forgo the need to perform the entire summation each time, instead simply adding the newest value of $\tilde{E}_x$ each time instead.

With the substitution of Eq. 25 in Eq. 24 alongside Eqs. 21a and 21b, we arrive at the formulation of FDTD in one dimension, arranged in iterative form,

$$\boxed{\begin{aligned} \tilde{D}_x(z_i, t_j) &= \tilde{D}_x(z_i, t_{j-1}) \\ &- \frac{1}{2}\left[H_y(z_i, t_j) - H_y(z_{i-1}, t_j)\right], && \text{(26a)} \\ \tilde{E}_x(z_i, t_j) &= \frac{\tilde{D}_x(z_i, t_j) - I(z_i, t_{j-1})}{\varepsilon_r + \frac{\sigma\Delta t}{\varepsilon_0}}, && \text{(26b)} \\ I(z_i, t_j) &= \frac{\sigma\Delta t}{\varepsilon_0}\sum_{n=0}^{j} \tilde{E}_x(z_i, t_n), && \text{(26c)} \\ H_y(z_i, t_{j+1}) &= H_y(z_i, t_j) \\ &- \frac{1}{2}\left[\tilde{E}_x(z_{i+1}, t_j) - \tilde{E}_x(z_i, t_j)\right]. && \text{(26d)} \end{aligned}}$$

Again, an important observation is that only Eqs. 26b and 26c contain information about the propagation medium. Thus, Eqs. 26a and 26d do not change in different media.

5 Stability

The stability conditions of FDTD, unlike many numerical methods, comes from physical reasoning rather than mathematical reasoning. FDTD propagates electromagnetic waves, which clearly cannot move faster than c_0, the speed of light in vacuum. As such, imagine an electromagnetic wave propagating in one dimension. If the distance it travels is Δx, then the minimum time required is $\Delta t_{min} = \Delta x / c_0$. Suppose this distance, Δx, is the length of one cell in the one-dimensional FDTD, then the timestep, Δt, must be no greater than Δt_{min}. So in one dimension, $\Delta t \leq \Delta x / c_0$. Now, consider a two-dimensional FDTD, now the maximum distance the wave could travel in one cell is the diagonal of each square cell. This leads us to a requirement of $\Delta t \leq \frac{\Delta x}{c_0 \sqrt{2}}$ where Δx is the size of the cell in both spatial dimensions. This trend continues, leading us to the Courant-Friedrichs-Levy (CFL) condition for an explicit method [3],

$$\Delta t \leq \frac{1}{c_0 \sqrt{\sum_i \frac{1}{(\Delta x_i)^2}}}, \tag{27}$$

where the sum in the denominator is over all spatial dimensions. For example, the stability condition for a three-dimensional FDTD would be $\Delta t \leq \frac{\Delta x}{c_0 \sqrt{3}}$ if the size of the cells were $\Delta x \times \Delta x \times \Delta x$. For n dimensions with equal spacing,

$$\Delta t \leq \frac{\Delta x}{c_0 \sqrt{n}}. \tag{28}$$

For simplicity in this paper, we choose

$$\Delta t = \frac{\Delta x}{2 c_0}, \tag{29}$$

which guarantees stability in one-, two-, three-, and even four-dimensional FDTD simulations. However, this is not the fastest or most efficient choice of Δt.

6 Boundary Conditions

In code, we cannot simulate infinite fields. This applies not only to the one-dimensional method, but all higher dimensional methods as well. As such, we will talk about x as a general spatial coordinate here, instead of z. Since we cannot store fields with infinite size in any computer code, there must be some finite number of x values, say N. Then there must be some initial and final x_i, say x_0 and x_{N-1} respectively. This section discusses three techniques to handle the non-existent values of x_{-1} and x_N.

6.1 Hard Boundaries

One of the simplest ways to handle the boundaries is to force $x_i = 0$ for $i < 0$ and $x_i = 0$ or $i > N - 1$. However, this results in erroneous reflections, which are not ideal [2].

6.2 Cyclic Boundaries

A similar, but slightly more efficient boundary condition is the cyclic boundary condition in which

$$i = \begin{cases} i + N\left\lceil\left|\frac{i}{N}\right|\right\rceil & \text{if } i < 0, \\ i & \text{if } 0 \leq i \leq N-1, \\ i - N\left\lfloor\left|\frac{i}{N}\right|\right\rfloor & \text{if } i > N-1. \end{cases} \tag{30}$$

Where $\lceil\ \rceil$ and $\lfloor\ \rfloor$ are the ceiling and floor functions respectively. This condition makes it so that leaving one side of the computational domain results in appearing on the other.

As an example, let $N = 100$ and suppose the values of x_{128} and x_{-128} are required. In the first case, we are in the bottom case of Eq. 30, which evaluates to $i = 128 - 100\lfloor 1.28\rfloor = 28$. Likewise, for x_{-128}, $i = -128 + 100\lceil 1.28\rceil = 72$. And so $x_{128} \to x_{28}$ and $x_{-128} \to x_{72}$—values that exist within the domain.

In terms of erroneous propagation, this method is not necessarily any better than the previous one. Instead of erroneous reflections, now certain parts of the boundary act as sources, assuming a wave propagated through the opposing side.

6.3 Perfectly Matched Layer

A much more powerful boundary condition is the perfectly matched layer (PML) [4]. This is an absorbing boundary condition that allows us to imitate infinite fields. As the waves reach the boundary, they are entirely absorbed, with no reflections. Within the domain, this is the same as if the waves were allowed to continue propagating to infinity.

First, consider the reflection coefficient at the boundary between medium A and medium B,

$$\Gamma = \frac{\eta_A - \eta_B}{\eta_A + \eta_B}, \tag{31}$$

where η is the intrinsic impedance of either medium, given by

$$\eta = \sqrt{\frac{\mu}{\varepsilon}}. \tag{32}$$

We have considered μ to be constant for all media (with a value of μ_0) in all prior discussions. However, suppose near the edge of the computational domain, a medium was placed with a specifically chosen μ such that $\Gamma = 0$, hence the name "perfectly matched layer". Then, there would be no reflections from this medium. This solves the problem of reflections from this medium, but waves would still reflect from the boundary of the domain, if not sufficiently attenuated in this new medium. Just as in previous discussions, the conductivity, σ, represents the attenuatation of a medium. Thus, we may introduce fictitious media near the boundaries of the computational domain with specifically chosen constants μ and σ such that they do not reflect any waves and attenuate the waves that enter them. These media are the perfectly matched layers, since their intrinsic impedance exactly matches that of the neighboring domain. Figure 1 illustrates the placement of the PML in a two-dimensional domain. It is important to note that the following discussion will be in the general three-dimensional case.

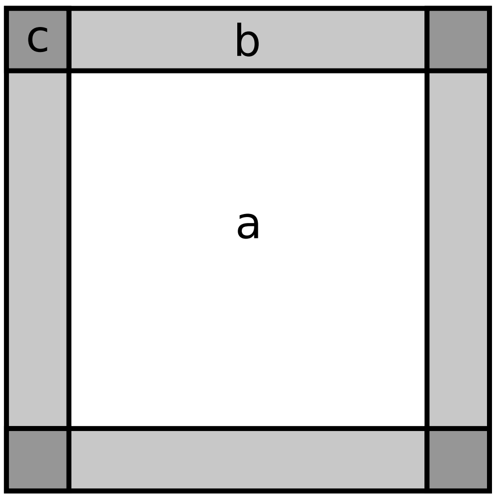

Figure 1: Placement of the PML in a two-dimensional domain. In (a), no PML is present and thus no material constants have been chosen or modified. The area marked by (b) is where the PML for the y direction has been placed, waves will be attenuated in this region to prevent them from reflecting from the boundary of the domain. The PMLs in the x and y directions overlap in (c).

To create these PMLs, we introduce fictitious dielectric constants and permeabilities—in the same form as before—for each direction, which are functions of position in any spatial direction, $r \in \{x, y, z\}$,

$$\begin{aligned} \varepsilon^*_{Fm}(r) &= \varepsilon_{Fm}(r) + \frac{\sigma_{Dm}(r)}{i\omega\varepsilon_0}, \\ \mu^*_{Fm}(r) &= \mu_{Fm}(r) + \frac{\sigma_{Hm}(r)}{i\omega\mu_0}, \\ &\text{for } m \in \{x, y, z\}, \end{aligned} \tag{33}$$

where σ_{Dm} denotes the electric conductivity in the m direction, and σ_{Hm} denotes the magnetic conductivity in the m direction. Note that m and r are independent. The spatial dependence is denoted by r, which can be any of the three spatial coordinates $\{x, y, z\}$. However, m denotes the direction that the constant applies in. For example, $\varepsilon^*_{Fx}(y)$ denotes the x-axis dielectric constant as a function of y, which may change with y, but not with x or z. It is very important to note that ε^*_{Fm} and μ^*_{Fm} have nothing at all to do with ε^*_r, introduced in Eq. 4, the relative permittivity of the propagation medium. These fictitious constants only apply in the PMLs, which reside near the boundaries of the computational domain.

We introduce $\boldsymbol{\varepsilon}^*_{\boldsymbol{F}}(\mathbf{r})$ and $\boldsymbol{\mu}^*_{\boldsymbol{F}}(\mathbf{r})$, which are functions of position,

$$\boldsymbol{\varepsilon}^*_{\boldsymbol{F}} = \begin{bmatrix} \varepsilon^*_{Fx}(x) \cdot \varepsilon^*_{Fx}(y) \cdot \varepsilon^*_{Fx}(z) \\ \varepsilon^*_{Fy}(x) \cdot \varepsilon^*_{Fy}(y) \cdot \varepsilon^*_{Fy}(z) \\ \varepsilon^*_{Fz}(x) \cdot \varepsilon^*_{Fz}(y) \cdot \varepsilon^*_{Fz}(z) \end{bmatrix}, \tag{34a}$$

$$\boldsymbol{\mu}^*_{\boldsymbol{F}} = \begin{bmatrix} \mu^*_{Fx}(x) \cdot \mu^*_{Fx}(y) \cdot \mu^*_{Fx}(z) \\ \mu^*_{Fy}(x) \cdot \mu^*_{Fy}(y) \cdot \mu^*_{Fy}(z) \\ \mu^*_{Fz}(x) \cdot \mu^*_{Fz}(y) \cdot \mu^*_{Fz}(z) \end{bmatrix}. \tag{34b}$$

Which, after transforming Eqs. 9a, 9b, and 9c to the frequency domain—which requires identities derived in Appendices A and B—allows us to write

$$\boxed{\begin{aligned} i\omega\boldsymbol{\mu}^*_{\boldsymbol{F}}\odot\mathbf{H}(\omega) &= -c_0\boldsymbol{\nabla}\times\tilde{\mathbf{E}}(\omega), && (35a)\\ i\omega\boldsymbol{\varepsilon}^*_{\boldsymbol{F}}\odot\tilde{\mathbf{D}}(\omega) &= c_0\boldsymbol{\nabla}\times\mathbf{H}(\omega), && (35b)\\ \tilde{\mathbf{D}}(\omega) &= \varepsilon_r(\omega)\tilde{\mathbf{E}}(\omega) + \frac{\sigma}{i\omega\varepsilon_0}\tilde{\mathbf{E}}(\omega). && (35c) \end{aligned}}$$

Where $\odot$ denotes the Hadamard product (elementwise product). This is the restatement of Maxwell's equations in the frequency domain in the context of a PML. We must be state them in the frequency domain first, since our assumed form of ε^*_{Fm} and μ^*_{Fm} are frequency-dependent, shown in Eq. 33. As before, only Eq. 35c contains information about the propagation medium. However, this time Eqs. 35a and 35b are the only equations to contain information about the PML.

There are two restrictions for these fictitious constants to form a PML [5],

1. The impedance must be constant across the boundary of the PML,

$$\eta_0 = \eta_m = \sqrt{\frac{\mu^*_{Fm}(r)}{\varepsilon^*_{Fm}(r)}}, \tag{36}$$

   where η_0 is the impedance of the non-PML domain, shown as (a) in Fig. 1. Since we are working in Gaussian units [2, 6],

$$\eta_0 = 1. \tag{37}$$

2. The constants for the directions perpendicular to the boundary must be the inverse of the constants in the transverse directions,

$$\begin{aligned} \varepsilon^*_{Fm_\perp}(r) &= \frac{1}{\varepsilon^*_{Fm_\parallel}(r)} \; \forall m_\parallel, \\ \mu^*_{Fm_\perp}(r) &= \frac{1}{\mu^*_{Fm_\parallel}(r)} \; \forall m_\parallel. \end{aligned} \tag{38}$$

   Where $m_\perp$ denotes the direction of perpendicular to the boundary and $m_\parallel$ denotes all of the directions parallel to the boundary. For example, in region (b) in Fig. 1, the direction perpendicular to the boundary is $m_\perp = y$, and the transverse direction in the two-dimensional case would be $m_\parallel = x$ (in the three-dimensional case we would also need to include the $m_\parallel = z$ direction). So for this example, $\varepsilon^*_{Fy}(r) = 1/\varepsilon^*_{Fx}(r)$ and $\mu^*_{Fy}(r) = 1/\mu^*_{Fx}(r)$.

To satisfy both of these conditions, we may choose, for all of the parallel coordinates,

$$\begin{aligned} \varepsilon_{Fm_\parallel}(r) &= \mu_{Fm_\parallel}(r) = 1, \\ \sigma_{Dm_\parallel}(r) &= \sigma_D(r), \\ \sigma_{Hm_\parallel}(r) &= \frac{\mu_0}{\varepsilon_0}\sigma_D(r). \end{aligned} \tag{39}$$

Substituting this into Eq. 33 and using Eq. 38 to find the constants in the perpendicular direction yields

$$\begin{cases} \varepsilon^*_{Fm_\parallel}(r) = 1 + \dfrac{\sigma_D(r)}{i\omega\varepsilon_0} \\ \mu^*_{Fm_\parallel}(r) = 1 + \dfrac{\sigma_D(r)}{i\omega\varepsilon_0} \end{cases}, \tag{40a}$$

$$\begin{cases} \varepsilon^*_{Fm_\perp}(r) = \left(1 + \dfrac{\sigma_D(r)}{i\omega\varepsilon_0}\right)^{-1} \\ \mu^*_{Fm_\perp}(r) = \left(1 + \dfrac{\sigma_D(r)}{i\omega\varepsilon_0}\right)^{-1} \end{cases}. \tag{40b}$$

Which clearly satisfies both conditions.

By gradually increasing σ_D in the PML, waves in the PML will be attenuated due to Eqs. 35a and 35b. We have not transformed Eqs. 35a, 35b, and 35c back into the time domain, since doing so is tricky and can be done after the dimensionality for an implementation is chosen.

7 Heterogeneous Media

While it has not been explicitly stated, all of the material constants pertaining to the propgation medium may be spatially dependent. We have already seen that the fictitious material constants of the PML were spatially dependent in order to attenuate waves. In Eq. 9b and 35c, ε_r and σ may be written as functions of position, $\mathbf{r}$. Consequently, in the formulation of FDTD in one-dimension, ε_r and σ could have been written as $\varepsilon_r(z_i)$ and $\sigma(z_i)$ respectively.

8 Formulation of FDTD in Two Dimensions with PML

Before we transform Eqs. 35a, 35b, and 35c back into the time domain and discretize them, we must choose between the transverse magnetic (TM) mode and transverse electric (TE) mode. For waves propagating in the xy-plane, the fields involved with each mode are

$$\text{TM} \begin{cases} \tilde{E}_z \\ H_x \\ H_y \end{cases}, \tag{41a}$$

$$\text{TE} \begin{cases} H_z \\ \tilde{E}_x \\ \tilde{E}_y \end{cases}. \tag{41b}$$

Arbitrarily, again, we will choose the TM mode. We start by implementing the PML in the x direction,

$$PML_x \begin{cases} \varepsilon^*_{Fx}(x) = \mu^*_{Fx}(x) = \left(1 + \dfrac{\sigma_D(x)}{i\omega\varepsilon_0}\right)^{-1} \\ \varepsilon^*_{Fy}(x) = \mu^*_{Fy}(x) = 1 + \dfrac{\sigma_D(x)}{i\omega\varepsilon_0} \\ \varepsilon^*_{Fz}(x) = \mu^*_{Fz}(x) = 1 + \dfrac{\sigma_D(x)}{i\omega\varepsilon_0} \end{cases}. \tag{42a}$$

And similarly for the y direction,

$$PML_y \begin{cases} \varepsilon^*_{Fy}(y) = \mu^*_{Fy}(y) = \left(1 + \dfrac{\sigma_D(y)}{i\omega\varepsilon_0}\right)^{-1} \\ \varepsilon^*_{Fx}(y) = \mu^*_{Fx}(y) = 1 + \dfrac{\sigma_D(y)}{i\omega\varepsilon_0} \\ \varepsilon^*_{Fz}(y) = \mu^*_{Fz}(y) = 1 + \dfrac{\sigma_D(y)}{i\omega\varepsilon_0} \end{cases}. \tag{43a}$$

Since the waves will propagate in the xy-plane, there is no z dependence in our two-dimensional formulation, so the PML is the z direction is ignored,

$$PML_z \begin{cases} \varepsilon^*_{Fz}(z) = \mu^*_{Fz}(z) = 1 \\ \varepsilon^*_{Fx}(z) = \mu^*_{Fx}(z) = 1 \\ \varepsilon^*_{Fy}(z) = \mu^*_{Fy}(z) = 1 \end{cases}. \tag{44a}$$

Thus, Eqs. 35a, 35b, and 35c become

$$i\omega\left(1+\frac{\sigma_D(x)}{i\omega\varepsilon_0}\right)^{-1}\left(1+\frac{\sigma_D(y)}{i\omega\varepsilon_0}\right)H_x(\omega) = -c_0\frac{\partial\tilde{E}_z}{\partial y}(\omega), \tag{45a}$$

$$i\omega\left(1+\frac{\sigma_D(x)}{i\omega\varepsilon_0}\right)\left(1+\frac{\sigma_D(y)}{i\omega\varepsilon_0}\right)^{-1}H_y(\omega) = c_0\frac{\partial\tilde{E}_z}{\partial x}(\omega), \tag{45b}$$

$$i\omega\left(1+\frac{\sigma_D(x)}{i\omega\varepsilon_0}\right)\left(1+\frac{\sigma_D(y)}{i\omega\varepsilon_0}\right)\tilde{D}_z(\omega) = c_0\left(\frac{\partial H_y}{\partial x}(\omega)-\frac{\partial H_x}{\partial y}(\omega)\right), \tag{45c}$$

$$\tilde{D}_z(\omega) = \varepsilon_r(\omega)\tilde{E}_z(\omega)+\frac{\sigma}{i\omega\varepsilon_0}\tilde{E}_z(\omega). \tag{45d}$$

We note again the important fact that Eq. 45d is the only part of this formulation to contain information about the propgation medium. All other parts, Eqs. 45a, 45b, and 45c, only contain information about propagation in free space and the PML.

We define discrete variables just as before,

$$\begin{cases} x_i = i\Delta x \\ y_j = j\Delta x \\ t_k = \left(k+\frac{1}{2}\right)\Delta t \end{cases} \quad \text{for } \tilde{E}_z \text{ and } \tilde{D}_z,$$
$$\begin{cases} x_i = \left(i+\frac{1}{2}\right)\Delta x \\ y_j = \left(j+\frac{1}{2}\right)\Delta x \\ t_k = k\Delta t \end{cases} \quad \text{for } H_x \text{ and } H_y,$$
$$\begin{cases} x_i = i\Delta x \\ y_j = j\Delta x \end{cases} \quad \text{for } \varepsilon_r,\ \sigma \text{ and, } \sigma_D. \tag{46}$$

Using this, we discretize Eqs. 45a, 45b, 45c, and 45d and transform them to the time-domain[1], treating Eq. 45d exactly as in previous discussions. To save space, we first introduce,

$$\partial_x\tilde{E}_z(x_i,y_i,t_k) = \tilde{E}_z(x_{i+1},y_j,t_k) - \tilde{E}_z(x_i,y_j,t_k), \tag{47a}$$

$$\partial_y\tilde{E}_z(x_i,y_i,t_k) = \tilde{E}_z(x_i,y_{j+1},t_k) - \tilde{E}_z(x_i,y_j,t_k), \tag{47b}$$

$$I_z(x_i,y_i,t_k) = \frac{\sigma(x_i,y_j)\Delta t}{\varepsilon_0}\sum_{n=0}^{k}\tilde{E}_z(x_i,y_j,t_n), \tag{47c}$$

$$I_{H_x}(x_i,y_i,t_k) = \sum_{n=0}^{k}\partial_y\tilde{E}_z(x_i,y_j,t_n), \tag{47d}$$

$$I_{H_y}(x_i,y_i,t_k) = \sum_{n=0}^{k}\partial_x\tilde{E}_z(x_i,y_j,t_n), \tag{47e}$$

$$f_{m1}(m_i) = \frac{\sigma_D(m_i)\Delta t}{2\varepsilon_0},\ m\in\{x,y\}, \tag{47f}$$

$$f_{m2}(m_i) = \frac{1}{1+\sigma_D(m_{i+\frac{1}{2}})\frac{\Delta t}{2\varepsilon_0}},\ m\in\{x,y\}, \tag{47g}$$

[1]See Appendix D for explaination on why so many steps were skipped here.

$$f_{m3}(m_i) = \frac{1-\sigma_D(m_{i+\frac{1}{2}})\frac{\Delta t}{2\varepsilon_0}}{1+\sigma_D(m_{i+\frac{1}{2}})\frac{\Delta t}{2\varepsilon_0}},\ m \in \{x, y\}, \tag{47h}$$

$$g_{m2}(m_i) = \frac{1}{1+\sigma_D(m_i)\frac{\Delta t}{2\varepsilon_0}},\ m \in \{x, y\}, \tag{47i}$$

$$g_{m3}(m_i) = \frac{1-\sigma_D(m_i)\frac{\Delta t}{2\varepsilon_0}}{1+\sigma_D(m_i)\frac{\Delta t}{2\varepsilon_0}},\ m \in \{x, y\}, \tag{47j}$$

$$G_{az}(x_i, y_j) = \frac{1}{\varepsilon_r(x_i, y_j) + \frac{\sigma(x_i,y_j)\Delta t}{\varepsilon_0}}, \tag{47k}$$

$$G_{bz}(x_i, y_j) = \frac{\sigma(x_i, y_j)\Delta t}{\varepsilon_0}. \tag{47l}$$

$$\boxed{\begin{aligned}
\tilde{D}_z(x_i, y_i, t_k) &= g_{x3}(x_i)g_{y3}(y_j)\tilde{D}_z(x_i, y_j, t_{k-1}) \\
&\quad + \frac{g_{x2}(x_i)g_{y2}(y_j)}{2}\left[H_y(x_i, y_i, t_k) - H_y(x_{i-1}, y_i, t_k)\right. \\
&\quad \left.-H_x(x_i, y_j, t_k) + H_x(x_i, y_{j-1}, t_k)\right], && \text{(48a)} \\
\tilde{E}_z(x_i, y_i, t_k) &= G_{az}(x_i, y_j)\left(\tilde{D}_z(x_i, y_j, t_k) - I_z(x_i, y_j, t_{k-1})\right), && \text{(48b)} \\
I_z(x_i, y_i, t_k) &= I_z(x_i, y_j, t_{k-1}) + G_{bz}(x_i, y_j)\tilde{E}_z, && \text{(48c)} \\
I_{H_x}(x_i, y_i, t_k) &= I_{H_x}(x_i, y_j, t_{k-1}) + \partial_y \tilde{E}_z(x_i, y_j, t_k) && \text{(48d)} \\
I_{H_y}(x_i, y_i, t_k) &= I_{H_y}(x_i, y_j, t_{k-1}) + \partial_x \tilde{E}_z(x_i, y_j, t_k) && \text{(48e)} \\
H_x(x_i, y_j, t_{k+1}) &= f_{y3}(y_j)H_x(x_i, y_j, t_k) - \frac{f_{y2}(y_j)\partial_y \tilde{E}_z(x_i, y_j, t_k)}{2} \\
&\quad - f_{x1}(x_i)I_{H_x}(x_i, y_j, t_k) && \text{(48f)} \\
H_y(x_i, y_j, t_{k+1}) &= f_{x3}(x_i)H_y(x_i, y_j, t_k) + \frac{f_{x2}(x_i)\partial_x \tilde{E}_z(x_i, y_j, t_k)}{2} \\
&\quad + f_{y1}(y_j)I_{H_y}(x_i, y_j, t_k) && \text{(48g)}
\end{aligned}}$$

Which allows us to write the FDTD formulation in 2D with a PML, arranged in iterative fashion above.

As mentioned before, gradually increasing σ_D in the PML will cause the waves in the PML to be attenuated without reflection. However, instead of gradually increasing σ_D, we may instead gradually increase $\frac{\sigma_D(m_i)\Delta t}{2\varepsilon_0}$, since all of the parts of Eq. 47—and thus also Eq. 48—with dependence on σ_D are in this form. We gradually increase this through an auxiliary parameter [2],

$$xn(i) = \frac{1}{3}\left(\frac{i}{L}\right)^3,\ i \in \{1, 2, 3, \cdots, L\}, \tag{49}$$

where L is the depth of the PML in voxels, or gridpoints. At the start of the PML, farthest from the computational boundary, $i = 1$. As we approach the boundary, i will approach L in integer steps. We will set this auxiliary parameter equal to

$$xn(i) = \frac{\sigma_D(m_i)\Delta t}{2\varepsilon_0}. \tag{50}$$

So that as the auxiliary parameter increases, so will $\frac{\sigma_D(m_i)\Delta t}{2\varepsilon_0}$. We rewrite the f and g functions in Eq. 47 to reflect this definition of the auxiliary paramter,

$$f_{m1}(m_i) = xn(i),\ m \in \{x, y\}, \tag{51a}$$

$$f_{m2}(m_i) = \frac{1}{1 + xn\left(i + \frac{1}{2}\right)},\ m \in \{x, y\}, \tag{51b}$$

$$f_{m3}(m_i) = \frac{1 - xn\left(i + \frac{1}{2}\right)}{1 + xn\left(i + \frac{1}{2}\right)},\ m \in \{x, y\}, \tag{51c}$$

$$g_{m2}(m_i) = \frac{1}{1 + xn\left(i\right)},\ m \in \{x, y\}, \tag{51d}$$

$$g_{m3}(m_i) = \frac{1 - xn(i)}{1 + xn(i)},\ m \in \{x, y\}. \tag{51e}$$

It is important to note the difference between f_{m2}, f_{m3} and g_{m2}, g_{m3}. That is, f_{m2} and f_{m3} evaluate the auxiliary function at half-steps, where

$$xn\left(i + \frac{1}{2}\right) = \frac{1}{3}\left(\frac{i + \frac{1}{2}}{L}\right)^3. \tag{52}$$

A final note is that our choice of how we gradually increased the auxiliary parameter was somewhat arbitrary. The cubic power in Eq. 50 was found empirically [2].

9 Implementation in Python 3

Equation 48 was implemented in standard Python 3 with the Numpy library. Plotting was done with Matplotlib. Appendix E contains the implementation. The following section contains results produced by this implementation.

10 Results

A variety of geometries were simulated with the code in Appendix E. All simulations were done in free-space with $\varepsilon = \varepsilon_0$ and $\sigma = 0$, except for any geometry added. A point source with finite spatial and temporal width offset above the center of the domain was used in all simulations. The pulse just a few timesteps after its maximum is shown in Fig. 2.

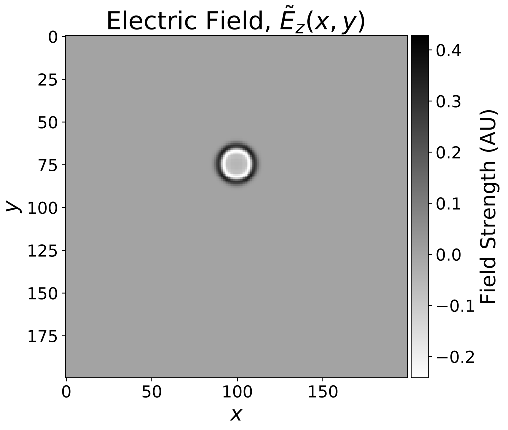

Figure 2: The pulse used in all simulations before interacting with any geometry. Shown here is the pulse a few timesteps after its peak.

10.1 Demonstration of the PML

First, we demonstrate the effectiveness of the PML. Without it, waves would reflect from the boundary, and thus the circular symmetry of the pulse used would not persist without it. To demonstrate that the PML minimizes reflections from the boundary, a free-space simulation was done with a PML at the edge of each boundary, 10 voxels thick. The pulse, several hundred timesteps after its peak, is shown in Fig. 3. This figure demonstrates that reflections from the boundary are sufficiently attenuated by the PML, maintaining the circular symmetry of the pulse in the region of the domain that is not part of the PML.

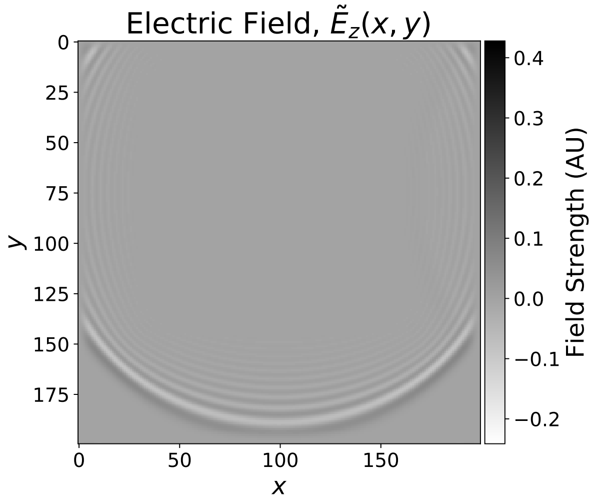

Figure 3: The pulse being attenuated in the PML, preventing erroneous reflections from effecting the domain.

10.2 Reflection from Barrier

A simulation was performed in which a 20 voxel thick barrier with $\varepsilon_r = 2$ and $\sigma = 0.005$ (arbitrary units) was placed across the entire x-axis centered in the domain. Figure 4 shows the waves shortly after the initial interaction with the barrier. This figure demonstrates both the change in propagation speed in the barrier (since $\varepsilon_r > 1$) and the attenuation of the wave in the barrier (since $\sigma > 0$). Moreover, this figure also demonstrates why the ficticious material properties of the PML had to be chosen so particularly, since boundaries with mismatched material properties will cause reflection. Figure 5 shows the pulse long after the initial interaction with the barrier. Parts of the pulse that transmitted through the barrier have been significantly attenuated and have traveled a much shorter distance compared to those that never interacted with the barrier.

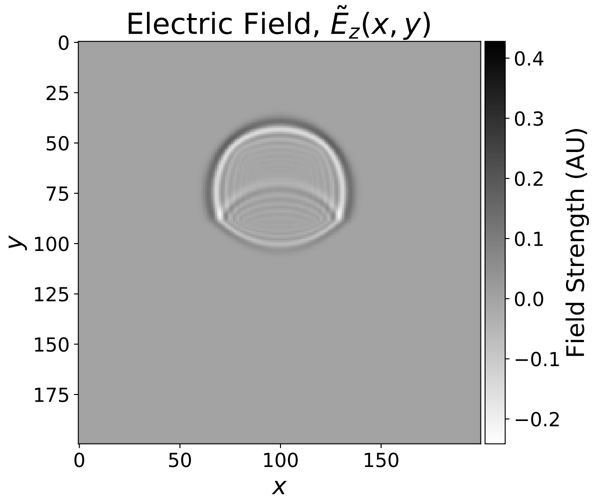

Figure 4: The pulse beginning to propagate through and reflect from a barrier placed across the x-axis, centered in the domain.

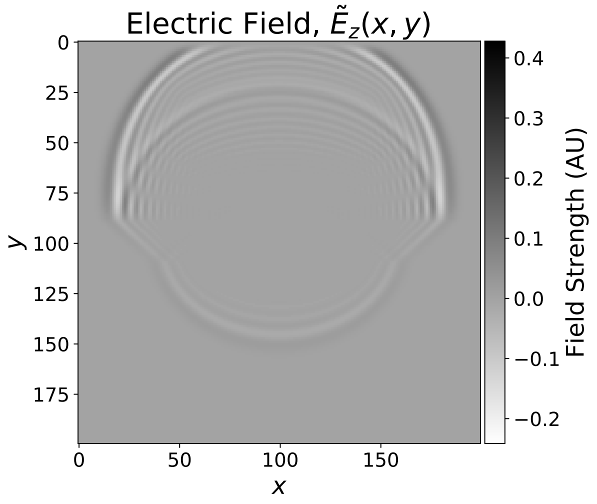

Figure 5: The pulse long after its initial interaction with the barrier placed across the x-axis, centered in the domain.

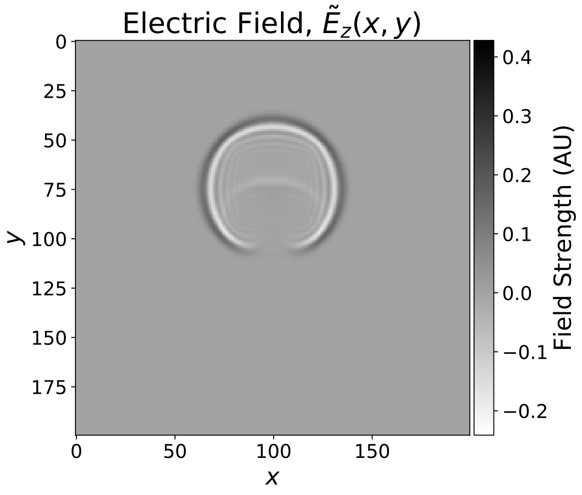

Figure 6: The pulse shortly after interacting with a small square object placed at the origin.

10.3 Diffraction from Small Object

A small object 20 by 20 voxels was placed at the origin to simulate the effects of diffraction. The object's material constants were $\varepsilon_r = 1$ and $\sigma = 0.01$. Figure 6 shows the pulse beginning to interact with the object, effectively removing part of its circular wave. Figure 7 shows the same pulse long after interacting with the object, demonstrating some diffraction and interference effects in the same region that was originally significantly effected.

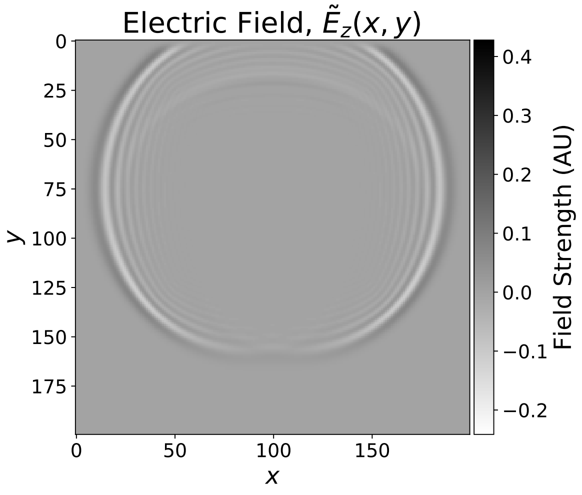

Figure 7: The pulse long after interacting with a small square object placed at the origin, demonstrating some diffraction events on the side opposite the object from the origin of the pulse.

10.4 Single Slit Interference

A classic single slit geometry was simulated as well. A similar barrier to before, except 10 voxel thick and with material constants $\varepsilon_r = 2$ and $\sigma = 0.05$ was placed across the x-axis, centered in the domain. A slit was removed from the center of the barrier, 20 voxels wide. Figure 8 shows the pulse initially interacting with the slit, demonstrating significant interference and diffraction effects as well as minimal transmission into the barrier. Figure 9 shows the pulse long after it has interacted with the slit. The classic central maximum and adjacent minima predicted in single slit interference experiments can be seen in this figure.

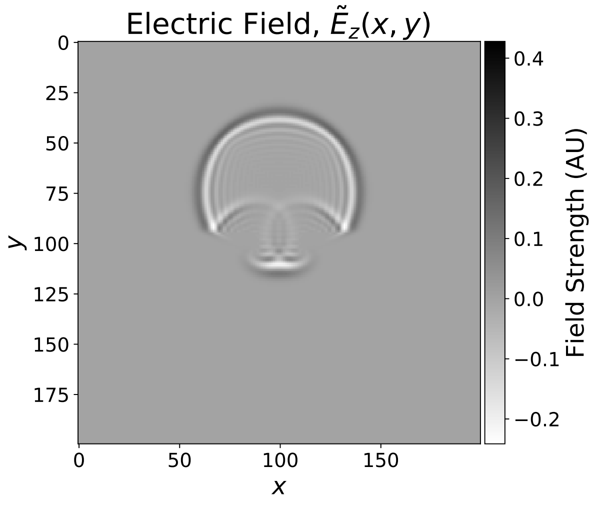

Figure 8: The pulse initially interacting with a single slit in a barrier.

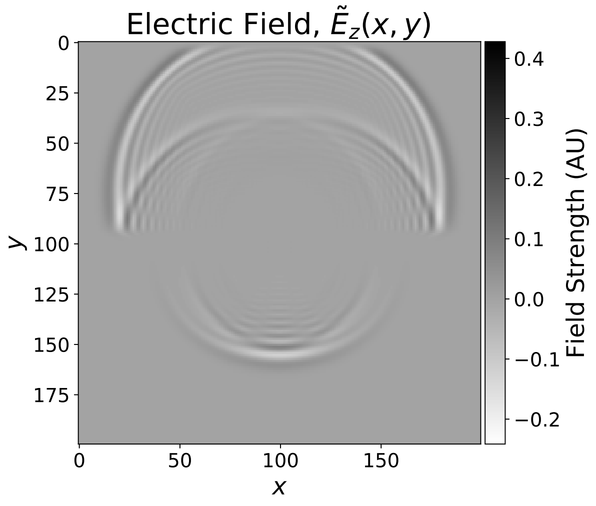

Figure 9: The pulse after its interaction with a single slit, demonstrating interference effects.

10.5 Double Slit Interference

Finally, the famous double slit interference situation was simulated. The same 10 voxel thick, $\varepsilon_r = 2$ $\sigma = 0.05$, barrier as in the previous section was used. Two 10 voxel wide sections were removed, leaving a 10 by 10 voxel block of the barrier in the center of the domain. Figure 10 shows the pulse initially interacting with the two slits, showing resemblance to the initial interaction with the single slit shown in Fig. 8. However, as shown in Fig. 11, many timesteps after the pulse's interaction with the two slits, the interference pattern is significantly more pronounced than in the single-slit case. Many minima and maxima are present in the interference pattern below the two slits, as observed in many experiments before.

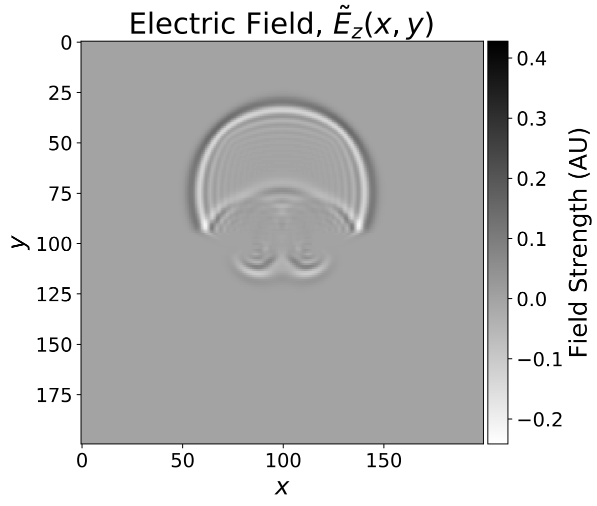

Figure 10: The pulse initially interacting with two nearby slits in the barrier.

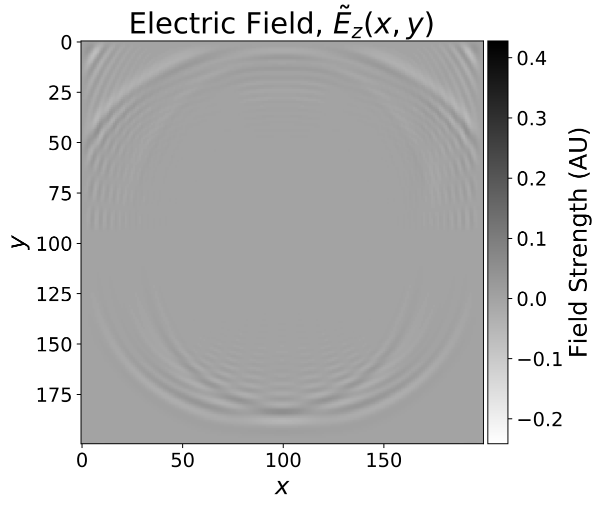

Figure 11: The pulse long after its interaction with the double-slit geometry. Classic interference patterns are present.

11 Conclusion

I have presented the mathematical foundations for the finite-difference time-domain method for electromagnetic wave propagation. An implementation in two-dimensions with a perfectly matched layer was implemented and used to simulate a variety of geometry involving heterogenous lossy dielectric materials. This implementation is provided in Appendix E for anyone who may be insterested in the simulation of electromagnetic waves.

Acknowledgements

I would like to thank F.D.C. Willard for helpful discussions and encouragement regarding this work.

References

[1] J. D. Jackson, *Classical Electrodynamics*. Wiley, 1999.

[2] D. M. Sullivan, *Electromagnetic Simulation Using The FDTD Method.* IEEE Press, 2000.

[3] R. Courant, K. Friedrichs, and H. Lewy, “Über die partiellen differenzengleichungen der mathematischen physik,” *Mathematische Annalen*, vol. 100, no. 1, pp. 32–74, 1928.

[4] J. P. Berenger, “A perfectly matched layer for the absorption of electromag-

netic waves," *Journal of Computational Physics*, vol. 114, pp. 185–200, 1994.

[5] Z. S. Sacks, D. M. Kingsland, R. Lee, and J. F. Lee, "A perfectly matched anisotropic absorber for use as an absorbing boundary condition," *IEEE Transactions on Antennas and Propagation*, vol. 43, pp. 1460–1463, 1995.

[6] M. Kitano, "The vacuum impedance and unit systems," 2006. arXiv:physics/0607056v2.

[7] J. B. Schneider, "Understanding the finite-difference time-domain method." `www.eecs.wsu.edu/~schneidj/ufdtd`, 2010. [Online, accessed 2019].

[8] A. Taflove and S. C. Hagness, *Computational Electrodynamics*. Artech House, 2000.

A Inverse Fourier Transform of $E(\omega)/i\omega$

First, we must develop some Fourier transforms of useful functions. Starting with the constant function, $x(t) = a$. Instead of directly evaluating the transform, we instead look at the inverse transform of $X(\omega)$,

$$\mathcal{F}^{-1}(X(\omega)) = x(t) = a = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{i\omega t}d\omega.$$

Clearly, then $X(\omega)$ must be $2\pi a\delta(\omega)$. So,

$$a \rightleftharpoons 2\pi a\delta(\omega). \tag{53}$$

Now, we examine the signum function,

$$sgn(t) = \begin{cases} +1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0 \end{cases}. \tag{54}$$

Which has the property

$$\frac{d}{dt}sgn(t) = 2\delta(t). \tag{55}$$

Taking the Fourier transform of this function follows, using integration by parts

$$\begin{aligned}\mathcal{F}(sgn(t)) &= \int_{-\infty}^{\infty} sgn(t)e^{-i\omega t}dt \\ &= \frac{-1}{i\omega}sgn(t)e^{-i\omega t}\bigg|_{-\infty}^{\infty} - \int_{-\infty}^{\infty}\frac{-2}{i\omega}\delta(t)e^{-i\omega t}dt \\ &= \frac{2}{i\omega}.\end{aligned}$$

So,

$$sgn(t) \rightleftharpoons \frac{2}{i\omega}. \tag{56}$$

The last function we need is the Heaviside step function,

$$H(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}. \tag{57}$$

Noticing that $H(t) = \frac{sgn(t)+1}{2}$ we take the Fourier transform of H,

$$\begin{aligned}\mathcal{F}(H(t)) &= \mathcal{F}\left(\frac{sgn(t)+1}{2}\right)\\ &= \mathcal{F}\left(\frac{sgn(t)}{2}\right) + \mathcal{F}\left(\frac{1}{2}\right)\\ &= \frac{1}{i\omega} + \pi\delta(\omega).\end{aligned}$$

So,

$$H(t) \rightleftharpoons \frac{1}{i\omega} + \pi\delta(\omega). \tag{58}$$

Now, consider a general function, $f(t)$ which can be written as the integral from $-\infty$ to t of some other function, $g(t)$,

$$f(t) = \int_{-\infty}^{t} g(\tau)d\tau. \tag{59}$$

This can be expressed as a convolution integral with the Heaviside function,

$$f(t) = \int_{-\infty}^{t} g(\tau)d\tau = \int_{-\infty}^{\infty} g(\tau)H(t-\tau)d\tau = (g * H)(t). \tag{60}$$

Fourier's convolution theorem states that for any two functions, $\theta(t)$ and $\varphi(t)$, whose Fourier transforms exist,

$$\mathcal{F}((\theta * \varphi)(t)) = \Theta(\omega)\Phi(\omega). \tag{61}$$

Applying this to our function, $f(t)$, yields,

$$\mathcal{F}(f(t)) = G(\omega)\left[\frac{1}{i\omega} + \pi\delta(\omega)\right]. \tag{62}$$

So,

$$\int_{-\infty}^{t} g(\tau)d\tau \rightleftharpoons G(\omega)\left[\frac{1}{i\omega} + \pi\delta(\omega)\right]. \tag{63}$$

Finally, we can use this identity to find $\mathbf{E}(t)$, given $\mathbf{E}(\omega)$. We apply the above identity to each of the components of $\mathbf{E}(\omega)\left[\frac{1}{i\omega} + \pi\delta(\omega)\right]$. Then,

$$\int_{-\infty}^{t} \mathbf{E}(\tau)d\tau \rightleftharpoons \mathbf{E}(\omega)\left[\frac{1}{i\omega} + \pi\delta(\omega)\right]. \tag{64}$$

We may assume that $\mathbf{E}(t)$ is zero for $t < 0$ and $\lim_{\omega\to 0}\mathbf{E}(\omega) = 0$, in which case the above equation reduces to the desired identity,

$$\int_{0}^{t} \mathbf{E}(\tau)d\tau \rightleftharpoons \frac{\mathbf{E}(\omega)}{i\omega}. \tag{65}$$

B Fourier Transform of df/dt

Given a function $f(t)$ and its Fourier transform $F(\omega)$ with the condition $\lim_{t\to 0} f(t) = 0$, the Fourier transform of its first derivative, df/dt is found with integration by parts,

$$\begin{aligned}\mathcal{F}\left(\frac{df}{dt}(t)\right) &= \int_{-\infty}^{\infty} \frac{df}{dt}(t)e^{-i\omega t}dt \\ &= f(t)e^{-i\omega t}\Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty} f(t)i\omega e^{-i\omega t}dt \\ &= i\omega \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt \\ &= i\omega F(\omega).\end{aligned}$$

Which is the identity,

$$\frac{df}{dt}(t) \rightleftharpoons i\omega F(\omega). \tag{66}$$

C The Lagrange Remainder Theorem

Theorem 1. *Let $f(x)$ be a function which is continuous on $[a,b]$ and $(n+1)$ differientiable on (a,b). Then, for each $x \in [a,b]$ there exists $\eta \in [a,b]$ such that*[2]

$$f(x) = f(a)+f'(a)(x-a)+\frac{f''(a)}{2}(x-a)^2+\cdots+\frac{f^{(n)}(a)}{n!}(x-a)^n+\frac{f^{(n+1)}(\eta)}{(n+1)!}(x-a)^{n+1}. \quad (67)$$

Proof. Let T_n be the nth Taylor polynomial of f, centered at a. That is,

$$T_n(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n. \quad (68)$$

Now choose $K \in \mathbb{R}$ such that

$$f(b) = T_n(b) + K(b-a)^{n+1}. \quad (69)$$

We will show that this K is coefficient of the final term in Eq. 67. First we consider the function F, defined as

$$F(x) = f(x) - T_n(x) - K(x-a)^{n+1}. \quad (70)$$

Notice that $F^{(k)}(a) = 0$ for $k \leq n$ since

$$T_n^{(k)}(x) = f^{(k)}(a) + f^{(k+1)}(a)(x-a) + \cdots + \frac{f^{(n)}(a)}{(n-k)!}(x-a)^{n-k} \text{ for } k \leq n. \quad (71)$$

So $T_n^{(k)}(a) = f^{(k)}(a)$ for $k \leq n$. And,

$$\frac{d^k}{dx^k}K(x-a)^{n+1} = \frac{K(n+1)!}{(n+1-k)!}(x-a)^{n+1-k}. \quad (72)$$

This demonstrates that $F^{(k)}(a) = 0$ for $k \leq n$. Our choice of K guarantees $F(b) = 0$, so we see that $F(a) = F(b) = 0$. This allows us to apply Rolle's theorem to F. The theorem is a special case of the mean value theorem, and it states that for any function, g, that is continuous on $[a,b]$ and differientiable on (a,b), if $g(a) = g(b)$ then there exists at least one $c \in [a,b]$ such that $g'(c) = 0$. Applying this theorem to F tells us that there exists a $c_1 \in [a,b]$ such that $F'(c_1) = 0$. Now we examine F'. We have shown above that $F'(a) = 0$,

[2] The final term in this equation is called the Lagrange remainder of the nth Taylor polynomial of f. This is because it is the error between f and the nth Taylor polynomial of f, T_n. That is to say, $f(x) - T_n(x) = \frac{f^{(n+1)}(\eta)}{(n+1)!}$ for some $\eta \in [a,b]$

and we have just demonstrated that $F'(c_1) = 0$, thus $F'(a) = F'(c_1) = 0$. Applying Rolle's theorem now to F' tells us that there exists $c_2 \in [a, c_1]$ such that $F''(c_2) = 0$. We continue this process on the higher derivatives of F up to $F^{(n+1)}$. Here, Rolle's theorem guarantees the existence of a $c_{n+1} \in [a, c_n]$ such that $F^{(n+1)}(c_{n+1}) = 0$. However, since $T_n^{(n+1)} \equiv 0$, we know that $F^{(n+1)}$ is

$$F^{(n+1)}(x) = f^{(n+1)}(x) - K(n+1)!. \tag{73}$$

So, using the fact that $F^{(n+1)}(c_{n+1}) = 0$,

$$K = \frac{f^{(n+1)}(c_{n+1})}{(n+1)!}. \tag{74}$$

Let $\eta = c_{n+1}$. Since $[a, c_{n+1}] \subseteq [a, c_n] \subseteq [a, c_{n-1}] \subseteq \cdots \subseteq [a, b]$, we know $\eta \in [a, b]$. Moreover, since our choice of K in Eq. 69 could be repeated with b replaced by any $x \in [a, b]$, the entire process could be done with any point in $[a, b]$, yielding a not-necessarily different η for each point. However, each η would still be in $[a, b]$ since $[a, x] \subseteq [a, b] \ \forall \ x \in [a, b]$. Plugging this expression for K back into Eq. 69 shows that for each $x \in [a, b]$, there exists $\eta \in [a, b]$ such that

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \frac{f^{(n+1)}(\eta)}{(n+1)!}(x-a)^{n+1}. \tag{75}$$

□

D Unproven Formula

In [2], the author implicitly states that the inverse Fourier transform of Eq. 45c is

$$\frac{\partial \tilde{D}_z}{\partial t} + \frac{\sigma_D(x)+\sigma_D(y)}{\varepsilon_0}\tilde{D}_z + \frac{(\Delta t)^2\sigma_D(x)\sigma_D(y)}{4\varepsilon_0^2}\frac{\partial \tilde{D}_z}{\partial t} = c_0\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right). \tag{76}$$

Which suggests the identity,

$$\frac{1}{i\omega}\tilde{D}_z \rightleftharpoons \frac{(\Delta t)^2}{4}\frac{\partial \tilde{D}_z}{\partial t}.$$

Not only is this certainly wrong, it fails to even make sense, since Δt should only appear after discretization.

I provide my work below which lead me to this incorrect identity from the author's equations here. The author starts with (Eq. (3.23a) in [2])

$$i\omega\left(1+\frac{\sigma_D(x)}{i\omega\varepsilon_0}\right)\left(1+\frac{\sigma_D(y)}{i\omega\varepsilon_0}\right)\tilde{D}_z(\omega) = c_0\left(\frac{\partial H_y}{\partial x}(\omega) - \frac{\partial H_x}{\partial y}(\omega)\right), \tag{77}$$

which is the same as Eq. 45c used in this work. Using the following definitions (Eq. 3.15 in [2])

$$gi2(i) = \frac{1}{1+\sigma_D(i)\frac{\Delta t}{2\varepsilon_0}}, \tag{78a}$$

$$gi3(i) = \frac{1-\sigma_D(i)\frac{\Delta t}{2\varepsilon_0}}{1+\sigma_D(i)\frac{\Delta t}{2\varepsilon_0}}, \tag{78b}$$

where i is the discretized spatial coordinate, $x_i = i\Delta x$ and $y_j = j\Delta y$, the author claims that the discrete form of Eq. 77 in the time-domain is (rewritten in the notation used in this work)

$$\begin{aligned}\tilde{D}_z(x_i,y_j,t_k) = {} & gi3(x_i)gj3(y_j)\tilde{D}_z(x_i,y_j,t_{k-1}) \\ & + \frac{gi2(x_i)gj2(y_j)}{2}\left[H_y(x_i,y_j,t_k) - H(x_{i-1},y_j,t_k)\right. \\ & \left. - H_x(x_i,y_j,t_k) + H_x(x_i,y_{j-1},t_k)\right].\end{aligned} \tag{79}$$

We will work backwards to un-discretize this equation to see if this is indeed the inverse Fourier transform of Eq. 77. Substituting Eqs. 29, 78a, and 78b and assuming $\Delta x = \Delta y$

yields,

$$\begin{aligned}\tilde{D}_z(x_i, y_j, t_k) = &\left(\frac{1-\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}}{1+\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}}\right)\left(\frac{1-\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}}{1+\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}}\right)\tilde{D}_z(x_i, y_j, t_{k-1}) \\ &+ \left(1+\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}\right)^{-1}\left(1+\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}\right)^{-1}\frac{\Delta t}{\Delta x}c_0\left[H_y(x_i, y_j, t_k)\right. \\ &\left.-H(x_{i-1}, y_j, t_k) - H_x(x_i, y_j, t_k) + H_x(x_i, y_{j-1}, t_k)\right].\end{aligned} \tag{80}$$

The final bracketed term, over Δx, is clearly just the finite differences of $\partial H_y/\partial x$ and $\partial H_x/\partial y$ (with the assumption that $\Delta x = \Delta y$). Knowing this, multiplying both sides by $\frac{1}{\Delta t}\left(1+\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}\right)\left(1+\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}\right)$, and subtracting the $\tilde{D}_z(x_i, y_j, t_{k-1})$ term to the left side gives

$$\begin{aligned}&\frac{1}{\Delta t}\left[\left(1+\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}\right)\left(1+\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}\right)\tilde{D}_z(x_i, y_j, t_k)\right. \\ &\left.-\left(1-\sigma_D(x_i)\frac{\Delta t}{2\varepsilon_0}\right)\left(1-\sigma_D(y_j)\frac{\Delta t}{2\varepsilon_0}\right)\tilde{D}_z(x_i, y_j, t_{k-1})\right] \\ &= c_0\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right).\end{aligned} \tag{81}$$

And so the right side of Eq. 77 is recovered (since $\frac{\partial H_y}{\partial x}(\omega) \rightleftharpoons \frac{\partial H_y}{\partial x}(t)$ and likewise for H_x). So we work only with the left side of Eq. 81 now, which we expect to be the inverse Fourier transform of the left side of Eq. 77. Fully expanding the left side shows

$$\begin{aligned}\rightarrow \frac{1}{\Delta t}&\left[\left(1+\frac{\Delta t}{2\varepsilon_0}\left(\sigma_D(x_i)+\sigma_D(y_j)\right)+\frac{(\Delta t)^2}{4\varepsilon_0^2}\sigma_D(x_i)\sigma_D(y_j)\right)\tilde{D}_z(x_i, y_j, t_k)\right. \\ &\left.-\left(1-\frac{\Delta t}{2\varepsilon_0}\left(\sigma_D(x_i)+\sigma_D(y_j)\right)+\frac{(\Delta t)^2}{4\varepsilon_0^2}\sigma_D(x_i)\sigma_D(y_j)\right)\tilde{D}_z(x_i, y_j, t_{k-1})\right], \\ \rightarrow \frac{\tilde{D}_z(x_i, y_j, t_k)-\tilde{D}_z(x_i, y_j, t_{k-1})}{\Delta t} &+ \frac{\sigma_D(x_i)+\sigma_D(y_j)}{\varepsilon_0}\frac{\tilde{D}_z(x_i, y_j, t_k)+\tilde{D}_z(x_i, y_j, t_{k-1})}{2} \\ &+ \frac{\Delta t\sigma_D(x_i)\sigma_D(y_j)}{4\varepsilon_0^2}\left(\tilde{D}_z(x_i, y_j, t_k)-\tilde{D}_z(x_i, y_j, t_{k-1})\right).\end{aligned}$$

We will use the same fact the author uses,

$$\frac{\tilde{D}_z(x_i, y_j, t_k)+\tilde{D}_z(x_i, y_j, t_{k-1})}{2} = \tilde{D}_z(x_i, y_j, t_{k-1/2}) = \tilde{D}_z(t), \tag{82}$$

meaning that $\tilde{D}_z(t)$ should be the average of the timesteps immediately before and after it. We will also use the definition of the central difference approximation, just as before, to arrive at an expression for this left side in continuous form

$$\rightarrow \frac{\partial \tilde{D}_z}{\partial t} + \frac{\sigma_D(x) + \sigma_D(y)}{\varepsilon_0} \tilde{D}_z(t) + \frac{\sigma_D(x)\sigma_D(y)\,(\Delta t)^2}{4\varepsilon_0^2} \frac{\partial \tilde{D}_z}{\partial t}. \tag{83}$$

Which is the left side of Eq. 76 as claimed. Already this must be incorrect, since we have a Δt in a function which is supposed to be continuous. However, now we take the Fourier transform of this and compare it to the left side of Eq. 77. Using the identity proven in Appendix B, we see that the Fourier transform of Eq. 83 is

$$i\omega \tilde{D}_z(\omega) + \frac{\sigma_D(x) + \sigma_D(y)}{\varepsilon_0} \tilde{D}_z(\omega) + \frac{\sigma_D(x)\sigma_D(y)\,(\Delta t)^2}{4\varepsilon_0^2} i\omega \tilde{D}_z(\omega). \tag{84}$$

To compare it to the left side of Eq. 77, we expand Eq. 77,

$$\rightarrow i\omega \tilde{D}_z(\omega) + \frac{\sigma_D(x) + \sigma_D(y)}{\varepsilon_0} \tilde{D}_z(\omega) + \frac{\sigma_D(x)\sigma_D(y)}{i\omega\varepsilon_0^2} \tilde{D}_z(\omega). \tag{85}$$

If the author's transformation is correct and the above un-discretization and Fourier transform is correct, then Eqs. 84 and 85 should be equivalent. Which suggests,

$$\Delta t = \frac{2}{i\omega}. \tag{86}$$

Clearly, something is wrong here. I cannot make sense of it, and not for lack of trying. The author of [2] has described a PML technique different from most other literature found [7,8]. As such, it is difficult to reconstuct much of the derivations left out of [2]. However, the implementation still works, so we press forward.

E Implementation of Two Dimensional FDTD with PML in Python 3

```
#!/bin/env python3

#============================================================================#
"""
This software is a minimum working example of the 2D FDTD method with PML
as described in the accompanying paper by P. D. Cook.

This program was written for Python 3.7.5.

This software is being provided "as is", without any express or implied
warranty. In particular, the authors do not make any representation or
warranty of any kind concerning the merchantability of this software or its
fitness for any particular purpose.
"""
#============================================================================#


import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from mpl_toolkits.axes_grid1 import make_axes_locatable
from mpl_toolkits.mplot3d import Axes3D

c  = 3E8      # speed of light in meters per second^2

def FDTD2D(media,pulse,Nt,PMLSize = 5):
    """
        FDTD2D takes a Media object, Pulse object, and the number of timesteps.
        Using the Finite Difference Time Domain method, it propagates the
        electric and magnetic fields forward in time.

        Returns:
         Ez - electric field in the z direction as a function of time and space
         Dz - electric displacement in the z direction as a function of time
                and space
         Hx - magnetic field in the x direction as a function of time and space
         Hy - magnetic field in the y direction as a function of time and space
    """

    Ny, Nx = media.shape
    dx     = media.dx
```

```
dt     = media.dt

assert(Nx >= 2*PMLSize and Ny >= 2*PMLSize)

times = np.arange(0,Nt*dt,dt)

# all arrays are indexed as [ti,yi,xi] (unless there is no t component, in
#   which case they are indexed as [yi,xi])
Dz  = np.zeros((Nt,Ny,Nx)) # electric displacement
Ez  = np.zeros((Nt,Ny,Nx)) # electric field
Hx  = np.zeros((Nt,Ny,Nx)) # magnetic field
Hy  = np.zeros((Nt,Ny,Nx)) # magnetic field
Ihx = np.zeros((Nt,Ny,Nx)) # integral of dyE
Ihy = np.zeros((Nt,Ny,Nx)) # integral of dxE
Iz  = np.zeros((Nt,Ny,Nx)) # integral of Ez

# read in the coefficient arrays for the media
if np.isscalar(media.Gaz) and np.isscalar(media.Gbz):
    Gaz = media.Gaz*np.ones((Ny,Nx))
    Gbz = media.Gbz*np.ones((Ny,Nx))
else:
    Gaz = media.Gaz
    Gbz = media.Gbz

# PML arrays
n1 = (1/(3*PMLSize**3))*np.arange(1,PMLSize+1)**3
n2 = (1/(3*PMLSize**3))*(np.arange(1,PMLSize+1)+1/2)**3

fx1 = np.zeros((Ny,Nx))
fx1[:,:PMLSize] = n1[::-1]
fx1[:,-1*PMLSize:] = n1

fx2 = np.ones((Ny,Nx))
fx2[:,:PMLSize] = 1/(1+n2[::-1])
fx2[:,-1*PMLSize:] = 1/(1+n2)

fx3 = np.ones((Ny,Nx))
fx3[:,:PMLSize] = (1-n2[::-1])/(1+n2[::-1])
fx3[:,-1*PMLSize:] = (1-n2)/(1+n2)

gx2 = 1/(1+fx1)
gx3 = (1-fx1)/(1+fx1)

fy1 = np.zeros((Ny,Nx))
fy1.T[:,:PMLSize] = n1[::-1]
```

```
fy1.T[:,-1*PMLSize:] = n1

fy2 = np.ones((Ny,Nx))
fy2.T[:,:PMLSize] = 1/(1+n2[::-1])
fy2.T[:,-1*PMLSize:] = 1/(1+n2)

fy3 = np.ones((Ny,Nx))
fy3.T[:,:PMLSize] = (1-n2[::-1])/(1+n2[::-1])
fy3.T[:,-1*PMLSize:] = (1-n2)/(1+n2)

gy2 = 1/(1+fy1)
gy3 = (1-fy1)/(1+fy1)

# get the pulse and save it
Pz = pulse.getPulse(times)

# FDTD loop
for ti, t in enumerate(times):
    if ti == 0: continue
    # I use a numpy trick here to do h[ti][yi][xi-1], by using np.roll
    # axis = 0 corresponds to the y axis and axis = 1
    #   corresponds to the x axis

    # calculate electric displacement
    Dz[ti] = gx3*gy3*Dz[ti-1]+0.5*gx2*gy2*(Hy[ti-1] - \
        np.roll(Hy[ti-1],-1,axis=1) - Hx[ti-1] + \
        np.roll(Hx[ti-1],-1,axis=0)) \
        + Pz[ti]    # add pulse

    # calculate electric field
    Ez[ti] = Gaz*(Dz[ti]-Iz[ti-1])

    # calculate integral of Ez
    Iz[ti] = Iz[ti-1] + Gbz*Ez[ti]

    # calculate magnetic field
    dxE     = np.roll(Ez[ti],1,axis=1) - Ez[ti]
    Ihy[ti] = Ihy[ti-1] + dxE
    Hy[ti]  = fx3*Hy[ti-1] + 0.5*fx2*dxE + fy1*Ihy[ti]

    dyE     = np.roll(Ez[ti],1,axis=0) - Ez[ti]
    Ihx[ti] = Ihx[ti-1] + dyE
    Hx[ti]  = fy3*Hx[ti-1] - 0.5*fy2*dyE - fx1*Ihx[ti]

# return arrays
```

```
    return Ez, Dz, Hx, Hy

class Media:
    """
        The Media class stores the Gaz and Gbz coefficient
        arrays for the media, which are calculated from
        spacially dependent epsilon_r (relative permittivity) and
        sigma (conductivity) coefficients. It also stores
        the size of the media and the spacings (dx and dt).

        For free space epsilon_r = 1 and sigma = 0.

        The default instance of this class is free space.
    """
    def __init__(self, Nx, Ny, dx, epsilon_r = 1, sigma = 0):

        epsilon_0 = 8.89E-12 # permittivity of free space

        self.shape = (Nx, Ny) # shape of the media
        self.dx    = dx       # spacing
        self.dt    = dx/(2*c) # timestep

        # if epsilon_r and sigma are constant across all space,
        #   then they need to be made into arrays
        if np.isscalar(epsilon_r) and np.isscalar(sigma):
             epsilon_r = epsilon_r*np.ones((Ny,Nx))
             sigma   = sigma*np.ones((Ny,Nx))

        # if only epsilon_r is constant across space,
        #   then it will be made into an array with the
        #   same shape as sigma
        elif np.isscalar(epsilon_r):
            assert(sigma.shape==(Ny,Nx))
            epsilon_r = epsilon_r*np.ones(sigma.shape)

        # if only sigma is constant across space,
        #   then it will be made into an array with the
        #   same shape as epsilon_r
        elif np.isscalar(sigma):
            assert(epsilon_r.shape==(Ny,Nx))
            sigma = sigma*np.ones(epsilon_r.shape)

        # if both epsilon_r and sigma are spacially dependent
        #   then Gaz and Gbz can immediately be calculated
        else:
```

```
        assert(sigma.shape   == (Ny,Nx))
        assert(epsilon_r.shape == (Ny,Nx))

    self.Gaz = 1/(epsilon_r+(sigma*self.dt/epsilon_0))
    self.Gbz = sigma*self.dt/epsilon_0

class Pulse:
    """
        The Pulse class defines a pulse to set as the electric displacement
            in the simulation.
        It creates a 2D gaussian pulse in both time and space
        It is definted by:
            center        - where the pulse will be centered in space, (x0,y0)
            radial_center - where the pulse will peak radially from the center
                                for example:
                                radial_center = 0:          radial_center > 0:

                                --------                    --------
                                --------                    --XXXX--
                                ---XX---                    --X--X--
                                ---XX---                    --X--X--
                                --------                    --XXXX--
                                --------                    --------

            spacial_spread  - the standard deviation of the gaussian in space

            temporal_center - where the peak of the pulse will be in time

            temporal_spread - the standard deviation of the gaussian in time

    """
    def __init__(self, media, center, radial_center, spacial_spread, \
                    temporal_center, temporal_spread):

        self.temporal_center = temporal_center
        self.temporal_spread = temporal_spread

        Ny, Nx = media.shape
        dx     = media.dx
        x0, y0 = center

        x, y    = np.meshgrid(np.linspace(-Nx/2,Nx/2,Nx)*dx, \
                     np.linspace(-Ny/2,Ny/2,Ny)*dx)
        d       = np.sqrt((x-x0)**2+(y-y0)**2)
        self.g_space = np.exp(-( (( d - radial_center )**2) / \
                ( 2.0 * spacial_spread**2 ) ) )

    def getPulse(self, times):
        """
            The getPulse method generates the pulse at the supplied times.
        """

        g_time = np.exp(-(times-self.temporal_center)**2 / \
                    (2 * self.temporal_spread**2) )

        return (g_time*(self.g_space*np.ones((g_time.size,\
                    self.g_space.shape[0],self.g_space.shape[1]))).T).T

def Animate(F, title="2D FDTD", excludePML = 0, cmin = None, cmax = None):
    """
        Animate will animate the propagation of a given field
    """

    if excludePML: F = F[:,excludePML:-excludePML,excludePML:-excludePML]

    Nt, Ny, Nx = F.shape

    fig, ax = plt.subplots(1,1)

    if cmin is None: cmin = np.min(F)/3
    if cmax is None: cmax = np.max(F)/3
    im = ax.imshow(F[0], animated=True, clim=(cmin,cmax),\
                cmap=plt.get_cmap('gist_yarg'))

    def updatefig(ti):
        ti %= Nt
        im.set_array(F[ti])
        return [im]

    ani = animation.FuncAnimation(fig, updatefig, interval=5, blit=True)
    ax.set_title(title,size=24)
    ax.set_xlabel("$x$",size=20)
    ax.set_ylabel("$y$",size=20)
    ax.tick_params(axis='both', which='major', labelsize=16)
    divider = make_axes_locatable(ax)
    cax = divider.append_axes('right', size='5%', pad=0.05)
    cbar=fig.colorbar(im, cax=cax)
    cax.set_ylabel("Field Strength (AU)",size=20)
    cax.tick_params(axis='both', which='major', labelsize=16)
    plt.show()
```

```python
def Energy(Ez,Hx,Hy):
    """
        Calculate the energy in the domain
    """
    Nt, Ny, Nx = Ez.shape
    E_Ez = np.sum(Ez**2, axis=(1,2))
    E_Hx = np.sum(Hx**2, axis=(1,2))
    E_Hy = np.sum(Hy**2, axis=(1,2))
    return E_Ez+E_Hx+E_Hy

################################################################################

media = Media(200,200,0.1)
pulse = Pulse(media, (0*media.dx,-25*media.dx), \
             10*media.dx, 10*media.dx, 10*media.dt, 2*media.dt)
Ez, Dz, Hx, Hy = FDTD2D(media, pulse, 500, PMLSize = 10)
E = Energy(Ez,Hx,Hy)
Animate(Ez,"Electric Field, $\\tilde{E}_z(x,y)$")
```