Institut für Physik
der Karl-Franzens-Universität Graz

Two Dimensional Finite Difference Time Domain Computation of Electromagnetic Fields in Python

Abschlussarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

vorgelegt von

Gernot Ohner

Betreuer: Assoc.-Prof. Dr. Peter Puschnig

Graz, im März 2018

Abstract

In this thesis the finite difference time domain method in Python will be investigated. First some background material will be presented, followed by a discussion of the basic Yee algorithm in one and two dimensions. This treatment is then extended to include lossy materials. For the two dimensional case two implementation of the perfectly matched layer will be introduced and various resulting simulations are shown. The naive implementation based on loops is compared to an implementation based on matrix operations. A trivial simulation is compared to a non-trivial simulation to investigate the possible dependence of the computation time on the field values. Moreover the computation times of the simulation without a perfectly matched layer is compared to that of a simulation with a Bérenger split field PML and a convolutional PML. We find that the computation times of the different implementations scale roughly according to their number of updating fields.

Contents

Chapter 1

Introduction

The finite difference time domain method, in short FDTD, is used to numerically compute the propagation of electromagnetic waves, that is, to solve the Maxwell equations for arbitrary environments. Finite difference methods have long been used in computational fluid dynamic problems by for example von Neumann. In 1966 Kane S. Yee introduced the formulation with staggered grids and half integer indices known today [1]. This greatly increased the range of solvable problems and lead to its widespread adoption and application in a vast range of areas. Further advancement was made with the introduction of first the absorbing boundary conditions (by Mur [2]) and then the perfectly matched layer (by Bérenger [3]). We will treat the latter in Chapter 4.

Numerous books have been written on the topic. Among those are Taflove & Hagness [4] and Inan & Marshall [5] which will both be cited extensively in this thesis.

Advantages and Disadvantages

Before we begin the exploration of the FDTD method, we will try to motivate our choice. Why use the FDTD method? What are its advantages?

Advantages

1. Fields at each lattice point depend only on the neighboring lattice points and not on the entire matrix of field values. Thus, no linear algebra is required which is highly advantageous from a computational performance standpoint.

2. In general previous field values do not need to be stored. Only when dealing with complex materials like dispersive ones or during some boundary treatments situations occur when we need to store previous values of the fields.

3. Once the algorithm is implemented, the treatment of new problems with different environments does not require any analytic calculations, such as the derivation of the Green functions. [4]

4. Matrix implementations of the FDTD method, such as the one described in Chapter 5 are highly amenable to GPU processing [4, p977]. See Section 6.2.3

Disadvantages

1. Due to the explicit nature of the finite difference approximations, the method is only conditionally stable: Stability requires small time steps. See Section 2.3
2. The method suffers from problems of numerical phase anisotropy. This means that the phase velocity of the numerically calculated wave differs from c depending on the direction of propagation on the grid.
3. Avoiding undesired reflections from the boundary of the domain of computation requires potentially costly treatments. See Chapter 4.

Chapter 2

Background

In this chapter some background information will be discussed that makes it easier to introduce the more novel aspects of the FDTD method.

2.1 Maxwell Equations

The Maxwell Equations are the four coupled partial differential equations (first order in space, first order in time) that govern the propagation of electromagnetic waves.

Flux equations For the FDTD method, the flux equations

$$\nabla \boldsymbol{B} = 0$$
$$\nabla \boldsymbol{D} = \rho$$

are not relevant, since either a) a source free medium will be considered, and there the flux equations are implicit in the curl equations [4, p54]; or b) the sources are introduced directly via forced field excitation.

Curl equations In a medium that is linear, isotropic and non-disperisve, but potentially lossy, the Maxwell curl equations simplify to

$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \boldsymbol{E} - \frac{1}{\mu}\sigma^* \boldsymbol{H} \tag{2.1}$$
$$\frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{\epsilon}\nabla \times \boldsymbol{H} - \frac{1}{\epsilon}\sigma \boldsymbol{E} \tag{2.2}$$

These two vector equations are the starting point for the FDTD method.

Splitting vector into scalar fields In Cartesian coordinates equations (2.1) split into 6 scalar equations, one for each direction and field. For the lossless case $\sigma = \sigma^* = 0$ we find

$$\begin{aligned}
\frac{\partial E_x}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right) & \frac{\partial H_x}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) \\
\frac{\partial E_y}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right) & \frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) & \frac{\partial H_z}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right)
\end{aligned} \tag{2.3}$$

Dimensional reduction In this thesis the FDTD method will be treated only in one and two dimensions and Cartesian coordinates. Evidently a reduction from the usual three dimensional space to a two dimensional problem corresponds to a reduction by one dimension. This dimension can be chosen to be z without loss of generality. What this means in the context of the FDTD method is not that this component of the Maxwell equations is simply ignored and the vectors in them reduced to (2,1) vectors. Instead we posit that in this direction nothing ever changes. That is, the fields may have non-zero z components, but all derivatives with respect to z are set to 0. Reduction to two dimensions then yields Equations (2.4).

$$\begin{aligned}
\frac{\partial E_x}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y}\right) & \frac{\partial H_x}{\partial t} &= -\frac{1}{\mu}\frac{\partial E_z}{\partial y} \\
\frac{\partial E_y}{\partial t} &= -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x} & \frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\frac{\partial E_z}{\partial x} \\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) & \frac{\partial H_z}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right)
\end{aligned} \tag{2.4}$$

Modes If we then have either an isotropic medium or an anisotropic medium where ϵ and μ have no off-diagonal elements, the equations decouple into two sets of independent equations. The set of coupled equations consisting of E_x, E_y and H_z is called the transverse electric (TE) mode. Conversey, the set consisting of H_x, H_y and E_z is called the transverse magnetic (TM) mode. Due to the different nature of the continuity equations for the E and H fields [6], the TE and TM mode describe very different physical situations. As an example, Taflove writes [4, pp55]:

> We note that the TE mode sets up the E field lines in a plane perpendicular to the infinitely long axis (the z axis) of the structure. If the structure is metallic, a substantial E field can be supported immediately adjacent and perpendicular to the structure surface without violating the boundary condition of zero E field tangential to a perfectly conducting surface. As a result the TE mode can support propagating electromagnetic fields bound closely to, or guided by, the surface of

a metal structure (the "creeping wave" being a classic example for curved metal surfaces).

The TM mode, on the other hand, does not support surface waves. The presence or absence of surface waves can have important implications for scattering and radiation problems.

Reduction to one dimension modifies the Maxwell curl equations to read

$$\begin{aligned}
&\frac{\partial E_x}{\partial t} = 0 && \frac{\partial H_x}{\partial t} = 0 \\
&\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x} && \frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \\
&\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon}\frac{\partial H_y}{\partial x} && \frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\frac{\partial E_y}{\partial x}
\end{aligned} \tag{2.5}$$

In one dimension, the TE mode consists only of E_y and H_z and the TM mode consists only of H_y and E_z.

2.2 Leapfrog algorithm

The leapfrog algorithm is a method for the numerical solution of hyperbolic partial differential equations (PDEs) It was introduced to avoid the unconditional instability of simpler methods like the first-order Euler method.

Hyperbolic PDE Formally, a hyperbolic PDE is an equation of the form [7].

$$Au_{x,y} + 2Bu_{x,y} + Cu_{y,y} + Du_x + Eu_y + F = 0$$

with

$$\det\left(\begin{bmatrix} A & B \\ B & C \end{bmatrix}\right) > 0$$

Practically this means that it has solutions that behave like waves and disturbances propagate with a finite speed. In fact a so called convection equation of the following form is a prototypical hyperbolic PDE: [5]

$$\frac{\partial V}{\partial t} + v_p\frac{\partial V}{\partial x} = 0$$

That Maxwell's curl equations are, in fact, hyperbolic partial differential equations is the basis for the entire enterprise of the FDTD method.

Leapfrog algorithm for one field When used for the calculation of a single field, the leapfrog method uses second-order centered differences for both spatial and time derivatives. A second order finite difference approximation is described by the following equation:

$$\frac{\partial f(x)}{\partial x} = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} + \mathcal{O}(\Delta x^2) = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2)$$

This finite difference approximation is accurate to $O(\Delta x^2)$, which means that if Δx is halved, the error of the approximation drops to one fourth its previous value. As we can see from the example of an application to convection equation above, the spatial derivative is calculated from the field values at time step n, whereas the time derivative is calculated from the field values at time steps $n-1$ and $n+1$. [5, p57]

$$\frac{V_i^{n+1} - V_i^{n-1}}{2\Delta t} + v_p \frac{V_{i+1}^n - V_{i-1}^n}{2\Delta x} = 0$$

Update equation This equation can then be rearranged to give the update equation for the leapfrog method.

$$V_i^{n+1} = V_i^{n-1} - \left(\frac{v_p \Delta t}{\Delta x}\right)\left[V_{i+1}^n - V_{i-1}^n\right] \tag{2.6}$$

Interleaved Leapfrog algorithm for coupled fields The leapfrog algorithm generalizes very naturally to the treatment of coupled fields. An example might be:

$$\frac{\partial V}{\partial t} + C\frac{\partial J}{\partial x} = 0$$
$$\frac{\partial J}{\partial t} + L\frac{\partial V}{\partial x} = 0$$

Time evolution The time evolution of the interleaved leapfrog algorithm is a two step procedure. First the values of the one field (J) are computed and stored on the basis of the values of the other field (V) previously stored in memory. Then the values of the V field are computed and stored on the basis of the values of J that were just calculated. Then the computation of J begins anew based on the new values of V. This two step process is repeated for the desired number of times. [4, p60] As the derivation is straightforward and analogous to the one above, we will only show the resulting update equations here, using the naming conventions of Inan [5] for the factors C and L.

$$V_i^{n+1} = V_i^{n-1} - \left(\frac{\Delta t}{C\Delta x}\right)\left[J_{i+1}^n - J_{i-1}^n\right] \tag{2.7}$$

$$J_i^{n+1} = J_i^{n-1} - \left(\frac{\Delta t}{L\Delta x}\right)\left[V_{i+1}^n - V_{i-1}^n\right] \tag{2.8}$$

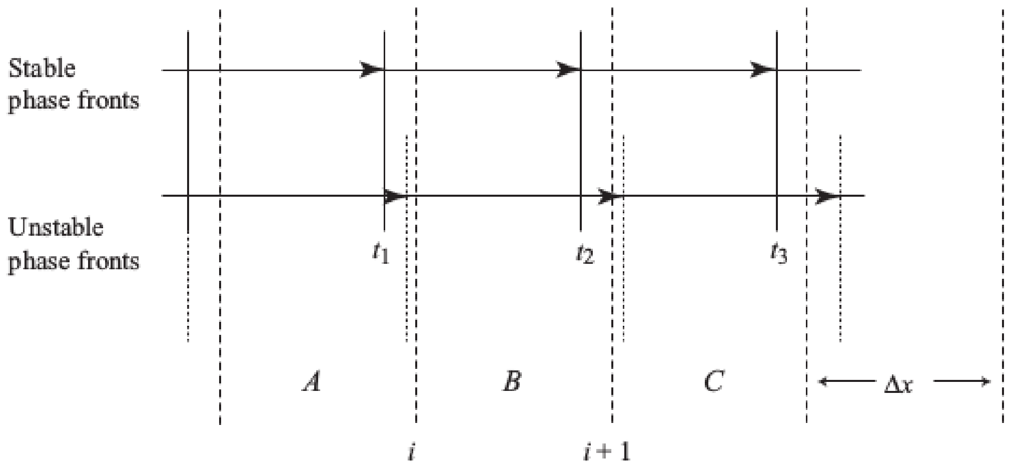


Figure 2.1: A visualization of the CFL criterion in one dimension. Taken from Inan [5, p117].

2.3 Numerical Stability

This type of timestepping is explicit, as opposed to implicit. One of the disadvantages of a fully explicit algorithm is, that, in order for the algorithm to remain stable, the temporal step size is limited by the spatial step size. This condition is called the Courant-Friedrichs-Lewy (CFL) criterion. For the somewhat laborious derivation of the stability criterion the reader is referred to Inan [5, pp132]. Here we only show the result for one dimension (Eq. (2.9)), two dimensions (Eq. (2.10)) and a square grid ($\Delta = \Delta x = \Delta y$) of two dimensions (Eq. (2.11)).

$$\Delta t \leq \frac{\Delta x}{v_p} \tag{2.9}$$

$$\Delta t \leq \frac{1}{v_p\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2}}} \tag{2.10}$$

$$\Delta t \leq \frac{\Delta}{\sqrt{2}v_p} \tag{2.11}$$

In these equations v_p is the phase velocity, which for the convection equation above has the value $v_p = \sqrt{\frac{1}{LC}}$. These results apply both to the convection equations and to the Maxwell curl equations. We note that in the Maxwell equations the factors C and L are replaced by ϵ and μ, respectively, and thus the resulting phase velocity is the speed of light.

Visualization of the CFL criterion Figure 2.1 shows a visual interpretation of the CFL criterion in one dimension. The dotted line represents a wave in a simulation with Δt that is too large and violates the CFL criterion. This results in the possibility that the simulation skips a cell. In the figure, such a situation is visualized. The wave represented by the dotted line is in cell A at time t_1 and in cell C at time t_2, without ever being in cell B. This leads the simulation to be unstable [5, p. 117].

Implicit methods on the other hand evaluate the partial derivatives in a PDE at the time step $n + 1$. This makes them unconditionally stable, removing the CFL limit but has high computational cost [5, p328] as it requires solving a (tridiagonal) matrix equation at each time step.

Chapter 3

Yee Algorithm

The Yee algorithm is an interleaved leapfrog algorithm of the form discussed in Section 2.2. The computed fields E and H are staggered by half a step in space and in time with respect to each other. This reduces the effective time-step to $\frac{1}{2}\Delta t$. The timestepping procedure thus generated is shown in Figure 3.1

The naive starting point for a simulation of two fields is a Cartesian collocated unstaggered grid. In this simplest possible grid both fields, E and H, are defined at the same point on the grid, i.e. "unstaggered" and all components of a field (e.g. B_x and B_y) are defined at the same point on the grid i.e. "collocated". This grid has problems with comparatively large numerical phase-velocity anisotropy, a problem beyond the scope of this thesis which requires the choice of a very fine discretization, leading to long computation times. In 1966 Kane S. Yee introduced the staggered uncollocated grid that is now known as a Yee grid. The Yee grid was a seminal advancement and greatly increased the popularity of the FDTD method.

Discretization details Before the discussion of the Yee grid, some details of the discretization are defined. We define Δx and Δy as the spatial step sizes in x and y direction, respectively. Then, to reference the position (iΔx, jΔy), we use the shorthand (i,j). Due to the staggering of the fields, i and j are *not* necessarily integers, but rational numbers. Furthermore we define n_x and n_y as the number of repetitions of the Yee cell in x and y direction, respectively, that make

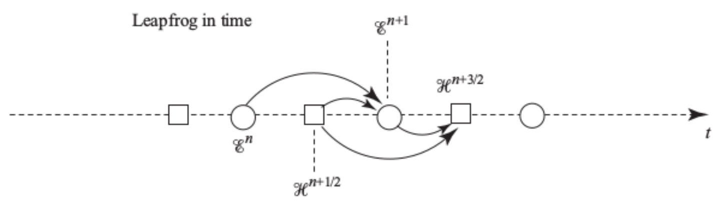


Figure 3.1: Leapfrog time evolution procedure. Taken from Inan [5, p74]

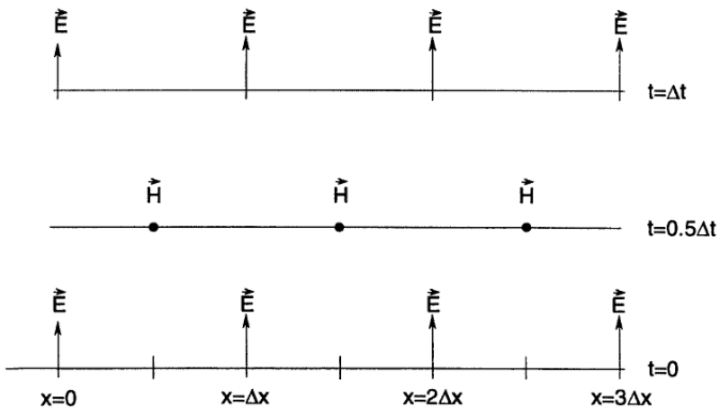


Figure 3.2: Space-time chart of the Yee algorithm for a one-dimensional wave propagation example showing the use of central differences for the space derivatives and leapfrog for the time derivatives. Initial conditions for both electric and magnetic fields are zero everywhere in the grid. Taken from Taflove [4, p61]

up the domain of simulation. That is $nx = \frac{l_x}{\Delta x}$ with l_x the length of the simulation domain in x direction and analogously for y.

3.1 Yee algorithm in one dimension

We start with the Yee grid for a one-dimensional simulation. As mentioned above, the Yee grid is a staggered grid. The electric field E is defined on integer indices $(0, 1, 2...)$ and the H field on half integer indices $(\frac{1}{2}, 1\frac{1}{2}, 2\frac{1}{2}, ...)$. Furthermore the E field is updated at integer time steps and the H field is updated at half integer time steps. Fig. 3.2 shows an illustration of the grid.

Arbitrariness of index choice We note that the above choice of defining E, rather than H, at integer indices was arbitrary and could have been made otherwise. Here it is only important to be consistent. The choice was made because the literature predominantly favors this option as well.

Environmental parameters ϵ and σ must be known at every position of the E field and μ and σ^* must be known at every position of the H field. In the basic Yee algorithm, these parameters can vary arbitrarily.

TM

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

TE

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x}$$
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\frac{\partial E_y}{\partial x}$$

To keep the exposition short, we only show the procedure for the TM mode. We can then apply a finite difference approximation via the leapfrog algorithm. The reason the equations above have only Δt and not $2\Delta t$ as prescribed in the leapfrog algorithm is that they have an effective time step of $\Delta t/2$.

$$\frac{H_z|_{i+\frac{1}{2}}^{n+1/2} - H_z|_{i+\frac{1}{2}}^{n-\frac{1}{2}}}{\Delta t} = -\frac{1}{\mu_{i+\frac{1}{2}}}\frac{\left[E_y|_{i+1}^{n} - E_y|_{i}^{n}\right]}{\Delta x} \tag{3.1}$$

$$\frac{E_y|_{i}^{n+1} - E_y|_{i}^{n}}{\Delta t} = -\frac{1}{\epsilon_i}\frac{\left[H_z|_{i+\frac{1}{2}}^{n+\frac{1}{2}} - H_z|_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right]}{\Delta x} \tag{3.2}$$

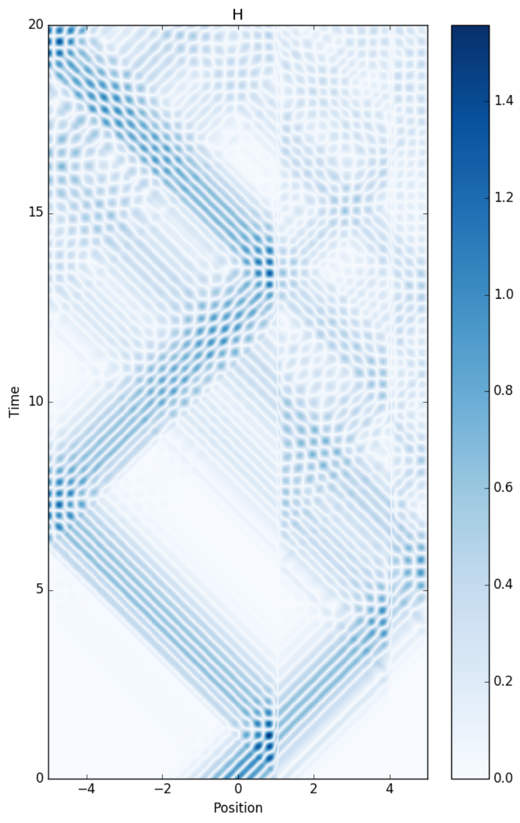
And solving algebraically for the field values at the new time step, $E_y|_i^{n+1}$ and $H_z|_{i+\frac{1}{2}}^{n+1/2}$, we obtain the update equations.

$$H_z|_{i+\frac{1}{2}}^{n+1/2} = H_z|_{i+\frac{1}{2}}^{n-\frac{1}{2}} - \frac{\Delta t}{\mu_{i+\frac{1}{2}}\Delta x}\left[E_y|_{i+1}^{n} - E_y|_{i}^{n}\right] \tag{3.3}$$

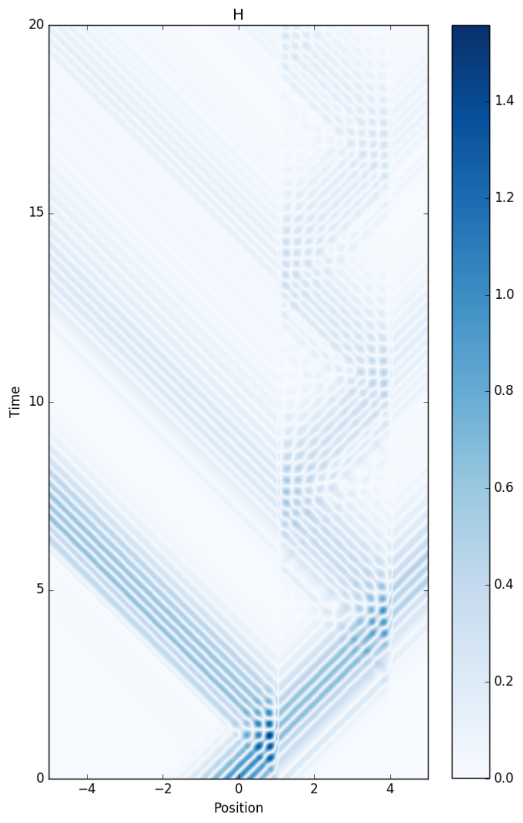
$$E_y|_{i}^{n+1} = E_y|_{i}^{n} - \frac{\Delta t}{\epsilon_i \Delta x}\left[H_z|_{i+\frac{1}{2}}^{n+\frac{1}{2}} - H_z|_{i-\frac{1}{2}}^{n+\frac{1}{2}}\right] \tag{3.4}$$

With these update equations, the propagation in x direction of an electromagnetic wave in a one dimensional wire can be simulated.

Example simulation A resulting simulation can be seen in fig 3.3a The figure shows the propagation of the H_z field component after the wire has been excited with a Gaussian of the form $H(x) = \exp(-ax^2)\sin(bx)$. The x axis represents space and the y axis time. The space from 1.5 to 4 is a medium with $\epsilon = 1.5\epsilon_0$, to showcase the reflection from a boundary between different media. The rest is vacuum. Evidently the wave is not only reflected at the medium but also at the boundary of the computational domain. This is an important problem that will be treated in significant detail. Naturally, for this figure to be created, the field values at every time step were stored in memory. As mentioned before, this is not necessary for the calculation itself but only for the visualization. Figure 3.3b shows the same simulation but with



(a) without ABC



(b) with ABC

Figure 3.3: Result of a 1D simulation using a source of the form $H(x) = \exp\left(-ax^2\right)\sin(bx)$ centered at x = 0. The space from 1.5 to 4 is a medium with $\epsilon = 1.5\epsilon_0$ while the rest is vacuum.

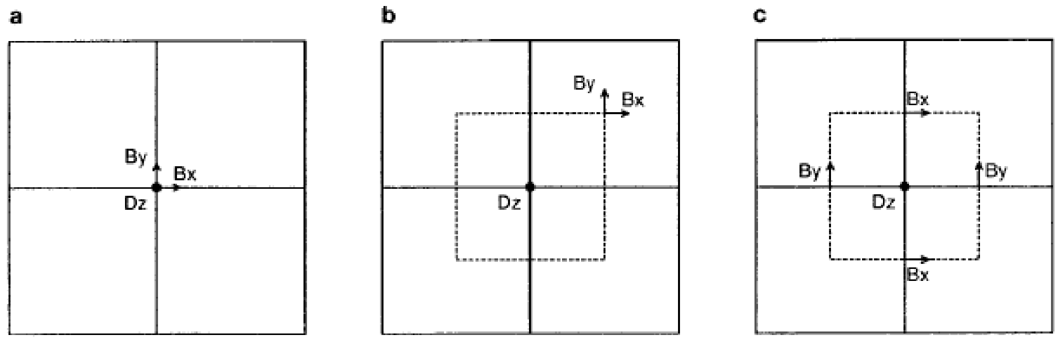


Figure 3.4: Various two dimensional grids of the TM mode. a)A collocated unstaggered grid. b) A collocated staggered grid. c) an uncollocated staggered grid, i.e. a Yee grid. Taken from Liu 1996 who shows D and B instead of E and H [9].

Table 3.1: Positions of field components in the Yee grid

TM	x	y	t
B_x	i	j+1/2	n+1/2
B_y	i+1/2	j	n+1/2
E_z	i	j	n

TE	x	y	t
E_x	i	j+1/2	n
E_y	i+1/2	j	n
H_z	i	j	n+1/2

absorbing boundary conditions [8]. This boundary treatment, that will not be further discussed in this thesis, removes the spurious reflections that occur on the boundary of the computational domain.

3.2 Yee algorithm in two dimensions

We now expand the Yee grid to two dimensions. As discussed in section 2.1 there is a choice between the TE and the TM mode. As mentioned above, the Yee grid is uncollocated, meaning that the components of a field (e.g. E_x and E_y) are not defined at the same positions. Which field components are defined at which indices for each mode is shown in Table 3.1 and illustrated in Fig. 3.5 The mesh spanning the space of the simulation is then made up of a lattice of repetitions of this Yee cell. Analogously to the one dimensional case, the field values at the next time step depend on the surrounding values at the current time step.

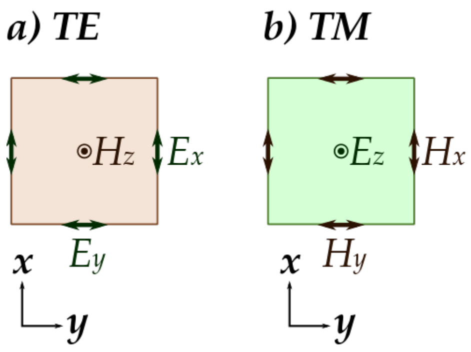


Figure 3.5: Illustration of the two dimensional Yee cells for both modes. Attribution: By FDominec (Own work) [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)], via Wikimedia Commons

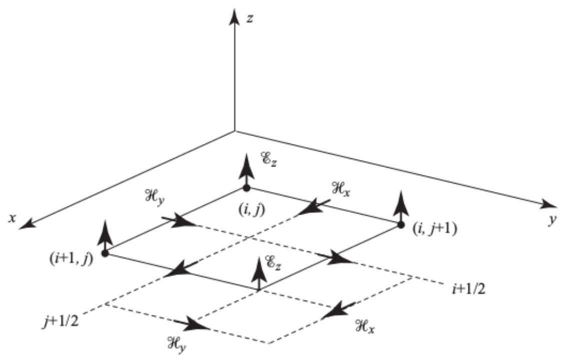


Figure 3.6: An FDTD unit cell for transverse magnetic waves. The small vectors with thick arrows are placed at the point in the mesh at which they are defined and stored. For Example E_z is defined/stored at mesh point (i,j), while H_y is defined/-stored at mesh points $(i + 1/2,\ j)$. Taken from Inan [5, p81]

Two dimensional Yee algorithm The two modes of the Maxwell equations in two dimensions shown in section 2.1 are

TE

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y}\right)$$
$$\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x}$$
$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right)$$

TM

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y}$$
$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x}$$
$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right)$$

Using the same method of discretization in space and in time as for the one dimensional problem above, we can then solve these equations algebraically for $E_x|_{i,j}^{n+1}$, $E_y|_{i,j}^{n+1}$ and $H_z|_{i+\frac{1}{2}}^{n+1/2}$ to obtain the update equations for the TE mode.

$$E_x\big|_{i,j}^{n+1} = E_x\big|_{i,j}^{n} + \frac{\Delta t}{\epsilon_i \Delta y}\left[H_z\big|_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} - H_z\big|_{i,j-\frac{1}{2}}^{n+\frac{1}{2}}\right] \tag{3.5}$$

$$E_y\big|_{i,j}^{n+1} = E_y\big|_{i,j}^{n} - \frac{\Delta t}{\epsilon_i \Delta x}\left[H_z\big|_{i+\frac{1}{2},j}^{n+\frac{1}{2}} - H_z\big|_{i-\frac{1}{2},j}^{n+\frac{1}{2}}\right] \tag{3.6}$$

$$H_z\big|_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1/2} = H_z\big|_{i+\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{\mu_{i,j+\frac{1}{2}} \Delta y}\left[E_x\big|_{i+\frac{1}{2},j+1}^{n} - E_x\big|_{i+\frac{1}{2},j}^{n}\right] - \frac{\Delta t}{\mu_{i+\frac{1}{2},j} \Delta x}\left[E_y\big|_{i+1,j+\frac{1}{2}}^{n} - E_y\big|_{i,j+\frac{1}{2}}^{n}\right] \tag{3.7}$$

3.3 Lossy media in one dimension

In this section the update equations for a potentially lossy material are derived. In this case σ and σ^* may be non-zero for some points in the simulation. The magnetic conductivity σ^* is zero most of the time, but may be non-zero in some materials that show magnetic relaxation effects. [5, p203] The starting point are the Maxwell equations 2.1 shown again here for convenience.

$$\frac{\partial \boldsymbol{H}}{\partial t} = -\frac{1}{\mu}\nabla \times \boldsymbol{E} - \frac{1}{\mu}\sigma^* \boldsymbol{H}$$
$$\frac{\partial \boldsymbol{E}}{\partial t} = \frac{1}{\epsilon}\nabla \times \boldsymbol{H} - \frac{1}{\epsilon}\sigma \boldsymbol{E}$$

splitting the vector fields into scalar fields, a assuming Cartesian coordinate system results in the lossy equivalent of equation (2.3)

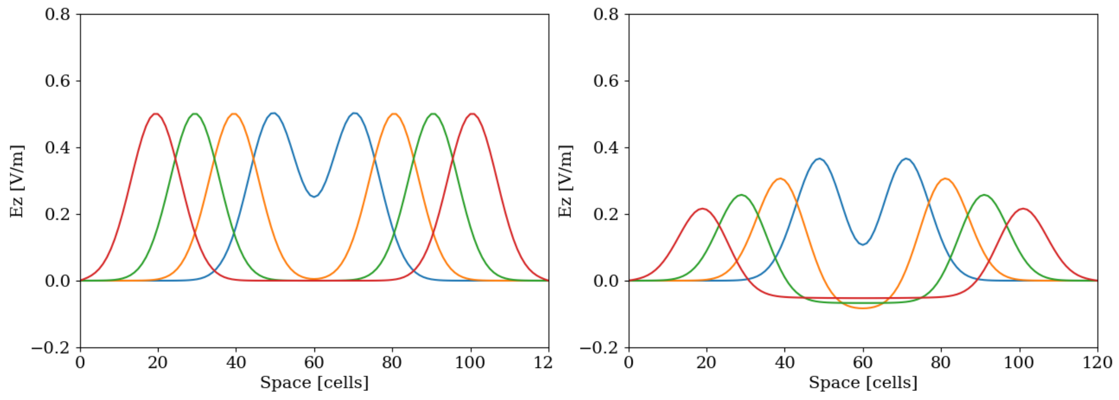
$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right) - \frac{\sigma}{\epsilon}E_x \tag{3.8}$$
$$\frac{\partial E_y}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right) - \frac{\sigma}{\epsilon}E_y \tag{3.9}$$
$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) - \frac{\sigma}{\epsilon}E_z \tag{3.10}$$

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) - \frac{\sigma^*}{\mu}H_x \tag{3.11}$$
$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) - \frac{\sigma^*}{\mu}H_y \tag{3.12}$$
$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right) - \frac{\sigma^*}{\mu}H_z \tag{3.13}$$

Reduction to one dimension As before we reduce to one dimension and chose the TE mode, that is, the equations of E_y and H_z which means that x is the only direction in which something changes.

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial x}\right) - \frac{\sigma}{\epsilon}E_y \tag{3.14}$$
$$\frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x}\right) - \frac{\sigma^*}{\mu}H_z \tag{3.15}$$

Then we apply the finite difference approximations in space and time



(a) without loss, $\sigma = 0$ S/m (b) with loss, $\sigma = 0.005$ S/m

Figure 3.7: A one dimensional simulation a) without and b) with electrical losses. The differently colored lines represent the state of the field at different points in time, specifically after 10 (blue), 20 (orange), 30 (green) and 40 (red) time steps.

$$\frac{E_y|_i^{n+1} - E_y|_i^n}{\Delta t} = -\frac{1}{\epsilon}\left(\frac{H_z|_{i+1/2}^{n+1/2} - H_z|_{i-1/2}^{n+1/2}}{\Delta x}\right) - \frac{\sigma}{\epsilon}E_y|_i^{n+1/2} \tag{3.16}$$

$$\frac{H_z|_{i+1/2}^{n+1/2} - H_z|_{i+1/2}^{n-1/2}}{\Delta t} = -\frac{1}{\mu}\left(\frac{E_y|_{i+1}^n - E_y|_i^n}{\Delta x}\right) - \frac{\sigma^*}{\mu}H_z|_{i+1/2}^n \tag{3.17}$$

Here, the problem arises that the term $\sigma E^{n+1/2}$ requires the value of the electric field at a half integer point where it is not defined, and likewise for $\sigma^* H^n$. This problem is solved by the so-called semi-implicit assumption.

$$E|^{n+1/2} = \frac{E|^{n+1} + E|^n}{2}$$

which has been found to yield numerically stable and accurate results for all non-negative values of σ [4, p64].

$$\frac{E_y|_i^{n+1} - E_y|_i^n}{\Delta t} = -\frac{1}{\epsilon}\left(\frac{H_z|_{i+1/2}^{n+1/2} - H_z|_{i-1/2}^{n+1/2}}{\Delta x}\right) - \frac{\sigma}{\epsilon}\frac{E_y|_i^{n+1} + E_y|_i^n}{2} \tag{3.18}$$

$$\frac{H_z|_{i+1/2}^{n+1/2} - H_z|_{i+1/2}^{n-1/2}}{\Delta t} = -\frac{1}{\mu}\left(\frac{E_y|_{i+1}^n - E_y|_i^n}{\Delta x}\right) - \frac{\sigma^*}{\mu}\frac{H_z|_{i+1/2}^n + H_z|_i^{n-1/2}}{2} \tag{3.19}$$

Table 3.2: Coefficients for lossy media

Update Equation	Lossless C1	Lossless C2	Lossy C1	Lossy C2
E_x, E_y, E_z	1	$\frac{\Delta t}{\epsilon}$	$\frac{2\epsilon-\sigma\Delta t}{2\epsilon+\sigma\Delta t}$	$\frac{2\Delta t}{(2\epsilon+\sigma\Delta t)}$
H_x, H_y, H_z	1	$\frac{\Delta t}{\epsilon}$	$\frac{2\mu-\sigma^*\Delta t}{2\mu+\sigma^*\Delta t}$	$\frac{2\Delta t}{(2\mu+\sigma^*\Delta t)}$

Algebraically isolating the terms $E_y|_i^{n+1}$ and $H_z|_{i+1/2}^{n+1/2}$ respectively yields the update equations. Here adapted from Inan, the equations for the TM mode for the interleaved leapfrog method in lossy media are

$$E_y|_i^{n+1} = \left(\frac{2\epsilon - \sigma\Delta t}{2\epsilon + \sigma\Delta t}\right) E_y|_i^n - \left(\frac{2\Delta t}{(2\epsilon + \sigma\Delta t)\Delta x}\right)(H_z|_{i+1/2}^{n+1/2} - H_z|_{i-1/2}^{n+1/2}) \tag{3.20}$$

$$H_z|_{i+1/2}^{n+1/2} = \left(\frac{2\mu - \sigma^*\Delta t}{2\mu + \sigma^*\Delta t}\right) H_z|_{i+1/2}^{n-1/2} - \left(\frac{2\Delta t}{(2\mu + \sigma^*\Delta t)\Delta x}\right)(E_y|_{i+1}^n - E_y|_i^n) \tag{3.21}$$

In Figure 3.7 a 1D simulation with and without losses is shown. Each colored line represents the value of the electric field at a different time step.

3.4 Lossy media in two dimensions

The same procedure can be applied to derive the update equations for a two dimensional lossy simulation. The TM mode of the Maxwell equations above is taken as a starting point and reproduced here for convenience.

$$\begin{aligned}
\frac{\partial H_x}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) - \frac{\sigma^*}{\mu}H_x \\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) - \frac{\sigma^*}{\mu}H_y \\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) - \frac{\sigma}{\epsilon}E_z
\end{aligned}$$

The approach is perfectly analogous to the one for the one dimensional method. First, the Maxwell equations are transformed from differential equations to difference equations by the leapfrog algorithm, again using the semi-implicit approximation for the loss terms. Then, algebraically solving for the field value at the new time point ($n+1$ for the E field and $n+1/2$ for the H field) yields the update equations.

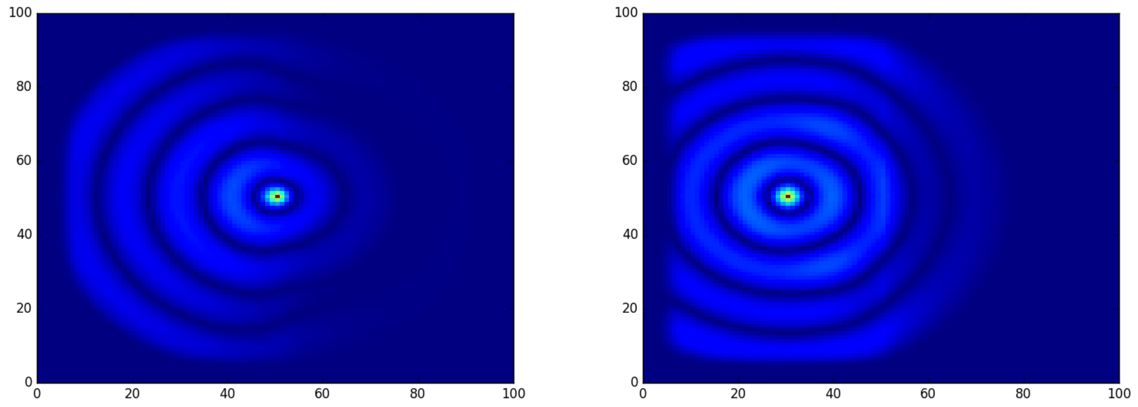


Figure 3.8: 2D simulations of a 100 × 100 grid with a conductivity of $\sigma = 0.01$ S/m on the right half. Plotted is the field value of H_z.

$$H_x|_{i,j+1/2}^{n+1/2} = \left(\frac{2\mu - \sigma^*\Delta t}{2\mu + \sigma^*\Delta t}\right) H_y|_{i,j+1/2}^{n-1/2} - \left(\frac{2\Delta t}{(2\mu + \sigma^*\Delta t)\Delta x}\right)(E_z|_{i,j+1}^{n} - E_z|_{i,j}^{n}) \tag{3.22}$$

$$H_y|_{i+1/2,j}^{n+1/2} = \left(\frac{2\mu - \sigma^*\Delta t}{2\mu + \sigma^*\Delta t}\right) H_y|_{i+1/2,j}^{n-1/2} - \left(\frac{2\Delta t}{(2\mu + \sigma^*\Delta t)\Delta x}\right)(E_z|_{i+1,j}^{n} - E_z|_{i,j}^{n}) \tag{3.23}$$

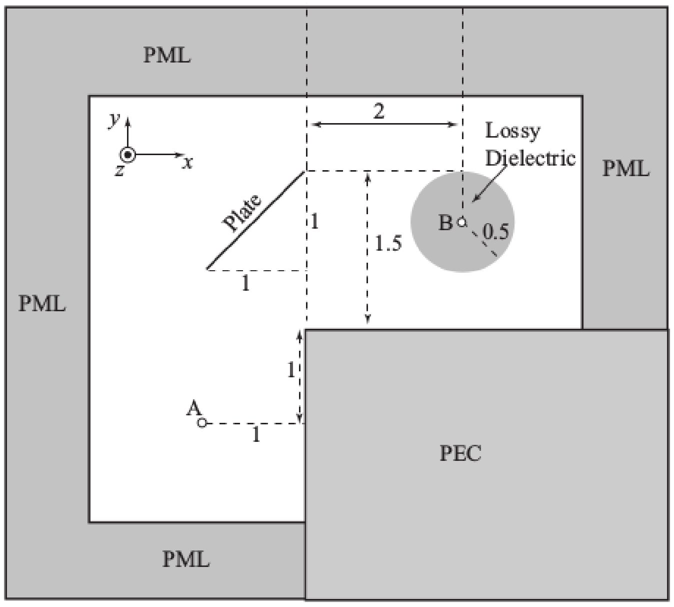
$$E_z|_{i,j}^{n+1} = \left(\frac{2\epsilon - \sigma\Delta t}{2\epsilon + \sigma\Delta t}\right) E_z|_{i}^{n} + \left(\frac{2\Delta t}{(2\epsilon + \sigma\Delta t)\Delta x}\right)(H_x|_{i,j+1/2}^{n+1/2} - H_x|_{i,j-1/2}^{n+1/2}) - (H_y|_{i+1/2,j}^{n+1/2} - H_y|_{i-1/2,j}^{n+1/2}) \tag{3.24}$$

For clarity we show the factors of the update equations in the lossless case compared to the lossy case in Table 3.2. It is easily seen that these equations reduce to the lossless update equations (3.2) if σ and σ^* are 0, as expected.

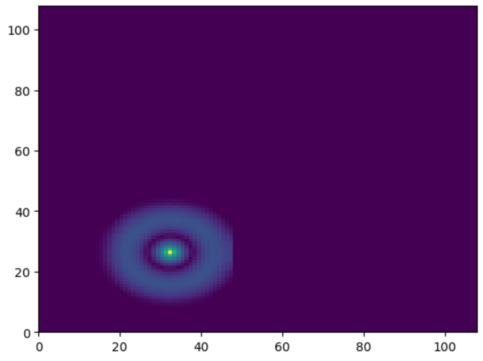
A simulation of the FDTD method for a lossy material is shown in Figure 3.8. In both cases the right half of the simulated domain has $\sigma = 0.01 S/m$. In the left figure the source is placed on the border between the lossy and the lossless material, to show the different behavior. In the right figure the source is placed in the lossless half, and careful examination reveals that the wave is partially reflected at the boundary to the lossy half.

Example of arbitrary environmental parameters In Figure 3.9a a system of scattering objects containing lossy dielectric and perfect electric conductors (PEC) is shown. A sine source was used to excite the field at the source point. This is intended to illustrate the ability of the FDTD method to treat systems with almost arbitrary features. A perfectly matched layer (PML) is used to truncate the domain of computation. This boundary treatment will be discussed in Chapter 4.

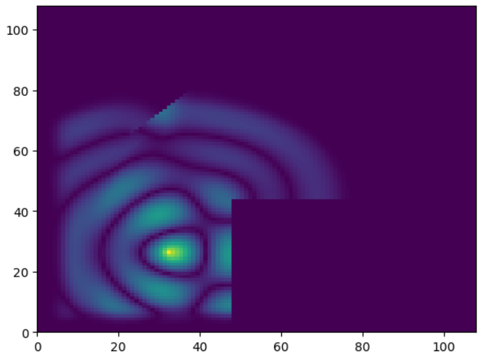
The dimensions are in meters. The plate is made of PEC. The sphere is made of lossy dielectric, with $\epsilon = 4.0\epsilon_0$ and $\sigma = 0.01 S/m$. The rest of the open area is free space. Use $\Delta x = \Delta y = 0.05m$ and $\Delta t = \Delta x/(\sqrt{2}c)$. Also, when modeling, all scatter objects and sources should be at least 10 cells away from the outer boundary. Let the PML be 10 cells thick.



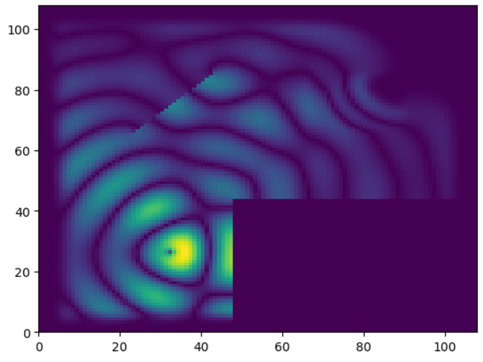
(a) A system of scattering objects. Taken from Inan (9.2.3) [5]



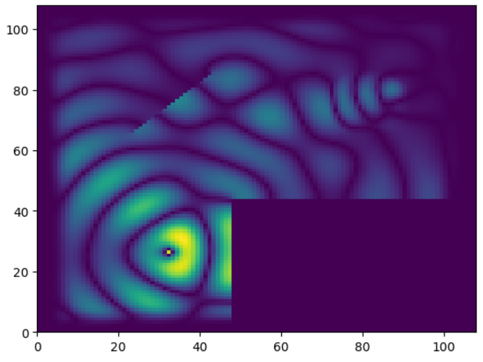
(b) t = 25



(c) t = 75



(d) t = 125



(e) t = 175

Figure 3.9: Snapshots at different time steps of a simulation of the environment posed by Figure 3.9a

Chapter 4

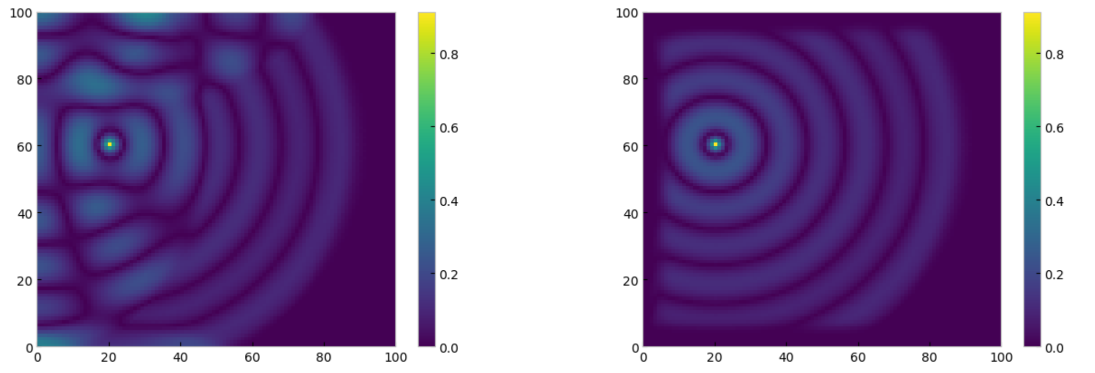
Perfectly Matched Layers

4.1 Boundary Problem

Boundary problem The structure of the Yee cell inevitably leads to a problem with the calculation of the outermost points. Taking a look at the update equations and Figure 3.5, we see that the calculation of e.g. E_x depends on the values of H_z on either side of it. But one of the E_x components in the outermost Yee cell does not have a H_z to one side - a problem that is evidently not solved by adding an additional row of cells at the offending position. There are multiple solutions to this problem.

Absorbing Boundary Conditions One possibility is to use special update equations for these border points that only depend on the fields we actually have in storage (versus usually they would need the H field one point further out). Absorbing boundary conditions essentially allow only the outwardly propagating part of the wave. This works well in one dimension, but runs into difficulties in higher dimensions. It still works, but requires storing past field values which is not desirable. We show the behavior in 1D in Fig 3.3b but will not further discuss them.

Dirichlet Boundary Conditions A simpler approach is to use Dirichlet boundary conditions (in particular with boundary value = 0) and never update the outermost cells of the electric field. This poses a new problem. Since the boundary points now have an electric field value always equal to zero, they act as a perfect electric conductor (PEC) and thus completely reflect incident waves. This is undesirable, as the boundary of the computational domain is supposed to simulate the extension to infinity, rather than an impenetrable wall. While it would be desirable for the domain of computation to be so large that the simulated wave never reaches its limits, this is in general computationally unfeasible. Instead, if the wave reaches the boundary of the computational domain, its extension to infinity should be simulated. [4, p.230] In fact it is desirable to make the domain of computation as small as possible, since the



(a) no PML at t = 100 (b) Bérenger PML at t = 100

Figure 4.1: A comparison of the behavior with and without a perfectly matched layer.

FDTD methods computational complexity scales super-linearly with the number of grid points, as we will discuss in section 6. Consequently Dirichlet boundary conditions are typically used in conjunction with a so called perfectly matched layer (PML).

4.2 General points about the perfectly matched layer

Perfectly matched layers introduce an artificial lossy material at the boundary of the computational domain. This region is, as the name implies, perfectly matched with the neighboring cells, such that no reflection takes place at the boundary of the PML, i.e the reflection coefficient $\Gamma = 0$. With n = $\sqrt{\frac{\mu}{\epsilon}}$, μ and ϵ need to be chosen such that $n_1 = n_2$ to achieve such perfect matching. As mentioned before in Section 3.3 the magnetic losses described by σ^* occur only rarely, and thus the term is usually unphysical and non-zero only within the PML. An analysis of the reflection coefficient shows that this naive approach yields $\Gamma = 0$ only for $\theta = 0$, that is, normal incidence. To obtain a PML formulation that provides satisfactorily low reflection for all angles of incidence, Bérenger [3] introduced the split field formulation that will be discussed in the next section. We note, however, that even in this implementation, undesirably large reflections can occur for near-grazing waves. [4, p294]

In Fig 4.2, taken from Inan [5, p207] we can see the reflection coefficient as a function of the angle of incidence, where $\theta = 0$ is normal incidence.

4.3 Bérengers Split Field PML

In his 1994 paper Bérenger introduced the split field PML. It is based on the idea of an "unphysical absorber", i.e. an area that does not necessarily satisfly Maxwells equations. [8] In the PML region for a 2D TE mode, B_z is split into B_{zx} and B_{zy} with separate values

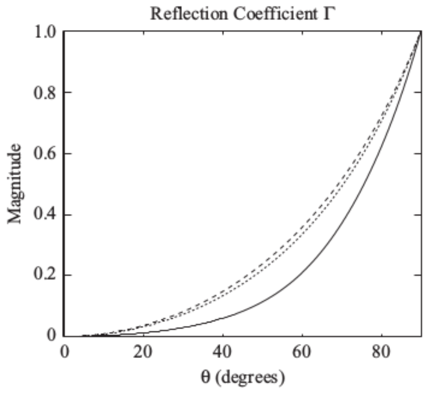
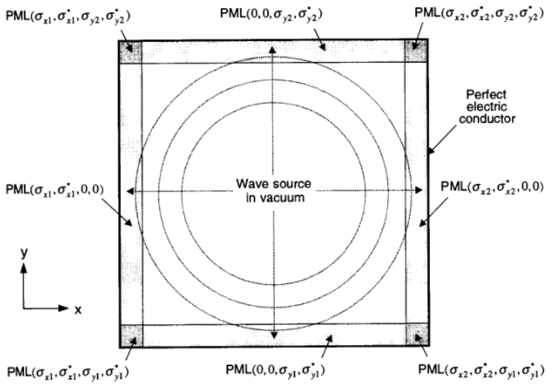


Figure 4.2: The reflection coefficient as a function of the angle of incidence, where $\theta = 0$ is normal incidence. Taken from Inan [5, p205]



PML($\sigma_{x1}, \sigma^*_{x1}, \sigma_{y2}, \sigma^*_{y2}$)
PML($0, 0, \sigma_{y2}, \sigma^*_{y2}$)
PML($\sigma_{x2}, \sigma^*_{x2}, \sigma_{y2}, \sigma^*_{y2}$)
Perfect electric conductor
PML($\sigma_{x1}, \sigma^*_{x1}, 0, 0$)
Wave source in vacuum
PML($\sigma_{x2}, \sigma^*_{x2}, 0, 0$)
y
x
PML($\sigma_{x1}, \sigma^*_{x1}, \sigma_{y1}, \sigma^*_{y1}$)
PML($0, 0, \sigma_{y1}, \sigma^*_{y1}$)
PML($\sigma_{x2}, \sigma^*_{x2}, \sigma_{y1}, \sigma^*_{y1}$)

Figure 4.3: Structure of the loss terms in a 2D simulation with Bérengers PML. Taken from Taflove [4, p280].

of conductivity, σ_x and σ_y acting upon them. In general $\sigma_x \neq \sigma_y$. Only in the corners the equality holds, as illustrated in Fig 4.3. Physical anisotropic media as a PML are possible, e.g. the Uniaxial PML, but are not discussed here. By making the material artificially anisotropic, a solution can be found that sets Γ to 0 for all angles of incidence. We will not show the derivation of this result here and refer the reader to the pertaining literature, i.e. Taflove [4, pp276]. Then the Maxwells equations in two dimensions take the form presented in Inan [5]

$$\epsilon \frac{\partial E_x}{\partial t} + \sigma_y E_x = \frac{\partial H_z}{\partial y} \tag{4.1}$$

$$\epsilon \frac{\partial E_y}{\partial t} + \sigma_x E_x = -\frac{\partial H_z}{\partial x} \tag{4.2}$$

$$\mu \frac{\partial H_{zx}}{\partial t} + \sigma^*_x H_{zx} = -\frac{\partial E_y}{\partial x} \tag{4.3}$$

$$\mu \frac{\partial H_{zy}}{\partial t} + \sigma^*_y H_{zy} = \frac{\partial E_x}{\partial y} \tag{4.4}$$

$$H_z = H_{zx} + H_{zy} \tag{4.5}$$

Then these equations are discretized as usual by using the leapfrog finite difference equations. For the loss term the semi-implicit approximation familiar from Section 3.3 was used again. This results in exactly the same update equations as the basic Yee algorithm for a lossy material except for the splitting of the z component. Below the update equations are shown again, this time fully explicit with regards to the indices, to emphasize the location dependence of ϵ, μ, σ and σ^*.

$$E_x|_{i+1/2,j}^{n+1} = \left(\frac{2\epsilon_{i,j} - \sigma_y|_{i,j}\Delta t}{2\epsilon_{i,j} + \sigma_y|_{i,j}\Delta t}\right) E_x|_{i+1/2,j}^{n} \tag{4.6}$$

$$+ \left(\frac{2\Delta t}{(2\epsilon_{i,j} + \sigma_y|_{i,j}\Delta t)\Delta y}\right) (H_z|_{i+1/2,j+1/2}^{n+1/2} - H_z|_{i+1/2,j-1/2}^{n+1/2}) \tag{4.7}$$

$$E_y|_{i,j+1/2}^{n+1} = \left(\frac{2\epsilon_{i,j} - \sigma_x|_{i,j}\Delta t}{2\epsilon_{i,j} + \sigma_x|_{i,j}\Delta t}\right) E_y|_{i,j+1/2}^{n} \tag{4.8}$$

$$- \left(\frac{2\Delta t}{(2\epsilon_{i,j} + \sigma_x|_{i,j}\Delta t)\Delta x}\right) (H_z|_{i+1/2,j+1/2}^{n+1/2} - H_z|_{i+1/2,j-1/2}^{n+1/2}) \tag{4.9}$$

$$H_{zx}|_{i+1/2,j+1/2}^{n+1/2} = \left(\frac{2\mu_{i,j} - \sigma_x^*|_{i,j}\Delta t}{2\mu_{i,j} + \sigma_x^*|_{i,j}\Delta t}\right) H_{zx}|_{i+1/2,j+1/2}^{n-1/2} \tag{4.10}$$

$$- \left(\frac{2\Delta t}{(2\mu_{i,j} + \sigma_x^*|_{i,j}\Delta t)\Delta x}\right) (E_y|_{i+1,j+1/2}^{n} - E_y|_{i,j+1/2}^{n}) \tag{4.11}$$

$$H_{zy}|_{i+1/2,j+1/2}^{n+1/2} = \left(\frac{2\mu_{i,j} - \sigma_y^*|_{i,j}\Delta t}{2\mu_{i,j} + \sigma_y^*|_{i,j}\Delta t}\right) H_{zy}|_{i+1/2,j+1/2}^{n-1/2} \tag{4.12}$$

$$+ \left(\frac{2\Delta t}{(2\mu_{i,j} + \sigma_y^*|_{i,j}\Delta t)\Delta y}\right) (E_x|_{i+1/2,j+1}^{n} - E_x|_{i+1/2,j}^{n}) \tag{4.13}$$

We note that in three dimensions, not only the z component but also all other components would have to be split, leading to twelve/eighteen equations rather than the usual six for three dimensions.

4.4 Convolutional Perfectly Matched Layer

The Bérenger PML and its physical cousin, the UPML are highly effective at terminating the computational domain, but have the disadvantage that they fail at treating evanescent waves. According to Inan, the reason for this is that it is only ”weakly causal” because the loss term in the semi-implicit approximation depends on the field value at $n + 1$. To solve this problem strongly causal perfectly matched layers have been introduced. The most popular implementation is the so called convolutional PML or CPML.

Update Equations The only way the update equations of the CPML are modified with respect to the basic Yee algorithm is the addition of an auxiliary variable [5, p227]. For simplicity we give the lossless update equations.

$$E_x\big|_{i,j}^{n+1} = E_x\big|_{i,j}^{n} + \frac{\Delta t}{\epsilon \Delta y}\left[H_z\big|_{i,j+\frac{1}{2}}^{n+\frac{1}{2},j} - H_z\big|_{i,j-\frac{1}{2}}^{n+\frac{1}{2}}\right] - \frac{\Delta t}{\epsilon}\Psi_{ey} \tag{4.14}$$

$$E_y\big|_{i,j}^{n+1} = E_y\big|_{i,j}^{n} - \frac{\Delta t}{\epsilon \Delta x}\left[H_z\big|_{i+\frac{1}{2},j}^{n+\frac{1}{2},j} - H_z\big|_{i-\frac{1}{2},j}^{n+\frac{1}{2}}\right] + \frac{\Delta t}{\epsilon}\Psi_{ex} \tag{4.15}$$

$$H_z\big|_{i+\frac{1}{2},j}^{n+1/2} = H_z\big|_{i+\frac{1}{2},j}^{n-\frac{1}{2}} + \frac{\Delta t}{\mu \Delta y}\left[E_x\big|_{i+1,j}^{n} - E_x\big|_{i,j}^{n}\right] - \frac{\Delta t}{\mu \Delta x}\left[E_y\big|_{i+1,j}^{n} - E_y\big|_{i,j}^{n}\right] + \frac{\Delta t}{\mu_0}\left(\Psi_{hy} - \Psi_{hx}\right) \tag{4.16}$$

where the Ψ terms are given by

$$\Psi_{ex}\big|_{i+1/2,j}^{n+1/2} = b_e y \psi_{ex}\big|_{i+1/2,j}^{n-1/2} + a_e y \frac{H\big|_{i+1/2,j+1/2}^{n+1/2} - H\big|_{i+1/2,j-1/2}^{n+1/2}}{\Delta y} \tag{4.17}$$

$$\Psi_{ey}\big|_{i,j+1/2}^{n+1/2} = b_e x \psi_{ey}\big|_{i,j+1/2}^{n-1/2} + a_e x \frac{H\big|_{i+1/2,j+1/2}^{n+1/2} - H\big|_{i-1/2,j+1/2}^{n+1/2}}{\Delta x} \tag{4.18}$$

$$\Psi_{hx}\big|_{i+1/2,j}^{n+1} = b_h y \psi_{hx}\big|_{i+1/2,j}^{n} + a_h y \frac{E\big|_{i+1,j+1/2}^{n} - E\big|_{i,j-1/2}^{n+1}}{\Delta y} \tag{4.19}$$

$$\Psi_{hy}\big|_{i+1/2,j}^{n+1} = b_h x \psi_{hy}\big|_{i+1/2,j}^{n} + a_h y \frac{E\big|_{i+1/2,j+1}^{n} - E\big|_{i+1/2,j}^{n+1}}{\Delta x} \tag{4.20}$$

and the constants a_i and b_i are given by

$$a_i = \frac{\sigma_i}{\sigma_i + \alpha_i}(e^{-(\sigma_i+\alpha_i)(\Delta t/\epsilon_0)} - 1) \tag{4.21}$$

$$b_i = e^{-(\sigma_i+\alpha_i)(\Delta t/\epsilon_0)} \tag{4.22}$$

Advantages of the CPML Unlike the Bérenger Split Field PML and the UPML, the CPML does not alter the update equations by modifying the prefactors cx1 etc. Instead the CPML term is additive. This means that it can be applied to any existing FDTD implementation, be it dispersive, anisotropic or non-linear. Moreover, Gvozdic et al [10] compare the performance of the UPML and CPML and find that the CPML exhibits a significantly lower reflection coefficient.

Disadvantages of the CPML The computation time of the CPML will be compared to the Bérenger Split Field PML and the basic implementation without a PML in Section 6.2.2. We find a substantial increase in the required computation time. Furthermore the CPML has 7 instead of 3 updating fields which leads to increased storage requirements. However various improvements to storage needs are possible. The Ψ need only be known in the PML region. The α-s, κ-s, and b-s can be stored in 1D arrays rather than 2D arrays, because e.g. b_x is invariant with respect to translation on the y axis and vice versa.

4.5 Technical points

Here we use the value of $\sigma_{\max}$ as given by Taflove [4, p293].

$$\sigma_{\max} = \frac{-\log(R_0)\cdot c\cdot(m+1)}{2\sqrt{\frac{\mu}{\epsilon}}\cdot d\cdot\Delta x}$$

Here R_0 is the desired reduction factor - level of reflection on the boundary of the computational domain deemed acceptable, for example 1e-6. m determines the steepness of the grading. Empirically it has been found that a value between 2 and 4 is most suitable. d is the thickness of the PML which is usually chosen to be 10 cells.

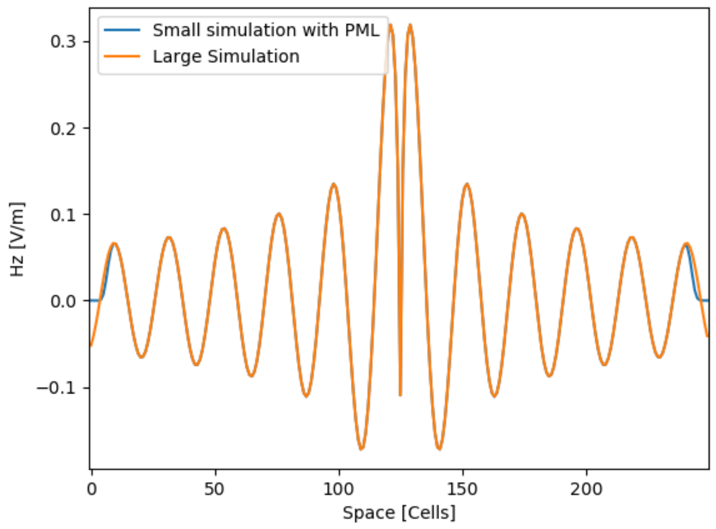
Grading the PML When testing an implementation of a PML, one finds that the actual reflection coefficient is significantly larger than the desired one, as specified by the reduction factor R_0 . The reason for this is the abrupt change from $\sigma = 0$ to a comparatively large value, i.e. one that is sufficient to reduce the fields by a factor of around 10^6 within 20 cells. Furthermore Inan [5, p210] notes, that due to the staggering of the fields, the electric and the magnetic component of the wave will not encounter the PML boundary at the same point in space. To reduce the reflection from the boundary of the supposedly perfectly matched layer, there is the possibility of grading the layer, such that σ does not change from 0 to σ_{max} in one step but gradually. Theoretically there are an infinite number of possible grading schemes, but polynomial grading and geometric grading are most often used. [4, p211] Choosing polynomial grading, the value of σ n steps from the boundary, taken from Taflove, is:

$$\sigma_n = \sigma_{\max}\left(\frac{t-n}{t}\right)^m \tag{4.23}$$

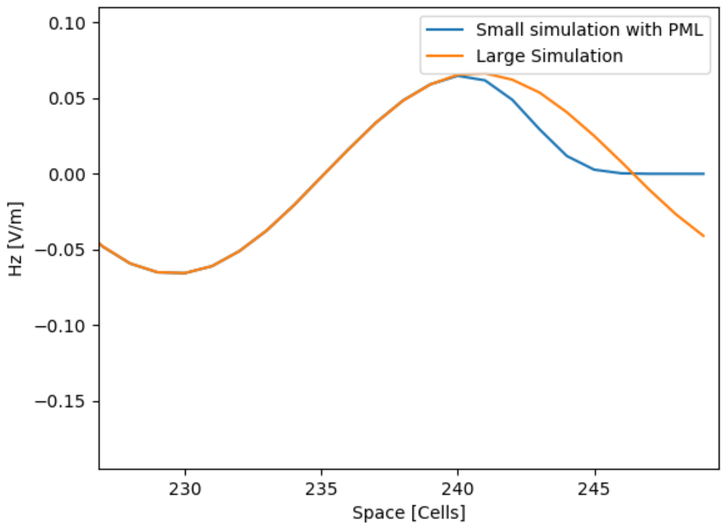
For the CPML, not only σ and σ^* but also α_i is graded. We note that α_i is graded in the opposite manner to σ, i.e. it is 0 at the boundary of the computational domain and rises inwards to a maximum value at the PML boundary. Even a properly graded PML will not provide a reduction to the specified factor R_0, as Inan notes: "The theoretical reflection coefficient [...] is based on an analytical treatment of the PML as a continuous lossy medium with thickness d; however, the discretization process limits the performance of the PML." [5, p212].

A potential pitfall in the scaling of the PML Careful attention has to be paid to correctly applying the grading of the PML. A mistake here can lead to a simulation that appears to work, but in fact has reflections orders of magnitudes bigger than it specified.

Testing a PML To test if a PML is actually working as intended, we follow a 2D analogue of the procedure Gvozdic et al [10] use. Two simulations are run: one small with $(n_x,\, n_y) = (250,\, 250)$ and the other large with $(n_x,\, n_y) = (750,\, 750)$. At $t = 300$, that is sufficient time for the wave to be reflected at the boundary of the computational domain of the smaller simulation a snapshot is taken of the behavior of both simulations. If the PML works there should be no difference in the field values of the two simulations, since the point of the PML is to simulate the propagation out of the computational domain, which is just what happens in the larger simulation. Such a snapshot is shown in Fig. 4.4a and zoomed into the region of interest in Fig. 4.4b.



(a) A cut-through of two simulations used to test the efficacy of the PML. The large simulations represents the ideal case of propagation past the computational domain. The small simulation using a PML should come as close as possible to this ideal.



(b) A zoom into the region of interest of Fig. 4.4a. The PML starts at cell 240.

Figure 4.4: Visualizations of the efficacy of a PML.

Chapter 5

Implementation

In this chapter the steps that need to be taken to proceed from the update equations obtained in the previous chapter to an actual implementation in code will be considered. In particular there are some complications that arise when trying to translate the notion of a half-integer index to a programming implementation. The implementations are shown in python. The reader is reminded that python uses zero-based arrays.

The index problem As arrays do not support half-integer indices, some adjustments have to be made. A mapping has to be applied that shifts the indices in a way that is consistent, does not create holes in the array and does not lead to double occupation. Preferably, it should also lead to a pattern of assignment that allows the update equations to be the same for all points on the lattice. This is possible for the Yee grid, but not, for example, for a hexagonal grid. Fortunately finding a mapping that adheres to these requirements is easier than it sounds and can be achieved by two simple rules. In the following x stands for any index, i.e. $x \in i, j$. These two rules solve the index problem. In the update equations of the E field,

- The indices of the E field stay the same.
- For the indices of the H field, x+1/2 is mapped to x
- For the indices of the H field, x-1/2 is mapped to $x - 1$

In the update equations of the H field,

- The indices of the H field stay the same.
- For the indices of the H field, $x + 1/2$ is mapped to $x + 1$
- For the indices of the H field, $x - 1/2$ is mapped to x

These rules are shown in Table 5.1

Table 5.1: A solution to the index problem in two dimensions

Field	**Yee index**	**array index**
H	$x + 1/2$	x
H	$x - 1/2$	$x - 1$
E	$x + 1/2$	$x + 1$
E	$x - 1/2$	x

Field sizes One of the main advantages of the FDTD method is that it does not require the storage of previous field values. This means that the used arrays can be of size (n_x, n_y) instead of (n_x, n_y, n_t). More specifically, inspection of Fig 3.5 shows that not all field components have the above number of positions on which they are defined. This a concern that will become particularly relevant when considering the implementation in a programming language, which is why we refer to the number of defined positions of a component as its "size". Specifically we see that the E_x field has n_x entries in x direction, but $n_y + 1$ entries in y direction. Analogously Ey has the size $(n_x + 1, n_y)$. Conversely, in the TM mode the H_x and H_y of size $(n_x, n_y + 1)$ and $(n_x + 1, n_y)$, respectively. Then H_z in the TE mode and E_z in the TM mode simply the size (n_x, n_y).

Code of the loop implementation The code for the Yee algorithm in two dimension for the TE mode in the implementation with loops then is:

```
for t in range(nt):

    for i in range(nx):
        for j in range(1,ny-1):
            ex[i,j] = ex[i,j] + cex * (hz[i,j] - hz[i,j-1])

    for i in range(1,nx-1):
        for j in range(ny):
            ey[i,j] = ey[i,j] - cey * (hz[i,j] - hz[i-1,j])

    for i in range(nx):
        for j in range(ny):
            hz[i,j] = hz[i,j] + chy*(ex[i,j+1] - ex[i,j])
                              - chx*(ey[i+1,j] - ey[i,j])
    hz[sourcepoint] = source(t)
```

Where $cex = \frac{\epsilon \Delta t}{\Delta x}$, $cey = \frac{\epsilon \Delta t}{\Delta y}$, $chx = \frac{\mu \Delta t}{\Delta x}$ and $chy = \frac{\mu \Delta t}{\Delta y}$.

Code of the matrix implementation An alternative, much more popular and computationally more efficient implementation is based on matrix manipulation packages such as numpy in python or eigen in c++. The reader is assumed to be familiar with the slicing notation in python. This considerably faster and more compact implementation based on numpy is shown below.

```
for t in range(nt):
    ex[:,1:-1] = ex[:,1:-1] + cex * (hz[:,1:] - hz[:,:-1])
    ey[1:-1,:] = ey[1:-1,:] - cey * (hz[1:,:] - hz[:-1,:])
    hz = hz - chx*(ey[1:,:]-ey[:-1,:]) + chy*(ex[:,1:]-ex[:,:-1])
    hz[sourcepoint] = source(t)
```

Inan claims that the matrix implementation takes a 1/10th of the computation time of the loop implementation. This result could not be reproduced. The speedup of the matrix implementation was measured via a simulation for a 100×100 lattice for 200 timesteps. This simulation was repeated 100 times. Defining the minimum computation time of the matrix implementation as the base case, the unoptimized loop implementation shown in Section 3.1 was found to have a minimum computation time 215 times that of the base case.

Code of the implementation with a Convolutional PML The convolutional PML discussed in the previous chapter is slightly more complicated and thus code for its implementation is shown below. This code was used in the comparison between the computing times of the PML implementations discussed in Section 6.2.2.

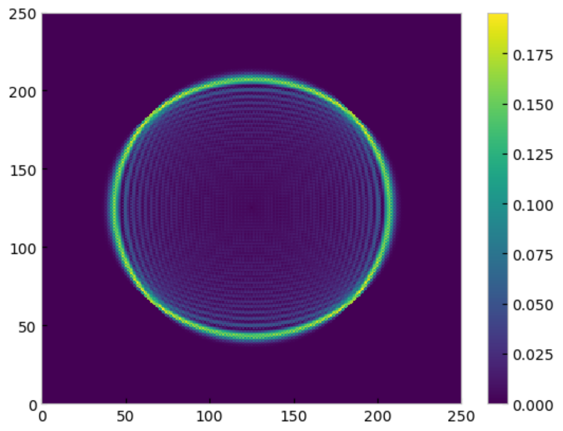
```
k = 1
alpha_x = np.ones((nx-1, ny))
alpha_y = np.ones((nx, ny-1))
alpha_mx = np.ones((nx, ny))
alpha_my = np.ones((nx, ny))

for i in range(10):
    alpha_x[i, :] = (i + 1) / 10
    alpha_x[-i - 1, :] = (i + 1) / 10
    alpha_mx[i, :] = (i + 1) / 10
    alpha_mx[-i - 1, :] = (i + 1) / 10
    alpha_y[:, i] = (i + 1) / 10
    alpha_y[:, -i - 1] = (i + 1) / 10
    alpha_my[:, i] = (i + 1) / 10
    alpha_my[:, -i - 1] = (i + 1) / 10
```

```
bex = np.exp(-(sigma_x/(k + alpha_x)) * (dt / const.epsilon_0))
bey = np.exp(-(sigma_y/(k + alpha_y)) * (dt / const.epsilon_0))
bhx = np.exp(-(sigma_mx/(k + alpha_mx)) * (dt / const.epsilon_0))
bhy = np.exp(-(sigma_my/(k + alpha_my)) * (dt / const.epsilon_0))

aex = (bex - 1) / dx
aey = (bey - 1) / dy
ahx = (bhx - 1) / dx
ahy = (bhy - 1) / dy

ex, ey, hz = fields
pex, pey, phx, phy = aux_fields
cex, cey, chy, chx, px, py, pm = constants

for t in range(time steps):
    phi_hy = bhy * phi_hy + ahy * (ex[:, 1:] - ex[:, :-1])
    phi_hx = bhx * phi_hx + ahx * (ey[1:, :] - ey[:-1, :])
    hz = hz - chy * (ey[1:, :] - ey[:-1, :])
            + chx * (ex[:, 1:] - ex[:, :-1])
            + pm * (phi_hy - phi_hx)
    hz[sourcepoint] = source(t)
    phi_ey = bey * phi_ey + aey * (hz[:, 1:] - hz[:, :-1])
    phi_ex = bex * phi_ex + aex * (hz[1:, :] - hz[:-1, :])
    ex[:, 1:-1] = ex[:, 1:-1]
                + cex * (hz[:, 1:] - hz[:, :-1]) + px * phi_ey
    ey[1:-1, :] = ey[1:-1, :]
                - cey * (hz[1:, :] - hz[:-1, :]) - py * phi_ex
```

Sources

Finding a non-trivial solution usually involves sources. There are numerous possible ways of doing that. One option for adding a source to the computation is in the form $E(i, j) = source$ [5]. This causes a small problem because it makes the source cell a perfect electric conductor (PEC) that therefore causes reflections. If this problem is relevant enough to warrant a different approach depends on the aim of the simulation. A different approach is an additive source of the form $E(i, j) = E(i, j) + source$

Source startup It is advisable to use a source that does not "turn on" too fast, as this will have spurious effects. To be specific, it will create superluminal waves. This field jitter is recognizable by its lack of rotational symmetry - it has corners at 45°, 135°, 225° and 315° [4,


(a) Superluminal waves from a source that turns on immediately. Notice the lack of circular symmetry.

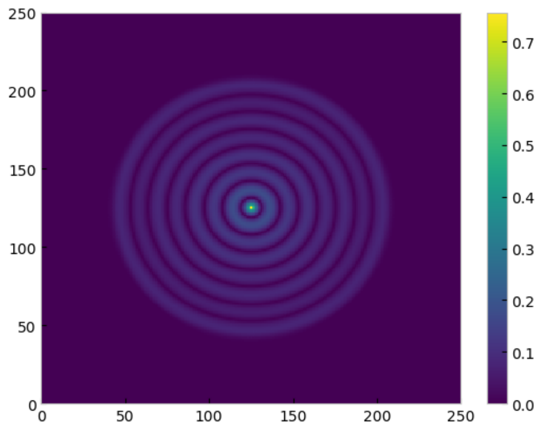

(b) No superluminal waves are caused by a source that turns on slowly.

Figure 5.1: A comparison of the behavior between two different sources, illustrating the presence and absence of "field jitter".

p114]. This phenomenon is shown in fig 5.1a, where a source of the form $E(x, y) = \exp\left(-n^2/20\right)$ was used, to isolate the field jitter. Here (x,y) is the source point and n is the current timestep.

Source choice For the reasons outlined above and others, a source of the form

$$E(x, y) = \sin(n/5)$$

was used in all simulations shown.

Chapter 6

Other considerations

6.1 Arithmetic Complexity

In this section the arithmetic complexity of a FDTD simulation is briefly discussed. The computational complexity describes the amount of resources required for running an algorithm. The special case in which this resource is the number of operations is called the arithmetic complexity. Specifically this is the number of operations required to obtain the output of the algorithm for an input of size N. Constant factors, such as the factor 2 that occurs by calculating two fields rather than one, as we have for the FDTD method, have no impact on the arithmetic complexity as the notation is defined such that $\mathcal{O}(cn) = \mathcal{O}(n), \forall c \neq f(n)$. The estimated number of operations for a FDTD simulation simply is the number of field values calculated (i.e. $n_x \times n_y$) times the number of times they are calculated (i.e. n_t). Consider a physical space of some determined size. Unlike the amount of time in seconds it takes until the wave reaches the boundary, the amount of time in time steps does evidently depend on the discretization constants. Since there is a limit on the size of the time step dependent on the spatial step size, a finer discretization requires a simulation to run for a greater number of time steps. This means that $nt \sim nx$ or $nt \sim ny$, up to some factor independent of n. Then, for a square grid, i.e. $n = n_x = n_y$, the computational complexity of a D dimensional simulation is $\mathcal{O}(n^{D+1})$. So for a 2D square grid, the arithmetic complexity of the FDTD method is $\mathcal{O}(n^3)$.

Example Consider a 2D simulation of a 5m by 5m room filled with 2.5 GHz radiation (WIFI standard 802.11b). Consideration of the numerical anisotropy error not further discussed in this thesis demand that the discretization constants are smaller than some fraction of the minimum wavelength, a factor designated by N_λ. Taflove finds that to achieve an error smaller than 0.1% the basic Yee algorithm requires $N_\lambda = 29$, i.e. the discretization constants need to be smaller than 1/29th [4] of the minimum wavelength. A rough calculation using $\lambda = c/f$, $\Delta = \frac{\lambda}{N_\lambda}$, $n = \frac{L}{\Delta}$ and $N \sim n^3$, can lead to an order-of-magnitude estimate for the number of operations. Using the numbers from above, and furthermore assuming 3 operations per update equation and

3 update equations, this leads to an estimate for the number of operations required for such a simulation on the order of 10^{10} or 10 GFLOP. A corresponding 3D simulation of the same room, assuming 5 m height, would then require on the order of 10 TFLOP. Note that the addition of another dimension also reduces the maximum stable time step via the CFL criterion from $\Delta t \leq \frac{\Delta}{\sqrt{2}v_p}$ to $\Delta t \leq \frac{\Delta}{\sqrt{3}v_p}$, but this difference is negligible in an order of magnitude estimate.

"Time to Steady State" approach Nagy et al [11], have a similar, slightly more sophisticated approach to calculating the arithmetic complexity of the FDTD method for a square grid in 2 dimensions. F_{iter} is the number of numerical operations in a single iteration and can be estimated by the number of algebraic operations per time step.

$$F_{iter} = 28n^2 \approx n^2$$

F_{steady} is the number of time steps until steady state, which they estimate as two times the number of time steps the wave requires to reach the boundary, taking into account the local group velocity of the wave.

$$F_{steady} \approx 2\sqrt{2}n\sqrt{\frac{\epsilon}{\epsilon_0}}$$

The total number of operations then is

$$F = F_{steady} \times F_{iter} = 2\sqrt{2}n\sqrt{\frac{\epsilon}{\epsilon_0}}28n^2 \approx n^3$$

Nagy et al also take into account the thickness of the PML, using $n = n_{FDTD} + 2n_{PML}$ and a retain the factor $\sqrt{\frac{\epsilon}{\epsilon_0}}$ in the final approximation. As the PML usually has a thickness of 10 cells and the latter factor is < 10 in most materials, we have judged this impact as negligible in an estimate of computational complexity.

6.2 Computation times

In this section compares the computation times of the FDTD method in different implementations and simulations. In order to decouple the results from the performance of the particular computer the simulations were run on, we give the results not in seconds. Instead we will define a base case for each comparison and set its computation time to 1. Then the computation time of the test case will be shown as a multiple or fraction of the base case. Each simulation is repeated a number times and the average and standard deviation are shown.

6.2.1 Dependence on field values

First, a possible dependence of the computation time on the field values is investigated. In particular we consider the extreme case of a trivial simulation: the field values are set to 0 at

Table 6.1: Computation times of a trivial and a non-trivial simulation

Method	**Minimum**	**Mean**	**Standard deviation**
trivial	1	1.052	0.064
non-trivial	1.004	1.056	0.062

the beginning and there are no sources, i.e. the field values are zero at all time and space points. For the non-trivial simulation a sine wave source at the center is used, as always. We reason that if the computation time of the trivial solution is significantly different from the non-trivial solution, it follows that different non-trivial simulations will also have differing computation times.

A simulation the basic Yee algorithm for a 100×100 lattice was run for 200 time steps. This simulation was executed 50000 times for the trivial and the non-trivial solution each. Minimum, mean and standard deviation of the computation times are shown in Table 6.1. The minimum value of the trivial solution is the base case that is set to 1 and all other values are shown as multiples of this base case. We emphasize that the minimum value is the important one, as, according to the documentation of the timeit python package:

> In a typical case, the lowest values gives a lower bound for how fast your machine can run the given code snippet; higher values in the result vector are typically not caused by variability in Python's speed but by other processes interfering with your timing accuracy.

Nonetheless we report the mean and standard deviation for completeness sake.

The non-trivial solution has a simulation time of 1,004. We conclude that there is in fact some tiny dependence of the computation time on the field values of the simulation on the order of half a percentage point. Presumably this difference depends on how the used programming language handles the storage and computation of floating point numbers. It persists when for the trivial simulation a "null source" is used, that is a source that mimics a real source in the way the source function in the simulation is called, the argument given, etc, but always returns 0.

6.2.2 PML computation times

As mentioned above arithmetic complexity does not concern itself with other things we care about, such as a twofold increase in computation time. In our simulations, such a factor may for example be introduced by the number of updating fields. In this section we will consider the number of operations without the PML, with the Bérenger PML and with the CPML for one mode of a 2D simulation and then compare the results to measured computation times.

Table 6.2: Computation times of two PML implementations compared with a PML-less implementation.

Method	**Minimum**	**Mean**	**Standard deviation**
no PML	1	1.51	0.49
Bérenger	1.46	2.58	0.56
CPML	2.16	2.52	0.54

We do not consider the pre-factor fields a, b, c in an update equation like

$$E = aE + b\frac{\partial B}{\partial x} + c\frac{\partial B}{\partial x}$$

as these can be calculated at the beginning of the simulation and do not change over its course. Thus their computational impact is negligible. This may be different for dispersive media in particular, because then ϵ and μ are functions of the frequency and may need to be calculated at each time steps and position.

The basic Yee algorithm has three updating scalar fields. For the TE mode those are Ex, Ey and Hz. The Bérenger Split Field PML has 5 updating fields (E_x, E_y, H_{zx}, H_{zy} and H_z), although there is reason to believe that the operation $H_z = H_{zx} + H_{zx}$ is not as computationally intensive as other update equations. The CPML has 7 updating fields: the normal 3 + the 4 Ψs. The number of update equations per time step is then 3, 5 and 7, respectively. Consequently, since the number of operations per update equation is roughly similar for all three methods, we would also expect the computation times to exhibit a ratio of between 3 : 4 : 7 and 3 : 5 : 7, or setting the basic Yee algorithm to 1, between $1 : 1\frac{1}{3} : 2\frac{1}{3}$ and $1 : 1\frac{2}{3} : 2\frac{1}{3}$.

The simulation was for a 200×200 grid of vacuum, run for 300 time steps. It was repeated 1000 times for each method. Table 6.2 shows the minimum, mean and standard deviation of the computation times. The minimum time of the Yee algorithm without a PML is set to 1 and all other values are shown as multiples of this base case.

In short, the three implementations have a ratio of computation times of 1 : 1.46 : 2.16. This roughly corresponds a naive estimate based on the number of update equations per time step alone.

6.2.3 GPU Speedup

Taflove finds a comparative speed advantage of the simulation run on the GPU of approximately 7:1 versus the CPU based simulation. More recently Wang et al [12] investigate (CUDA based) GPU acceleration of a FDTD simulation of unmagnetized plasma. They find a speedup ratio of between 4.45 : 1 and 2.47 : 1. This advantage decreases as the simulation is run for more time steps. Naturally, this advantage depends on the hardware used.

Chapter 7

Conclusion

In conclusion we can say that the FDTD method in its basic implementation is both quite powerful and yet straightforward. In fact, the boundary treatments like the ones discussed in Chapter 4 and their derivation are significantly more complicated than the basic Yee algorithm itself. Furthermore the creation of the mesh of ϵ and μ seems somewhat laborious, in particular in three dimensions.

We found the literature on the topic to be both extensive and helpful. They present numerous extensions to the Yee algorithm such as a hexadecimal grid for the treatment of numerical phase anisotropy. This grid was considered as well for this thesis but the grid-to-array mapping rules proved somewhat more demanding in this case than in that of the Yee grid.

While writing this thesis, the author learned numerous things, among them the importance of numerical stability conditions (see Section 2.3) and the necessity of considering the limits of numerical simulations as showcased in the superluminal waves created by fast source start-up (see Section 5).

List of Figures

List of Tables

Bibliography

[1] Kane S. Yee. “Numerical solution of initial boundary value problems involving Maxwell’s equations in isotropic media”. In: *IEEE Transactions on Antennas and Propagation* (1966), pp. 302–307. DOI: https://doi.org/10.1109/tap.1966.1138693.

[2] G. Mur. “Absorbing Boundary Conditions for the Finite-Difference Approximation of the Time-Domain Electromagnetic-Field Equations”. In: *IEEE Transactions on Electromagnetic Compatibility* EMC-23.4 (Nov. 1981), pp. 377–382. ISSN: 0018-9375. DOI: https://doi.org/10.1109/TEMC.1981.303970.

[3] Jean-Pierre Berenger. “A Perfectly Matched Layer for the Absorption of Electromagnetic Waves”. In: *Journal of Computational Physics* 114.2 (Oct. 1994), pp. 185–200. DOI: http://dx.doi.org/10.1006/jcph.1994.1159.

[4] Allen Taflove. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. 3rd ed. Boston: Artech House, 2005.

[5] Umran S. Inan and Robert A. Marshall. *Numerical Electromagnetics: The FDTD Method*. 1st ed. Cambridge: Cambridge University Press, 2011. DOI: https://doi.org/10.1017/cbo9780511921353.

[6] John David Jackson. *Classical Electrodynamics*. 3rd ed. New York, NY: Wiley, 1999. DOI: https://doi.org/10.1002/3527600434.eap109.

[7] Eric W. Weisstein. *Hyperbolic Partial Differential Equations From MathWorld—A Wolfram Web Resource*. Last visited on 25/2/2018. URL: http://mathworld.wolfram.com/HyperbolicPartialDifferentialEquation.html.

[8] Ulrich Hohenester et al. “Simulations and modeling for nanooptical spectroscopy”. In: Unpublished, Jan. 2017.

[9] Yen Liu. “Fourier Analysis of Numerical Algorithms for the Maxwell Equations”. In: *Journal of Computational Physics* 124 (Feb. 1993), pp. 396–416. DOI: https://doi.org/10.1006/jcph.1996.0068.

[10] B. D. Gvozdic and D. Z. Djurdjevic. “Performance advantages of CPML over UPML absorbing boundary conditions in FDTD algorithm”. In: *Journal of Electrical Engineering* 68 (Jan. 2017), pp. 47–53. DOI: https://doi.org/10.1515/jee-2017-0006.

[11] L. Nagy, R. Dady, and A. Farkasvolgyi. “Algorithmic complexity of FDTD and ray tracing method for indoor propagation modelling”. In: *3rd European Conference on Antennas and Propagation*. Mar. 2009, pp. 2262–2265.

[12] X.-m. Wang et al. “GPU-Accelerated Parallel Finite-Difference Time-Domain Method for Electromagnetic Waves Propagation in Unmagnetized Plasma Media”. In: *ArXiv e-prints* (Sept. 2017). arXiv: `1709.00821 [physics.comp-ph]`.