Implementation of Parallel-1D FDTD Algorithm using Schwarz Alternating Method

Undergraduate Project Proposal

by

Karlo Pio Dungog Alaan
2015-07559
B.S. Electronics and Communications Engineering
and
Jan Marvin Delos Reyes Moyco
2015-09359
B.S. Electronics and Communications Engineering

Adviser:

Adrian Vidal
Jaybie Agullo De Guzman
Bernalyn Decena

University of the Philippines, Diliman
August 2021

Abstract

Implementation of Parallel-1D FDTD Algorithm using Schwarz Alternating Method

Students predominantly have no access to powerful computational devices when doing electromagnetic simulations. These usually require powerful hardware because of the scale and complexities of the problems. Finite Difference Time Domain (FDTD) is one of the popular numerical methods for solving electromagnetic problems because of its simplicity and ease of implementation. Parallel FDTD algorithms have already shown significant advantage over traditional serial FDTD algorithms but they lack a method of validating the data in each subdomain. Schwarz Alternating method is a common domain decomposition method used in solving partial differential equations. A new method combining the a parallelized FDTD via the Schwarz method is developed in this project. This can narrow the gap between the lack of access to powerful hardware and the need for electromagnetic simulations by utilizing parallel computing with relatively inexpensive hardware. The current progress of the project is the development of a serial 1D FDTD algorithm in Python.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of the Project

In 2020, the COVID-19 pandemic forced every educational institution in the world to forego face-to-face classes and moved its operations online. This change in mode of learning exacerbated already existing problems of students with the lack of devices for learning or research as well as internet connectivity. Before the pandemic began, data from Philippine Statistics Authority [?] show that less than 30% of households in the Philippines have access to a desktop or laptop. During the pandemic, an estimated 6.8 million students lack access to proper Information and Communications Technologies (ICT) devices (phone, tablet, laptop, etc) for online learning and 6.8 million lack access to a stable internet connection in the Philippines [?]. Another survey conducted by Social Weather Stations shows that only 13% of enrolled Filipino students have access to a desktop or laptop [?] which is usually needed for tech-related or engineering courses. This lack of access to computing resources are apparent despite the decrease in the price of computers in recent years.

These problems are more apparent in tertiary education since in general, college students need some sort of ICT devices for their studies. For example, in a survey conducted by UP Media and Public Relations Office, 28% students in the UP System do not have access to any ICT device for learning [?] while majority of the students in UP Diliman (approximately 97% of the total student population) have some sort of access to an internet connection [?] as shown on Figure 1.1.

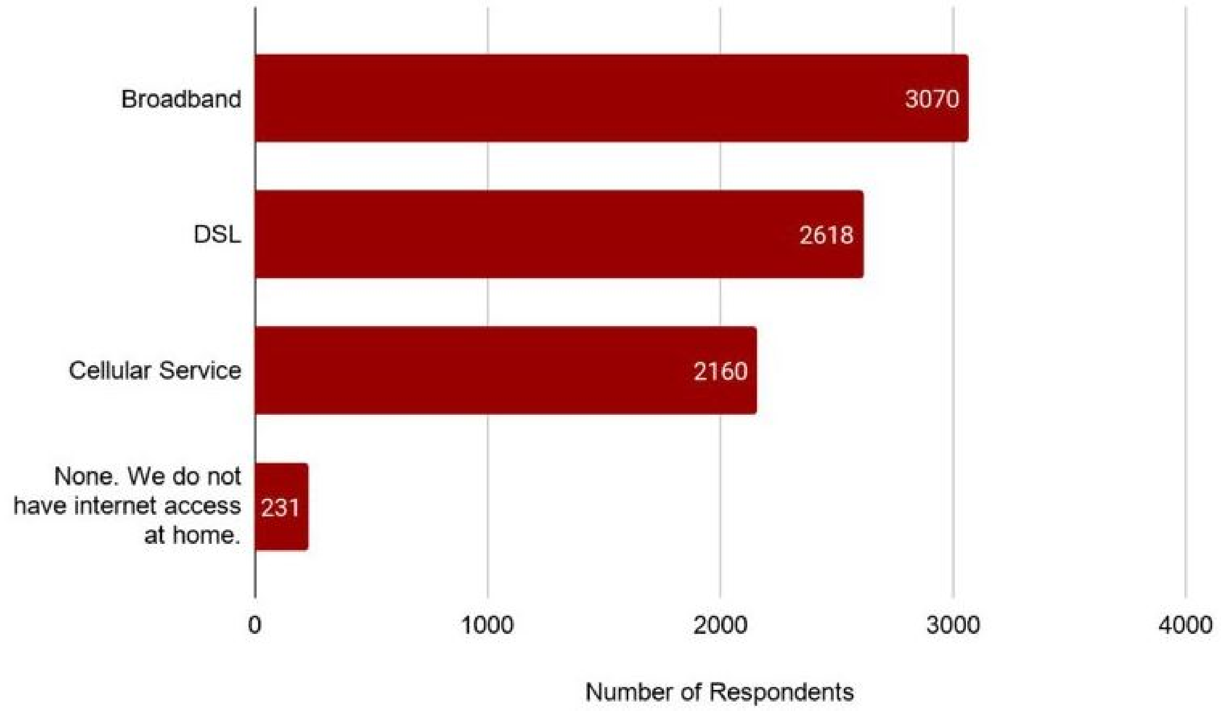


Figure 1.1: The internet sources of the respondents in the survey shows that most students have access to an internet connection in different forms [?].

One of the methods used to increase accessibility to computational resources without acquiring expensive hardware is cloud computing [?]. This is a type of computing where the computing resources can be rented on-demand without buying any hardware [?] can offload the burden of acquiring new hardware, maintaining the hardware, and upgrading the hardware while retaining the computational resources at a fraction of the cost [?]. This will be advantageous to anyone without access to high computational resources because renting computing power is much lower than acquiring and maintaining hardware that provides the same computing power. This advantage is especially beneficial in processes that involves large amounts of data processing like electromagnetic (EM) simulations because of the need for high computational power in a short period of time, cloud computing can be used to augment the researcher's computational needs in a certain amount of time without spending a large amount and tweaking the hardware to suit research requirements.

In EM research, simulating and modelling EM events is usually difficult to do because current problems in EM research need complex design and large structure to provide adequate solutions [?] and require long simulation time. In addition, EM problems that has real world application rarely has exact analytical solution and using approximate solutions from numerical methods that are tailored for solving Maxwell's Equation are the next best thing [?]. There are several methods of computing Maxwell's equations numerically like Finite Difference Time Domain (FDTD) [?], Finite Element Method (FEM) [?], Method of Moment (MoM) [?], etc. FDTD, in particular, has been a very popular tool for EM simulations [?] due to its ease of implementation and good accuracy of the simulation [?].

To improve the long simulation time issue common in complex EM problems, parallel computing can be integrated to the simulation of EM problems. Parallel computing is the method of splitting a numerical problem into several sub-problems such that the computation of each sub-problem can be done simultaneously [?]. This method improves the simulation time of an EM problem because of the simultaneous processing of each part of the main problem. The computational load on each sub problem is less than the total problem so in theory, a problem can be solved without investing too much on high performance processors.

Overall, cloud computing can be used to minimize the cost of EM research due to the hardware requirements while parallel computing can reduce the simulation time of complex EM problems. Thus, combining FDTD with parallel processing and cloud computing can be a good solution in creating an efficient, accurate, and accessible system for EM simulations.

1.2 Project Overview

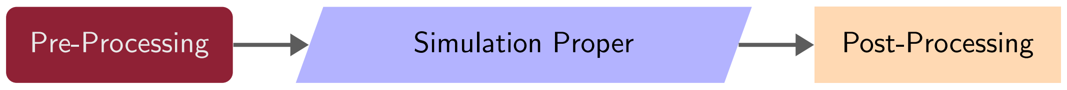


Figure 1.2: Block Diagram for the EEE 190 Project

The building blocks of the project shown in Figure 1.2 are the major components of the project:

1. **Pre-Processing** - is the part where the input data is processed, simulation parameters are initialized (creation of variables and computing constants), and the computational domain is created for the simulation proper.
2. **Simulation Proper** - is where the proposed algorithm will be used and the necessary measurements are done.

3. **Post-Processing** - is the part where the data is consolidated for analysis, data visualization, and storage.

1.3 Documentation Flow and Organization

In the succeeding chapters, the review of related works will explain the relevant concepts that will be used in the project. The problem statement and objectives explains the project's scope as well as expected objectives at the project completion. The methodology is the detailed explanation of how the project will be accomplished as well as the method of verification of the results. The initial simulation results of the project will be presented in the preliminary findings and the project schedule and deliverables will discuss the timeline of the project as well as the deliverables and their relation to the project's objectives.

Chapter 2

Finite-Difference Time Domain

This chapter will focus on the basics of the Finite-Difference Time Domain method. It aims to familiarize the reader with the foundations of the method, starting with Maxwell's equations. From there, the Finite Difference method was used to approximate the equations for easier numerical solving. The basic FDTD algorithm is also shown in this chapter. Lastly, the derivation of the one-dimensional FDTD is presented here as well.

2.1 Maxwell's Equations

Maxwell's equations are the foundation of electromagnetic theory. The equations are shown in Equation 2.1.

$$\nabla \cdot \mathbf{D} = \rho_v \qquad \text{Gauss' Law} \tag{2.1a}$$

$$\nabla \cdot \mathbf{B} = 0 \qquad \text{Gauss' Law for Magnetism} \tag{2.1b}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \qquad \text{Ampere's Law} \tag{2.1c}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \qquad \text{Faraday's Law of Induction} \tag{2.1d}$$

2.2 Finite Difference Method

Both Curl Equations consist of derivatives. Since computers cannot inherently solve for derivatives, some approximations can be made for them to calculate derivatives. The Finite Difference Method is used to get an appropriate approximation for the Curl Equations.

The Finite Difference Method calculates the derivative at the midpoint of two points of a function. In approximating the Curl Equations, special care must be given when formulating the approximated equations. Particularly, each term of both sides of the equation must exist in at the same point in time. This is to ensure that the equations are stable during the simulations [**?**].

Using the Finite Difference Method, the Curl Equations can be approximated as shown in Equations 2.2 and 2.3.

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad \Rightarrow \quad \nabla \times \mathbf{E}(t+\Delta t) \simeq -\mu \frac{\mathbf{H}(t+\frac{\Delta t}{2}) - \mathbf{H}(t-\frac{\Delta t}{2})}{\Delta t} \tag{2.2}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} \quad \Rightarrow \quad \nabla \times \mathbf{H}(t+\frac{\Delta t}{2}) \simeq \varepsilon \frac{\mathbf{E}(t+\Delta t) - \mathbf{E}(t)}{\Delta t} \tag{2.3}$$

From the above equations, another set of equations called the Update Equations can be defined. These equations will be used to compute the field values at the next time step inside the FDTD loop. The Update Equations can be seen in Equation 2.4.

$$\mathbf{H}|_{t+\frac{\Delta t}{2}} = \mathbf{H}|_{t-\frac{\Delta t}{2}} - \frac{\Delta t}{\mu}\left(\nabla \times \mathbf{E}|_t\right) \tag{2.4a}$$

$$\mathbf{E}|_{t+\Delta t} = \mathbf{E}|_t + \frac{\Delta t}{\varepsilon}\left(\nabla \times \mathbf{H}|_{t+\frac{\Delta t}{2}}\right) \tag{2.4b}$$

2.3 The Basic FDTD Algorithm

The type of FDTD used in this project is the Alternating Direction Implicit (ADI) since it has less stringent stability criterion [**?**]. The basic operating principle of the FDTD algorithm is the leapfrog method [**?**]. It uses values from another field to compute the values of the current field. In FDTD, the electric field is computed based on the magnetic field, and conversely, the magnetic field is computed based on the electric field. This cycle continues until the set number of iterations is finished.

The FDTD algorithm updates the values of the fields via the Update equations in 2.4. The basic FDTD algorithm flowchart can be seen in Fig. 2.1. In order to reduce computation time, constants such as the time step and permittivity can be initialized at the beginning. Both fields are also initialized to zero.

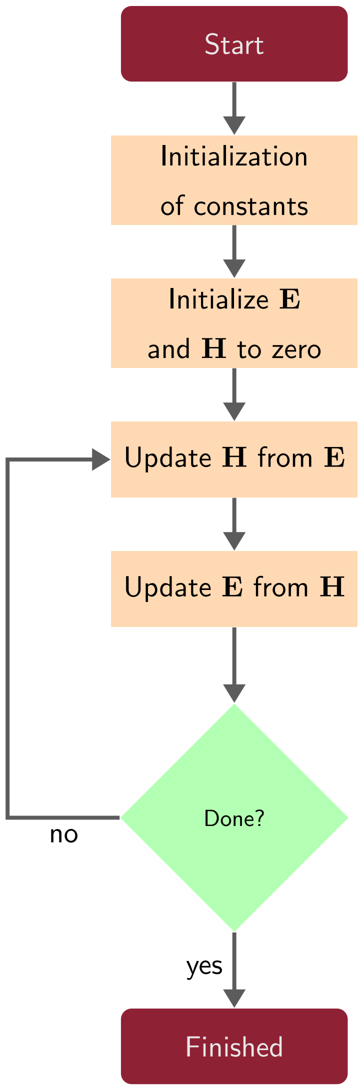


Figure 2.1: Basic FDTD Algorithm

2.4 Reduction to 1D Using Yee Grid Scheme

The Equations in 2.2 deal with the time components of Maxwell's Curl Equations. However, these equations still exist in three dimensions. In compliance to the objectives of this project, these equations need to be reduced to one dimension only. This can be achieved using Yee Cells [?].

Yee cells are used to discretize the space components of the Curl Equations. In a Yee cell, the electric and magnetic fields are arranged such that they are staggered inside the grid [?]. A visual representation of the Yee cell with the fields can be seen in Figure 2.2 [?].

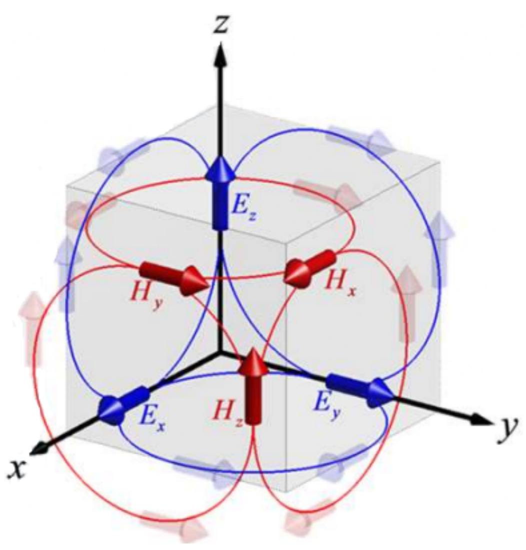
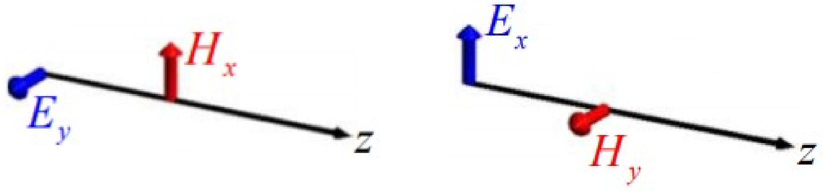


Figure 2.2: Three-dimensional Yee Cell [?].

Equations for the x, y, and z components of the Curl Equations can be deduced from one Yee cell. However, the derivations will no longer be included in this document for the purpose of conciseness. The focus is shifted towards the reduced one-dimensional Yee cell, as shown in Figure 2.3 [?]. There are two configurations or modes of the one-dimensional Yee cell: the E_y/H_x mode and the E_x/H_y mode. For this project, the E_y/H_x mode will be used since the two are functionally the same. As seen in the figure, only the y component of the electric field remains, while only the x component of the magnetic field remains.



(a) E_y/H_x mode (b) E_x/H_y mode

Figure 2.3: One-dimensional Yee cells [?].

The E_y/H_x mode is possible since the derivatives along x and y computed using the three-dimensional analysis can be reduced to zero. The field values along those directions are constant since material properties only change along the propagation path, which is the z direction. The Curl Equations can be reduced as seen in Equation 2.5, where k is the array index.

$$-\frac{\mathbf{E}_y^{k+1}(t)-\mathbf{E}_y^{k}(t)}{\Delta z} = \frac{\mu^k}{c_0}\frac{\mathbf{H}_x^{k}(t+\frac{\Delta t}{2})-\mathbf{H}_x^{k}(t-\frac{\Delta t}{2})}{\Delta t} \tag{2.5a}$$

$$\frac{\mathbf{H}_x^{k}(t+\frac{\Delta t}{2})-\mathbf{H}_x^{k-1}(t+\frac{\Delta t}{2})}{\Delta z} = \frac{\varepsilon^k}{c_0}\frac{\mathbf{E}_y^{k}(t+\Delta t)-\mathbf{E}_y^{k}(t)}{\Delta t} \tag{2.5b}$$

Using Equation 2.5, a more refined set of the Update Equations can be derived, specifically for the E_y/H_x mode. The new equations can be seen in Equation 2.6, and will be used for the entirety of the project.

$$\mathbf{E}_y^k\Big|_{t+\Delta t} = \mathbf{E}_y^k\Big|_t + \left(\frac{C_0\Delta t}{\varepsilon^k}\right)\left(\frac{\mathbf{H}_x^k\big|_{t+\frac{\Delta t}{2}} - \mathbf{H}_x^{k-1}\big|_{t+\frac{\Delta t}{2}}}{\Delta z}\right) \tag{2.6a}$$

$$\mathbf{H}_x^k\Big|_{t+\frac{\Delta t}{2}} = \mathbf{H}_x^k\Big|_{t-\frac{\Delta t}{2}} - \left(\frac{C_0\Delta t}{\mu^k}\right)\left(\frac{\mathbf{E}_y^{k+1}\big|_t - \mathbf{E}_y^k\big|_t}{\Delta z}\right) \tag{2.6b}$$

2.5 More Detailed 1D FDTD

The scope for this project is to use a FDTD in one dimension, which is the simplest type of FDTD that can be used to show if the parallel configuration is better than serial configuration. The most basic algorithm of FDTD is the leapfrog method [?] that alternates in solving for the electric and magnetic field in each Yee cell using the update equations [?]. The update equations used in this project are the Ey/Hx form of the FDTD Update equation in 1D [?] and this form is chosen arbitrarily since the other form Ex/Hy is similar to the Ey/Hx form. The difference is the convention of the directions of the electric and magnetic fields in the simulation space.

$$\tilde{\mathbf{H}}_x|^k_{t+\frac{\Delta t}{2}} = \tilde{\mathbf{H}}_x|^k_{t-\frac{\Delta t}{2}} + (m_{\mathbf{H}_x}|^k)(\frac{\mathbf{E}_y|^{k+1}_t - \mathbf{E}_y|^k_t}{\Delta z}) \tag{2.7}$$

$$\mathbf{E}_y|^k_{t+\Delta t} = \mathbf{E}_y|^k_t + (m_{\mathbf{E}_y}|^k)(\frac{\tilde{\mathbf{H}}_x|^k_{t+\frac{\Delta t}{2}} - \tilde{\mathbf{H}}_x|^{k-1}_{t+\frac{\Delta t}{2}}}{\Delta z}) \tag{2.8}$$

$$m_{\mathbf{E}_y}|^k = \frac{c_0\Delta t}{\epsilon_{yy}|^k} \tag{2.9}$$

$$m_{\mathbf{H}_x}|^k = \frac{c_0\Delta t}{\mu_{xx}|^k} \tag{2.10}$$

The equation above shows the update equations and the update coefficients for both the electric and magnetic fields for the FDTD algorithm. There are two important points about the update equations that need to be discussed: their dependence on previous value and adjacent values. First, the first term in the update equation is the previous field value (for both electric and magnetic field) so, every value in the k-th position in space is dependent on the previous time step. Second, the dependence on adjacent cells, since the electric field has a term for the magnetic field in k-1 position, this means that the electric field is dependent on the adjacent magnetic field value while the magnetic field is dependent on the adjacent electric field value (because it has an electric field in k+1 position).

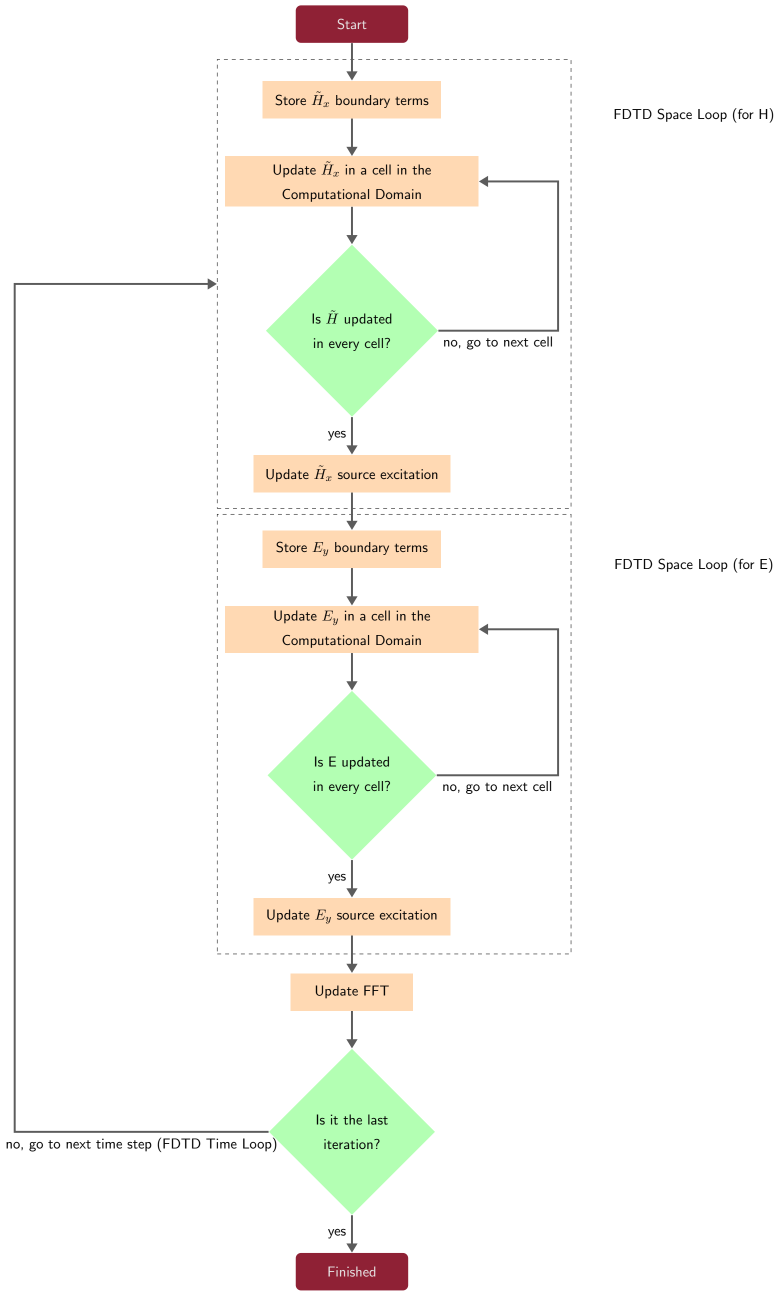

Start
Store $\tilde{H}_x$ boundary terms
FDTD Space Loop (for H)
Update $\tilde{H}_x$ in a cell in the Computational Domain
Is $\tilde{H}$ updated in every cell?
no, go to next cell
yes
Update $\tilde{H}_x$ source excitation
Store E_y boundary terms
FDTD Space Loop (for E)
Update E_y in a cell in the Computational Domain
Is E updated in every cell?
no, go to next cell
yes
Update E_y source excitation
Update FFT
Is it the last iteration?
no, go to next time step (FDTD Time Loop)
yes
Finished

Figure 2.4: 1D FDTD Algorithm

The flowchart shown in Fig 2.4 is the main algorithm of FDTD that computes for the field values in space and time. There are other steps like the pre-processing and post-processing that will be discussed in the implementation part of this chapter. In general, the FDTD algorithm has two main loops: the main loop that iterates through every time step, which will be referred here as the FDTD time loop and the inner loop that iterates through every cell in the computational domain, referred here as the FDTD space loop.

First, the FDTD time loop is where the main steps of the algorithm are done. There are three main steps:

1. Updating the magnetic field values
2. Updating the electric field values
3. Updating the Fourier Transform and checking the stop condition

The first and second steps are similar, the only difference is what field values are updated. This first two step can be referred to as FDTD space loop since these steps are looping in the computational space of the simulation. The initial step in this part is to store the boundary values of magnetic or electric fields for the computation of the boundary condition in the next step. Next, to update the field values in the computational domain including the external boundary conditions of the domain, the FDTD space loop is involved since the iteration will be done through every cell in the computational domain. The update equation specified in 2.8 and 2.7 will be used to update each cell and after updating the cells in the whole computational domain.

The injection of source excitation is done after the update equations are done where the source excitation (computed in the pre-processing stage) will be used by injecting, depending on the method of excitation into the specified injection point (k_{inj}). This will affect the field values since the field values have a dependence on the adjacent cells and so, the effect of the source excitation will be seen after the current iteration in the FDTD time loop is finished. The method on how the source excitation is injected into k_{inj} and what types of sources will be used in the project will be discussed in the implementation part.

After the storing of boundary values of the electric and magnetic fields, FDTD space loop for electric and magnetic fields, the last step in the FDTD time loop is the update of Fourier transform and checking of the stop condition. The Fourier transform of each iteration in the FDTD time loop can be used to compute for the reflection and transmission coefficients for a specified frequency since FDTD is a time-based method, a single simulation run can include a wide range of frequencies [?]. The equation used for calculating the reflection and transmission coefficients are shown below:

$$R(f) = (\frac{FFT[E_{ref}(t)]}{FFT[E_{src}(t)]})^2 \tag{2.11}$$

$$T(f) = (\frac{FFT[E_{trn}(t)]}{FFT[E_{src}(t)]})^2 \tag{2.12}$$

$$C(f) = R(f) + T(f) \tag{2.13}$$

The reflected and transmitted fields are converted into the frequency domain by Fast Fourier Transform, and the fields are normalized with respect to the source excitation. The law of conservation of energy is also checked by getting the sum of the reflected and transmitted fields [?]. This equation holds true in this situation because in the current FDTD model, the conductivity or the absorption loss after passing through a material is not taken into account so there should be no loss of energy because of the conductivity of the materials. The reflected and transmitted fields are measured at the first and last cells of the computational domain for the reflected and transmitted fields respectively.

The stop condition for the FDTD time loop is when the current iteration reaches the time specified in the pre-processing stage. If the current iteration reaches the last timestep, the FDTD time loop is finished and the algorithm will proceed to post-processing, otherwise, the algorithm continues to the next iteration of the FDTD time loop.

Chapter 3

Schwarz Method

In this chapter, the concept of Domain Decomposition is presented as a way to numerically solve problems. The Schwarz method, which is a Domain Decomposition scheme, will be tackled on a surface level in this chapter as well.

3.1 Domain Decomposition

Domain decomposition is a general method of solving a global large problem by dividing it into smaller simple smaller local subproblems [?]. It is usually used in conjunction with parallel computing because each subproblem can be assigned to a processor in a parallel computing setup [?]. Since most modern processors have multiple processing cores, parallelization is becoming more viable [?].

There are two types of domain decomposition in terms of how the computational domain is partitioned: the non-overlapping and the overlapping domains. Overlapping methods, also known as Schwarz methods, has an overlap region between each adjacent subdomains while non-overlapping methods have no overlap region in the adjacent subdomains [?]. For this project, only the overlapping type will be considered.

3.2 Schwarz Alternating Method

The Schwarz Alternating Method or Schwarz Multiplicative Method is a type of overlapping method of domain decomposition [?] which uses the overlapping region between the different subdomain to check the convergence of the whole computational domain/simulation space [?]. This particular method is the simplest form of overlapping domain type of domain decomposition that uses Dirichlet boundary conditions [?] at the external boundaries of the computational domain. The basic algorithm of the method is shown below:

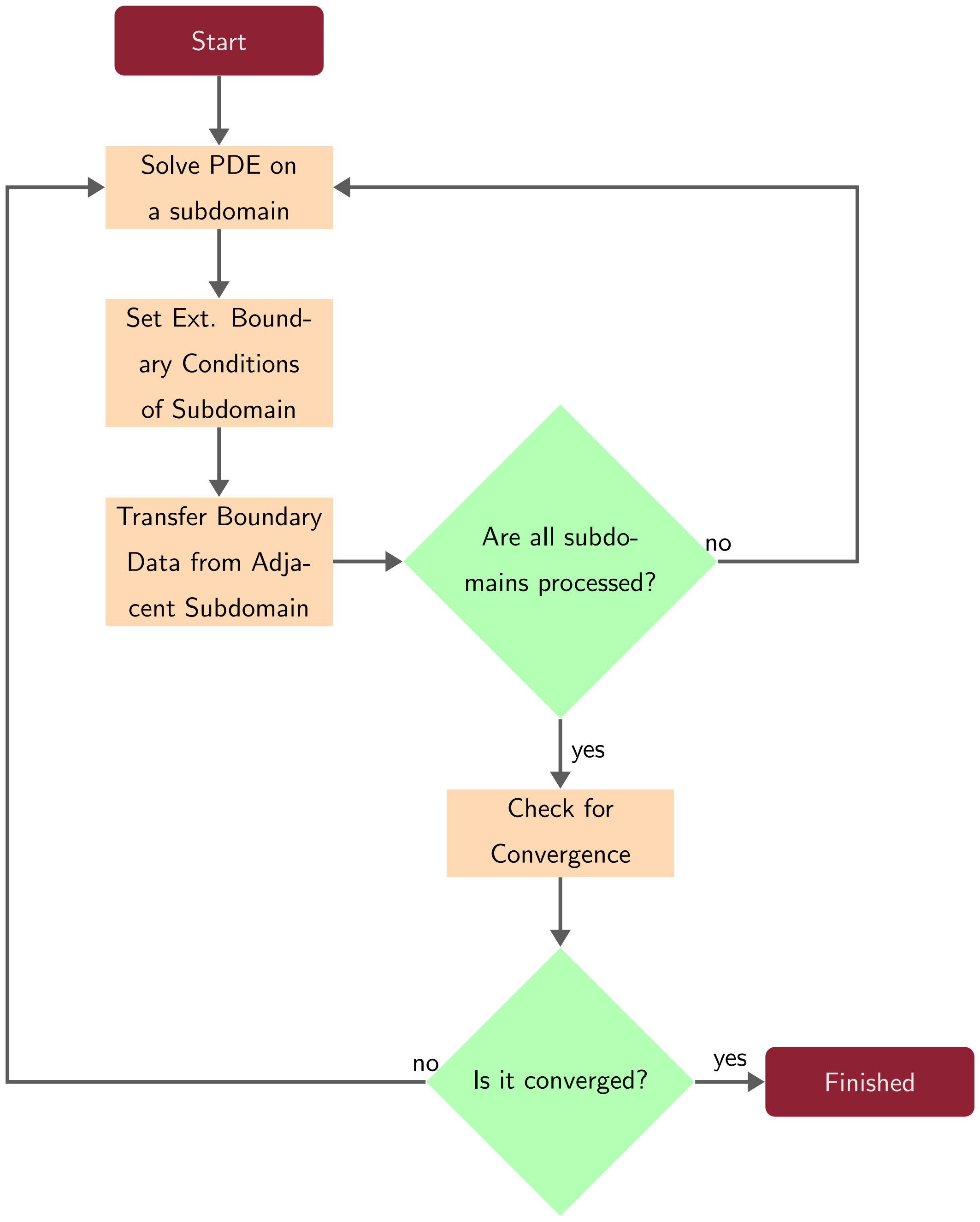

Start
Solve PDE on a subdomain
Set Ext. Boundary Conditions of Subdomain
Transfer Boundary Data from Adjacent Subdomain
Are all subdomains processed?
no
yes
Check for Convergence
Is it converged?
no
yes
Finished

Figure 3.1: Schwarz Alternating Algorithm [?]

This domain decomposition method decomposes the whole computational domain into several subdomains where each subdomain has an overlapping region with each adjacent subdomain as a pre-processing step and the algorithm can be divided into two parts: the first part where it can be done asynchronously and the second part where each subdomain must do it synchronously.

These steps can be done by each subdomain asynchronously with respect to the other subdomains:

1. Solve the partial differential equation on a subdomain
2. Set the external boundary values of the subdomain (not including the inner boundaries as a consequence of partitioning the computational domain)

   After the first two steps, each subdomain must wait for their adjacent subdomain to finish before continuing the algorithm. The next half of the algorithm must be done synchronously because it requires the data transfer between the adjacent subdomains.
3. Transfer the data from the adjacent subdomain/s to the inner boundary of the subdomain
4. Check for convergence of each subdomain

In the second part, each subdomain must finish the first part so that the data that will be transferred to the inner boundaries of each subdomain will be the updated value that has been computed in the first part. The importance of the overlapping region of this algorithm is for the convergence of the numerical solution. The convergence is validated by comparing the values in the overlapping regions of each subdomain such that they satisfy the condition shown below. If the convergence is achieved, the algorithm will proceed to post-processing, otherwise the algorithm will start to step 1 in the first half of the algorithm.

$$error \leq \epsilon_{tol} \tag{3.1}$$

A more in-depth description of the convergence method used in this project will be done in the implementation part.The *error* is the error function that will be used to calculate the difference in values between overlapping regions while the ϵ_{tol} is the error tolerance that the algorithm uses to determine if the solution has converged. The important point in this part is that the Schwarz Alternating Method can validate whether each subdomain converges to one solution using the overlapping region/s in each subdomain.

Chapter 4

Related Work

4.1 Finite-Difference Time Domain

The earliest known work related to numerically solving for partial difference equations was published in 1928 [?]. Since then up until the 1960's, there only one literature that is related to time-dependent finite difference methods [?]. In 1966, Kane Yee [?] applied finite-difference methods in solving for Maxwell's curl equations. The concept of staggering the electric and magnetic fields across space and time in a grid was also introduced in this work. The term "Finite Difference Time Domain" was coined by Taflove in 1980 [?]. Since then, the Finite Difference Time Domain is used in scientific and engineering applications dealing with electromagnetic wave interactions. Its most prominent use is in the field of wireless communications, but other fields of study such as biomedicine and quantum mechanics have used FDTD as well [?]. Although the study and application of FDTD has been around for quite some time, research on the subject still continue based on recent studies [?, ?, ?].

4.1.1 Properties of the FDTD Method and Comparisons to Other Methods

As mentioned in Chapter 1.1, there are several other ways to numerically compute partial differential equations. This subsection will show some comparisons between FDTD and the aforementioned methods. This subsection will also shed some light on why FDTD is the most appropriate method for this study.

Finite Element Method

The finite element method is a widely used numerical method for solving PDEs in *space* variables — unlike FDTD which handles variables in space *and* time. Similar to FDTD, it divides a problem into smaller parts called finite elements. These finite elements are made by constructing a 3D mesh of the object being modelled. These meshes can be of different shapes [**?**], which are in contrast to the usual cubic shapes of the FDTD subdomains (Yee Cells).

When compared to FDTD, FEM can handle more complex geometries, since the meshes can be configured in different ways other than a cube [**?**]. However, this leads to computational complexity. This is where FDTD has an advantage — it is easier to implement. Also, FEM was initially developed for structural analysis [**?**], while FDTD was formulated for the purpose of numerically computing Maxwell's equations [**?**].

Method of Moments

The Method of Moments (MoM) uses the integral form of Maxwell's Equations instead of their differential forms [**?**] [**?**]. Like the Finite Element Method, it also constructs meshes on the surface of the model. However, unlike both FDTD and FEM, MoM operates only on the boundaries of the computational domain rather than the entire space.

When compared to the two other methods, MoM is more efficient when the area-to-volume ratio is small. However, it becomes more computationally expensive as the complexity increases. Matrix sizes also increase with complexity [**?**].

More Properties of FDTD

Among the properties mentioned above, here are some more properties of FDTD according to Taflove [**?**]:

1. It is intuitive and easy to implement
2. It is accurate and robust
3. It is a time-domain technique — it can cover a wide range of frequency responses using a single simulation
4. It does not require matrix inversion
5. It efficiently scales well with parallel CPUs

The properties mentioned above were the main driving force for several researches. Studies have used the FDTD method for its simplicity [**?**, **?**, **?**]. The efficiency of FDTD in solving for

Maxwell's equations was a key feature for one study [?]. Another study mentioned that they chose FDTD for its wide coverage of frequencies [?]. In this other study, they mentioned that they used FDTD because it does need matrix inversion [?].

The project is an exploration of combining an electromagnetic wave simulation method with Schwarz Decomposition Method, which would verify if the data between two subdomains are accurate with each other. In order to maximize the chance of success, a suitable method must be chosen. Using the above properties, especially the simpleness, easiness to implement, and accuracy, FDTD is an excellent choice for this project.

4.1.2 Implementations of FDTD

This subsection focuses on how studies implement FDTD. For this document, there will be two classes of implementations. First is the number of FDTD dimensions, and the second is how the FDTD method is combined with other methods.

Number of FDTD Dimensions

The FDTD method was originally formulated to solve for Maxwell's equations. Since Maxwell's equations describe quantities that are in three spatial dimensions, FDTD also has to do operations along the three dimensions. However, due to the staggered nature of the fields in the Yee cell, the FDTD method can be reduced to 2D or 1D [?], depending on the need of the user.

In the researches found under FDTD, most of them were either 3D [?, ?, ?] or 2D [?, ?]. Only one study was found to incorporate 1D FDTD [?]. However, since this project aims to combine FDTD with the Schwarz method, it would be best to start with simple, one-dimensional models. Also, this 1D setup can be scaled up to 2D and 3D should it prove to be successful. Implementing 1D FDTD is easy since references for this are readily available [?, ?, ?, ?].

Configuration of Computational Domain

FDTD operates on cells within its computational space. These cells can either be equally divided according to the size of the computational domain or be non-uniformly divided based on the complexity of whatever model is in the computational space. The former is the more basic way of implementing FDTD, and many studies still use this. The latter, however can be used to improve the performance and efficiency of the traditional FDTD implementation.

Many of the structures being modelled nowadays are not uniform — some of their parts might be more complex than other parts. By making the cells smaller at the more complex parts, the accuracy of the modelled EM phenomenon is increased. At the same time, less computational resources are allotted to the larger cells located along the less complex parts of the model. Studies have shown the improvements in computation time when using non-uniform grids. A study [?] showed that using non-uniform grid greatly decreases the computation time, thus improving efficiency. In another study [?], a modified FDTD algorithm that uses non-uniform grids was compared to the conventional FDTD implementation. The study noted that the computational efficiency of the former was superior to the conventional method.

Another class of FDTD configuration is the non-uniform computational domain. In all of the previous works mentioned, none of them specified whether they did the study in a serial or in a parallel-configured computer. However, in one study [?], a parallelized FDTD method in conjunction with a variable domain size was used. When compared to the previous computational domain configuration, the domain size per CPU core is non-uniform rather than cell sizes within a domain. This gives off an opportunity to assign more complex parts of the simulation to more powerful CPU cores.

The aim of this project is more of a proof of concept – that FDTD and Schwarz can be combined. Thus, the focus will be on trying to make them work together first. Computational efficiency of the devised method will only be secondary. Also, the project will deal only with one dimension. The need to vary the cell or computational domain size is unnecessary since the efficiency gains will not be seen with such a simple model. Thus, the uniform grid configuration is more fitting to be used in this project.

FDTD in Conjunction with other methods

Several studies have used the FDTD method alongside other techniques in order to optimize its performance. In one study [?], they used a Laplace Transform method to simplify the conductivity of their model. Other studies used different methods to improve the performance of their simulations. The moving window method was used in another study [?] to reduce the computational resource usage of their simulation, while another study [?] used topology optimisation for the same reason. Other methods were also presented by one study [?], such as Alternate Direction Scheme, subgridding, and hybrid approaches to be used in conjuction with FDTD. Another class of FDTD optimisation is to perform it on a parallelized system. This will be further discussed in subsection 4.2.1.

Following the idea that FDTD can be used alongside other methods, it can be inferred that the Schwarz method can also be used along with FDTD.

4.1.3 Simulation Models in Other Works

Due to the ubiquity of FDTD in the electromagnetic simulation space, an extensive amount of studies used it to model their objects. Examples of structures modelled using FDTD in other works are thin conducting sheets [?], plasma sheath of a hypersonic vehicle [?], graphene sheets [?], [?], 3D photonic crystals [?], and transmission lines [?], among many others.

For this project, complex structures such as those mentioned above will not be used. Two simple models will be used instead. The first model is a computational domain split into cells with varying refractive indices. The second model is called Bragg Grating. More information on these two models will be discussed in subsection 6.1.2

4.2 Parallel Computing

Parallel computing can be divided into two types depending on how the memory is accessed: shared memory (SM) and distributed memory (DM) architectures [?, ?, ?].

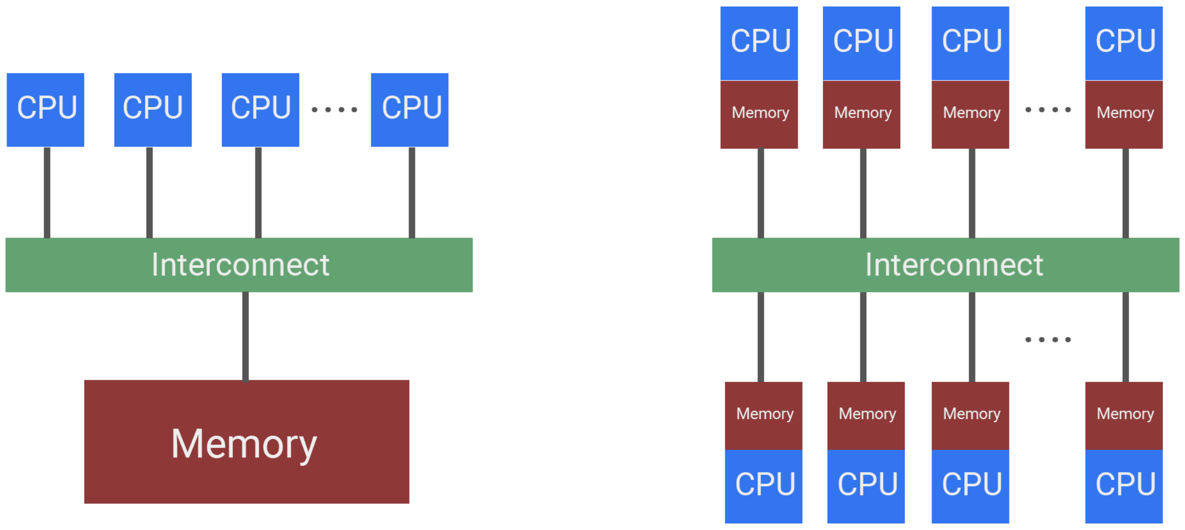


Figure 4.1: Shared Memory (SM) (left) and Distributed Memory (DM) (right) architectures [?].

Figure 4.1 shows the two memory architectures used in parallel computing [?]. SM architecture shows that each processor shares a memory block in the system while in DM, each processor has an independent memory block (memory is 'distributed' in each processor) that cannot be accessed by other processor. In the context of FDTD and computational domain, SM assigns each subdomain to a thread or process in a node while DM assigns each subdomain to a node in a cluster of computers [?].

In order to facilitate communication between the different processors, message passing interface (MPI) protocol like MPICH or Intel MPI library is usually used (for DM) [?, ?, ?] and multi-threading libraries like OpenMP (for SM) [?, ?, ?]. MPI is a protocol generally used for communicating between independent processors in a network [?] while OpenMP can be thought of as MPI's equivalent in a multi-threading context (where there are multiple processes in a single CPU) that is popular in parallel computing research [?]. There are also research [?] that combines the use of MPI and OpenMP such that they have a hybrid memory architecture (SM and DM) such that the computational domain is divided into several nodes while each node has multiple threads that divide the subdomain into smaller subsubdomain.

In the context of this project, the primary focus will be on SM architecture that utilizes OpenMP because it is generally easier to implement given the small timeframe of the project and will be cheaper to implement because of the rising demand of cheap multi-core computers that has good single core performance [?]. MPI and DM can still be considered for the project if time allows for it as the project progresses.

4.2.1 Parallel FDTD Research

Research in implementing FDTD in parallel configuration has already progress in the past decade. A common methodology in these studies [?, ?, ?, ?, ?, ?] shows that parallel FDTD algorithm use spatial division (or domain decomposition) of the computational domain is done and mapping each subdomain to either a thread (for SM type of parallel computing) or node (for DM type of parallel computing). This method of domain decomposition is intuitive because of how FDTD algorithm works that loops in the whole computational domain. Improvements on this methodology are done in [?, ?, ?, ?, ?] by introducing overlaps in the subdomains (also referred as 'ghost cells') that serves as storage of the boundary values of the adjacent subdomains for more efficient processing in each time step of the FDTD algorithm. Research in [?] also shows a different method of setting the computational domain by using a non-uniform cell size such that smaller cell size is present in parts of the domain which has intricate/complex designs while larger cell size has simpler designs in the domain to increase the efficiency and minimize load problems in the computational domain.

Another technique that is implemented in recent research is dividing the computational domain in only one direction to minimize the data communication overhead between adjacent subdomains [?, ?, ?, ?]. This minimizes the amount of data transfer between each subdomains since fewer artificial boundaries between subdomains are created. Other researches include [?] that uses a different data structure (compressed octree) to alleviate load balancing problems in the computational domain brought about by complex geometries of the simulation models. Although these method are considered to be a parallel process, the data transfer between adjacent subdomain is still a synchronous event that needs to be done by each subdomain. This may result in some subdomain 'waiting' for all of the subdomain to be done processing before initiating the data transfer of the adjacent subdomains.

Data transfer between adjacent subdomains is evident across different methodologies [?, ?, ?, ?, ?, ?, ?, ?] by using a master node/thread that synchronizes every subdomain's work in each iteration [?]. Improvements on minimizing the data communication overhead are also done in [?] by using a separate unit for communicating between adjacent subdomains and using a different communication protocol (Scalable Coherent Boundary Protocol or SCI) than the traditional MPI that results in 40% increase in speedup than the common method.

Since most researches in parallel FDTD uses DM architecture, they have different procesors as well as memory blocks used in each node/subdomain. These nodes might have small discrepancies in how they process the data as well as the amount of noise introduced while processing the data that might cause cascading errors in the results. Utilizing Schwarz Algorithm (as discussed in chapter 3) might help in minimizing errors that appear (as a validation method for the results) so that the system is reliable and accurate since most research in parallel FDTD do not check for the convergence of each subdomain. Also, taking advantage of consumer grade multi-core computers in parallel computing is a good compromise in performance and cost when compared to using supercomputers or clusters of high performance computers.

4.3 Schwarz Method

The Schwarz alternating method was first proposed in 1870, but is still a popular topic of study up to the present. It is a method for solving Laplace PDEs on irregular domains. It's main idea is to solve for an irregularly shaped domain using its parts that are regularly-shaped [?].

Several studies [?, ?, ?] have used Schwarz method in conjunction with other numerical methods, but these studies do not deal with electromagnetic problems or FDTD. There has not been any literature on a combination of the FDTD and Schwarz methods as far as the proponents' knowledge goes. However, there are studies which solved electromagnetic problems using the Schwarz method in conjunction with other methods. In [?], a convection-diffusion problem was solved using parallel iterative methods. In [?], an optimal iteration was developed to extend the application range of the Schwarz method. In [?], the Schwarz method was modified to use partial overlapping regions to solve for a boundary-value problem. The studies presented above show that

Schwarz method can be modified and be used in conjunction with other computing techniques. This implies the possibility of using the Schwarz and FDTD method together.

This project aims to combine Schwarz method with the FDTD method in a parallel computing environment. Section 4.2 provides some studies that already implemented FDTD in parallel. However, none of them were able to show whether the exchange of information between the sub-domains are accurate. As stated previously, it may be possible to use Schwarz in conjunction with the FDTD method. If that is the case, then it can be an improvement to the current parallel FDTD studies.

4.4 Cloud Computing

4.4.1 Infrastructure as a Service

Infrastructure as a Service (IaaS) is a type of cloud computing service model that are used when the cloud service providers will provide the memory, storage, computing resources and network while the customer will provide the necessary operating system and software necessary for their needs [**?**]. This service allows for the customer the “lowest level of control of the resources in the cloud” [**?**]. In essence, this is basically a virtual machine in the cloud where the user can set their own computational needs and install their custom software. This service model is beneficial to parallel computing as it can reduce the cost of acquiring hardware and maintenance while it is being used

Popular IaaS from different service providers include: Amazon’s Elastic Compute Cloud (EC2), Google Cloud Compute Engine, and Microsoft Azure Virtual Machines. These services have similar features that offer pre-installed virtual machine instances or installing custom images that contain necessary software and configuration for the user’s needs and duplicating an instance of a newly created VM can also be done [**?**, **?**, **?**]. These features are advantageous in creating a cluster of nodes in the cloud since it will be cost-effective and easy to creating a cluster of similar nodes by using the different cloud platforms.

Although commercial cloud services are easier to use and popular in research, open source cloud frameworks such as SciCloud and Hadoop are also being developed as a way to creating a private cloud service in universities for efficient use of computational resources [**?**].

4.4.2 FDTD Cloud Computing Research

As discussed in section 4.2.1, many researches are done on parallel computing and some of them have already explored the idea of implementing these algorithms in the cloud [?, ?, ?, ?].

Most of them [?, ?, ?] uses Amazon EC2 while [?] uses cloud computing framework like SciCloud and Hadoop. In [?], identical setups (FDTD code, MPI libraries, and configuration) are done on a local cluster of computers and on Amazon EC2 instance. The results of a series of tests shows that there are no significant drawbacks on using a cloud setup in comparison on a local setup. In another study [?], the use of cloud computing in solving EM problems using parallel FDTD is not worse than using a supercomputer. The results of these studies show that cloud computing does not provide any significant advantage in the speed and accuracy of the algorithm since their results are similar to local setups.

In these studies, the implementation of the algorithm in the cloud is similar to how the algorithm is implemented on-site so it seems that the main purpose of utilizing cloud computing in these studies is to ascertain the viability of cloud computing for research. The ability of cloud computing to assign resources depending on the demand also helps balance the computational power and cost since as most research in parallel FDTD show, efficiency usually drops as the number of nodes (processors) increase in a simulation run. So, it is evident that parallel computing and cloud computing is an effective and viable solution for cost-effective computing resources as the studies shows positive results in using cloud resources.

4.5 Synthesis

The works presented in each subsection greatly aided in the positioning of this project among the current studies in FDTD, Schwarz method, and parallel computing. They provided information on how and why the methods were used in the context of numerically solving for electromagnetic phenomena. With the help of the works mentioned, the proponents now have a foundation to which project objectives can be built upon. Other aspects of the project such as the scope and methodology can also be greatly influenced by these works. Combining all of the ideas from the different literatures, the proponents believe that this undergraduate project can help in alleviating the need of powerful hardware in solving electromagnetic problems.

Chapter 5

Problem Statement and Objectives

5.1 Problem Statement

Students predominantly have no access to powerful computational devices when doing electromagnetic simulations. Such simulation jobs are difficult to do on low-end hardware due to their mathematical complexity, complex design, and scale. Electromagnetics usually involves partial differential equations, which needs to be computed numerically for computers to solve them. The Finite Difference Time-Domain algorithm is a popular numerical method used for EM simulation problems due to its simplicity and fast processing.

The gap between the lack of powerful hardware and the need for EM simulations can be narrowed by implementing an established FDTD algorithm in a parallel environment. This implementation is parallelized using domain decomposition via the Schwarz method. By implementing this method, workflows may run faster and can be more accessible while maintaining accuracy and reliability.

5.2 Specific Objectives

1. Develop serial Finite Difference Time Domain code in 1D based on currently existing FDTD algorithms.

2. Implement the developed 1D FDTD code in serial and parallel configuration using similar simulation parameters.

3. Use Schwarz method to validate whether or not the data computed in the parallel configuration is accurate with respect to the serial configuration and the analytical solution.

4. Assess the speedup and efficiency of our developed code simulation results on the three configurations, then compare the results to the open-source FDTD code run on the same configurations.

5.3 Scopes and Limitations

This part discusses the scope and limitations of the project to properly set the boundaries of the research.

5.3.1 FDTD

For this project, only Ampere's Law and Faraday's Law of Induction will be focused on. The project's scope is only 1-dimensional plane wave propagation, and the last two Maxwell's equations are enough to model it. Ampere's Law and Faraday's Law of Induction shall be known as the Curl Equations for the rest of this document.

A few assumptions for Maxwell's Equations are made for this project. These assumptions will simplify the calculations to be made, and are enough to satisfy the goal of this project. The assumptions are as follows [**?**]:

1. There is no charges or current sources in the system.
2. The materials used are linear, isotropic, and non-dispersive.
3. Cell size (Δz) is uniform in each part of the computational domain.

As a result of these assumptions, the Curl Equations will reduce to the following:

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} \quad \text{and} \quad \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{5.1}$$

5.3.2 Schwarz Alternating Method

The assumptions for the Schwarz Alternating Method for this project is shown below:

1. Length of overlapping region must be greater than 0 ($\Omega_i \cap \Omega_{i+1}$).
2. Each subdomain must have an equal amount of cells.
3. Convergence is checked after each cell in the computational domain is solved.

The first assumption deals with the requirement of convergence [?] for the algorithm since the overlapping region is used to check for convergence. The second assumption is related to the FDTD-Schwarz Algorithm since having a uniform size in each subdomain can simplify the implementation of the algorithm. Also, this assumption will prevent any load-balancing techniques to be done. The last assumption is also done to prevent the algorithm from going to the next iteration without ensuring that the current iteration has converged successfully.

5.3.3 Configuration

In this project's context, although the proposed solution is parallel computing in the cloud, the methodology (in chapter 6) ignores the cloud computing aspect since the steps that need to be done on-site and in the cloud setup are similar (as shown in subsection 4.4.2). The only difference is on how the set up the virtual machines or local computers before installing the necessary software and configurations for the parallel FDTD algorithm.

Chapter 6

Methodology

6.1 Design

6.1.1 Parallel FDTD via Schwarz Alternating Method

The FDTD algorithm and Schwarz Alternating Method will be combined to create an FDTD algorithm that can be used in a parallel configuration. The purpose of including the Schwarz Alternating Method is to have a consistent way of validating the convergence of the solution when the computational domain is partitioned into subdomains. Most of the research [?, ?, ?, ?, ?, ?] done does not check the convergence of the solution, but simply partition the computational domain and facilitate the data exchange between adjacent subdomains due to the dependence of the equations 2.8 and 2.7 to the adjacent cells. The flowchart for the combined algorithm of FDTD and Schwarz Alternating Method is shown below:

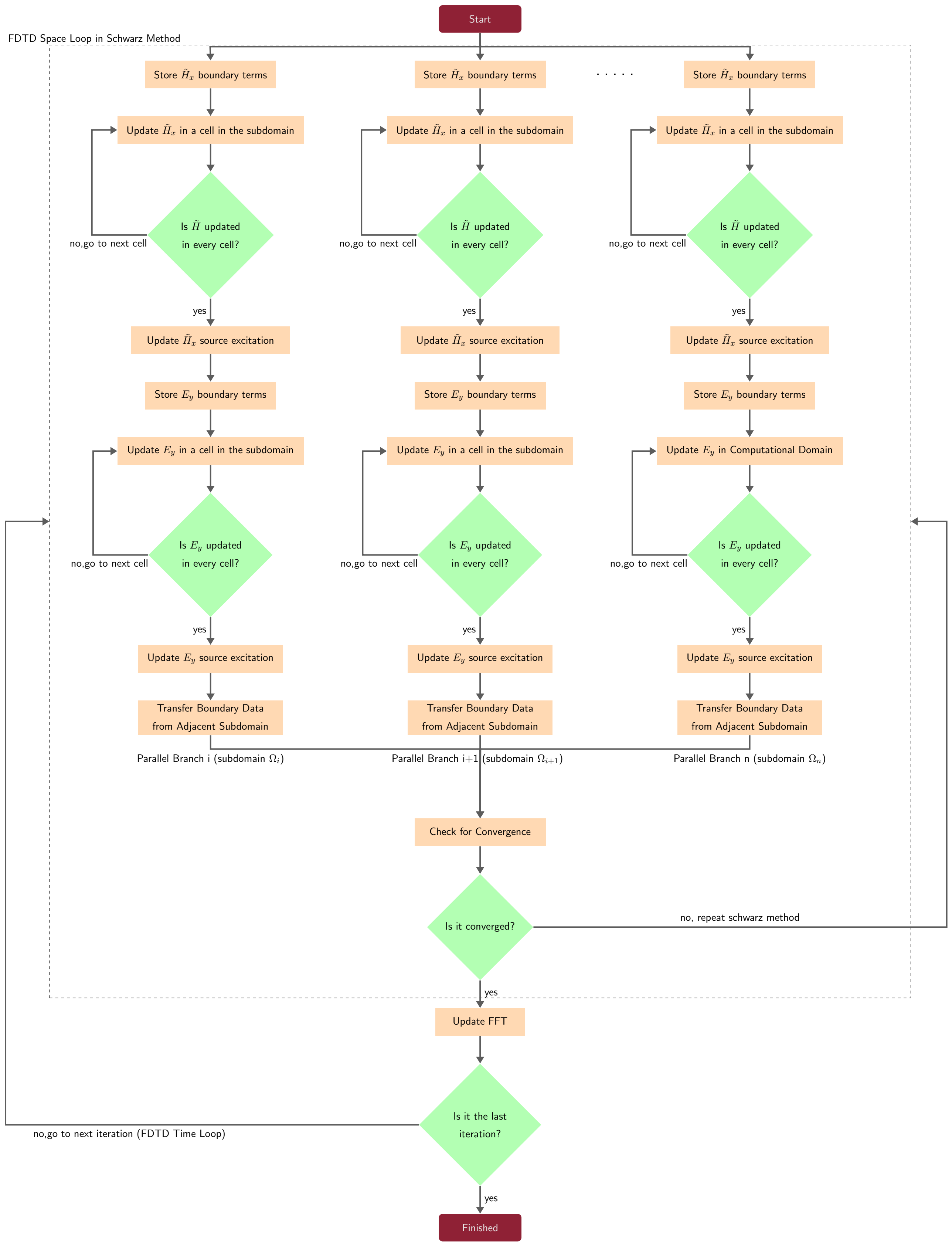

Start
FDTD Space Loop in Schwarz Method
Store $\tilde{H}_x$ boundary terms
Update $\tilde{H}_x$ in a cell in the subdomain
Is $\tilde{H}$ updated in every cell?
no,go to next cell
yes
Update $\tilde{H}_x$ source excitation
Store E_y boundary terms
Update E_y in a cell in the subdomain
Is E_y updated in every cell?
no,go to next cell
yes
Update E_y source excitation
Transfer Boundary Data from Adjacent Subdomain
Parallel Branch i (subdomain Ω_i)
Store $\tilde{H}_x$ boundary terms
Update $\tilde{H}_x$ in a cell in the subdomain
Is $\tilde{H}$ updated in every cell?
no,go to next cell
yes
Update $\tilde{H}_x$ source excitation
Store E_y boundary terms
Update E_y in a cell in the subdomain
Is E_y updated in every cell?
no,go to next cell
yes
Update E_y source excitation
Transfer Boundary Data from Adjacent Subdomain
Parallel Branch i+1 (subdomain Ω_{i+1})
· · · · ·
Store $\tilde{H}_x$ boundary terms
Update $\tilde{H}_x$ in a cell in the subdomain
Is $\tilde{H}$ updated in every cell?
no,go to next cell
yes
Update $\tilde{H}_x$ source excitation
Store E_y boundary terms
Update E_y in Computational Domain
Is E_y updated in every cell?
no,go to next cell
yes
Update E_y source excitation
Transfer Boundary Data from Adjacent Subdomain
Parallel Branch n (subdomain Ω_n)
Check for Convergence
Is it converged?
no, repeat schwarz method
yes
Update FFT
Is it the last iteration?
no,go to next iteration (FDTD Time Loop)
yes
Finished

Figure 6.1: FDTD-Schwarz Algorithm Flowchart

The combined algorithm has three major processes,

1. FDTD time loop

2. Schwarz Algorithm

3. FDTD space loop

To start, after the pre-processing is done, the algorithm will enter the FDTD Time Loop which involves these steps:

1. Schwarz Algorithm

2. Updating the Fourier Transform and checking the stop condition

Similar to what was done in the FDTD algorithm, the FDTD time loop is the outer loop of the algorithm that iterates through time which means that after every iteration, there will be a 'snapshot' of the current field values in the current time step. Since the Schwarz Algorithm is integrated, it will be done for every iteration to make sure that each 'snapshot' in every time step converges. After the Schwarz algorithm is done in this iteration, the Fourier transform is updated for the reflectance and transmittance coefficients. Fourier transform is outside of the Schwarz algorithm to prevent wasteful computation since after every Schwarz Algorithm run, the solution is converges unlike if after every iteration inside the Schwarz the Fourier transform is included, there will be unnecessary computations since the Fourier Transform of the fields will be wasted if the subdomains do not converges. The stop condition is similar to the stop condition of the FDTD algorithm where the algorithm will stop and proceed to post-processing if the current time step is equal to the last time step specified in the pre-processing stage.

The Schwarz Algorithm and FDTD Space Loop are integrated with each other such that the first step in the Schwarz loop in subsection 3.2 where the PDEs in the subdomains are solved is the FDTD Space Loop. In simpler terms, the first half of Schwarz loop where the operation can be done asynchronous with respect to each subdomain is the FDTD Space Loop because it will solve for the PDEs (FDTD update equations) and solve the external boundary conditions of each subdomain.

The steps in Schwarz Algorithm (in the FDTD-Schwarz Algorithm):

1. FDTD space loop

2. Transfer the data from the adjacent subdomain/s to the inner boundary of the subdomain

3. Check for convergence of each subdomain

The second half of the algorithm will still be done synchronously with respect to each subdomain because it needs the updated values in each subdomain for the internal boundaries of the subdomains. After the transfer of data, the convergence will be validated using the error equation and the condition in equation 3.1 similar to the process in Schwarz Algorithm. At the end of Schwarz-FDTD Space Loop, if the solution converges, the algorithm will proceed to step 2 of FDTD Time Loop, otherwise the algorithm will start step 1 of Schwarz-FDTD Space Loop and run the FDTD Space Loop again.

The project will implement this combined algorithm because this algorithm provides FDTD to validate the results of its parallel configuration. The convergence of the solution in each iteration will be important because the results of each subdomain might not be correct if the convergence is not validated in every time step.

6.1.2 Simulation Models

The simulation models that will be used in the project will be discussed here. These models are **plane wave propagation in different media** and **bragg grating**.

Plane wave propagation in various media

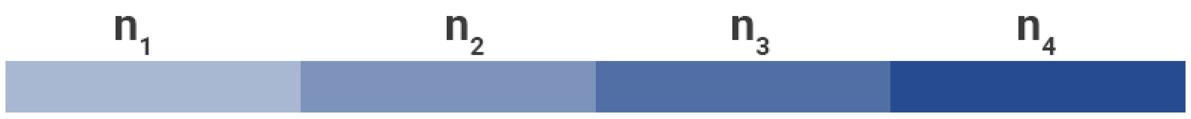


Figure 6.2: Different materials with increasing refractive indices from left to right

Plane wave propagation is one of the simplest EM simulation that can be done in FDTD. As shown on Figure 6.2, the different materials are stacked side to side such that their refractive index is increasing from left to right.

Bragg Grating

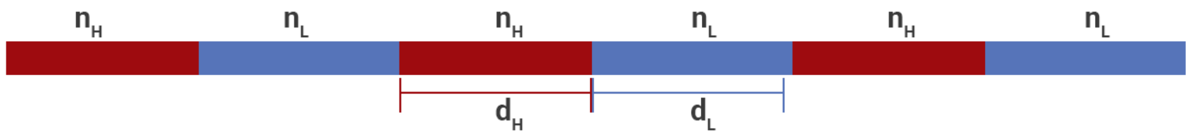


Figure 6.3: Bragg grating in 1D

This type of grating uses an alternating configuration of different materials with high and low refractive index [?] as shown in Figure 6.3. This type of grating is primarily used in preventing certain wavelengths of electromagnetic waves from passing through the grating [?]. This model is selected as one of the simulation models that will be used in the project since this can be done on 1D assuming that the other directions (y and z axis) have similar refractive index.

$$d_H = \frac{\lambda}{4n_H} \tag{6.1}$$

$$d_L = \frac{\lambda}{4n_L} \tag{6.2}$$

The design of a Bragg grating model can be done by adjusting thickness of each material based on 6.1 and 6.2 since the wavelength λ is proportional to the thickness of the two materials (d_H, d_L). This means that the higher the target frequency to reflect, thinner thickness for each material in the grating is needed.

6.1.3 Machine Configuration

The hardware specifications of the machine that will be used in this project are not yet determined due to different constraints but the minimum general specifications are determined and shown on the table below:

Table 6.1: Machine Configuration

Specification	**Value**
Operating System	Linux (Ubuntu)
CPU	Multi-core processor
Programming Language/s	C++ and Python 3

The operating system chosen was Linux (specifically Ubuntu) because of convenience but it is expected that the code base will be compatible with any Linux distributions as long as the dependencies are properly installed. The CPU of the machine must be a multi-core processor since the project can take advantage of the number of cores in increasing the amount of subdomain in each simulation. The programming languages used will be C++ and Python in combination with each other since C++ is generally faster in computing then Python [**?**] while Python's modules in data visualization is easier to use than C++'s libraries.

6.2 Implementation

In this part, the specific details of the algorithm as well as simulation parameters will be discussed in detail as well as the system's input/output handling for ease of use. The main algorithm process will not be discussed here because it is already discussed in section 6.1.

6.2.1 Pre-Processing

In the pre-processing stage, the initialization of the simulation parameters as well as the input handling of the system occurs. There are two main parts in the pre-processing:

1. Input handling; and
2. Initialization of simulation parameters and computational domain.

6.2.1.1 Input Handling

In this part, the details about the input files and formats will be discussed. The file type of the input file is a **.csv file**. This file type was chosen because it can be easily read by the system through built-in libraries for simpler parsing. The input name of the file must be in this format "**sim_model.csv**"The format of the csv file will be as shown below:

Table 6.2: Format of Input File

Column 1	**Column 2**	**Column 3**	**Column 4**
d_1	μ_1	ϵ_1	f_{MAX}
d_2	μ_2	ϵ_2	source
d_3	μ_3	ϵ_3	n_{model}
d_4	μ_4	ϵ_4	t_{sim}
.	.	.	
.	.	.	
.	.	.	
d_n	μ_n	ϵ_n	

So, in this format, columns 1 to 3 describe the model's layer where each row corresponds to a model's thickness in meters (d), magnetic permeability (μ), electric permittivity (ϵ), and n specifies how many layers in the simulation model. Column 4 describes the source excitation that will be used in the simulation where f_{MAX} is the maximum frequency that will be included in the source while source type specifies whether the source is a gaussian source (integer 0) or sinusoidal source (integer 1) as well as the total number of layers (n) of the model and t_{sim} specifies how long the simulation time (in seconds). The input file does not indicate how many cells will have in each layer because the cell size Δz is computed at every simulation so it is better to specify the physical distance of the layers in the simulation model.

6.2.1.2 Initialization of simulation parameters

The simulation parameters that are not needed to be computed at every iteration are done here to save computation time. The list of simulation parameters initialize are shown below:

Table 6.3: Initialized Simulation Parameters

Parameter	
c_0	299792458.0
ϵ_0	8.8541878128e-12
μ_0	1.25663706212e-06
Boundary terms	Array of size (1,2)
Δz	0
Δt	0
Source	Vector
Update Coefficients (for E and $\tilde{H}$)	Vector

So, the initialization of these parameters are done before the simulation proper to lessen computation time per iteration since they are only needed to be initialized once. The following subsections discuss how the cell size and time step and update coefficients are computed while the computation for the source excitation will be discussed in the next section.

6.2.1.3 Setup of Computational Domain

The computational domain is the space where the electric and magnetic fields are computed in the FDTD algorithm as well as where the simulation model is located. To set up the computational domain, there are several considerations that need to be taken into account to ensure the convergence of the simulation as well as to prevent numerical dispersion as discussed in [**?**].

Grid Dispersion

In computing for a good grid resolution for the computational domain, two factors are usually considered: (1) Minimum Wavelength and (2) Minimum Dimension [?]. This is related to the concept of grid dispersion that can cause numerical instability in FDTD. This project will only deal with uniform grid resolution (meaning constant delta z) so that the computations will be simplified in 1 dimension context.

$$\lambda_{min} = \frac{c_0}{f_{max} n_{max}} \tag{6.3}$$

The minimum wavelength is usually considered when computing the grid resolution to prevent numerical dispersion. The computation of the minimum wavelength for the grid size is done because the smallest wavelength in the FDTD simulation must be sampled properly to satisfy Nyquist sampling theorem [?]. The minimum wavelength is computed using equation 6.3 that considers the max frequency needed in the simulation f_{MAX} and the largest refractive index in the computational domain n_{MAX}.

$$\Delta z_\lambda \approx \frac{\lambda_{min}}{N_\lambda}, \;\; N_\lambda \geq 10 \tag{6.4}$$

The rule of thumb in computing the grid resolution when considering the minimum wavelength is that the number of cells (N_λ) that resolve the minimum wavelength must at least be 10 cells. This can be increased depending on the type of material in the computational domain. For example, the recommended number of cells N_λ for low contrast dielectrics are 10 - 20, 20 - 30 for high contrast dielectrics, 40 - 60 for metallic materials, and 100 - 200 for plasmonic devices. In this project, the focus will be on dielectrics so the number of cells will be at a maximum of 40 cells to ensure that the refractive index of most dielectrics are considered.

Device Dimension

This part considers the simulation model used in the computational domain. The consideration with regards to the device dimensions are that the minimum dimensions in the simulation model must be resolved in the grid resolution. The minimum dimension d_{min} of the simulation model is the smallest feature of the model and in the context of the project, since the scope is only on one dimension (z dimension), this means that the minimum thickness of a layer should be considered [?].

$$\Delta z_d \approx \frac{d_{min}}{N_d}, \;\; N_d \geq 1 \tag{6.5}$$

The grid resolution due to the minimum dimension in the model is shown in 6.5. As shown, the minimum dimension of the model is considered by setting the number of cells that represent the minimum dimension to at least 1 cell. However, it is recommended to set it at $N_d = 4$ so that the features of the model are properly represented [?].

$$\Delta z = min[\Delta z_\lambda, \Delta z_d] \tag{6.6}$$

The final grid resolution can be computed by selecting the minimum grid resolution between the grid resolution due to minimum wavelength and the grid resolution due to minimum model dimension. This cell size or grid resolution will be used in the FDTD simulation.

There is another consideration in grid resolution, the critical dimension of the model, which is the actual dimension (total dimension) of the simulation model. This consideration will not be considered in this project because of the project's scope in 1D and the models used fit in the Yee cells without any problem. This consideration is important when creating a higher dimension FDTD simulation because the cells might not have homogenous material if the critical dimension is not considered. This consideration makes sure that the model in the computational domain fits each cell properly such that the border of the model is also the border of a Yee cell.

Courant-Friedrichs-Lewy Stability Criterion (CFL Condition)

In an FDTD simulation, the change in electric and magnetic fields in a cell is only felt by the adjacent cells in each iteration, this results in the propagation of the fields to be 1 cell per 2 time step because of the staggered scheme of the electric and magnetic fields in the computational domain. This is also related to the fact that the speed of EM waves are different in different media and the maximum speed an EM wave can propagate is its speed in vacuum c_0. This is described by the *Courant Stability Criterion*:

$$\Delta t_{3D} < \frac{n_{min}}{c_0\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}} \tag{6.7}$$

$$\Delta t_{1D} < \frac{n_{min}\Delta z}{2c_0} \tag{6.8}$$

The basic description of CFL condition (as shown on equation 6.8) for FDTD is that the time step is restricted by the speed by which the EM wave is propagating in the highest speed (lowest refractive index) and the size of each cell in the FDTD grid [?] and [?]. The constant coefficient 0.5 is added in the CFL condition as a safety margin to make sure that the simulation will be stable [?]. If the CFL is not satisfied, the FDTD simulation will be unstable as time passes because as explained previously, in FDTD an EM wave cannot physically travel in more than one cell in two time steps [?]. The CFL condition prevents instability in FDTD simulation by restricting the size of the time step based on the inherent instability of FDTD if the EM waves travel in more than one cell size.

Layout of Computational Domain

Serial Configuration

After resolving the grid resolution and time resolution of the computational domain, the whole computational domain can be created. Following [?], when creating the computational domain, adding spacers (cells that have a refractive index of air/vacuum) can be used so that there is a space between the boundary of the computational domain and the simulation model. This will help in the visualization of the reflected fields and transmitted fields after it hits the simulation model or passes through it.

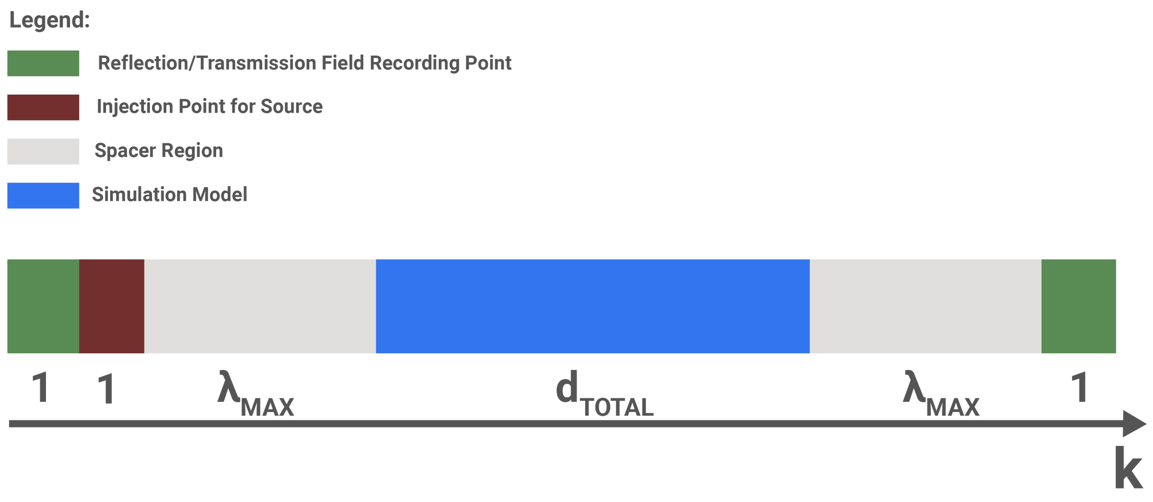


Figure 6.4: Layout of Computational Domain for 1D FDTD

As shown in Figure 6.4, The first and last cells are the reflection and the transmission recording point of the simulation (this will be further explained in the next subsection on why it works). The simulation model is in the center of the computational domain and the spacer regions, which at least have a minimum size of λ_{MAX} is on either side of the simulation model. The point of injection for the source is located at the left side of the computational domain, near the reflection recording point. The selected sizes are also shown on the figure above as shown where the recording point and the injection point of the source in the simulation is only 1 cell, while the spacer region is at least the size of the largest wavelength in the source while the size of the simulation model d_{TOTAL} is dependent on the input file.

Another function of the spacer region in the computational domain is to make the number of cells in the computational domain divisible to 2 so that the partitioning of the computational domain in the parallel configuration is easy and less prone to more complexities such that the subdomains have equal lengths.

Parallel Configuration

The process of creating and layout of the computational domain in the parallel configuration is the same as the serial configuration. An additional process is added after the layout of the computational domain is created. This process is the *partitioning of the computational domain into subdomains*. The number of subdomains will correspond to how many processes will be run in parallel in the FDTD simulation. The selected formula for getting the number of subdomains is shown below:

$$No.of subdomain = 2^x, \;\; x \in [1,5] \tag{6.9}$$

This equation (6.9) such that x is the order of simulations that will be done in this project (i.e. x = 1 is the first simulation run for parallel configuration where the number of subdomains will be 2). This ensures that the number of subdomains will always be even and the computational domain can be divided equally (every subdomain has equal length) so complexities on the length of subdomain that is not an integer value is prevented.

After the cell size and time step are computed, the update coefficients (see equation 2.9 and 2.10) can be calculated since they are dependent on these parameters. These will be a vector that have different values depending on what the refractive index is on each cell. These will be used in the update equations (equations 2.8 and 2.7) in order for the FDTD to take into account the material properties that the EM wave is propagating into while the time step and cell size is still considered.

6.2.2 Simulation Proper

This subsection will discuss the specific configurations of the two main methods used in this project. First, the FDTD Method's assumptions, boundary conditions, and source excitations are discussed. Second, the Schwarz Alternating Method's assumptions, initial guess, boundary conditions, and the convergence of the computational domain.

6.2.2.1 FDTD Algorithm

In subsection 2.5 the algorithm is discussed in the context of a one dimensional computational domain. The boundary conditions and source excitations that will be used in this project is discussed.

Boundary Conditions

As explained in subsection 2.5, there is a boundary in the computational domain because the simulation cannot extend the space infinitely so the computational domain is a closed space. However, to make the simulation behave similar to reality, the boundaries must act like the plane waves will not reflect in the boundaries and pass through them, making the computational domain like an open space. There are simpler boundary conditions (Dirichlet Boundary Conditions) that act like a barrier that reflects any EM wave that will not be discussed here but it will be shown in Chapter 7.

Perfectly Absorbing Boundary Condition (PABC)

PABC is a boundary condition used in FDTD as discussed by [**?**] and [**?**], that takes advantage of how the EM fields move in the Yee grid. The problem of FDTD update equations is that they are dependent on the adjacent cells. At the left side boundary of the computational domain (k = 0), the electric field update equation (equation 2.8) shows that it requires magnetic field values outside the computational domain (k = -1) . Similarly, at the right side boundary of the computational domain ($k = N_z$), the magnetic field update equation (equation 2.7) shows that it requires electric field values outside the computational domain ($k = N_z + 1$)[**?**].

As discussed in subsection 6.2.1.3, the EM waves cannot travel more than one cell per two time step [**?**] in FDTD so since the movement of the electric and magnetic fields are known, predicting the fields at the boundary by saving the value at the boundary in previous timesteps can make the fields propagate outside the computational domain. This is done in the FDTD algorithm (see figure 2.4) by saving the boundary terms in each iteration [**?**]. The update equations for the PABC boundary condition used are shown below:

$$z_{low} = [0,0], \;\; z_{low}[0] = z_{low}[1], \;\; z_{low}[1] = \tilde{H}_x|^0, \tag{6.10}$$

$$E_y|^0_{t+\Delta t} = E_y|^0_t + m^0_{E_y}\left(\frac{\tilde{H}_x|^0_{t+\frac{\Delta t}{2}} - z_{low}[0]}{\Delta z}\right) \tag{6.11}$$

$$z_{high} = [0,0], \;\; z_{high}[0] = z_{high}[1], \;\; z_{high}[1] = E_y|^{N_z} \tag{6.12}$$

$$\tilde{H}_x|^{N_z}_{t+\frac{\Delta t}{2}} = \tilde{H}_x|^{N_z}_{t-\frac{\Delta t}{2}} + m^{N_z}_{H_x}\left(\frac{z_{high}[0] - E_y|^{N_z}_t}{\Delta z}\right) \tag{6.13}$$

As shown on the equation above, each boundary value for each update equation is saved into a two sets of queue data structure, for the electric field update equation only the magnetic field at $k = 0$ is saved while in the magnetic field update equation, the electric field at $k = N_z$ is saved into its queue. The data queues are initialized to 0 at first since there is no field change in the computational domain. The saved boundary values are now used in the update equations as shown above. This boundary condition is only acceptable if the time step is computed using equation 6.8 where the refractive index n_{min} is the refractive index at the boundary of the computational domain and it can only be done in one direction so PABC cannot be used in higher dimensions of FDTD.

Source Excitation

In FDTD simulations, there are two most commonly used sources: (1) Gaussian Pulse and (2) Sinusoidal Source. These two sources will be used as the excitation for the simulations in this project.

Gaussian Pulse Source

A Gaussian pulse is a type of excitation that has a similar appearance to a Gaussian bell curve [?]. It is one of the popular source excitations in FDTD that are used in several studies [?, ?, ?, ?, ?, ?, ?] because it is composed of many frequency components. This allows a single FDTD simulation to analyze multiple frequencies. The equation of a Gaussian pulse in 1D is shown below:

$$g(t) = e^{-(\frac{t-t_0}{\tau})^2} \tag{6.14}$$

$$\tau = \frac{1}{2f_{MAX}} \tag{6.15}$$

In equation 6.14, t_0 is the time delay of the Gaussian pulse which has usual value of at least six times the pulse duration τ. This time delay is necessary so that the source is “eased” into the computational domain [?] and not doing so may result in abrupt change in fields that may cause numerical instability. τ, on the other hand, is the width of the pulse or duration and it is determined such that the pulse will have enough energy for the max frequency (f_{MAX}).

$$E_{src} = e^{-(\frac{t-t_0}{\tau})^2} \tag{6.16}$$

$$\tilde{H}_{src} = -\sqrt{\frac{\epsilon^{k_{inj}}}{\mu^{k_{inj}}}} e^{-(\frac{t-t_0+\delta t}{\tau})^2} \tag{6.17}$$

$$\delta t = \frac{n_{inj}\Delta z}{2c_0} + \frac{\Delta t}{2} \tag{6.18}$$

In FDTD, there are two fields to consider - **electric and magnetic fields**, it is important to have the electric and magnetic field component of the Gaussian pulse. They are similar to equation 6.14 but the only difference is the adjustment of the magnitude of the magnetic field because of the difference in magnitude between the two fields. Another difference is that the magnetic field must be adjusted such that it is delayed by half a time step because of how the FDTD uses staggered electric and magnetic fields and by convention, it is decided that the magnetic field exists at half time steps while electric field exists at whole time steps. δt is the adjustment in that regard where the first term is the time delay by half a cell while the second term is the time step difference by half [**?**].

Sinusoidal Pulse Source

A Sinusoidal source is a source excitation in the form of a sine wave. It is commonly used by adjusting the amplitude at $t < t_0$ to increase exponentially up to "ease the injection of the source similar to how the Gaussian pulse is "eased" into the computational domain [**?**].

$$A(t) = \begin{cases} e^{-(\frac{t-t_0}{\tau})^2} & \text{if } t < t_0 \\ 1 & \text{if } t \geq t_0 \end{cases} \qquad \tau = \frac{3}{f_0} \tag{6.19}$$

$$g(t) = A(t)\sin(2\pi f_0 t) \tag{6.20}$$

The constant f_0 is similar to f_{MAX} where f_0 must be the frequency of interest since the sinusoidal source usually only has one frequency component.

$$E_{src} = A(t)\sin(2\pi f_0 t) \tag{6.21}$$

$$\tilde{H}_{src} = -\sqrt{\frac{\epsilon^{k_{inj}}}{\mu^{k_{inj}}}} A(t)\sin(2\pi f_0 t) \tag{6.22}$$

$$\tag{6.23}$$

Similar to what was done in Gaussian pulse, the sinusoidal source has two components: electric and magnetic field and they use the same equation (6.20). The difference is the adjustment of the magnitude in the magnetic field using the permittivity and permeability in the injection point.

Total Field/Scatter Field (TF/SF)

In FDTD, there are several ways of injecting a source in the computational domain. First, hard source method which overwrites the field values using the values of the source excitation. This results in a barrier in the injection point in the computational domain that can skew the results because in reality, there should be no barrier in the injection point. Second, soft source method which adds the value of the source excitation to the previous value of the field values. Similar to the hard source, this produces a mirror pulse that propagates in the opposite direction but the injection point no longer acts like a barrier. These two methods are good starting point in FDTD simulations but they are rarely used in researches because they produce sources that travel in the two directions whereas in normal cases, sources only travel in 1 direction.

$$E_{total} = E_{incident} + E_{scatter} \tag{6.24}$$

$$\tilde{H}_{total} = \tilde{H}_{incident} + \tilde{H}_{scatter} \tag{6.25}$$

TF/SF is the method of injecting the source excitation by dividing the computational domain into two regions: Total Field (TF) and Scatter Field (SF) [?]. This also decomposes the electric field and magnetic fields as shown in equations 6.24 and 6.25 where the total field is composed of the incident fields which are the fields that propagate in the direction of the EM wave propagation while the scatter fields are the fields that are reflected and goes in the direction opposite of the direction of propagation [?].

TF/SF is usually done to minimize the load on the boundaries of the computational domain and to prevent the incident fields to interact with the boundaries [?] as this can prevent unnecessary reflections inside the computational domain. In the case of one dimension, the computational domain is split onto total and scatter field in Figure 6.4 where the injection point for the source excitation is the boundary between the scatter field (left) and the total field (right). This partitioning can help in injecting the source as a "one-way source" where there will be no source that propagates in the opposite direction [?].

$$\tilde{H}_x|_{t+\frac{\Delta t}{2}}^{k_{inj}-1} = \tilde{H}_x|_{t+\frac{\Delta t}{2}}^{k_{inj}-1} + (m_{H_x}|^{k_{inj}-1})\frac{(E_y|_t^{k_{inj}} - E_y^{src}|_t^{k_{inj}}) - E_y|_t^{k_{inj}-1}}{\Delta z} \tag{6.26}$$

$$E_y|_{t+\Delta t}^{k_{inj}} = E_y|_t^{k_{inj}} + (m_{E_y}|^{k_{inj}})\frac{\tilde{H}_x|_{t+\frac{\Delta t}{2}}^{k_{inj}} - (\tilde{H}_x|_{t+\frac{\Delta t}{2}}^{k_{inj}-1} + \tilde{H}_x^{src}|_{t+\frac{\Delta t}{2}}^{k_{inj}-1})}{\Delta z} \tag{6.27}$$

Since the computational domain is split into TF and SF, adjustments in the FDTD update equations must be made because of their dependence on the adjacent cells. The main cells in consideration are the cells k_{inj} and $k_{inj} - 1$. First for the magnetic field update equation, the previous update equation (equation 2.7) is modified to make sure that the electric field on the k_{inj} cell is a SF quantity by subtracting the source term for electric field. The modification is $(E_y|_t^{k_{inj}} - E_y^{src}|_t^{k_{inj}})$ as shown in equation 6.26.The update equation of the electric field on the other hand needs to make sure that the magnetic fields in the $k_{inj} - 1$ cell is a TF quantity. This is done by modifying the original equation (2.8) and adding the magnetic field component of the source by using this term $(\tilde{H}_x|_{t+\frac{\Delta t}{2}}^{k_{inj}-1} + \tilde{H}_x^{src}|_{t+\frac{\Delta t}{2}}^{k_{inj}-1})$ as shown in equation 6.27. These modifications in the original update equations are only applicable on the cells k_{inj} and $k_{inj} - 1$ since these cells are the only ones with inconsistent quantities. Any cells that are not the mentioned cells will always be either a TF quantity or an SF quantity [?].

6.2.2.2 Schwarz Alternating Method

In this part, the specific configuration of Schwarz Alternating Method will be discussed, more specifically on how the convergence of each subdomain is done.

Initial Guess

As discussed in subsection 3.2, the Schwarz Alternating Method needs an initial guess as the pre-processing stage of the main algorithm. This is needed in order to have a starting point in the algorithm. However, in the combined FDTD and Schwarz Alternating Method, the initial guess is not necessary. This is because the FDTD will have a good initial value at the beginning of the algorithm by setting all field values to 0. The initial value in every simulation is set to 0 (in all field values) because at the beginning of the simulation, there is no source excitation and one of the assumptions in the FDTD algorithm is that there is no EM source inside the computational domain.

Convergence

The main validation method of Schwarz Alternating Method is to check for convergence after solving the PDEs and the boundary conditions [?]. The convergence condition in equation 3.1 is the condition in the algorithm to determine if the subdomains converge. Most researches [?, ?, ?, ?, ?, ?, ?], uses L2 norm as the loss function for the convergence of Schwarz Alternating Method. The equation shown below is the general L2 Norm equation [?] and the equations that will be used in the project.

$$X_{\Omega_i} = \frac{||X_{serial} - X_{\Delta_i}||_2}{||X_{serial}||_2}, \quad i \in [left, right] \tag{6.28}$$

$$E_{error} = [E_{\Omega_{left}}, E_{\Omega_{right}}], \quad \tilde{H}_{error} = [\tilde{H}_{\Omega_{left}}, \tilde{H}_{\Omega_{right}}] \tag{6.29}$$

$$||E_{error}||_2 \leq \epsilon_{machine} \tag{6.30}$$

$$||\tilde{H}_{error}||_2 \leq \epsilon_{machine} \tag{6.31}$$

In subsection 6.1.1, the algorithm discussed that each subdomain is processed in each iteration of the Schwarz algorithm. This first part of the algorithm is discussed as the asynchronous part of the algorithm. The convergence of the subdomains is the second part of the algorithm which must be done synchronously because the first part of the algorithm needs to be finished. In the convergence, since there are two parameters that are computed in each subdomain (electric and magnetic fields), each subdomain needs to satisfy two convergence conditions (equations 6.30 and 6.31) to validate the convergence of the subdomain. Also, the values in each overlapping region in each subdomain are compared to the values in the serial configuration (in the same position in the computational domain). This is shown in equation 6.28 where the relative error of the values in the overlapping region when compared to the serial configuration's values are done. This is done on both subdomains that are connected (equation 6.29) and the L_2 norm of the E_{error} and H_{error} are computed as the parameter compared to the error tolerance.

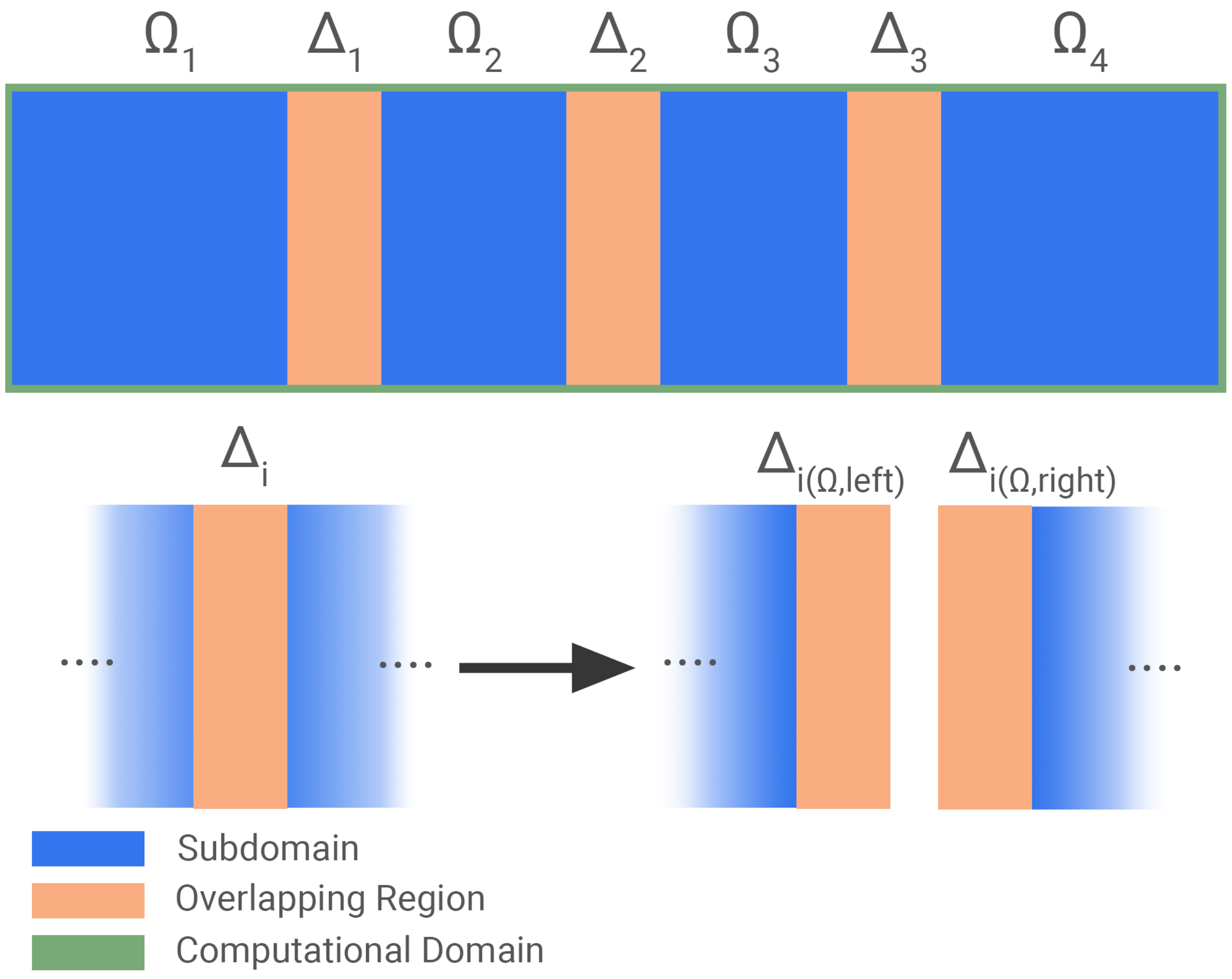


Figure 6.5: Layout of Partitioned subdomains in the Computational Domain (top), Composition of an overlapping region (bottom)

The process of convergence in Schwarz Alternating Method is done in each subdomain at least once because if a certain subdomain is in the middle, meaning they have two overlapping regions, the convergence process is done on each overlapping region. For example, in Figure 6.5, the computational domain is partitioned into 4 subdomains (Ω_1 - Ω_4) and thus it has three overlapping regions (Δ_1 - Δ_3). This shows that Ω_2 and Ω_3 have two overlapping regions ($\Delta_{i(\Omega,left)}$ and $\Delta_{i(\Omega,right)}$ in Fig. 6.5). In each subdomain, there will always be at least one overlapping region (in the case of a subdomain at the edge of the computational domain) and two overlapping region for the left and right adjacent subdomains (for the subdomains positioned in the middle of the computational domain).

The decided data transfer flow is that the convergence on Δ_1 is done on Ω_1 node, meaning

Pseudocode 1 Data Transfer in each subdomain for convergence checking

```
1: for subdomain $i = 1, 2, \dots n-1$ do
2:     Transfer $\Delta_{i(\Omega_{i+1})}$ data to $\Omega_i$
3:     Process convergence by using $\Delta_{i(\Omega_i)}$ and $\Delta_{i(\Omega_{i+1})}$ in $\Omega_i$
4: end for
```

the data from Δ_1 of Ω_2 is transferred to Ω_1 so that the convergence is processed. This pattern is repeated such that the left subdomain of the overlapping region is the node that will process the convergence. This means that the rightmost subdomain will not process the convergence on their connected overlapping region.

6.2.3 Post-Processing

After the simulation run has finished, the post-processing stage will begin and consolidate the simulation results and start data visualization. In the serial configuration, the process is simple: the simulation results are saved into several output files in .csv format. The format of the output file is shown below:

Table 6.4: Format of Output File

	Column 0	**Column 1**	**Column 2**	**.....**	**Column N_z**
Row 0	$X\|_{t=0}^{k=0}$	$X\|_{t=0}^{k=1}$	$X\|_{t=0}^{k=2}$	.....	$X\|_{t=0}^{k=N_z}$
Row 1	$X\|_{t=1}^{k=0}$	$X\|_{t=1}^{k=1}$	$X\|_{t=1}^{k=2}$	.....	$X\|_{t=1}^{k=N_z}$
Row 2	$X\|_{t=2}^{k=0}$	$X\|_{t=2}^{k=1}$	$X\|_{t=2}^{k=2}$	.....	$X\|_{t=2}^{k=N_z}$
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
Row N_t	$X\|_{t=N_t}^{k=0}$	$X\|_{t=N_t}^{k=1}$	$X\|_{t=2}^{k=N_t}$	.....	$X\|_{t=N_t}^{k=N_z}$

This format will be applied to the electric field, magnetic field, reflectance, and transmittance that will be saved in csv files. Each row in the csv file correspond to an iteration up to the last iteration N_t while the columns indicate the cell in the computational domain from k=0 to $k = N_z$.

In parallel configuration, the process is similar to the serial configuration but there is an extra initial step where the data from the different subdomains are compiled to create the solution for the whole computational domain. After that, the different simulation results are saved into csv files with formats similar to Table 6.4.

Data Plots

The simulation results will be visualised by plotting each iteration of the whole computational domain. Two plots will be created:

1. Superimposed plots of the electric and magnetic fields; and
2. Superimposed plots of reflectance, transmittance, and the sum of the previous two parameters

These plots are then compiled to create a video file to make the visualization of the FDTD simulation better to see the propagation of the plane waves.

To facilitate the creation of a video from a set of images, the filename format of the images will be: "**[parameter]_image_[iteration].jpeg**". The format specifies that [parameter] is the simulation parameters that are plotted, in this project there will only be two cases: (1) **EyHx** or (2) **TrRf** where they pertain to the electric field/magnetic field superimposition and reflectance/transmittance plots respectively.

The ffmpeg library is used to create a video from a set of images [**?**]. The bash command for the creation of a video from set of images is shown below:

ffmpeg -f image2 -framerate 200 -i [parameter]_images_%07d.jpeg -s [Width]x[Height] [output name].mp4

The bash command above creates a video from a set of images that are created by the program in .mp4 format. In this way, the behavior of the electric and magnetic fields inside the computational domain is easily visualize and data can be compressed.

6.3 Testing

The testing part of the methodology discusses how the testing will be done on each machine configuration and the different variables that will be changed in the configuration. The simulation accuracy of the project will also be discussed as well as the measurement of different metrics of performance for the parallel configuration.

6.3.1 Testing Configuration

The simulations done in this project are divided in two parts: (1) Serial configuration and (2) Parallel Configuration. The first part, is the reference simulation where the FDTD algorithm (see subsection 2.5) is done without any parallelization schemes. This is used as a reference so that the parallel-FDTD algorithm (see subsection 6.1.1) can be compared to a standard FDTD simulation to determine the accuracy of the parallel-FDTD algorithm. The second part is the simulations for the parallel-FDTD algorithm in which the parallelization scheme is used. In both parts, simulations in the chosen open source FDTD tools will be done to determine the accuracy of the project's algorithm with other tools.

In these simulations, there will be a default configuration of simulation parameters (in each machine configuration) where the independent variables will be changed one at a time to find out what parameters affect the performance and accuracy of the algorithm.

Table 6.5: Default machine configuration

Machine Configuration	Independent Variables	Default Value
Serial Configuration	Total Grid Size	$3(d_{total})$
	Simulation Time	100τ
	Model Complexity	Plane wave propagation in different media
	Source Excitation	Gaussian Pulse
Parallel Configuration	Total Grid Size	$3(d_{total})$
	Simulation Time	100τ
	Model Complexity	Plane wave propagation in different media
	Source Excitation	Gaussian Pulse
	No. of subdomain	2^3
	Size of overlapping region	$30\%(d_{subdomain})$

In table 6.5, the values that will be used in each configuration by default (before any independent variable are changed) is shown. Each independent variable that will be tested are explained below:

- **Total Grid Size** - is the size of the whole computational domain (see Figure 6.4) and the value is dependent on the size of the simulation model used (in this case 3x the size of the simulation model). This means that the spacer region must be increased such that the total size of the computational domain is 3x the size of the simulation model.

- **Simulation Time** - is the time in which the FDTD time loop will iterate. This is not the actual run time of the simulation but the set simulation time (t_{sim}) in the input file. This value is dependent on τ, which is based on the maximum frequency of interest in the source excitation (see equations 6.15 and 6.19) and in this case it is 100x the value of τ.

- **Model Complexity** - is the simulation model that will be used in the simulation run. The selected default simulation model is the Plane wave propagation in different media (Figure 6.2).
- **Source Excitation** - is the type of EM source that will be used in the simulations. The selected default source excitation is Gaussian pulse (equation 6.14).
- **No. of subdomain** - is the number of subdomains that will be used in the parallel configuration. The selected default value is $2^3 = 8$ subdomains.
- **Size of overlapping region** - is the portion of the subdomain that *overlaps* with the other subdomain. This value is dependent on the size of the subdomain (since each subdomain has equal size) and in this case it is 30% of the size of each subdomain.

Table 6.6: List of independent and dependent variables for each machine configuration

Machine Configuration	**Independent Variables**	**Dependent Variables**
Serial Configuration	Total Grid Size Simulation Time Model Complexity Source Excitation	Speedup Efficiency Simulation Runtime Percent Error
Parallel Configuration	Total Grid Size Simulation Time Model Complexity No. of subdomain Size of overlapping region Source Excitation	Speedup Efficiency Simulation Runtime Percent Error Convergence Time

Table 6.6 shows the different independent and dependent variables that will be used in the project. The independent variables are the parameters that will be varied one at a time (while using the default configuration for the other parameters) in each simulation run while the dependent variables are the metrics and measurements parameters that will indicate about the performance of the simulator as well its accuracy.

6.3.2 Simulation Accuracy

In this part, the accuracy of the simulation with regards to each configuration as well as to other simulation results from open source tools will be discussed.

Convergence Condition in Schwarz Algorithm

The convergence condition of each subdomain in Schwarz Alternating Algorithm is one of the most important aspect of that method because this convergence condition determines if the algorithm will repeat to the starting point or finish. An important parameter of the convergence condition is ϵ_{tol} (see equation 3.1) or the error tolerance of the algorithm.

The error tolerance $\epsilon_{machine}$ used in the convergence of Schwarz Alternating method was selected based on [?] where it is also referred as machine epsilon and defined as "Difference between 1 and the least value greater than 1 that is representable" [?]. This definition of the error tolerance can give enough accuracy between the values in the serial configuration and parallel configuration such that when the algorithm achieve convergence, there will be no discrepancy between the serial configuration and parallel configuration.

In order to get the value of the machine epsilon, there are built-in libraries for C++ and Python that can be used in order to set the error tolerance without setting a constant in the code (since it might be machine dependent).For C++, the standard library (std) can be used by using the statement "**std::numeric_limits<double>::epsilon()**" [?]. In Python, the *sys library* is the library that can be used to get the machine epsilon, using the statement "**sys.float_info.epsilon**" [?].

Error Analysis

The accuracy of simulation results of the parallel configuration is compared to the serial configuration and both the serial and parallel configuration will be compared to the results of the simulation from open source FDTD tools. To measure the difference in results, percent error is used shown below:

$$Percent\,error\ =\ |\frac{X_{reference}-X_{simulation}}{X_{reference}}|\times 100\% \tag{6.32}$$

As shown in equation 6.32, the formula used is relative error that is converted into percentage. In the context of this project, the electric and magnetic field is the parameters that will be compared because these are the main products of the FDTD algorithm. The equations below show the specific equations that will be used for each configuration.

$$E_{\%_{error}} = |\frac{E_{serial/other} - E_{parallel/serial}}{E_{serial/other}}| \times 100\% \tag{6.33}$$

$$\tilde{H}_{\%_{error}} = |\frac{\tilde{H}_{serial/other} - \tilde{H}_{parallel/serial}}{\tilde{H}_{serial/other}}| \times 100\% \tag{6.34}$$

The percent error for both the electric and magnetic fields are computed depending on the configurations. In the case of serial configuration, the reference simulation results is the results from the open source FDTD tools. In parallel configuration, the reference simulation results are the simulation results from the serial configuration and the open source FDTD tools. Another aspect that can be done to check the accuracy of the results is to superimpose the plots of each simulation results to see how close the results are with each other and to determine if some parts in the computational domain have a higher error or lesser error.

6.3.3 Performance Analysis

This part discusses on how the performance of the parallel configuration is measured using the **speedup** and **efficiency** metrics.

6.3.3.1 Speedup

Most researches [?, ?, ?] in parallel computing use relative speedup in measuring the effect of increasing the number of subdomain in a simulation runtime. The equation for relative speedup is shown below:

$$S_P = \frac{T(1)}{T(P)} \tag{6.35}$$

In [?], the relative speedup is defined as the ratio of the successful runtime of an algorithm using one processor and the successful runtime of the same algorithm using P number of processors.

In the metric of speedup, since each subdomain corresponds to a node/processor, the expected result is that the speedup of the configuration will be better as the number of subdomains increases because each subdomain will have smaller area to compute when the subdomains have increased in number. Also, the serial configuration will be used as a reference value since the speedup of the serial configuration is 1. Based on equation 6.35 , as the number of subdomains approach infinity, the speedup will approach 0. This suggest that a smaller speedup value results in better performance. This metric, however is not the only measurement of performance that is needed to be considered so another parameter, efficiency is discussed next.

6.3.3.2 Efficiency

Same as in subsubsection 6.3.3.1, most researches in parallel FDTD research uses another metric for measuring the efficiency of the algorithm. This metric is called **relative parallel efficiency**.

$$E(P) = \frac{S_P}{P} \tag{6.36}$$

Relative parallel efficiency is defined in [**?**] as the ratio between the speedup (equation 6.35) and the number of processors P. This metric is useful for measuring the performance of an algorithm on how efficient it can use each processor.

The efficiency (equation 6.36) can be used to describe on how efficient the algorithm in utilizing the processors used in the algorithm. This is because as the number of processors decrease, the efficiency increases while the speedup is directly proportional to the efficiency. The ideal efficiency value is 0 or when the speedup reaches 0 and the number of processors used is 1. This metric, along with speedup can be used to describe the performance of the parallel configuration of the project.

Simulation Runtime and Convergence Time

The last two parameter that measures the total simulation runtime and convergence time in the Schwarz Alternating Method. These parameters are also important to measure in the testing part of the methodology because they can give some description on how long the simulation will run. The convergence time will always be less than the total simulation runtime because in every simulation runtime, there will be multiple convergence time because the Schwarz Alternating Method is done on every subdomain.

The simulation runtime is the total amount of time (in seconds) on how long the simulation runs starting from the pre-processing stage to the post-processing stage. Convergence time, on the other hand is the amount of time from the start of an instance of Schwarz Alteranting Method to the end when all subdomain have converged.

These parameters can help in identifying what independnet variables have significant effect of the runtime of the simulations.

6.3.4 Data Visualization

Since in the testing, there will be numerous simulation results from numerous simulation runs, data visualization is necessary to see the different trends from the data. The superimposition of different plots on each other can help in determining the patterns that emerge in the data. Tabulating the metrics can also be used to compare the values closely and make sure that the analysis can be done on the data properly. These methods will be used after the simulation results are completed in each configuration.

Chapter 7

Preliminary Findings

In this part, the initial simulation results from our current implementation of 1D FDTD is presented. The FDTD algorithm used in this part is similar to subsection 2.5. The difference is on the boundary condition used which is only the Dirichlet Boundary condition, which the boundary at the end of the computational domain acts like a perfect electric conductor and perfect magnetic conductor.

The simulation parameters used are $f_{MAX} = 2.4GHz$, Gaussian Pulse and No Source, Dirichlet Boundary condition, Perfectly Absorbing Boundary Condition (PABC), and the computational domain is composed of only air or vacuum while the simulation model used is a single layer of glass ($\mu = \frac{3}{8}$, $\epsilon = 8$). The following figures are the simulation results when: (1) No source excitation, (2) Gaussian pulse using Hard source method, (3) Gaussian pulse using Soft source method, and (4) Gaussian pulse using Soft source with PABC.

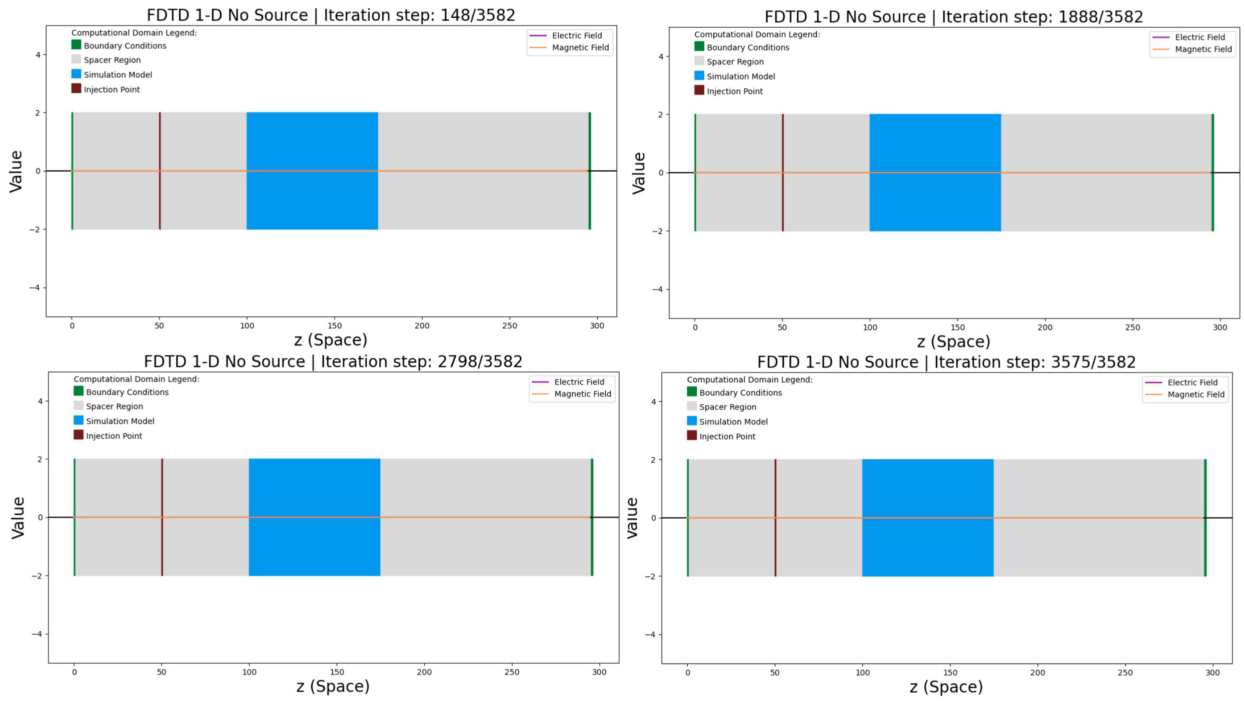


Figure 7.1: FDTD Simulation Results at different time step (No Source Excitation)

As shown in Figure 7.1, the FDTD simulation shows that there is no change in the electric and magnetic field. This means that the simulation is stable enough and does not show any errors when there is no source excitation. The results are expected since there is no outside EM source to excite the computational domain, there will be no change in the initial electric and magnetic field values.

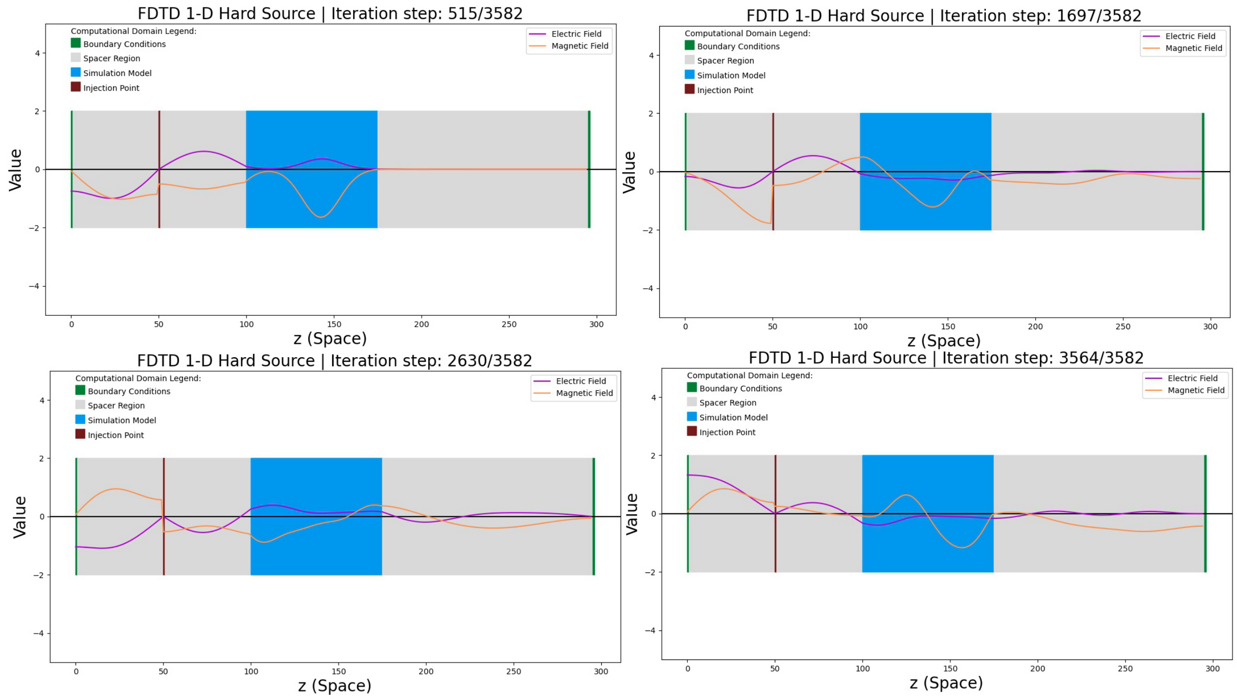


Figure 7.2: FDTD Simulation Results at different time step (Gaussian Pulse in Hard Source Method)

As shown in Figure 7.2, the FDTD simulation has injected the Gaussian Pulse in the center of the computational domain. Two pulses were produced (propagates to the left and right direction) and since the boundary condition used are Dirichlet Boundary conditions, the pulses bounces off of the boundaries and cannot propagate outside the computational domain. The injection point also acts like a wall between the two pulses so they cannot interact with each other. This is because Hard source method forces the injection by overwriting the field value at the injection point and disregards the previous value. Also, the fields are distorted (slower and change in shape) when it enters the glass material and reverts back to its original shape and speed after exiting the glass material. It is also important to note that small reflections occur as the fields enter a change in material (when it enter or exit the glass).

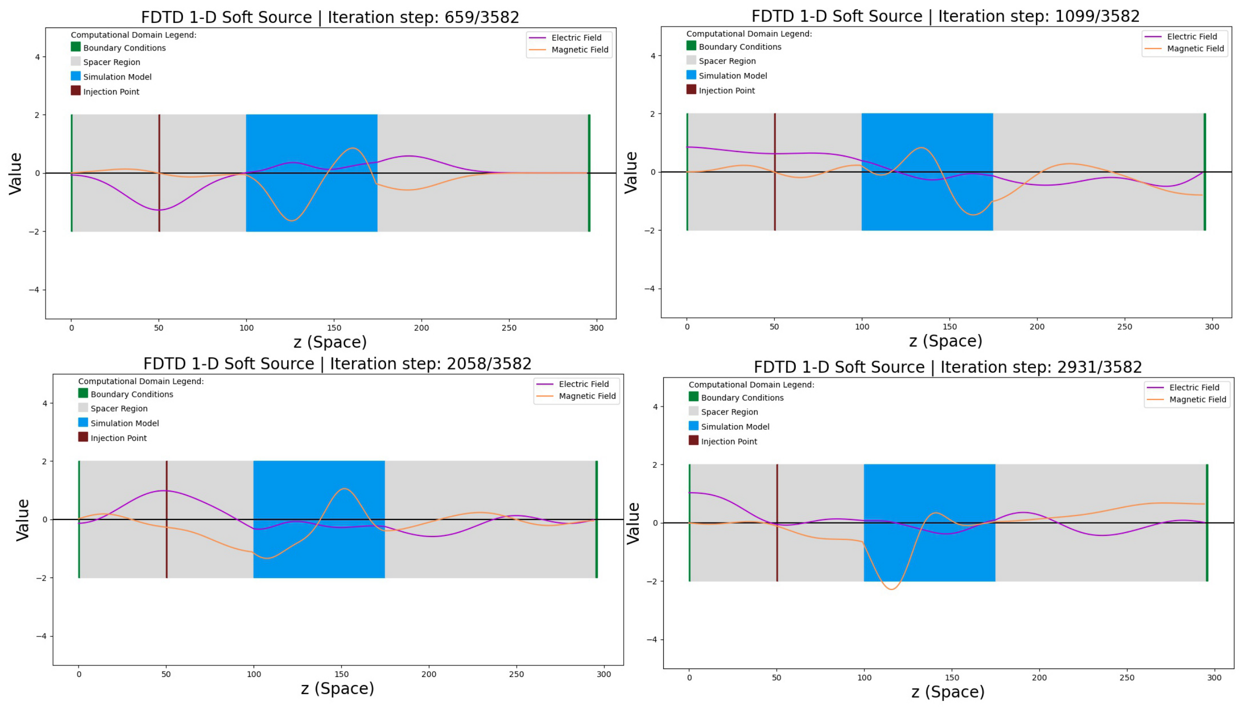


Figure 7.3: FDTD Simulation Results at different time step (Gaussian Pulse in Soft Source Method)

The next figure (Figure 7.3) shows the simulation results similar to the one in Figure 7.2 but the method of excitation is soft source method where the value of the source at the injection point is added onto the previous value of the fields so that the two pulses can interact with each other. However, since the boundary conditions are Dirichlet and the material is lossless, the pulses will not fade but bounce around the computational domain until the simulation ends.

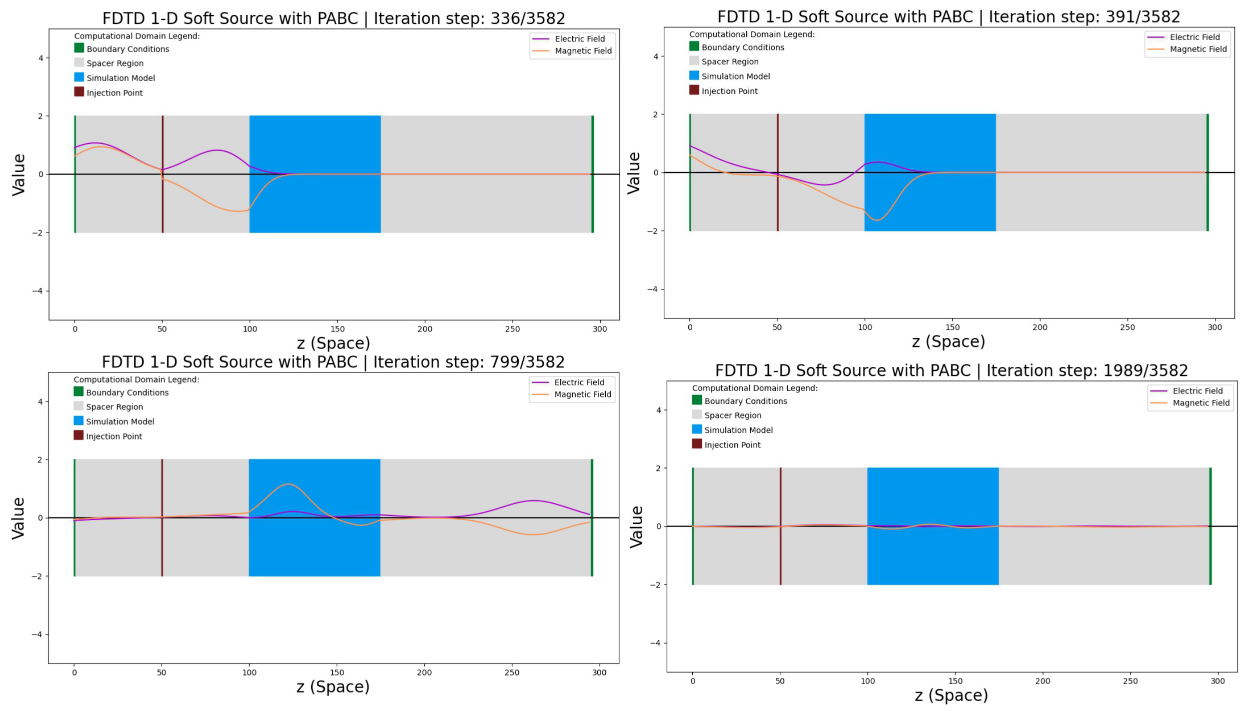


Figure 7.4: FDTD Simulation Results at different time step (Gaussian Pulse in Soft Source Method and PABC)

The last figure (Figure 7.4) shows similar results and observation as the previous figure (Fig. 7.3). The main difference is the boundary condition used which is PABC. The results show that the fields can now propagate outside of the computational domain as shown in the figure as the time step increase, there are fewer field values that change and it becomes smaller. However, it is not a perfect boundary condition as since it is also noticed that small reflections (similar to what happens when field enter or exit a new material) occur as shown in time step 799 when the reflection of the fields enter the glass material.

Chapter 8

Results and Discussion

8.1 Problems Encountered

In this section, problems that were encountered during the implementation of the project will be presented. Actions done to solve these challenges will also be included in this section.

8.1.1 Ringing in the computational domain when source is sinusoidal

When first implementing the 1-D FDTD algorithm, some erratic behavior was observed when the waves are propagating across the computational domain. This happens when the source type is sinusoidal and the boundary conditions is PABC.

The problem was caused by an error in the initialization of two of the vectors used to store the field values. There was a misuse of the `range()` function of the Xtensor library.

8.1.2 Unwanted energy propagating in the wrong direction

It has been observed that some energy was propagating to the left of the source injection point. This is unwanted behavior since the implementation of the source should be one-sided. The culprit of this problem was a mistype in the code — the places of two variables in the code were unknowingly exchanged.

From Week 1 and 2

1. Ringing in the computational domain when source is sinusoidal
2. PABC does not completely allow the pulse to propagate freely, some energy is still reflected back

From Week 3

8.2 Features

1. Boundary conditions - PABC and Dirichlet
2. Sources - sinusoidal, Gaussian pulse, ****
3. Excitation method - hard, soft, TF/SF

8.3 Code Dependencies/Libraries used

1. Xtensor library - useful for matrix oprations
2. Xtl - prerequisite of xtensor
3. HighFive - used for plotting of results(??)
4. plotly
5. matplotlib

Chapter 9

Project Schedule and Deliverables

9.1 Gantt Chart

The Gantt Chart for this project is shown in Figure 6.1. It is assumed that the project will run for twelve weeks. The chart is color-coded according to which proponent is responsible for a certain task. The blue-colored bars are assigned to Mr. Moyco, while the purple-colored bars are assigned to Mr. Alaan. The green-colored bars are assigned to both members of the team.

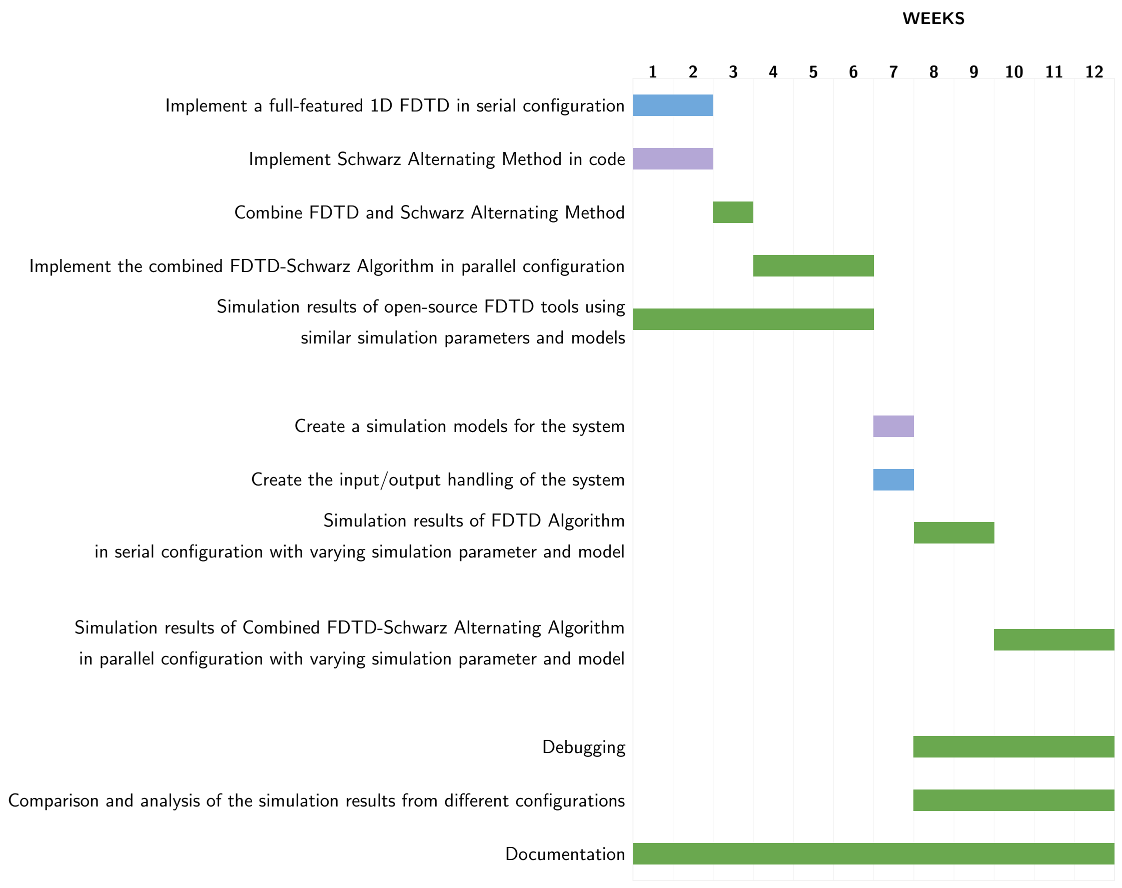


Figure 9.1: Gantt chart of the Project Proposal

9.2 Halfway-point Deliverables

The following list shows the expected outputs of the group at the half-way point of the project. For this project, the proponents deemed the half-way point to be after the third week of the entire project duration.

1. Serial FDTD with features
2. Schwarz Alternating Method Code
3. Combination of the two algorithms implemented in series

Item 1 is assigned to Mr. Moyco, while item 2 is assigned to Mr. Alaan. Item 3 is assigned to both members of the team.

9.3 Final Deliverables

The following list shows the expected outputs of the group at the end of the project.

1. Implementation of the codes in parallel configuration
2. Comparison of the simulation results of three configurations
3. Documentation

All of the tasks listed above will be worked on by both members of the team.

Bibliography

[1] UP Diliman University Student Council. Position paper on ending the semester and mass promotion. https://www.facebook.com/USCUPDiliman/photos/pcb.2966931010033920/2966930166700671, Apr. 2020.

[2] Raymond Rumpf. Formulation of one-dimensional fdtd. https://empossible.net/wp-content/uploads/2020/01/Lecture-Formulation-of-1D-FDTD.pdf, 1 2020.

[3] Irina Tezaur, Alejandro Mota, and Coleman Alleman. The schwarz alternating method for concurrent multiscale coupling in solid mechanics. https://old-www.sandia.gov/~ikalash/_assets/documents/tezaur_coupled_2017_schwarz.pdf, 2017.

[4] Remi Busseuil, Gabriel Almeida, Luciano Ost, Sameer Varyani, Gilles Sassatelli, and Michel Robert. Adaptation strategies in multiprocessors system on chip. In *IFIP Advances in Information and Communication Technology*, pages 233–257, Sep. 2010.

[5] Dennis Mapa. Functional literacy rate of filipinos by exposure to different forms of mass media ranges from 92.6 percent to 97.1 percent in 2019. Technical report, Philippine Statistics Authority, 2020.

[6] Janvic Mateo. Tech woes among top concerns for distance learning. https://www.philstar.com/headlines/2020/08/02/2032336/tech-woes-among-top-concerns-distance-learning, Aug. 2020.

[7] CNN Philippines. Sws: Over 5 out of 10 enrolled filipino students use devices for distance learning. https://www.cnnphilippines.com/news/2021/3/2/filipino-students-devices-distance-learning.html, Mar. 2021.

[8] Sofia Abrogar. Over 5,000 up students may be unable to continue under remote learning. https://www.tinigngplaridel.net/news/2020/over-5000-up-students-may-be-unable-to-continue-under-remote-learning/, 2020.

[9] Queensland Government. Benefits of cloud computing. `https://www.business.qld.gov.au/running-business/it/cloud-computing/benefits`, 2017.

[10] Amazon Web Services. What is cloud computing? `https://aws.amazon.com/what-is-cloud-computing/`.

[11] Michael Armbrust, Armando Fox, Rean Griffith, Anthony Joseph, Randy Katz, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, and Andrew Konwinski. Above the clouds: A berkeley view of cloudcomputing. Technical report, University of California Berkeley, 2009.

[12] Ingo Wolff. Recent improvements of finite difference time domain simulation techniques. In *2016 IEEE MTT-S International Microwave Symposium (IMS)*. IEEE, May 2016.

[13] Raymond Rumpf. Introduction to cem. `https://empossible.net/wp-content/uploads/2019/08/Lecture-0b-Introduction-to-CEM.pdf`, 2019.

[14] Gizem Toroglu and Levent Sevgi. Finite-difference time-domain (fdtd) matlab codes for first- and second-order em differential equations [testing ourselves]. *IEEE Antennas and Propagation Magazine*, 56, Apr. 2014.

[15] M. K. Haldar. Introducing the finite element method in electromagnetics to undergraduates using matlab. *The International Journal of Electrical Engineering & Education*, 43, Jul. 2006.

[16] Ercument Arvas and Levent Sevgi. A tutorial on the method of moments [testing ourselves]. *IEEE Antennas and Propagation Magazine*, 54, Jun. 2012.

[17] Omar Ramadan. Efficient parallel fdtd algorithm for modeling infinite graphene sheet simulations. In *2017 8th International Conference on Information and Communication Systems (ICICS)*. IEEE, Apr. 2017.

[18] George Karniadakis and Robert Kirby II. *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, 2003.

[19] T. Namiki and K. Ito. A new fdtd algorithm free from the cfl condition restraint for a 2d-te wave. In *IEEE Antennas and Propagation Society International Symposium. 1999 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.99CH37010)*. IEEE, 1999.

[20] Raymond Rumpf. Lecture 4 (fdtd) – electromagnetics and fdtd. `https://www.youtube.com/watch?v=TbzFef5gUms&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=5`, 2014.

[21] Kane Yee. Numerical solution of the initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Microwave Theory and Techniques*, 44:61–69, 1998.

[22] Jennifer E. Houle and Dennis M. Sullivan. *Electromagnetic Simulation Using the FDTD Method with Python*. Wiley, Dec. 2019.

[23] Raymond Rumpf. Lecture 8 (fdtd) – review and walkthrough of 1d fdtd. `https://www.youtube.com/watch?v=jlPXKXKSXfs&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=9`, 2014.

[24] Louis Poirel. Algebraic domain decomposition methods for hybrid (iterative/direct) solvers. Master's thesis, Université de Bordeaux, Nov. 2018.

[25] Barry Smith, Peter Bjorstad, and William Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1st edition, 1996.

[26] A Edelman. 6.338j/18.337j applied parallel computing. `http://courses.csail.mit.edu/18.337/2009/`, 2009.

[27] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An Introduction to Domain Decomposition Methods:algorithms, theory and parallel implementation*. HAL, 6th edition, 2021.

[28] E Maroudas, Nikos Vilanakis, C Antonopoulos, E Mathioudakis, Yiannis Saridakis, and M Vavalis. Schwarz splitting for the steady state problem of saltwater intrusion in coastal aquifers. *International Journal of Mathematical Models and Methods in Applied Sciences*, 9:733–739, Jun. 2015.

[29] R. Courant, K. Friedrichs, and H. Lewy. Uber die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100, 12 1928.

[30] Allen Taflove and S. Hagness. *Computational electrodynamics: the finite-difference time-domain method. 3rd ed*, volume 67-106. Artech House Publishers, June 2005.

[31] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14(3):302–307, 1966.

[32] Allen Taflove. Application of the finite-difference time-domain method to sinusoidal steady-state electromagnetic-penetration problems. *IEEE Transactions on Electromagnetic Compatibility*, EMC-22, 8 1980.

[33] Der-Han Huang, Andreas C. Cangellaris, and Xu Chen. Stochastic-galerkin finite-difference time-domain for waves in random layered media. In *2020 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, pages 1–4, 2020.

[34] Antonio P. Thomson, Atef Z. Elsherbeni, and Mohammed Hadi. A practical fourth order finite-difference time-domain algorithm for the solution of maxwell's equations. In *2020 International Applied Computational Electromagnetics Society Symposium (ACES)*, pages 1–2, 2020.

[35] Abdullah M. Algarni, Atef Z. Elsherbeni, and Mohammed Hadi. Implementation of passive and active circuit elements in cylindrical finite-difference time-domain formulation. In *2020 International Applied Computational Electromagnetics Society Symposium (ACES)*, pages 1–2, 2020.

[36] Daryl L. Logan. *A First Course in the Finite Element Method*. Cengage Learning, 5 edition, 2012.

[37] Martin Costabel. Principles of boundary element methods. *Computer Physics Reports*, pages 243–274, 1987.

[38] J.T. Katsikadelis. *Boundary Elements: Theory and Applications*. Elsevier, 1st edition, 2002.

[39] Mohaira Ahmad and Lixia Yang. Modeling of 1d graphene based on ltjec-fdtd method. In *2016 11th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*. IEEE, Oct. 2016.

[40] Jiang Fan Liu, Xiao Li Xi, Guo Bin Wan, and Li Li Wang. Simulation of electromagnetic wave propagation through plasma sheath using the moving-window finite-difference time-domain method. *IEEE Transactions on Plasma Science*, 39:852–855, Mar. 2011.

[41] Hyungwoo Kim, Jangwon Lee, Jongsuh Lee, Jaeyub Hyun, and Semyung Wang. Topology optimization of a magnetic resonator using finite-difference time-domain method for wireless energy transfer. In *IEEE Transactions on Magnetics*, volume 52. Institute of Electrical and Electronics Engineers Inc., Mar. 2016.

[42] Shuichi Aono, Masaki Unno, and Hideki Asai. A novel fdtd algorithm based on alternating-direction explicit method with pml absorbing boundary condition. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Jan. 2010.

[43] S.G. Garcia, R.G. Rubio, A.R. Bretones, and R.G. Martin. On the dispersion relation of adi-fdtd. *IEEE Microwave and Wireless Components Letters*, 16, 6 2006.

[44] Atef Elsherbeni and Veysel Demir. *The Finite-Difference Time-Domain Method for Electromagnetics with MATLAB Simulations*. SciTech Publishing, 2nd edition, 2015.

[45] Jingjun Song, Lijuan Shi, Shitian Zhang, Qinliang Li, and Lixia Yang. Research on non-uniform nested grid fdtd method based on ionospheric zakharov model. In *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*, pages 1–4, 2018.

[46] Naixing Feng, Yuxian Zhang, Jinfeng Zhu, Qingsheng Zeng, and Guo Ping Wang. High-accurate non-uniform grids for system-combined adi-fdtd method in near-field scattering with proper cfl factor. *IEEE Access*, 9:18550–18559, 2021.

[47] L. Catarinucci, P. Palazzari, and L. Tarricone. A parallel variable-mesh fdtd algorithm for the solution of large electromagnetic problems. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8 pp.–, 2005.

[48] Joao Costa, Kevin Roy, Fumie Costen, and Anthony Brown. Toward large scale frequency dependent fdtd for uwb systems. In *2007 IEEE Antennas and Propagation Society International Symposium*. IEEE, Jun. 2007.

[49] Kamal Abboud, Toufic Abboud, Francois Bereux, and Gilles Peres. Thin conducting sheet (tcs) in 1d fdtd : Theory and implementation. In *2008 International Symposium on Electromagnetic Compatibility - EMC Europe*. IEEE, Sep. 2008.

[50] Toshihiro Hanawa and Soichiro Ikuno. Large-scale simulation for optical propagation in 3-d photonic crystal using the fdtd method with parallel processing. *IEEE Transactions on Magnetics*, 43, Apr. 2007.

[51] Jie Luo, Zhihong Ye, and Cheng Liao. A mpi-based parallel fdtd-tl method for the emi analysis of transmission lines in cavity excited by ambient wave. *IEEE Transactions on Electromagnetic Compatibility*, 62:212–217, Feb. 2020.

[52] Wenhua Yu, Raj Mittra, Xiaoling Yang, and Yongjun Liu. Performance analysis of parallel fdtd algorithm on different hardware platforms. In *2009 IEEE Antennas and Propagation Society International Symposium*. IEEE, Jun. 2009.

[53] Tomasz Ciamulski and Maciej Sypniewski. Different implementations of parallel processing for fdtd simulator. In *MIKON 2008 - 17th International Conference on Microwaves, Radar and Wireless Communications*, pages 1–3, 2008.

[54] Tomasz Ciamulski, Mats Hjelm, and Maciej Sypniewski. Parallel fdtd calculation efficiency on computers with shared memory architecture. In *2007 Workshop on Computational Electromagnetics in Time-Domain*. IEEE, Oct. 2007.

[55] Kuisong Zheng, Huan Luo, Zongmin Mu, Jianying Li, and Gao Wei. Parallel tss-fdtd method for analyzing underwater low-frequency electromagnetic propagation. *IEEE Antennas and Wireless Propagation Letters*, 15, Nov. 2016.

[56] Jundong Tan, Zihao Li, Qi Guo, and Yunliang Long. An efficient parallel implicit solver for lod-fdtd algorithm in cloud computing environment. *IEEE Antennas and Wireless Propagation Letters*, 17, Jul. 2018.

[57] Xiaoyang Yan, Weiwen Zhang, Huifang Deng, and Shehui Bu. The parallelization of three-dimensional electro-magnetic particle model using both mpi and openmp. In *2010 International Conference on Computational and Information Sciences*. IEEE, Dec. 2010.

[58] Haiming Lin, Xiaohu Liu, Kangyu Jia, and Wei Fu. A parallel domain decomposition fdtd algorithm based on cloud computing. In *2013 International Conference on Information Science and Cloud Computing Companion*. IEEE, Dec. 2013.

[59] Bin Yuan, Yao Zhang, and Xiu Wang. A suitable computing architecture and its algorithm for parallel fdtd. In *Asia-Pacific Microwave Conference 2011*, pages 66–69, 2011.

[60] Xu Zhang, Xiaoming Liu, Shaohua Liu, Junsheng Yu, Ran Ding, Bingran Liu, and Rui Xiao. A more efficient mechanism of fdtd method based on message passing interface. In *2008 International Conference on Microwave and Millimeter Wave Technology*. IEEE, Apr. 2008.

[61] Zengrui Li, Limei Luo, and Junhong Wang. A study of parallel fdtd method on high-performance computer clusters. In *2009 International Joint Conference on Computational Sciences and Optimization*. IEEE, Apr. 2009.

[62] T. Hanawa, M. Kurosawa, and S. Ikuno. Investigation on 3-d implicit fdtd method for parallel processing. *IEEE Transactions on Magnetics*, 41, May 2005.

[63] Wenhua Yu, Yongjun Liu, Tao Su, Neng-Tien Hunag, and R. Mittra. A robust parallel conformal finite-difference time-domain processing package using the mpi library. *IEEE Antennas and Propagation Magazine*, 47, Jun. 2005.

[64] J Dong, S L Chai, and J J Mao. An automatic load-balanced parallel multilevel fast multipole method for large scale electromagnetic scattering problem. In *2005 Asia-Pacific Microwave Conference Proceedings*, volume 3, pages 4 pp.–, 2005.

[65] Alexey B. Gnilenko and Valeriy I. Magro. Method of partial overlapping regions for solving electromagnetic problems. In *2017 XXIInd International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED)*, pages 36–41, 2017.

[66] Sanjay K. Khattri, Gunnar E. Fladmark, and Helge K. Dahle. *Control Volume Finite Difference On Adaptive Meshes*. Springer Berlin Heidelberg, 2007.

[67] Ke Li, Kejun Tang, Tianfan Wu, and Qifeng Liao. D3m: A deep domain decomposition method for partial differential equations. *IEEE Access*, 8, 2020.

[68] P. Spiteri, R. Guivarch, D. El Baz, and Chau Ming. Parallelization of subdomain methods with overlapping for linear and nonlinear convection-diffusion problems. In *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2003. Proceedings.*, pages 341–348, 2003.

[69] M. A. Gnatyuk and V. M. Morozov. On the schwarz alternating method for solving electromagnetic problems. In *2015 XXth IEEE International Seminar/Workshop on Direct and Inverse Problems of Electromagnetic and Acoustic Wave Theory (DIPED)*, pages 132–135, 2015.

[70] P M Mell and T Grance. The nist definition of cloud computing. Technical report, National Institute of Standards and Technology, 2011.

[71] IBM. Iaas (infrastructure-as-a-service). `https://www.ibm.com/cloud/learn/iaas#toc-iaas-platf-uSme6HXr`, Jul. 2019.

[72] Amazon. Amazon ec2. `https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc`, 2021.

[73] Google. Compute engine. https://cloud.google.com/compute.

[74] Microsoft. Windows virtual machines in azure. https://docs.microsoft.com/en-us/azure/virtual-machines/windows/overview, Nov. 2019.

[75] Satish Srirama, Oleg Batrashev, and Eero Vainikko. Scicloud: Scientific computing on the cloud. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 579–580, 2010.

[76] Dipanjan Gope, Vikram Jandhyala, Xiren Wang, Don Macmillen, Raul Camposano, Swagato Chakraborty, James Pingenot, and Devan Williams. Towards system-level electromagnetic field simulation on computing clouds. In *2011 IEEE 20th Conference on Electrical Performance of Electronic Packaging and Systems*. IEEE, Oct. 2011.

[77] Tim McDonald, Robert Fisher, Greg Rigden, and Rod Perala. Parallel fdtd electromagnetic effects simulation using on-demand cloud hpc resources. In *2013 IEEE International Symposium on Electromagnetic Compatibility*. IEEE, Aug. 2013.

[78] S.D. Gedney. A comparison of the performance of finite difference time-domain, finite element time-domain, and discrete surface integral equation methods on high performance parallel computers. In *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting*. IEEE, 1994.

[79] Raymond Rumpf. Lecture 9 (fdtd) – examples of 1d fdtd. https://www.youtube.com/watch?v=hbjIt_ZGQa4&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=10, 2014.

[80] Regina C. Allil, Fábio V. de Nazaré, and Marcelo Martins M. Werneck. *Fiber Bragg Gratings: Theory, Fabrication, and Applications*, chapter Introduction. SPIE, Sep. 2017.

[81] Naser Tamimi. How fast is c++ compared to python? https://towardsdatascience.com/how-fast-is-c-compared-to-python-978f18f474c7, Dec. 2020.

[82] Raymond Rumpf. Lecture 6 (fdtd) – implementation of 1d fdtd. https://www.youtube.com/watch?v=hNN7EtZsJuU&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=7, 2014.

[83] Umran Inan and Robert Marshall. *Numerical Electromagnetics The FDTD Method*. Cambridge University Press, 2011.

[84] Raymond Rumpf. Lecture 7 (fdtd) – learning from 1d fdtd. https://www.youtube.com/watch?v=H5Qq3JGwk1I&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=8, 2014.

[85] Martin H. Weik. *Gaussian pulse*, pages 676–676. Springer US, Boston, MA, 2001.

[86] Huaguang Bao and Rushan Chen. An efficient domain decomposition parallel scheme for leapfrog adi-fdtd method. *IEEE Transactions on Antennas and Propagation*, 65, Mar. 2017.

[87] Raymond Rumpf. Lecture 10 (fdtd) – enhancing 1d fdtd. `https://www.youtube.com/watch?v=ZrgGIkBvbvM&list=PLLYQF5WvJdJWoU9uEeWJ6-MRzDSziNnGt&index=11`, 2014.

[88] Zhaoyang Cai, Bin Chen, Yun Yi, Run Xiong, and Yunfei Mao. A novel tf/sf boundary for cylindrical fdtd method with ground. In *2012 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, volume 2, pages 1–4, 2012.

[89] Alejandro Mota, Irina Tezaur, and Coleman Alleman. The schwarz alternating method in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 319, Jun. 2017.

[90] A. Malagón-Romero and A. Luque. A domain-decomposition method to implement electrostatic free boundary conditions in the radial direction for electric discharges. *Computer Physics Communications*, 225, Apr. 2018.

[91] Seongjai Kim. *Numerical Methods for Partial Differential Equations*. Mississippi State University, Sep. 2017.

[92] Hendrik Borchardt. Iterative domain decomposition methods foreigenvalue problems. Master's thesis, RWTH Aachen University, Apr. 2020.

[93] Rowel Atienza. Foundations of machine learning. `https://github.com/roatienza/ml`, 2020.

[94] The FFmpeg developers. Stream selection. `https://ffmpeg.org/ffmpeg.html#Stream-selection`, 2021.

[95] Python Software Foundation. sys system-specific parameters and functions. `https://docs.python.org/3/library/sys.html`.

[96] The C++ Resources Network. std::numeric_limits. `https://www.cplusplus.com/reference/limits/numeric_limits/`.

[97] K.H. Hoffmann and Jun Zou. Parallel efficiency of domain decomposition methods. *Parallel Computing*, 19, Dec. 1993.