UNCONDITIONALLY STABLE FINITE DIFFERENCE TIME DOMAIN METHODS FOR FREQUENCY DEPENDENT MEDIA

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Engineering and Physical Sciences

2010

By
Hasan Khaled Rouf
School of Electrical and Electronic Engineering

Contents

List of Tables

List of Figures

Abstract

The efficiency of the conventional, explicit finite difference time domain (FDTD) method is constrained by the upper limit on the temporal discretization, imposed by the Courant–Friedrich–Lewy (CFL) stability condition. Therefore, there is a growing interest in overcoming this limitation by employing unconditionally stable FDTD methods for which time-step and space-step can be independently chosen. Unconditionally stable Crank Nicolson method has not been widely used in time domain electromagnetics despite its high accuracy and low anisotropy. There has been no work on the Crank Nicolson FDTD (CN–FDTD) method for frequency dependent medium.

In this thesis a new three-dimensional frequency dependent CN–FDTD (FD–CN–FDTD) method is proposed. Frequency dependency of single–pole Debye materials is incorporated into the CN–FDTD method by means of an auxiliary differential formulation. In order to provide a convenient and straightforward algorithm, Mur's first-order absorbing boundary conditions are used in the FD–CN–FDTD method. Numerical tests validate and confirm that the FD–CN–FDTD method is unconditionally stable beyond the CFL limit.

The proposed method yields a sparse system of linear equations which can be solved by direct or iterative methods, but numerical experiments demonstrate that for large problems of practical importance iterative solvers are to be used. The FD–CN–FDTD sparse matrix is diagonally dominant when the time-step is near the CFL limit but the diagonal dominance of the matrix deteriorates with the increase of the time-step, making the solution time longer. Selection of the matrix solver to handle the FD–CN–FDTD sparse system is crucial to fully harness the advantages of using larger time-step, because the computational costs associated with the solver must be kept as low as possible. Two best–known iterative solvers, Bi-Conjugate Gradient Stabilised (BiCGStab) and Generalised Minimal Residual (GMRES), are extensively studied in terms of the number of iteration requirements for convergence, CPU time and memory requirements. BiCGStab outperforms GMRES in every aspect. Many of these findings do not match with the existing literature on frequency–independent CN–FDTD method and the possible reasons for this are pointed out.

The proposed method is coded in Fortran and major implementation techniques of the serial code as well as its parallel implementation in Open Multi-Processing (OpenMP) are presented. As an application, a simulation model of the human body is developed in the FD–CN–FDTD method and numerical simulation of the electromagnetic wave propagation inside the human head is shown.

Finally, this thesis presents a new method modifying the frequency dependent alternating direction implicit FDTD (FD–ADI–FDTD) method. Although the ADI–FDTD method provides a computationally affordable approximation of the CN–FDTD method, it exhibits a loss of accuracy with respect to the CN-FDTD method which may become severe for some practical applications. The modified FD–ADI–FDTD method can improve the accuracy of the normal FD–ADI–FDTD method without significantly increasing the computational costs.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Fumie Costen for her advice, guidance, support and encouragement throughout this research. Her indefatigable enthusiasm and counsel, in particular, emphasis on simplicity and elegance in research, insightful and thought-provoking feedback have been invaluable to me. Thank you, Fumie!

I am grateful to Dr. Michael Bane of Research Computing Services for providing technical advices and supports on many occasions of this research.

Thanks to the National Grid Service (NGS), Research Computing Services (RCS) at The University of Manchester and Kyushu University Computing Systems for Research, Japan for their computational resources that were used to run the numerical simulations.

I would like to thank The University of Manchester for granting me the Overseas Research Students (ORS) Award for this study.

Above all, the endless supports that I have got from my loving parents and siblings are beyond recognition.

Chapter 1

Introduction

1.1 Background

To solve the physical field problems several techniques are used, which can be classified into experimental, analytical and numerical. Experiments are expensive, time consuming, sometimes hazardous and usually do not allow much flexibility in parameter variation [3]. For many practical and complex problems, analytical solutions do not exist or are not obtainable. Numerical methods do not suffer from these limitations, yet can give approximate solutions of sufficient accuracy. With the availability of the digital computers, since 1960s studies on the numerical methods for electromagnetic field problems started.

A number of numerical techniques are commonly used in the study of electromagnetic problems, namely, method of moments (MoM) [4, 5], transmission line matrix (TLM) method [6], finite element method (FEM) [7, 8], finite difference time domain (FDTD) method [9]. Each method has advantages in some application areas, but there is no method that is the best in all the areas of applications. FEM, TLM and FDTD methods are based on the discretization of Maxwell's equations over the entire computational domain. On the other hand, integral methods, such as, MoM are based on the discretization of certain integral equations involving the Green's function. In the integral methods there are fewer number of unknowns but the matrix to be solved is a full matrix, while in the differential methods the matrix is sparse. FEM and MoM methods solve the Maxwell's equations in frequency domain while FDTD and TLM methods work

in time domain. Frequency domain methods give the solutions for a specific frequency and therefore repeated simulation runs are required to obtain the system response over a range of frequencies. Because of this, for wide-band applications a time domain method like TLM or FDTD should be used. In many ways TLM method is similar to FDTD method and shares many of the advantages and disadvantages of FDTD method. But TLM method requires more storage space for computation than FDTD method.

This thesis is on the FDTD method. FDTD method was originally developed by K.S. Yee in 1966 [9]. Initially it had drawn little interest because of the constraints of the computational resources [10]. However, the availability of powerful computing resources later made it a very popular numerical technique for computational electrodynamics. Yee's FDTD method discretizes Maxwell's curl equations to solve for both electric and magnetic fields. It uses central-difference equations for both space and time derivatives. The FDTD algorithm progresses in a leap-frog manner which means that, first electric field is solved for a certain instant, then magnetic field is solved for the next instant and this progression repeats. The method is explicit because the fields at a certain instant are calculated using the fields at the previous instant.

FDTD methods have several advantages. They are said to be the most straightforward, robust and widely applicable electromagnetic modelling techniques. They are easy to understand and easy to implement in computer programmes. FDTD methods operate in the time domain. So a single simulation run can get the solutions for a wide frequency range. By specifying the material parameters at different points in the computational domain the FDTD method can easily model various materials. On the other hand, FDTD is not good at modelling the complex geometries with a high precision. The method becomes computationally intensive and requires large amount of memory when very fine spatial discretization needs to be used for accurate modelling.

1.2 Research Context

The standard explicit FDTD method has some major limitations. Media parameters in standard FDTD method are specified as frequency-independent constants

while, in reality, they depend on frequency. Media parameters can be considered constant only within a narrow frequency band, but if a broadband pulse propagates through such a medium, the frequency dependency of the medium has to be taken into account [11]. The merit that a single FDTD simulation run can cover a wide range of frequencies can fully be attained when standard FDTD method is modified to accommodate frequency dependent materials. This is particularly important as many current and emerging technological applications use wide frequency bands and for their accurate studies incorporation of frequency dependent materials in FDTD method is essential.

One of the main drawbacks of the conventional FDTD method is the reduced computational efficiency resulting from the upper limit on the time-step that needs to satisfy the Courant–Friedrich–Lewy (CFL) stability condition [12]. This condition imposes an upper bound on the time-step depending on the minimum spatial step. Thus, when very small spatial step relative to the wavelength of interest is employed to accurately model the fine geometrical details of a given application, an unnecessarily small time-step is enforced, with an increase of the total CPU time. Therefore, there is a growing interest in overcoming this limitation by employing unconditionally stable implicit FDTD methods, for which time-step and space-step can be independently chosen [13][14]. This trend will continue because high accuracy in modelling is increasingly in demand.

An alternative to the explicit FDTD method is provided by the Crank Nicolson FDTD (CN–FDTD) method [15] which presents unconditional stability beyond the CFL limit. Both methods share in common the discretization of the time and space derivatives by centred differences, with the only difference being that the fields affected by the curl operator are averaged in time by the CN–FDTD method, whereas in explicit FDTD method they are not. The resulting scheme is a fully implicit marching–on–in–time algorithm with the same potential of the classical FDTD method. However, despite its accuracy and low anisotropy [16] it has not been widely used in time domain electromagnetics as it involves large sparse matrix computations. Instead, there have been many works attempting to simplify or approximate its implementation, such as, alternating direction implicit (ADI–FDTD) method [17], CN Douglas Gunn method [18], CN cycle sweep

method [19], CN approximate factorization splitting method [20]. Such approximations suffer up to some extent of numerical errors, which may become severe for some practical applications [21]. All these works on simplification or approximation of CN–FDTD method have been limited to frequency-independent materials. Quite recently [22] investigated the original CN–FDTD method for frequency-independent materials. To the best knowledge of the author, there has been no work on the CN–FDTD method for frequency dependent media.

1.3 Contributions and Outline of the Thesis

The purpose of this research project is to develop novel unconditionally stable FDTD algorithms for the frequency dependent media. A three-dimensional frequency dependent CN–FDTD (FD–CN–FDTD) method is proposed in this thesis. This thesis also presents an accuracy improved frequency dependent ADI–FDTD (FD–ADI–FDTD) method.

Chapter 2 provides the background knowledge of the FDTD method and literature review covering the state-of-the-art in the developments and studies of the FDTD methods.

The proposed three-dimensional FD-CN-FDTD method is presented in Chapter 3. The formulation of the FD-CN-FDTD method and the inclusion of Mur's absorbing boundary condition are described. The proposed method is unconditionally stable beyond the CFL limit and has higher accuracy than other unconditionally stable methods, such as, ADI–FDTD method.

Numerical validation of the proposed method is shown in Chapter 4. By carrying out several numerical tests with the FD–CN–FDTD method, this chapter shows the average error of the method, effects of the CFL number and spatial resolution, effects of the conductivity and the transmission and reflection coefficients errors.

The FD-CN-FDTD method requires solution of a sparse system of linear equations. An efficient sparse matrix solver is essential to fully harness the advantages

of using larger time–step beyond CFL limit. Chapter 5 deals with the issues related to the solution of the FD–CN–FDTD method. The effects of time–step beyond CFL limit on the characteristics of the FD-CN-FDTD sparse matrix and effectiveness of different solvers, of direct and iterative types, to solve the sparse system are extensively studied in this chapter.

An application of the FD–CN–FDTD method is described in Chapter 6. A simulation model of the human body is developed in the FD–CN–FDTD method with all the fine structures and frequency dependent dielectric properties of the human tissues. Numerical simulation of electromagnetic wave propagation inside the human head is shown. The implications of this study for further research on bioelectromagnetics is also explained in this chapter.

In Chapter 7 a method is presented modifying the FD–ADI–FDTD method in order to improve the accuracy. The ADI–FDTD method is a computationally affordable approximation of the CN–FDTD method [23], found by adding a perturbation term to the latter. However, the ADI–FDTD method exhibits a loss of accuracy with respect to the CN-FDTD method that may become severe for some practical applications [23]. By numerical experiments in Chapter 7 it is shown that, the modified FD–ADI–FDTD method is more accurate than normal FD–ADI–FDTD method and it does not significantly increase the computational costs.

Major implementation techniques of the proposed FD–CN–FDTD and modified FD–ADI–FDTD methods are presented in Chapter 8. The FD–CN–FDTD method is implemented in serial Fortran code and its parallel implementation is performed in Open Multi-Processing (OpenMP). There are scarce precedence of implementation of the CN–FDTD method in computer programmes and no precedence of implementation of the frequency dependent CN–FDTD method which is more complicated. Therefore, the implementation techniques described in this chapter bear significance and would be useful for further research on the CN–FDTD method.

Chapter 9 concludes the thesis with a short discussion on the works presented and highlights their limitations. Some possible future research works are also suggested in this chapter.

Chapter 2

Finite Difference Time Domain Method

The FDTD method is said to be the most straightforward, robust and versatile electromagnetic modelling technique. It is one of the most widely used numerical techniques used in computational electrodynamics. The method owes its success to the power and simplicity it provides [24]. This chapter provides the background knowledge of FDTD methods and literature review covering the state-of-the-art in the development and studies of the FDTD method.

2.1 FDTD Method

2.1.1 Finite Difference Concept

The concept of finite differencing is briefly described first because it is the basis of the FDTD method [3]. The increment of a function $f(x)$ at a certain point x_0 can be written as

$$\Delta f(x_0) = f(x_0 + \Delta x) - f(x_0) \tag{2.1}$$

Then the difference quotient or slope of the function $f(x)$ with respect to x can be expressed as

$$\frac{\Delta f(x_0)}{\Delta x} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \tag{2.2}$$

As Δx approaches zero, the derivative of the function $f(x)$ with respect to x can be written as

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{\Delta f(x_0)}{\Delta x} = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \tag{2.3}$$

Thus, when Δx is very small the derivative of a function can be approximated by its difference quotient: $\frac{df}{dx} \approx \frac{\Delta f}{\Delta x}$. Now the derivate of the function $f(x)$ can be expressed in three ways – forward, backward and central difference and their expressions are:

$$f'(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} \qquad \text{forward difference} \tag{2.4}$$

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} \qquad \text{backward difference} \tag{2.5}$$

$$f'(x_0) \approx \frac{f(x_0 + \frac{\Delta x}{2}) - f(x_0 - \frac{\Delta x}{2})}{\Delta x} \qquad \text{central difference} \tag{2.6}$$

Similarly the second derivative of $f(x)$ can be expressed as

$$f''(x_0) \approx \frac{f'(x_0 + \frac{\Delta x}{2}) - f'(x_0 - \frac{\Delta x}{2})}{\Delta x}$$
$$= \frac{1}{\Delta x}\left[\frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} - \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x}\right]$$

$$\therefore\ f''(x_0) \approx \frac{f(x_0 + \Delta x) - 2f(x_0) + f(x_0 - \Delta x)}{(\Delta x)^2} \tag{2.7}$$

These approximations of derivatives in terms of the values at a discrete set of points are called finite difference approximations [3]. Taylor's expansion series give the general approach for the above finite difference approximations. During the approximations higher-order terms in the Taylor series are usually truncated considering them negligible which introduces some degree of errors. According to

Taylor's expansion,

$$f(x_0 + \Delta x) = f(x_0) + \Delta x f'(x_0) + \frac{1}{2!}(\Delta x)^2 f''(x_0) + \frac{1}{3!}(\Delta x)^3 f'''(x_0) + \dots \quad (2.8)$$

and

$$f(x_0 - \Delta x) = f(x_0) - \Delta x f'(x_0) + \frac{1}{2!}(\Delta x)^2 f''(x_0) - \frac{1}{3!}(\Delta x)^3 f'''(x_0) + \dots \quad (2.9)$$

Adding (2.8) and (2.9) and truncating higher-order terms (considering them negligible) (2.7) is obtained. Subtracting (2.9) from (2.8) and truncating higher-order terms yield (2.6). The orders of the terms that were truncated make both of (2.6) and (2.7) second-order accurate. Similarly, rearranging (2.8) and (2.9) and dropping higher-order terms give (2.4) and (2.5), respectively. These forward and backward differences are first-order accurate. As the forward difference attempts to predict the future behaviour of the function using the values of current and previous time steps it is always unstable. The central difference is conditionally stable. In general, the backward difference is unconditionally stable but it involves an implicit update procedure which requires to solve a matrix equation at each time step [3][25].

2.1.2 Yee's FDTD Method

The FDTD method, developed by Yee [9], discretizes Maxwell's curl equations in both time and spatial domains using the central difference approximations. The resulting equations are then solved numerically to get the electric and magnetic fields at each time step in an explicit leapfrog manner. First electric field is solved for a certain instant, then magnetic field is solved in the next instant and this progression repeats. The FDTD method is second-order accurate.
Maxwell's curl equations for a isotropic and linear medium are:

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \quad (2.10)$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \quad (2.11)$$

where, **E** is the electric field, **H** is the magnetic field, **D** is the electric flux density, **B** is the magnetic flux density, **J** is the conduction current density. The constitutive relationships, $\mathbf{D} = \epsilon\mathbf{E}$ and $\mathbf{B} = \mu\mathbf{H}$, are used in (2.10) and (2.11) and then for the source-free medium ($\mathbf{J} = 0$), (2.10) and (2.11) can be written as following six coupled partial differential equations:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) \tag{2.12}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \tag{2.13}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right) \tag{2.14}$$

$$\frac{\partial E_x}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right) \tag{2.15}$$

$$\frac{\partial E_y}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right) \tag{2.16}$$

$$\frac{\partial E_z}{\partial t} = \frac{1}{\epsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) \tag{2.17}$$

where μ is the permeability and ϵ is the permittivity.

Applying central finite difference approximations of Section 2.1.1 on both space and time derivatives of (2.12), (2.13), (2.14), (2.15), (2.16), (2.17) explicit Yee FDTD equations are obtained. To derive these equations components of **E** and **H** are placed about a unit cell of lattice as shown in Fig. 2.1 and evaluated at alternate half-time steps. Thus following discretized equations are obtained,

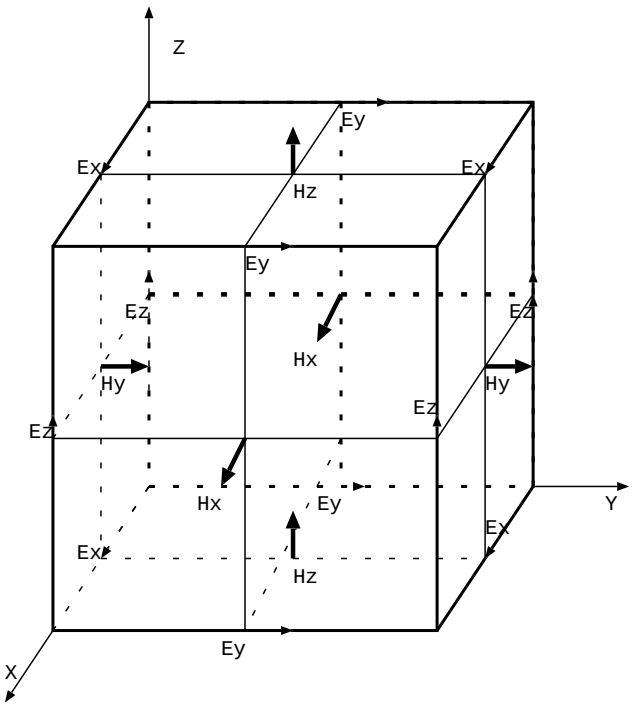


Figure 2.1: Positions of the field components in the Yee cell

where, Δt is temporal discretization and Δx, Δy and Δz are spatial discretizations in x, y and z directions, respectively:

$$H_x^{n+\frac{1}{2}}{}_{(i,j+\frac{1}{2},k+\frac{1}{2})} = H_x^{n-\frac{1}{2}}{}_{(i,j+\frac{1}{2},k+\frac{1}{2})} \tag{2.18}$$
$$+\frac{\Delta t}{\mu(i,j+\frac{1}{2},k+\frac{1}{2})\Delta z}\left[E_y^n{}_{(i,j+\frac{1}{2},k+1)} - E_y^n{}_{(i,j+\frac{1}{2},k)}\right]$$
$$+\frac{\Delta t}{\mu(i,j+\frac{1}{2},k+\frac{1}{2})\Delta y}\left[E_z^n{}_{(i,j,k+\frac{1}{2})} - E_z^n{}_{(i,j+1,k+\frac{1}{2})}\right]$$

$$H_y^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j,k+\frac{1}{2})} = H_y^{n-\frac{1}{2}}{}_{(i+\frac{1}{2},j,k+\frac{1}{2})} \tag{2.19}$$
$$+\frac{\Delta t}{\mu(i+\frac{1}{2},j,k+\frac{1}{2})\Delta x}\left[E_z^n{}_{(i+1,j,k+\frac{1}{2})} - E_z^n{}_{(i,j,k+\frac{1}{2})}\right]$$
$$+\frac{\Delta t}{\mu(i+\frac{1}{2},j,k+\frac{1}{2})\Delta z}\left[E_x^n{}_{(i+\frac{1}{2},j,k)} - E_x^n{}_{(i+\frac{1}{2},j,k+1)}\right]$$

$$H_z^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j+\frac{1}{2},k)} = H_z^{n-\frac{1}{2}}{}_{(i+\frac{1}{2},j+\frac{1}{2},k)} \tag{2.20}$$
$$+\frac{\Delta t}{\mu(i+\frac{1}{2},j+\frac{1}{2},k)\Delta y}\left[E_x^n{}_{(i+\frac{1}{2},j+1,k)} - E_x^n{}_{(i+\frac{1}{2},j,k)}\right]$$
$$+\frac{\Delta t}{\mu(i+\frac{1}{2},j+\frac{1}{2},k)\Delta x}\left[E_y^n{}_{(i,j+\frac{1}{2},k)} - E_y^n{}_{(i+1,j+\frac{1}{2},k)}\right]$$

$$E_x^{n+1}{}_{(i+\frac{1}{2},j,k)} = E_x^n{}_{(i+\frac{1}{2},j,k)} \tag{2.21}$$
$$+\frac{\Delta t}{\epsilon(i+\frac{1}{2},j,k)\Delta y}\left[H_z^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j+\frac{1}{2},k)} - H_z^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j-\frac{1}{2},k)}\right]$$
$$+\frac{\Delta t}{\epsilon(i+\frac{1}{2},j,k)\Delta z}\left[H_y^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j,k-\frac{1}{2})} - H_y^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j,k+\frac{1}{2})}\right]$$

$$E_y^{n+1}{}_{(i,j+\frac{1}{2},k)} = E_y^n{}_{(i,j+\frac{1}{2},k)} \tag{2.22}$$
$$+\frac{\Delta t}{\epsilon(i,j+\frac{1}{2},k)\Delta z}\left[H_x^{n+\frac{1}{2}}{}_{(i,j+\frac{1}{2},k+\frac{1}{2})} - H_x^{n+\frac{1}{2}}{}_{(i,j+\frac{1}{2},k-\frac{1}{2})}\right]$$
$$+\frac{\Delta t}{\epsilon(i,j+\frac{1}{2},k)\Delta x}\left[H_z^{n+\frac{1}{2}}{}_{(i-\frac{1}{2},j+\frac{1}{2},k)} - H_z^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j+\frac{1}{2},k)}\right]$$

$$E_z^{n+1}{}_{(i,j,k+\frac{1}{2})} = E_z^n{}_{(i,j,k+\frac{1}{2})} \tag{2.23}$$
$$+\frac{\Delta t}{\epsilon(i,j,k+\frac{1}{2})\Delta x}\left[H_y^{n+\frac{1}{2}}{}_{(i+\frac{1}{2},j,k+\frac{1}{2})} - H_y^{n+\frac{1}{2}}{}_{(i-\frac{1}{2},j,k+\frac{1}{2})}\right]$$
$$+\frac{\Delta t}{\epsilon(i,j,k+\frac{1}{2})\Delta y}\left[H_x^{n+\frac{1}{2}}{}_{(i,j-\frac{1}{2},k+\frac{1}{2})} - H_x^{n+\frac{1}{2}}{}_{(i,j+\frac{1}{2},k+\frac{1}{2})}\right]$$

Field components in the FDTD equations of (2.18), (2.19), (2.20), (2.21), (2.22), (2.23) lie in non-integer coordinates. To implement them in the computational systems they are transformed into integer coordinates which finally gives a new set of equations:

$$H_x^{n+1}{}_{(i,j,k)} = H_x^n{}_{(i,j,k)} + \frac{\Delta t}{\mu(i,j,k)\Delta z}\left[E_y^n{}_{(i,j,k)} - E_y^n{}_{(i,j,k-1)}\right] - \frac{\Delta t}{\mu(i,j,k)\Delta y}\left[E_z^n{}_{(i,j,k)} - E_z^n{}_{(i,j-1,k)}\right] \tag{2.24}$$

$$H_y^{n+1}{}_{(i,j,k)} = H_y^n{}_{(i,j,k)} + \frac{\Delta t}{\mu(i,j,k)\Delta x}\left[E_z^n{}_{(i,j,k)} - E_z^n{}_{(i-1,j,k)}\right] - \frac{\Delta t}{\mu(i,j,k)\Delta z}\left[E_x^n{}_{(i,j,k)} - E_x^n{}_{(i,j,k-1)}\right] \tag{2.25}$$

$$H_z^{n+1}{}_{(i,j,k)} = H_z^n{}_{(i,j,k)} + \frac{\Delta t}{\mu(i,j,k)\Delta y}\left[E_x^n{}_{(i,j,k)} - E_x^n{}_{(i,j-1,k)}\right] - \frac{\Delta t}{\mu(i,j,k)\Delta x}\left[E_y^n{}_{(i,j,k)} - E_y^n{}_{(i-1,j,k)}\right] \tag{2.26}$$

$$E_x^{n+1}{}_{(i,j,k)} = E_x^n{}_{(i,j,k)} + \frac{\Delta t}{\epsilon(i,j,k)\Delta y}\left[H_z^n{}_{(i,j+1,k)} - H_z^n{}_{(i,j,k)}\right] - \frac{\Delta t}{\epsilon(i,j,k)\Delta z}\left[H_y^n{}_{(i,j,k+1)} - H_y^n{}_{(i,j,k)}\right] \tag{2.27}$$

$$E_y^{n+1}{}_{(i,j,k)} = E_y^n{}_{(i,j,k)} + \frac{\Delta t}{\epsilon(i,j,k)\Delta z}\left[H_x^n{}_{(i,j,k+1)} - H_x^n{}_{(i,j,k)}\right] - \frac{\Delta t}{\epsilon(i,j,k)\Delta x}\left[H_z^n{}_{(i+1,j,k)} - H_z^n{}_{(i,j,k)}\right] \tag{2.28}$$

$$E_z^{n+1}{}_{(i,j,k)} = E_z^n{}_{(i,j,k)} + \frac{\Delta t}{\epsilon(i,j,k)\Delta x}\left[H_y^n{}_{(i+1,j,k)} - H_y^n{}_{(i,j,k)}\right] - \frac{\Delta t}{\epsilon(i,j,k)\Delta y}\left[H_x^n{}_{(i,j+1,k)} - H_x^n{}_{(i,j,k)}\right] \tag{2.29}$$

(2.24), (2.25), (2.26), (2.27), (2.28), (2.29) are the main equations of the explicit FDTD method.

2.1.3 Features of the FDTD Method

The FDTD equations of (2.24) through (2.29) imply some obvious strengths and weaknesses of the FDTD method. Some of these are mentioned below:

- The FDTD equations can be easily implemented in computer programmes.
- By defining the FDTD grid properly and specifying the values of ϵ and μ at the grid points, computational space with any medium can be easily modelled.
- Since the problems need to be mapped into the cells of the computational space, the bigger the problem, the bigger the computational space. Therefore, the requirements of the memory increases with the size of the problem as all the values of **E** and **H** fields and media parameters at each grid point need to be stored.
- By using the rectangular grids of the explicit FDTD method it is not possible to accurately model the curves in the problem geometry. This is usually done by staircasing in the computational space but it leads to large computational errors.
- As the FDTD method is a time-domain technique only a single simulation run is required to get the results for the whole frequency range of interest. Because of this the FDTD method can be an attractive choice for modelling wideband systems. However, in (2.24) through (2.29) the media parameters are considered to be independent of the frequency, whereas in wideband systems these parameters vary with the frequency. So inaccuracies will appear if the standard FDTD method is used in modelling problems in wideband frequency without any modification.
- From the spatial discretization viewpoint there can be two extreme possibilities: electrically *small* geometrical details and electrically *large* geometrical details [26]. For electrically small geometrical problems, the spatial variations of the fields are dominated by the geometry instead of by the wavelength and a fine discretization must resolve the geometrical details. On the other hand, for the electrically large geometrical problems, the spatial variations of the fields are dominated by the minimum wavelength and a proper space-step must be chosen to achieve an accurate overall resolution.

For the intermediate region (electrical size of geometrical details comparable to the wavelength) the space-steps must accurately resolve both spatial variations.

- The choice of the temporal discretization Δt in Yee FDTD method is restricted by the CFL stability criterion [27] (described in Section 2.4). This upper bound on the temporal discretization affects the computational efficiency of the FDTD method.

2.2 Absorbing Boundary Condition

The computational domain of the FDTD method needs to be appropriately terminated because computational resources are limited. The original Yee's FDTD method does not explicitly contain any boundary information. The FDTD method enables to compute the electric and magnetic fields inside the computational domain through the update equations that use the field values at previous time steps at these locations and those at the nearest neighbours. But the electric fields at the boundaries can not be calculated in this way because values of the magnetic fields outside the domain are required. A boundary condition would allow to calculate the electric fields at the boundaries by using the field values available in the interior region. Without introducing boundary condition the FDTD method can not be used to solve practical problems.

An absorbing boundary condition (ABC) on the periphery of the computational domain simulates it to be extended to the infinity. The ABC needs to absorb the reflections of outgoing waves to an acceptable level that otherwise will make the desired simulation data spurious. Different ways to approach the outer boundary condition issues lead to the development of different ABCs. One of the oldest and well-known boundary condition is Mur's absorbing boundary condition [28]. It is relatively simple and has been successfully used to solve many engineering problems. Its implementation is straightforward and it is relatively less computationally demanding. However, Mur's ABC has room for improvement in terms of the accuracy of the solution it generates [29]. To improve its accuracy, Mei and Fang [30] have proposed the so called super absorption technique, while Chew [31] has introduced Liao's boundary condition. Both of these exhibit better characteristics than that of Mur especially for obliquely incident waves [32].

However, many of these boundary conditions can suffer from either unstability or inaccurate solutions. Therefore, the quest for robust and effective boundary conditions continued until the perfectly matched layer (PML) was introduced by Bérenger [33]. PML is actually an artificial (mathematical) anisotropic material which is inserted in the periphery of the computational domain in order to absorb the outgoing waves. This anisotropic material can be perfectly matched to free space at all incident angles and frequencies, provided that the interface is an infinite plane. Thus an infinite PML can absorb the incoming waves at all frequencies and for all incident angles. Since the pioneering work of Bérenger different versions of PML have been proposed in the literature: unsplit PML (UPML) [34], stretched coordinate PML [35][36], time convolution PML [37]. While the main concept of these versions of PMLs is still the same, different versions lead to different computer codes when implemented in the FDTD method [38].

Mur's ABC can give reasonably good results in the FDTD simulations of objects like waveguides, patch antennas and microwave circuits without paying the heavy computational expenses required by some other types of ABCs. Although higher-order Mur's ABCs are superior to first-order ABCs their implementation is more complex. Second-order Mur's ABC requires tangential derivatives on the boundary but sufficient information is not available to perform the derivative on the corners. This is why second-order Mur's ABC is not very good in handling the corner regions of the boundary [39]. Higher-order Mur's ABCs are not well-suited for parallel processing because they require exchange of field information that places a heavier burden in terms of communication than do the first-order Mur's ABC [40]. In this thesis all the proposed FDTD methods are terminated by first-order Mur's ABC.

Mur's ABC can be viewed as an approximation of the Engquist-Majda boundary condition [41]. For one dimension a plane wave propagating along the negative x-direction (Fig. 2.2) can be represented by the function $\phi(x+ct)$ and satisfies [27]:

$$\left(\frac{\partial}{\partial x} - \frac{1}{c}\frac{\partial}{\partial t}\right)\phi(x,t) = 0 \tag{2.30}$$

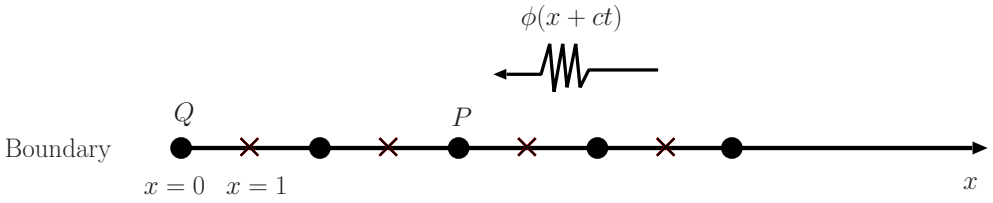


Figure 2.2: Explanation of Mur's absorbing boundary condition

Here ϕ is any function of $(x+ct)$, c is the wave propagation velocity in free space and t is the time. In Fig. 2.2 the field inside the computational domain (for example, point P) can be computed through the FDTD update equations but the field at a point on the boundary (for example, point Q) has to be computed by using an ABC. In Engquist-Majda ABC the field at the boundary of the domain is expressed in terms of the known fields in the interior of the domain. Thus when the wave is normally-incident plane wave using forward difference formula of (2.4) the field at the boundary $x=0$ and at time $(n+1)\Delta t$ can be approximated as:

$$\frac{\phi^n_{x=1}-\phi^n_{x=0}}{\Delta x}=\frac{1}{c}\frac{\phi^{n+1}_{x=0}-\phi^n_{x=0}}{\Delta t} \tag{2.31}$$

$$\therefore\ \phi^{n+1}_{x=0}=\big(1-\frac{c\Delta t}{\Delta x}\big)\phi^n_{x=0}+\frac{c\Delta t}{\Delta x}\phi^n_{x=1} \tag{2.32}$$

In (2.32) the field at the domain boundary is expressed in terms of the fields at the boundary and adjacent to the boundary (inside the domain), both sampled at the previous time step. Equation (2.32) is valid for a normally-incident plane wave because (2.30) is an 1-D wave equation. In Mur's ABC, unlike Engquist-Majda boundary condition, (2.30) is approximated by using central differencing in both the time and spatial domains:

$$\frac{1}{\Delta x}\left[\phi^{n+\frac{1}{2}}_{x=1}-\phi^{n+\frac{1}{2}}_{x=0}\right]=\frac{1}{c\Delta t}\left[\phi^{n+1}_{x=\frac{1}{2}}-\phi^{n}_{x=\frac{1}{2}}\right] \tag{2.33}$$

(2.33) is second-order accurate. However it includes neither the field at the boundary nor at time $(n+1)\Delta t$. Therefore, Mur represented $\phi^{(n+\frac{1}{2})}_{x=0}$ and $\phi^n_{x=\frac{1}{2}}$ averaging the two adjacent fields in both time and spatial domains:

$$\phi_{x=0}^{n+\frac{1}{2}} = \frac{1}{2}\left(\phi_{x=0}^{n+1} + \phi_{x=0}^{n}\right) \tag{2.34}$$

$$\phi_{x=\frac{1}{2}}^{n} = \frac{1}{2}\left(\phi_{x=1}^{n} + \phi_{x=0}^{n}\right) \tag{2.35}$$

Substituting (2.34) and (2.35) into (2.33) yields Mur's first-order boundary condition:

$$\phi_{x=0}^{n+1} = \phi_{x=1}^{n} + \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\left[\phi_{x=1}^{n+1} - \phi_{x=0}^{n}\right] \tag{2.36}$$

If the boundary is not too close to the simulated objects and excitation sources, Mur's first-order ABC can give results of acceptable accuracy. An advantage of Mur's first-order ABC is that it can be easily incorporated in a code designed for parallel processing [40]. This is because the computation of the fields at the boundaries only requires the knowledge of previous values at the same locations and the field values adjacent to the boundary at the same time step. Thus all the information needed for parallel processing is available in the individual parallel computational subdomains. However, for the problems dealing with propagation of waves at highly oblique angles or requiring very high accuracy Mur's first-order ABC might not be the most suitable boundary condition.

2.3 Frequency Dependent Media

In the explicit FDTD equations of (2.24) through (2.29), the media parameters are specified as frequency-independent constants but, in reality, these parameters depend on the frequency. Media parameters can be considered constant only within a narrow frequency band. If a broadband pulse is propagated through such a medium the frequency dependence of the medium has to be taken into account [11]. An example of how relative permittivity and conductivity of the grey matter of human brain vary over frequency is shown in Fig. 2.3. Tissues of the human body are frequency dependent and the dielectric properties of these tissues were measured by Gabriel *et al* in the frequency range of 10 Hz to 20

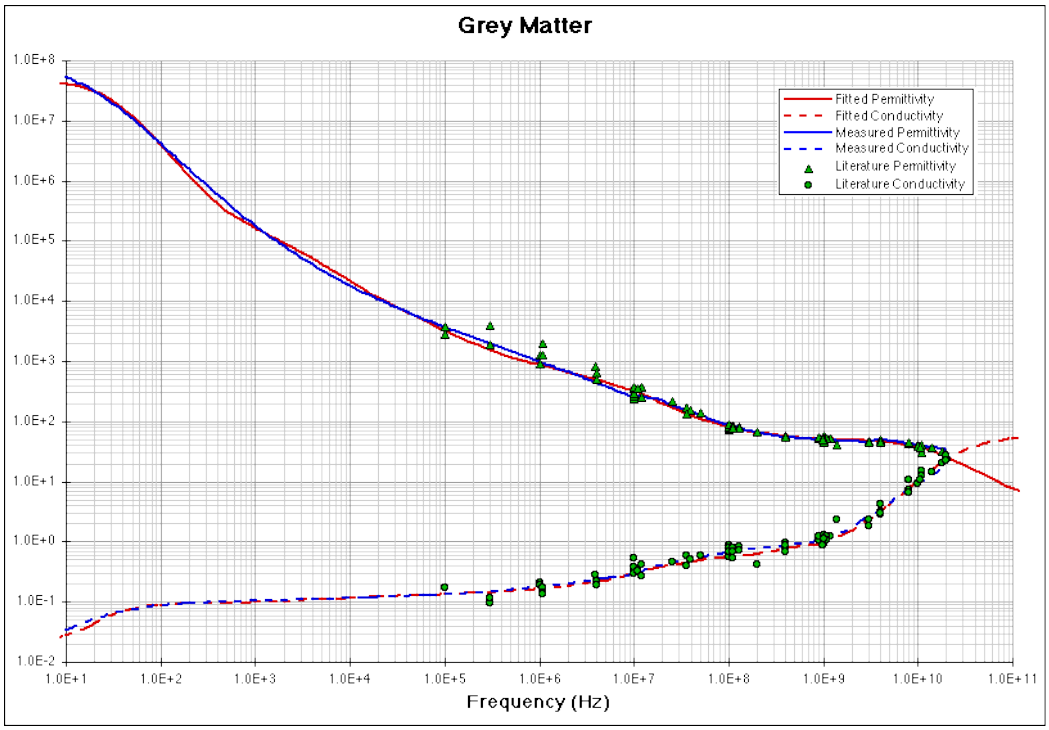


Figure 2.3: Dielectric properties of the grey matter of human brain varies with the frequency. (From the database of dielectric properties of human tissues compiled by Gabriel *et al* [1])

GHz [42] [43] [44] [1]. The variation of the dielectric constant with frequency is called dispersion and the medium for which ϵ and μ are functions of frequency is called dispersive medium. When ϵ and μ are independent of frequency it is non-dispersive medium. A dispersive medium responds to an electromagnetic field as the superposition of two responses: an instantaneous response (also called infinite frequency response), and a retarded response coming from the energy initially absorbed by the medium and subsequently returned by it. This second response is delayed in time because of material inertia and is responsible for the energy dispersion [24]. The ways in which physical processes in a material can affect the electric field are described through complex permittivity of the form [45]: $\epsilon = \epsilon' - \jmath\epsilon''$. The real part of the permittivity ϵ' is a measure of how much energy from an external electric field is stored in a material. The imaginary part of the permittivity ϵ'' is called the loss factor and is a measure of how dissipative or lossy a material is to an external electric field.

Three models are mainly used to represent frequency dispersive materials:

Debye Model, *Lorentz-Drude Model*, *Cole-Cole Model*. At wide frequency band the frequency dispersion characteristics of the dispersive media can be explained by the relaxation time required for dipoles to become oriented or polarized, on the application of an external electric field [46]. At such frequencies the polar molecules of these media tend to rotate as if they are in a damping frictional medium. In Debye and Cole-Cole models, the relaxation time of a material τ_{D} is used to describe the dispersive materials. Lorentz-Drude model is based on the motion of bounded charges and gives a system with a couple of resonant frequencies [47]. Debye model is the most widely used model in the FDTD method because of its simplicity of implementation. One of the objectives of this research is to observe the interaction of electromagnetic waves with the human body. Cole-Cole and Debye models are the most appropriate models to represent frequency dispersive biological tissues. Out of these, the Debye model is chosen to use in the FDTD methods proposed in this thesis because it is widely used and easy to implement.

The single-pole Debye model is given as:

$$\epsilon_r = \epsilon_\infty + \frac{\epsilon_{\mathrm{S}} - \epsilon_\infty}{1 + \jmath\omega\tau_{\mathrm{D}}} \tag{2.37}$$

where, ϵ_r is the relative permittivity, ϵ_{S} is the static permittivity, ϵ_∞ is the optical permittivity, τ_{D} is the characteristic relaxation time of the dipole moment of the molecule, ω is the angular frequency. This model can be extended to account for the losses due to the conduction currents and the static conductivity as follows:

$$\epsilon_r = \epsilon_\infty + \frac{\epsilon_{\mathrm{S}} - \epsilon_\infty}{1 + \jmath\omega\tau_{\mathrm{D}}} - \jmath\frac{\sigma}{\omega\epsilon_0} \tag{2.38}$$

where, ϵ_0 is the free space permittivity. The wideband frequency dispersion characteristics of the dielectric materials can be more accurately described by introducing multiple dispersion processes. Multiple Debye terms are included in the multi-pole Debye model with relaxation times well separated:

$$\epsilon_r = \epsilon_\infty + \sum_k \frac{\Delta\epsilon_k}{1 + \jmath\omega\tau_{Dk}} - \jmath\frac{\sigma}{\omega\epsilon_0} \tag{2.39}$$

where, τ_{Dk} is the relaxation time and $\Delta\epsilon_k$ is the change in the relative permittivity over the k^{th} dispersion region. For example, in the case of 2-pole Debye model $k = 1, 2$, $\Delta\epsilon_1 = \epsilon_{\rm S} - \epsilon_{\rm m}$ and $\Delta\epsilon_2 = \epsilon_{\rm m} - \epsilon_\infty$, where, $\epsilon_{\rm m}$ is the intermediate relative permittivity. Although multi-pole Debye models account for the different dispersion regions, their implementation in the FDTD method may be quite complicated. On the other hand, with the advantage of less complicated implementation it may be the case that single-pole Debye model can adequately describe the materials of a certain problem.

2.4 Unconditionally Stable FDTD Method

To ensure the accuracy of the FDTD method spatial sampling must be smaller compared to the wavelength (usually less than or equal to one-tenth of the wavelength) [3]. That means, for higher frequencies when the wavelength is very small spatial sampling should be even smaller. In the FDTD method time- and space-steps are related. Because, for example, if the time-step is very large the wave has to travel over several spatial cells in one time-step. But the field values at each cell is updated based on the values at the adjacent cells. So a large time-step may cause major problem and the simulation may become unstable. This frames the so called Courant-Friedrichs-Lewy (CFL) stability condition [48][49] which the explicit FDTD method must satisfy to maintain its stability:

$$\Delta t \leq \frac{1}{c}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)^{-\frac{1}{2}} \tag{2.40}$$

Thus, there is an upper bound on the time-step depending on the minimum spatial step. This restriction renders the FDTD method inefficient. To overcome this limitation there is a growing interest in the unconditionally stable implicit FDTD methods for which time-step and space-step can be independently chosen. The Crank–Nicolson [15] FDTD (CN–FDTD) method is such a method which provides unconditional stability beyond the CFL limit. In the CN–FDTD method the time and space derivatives are discretized by centred differences while

the fields affected by the curl operators are averaged in time. In the standard FDTD method unknown values are solved explicitly using the values at the previous time steps while in implicit methods, such as in CN–FDTD method, all the values of either **E** or **H** are computed at the same time by solving a system of linear equations. Implicit methods allow relatively larger time-step by doing more computations at each step of the simulation.

Maxwell's equations ((2.12) through (2.17)) can be written as

$$\frac{\partial U}{\partial t} = AU + BU \tag{2.41}$$

where,

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{\epsilon}\frac{\partial}{\partial y} \\ 0 & 0 & 0 & \frac{1}{\epsilon}\frac{\partial}{\partial z} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\epsilon}\frac{\partial}{\partial x} & 0 \\ 0 & \frac{1}{\mu}\frac{\partial}{\partial z} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\mu}\frac{\partial}{\partial x} & 0 & 0 & 0 \\ \frac{1}{\mu}\frac{\partial}{\partial y} & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.42}$$

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 & -\frac{1}{\epsilon}\frac{\partial}{\partial z} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\epsilon}\frac{\partial}{\partial x} \\ 0 & 0 & 0 & -\frac{1}{\epsilon}\frac{\partial}{\partial y} & 0 & 0 \\ 0 & 0 & -\frac{1}{\mu}\frac{\partial}{\partial y} & 0 & 0 & 0 \\ -\frac{1}{\mu}\frac{\partial}{\partial z} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\mu}\frac{\partial}{\partial x} & 0 & 0 & 0 & 0 \end{pmatrix} \tag{2.43}$$

$$U = \begin{bmatrix} E_x & E_y & E_z & H_x & H_y & H_z \end{bmatrix}^T \tag{2.44}$$

Using finite differences for the time derivatives and averaging the fields over time, from (2.41) the CN–FDTD method is found:

$$\frac{U^{n+1}-U^n}{\Delta t} = (A+B)\frac{U^{n+1}+U^n}{2}$$

$$\therefore\ U^{n+1}-U^n = (\frac{\Delta t}{2}A + \frac{\Delta t}{2}B)(U^{n+1}+U^n)$$

$$\therefore\ U^{n+1} - U^{n+1}(\frac{\Delta t}{2}A + \frac{\Delta t}{2}B) = U^n + U^n(\frac{\Delta t}{2}A + \frac{\Delta t}{2}B)$$

$$\therefore\ (I - \frac{\Delta t}{2}A - \frac{\Delta t}{2}B)U^{n+1} = (I + \frac{\Delta t}{2}A + \frac{\Delta t}{2}B)U^n \tag{2.45}$$

Here, I is a 6×6 identity matrix. Now, to write (2.45) in a factored form the formula, $(1 \pm a_1)(1 \pm a_2) = 1 \pm a_1 \pm a_2 \pm a_1 a_2$ is used. Assuming $a_1 = \frac{\Delta t}{2}A$, $a_2 = \frac{\Delta t}{2}B$ and adding $a_1 a_2 = \frac{\Delta t^2}{4}AB$ on both sides of (2.45)

$$(I - \frac{\Delta t}{2}A - \frac{\Delta t}{2}B + \frac{\Delta t^2}{4}AB)U^{n+1} = (I + \frac{\Delta t}{2}A + \frac{\Delta t}{2}B + \frac{\Delta t^2}{4}AB)U^n + \frac{\Delta t^2}{4}AB(U^{n+1} - U^n)$$

$$\therefore\ (I - \frac{\Delta t}{2}A)(I - \frac{\Delta t}{2}B)U^{n+1} = (I + \frac{\Delta t}{2}A)(I + \frac{\Delta t}{2}B)U^n + \frac{\Delta t^2}{4}AB(U^{n+1} - U^n) \tag{2.46}$$

The CN–FDTD equation (2.46) involves continuous spatial derivatives of fields. Its discretisation in space yields a large sparse matrix and therefore until recently it has not been used in time domain electromagnetics. Instead, there have been many works attempting to simplify or approximate its implementation: the alternating direction implicit (ADI–FDTD) method [17][50], the CN

Douglas Gunn method [18], the CN cycle sweep method [19], the CN approximate factorization splitting method [20]. However, these approximations suffer up to some extent of numerical errors, which may become severe for some practical applications [51, 21, 52]. Among these methods, the ADI–FDTD method has drawn much interest from the researchers over the last decade. The ADI–FDTD method drops the last term of (2.46) and then splits it into two steps:

Step-1

$$(I - \frac{\Delta t}{2}A)U^{n+\frac{1}{2}} = (I + \frac{\Delta t}{2}B)U^{n} \tag{2.47}$$

Step-2

$$(I - \frac{\Delta t}{2}B)U^{n+1} = (I + \frac{\Delta t}{2}A)U^{n+\frac{1}{2}} \tag{2.48}$$

The advantage of ADI–FDTD method over CN–FDTD method is that the computational overhead is smaller because tridiagonal matrix systems are required to solve (rather than sparse matrix systems). So, the ADI–FDTD method is a computationally affordable approximation of the CN–FDTD method [23][21], found by adding a perturbation term to the latter. Because of the omission of the last term of (2.46) the ADI–FDTD method leads to truncation error which is a function of $(\frac{\Delta t}{2})^2$ times the space derivatives of the field [23]. This means that the truncation error increases with larger Δt, thus imposing a restriction on Δt, particularly when accuracy of the problem is crucial. Compared to the explicit FDTD method the ADI–FDTD method has larger numerical dispersion which increases with the temporal discretization. Dispersion analyses of the ADI–FDTD method have been shown in [53] assuming lossless media and in [54] for lossy media. The ADI–FDTD method also experiences numerical errors from the source condition [55]. In all, the ADI–FDTD method improves the computational efficiency at the cost of accuracy. On the other hand, although the CN–FDTD method has higher accuracy and lower anisotropy than the ADI–FDTD method [20] the computational costs are also higher. Ideally, an FDTD method should take merits of both of the methods.

There have been works to improve the accuracy of ADI–FDTD method towards CN–FDTD method in different ways [56][57]. The iterative ADI–FDTD

method of [56] involves loop of iterations at each time steps making it more computationally expensive than normal ADI–FDTD method. An alternative technique has been suggested by [57] employing an average approximation of some of the implicit fields. [56] and [57] showed the improvements of ADI–FDTD method for two dimensional problems while [58] and [59] reported these error reduction methods diverge in three dimensional cases. To reduce the numerical dispersion of the ADI–FDTD method higher-order methods have been proposed [60]. Another development in the area of unconditionally stable FDTD method is locally one–dimensional (LOD)–FDTD method [61]. The main advantage of the LOD–FDTD method is that the algorithm is quite simple with a subsequent reduction in computational time, while maintaining the accuracy comparable to the ADI–FDTD method [62]. The LOD–FDTD method is more efficient than other unconditionally stable methods due to lesser arithmetic operations. This fact has motivated other researchers to extend and improve the LOD–FDTD method from different viewpoints [63][64]. Recently several unconditionally stable split–step FDTD methods have also been proposed [65][66].

Studies on the original CN–FDTD method without any approximation is scarce because of the requirements of large sparse matrix computations. However, with the massive advancement of the technology of memory and computational resources, handling huge sparse matrices is no longer a bottleneck. This, together with the extensive researches during last two decades that resulted in highly sophisticated, robust, efficient and economical sparse solvers, makes the CN–FDTD method a promising affordable alternative to the explicit FDTD method. Quite recently [22] investigated the original CN–FDTD method for frequency-independent materials. To solve the generated sparse matrix of the frequency-independent CN–FDTD method [67] used iterative solvers and [68] used preconditioned iterative solvers.

2.5 Other FDTD Methods

From the beginning there have been a lot of research on further development of the original FDTD method. Some of these have dealt with key aspects of the method and provided enhanced solutions (mitigating dispersion, material modelling, geometrical modelling etc), while others have described completely new

concepts providing the method with a new dimension (PML ABCs, unconditionally stable techniques etc). Some of the most interesting developments are described here.

One of the major limitations of the FDTD method is its numerical dispersion [27]. Numerical dispersion causes the phase velocity in the FDTD grid to become a function of frequency and propagation angle. As a consequence, phase errors appear in narrowband simulations and pulse distortion appears in broadband simulations [69]. Because of its cumulative effect numerical dispersion becomes a dominant factor in large-scale simulations unless very small spatial discretization is used. But lowering the spatial discretization leads to prohibitively large memory requirements and computational costs. To address this issue a number of higher-order FDTD methods have been proposed: second-order accurate in time and fourth-order accurate in space (2,4), second-order accurate in time and sixth-order accurate in space (2,6) [70], fourth-order accurate in time and space (4,4) [71]. Higher-order FDTD methods exhibit reduced dispersion error levels with lower computational costs and memory requirements. This is particularly attractive for the analysis of electrically large problems because higher-order FDTD method allows an increased spatial discretization while maintaining a specified accuracy level [72]. Other noteworthy alternatives for low dispersion methods to accurately simulate electrically large problems are Pseudo Spectral Time Domain (PSTD) method [73], Multi-resolution in Time Domain (MRTD) techniques [74].

Although the FDTD method has the ability to handle problems incorporating materials with geometrical inhomogeneities, problems arise when the media interfaces are not plane because in the original form of the FDTD method it is difficult to model the structures that cannot fit well into Cartesian coordinates. Approximating the interfaces in such complex geometries through staircasing can lead to significant errors [75]. Errors also appear because of the assumption of constant fields inside the cells neighbouring material terminations (e.g. edges, corners, slots, wires etc). In solving this problem [76][77] suggested combining the FDTD method with another method that is suitable for accurately modelling problematic geometrical details: FDTD/FETD, FDTD/MoM methods.

Discretization of the entire computational space with a fine grid to accurately

handle the geometrical details and field variations will make the method computationally intensive. To circumvent this problem a few methods of obtaining a more refined mesh in a subregion have been reported. They can be divided into three main categories [78], namely: sequential computations, space-only subgridding and subgridding in space and time. In sequential computations, computation in the whole problem space is performed using a coarse grid followed by re-computation in a limited volume using a finer mesh [79]. Fields calculated by using coarse grid are used as the boundary values in the fine grid computation. In the space-only subgridding various grid-steps in different directions are implemented [80]. [81] improved the method to second-order accurate. However, numerical dispersion varies considerably with the density of the mesh [82] and due to the CFL stability condition, time-step is restricted by the smallest mesh size throughout the computational space making it computationally inefficient. Higher efficiency can be achieved with subgridding in space and time technique [83] as the time-step is set for each mesh separately. Bérenger proposed a new subgridding technique called Huygens Subgridding (HSG) [84] that connects the main grid and the subgrid regions using the Huygens-Kirchhoff principle [85]. HSG has the advantages of allowing arbitrarily large ratio of spatial resolutions. Furthermore, it can significantly reduce the spurious numerical reflections resulted at the interfaces between the main grid and the subgrid regions.

An alternative strategy to handle the staircasing error to simulate the objects having boundaries not coinciding with the Cartesian coordinates is to employ the conformal FDTD technique [86]. Conformal FDTD method requires less memory than the conventional non-uniform mesh methods and does not suffer from late time instability problems as many of the subgridding methods do.

Developments in the research on unconditionally stable FDTD methods have already been described in Section 2.4. As extremely powerful supercomputers are available these days for handling large-scale computations, parallelization of the FDTD method is becoming popular. The parallel FDTD algorithm gains the computational efficiency by distributing the computational burden over a cluster of processors. It also enables one to solve large problems that could be beyond the scope of a single processor because of CPU time limitations. Robust strategies of combining different algorithms with the parallelized FDTD have also been

suggested [40]; for example, parallelizing the conformal FDTD method and enhancing it with either subgridding, the ADI–FDTD algorithm or both.

As many current and emerging technological applications involve electromagnetic wave interactions with the materials having frequency dependent dielectric properties [12], development of efficient FDTD methods capable of handling frequency dependent materials is important. Frequency dependency has been incorporated in the FDTD method mainly by the auxiliary differential equation method [87], the z–transform method [88] and the discrete convolution method [89]. To be able to handle practical applications, frequency dependency has been included in many of the major FDTD techniques, namely, ADI–FDTD method [90][91], subgridding FDTD method [92], higher-order FDTD method [93], LOD–FDTD method [62], split-step FDTD method [94], conformal FDTD method [95]. Inclusion of frequency dependent materials is a significant development but it comes at the cost of higher memory and computational time requirements.

Since the introduction of Yee's FDTD method many variations and extensions of it have been proposed and the literature on the FDTD techniques is extensive. Major trends and developments of FDTD researches have been surveyed here but this is obviously not exhaustive.

Chapter 3

Frequency Dependent Crank Nicolson FDTD Method

3.1 Introduction

The main drawback of the conventional, explicit FDTD method is the reduced computational efficiency resulting from the upper limit on the time-step that needs to satisfy the CFL stability condition [27]. Thus, when very small spatial step relative to the wavelength of interest is employed to accurately model the fine geometrical details of a given application, an unnecessarily small time-step is enforced, with an increase of the total CPU time. An alternative to the explicit FDTD method is provided by the CN–FDTD method [15], which presents unconditional stability beyond the CFL limit. Both methods share in common the discretization of the time and space derivatives by centred differences, with the only difference being that the fields affected by the curl operator are averaged in time by the CN–FDTD method, whereas in the explicit FDTD method they are not. The resulting method is a fully implicit marching–on–in–time algorithm with the same potential of the classical FDTD method. However, despite its accuracy and low anisotropy [16] the CN–FDTD method has not been widely used in time domain electromagnetics as it involves large sparse matrix computations. Instead, there have been many works attempting to simplify or approximate its implementation [96]. Such approximations suffer from some extent of numerical errors, which may become severe for some practical applications [21]. With the massive advancement of the technology of memory and computational resources, handling huge sparse matrices is no longer a bottleneck. This, together with the

extensive researches during last two decades that resulted in highly sophisticated, robust, efficient and economical sparse solvers, makes the CN–FDTD method a promising affordable alternative to the explicit FDTD method.

In this chapter a new three-dimensional frequency dependent CN-FDTD method (FD–CN–FDTD) is proposed. Frequency dependence of single–pole Debye materials is incorporated into the CN–FDTD method by means of an auxiliary differential formulation [87]. Mur's first-order absorbing boundary condition [28] is used to terminate the boundaries in the FD–CN–FDTD method.

3.2 Formulation of the FD–CN–FDTD Method

The differential time domain Maxwell's equations in material independent form are

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{3.1}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \tag{3.2}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{3.3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{3.4}$$

where

$$\mathbf{D} = \epsilon \mathbf{E} \tag{3.5}$$

$$\mathbf{B} = \mu \mathbf{H} \tag{3.6}$$

Here **E** is the electric field, **H** is the magnetic field, **D** is the electric flux density, **B** is the magnetic flux density, **J** is the conduction current density, ρ is the charge density, ϵ is the permittivity of the medium, μ is the permeability of the medium. (3.1) through (3.6) provides all the information to describe the behaviour of the field in the linear, isotropic and non–magnetic medium provided that the initial field distribution is specified. In fact, the two curl equations ((3.1) and (3.2)) contain the two divergence equations ((3.3) and (3.4)) and are only required to be considered. For most materials the relative permeability, μ_r, is very near to the unity [p.85, [97]]; therefore, in this thesis permeability of all the materials is considered to be that of the vacuum (μ_0) *i.e.* $\mu = \mu_r\mu_0 = 4\pi \times 10^{-7}$ H/m .

For many materials of interest, the constitutive parameters change over a wide band of frequencies [45]. Such frequency dependent materials can be described by the Debye model. In the FD–CN–FDTD method single–pole Debye model is used which defines the complex relative permittivity as

$$\epsilon_r = \epsilon_\infty + \frac{\epsilon_S - \epsilon_\infty}{1 + \jmath\omega\tau_D} - \jmath\frac{\sigma}{\omega\epsilon_0} \tag{3.7}$$

where ϵ_r is the complex relative permittivity, ϵ_0 is the free space permittivity, ϵ_S is the static permittivity, ϵ_∞ is the optical permittivity, τ_D is the characteristic relaxation time, σ is the static conductivity, ω is the angular frequency.

Using (3.5) and (3.7) the constitutive relationship for single–pole Debye electrically–dispersive media can be found (in frequency domain):

$$\begin{aligned}
\mathbf{D} &= \epsilon_0\epsilon_r\mathbf{E} \qquad (3.8)\\
&\epsilon_0(\epsilon_\infty + \frac{\epsilon_S - \epsilon_\infty}{1 + \jmath\omega\tau_D} - \jmath\frac{\sigma}{\omega\epsilon_0})\mathbf{E}\\
&= (\epsilon_0\epsilon_\infty + \frac{\epsilon_0\epsilon_S - \epsilon_0\epsilon_\infty}{1 + \jmath\omega\tau_D} + \frac{\sigma}{\jmath\omega})\mathbf{E}\\
&= \frac{(\jmath\omega)^2\epsilon_0\epsilon_\infty\tau_D + \jmath\omega(\epsilon_0\epsilon_S + \sigma\tau_D) + \sigma}{\jmath\omega(1 + \jmath\omega\tau_D)}\mathbf{E}
\end{aligned}$$

(3.8) is simplified to

$$(\jmath\omega)^2\tau_{\mathrm{D}}\mathbf{D} + \jmath\omega\mathbf{D} = (\jmath\omega)^2\epsilon_0\epsilon_\infty\tau_{\mathrm{D}}\mathbf{E} + \jmath\omega(\epsilon_0\epsilon_{\mathrm{S}} + \sigma\tau_{\mathrm{D}})\mathbf{E} + \sigma\mathbf{E} \tag{3.9}$$

Mapping $(\jmath\omega)^m$, in frequency domain, into $\frac{\partial^m}{\partial t^m}$, in time domain, (3.9) can be written as a differential equation in time domain:

$$\tau_{\mathrm{D}}\frac{\partial^2\mathbf{D}}{\partial t^2} + \frac{\partial\mathbf{D}}{\partial t} = \epsilon_0\epsilon_\infty\tau_{\mathrm{D}}\frac{\partial^2\mathbf{E}}{\partial t^2} + (\epsilon_0\epsilon_{\mathrm{S}} + \sigma\tau_{\mathrm{D}})\frac{\partial\mathbf{E}}{\partial t} + \sigma\mathbf{E} \tag{3.10}$$

For x-direction, when (3.10) is discretized and the last term is averaged over time, the following equation is obtained

$$\begin{aligned}
\tau_{\mathrm{D}(i,j,k)}\frac{{D_x}^{n+1}{}_{(i,j,k)} - 2{D_x}^{n}{}_{(i,j,k)} + {D_x}^{n-1}{}_{(i,j,k)}}{(\Delta t)^2} & \\
+\frac{{D_x}^{n+1}{}_{(i,j,k)} - {D_x}^{n}{}_{(i,j,k)}}{\Delta t} & \\
= \epsilon_0\epsilon_{\infty(i,j,k)}\tau_{\mathrm{D}(i,j,k)}\frac{{E_x}^{n+1}{}_{(i,j,k)} - 2{E_x}^{n}{}_{(i,j,k)} + {E_x}^{n-1}{}_{(i,j,k)}}{(\Delta t)^2} & \\
+\left(\epsilon_0\epsilon_{\mathrm{S}(i,j,k)} + \sigma_{(i,j,k)}\tau_{\mathrm{D}(i,j,k)}\right)\frac{{E_x}^{n+1}{}_{(i,j,k)} - {E_x}^{n}{}_{(i,j,k)}}{\Delta t} & \\
+\sigma_{(i,j,k)}\frac{{E_x}^{n+1}{}_{(i,j,k)} + {E_x}^{n}{}_{(i,j,k)}}{2} &
\end{aligned} \tag{3.11}$$

(3.11) can be simplified to obtain ${E_x}^{n+1}{}_{(i,j,k)}$

$$\begin{aligned}
{E_x}^{n+1}{}_{(i,j,k)} = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}{D_x}^{n+1}{}_{(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)}{D_x}^{n}{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)}{D_x}^{n-1}{}_{(i,j,k)} & \\
-\frac{\nu_5(i,j,k)}{\nu_4(i,j,k)}{E_x}^{n}{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)}{E_x}^{n-1}{}_{(i,j,k)} &
\end{aligned} \tag{3.12}$$

where,

$$\nu_1(i,j,k) = \frac{\tau_{\mathrm{D}(i,j,k)}}{(\Delta t)^2} + \frac{1}{\Delta t} \tag{3.13}$$

$$\nu_2(i,j,k) = \frac{-2\tau_{\mathrm{D}^{(i,j,k)}}}{(\Delta t)^2} - \frac{1}{\Delta t} \tag{3.14}$$

$$\nu_3(i,j,k) = \frac{\tau_{\mathrm{D}^{(i,j,k)}}}{(\Delta t)^2} \tag{3.15}$$

$$\nu_4(i,j,k) = \frac{\epsilon_0\epsilon_{\infty}{}^{(i,j,k)}\tau_{\mathrm{D}^{(i,j,k)}}}{(\Delta t)^2} + \frac{\epsilon_0\epsilon_{\mathrm{S}^{(i,j,k)}} + \sigma^{(i,j,k)}\tau_{\mathrm{D}^{(i,j,k)}}}{\Delta t} + \frac{\sigma^{(i,j,k)}}{2} \tag{3.16}$$

$$\nu_5(i,j,k) = \frac{-2\epsilon_0\epsilon_{\infty}{}^{(i,j,k)}\tau_{\mathrm{D}^{(i,j,k)}}}{(\Delta t)^2} - \frac{\epsilon_0\epsilon_{\mathrm{S}^{(i,j,k)}} + \sigma^{(i,j,k)}\tau_{\mathrm{D}^{(i,j,k)}}}{\Delta t} + \frac{\sigma^{(i,j,k)}}{2} \tag{3.17}$$

$$\nu_6(i,j,k) = \frac{\epsilon_0\epsilon_{\infty}{}^{(i,j,k)}\tau_{\mathrm{D}^{(i,j,k)}}}{(\Delta t)^2} \tag{3.18}$$

For the source-free medium ($\mathbf{J} = 0$) using (3.6), Maxwell's curl equations ((3.1) and (3.2)) can be written in scalar form:

$$\frac{\partial H_x}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y}\right) \tag{3.19}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}\right) \tag{3.20}$$

$$\frac{\partial H_z}{\partial t} = \frac{1}{\mu}\left(\frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}\right) \tag{3.21}$$

and

$$\frac{\partial D_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{3.22}$$

$$\frac{\partial D_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \tag{3.23}$$

$$\frac{\partial D_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{3.24}$$

Now in the CN method the time and space derivatives are discretized by centred differences while the fields affected by the curl operators are averaged in time. The method uses the same Yee grid as the conventional FDTD. Using this approach ${H_x}^{n+1}{}_{(i,j,k)}$, ${H_y}^{n+1}{}_{(i,j,k)}$ and ${H_z}^{n+1}{}_{(i,j,k)}$ are obtained from (3.19), (3.20) and (3.21) while ${D_x}^{n+1}{}_{(i,j,k)}$, ${D_y}^{n+1}{}_{(i,j,k)}$ and ${D_z}^{n+1}{}_{(i,j,k)}$ are obtained from (3.22) , (3.23) and (3.24), respectively:

$${H_x}^{n+1} = {H_x}^n + \frac{\Delta t}{2\mu}\left(\frac{\partial {E_y}^{n+1}}{\partial z} - \frac{\partial {E_z}^{n+1}}{\partial y} + \frac{\partial {E_y}^n}{\partial z} - \frac{\partial {E_z}^n}{\partial y}\right) \tag{3.25}$$

$${H_y}^{n+1} = {H_y}^n + \frac{\Delta t}{2\mu}\left(\frac{\partial {E_z}^{n+1}}{\partial x} - \frac{\partial {E_x}^{n+1}}{\partial z} + \frac{\partial {E_z}^n}{\partial x} - \frac{\partial {E_x}^n}{\partial z}\right) \tag{3.26}$$

$${H_z}^{n+1} = {H_z}^n + \frac{\Delta t}{2\mu}\left(\frac{\partial {E_x}^{n+1}}{\partial y} - \frac{\partial {E_y}^{n+1}}{\partial x} + \frac{\partial {E_x}^n}{\partial y} - \frac{\partial {E_y}^n}{\partial x}\right) \tag{3.27}$$

$${D_x}^{n+1} = {D_x}^n + \frac{\Delta t}{2}\left(\frac{\partial {H_z}^{n+1}}{\partial y} - \frac{\partial {H_y}^{n+1}}{\partial z} + \frac{\partial {H_z}^n}{\partial y} - \frac{\partial {H_y}^n}{\partial z}\right) \tag{3.28}$$

$$D_y{}^{n+1} = D_y{}^n + \frac{\Delta t}{2}\left(\frac{\partial H_x{}^{n+1}}{\partial z} - \frac{\partial H_z{}^{n+1}}{\partial x} + \frac{\partial H_x{}^n}{\partial z} - \frac{\partial H_z{}^n}{\partial x}\right) \tag{3.29}$$

$$D_z{}^{n+1} = D_z{}^n + \frac{\Delta t}{2}\left(\frac{\partial H_y{}^{n+1}}{\partial x} - \frac{\partial H_x{}^{n+1}}{\partial y} + \frac{\partial H_y{}^n}{\partial x} - \frac{\partial H_x{}^n}{\partial y}\right) \tag{3.30}$$

Substituting the values of $H_y^{n+1}{}_{(i,j,k)}$ and $H_z^{n+1}{}_{(i,j,k)}$ from (3.26) and (3.27) in (3.28) to get $D_x^{n+1}{}_{(i,j,k)}$ and then using the value of $D_x^{n+1}{}_{(i,j,k)}$ in (3.12) give a equation of $E_x{}^{n+1}$, $E_y{}^{n+1}$ and $E_z{}^{n+1}$. Taking all the $(n+1)$ terms on the left hand side this equation eventually becomes

$$\begin{aligned}
&E_x{}^{n+1} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_x{}^{n+1}}{\partial y^2} + \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_y{}^{n+1}}{\partial x \partial y} \\
&\quad + \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_z{}^{n+1}}{\partial z \partial x} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_x{}^{n+1}}{\partial z^2} \\
&= \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}D_x{}^n + \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\frac{\Delta t}{2}\frac{\partial H_z{}^n}{\partial y} + \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_x{}^n}{\partial y^2} \\
&- \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_y{}^n}{\partial x \partial y} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\frac{\Delta t}{2}\frac{\partial H_y{}^n}{\partial z} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_z{}^n}{\partial z \partial x} \\
&+ \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{1}{\mu}\frac{\partial^2 E_x{}^n}{\partial z^2} + \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{\partial H_z{}^n}{\partial y} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\left(\frac{\Delta t}{2}\right)^2\frac{\partial H_y{}^n}{\partial z} \\
&\quad + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)}D_x{}^n + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)}D_x{}^{n-1} - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)}E_x{}^n - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)}E_x{}^{n-1}
\end{aligned} \tag{3.31}$$

When (3.31) is discretized in space and same terms are factored out properly (3.32) is obtained

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)\right]E_x^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j+1,k)}-\frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j-1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k-1)} \\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i,j+1,k)}-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i-1,j+1,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i-1,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i-1,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i-1,j,k)} \\
&=P_2(i,j,k)D_x^n{}_{(i,j,k)}+P_3(i,j,k)D_x^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j+1,k)}-\frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k+1)}+\frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k)} \\
-P_6(i,j,k)E_x^{n-1}{}_{(i,j,k)}&-\left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)+P_5(i,j,k)\right]E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j+1,k)}+\frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j-1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j+1,k)}+\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j+1,k)}-\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k+1)}-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k)}
\end{aligned}
\tag{3.32}
$$

In (3.32) $P_1(i,j,k)$, $P_2(i,j,k)$, $P_3(i,j,k)$, $P_4(i,j,k)$, $P_5(i,j,k)$ and $P_6(i,j,k)$ have the following values:

$$
P_1(i,j,k)=\frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}(\frac{\Delta t}{2})^2\frac{1}{\mu} \tag{3.33}
$$

$$P_2(i,j,k) = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} \tag{3.34}$$

$$P_3(i,j,k) = \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} \tag{3.35}$$

$$P_4(i,j,k) = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}\Delta t \tag{3.36}$$

$$P_5(i,j,k) = \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} \tag{3.37}$$

$$P_6(i,j,k) = \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} \tag{3.38}$$

Now for y-direction (3.11) turns into

$$\begin{aligned} E_y^{n+1}{}_{(i,j,k)} = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} D_y^{n+1}{}_{(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_y^{n}{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} D_y^{n-1}{}_{(i,j,k)} \\ - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_y^{n}{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} E_y^{n-1}{}_{(i,j,k)} \end{aligned} \tag{3.39}$$

$H_x^{n+1}{}_{(i,j,k)}$ and $H_z^{n+1}{}_{(i,j,k)}$ from (3.25) and (3.27) are substituted into (3.29). Then the obtained $D_y^{n+1}{}_{(i,j,k)}$ is put into (3.39) which, in discretized form, gives

$$\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)\right]E_y^{n+1}{}_{(i,j,k)} \quad (3.40)\\
&-\frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k-1)}\\
&-\frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i+1,j,k)}-\frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i-1,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k)}\\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k+1)}+\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k)}\\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j,k)}-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j,k)}\\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j-1,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j-1,k)}\\
&=P_2(i,j,k)D_y^n{}_{(i,j,k)}+P_3(i,j,k)D_y^{n-1}{}_{(i,j,k)}\\
&+\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k+1)}-\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k)}\\
&-\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i+1,j,k)}+\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i,j,k)}\\
-P_6(i,j,k)E_y^{n-1}{}_{(i,j,k)}&-\left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)+P_5(i,j,k)\right]E_y^n{}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k-1)}\\
&+\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i-1,j,k)}\\
&-\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k+1)}-\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k)}\\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j-1,k)}-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j-1,k)}
\end{aligned}$$

Similarly for z-direction (3.11) is written as

$$\begin{aligned}
E_z^{n+1}{}_{(i,j,k)}=\frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}D_z^{n+1}{}_{(i,j,k)}&+\frac{\nu_2(i,j,k)}{\nu_4(i,j,k)}D_z^n{}_{(i,j,k)}+\frac{\nu_3(i,j,k)}{\nu_4(i,j,k)}D_z^{n-1}{}_{(i,j,k)} \quad (3.41)\\
&-\frac{\nu_5(i,j,k)}{\nu_4(i,j,k)}E_z^n{}_{(i,j,k)}-\frac{\nu_6(i,j,k)}{\nu_4(i,j,k)}E_z^{n-1}{}_{(i,j,k)}
\end{aligned}$$

In this case ${H_x^{n+1}}_{(i,j,k)}$ and ${H_y^{n+1}}_{(i,j,k)}$ from (3.25) and (3.26), respectively, are substituted into (3.30) and the resulting ${D_z^{n+1}}_{(i,j,k)}$ is used in (3.41). This yields

$$\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)\right]{E_z^{n+1}}_{(i,j,k)}-\frac{P_1(i,j,k)}{\Delta x^2}{E_z^{n+1}}_{(i+1,j,k)} \quad (3.42)\\
&-\frac{P_1(i,j,k)}{\Delta x^2}{E_z^{n+1}}_{(i-1,j,k)}-\frac{P_1(i,j,k)}{\Delta y^2}{E_z^{n+1}}_{(i,j+1,k)}-\frac{P_1(i,j,k)}{\Delta y^2}{E_z^{n+1}}_{(i,j-1,k)}\\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}{E_x^{n+1}}_{(i+1,j,k)}-\frac{P_1(i,j,k)}{\Delta z\Delta x}{E_x^{n+1}}_{(i,j,k)}-\frac{P_1(i,j,k)}{\Delta z\Delta x}{E_x^{n+1}}_{(i+1,j,k-1)}\\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}{E_x^{n+1}}_{(i,j,k-1)}+\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n+1}}_{(i,j+1,k)}-\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n+1}}_{(i,j,k)}\\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n+1}}_{(i,j+1,k-1)}+\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n+1}}_{(i,j,k-1)}\\
&=P_2(i,j,k){D_z^{n}}_{(i,j,k)}+P_3(i,j,k){D_z^{n-1}}_{(i,j,k)}\\
&+\frac{P_4(i,j,k)}{\Delta x}{H_y^{n}}_{(i+1,j,k)}-\frac{P_4(i,j,k)}{\Delta x}{H_y^{n}}_{(i,j,k)}\\
&-\frac{P_4(i,j,k)}{\Delta y}{H_x^{n}}_{(i,j+1,k)}+\frac{P_4(i,j,k)}{\Delta y}{H_x^{n}}_{(i,j,k)}\\
&-P_6(i,j,k){E_z^{n-1}}_{(i,j,k)}-\left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)+P_5(i,j,k)\right]{E_z^{n}}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta x^2}{E_z^{n}}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x^2}{E_z^{n}}_{(i-1,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta y^2}{E_z^{n}}_{(i,j+1,k)}+\frac{P_1(i,j,k)}{\Delta y^2}{E_z^{n}}_{(i,j-1,k)}\\
&-\frac{P_1(i,j,k)}{\Delta x\Delta z}{E_x^{n}}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta z}{E_x^{n}}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta x\Delta z}{E_x^{n}}_{(i+1,j,k-1)}-\frac{P_1(i,j,k)}{\Delta x\Delta z}{E_x^{n}}_{(i,j,k-1)}\\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n}}_{(i,j+1,k)}+\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n}}_{(i,j,k)}\\
&+\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n}}_{(i,j+1,k-1)}-\frac{P_1(i,j,k)}{\Delta y\Delta z}{E_y^{n}}_{(i,j,k-1)}
\end{aligned}$$

(3.32), (3.40) and (3.42) are the three main equations of the FD–CN–FDTD method. These equations are valid for the interior computational space but not on the boundaries. (3.32) is valid for $i_{min}+1 < i \leq i_{max}$, $j_{min}+1 < j \leq j_{max}-1$ and $k_{min}+1 < k \leq k_{max}-1$. Henceforth, i_{min}, j_{min}, k_{min} are the lower boundaries and i_{max}, j_{max}, k_{max} are the upper boundaries in the x, y, z directions, respectively. (3.40) is valid for $i_{min}+1 < i \leq i_{max}-1$, $j_{min}+1 < j \leq j_{max}$ and $k_{min}+1 < k \leq k_{max}-1$. (3.42) is valid for $i_{min}+1 < i \leq i_{max}-1$, $j_{min}+1 < j \leq j_{max}-1$ and $k_{min}+1 < k \leq k_{max}$. To calculate the values on the boundaries suitable absorbing boundary conditions have to be added in the FD–CN–FDTD method.

3.3 Inclusion of Mur's First-Order Boundary Condition

In the FD–CN–FDTD method Mur's first-order absorbing boundary condition [28] (equations in Appendix B) is used. (3.32) does not hold when

- $i = i_{min}+1$
- $j = j_{min}+1$
- $k = k_{min}+1$
- $i = i_{min}+1$ and $j = j_{min}+1$
- $j = j_{min}+1$ and $k = k_{min}+1$
- $i = i_{min}+1$ and $k = k_{min}+1$
- $i = i_{min}+1$, $j = j_{min}+1$ and $k = k_{min}+1$
- $j = j_{max}$
- $k = k_{max}$

When $i = i_{min}+1$, (3.32) is to be modified using the equations of Mur's ABC. The modified equation is (3.43) which has been derived using (B.23) and (B.27)

$$\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)\right]E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j+1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j-1,k)} - \frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{3(i,j,k)}-1)E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{1(i,j,k)}-1)E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}h_{4(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}h_{2(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{7(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{5(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}h_{8(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}h_{6(i,j,k)} \\
&= P_2(i,j,k)D_x^n{}_{(i,j,k)} + P_3(i,j,k)D_x^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j+1,k)} - \frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)+P_5(i,j,k)\right]E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j-1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k)}
\end{aligned} \tag{3.43}$$

where

$$h_{1(i,j,k)} = \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}})}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.44}$$

$$h_{2(i,j,k)} = \frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}})\, E^n_{y\,(i+1,j,k)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{y\,(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.45}$$

$$h_{3(i,j,k)} = h_1(i, j+1, k) \tag{3.46}$$

$$h_{4(i,j,k)} = h_2(i, j+1, k) \tag{3.47}$$

$$h_{5(i,j,k)} = \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}})}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.48}$$

$$h_{6(i,j,k)} = \frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}})\, E^n_{z\,(i+1,j,k)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{z\,(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.49}$$

$$h_{7(i,j,k)} = h_5(i, j, k+1) \tag{3.50}$$

$$h_{8(i,j,k)} = h_6(i, j, k+1) \tag{3.51}$$

Similarly, for the case of $j = j_{min}+1$, using the Mur's ABC equation of (B.19), (3.32) is modified to (3.52)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)\right]E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j+1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2}h_{9(i,j,k)}E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}h_{10(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k-1)} \\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i-1,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}E_y^{n+1}{}_{(i-1,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i-1,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^{n+1}{}_{(i-1,j,k)} \\
&= P_2(i,j,k)D_x^n{}_{(i,j,k)} + P_3(i,j,k)D_x^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j+1,k)} - \frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k)} \\
-P_6(i,j,k)E_x^{n-1}{}_{(i,j,k)} &- \left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)+P_5(i,j,k)\right]E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j-1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k)}
\end{aligned}
\tag{3.52}
$$

where

$$h_{9(i,j,k)} = \frac{(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}})}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.53}$$

$$h_{10(i,j,k)} = \frac{(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}})\, E^n_{x\,(i,j+1,k)} - (\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{x\,(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.54}$$

In the same way, for the case of $k = k_{min} + 1$, using the Mur's ABC equation of (B.21), (3.32) is modified to (3.55)

$$\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\big)\right] E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} E_x^{n+1}{}_{(i,j+1,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta y^2} E_x^{n+1}{}_{(i,j-1,k)} - \frac{P_1(i,j,k)}{\Delta z^2} E_x^{n+1}{}_{(i,j,k+1)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta z^2} h_{11(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{12(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i-1,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i-1,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i,j,k+1)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i-1,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i-1,j,k)} \\
&\quad = P_2(i,j,k) D_x^n{}_{(i,j,k)} + P_3(i,j,k) D_x^{n-1}{}_{(i,j,k)} + \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j+1,k)} \\
&\quad - \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j,k)} - \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\big) + P_5(i,j,k)\right] E_x^n{}_{(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j-1,k)} + \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k+1)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k+1)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k)}
\end{aligned} \tag{3.55}$$

where

$$h_{11(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}})}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.56}$$

$$h_{12(i,j,k)} = \frac{(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}})\, E^n_{x\,(i,j,k+1)} - (\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{x\,(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.57}$$

For the case of $i = i_{min} + 1$ and $j = j_{min} + 1$, using the Mur's ABC equations of (B.23), (B.27) and (B.19), (3.32) is modified to (3.58)

$$\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\big)\right] E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} E_x^{n+1}{}_{(i,j+1,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y^2} h_{9(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} E_x^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2} E_x^{n+1}{}_{(i,j,k-1)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{3(i,j,k)} - 1) E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{1(i,j,k)} - 1) E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y^2} h_{10(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{4(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{2(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{7(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{5(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{8(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{6(i,j,k)} \\
&= P_2(i,j,k) D_x^n{}_{(i,j,k)} + P_3(i,j,k) D_x^{n-1}{}_{(i,j,k)} + \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j+1,k)} \\
&- \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j,k)} - \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\big) + P_5(i,j,k)\right] E_x^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j-1,k)} + \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k+1)} \\
&+ \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k+1)} \\
&+ \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k)}
\end{aligned} \tag{3.58}$$

For the case of $j = j_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.19) and (B.21), (3.32) is modified to (3.59)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\Big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\Big)\right] E_x^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2} E_x^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{9(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{10(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z^2} E_x^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{11(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{12(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i-1,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_y^{n+1}{}_{(i-1,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i-1,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^{n+1}{}_{(i-1,j,k)} \\
&= P_2(i,j,k) D_x^n{}_{(i,j,k)} + P_3(i,j,k) D_x^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j+1,k)} - \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k)} \\
&-P_6(i,j,k) E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\Big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\Big) + P_5(i,j,k)\right] E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j-1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k)}
\end{aligned}
\tag{3.59}
$$

For the case of $i = i_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.23), (B.27) and (B.21), (3.32) is modified to (3.60)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)\right]E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j+1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y^2}E_x^{n+1}{}_{(i,j-1,k)} - \frac{P_1(i,j,k)}{\Delta z^2}E_x^{n+1}{}_{(i,j,k+1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z^2}h_{11(i,j,k)}E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2}h_{12(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{3(i,j,k)}-1)E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{1(i,j,k)}-1)E_y^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta y}h_{4(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}h_{2(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{7(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{5(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta x}h_{8(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}h_{6(i,j,k)} \\
&\quad = P_2(i,j,k)D_x^n{}_{(i,j,k)} + P_3(i,j,k)D_x^{n-1}{}_{(i,j,k)} \\
&\quad +\frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j+1,k)} - \frac{P_4(i,j,k)}{\Delta y}H_z^n{}_{(i,j,k)} \\
&\quad -\frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z}H_y^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta y^2}+\frac{1}{\Delta z^2}\big)+P_5(i,j,k)\right]E_x^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_x^n{}_{(i,j-1,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2}E_x^n{}_{(i,j,k-1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y\Delta x}E_y^n{}_{(i-1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta x}E_z^n{}_{(i-1,j,k)}
\end{aligned}
\tag{3.60}
$$

For the case of $i = i_{min} + 1$, $j = j_{min} + 1$ and $k = k_{min} + 1$, using the Mur's ABC equations of (B.23), (B.27), (B.19) and (B.21), (3.32) is modified to (3.61)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\Big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\Big)\right] E_x^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y^2} E_x^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{9(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{10(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z^2} E_x^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{11(i,j,k)} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{12(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{3(i,j,k)} - 1) E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{1(i,j,k)} - 1) E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{4(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{2(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{7(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{5(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{8(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{6(i,j,k)} \\
&= P_2(i,j,k) D_x^n{}_{(i,j,k)} + P_3(i,j,k) D_x^{n-1}{}_{(i,j,k)} \\
&+ \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j+1,k)} - \frac{P_4(i,j,k)}{\Delta y} H_z^n{}_{(i,j,k)} \\
&- \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k+1)} + \frac{P_4(i,j,k)}{\Delta z} H_y^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_x^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\Big(\frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\Big) + P_5(i,j,k)\right] E_x^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_x^n{}_{(i,j-1,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_x^n{}_{(i,j,k-1)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta x} E_y^n{}_{(i-1,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_z^n{}_{(i-1,j,k)}
\end{aligned}
\tag{3.61}
$$

For the case of $j = j_{max}$, using the Mur's ABC equation of (B.20), (3.32) is modified to (3.62)

$$E_x^{n+1}{}_{(i,j,k)} - \frac{\left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j-1,k)}}\right) E_x^{n+1}{}_{(i,j-1,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.62}$$
$$= \frac{\left(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j-1,k)}}\right) E_x^n{}_{(i,j-1,k)} - \left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_x^n{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}}$$

For the case of $k = k_{max}$, using the Mur's ABC equation of (B.22), (3.32) is modified to (3.63)

$$E_x^{n+1}{}_{(i,j,k)} - \frac{\left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k-1)}}\right) E_x^{n+1}{}_{(i,j,k-1)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.63}$$
$$= \frac{\left(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k-1)}}\right) E_x^n{}_{(i,j,k-1)} - \left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_x^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}}$$

Again, (3.40) does not hold when

- $i = i_{min} + 1$
- $j = j_{min} + 1$
- $k = k_{min} + 1$
- $i = i_{min} + 1$ and $j = j_{min} + 1$
- $j = j_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{min} + 1$, $j = j_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{max}$
- $k = k_{max}$

When $i = i_{min}+1$, (3.40) is to be modified using the equations of Mur's ABC. The modified equation is (3.64) which has been derived using (B.23)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)\right]E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k+1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i+1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x^2}h_1{}_{(i,j,k)}E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}h_2{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k+1)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j-1,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j-1,k)} \\
&\quad = P_2(i,j,k)D_y^n{}_{(i,j,k)} + P_3(i,j,k)D_y^{n-1}{}_{(i,j,k)} \\
&\quad +\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k+1)} - \frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k)} \\
&\quad -\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)+P_5(i,j,k)\right]E_y^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k-1)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i-1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.64}
$$

Similarly, for the case of $j = j_{min} + 1$, using the Mur's ABC equations of (B.29) and (B.19), (3.40) is modified to (3.65)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big)\right] E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k+1)} \\
&- \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i-1,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{23(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{21(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k)} \\
&\qquad - \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{24(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{22(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{25(i,j,k)} - 1) E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{9(i,j,k)} - 1) E_x^{n+1}{}_{(i,j,k)} \\
&\qquad - \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{26(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{10(i,j,k)} \\
&\qquad = P_2(i,j,k) D_y^n{}_{(i,j,k)} + P_3(i,j,k) D_y^{n-1}{}_{(i,j,k)} \\
&\qquad + \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k+1)} - \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k)} \\
&\qquad - \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big) + P_5(i,j,k)\right] E_y^n{}_{(i,j,k)} \\
&\qquad + \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k-1)} \\
&\qquad + \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i-1,j,k)} \\
&\qquad - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k)} \\
&\qquad + \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k)} \\
&\qquad - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j,k)} \\
&\qquad + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.65}
$$

where

$$h_{21(i,j,k)} = \frac{(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}})}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.66}$$

$$h_{22(i,j,k)} = \frac{(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}})\, E^n_{z\,(i,j+1,k)} - (\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{z\,(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.67}$$

$$h_{23(i,j,k)} = h_{21}(i,j,k+1) \tag{3.68}$$

$$h_{24(i,j,k)} = h_{22}(i,j,k+1) \tag{3.69}$$

$$h_{25(i,j,k)} = h_{9}(i+1,j,k) \tag{3.70}$$

$$h_{26(i,j,k)} = h_{10}(i+1,j,k) \tag{3.71}$$

In the same way, for the case of $k = k_{min} + 1$, using the Mur's ABC equation of (B.25), (3.40) is modified to (3.72)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\left(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\right)\right] E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{27(i,j,k)} E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{28(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i-1,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y \Delta z} E_z^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_z^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z} E_z^{n+1}{}_{(i,j-1,k+1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_z^{n+1}{}_{(i,j-1,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^{n+1}{}_{(i+1,j-1,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^{n+1}{}_{(i,j-1,k)} \\
&= P_2(i,j,k) D_y^n{}_{(i,j,k)} + P_3(i,j,k) D_y^{n-1}{}_{(i,j,k)} \\
&+ \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k+1)} - \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k)} \\
&- \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\left(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\right) + P_5(i,j,k)\right] E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k-1)} \\
&+ \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i-1,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.72}
$$

where

$$h_{27(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}})}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.73}$$

$$h_{28(i,j,k)} = \frac{(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}})\, E^n_{y\,(i,j,k+1)} - (\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}})\, E^n_{y\,(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.74}$$

For the case of $i = i_{min} + 1$ and $j = j_{min} + 1$, using the Mur's ABC equations of (B.23), (B.29) and (B.19), (3.40) is modified to (3.75)

$$\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big)\right] E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k+1)} \\
&- \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{2(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{23(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{21(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x^2} h_{1(i,j,k)} E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{24(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{22(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{25(i,j,k)} - 1) E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y}(h_{9(i,j,k)} - 1) E_x^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{26(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{10(i,j,k)} \\
&= P_2(i,j,k) D_y^n{}_{(i,j,k)} + P_3(i,j,k) D_y^{n-1}{}_{(i,j,k)} + \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k+1)} \\
&- \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k)} - \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big) + P_5(i,j,k)\right] E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k-1)} + \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i+1,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i-1,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j-1,k)}
\end{aligned} \tag{3.75}$$

For the case of $j = j_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.29), (B.19) and (B.25), (3.40) is modified to (3.76)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)\right]E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k+1)} \\
&-\frac{P_1(i,j,k)}{\Delta z^2}h_{27(i,j,k)}E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2}h_{28(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{23(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{21(i,j,k)}-1)E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}h_{24(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}h_{22(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{25(i,j,k)}-1)E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}(h_{9(i,j,k)}-1)E_x^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}h_{26(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}h_{10(i,j,k)} \\
&= P_2(i,j,k)D_y^n{}_{(i,j,k)} + P_3(i,j,k)D_y^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k+1)} - \frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)+P_5(i,j,k)\right]E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k-1)} \\
&+\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.76}
$$

For the case of $i = i_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.23) and (B.25), (3.40) is modified to (3.77)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)\right]E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z^2}E_y^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta z^2}h_{27(i,j,k)}E_y^{n+1}{}_{(i,j,k)}-\frac{P_1(i,j,k)}{\Delta z^2}h_{28(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x^2}E_y^{n+1}{}_{(i+1,j,k)}-\frac{P_1(i,j,k)}{\Delta x^2}h_{1(i,j,k)}E_y^{n+1}{}_{(i,j,k)}-\frac{P_1(i,j,k)}{\Delta x^2}h_{2(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k+1)}-\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k+1)}+\frac{P_1(i,j,k)}{\Delta y\Delta z}E_z^{n+1}{}_{(i,j-1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j,k)}-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i+1,j-1,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^{n+1}{}_{(i,j-1,k)} \\
&=P_2(i,j,k)D_y^n{}_{(i,j,k)}+P_3(i,j,k)D_y^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k+1)}-\frac{P_4(i,j,k)}{\Delta z}H_x^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i+1,j,k)}+\frac{P_4(i,j,k)}{\Delta x}H_z^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_y^{n-1}{}_{(i,j,k)}-\left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2}+\frac{1}{\Delta x^2}\big)+P_5(i,j,k)\right]E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z^2}E_y^n{}_{(i,j,k-1)} \\
&+\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x^2}E_y^n{}_{(i-1,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k+1)}+\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k+1)}-\frac{P_1(i,j,k)}{\Delta z\Delta y}E_z^n{}_{(i,j-1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j,k)}+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i+1,j-1,k)}-\frac{P_1(i,j,k)}{\Delta x\Delta y}E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.77}
$$

For the case of $i = i_{min} + 1$, $j = j_{min} + 1$ and $k = k_{min} + 1$, using the Mur's ABC equations of (B.23), (B.29), (B.19) and (B.25), (3.40) is modified to (3.78)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big)\right] E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z^2} E_y^{n+1}{}_{(i,j,k+1)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{27(i,j,k)} E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z^2} h_{28(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x^2} E_y^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{1(i,j,k)} E_y^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{2(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z} (h_{23(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} (h_{21(i,j,k)} - 1) E_z^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{24(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{22(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} (h_{25(i,j,k)} - 1) E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} (h_{9(i,j,k)} - 1) E_x^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{26(i,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} h_{10(i,j,k)} \\
&= P_2(i,j,k) D_y^n{}_{(i,j,k)} + P_3(i,j,k) D_y^{n-1}{}_{(i,j,k)} \\
&+ \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k+1)} - \frac{P_4(i,j,k)}{\Delta z} H_x^n{}_{(i,j,k)} \\
&- \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i+1,j,k)} + \frac{P_4(i,j,k)}{\Delta x} H_z^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_y^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}\big) + P_5(i,j,k)\right] E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z^2} E_y^n{}_{(i,j,k-1)} \\
&+ \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_y^n{}_{(i-1,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k+1)} + \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k+1)} - \frac{P_1(i,j,k)}{\Delta z \Delta y} E_z^n{}_{(i,j-1,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i+1,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta y} E_x^n{}_{(i,j-1,k)}
\end{aligned}
\tag{3.78}
$$

For the case of $i = i_{max}$, using the Mur's ABC equation of (B.24), (3.40) is modified to (3.79)

$$E_y^{n+1}{}_{(i,j,k)} - \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i-1,j,k)}})\, E_y^{n+1}{}_{(i-1,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.79}$$
$$= \frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i-1,j,k)}})\, E_y^{n}{}_{(i-1,j,k)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}})\, E_y^{n}{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}}$$

For the case of $k = k_{max}$, using the Mur's ABC equation of (B.26), (3.40) is modified to (3.80)

$$E_y^{n+1}{}_{(i,j,k)} - \frac{(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k-1)}})\, E_y^{n+1}{}_{(i,j,k-1)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.80}$$
$$= \frac{(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k-1)}})\, E_y^{n}{}_{(i,j,k-1)} - (\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}})\, E_y^{n}{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}}$$

Finally, (3.42) does not hold when

- $i = i_{min} + 1$
- $j = j_{min} + 1$
- $k = k_{min} + 1$
- $i = i_{min} + 1$ and $j = j_{min} + 1$
- $j = j_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{min} + 1$, $j = j_{min} + 1$ and $k = k_{min} + 1$
- $i = i_{max}$
- $j = j_{max}$

When $i = i_{min}+1$, (3.42) is to be modified using the equations of Mur's ABC. The modified equation is (3.81) which has been derived using (B.27)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)\right]E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}E_z^{n+1}{}_{(i+1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x^2}h_5{}_{(i,j,k)}E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}h_6{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y^2}E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_z^{n+1}{}_{(i,j-1,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta z\Delta x}E_x^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta z\Delta x}E_x^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta x}E_x^{n+1}{}_{(i+1,j,k-1)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}E_x^{n+1}{}_{(i,j,k-1)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^{n+1}{}_{(i,j+1,k-1)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^{n+1}{}_{(i,j,k-1)} \\
&\quad = P_2(i,j,k)D_z^n{}_{(i,j,k)} + P_3(i,j,k)D_z^{n-1}{}_{(i,j,k)} \\
&\quad +\frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i,j,k)} \\
&\quad -\frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)+P_5(i,j,k)\right]E_z^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i-1,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j-1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k-1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k-1)}
\end{aligned}
\tag{3.81}
$$

Similarly, for the case of $j = j_{min} + 1$, using the Mur's ABC equations of (B.29), (3.42) is modified to (3.82)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big)\right] E_z^{n+1}{}_{(i,j,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i-1,j,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta y^2} E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{21(i,j,k)} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{22(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i,j,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i+1,j,k-1)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i,j,k-1)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j+1,k-1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j,k-1)} \\
&\quad = P_2(i,j,k) D_z^n{}_{(i,j,k)} + P_3(i,j,k) D_z^{n-1}{}_{(i,j,k)} \\
&\quad + \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i,j,k)} \\
&\quad - \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big) + P_5(i,j,k)\right] E_z^n{}_{(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i-1,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j-1,k)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k-1)} \\
&\quad - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k)} \\
&\quad + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k-1)}
\end{aligned}
\tag{3.82}
$$

In the same way, for the case of $k = k_{min} + 1$, using the Mur's ABC equations of (B.21) and (B.25), (3.42) is modified to (3.83)

$$\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big)\right] E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i+1,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i-1,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} E_z^{n+1}{}_{(i,j-1,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{33(i,j,k)} - 1) E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{11(i,j,k)} - 1) E_x^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{34(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{12(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{35(i,j,k)} - 1) E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{27(i,j,k)} - 1) E_y^{n+1}{}_{(i,j,k)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{36(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{28(i,j,k)} \\
&= P_2(i,j,k) D_z^n{}_{(i,j,k)} + P_3(i,j,k) D_z^{n-1}{}_{(i,j,k)} \\
&+ \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i,j,k)} \\
&- \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j,k)} \\
&- P_6(i,j,k) E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big) + P_5(i,j,k)\right] E_z^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i-1,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j-1,k)} \\
&- \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k-1)} \\
&- \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k)} \\
&+ \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k-1)}
\end{aligned} \tag{3.83}$$

where

$$h_{33(i,j,k)} = h_{11}(i+1,j,k) \tag{3.84}$$

$$h_{34(i,j,k)} = h_{12}(i+1,j,k) \tag{3.85}$$

$$h_{35(i,j,k)} = h_{27}(i, j+1, k) \tag{3.86}$$

$$h_{36(i,j,k)} = h_{28}(i, j+1, k) \tag{3.87}$$

For the case of $i = i_{min} + 1$ and $j = j_{min} + 1$, using the Mur's ABC equations of (B.27) and (B.29), (3.42) is modified to (3.88)

$$\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big)\right] E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{5(i,j,k)} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{6(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2} E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{21(i,j,k)} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{22(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i+1,j,k-1)} \\
&+\frac{P_1(i,j,k)}{\Delta z \Delta x} E_x^{n+1}{}_{(i,j,k-1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j+1,k-1)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^{n+1}{}_{(i,j,k-1)} \\
&= P_2(i,j,k) D_z^n{}_{(i,j,k)} + P_3(i,j,k) D_z^{n-1}{}_{(i,j,k)} + \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i+1,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i,j,k)} - \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j,k)} \\
&-P_6(i,j,k) E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big) + P_5(i,j,k)\right] E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i-1,j,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j+1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j-1,k)} - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k-1)}
\end{aligned} \tag{3.88}$$

For the case of $j = j_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.29), (B.21) and (B.25), (3.42) is modified to (3.89)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)\right]E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}E_z^{n+1}{}_{(i+1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x^2}E_z^{n+1}{}_{(i-1,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_z^{n+1}{}_{(i,j+1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y^2}h_{21(i,j,k)}E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2}h_{22(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{33(i,j,k)}-1)E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{11(i,j,k)}-1)E_x^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta x}h_{34(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}h_{12(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{35(i,j,k)}-1)E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{27(i,j,k)}-1)E_y^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}h_{36(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}h_{28(i,j,k)} \\
&\quad = P_2(i,j,k)D_z^n{}_{(i,j,k)} + P_3(i,j,k)D_z^{n-1}{}_{(i,j,k)} \\
&\quad +\frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i,j,k)} \\
&\quad -\frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)+P_5(i,j,k)\right]E_z^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i-1,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j-1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k-1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k-1)}
\end{aligned}
\tag{3.89}
$$

For the case of $i = i_{min}+1$ and $k = k_{min}+1$, using the Mur's ABC equations of (B.27), (B.21) and (B.25), (3.42) is modified to (3.90)

$$
\begin{aligned}
&\left[1+2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)\right]E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}E_z^{n+1}{}_{(i+1,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x^2}h_{5(i,j,k)}E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2}h_{6(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y^2}E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2}E_z^{n+1}{}_{(i,j-1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{33(i,j,k)}-1)E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}(h_{11(i,j,k)}-1)E_x^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta z\Delta x}h_{34(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z\Delta x}h_{12(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{35(i,j,k)}-1)E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}(h_{27(i,j,k)}-1)E_y^{n+1}{}_{(i,j,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}h_{36(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}h_{28(i,j,k)} \\
&\quad = P_2(i,j,k)D_z^n{}_{(i,j,k)} + P_3(i,j,k)D_z^{n-1}{}_{(i,j,k)} \\
&\quad +\frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x}H_y^n{}_{(i,j,k)} \\
&\quad -\frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y}H_x^n{}_{(i,j,k)} \\
&-P_6(i,j,k)E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\big)+P_5(i,j,k)\right]E_z^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2}E_z^n{}_{(i-1,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2}E_z^n{}_{(i,j-1,k)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x\Delta z}E_x^n{}_{(i,j,k-1)} \\
&\quad -\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k)} \\
&\quad +\frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y\Delta z}E_y^n{}_{(i,j,k-1)}
\end{aligned}
\tag{3.90}
$$

For the case of $i = i_{min} + 1$, $j = j_{min} + 1$ and $k = k_{min} + 1$, using the Mur's ABC equations of (B.27), (B.29), (B.21) and (B.25), (3.42) is modified to (3.91)

$$
\begin{aligned}
&\left[1 + 2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big)\right] E_z^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x^2} E_z^{n+1}{}_{(i+1,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{5(i,j,k)} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta x^2} h_{6(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y^2} E_z^{n+1}{}_{(i,j+1,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{21(i,j,k)} E_z^{n+1}{}_{(i,j,k)} - \frac{P_1(i,j,k)}{\Delta y^2} h_{22(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{33(i,j,k)} - 1) E_x^{n+1}{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x}(h_{11(i,j,k)} - 1) E_x^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta z \Delta x} h_{34(i,j,k)} + \frac{P_1(i,j,k)}{\Delta z \Delta x} h_{12(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{35(i,j,k)} - 1) E_y^{n+1}{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z}(h_{27(i,j,k)} - 1) E_y^{n+1}{}_{(i,j,k)} \\
&-\frac{P_1(i,j,k)}{\Delta y \Delta z} h_{36(i,j,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} h_{28(i,j,k)} \\
&= P_2(i,j,k) D_z^n{}_{(i,j,k)} + P_3(i,j,k) D_z^{n-1}{}_{(i,j,k)} \\
&+\frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i+1,j,k)} - \frac{P_4(i,j,k)}{\Delta x} H_y^n{}_{(i,j,k)} \\
&-\frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j+1,k)} + \frac{P_4(i,j,k)}{\Delta y} H_x^n{}_{(i,j,k)} \\
&-P_6(i,j,k) E_z^{n-1}{}_{(i,j,k)} - \left[2P_1(i,j,k)\big(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\big) + P_5(i,j,k)\right] E_z^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x^2} E_z^n{}_{(i-1,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y^2} E_z^n{}_{(i,j-1,k)} \\
&-\frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k)} + \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i+1,j,k-1)} - \frac{P_1(i,j,k)}{\Delta x \Delta z} E_x^n{}_{(i,j,k-1)} \\
&-\frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k)} + \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k)} \\
&+\frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j+1,k-1)} - \frac{P_1(i,j,k)}{\Delta y \Delta z} E_y^n{}_{(i,j,k-1)}
\end{aligned}
\tag{3.91}
$$

For the case of $i = i_{max}$, using the Mur's ABC equation of (B.28), (3.42) is modified to (3.92)

$$E_z^{n+1}{}_{(i,j,k)} - \frac{\left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i-1,j,k)}}\right) E_z^{n+1}{}_{(i-1,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.92}$$
$$= \frac{\left(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i-1,j,k)}}\right) E_z^{n}{}_{(i-1,j,k)} - \left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_z^{n}{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}}$$

For the case of $j = j_{max}$, using the Mur's ABC equation of (B.30), (3.42) is modified to (3.93)

$$E_z^{n+1}{}_{(i,j,k)} - \frac{\left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j-1,k)}}\right) E_z^{n+1}{}_{(i,j-1,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.93}$$
$$= \frac{\left(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j-1,k)}}\right) E_z^{n}{}_{(i,j-1,k)} - \left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_z^{n}{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}}$$

3.4 Calculation of Electric Fields

Using the equations derived so far, values of the electric fields are calculated. These equations can be summed up as:

- $E_x^{n+1}{}_{(i_{min}+1\leq i\leq i_{max}, j_{min}+1\leq j\leq j_{max}, k_{min}+1\leq k\leq k_{max})}$ corresponds to (3.32) and those mentioned in Table 3.1
- $E_y^{n+1}{}_{(i_{min}+1\leq i\leq i_{max}, j_{min}+1\leq j\leq j_{max}, k_{min}+1\leq k\leq k_{max})}$ corresponds to (3.40) and those mentioned in Table 3.2
- $E_z^{n+1}{}_{(i_{min}+1\leq i\leq i_{max}, j_{min}+1\leq j\leq j_{max}, k_{min}+1\leq k\leq k_{max})}$ corresponds to (3.42) and those mentioned in Table 3.3

By applying these equations to each FDTD grid position, a system of linear equations of $\boldsymbol{Au} = \boldsymbol{c}$ is set up. Here, $\boldsymbol{A}$ is the coefficient matrix, $\boldsymbol{u}$ represents a vector with the electric field components to be solved, and $\boldsymbol{c}$ is the excitation vector. It should be mentioned that the equations for the points i_{min}, j_{min}, k_{min} are excluded in order to form the matrix properly. When the three sets of

x direction	y direction	z direction	(3.32) replaced by
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max} - 1$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.43)
$i_{min} + 1 < i \leq i_{max}$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.52)
$i_{min} + 1 < i \leq i_{max}$	$j_{min} + 1 < j \leq j_{max} - 1$	$k = k_{min} + 1$	(3.55)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.58)
$i_{min} + 1 < i \leq i_{max}$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.59)
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max} - 1$	$k = k_{min} + 1$	(3.60)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.61)
$i_{min} + 1 \leq i \leq i_{max}$	$j = j_{max}$	$k_{min} + 1 \leq k \leq k_{max}$	(3.62)
$i_{min} + 1 \leq i \leq i_{max}$	$j_{min} + 1 \leq j \leq j_{max} - 1$	$k = k_{max}$	(3.63)

Table 3.1: The boundary equations that replace (3.32) and the corresponding scanning ranges in x, y and z directions.

x direction	y direction	z direction	(3.40) replaced by
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max}$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.64)
$i_{min} + 1 < i \leq i_{max} - 1$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.65)
$i_{min} + 1 < i \leq i_{max} - 1$	$j_{min} + 1 < j \leq j_{max}$	$k = k_{min} + 1$	(3.72)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max} - 1$	(3.75)
$i_{min} + 1 < i \leq i_{max} - 1$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.76)
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max}$	$k = k_{min} + 1$	(3.77)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.78)
$i = i_{max}$	$j_{min} + 1 \leq j \leq j_{max}$	$k_{min} + 1 \leq k \leq k_{max}$	(3.79)
$i_{min} + 1 \leq i \leq i_{max} - 1$	$j_{min} + 1 \leq j \leq j_{max}$	$k = k_{max}$	(3.80)

Table 3.2: The boundary equations that replace (3.40) and the corresponding scanning ranges in x, y and z directions.

equations are considered together and applied on all the grid points this exclusion is required to make the system uniform. Therefore, the electric field values at i_{min}, j_{min}, k_{min} are calculated separately using the Mur's ABC equations as described in Section 3.4.1. The system of equations $\boldsymbol{Au} = \boldsymbol{c}$ is solved to find the electric field values in the scanning range of ($i_{min} + 1 \leq i \leq i_{max}$, $j_{min} + 1 \leq j \leq j_{max}$, $k_{min} + 1 \leq k \leq k_{max}$).

x direction	y direction	z direction	(3.42) replaced by
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max} - 1$	$k_{min} + 1 < k \leq k_{max}$	(3.81)
$i_{min} + 1 < i \leq i_{max} - 1$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max}$	(3.82)
$i_{min} + 1 < i \leq i_{max} - 1$	$j_{min} + 1 < j \leq j_{max} - 1$	$k = k_{min} + 1$	(3.83)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k_{min} + 1 < k \leq k_{max}$	(3.88)
$i_{min} + 1 < i \leq i_{max} - 1$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.89)
$i = i_{min} + 1$	$j_{min} + 1 < j \leq j_{max} - 1$	$k = k_{min} + 1$	(3.90)
$i = i_{min} + 1$	$j = j_{min} + 1$	$k = k_{min} + 1$	(3.91)
$i = i_{max}$	$j_{min} + 1 \leq j \leq j_{max}$	$k_{min} + 1 \leq k \leq k_{max}$	(3.92)
$i_{min} + 1 \leq i \leq i_{max} - 1$	$j = j_{max}$	$k_{min} + 1 \leq k \leq k_{max}$	(3.93)

Table 3.3: The boundary equations that replace (3.42) and the corresponding scanning ranges in x, y and z directions.

3.4.1 Electric Fields at $\mathbf{i = i_{min}}$, $\mathbf{j = j_{min}}$, $\mathbf{k = k_{min}}$

Solution of the system of equations $\boldsymbol{Au} = \boldsymbol{c}$ gives the electric field values in the scanning range of $i_{min} + 1 \leq i \leq i_{max}$, $j_{min} + 1 \leq j \leq j_{max}$, $k_{min} + 1 \leq k \leq k_{max}$. Values of the electric fields for the rest of the boundary locations are found by using the following equations which come from Mur's first-order boundary condition.

$$E_x^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}}\right) E_x^{n+1}{}_{(i,j+1,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} + \frac{\left(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}}\right) E_x^{n}{}_{(i,j+1,k)} - \left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_x^{n}{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \quad [i_{min} + 1 \leq i \leq i_{max}, j = j_{min}, k_{min} \leq k \leq k_{max}]. \tag{3.94}$$

$$E_x^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}}\right) E_x^{n+1}{}_{(i,j,k+1)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} + \frac{\left(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}}\right) E_x^{n}{}_{(i,j,k+1)} - \left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_x^{n}{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \quad [i_{min} + 1 \leq i \leq i_{max}, j_{min} \leq j \leq j_{max}, k = k_{min}]. \tag{3.95}$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}}\right) E_y^{n+1}{}_{(i+1,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.96}$$
$$+\frac{\left(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}}\right) E_y^n{}_{(i+1,j,k)} - \left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_y^n{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}}$$
$$[i = i_{min}, j_{min} + 1 \leq j \leq j_{max}, k_{min} \leq k \leq k_{max}].$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}}\right) E_y^{n+1}{}_{(i,j,k+1)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.97}$$
$$+\frac{\left(\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k+1)}}\right) E_y^n{}_{(i,j,k+1)} - \left(\Delta t - \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_y^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu\epsilon_{(i,j,k)}}}$$
$$[i_{min} \leq i \leq i_{max}, j_{min} + 1 \leq j \leq j_{max}, k = k_{min}].$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}}\right) E_z^{n+1}{}_{(i+1,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.98}$$
$$+\frac{\left(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j,k)}}\right) E_z^n{}_{(i+1,j,k)} - \left(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_z^n{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j,k)}}}$$
$$[i = i_{min}, j_{min} \leq j \leq j_{max}, k_{min} + 1 \leq k \leq k_{max}].$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}}\right) E_z^{n+1}{}_{(i,j+1,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}} \tag{3.99}$$
$$+\frac{\left(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1,k)}}\right) E_z^n{}_{(i,j+1,k)} - \left(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}\right) E_z^n{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j,k)}}}$$
$$[i_{min} \leq i \leq i_{max}, j = j_{min}, k_{min} + 1 \leq k \leq k_{max}].$$

3.5 Calculation of Magnetic Fields and Electric Flux Densities

Once electric fields are calculated magnetic fields are calculated by the discretized form of (3.25), (3.26), (3.27) as follows.

$$H_x^{n+1}{}_{(i,j,k)} = H_x^n{}_{(i,j,k)} \tag{3.100}$$
$$+\frac{\Delta t}{2\mu}\left[\frac{1}{\Delta z}\left(E_y^{n+1}{}_{(i,j,k)} - E_y^{n+1}{}_{(i,j,k-1)}\right) - \frac{1}{\Delta y}\left(E_z^{n+1}{}_{(i,j,k)} - E_z^{n+1}{}_{(i,j-1,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta z}\left(E_y^n{}_{(i,j,k)} - E_y^n{}_{(i,j,k-1)}\right) - \frac{1}{\Delta y}\left(E_z^n{}_{(i,j,k)} - E_z^n{}_{(i,j-1,k)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}-1, j_{min}+1 \le j \le j_{max}, k_{min}+1 \le k \le k_{max}]$$

$$H_y^{n+1}{}_{(i,j,k)} = H_y^n{}_{(i,j,k)} \tag{3.101}$$
$$+\frac{\Delta t}{2\mu}\left[\frac{1}{\Delta x}\left(E_z^{n+1}{}_{(i,j,k)} - E_z^{n+1}{}_{(i-1,j,k)}\right) - \frac{1}{\Delta z}\left(E_x^{n+1}{}_{(i,j,k)} - E_x^{n+1}{}_{(i,j,k-1)}\right)\right.$$
$$\left.+\frac{1}{\Delta x}\left(E_z^n{}_{(i,j,k)} - E_z^n{}_{(i-1,j,k)}\right) - \frac{1}{\Delta z}\left(E_x^n{}_{(i,j,k)} - E_x^n{}_{(i,j,k-1)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}-1, j_{min}+1 \le j \le j_{max}, k_{min}+1 \le k \le k_{max}]$$

$$H_z^{n+1}{}_{(i,j,k)} = H_z^n{}_{(i,j,k)} \tag{3.102}$$
$$+\frac{\Delta t}{2\mu}\left[\frac{1}{\Delta y}\left(E_x^{n+1}{}_{(i,j,k)} - E_x^{n+1}{}_{(i,j-1,k)}\right) - \frac{1}{\Delta x}\left(E_y^{n+1}{}_{(i,j,k)} - E_y^{n+1}{}_{(i-1,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta y}\left(E_x^n{}_{(i,j,k)} - E_x^n{}_{(i,j-1,k)}\right) - \frac{1}{\Delta x}\left(E_y^n{}_{(i,j,k)} - E_y^n{}_{(i-1,j,k)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}-1, j_{min}+1 \le j \le j_{max}, k_{min}+1 \le k \le k_{max}]$$

Each of (3.100), (3.101), (3.102) shows the scanning ranges and magnetic fields beyond these ranges are not required elsewhere in the algorithm.

Finally electric flux densities are calculated by the discretized form of (3.28), (3.29), (3.30):

$$D_x^{n+1}{}_{(i,j,k)} = D_x^n{}_{(i,j,k)} \tag{3.103}$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta y}\left(H_z^{n+1}{}_{(i,j+1,k)} - H_z^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta z}\left(H_y^{n+1}{}_{(i,j,k+1)} - H_y^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta y}\left(H_z^n{}_{(i,j+1,k)} - H_z^n{}_{(i,j,k)}\right) - \frac{1}{\Delta z}\left(H_y^n{}_{(i,j,k+1)} - H_y^n{}_{(i,j,k)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}, j_{min}+1 \le j \le j_{max}-1, k_{min}+1 \le k \le k_{max}-1]$$

$$D_y^{n+1}{}_{(i,j,k)} = D_y^n{}_{(i,j,k)} \tag{3.104}$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta z}\left(H_x^{n+1}{}_{(i,j,k+1)} - H_x^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta x}\left(H_z^{n+1}{}_{(i+1,j,k)} - H_z^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta z}\left(H_x^n{}_{(i,j,k+1)} - H_x^n{}_{(i,j,k)}\right) - \frac{1}{\Delta x}\left(H_z^n{}_{(i+1,j,k)} - H_z^n{}_{(i,j,k)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}-1, j_{min}+1 \le j \le j_{max}, k_{min}+1 \le k \le k_{max}-1]$$

$$D_z^{n+1}{}_{(i,j,k)} = D_z^n{}_{(i,j,k)} \tag{3.105}$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta x}\left(H_y^{n+1}{}_{(i+1,j,k)} - H_y^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta y}\left(H_x^{n+1}{}_{(i,j+1,k)} - H_x^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta x}\left(H_y^n{}_{(i+1,j,k)} - H_y^n{}_{(i,j,k)}\right) - \frac{1}{\Delta y}\left(H_x^n{}_{(i,j+1,k)} - H_x^n{}_{(i,j,k)}\right)\right]$$
$$[i_{min}+1 \le i \le i_{max}-1, j_{min}+1 \le j \le j_{max}-1, k_{min}+1 \le k \le k_{max}]$$

Again each of (3.103), (3.104), (3.105) shows the scanning ranges and electric flux densities beyond these ranges are not required elsewhere in the algorithm.

Chapter 4

Detailed Study of the FD–CN–FDTD Method

In this chapter the proposed FD–CN–FDTD method is validated by numerical experiments. By calculating the average error of the method, both for non-lossy and lossy media, the effects of $CFLN$ and spatial resolution, χ, are observed. All the FDTD methods have their own limitations. Before simulating an unknown problem, it is essential to know these limitations. Because, by knowing these one can determine whether the problem in question lies within the confines, where the concerned FDTD method works properly. To understand the limitations of the FD–CN–FDTD method *i.e.* to find out the parameters for which it does not produce expected results, a number of numerical tests have been performed and are shown in this chapter.

4.1 Validation of the FD–CN–FDTD Method

In order to validate the FD–CN–FDTD method, numerical tests were conducted with a computational space of size 30×30×30 cells and composing of inhomogeneous, frequency dependent media. Half of the computational space was filled with medium 1 ($\epsilon_S = 71.66$, $\epsilon_\infty = 34.58$, $\sigma = 0.49$ S/m and $\tau_D = 5.65$ ps) and the other half with medium 2 ($\epsilon_S = 87.34$, $\epsilon_\infty = 49.13$, $\sigma = 0.69$ S/m and $\tau_D = 26.89$ ps) as shown in Fig. 4.1. A z–directed dipole source was placed at (10,15,15) in medium 1, with a time evolution of a modulated Gaussian pulse centred at 3 GHz. Signals were observed at (20,15,15) in medium 2. A uniform spatial sampling of $\Delta x = \Delta y = \Delta z = 10^{-3}$m was used. As a reference,

an identical setup was taken for the standard explicit frequency dependent (FD-)FDTD method. Henceforth, in this thesis Δt refers to the time-step used in the simulation, Δt_{CFL} to the maximum time-step allowed by the CFL stability condition and $CFLN$ to the CFL number defined as $CFLN \equiv \Delta t / \Delta t_{CFL}$. The first $600 \times \Delta t_{CFL}$ ($\Delta t_{CFL} = 1.9$ ps) time period of the E_z field components at the observation point are shown in Fig. 4.2, computed both with the FD-FDTD method when $CFLN = 1$ and with the FD–CN–FDTD method when $CFLN = 1, 3, 5$.

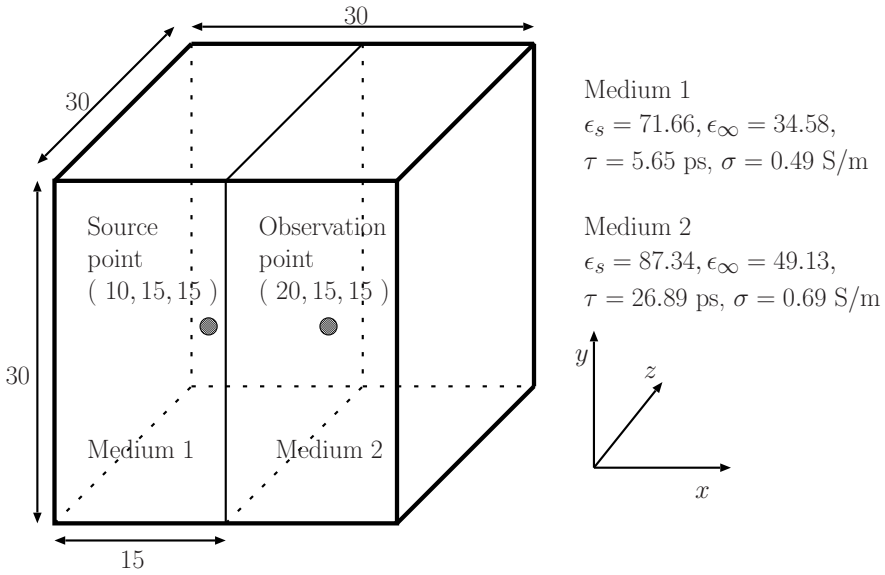


Figure 4.1: Computational space for the validation of the FD–CN–FDTD method.

Good agreement between the signals from the FD–CN–FDTD method and the explicit FD-FDTD method is observed. The explicit FD-FDTD method is stable only when Δt is within the CFL limit. On the other hand, the FD–CN–FDTD method is stable beyond the CFL limit in this numerical test, although numerical errors increasingly appear with higher $CFLN$.

4.2 Numerical Errors in the FD–CN–FDTD Method

As seen in Fig. 4.2, the FD–CN–FDTD method is able to use temporal discretization, Δt, above the CFL limit. But numerical accuracy is compromised at higher

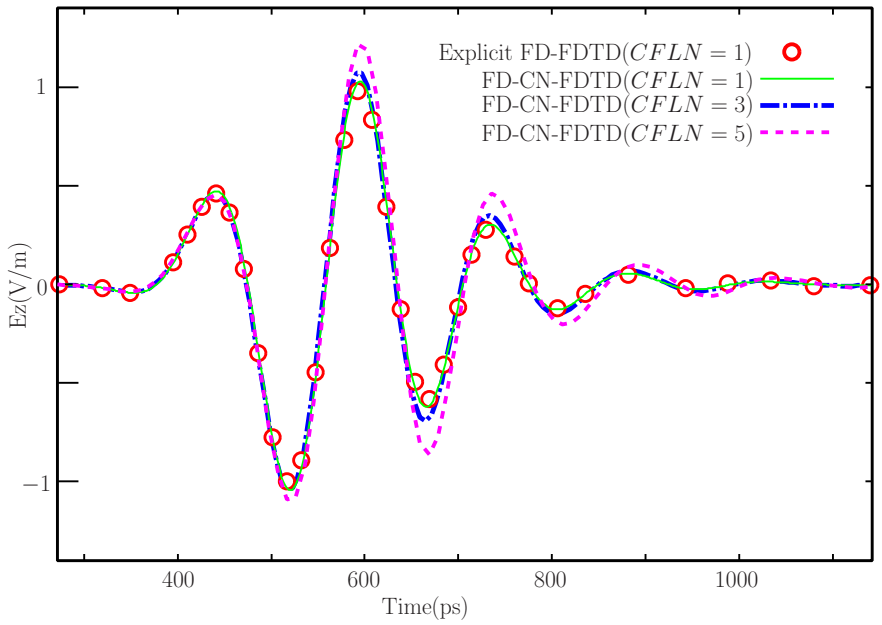


Figure 4.2: Observations from explicit FD–FDTD and FD–CN–FDTD methods.

$CFLN$. Average error of the FD–CN–FDTD method at different $CFLN$ is quantified by numerical tests. In the numerical tests, 160×160×160 cubic computational space filled with a lossy medium was considered. The medium parameters were $\epsilon_{\mathrm{S}} = 71.666161$, $\epsilon_{\infty} = 34.58062$, $\tau_{\mathrm{D}} = 5.6558308$ ps, $\sigma = 0.4993007$ S/m. Source excitation of a modulated Gaussian pulse centred at 3 GHz was located at the centre (80, 80, 80). Spatial sampling was uniform ($\Delta x = \Delta y = \Delta z = 10^{-3}$m), Δt_{CFL} was 1.9 ps and $CFLN$ was varied from 1 to 10. Observations were taken at all the grid points within 20 cells away from the source, converted into frequency domain and the average error, $\mathcal{E}$, was calculated using

$$\mathcal{E} = \sqrt{\frac{\sum_{f} \left(\mathcal{S}_{rcd} - \mathcal{S}_{rcd}^{ref}\right)^2}{\sum_{f} \left(\mathcal{S}_{rcd}^{ref}\right)^2}} \tag{4.1}$$

where, $\mathcal{S}_{rcd}$ is the frequency spectrum of the received signal using the FD–CN–FDTD method and $\mathcal{S}_{rcd}^{ref}$ is that of the reference signal which is calculated using the explicit FD–FDTD method. The average error for the lossy medium is shown in Fig. 4.3. Similar study was performed for the non-lossy medium where all

the parameters were the same as lossy medium, except that the conductivity, σ, was zero. The average error for the non-lossy medium is also plotted in Fig. 4.3. Fig. 4.3 shows when $CFLN \leq 5$ the average error is below 0.1 in both lossy and non-lossy cases.

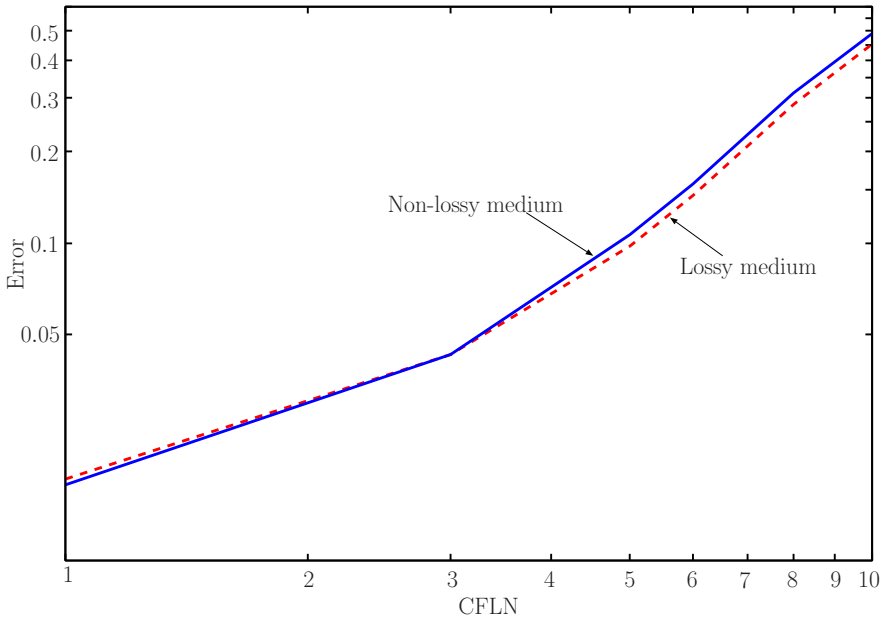


Figure 4.3: Average error of the FD–CN–FDTD method at different $CFLN$ for lossy and non-lossy media.

The effect of $CFLN$ on the numerical error has been observed above but numerical error also depends on the spatial discretization (Δs). In FDTD methods, the temporal discretization and the spatial discretization are related and depending on the spatial resolution, χ, the spatial discretization will have the value, $\Delta s = \lambda/\chi$, where, λ is the wavelength of the signal. If the spatial resolution is too large the method will become inefficient but for accurate modelling of objects with fine features, a high spatial resolution is required [99]. So there is a trade-off between these parameters and their selection is application dependent. To observe the effect of spatial resolution on the accuracy of FD–CN–FDTD, the same numerical tests described above were conducted with varying values of χ ($\chi = 40, 70$ and 100) and the average error was calculated using (4.1). Table 4.1 shows the change in error due to the varying values of χ and $CFLN$. It is seen that, as the spatial resolution is lowered the error is increased. The value of the spatial resolution should be chosen depending on the requirement of accuracy of

any particular application. Although higher $CFLN$ can reduce the total simulation time, it should not be set to very high value without considering other factors. The accuracy requirements for the concerned problem, spatial resolution and CFL number – each of these needs to be optimized properly considering the other.

$CFLN$	Average Error, $\mathcal{E}$		
	$\chi = 40$	$\chi = 70$	$\chi = 100$
1	0.0271318	0.0211179	0.0167486
3	0.0656218	0.0520507	0.0351224
5	0.182472	0.158372	0.12668
6	0.259739	0.23379	0.196922
8	0.438352	0.410313	0.365371
10	0.614931	0.592478	0.547573

Table 4.1: Average error at different spatial resolution (χ) for the lossy media

To observe how the FD–CN–FDTD method handles the simulation of free space, the same numerical tests as above were carried out with the whole computational space filled with air ($\epsilon_{\mathrm{S}} = 1.0$, $\epsilon_\infty = 1.0$, $\tau_{\mathrm{D}} = 0.0$ ps, $\sigma = 0.0$ S/m). The value of χ was 100, Δt_{CFL} was 1.9 ps and $CFLN$ was varied from 1 to 10. For $CFLN = 1$, 2 and 3 the FD–CN–FDTD method always works normally without any divergence. However, when $CFLN$ is above 3 the FD–CN–FDTD method works normally upto certain time steps and thereafter it starts diverging. When $CFLN$ = 4, 5, 6, 7, 8, 9 and 10, divergence starts after 940/4, 935/5, 936/6, 980/7, 984/8, 1026/9 and 1070/10 time steps, respectively. In all these numerical tests, the FD–CN–FDTD method works normally at least upto $900/CFLN$ time steps without any divergence. In the equivalent time of $900/CFLN$ steps (*i.e.* $(900/CFLN) \times \Delta t = 900 \times \Delta t_{CFL}$) the wave propagates $900 \times 1.9 \times 10^{-12} \times 3 \times 10^8$m $= 0.513$ m in the free space. On the other hand, in these tests the size of the computational space was 0.16m×0.16m×0.16m. Therefore, the FD–CN–FDTD method can handle the simulation of free space but in this case the method has the limitation of diverging after a certain number of time steps when $CFLN \geq 4$. When simulating the free space in the FD–CN–FDTD method, the total number of time steps the simulation will run needs to

be carefully selected so that the simulation stops before divergence starts. However, when the computational space consists of frequency dependent and lossy media, the FD–CN–FDTD method is always stable beyond the CFL limit at all the values of $CFLN$.

4.3 Handling the Lossy Media

Although in the real-world most media are lossy, majority of the recently proposed versions of FDTD methods considered the media to be non-lossy for the sake of simplicity. One of the strengths of the FD–CN–FDTD method is its ability to handle the lossy media. In this section, by several numerical tests the accuracy of the simulation of the lossy media in FD–CN–FDTD method is studied. Attenuation caused by the lossy media is calculated by using FD–CN–FDTD and compared against the theoretical attenuation. By studying the effects of varying spatial resolution, it is observed that in order to match the numerical and theoretical attenuation, a threshold of the spatial resolution needs to be maintained.

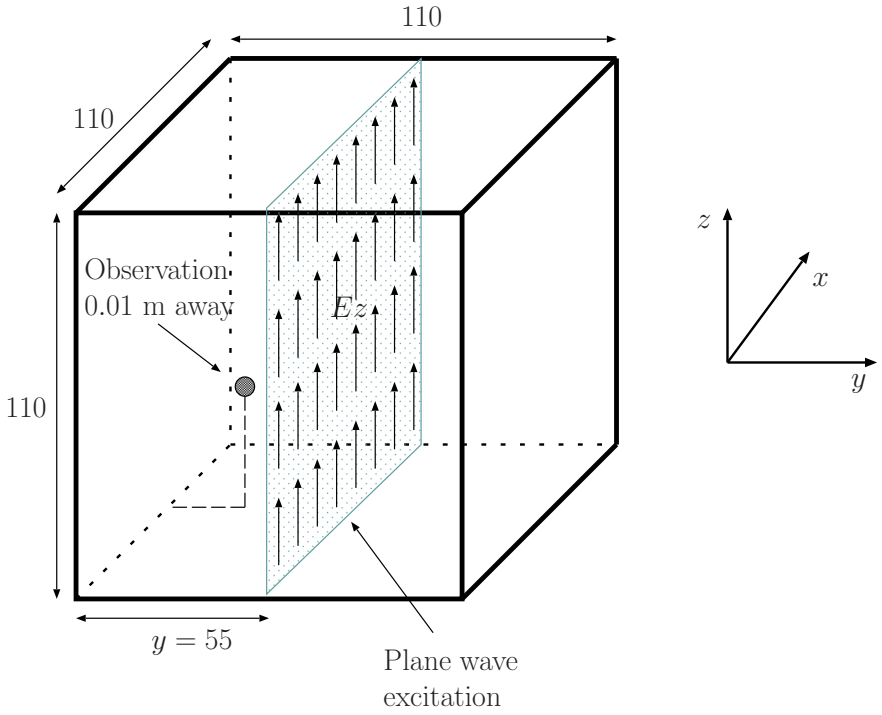


Figure 4.4: Computational environment with the plane wave source excitation.

Numerical tests were carried out with the computational environment shown

in Fig. 4.4. The size of the computational space was 110×110×110 cells and it was filled with three different sets of media parameters, differing only by conductance: $(\sigma, \epsilon_\infty, \epsilon_S, \tau_D) = (0.049\ \text{S/m}, 3.5, 6.2, 39.0\ \text{ps})$, $(0.49\ \text{S/m}, 3.5, 6.2, 39.0\ \text{ps})$ and $(4.9\ \text{S/m}, 3.5, 6.2, 39.0\ \text{ps})$. Spatial resolution was variable: 20, 50, 100, 150, 200, 250, 300 cells per wavelength, resulting in variable spatial discretization. To derive the theoretical attenuation, a plane wave excitation was assumed. In order to approximate the same in the numerical tests, the plane $y = 55$ of the computational space was excited (Fig. 4.4) by Gaussian pulses whose time domain signal is shown in Fig. 4.5 and frequency domain signal is shown in Fig. 4.6.

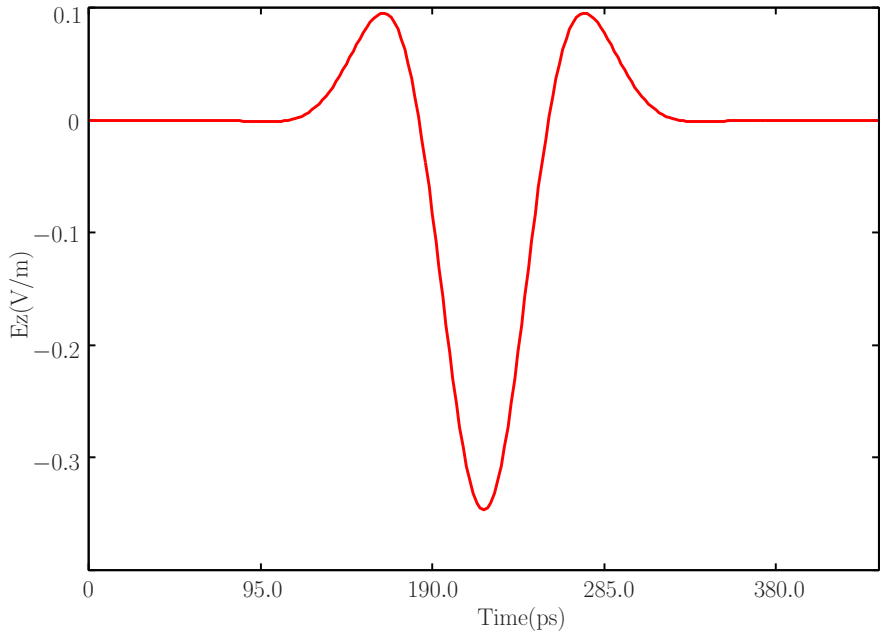


Figure 4.5: Excitation Signal in the Time Domain.

For a lossy medium the theoretical propagation constant, Γ, is calculated as

$$\Gamma = \jmath\omega\sqrt{\mu\left(\epsilon_0\epsilon_\infty + \frac{\epsilon_0\epsilon_S - \epsilon_0\epsilon_\infty}{1 + \jmath\omega\tau_D} - \jmath\frac{\sigma}{\omega}\right)} \tag{4.2}$$

where μ is the permeability, ϵ_0 is the free space permittivity, ϵ_S is the static permittivity, ϵ_∞ is the optical permittivity, τ_D is the characteristic relaxation time and ω is the angular frequency. Propagation constant has a real part, called attenuation constant (α) and an imaginary part, called phase constant (β):

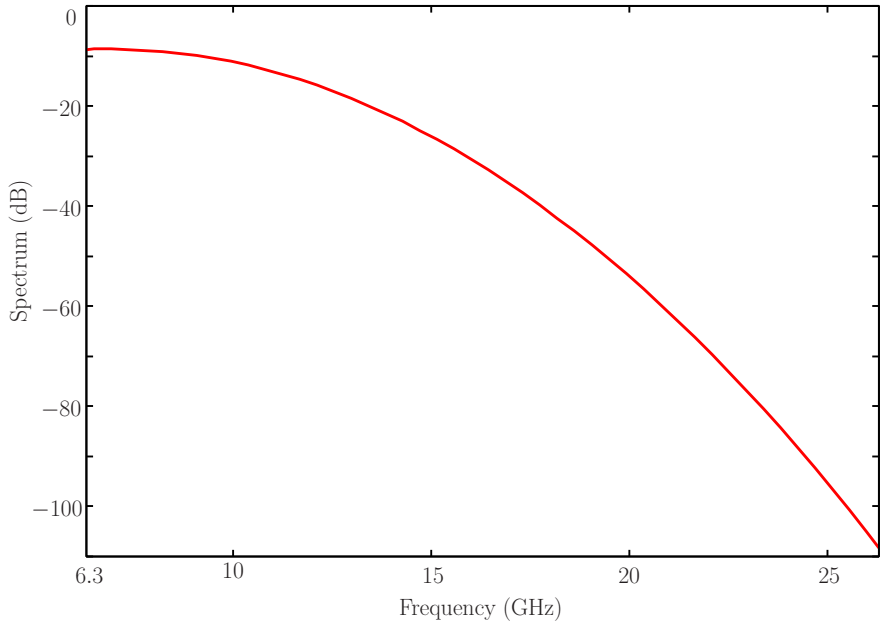


Figure 4.6: Excitation Signal in the Frequency Domain.

$$\Gamma = \alpha + \jmath\beta \tag{4.3}$$

With distance, d, from the source, the amplitude of the signal decays as $\exp(-\alpha d)$ *i.e.* $\exp(-\mathrm{Re}(\Gamma)\, d)$. The theoretical attenuation can be calculated from this expression.

In the numerical tests, observations were taken at 0.01 metres away from the source (*i.e.* $d = 0.01$ m). As the spatial resolution and thereby the spatial discretization were varying, the number of cells between the source plane and the observation point also varied. For spatial resolution $\chi = 150$ cells per wavelength, the observed signal in time domain is shown in Fig. 4.7 and in frequency domain in Fig. 4.8. Fig. 4.8 shows the attenuation of the observed signal at each frequency can be calculated by using

$$attenuation\,(f) = \frac{spectrum\ of\ observed\ signal\,(f)}{spectrum\ of\ excitation\,(f)} \tag{4.4}$$

In the same way, for each of the three different lossy media, at different spatial resolution, the observed signals were transformed into frequency domain and the

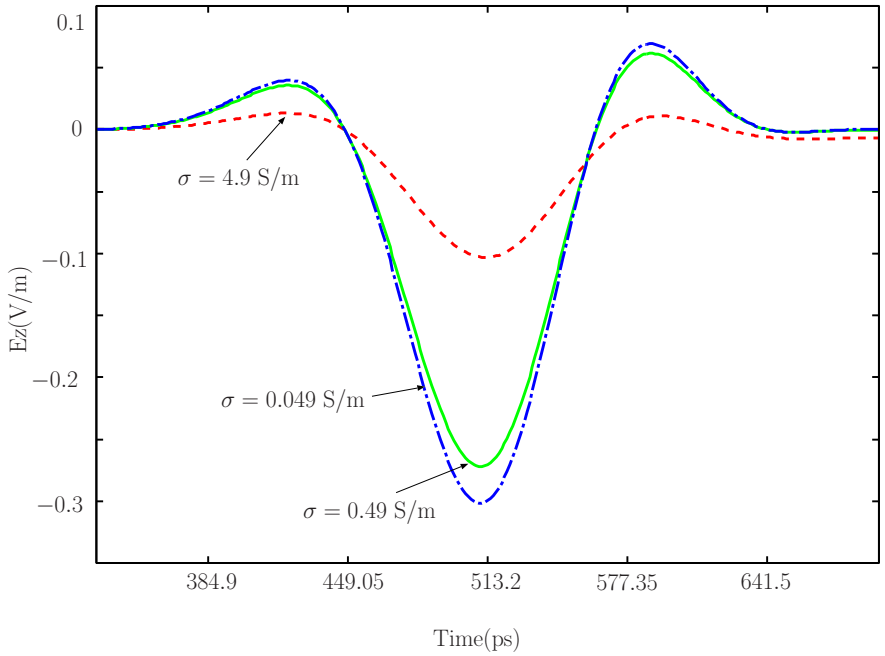


Figure 4.7: Observed signal (in time domain) at 0.01 m away from the plane wave source when $\chi = 150$.

attenuation was calculated using (4.4). Fig. 4.9, Fig. 4.10 and Fig. 4.11 show the attenuation at different spatial resolution for the three different lossy media along with the theoretical attenuation.

Fig. 4.9, Fig. 4.10 and Fig. 4.11 show for highly lossy medium a higher spatial resolution (or a smaller spatial discretization) is required by the FD–CN–FDTD method to match with the theoretical attenuation. The reason for this is, with the increase of conductivity the wavelength of the signal shortens requiring higher spatial resolution [100]. This study also manifests that there is a minimum threshold of the spatial resolution that needs to be ensured to get the acceptable level of attenuation that matches with the theoretical attenuation. The threshold of the spatial resolution varies with the conductivity. Table 4.2 shows the minimum threshold of the spatial resolution for different conductivity at or above which the numerical attenuation matches with the theoretical one. This table also shows the values of skin depth δ (the distance at which the wave attenuates to $1/e$ of the value at the surface) and Δs for these cases. The values of skin depth for $\sigma = 0.049$ S/m, 0.49 S/m and 4.9 S/m are 124, 78 and 37 cells, respectively, from the surface.

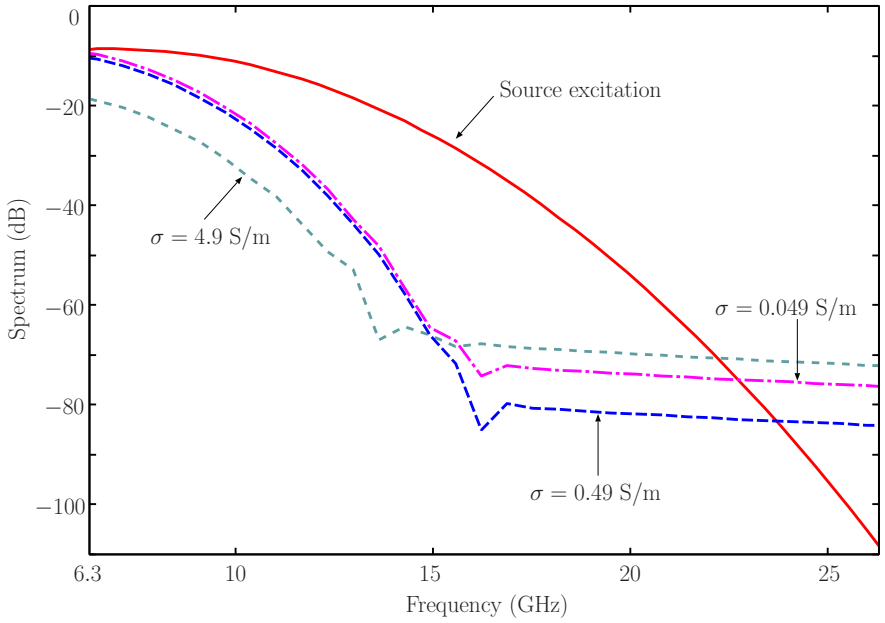


Figure 4.8: Observed signal (in frequency domain) at 0.01 m away from the plane wave source when $\chi = 150$.

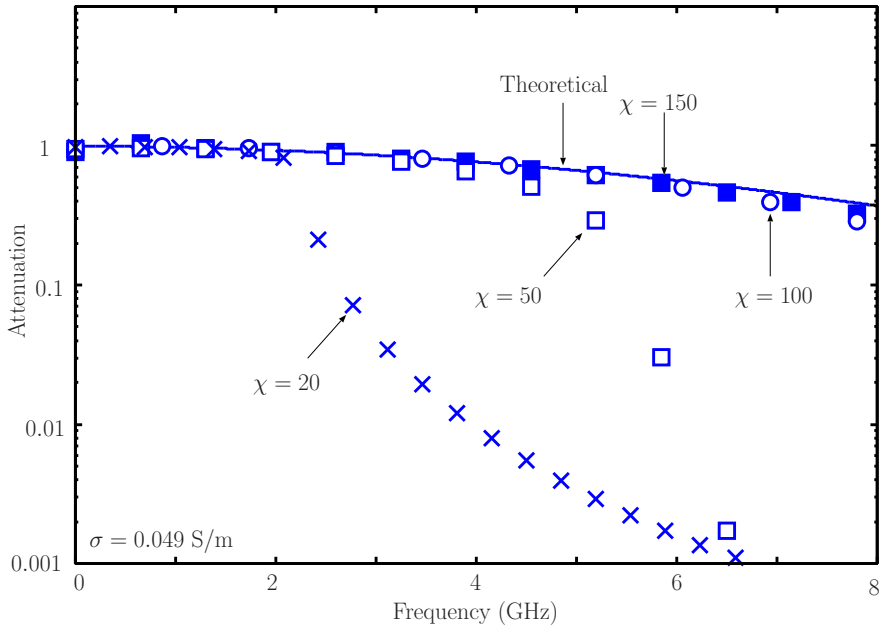


Figure 4.9: Attenuation in the medium with parameters $\sigma = 0.049\text{S/m}, \epsilon_\infty = 3.5, \epsilon_\text{S} = 6.2, \tau_\text{D} = 39.0$ ps, for different spatial resolution.

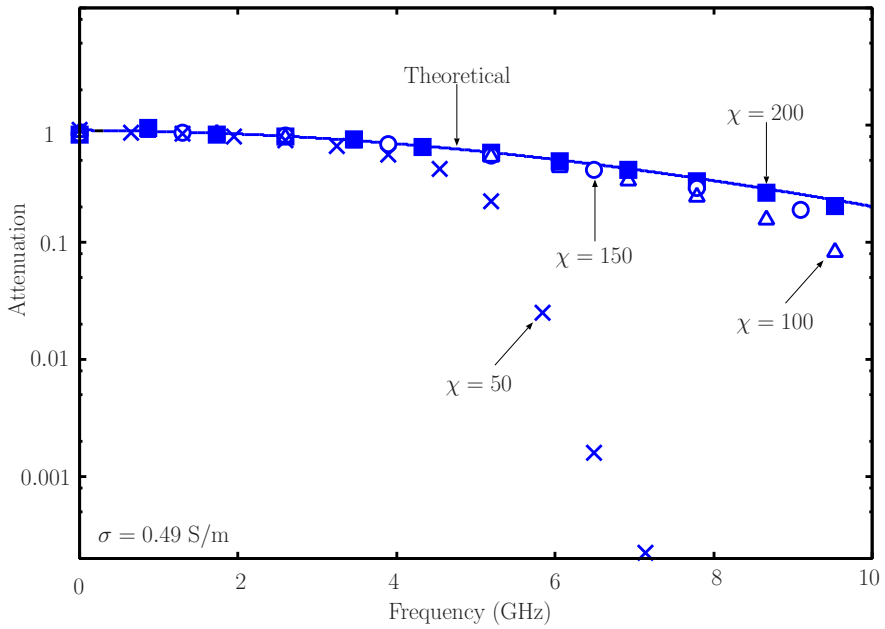


Figure 4.10: Attenuation in the medium with parameters $\sigma = 0.49\text{S/m}, \epsilon_\infty = 3.5, \epsilon_\text{S} = 6.2, \tau_\text{D} = 39.0$ ps, for different spatial resolution.

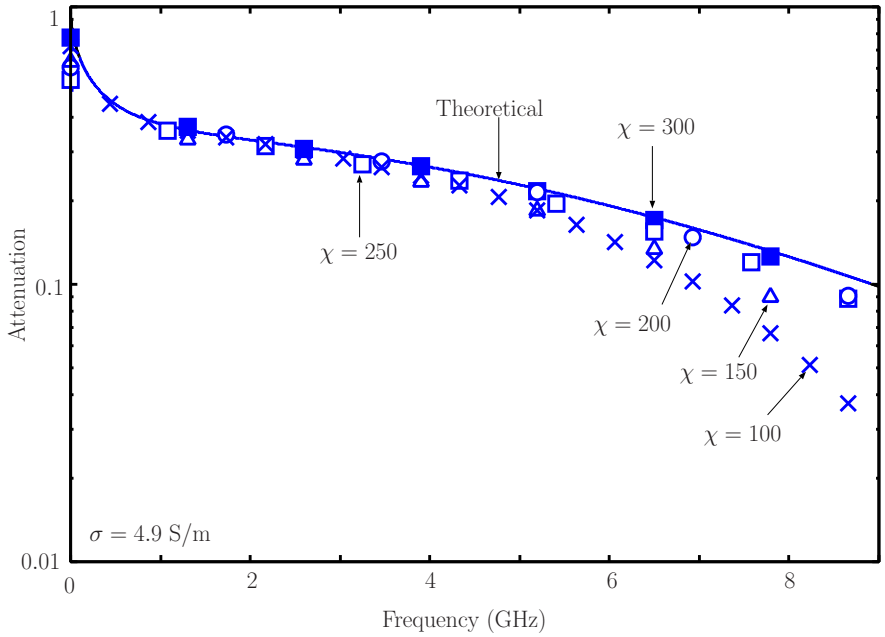


Figure 4.11: Attenuation in the medium with parameters $\sigma = 4.9\text{S/m}, \epsilon_\infty = 3.5, \epsilon_\text{S} = 6.2, \tau_\text{D} = 39.0$ ps, for different spatial resolution.

4.4 Dealing with the Inhomogeneous Media

The FD–CN–FDTD method is capable of simulating inhomogeneous, frequency dependent medium. How the FD–CN–FDTD method handles the interface between two different frequency dependent media is studied in this section. This is

σ(S/m)	0.049	0.49	4.9
χ	100	200	300
Δs (mm)	0.23	0.115	0.0765
δ (mm)	28.65	9.06	2.865

Table 4.2: Threshold of the spatial resolution (χ) required for the matching of numerical and theoretical attenuation curves.

done by analysing the reflection and transmission of the wave travelling from one medium to the other.

For a plane wave travelling from one conductive medium to another with an angle of incidence of 0° the theoretical reflection coefficient is defined as [p.151, [97]]

$$\frac{\eta_2 - \eta_1}{\eta_2 + \eta_1} \tag{4.5}$$

and the theoretical transmission coefficient is defined as [97]

$$\frac{2\eta_2}{\eta_2 + \eta_1} \tag{4.6}$$

where η_1 and η_2 are the characteristic or intrinsic impedance of the first and the second media. If ϵ and μ are the permittivity and permeability of the medium, characteristic impedance is related to these parameters by $\eta = \sqrt{\mu/\epsilon}$. For most materials the relative permeability, μ_r, is very nearly unity [p.85, [97]]; therefore, permeability of all the media in this study is considered to be that of the vacuum μ_0 *i.e.* $\mu = \mu_r\mu_0 = 4\pi \times 10^{-7}$ H/m . So, the relationship of theoretical reflection coefficient becomes

$$\frac{\sqrt{\epsilon_1} - \sqrt{\epsilon_2}}{\sqrt{\epsilon_1} + \sqrt{\epsilon_2}} \tag{4.7}$$

and that of theoretical transmission coefficient becomes

$$\frac{2\sqrt{\epsilon_1}}{\sqrt{\epsilon_1} + \sqrt{\epsilon_2}} \tag{4.8}$$

where ϵ_1 and ϵ_2 are the permittivity of the first and second media. The capability of the FD–CN–FDTD method in dealing with the media interface was studied by numerically calculating the reflection and transmission coefficients and comparing these with the theoretical coefficients. In the numerical experiments, computational space had the size of 120×120×120 cells and consisted of 2 media as shown in Fig. 4.12. Medium 1 filled the space for $1 \leq y \leq 80$, while medium 2 filled for $y \geq 81$. Three different cases, all having the same parameters for medium 1 but different parameters for medium 2, were tested. These are described in Table 4.3.

Case 1				
Medium	ϵ_S	ϵ_∞	τ_D (ps)	σ (S/m)
1	71.66616	34.58062	5.6558308	0.4993007
2	87.34172	49.1395	26.894634	0.6980397

Case 2				
Medium	ϵ_S	ϵ_∞	τ_D (ps)	σ (S/m)
1	71.66616	34.58062	5.6558308	0.4993007
2	207.34172	159.1395	26.894634	0.6980397

Case 3				
Medium	ϵ_S	ϵ_∞	τ_D (ps)	σ (S/m)
1	71.66616	34.58062	5.6558308	0.4993007
2	407.34172	389.1395	26.894634	0.6980397

Table 4.3: Media parameters of the computational space for studying media transition.

For case 1, case 2 and case 3, the permittivity of medium 2 was calculated using $\epsilon = \epsilon_0 \epsilon_r$ and Debye relationship of (3.7). These are $79.53107 - \jmath 19.59149$, $197.4865 - \jmath 23.62453$, and $395.4420 - \jmath 27.65755$, respectively.

In order to generate a plane-wave travelling along the y-axis and having a wave–front in the xz-plane, z-directed modulated Gaussian pulses, centred at 6.9 GHz, were excited at the plane, $y = 40$ (Fig. 4.12). Spatial resolution was variable:

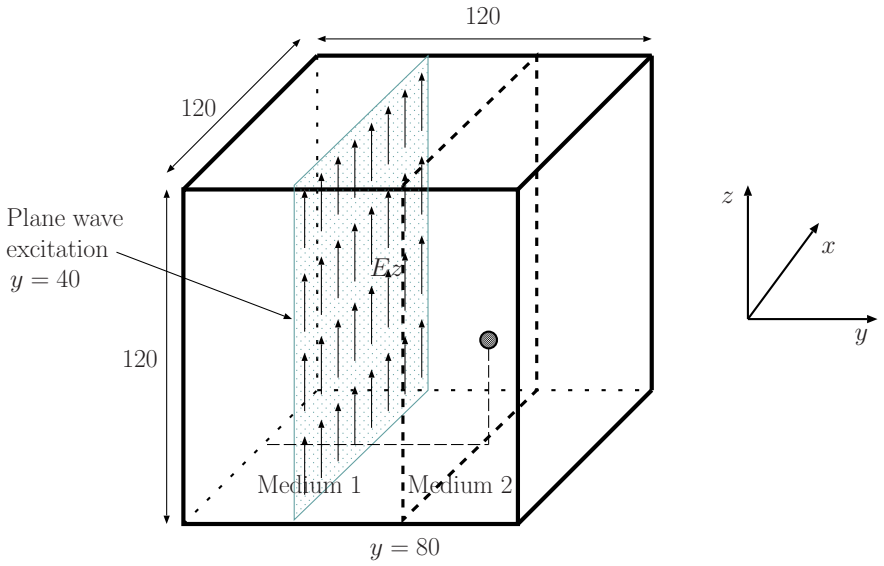


Figure 4.12: FD–CN–FDTD computational space for studying media transition.

20, 60, 100, 140, 180 cells per wavelength, making the spatial discretization vary accordingly. Observations were taken at 3 locations to get the incident, reflected and transmitted signals:

$Signal(i)$ is the incident signal obtained by filling the whole computational space with only medium 1 and taking the observation at $(60, 80, 60)$;

$Signal(r')$ is the signal that contains the reflections from medium 2 and taken at $(60, 80, 60)$ in Fig. 4.12;

$Signal(t)$ is the transmitted signal taken at $(60, 81, 60)$ in Fig. 4.12.

By deducting the incident signal, $signal(i)$, from the signal with reflections, $signal(r')$, the reflected signal can be found. Then the reflection coefficient can be calculated as

$$\frac{max\,|signal(r') - signal(i)|}{max\,|signal(i)|} \tag{4.9}$$

For the three cases, the numerical reflection coefficients were calculated using the FD–CN–FDTD method, with the varying values of spatial resolution (χ). Theoretical reflection coefficients were calculated using (4.7). Fig. 4.13 shows the theoretical and numerical reflection coefficients as a functions of χ. Both the

theoretical and numerical coefficients are seen to be in good match. Reflection coefficient error of the FD–CN–FDTD method, based on the theoretical coefficients, are calculated as

$$\frac{|numerical\ coefficient - theoretical\ coefficient|}{|theoretical\ coefficient|} \tag{4.10}$$

and shown in Fig. 4.14. Next using the incident and transmitted signals, the transmission coefficients are calculated as

$$\frac{max\,|signal(t)|}{max\,|signal(i)|} \tag{4.11}$$

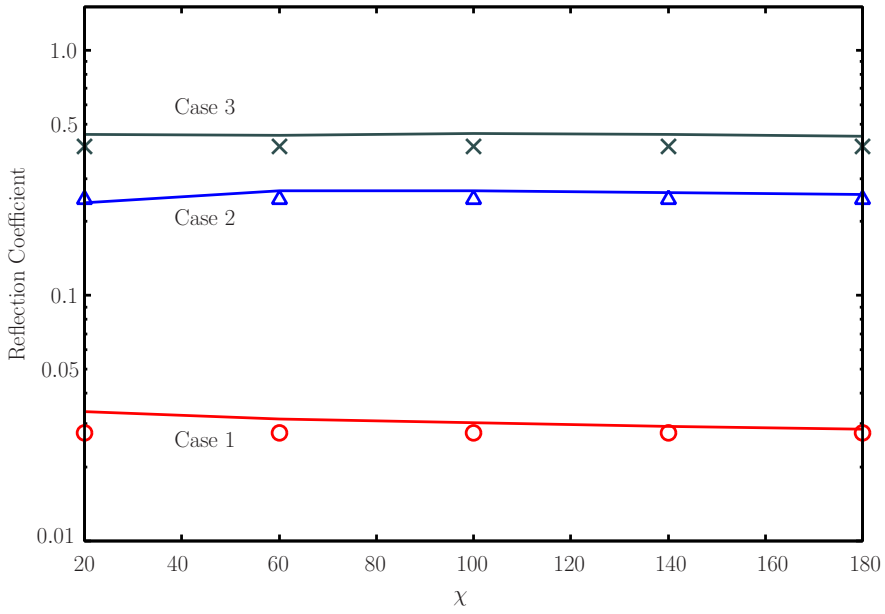


Figure 4.13: Theoretical (symbols) and numerical (solid line) reflection coefficients.

Using (4.11) the numerical transmission coefficients were calculated for the three cases with different χ and using (4.8) the theoretical transmission coefficients were calculated. Both of these coefficients are shown in Fig. 4.15 and found to be in good agreement. The error in the numerical calculation of the transmission coefficients were calculated using (4.10) and shown in Fig. 4.16.

The reason for the difference in the reflection (Fig. 4.13) and transmission (Fig. 4.15) coefficients among the three cases is the differing values of permittivity

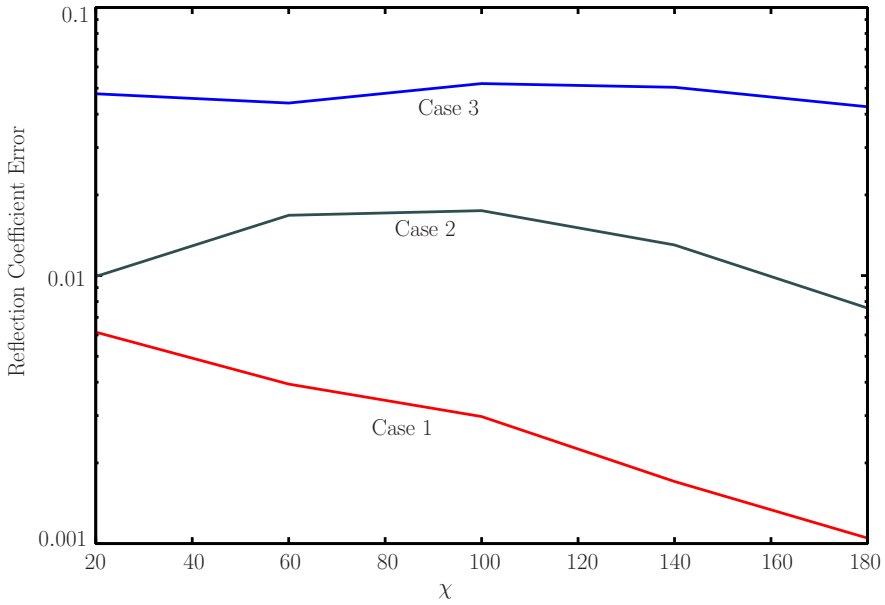


Figure 4.14: Reflection coefficient error.

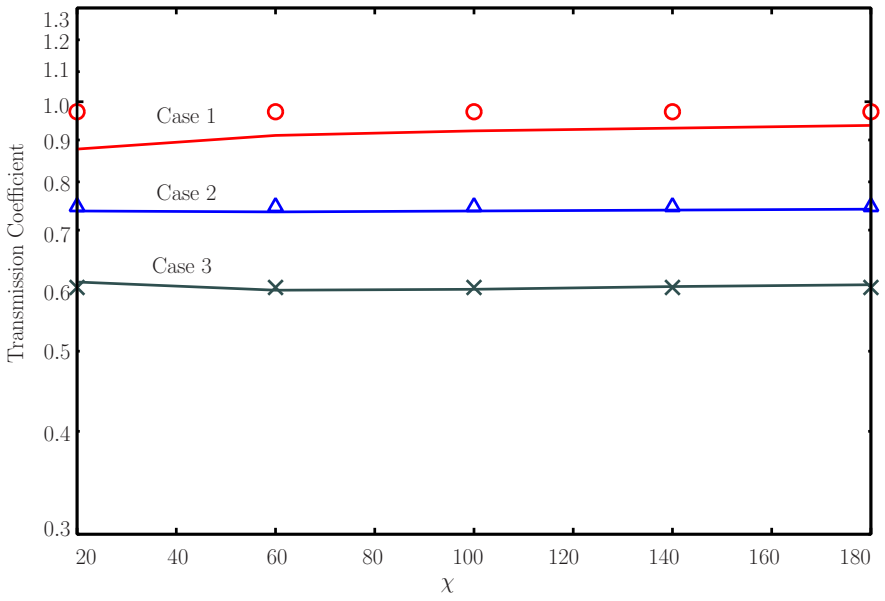


Figure 4.15: Theoretical (symbols) and numerical (solid line) transmission coefficients.

of medium 2 (because the wave propagates from medium 1 to medium 2). From case 1 to case 3, medium 2 has progressively higher values of permittivity. The speed of wave propagation in a dielectric material is related to the reciprocal of the square root of the permittivity of the material *i.e.* speed of propagation $= c \times \frac{1}{\sqrt{\epsilon_r}}$, where, c is the speed of light in free space. That means, with the increase

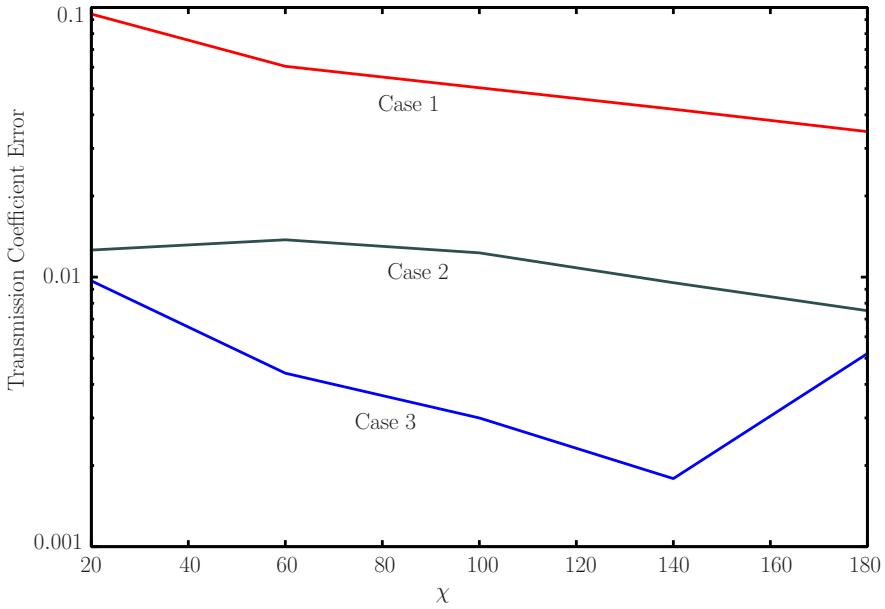


Figure 4.16: Transmission coefficient error.

of permittivity, the speed of propagation goes down. Therefore, as expected [101], in the case of lower permittivity (case 1) the transmission is higher (Fig. 4.15) than in the case of higher permittivity (case 3). On the other hand, as the permittivity increases, there is less penetration and more reflection (Fig. 4.13) [101].

4.5 Analytical Study of Numerical Stability

A numerically stable FDTD method does not increase the magnitude of the solution without bound as time progresses. If this is not the case, the method is unstable. In a stable FDTD method, a small error at any stage produces an smaller cumulative error in the successive stages and the opposite happens in an unstable method. Von Neumann method is often used to analyze the stability of FDTD methods [3]. In this method, the instantaneous electric and magnetic fields, distributed in space across the grid, are Fourier-transformed into the waves in spatial spectral domain and thereby growth factor or amplification factor is derived [27]. Eigenvalues of the amplification factor matrix are then computed to check the stability of the FDTD method. The method is numerically stable if all the eigenvalues are equal to or less than unity.

For the electric field, magnetic field and electric flux densities in x, y and z directions the FD–CN–FDTD equations (3.12), (3.39), (3.41), (3.100), (3.101), (3.102), (3.103), (3.104), (3.105) are re-written below:

$$E_x^{n+1}{}_{(i,j,k)} = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} D_x^{n+1}{}_{(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_x^{n}{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} D_x^{n-1}{}_{(i,j,k)} - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_x^{n}{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} E_x^{n-1}{}_{(i,j,k)} \quad (4.12)$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} D_y^{n+1}{}_{(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_y^{n}{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} D_y^{n-1}{}_{(i,j,k)} - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_y^{n}{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} E_y^{n-1}{}_{(i,j,k)} \quad (4.13)$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} D_z^{n+1}{}_{(i,j,k)} + \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_z^{n}{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} D_z^{n-1}{}_{(i,j,k)} - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_z^{n}{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} E_z^{n-1}{}_{(i,j,k)} \quad (4.14)$$

$$H_x^{n+1}{}_{(i,j,k)} = H_x^{n}{}_{(i,j,k)} + \frac{\Delta t}{2\mu}\left[\frac{1}{\Delta z}\left(E_y^{n+1}{}_{(i,j,k)} - E_y^{n+1}{}_{(i,j,k-1)}\right) - \frac{1}{\Delta y}\left(E_z^{n+1}{}_{(i,j,k)} - E_z^{n+1}{}_{(i,j-1,k)}\right) + \frac{1}{\Delta z}\left(E_y^{n}{}_{(i,j,k)} - E_y^{n}{}_{(i,j,k-1)}\right) - \frac{1}{\Delta y}\left(E_z^{n}{}_{(i,j,k)} - E_z^{n}{}_{(i,j-1,k)}\right)\right] \quad (4.15)$$

$$H_y^{n+1}{}_{(i,j,k)} = H_y^{n}{}_{(i,j,k)} + \frac{\Delta t}{2\mu}\left[\frac{1}{\Delta x}\left(E_z^{n+1}{}_{(i,j,k)} - E_z^{n+1}{}_{(i-1,j,k)}\right) - \frac{1}{\Delta z}\left(E_x^{n+1}{}_{(i,j,k)} - E_x^{n+1}{}_{(i,j,k-1)}\right) + \frac{1}{\Delta x}\left(E_z^{n}{}_{(i,j,k)} - E_z^{n}{}_{(i-1,j,k)}\right) - \frac{1}{\Delta z}\left(E_x^{n}{}_{(i,j,k)} - E_x^{n}{}_{(i,j,k-1)}\right)\right] \quad (4.16)$$

$$H_z^{n+1}{}_{(i,j,k)} = H_z^n{}_{(i,j,k)} \quad (4.17)$$
$$+\frac{\Delta t}{2\mu}\left[\frac{1}{\Delta y}\left(E_x^{n+1}{}_{(i,j,k)} - E_x^{n+1}{}_{(i,j-1,k)}\right) - \frac{1}{\Delta x}\left(E_y^{n+1}{}_{(i,j,k)} - E_y^{n+1}{}_{(i-1,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta y}\left(E_x^n{}_{(i,j,k)} - E_x^n{}_{(i,j-1,k)}\right) - \frac{1}{\Delta x}\left(E_y^n{}_{(i,j,k)} - E_y^n{}_{(i-1,j,k)}\right)\right]$$

$$D_x^{n+1}{}_{(i,j,k)} = D_x^n{}_{(i,j,k)} \quad (4.18)$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta y}\left(H_z^{n+1}{}_{(i,j+1,k)} - H_z^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta z}\left(H_y^{n+1}{}_{(i,j,k+1)} - H_y^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta y}\left(H_z^n{}_{(i,j+1,k)} - H_z^n{}_{(i,j,k)}\right) - \frac{1}{\Delta z}\left(H_y^n{}_{(i,j,k+1)} - H_y^n{}_{(i,j,k)}\right)\right]$$

$$D_y^{n+1}{}_{(i,j,k)} = D_y^n{}_{(i,j,k)} \quad (4.19)$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta z}\left(H_x^{n+1}{}_{(i,j,k+1)} - H_x^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta x}\left(H_z^{n+1}{}_{(i+1,j,k)} - H_z^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta z}\left(H_x^n{}_{(i,j,k+1)} - H_x^n{}_{(i,j,k)}\right) - \frac{1}{\Delta x}\left(H_z^n{}_{(i+1,j,k)} - H_z^n{}_{(i,j,k)}\right)\right]$$

$$D_z^{n+1}{}_{(i,j,k)} = D_z^n{}_{(i,j,k)} \quad (4.20)$$
$$+\frac{\Delta t}{2}\left[\frac{1}{\Delta x}\left(H_y^{n+1}{}_{(i+1,j,k)} - H_y^{n+1}{}_{(i,j,k)}\right) - \frac{1}{\Delta y}\left(H_x^{n+1}{}_{(i,j+1,k)} - H_x^{n+1}{}_{(i,j,k)}\right)\right.$$
$$\left.+\frac{1}{\Delta x}\left(H_y^n{}_{(i+1,j,k)} - H_y^n{}_{(i,j,k)}\right) - \frac{1}{\Delta y}\left(H_x^n{}_{(i,j+1,k)} - H_x^n{}_{(i,j,k)}\right)\right]$$

where, $\nu_1(i,j,k)$, $\nu_2(i,j,k)$, $\nu_3(i,j,k)$, $\nu_4(i,j,k)$, $\nu_5(i,j,k)$, $\nu_6(i,j,k)$ are defined in (3.13), (3.14), (3.15), (3.16), (3.17), (3.18). In the spatial spectral domain, components of the electric field, magnetic field and electric flux densities can be written as:

$$E_r^n{}_{(i,j,k)} = {E_r}^n e^{-\jmath(k_x i\Delta x + k_y j\Delta y + k_z k\Delta z)} \tag{4.21}$$

$$H_r^n{}_{(i,j,k)} = {H_r}^n e^{-\jmath(k_x i\Delta x + k_y j\Delta y + k_z k\Delta z)} \tag{4.22}$$

$$D_r^n{}_{(i,j,k)} = {D_r}^n e^{-\jmath(k_x i\Delta x + k_y j\Delta y + k_z k\Delta z)} \tag{4.23}$$

where $r = x, y, z$ and k_x, k_y, k_z are wave numbers along the x, y, z directions, respectively. Using (4.21), (4.22), (4.23) and assuming $\Delta x = \Delta y = \Delta z = \Delta s$, (4.12),(4.13),(4.14), (4.15), (4.16), (4.17), (4.18), (4.19), (4.20) can be written in the following way:

$$\begin{aligned} {E_x^{n+1}}_{(i,j,k)} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} {D_x^{n+1}}_{(i,j,k)} = {} & \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_x^n{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} {D_x^{n-1}}_{(i,j,k)} \\ & - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_x^n{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} {E_x^{n-1}}_{(i,j,k)} \end{aligned} \tag{4.24}$$

$$\begin{aligned} {E_y^{n+1}}_{(i,j,k)} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} {D_y^{n+1}}_{(i,j,k)} = {} & \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_y^n{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} {D_y^{n-1}}_{(i,j,k)} \\ & - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_y^n{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} {E_y^{n-1}}_{(i,j,k)} \end{aligned} \tag{4.25}$$

$$\begin{aligned} {E_z^{n+1}}_{(i,j,k)} - \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)} {D_z^{n+1}}_{(i,j,k)} = {} & \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)} D_z^n{}_{(i,j,k)} + \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)} {D_z^{n-1}}_{(i,j,k)} \\ & - \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)} E_z^n{}_{(i,j,k)} - \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)} {E_z^{n-1}}_{(i,j,k)} \end{aligned} \tag{4.26}$$

$$
\begin{aligned}
H_x^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_y^{n+1}{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_y^{n+1}{}_{(i,j,k)} e^{\jmath k_z \Delta z} \\
+ \frac{\Delta t}{2\mu\Delta s} E_z^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_z^{n+1}{}_{(i,j,k)} e^{\jmath k_y \Delta y} \\
= H_x^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_y^n{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_y^n{}_{(i,j,k)} e^{\jmath k_z \Delta z} \\
- \frac{\Delta t}{2\mu\Delta s} E_z^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_z^n{}_{(i,j,k)} e^{\jmath k_y \Delta y}
\end{aligned}
\tag{4.27}
$$

$$
\begin{aligned}
H_y^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_z^{n+1}{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_z^{n+1}{}_{(i,j,k)} e^{\jmath k_x \Delta x} \\
+ \frac{\Delta t}{2\mu\Delta s} E_x^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_x^{n+1}{}_{(i,j,k)} e^{\jmath k_z \Delta z} \\
= H_y^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_z^n{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_z^n{}_{(i,j,k)} e^{\jmath k_x \Delta x} \\
- \frac{\Delta t}{2\mu\Delta s} E_x^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_x^n{}_{(i,j,k)} e^{\jmath k_z \Delta z}
\end{aligned}
\tag{4.28}
$$

$$
\begin{aligned}
H_z^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_x^{n+1}{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_x^{n+1}{}_{(i,j,k)} e^{\jmath k_y \Delta y} \\
+ \frac{\Delta t}{2\mu\Delta s} E_y^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_y^{n+1}{}_{(i,j,k)} e^{\jmath k_x \Delta x} \\
= H_z^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_x^n{}_{(i,j,k)} - \frac{\Delta t}{2\mu\Delta s} E_x^n{}_{(i,j,k)} e^{\jmath k_y \Delta y} \\
- \frac{\Delta t}{2\mu\Delta s} E_y^n{}_{(i,j,k)} + \frac{\Delta t}{2\mu\Delta s} E_y^n{}_{(i,j,k)} e^{\jmath k_x \Delta x}
\end{aligned}
\tag{4.29}
$$

$$
\begin{aligned}
D_x^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\Delta s} H_z^{n+1}{}_{(i,j,k)} e^{-\jmath k_y \Delta y} + \frac{\Delta t}{2\Delta s} H_z^{n+1}{}_{(i,j,k)} \\
+ \frac{\Delta t}{2\Delta s} H_y^{n+1}{}_{(i,j,k)} e^{-\jmath k_z \Delta z} - \frac{\Delta t}{2\Delta s} H_y^{n+1}{}_{(i,j,k)} \\
= D_x^n{}_{(i,j,k)} + \frac{\Delta t}{2\Delta s} H_z^n{}_{(i,j,k)} e^{-\jmath k_y \Delta y} - \frac{\Delta t}{2\Delta s} H_z^n{}_{(i,j,k)} \\
- \frac{\Delta t}{2\Delta s} H_y^n{}_{(i,j,k)} e^{-\jmath k_z \Delta z} + \frac{\Delta t}{2\Delta s} H_y^n{}_{(i,j,k)}
\end{aligned}
\tag{4.30}
$$

$$
\begin{aligned}
D_y^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\Delta s} H_x^{n+1}{}_{(i,j,k)} e^{-\jmath k_z \Delta z} + \frac{\Delta t}{2\Delta s} H_x^{n+1}{}_{(i,j,k)} \\
+ \frac{\Delta t}{2\Delta s} H_z^{n+1}{}_{(i,j,k)} e^{-\jmath k_x \Delta x} - \frac{\Delta t}{2\Delta s} H_z^{n+1}{}_{(i,j,k)} \\
= D_y^n{}_{(i,j,k)} + \frac{\Delta t}{2\Delta s} H_x^n{}_{(i,j,k)} e^{-\jmath k_z \Delta z} - \frac{\Delta t}{2\Delta s} H_x^n{}_{(i,j,k)} \\
- \frac{\Delta t}{2\Delta s} H_z^n{}_{(i,j,k)} e^{-\jmath k_x \Delta x} + \frac{\Delta t}{2\Delta s} H_z^n{}_{(i,j,k)}
\end{aligned}
\tag{4.31}
$$

$$
\begin{aligned}
D_z^{n+1}{}_{(i,j,k)} - \frac{\Delta t}{2\Delta s} H_y^{n+1}{}_{(i,j,k)} e^{-\jmath k_x \Delta x} + \frac{\Delta t}{2\Delta s} H_y^{n+1}{}_{(i,j,k)} \\
+ \frac{\Delta t}{2\Delta s} H_x^{n+1}{}_{(i,j,k)} e^{-\jmath k_y \Delta y} - \frac{\Delta t}{2\Delta s} H_x^{n+1}{}_{(i,j,k)} \\
= D_z^n{}_{(i,j,k)} + \frac{\Delta t}{2\Delta s} H_y^n{}_{(i,j,k)} e^{-\jmath k_x \Delta x} - \frac{\Delta t}{2\Delta s} H_y^n{}_{(i,j,k)} \\
- \frac{\Delta t}{2\Delta s} H_x^n{}_{(i,j,k)} e^{-\jmath k_y \Delta y} + \frac{\Delta t}{2\Delta s} H_x^n{}_{(i,j,k)}
\end{aligned}
\tag{4.32}
$$

(4.24),(4.25),(4.26),(4.27),(4.28),(4.29),(4.30),(4.31),(4.32) can be written as

$$
\boldsymbol{W_1} U^{n+1} = \boldsymbol{W_2} U^n + \boldsymbol{W_3} U^{n-1} \tag{4.33}
$$

where $\boldsymbol{W_1}, \boldsymbol{W_2}$ and $\boldsymbol{W_3}$ are 9×9 matrices and U is a 9×1 vector. $\boldsymbol{W_1}, \boldsymbol{W_2}, \boldsymbol{W_3}$ and U are described in (4.34),(4.35),(4.36) and (4.37), respectively, below:

$$
\boldsymbol{W_1} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & -\zeta_a & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & -\zeta_a & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -\zeta_a \\
0 & -\Upsilon + \Upsilon e^{jk_z\Delta z} & \Upsilon - \Upsilon e^{jk_y\Delta y} & 1 & 0 & 0 & 0 & 0 & 0 \\
\Upsilon - \Upsilon e^{jk_z\Delta z} & 0 & -\Upsilon + \Upsilon e^{jk_x\Delta x} & 0 & 1 & 0 & 0 & 0 & 0 \\
-\Upsilon + \Upsilon e^{jk_y\Delta y} & \Upsilon - \Upsilon e^{jk_x\Delta x} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma e^{-jk_z\Delta z} - \Gamma & -\Gamma e^{-jk_y\Delta y} + \Gamma & 1 & 0 & 0 \\
0 & 0 & 0 & -\Gamma e^{-jk_z\Delta z} + \Gamma & 0 & \Gamma e^{-jk_x\Delta x} - \Gamma & 0 & 1 & 0 \\
0 & 0 & 0 & \Gamma e^{-jk_y\Delta y} - \Gamma & -\Gamma e^{-jk_x\Delta x} + \Gamma & 0 & 0 & 0 & 1
\end{pmatrix} \tag{4.34}
$$

$$
\boldsymbol{W_2} = \begin{pmatrix}
-\zeta_e & 0 & 0 & 0 & 0 & 0 & \zeta_b & 0 & 0 \\
0 & -\zeta_e & 0 & 0 & 0 & 0 & 0 & \zeta_b & 0 \\
0 & 0 & -\zeta_e & 0 & 0 & 0 & 0 & 0 & \zeta_b \\
0 & \Upsilon - \Upsilon e^{jk_z\Delta z} & -\Upsilon + \Upsilon e^{jk_y\Delta y} & 1 & 0 & 0 & 0 & 0 & 0 \\
-\Upsilon + \Upsilon e^{jk_z\Delta z} & 0 & \Upsilon - \Upsilon e^{jk_x\Delta x} & 0 & 1 & 0 & 0 & 0 & 0 \\
\Upsilon - \Upsilon e^{jk_y\Delta y} & -\Upsilon + \Upsilon e^{jk_x\Delta x} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\Gamma e^{-jk_z\Delta z} + \Gamma & \Gamma e^{-jk_y\Delta y} - \Gamma & 1 & 0 & 0 \\
0 & 0 & 0 & +\Gamma e^{-jk_z\Delta z} - \Gamma & 0 & -\Gamma e^{-jk_x\Delta x} + \Gamma & 0 & 1 & 0 \\
0 & 0 & 0 & -\Gamma e^{-jk_y\Delta y} + \Gamma & \Gamma e^{-jk_x\Delta x} - \Gamma & 0 & 0 & 0 & 1
\end{pmatrix} \tag{4.35}
$$

$$\boldsymbol{W_3} = \begin{pmatrix} -\zeta_f & 0 & 0 & 0 & 0 & 0 & \zeta_c & 0 & 0 \\ 0 & -\zeta_f & 0 & 0 & 0 & 0 & 0 & \zeta_c & 0 \\ 0 & 0 & -\zeta_f & 0 & 0 & 0 & 0 & 0 & \zeta_c \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.36}$$

$$U = \begin{pmatrix} E_x \\ E_y \\ E_z \\ H_x \\ H_y \\ H_z \\ D_x \\ D_y \\ D_z \end{pmatrix} \tag{4.37}$$

In (4.34),(4.35) and (4.36) following assumptions were made

$\Upsilon = \frac{\Delta t}{2\mu\Delta s}$
$\Gamma = \frac{\Delta t}{2\Delta s}$
$\zeta_a = \frac{\nu_1(i,j,k)}{\nu_4(i,j,k)}$
$\zeta_b = \frac{\nu_2(i,j,k)}{\nu_4(i,j,k)}$
$\zeta_c = \frac{\nu_3(i,j,k)}{\nu_4(i,j,k)}$
$\zeta_e = \frac{\nu_5(i,j,k)}{\nu_4(i,j,k)}$
$\zeta_f = \frac{\nu_6(i,j,k)}{\nu_4(i,j,k)}$

If in (4.33), the terms on the left hand side were at $(n+1)\Delta t$ and those on the right hand side were at $n\Delta t$, it would have been straightforward to find the growth factor. However, because of the presence of the term at $(n-1)\Delta t$ on the right hand side of (4.33), calculation of the growth factor of the FD–CN–FDTD

method is not trivial. To circumvent this, using a 9×9 null matrix $\boldsymbol{O}$ (4.33) can be written as

$$[\boldsymbol{W_1}]U^{n+1} + [\boldsymbol{O}]U^n = [\boldsymbol{W_2}]U^n + [\boldsymbol{W_3}]U^{n-1} \tag{4.38}$$

If $[\boldsymbol{I}]$ is a 9×9 identity matrix, an equation having the same time steps on both sides of the equality as in (4.38) (*i.e.* $n+1$ and n terms on the left hand side and n and $n-1$ terms on the right hand side) can be written

$$[\boldsymbol{O}]U^{n+1} + [\boldsymbol{I}]U^n = [\boldsymbol{I}]U^n + [\boldsymbol{O}]U^{n-1} \tag{4.39}$$

Using $\boldsymbol{W_1}$, $\boldsymbol{W_2}$, $\boldsymbol{W_3}$, $\boldsymbol{I}$, $\boldsymbol{O}$ as block matrices and combining (4.38) and (4.39)

$$\begin{bmatrix} \boldsymbol{W_1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} U^{n+1} \\ U^n \end{bmatrix} = \begin{bmatrix} \boldsymbol{W_2} & \boldsymbol{W_3} \\ \boldsymbol{I} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} U^n \\ U^{n-1} \end{bmatrix} \tag{4.40}$$

Assuming two vectors, each of size 18×1, as $[T^{n+1}] = \begin{bmatrix} U^{n+1} \\ U^n \end{bmatrix}$ and $[T^n] = \begin{bmatrix} U^n \\ U^{n-1} \end{bmatrix}$ (4.40) becomes

$$\begin{bmatrix} \boldsymbol{W_1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} T^{n+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{W_2} & \boldsymbol{W_3} \\ \boldsymbol{I} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} T^n \end{bmatrix} \tag{4.41}$$

Now the growth factor of the FD–CN–FDTD method can be determined from (4.41) by

$$\Lambda = \left[\boldsymbol{M}^{-1}\right][\boldsymbol{N}] \tag{4.42}$$

where both $[\boldsymbol{M}]$ and $[\boldsymbol{N}]$ have same dimension, 18×18, and

$$[\boldsymbol{M}] = \begin{bmatrix} \boldsymbol{W_1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{bmatrix} \tag{4.43}$$

$$[\boldsymbol{N}] = \begin{bmatrix} \boldsymbol{W_2} & \boldsymbol{W_3} \\ \boldsymbol{I} & \boldsymbol{O} \end{bmatrix} \tag{4.44}$$

For the FD–CN–FDTD method to be stable, eigenvalues of Λ have to be less than or equal to unity. Usually computer programmes like Mathematica is used to symbolically calculate the eigenvalues of Λ. Because the size of this symbolic computation is prohibitively large, it was not possible to successfully run this analytical computation to the completion and obtain the eigenvalues of Λ, neither in Mathematica nor in Matlab Symbolic Math Toolbox. Both of these programmes exhaust the available computing resources (memory) while handling the computation of large symbolic matrices of (4.42) having the size of 18×18 and involving matrix inversion as well.

On the computing resources of Research Computing Services (RCS) [102] the Mathematica code for symbolic computation of eigenvalues ran over 7 days 7 hours at about 8 GB of memory. Then on 7th day at 19th hour the memory used by the code progressively increased to 32.6 GB (by 8 days 3 hours) and it started swapping to the disk. After 8 days 11 hours it failed completely:

```
   Aug  8 23:13:44 caterpillar3
kernel: Out of memory: Killed process 22261 (MathKernel)
```

It was not possible to get the access to more powerful computing resources than RCS to run this Mathematica code to the end.

Chapter 5

Efficient Solvers for the FD–CN–FDTD Method

The proposed FD–CN–FDTD method requires solution of a large number of simultaneous linear equations. When the method is applied to electromagnetic problems most of the CPU time is spent on this solution of linear algebraic equations. Therefore, an efficient solution is essential to gain the benefit of the FD–CN–FDTD method. This chapter deals with the issues related to the solution of the FD–CN–FDTD method which lies at its core.

5.1 Sparse Matrix

The FD–CN–FDTD method yields a large set of linear equations resulting in a huge sparse matrix. These come from (3.32), (3.40), (3.42) and those mentioned in Tables 3.1, 3.2, 3.3. By applying these equations to all Yee–grid locations a system of linear equations of $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{c}$ is found. Here $\boldsymbol{A}$ is the coefficient matrix, $\boldsymbol{u}$ represents a vector with the electric field components to be solved and $\boldsymbol{c}$ is the excitation vector. The coefficient matrix $\boldsymbol{A}$ is highly sparse. Its size depends on the size of the computational space while its characteristics depend on media parameters: ϵ_{S}, ϵ_{∞}, σ and τ_{D} and temporal discretization, Δt. The size of matrix $\boldsymbol{A}$ is $(3\cdot(N_x-1)\cdot(N_y-1)\cdot(N_z-1))\times(3\cdot(N_x-1)\cdot(N_y-1)\cdot(N_z-1))$ where N_x, N_y, N_z are the size of the computational space in x, y and z directions, respectively.

When the problem space is homogeneous, the coefficient matrix $\boldsymbol{A}$ is symmetric and otherwise asymmetric. Fig. 5.1 shows the sparsity pattern of $\boldsymbol{A}$ in the case

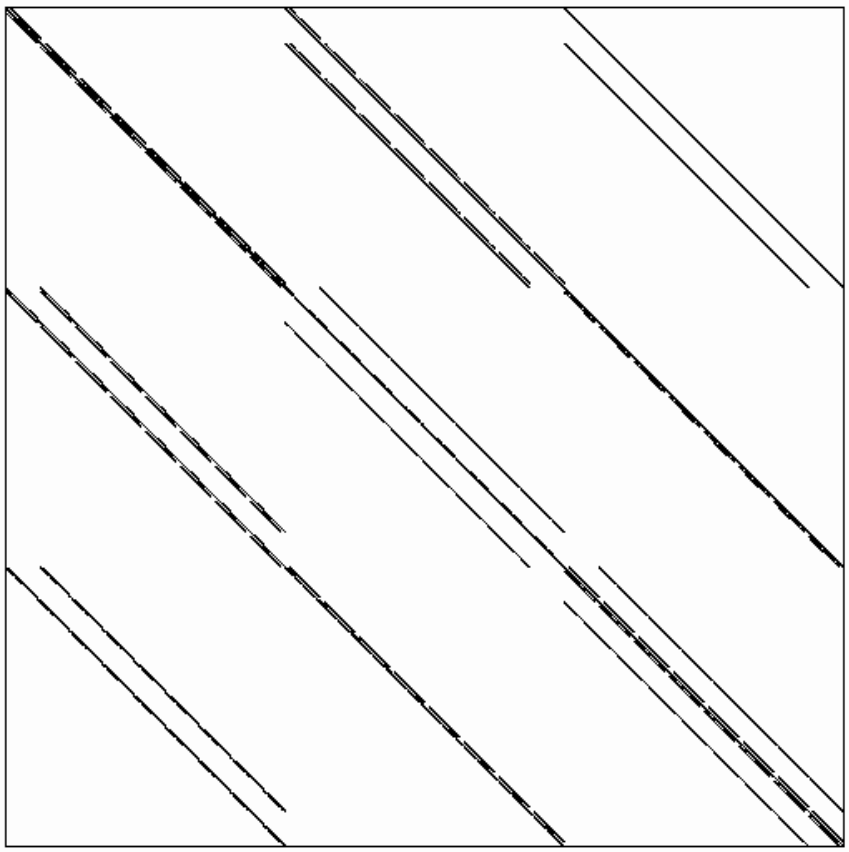


Figure 5.1: Sparsity pattern of the coefficient matrix $\boldsymbol{A}$ of the FD–CN–FDTD method, when $\epsilon_{\mathrm{S}} = 6.2$, $\epsilon_{\infty} = 3.5$, $\sigma = 0.029$ S/m and $\tau_{\mathrm{D}} = 39.0$ ps.

where the entire computational space is filled with Debye parameters $\epsilon_{\mathrm{S}} = 6.2$, $\epsilon_{\infty} = 3.5$, $\sigma = 0.029$ S/m and $\tau_{\mathrm{D}} = 39.0$ ps. The sparsity pattern of Fig. 5.1 is similar to that of three dimensional Finite Difference Frequency Domain (FDFD) method [103], and therefore the findings in this research could also be useful to the FDFD researchers.

5.1.1 Condition Number and Diagonal Dominance

The ease of solution of a linear system of equations can be measured by the condition number of $\boldsymbol{A}$. Condition number measures the stability or sensitivity of a matrix (and of the linear system it represents) to numerical operations. It is defined as $\kappa(\boldsymbol{A}) = \|\boldsymbol{A}\|\|\boldsymbol{A}^{-1}\|$ where $\|.\|$ is p-norm of the matrix[1] and p can be 1,

[1]The norm of a matrix is a scalar that gives some measure of the magnitude of the elements of the matrix. 1-norm condition number is considered in this thesis. 1-norm of matrix $\boldsymbol{A}$ is

2 or ∞. Although $\kappa(\boldsymbol{A})$ depends on the choice of norm, either of these norms typically gives roughly comparable values of it [104]. In this study 1-norm condition number was considered. A sparse matrix system with a high condition number is numerically ill-conditioned and is difficult to solve. Conversely a system with low condition number is well-conditioned and is relatively easy to solve.

In practice, however, computation of the condition number is quite difficult, time consuming and sometimes impossible. Because to compute the condition number, inversion of the matrix $\boldsymbol{A}$ is required which is computationally expensive, specially when $\boldsymbol{A}$ is large. For the FD–CN–FDTD method when the computational space was larger than 15×15×15 cells it was never possible to compute the condition number of the generated sparse matrix with the available computing resources (dual core AMD Opteron Processor 250 with 12GB memory and 1GB cache-size). For computational space of 15×15×15 cells 1-norm condition number was calculated for homogeneous and inhomogeneous media with different $CFLN$ and is presented in Table 5.1. For the homogeneous case, Debye parameters for the whole space was $\epsilon_{\mathrm{S}} = 4.8$, $\epsilon_{\infty} = 2.8$, $\sigma = 0.20$ S/m and $\tau_{\mathrm{D}} = 7.0$ ps. For the inhomogeneous case, the computational space consisted of 3 stratified media having equal size: one-third having the parameters $\epsilon_{\mathrm{S}} = 4.8$, $\epsilon_{\infty} = 2.8$, $\sigma = 0.20$ S/m and $\tau_{\mathrm{D}} = 7.0$ ps; one-third with the parameters $\epsilon_{\mathrm{S}} = 6.2$, $\epsilon_{\infty} = 3.5$, $\sigma = 0.029$ S/m and $\tau_{\mathrm{D}} = 39.0$ ps and the remaining part with the parameters $\epsilon_{\mathrm{S}} = 9.5$, $\epsilon_{\infty} = 4.2$, $\sigma = 0.019$ S/m and $\tau_{\mathrm{D}} = 77.0$ ps. Table 5.1 shows for high $CFLN$ the matrix becomes severely ill-conditioned, requiring high computation time to be solved. This finding is in line with that of [67] which reports the same for frequency–independent Crank–Nicolson method. Table 5.1 also shows that, irrespective of the media being homogeneous or inhomogeneous, the condition number increases at a similar rate with the $CFLN$.

Similar conclusions are found when diagonal dominance of the coefficient matrix[1] is considered instead of the condition number. Diagonal dominance of the

found by summing the absolute values of the elements in each column of $\boldsymbol{A}$ and then taking the largest of these column sums *i.e.* $\|\boldsymbol{A}\|_1 = \max\limits_{1\le j\le n} \sum\limits_{i=1}^{m} |a_{i,j}|$ [104]

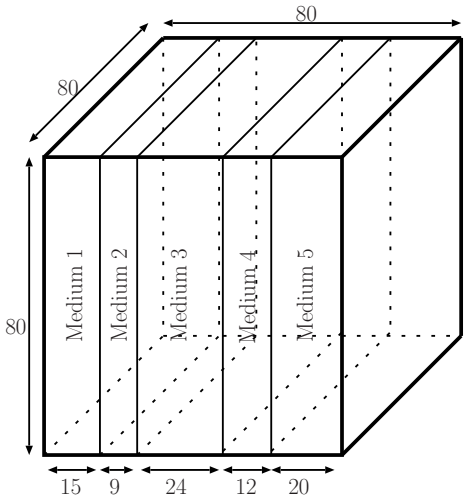
[1]If $a_{i,j}$ is the (i,j)-th element of matrix $\boldsymbol{A}$, then $\boldsymbol{A}$ is diagonally dominant if $|a_{i,i}| \geq \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}|$ for all the i rows.

$CFLN$	Condition Number (homogeneous medium)	Condition Number (inhomogeneous medium)
1	3.58934479888516	3.80026294877373
2	4.77889162596216	5.63798835383051
3	12.5571219884572	8.03725239471826
4	88.9828959798497	142.376370566024
5	57150.0480720380	45976.7592194050
6	1434509822.80038	883345491.644407
7	37699399327182.4	19798632060535.8
8	4.382859536859418E+017	2.061438929431951E+017
9	2.255666356921374E+021	9.217501368037234E+020
10	4.677124697624521E+024	1.923201534239998E+024
11	5.831503312083712E+027	2.104623342204125E+027
12	3.849638466963231E+030	1.317912729029270E+030
13	1.435933921690451E+033	5.075254678012810E+032
14	3.797326242305980E+035	1.282211916790933E+035
15	6.933718165858368E+037	2.241768644600929E+037
16	9.136650096500494E+039	2.835285743200455E+039
17	9.015011444539032E+041	2.691888048797409E+041
18	6.870749991581516E+043	1.979287789772085E+043
19	4.153048717710373E+045	1.157282615412316E+045
20	2.036431555022701E+047	5.503870731510703E+046

Table 5.1: 1-norm condition number at different $CFLN$ for homogeneous and inhomogeneous media for the computational space of 15×15×15 cells

FD–CN–FDTD coefficient matrix improves when $CFLN$ decreases, leading to matrices which are easier to solve. To study the effect of $CFLN$ on diagonal dominance an inhomogeneous cubic space of size 80×80×80 cells having 5 different media as shown in Fig. 5.2 was considered. A homogeneous computational space was also considered which had Debye parameters $\epsilon_{\mathrm{S}} = 6.2$, $\epsilon_{\infty} = 3.5$, $\sigma = 0.029$ S/m and $\tau_{\mathrm{D}} = 39.0$ ps for the whole cubic space of Fig. 5.2. Computational space of such large size was used to demonstrate the merits of using

diagonal dominance over condition number which allowed the computation of maximum 15×15×15 cells. Whatever be the size of the computational space, both diagonal dominance and condition number provide the same observation for the FD–CN–FDTD coefficient matrix.



Medium	ϵ_s	ϵ_∞	τ (ps)	σ (S/m)
Medium 1	71.66	34.58	5.65	0.021
Medium 2	6.2	3.5	39.0	0.029
Medium 3	9.5	4.2	77.0	0.037
Medium 4	87.34	49.13	26.89	0.045
Medium 5	4.8	2.8	7.0	0.053

Figure 5.2: Computational environment for numerical studies using the FD–CN–FDTD method

Fig. 5.3 shows how the absolute values of diagonal and the sum of absolute values of off-diagonal elements of $\boldsymbol{A}$ vary with $CFLN$, both for the homogeneous and inhomogeneous cases. A representative row of $\boldsymbol{A}$ has been taken, corresponding to the interior computational space, which comprises nearly the whole coefficient matrix (except the boundary–contributed rows). For low $CFLN$ all the rows of the coefficient matrix are strictly diagonally dominant except a very few which are contributed by the boundary cases, whereas high $CFLN$ deteriorates this property. Fig. 5.3 shows that the advantageous diagonal property of the coefficient matrix is lost with increased $CFLN$ irrespective of the media parameters or homogeneity. This perfectly matches to the observation of Table 5.1 in terms of condition number. For practical problems the diagonal–dominance criterion is simpler to handle than the condition–number one because of computational expenses. An additional advantage of the diagonal dominance criterion is that it points a direction to research in the building of appropriate preconditioners to ease the solution at higher $CFLN$.

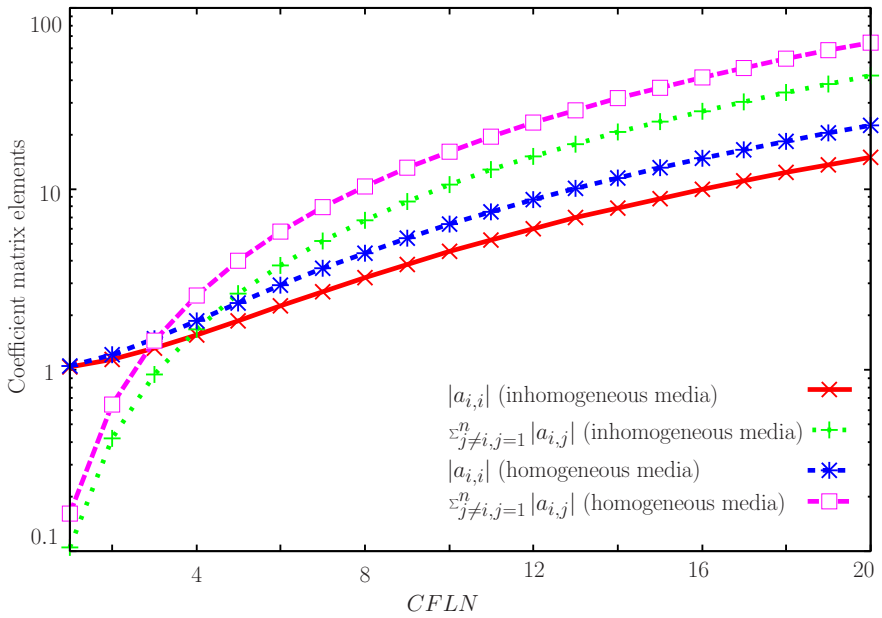


Figure 5.3: Absolute values of diagonal and sum of absolute values of off-diagonal entries of the coefficient matrix changes with $CFLN$

5.2 Direct Methods

A sparse matrix system is solved at each time step of the FD–CN–FDTD method. Methods of solution for such sparse systems fall into two categories – the *direct* and *iterative* methods. Direct solvers are extremely robust and reliable and give the exact solution if there is no rounding errors. Their latest implementations have improved the memory efficiency and have efficient reordering techniques, which improve the performance to a great extent. Out of the two types of methods, first a version of sparse Gaussian elimination based direct solver [105, 106] was investigated to solve the FD–CN–FDTD method. Following main steps are executed, in sequence, in the direct solver used in this study [107]:

1. First, reordering of the rows and the columns of the coefficient matrix are done. This is performed in such a way that the factors get little fill-in compared to the original matrix. Fill-in refers to those entries in the matrix that are initially zero but changed to a non-zero value during the operation of a certain algorithm. If possible, reordering is done to turn the coefficient matrix into special structure like block-triangular form.

2. Next, the reordered matrix is analyzed and a pivot ordering is computed in order to do symbolic factorization. The nonzero structure of the factors

are, thereby, determined and suitable data structures are formed for these factors.

3. Based on the above two steps, the coefficient matrix is decomposed in factors of lower and upper triangular matrices (LU factors)

4. After the LU factorization, forward and backward triangular sweeps are executed using the factors to obtain the solution.

Following these steps, iterative refinement are optionally performed to improve the accuracy of the solution.

The direct solver used with the FD–CN–FDTD method chooses a pivot sequence to decompose $\boldsymbol{A}$ into LU factors, in such a way that the sparsity is preserved in them. A full Markowitz search technique is used to find the best pivot and reduce the fill-ins (*i.e.* not to waste memory). At each time step of the FD–CN–FDTD algorithm, a new vector $\boldsymbol{c}$ is calculated for the right hand side, while $\boldsymbol{A}$ is required to be factorized only once (which dominates the computational time) before the beginning of the FDTD iteration[1]. Once factorized, the same factors are repeatedly used at each time step to obtain $\boldsymbol{u}$. For this reason, this method showed the potential to become more computationally efficient than the iterative methods when a large number of FDTD iterations are needed, since at each time step only forward and backward solutions are required.

Using direct solver in the FD–CN–FDTD method the same numerical test mentioned in Section 4.1 was performed. The size of the computational space was (30×30×30) cells; half of which was filled with medium 1 ($\epsilon_{\mathrm{S}} = 71.66$, $\epsilon_{\infty} = 34.58$, $\sigma = 0.49$ S/m and $\tau_{\mathrm{D}} = 5.65$ ps) and the other half with medium 2 ($\epsilon_{\mathrm{S}} = 87.34$, $\epsilon_{\infty} = 49.13$, $\sigma = 0.69$ S/m and $\tau_{\mathrm{D}} = 26.89$ ps) as shown in Fig. 5.4. Source excitation of a modulated Gaussian pulse centred at 3 GHz was placed at (10,15,15). Spatial sampling was uniform, with $\Delta x = \Delta y = \Delta z = 10^{-3}$m and $CFLN \equiv \Delta t/\Delta t_{CFL} = 1, 3, 5$ were tested, where $\Delta t_{CFL} = 1.9$ ps

[1]In this thesis, *FDTD iteration* is used to refer to the iteration (time stepping) required to complete the simulation, in order to avoid confusion with the iteration required to converge in the iterative solvers

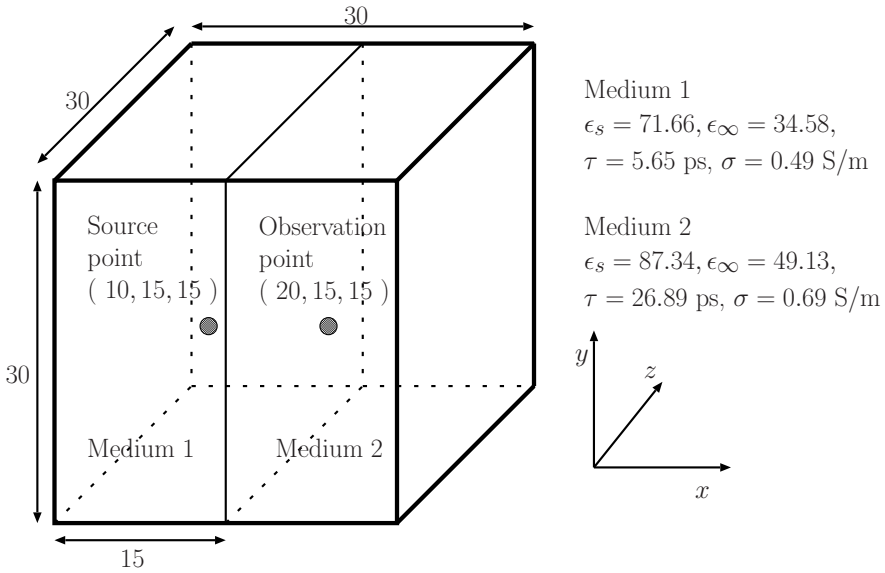


Figure 5.4: FDTD problem space for simulation with the FD–CN–FDTD method.

In this numerical test in which the sparse matrix system was solved by the direct solver, CPU time required for LU decomposition was 633 minutes and average CPU time per FDTD iteration was 6.489 seconds when $CFLN = 1$ on dual AMD Opteron 280 with 8 GB of memory. When the computational space was filled with homogeneous material of medium 1 or when $CFLN > 1$ there were no significant differences in these values (*i.e.* the CPU time for LU decomposition and average CPU time per FDTD iteration). This test was in double precision computation and required 2.4 GB of memory. With the available computational resources (National Grid Services [108], that also has a limit on the number of hours a programme can run on) it was never possible to successfully use direct solver in the FD–CN–FDTD method when the computational space was larger than 30×30×30 cells.

Despite being robust and reliable, the direct solver is not practical to use in the FD–CN–FDTD method for real applications. They are computationally more expensive and require excessively large memory (their memory requirements grow as a nonlinear function of the matrix size [109]). It might be possible to use direct solver in the FD–CN–FDTD method for larger problems by doing distributed memory parallelization (using Message Passing Interface).

5.3 Iterative Methods

Direct methods can be used to solve problems of upto 30×30×30 cells of computational space but for larger problems involving unknown variables of the order of millions the computation is prohibitively expensive and requires excessively large memory. For example, when 30×30×30 cells computational space is modelled in the FD–CN–FDTD method the direct solver using sparse Gaussian elimination requires 2.4 GB of memory whereas iterative solvers BiCGStab and GMRES, respectively, require only 62 MB and 65 MB of memory. Thus for practical problems iterative solvers have to be used [110] [104].

Iterative methods work by repeatedly improving an approximate solution until it is accurate enough [104]. To solve a linear system $\boldsymbol{Au} = \boldsymbol{c}$, iterative methods require an initial guess $\boldsymbol{u}^{(0)}$ that approximates the true solution. If no good approximation to the solution is known $\boldsymbol{u}^{(0)}$ can be taken as zero. In the iterative process $\boldsymbol{u}^{(0)}$ is used to generate a new guess $\boldsymbol{u}^{(1)}$, which is then used to generate yet another guess $\boldsymbol{u}^{(2)}$, and so on. In this way after k iterations $\boldsymbol{u}^{(k)}$ is generated. If $\boldsymbol{u}^{(k)}$ is sufficiently close to the solution, the iteration is stopped and $\boldsymbol{u}^{(k)}$ is accepted as an adequate approximation to the solution. Residual vector, $\boldsymbol{r}^{(k)} = \boldsymbol{c} - \boldsymbol{Au}^{(k)}$, is considered to decide if $\boldsymbol{u}^{(k)}$ is sufficiently close to the solution. As soon as the residual vector meets a pre-specified stopping criterion (for example, given a very small value of threshold ϵ, $\|\boldsymbol{r}^{(k)}\| \leq \epsilon$), the iterative process stops and convergence is said to be achieved. A good number of iterative methods have been developed but they mainly belong to two categories [2]:

Stationary methods: Stationary methods are the older of the iterative methods. They are simpler to understand and implement but usually not very effective. Some examples of stationary methods: Jacobi method, Gauss-Seidel method, Successive Over-Relaxation (SOR) method and Symmetric Successive Over-Relaxation (SSOR) method.

Non-stationary methods: Non-stationary methods were developed relatively recently. Although their analysis is usually harder to understand, they can be highly effective. Some examples of non-stationary methods: Conjugate Gradient (CG) method, Minimal Residual (MINRES) method, GMRES method, BiConjugate Gradient (BiCG) method, Quasi-Minimal Residual

(QMR) method, Conjugate Gradient Squared (CGS) method and BiCGStab method.

Some methods are only important to understand the historical development of iterative methods and are not relevant for solving the FD–CN–FDTD sparse matrix system. As the convergence of stationary methods is slow and guaranteed for a limited class of matrices, these are not considered in this study. Non-stationary methods that represent the current state-of-the-art for solving large sparse linear systems have been used in this study. These are also called Krylov subspace methods.

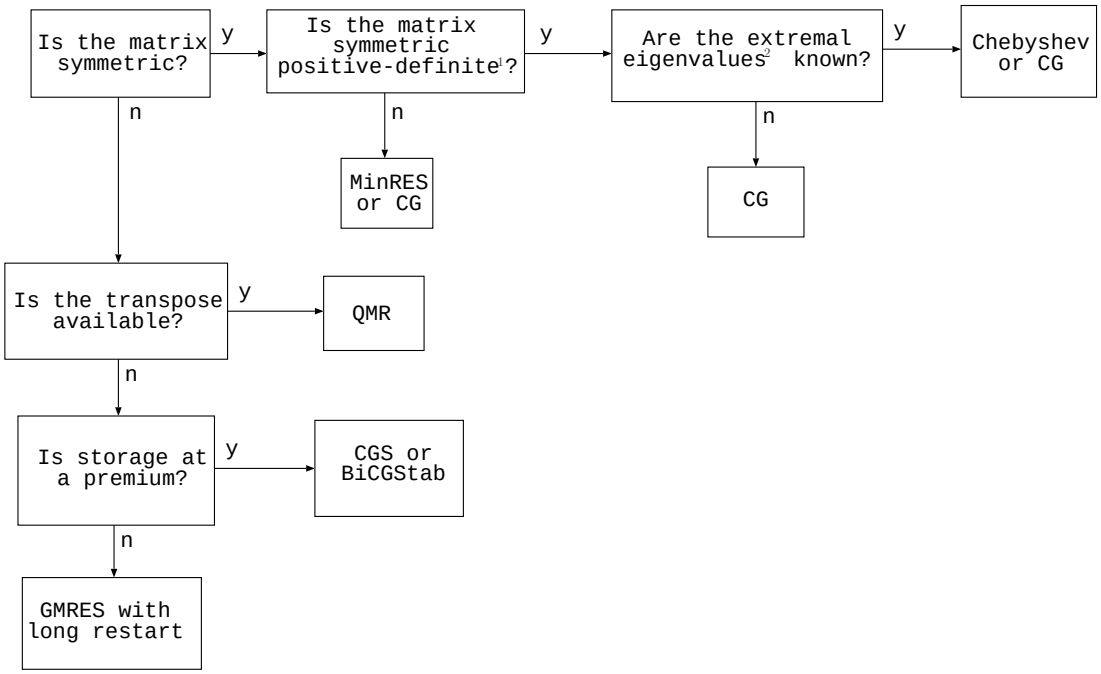


Figure 5.5: Choosing an effective iterative method [2]

Although there exists many iterative methods a certain method may work well for one problem but not for another. An iterative method might be convergent but the convergence might be too slow to be of practical value. Therefore, it is essential to find the most effective method for the concerned problem. For this, both the computations required per iteration and the number of iterations necessary for convergence need to be considered. A flowchart is suggested in the

[1]If $a_{i,j}$ is the (i,j)-th element of matrix $\boldsymbol{A}$, then $\boldsymbol{A}$ is symmetric positive definite if $a_{i,j} = a_{j,i}$ for all the i,j and if it satisfies $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all nonzero vectors $\boldsymbol{x}$

[2]Maximal and minimal eigenvalues

Appendix D of [2] as shown in Fig. 5.5 to choose an effective iterative method from the best known methods for which extensive computational experience has been gathered. Depending on the homogeneity of the computational space the FD–CN–FDTD method yields symmetric and asymmetric matrices. Since for homogeneous problems the coefficient matrix is symmetric, as shown in Fig. 5.5, iterative solvers like CG method can be used. However practical problems are inhomogeneous and iterative solvers for asymmetric matrices have to be used. These are GMRES, BiCGStab, QMR or CGS. Out of these, QMR requires transpose matrix-vector product making the computational costs per iteration higher. On the other hand, CGS suffers from irregular convergence, which may lead to substantial build-up of rounding errors because CGS algorithm is based on squaring the residual polynomial [110] [2]. Therefore, this study focuses on GMRES(m) [111] and BiCGStab [112]. GMRES is said to be a very robust solver for nonsymmetric matrices. It leads to the smallest residual for a fixed number of iteration steps. But these steps become increasingly expensive and in order to limit the increasing storage requirements and work per iteration step, restarting is necessary. It is quite difficult to chose the appropriate number of iterations m after which GMRES restarts. If m is too small GMRES(m) may be slow to converge, or fail to converge entirely. If it is unnecessarily large excessive work and more storage are incurred as penalty. BiCGStab is a fast and smoothly converging variant of BiCG method. The advantage of this method is that its computational costs per iteration are similar to that of CGS but it avoids the irregular convergence patterns of CGS while maintaining about the same rate of convergence. Also BiCGStab does not require transpose matrix-vector product like QMR.

5.3.1 Performance Study of BiCGStab and GMRES

In this section the performance of BiCGStab and GMRES, both for the homogeneous and inhomogeneous media cases, are compared. Two cases are considered for study. The first one consists of an inhomogeneous medium, in a cubic space of size 80×80×80 cells, with 5 different media as shown in Fig. 5.2. The second one involves the same cubic space of the previous case, now filled with a homogeneous medium with Debye parameters $\epsilon_{\mathrm{S}} = 6.2$, $\epsilon_{\infty} = 3.5$, $\sigma = 0.029$ S/m and $\tau_{\mathrm{D}} = 39.0$ ps. In both cases a z-directed dipole hard source with a time variation

given by a Gaussian pulse centred at 6.9 GHz was placed at the centre of the computational space. Spatial sampling was uniform: $\Delta x = \Delta y = \Delta z = \Delta s = 10^{-3}$m. The time-step is taken equal or above the CFL stability condition of the explicit FDTD: $\Delta t = CFLN \times \Delta s/(c\sqrt{3})$, where, c is the free space light–speed. The level of accuracy in waveform compared with the explicit frequency dependent FDTD is the same as the one presented in Fig. 4.2. In the iterative solution algorithms the **E** values at the previous time step are used as the initial value.

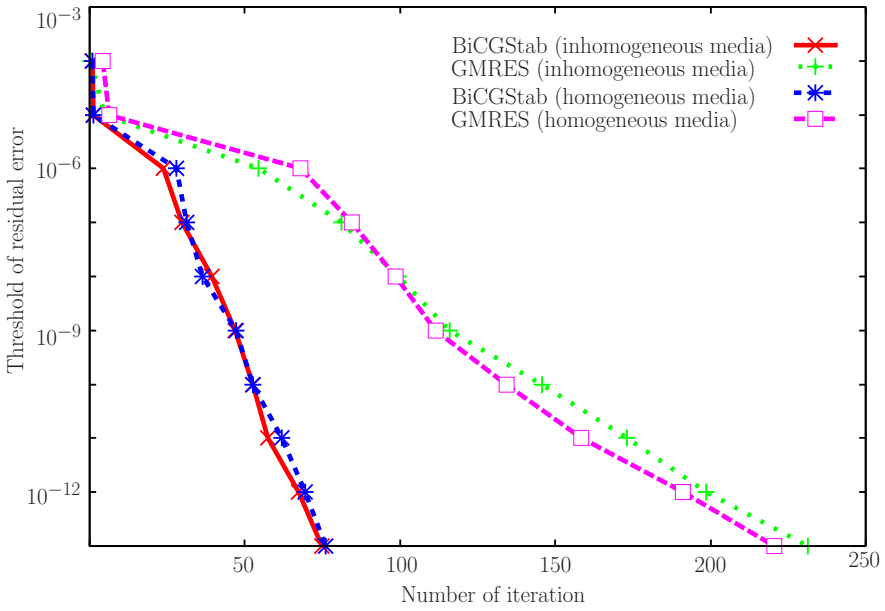


Figure 5.6: Threshold of residual error versus number of iteration required by BiCGStab and GMRES for homogeneous and inhomogeneous cases when $CFLN = 20$

Fig. 5.6 shows the convergence pattern for $CFLN = 20$, plotting the threshold of residual error as a function of the number of iterations required by the iterative solvers to converge. The number of iterations required to achieve a specified accuracy is demonstrated in this figure. For example, to make the residual error lower than 10^{-8} BiCGStab requires about 45 iterations whereas GMRES requires about 97 iterations in both homogeneous and inhomogeneous cases. The convergence rate of the solvers is weakly affected by homogeneity. In this numerical test, the value of $CFLN$ is quite high (20) and for the case of $CFLN$ having a value lower than this, the iteration numbers would certainly be lower than those shown in Figure 5.6.

Figure 5.7 shows how the average number of iterations, required by BiCGStab

and GMRES to converge, increases with the CFL number. Stopping criteria in this case was 10^{-13} and the reason for selecting this small value of convergence tolerance is, in the FD–CN–FDTD method, unlike the frequency–independent CN–FDTD method, $\mathbf{D} = \epsilon\mathbf{E}$ is used and therefore $\mathbf{D}$ can have a value of such small order because of ϵ (permittivity). GMRES stagnates when convergence tolerance is below 10^{-13} while BiCGStab can work even at a lower convergence tolerance. Both solvers require more iterations to converge as $CFLN$ goes up but the rate of increase of iteration numbers with $CFLN$ is higher for GMRES than for BiCGStab. Homogeneity does not affect significantly this rate, particularly, for BiCGStab. The change of iteration number with $CFLN$ for convergence tolerance values from 10^{-12} to 10^{-3} can be assumed from Figure 5.6. In the FD–CN–FDTD method, the total number of FDTD iterations required to complete the simulation decreases with $CFLN$, but the increase of computational costs per FDTD iteration with $CFLN$, as shown in Fig. 5.7, can undermine this positive effect unless the solution is very efficient.

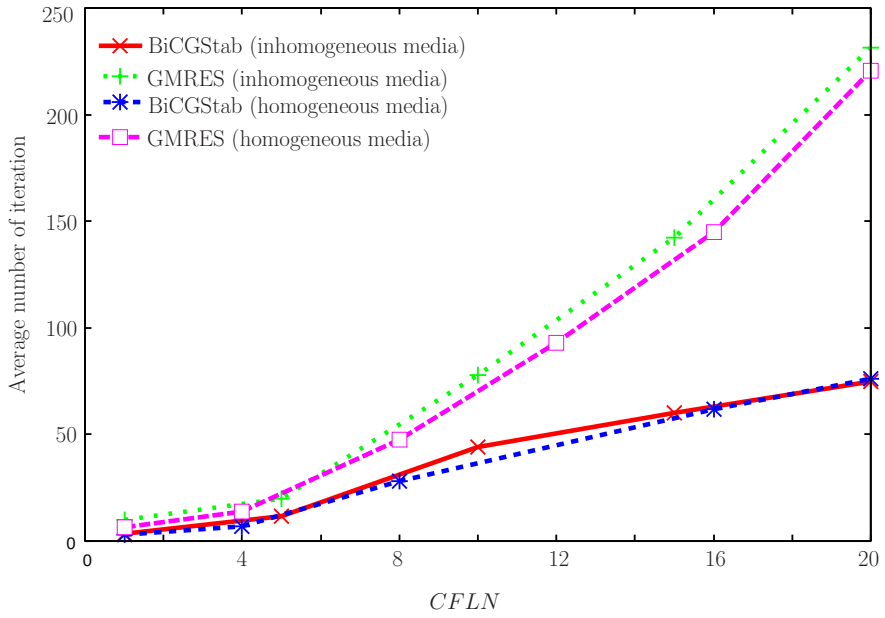


Figure 5.7: Average number of iterations required to converge at different CFL numbers (convergence tolerance = 10^{-13})

Fig. 5.8 plots the CPU time required by the FD–CN–FDTD method with BiCGStab or GMRES as a function of $CFLN$. The stopping criterion of 10^{-13}

Computational Size (cells)	40^3	60^3	80^3
BiCGStab	145MB	487MB	1.1GB
GMRES	151MB	507MB	1.2GB

Table 5.2: Memory required by BiCGStab and GMRES for different computational spaces

was used and the simulation was performed to reach a fixed time instant by letting the code run for $1200/CFLN$ time steps on a dual AMD Opteron 280 with 8GB of memory. CPU time decreases with the $CFLN$, for both solvers, although GMRES requires more CPU time than BiCGStab. The trend of the curves in Fig. 5.8 also manifests that the difference between the CPU time required by the two solvers becomes narrower with the increase of $CFLN$. Table 5.2 presents the memory required by the two solvers for three different computational space sizes. GMRES always requires more memory than BiCGStab.

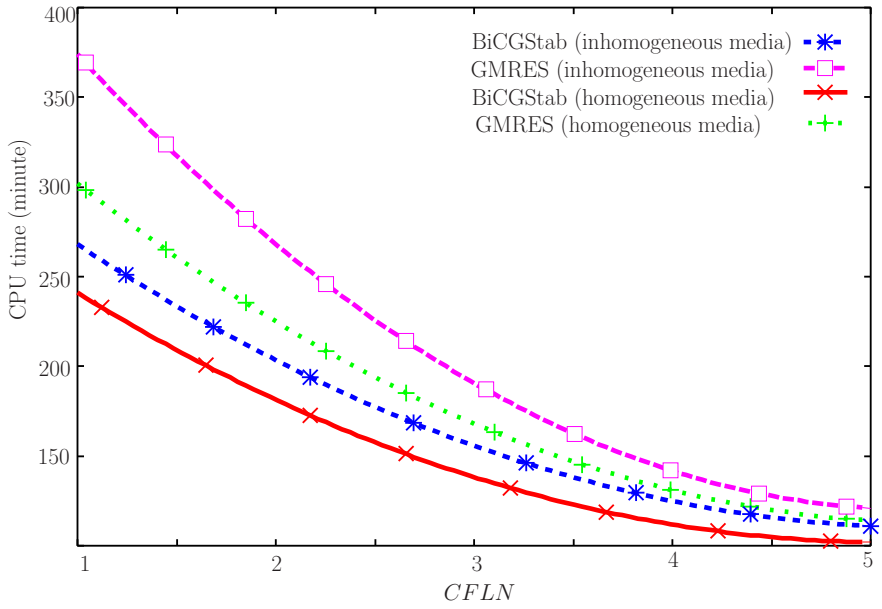


Figure 5.8: CPU time required by BiCGStab and GMRES at different $CFLN$

From all the above, it can be concluded that BiCGStab outperforms GMRES in computational efficiency. This finding is in contrary to that of [67] which reports GMRES is the fastest for the frequency–independent CN–FDTD method presented there. The work of [67] is based on (3.1) and (3.2) while the FD–CN–FDTD method additionally involves (3.10) which has second-order time derivative

terms. The FD–CN–FDTD method involves nine field components in place of six for the CN–FDTD method and the sparsity pattern of the former has more bands than the latter [68]. Apart from this, the problems concerning simulation implementation, optimization and parameters tuning have an obvious influence in concluding which solver is the most efficient.

5.4 Preconditioners

The rate of convergence of the iterative methods depends greatly on the spectrum of the coefficient matrix[1]. Often a second matrix, called a preconditioner, is used with the iterative methods to transform the coefficient matrix into one with a more favourable spectrum[2] [2]. A good preconditioner improves the convergence of the iterative method but the extra computational costs of constructing and applying the preconditioner should be minimal. Without a preconditioner the iterative method may even fail to converge. On the other hand, an inappropriate preconditioner can be counter-productive by making the convergence more difficult or giving wrong results. Finding a preconditioner suitable for a particular problem is challenging and beyond the main focus of this thesis.

To solve for the FD–CN–FDTD method two preconditioners were applied: Incomplete LU with no fill-in or ILU(0) and Sparse Approximate Inverse (SAI). ILU(0) did not give any improvement in convergence (rather convergence deteriorated). However, SAI gave slight improvement in reducing the number of iterations to converge but there were two major setbacks making it unsuitable for use. The time to compute the approximate inverse preconditioner is too large which makes the total CPU time longer than that without any preconditioner. [68] showed SAI can reduce iteration numbers for the frequency-independent CN–FDTD method but did not mention the total CPU time. A second problem is that memory requirements of SAI restricted the maximum computational space to only 30×30×30 cells. SAI also showed lack of robustness when used with the FD–CN–FDTD method.

[1]The set of the eigenvalues of a matrix is called its spectrum

[2]For example, by concentrating the spectrum of the preconditioned matrix. The ratio of the maximum and minimum eigenvalues gives a reasonable idea if a symmetric matrix is ill or well conditioned. When the spectrum of the matrix is concentrated this ratio becomes smaller, implying a relatively well conditioned matrix[2]

5.5 Conclusion

This chapter has presented Krylov–based approach to solve the FD–CN–FDTD sparse matrix system. Two best–known iterative methods, GMRES and BiCGStab, were compared in terms of the number of iteration requirements for convergence with different $CFLN$, CPU time and memory requirements. BiCGStab outperforms GMRES when used with the FD–CN–FDTD method in every aspect of the study. However there have been a lot of research on the development of preconditioners for GMRES and Fig. 5.8 indicates that GMRES tends to narrow down the difference in CPU time with BiCGStab at higher $CFLN$. Therefore, the potential of using GMRES to use in the FD–CN–FDTD method can not be ruled out. In this chapter it has also been pointed out that the degradation of diagonal–dominance of the coefficient matrix with increased $CFLN$ is a main reason for the increase of the CPU time needed by the solvers. Furthermore, it was found that ILU(0) and SAI preconditioners can not improve the computational efficiency of the FD–CN–FDTD method. Many of these findings about the frequency dependent CN–FDTD method do not match with the existing literature on the frequency–independent CN–FDTD method and possible reasons for this are also mentioned. Further work is needed to tailor suitable preconditioners to improve the iterative solver convergence.

Chapter 6

Modelling Human Body in the FD-CN-FDTD Method

With the rapid development of the wireless communications technology and the widespread use of mobile phones and other wireless devices there have been increasing public concern about the hazardous effect of electromagnetic radiation on the human body [113]. On the other hand, electromagnetic radiation can be exploited for positive purposes as well. For example, one of the fascinating applications of bioelectromagnetics is the treatment of neurological diseases using electromagnetic therapy in which certain tissues of the human body are stimulated by electromagnetic fields. Recently researches on the effects of mobile phone radiation on mice suggested that, mobile phones might protect against Alzheimer's diseases [114]. However, there are ethical restrictions on doing experiments on the living human being in order to study dosimetry or other bioelectromagnetic aspects. Therefore, numerical simulation is the possible alternative way for this. Out of many numerical methods, the FDTD method is the most suitable technique for this, because of its robustness and simplicity.

This chapter describes the application of the proposed FD–CN–FDTD method to model a real–life application. Using the FD–CN–FDTD method a simulation model of the human body is developed. Electromagnetic properties of the human tissues depend on the frequency. For accurate numerical studies of the interaction of electromagnetic waves with the human body, the FDTD method should have the ability to handle frequency dependent media. Therefore, this is an appropriate application for the proposed frequency dependent CN–FDTD method.

6.1 Numerical Modelling of the Human Body

In FDTD methods, a numerical model of the human body can be developed by appropriately discretizing the computational space (*i.e.* human body) and then assigning each discretized space, which represents the tissues and organs, its corresponding dielectric parameters. The dielectric parameters of the human tissues were experimentally measured and compiled by Gabriel *et al* in the frequency range of 10 Hz to 20 GHz [42][43][44][1]. Mathematically these parameters, which vary over the frequency, can be modelled by Debye and Cole-Cole models. This thesis uses the single–pole Debye model because of its simplicity of implementation in the FDTD method [115]. From the Gabriel's data, by using Newton's method and least square fitting technique, the dielectric parameters of the tissues for the Debye model can be obtained [115]. For all the tissues of the human body, the single–pole Debye parameters are available in our research group and are presented in Table 6.1.

A number of computational models of the human body, which mathematically represents the human anatomy, have been developed [116]. With the advancement of medical imaging technology, for example, computed tomography (CT) and magnetic resonance imaging (MRI), it has been possible to develop high-resolution computational human body models. These medical imaging technologies can provide high-resolution cross-sectional digital image of the internal anatomy of the human body [116]. When the pixel data obtained from such medical images are extended to three dimensions it gives the cuboidal representation, known as voxels. Voxels can be used to digitally represent the three-dimensional human body. Each voxel contains an uniform volume of the human body such that it can be assigned with an identifying number that corresponds to a particular tissue or organ. Such models are called voxel models or phantoms. In this thesis the whole-body voxel model of [117] which is based on the MRI data has been used. To model the human body in the FD–CN–FDTD method, the geometrical features of the human body is read from the 2-mm resolution voxel model of [117]. This voxel model consists of 320×160×866 voxels for the male and 320×160×804 voxels for the female models. However, this thesis only deals with the male model. For each voxel, that represents a certain tissue, the corresponding single–pole Debye parameters from Table 6.1 were mapped. In this way, the realistic human body with the frequency dependent dielectric parameters

Tissue	ϵ_∞	ϵ_S	τ_D (ps)	σ (S/m)
Bladder	9.6746378	19.3315	6.94441	0.30010962
Blood	30.597572	62.9005	7.00378	1.2557440
Bone Cortical	6.6990891	12.9266	10.6297	0.0686324313
Breast fat	3.0959890	5.48171	8.01535	0.0303096939
Cartilage	19.391968	44.5586	9.9898	0.49346399
Cerebellum	35.194771	58.1546	22.7492	0.82605290
Cerebrospinal fluid	33.147984	70.3996	6.05165	2.1439056
Colon	34.665340	61.1213	10.386	0.71641898
Cornea	32.372139	57.8425	9.57875	1.0688734
Fat	3.9981320	5.53071	7.8745	0.0371063352
Gall Bladder	28.180500	60.7348	5.30142	1.0422504
Bile	33.478043	72.1819	5.38703	1.5769558
Grey Matter	33.057121	56.444	11.7277	0.59515452
Heart	38.085503	65.0845	13.6874	0.77800655
Kidney	39.859943	67.5083	20.1944	0.85720307
Liver	27.985529	50.1529	11.8828	0.52030164
Nerve (Spinal Chord)	20.980206	34.7488	12.1644	0.35986480
Ovary	33.372314	59.2252	22.1513	0.79014659
Small Intestine	39.190655	65.79	15.2864	1.7013499
Spleen	37.115120	62.585	14.2244	0.84441692
Duodenum	31.503292	66.2792	6.32267	0.91899437
Tendon	16.789112	46.7146	6.88525	0.49999046
Prostate	31.306032	62.0644	7.03912	0.93375188
Thyroid	27.738815	60.5505	5.90674	0.80898720
Tongue	28.257284	56.5226	6.81527	0.69344097
Trachea	22.140263	43.1848	7.49705	0.56548506
Uterus	33.134838	63.0676	7.76051	0.96926498
Vitreous Humour	4.2372255	69.0152	2.42065	1.5027161
White Matter	24.371387	41.2808	11.1905	0.34836939
Average Brain	28.713840	48.8598	11.4971	0.47177213
Average Muscle	28.001335	56.9314	6.22154	0.74710500
Average Lung	21.436010	38.2542	9.12811	0.45216662
Average Bone	6.3775983	12.3227	10.6741	0.11563171
Average Skin	29.850496	47.9301	14.536	0.54073584

Table 6.1: Single–pole Debye parameters for the human tissues

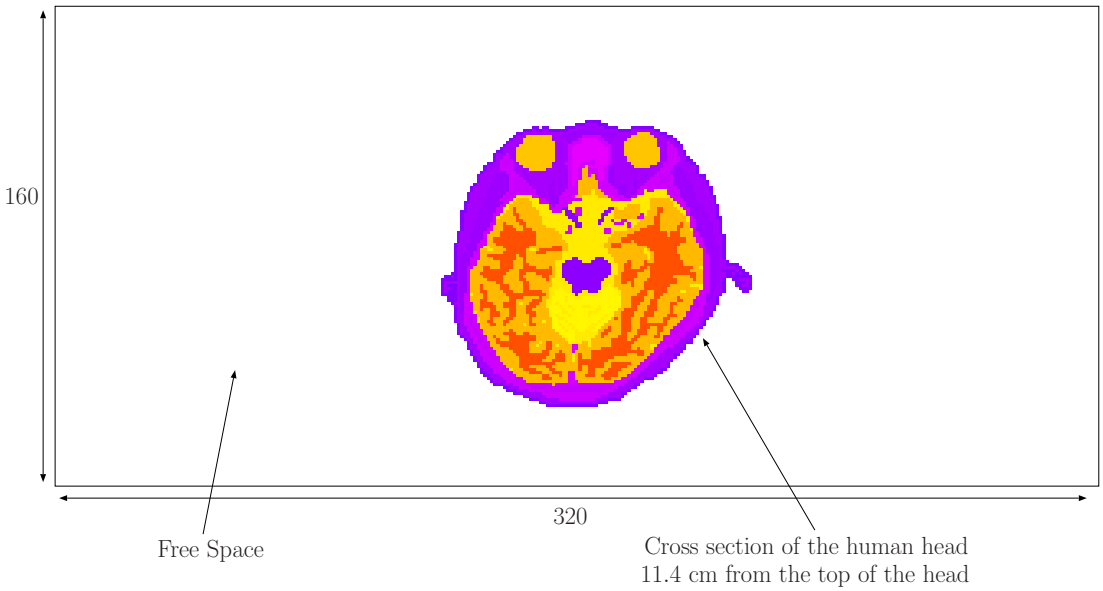


Figure 6.1: A cross section of the human head at 11.4 cm from the top of the head. The area outside the head upto the boundary is free space.

is modelled in the FD–CN–FDTD computational space. A cross-section of the human body from this model is shown in Fig. 6.1. The cross-section of Fig. 6.1 is for $z = 57$ plane which lies in the human head and has the xy dimensions of 320×160. The area outside the head upto the boundary is free space.

The numerical model of the human body as described above can be useful for several purposes. For example, to estimate the specific absorption rate (SAR), for numerical study of the bioelectromagnetic therapies and to design the body-centric wireless networks. This thesis focuses on the bioelectromagnetic therapies.

6.2 Use of the FD–CN–FDTD Method to Model Bioelectromagnetic Therapies

The application of electromagnetic fields onto the human to treat the diseases is called bioelectromagnetic therapy. Some of the treatable diseases in this way are osteoporosis, arthritis, dystonia, Alzheimer's disease, essential tremor. This thesis deals with one of the bioelectromagnetic therapies, called deep brain stimulation (DBS), which is a surgical treatment used for the treatment of neurological disorders, such as, Parkinson's disease and essential tremor. DBS uses an implanted electrode to deliver electrical stimulation to precisely targeted areas in

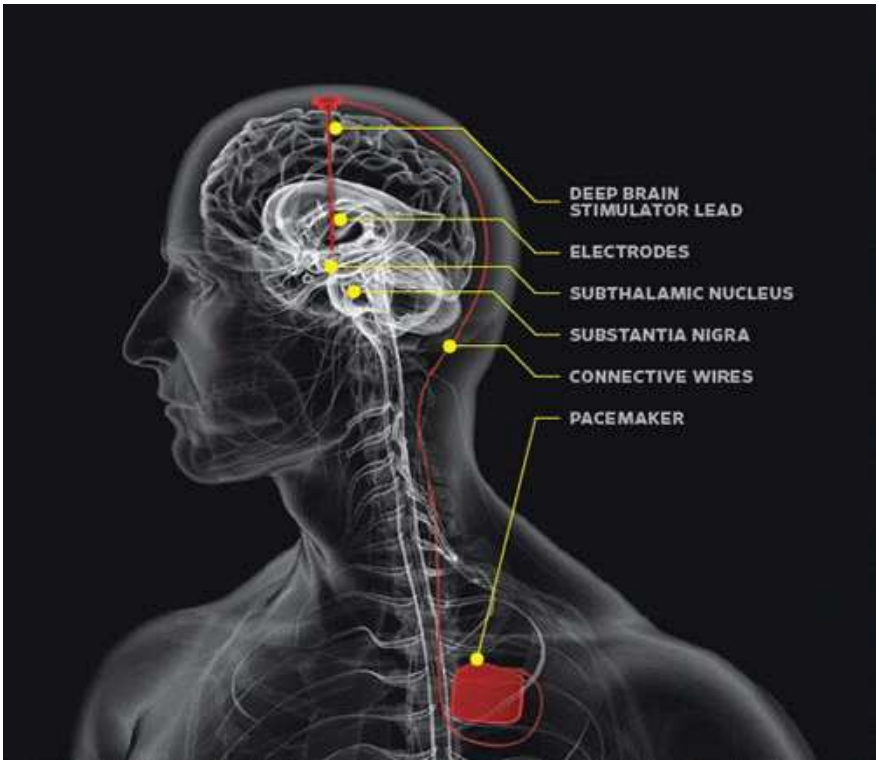


Figure 6.2: Components of a typical DBS system [1]

the brain. A typical DBS system includes three components: neurostimulator, electrodes and insulated wire extension as shown in Fig. 6.2. Neurostimulator is the power source for the DBS system which runs by a small battery and contains a programmed computer chip. It is implanted beneath the skin in the chest. The electrodes are implanted inside the brain and the wire extension connects the electrode to the neurostimulator and runs underneath the skin. DBS system sends the electrical stimulation to the targeted areas within the brain that control the movement of the body and muscle function, in order to make these areas function better. As the targeted area differs depending on the disease being treated, the location of the electrode inside the brain also varies. For Parkinson's disease the preferred site of stimulation is the subthalamus nucleus (STN), for essential tremor it is the ventro intermediate nucleus of the thalamus (Vim) and for obsessive-compulsive disorder (OCD) it is the anterior limb of the internal capsule (AIC). This thesis considers Parkinson's disease and therefore the targeted structure is always STN. STN is a small structure surrounded by several other nuclei and multiple fiber tracts as shown in Fig. 6.3.

[1] http://www.wired.com/wired/archive/15.03/images/FF_156_brain4_f.jpg (Date of download : 02/03/2010)

Although DBS can provide remarkable therapeutic benefits for neurological disorders, it has a number of potential risks and side effects because it requires to place the implants invasively inside the brain. Some of these risks and side effects include: paralysis, coma and even death, brain stroke, seizures, infection, allergic response to the materials of the implants, confusion and attention problems, pain at the surgery sites, double vision, dizziness, headache. Also any such invasive electromagnetic therapy can be unethical if large experiential knowledge is not available. Even if experiential knowledge is available, accurate prediction of excitation of the targeted tissues inside the brain is important.

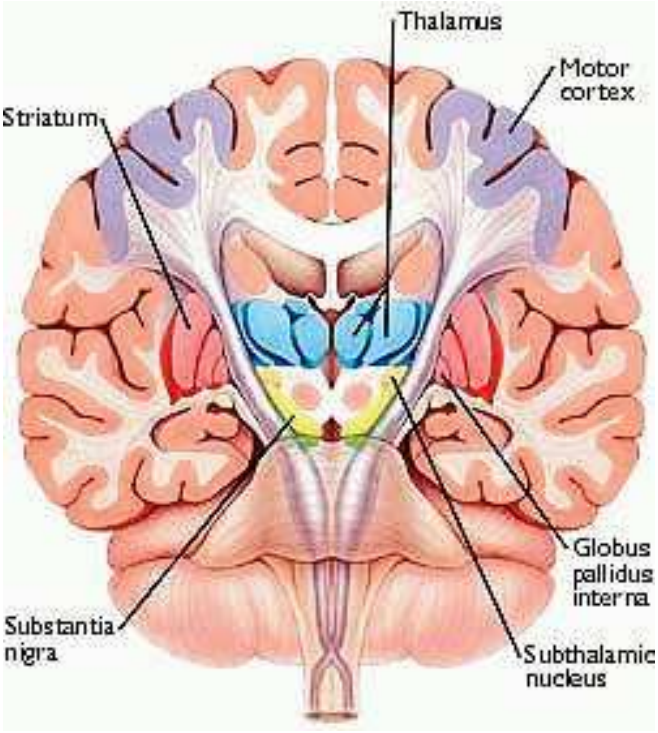


Figure 6.3: Location of the subthalamic nucleus (STN) inside the human head [1]

As a precursor of the future research, in this thesis, using the FD–CN–FDTD method the DBS system is modelled simplistically to suggest FD–CN–FDTD's suitability for bioelectromagnetics research. There are two aims of this research. First, to accurately predict the excitation of the targeted tissues inside the brain. Such predictions will help the clinicians better understand the mechanisms of a bioelectromagnetic treatment procedure, its limitations and implications from

[1] http://www.lloydtan-trust.com/index.php?page=living_sub&type=causation (Date of download : 02/03/2010)

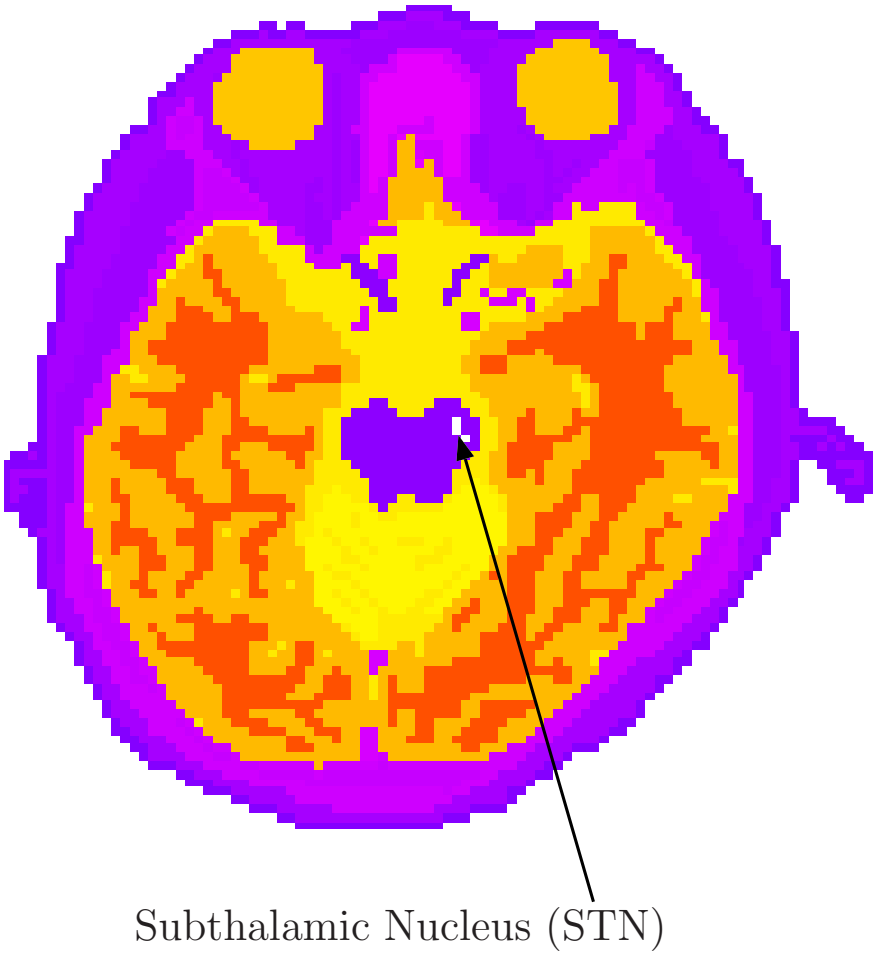


Figure 6.4: Location of the STN, inside the human head model, used in the numerical simulation

an electrical point of view as well as make the appropriate parameters set-up required for an effective treatment. Second and the ultimate objective is to develop a technique to overcome the invasiveness nature of the DBS procedure, so that the possible risk factors and side effects can be avoided. With these aims in mind, electrical field distribution inside the human head is studied by using the numerical human body model in the FD–CN–FDTD method.

In order to model simplified DBS system, it is sufficient to consider the area from the top of the head up to the section above the upper chest in the numerical human body model. Therefore, the size of the FD–CN–FDTD computational

space was 320×160×220 voxels from the top of the head (220×2mm = 44cm). By careful calculation, the location of STN (Fig. 6.3) was spotted inside the human head model at (169,86,57), (169,87,57), (170,88,57) points as shown in Fig. 6.4. In order to resemble the stimulation of STN in the DBS procedure, source excitations of z–directed modulated Gaussian pulse centred at 3GHz were placed at these points. Then the simulation was run for $1000/CFLN$ time steps with $CFLN = 1$ and 3. Observations were taken at several points around the head. This thesis puts forward the idea that, if source excitations based on these observations are applied at these points around the head, by using some algorithms it would be possible to stimulate the targeted STN. Twenty five observation points were chosen around the head with nearly equal distance from one another, as shown in Fig. 6.5. The locations of these twenty five points are given in Table 6.2.

It should be pointed out that, the location of the STN is on the $z = 57$ plane and all the twenty five observation points also lie on the $z = 57$ plane. Observed signals at the twenty five locations are shown in Fig. 6.6 for $CFLN = 1$ and 3. Signals for $CFLN = 3$ agree well with the corresponding signals for $CFLN = 1$. The amount of CPU time saved when $CFLN = 3$ is used instead of $CFLN = 1$ is shown in Table 8.4 in Chapter 8. Fig. 6.6 shows that the observed signals are phase shifted from one observation location to the next as expected. Because of the heterogeneities of the tissues of the human head, the amplitude of these signals vary a lot from one observation location to the next. The time delays or phase shifts in both cases of $CFLN = 1$ and $CFLN = 3$ follow the same pattern. For example, both $CFLN = 1$ and $CFLN = 3$ show conspicuous change of phase from loc-12 onward and from loc-18 onward in Fig. 6.6.

Electrical field distributions at different time steps on the $z = 57$ plane in the human head model are shown in Fig. 6.7 and Fig. 6.8 for $CFLN = 1$ and 3, respectively. The electrical field distributions shown in Fig. 6.7 and Fig. 6.8 include the fields in the free space surrounding the head (Fig. 6.1). To produce Fig. 6.7 and Fig. 6.8, absolute values of the electrical fields were used. Therefore, the minimum values of the electrical field distributions are zero at all the time steps. The changes of electrical field distributions over time when $CFLN = 1$ match quite well with those when $CFLN = 3$.

Observation point	*Location in the numerical model*
loc-1	$(135, 51, 57)$
loc-2	$(130, 57, 57)$
loc-3	$(127, 64, 57)$
loc-4	$(124, 78, 57)$
loc-5	$(123, 85, 57)$
loc-6	$(119, 92, 57)$
loc-7	$(122, 98, 57)$
loc-8	$(124, 106, 57)$
loc-9	$(127, 113, 57)$
loc-10	$(130, 120, 57)$
loc-11	$(138, 127, 57)$
loc-12	$(146, 134, 57)$
loc-13	$(160, 134, 57)$
loc-14	$(175, 134, 57)$
loc-15	$(180, 127, 57)$
loc-16	$(190, 120, 57)$
loc-17	$(195, 113, 57)$
loc-18	$(201, 106, 57)$
loc-19	$(205, 98, 57)$
loc-20	$(207, 92, 57)$
loc-21	$(208, 85, 57)$
loc-22	$(207, 78, 57)$
loc-23	$(206, 71, 57)$
loc-24	$(203, 61, 57)$
loc-25	$(197, 51, 57)$

Table 6.2: Locations of the twenty five observation points around the human head model

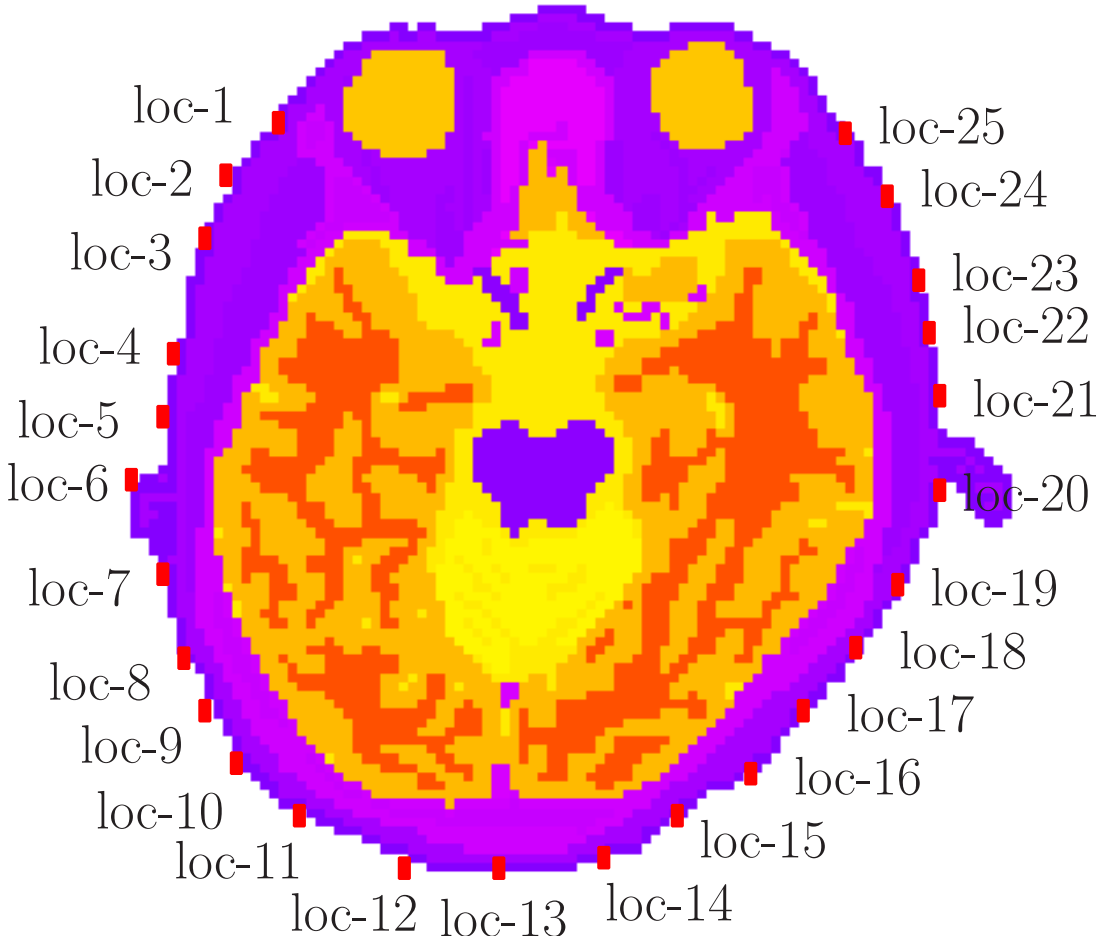
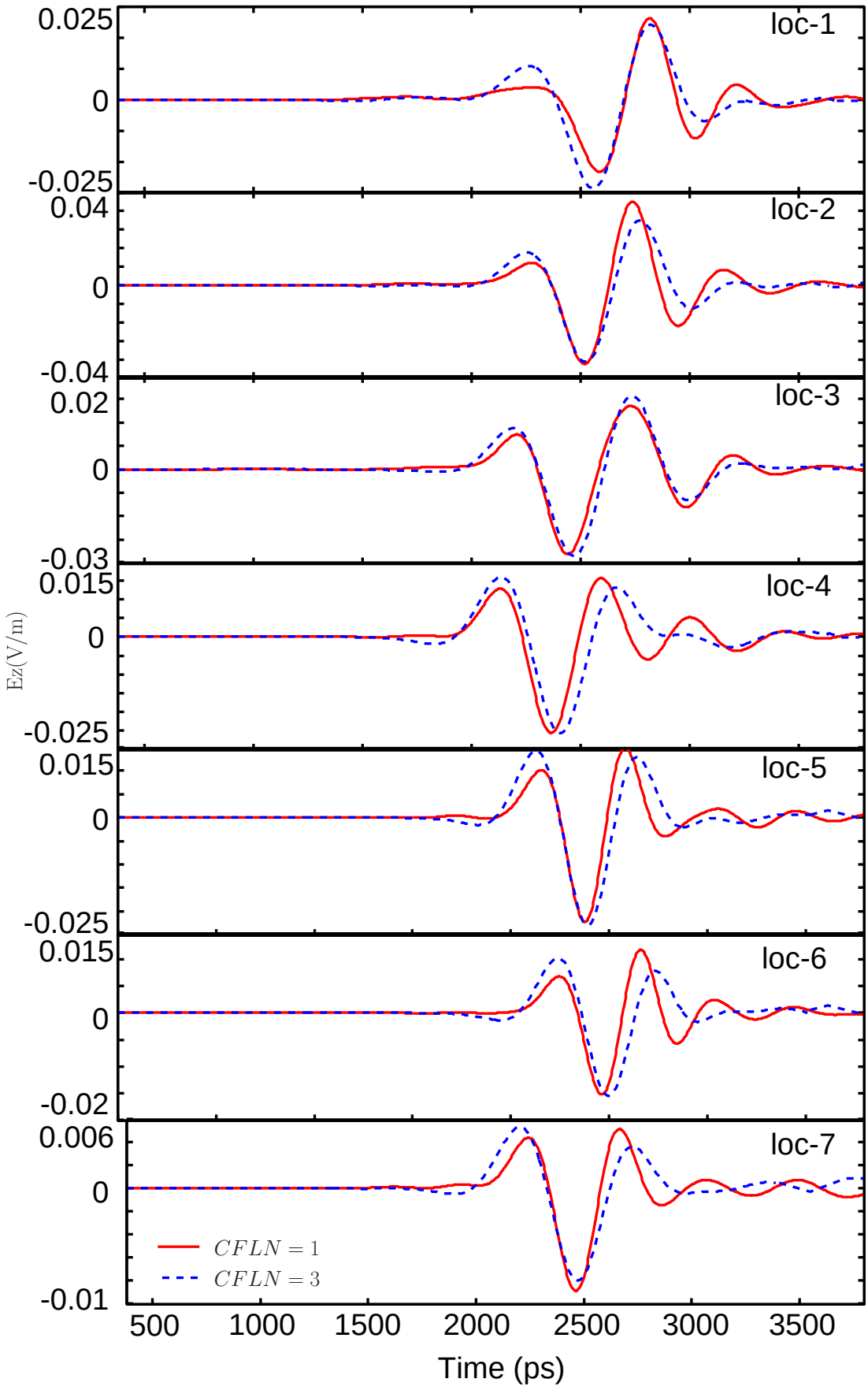
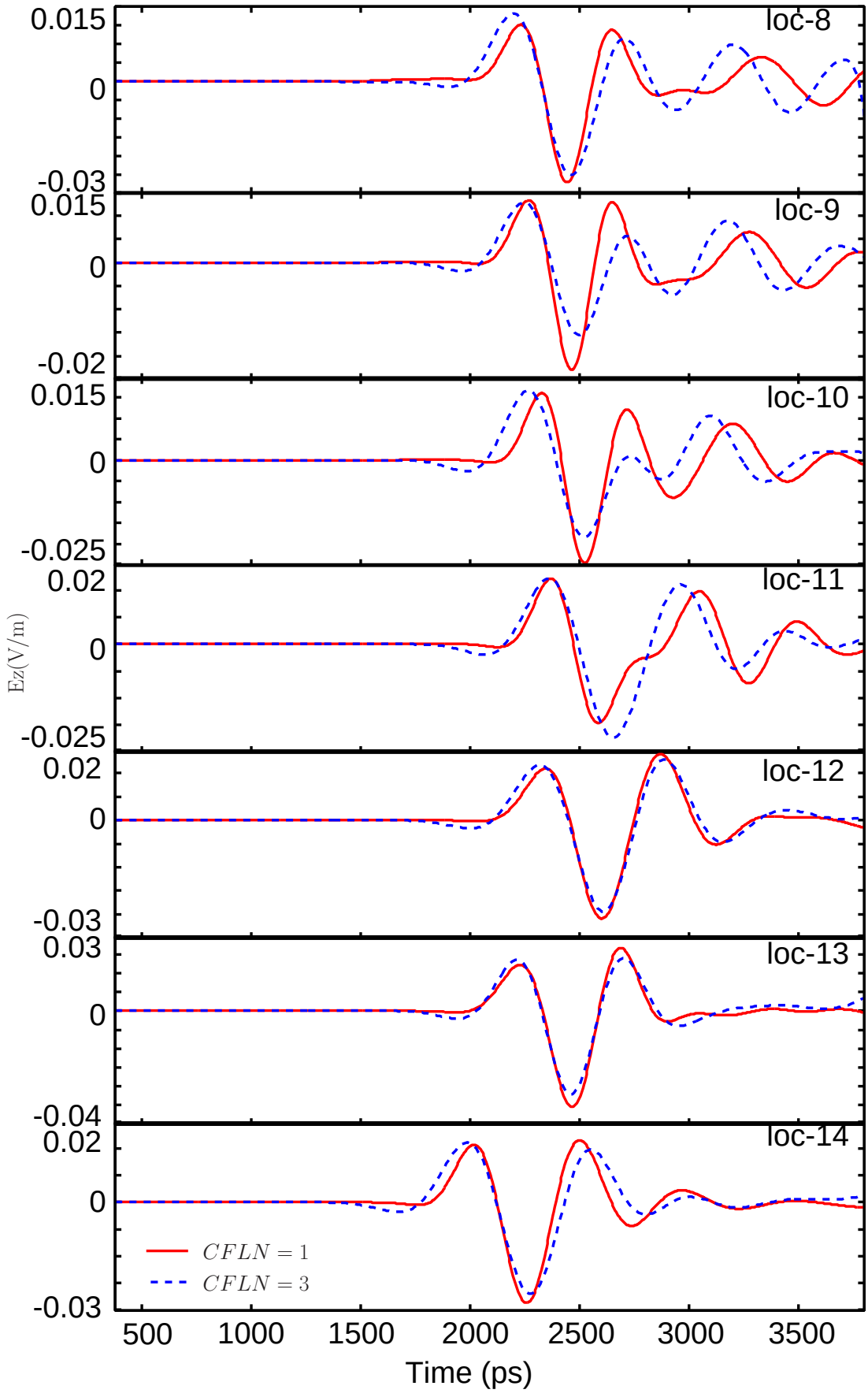
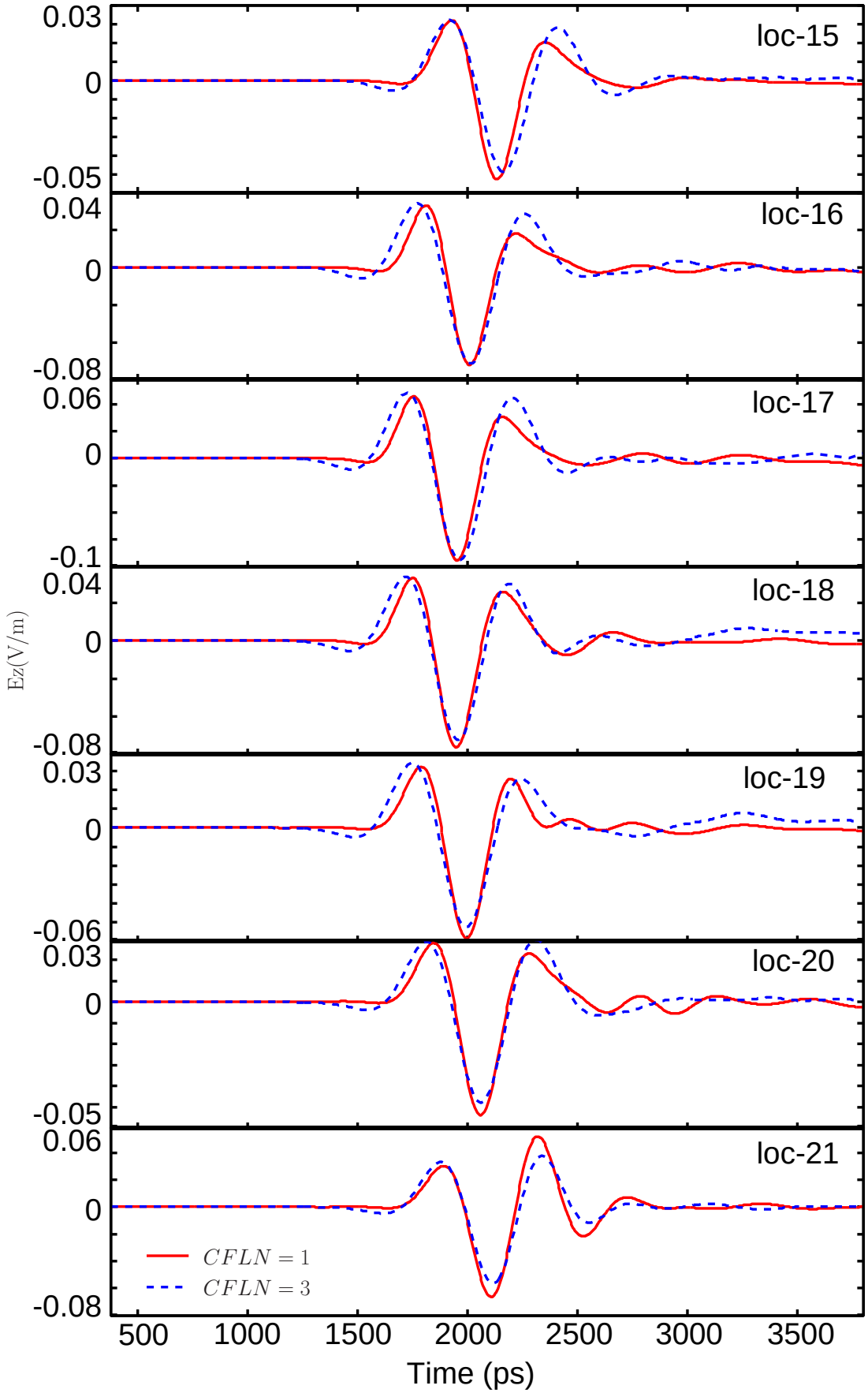


Figure 6.5: Locations of the twenty five observation points around the human head model

From these observed signals around the head, using FDTD–based time reversal algorithms as reported in [118], the excitation at the STN can be reproduced. For this further research on the backpropagation and target localization by FDTD time reversal algorithms is required. Using such algorithms, if appropriate electromagnetic wave excitations are applied at the appropriate locations around the head it would be possible to pinpoint the target location inside the brain, which at present is stimulated by inserting DBS electrodes. With such non-invasive procedure, there will be no associated risk factors and side effects of invasive DBS system.


loc-1
loc-2
loc-3
loc-4
loc-5
loc-6
loc-7
0.025
0
-0.025
0.04
0
-0.04
0.02
0
-0.03
0.015
0
-0.025
0.015
0
-0.025
0.015
0
-0.02
0.006
0
-0.01
Ez(V/m)
CFLN = 1
CFLN = 3
500
1000
1500
2000
2500
3000
3500
Time (ps)


loc-8
loc-9
loc-10
loc-11
loc-12
loc-13
loc-14
0.015
0
-0.03
0.015
0
-0.02
0.015
0
-0.025
0.02
0
-0.025
0.02
0
-0.03
0.03
0
-0.04
0.02
0
-0.03
Ez(V/m)
CFLN = 1
CFLN = 3
500
1000
1500
2000
2500
3000
3500
Time (ps)


loc-15
loc-16
loc-17
loc-18
loc-19
loc-20
loc-21
0.03
0
-0.05
0.04
0
-0.08
0.06
0
-0.1
0.04
0
-0.08
0.03
0
-0.06
0.03
0
-0.05
0.06
0
-0.08
Ez(V/m)
CFLN = 1
CFLN = 3
500
1000
1500
2000
2500
3000
3500
Time (ps)

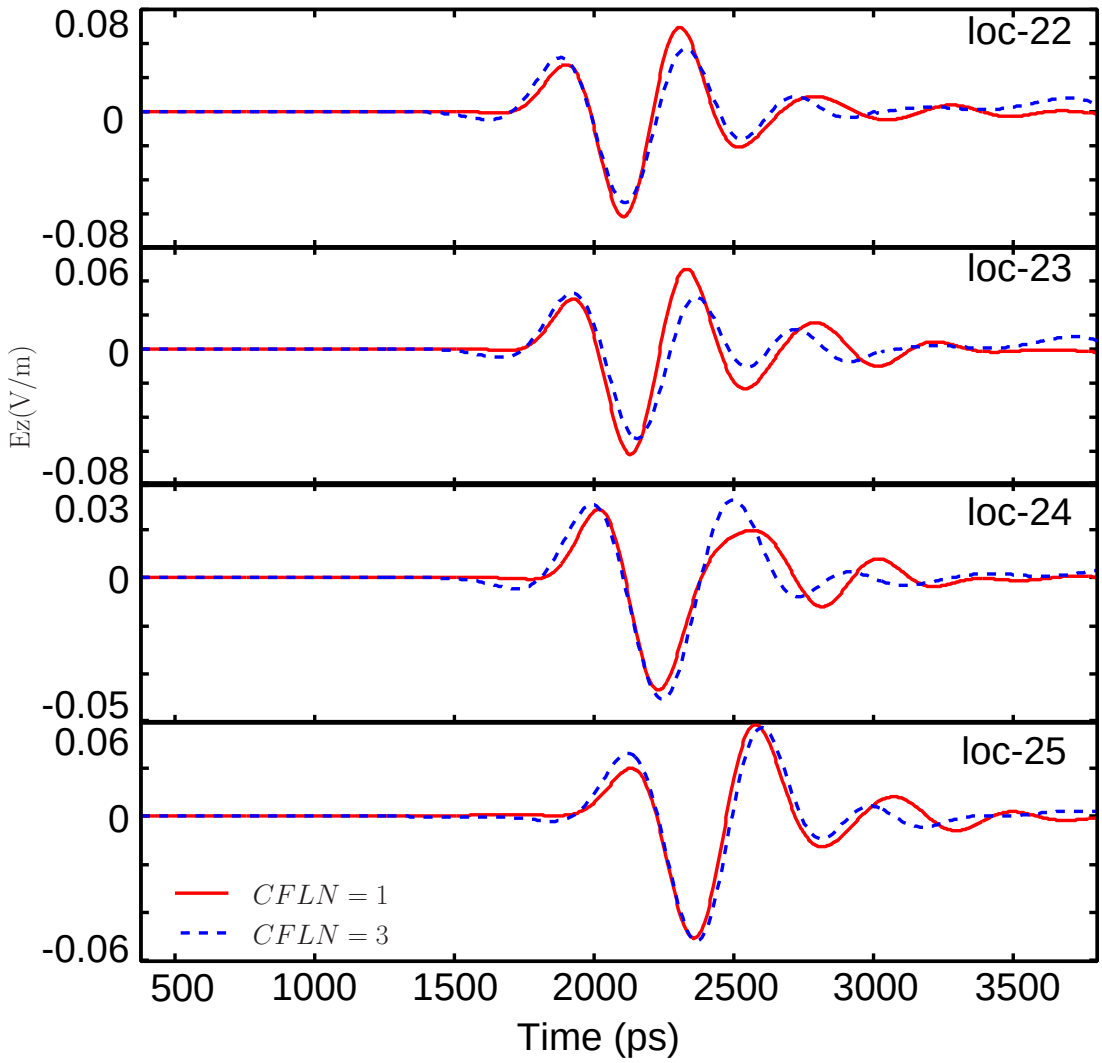


Figure 6.6: Observed signals at the selected twenty five locations around the human head when $CFLN = 1$ and 3.

6.3 Conclusion

In this chapter, an application of the proposed FD–CN–FDTD method has been described. With the FD–CN–FDTD method the human body has been modelled with all their fine structures and frequency dependent dielectric properties. Numerical simulation of electromagnetic wave propagation inside the human head has been shown. The implications of this study for further research on bioelectromagnetic therapies, such as DBS, using the FD–CN–FDTD method has also been mentioned.

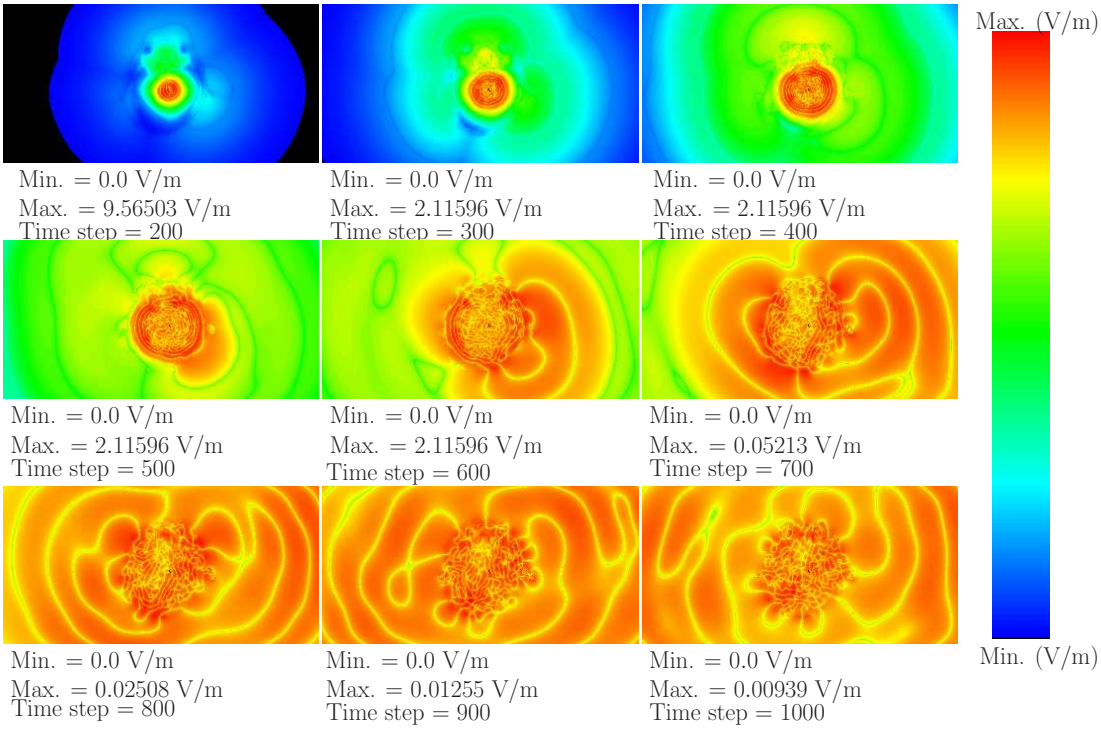


Figure 6.7: Electrical field distributions at different time steps of the numerical simulation on the $z = 57$ plane within the human head model ($CFLN = 1$)

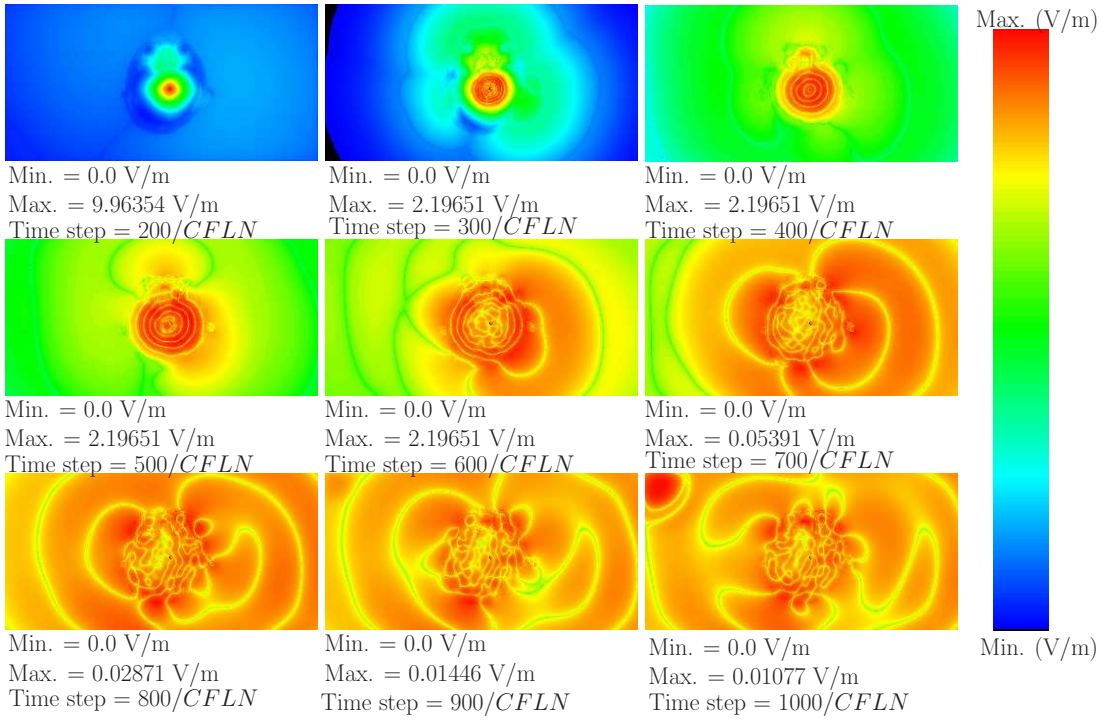


Figure 6.8: Electrical field distributions at different time steps of the numerical simulation on the $z = 57$ plane within the human head model ($CFLN = 3$)

Chapter 7

Modified Frequency Dependent ADI–FDTD Method

In this chapter a new method modifying the frequency dependent alternating direction implicit finite difference time domain (FD–ADI–FDTD) is presented. The method can improve the accuracy of the FD–ADI–FDTD method without significant increase of computational costs. The derivation of the modified method is presented in Section 7.2 followed by its validation by numerical experiments.

7.1 Limitations of the FD–ADI–FDTD Method

A major breakthrough in FDTD research has been the development of the alternating direction implicit (ADI)–FDTD method [17], since it removes the bound on the time-step of conventional FDTD method imposed by CFL stability condition. Although Crank Nicolson (CN)–FDTD method [96][119] has similar capability, the advantage of ADI–FDTD method over CN–FDTD method is that the computational overhead is smaller because tridiagonal matrix systems are required to solve (rather than sparse matrix systems). In fact, the ADI–FDTD method is a computationally affordable approximation of the CN–FDTD method [23][21], found by adding a perturbation term to the latter. This term permits to split the fully implicit step advancing from n to $n+1$ in CN–FDTD method, into two tridiagonally implicit substeps in ADI–FDTD method going from n to $n+1/2$ and from $n+1/2$ to $n+1$. The ADI–FDTD method exhibits a loss of accuracy with respect to the CN-FDTD method that may become severe for some practical applications, especially when large time-steps are used [23].

In summary, the ADI–FDTD method improves the computational efficiency at the cost of accuracy, while the CN–FDTD method exhibits higher accuracy at the cost of computational costs. Frequency dependent ADI–FDTD and CN–FDTD methods also have the same characteristics. There have been efforts to improve the accuracy of ADI-FDTD method. For instance, the approach of [56] is based on an iterative procedure ideally converging to the CN-FDTD method. Since this approach involves loop of iterations at each time steps, it is computationally more expensive than normal ADI–FDTD method. An alternative solution is given by [57] employing an average approximation of some of the implicit fields. Although both of these techniques are stable in 2D, their generalization to 3D seems to become unstable [59][58].

All these researches to improve the ADI–FDTD method considered frequency independent material. However the inclusion of the frequency dependency of material parameters in FDTD methods is essential for the simulation of broadband systems. Formulations of the ADI–FDTD method to handle the frequency dependent features of the media have been studied using differential equation method in [91] and using Z-transform method in [120]. In this thesis, the approach by [57] is extended for the frequency dependent ADI–FDTD method of [91].

7.2 Modified Frequency Dependent ADI–FDTD Method

From Maxwell's equation: $\nabla \times \mathbf{E} = -\mu \dfrac{\partial \mathbf{H}}{\partial t}$ and $\nabla \times \mathbf{H} = \dfrac{\partial \mathbf{D}}{\partial t} + \sigma \mathbf{E}$, the equations for 2D TM polarization are:

$$\frac{\partial D_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma E_z \tag{7.1}$$

$$\frac{\partial H_x}{\partial t} = -\frac{1}{\mu}\frac{\partial E_z}{\partial y} \tag{7.2}$$

$$\frac{\partial H_y}{\partial t} = \frac{1}{\mu}\frac{\partial E_z}{\partial x} \tag{7.3}$$

In modified FD–ADI–FDTD method, frequency dependent media have been modelled by the single–pole Debye relationship $\mathbf{D} = \epsilon_0\epsilon_r\mathbf{E}$ with $\epsilon_r = \epsilon_\infty + \frac{\epsilon_S - \epsilon_\infty}{1 + \jmath\omega\tau_D}$. The frequency domain constitutive relationship for this medium can be translated into time domain using an auxiliary differential formulation [87]

$$\tau_D \frac{\partial D_z}{\partial t} - \tau_D \epsilon_0 \epsilon_\infty \frac{\partial E_z}{\partial t} = \epsilon_0 \epsilon_S E_z - D_z \tag{7.4}$$

Putting (7.1) into (7.4) time derivative of E_z is obtained:

$$\frac{\partial E_z}{\partial t} = -(\frac{\epsilon_S}{\tau_D \epsilon_\infty} + \frac{\sigma}{\epsilon_0 \epsilon_\infty})E_z + \frac{1}{\tau_D \epsilon_0 \epsilon_\infty} D_z + \frac{1}{\epsilon_0 \epsilon_\infty}\frac{\partial H_y}{\partial x} - \frac{1}{\epsilon_0 \epsilon_\infty}\frac{\partial H_x}{\partial y} \tag{7.5}$$

(7.1), (7.2), (7.3), (7.5) can be written as

$$\frac{\partial V}{\partial t} = PV + QV \tag{7.6}$$

where

$$P = \begin{pmatrix} 0 & \frac{1}{\tau_D \epsilon_0 \epsilon_\infty} & 0 & \frac{1}{\epsilon_0 \epsilon_\infty}\frac{\partial}{\partial x} \\ 0 & 0 & 0 & \frac{\partial}{\partial x} \\ 0 & 0 & 0 & 0 \\ \frac{1}{\mu}\frac{\partial}{\partial x} & 0 & 0 & 0 \end{pmatrix} \tag{7.7}$$

$$Q = \begin{pmatrix} -(\frac{\epsilon_S}{\tau_D \epsilon_\infty} + \frac{\sigma}{\epsilon_0 \epsilon_\infty}) & 0 & -\frac{1}{\epsilon_0 \epsilon_\infty}\frac{\partial}{\partial y} & 0 \\ -\sigma & 0 & -\frac{\partial}{\partial y} & 0 \\ -\frac{1}{\mu}\frac{\partial}{\partial y} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.8}$$

$$V = \begin{bmatrix} E_z & D_z & H_x & H_y \end{bmatrix}^T \tag{7.9}$$

Now using finite differences for the time derivative and averaging the fields over time in (7.6), the CN–FDTD method is found

$$\frac{V^{n+1}-V^n}{\Delta t} = (P+Q)\frac{V^{n+1}+V^n}{2}$$

$$\therefore\ V^{n+1}-V^n = (\frac{\Delta t}{2}P + \frac{\Delta t}{2}Q)(V^{n+1}+V^n)$$

$$\therefore\ V^{n+1} - V^{n+1}(\frac{\Delta t}{2}P + \frac{\Delta t}{2}Q) = V^n + V^n(\frac{\Delta t}{2}P + \frac{\Delta t}{2}Q)$$

$$\therefore\ (I - \frac{\Delta t}{2}P - \frac{\Delta t}{2}Q)V^{n+1} = (I + \frac{\Delta t}{2}P + \frac{\Delta t}{2}Q)V^n \tag{7.10}$$

where I is a 4×4 identity matrix. The CN–FDTD equation of (7.10) can also be written as

$$(I - \frac{\Delta t}{2}P)(I - \frac{\Delta t}{2}Q)V^{n+1} = \tag{7.11}$$
$$(I + \frac{\Delta t}{2}P)(I + \frac{\Delta t}{2}Q)V^n + \frac{\Delta t^2}{4}PQ(V^{n+1}-V^n)$$

In the ADI–FDTD method the last term of (7.11) is dropped and then solved in two steps leading to truncation error which is a function of $(\frac{\Delta t}{2})^2$ times the space derivatives of the field [23]. This means that the truncation error increases with larger Δt, thus imposing a restriction on Δt, particularly when accuracy of the problem is crucial for strong–gradient fields. Instead, following the strategy of [57], using the Peaceman-Rachford algorithm [25], the CN–FDTD equation of (7.11) can be written in a two–step implicit form, without dropping any term:

Step-1

$$(I - \frac{\Delta t}{2}P)V^{n+\frac{1}{2}} = (I + \frac{\Delta t}{2}Q)V^n + \frac{\Delta t^2}{8}PQ(V^{n+1}-V^n) \tag{7.12}$$

Step-2

$$(I - \frac{\Delta t}{2}Q)V^{n+1} = (I + \frac{\Delta t}{2}P)V^{n+\frac{1}{2}} + \frac{\Delta t^2}{8}PQ(V^{n+1} - V^n) \tag{7.13}$$

In (7.12) and (7.13) the last term of (7.11) is halved at each half–step. By starting with (7.13) and then using (7.12) it can be shown that the pair of equations (7.12) and (7.13) are equivalent to (7.11) in the following way:

$$\begin{aligned}
&(I - \frac{\Delta t}{2}P)(I - \frac{\Delta t}{2}Q)V^{n+1} \\
&= (I + \frac{\Delta t}{2}P)(I - \frac{\Delta t}{2}P)V^{n+\frac{1}{2}} + \frac{\Delta t^2}{8}PQ(I - \frac{\Delta t}{2}P)(V^{n+1} - V^n) \\
&= (I + \frac{\Delta t}{2}P)\Big((I + \frac{\Delta t}{2}Q)V^n + \frac{\Delta t^2}{8}PQ(V^{n+1} - V^n)\Big) \\
&\quad + \frac{\Delta t^2}{8}PQ(I - \frac{\Delta t}{2}P)(V^{n+1} - V^n) \\
&= (I + \frac{\Delta t}{2}P)(I + \frac{\Delta t}{2}Q)V^n + \frac{\Delta t^2}{8}PQ(I + \frac{\Delta t}{2}P)(V^{n+1} - V^n) \\
&\quad + \frac{\Delta t^2}{8}PQ(I - \frac{\Delta t}{2}P)(V^{n+1} - V^n) \\
&= (I + \frac{\Delta t}{2}P)(I + \frac{\Delta t}{2}Q)V^n + \frac{\Delta t^2}{4}PQ(V^{n+1} - V^n)
\end{aligned}$$

The right hand sides of (7.12) and (7.13) have $(n+1)$ terms that need to be updated. This gives rise to the issue of finding an effective iterative procedure. To avoid this problem, from the approximations:

$$V^n = \frac{V^{n+\frac{1}{2}} + V^{n-\frac{1}{2}}}{2} \text{ and } V^{n+\frac{1}{2}} = \frac{V^{n+1} + V^n}{2}$$

V^{n+1} and $V^{n+\frac{1}{2}}$ can be obtained by extrapolation [57] as

$$V^{n+1} = 2V^{n+\frac{1}{2}} - V^n \tag{7.14}$$

$$V^{n+\frac{1}{2}} = 2V^n - V^{n-\frac{1}{2}} \tag{7.15}$$

Using (7.14) in the last term of either (7.12) or (7.13) gives

$$\frac{\Delta t^2}{8}PQ(V^{n+1}-V^n)=\frac{\Delta t^2}{4}PQ(V^{n+\frac{1}{2}}-V^n) \tag{7.16}$$

Again putting (7.15) into (7.16) gives

$$\frac{\Delta t^2}{8}PQ(V^{n+1}-V^n)=\frac{\Delta t^2}{4}PQ(V^{n}-V^{n-\frac{1}{2}}) \tag{7.17}$$

Use of (7.16) and (7.17) in (7.12) and (7.13), respectively, yields a couple tridiagonally implicit set of equations

Step-1

$$(I-\frac{\Delta t}{2}P)V^{n+\frac{1}{2}}=(I+\frac{\Delta t}{2}Q)V^n+\frac{\Delta t^2}{4}PQ(V^{n}-V^{n-\frac{1}{2}}) \tag{7.18}$$

Step-2

$$(I-\frac{\Delta t}{2}Q)V^{n+1}=(I+\frac{\Delta t}{2}P)V^{n+\frac{1}{2}}+\frac{\Delta t^2}{4}PQ(V^{n+\frac{1}{2}}-V^{n}) \tag{7.19}$$

In (7.18) and (7.19), unlike normal ADI–FDTD method, the last term is not omitted. The existence of this last term reduces the truncation error. This approach is explicit in nature and therefore simpler than the iterative ADI method of (7.12) and (7.13).

First half-step

Now replacing (7.7), (7.8) and (7.9) into (7.18), the first substep of equations are found

$$\begin{aligned}{E_z}^{n+\frac{1}{2}}-\frac{\Delta t}{2\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{D_z}^{n+\frac{1}{2}}-\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} \\ =[1+\frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty}+\frac{\sigma}{\epsilon_0\epsilon_\infty})]{E_z}^{n}+\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_x}^{n}}{\partial y} \\ -(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big({E_z}^{n}-{E_z}^{n-\frac{1}{2}}\big)-(\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big(\frac{\partial {H_x}^{n}}{\partial y}-\frac{\partial {H_x}^{n-\frac{1}{2}}}{\partial y}\big)\end{aligned} \tag{7.20}$$

$$D_z{}^{n+\frac{1}{2}} - \frac{\Delta t}{2}\frac{\partial H_y{}^{n+\frac{1}{2}}}{\partial x} = -\frac{\Delta t}{2}\sigma E_z{}^{n} + D_z{}^{n} + \frac{\Delta t}{2}\frac{\partial H_x{}^{n}}{\partial y} \tag{7.21}$$

$$H_x{}^{n+\frac{1}{2}} = \frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial E_z{}^{n}}{\partial y} + H_x{}^{n} \tag{7.22}$$

$$\begin{aligned} -\frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial E_z{}^{n+\frac{1}{2}}}{\partial x} + H_y{}^{n+\frac{1}{2}} = H_y{}^{n} \\ -(\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\big(\frac{\partial E_z{}^{n}}{\partial x} - \frac{\partial E_z{}^{n-\frac{1}{2}}}{\partial x}\big) \\ -(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\big(\frac{\partial^2 H_x{}^{n}}{\partial x\partial y} - \frac{\partial^2 H_x{}^{n-\frac{1}{2}}}{\partial x\partial y}\big) \end{aligned} \tag{7.23}$$

Substituting $D_z{}^{n+\frac{1}{2}}$ from (7.21) into (7.20) gives

$$\begin{aligned} E_z{}^{n+\frac{1}{2}} - \frac{\Delta t}{2\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\left(\frac{\Delta t}{2}\frac{\partial H_y{}^{n+\frac{1}{2}}}{\partial x} - \frac{\Delta t}{2}\sigma E_z{}^{n} + D_z{}^{n} + \frac{\Delta t}{2}\frac{\partial H_x{}^{n}}{\partial y}\right) \\ -\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial H_y{}^{n+\frac{1}{2}}}{\partial x} = [1 + \frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})]E_z{}^{n} + \frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial H_x{}^{n}}{\partial y} \\ -(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big(E_z{}^{n} - E_z{}^{n-\frac{1}{2}}\big) - (\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big(\frac{\partial H_x{}^{n}}{\partial y} - \frac{\partial H_x{}^{n-\frac{1}{2}}}{\partial y}\big) \end{aligned} \tag{7.24}$$

Putting ${E_z}^{n+\frac{1}{2}}$ from (7.24) into (7.23) results in

$$
\begin{aligned}
{H_y}^{n+\frac{1}{2}} - \Big[(\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big]\frac{\partial^2 {H_y}^{n+\frac{1}{2}}}{\partial x^2} \qquad (7.25)\\
= \frac{\partial {E_z}^n}{\partial x}\Big\{\frac{\Delta t}{2}\frac{1}{\mu}[1 + \frac{\Delta t}{2}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})] - (\frac{\Delta t}{2})^2\frac{\sigma}{\mu} - (\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\Big\}\\
+\frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {D_z}^n}{\partial x} + \frac{\partial^2 {H_x}^n}{\partial x\partial y}\Big\{(\frac{\Delta t}{2})^2\frac{1}{\mu} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty} - (\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\Big\}\\
+(\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\frac{\partial {E_z}^{n-\frac{1}{2}}}{\partial x} + (\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\frac{\partial^2 {H_x}^{n-\frac{1}{2}}}{\partial x\partial y} +\\
{H_y}^n - (\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\big(\frac{\partial {E_z}^n}{\partial x} - \frac{\partial {E_z}^{n-\frac{1}{2}}}{\partial x}\big)\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\big(\frac{\partial^2 {H_x}^n}{\partial x\partial y} - \frac{\partial^2 {H_x}^{n-\frac{1}{2}}}{\partial x\partial y}\big)
\end{aligned}
$$

Factoring out the same derivatives (7.25) simplifies into

$$
\begin{aligned}
{H_y}^{n+\frac{1}{2}} - \Big[(\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big]\frac{\partial^2 {H_y}^{n+\frac{1}{2}}}{\partial x^2} \qquad (7.26)\\
= \frac{\partial {E_z}^n}{\partial x}\Big\{\frac{\Delta t}{2}\frac{1}{\mu}[1 + \frac{\Delta t}{2}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})]\\
-(\frac{\Delta t}{2})^2\frac{\sigma}{\mu} - (\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} - (\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big\}\\
+\frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {D_z}^n}{\partial x} + \frac{\partial^2 {H_x}^n}{\partial x\partial y}\Big\{(\frac{\Delta t}{2})^2\frac{1}{\mu} - (\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\Big\}\\
+\frac{\partial {E_z}^{n-\frac{1}{2}}}{\partial x}\Big\{(\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big\}\\
+\frac{\partial^2 {H_x}^{n-\frac{1}{2}}}{\partial x\partial y}\Big\{(\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big\} + {H_y}^n
\end{aligned}
$$

After discretization (7.26) becomes

$$
\begin{aligned}
H_y^{n+\frac{1}{2}}{}_{(i,j)} - \Big[(\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big] \qquad (7.27)\\
\frac{1}{\Delta x^2}\Big\{H_y^{n+\frac{1}{2}}{}_{(i+1,j)} - 2H_y^{n+\frac{1}{2}}{}_{(i,j)} + H_y^{n+\frac{1}{2}}{}_{(i-1,j)}\Big\}\\
= \frac{1}{\Delta x}\Big\{\frac{\Delta t}{2}\frac{1}{\mu}[1 + \frac{\Delta t}{2}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})] - (\frac{\Delta t}{2})^2\frac{\sigma}{\mu} - (\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big\}\Big\{E_z^n{}_{(i,j)} - E_z^n{}_{(i-1,j)}\Big\} + \frac{\Delta t}{2}\frac{1}{\mu}\frac{1}{\Delta x}\Big\{D_z^n{}_{(i,j)} - D_z^n{}_{(i-1,j)}\Big\}\\
+\frac{1}{\Delta x\Delta y}\Big\{(\frac{\Delta t}{2})^2\frac{1}{\mu} - (\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty}\Big\}\Big\{H_x^n{}_{(i,j+1)} - H_x^n{}_{(i,j)} - H_x^n{}_{(i-1,j+1)} + H_x^n{}_{(i-1,j)}\Big\}\\
+\frac{1}{\Delta x}\Big\{(\frac{\Delta t}{2})^3\frac{\sigma}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\rm S}}{\tau_{\rm D}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big\}\\
\Big\{E_z^{n-\frac{1}{2}}{}_{(i,j)} - E_z^{n-\frac{1}{2}}{}_{(i-1,j)}\Big\} + \frac{1}{\Delta x\Delta y}\Big\{(\frac{\Delta t}{2})^3\frac{1}{\mu\tau_{\rm D}\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big\}\\
\Big\{H_x^{n-\frac{1}{2}}{}_{(i,j+1)} - H_x^{n-\frac{1}{2}}{}_{(i,j)} - H_x^{n-\frac{1}{2}}{}_{(i-1,j+1)} + H_x^{n-\frac{1}{2}}{}_{(i-1,j)}\Big\} + H_y^n{}_{(i,j)}
\end{aligned}
$$

In the modified FD–ADI–FDTD method Mur's first-order absorbing boundary conditions [28] (equations in Appendix B) are used. (7.27) does not hold when $i = i_{min}$ or $i = i_{max}$ (hereafter, i_{min} and i_{max} refer to the lower and upper boundaries along the x direction, j_{min} and j_{max} refer to those along the y direction). When $i = i_{min}$ and $i = i_{max}$, for $H_y{}^{n+\frac{1}{2}}$ (Figure 7.1) (7.28) and (7.29), respectively, are used. These equations come from Mur's ABC.

$$
\begin{aligned}
H_y^{n+\frac{1}{2}}{}_{(i,j)} - \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j)}})}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}H_y^{n+\frac{1}{2}}{}_{(i+1,j)} \qquad (7.28)\\
= \frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j)}})\,H_y^n{}_{(i+1,j)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j)}})\,H_y^n{}_{(i,j)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}
\end{aligned}
$$

$$
\begin{aligned}
H_y^{n+\frac{1}{2}}{}_{(i,j)} - \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i-1,j)}})}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}H_y^{n+\frac{1}{2}}{}_{(i-1,j)} \qquad (7.29)\\
= \frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i-1,j)}})\,H_y^n{}_{(i-1,j)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j)}})\,H_y^n{}_{(i,j)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}
\end{aligned}
$$

Now (7.27), (7.28) and (7.29) can be applied to all the grid points of the computational space. This will form a system of linear equations of $\boldsymbol{Au} = \boldsymbol{c}$, where

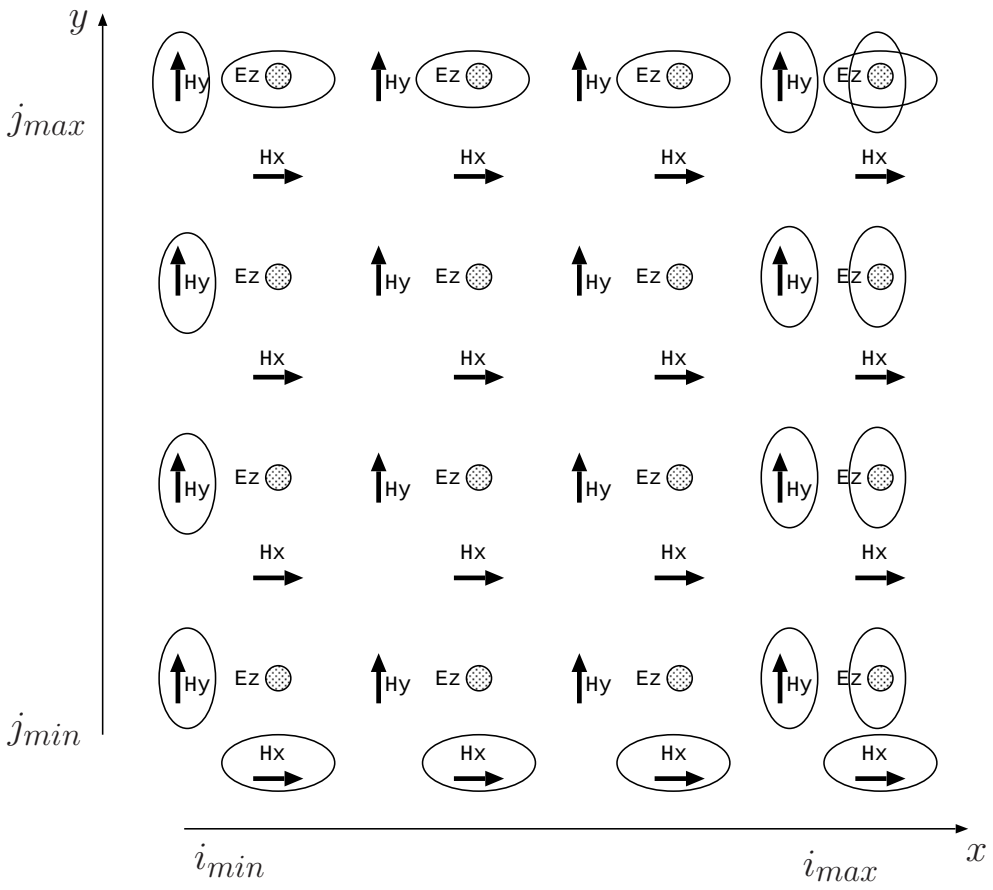


Figure 7.1: Locations on the grid of the boundary values of **E** and **H** that are calculated using the Mur's ABCs (first half-step). Only those boundary values that are used in the modified FD–ADI–FDTD equations are calculated.

$\boldsymbol{A}$ is a tridiagonal matrix, $\boldsymbol{u}$ is unknown field vector ${H_y}^{n+\frac{1}{2}}$ and $\boldsymbol{c}$ is the excitation vector. The solution of this tridiagonal system of equations provides the values of H_y at half-step. Then using ${H_y}^{n+\frac{1}{2}}$ in (7.21) ${D_z}^{n+\frac{1}{2}}$ is solved explicitly:

$$ {D_z}^{n+\frac{1}{2}} = \frac{\Delta t}{2}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} - \frac{\Delta t}{2}\sigma {E_z}^{n} + {D_z}^{n} + \frac{\Delta t}{2}\frac{\partial {H_x}^{n}}{\partial y} \tag{7.30} $$

The discretization of (7.30) gives

$$ \begin{aligned} D_z^{n+\frac{1}{2}}{}_{(i,j)} &= \frac{\Delta t}{2}\frac{1}{\Delta x}\Big\{H_y^{n+\frac{1}{2}}{}_{(i+1,j)} - H_y^{n+\frac{1}{2}}{}_{(i,j)}\Big\} \\ -\frac{\Delta t}{2}\sigma E_z^n{}_{(i,j)} + D_z^n{}_{(i,j)} &+ \frac{\Delta t}{2}\frac{1}{\Delta y}\Big\{H_x^n{}_{(i,j+1)} - H_x^n{}_{(i,j)}\Big\} \end{aligned} \tag{7.31} $$

(7.31) can not be used when $i = i_{max}$ and $j = j_{max}$. However, $D_z(i_{max})$ and $D_z(j_{max})$ are not required elsewhere in the scheme and therefore, are not calculated. Using ${D_z}^{n+\frac{1}{2}}$ and ${H_y}^{n+\frac{1}{2}}$ in (7.20) ${E_z}^{n+\frac{1}{2}}$ is explicitly found:

$$\begin{aligned} {E_z}^{n+\frac{1}{2}} &= \frac{\Delta t}{2\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{D_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} \\ &+[1+\frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty}+\frac{\sigma}{\epsilon_0\epsilon_\infty})]{E_z}^{n} + \frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_x}^{n}}{\partial y} \\ -(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big({E_z}^{n}-{E_z}^{n-\frac{1}{2}}\big) &- (\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big(\frac{\partial {H_x}^{n}}{\partial y}-\frac{\partial {H_x}^{n-\frac{1}{2}}}{\partial y}\big) \end{aligned} \tag{7.32}$$

The discretization of (7.32) gives

$$\begin{aligned} E_z^{n+\frac{1}{2}}{}_{(i,j)} &= \frac{\Delta t}{2\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}D_z^{n+\frac{1}{2}}{}_{(i,j)} \\ +\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{1}{\Delta x}\left\{H_y^{n+\frac{1}{2}}{}_{(i+1,j)}-H_y^{n+\frac{1}{2}}{}_{(i,j)}\right\} &+ [1+\frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty}+\frac{\sigma}{\epsilon_0\epsilon_\infty})]E_z^n{}_{(i,j)} \\ +\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{1}{\Delta y}\left\{H_x^n{}_{(i,j+1)}-H_x^n{}_{(i,j)}\right\} &- (\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\big(E_z^n{}_{(i,j)}-E_z^{n-\frac{1}{2}}{}_{(i,j)}\big) \\ -(\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\Big\{\frac{1}{\Delta y}\left\{H_x^n{}_{(i,j+1)}-H_x^n{}_{(i,j)}\right\} &- \frac{1}{\Delta y}\left\{H_x^{n-\frac{1}{2}}{}_{(i,j+1)}-H_x^{n-\frac{1}{2}}{}_{(i,j)}\right\}\Big\} \end{aligned} \tag{7.33}$$

(7.33) can not be used when $i = i_{max}$ and $j = j_{max}$. Therefore, $E_z(i_{max})$ and $E_z(j_{max})$ (Figure 7.1) are determined by Mur's ABC using (7.34) and (7.35), respectively:

$$\begin{aligned} E_z^{n+\frac{1}{2}}{}_{(i,j)} &= \frac{(\Delta t-\Delta x\sqrt{\mu\epsilon_{(i-1,j)}})}{\Delta t+\Delta x\sqrt{\mu\epsilon_{(i,j)}}}E_z^{n+\frac{1}{2}}{}_{(i-1,j)} \\ +\frac{(\Delta t+\Delta x\sqrt{\mu\epsilon_{(i-1,j)}})\,E_z^n{}_{(i-1,j)}-(\Delta t-\Delta x\sqrt{\mu\epsilon_{(i,j)}})\,E_z^n{}_{(i,j)}}{\Delta t+\Delta x\sqrt{\mu\epsilon_{(i,j)}}} \end{aligned} \tag{7.34}$$

$$\begin{aligned} E_z^{n+\frac{1}{2}}{}_{(i,j)} &= \frac{(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j-1)}})}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}}E_z^{n+\frac{1}{2}}{}_{(i,j-1)} \\ +\frac{(\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j-1)}})\,E_z^n{}_{(i,j-1)}-(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j)}})\,E_z^n{}_{(i,j)}}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}} \end{aligned} \tag{7.35}$$

Finally ${H_x}^{n+\frac{1}{2}}$ is obtained using (7.22)

$$ {H_x}^{n+\frac{1}{2}} = \frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {E_z}^n}{\partial y} + {H_x}^n \tag{7.36} $$

(7.36) is discretized to (7.37)

$$ H_x^{n+\frac{1}{2}}{}_{(i,j)} = \frac{\Delta t}{2}\frac{1}{\mu}\frac{1}{\Delta y}\Big\{E_z^n{}_{(i,j)} - E_z^n{}_{(i,j-1)}\Big\} + H_x^n{}_{(i,j)} \tag{7.37} $$

(7.37) can be used for all the grid points except for $j = j_{min}$. $H_x(j_{min})$ (Figure 7.1) is calculated using (7.38) which comes from Mur's ABC:

$$ H_x^{n+\frac{1}{2}}{}_{(i,j)} = \frac{(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1)}})}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j)}}} H_x^{n+\frac{1}{2}}{}_{(i,j+1)} \tag{7.38} $$
$$ + \frac{(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1)}})\, H_x^n{}_{(i,j+1)} - (\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j)}})\, H_x^n{}_{(i,j)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j)}}} $$

All the equations to calculate ${E_z}^{n+\frac{1}{2}}$, ${D_z}^{n+\frac{1}{2}}$, ${H_x}^{n+\frac{1}{2}}$ and ${H_y}^{n+\frac{1}{2}}$ for the first half-step have been derived above.

Second half-step

Next, using (7.7), (7.8) and (7.9) into (7.19) the second substep of equations is obtained:

$$ [1 + \frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})]{E_z}^{n+1} + \frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_x}^{n+1}}{\partial y} \tag{7.39} $$
$$ = {E_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{D_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} $$
$$ -(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\Big({E_z}^{n+\frac{1}{2}} - {E_z}^n\Big) - (\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\Big(\frac{\partial {H_x}^{n+\frac{1}{2}}}{\partial y} - \frac{\partial {H_x}^n}{\partial y}\Big) $$

$$ \frac{\Delta t}{2}\sigma {E_z}^{n+1} + {D_z}^{n+1} + \frac{\Delta t}{2}\frac{\partial {H_x}^{n+1}}{\partial y} = {D_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} \tag{7.40} $$

$$\frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {E_z}^{n+1}}{\partial y} + {H_x}^{n+1} = {H_x}^{n+\frac{1}{2}} \tag{7.41}$$

$$\begin{aligned} {H_y}^{n+1} &= \frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {E_z}^{n+\frac{1}{2}}}{\partial x} + {H_y}^{n+\frac{1}{2}} \\ -(\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big(\frac{\partial {E_z}^{n+\frac{1}{2}}}{\partial x} - \frac{\partial {E_z}^{n}}{\partial x}\Big) \\ -(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big(\frac{\partial^2 {H_x}^{n+\frac{1}{2}}}{\partial x\partial y} - \frac{\partial^2 {H_x}^{n}}{\partial x\partial y}\Big) \end{aligned} \tag{7.42}$$

Substitution of ${H_x}^{n+1}$ from (7.41) into (7.39) gives

$$\begin{aligned} &[1 + \frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})]{E_z}^{n+1} - (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\frac{\partial^2 {E_z}^{n+1}}{\partial y^2} \\ &= -\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_x}^{n+\frac{1}{2}}}{\partial y} + {E_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{D_z}^{n+\frac{1}{2}} \\ &+\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} - (\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\Big({E_z}^{n+\frac{1}{2}} - {E_z}^{n}\Big) \\ &-(\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\Big(\frac{\partial {H_x}^{n+\frac{1}{2}}}{\partial y} - \frac{\partial {H_x}^{n}}{\partial y}\Big) \end{aligned} \tag{7.43}$$

After factoring out the coefficients properly, (7.43) can be re-written as

$$\begin{aligned} &[1 + \frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})]{E_z}^{n+1} - (\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\frac{\partial^2 {E_z}^{n+1}}{\partial y^2} \\ &= -\left\{\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty} + (\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\right\}\frac{\partial {H_x}^{n+\frac{1}{2}}}{\partial y} \\ &+\left\{1 - (\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\right\}{E_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{D_z}^{n+\frac{1}{2}} \\ &+\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_\infty}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} + (\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}{E_z}^{n} + (\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_\infty}\frac{\partial {H_x}^{n}}{\partial y} \end{aligned} \tag{7.44}$$

(7.44) is discretized as

$$
\begin{aligned}
&[1+\frac{\Delta t}{2}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_{\infty}}+\frac{\sigma}{\epsilon_0\epsilon_{\infty}})]{E_z}^{n+1}{}_{(i,j)} \qquad (7.45)\\
&-(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_{\infty}}\frac{1}{\Delta y^2}\left\{{E_z}^{n+1}{}_{(i,j+1)}-2{E_z}^{n+1}{}_{(i,j)}+{E_z}^{n+1}{}_{(i,j-1)}\right\}\\
=&-\left\{\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_{\infty}}+(\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_{\infty}}\right\}\frac{1}{\Delta y}\left\{{H_x}^{n+\frac{1}{2}}{}_{(i,j+1)}-{H_x}^{n+\frac{1}{2}}{}_{(i,j)}\right\}\\
&+\left\{1-(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_{\infty}}\right\}{E_z}^{n+\frac{1}{2}}{}_{(i,j)}+\frac{\Delta t}{2}\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_{\infty}}{D_z}^{n+\frac{1}{2}}{}_{(i,j)}\\
&+\frac{\Delta t}{2}\frac{1}{\epsilon_0\epsilon_{\infty}}\frac{1}{\Delta x}\left\{{H_y}^{n+\frac{1}{2}}{}_{(i+1,j)}-{H_y}^{n+\frac{1}{2}}{}_{(i,j)}\right\}\\
&+(\frac{\Delta t}{2})^2\frac{\sigma}{\tau_{\mathrm{D}}\epsilon_0\epsilon_{\infty}}{E_z}^{n}{}_{(i,j)}+(\frac{\Delta t}{2})^2\frac{1}{\tau_{\mathrm{D}}\epsilon_0\epsilon_{\infty}}\frac{1}{\Delta y}\left\{{H_x}^{n}{}_{(i,j+1)}-{H_x}^{n}{}_{(i,j)}\right\}
\end{aligned}
$$

(7.45) can not be applied to the computational grids when $j = j_{min}$ and $j = j_{max}$. Following Mur's ABC, (7.46) and (7.47) are used for ${E_z}^{n+1}$ (Figure 7.2) when $j = j_{min}$ and $j = j_{max}$, respectively:

$$
\begin{aligned}
&{E_z}^{n+1}{}_{(i,j)}-\frac{(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j+1)}})}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}}{E_z}^{n+1}{}_{(i,j+1)} \qquad (7.46)\\
=&\frac{(\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j+1)}})\,{E_z}^{n}{}_{(i,j+1)}-(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j)}})\,{E_z}^{n}{}_{(i,j)}}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}}
\end{aligned}
$$

$$
\begin{aligned}
&{E_z}^{n+1}{}_{(i,j)}-\frac{(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j-1)}})}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}}{E_z}^{n+1}{}_{(i,j-1)} \qquad (7.47)\\
=&\frac{(\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j-1)}})\,{E_z}^{n}{}_{(i,j-1)}-(\Delta t-\Delta y\sqrt{\mu\epsilon_{(i,j)}})\,{E_z}^{n}{}_{(i,j)}}{\Delta t+\Delta y\sqrt{\mu\epsilon_{(i,j)}}}
\end{aligned}
$$

When (7.45), (7.46) and (7.47) are applied to all the grid locations, like the first substep, another tridiagonal system of equations is found. The values of ${E_z}^{n+1}$ are found by solving this tridiagonal system of equations. Then using (7.41) ${H_x}^{n+1}$ is solved explicitly:

$$
{H_x}^{n+1}={H_x}^{n+\frac{1}{2}}-\frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {E_z}^{n+1}}{\partial y} \qquad (7.48)
$$

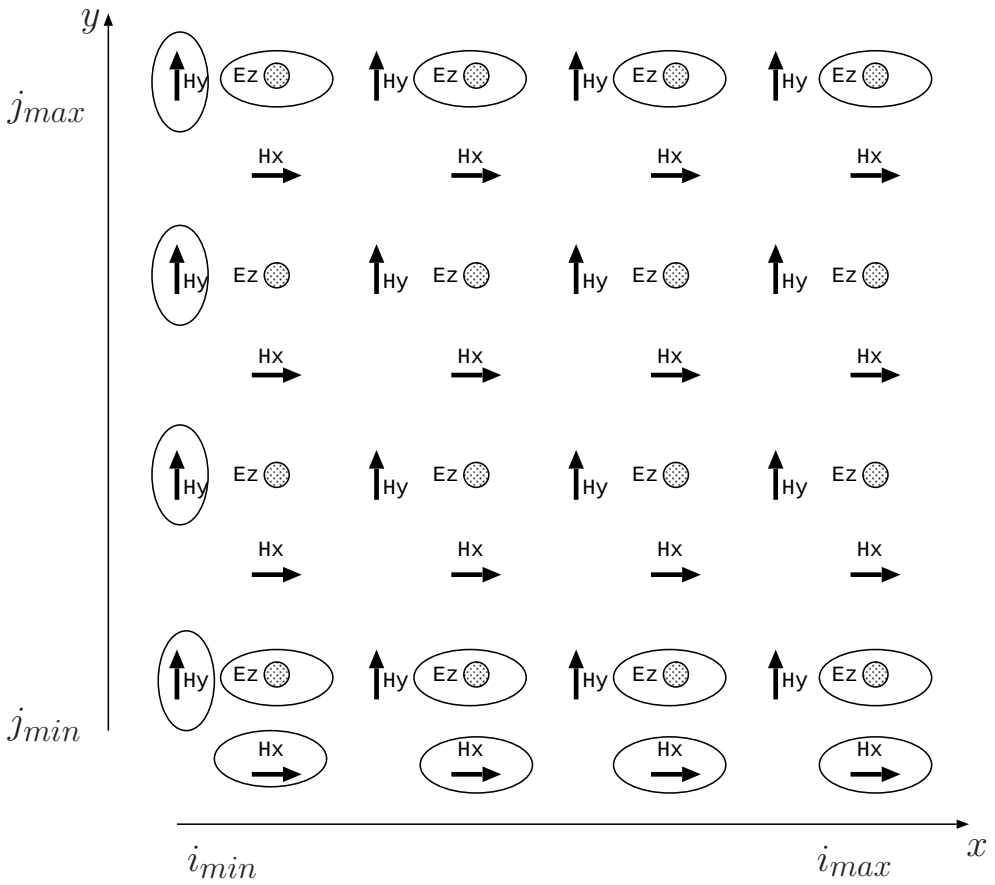


Figure 7.2: Locations on the grid of the boundary values of **E** and **H** that are calculated using the Mur's ABCs (second half-step). Only those boundary values that are used in the modified FD–ADI–FDTD equations are calculated.

(7.48) is discretized to (7.49)

$$H_x^{n+1}{}_{(i,j)} = H_x^{n+\frac{1}{2}}{}_{(i,j)} - \frac{\Delta t}{2}\frac{1}{\mu}\frac{1}{\Delta y}\Big\{E_z^{n+1}{}_{(i,j)} - E_z^{n+1}{}_{(i,j-1)}\Big\} \tag{7.49}$$

As (7.49) is not applicable when $j = j_{min}$, $H_x(j_{min})$ (Figure 7.2) is determined using Mur's ABC

$$H_x^{n+1}{}_{(i,j)} = \frac{(\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j+1)}})}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j)}}} H_x^{n+1}{}_{(i,j+1)} \tag{7.50}$$
$$+ \frac{(\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j+1)}})\, H_x^{n}{}_{(i,j+1)} - (\Delta t - \Delta y\sqrt{\mu\epsilon_{(i,j)}})\, H_x^{n}{}_{(i,j)}}{\Delta t + \Delta y\sqrt{\mu\epsilon_{(i,j)}}}$$

Next (7.42) is used to solve for ${H_y}^{n+1}$ explicitly:

$$
\begin{aligned}
{H_y}^{n+1} = \frac{\Delta t}{2}\frac{1}{\mu}\frac{\partial {E_z}^{n+\frac{1}{2}}}{\partial x} + {H_y}^{n+\frac{1}{2}} \qquad (7.51)\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu}\Big(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty}\Big)\Big(\frac{\partial {E_z}^{n+\frac{1}{2}}}{\partial x} - \frac{\partial {E_z}^{n}}{\partial x}\Big)\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big(\frac{\partial^2 {H_x}^{n+\frac{1}{2}}}{\partial x\partial y} - \frac{\partial^2 {H_x}^{n}}{\partial x\partial y}\Big)
\end{aligned}
$$

The discretization of (7.51) gives

$$
\begin{aligned}
H_y^{n+1}{}_{(i,j)} = \frac{\Delta t}{2}\frac{1}{\mu}\frac{1}{\Delta x}\Big\{E_z^{n+\frac{1}{2}}{}_{(i,j)} - E_z^{n+\frac{1}{2}}{}_{(i-1,j)}\Big\} + H_y^{n+\frac{1}{2}}{}_{(i,j)} \qquad (7.52)\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu}(\frac{\epsilon_{\mathrm{S}}}{\tau_{\mathrm{D}}\epsilon_\infty} + \frac{\sigma}{\epsilon_0\epsilon_\infty})\Big(\frac{1}{\Delta x}\Big\{E_z^{n+\frac{1}{2}}{}_{(i,j)} - E_z^{n+\frac{1}{2}}{}_{(i-1,j)}\Big\} - \frac{1}{\Delta x}\Big\{E_z^{n}{}_{(i,j)} - E_z^{n}{}_{(i-1,j)}\Big\}\Big)\\
-(\frac{\Delta t}{2})^2\frac{1}{\mu\epsilon_0\epsilon_\infty}\Big(\frac{1}{\Delta x\Delta y}\Big\{H_x^{n+\frac{1}{2}}{}_{(i,j+1)} - H_x^{n+\frac{1}{2}}{}_{(i,j)} - H_x^{n+\frac{1}{2}}{}_{(i-1,j+1)} + H_x^{n+\frac{1}{2}}{}_{(i-1,j)}\Big\}\\
-\frac{1}{\Delta x\Delta y}\Big\{H_x^{n}{}_{(i,j+1)} - H_x^{n}{}_{(i,j)} - H_x^{n}{}_{(i-1,j+1)} + H_x^{n}{}_{(i-1,j)}\Big\}\Big)
\end{aligned}
$$

(7.52) can not be used when $i = i_{min}$ and $j = j_{max}$. As $H_y(j_{max})$ is not required elsewhere in the scheme, it is not calculated. $H_y(i_{min})$ (Figure 7.2) is determined using (7.53) which comes from Mur's ABC.

$$
\begin{aligned}
H_y^{n+1}{}_{(i,j)} = \frac{(\Delta t - \Delta x\sqrt{\mu\epsilon_{(i+1,j)}})}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}H_y^{n+1}{}_{(i+1,j)} \qquad (7.53)\\
+\frac{(\Delta t + \Delta x\sqrt{\mu\epsilon_{(i+1,j)}})\,H_y^{n}{}_{(i+1,j)} - (\Delta t - \Delta x\sqrt{\mu\epsilon_{(i,j)}})\,H_y^{n}{}_{(i,j)}}{\Delta t + \Delta x\sqrt{\mu\epsilon_{(i,j)}}}
\end{aligned}
$$

Finally using (7.40) ${D_z}^{n+1}$ is calculated explicitly

$${D_z}^{n+1} = {D_z}^{n+\frac{1}{2}} + \frac{\Delta t}{2}\frac{\partial {H_y}^{n+\frac{1}{2}}}{\partial x} - \frac{\Delta t}{2}\sigma {E_z}^{n+1} - \frac{\Delta t}{2}\frac{\partial {H_x}^{n+1}}{\partial y} \qquad (7.54)$$

(7.54) is discretized to (7.55)

$$D_z^{n+1}{}_{(i,j)} = D_z^{n+\frac{1}{2}}{}_{(i,j)} + \frac{\Delta t}{2}\frac{1}{\Delta x}\left\{H_y^{n+\frac{1}{2}}{}_{(i+1,j)} - H_y^{n+\frac{1}{2}}{}_{(i,j)}\right\} \\ -\frac{\Delta t}{2}\sigma E_z^{n+1}{}_{(i,j)} - \frac{\Delta t}{2}\frac{1}{\Delta y}\left\{H_x^{n+1}{}_{(i,j+1)} - H_x^{n+1}{}_{(i,j)}\right\} \tag{7.55}$$

(7.55) is not applicable when $i = i_{max}$ and $j = j_{max}$. But as the values of $D_z(i_{max})$ and $D_z(j_{max})$ are not required elsewhere in the scheme, they are not calculated. All the equations to calculate the values of **E**, **D** and **H** have been now derived. A close look to these equations shows that the tridiagonal matrices of the modified FD–ADI–FDTD method are those of the normal FD–ADI–FDTD method [91] except for additional terms which do not increase the computational burden significantly.

7.3 Numerical Validation

To validate the proposed modified FD–ADI–FDTD method, numerical tests were conducted for a 2D computational space of 400×400 cells (in x and y directions) consisting of 2 media. Half of the computational space ($x \leq 200$) had the parameters $\epsilon_S = 9.5$, $\epsilon_\infty = 4.2$, $\sigma = 0.0$ S/m and $\tau_D = 77.0$ ps and the other half ($x > 200$) had $\epsilon_S = 6.2$, $\epsilon_\infty = 3.5$, $\sigma = 0.0$ S/m and $\tau_D = 39.0$ ps. A line source was applied at (180,200) in the first medium. The excitation waveform was a Gaussian pulse centred at 2.3 GHz. A uniform spatial sampling of $\Delta x = \Delta y = \Delta s = 0.3 \times 10^{-3}$m was used. The time-step was variably taken at or above the CFL stability condition of the explicit FDTD method: $\Delta t = CFLN \times \Delta s/(c\sqrt{2})$ with $CFLN$ referred to as CFL number and c the free space light–speed.

A test point, 40 cells away into the second medium, at (220,200) was taken. The solution provided by the FD–ADI–FDTD method [91] for $CFLN = 1$ was used as the reference. Results for the time evolution of E_z at the test point for the FD-ADI–FDTD method and the modified FD–ADI–FDTD method are shown in Figure 7.3 and 7.4, respectively, when $CFLN$ was varied from 1 to 30. Noticeable errors are seen when FD–ADI–FDTD method is used while in the

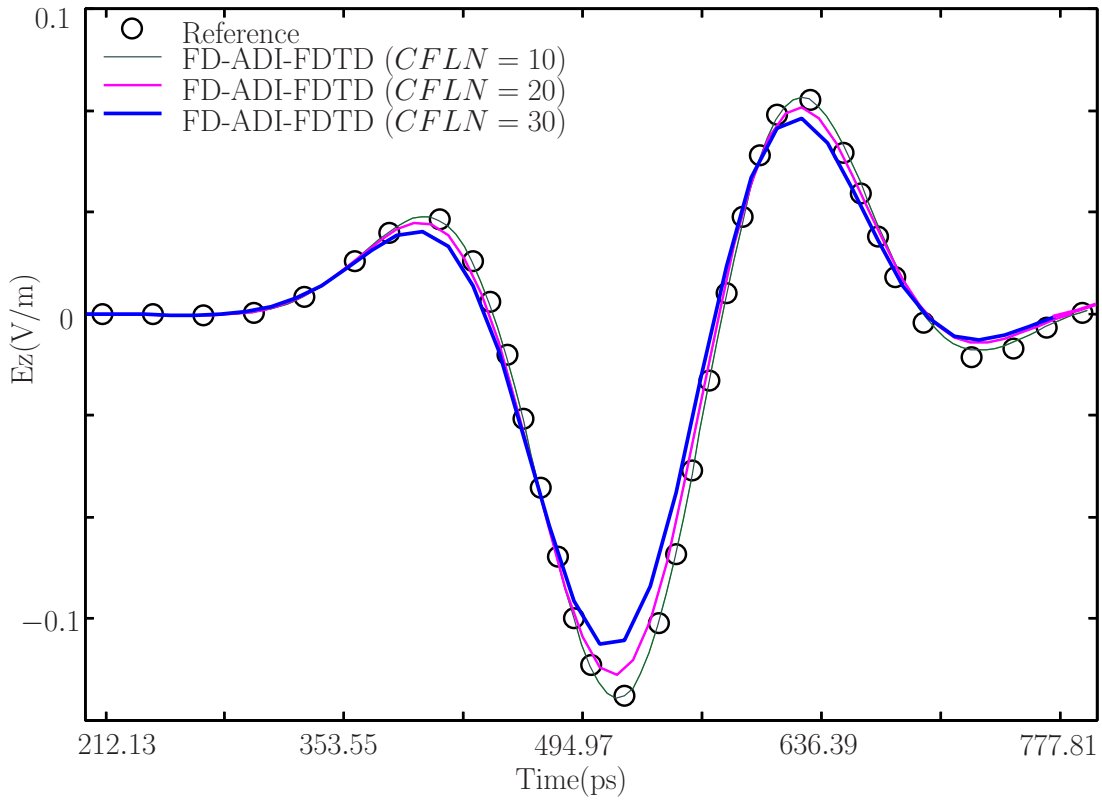


Figure 7.3: Observed signals of the normal FD–ADI–FDTD method with changing CFL numbers.

case of modified FD–ADI–FDTD method these are significantly reduced. Splitting errors resulting from the dropped last term of (7.11) account for these errors, which have been taken care of in the modified FD–ADI–FDTD method.

To quantify the improvement of the modified FD–ADI–FDTD method, an average error $\mathcal{E}$ calculated over the whole frequency band of the transient excitation has been defined:

$$\mathcal{E} = \sqrt{\frac{\sum_f \left(\mathcal{S}_{rcd} - \mathcal{S}_{rcd}^{ref}\right)^2}{\sum_f \left(\mathcal{S}_{rcd}^{ref}\right)^2}} \tag{7.56}$$

where $\mathcal{S}_{rcd}$ is the amplitude spectrum of the field received at the test point by the modified FD–ADI–FDTD method and $\mathcal{S}_{rcd}^{ref}$ is that of the reference. The average errors for FD–ADI–FDTD and modified FD–ADI–FDTD methods for

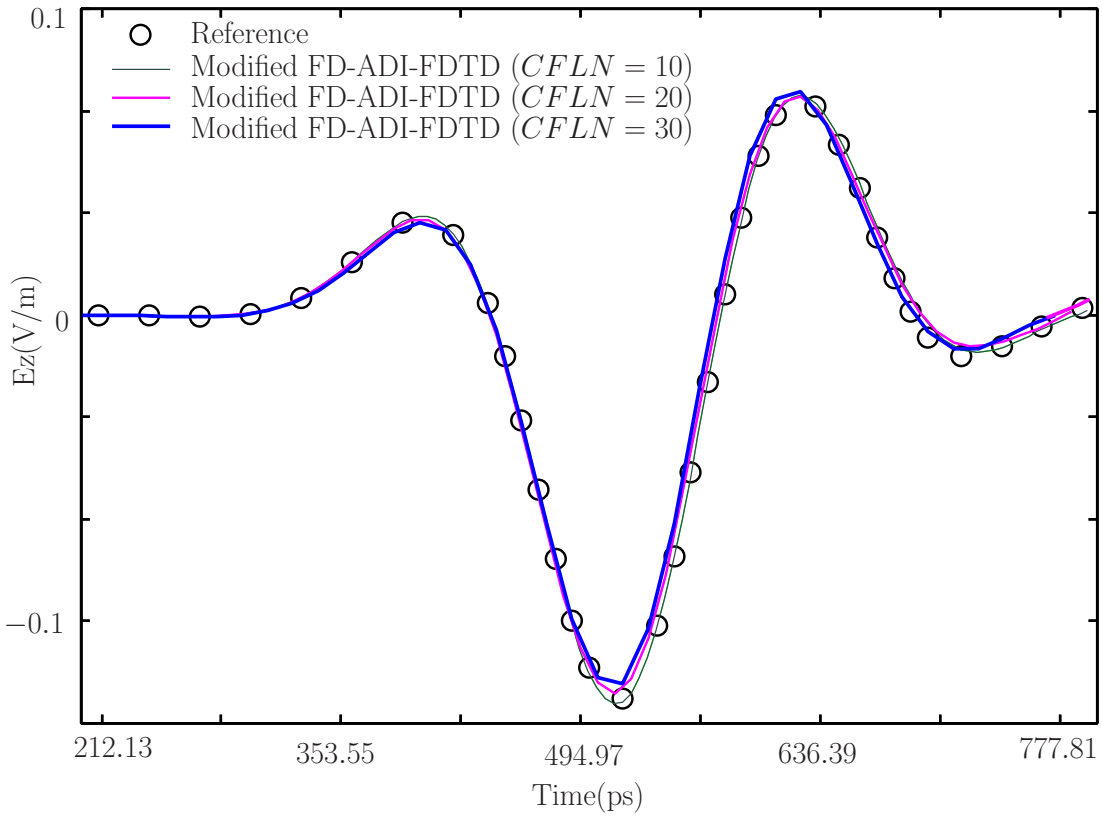


Figure 7.4: Observed signals of the modified FD–ADI–FDTD method with varying CFL numbers.

$CFLN$	FD–ADI–FDTD	Modified FD–ADI–FDTD
10	0.096932	0.052911
20	0.471145	0.091383
30	0.718453	0.124622

Table 7.1: Average error of normal FD–ADI–FDTD method and modified FD–ADI–FDTD method at different $CFLN$

three values of $CFLN > 1$ are given in Table 7.1. The proposed method reduces the errors of the FD–ADI–FDTD method without significant increase in the computational costs.

7.4 Conclusion

A method capable of reducing the numerical errors in the FD–ADI–FDTD method has been presented and numerically verified in this chapter. Like the normal FD–ADI–FDTD method, the modified method still requires only to solve the tridiagonal matrix systems but can reduce the perturbations introduced in the CN–FDTD method to formulate the ADI–FDTD method. In other words, the improvement of accuracy comes without significant increase in the computational costs.

Chapter 8

Implementation of the Proposed Methods

The implementation of the FD–CN–FDTD method in serial code as well as its parallel implementation in OpenMP are presented in this chapter. As the power of computation and technology of memory are galloping fast, in parallel with the development of sophisticated sparse matrix solvers, the CN–FDTD method poses itself as a promising, affordable alternative to the conventional explicit FDTD method. However, there are scarce precedence of implementation of the CN–FDTD method in computer programmes and no precedence of implementation of the frequency dependent CN–FDTD method which is more complicated. Therefore, the implementation techniques described in this chapter bear significance and would be useful for future researchers on the CN–FDTD method. Also the implementation of the modified FD–ADI–FDTD method proposed in this thesis is described in this chapter.

8.1 Introduction

Ideally, implementation of a numerical method in computer code should satisfy a number of criteria. For example, it should be simple, easily understood, portable and easily parallelizable. Simplicity in coding ensures potential bugs and problems can be avoided and other researchers can re-use the code with simple modification as required. Portability means the code can be compiled on different architectures and operating systems without any change in it. Parallelization of the code should enable a larger computation than a CPU can handle.

All the unconditionally stable FDTD methods presented in this thesis were implemented in Fortran 90. In the scientific computation, specially when high performance is required, Fortran is the most widely used programming language [121]. It is said to be a natural language for expressing science and engineering ideas. In addition to normal features present in other high-level programming languages, Fortran provides some additional merits:

- It allows variable-dimension array arguments in the subroutines.
- It has a rich set of useful generic-precision intrinsic functions.
- The new intrinsic functions of Fortran 90 allow very sophisticated array manipulations.
- Fortran 90 supports various useful features of C (column independent code, pointers, dynamic memory allocation) and C++ (operator overloading, primitive objects).
- The good design of Fortran allows maximal speed of execution.
- There is a huge amount of high quality scientific legacy code written in Fortran and much of which is publicly available.

The code can be parallelized using OpenMP in shared memory architecture or message passing interface (MPI) in distributed memory architecture. Adding OpenMP constructs to the serial Fortran code is straightforward to get some level of parallelization in shared memory. Use of MPI to distribute the computational load among several nodes can get the most benefit of parallelization but its implementation is complicated.

In this thesis, two implicit FDTD methods, namely, FD–CN–FDTD method and modified FD–ADI–FDTD method have been proposed. Since the FD–CN–FDTD method yields sparse matrices and the modified FD–ADI–FDTD method yields tridiagonal matrices, their implementation using standard matrix structures and algorithms will not give fast computation with the least memory requirements. Therefore, for efficient implementation of the proposed methods, the special characteristics of the matrices have to be exploited.

8.2 Implementation of the FD–CN–FDTD Method

Some of the key techniques used in the implementation of the FD-CN-FDTD method in Fortran code are described in this section. As explained in Chapter 5, the FD-CN-FDTD method can use either direct or iterative solvers to solve the sparse system of linear equations $\boldsymbol{Au} = \boldsymbol{c}$. Fig. 8.1 shows the flowchart of the FD-CN-FDTD method in both approaches of direct and iterative solvers. When the direct solver is used, the sparse matrix $\boldsymbol{A}$ generated by the FD-CN-FDTD method is first decomposed in factors of lower and upper triangular matrices (LU factors). Then during the iterations of the simulation (FDTD iteration), these factors are repeatedly used with a new value of vector $\boldsymbol{c}$ at each iteration and the system is solved by forward and backward triangular sweeps. In the iterative solver approach, at each step of the simulation, an initial estimation of the solution is made and by repeated use of a certain algorithm (depending on the iterative method used) this initial estimation approaches to the desired solution. In both cases, the first step is to form the sparse matrix from the FD–CN–FDTD equations (3.32), (3.40), (3.42) and those mentioned in Tables 3.1, 3.2, 3.3. As this matrix is mostly filled with zeroes and only a few non-zeroes, advantages can be taken from the sparseness of the matrix. Because, there is no necessity of storing the zeroes and only the non-zero values and their positions in the matrix (row and column indices) are important. To form the sparse matrix in the FD–CN–FDTD method, the function of Fig. 8.2 is used.

In the function of Fig. 8.2, the values of `(i,j,k)` are passed to `(p,q,r)`, where, `(i,j,k)` are the locations of the grid points in the computational space. `iminplusone` $= i_{min} + 1$, `jminplusone` $= j_{min} + 1$, `kminplusone` $= k_{min} + 1$, where, $i_{min}, j_{min}, k_{min}$ and $i_{max}, j_{max}, k_{max}$ are the lower and upper limits of the computational space. Based on the values of `(i,j,k)`, the function `mat_ind(p,q,r)` of Fig. 8.2 returns the values which are used to calculate the row and column indices of each of the elements of the sparse matrix. The elements of the sparse matrix come from the coefficients of E_x, E_y and E_z of the FD–CN–FDTD equations (3.32), (3.40), (3.42) and those mentioned in Tables 3.1, 3.2, 3.3.

The sequence in the unknown vector $\boldsymbol{u}$ of the system of equation $\boldsymbol{Au} = \boldsymbol{c}$ of

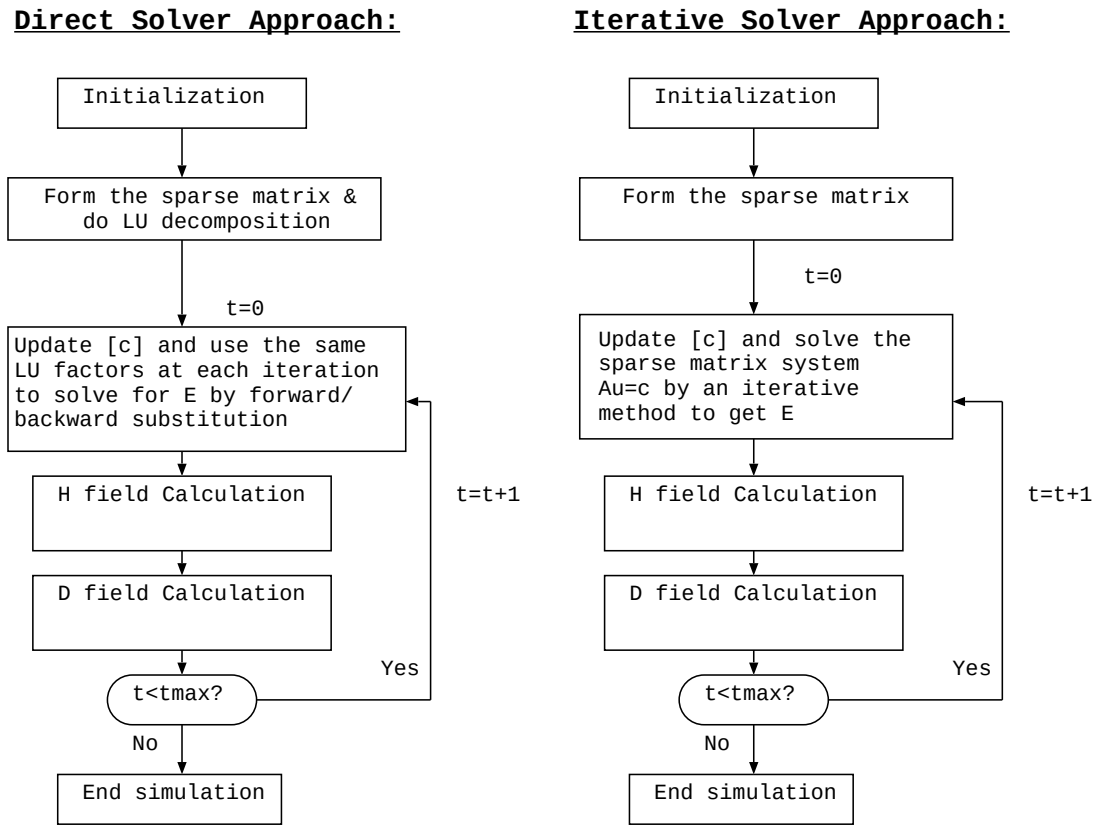


Figure 8.1: Flowchart of the FD–CN–FDTD method in direct and iterative solvers approaches

```
integer function mat_ind (p,q,r)
implicit none
integer, intent(in):: p,q,r
integer pmax, qmax, rmax

pmax=imax-iminplusone+1
qmax=jmax-jminplusone+1
rmax=kmax-kminplusone+1

mat_ind=(p-iminplusone)*qmax*rmax+(q-jminplusone)*rmax+r-kminplusone+1

end function mat_ind
```

Figure 8.2: Function for computing the matrix indices in the implementation of the FD–CN–FDTD method

the FD–CN–FDTD method is useful in determining the row and column indices of the elements of the sparse matrix. Fig. 8.3 shows the sequence of unknowns in the vector $\boldsymbol{u}$, when the FD–CN–FDTD equations are applied to all the grid points. In $\boldsymbol{u}$, unknown values of E_x at all the grid points come first, followed by those of E_y and E_z, respectively. When the FD–CN–FDTD equations are applied to all the grid points, first (3.32) is used at all those points, followed by (3.40) and (3.42). Row indices for the coefficients of E_x, E_y and E_z of (3.32) are calculated by passing `(i,j,k)` to the function `mat_ind(p,q,r)`. Now, in the Fortran code the total number of grid points is calculated as

```
max_mul=(imax-iminplusone+1)*(jmax-jminplusone+1)*(kmax-kminplusone+1)
```

Therefore, the row indices for the coefficients of E_x, E_y and E_z of (3.40) and (3.42) are calculated by `max_mul+mat_ind(p,q,r)` and `2*max_mul+mat_ind(p,q,r)`, respectively.

Each column of $\boldsymbol{A}$ is multiplied with the unknown vector $\boldsymbol{u}$ which has the sequence of Fig. 8.3. Therefore, for either of (3.32), (3.40) or (3.42), the column indices for the coefficients of E_x, E_y and E_z are calculated by `mat_ind(p,q,r)`, `max_mul+mat_ind(p,q,r)` and `2*max_mul+mat_ind(p,q,r)`, respectively. In this way, all the non-zero values of the matrix $\boldsymbol{A}$ and their row and column indices are calculated and stored in coordinate sparse matrix format (COO) as shown in Fig. 8.4. So if there are n non-zeroes in matrix $\boldsymbol{A}$, two integer arrays each of size n and a double precision array of size n are required.

The memory requirements for storing the sparse matrix can be further reduced by converting the COO format to compressed sparse row (CSR) format. CSR is the most popular sparse matrix storage format as it is more convenient to perform typical matrix computations in this format [110]. Many of the useful sparse matrix software packages are compatible with this format. In CSR format the sparse matrix is represented by three arrays:

- A double precision array, val, of the length of the number of non-zeroes present in the sparse matrix. It stores the non-zero elements of the matrix row by row.
- An integer array, jj, of the length of the number of non-zeroes present in the sparse matrix. It contains the column indices of the non-zero elements stored in the above-mentioned double precision array.

$$
[\boldsymbol{u}] \quad = \quad \begin{bmatrix}
E_x(i_{min}, j_{min}, k_{min}) \\
E_x(i_{min}, j_{min}, k_{min}+1) \\
E_x(i_{min}, j_{min}, k_{min}+2) \\
\cdots \\
E_x(i_{min}, j_{min}, k_{max}) \\
E_x(i_{min}, j_{min+1}, k_{min}) \\
E_x(i_{min}, j_{min+1}, k_{min}+1) \\
E_x(i_{min}, j_{min+1}, k_{min}+2) \\
\cdots \\
E_x(i_{min}, j_{min+1}, k_{max}) \\
\cdots \\
\cdots \\
E_x(i_{min}, j_{max}, k_{max}) \\
E_x(i_{min+1}, j_{min}, k_{min}) \\
E_x(i_{min+1}, j_{min}, k_{min}+1) \\
\cdots \\
\cdots \\
\cdots \\
E_x(i_{max}, j_{max}, k_{max}) \\
E_y(i_{min}, j_{min}, k_{min}) \\
E_y(i_{min}, j_{min}, k_{min}+1) \\
\cdots \\
\cdots \\
\cdots \\
\cdots \\
E_y(i_{max}, j_{max}, k_{max}) \\
E_z(i_{min}, j_{min}, k_{min}) \\
E_z(i_{min}, j_{min}, k_{min}+1) \\
\cdots \\
\cdots \\
\cdots \\
\cdots \\
E_z(i_{max}, j_{max}, k_{max})
\end{bmatrix}
$$

Figure 8.3: The sequence in the unknown vector $\boldsymbol{u}$ of the system of equation $\boldsymbol{Au} = \boldsymbol{c}$ of the FD–CN–FDTD method

- An integer array, *ptr*, of length $s + 1$, where, s is the order of the sparse matrix. It contains the pointers (indices) to the beginning of each row in either of the previous two arrays. The last element of *ptr* array always has the value $nnz + 1$, where, nnz is the number of non-zeroes in the sparse matrix.

$RowIndex$	$r_1,\quad r_2,\quad r_3,\quad \ldots\ldots\ldots\quad r_n$
$ColumnIndex$	$c_1,\quad c_2,\quad c_3,\quad \ldots\ldots\ldots\quad c_n$
$Values$	$val_1,\quad val_2,\quad val_3,\quad \ldots\ldots\ldots\quad val_n$

Figure 8.4: Coordinate (COO) storage format of sparse matrix

For example, the matrix $\boldsymbol{G}$ in (8.1) is represented in CSR format in (8.2), (8.3) and (8.4). val in (8.2) and jj in (8.3) can be easily understood from the above description of the CSR arrays. ptr array holds the position of the left-most non-zero element among all the non-zero elements of the matrix (row by row). Thus the contents of the ptr array in (8.4) are the positions of 2, 4, 7, 8 in val, which holds all the non-zero elements of the matrix. As the total number of non-zero elements in $\boldsymbol{G}$ is 8, the last element of ptr is $8 + 1 = 9$.

$$\boldsymbol{G} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & 4 & 3 & 5 \\ 7 & 0 & 6 & 0 \\ 0 & 0 & 0 & 8 \end{pmatrix} \tag{8.1}$$

$$val = 2 \quad 1 \quad 4 \quad 3 \quad 5 \quad 7 \quad 6 \quad 8 \tag{8.2}$$

$$jj = 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 1 \quad 3 \quad 4 \tag{8.3}$$

$$ptr = 1 \quad 3 \quad 6 \quad 8 \quad 9 \tag{8.4}$$

The FD–CN–FDTD sparse matrix in COO format is converted to CSR format using SPARSEKIT package [122]. In order to solve the sparse system, $\boldsymbol{Au} = \boldsymbol{c}$, using direct or iterative solvers, Harwell Subroutine Library (HSL) packages [123]

were used. For this, the matrix in CSR format is again required to convert to `HSL_ZD11` type which is the HSL's own derived type for sparse matrices. `HSL_ZD11` type is used to allow the exchange of data between HSL subprograms and other codes [124]. An excerpt of the code for the solution of $\boldsymbol{Au} = \boldsymbol{c}$ using HSL's BiCGStab routine, MI–26, is shown in Fig. 8.5. In Fig. 8.5, `iact` is an integer variable whose value decides the next course of action in the solution of $\boldsymbol{Au} = \boldsymbol{c}$. Value of `iact` determines whether an error is occurred or further iteration is required to reach the convergence or convergence is achieved and computation should be terminated. The details of other variables `n`, `w`, `ldw`, `locy`, `locz`, `resid`, `icntl`, `cntl`, `info26`, `isave`, `rsave`, `info65` can be found in the HSL documentations [125][126]. At each FDTD iteration using a sparse matrix solver routine, like MI–26 in Fig. 8.5, the values of electric fields are obtained. Thereafter, calculation of magnetic fields and electric flux densities is quite straightforward using the explicit equations of Section 3.5.

```
iact = 0
do
call mi26ad(iact,n,w,ldw,locy,locz,resid,icntl,cntl,info26,isave,rsave)
if (iact == -1) then
write(*,*) "Error in solver loop"
exit
else if (iact == 1) then
if (resid <= tolerance) then
exit
end if
else if (iact == 2) then
call mc65_matrix_multiply_vector(a, w(:,locz), w(:,locy), info65)
end if
end do
```

Figure 8.5: An excerpt of the FD–CN–FDTD code where $\boldsymbol{Au} = \boldsymbol{c}$ is solved using the BiCGStab method

When the sparse matrix system is solved in iterative methods, multiplications of a matrix and a vector are required. For this matrix-vector multiplication, HSL documentations suggested to use HSL routine MC–65 (`mc65_matrix_multiply_vector` in Fig. 8.5). However there are other subroutines for sparse matrix-vector multiplications, for example, `amux` from SPARSEKIT

Subroutine	% of total CPU time used by the subroutine	Total CPU time
`mc65_matrix_multiply_vector`	47.8	84 min 31 sec
`mi26ad`	18.5	

Table 8.1: Performance when matrix-vector multiplication subroutine, mc65_matrix_ multiply_vector, is used in the implementation of the FD–CN–FDTD method ($CFLN = 1$)

Subroutine	% of total CPU time used by the subroutine	Total CPU time
`amux`	42.4	70 min 49 sec
`mi26ad`	25.3	

Table 8.2: Performance when matrix-vector multiplication subroutine, amux, is used in the implementation of the FD–CN–FDTD method ($CFLN = 1$)

[122]. Matrix-vector multiplications account for a significant portion of the computations. Therefore, for computationally efficient implementation of the FD–CN–FDTD method, this was carefully studied with Fortran profiler (`gprof`). For the numerical tests described in Section 5.3.1 (Fig. 5.2), when the simulation is run for 1200 time steps with $CFLN = 1$ on AMD Athlon 64 X2 4200+ Dual Core Processor, the performance of matrix-vector multiplication subroutines, `mc65_matrix_multiply_vector` and `amux` are shown in Table 8.1 and Table 8.2. Intel Fortran Compiler was used in these tests and in all the studies described in this thesis except the parallelization described in Section 8.3. In Table 8.1 and Table 8.2, the performance is shown in terms of the percentage of total CPU time used by a particular subroutine and total CPU time required by the whole FD–CN–FDTD code. `amux` accounted for 42.4% of the total time spent to run the code in comparison to 47.8% for `mc65_matrix_multiply_vector`. Depending on the choice of matrix-vector multiplication subroutine, the percentage of time used by the BiCGStab solver subroutine, `mi26ad`, is also affected. The percentage of time for `mi26ad` includes Basic Linear Algebra Subprograms (BLAS) routines: `dnrm2`, `daxpy`, `ddot`, `dcopy` and `dscal`, which are called by `mi26ad` during its execution. The most noticeable observation in Table 8.1 and Table 8.2 is the reduction of total CPU time when `amux` is used instead of `mc65_matrix_multiply_vector`. An advantage of using `amux` is that it can do the matrix-vector multiplication when

$CFLN$	% of total CPU time	
	`amux`	`mi26ad`
1	42.4	25.3
2	44.4	28.1
3	48.7	30.7
4	51.8	31.3
5	56.4	32.5

Table 8.3: Performance of the two most computationally expensive subroutines at different $CFLN$

the sparse matrix is stored in CSR format, while `mc65_matrix_multiply_vector` requires the matrix in `HSL_ZD11` format. So, use of `amux` simplifies the implementation by not requiring to convert the sparse matrix into `HSL_ZD11` format.

As in each time step of the FD–CN–FDTD method, usually matrix-vector multiplication needs to be performed a number of times, it has a significant contribution in the overall computational performance. This is because, as shown in Fig. 8.5, matrix-vector multiplications need to be done repeatedly until the solution converges. Furthermore, Section 5.3.1 (Fig. 5.7) showed, in solving the sparse matrix system, more iterations are required to converge at higher $CFLN$. This means that the number of matrix-vector multiplications would increase at higher $CFLN$. For the same numerical test mentioned above, the performance of `amux` and `mi26ad` subroutines at different $CFLN$ is shown in Table 8.3. These two are the most computationally expensive subroutines used in the implementation of the FD–CN–FDTD method. Table 8.3 shows that with the increase of $CFLN$, the percentage of total CPU time used by both of the subroutines increases. Preconditioners are usually used with the sparse matrix by matrix-vector multiplication operation. Therefore, if a suitable preconditioner is found for the FD–CN–FDTD method in future, the computational effects of matrix-vector multiplications need to be taken into account.

In order to reduce the computational costs and memory requirements, some techniques were used during the implementation of the FD–CN–FDTD method.

To implement the FD–CN–FDTD equations (3.32), (3.40), (3.42) and those mentioned in Tables 3.1, 3.2, 3.3, instead of repeatedly calculating the coefficients of E_x, E_y, E_z, H_x, H_y, H_z, D_x, D_y, D_z, they are calculated only once before the formation of the sparse matrix, stored and reused. In most cases of the simulation, uniform space discretization of $\Delta x = \Delta y = \Delta z = \Delta s$ is used. Therefore, in the implementation the constant multiplying terms of the FD–CN–FDTD equations $\frac{P_1(i,j,k)}{\Delta x^2}$, $\frac{P_1(i,j,k)}{\Delta y^2}$, $\frac{P_1(i,j,k)}{\Delta z^2}$, $\frac{P_1(i,j,k)}{\Delta x \Delta y}$, $\frac{P_1(i,j,k)}{\Delta y \Delta z}$, $\frac{P_1(i,j,k)}{\Delta z \Delta x}$ can be equated to $\frac{P_1(i,j,k)}{\Delta s^2}$. The number of arithmetic operations and the computational time are saved in these ways.

Setting up the computational environment: Fig. 8.6 shows the way in which the computational environment of the FD–CN–FDTD method is set up. As the FD–CN–FDTD method is meant for inhomogeneous media, for each grid point (i, j, k) corresponding media parameters have to be defined. The coefficients for each field component of the FD–CN–FDTD equations are functions of the media parameters ((3.33), (3.34), (3.35), (3.36), (3.37), (3.38) and (3.13), (3.14), (3.15), (3.16), (3.17), (3.18)). Therefore, their values also depend on the location of the grid point in the computational space and need to be calculated based on the media parameters at that grid point. However, storing media parameters and coefficients for all the grid points will take a lot of memory. Therefore, a technique is used to save the memory.

The computational space consists of different type of media, each having different parameters. Each of the media is identified by an integer `m`. An array named `idmed(i,j,k)` holds the media type (`m`) for each (i, j, k). The coefficients are calculated for each media type, rather than for each (i, j, k). The total number of media present in the computational space is limited and usually less than the total number of grid points in the computational space. Therefore, this approach can avoid expensive calculation and storing of data for each (i, j, k). In the skeleton code of Fig. 8.6 `dt`, `dx`, `dy`, `dz` are Δt, Δx, Δy, Δz and `debcoeff1`, `debcoeff2`, `debcoeff3`, `debcoeff4`, `debcoeff5`,`debcoeff6`, `p1`, `p2`, `p3`, `p4`, `p5`, `p6` are $\nu_1(i, j, k)$, $\nu_2(i, j, k)$, $\nu_3(i, j, k)$, $\nu_4(i, j, k)$, $\nu_5(i, j, k)$, $\nu_6(i, j, k)$, $P_1(i, j, k)$, $P_2(i, j, k)$, $P_3(i, j, k)$, $P_4(i, j, k)$, $P_5(i, j, k)$, $P_6(i, j, k)$, respectively ((3.33)–(3.38), (3.13)–(3.18)). `tau_d`, `sigma`, `eps_s`, `eps_inf`, `eps_0`, `mu` are τ_{D}, σ, ϵ_{S}, ϵ_∞, ϵ_0, μ, respectively. In this example, the computational space is formed by 2 different

media. As seen in Fig. 8.6 it is not necessary to store the media parameters `tau_d`, `sigma`, `eps_s`, `eps_inf` for all the grid points, rather, they are stored only for the two types of the media. The coefficients (for example, `debcoeff1`, `debcoeff2`, `p1`, `p2`) are calculated and stored only for the two types of media as well. Fig. 8.6 also shows how this technique is used in the formation of the sparse matrix. The first and second rows of the array `matrix_A_ind` hold the row and column indices of the matrix, respectively and the array `matrix_A_val` holds the value of the corresponding non-zero elements. In the last `do` loop of Fig. 8.6 by using the array `idmed` the media type (`m`) for a point (i, j, k) is determined first and then the value of `p1` for that media type is used to calculate the non-zero elements of the matrix. This approach saves the amount of memory, computations and requires less data reading.

Setting up the computational environment for human body model: Chapter 6 presented the modelling of human body in the FD–CN–FDTD method. In this model the geometrical features of the human body is read from the 2-mm resolution voxel model or phantom of [117], hereafter called NICT (National Institute of Information and Communications Technology, Japan) phantom. Each voxel represents an uniform volume of the human body such that it can be assigned with an identifying number that corresponds to a particular tissue or organ. The male NICT phantom consists of 320×160×866 voxels *i.e.* `imax = 320`, `jmax = 160`, `kmax = 866`. The NICT phantom has 866 files corresponding to `k = 1` to `kmax`, each containing the identifying numbers of all the tissues on the cross section of xy plane having the dimensions of `imax`×`jmax` = 320×160. Each of these files has the name '`numXXX`', where '`XXX`' is the value of `k` (*i.e.* 001 to 866). Fig. 8.7 shows the code for reading the identifying numbers of the tissues from the '`numXXX`' files and storing them into the `idmed` array. Here `nom` is a character variable that holds the NICT phantom filename and integer variables `d0`, `d1`, `d2` are used to produce the '`numXXX`' filenames in a sequential order. The code in Fig. 8.7 thus places the tissues represented by the identifying numbers in the computational space of the FD–CN–FDTD code. For each of these tissues the corresponding Debye parameters (Table 6.1) are stored appropriately in the arrays `tau_d`, `sigma`, `eps_s`, `eps_inf`. Then the calculation of the coefficients and rest of the implementation follow the same techniques described earlier (Fig. 8.6). In this way, the human body is modelled in the FD–CN–FDTD computational space.

```
integer m !media id
integer, parameter :: med=2 !in total, 2 types of media
integer, dimension (imin:imax,jmin:jmax,kmin:kmax) :: idmed

           ...........
           ...........
       <more declarations>
           ...........
           ...........

!defining which grid point has which type of media. the value
!of x1 determines the dividing plane in the computational space

do k = kmin,kmax
     do j = jmin,jmax
        do i = imin,imax

 if (i.le.x1) then
 idmed(i,j,k)=1
 else
 idmed(i,j,k)=2
 end if

        end do
     end do
 end do

!assigning the media parameters
!t1, s1, es1, es1 are the values of the media parameters

tau_d(1)= t1
sigma(1)= s1
eps_s(1)= es1
eps_inf(1)= es2

!t2, s2, es2, es2 are the values of the media parameters

tau_d(2)= t2
sigma(2)= s2
eps_s(2)= es2
eps_inf(2)= es2

         ...........
         ...........
      <more assigments>
         ...........
         ...........
```

contd.

```
! calculation of the coeffecients
do m=1,med

   debcoeff1(m)=tau_d(m)/(dt**2.0)+1.0/dt
   debcoeff2(m)=-2.0*tau_d(m)/(dt**2.0)-1.0/dt
   debcoeff3(m)=tau_d(m)/(dt**2.0)
   debcoeff4(m)=eps_0*eps_inf(m)*tau_d(m)/(dt**2.00 &
                +(eps_0*eps_s(m)+sigma(m)*tau_d(m))/dt+sigma(m)/2.0
   debcoeff5(m)=-2.0*eps_0*eps_inf(m)*tau_d(m)/(dt**2.0) &
                -(eps_0*eps_s(m)+sigma(m)*tau_d(m))/dt+sigma(m)/2.0
   debcoeff6(m)=eps_0*eps_inf(m)*tau_d(m)/(dt**2.0)
   p1(m)=((dt/2.0)**2.0)*debcoeff1(m)/debcoeff4(m)/mu
   p2(m)=debcoeff1(m)/debcoeff4(m)+debcoeff2(m)/debcoeff4(m)
   p3(m)=debcoeff3(m)/debcoeff4(m)
   p4(m)=debcoeff1(m)*dt/debcoeff4(m)
   p5(m)=debcoeff5(m)/debcoeff4(m)
   p6(m)=debcoeff6(m)/debcoeff4(m)

end do
          ...........
          ...........

!matrix A is being formed now
 do k = kminplusone,kmax
     do j = jminplusone,jmax
         do i = iminplusone,imax
 m=idmed(i,j,k)
          ...........
          ...........
    <programme statements>
          ...........
          ...........
matrix_A_ind(1,next)=mat_ind(i,j,k)
matrix_A_ind(2,next)=mat_ind(i,j,k)
matrix_A_val(next)=1.0+2.0*p1(m)/(dy**2.0)+2.0*p1(m)/(dz**2.0)
next=next+1
          ...........
          ...........
    <programme statements>
          ...........
          ...........

       end do
     end do
  end do
          ...........
          ...........
          ...........
```

Figure 8.6: Skeleton code showing the computational environment set-up and memory saving techniques in the implementation.

```
do k = 1,kmax
   d2=k/100; d1=(k-d2*100)/10; d0=(k-d2*100)-d1*10;
   nom='num' // CHAR(48+d2) // CHAR(48+d1) // CHAR(48+d0)
  open(unit=53, file=nom, status="old", iostat=err)
      if (err.NE.0) then
       print *, "Error opening file: '", nom, "'"
        stop
      end if

   do j = 1, jmax
     do i = 1, imax
       read(53, *, iostat=err) idmed (i,j,k)
       if (err.NE.0) then
         print *, "Error reading file: '", nom, "'"
         stop
       end if
     end do
   end do

  close(53)
end do
```

Figure 8.7: Code for reading the data from the human body phantom.

8.3 Parallelization of the FD–CN–FDTD Method in Shared Memory Architecture

The simulation of 2-millimetre resolution human body model in the FD–CN–FDTD method requires enormously large amount of memory and very long CPU time. One way to overcome these constraints is parallelising the FD–CN–FDTD code. In this thesis, parallelization has been done using OpenMP in a shared memory architecture. OpenMP provides the specifications for parallelizing the programmes written in C, C++ and Fortran in a shared memory environment. It consists of a set of compiler directives, runtime routines and environment variables using which the shared memory parallelism is specified. OpenMP provides an elegant and portable interface to parallelize the programme on various architectures and systems by small incremental changes to the source code.

In the parallel execution of the code, OpenMP uses the fork-join model as

shown in Fig. 8.8. An OpenMP programme begins as a single thread of execution. When it encounters a parallel construct, it creates ('forks') the required number of threads and becomes the master thread of all the threads. Programme statements within the parallel construct are executed in parallel by each thread of the team of threads. At the end of the parallel construct, threads synchronize back ('join') and only the master thread continues the execution. There can be any number of parallel regions and different number of threads in each parallel region, as shown in Fig. 8.8.

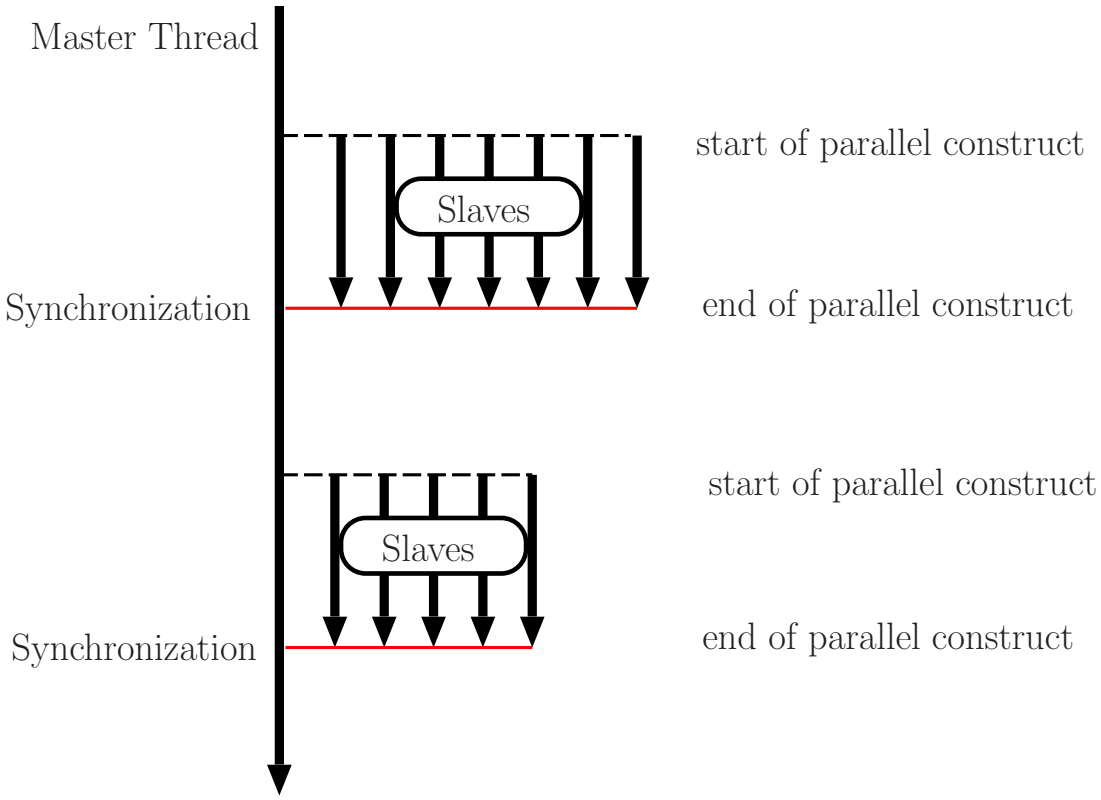


Figure 8.8: Execution model of OpenMP

To parallelize the code in OpenMP, appropriate OpenMP directives are added in the appropriate places in the source code. Then the code is compiled with a compiler that supports OpenMP and with the appropriate compiler options. The compiler interprets the OpenMP directives and parallelizes the code. In the serial implementation of the FD–CN–FDTD method, there are a number of nested `do` loops to cover the whole three-dimensional computational space as shown in Fig. 8.9. Computational efficiency can be achieved by distributing these loops over a number of threads using OpenMP directives. This will make

```
do k = kminplusone,kmax
    do j = jminplusone,jmax
        do i = iminplusone,imax
        ...........
        ...........
   <programme statements>
        ...........
        ...........
        end do
    end do
end do
```

Figure 8.9: Nested do loops in the serial code

the programme statements inside the loops execute in parallel by each thread. For example, OpenMP directives are used during the formation of matrix $\boldsymbol{A}$ of the FD–CN–FDTD method and shown in Fig. 8.10. In Fig. 8.10, the starting and ending OpenMP directives are, respectively

```
!$OMP PARALLEL DO SHARED(max_mul,p1,permittivity,matrix_A_val,
  matrix_A_ind) PRIVATE(i,j,k,m,next)
```

and

```
!$OMP END PARALLEL DO
```

In the starting OpenMP directive, the data scoping clauses, `SHARED` and `PRIVATE`, were used to control access to the variables properly. If no such access control mechanism is used, the problem of 'race conditions' can arise. In race condition, the programme will run but due to the 'racing' of the threads to modify a variable, it can produce wrong results. In Fig. 8.10 the clause `SHARED` specifies `max_mul,p1,permittivity,matrix_A_val,matrix_A_ind` as shared variables among all the threads for the duration of the parallel construct. All the threads use the same storage area for each shared variable and have access to that storage area. On the other hand, the clause `PRIVATE` makes the variables `i,j,k,m,next` 'local' to each thread. They will have multiple storage locations, one within the execution context of each thread, for the duration of the parallel construct. When read and write operations are performed by a thread on these private variables, they will refer to the local copy of the variable within that

```
! formation of matrix a

!$OMP PARALLEL DO SHARED(max_mul,p1,permittivity,matrix_A_val,
 matrix_A_ind) PRIVATE(i,j,k,m,next)

 do k = kminplusone,kmax
     do j = jminplusone,jmax
         do i = iminplusone,imax
         ...........
         ...........
    <codes to produce matrix a>
         ...........
         ...........
         end do
     end do
 end do

!$OMP END PARALLEL DO
```

Figure 8.10: An excerpt of the code showing the use of OpenMP directives in the nested do loops during the formation of matrix $\boldsymbol{A}$

thread. Their memory locations are inaccessible to other threads. When the code of Fig. 8.10 is run, the loops are distributed across the threads and for a certain loop a thread will have its own private values of `i,j,k`. Since the media parameter index `m` depends on the location in the computational space *i.e.* `i,j,k`, it is also local to that thread. The integer variable `next` is used for the purpose of indexing the matrix elements and is local because its value can not be changed by any operation outside a particular loop. Thus the race condition is avoided and the set of statements within the OpenMP starting and closing directives are executed in parallel. Likewise, at each time step of the FD–CN–FDTD method, vector $\boldsymbol{c}$ is computed using the OpenMP code which looks like Fig. 8.11.

In Section 8.2 it was mentioned that at each time step of the FD–CN–FDTD method, matrix-vector multiplications need to be done repeatedly until the convergence is achieved and a significant amount of computational time is spent on this operation. To reduce the computational time, matrix-vector multiplication has been parallelized using OpenMP orphaned directives. Matrix-vector multiplication is done by calling the subroutine `amux` and the operations within that

```
! computation of vector c

!$OMP PARALLEL DO SHARED(max_mul,p1,p2,p3,p4,p5,p6,permittivity,
 ex,ey,ez,hx,hy,hz,flux_density_dx,flux_density_dy,
 flux_density_dz,b) PRIVATE(i,j,k,m)

  do k = kminplusone,kmax
      do j = jminplusone,jmax
          do i = iminplusone,imax
           ...........
           ...........
       <codes to compute vector c>
           ...........
           ...........
          end do
      end do
  end do

!$OMP END PARALLEL DO
```

Figure 8.11: An excerpt of the code showing the use of OpenMP directives in the computation of vector $\boldsymbol{c}$

subroutine also need to be parallelized. Usual OpenMP directives are used before calling `amux` as shown in Fig. 8.12 and orphaned directives are used inside the subroutine as shown in Fig. 8.13. Orphan directives appear outside the lexical scope (*i.e.* between `PARALLEL` and `END PARALLEL` in Fig. 8.12), but inside the dynamic scope (meaning that, the lexical scope as well as all statements executed as a result of the execution of statements within the lexical scope *i.e.* statements in `amux`) of the parallel region.

```
!matrix-vector multiplication using amux
!$OMP PARALLEL
call amux ( mat_size, w(:,locz), w(:,locy), ao, jao, iao )
!$OMP END PARALLEL
```

Figure 8.12: OpenMP code for matrix-vector multiplication

Most of the `do` loops in the serial implementation of the FD–CN–FDTD method were parallelized using OpenMP directives as described above. However, it is not computationally efficient to parallelize the smaller `do` loops which

```
!$OMP PARALLEL DO SHARED(ia,a,x,ja,y,t) PRIVATE(i,k)

  do i = 1, n
!
!  Compute the inner product of row I with vector X.
!
    t = 0.0D+00
    do k = ia(i), ia(i+1)-1
      t = t + a(k) * x(ja(k))
    end do

    y(i) = t

  end do
!$OMP END PARALLEL DO
```

Figure 8.13: Orphaned OpenMP directives inside the matrix-vector multiplication subroutine amux

```
! calculating Hx in the interior space

!$OMP PARALLEL SHARED(ey,ez,hx) PRIVATE(i,j,k)
  !$OMP DO
       do k = kminplusone,kmax
          do j = jminplusone,jmax
              do i = iminplusone,imax

hx(current_step,i,j,k)=(dt/2.0/mu/dz)*(ey(current_step,i,j,k) &
-ey(current_step,i,j,k-1))-(dt/2.0/mu/dy) &
*(ez(current_step,i,j,k)-ez(current_step,i,j-1,k)) &
+hx(previous_step1,i,j,k)+(dt/2.0/mu/dz)*(ey(previous_step1,i,j,k) &
-ey(previous_step1,i,j,k-1))-(dt/2.0/mu/dy) &
*(ez(previous_step1,i,j,k)-ez(previous_step1,i,j-1,k))

              end do
          end do
       end do

  !$OMP END DO NOWAIT
!$OMP END PARALLEL
```

Figure 8.14: Use of NOWAIT clause in the OpenMP code

have small range between the minimum and maximum integer values over which it runs. During the parallelization of a `do` loop, parallel overhead costs have to be taken into account. When a loop is run in parallel, it adds the runtime costs because the master thread needs to start the slaves, the loops need to be divided across the threads and the threads need to be synchronized back at the end of the parallelization. All these add up to parallel overhead costs. Therefore, smaller `do` loops were not parallelized.

To enhance the computational efficiency, where appropriate `NOWAIT` clause was used with the OpenMP `do` loop. An example is shown in Fig. 8.14, where ${H_x}^{n+1}$ is being calculated explicitly after the calculation of electric fields. When the parallel OpenMP block is closed with `!$OMP END DO` directive, all the threads that execute the statements inside the parallel block are synchronized by this directive. If any thread finishes earlier than other threads, it has to wait until all the threads finish. Sometimes this can be wasteful. The `NOWAIT` clause of `!$OMP DO` loop removes this synchronization point and allows the thread that finishes early to proceed. In Fig. 8.14, to calculate `hx(current_step,i,j,k)`, a thread does not need the computed values from another thread and thereby, does not need to wait. Because, all the required values to calculate `hx(current_step,i,j,k)` (*i.e.* the electric field and magnetic field arrays on the right hand side in Fig. 8.14) are already available. Therefore, by removing the synchronization at the end of the loop and letting the thread that finishes early to go ahead to do the next computation, computational efficiency was achieved.

Numerical modelling of the human body in the FD–CN–FDTD method, as described in Chapter 6, was studied for the cases of $CFLN = 1$ and 3. Its implementation in OpenMP was compiled by Hitachi `f90` compiler with `-Oss` option and run on Hitachi SR16000 Model L2, POWER6 4.7GHz (dualcore)x16 processors. Table 8.4 shows the achieved improvement by the OpenMP code over the serial code, in terms of CPU time, when it was run for $1050/CFLN$ time steps. With the OpenMP code, the CPU time has been greatly reduced when $CFLN = 1$ and as expected, the most significant reduction in CPU time has been achieved in the case of $CFLN = 3$. However, the scaling is not perfect for the number of threads (32) the code uses. In OpenMP, usually, it is hard to obtain perfect speed-ups even when the parallelization is done correctly [127].

Therefore, this performance of the OpenMP code is not very unusual. Understanding the details of underlaying hardware and using vendor-supplied parallel mathematical operation libraries may improve the performance to some extent. But the ultimate benefits of parallelization can be fully achieved by using MPI in the dynamic memory architecture, although its implementation is complicated.

An interesting observation in Table 8.4 is the increase of speed-up by OpenMP parallelization at higher $CFLN$. Speed-up by OpenMP for $CFLN = 1$ and 3 are 2.63 and 5.85, respectively. Further tests with different $CFLN$ need to be performed to check whether the increase of speed-up is always linear with the $CFLN$. If it is found that better speed-up always comes at higher $CFLN$, it would indicate that the use of the FD–CN–FDTD method is more appropriate while parallelized.

$CFLN = 1$		
Code type	CPU time	Speed-up
Serial	39hr 53min	1
OpenMP	15hr 11min	2.63

$CFLN = 3$		
Code type	CPU time	Speed-up
Serial	37hr 38min	1
OpenMP	6hr 26min	5.85

Table 8.4: Speed-up by the OpenMP code on 32 cores at different $CFLN$.

8.4 Implementation of the Modified FD–ADI–FDTD Method

In Chapter 7 modified FD–ADI–FDTD method has been proposed. This accuracy improved modified FD–ADI–FDTD method is a two-step process and requires to solve a tridiagonal matrix system at each step as explained in Section 7.2. Fig. 8.15 shows the flowchart of this method. In the first step, ${H_y}^{n+\frac{1}{2}}$ is

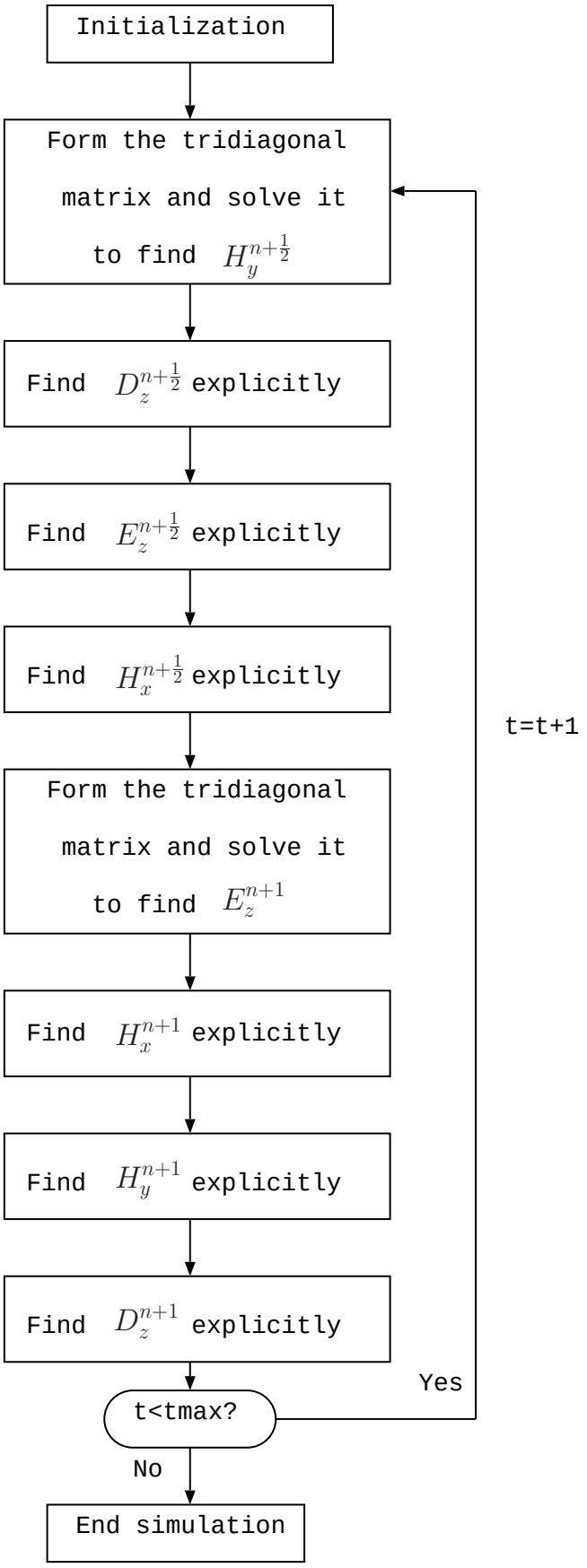


Figure 8.15: Flowchart of the modified FD–ADI–FDTD method

found by solving a tridiagonal matrix system, followed by the explicit update of ${D_z}^{n+\frac{1}{2}}$, ${E_z}^{n+\frac{1}{2}}$ and ${H_x}^{n+\frac{1}{2}}$. In the second step, ${E_z}^{n+1}$ is found by solving another tridiagonal matrix system, followed by the explicit update of ${H_x}^{n+1}$, ${H_y}^{n+1}$ and ${D_z}^{n+1}$. Linear Algebra PACKage (LAPACK) [128] tridiagonal matrix solver subroutine `DGTSV` was used in the implementation of the modified FD–ADI–FDTD method.

The modified FD–ADI–FDTD equations, described in Section 7.2, show that the coefficients of E_z, D_z, H_x, H_y are repeatedly used during the computation of the method. These coefficients are functions of the media parameters. For inhomogeneous media, the media parameters are space–dependent but storing these parameters for all the grid points will take a lot of memory. Also the calculation of all the coefficients for each grid point and storing them are expensive, both in terms of computational time and memory. To handle this, a strategy similar to the one described in Section 8.2 was followed. An integer `m` is used to represent each type of different media present in the computational space. For each grid point (i, j) an array `idmed(i,j)` holds the media type `m`. For the calculations involving the media parameters and the coefficients for the field components of modified FD–ADI–FDTD equations, the value of `m` at a point (i, j) is first checked. Then the media parameters and the coefficients for that `m` are used in the computation. Thus, for each type of media the coefficients are calculated only once in the beginning of the execution of the code. This approach saves memory and computational time because the total number of media present in the computational space is usually limited and far less than the total number of grid points.

Chapter 9

Conclusion and Future Works

9.1 Conclusion

The efficiency of conventional explicit FDTD method is constrained by the upper limit on the temporal discretization, imposed by the CFL stability condition. Therefore, there is a growing interest in overcoming this limitation by employing unconditionally stable FDTD methods for which time-step and space-step can be independently chosen. This trend will continue because high accuracy in modelling is increasingly in demand. For wideband applications the FDTD methods should take the frequency dependency of the medium into account. In this thesis two unconditionally stable FDTD methods have been presented for the frequency dependent media. The first one is three-dimensional FD–CN–FDTD method, which has higher accuracy than other unconditionally stable methods, such as, ADI–FDTD method. The formulation of the FD–CN–FDTD method was presented in Chapter 3. To model frequency dependent media single–pole Debye model has been used in the FD-CN-FDTD method. By means of an auxiliary differential equation the frequency dependency has been incorporated into the proposed method. Numerical experiments validate and confirm that the method is unconditionally stable for time-steps over the CFL limit of conventional FDTD method. Although in the real-world most media are lossy, majority of the recently proposed versions of FDTD methods considered the media to be non-lossy for the sake of simplicity. The proposed FD–CN–FDTD method is capable of simulating frequency dependent as well as lossy media and for such media it is always stable at all the values of $CFLN$. The FD–CN–FDTD method can handle the simulation of free space but in this case the method has the limitation

of diverging after a certain number of time steps when $CFLN \geq 4$. Therefore, when simulating the free space in the FD–CN–FDTD method, the total number of time steps the simulation will run needs to be carefully selected so that the simulation stops before divergence starts.

The FD–CN–FDTD method requires solution of a large number of simultaneous linear equations. When the method is applied to electromagnetic problems, most of the CPU time is spent on this solution of linear algebraic equations. Therefore, an efficient solution is essential to gain the benefit of the FD–CN–FDTD method. Chapter 5 dealt with the issues related to the solution of FD–CN–FDTD sparse matrix system, which lies at the core of the method. The FD–CN–FDTD sparse matrix system can be solved by direct or iterative methods. Through numerical experiments it was shown that although direct solvers are robust and reliable, they are not practical to use in the FD–CN–FDTD method for real applications. They are computationally more expensive and require excessively large memory. For practical problems iterative solvers have to be used in the FD–CN–FDTD method. It was found out that, when $CFLN$ is increased more CPU time is needed by the iterative solvers to solve the FD–CN–FDTD sparse matrix system. Studies on the generated sparse matrix revealed that, the degradation of diagonal–dominance of the sparse matrix with increased $CFLN$ is the main reason for the increase of the CPU time needed by the solvers. Two best–known iterative methods, GMRES and BiCGStab, have been extensively studied and compared to solve the FD–CN–FDTD sparse matrix system. The comparison was made in terms of the number of iterations required to converge at different $CFLN$, CPU time and memory requirements. BiCGStab outperforms GMRES when used with FD–CN–FDTD in every aspect of the study. However, GMRES tends to narrow down the difference in CPU time with BiCGStab at higher $CFLN$ and there have been a lot of research on the development of preconditioners for GMRES. Therefore, the potential of using GMRES to use in the FD–CN–FDTD method can not be ruled out.

Many of these findings about the frequency dependent CN–FDTD method do not match with the existing literature on the frequency–independent CN–FDTD method. For example, [67] reports that GMRES is the fastest for the frequency–independent CN–FDTD method presented there. The work of [67] is

based on (3.1) and (3.2) while the FD–CN–FDTD method additionally involves (3.10) which has second-order time derivative terms. The FD–CN–FDTD method involves nine field components in place of six for the CN–FDTD method and the sparsity pattern of the former has more bands than the latter [68]. Apart from this, the concerned problem of simulation, implementation, optimization and parameters tuning have an obvious influence in concluding which solver is the most efficient.

Several studies were carried out with the commonly used preconditioners, such as, ILU(0) and SAI. Neither of these can improve the computational efficiency of the FD–CN–FDTD method. Further work is needed to tailor suitable preconditioners to improve the convergence of the iterative solvers. The change of diagonal–dominance with the $CFLN$ points a research direction in the building of appropriate preconditioners to ease the solution at higher $CFLN$.

An important finding in this thesis is that, although at higher $CFLN$ the FD-CN-FDTD method reduces the total number of iterations to complete the simulation, the computational costs at each of these iterations increase with $CFLN$. This can undermine the benefit of using temporal discretization above the CFL limit, unless the sparse solver is very efficient. This issue can potentially be addressed by either parallelising the FD–CN–FDTD code or using appropriate preconditioners. The use of BiCGStab and GMRES in the FD–CN–FDTD method, as was done in this thesis, is relevant in this context. The algorithm of BiCGStab involves performing matrix-vector product and arithmetic operations on the vectors (*e.g.* vector-vector addition and dot-product of two vectors). As these operations possess inherent parallelism, in the parallelisation of the FD–CN–FDTD method BiCGStab is likely to be the appropriate solver. On the other hand, a lot of research on the preconditioners for GMRES has been carried out by the researchers of applied mathematics. The acquired knowledge on such preconditioners and their computer code might be useful for the development of suitable preconditioners for the FD–CN–FDTD method.

The implementation of the FD–CN–FDTD method in serial code is presented in Chapter 8. There are scarce precedence of implementation of CN–FDTD

method in computer programmes and no precedence of implementation of frequency dependent CN–FDTD method which is more complicated. Therefore the presented implementation techniques bear significance and would be useful for further research on the CN–FDTD method. Chapter 8 also presented the parallel implementation of the FD–CN–FDTD method in OpenMP. However, with OpenMP the perfect speed-up was not achieved. This is not unusual when parallelisation is done using OpenMP. The benefits of parallelization can be fully achieved by using MPI in the distributed memory architecture.

Chapter 6 described an application of the FD–CN–FDTD method. Using the FD–CN–FDTD method a simulation model of the human body was developed with all the fine structures and frequency dependent dielectric properties of the human tissues. With a view to study bioelectromagnetic therapies, specifically DBS, numerical simulation of electromagnetic wave propagation inside the human head has been shown in this chapter. However, modelling of the DBS system has been performed in a very simplistic way. A more realistic model should use the parameters described in, for example, [129]. Numerical simulation of the human body in the FD–CN–FDTD method presented in this thesis forms a significant footing for further research on bioelectromagnetics using this method. It will be useful for the research towards the understanding of bioelectromagnetic therapies, to accurately predict the excitation of the targeted tissues inside the human head in DBS system and to develop techniques for non-invasive DBS. Some research directions to develop such non-invasive DBS technique have also been suggested in Chapter 6.

The second unconditionally stable FDTD method presented in this thesis is the accuracy improved FD–ADI–FDTD method. The ADI–FDTD method provides a computationally affordable approximation of the CN–FDTD method by adding a perturbation term to the latter. Therefore, the ADI–FDTD method exhibits a loss of accuracy with respect to the CN–FDTD method that may become severe for some practical applications, especially when large time-steps are used. The modified frequency dependent ADI–FDTD method presented in this thesis can improve the accuracy of the normal frequency dependent ADI–FDTD method. The modified method does not increase the computational costs significantly because it still requires to solve the same tridiagonal matrix system

as is required by the normal method. The modified FD–ADI–FDTD method presented in this thesis is for two-dimension and needs to be extended to three-dimension.

9.2 Suggested Future Works

Based on the discussions and limitations of the works presented in this thesis, some potential future research-works are suggested below.

In the FD-CN-FDTD method, the CPU time spent on the solution of the sparse matrix system becomes increasingly longer with the increase of $CFLN$, because the matrix becomes ill-conditioned. To take the ultimate benefit of using higher $CFLN$, the solution time of the FD–CN–FDTD sparse matrix system needs to be improved. For this further research is needed to find a suitable preconditioner which will improve the convergence of the iterative solver. The fact that, the diagonal–dominance of the coefficient matrix of the FD–CN–FDTD method deteriorates when $CFLN$ increases, gives some hints on this issue. The idea presented in [130] to make a given matrix strictly diagonally dominant can be useful in building a suitable preconditioner.

Numerical simulation of practical applications, such as, modelling whole human body requires enormously large amount of memory and very long simulation time. One of the ways to overcome these constraints is to parallelize the FD–CN–FDTD code. This thesis presented parallelization in OpenMP in shared memory architecture. But to achieve better performance, parallelization has to be done in MPI in the distributed memory architecture.

Modelling of the human body in the FD–CN–FDTD method as presented in Chapter 6 serves as the foundation for many future researches on bioelectromagnetics. For example, research should be conducted on the backpropagation and target localization by FDTD time reversal algorithms [118] in order to develop a non-invasive DBS technique. One application of the numerical human body model is estimation of the parameters that describe the electromagnetic field interaction with the human body, such as, specific absorption rate (SAR). By estimating SAR the amount of radio wave energy absorbed by the human

body during the use of mobile phones or other wireless devices can be measured. Another potential research can be numerical modelling of body-centric wireless networks. It will be interesting to perform FD–CN–FDTD based on-body propagation channel modelling and study the effects of the dynamic on-body environment on the communication path between body-worn devices, considering various body postures.

Appendix A

List of Publications

Journals, Peer-reviewed

- Hasan Khaled Rouf, Fumie Costen and Salvador Garcia, "3D Crank–Nicolson finite difference time domain method for dispersive media," IET Electronics Letters, Volume 45, Issue 19, 2009, DOI: 10.1049/el.2009.1940. *(Chapter 3 and Chapter 4)*

- Hasan Khaled Rouf, Fumie Costen and Salvador Garcia, "On the solution of 3D frequency dependent Crank–Nicolson FDTD scheme," Journal of Electromagnetic Waves and Applications, Volume 23, Issue 16, 2009, DOI: 10.1163/156939309790109261. *(Chapter 5)*

- Hasan Khaled Rouf, Fumie Costen and Salvador Garcia, "Reduction of the numerical errors in frequency dependent ADI–FDTD," IET Electronics Letters, Volume 46, Issue 7, 2010, DOI: 10.1049/el.2010.0322. *(Chapter 7)*

Conference Publications, Refereed

- Hasan Khaled Rouf, Fumie Costen and Salvador Garcia, "Improving the accuracy of frequency dependent ADI–FDTD," IEEE AP–S International Symposium and USNC/URSI Radio Science Meeting, 11–17 July, 2010, Toronto, Ontario, Canada.

- Hasan Khaled Rouf, Fumie Costen and Salvador Garcia, "A frequency dependent implicit FDTD scheme," IEEE AP–S International Symposium

and USNC/URSI Radio Science Meeting, 1–5 June, 2009, SC, USA.

- Hasan Khaled Rouf, Fumie Costen and Anthony Brown, "Overcoming CFL constraint of FDTD in frequency dependent environment," Fifth International Conference of Applied Mathematics and Computing, Plovdiv, Bulgaria, August 12–18, 2008.

Conference Publications, Other

- Hasan Khaled Rouf, Fumie Costen, Salvador Garcia and Seiji Fujino, "Efficient solvers for the frequency dependent Crank Nicolson FDTD method," International Kyoto Forum on Krylov Subspace Methods, 25–27 March, 2010, Kyoto, Japan.

- Hasan Khaled Rouf, Fumie Costen and Anthony Brown, "Frequency dependent Crank Nicholson Finite Difference Time Domain (FDTD) method for Ultra Wide Band (UWB) systems," 10th International PhD Workshop, 18–21 October, 2008, Gliwice, Poland.

- Hasan Khaled Rouf, Fumie Costen and Anthony Brown, "Frequency dependent Crank Nicholson scheme for 3D electromagnetic field problems," Graduate Research Conference, September, 2008, University of Manchester, Manchester, UK.

Publications, In progress

- Hasan Khaled Rouf and Fumie Costen, "Numerical modelling of the human body in frequency dependent Crank Nicolson FDTD method," IET Microwaves, Antennas and Propagation, (in progress). (Chapter 6).

Appendix B

Mur's ABC

Numerical Formulation from One-Way Wave Equation

From the Maxwell's equation $\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} = -\frac{\partial \mu \mathbf{H}}{\partial t}$, taking curl on both sides of equality

$$\nabla \times \nabla \times \mathbf{E} = -\nabla \times \frac{\partial \mu \mathbf{H}}{\partial t} = -\mu \frac{\partial \nabla \times \mathbf{H}}{\partial t} = -\mu \frac{\partial \frac{\partial \mathbf{D}}{\partial t}}{\partial t} = -\mu \frac{\partial^2 \mathbf{D}}{\partial t^2} = -\mu\epsilon \frac{\partial^2 \mathbf{E}}{\partial t^2} \quad \text{(B.1)}$$

Assuming that boundary is source-free ($\nabla \bullet \mathbf{E} = 0$), utilizing $\nabla\nabla \times \mathbf{E} = \nabla(\nabla \bullet \mathbf{E}) - (\nabla \bullet \nabla)\mathbf{E}$, (B.1) is modified as follows:

$$-(\nabla \bullet \nabla)\mathbf{E} + \mu\epsilon \frac{\partial^2 \mathbf{E}}{\partial t^2} = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} - \mu\epsilon \frac{\partial^2}{\partial t^2})\mathbf{E} =$$

$$(\frac{\partial^2}{\partial x^2} - \mu\epsilon \frac{\partial^2}{\partial t^2}(1 - \frac{\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}}{\mu\epsilon \frac{\partial^2}{\partial t^2}}))\mathbf{E} =$$

$$\left(\frac{\partial}{\partial x} - \sqrt{\mu\epsilon}\frac{\partial}{\partial t}\sqrt{1 - \frac{\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}}{\mu\epsilon \frac{\partial^2}{\partial t^2}}}\right) \bullet \left(\frac{\partial}{\partial x} + \sqrt{\mu\epsilon}\frac{\partial}{\partial t}\sqrt{1 - \frac{\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}}{\mu\epsilon \frac{\partial^2}{\partial t^2}}}\right)\mathbf{E} = 0 \quad \text{(B.2)}$$

[1]Appendix B is cited from [98]

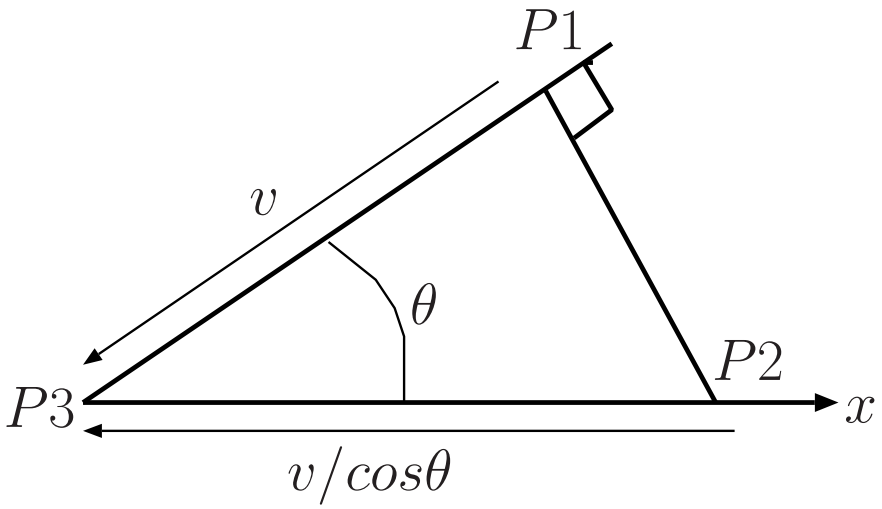


Figure B.1: Phase velocity concept

(B.2) means that forward travelling wave × backward travelling wave = 0.

Propagation constants Γ for x, y and z directions satisfy the following:

$$\Gamma_x^2 + \Gamma_y^2 + \Gamma_z^2 + \omega^2\epsilon\mu = 0 \tag{B.3}$$

The propagation constant Γ consists of attenuation element β and phase element κ as follows

$$\begin{aligned}
\Gamma_x &= \jmath\kappa_x + \beta_x = \jmath\frac{2\pi}{\lambda_x} + \beta_x \\
\Gamma_y &= \jmath\kappa_y + \beta_y = \jmath\frac{2\pi}{\lambda_y} + \beta_y \\
\Gamma_z &= \jmath\kappa_z + \beta_z = \jmath\frac{2\pi}{\lambda_z} + \beta_z
\end{aligned} \tag{B.4}$$

The assumption of $\omega t - \Gamma_x x - \Gamma_y y - \Gamma_z z =$ constant leads to the phase velocity for the x direction as follows.

$$\frac{\partial x}{\partial t} = \frac{\omega}{\Gamma_x} \tag{B.5}$$

It is assumed that the plane wave arrives with the angle of θ as in Fig. B.1. When the wave moves from the point P1 to P3 (Fig. B.1), the point P2 moves

to P3 along x-axis. In the free space where the wave moves a distance υ from P1 to P3, the length between P2 to P3, that is, $\frac{\upsilon}{\cos\theta}$, is the phase velocity in the x direction.

$$\frac{\partial x}{\partial t} = \frac{\upsilon}{\cos\theta} \tag{B.6}$$

On the other hand, in the free space where it can be assumed that the attenuation constant β is zero, these phase constants κ satisfy the following.

$$\kappa_x^2 + \kappa_y^2 + \kappa_z^2 = \omega^2\epsilon\mu \tag{B.7}$$

Therefore, phase velocity for the x direction is

$$\frac{\omega}{\kappa_x} = \frac{\omega}{\sqrt{\omega^2\epsilon\mu - \kappa_y^2 - \kappa_z^2}} = \frac{1}{\sqrt{\epsilon\mu(1 - (\frac{\kappa_y}{\omega})^2 - (\frac{\kappa_z}{\omega})^2)}} \tag{B.8}$$

At $x = i_{min}$, that is, the boundary parallel to the y-z plain, the forward travelling wave = 0 and at $x = i_{max}$, the backward travelling wave = 0. At $x = i_{min}$, where it is assumed that κ_y and κ_z are nearly zero, one-term Taylor approximation yields

$$(\frac{\partial}{\partial x} - \sqrt{\mu\epsilon}\frac{\partial}{\partial t})\mathbf{E} = (\frac{\partial}{\partial x} - \frac{\kappa_x}{\omega}\frac{\partial}{\partial t})\mathbf{E} = 0 \tag{B.9}$$

In the same way, at $x = i_{max}$, one-term Taylor approximation yields

$$(\frac{\partial}{\partial x} + \sqrt{\mu\epsilon}\frac{\partial}{\partial t})\mathbf{E} = (\frac{\partial}{\partial x} + \frac{\kappa_x}{\omega}\frac{\partial}{\partial t})\mathbf{E} = 0 \tag{B.10}$$

At $x = i_{min}$, two-terms Taylor approximation yields

$$\left(\frac{\partial}{\partial x} - \sqrt{\mu\epsilon}\frac{\partial}{\partial t}(1 - \frac{1}{2}\frac{\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}}{\mu\epsilon\frac{\partial^2}{\partial t^2}})\right)\mathbf{E} = \left(\sqrt{\mu\epsilon}\frac{\partial}{\partial t}\frac{\partial}{\partial x} - \mu\epsilon\frac{\partial^2}{\partial t^2} + \frac{1}{2}(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2})\right)\mathbf{E} = 0 \tag{B.11}$$

In the same way, at $x = i_{max}$, two-terms Taylor approximation yields

$$\left(\frac{\partial}{\partial x} + \sqrt{\mu\epsilon}\frac{\partial}{\partial t}(1 - \frac{1}{2}\frac{\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}}{\mu\epsilon\frac{\partial^2}{\partial t^2}}) \right) \mathbf{E} = \left(\sqrt{\mu\epsilon}\frac{\partial}{\partial t}\frac{\partial}{\partial x} + \mu\epsilon\frac{\partial^2}{\partial t^2} - \frac{1}{2}(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}) \right) \mathbf{E} = 0 \tag{B.12}$$

Discretization of these equations is required to implement these ABCs into the FDTD method, following Taylor's series approximations in (B.13) of the solutions to the partial differential equations:

$$\begin{aligned} \mathbf{E}|_{x+h} &= \mathbf{E}|_x + h\frac{\partial \mathbf{E}|_x}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 \mathbf{E}|_x}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3 \mathbf{E}|_x}{\partial x^3} \\ \mathbf{E}|_{x-h} &= \mathbf{E}|_x - h\frac{\partial \mathbf{E}|_x}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 \mathbf{E}|_x}{\partial x^2} - \frac{1}{6}h^3\frac{\partial^3 \mathbf{E}|_x}{\partial x^3} \end{aligned} \tag{B.13}$$

(B.13) produces the following approximations for the first and second derivative of $\mathbf{E}$:

$$\begin{aligned} \frac{\partial \mathbf{E}|_x}{\partial x} &\sim \frac{\mathbf{E}|_{x+h} - \mathbf{E}|_{x-h}}{2h} \\ \frac{\partial^2 \mathbf{E}|_x}{\partial x^2} &\sim \frac{\mathbf{E}|_{x+h} - 2\mathbf{E}|_x + \mathbf{E}|_{x-h}}{h^2} \end{aligned} \tag{B.14}$$

(B.14) is second-order accurate in the discretization, *i.e.* $O(h^2)$. This means that the error will be reduced by one-fourth if a spatial discretization is reduced by one-half. If terms which are $O(h^2)$ and higher than $O(h^2)$ are ignored, the following first-order forward and backward approximations to $\frac{\partial \mathbf{E}|_x}{\partial x}$ is obtained:

$$\begin{aligned} \frac{\partial \mathbf{E}|_x}{\partial x} &\sim \frac{\mathbf{E}|_{x+h} - \mathbf{E}|_x}{h} \\ \frac{\partial \mathbf{E}|_x}{\partial x} &\sim \frac{\mathbf{E}|_x - \mathbf{E}|_{x-h}}{h} \end{aligned} \tag{B.15}$$

At the y-z plane boundary, the boundary condition is placed to $E_{y(i,j,k)}$ and $E_{z(i,j,k)}$. As an example, the boundary condition is placed to $E_{y(i,j,k)}$ at $x = i_{min}$.

First-order Mur's ABC

Based on (B.15) and (B.14), (B.9), (B.10), (B.11), (B.12) are discretized as follows: In case of the first-order approximation, (B.9) is discretized as

$$E_y^{n+1}{}_{(i_{min},j,k)} = \frac{\Delta t - \Delta x\sqrt{\mu^{n+1}{}_{(i_{min+1},j,k)}\epsilon^{n+1}{}_{(i_{min+1},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{min},j,k)}\epsilon^{n+1}{}_{(i_{min},j,k)}}} E_y^{n+1}{}_{(i_{min+1},j,k)} \quad \text{(B.16)}$$
$$+\frac{\Delta t + \Delta x\sqrt{\mu^{n}{}_{(i_{min+1},j,k)}\epsilon^{n}{}_{(i_{min+1},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{min},j,k)}\epsilon^{n+1}{}_{(i_{min},j,k)}}} E_y^{n}{}_{(i_{min+1},j,k)}$$
$$-\frac{\Delta t - \Delta x\sqrt{\mu^{n}{}_{(i_{min},j,k)}\epsilon^{n}{}_{(i_{min},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{min},j,k)}\epsilon^{n+1}{}_{(i_{min},j,k)}}} E_y^{n}{}_{(i_{min},j,k)}$$

Similarly, (B.10) is discretized as

$$E_y^{n+1}{}_{(i_{max},j,k)} = \frac{\Delta t - \Delta x\sqrt{\mu^{n+1}{}_{(i_{max-1},j,k)}\epsilon^{n+1}{}_{(i_{max-1},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{max},j,k)}\epsilon^{n+1}{}_{(i_{max},j,k)}}} E_y^{n+1}{}_{(i_{max-1},j,k)} \quad \text{(B.17)}$$
$$+\frac{\Delta t + \Delta x\sqrt{\mu^{n}{}_{(i_{max-1},j,k)}\epsilon^{n}{}_{(i_{max-1},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{max},j,k)}\epsilon^{n+1}{}_{(i_{max},j,k)}}} E_y^{n}{}_{(i_{max-1},j,k)}$$
$$-\frac{\Delta t - \Delta x\sqrt{\mu^{n}{}_{(i_{max},j,k)}\epsilon^{n}{}_{(i_{max},j,k)}}}{\Delta t + \Delta x\sqrt{\mu^{n+1}{}_{(i_{max},j,k)}\epsilon^{n+1}{}_{(i_{max},j,k)}}} E_y^{n}{}_{(i_{max},j,k)}$$

(B.16) and (B.17) are obtained using (B.18).

$$\frac{\partial E_y^{n+1}{}_{(i_{min},j,k)}}{\partial x} = \quad \text{(B.18)}$$
$$\frac{\frac{E_y^{n+1}{}_{(i_{min+1},j,k)} - E_y^{n+1}{}_{(i_{min},j,k)}}{\Delta x} + \frac{E_y^{n}{}_{(i_{min+1},j,k)} - E_y^{n}{}_{(i_min,j,k)}}{\Delta x}}{2}$$
$$\frac{\partial E_y^{n+1}{}_{(i_{min},j,k)}}{\partial t} =$$
$$\frac{\frac{E_y^{n+1}{}_{(i_{min+1},j,k)} - E_y^{n}{}_{(i_{min+1},j,k)}}{\Delta t} + \frac{E_y^{n+1}{}_{(i_{min},j,k)} - E_y^{n}{}_{(i_{min},j,k)}}{\Delta t}}{2}$$

(B.16) and (B.17) are called Mur's first-order ABC which is versatile on all types of models. This algorithm can result in a large percentage of the outgoing energy

being reflected from the boundary if materials with high dielectric constants are present. In the same way as (B.16) and (B.17) all the equations for Mur's first-order ABC are derived below:

Mur's ABC to update $E_{x(i,j,k)}$

$$E_x^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta y\sqrt{\mu_{(i,j+1,k)}\epsilon_{(i,j+1,k)}})\, E_x^{n+1}{}_{(i,j+1,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.19}$$
$$+\frac{(\Delta t + \Delta y\sqrt{\mu_{(i,j+1,k)}\epsilon_{(i,j+1,k)}})\, E_x^n{}_{(i,j+1,k)} - (\Delta t - \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_x^n{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min}+1 \le i \le i_{max}, j = j_{min}, k_{min} \le k \le k_{max}].$$

$$E_x^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta y\sqrt{\mu_{(i,j-1,k)}\epsilon_{(i,j-1,k)}})\, E_x^{n+1}{}_{(i,j-1,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.20}$$
$$+\frac{(\Delta t + \Delta y\sqrt{\mu_{(i,j-1,k)}\epsilon_{(i,j-1,k)}})\, E_x^n{}_{(i,j-1,k)} - (\Delta t - \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_x^n{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min}+1 \le i \le i_{max}, j = j_{max}, k_{min} \le k \le k_{max}].$$

$$E_x^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu_{(i,j,k+1)}\epsilon_{(i,j,k+1)}})\, E_x^{n+1}{}_{(i,j,k+1)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.21}$$
$$+\frac{(\Delta t + \Delta z\sqrt{\mu_{(i,j,k+1)}\epsilon_{(i,j,k+1)}})\, E_x^n{}_{(i,j,k+1)} - (\Delta t - \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_x^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min}+1 \le i \le i_{max}, j_{min} \le j \le j_{max}, k = k_{min}].$$

$$E_x^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu_{(i,j,k-1)}\epsilon_{(i,j,k-1)}})\, E_x^{n+1}{}_{(i,j,k-1)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.22}$$
$$+\frac{(\Delta t + \Delta z\sqrt{\mu_{(i,j,k-1)}\epsilon_{(i,j,k-1)}})\, E_x^n{}_{(i,j,k-1)} - (\Delta t - \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_x^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min}+1 \le i \le i_{max}, j_{min} \le j \le j_{max}, k = k_{max}].$$

Mur's ABC to update $E_{y(i,j,k)}$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta x\sqrt{\mu_{(i+1,j,k)}\epsilon_{(i+1,j,k)}})\, E_y^{n+1}{}_{(i+1,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.23}$$
$$+\frac{(\Delta t + \Delta x\sqrt{\mu_{(i+1,j,k)}\epsilon_{(i+1,j,k)}})\, E_y^n{}_{(i+1,j,k)} - (\Delta t - \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_y^n{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i = i_{min}, j_{min} + 1 \le j \le j_{max}, k_{min} \le k \le k_{min}].$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta x\sqrt{\mu_{(i-1,j,k)}\epsilon_{(i-1,j,k)}})\, E_y^{n+1}{}_{(i-1,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.24}$$
$$+\frac{(\Delta t + \Delta x\sqrt{\mu_{(i-1,j,k)}\epsilon_{(i-1,j,k)}})\, E_y^n{}_{(i-1,j,k)} - (\Delta t - \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_y^n{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i = i_{max}, j_{min} + 1 \le j \le j_{max}, k_{min} \le k \le k_{min}].$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu_{(i,j,k+1)}\epsilon_{(i,j,k+1)}})\, E_y^{n+1}{}_{(i,j,k+1)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.25}$$
$$+\frac{(\Delta t + \Delta z\sqrt{\mu_{(i,j,k+1)}\epsilon_{(i,j,k+1)}})\, E_y^n{}_{(i,j,k+1)} - (\Delta t - \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_y^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min} \le i \le i_{max}, j_{min} + 1 \le j \le j_{max}, k = k_{min}].$$

$$E_y^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta z\sqrt{\mu_{(i,j,k-1)}\epsilon_{(i,j,k-1)}})\, E_y^{n+1}{}_{(i,j,k-1)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.26}$$
$$+\frac{(\Delta t + \Delta z\sqrt{\mu_{(i,j,k-1)}\epsilon_{(i,j,k-1)}})\, E_y^n{}_{(i,j,k-1)} - (\Delta t - \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_y^n{}_{(i,j,k)}}{\Delta t + \Delta z\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min} \le i \le i_{max}, j_{min} + 1 \le j \le j_{max}, k = k_{max}].$$

Mur's ABC to update $E_{z(i,j,k)}$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{(\Delta t - \Delta x\sqrt{\mu_{(i+1,j,k)}\epsilon_{(i+1,j,k)}})\, E_z^{n+1}{}_{(i+1,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.27}$$
$$+\frac{(\Delta t + \Delta x\sqrt{\mu_{(i+1,j,k)}\epsilon_{(i+1,j,k)}})\, E_z^n{}_{(i+1,j,k)} - (\Delta t - \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}})\, E_z^n{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i = i_{min}, j_{min} \le j \le j_{max}, k_{min} + 1 \le k \le k_{min}].$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta x\sqrt{\mu_{(i-1,j,k)}\epsilon_{(i-1,j,k)}}\right) E_z^{n+1}{}_{(i-1,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.28}$$
$$+\frac{\left(\Delta t + \Delta x\sqrt{\mu_{(i-1,j,k)}\epsilon_{(i-1,j,k)}}\right) E_z^{n}{}_{(i-1,j,k)} - \left(\Delta t - \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}\right) E_z^{n}{}_{(i,j,k)}}{\Delta t + \Delta x\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i = i_{max}, j_{min} \leq j \leq j_{max}, k_{min} + 1 \leq k \leq k_{min}].$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta y\sqrt{\mu_{(i,j+1,k)}\epsilon_{(i,j+1,k)}}\right) E_z^{n+1}{}_{(i,j+1,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.29}$$
$$+\frac{\left(\Delta t + \Delta y\sqrt{\mu_{(i,j+1,k)}\epsilon_{(i,j+1,k)}}\right) E_z^{n}{}_{(i,j+1,k)} - \left(\Delta t - \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}\right) E_z^{n}{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min} \leq i \leq i_{max}, j = j_{min}, k_{min} + 1 \leq k \leq k_{max}].$$

$$E_z^{n+1}{}_{(i,j,k)} = \frac{\left(\Delta t - \Delta y\sqrt{\mu_{(i,j-1,k)}\epsilon_{(i,j-1,k)}}\right) E_z^{n+1}{}_{(i,j-1,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}} \tag{B.30}$$
$$+\frac{\left(\Delta t + \Delta y\sqrt{\mu_{(i,j-1,k)}\epsilon_{(i,j-1,k)}}\right) E_z^{n}{}_{(i,j-1,k)} - \left(\Delta t - \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}\right) E_z^{n}{}_{(i,j,k)}}{\Delta t + \Delta y\sqrt{\mu_{(i,j,k)}\epsilon_{(i,j,k)}}}$$
$$[i_{min} \leq i \leq i_{max}, j = j_{max}, k_{min} + 1 \leq k \leq k_{max}].$$

In the implementation of FD–CN–FDTD and modified FD–ADI–FDTD methods $\epsilon = \epsilon_0\epsilon_r$ is used in the above Mur's ABC equations where ϵ_r comes from (3.7) which is a function of Debye media parameters and angular frequency. For Mur's ABCs the centre frequency is used to compute the real part of ϵ in (3.7). Thus the Mur's ABCs are frequency independent and fixed at the centre frequency.

Bibliography

[1] C. Gabriel. Compilation of the dielectric properties of body tissues at RF and microwave frequencies. *Report N.AL/OE-TR- 1996-0037, Occupational and Environmental Health Directorate, Radiofrequency Radiation Division, Brooks Air Force Base, Texas (USA)*, June 1996.

[2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM Press, Philadelphia, 1993.

[3] M.N. Sadiku. *Numerical Techniques in Electromagnetics*. CRC Press, second edition edition, 2000.

[4] R.F. Harrington. *Field Computation by Moment Methods*. The McMillan Company, New York, 1968.

[5] R.F. Harrington. The method of moments - a personal review. *IEEE Antennas and Propagation Society International Symposium, 2000*, 3, July 2000.

[6] P. B. Johns and R. L. Beurle. Numerical solution of two dimensional scattering problems using a transmission-line matrix. *Proceedings of IEE*, 118(12):1203–1208, 1971.

[7] M. N. O. Sadku. A simple introduction to finite element analysis of electromagnetics problems. *IEEE Transactions on Education*, 32(2):85–93, 1989.

[8] K. Ise, K. Inoue, and M. Koshiba. Three-dimensional finite-element solution of dielectric scattering obstacles in a rectangular waveguide. *IEEE Transactions on Microwave Theory and Techniques*, 38(9):1352–1359, September 1990.

[9] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, AP-14, 1966.

[10] K. L. Shlager and J. B. Schneider. A selective survey of the finite difference time domain literature. *IEEE Antennas and Propagation Magazine*, 37-4:39–56, 1995.

[11] Akira Ishimaru. *Electromagnetic Wave Propagation, Radiation and Scattering*. Prentice Hall, 1991.

[12] A. Taflove and S. Hagness. *Computational Electromagnetics. The Finite–Difference Time–Domain Method*. Artech House, Boston, MA, 2000.

[13] S. Gonzalez Garcia1, Fumie Costen, M. Fernandez Pantoja1, Anthony Brown, and A. Rubio Bretones. Open issues in unconditionally stable schemes. In *Progress in Electromagnetics Research Symposium Abstracts*, 2009.

[14] Kung F. and Chuah H.T. Stability of classical finite-difference time-domain (FDTD) formulation with nonlinear elements–a new perspective. *Journal of Electromagnetic Waves and Applications*, 17(9):1313–1314, 2003.

[15] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43:50–67, 1947.

[16] E. L. Tan. Efficient algorithms for Crank Nicolson based finite difference time domain methods. *IEEE Transactions on Microwave Theory and Techniques*, 56(2):408–413, February, 2008.

[17] T. Namiki. A new FDTD algorithm based on alternating-direction implicit method. *IEEE Transactions on Microwave Theory and Techniques*, 47:2003–2007, 1999.

[18] G. Sun and C. Trueman. Unconditionally stable Crank-Nicolson scheme for solving two dimensional Maxwell's equations. *Electronics Letters*, 39:595 – 597, 2003.

[19] G. Sun and C. Trueman. Efficient implementations of the Crank-Nicolson scheme for the finite-difference time-domain method. *IEEE Transactions on Microwave Theory and Techniques*, 54(5):2275– 2284, May 2006.

[20] G. Sun and C. Trueman. Unconditionally-stable FDTD method based on Crank-Nicolson scheme for solving three dimensional Maxwell's equations. *Electronics Letters*, 40:589–590, 2004.

[21] S. G. Garcia, R. G. Rubio, A. R. Bretones, and R. G. Martin. On the dispersion relation of ADI–FDTD. *IEEE Microwave and Wireless Component Letters*, 16(6):354–356, June 2006.

[22] Y. Yang, R. Chen, and E. Yung. The unconditionally stable Crank Nicolson FDTD method for three-dimensional Maxwell's equations. *Microwave and Optical Technology Letters*, 48:1619–1622, 2006.

[23] S. G. Garcia, T.W. Lee, and S.C. Hagness. On the accuracy of the ADI-FDTD method. *IEEE Antennas and Wireless Propagation Letters*, 1:31–34, 2002.

[24] S. Gonzalez Garcia, A. Rubio Bretones, B. Garcia Olmedo, and R. Gomez Martin. *Finite Difference Time Domain Methods*, pages 91–132. WIT Press, 2003.

[25] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. CRC Press Inc, August 1995.

[26] S. Gonzalez Garcia, A. Rubio Bretones, B. Garcia Olmedo, and R. Gomez Martin. *New Trends in FDTD Methods in Computational Electrodynamics: Unconditionally Stable Schemes*, pages 55–96. Transworld Research Network, 2005.

[27] A. Taflove and S. Hagness. *Computational Electrodynamics: The Finite–Difference Time–Domain Method, third ed.* Artech House, Boston, MA, 2005.

[28] G. Mur. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic-field equations. *IEEE Transactions on Electromagnetic Compatibility*, 23:377–382, 1981.

[29] A. Taflove. *Advances in computational electrodynamics: the finite-difference time-domain method*. Artech House, Boston, MA, 1998.

[30] K. Mei and J. Fang. Superabsorption - a method to improve absorbing boundary conditions. *IEEE Transactions on Antennas and Propagation*, 40(9):1001–1010, September 1992.

[31] Z. P. Liao, H. L. Wong, G. P. Yang, and Y. F. Yuan. A transmitting boundary for transient wave analysis. *Scietia Sinica*, 28:1063–1076, 1984.

[32] V. Betz and R. Mittra. Comparison and evaluation of boundary conditions for the absorption of guided waves in an FDTD simulation. *IEEE Microwave and Guided Wave Letters*, 2(12):499–501, December 1992.

[33] J.-P. Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comp. Phys.*, 114:185–200, 1994.

[34] S. Gedney. An anisotropic perfectly matched layer – absorbing medium for the truncation of FDTD lattices. *IEEE Transactions on Antennas and Propagation*, 44(12):1630–1639, December 1996.

[35] W. Chew and W. Wood. A 3-D perfectly matched medium from modified Maxwell's equations with stretched coordinates. *Microwave and Optical Technology Letters*, 7:599–604, 1994.

[36] C. Rapaport. Perfectly matched absorbing boundary conditions based on anisotropic lossy mapping of space. *IEEE Microwave and Guided Wave Letters*, 5:90–92, 1995.

[37] J. Roden and S. Gedney. Convolution PML (CPML): An efficient FDTD implementation of the CFS-PML for arbitrary medium. *Microwave and Optical Technology Letters*, 27(5):334–339, 2000.

[38] D. Werner and R. Mittra. A new field scaling interpretation of Berenger's PML and its comparison to other PML formulations. *Microwave and Optical Technology Letters*, 16(2):103–106, 1997.

[39] J. Svigelj and R. Mittra. The dispersive boundary condition applied to nonuniform orthogonal meshes. *IEEE Transactions on Microwave Theory and Techniques*, 47(3), March, 1999.

[40] Wenhua Yu, Raj Mittra, Tao Su, Yongjun Liu, and Xiaoling Yang. *Parallel Finite-Difference Time-Domain Method*. Artech House Publishers, 2006.

[41] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Mathematics of Computation*, 31:629–651, 1977.

[42] C. Gabriel, S. Gabriel, and E. Corthout. The dielectric properties of biological tissues: I. literature survey. *Physics in Medicine and Biology*, 41:2231–2249, 1996.

[43] S. Gabriel, R.W. Lau, and C. Gabriel. The dielectric properties of biological tissues: II. measurements in the frequency range 10 Hz to 20 GHz. *Physics in Medicine and Biology*, 41:2251–2269, 1996.

[44] S. Gabriel, R.W. Lau, and C.Gabriel. The dielectric properties of biological tissues: III. parametric models for the dielectric spectrum of tissues. *Physics in Medicine and Biology*, 41:2271–2293, 1996.

[45] John A. Buck William H. Hayt. *Engineering Electromagnetics*. McGraw-Hill Publishing Co., 2001.

[46] P. Debye. *Polar Molecules*. Dover, New York, 1929.

[47] L. Xu and S. C. Hagness. A confocal microwave imaging algorithm for breast cancer detection. *IEEE Microwave and Wireless Component Letters*, 11:130–132, 2001.

[48] A. Taflove and M.E. Brodwin. Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations. *IEEE Transactions on Microwave Theory and Techniques*, 23(8):623–630, 1975.

[49] M.N.O. Sadiku, S.O. Agbo, and V. Bemmel. Stability criterion for finite-difference time-domain algorithm. *IEEE Southeastcon Proceedings*, 1:48–50, April 1990.

[50] F. Zheng, Z. Chen, and J. Zhang. A finite difference time domain method without the Courant stability conditions. *IEEE Microwave and Guided Wave Letters*, 9:441–443, 1999.

[51] S. G. Garcia, T. Lee, and S. C. Hagness. On the accuracy of the ADI–FDTD method. *IEEE Antennas and Wireless Propagation Letters*, 1(1):31–34, 2002.

[52] S. G. Garcia, F. Costen, A. R. Bretones, and A. Brown. State–of–the–art in unconditionally stable FDTD schemes. In *IEEE–AP/URSI 2007*, July 2007.

[53] F. Zheng and Z. Chen. Numerical dispersion analysis of the unconditionally stable 3-D ADI-FDTD method. *IEEE Transactions on Microwave Theory and Techniques*, 49(5):1006–1009, May 2001.

[54] W. Fu and E.L. Tan. Stability and dispersion analysis for ADI-FDTD method in lossy media. *IEEE Transactions on Antennas and Propagation*, 55(4):1095 – 1102, April 2007.

[55] T. W. Lee and S. C. Hagness. Wave source conditions for the unconditionally stable ADI-FDTD method. In *IEEE Antennas and Propagation Society International Symposium*, pages 142–145, 2001.

[56] Shumin Wang, F.L. Teixeira, and Ji Chen. An iterative ADI-FDTD with reduced splitting error. *IEEE Microwave and Wireless Components Letters*, 15(2):92–94, February 2005.

[57] I. Ahmed and Z. Chen. Error reduced ADI-FDTD methods. *IEEE Antennas and Wireless Propagation Letters*, 4:323–325, 2005.

[58] B.D. Welferta. Analysis of iterated ADI-FDTD schemes for Maxwell's curl equations. *Journal of Computational Physics*, 222(1):9–27, 2007.

[59] I. Ahmed and Z. Chen. Accuracy improved ADI-FDTD methods. *International Journal of Numerical Modeling, Electronic networks, devices and fields*, 20:35–46, November 2006.

[60] J. Chen, Z. Wang, and Y. Chen. Higher-order alternative direction implicit FDTD method. *Electronics Lett.*, 38:1321–1322, 2002.

[61] J. Shibayama, M. Muraki, J. Yamauchi, and H. Nakano. Efficient implicit FDTD algorithm based on locally one-dimensional scheme. *Electronics Letters*, 41(19):10461047, September 2005.

[62] J. Shibayama, R. Takahashi, J. Yamauchi, and H. Nakano. Frequency-dependent LOD-FDTD implementations for dispersive media. *Electronics Letters*, 42(19):10841086, September 2006.

[63] E. L. Tan. Unconditionally stable LOD-FDTD method for 3-D Maxwell's equations. *IEEE Microwave Theory and Wireless Component Letters*, 17(2):8587, February 2007.

[64] I. Ahmed, Eng-Kee Chua, Er-Ping Li, and Zhizhang Chen. Development of the three-dimensional unconditionally stable LOD-FDTD method. *IEEE Transactions on Antennas and Propagation*, 56(11):3596–3600, November 2008.

[65] J. Lee and B. Fornberg. A split step approach for the 3-D Maxwell's equations. *Journal of Computations and Applied Mathematics*, 158(2):485505, September 2003.

[66] W. Fu and E. L. Tan. Development of split step FDTD method with higher order spatial accuracy. *Electronics Letters*, 40, 2004.

[67] Y. Yang, R. Chen, D. Ding, Z. Fan, and E. Yung. Application of iterative solvers in 3D Crank-Nicolson FDTD method for simulating resonant frequencies of the dielectric cavity. In *Asia-Pacific Microwave Conference,*, 2007.

[68] Y. Yang, Z. Fan, D. Ding, and S. Liu. Application of the preconditioned GMRES to the Crank-Nicolson finite-difference time-domain algorithm for 3D full-wave analysis of planar circuits. *Microwave and Optical Technology Letters*, 50:1458–1463, 2008.

[69] Fernando L. Teixeira. FDTD/FETD methods: a review on some recent advances and selected applications. *Journal of Microwaves and Optoelectronics*, 6(1):83–95, June 2007.

[70] C.W. Manry, S.L. Broschat, and J.B. Schneider. Higher-order FDTD methods for large problems. *Applied Computational Electromagnetics Society Journal*, 10(2):17–29, 1995.

[71] T. Deveze, L. Beaulieu, and W. Tabbara. A fourth order scheme for the FDTD algorithm applied to Maxwell's equations. *IEEE Antennas and Propagation Society International Symposium, 1992*, pages 346 – 349.

[72] G. Dolmans. Applicability of higher-order FDTD schemes with Berenger's ABC for indoor fading channels. *IEEE Antennas and Propagation Society International Symposium, 1997*, 3:1900–1903.

[73] Q. H. Liu. The PSTD algorithm: A time-domain method requiring only two cells per wavelength. *Microwave and Optical Technology Letters*, 15(3):158165, 1977.

[74] M. Krumpholz and L.P.B. Katehi. MRTD: New time-domain schemes based on multiresolution analysis. *IEEE Transactions on Microwave Theory and Techniques*, 44(4):555571, 1996.

[75] R. Holland. Pitfalls of staircase meshing. *IEEE Transactions on Electromagnetic Compatibility*, 35(4):434439, 1993.

[76] R. Wu and T. Itoh. Hybrid finite-difference time-domain modeling of curved surfacesusing tetrahedral edge elements. *IEEE Transactions on Antennas and Propagation*, 45(8):1302–1309, 1997.

[77] Z. Huang, K.R. Demarest, and R.G. Plumb. An FDTD/MoM hybrid technique for modeling complex antennas in the presence of heterogeneous grounds. *IEEE Transactions on Geoscience and Remote Sensing*, 37(6):26922698, 1999.

[78] M. Okoniewski, E. Okoniewska, and M.A. Stuchly. Three-dimensional subgridding algorithm for FDTD. *IEEE Transactions on Antennas and Propagation*, 45(3):422 – 429, 1997.

[79] K. S. Kunz and L. Simpson. A technique for increasing the resolution of finite-difference solution of the Maxwell's equations. *IEEE Transactions on Electromagnetic Compatibility*, 23:419–422, 1981.

[80] D. H. Choi and W. J. R. Hoefer. A graded mesh FDTD algorithm for eigenvalue problems. *17th European Microwave Conference Digest*, pages 413–417, September 1987.

[81] S. Xiao and R. Vahldieck. A fast FDTD analysis of guided wave structures using a continuously variable mesh with second order accuracy. *Journal of Electronics and Telecommunication Engineering*, 41:3–14, 1995.

[82] J. Svigelj and R. Mittra. Grid dispersion error using the nonuniform orthogonal finite-difference time-domain method. *Microwave and Optical Technology Letters*, 10:199–201, 1995.

[83] S.S. Zivanovic, K.S. Yee, and K.K. Mei. A subgridding method for the time-domain finite-difference method to solve Maxwell's equations. *IEEE Transactions on Microwave Theory and Techniques*, 39:471–479, 1991.

[84] J.-P. Bérenger. A Huygens Subgridding for the FDTD method. *IEEE Transactions on Antennas and Propagation*, 54:3797–3804, 2006.

[85] C. Huygens. *Treatise on Light*. `http://www.gutenberg.org/etext/14725`, 1690.

[86] W. Yu and R. Mittra. A conformal finite difference time domain technique for modeling curved dielectric surfaces. *IEEE Microwave and Guided Wave Letters*, 11(1):25–27, 2001.

[87] R. Joseph, S. Hagness, and A. Taflove. Direct time integration of Maxwell's equations in linear dispersive media with absorption for scattering and propagation of femtosecond electromagnetic pulses. *Optics Letters*, 16(18):1412–1414, 1991.

[88] D. M. Sullivan. Frequency-dependent FDTD methods using Z transforms. *IEEE Transaction on Antennas and Propagation*, 40:1223–1230, 1992.

[89] R. J. Luebbers, F. Hunsberger, K. S. Kunz, R. B. Standler, and M. Schneider. A frequency-dependent finite-difference time-domain formulation for dispersive materials. *IEEE Transactions on Electromagnetic Compatibility*, 32-3:220–227, 1990.

[90] S. G. Garcia, R. G. Rubio, A. R. Bretones, and R. G. Martin. Extension of the ADI-FDTD method to Debye media. *IEEE Transactions on Antennas and Propagation*, 51-11:3183–3186, 2003.

[91] F. Costen and A. Thiry. Alternative formulation of three dimensional frequency dependent ADI-FDTD method. *IEICE Electronics Express*, 1:528–533, 2004.

[92] Fumie Costen and Jean-Pierre Berenger. Extension of the FDTD Huygens subgridding to frequency dependent media. *Journal Annals of Telecommunications*, 65:211–217, April 2010.

[93] K.P. Prokopidis, E.P. Kosmidou, and T.D. Tsiboukis. An FDTD algorithm for wave propagation in dispersive media using higher-order schemes. *Journal of Electromagnetic Waves and Applications*, 18(9):1171–1194, 2004.

[94] O. Ramadan. An improved implicit split-step algorithm for dispersive band-limited FDTD applications. *IEEE Microwave and Wireless Components Letters*, 18(8):497 – 499, 2008.

[95] Y. Zhao and Y. Hao. A conformal dispersive FDTD method for modelling of nano-plasmonic waveguides. *IEEE Antennas and Propagation Society International Symposium, 2007*, pages 4461–4464.

[96] Hasan Khaled Rouf, Fumie Costen, and Salvador G. Garcia. 3-D Crank–Nicolson finite difference time domain method for dispersive media. *Electronics Letters*, 45(19):961–962, September 10 2009.

[97] E. C. Jordan and K. G. Balmain. *Electromagnetic Waves and Radiating Systems*. Prentice Hall, second edition, 1968.

[98] Fumie Costen. *High Speed Computational Modelling in the Application of UWB Signals*. PhD thesis, Kyoto University, Japan, 2005.

[99] A.C. Cangellaris and R. Lee. On the accuracy of numerical wave simulations based on finite methods. *Journal of Electromagnetic Waves and Applications*, 6(12):1635–1653, 1992.

[100] A. Thiry. *Efficient FDTD for Broadand Systems*. PhD thesis, The University of Manchester, UK, 2006.

[101] G. Stratis and D. Demetriou. Numerical study of reflection and transmission coefficients for different inhomogeneous walls. *IEEE Antennas and Propagation Society International Symposium*, 1:590–593, August 1999.

[102] Research Computing Services, The University of Manchester, UK. `http://www.rcs.manchester.ac.uk/home`.

[103] M.H.A. Sharkawy, V. Demir, and A.Z. Elsherbeni. An efficient ILU preconditioning for highly sparse matrices constructed using the FDFD method. *IEEE Antennas and Propagation Magazine*, 49(6):135–139, 2007.

[104] David S. Watkins. *Fundamentals of Matrix Computations*, page 521. Wiley-Interscience, Second Edition, 2002.

[105] S. Salvini and G. Shaw. An evaluation of new NAG library solvers for large sparse unsymmetric linear systems. *NAG Technical Report TR2/96, Numerical Algorithms Group Ltd., UK*, 1996.

[106] I. S. Duff and J. K. Reid. *MA48, a Fortran code for direct solution of sparse unsymmetric linear systems of equations*. Rutherford Appleton Laboratory, Oxfordshire, 1993.

[107] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000.

[108] The National Grid Service (NGS), UK. `http://www.ngs.ac.uk/`.

[109] K. Nakajima. Preconditioned iterative linear solvers for unstructured grids on the earth simulator. In *Proceedings of Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004.*, pages 150–159, July 2004.

[110] Y. Saad. *Iterative Methods for Sparse Linear Systems*, page 297. PWS Publishing Company, Boston, MA, 1996.

[111] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual method for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[112] H. van der Vorst. BiCGSTAB: a fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13:631–644, 1992.

[113] Y. Liu, Z. Liang, , and Z. Yang. Computation of electromagnetic dosimetry for human body using parallel FDTD algorithm combined with interpolation technique. *Progress in Electromagnetics Research*, PIER 82:95–107, 2008.

[114] G. Arendash, J. Ramos, T. Mori, M. Mamcarz, X. Lin, M. Runfeldt, L. Want, G. Zhang, V. Sava, J. Tan, and C. Cao. Electromagnetic field treatment protects against and reverses cognitive impairment in alzheimer's disease mice. *Journal of Alzheimer's Disease*, 19(1):191–210, January 2010.

[115] T. Wuren, T. Takai, M. Fujii, and I. Sakagami. Effective 2-Debye-pole FDTD model of electromagnetic interaction between whole human body and UWB radiation. *IEEE Microwave and Wireless Components Letters*, 17(7):483–485, 2007.

[116] Martin Caon. Voxel-based computational models of real human anatomy: a review. *Radiation and Environmental Biophysics*, 42(4):229–235, February, 2004.

[117] T. Nagaoka, S. Watanabe, K. Sakurai, E. Kunieda, S. Watanabe, M. Taki, and Y. Yamanaka. Development of realistic high-resolution whole-body voxel models of Japanese adult males and females of average height and weight, and application of models to radio-frequency electromagnetic-field dosimetry. *Physics in Medicine and Biology*, 49:1–16, 2004.

[118] C.M. Rappaport P. Kosmas. FDTD-based time reversal for microwave breast cancer detection : localization in three dimensions. *IEEE Transactions on Microwave Theory and Techniques*, 54(4):1921–1927, April, 2006.

[119] Hasan Khaled Rouf, Fumie Costen, Salvador G. Garcia, and Seiji Fujino. On the solution of 3-D frequency dependent Crank–Nicolson FDTD scheme. *Journal of Electromagnetic Waves and Applications*, 23:2163–2175, 2009.

[120] X. T. Dong, N. V. Venkatarayalu, B. Guo, W. Y. Yin, and Y. B. Gan. General formulation of unconditionally stable ADI-FDTD method in linear dispersive media. *IEEE Transactions on Microwave Theory and Techniques*, pages 170–174, 2004.

[121] V. Decyk, C. Norton, and H. Gardner. Why fortran?. *Computing in Science and Engineering*, 9:68–71, 2007.

[122] Y. Saad. *SPARSKIT: A basic toolkit for sparse matrix computations (Version 2).* `http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html`, Research Institute for Advanced Computer Science, NASA Ames Research Center.

[123] Rutherford Appleton Laboratory Numerical Analysis Group. *Harwell Subroutine Library (HSL) 2007 for Researchers.* `http://www.hsl.rl.ac.uk/hsl2007/hsl20074researchers.html`.

[124] Rutherford Appleton Laboratory Numerical Analysis Group. *ZD11: Derived type for sparse matrix storage schemes.* `http://hsl.rl.ac.uk/hsl2007/distrib/packages/hsl_zd11/hsl_zd11.pdf`.

[125] Rutherford Appleton Laboratory Numerical Analysis Group. *MI26: Unsymmetric system: BiConjugate Gradient Stabilized (BiCGStab) method.* `http://hsl.rl.ac.uk/hsl2007/distrib/packages/mi26/mi26.pdf`.

[126] Rutherford Appleton Laboratory Numerical Analysis Group. *MC65: Construct and manipulate sparse matrix objects.* `http://hsl.rl.ac.uk/hsl2007/distrib/packages/hsl_mc65/hsl_mc65.pdf`.

[127] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, and J. McDonald. *Parallel Programming in OpenMP.* Morgan Kaufmann, 2001.

[128] *LAPACK: Linear Algebra PACKage.* `http://www.netlib.org/lapack/`.

[129] E. Moro, R. Esselink, J. Xie, M. Hommel, A. Benabid, and P. Pollak. The impact on Parkinson's disease of electrical parameter settings in STN stimulation. *Neurology*, 59:706–713, 2002.

[130] J. Yun, Y. Plamen, and Y. Yalamov. A method for constructing diagonally dominant preconditioners based on Jacobi rotations. *Applied Mathematics and Computation*, 174(1):74–80, March 2006.

[131] Kyushu University Computing Systems for Research, Japan. `http://www.cc.kyushu-u.ac.jp/scp/system/SR16000/intro.html`.