Advanced Modelling of a Borehole Radar Environment with the Finite Difference Time Domain Method

Peter W. Futter

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at the University of Stellenbosch.

Advisor

Prof. David B. Davidson

December 2001

– Declaration –

"I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree."

Signature ....[illegible].......... Peter W. Futter

Date ..26-11-2001..........

Abstract

Over the last decade, as the mining industry of South Africa is moving to ever deeper mines, the borehole radar is becoming an increasingly important field of research.
In December 2000, Burger completed his thesis on Electromagnetic Modelling of a Borehole Radar Environment with the FDTD Method. The goal of this thesis is to extend the research presented in Burger's thesis, considering how more advanced modelling techniques can be applied to the FDTD analysis of the borehole radar environment.
Some of these techniques include implementation of dispersive and conductive material models, and developing Uniaxial Perfectly Matched Layer boundary conditions for matching these model. Simulations were run to measure the performance of these boundary condition for matching dispersive and conductive materials.
The thesis also includes the implementation of a parallel version of the FDTD algorithm using the Message Passing Interface library.
Finally several realistic borehole models where simulated to test the accuracy of the code and to show how the code can be used to model real world problems.

Opsomming

Gedurende die laaste dekade, soos die Suid-Afrikaanse myn industrie al hoe dieper myne gebou het, het die boorgat radar 'n belangrike navorsingsveld geword.
Burger se Meersters graad tesis, Desember 2000, het gehandel oor die electromagnetiese modellering van die boorgat radar omgewing met die Eindige Verskil Tyd Gebied [EVTG] tegniek. Die doel van die huidige tesis is om Burger se werk verder te voer, deur die toepassing van meer gevorderde modellerings tegnieke op die EVTG analise van die boorgat radar omgewing te ondersoek.
Sommige van hierdie tegnieke sluit in: die implimentering van dispersiewe en geleidende materiaal modelle en die ontwikkeling van 'Uniaksiaal, Perfek Aangepaste Laag' rand voorwaardes om hierdie modelle mee aan te pas. Simulasies is uitgevoer om die effektiwiteit van hierdie randvoorwaardes vir die aanpassing van dispersiewe en geleidende materiale te evalueer.
Die tesis sluit ook 'n parallele implimentering van die EVTG algoritme in, wat gebruik maak van die 'Message Passing Interface' funksie biblioteek.
Ten slotte is 'n paar realistiese boorgat modelle gesimuleer om die toepassing van die kode op praktiese probleme te demonstreer en om die akkuraatheid daarvan te evalueer.

Acknowledgements

I would like to thank the following people for their support in my work on this thesis:

- Prof. David B. Davidson for leading me through the project.
- The NRF for granting me a bursary for 2000 and 2001.
- The DSP lab for supplying endless amounts of coffee.
- My house mates, office mates and friends who put up with an over-coffeed over-stressed Masters student for the last two years.
- Marius van Wyk for running simulations in CST.
- My parents who have provided this amazing opportunity.
- My surfboards who help take my mind of work when I needed it most.
- And of course LaTeX for making writing the report a pleasure.

Nomenclature

FDTD	- Finite Difference Time Domain
ABC	- Absorbing Boundary Condition
PML	- Perfectly Matched Layer
UPML	- Uniaxial Perfectly Matched Layer
ADE	- Auxiliary Differential Equation
3D	- Three Dimentional
2D	- Two Dimentional
MPI	- Message PAssing Interface
GUI	- Graphical User Interface
MoM	- Method of Moments

Contents

List of Figures

Chapter 1

Introduction

In December 1999 Burger completed his Masters Thesis, Electromagnetic Modelling of a Borehole Radar Environment using the Finite Difference Time Domain [FDTD] Method. Although he made extensive progress on the FDTD algorithm that he was working on, there was still wide scope for including more advanced modelling techniques in the algorithm. For an introduction to the project and the FDTD Method consult his thesis [1].

This thesis deals with the development and implementation of advanced FDTD methods, which will add greater functionality and modelling capabilities to the FDTD code developed by Burger.

In this chapter, a brief introduction to the *Deepmine* Project is presented, including specific reference to what the value of developing this FDTD algorithm is to the project. Finally, a description of the actual goals for this thesis and an overview of this report is presented.

1.1 The *Deepmine* Project

In South Africa, the mining industries plays a critical part in the countries economy. Due to diminishing geological reserves, mines have been forced to depths below $3.5km$. But even the reserves at these depths will diminish in the future. According to Trickett [2], it is therefore imperative that the mining industry pursue technologies that enable mining at ultra-deep levels. This was the backbone idea that lead to the creation of the *Deepmine* project in March 1998.

Since the creation of the *Deepmine* project, extensive work has been done on developing a borehole radar for the estimation of subterranean features, including faults and water fissures. In March 2001, a group of researchers went to Kleinsee (near the Nambian border) to take measurements using the borehole radar developed for the *Deepmine* project. Figure 1.1 shows some photographs taken of the borehole, and the radar being lowered into the borehole.

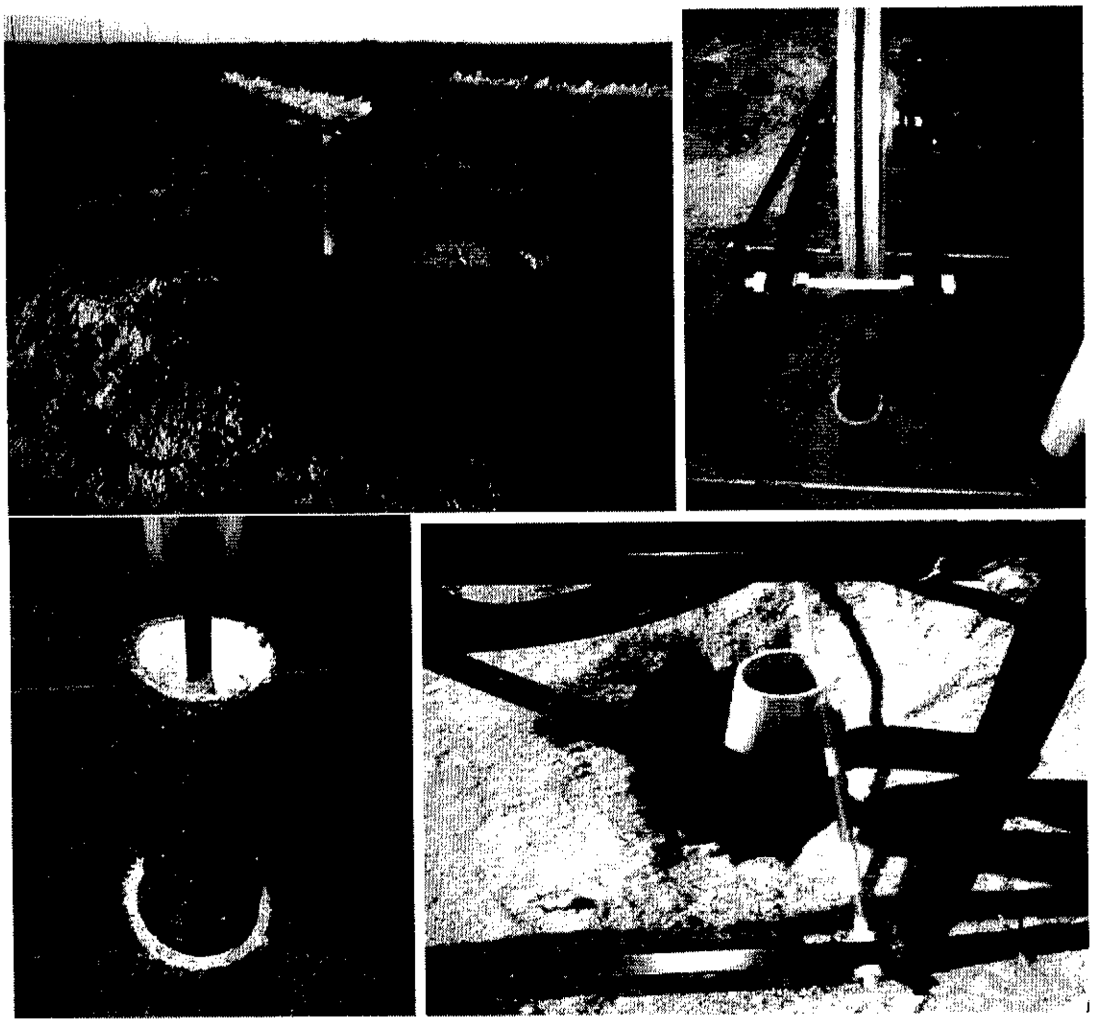

Figure 1.1: Photographs from the Kleinsee Expedition

It is of great value to the *Deepmine* project to be able to simulate such a borehole environment. The FDTD algorithm that was developed for this thesis, was developed with the intention of simulating these borehole measurements. In chapter 5 an example of such a borehole simulation is presented.

1.2 Thesis Goals

The FDTD code written by Burger offered good modelling for such a borehole environment. However, there was still scope for further development on the code, to add greater functionality and accuracy. These are the following aspects that were identified as key areas of development for this thesis:

- Mur second order Absorbing Boundary Condition (ABC) were used in the original FDTD code. However the performance of this boundary condition is greatly exceeded by the Uniaxial Perfectly Matched Layer (UPML). The UPML is theoretically capable of **zero reflection**, even for conductive and dielectric media. The

Mur second order ABCs will be replaced by the UPML.

- A model will be implemented such that the UPML can be used to terminate dispersive media. This model is similar to the Auxiliary Differential Equation (ADE) Method, and will replace the Recursive Convolution Method used in the original code. A Debye model is used to model the dispersive properties of the media.
- A model will also be introduced to the code to include modelling capabilities for dielectric and conductive media. The UPML model will also be adapted to terminate these media.
- A parallel version of the code (using the Message Passing Interface (MPI)) will implemented. This will reduce the time required to run a simulation, and also allow much larger simulation to be solved.
- Due to the increasing size of the code, it proved necessary to restructure the code. The code will be split up into several source files and modules will be used to define variables. This will make the code much more readable, and make further development on the code substantially easier.

The thin-wire antenna model and the impedance loaded antenna models which were developed in the original code will still be used. All the text files that were used for setting up the dimensions in the original code, are also still used.

1.3 Overview of this Report

In chapter 2 a detailed description of the UPML is presented. This chapter includes a derivation, which explains how the UPML can be used to absorb incident waveforms. The derivation of the update equations is also presented.

Chapter 3 discusses the implementation of material models for dispersive and conductive materials. The chapter shows how the UMPL can be extended to match these materials.

Chapter 4 looks at the development of the parallel FDTD algorithm. The chapter introduces the MPI library, and discusses certain functions that were used from the library. It also discusses the finer points of how the FDTD algorithm can be split up between several processes, and what inter-process communication is required.

Chapter 5 shows some results of simulations that were run with the code. The results are compared to results of obtained using other numerical codes.

Finally, in chapter 6 a conclusion is presented. The chapter discusses how effectively the goals of this thesis were achieved, as well as presenting some ideas for future development on the code.

Chapter 2

The PML ABC for Anisotropic Media

One of the greatest challenges with an FDTD implementation is to accurately and efficiently model an unbounded region. In order to maximize efficiency, the simulation grid should only enclose the area closest to the structures of interest. However, it must appear that the fields propagate in an infinite region beyond the grid.
This is best achieved by implementing some form of Absorbing Boundary Condition [ABC], which reduces reflections of the waves leaving the grid.
This chapter will consider several formulations of Perfectly Matched Layer ABC, giving a detailed discussion on the uniaxial formulation. The implementation of the 3D UPML is then discussed. Finally, the performance of a 2D example is presented. A 2D example is used because of the large amount of memory required to obtain a reference signal.

2.1 Introduction to the Perfectly Matched Layer

There are several different types of ABCs, but by far the most successful is the Perfectly Matched Layer [PML], introduced by Berenger [3] . As the name suggests, the PML is a layer of material, a few cells wide, that is modelled to match the inside of the computational grid. The layer must absorb all outgoing fields, minimizing their interaction with the fields inside the grid. There are currently three accepted formulations of the PML:

- Berenger split field formulation
- Stretched co-ordinate formulation
- Uniaxial anisotropic formulation

The split field formulation was the original formulation presented by Berenger [4]. The PML is modelled by splitting the transverse field components into two orthogonal components, allowing an extra degree of freedom. Although this method performs well, the implementation can be confusing, and the material that is modelled in the PML does not actually exist.
The stretched co-ordinate formulation [5] maps Maxwell's equations into a complex co-ordinate space, offering a more compact way of writing the split field equations. Once

again this formulation is largely theoretical.
The concept of modelling the PML ABC as a uniaxial anisotropic material (UPML)for the FDTD method was first introduced by Gedney [6]. The uniaxial formulations is the only formulation that has true physical meaning. Here the PML is modelled as a uniaxial anisotropic medium, with the materials properties to be described in section 2.2.1

2.2 The Uniaxial PML

Because the UPML formulation is the only formulation with true physical significance, it was the formulation that was implemented in the code. The next section briefly discusses some of the properties of uniaxial anisotropic materials, and how these properties can be utilized to implement a UPML ABC.

2.2.1 Uniaxial Anisotropic Materials

In an electrically anisotropic material the D field can no longer be expressed as the product of the scaler ϵ and the vector E field. Each of the D field components are a function of all the E field components. The scaler ϵ is replaced by the tensor $\bar{\bar{\epsilon}}$ with $D = \bar{\bar{\epsilon}}E$ where

$$\bar{\bar{\epsilon}} = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} & \epsilon_{13} \\ \epsilon_{21} & \epsilon_{22} & \epsilon_{23} \\ \epsilon_{31} & \epsilon_{32} & \epsilon_{33} \end{bmatrix} \tag{2.1}$$

A uniaxial anisotropic material is a special case of the anisotropic materials. It describes materials with tensors which only have diagonal elements,

$$\bar{\bar{\epsilon}} = \begin{bmatrix} \epsilon_{11} & 0 & 0 \\ 0 & \epsilon_{22} & 0 \\ 0 & 0 & \epsilon_{33} \end{bmatrix} \tag{2.2}$$

where two of the three diagonal elements are equal.
It will be shown that the uniaxial anisotropic material is ideal for an ABC because it is theoretically capable of transmitting an incident plane wave with ***zero reflection***, regardless of the angle of incidence.

2.2.2 Absorbing Properties of the Uniaxial Medium

In this section, Gedney's derivation [7], on how the UPML is matched to truncate the FDTD grid, is presented because it is essential in understanding of the workings of the UPML.
Consider the scenario shown in figure 2.1 where a polarized time harmonic plane wave is incident on a uniaxial anisotropic half-space. The boundary between the two media is in a constant x plane. Assume that the incident wave has the form $H^{inc} = H_0 e^{-j\beta_x x - j\beta_y y}$.

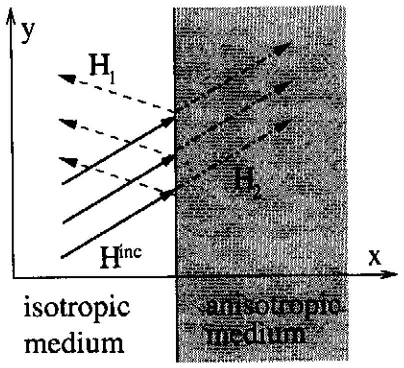

Figure 2.1: The uniaxial PML

Expressions for the reflected (H_1) and transmitted (H_2) waves will be derived. The fields of the plane wave in the uniaxial medium are described by:

$$\begin{aligned} \vec{\beta}^a \times \vec{E} &= \omega \bar{\bar{\mu}} \vec{H} \\ \vec{\beta}^a \times \vec{H} &= -\omega \bar{\bar{\epsilon}} \vec{E} \end{aligned} \tag{2.3}$$

where $\vec{\beta}^a = \hat{x}\beta_x^a + \hat{y}\beta_y^a$, and the permittivity and permeability are the uniaxial tensors

$$\bar{\bar{\epsilon}} = \epsilon_1 \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & b \end{bmatrix}, \ \bar{\bar{\mu}} = \mu_1 \begin{bmatrix} c & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{bmatrix} \tag{2.4}$$

which clearly illustrates the uniaxial medium is rotationally symmetric around the x axis. From equation(2.3) we derive the wave equation as

$$\vec{\beta}^a \times (\bar{\bar{\epsilon}}^{-1}\vec{\beta}^a) \times \vec{H} + \omega^2 \bar{\bar{\mu}} \vec{H} = 0 \tag{2.5}$$

If the curl operations are executed on the vectors, the following matrix expression is obtained from the wave equation:

$$\begin{bmatrix} k^2c - (\beta_y^a)^2 b^{-1} & \beta_y^a \beta_x^a b^{-1} & 0 \\ \beta_x^a \beta_y^a b^{-1} & k^2 d - (\beta_x^a)^2 b^{-1} & 0 \\ 0 & 0 & k^2 d - (\beta_x^a)^2 b^{-1} - (\beta_y^a)^2 b^{-1} \end{bmatrix} \begin{bmatrix} H_x \\ H_y \\ H_z \end{bmatrix} = 0 \tag{2.6}$$

Here $k^2 = \omega^2 \epsilon_1 \mu_1$. If the matrix is solved for β_x^a, there are four eigenmode solutions that can be decoupled into forward and backward TE_z and TM_z modes. For the TE_z mode the dispersion relationship is

$$k^2 - (\beta_x^a)^2 b^{-1} d^{-1} - (\beta_y^a)^2 a^{-1} d^{-1} = 0 \tag{2.7}$$

In order to calculate the reflection coefficient at the boundary of the uniaxial material, the fields in the isotropic medium are expressed as the superposition of the incident and reflected field.

$$\begin{aligned} \vec{H}_1 &= \hat{z} H_0 (1 + \Gamma e^{2j\beta_x^i x}) e^{-j\beta_x^i x - j\beta_y^i y} \\ \vec{E}_1 &= [-\hat{x}\frac{\beta_y^i}{\omega\epsilon_1}(1 + \Gamma e^{2j\beta_x^i x}) + \hat{y}\frac{\beta_x^i}{\omega\epsilon_1}(1 + \Gamma e^{2j\beta_x^i x})] H_0 e^{-j\beta_x^i x - j\beta_y^i y} \end{aligned} \tag{2.8}$$

The wave that is transmitted through the uniaxial medium is described by

$$\begin{aligned}\vec{H}_2 &= \hat{z}H_0\tau e^{-j\beta_x^a x - j\beta_y^a y}\\ \vec{E}_2 &= [-\hat{x}\frac{\beta_y^a}{\omega\epsilon_1 a} + \hat{y}\frac{\beta_x^a}{\omega\epsilon_1 b}]H_0\tau e^{-j\beta_x^a x - j\beta_y^a y}\end{aligned} \tag{2.9}$$

where Γ and τ are the reflection and transmission coefficients respectively. When the boundary conditions are enforced, we have

$$\begin{aligned}\Gamma &= \frac{\beta_x^i - \beta_x^a\, b^{-1}}{\beta_x^i + \beta_x^a\, b^{-1}}\\ \tau &= \frac{2\beta_x^i}{\beta_x^i + \beta_x^a\, b^{-1}}\end{aligned} \tag{2.10}$$

In order for the tangential field components to be continuous over the boundary, the tangential phase velocities must be equal i.e. $\beta_y^a = \beta_y^i$.

Our goal is to find the elements of $\bar{\bar{\epsilon}}$ and $\bar{\bar{\mu}}$ that will give $\Gamma = 0$ for all angles of incidence. Clearly from equation(2.10) the medium must have $\beta_x^a = \beta_x^i b$ for this to be true. If equation(2.7) is rearranged for β_x^a we have

$$\begin{aligned}\beta_x^a &= \sqrt{(bd)k^2 - (a^{-1}b)(\beta_y^a)^2}\\ &= \sqrt{(bd)k^2 - (a^{-1}b)(\beta_y^i)^2}\end{aligned} \tag{2.11}$$

Choosing $d = b$ and $a = b^{-1}$, we have

$$\begin{aligned}\beta_x^a &= \sqrt{b^2(k^2 - (\beta_y^i)^2)}\\ &= b(\beta_x^i)\end{aligned} \tag{2.12}$$

giving the relationship for ***zero reflection*** for all angles of incidence.
A similar procedure can be performed for the TM_z mode dispersion relationship. Zero reflection is obtained in the case where $b = d$ and $c^{-1} = d$.
It is now clear that, if a plane wave is incident on a uniaxial medium boundary in any constant x plane, and has permittivity and permeability tensors

$$\begin{aligned}\bar{\bar{\epsilon}} &= \epsilon_1\bar{\bar{s}}\\ \bar{\bar{\mu}} &= \mu_1\bar{\bar{s}}\\ \bar{\bar{s}} &= \begin{bmatrix} s_x^{-1} & 0 & 0\\ 0 & s_x & 0\\ 0 & 0 & s_x\end{bmatrix}\end{aligned} \tag{2.13}$$

the wave will be purely transmitted into the uniaxial medium with zero reflection, for all angles of incidence, polarization and frequency of incident wave.

If s_x is chosen to be $s_x = (1 + \frac{\sigma_x}{j\omega\epsilon_0})$ then

$$\beta_x^a = (1 - \frac{j\sigma_x}{\omega\epsilon_0})\beta_x^i \tag{2.14}$$

The fields in the uniaxial medium can now be rewritten as

$$\begin{aligned} H_2 &= \hat{z} H_0 e^{-j\beta_x^a x - j\beta_y^a y} e^{-\sigma_x \eta_1 \epsilon_{r1} \cos(\theta_i) x} \\ E_2 &= (-\hat{x} s_x \eta_1 \sin(\theta_i) + \hat{y} \eta_1 \cos(\theta_i)) H_0 e^{-j\beta_x^a x - j\beta_y^a y} e^{-\sigma_x \eta_1 \epsilon_{r1} \cos(\theta_i) x} \end{aligned} \tag{2.15}$$

where θ_i is the angle of incidence measured from the x-axis and η_1 is the wave impedance. As the wave propagates further into the uniaxial medium it is attenuated along the x-axis.

The uniaxial medium performs ideally as a PML because it exhibits the following properties:

- The tangential phase velocities and therefore the wave impedances are equal across the boundary, making the boundary perfectly matched.
- The normal phase velocity (equation 2.14) reduces as as the wave propagates into the PML.
- The uniaxial medium is theoretically capable of totally absorbing an incident plane wave.

Equation (2.13) gives the tensor for the case of a uniaxial medium boundary in a constant x plane. It can be shown that the general tensor, for uniaxial medium boundaries in planes of constant x, y and z can be written as

$$\begin{aligned} \epsilon &= \epsilon_1 \overline{\overline{s}} \\ \mu &= \mu_1 \overline{\overline{s}} \\ \overline{\overline{s}} &= \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_x s_z}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix} \end{aligned} \tag{2.16}$$

2.2.3 Performance of the UPML as an ABC

The previous section deals with the theoretical performance of a uniaxial half-space as an absorbing medium. It was shown that under these ideal conditions the medium is capable of completely absorbing an incident plane wave.
However, two assumptions were made that prevent the UPML from being a perfect absorber when it is implemented in an FDTD code.

The first problem is that a PML ABC must be finite thickness. In the previous section it was assumed that the uniaxial medium was a half-space. When the uniaxial medium is used as a ABC it is only a few cells wide and must be terminated by a boundary. If the boundary is assumed to be PEC then the incident fields are reflected off the boundary and back into the FDTD region. It can be shown that for a PML of thickness d the actual reflection is given by [7]

$$R(\theta_i) = e^{-2\sigma_x \eta \epsilon_r d \cos(\theta_i)} \tag{2.17}$$

and is reduced by using a thicker PML and a higher conductivity. Unfortunately the reflection error increases with larger angles of incidence.
For an efficient FDTD implementation of the UPML the thickness, d should be small. Therefore, to minimize the reflection error the conductance is usually chosen as large as possible.

The second problem is due to the assumption that the FDTD grid is continuous, i.e. reflections occur due to the discretization of the grid. Berenger postulated [8] that the largest discretization reflections occurs at the interface of the UPML.
He proposed that discretization errors can be reduced by spatially scaling the conductance in the UPML. If the conductance is scaled **only along the normal axis**, the material is still matched at the boundary. Several ways of scaling the conductance have been suggested, but the most successful has been the polynomial scaling [9] where

$$\sigma_x(x) = \sigma_{max}(\frac{x}{d})^m \tag{2.18}$$

yielding a reflection error of

$$R(\theta_i) = e^{-2\sigma_{max}\eta\epsilon_r d\frac{\cos(\theta_i)}{m+1}} \tag{2.19}$$

where m is the order of the polynomial (typically in the range of 2 and 4), and d is the thickness of the ABC.
If the conductance is polynomially scaled, then two different reflection errors can be distinguished for the following cases. Given m,d and $R(\theta_i)$

- If the conductance is chosen small (not usually the case) the reflection error is dominated by the reflection off the PEC wall. Equation(2.19) predicts the error accurately.
- If the conductance is chosen large, which is normally the case, the discretization error can dominate the reflection error. Equation(2.19) no longer predicts the error accurately.

Clearly a trade off exists between the two types of reflection errors and an optimal operation point must be found. In [10] a detailed description of the optimal design of the PML is presented. However, a more general solution is desired.
Through extensive experimental studies, demonstrated in [7], the following parameters were found to produce the smallest reflection errors, while still minimizing the discretization errors for a broad range of applications.
It was shown that for a 10 cell think PML ($d = 10$), with $R(0) = e^{-16}$, or $d = 5$ and $R(0) = e^{-8}$ the optimal choice of σ_{max} was given as:

$$\sigma_{opt} = \frac{(m+1)}{150\pi\sqrt{\epsilon_r}\Delta x} \tag{2.20}$$

This expression was found to be robust for a large number of applications.

2.2.4 Implementation - UPML Update Equations

Maxwell's curl equations can now be written as

$$\begin{aligned}\nabla \times \vec{E} &= -j\omega\mu\bar{\bar{s}}\vec{H} \\ \nabla \times \vec{H} &= j\omega\epsilon\bar{\bar{s}}\vec{E}\end{aligned} \tag{2.21}$$

In order to obtain simpler expressions for the update equations, expressions for the D and B fields are also used.
Using the general tensor given in equation(2.16) the D field is given by

$$\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} = \epsilon \begin{bmatrix} \frac{s_z}{s_x}E_x \\ \frac{s_x}{s_y}E_y \\ \frac{s_y}{s_z}E_z \end{bmatrix} \tag{2.22}$$

Substituting the D field into equation(2.21) we get

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = j\omega \begin{bmatrix} s_y & 0 & 0 \\ 0 & s_z & 0 \\ 0 & 0 & s_x \end{bmatrix} \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \tag{2.23}$$

When s_x is chosen such that $s_x = 1+\frac{\sigma_x}{j\omega\epsilon_0}$, and s_y and s_z are chosen in a similar manner, and used in equation(2.23), after the equation is transformed into the time domain ($j\omega \Rightarrow \frac{d}{dt}$) we have

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = \frac{d}{dt} \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} + \frac{1}{\epsilon_0} \begin{bmatrix} \sigma_y & 0 & 0 \\ 0 & \sigma_z & 0 \\ 0 & 0 & \sigma_x \end{bmatrix} \begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \tag{2.24}$$

The update equations for the $\hat{x}$ components of all the fields are derived. The $\hat{y}$ and $\hat{z}$ vector components can be derived in the same way. The D_x component is obtained from (2.24)

$$\frac{dD_x}{dt} + \frac{\sigma_y}{\epsilon_0}D_x = \frac{d}{dy}H_z - \frac{d}{dz}H_y \tag{2.25}$$

The partial derivatives in both the space and time domains are now discretized. Time averaging is applied to the loss term $\frac{\sigma_y}{\epsilon_0}D_x = \frac{\sigma_y}{2\epsilon_0}(D_x|^{n+1}_{i+\frac{1}{2},j,k} + D_x|^{n}_{i+\frac{1}{2},j,k})$ and then the equation is rearranged

$$\begin{aligned} D_x|^{n+1}_{i+\frac{1}{2},j,k} =& \left(\frac{2\epsilon_0 - \sigma_y\Delta t}{2\epsilon_0 + \sigma_y\Delta t}\right)D_x|^{n}_{i+\frac{1}{2},j,k} + \left(\frac{2\epsilon_0\Delta t}{2\epsilon_0 + \sigma_y\Delta t}\right) \\ & \left[\frac{H_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k} - H_z|^{n+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2},k}}{\Delta y} - \frac{H_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k+\frac{1}{2}} - H_y|^{n+\frac{1}{2}}_{i+\frac{1}{2},j,k-\frac{1}{2}}}{\Delta z}\right] \end{aligned} \tag{2.26}$$

The E_x component is derived from the D_x component using the relationship given by equation(2.22).

$$\begin{aligned} s_x D_x &= \epsilon s_z E_x \\ (j\omega + \frac{\sigma_x}{\epsilon_0})D_x &= \epsilon(j\omega + \frac{\sigma_z}{\epsilon_0})E_x \end{aligned} \tag{2.27}$$

The equation is then transformed into the time domain ($j\omega \Rightarrow \frac{d}{dt}$) giving

$$(\frac{d}{dt}(D_x) + \frac{\sigma_x}{\epsilon_0}D_x) = \epsilon(\frac{d}{dt}(E_x) + \frac{\sigma_z}{\epsilon_0}E_x) \tag{2.28}$$

Once again the partial derivatives are discretized, and time averaging is used for the loss term. Both sides of the equation are then multiply through by $2\epsilon_0\Delta t$ and then it is rearranged

$$\begin{aligned} E_x|^{n+1}_{i+\frac{1}{2},j,k} =& \frac{(2\epsilon_0 - \sigma_z\Delta t)}{(2\epsilon_0 + \sigma_z\Delta t)}E_x|^{n}_{i+\frac{1}{2},j,k} + \frac{1}{\epsilon(2\epsilon_0 + \sigma_z\Delta t)} \\ & \left[(2\epsilon_0 + \sigma_x\Delta t)D_x|^{n+1}_{i+\frac{1}{2},j,k} - (2\epsilon_0 - \sigma_x\Delta t)D_x|^{n}_{i+\frac{1}{2},j,k}\right] \end{aligned} \tag{2.29}$$

We now calculate the B and H fields

$$\begin{aligned} B_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} =& (\frac{2\epsilon_0 - \sigma_y\Delta t}{2\epsilon_0 - \sigma_y\Delta t})B_x|^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} + (\frac{2\epsilon_0\Delta t}{2\epsilon_0 + \sigma_y\Delta t}) \\ & \left[\frac{E_z|^{n+1}_{i,j+1,k+\frac{1}{2}} - E_z|^{n+1}_{i,j,k+\frac{1}{2}}}{\Delta y} - \frac{E_y|^{n+1}_{i,j+\frac{1}{2},k+1} - E_y|^{n+1}_{i,j+\frac{1}{2},k}}{\Delta z}\right] \end{aligned} \tag{2.30}$$

and H

$$\begin{aligned} H_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} =& \frac{(2\epsilon_0 - \sigma_z\Delta t)}{(2\epsilon_0 + \sigma_z\Delta t)}H_x|^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} + \frac{1}{\mu(2\epsilon_0 + \sigma_z\Delta t)} \\ & \left[(2\epsilon_0 + \sigma_x\Delta t)B_x|^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} - (2\epsilon_0 - \sigma_x\Delta t)B_x|^{n-\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}}\right] \end{aligned} \tag{2.31}$$

At this point it should be mentioned that when the value of σ is being calculated, careful consideration should be paid to the position of the ifields, remembering the half spatial step between the different vector components.
It should also be mentioned that the UPML update equations presented in this section can be used as a general set of FDTD update equations. It can be shown that for any point outside of the PML (where $\sigma = 0$) the equations simplify to the general update equations. This is of great advantage, because it implies that a single set of update equations can be used for the ABC and the field points inside the computational grid. This means a cleaner and more efficient implementation is possible.

2.3 Performance - A 2D Simulation

In order to test the performance of the UPML, a 2D example was implemented to measure the reflections error (It was not possible to perform this example in 3D because of the large memory requirements for calculating the reference signal). Figure 2.2 shows the setup of the grid for the example. The grid was fed at the centre by the differentiated Gaussian pulse point source

$$E_z(t) = -2[\frac{t - t_0}{t_w}]e^{-[\frac{t-t_0}{t_w}]^2} \tag{2.32}$$

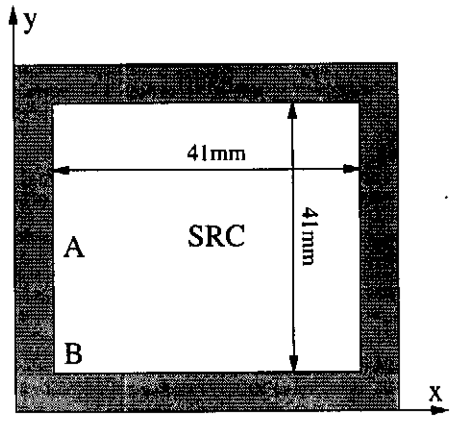

Figure 2.2: Grid for example

with $t_w = 26.53\ ps$ and $t_0 = 4t_w$. A 1x1 mm cell size and a time step of 98% of the Courant limit were used for the simulation. The E field was measured two cells away from the UPML at points A and B. In order to calculate the reflection error at A and B the fields must be compared to a reference signal. This was obtained by extending the grid from a 41x41 cell grid to a 1021x1021 cell grid and measuring the fields at the same position relative to the source as points A an B. The reflection error at time step n was then calculated as

$$R_{error}(n) = \frac{E_z(n) - E_{z,ref}(n)}{E_{z,max}} \tag{2.33}$$

where $E_{z,ref}(n)$ is the reference signal at time step n and $E_{z,max}$ is the maximum value of sequence E_z.
The simulation was run using 5 and 10 layer UPML. The respective predicted reflection errors are (-69dB) and (-139dB) when the optimum conductivity given by equation(2.20) is used. Figure 2.3 and 2.4 show the results obtained for the example.
In both cases the reflection measured at A is smaller than the reflection measured at B. This is due to the summation of reflections off both UPML walls at B, causing the larger reflection error. The 5 layer simulations seems much closer to the predicted value than the 10 layer simulation. This is probably caused by reflections that occur deeper in the PML in the 10 layer simulation. The reflection error measured at A, for the 10 layer simulation, is not nearly as smooth as the other reflection errors. This is probably numerical noise due to the simulation only being run in single precision. It only occurs for this single case, because the values of the reflection error are so small.

2.4 Conclusion

This chapter has discussed how a uniaxial anisotropic material can be modelled as a PML ABC. The formulation of how the a uniaxial half-space can be considered a perfect absorber was presented in detail. Then certain shortcomings of the theoretical model were considered. The formulation of the update equations for the PML vector components was derived, and the implementation thereof was discussed. Finally a 2D simulation was run to measure the performance of the PML as an ABC.

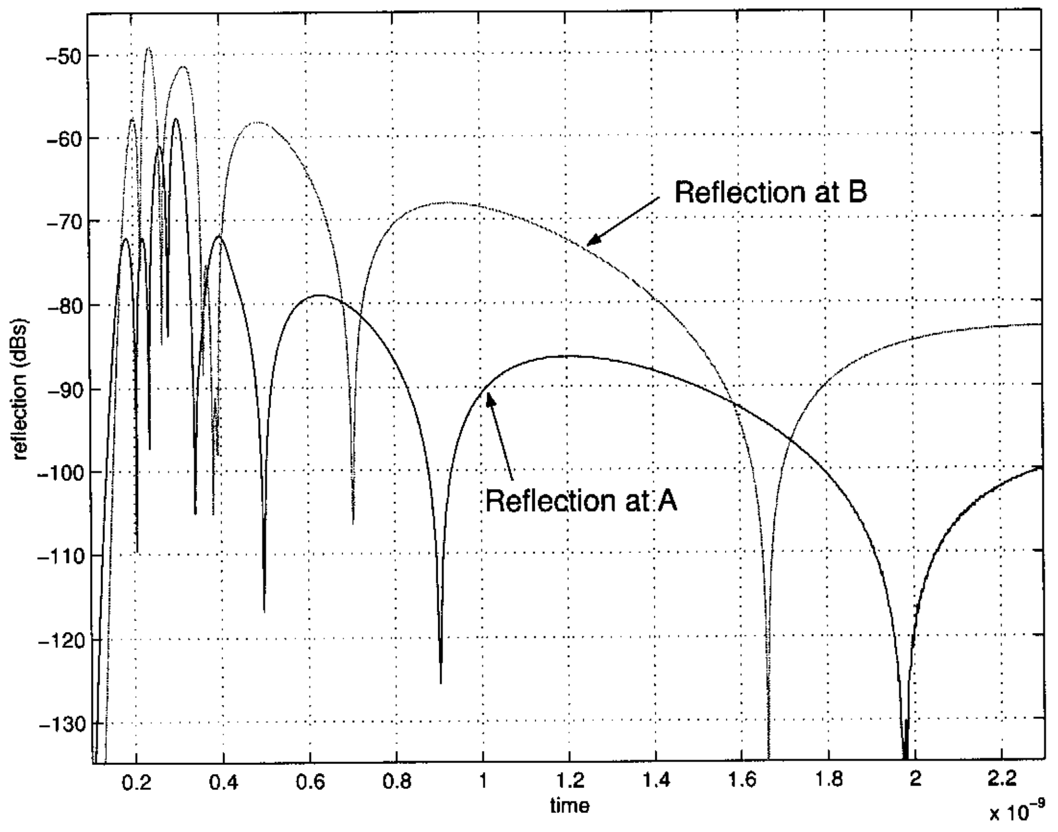

Figure 2.3: 5 layer UPML

It is clear from the results shown in this chapter that the UPML out-performs all previous ABCs by obtaining almost negligible reflection errors.

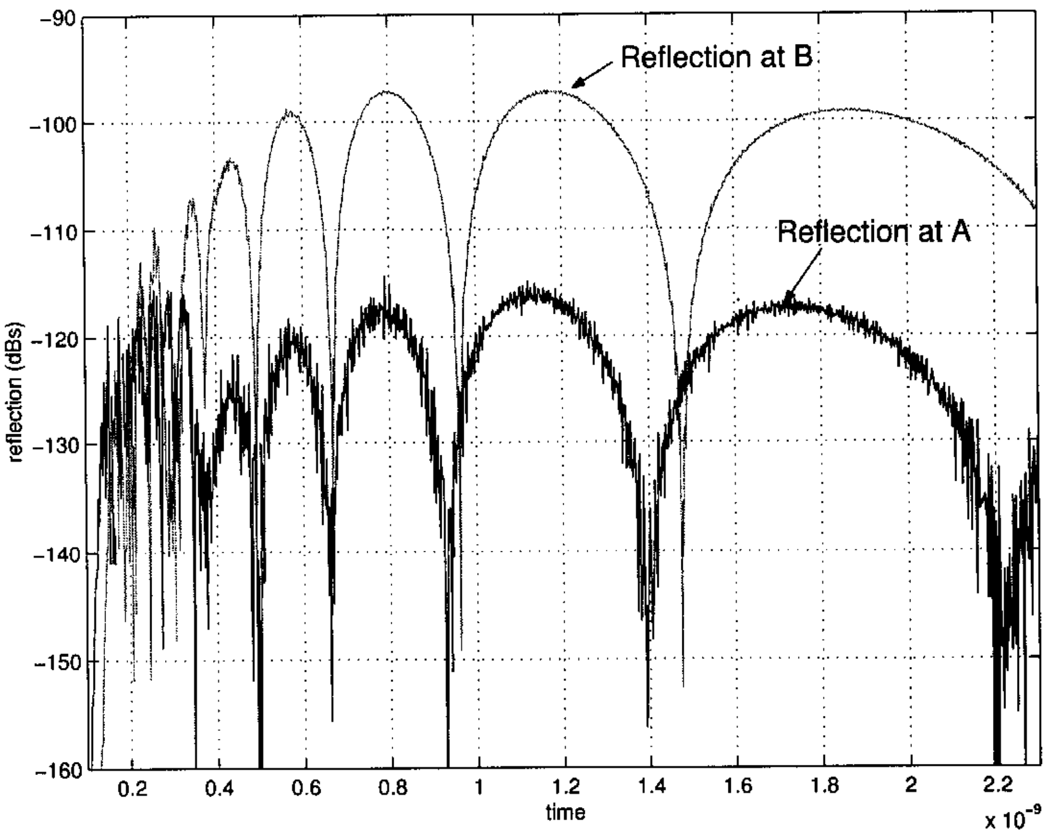

Figure 2.4: 10 layer UPML

Chapter 3

Modelling Dispersive and Conductive Materials

Due to the cell-like structure of the FDTD computational grid, the algorithm is easily adapted to include material models. Once suitable update equations are included in the algorithm, materials are modelled simply by assigning material properties to each cell. Boundary conditions between stratified media are automatically enforced by the update equations.

In this chapter, derivations for both dispersive and conductive PML ABCs are present. It is also shown how the update equations for the PML update equations simplify to the general update equations for both types of materials. The effect that the materials have on modelling the geometry is also discussed.

3.1 Methods for Modelling Dispersive Media

This section involves the derivation of the update equations for dispersive media, using the Debye model to approximate the frequency dependence of the media. The update equations for the PML will be derived first, because outside of the PML, these equations simplify to the general update equations for dispersive media.

3.1.1 The Debye Model

In [1] it was shown that the frequency dependency relative permittivity of water can be accurately modelled using the Debye first order formulation. The Debye approach models the frequency dependency of the relative permittivity as:

$$\epsilon_r(\omega) = (\epsilon_\infty + \frac{\epsilon_s - \epsilon_\infty}{1 + j\omega t_0}) \tag{3.1}$$

where ϵ_s is the DC permittivity, ϵ_∞ is the permittivity at very high frequencies and t_0 is the Debye relaxation constant. These constants for distilled water are given in Table 3.1 Figure 3.1 shows the value of the Debye model for distilled water.

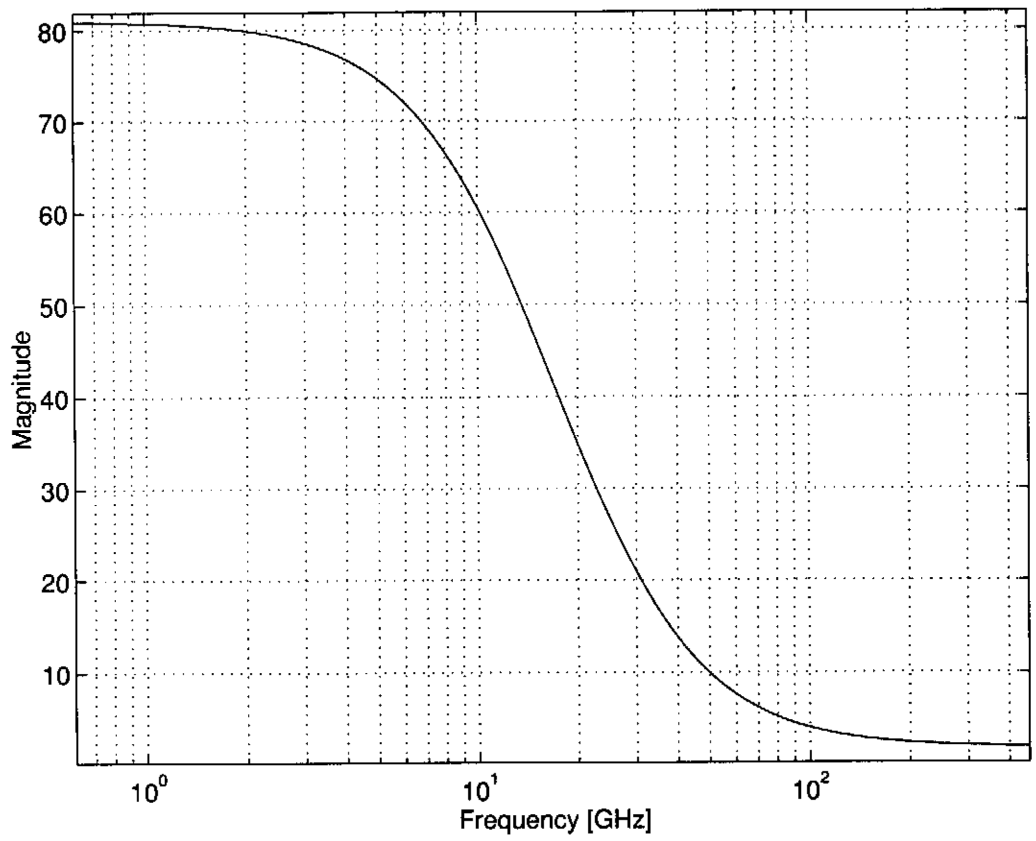

Figure 3.1: Debye modelling of material frequency dependence for water

ϵ_s	81
ϵ_∞	1.8
t_0	9.4 x 10^{-12}

Table 3.1: Debye constants for distilled water

3.1.2 PML Boundary Conditions for Dispersive Media

In [1] a 2nd order Mur ABC was implemented. The Mur formulation is dependant on the speed of the wave traveling in the medium, which it must terminate. When the medium has dispersive properties, the Mur ABC performs poorly, because the different frequency components of the incident wave travel at different speeds. This implies that the ABC cannot be matched to the dispersive media, causing large reflections for certain incident wave forms.

It was shown in section 2.2.3 that the PML is perfectly matched to a medium provided the medium extends normally through the UPML. This is also true for dispersive materials. Therefore the PML was chosen as an ABC, because of its theoretical ability to offer a **zero reflection** coefficient, even for dispersive and stratified media.

The formulation of the PML update equations is very similar, but slightly more com-

plicated than those presented in section 2.2.4. Again the D and B field components are used, but now it is necessary to introduce another variable.
From the general set of PML update equations we have,

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = j\omega\epsilon_0\hat{\epsilon}_r(\omega) \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_x s_z}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix} \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} \tag{3.2}$$

with now the Debye model is included in $\epsilon_r(\omega)$.
Now a variable T is introduced to simplify the update equation by breaking them up into another stage, and the D field component is used, giving:

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \epsilon_0 \begin{bmatrix} \frac{s_z}{s_x}E_x \\ \frac{s_x}{s_y}E_y \\ \frac{s_y}{s_z}E_z \end{bmatrix} \tag{3.3}$$

and

$$[D] = \epsilon_r(\omega)[T] \tag{3.4}$$

Therefore the update equations for the PML with $s = 1 + \frac{\sigma}{j\omega\epsilon_0}$ with dispersive media are given by:

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} + \frac{1}{\epsilon_0}\begin{bmatrix} \sigma_y & 0 & 0 \\ 0 & \sigma_z & 0 \\ 0 & 0 & \sigma_x \end{bmatrix}\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \tag{3.5}$$

and substituting the Debye model for $\epsilon_r(\omega)$ into equation 3.4 and rearranging for T we get:

$$\begin{aligned} (j\omega t_0\epsilon_\infty + \epsilon_s)T_x &= (1 + j\omega t_0)D_x \\ t_0\epsilon_\infty\frac{d}{dt}T_x + \epsilon_s T_x &= t_0\frac{d}{dt}D_x + D_x \end{aligned} \tag{3.6}$$

Finally, the value of the E field can be determined by equation 3.3 and performing a fourier transform into the time domain:

$$\begin{aligned} \epsilon_0(1 + \frac{\sigma_z}{j\omega})E_x &= (1 + \frac{\sigma_x}{j\omega})T_x \\ \epsilon_0\frac{d}{dt}E_x + \epsilon_0\sigma_z E_x &= \frac{d}{dt}T_x + \sigma_x T_x \end{aligned} \tag{3.7}$$

The final update equations are obtained by implementing the discrete derivatives, and using central averaging on the other terms. For example the $\frac{d}{dt}T_x + \sigma_x T_x$ will become

$$\frac{T_x^{n+1} - T_x^n}{\Delta t} + \sigma_x\frac{T_x^{n+1} + T_x^n}{2} \tag{3.8}$$

Once an expression for the E field has been implemented, the standard update equations for the B and H fields are used.

If the equations obtained for the dispersive media PML are compared to those for the free space case, it is clear that they are almost identical. When the dispersive PML equations are implemented, an intermediate step can simply be applied to the free space implementation. This step is the calculation of the variable T in equation 3.6. Equation 3.7 is the same as that of the free space PML equation, with D replaced by T.
In section 5.1 a simulation was run to measure the performance of the PML for terminating dispersive media.

3.1.3 Implementing the General Dispersive Media Update Equations

In the PML the variable $s = 1 + \frac{\sigma}{j\omega\epsilon_0}$, where σ is a function of the depth in the PML. However, if σ is set equal to zero ($s = 1$) for all points outside of the PML, the dispersive media PML update equations simplify as follows:

$$\begin{aligned} T &= \epsilon_0 E \\ D &= \epsilon_r(\omega) T \\ \Rightarrow D &= \epsilon_0 \epsilon_r(\omega) E \end{aligned} \tag{3.9}$$

Here, $\epsilon_r(\omega)$ is still modelled using the Debye model. This set of equations is identical to the equations that are obtained if the ADE method is used to model the dispersive media.
Therefore the update equations for the dispersive media PML simplify to the general update equations for modelling the dispersive media. This is done simply by setting $\sigma = 0$ for all points outside the PML.
The update equations for the PML offer a simple general solution to modelling dispersive media. For points in the PML, the value of σ is matched to the properties of the material normal to the PML. For all points outside the PML, σ is set equal to zero, to provide the general update equations for dispersive media.

3.2 Methods for Modelling Conductive Media

This section presents the derivation of the update equations for modelling conductive material. Once again it will be shown that the PML update equations simplify to the general conductive media update equations for points outside of the PML.

3.2.1 Modelling the Conductivity

The conductivity of the material is modelled in a similar way to the model used for dispersive media. The following expression is used to model the conductivity:

$$D = j\omega\epsilon_0(\epsilon_r + \frac{\sigma}{j\omega\epsilon_0})E \tag{3.10}$$

where ϵ_r is the relative permittivity and σ is the conductance of the material. This model is used in the derivation of the PML boundary conditions presented in the next section.

3.2.2 PML Boundary Conditions for Conductive Media

The update equations for conductive media PML are derived in the same way as those for dispersive media. The expression given for the conductive media in equation 3.10 is frequency dependent. When the expression is transformed into the time domain a variable is introduced to break the update equations up into simpler expression. A full derivation for the PML boundary conditions for conductive media is presented in appendix A.
In [7] a similar formulation is presented to the derivation in appendix A. However, the variables in appendix A are rearranged slightly differently. This allows for the implementation to make use of the general UPML update equations presented in section 2.2.4, with an intermediate expression included. This expression is the calculation of the variable T (equation A.5), which is then used to calculate the E field.

Once again the PML offers **zero reflection** if the PML is matched to the material properties of the material adjacent to the surface of the layer.
In section 5.1 a simulation was run to measure the performance of the PML for terminating conductive media.

3.2.3 Implementing the General Conductive Media Update Equations

As with the dipersive media PML update equations, the conductive media PML update equations offer a general solution to modelling the conductive media. If $s = 1$ outside of the PML the update equations simplify to the general update equations for conductive media.

3.3 Matching the PML to Stratified Media

In this chapter it was shown that in order for the PML to be matched to a material, the material should extend normally into the PML. For the stratified case, this means that the PML is perfectly matched at a point if the material properties of the material that is adjacent to the point extend into the PML.

However one topic concerning the implementation of the stratified media PML has not yet been addressed. That is the assigning of the value of σ_{opt} for the PML. In equation 2.20 it is shown that σ_{opt} is dependent on the relative permittivity of the material. For the stratified media case this proposes an interesting problem, which is not really addressed in related literature. Two possibilities exist to overcoming the problem:

- Use a PML for each stratified layer, where values of σ_{opt} are calculated for each layer.
- Use a single PML, where σ_{opt} is calculated from some kind of average value of ϵ_r for all the layers.

When the first method was implemented, it was found that reflections occur at the boundaries between the different PML layers, giving unstable results.
Therefore the only solution is to calculate some kind of average of ϵ_r for all the layers, which is used to determine σ_{opt}. This is not the ideal solution, because if the different layers have large differences in their relative permittivities, it could lead to large reflections, because the PML will not be perfectly matched to the material.

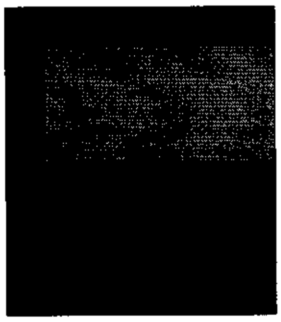

Figure 3.2: Stratified Media example

In order to clarify the implementation a hypothetical example is considered. The geometry is shown in figure 3.2. Here two materials, A and B, are modelled and terminated using a PML. If material A has a relative permittivity of ϵ_A and material B of ϵ_B, the following can be said about the modelling of the points 1-4:

- Point 1 will be modelled using a permittivity ϵ_A, and σ_{opt} will be calculated using ϵ_A.
- Point 2 will be modelled using a permittivity ϵ_A, and σ_{opt} will be calculated using an average of ϵ_A and ϵ_B.
- Point 3 will be modelled using a permittivity ϵ_B, and σ_{opt} will be calculated using an average of ϵ_A and ϵ_B.
- Point 4 will be modelled using a permittivity ϵ_B, and σ_{opt} will be calculated using ϵ_B.

3.4 Modelling the Geometry in Conductive and Dispersive Media

3.4.1 The Thin-wire Update Equations

The thin-wire antenna model, that was developed in [1], was used in this thesis.
When the antenna is surrounded by a material, the free space expressions for the E_x and E_y fields on the wire, are replaced by the the material update equations. The E_z component on the wire is modelled as metal, and therefore is unaffected by the surrounding

material. The magnetic field components are also unaffected by the surrounding material, and therefore the free space update equations are used.

The update equations for the lumped elements remain unchanged when the antenna is surrounded by a material. This is because the fields refer to internal fields in the resistor or capacitor.

3.4.2 Modelling a Borehole and Other Geometries

One of the prime interests of this thesis is to be able to simulate the measurements taken in a borehole. This is discussed in more detail in chapter 5.
The modelling of the borehole requires the ability to model stratified media, where the layers are cylindrical (surrounding the borehole).
It is also of interest to be able to model horizontally stratified media.
Both cases are modelled in the following way.

- Each different material is assigned an integer, *mat_num*.
- The properties of each material are stored in a matrix, where *mat_num* is used to index the matrix. For example $mat\_prop\ (mat\_num, 3) = 4.815$.
- Finally a three dimensional matrix, the same size as the computational grid, stores the number of the material for each point in the grid. When the fields are updated, the matrix is tested for a material at each point. If a material is present the properties are obtained from *mat_prop* and the special update equations for the material are used.

3.4.3 Stability Consideration

It should be noted that for dielectric materials, the wavelength is reduced by a factor of $\frac{1}{\sqrt{\epsilon_r}}$ to that of the free space wavelength. This should be considered when the FDTD cell size is chosen. The rule of thumb is that the cells size should be less than or equal to a tenth of a wavelength.
If the material has dispersive properties, then ϵ_{max} is used when determining the cell size, and matching the PML.

3.5 Results

Several simulations were run to test the accuracy of the two material models presented in this chapter. The results obtained using the code that was developed were compared to results obtained with other commercial packages. The results are presented in Chapter 5.

3.6 Conclusion

In this chapter implementations for dispersive and conductive media models were presented. It was shown that the PML update equations can be used as a general set of update equations for modelling the materials.
The accuracy of the two models was tested in several examples presented in Chapter 5

The implementation of these two models allows the code to model a large range of materials, with varying geometries. The fact that the ABCs can now be matched to the material that is modelled, means that previous reflections off non-perfect ABCs no longer interfere with the accuracy of the material models.

Chapter 4

The Message Passing Interface [MPI]

Today's personal and super computer capabilities have not done away with the need for parallel processing applications. Rather, it has taken the level of performance of parallel processing algorithms to a new level, allowing faster solutions to substantially larger problems. For an introduction to the implementation of a parallel FDTD algorithm on a CM-2 machine consult [11].
The following chapter considers the implementation of a two dimensional parallel architecture, which utilizes the MPI library for inter-processes communication. Functions that were used from the MPI library are described, including the Cartesian mapping and the derived data types that were used. The actual inter-process communication that is required is discussed in section 4.2.4 and 4.2.6.
Several simulations were run to measure the performance of the MPI algorithm. Finally a conclusion is presented, which ties up the work presented in this chapter.

4.1 The MPI Library

Over the last one and a half decades there has been a huge development in the parallel processing community, resulting in codes, which offer highly efficient functions to be used for inter-process communications. An introduction to the developement of parallel processing is presented in [12].
The MPI library, which enables message passing for programs exploiting multiple processors, was completed in 1994, and is fast becoming the standard.
The library contains a group of functions, which can be called in C, C++ and Fortran. The functions are easy to use and are well documented. For these reasons the MPI library was chosen to develop a parallel FDTD algorithm.
The MPI Users Guide [13] provides a detailed description of the MPI functions, while [14] provides an excellent tutorial, including Fortran and C code extracts, for the MPI library.

4.2 Implementing a Parallel FDTD Algorithm

The interest in implementing a parallel FDTD algorithm lies in the following two fundamental ideas. Assuming that inter-process communication time overheads are negligible, the following is theoretically possible for a parallel FDTD algorithm executed on N machines

- The computational time can be reduced by a factor of N.
- The memory requirement can be shared equally between the N machines.

For example, consider two FDTD simulations, where simulation one has a computational grid ten times larger than simulation two. If simulation one is solved using an MPI algorithm on ten machines, it could take the same time to run as simulation two, run on a single machine. It is clear that, by developing a parallel FDTD algorithm, the scope of problems that can be solved becomes substantially larger.

It is shown in [13] that in order for an MPI algorithm to be efficient, the computational loads of each process must be balanced. The FDTD algorithm can be described as a SPMD (Single Program Multiple Data) parallel architecture. These means that each process executes the same program. The differences between processes lie in the position of sources, boundary conditions and material properties. If the FDTD computational grid is divided up equally between the process, the load requirements for each process will be almost the same, and a efficient FDTD algorithm can be developed.
Figure 4.1 shows the flow diagram of the MPI FDTD algorithm that was developed. The remainder of this section describes how each of the steps in the flow diagram are implemented using functions from the MPI library. Several extracts from the Fortran 90 code are included.

4.2.1 Compiling and Running using the MPI Library

In order to use the functions offered in the MPI library it is necessary to include the MPI header file. This is done by inserting the following line at the beginning of the program:

```
#include "mpif.h"
```

The MPI libraries are linked dynamically i.e. they are only linked when the code is executed. The code can be compiled using the standard f90 compiler options, and **'-lmpi'** is added at the end of the command line. The following line will compile a 64 bit executable, **FDTD_mpi**, from the source code **FDTD_mpi.f90**, with level 2 optimization, linking the MPI libraries:

```
f90 -64 -O2 FDTD_mpi.f90 -o FDTD_mpi -lmpi
```

The mpirun command is the primary job launcher for the Message Passing Toolkit implementation of MPI. The parameter **'-np'** is passed before the executable file giving the number of processes to launch. Here N processes will be launched:

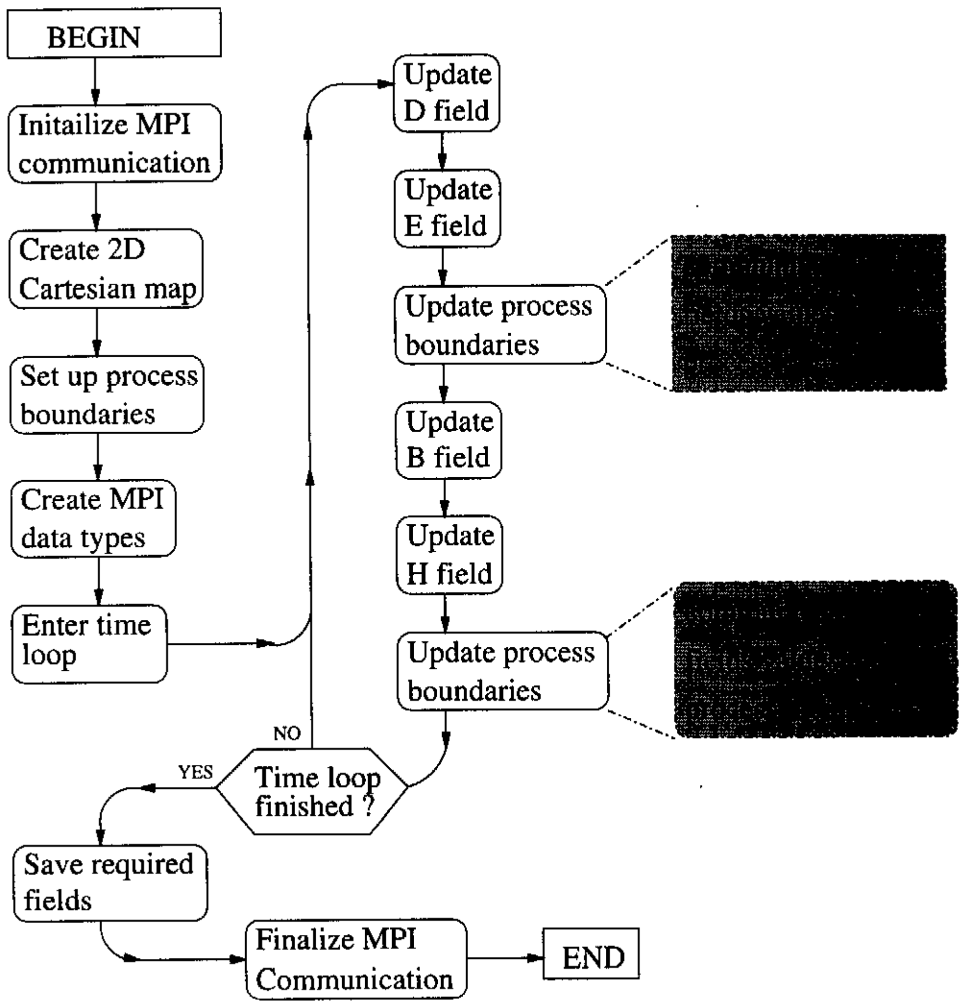

Figure 4.1: Flow diagram for the MPI FDTD algorithm

```
mpirun -np N FDTD_mpi
```

4.2.2 Initializing the Communication World

When mpirun is called, MPI groups all N processes in a single communicator, MPI_comm_world. MPI_comm_world is the default communicator, and is created when the MPI_init is called, enabling all N processes to communicate with one another. It is important to note that, typically each process is run on its own processor, but its not always true. It is possible to run more that one process on a single processor, but it is usually not practical. The following code extract is the typical structure for setting up an MPI program:

```
call MPI_init( err )
call MPI_comm_size( MPI_comm_world, size, err )
      .
      .
call MPI_finalize( err )
```

with the following parameters returned after the functions are called:

```
MPI_comm_world : MPI default communicator handle,
                 created when MPI_init is called
size           : number of processes, N, in
                 MPI_comm_world
err            : error if function call failed
```

4.2.3 Implementing the Two Dimensional Cartesian Topology

The MPI library has a group of functions, which can be used to regulate the way in which the process are mapped to a problem. For an FDTD algorithm, the Cartesian mapping function is ideal for dividing up the computational grid into subspaces and assigning a process to each subspace.

By using a Cartesian map the program is given a more logical structure. It also offers a much simpler and cleaner solution to writing a generic MPI FDTD algorithm i.e. the code can be run with any number of processes without being recompiled.

It was not practical to implement a three dimensional Cartesian map because it would require a large cluster to be run on, which unfortunately was not available. However, the MPI Cartesian functions are easily scaled to three dimensions (for running larger problems). Figure 4.2 shows how the FDTD computational area is mapped for a 3x3 two dimensional Cartesian map in the xy plane. This is implemented by calling the following

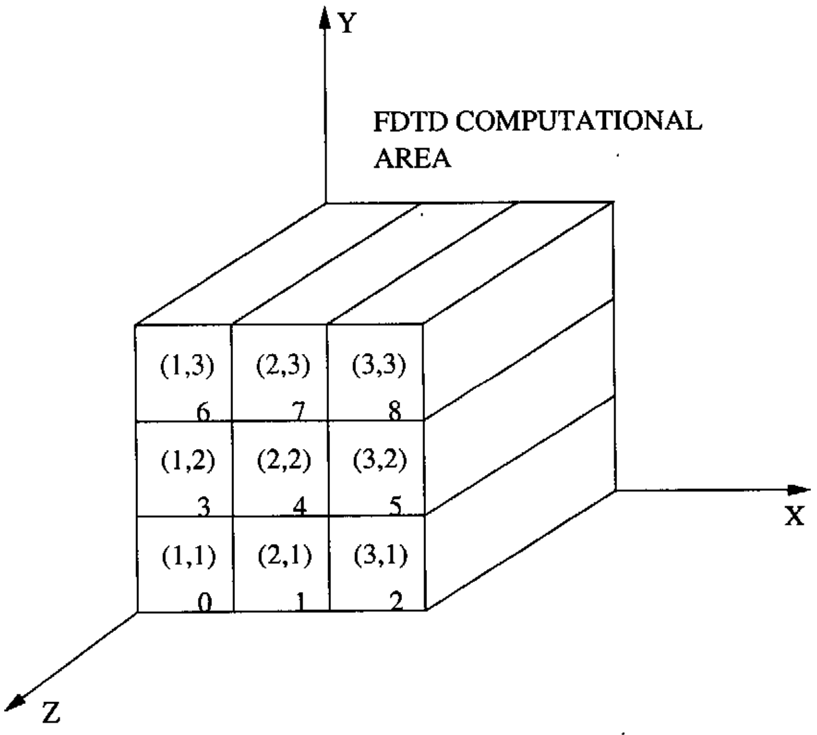

Figure 4.2: 2D Cartesian topology

function:

```
call MPI_cart_create( MPI_comm_world, D, dim(1:3), period,
                      reorder, 2D_comm, err )
```

where

```
MPI_comm_world : default communicator
D              : number of dimensions = 2
dim(1:3)       : number of process in each direction
period         : no periodicity = 0
reorder        : no reordering of processes = 0
2D_comm        : new communicator
err            : error if function call failed
```

When the **2D_comm** is created each process is assigned a rank (from 0 to N−1), and an x and y Cartesian coordinate. This is also illustrated in figure 4.2.

Once the process have been mapped, several MPI functions are available for handling the Cartesian map.
The first function that is used, returns the rank of the neighbouring processes. This information is essential for directing inter-process communication (see section 4.2.6). The following two function calls store the rank of the neighbouring processes in a four element array.

```
call MPI_cart_shift(2D_comm, 0, 1, neighbour_rank(UP),
                    neighbour_rank(DOWN),  err)
call MPI_cart_shift(2D_comm, 1, 1, neighbour_rank(LEFT),
                    neighbour_rank(RIGHT), err)
```

If the process does not have a neighbour on one of its sides i.e. it is on the edge of the Cartesian map, the function returns a rank of -1 for that side. The array for process (3,3) in figure 4.2 would have the following form:

```
neighbour_rank(UP)    = -1
neighbour_rank(DOWN)  =  5
neighbour_rank(LEFT)  =  7
neighbour_rank(RIGHT) = -1
```

This function is also used for determining where PML boundary conditions must be applied i.e. absorbing layers are only added to edges of processes where the array has a value of -1.

The Cartesian mapping also helps simplify the mapping of the geometry across the various number of processes. The following function returns the Cartesian coordinates of the process **rank** in the two element array **coords**:

```
call MPI_cart_coords(2D_comm, rank, 2, coords, err)
```

Once the Cartesian coordinates of the process are obtained, coordinates local to a process can be converted to global coordinates for mapping geometry.

4.2.4 Inter-process Boundaries

In order for the FDTD algorithm to be continuous over the process boundaries, it is necessary to passes certain field values between the processes. This is explained further by considering the dependencies of the field components.
For example, in the plane $y = 1$ in the subspace assigned to process (2,2), shown in figure 4.2
$D_x|_{i,1,k}$ is a function of $(H_y|_{i,1,k}; H_y|_{i,1,k-1}; H_z|_{i,1,k}; H_z|_{i,0,k})$. The value of $H_z|_{i,0,k}$ is $H_z|_{i,ny,k}$ in the process (2,1). This value must be communicated from the process (2,1) to calculate the values of $D_x|_{i,1,k}$ correctly.
Similarly in the plane $y = ny$ in the same subspace $B_x|_{i,ny,j}$ is a function of $(E_y|_{i,ny,k+1}; E_y|_{i,ny,k}; E_z|_{i,ny+1,k}; E_z|_{i,ny,k})$. Here $E_z|_{i,ny+1,k}$ is $E_z|_{i,1,k}$ in process (2,3).

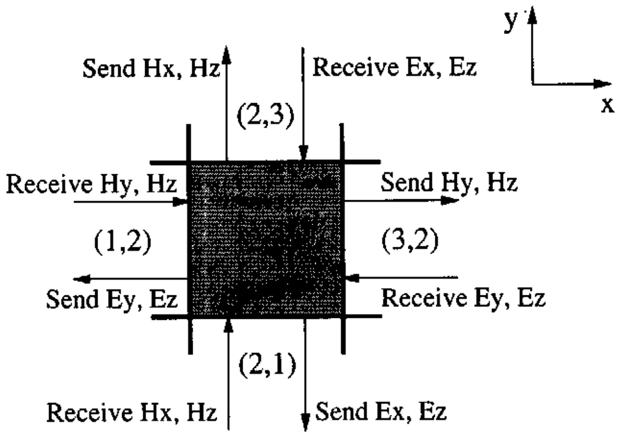

Figure 4.3: Inter-process communication

Figure 4.3 shows the communication required for process (2,2) in the Cartesian topology. In order to simplify the inter-process communication, the size of the field matrices is extended by one in the X and Y directions [15]. These the additional planes are used as buffers to receive outstanding field values from neighbouring processes. The matrices are allocated as follows:

```
E_x( 1:nx+1,1:ny+1,nz )    H_x( 0:nx,0:ny,nz)
E_y( 1:nx+1,1:ny+1,nz )    H_y( 0:nx,0:ny,nz)
E_z( 1:nx+1,1:ny+1,nz )    H_z( 0:nx,0:ny,nz)
```

4.2.5 Defining the Derived Data Types

In Fortran 90 a three dimensional matrix, with indexing (x,y,z), has contiguous data in the X direction, and discontiguous data in both the Y and Z directions. The MPI library offers functions for defining derived data types for both contiguous and discontiguous data.
In order to reduce the time overhead associated with inter-process communication, the data is first grouped before being sent. It can then be sent using a single communication instruction.

Section 4.2.4 showed that fields lying in both the xz and yz planes on process boundaries must be sent or received between processes. Two new data types are needed for grouping these planes before sending or receiving them. The data types are defined in the following way. Firstly a column and a row vector of are defined and committed to the communicator:

```
! define data types call MPI_type_contiguous( nx+1, MPI_real,
colX, err ) call MPI_type_vector( ny+1, 1, nx+1, MPI_real, rowY,
err ) ! commit data types call MPI_type_commit( colX, err) call
MPI_type_commit( rowY, err)
```

Here the contiguous data type is used for the X direction and the discontiguous data type for the Y direction. These vector data types are then used to create two two dimensional matrices.

```
call MPI_type_vector( nz, 1, nx+1, colX, matXZ, err )
call MPI_type_vector( nz, 1, ny+1, rowY, matYZ, err )
call MPI_type_commit( matXZ, err)
call MPI_type_commit( matYZ, err)
```

Because the data is discontiguous in the Z direction, both matrices are created using the discontiguous data type.

4.2.6 Inter-process Communications

The MPI library offers a range of different send and receive functions. Non-blocking send and receive functions were used to perform the required inter-process communication in the parallel algorithm.
The following extract of code shows how inter-process communication in the X direction of E field values is handled.

```
!   X direction communication
    send    = neighbour_rank(LEFT)
    receive = neighbour_rank(RIGHT)
    IF(send.NE.-1)THEN
       CALL MPI_SEND(EYS(1,1,1),1,matYZ,send,tag,2D_comm,stat,err)
       CALL MPI_SEND(EZS(1,1,1),1,matYZ,send,tag,2D_comm,stat,err)
    ENDIF
    IF(receive.NE.-1)THEN
       CALL MPI_RECV(EYS(NX+1,1,1),1,matYZ,receive,tag,2D_comm,stat,err)
       CALL MPI_RECV(EZS(NX+1,1,1),1,matYZ,receive,tag,2D_comm,stat,err)
    ENDIF
```

where

```
EYS(1,1,1)     :  the address of the data group to send
```

```
EZS(1,1,1)     :       "          "
EYS(NX+1,1,1)  :  the address of the data group to receive
EZS(NX+1,1,1)  :       "          "
matYZ          :  the data type to send
receive        :  the rank of process in the positive X direction
send           :  the rank of process in the negative X direction
```

Each process is tested to see if it lies on the edge of the computational grid (send or receive equals -1). If the process does lie on the edge, no fields are sent or received.
The passing of the field values between processes is handled generically, and is independent of the number of process used.

4.3 Performance

A simulation was set up to test the performance of the MPI algorithm. A 60x60x50 computational grid was used, and all simulations were run for 200 time steps.
Firstly, the simulation was run using the standard FDTD algorithm to obtain a benchmark time, to which the MPI execution times can be compared. Then MPI simulations, with different number of processes, were run, and execution times were measured. Simulations were run using both one and two dimensional mapping, so that a comparison could be made.
Unfortunately a cluster was not available to test the MPI algorithm properly. All simulations were executed on a Dual SGI i.e. multiple processes were launched on each processor. Execution times are the actual dedicated CPU times for each process. This means that the results present here do not give a true reflection of the actual performance of the algorithm, because the process times given do not include networking time overheads.
Figure 4.4 shows the execution times for both the one and two dimensional MPI algorithms, for the different number of processes.
The performance of the parallel algorithms can be measured by the following two parameters [15]:

$$\begin{aligned} S &= \frac{T_1}{T_n} \\ E &= \frac{S}{n} \end{aligned} \tag{4.1}$$

where T_1 is the single process execution time and T_n is the execution time for n processes. The scalability gives the ratio between the single process and the n process times, while the efficiency measures the actual performance of the algorithm.
Figure 4.5 shows a plot of the scalability of the two algorithms. It also shows the ideal case, where communication time overheads are zero, which increases linearly with the number of processes. It is clear that both have the one dimensional, and more so the two dimensional algorithm, have non-zero time overheads, especially for a large number of processes.
Figure 4.6 shows a comparison of the two algorithms efficiency. The ideal case would have an efficiency of 100% for any number of processes. The performance of both algorithms

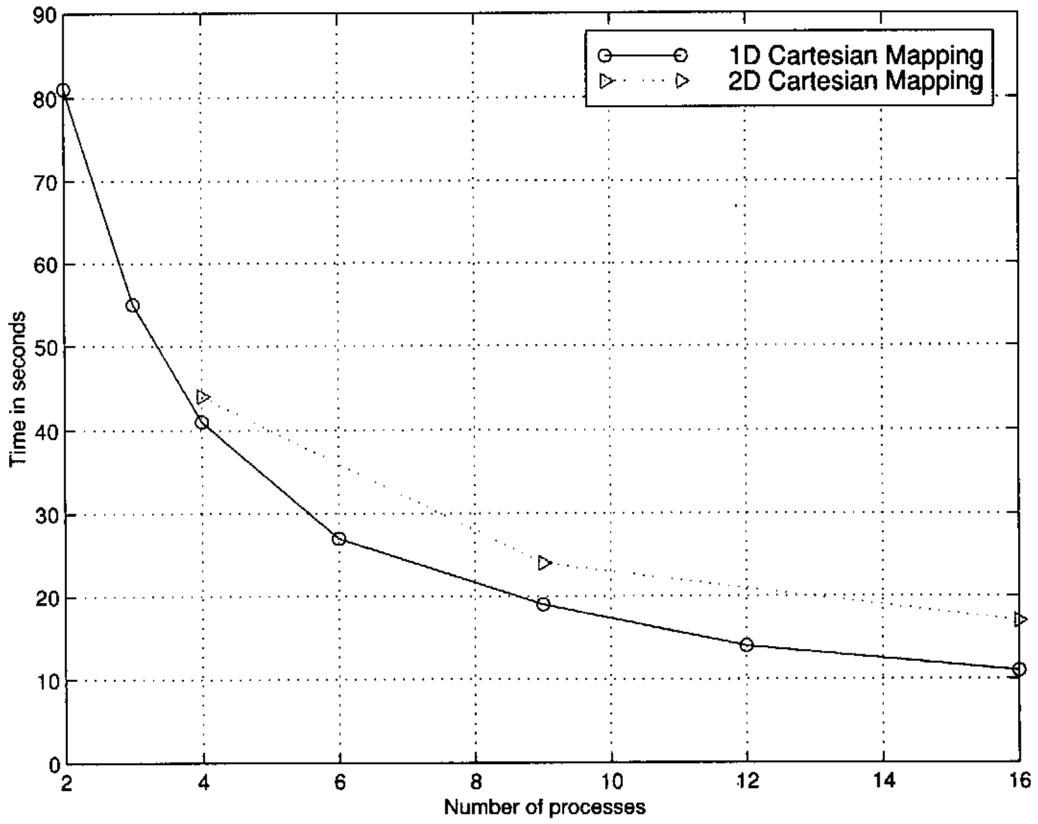

Figure 4.4: Process Times

deteriorates as the number of processes increase and the communication times become larger.

4.4 Conclusion

In this chapter, a detailed description of how a parallel FDTD algorithm can be implemented, was presented. Several MPI functions are also discussed that can be used to implement a generic solution, which allows for mapping of geometry and boundary conditions regardless of the number of processes used.

Several tests were run to measure the performance of the algorithm. Although it is clear that communication time overheads are not negligible for simulations with a large number of processes, this does not make the algorithm impractical to use. The strength of the MPI FDTD algorithm lies in the fact that, if large machines are available, the algorithm can provide a faster solution to huge problems that can never be solved on a single machine, using the standard FDTD algorithm.

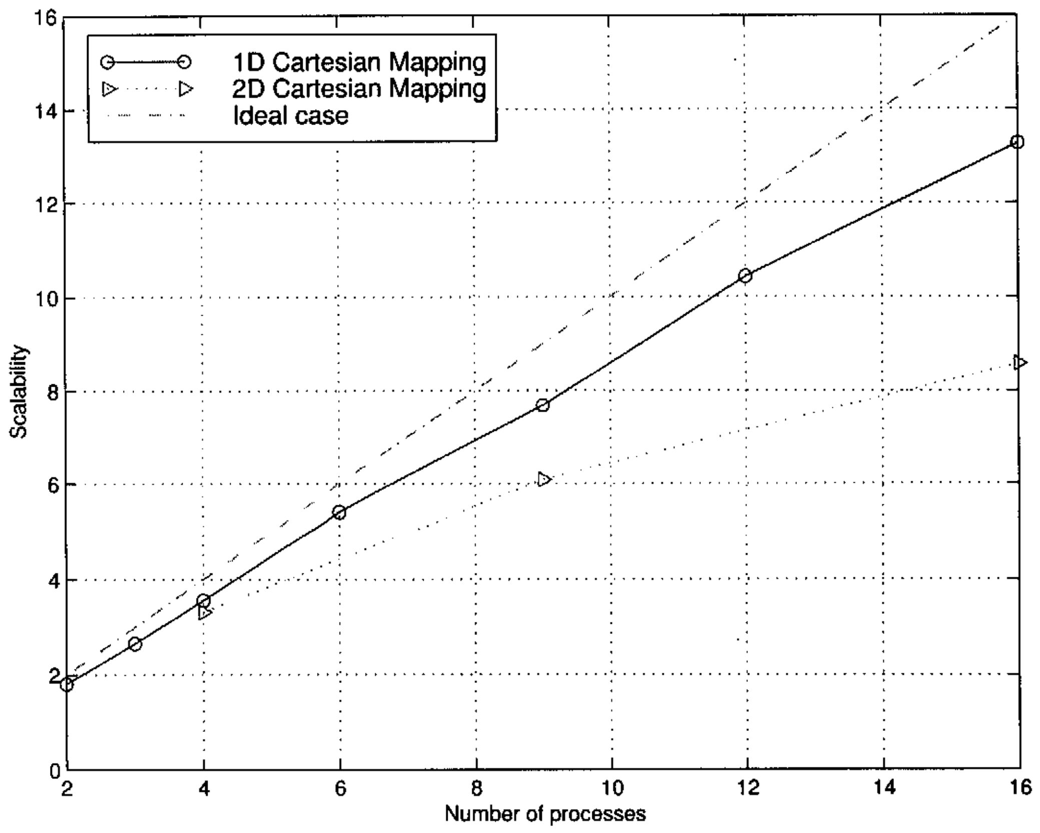

Figure 4.5: Process Scalability

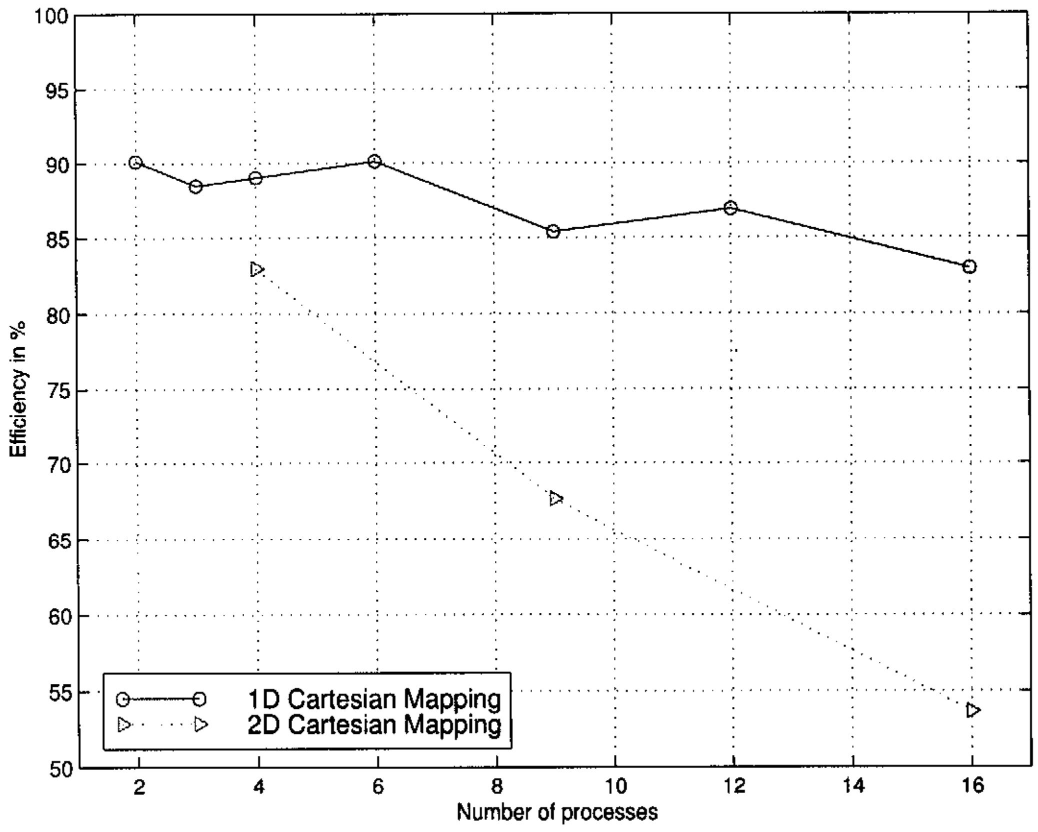

Figure 4.6: Process Efficiency

Chapter 5

Simulations

In order to measure the accuracy of the various FDTD implementations presented in this thesis, several realistic problems were set up and the results were compared to results obtained from other numerical packages.
Firstly, a simulation is set up to measure the performance of PML models developed in chapter 3. Then three GPR simulations are presented, to measure the accuracy of the code and to show how the code can be used to model real world problems.

5.1 PML Performance Measurements

In this section a simulation was set up to measure the performance of the PML models developed in chaper 3. The simulation is set up in the same manner as the simulation in section 2.3, however this example uses the full three dimensional update equations.
The simulation was run using a grid with $NX = 100$, $NY = 60$ and $NZ = 100$, and was excited with a $1GHz$ differential Gaussian pulse. The reflection error for the 10 cell PML was determined by equation 2.17. The reflection was measured two cells from the PML interface in the Y direction. The reference signal was obtained by extending the grid in the Y direction so that $NY = 1000$. The simulation was run to measure the PML performance for homogeneous free space, conductive and dielectric material models.
The conductive media was modelled as dry clay (see section 5.2.1 for properties). Three different simulations were run for the dispersive media, to illustrate how the PML can be mismatched to the material.
Figure 5.1 shows the results obtained for the free space and conductive materials. The maximum reflection error for both materials is about $-113dB$, showing that the PML is an excellent ABC for these two materials.
For the homogeneous dispersive media model, the PML matching is slightly more complex. Careful consideration must be paid to the frequency dependency of the permittivity, relative to the bandwidth of the source. This is illustrated by using three homogeneous dispersive materials, all with different Debye relaxation constants (all three materials have $\epsilon_s = 4.815$ and $\epsilon_\infty = 1.8$). Figure 5.2 shows the three Debye models that where used to model the permittivity frequency dependency for three simulations.
The figure shows that for the three materials, in the frequency band from $100MHz$ to

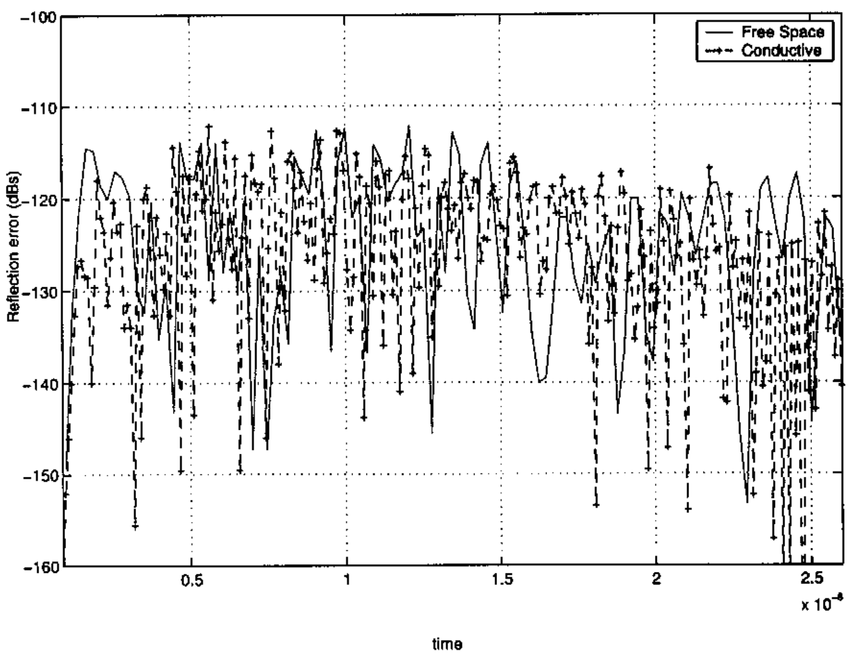

Figure 5.1: PML Performance test

$1GHz$ (where most of the excitation energy lies) the relative permittivity for:

- Material 1 is 1.8 across the whole band.
- Material 2 varies between 3.7 and 1.8.
- Material 3 varies between 4.815 and 3.

For the simulation, the PML is matched to the maximum value of the relative permittivity, $\epsilon_s = 4.815$. Figure 5.3 shows the reflection error for the three materials.
Due to the the PML not being matched to the first material at high frequencies, a larger reflection occurs in the early time response. In the later time response the reflection error is reduce, because the PML is matched at lower frequencies.
Because the PML of the second material is better matched than the first material, the earlier time response now has a smaller reflection error.
For the third material the PML matching is still better, and no increase in the reflection error can be seen in the early time response.

The PML mismatching that occurs in the first simulation can be overcome by matching the PML to a lower value of the relative permittivity. This would increase the later time response reflection error slightly, but it would reduce the reflection error that occurs in the initial time response, giving a reflection error that varies less.
If the frequency dependency of the relative permittivity is considered when the PML is matched, the PML performs excellently as an ABC for terminating dispersive media.

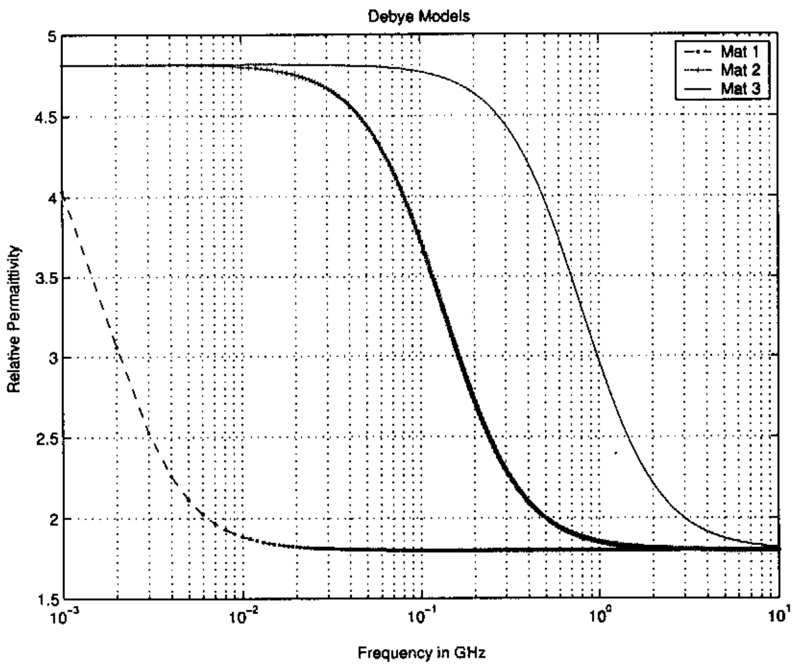

Figure 5.2: Debye Models for PML tests

5.2 GPR Borehole Measurements

In the mining industry it is of great value if the position of gold reefs can be estimated. One way of doing this is by estimating the properties of the rock from the way an electromagnetic wave propagates through it. Such measurements are made by exciting an antenna in a borehole, while measuring the field with another antenna in another borehole. Subterranean properties can be extracted from the fields that are measured.

A simple model is used in the first example to model such a measurement. In the second example a more complex model is used in order to model the actual borehole measurement more realistically. In both examples the antenna is excited with a Gaussian pulse, and the field is measured at some point from the antenna. In the third example, a horizontally layered problem is considered.

5.2.1 Example 1: Simple Model

Figure 5.4 shows a vertical cut through the borehole. In this example, the clay that surrounds the borehole is modelled as dry clay. The dry clay is modelled as a lossy dielectric with the following properties:

$$\begin{aligned} \epsilon_r &= 4.815 \\ \sigma &= 0.00123 \end{aligned} \tag{5.1}$$

The antenna is excited with a Gaussian pulse with a bandwidth of 100 MHz, giving a minimum wavelength of $\lambda_{min} = 1.3672m$ in the clay. In order to model the geometry

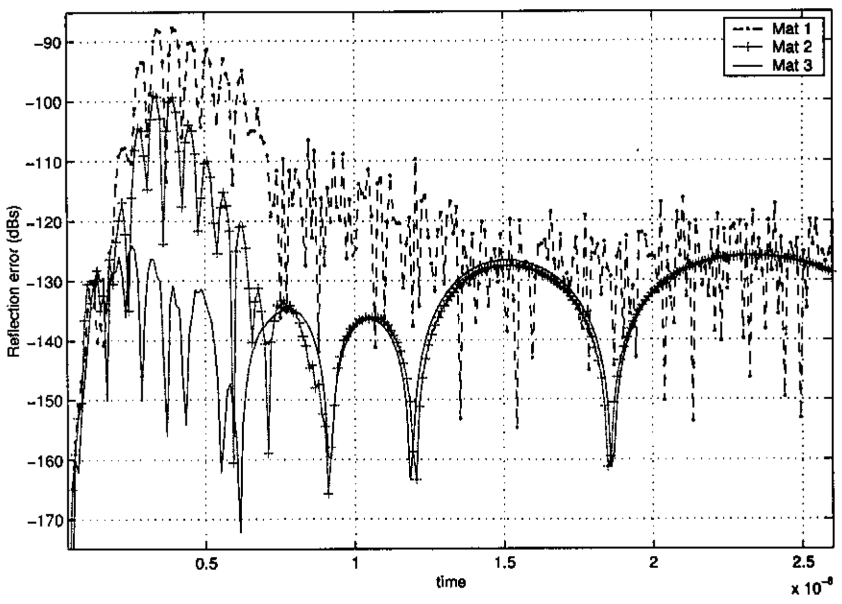

Figure 5.3: PML Performance test

accurately a spatial segmentation of at least 10 samples per wavelength should be used.

Initially, in order to model the borehole as a single cell, a spatial segmentation of 60mm is used in both the X and Y directions. A spatial segmentation of 100mm was used in the Z direction. A computational grid of $(NX = 41, NY = 78, NZ = 41)$, including the 10 cell PML on each wall, was used. The results of the simulation were compared to those obtained using the CST[1] [16] package. However, after the first run, the results were far from agreeing. Figure 5.5 shows a comparison between the probe signals in the frequency domain, after the first run. The results from the FDTD algorithm had a substantially lower centre frequency. After some speculation, the follow areas were identified as causing the difference between the two results.

- Firstly, it was discovered that the finite volume codes excitation wave had a built in time delay. Obviously, if the results are to be compared in the time domain, it is essential that the excitation waveforms are identical (this will only affect the phase in the frequency domain).
- Secondly, the two probe signals had different sampling periods, due to the difference in mesh sizes. In order to compare the signals accurately in the frequency domain it is essential that they have the same sampling period. After the FDTD simulation was run, the probe signal was re-sampled so that the two signals have the same sampling period [2].
- The single cell approximation of the borehole was not good enough. Although using a finer mesh would result in the simulation taking longer to run, it was necessary to model the cylindrical nature of the borehole more accurately.

[1]This is a commercial package, based on a finite volume method

[2]Re-sampling, Discrete Fourier Transforms and all graphs were obtained using MATLAB

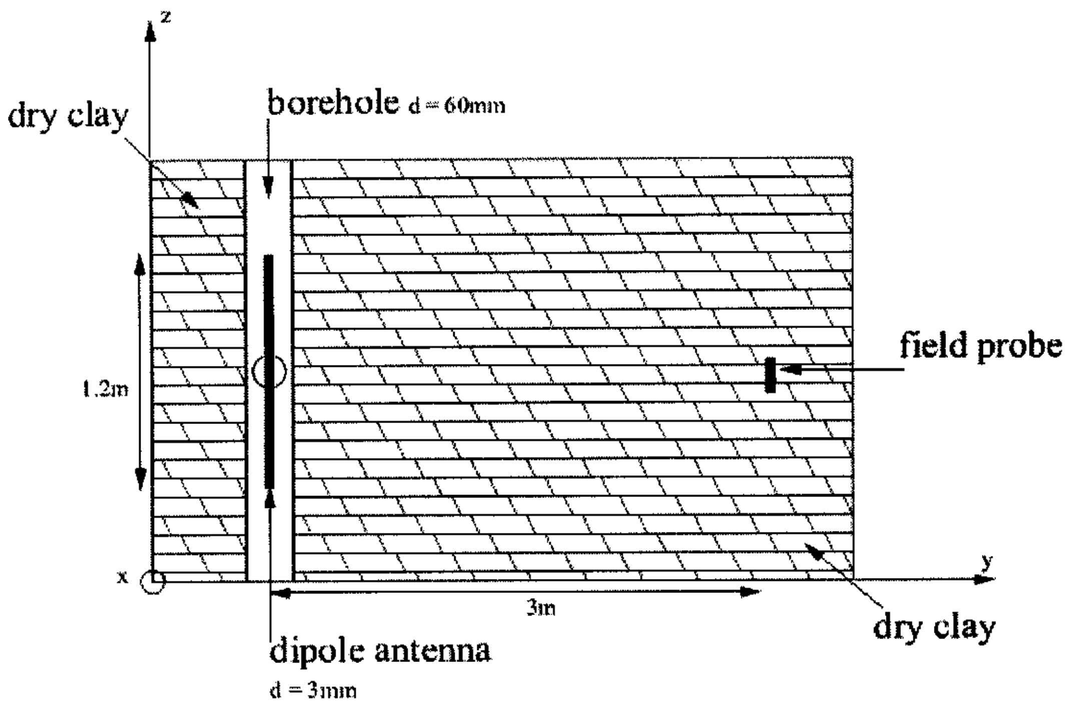

Figure 5.4: Borehole geometry

- Another problem was caused by the borehole not being matched to the PML in the Z direction. Although the PML is perfectly matched in both the horizontal planes, it is not matched in the Z direction (see section 3.3). The PML was matched to the dielectric ($\epsilon_r = 4.815$) i.e. it is not matched over the borehole. The reflection off the PML interferes with the signal measured at the probe.

The simulation was re-run with the following changes made:

- Both simulations were excited with exactly the same time waveform, allowing the results to be compared in the time domain.
- A finer mesh was used in the X and Y direction, enabling the borehole to be modelled as a 'circle' with a 5 cell diameter.
- The computational grid was extended in the Z direction. This reduces the effects that any reflections off the PML will have on the probe signal because the reflections take longer to reach the probe and will be more attenuated, due to the longer path length.

The final parameters for the simulation, including 10 cell PMLs, where:

$$\begin{array}{ll} \Delta x = 0.012m & NX = 51 \\ \Delta y = 0.012m & NY = 285 \\ \Delta z = 0.01m & NZ = 100 \end{array} \tag{5.2}$$

and the time parameters are:

$$\Delta t = 2.8203e^{-11}s \; NT = 8000 \tag{5.3}$$

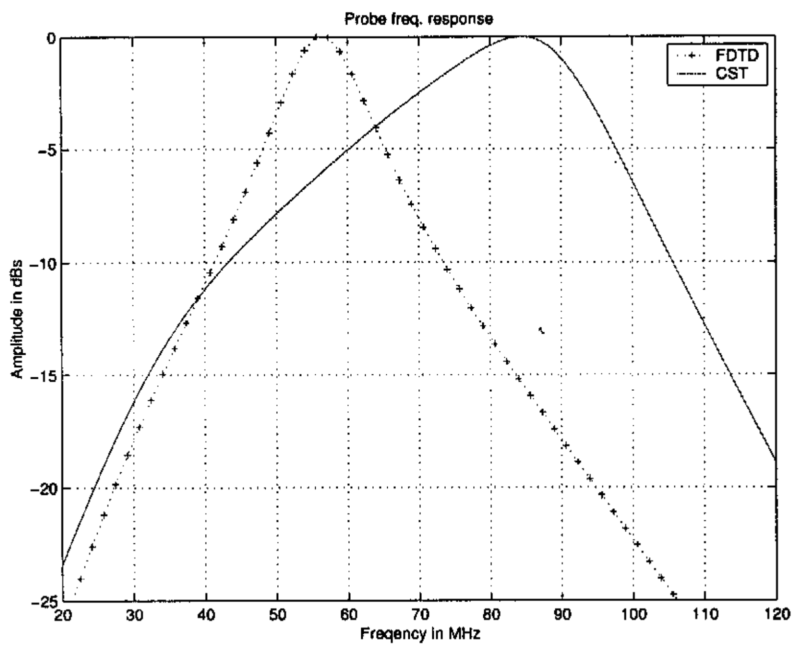

Figure 5.5: Frequency comparison after the first run

Clearly, the second simulation will take much longer to run than the first.
Figure 5.6 shows the two probe signals in the time domain. For the first five peaks the two signals are almost indistinguishable, but then the signals start to differ slightly.

The difference between the two signals is caused by the different modelling capabilities of the two codes. CST has the ability to model circles accurately due to its conformal meshing capabilities, while the FDTD algorithm approximates the circle using a square mesh. The borehole was approximated using a circle with a diameter of five cells. Using only five cells for the circle causes a stairstepped approximation of the borehole. However if a finer mesh was used (to give a better approximation of the round borehole) the simulation would take several days to run.
Figure 5.6 also shows that the two signals have exactly the same centre frequency. This is also illustrated in figure 5.7, where the frequency response of the two signals is shown. Here it is clear that the two signals have the same centre frequency.

This example has illustrated that the conductive media model that was implemented for the FDTD algorithm is accurate. It has also illustrated several other problems with the algorithm.
Firstly, stratified media provide a problem concerning matching the PML. Even though the PML can be perfectly matched to the individual materials, when more than one material is modelled, larger reflections off the PML can be expected. This can be partially overcome by using a larger computational grid.
Secondly, the FDTD algorithm (using rectangular coordinates) models curved structures poorly. This can be improved by using a finer mesh, but this is not always practical due to large RAM requirements. A better option would be to include either non-rectilinear

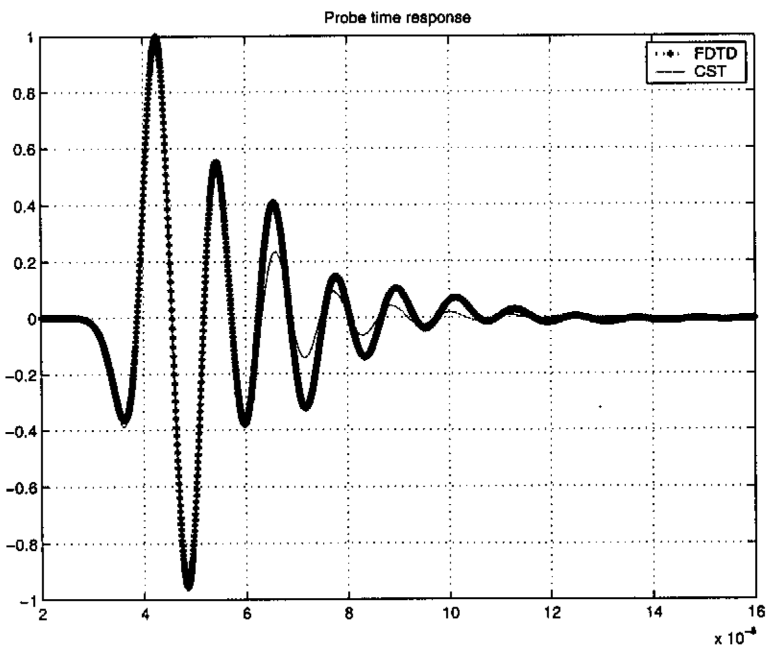

Figure 5.6: Comparison of results

meshing [17], or cylindrical update equations.

5.2.2 Example 2: Cylindrically Layered Model

In this example, the borehole geometry is extended to include a more realistic model of the borehole. The model consists of three different material layers surrounding the borehole, shown in figure 5.8. The first layer is modelled as polyflip (this can be seen clearly in the photographs shown in figure 1.1), the second is modelled as wet clay (the clay becomes wet from the drilling technique), and the final layer is modelled as dry clay. The properties of the materials are given in table 5.1.

Material	ϵ_r	σ **(S/m)**
polyflip	20	0.0176
wet clay	70	0.15
dry clay	4.815	0.00123

Table 5.1: Material properties

The simulation is excited with the same source as the previous simulation. A value of $\epsilon_r = 8$ (see section 3.3) was chosen to match the PML in the Z direction. This value was chosen much closer to the dry clay permittivity than the other materials, because at the PML boundary the dry clay occupies the largest number of cells. However it is still necessary to extend the computational grid in the Z direction to reduce the interference of the reflections off the PML. The PML can be perfectly matched in the X and Y directions

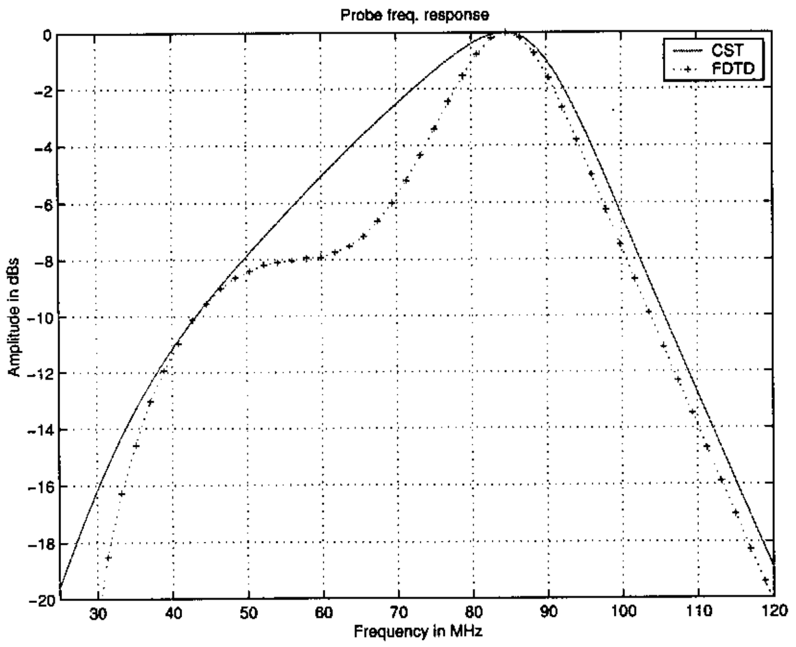

Figure 5.7: Comparison of results

($\epsilon_r = 4.815$). The final parameters for the simulation, including 10 cell PMLs, were:

$$\begin{array}{cc} \Delta x = 0.012m & NX = 60 \\ \Delta y = 0.012m & NY = 295 \\ \Delta z = 0.01m & NZ = 200 \end{array} \tag{5.4}$$

and the time parameters are:

$$\Delta t = 2.8203e^{-11}s \; NT = 8000 \tag{5.5}$$

Figure 5.8 shows a cross section of one quarter of the material matrix that is used to model the borehole. It is clear that the edges of circles are not modelled accurately at all, with the free free space circle being modelled as a square. Table 5.2 gives the dimensions for the problem.

Material	**Dimension**
air	diameter 60mm
polyflip	diameter 200mm
wet clay	diameter 300mm
dry clay	extends to infinity

Table 5.2: Material Layer Dimensions

Figure 5.9 shows a comparison results from the simulation and those obtained using CST. Once again the results differ slightly, but this is due to the poor modelling of the circular nature of the borehole.

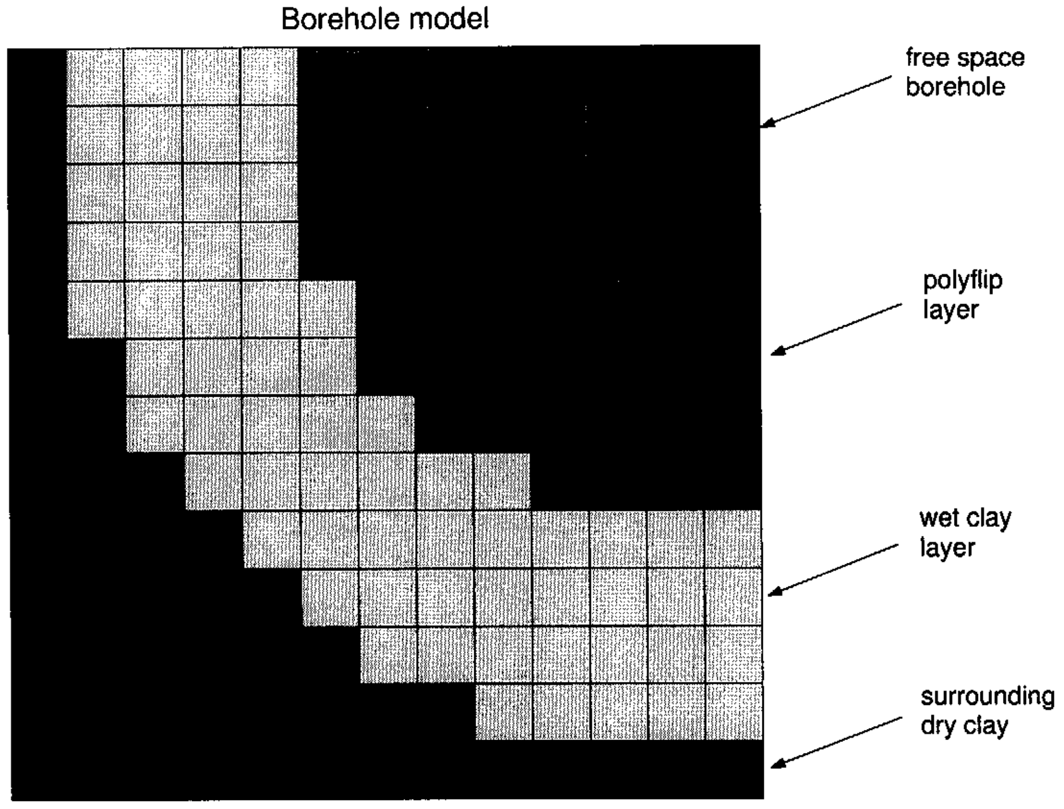

Figure 5.8: Borehole model

5.2.3 Example 3: Layered Dispersive Media Model

In this section a slightly different type of problem is considered. The problem also concerns a stratified media problem, however in this example the layers lie horizontally (this could represent different reefs underground). Once again a dipole is used as a source for the example, but in this example the borehole is not modelled.
Figure 5.10 illustrates the layout of the problem. The dipole is excited with a 1 GHz Differential Gaussian pulse, and the field is probed in two different places. Material one has a dielectric constant of 2.46, while material two has a dielectric constant of 1.3.
A spatial segmentation of $\frac{\lambda}{20}$ at 1 GHz was used, giving $\triangle x = \triangle y = \triangle z = 0.015m$. Once again, the PML matching must be considered carefully. In this example the PML can be matched perfectly to material one ($\epsilon_r = 2.46$) in the Z direction. However this is not true in the X and Y directions. The average value of $\epsilon_{ave} = 2.3$ was used to match the PML in these directions. It was chosen closer to the value of material two so that the PML is better matched to the middle material layer. The computational grid was extended in the X and Y directions in order to minimize the reflections off the PML's interference with the probe signals. The parameters for the simulation are:

$$\begin{array}{ll} \triangle x = 0.015m & NX = 100 \\ \triangle y = 0.015m & NY = 100 \\ \triangle z = 0.015m & NZ = 47 \end{array} \tag{5.6}$$

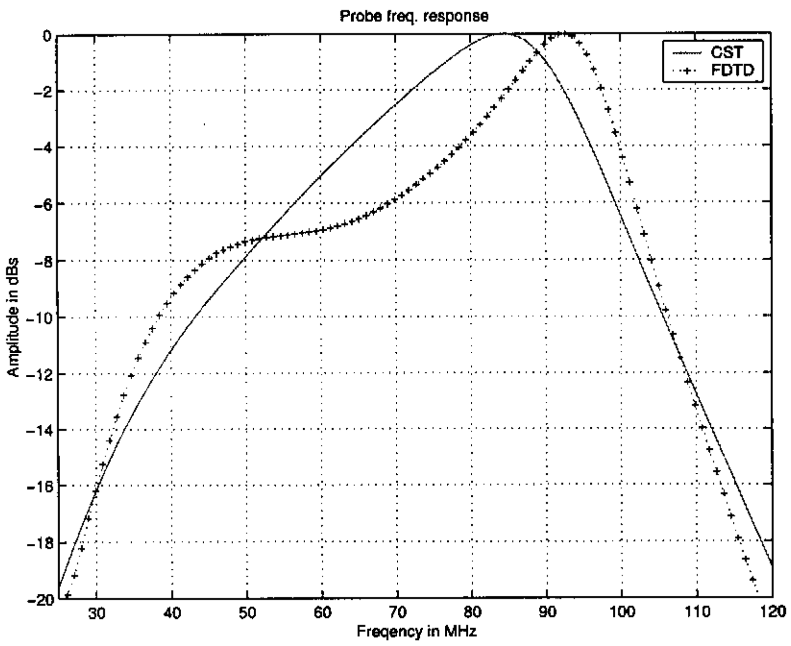

Figure 5.9: Comparison of results

and the time parameters are:

$$\triangle t = 2.889e^{-11}s \; NT = 1500 \tag{5.7}$$

In order to determine the accuracy of the results, the same problem was set up using another commercial package called FEKO [18]. However, unlike CST and the FDTD algorithm, FEKO uses a Method of Moments (MoM) computational engine i.e. it is a frequency domain code and includes the radiation condition.
The FDTD simulation was excited with a Differential Gaussian pulse with a bandwidth of 1 GHz. The FEKO simulation was run from 1.5 to 8.5 GHz, with a frequency step of $\triangle f_{FEKO} = 16$ MHz.
Once again the FDTD time signal needs to be re-sampled such that $\triangle f_{FEKO} = \frac{1}{Nt_{new}}$ [3] i.e. the two simulations will have data at the same frequency points.
It is also necessary to normalise the FDTD probe signals by the source energy because the FEKO simulation excites each frequency point with the same amount of energy.
Figure 5.11 shows a comparison of the two probe signals for the two simulation. The results correlate very well, with the maximum difference less than 2 dBs.

5.3 Conclusion

In this chapter simulations where run to measure the performance of the PML for dispersive and conductive media. The simulations show that the PML is capable of terminating

[3]It is of great importance that $\triangle f_{FEKO}$ is not too large, otherwise it will result in the FDTD time wave form being under-sampled. Check the Nyquist rate [19]

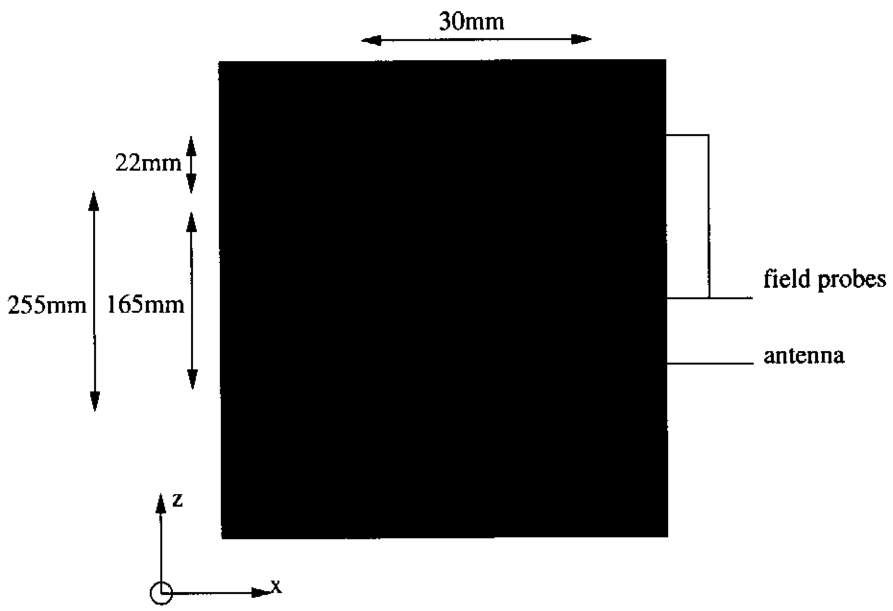

Figure 5.10: Geometry for the Example

these materials with average reflections errors less than -120 dBs. When the PML is matched to dispersive media the frequency dependency of the relative permittivity should be taken into account.

This chapter also deals with three different real world GPR examples, where the results were compared to other commercial packages. This chapter has shown that the implementation of the FDTD algorithm, and the various material models is accurate. Other important issues have also been considered.

The chapter has looked at the various aspects that must be taken into consideration when different codes are compared.
This chapter has also discussed what measures must be taken when the code is used to model stratified media.

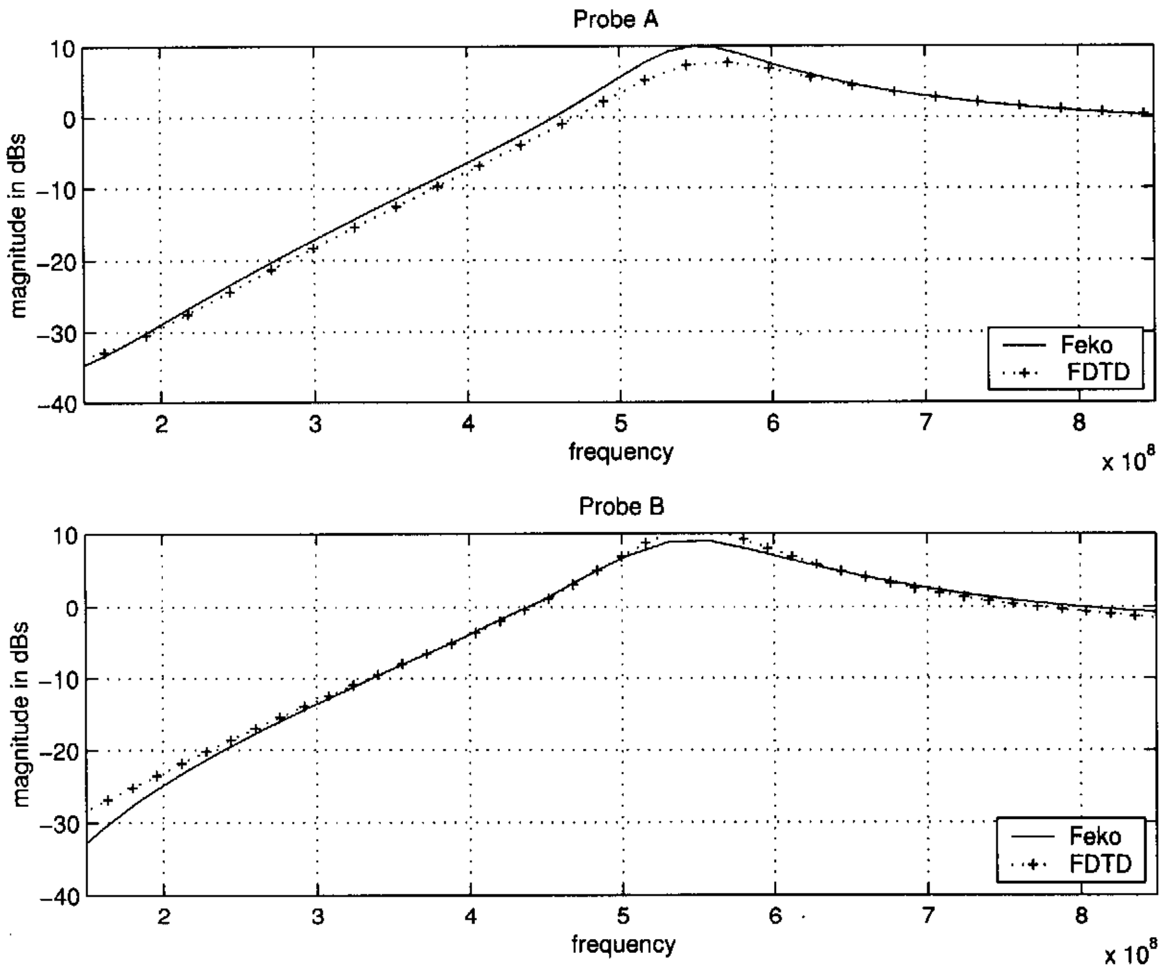

Figure 5.11: Comparison of Probe Signals

Chapter 6

General Conclusion and Recommendations

The goal of this thesis was to add functionality and accuracy to the FDTD code, originally developed by Burger [1]. During the development of the work in this thesis, challenging issues concerning the implementation of the advanced PML ABCs, various material modelling techniques, and the development of a parallel algorithm have been addressed.
This chapter wraps up the all the work that has been presented in this thesis, as well as presenting ideas for future development opportunities.

6.1 Achievements

The main aim of this thesis was to improve the modelling capability of the FDTD code developed in [1], making the algorithm more efficient and more accurate for simulating the borehole environment.
This section discusses the major achievements that were made in the development on the FDTD code:

- The implementation of the UPML, to replace the Mur second order ABCs, was a major contribution to the code. This has led to the an improvement in the algorithm's accuracy, by greatly reducing reflections off computational grid edges. The UPML equations also can be used as a general set of update equations, which now also include the B and D fields.
- The Recursive Convolution Model was replaced by a model similar to the Auxiliary Differential Equation model, and PML ABCs were implemented for dispersive media. Once again the PMLs allow for more accurate modelling.
- A model for dielectric and conductive media was introduced to the algorithm, with PML boundary conditions. This allows a larger range of material types to be modelled, which do not exhibit dispersive properties, for example the clay surrounding the borehole in chapter 5.

- The development of the parallel FDTD algorithm has led to the algorithm becoming far more efficient, allowing much larger problems to be split up over several computers and produce results much faster.
- Finally, work was done on the code, to make it more modular. The code was broken up in to several different files containing different sets of subroutines. This will allow for much easier understanding of the code and future development on the code.

6.2 Possible Future Development

Although much functionality was added to the FDTD algorithm that has been developed, there are still aspect of the algorithm that can be enhanced to improve the performance and make the code more user friendly. Here are some ideas which would be worth investigating.

- In order to model the cylindrical nature of the borehole more accurately, it could be of great value to implement the update equations in cylindrical coordinates. This would be a large task, because all boundary conditions and material models would have to be derived again. However, it would lead to more accurate results.
- In the event of a large cluster being available for running simulations on, it could be well worth scaling the two dimensional MPI FDTD algorithm to a full three dimensional algorithm. It would not require too much extra work, as the MPI algorithm was written considering the possibility of later being scaled. This would add great efficiency and allow far larger simulations to be solved.
- It would be of great value if a near to far field transformation was included in the code. This would enable the code to calculate far field patterns for antennas etc.
- It may be worth developing a more advanced preprocessor, which could import geometries that were drawn in some kind of drawing package. This would greatly increase the complexity of the problems that could be solved.
- Finally, if some kind of Graphic User Interface (GUI) was added to the code, it would make the code far easier to use, and could be the final step in developing a commercial package.

Appendix A

UPML Conductive Boundary Conditions

From $\nabla \times \vec{H} = j\omega\epsilon\bar{\bar{s}}\vec{E}$

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = j\omega\epsilon_0(\epsilon_r + \frac{\sigma}{j\omega\epsilon_0}) \begin{bmatrix} \frac{s_y s_z}{s_x} & 0 & 0 \\ 0 & \frac{s_x s_z}{s_y} & 0 \\ 0 & 0 & \frac{s_x s_y}{s_z} \end{bmatrix} \begin{bmatrix} E_x \\ E_y \\ E_z \end{bmatrix} \tag{A.1}$$

Now introduce the D field, and a variable T, where :

$$\begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \epsilon_0 \begin{bmatrix} \frac{s_z}{s_x}E_x \\ \frac{s_x}{s_y}E_y \\ \frac{s_y}{s_z}E_z \end{bmatrix} \tag{A.2}$$

and

$$[D] = (\epsilon_r + \frac{\sigma}{j\omega\epsilon_0})[T] \tag{A.3}$$

The update equations for the D field in the PML, with $s = 1 + \frac{\sigma}{j\omega\epsilon_0}$ is given by:

$$\begin{bmatrix} \frac{d}{dy}H_z - \frac{d}{dz}H_y \\ \frac{d}{dz}H_x - \frac{d}{dx}H_z \\ \frac{d}{dx}H_y - \frac{d}{dy}H_x \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} + \frac{1}{\epsilon_0}\begin{bmatrix} \sigma_y & 0 & 0 \\ 0 & \sigma_z & 0 \\ 0 & 0 & \sigma_x \end{bmatrix}\begin{bmatrix} D_x \\ D_y \\ D_z \end{bmatrix} \tag{A.4}$$

Equation A.3 is rearranged and then transformed into the time domain.

$$\begin{aligned} (j\omega\epsilon_0\epsilon_r + \sigma)T &= (j\omega\epsilon_0)D \\ \epsilon_r\frac{d}{dt}T + \frac{\sigma}{\epsilon_0}T &= \frac{d}{dt}D \\ \epsilon_r\frac{T^{n+1} - T^n}{\Delta t} + \frac{\sigma(T^{n+1} + T^n)}{2\epsilon_0} &= \frac{D^{n+1} - D^n}{\Delta t} \\ 2\epsilon_0\epsilon_r(T^{n+1} - T^n) + \sigma\Delta t(T^{n+1} + T^n) &= 2\epsilon_0(D^{n+1} - D^n) \\ (\sigma\Delta t + 2\epsilon_0\epsilon_r)T^{n+1} + (\sigma\Delta t - 2\epsilon_0\epsilon_r)T^n &= 2\epsilon_0(D^{n+1} - D^n) \\ T^{n+1} &= \frac{2\epsilon_0}{2\epsilon_0\epsilon_r + \sigma\Delta t}[T^n(\frac{\epsilon_r - \sigma\Delta t}{2\epsilon_0}) + D^{n+1} - D^n] \end{aligned} \tag{A.5}$$

Finally, the value of the E field can be determined from equation A.2

$$\begin{aligned}\epsilon_0(1+\frac{\sigma_z}{j\omega})E_x &= (1+\frac{\sigma_x}{j\omega})T_x \\ \epsilon_0\frac{d}{dt}E_x + \epsilon_0\sigma_z E_x &= \frac{d}{dt}T_x + \sigma_x T_x\end{aligned} \tag{A.6}$$

The final update equations are obtain by implementing the discrete derivatives, and using central averaging on the other terms. Once the expression of the E field has been implemented, the standard update equations for the B and H fields are used.

Bibliography

[1] E. H. Burger, "Electromagnetic Modelling of a Borehole Radar Environment with the FDTD Method," Master's thesis, University of Stellenbosch, Stellenbosch, 7600, South Africa, December 2000.

[2] J. C. Trickett, I. Mason, H. Eybers, M. Meyering, F. Stevenson, D. Vogt, J. Hargreaves, and R. Fynn, "The application of borehole radar to South Africa's ultra-deep gold mining environment," in *8th Ground Penetrating Radar Conference 2000, Australia*, pp. 676–681, May 2000.

[3] J.-P. Bérenger, "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *Journal of Computational Physics*, vol. 114, no. 1, pp. 185–200, 1994.

[4] A. Taflove, "The Perfectly Matched Layer Absorbing Medium," in *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, ch. 7, pp. 181–194, Boston, MA: Artech House, 1995.

[5] W. C. Chew and W. H. Weedon, "A 3D Perfectly Matched Medium from Modified Maxwell's Equations with Stretched Coordinates," *Microwave and Optical Technologies Letters*, vol. 7, pp. 599–604, Sept. 1994.

[6] S. D. Gedney, "An Anisotropic Perfectly Matched Layer-Absorbing Medium for the Truncation of FDTD Lattices," *IEEE Transactions on Antennas and Propagation*, vol. 44, pp. 1630–1639, Dec. 1996.

[7] S. D. Gedney, "The Perfectly Matched Layer Absorbing Medium," in *Advances in Computational Electrodynamics: The Finite-Difference Time-Domain Method* (A. Taflove, ed.), ch. 5, pp. 263–344, Boston, MA: Artech House, 1998.

[8] J.-P. Bérenger, "Improved PML for the FDTD Solution of Wave-Structure Interaction Problems," *IEEE Transactions on Antennas and Propagation*, vol. 45, pp. 466–473, Mar. 1997.

[9] J.-P. Bérenger, "Perfectly Matched Layer for the FDTD Solution of Wave-Structure Interaction Problems," *IEEE Transactions on Antennas and Propagation*, vol. 44, pp. 110–117, Jan. 1996.

[10] D. C. Wittwer and R. W. Ziolkowski, "How to Design the Imperfect Bérenger PML," *Electromagnetics*, vol. 16, no. 4, pp. 465–485, 1996.

[11] D. B. Davidson and R. W. Ziolkowski, "A Connection Machine (CM-2) Implementation of a 3D Parallel FDTD Code for Electromagnetic Field Simulation," *International Journal of Numerical Modelling : Electronic Netwroks, Devices and Fields*, vol. 8, pp. 221–232, 1995.

[12] D. B. Davidson, *Parallel Algorithms for Electromagnetic Moment Method Formulations.* PhD thesis, University of Stellenbosch (South Africa), November 1991.

[13] P. S. Pacheco, "A User's Guide to MPI," Tech. Rep. CA 94117, University of San Francisco, San Francisco, March 1998.

[14] "Message Passing Interface (MPI)."
`http://www.mhpcc.edu/training/workshop/mpi/MAIN.html`.

[15] C. Guiffaut and K. Mahdjoubi, "A Parallel FDTD Algorithm using the MPI Library," *IEEE Antennas and Propagation Magazine*, vol. 43, pp. 94–103, April 2001.

[16] "CST Home Page."
`http://www.CST.de`.

[17] D. Davidson and R. W. Ziolkowski, "Body-of-revolution finite-difference time-domain modelling of space-time focusing by a three-dimensional lens," *J. Opt. Soc. Am. A*, vol. 11, pp. 1471–1490, April 1994. *Special Issue on 3D Scattering.*

[18] "FEKO Home Page."
`http://www.FEKO.co.za`.

[19] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms and Applications.* Prentice Hall, third ed., 1996.