STREAMLINE UPWIND/ PETROV-GALERKIN FEM BASED TIME-ACCURATE
SOLUTION OF 3D TIME-DOMAIN MAXWELL’S EQUATIONS FOR DISPERSIVE
MATERIALS

By
Srijith Rajamohan

William Kyle Anderson
Professor of Computational Engineering
(Chairperson)

Kidambi Sreenivas
Research Professor of Computational Engineering
(Committee Member)

Li Wang
Research Assistant Professor of Computational Engineering
(Committee Member)

John V. Matthews, III
Associate Professor of Mathematics
(Committee Member)
STREAMLINE UPWIND/ PETROV-GALERKIN FEM BASED TIME-ACCURATE SOLUTION OF 3D TIME-DOMAIN MAXWELL’S EQUATIONS FOR DISPERSIVE MATERIALS

By

Srijith Rajamohan

A Dissertation Submitted to the Faculty of the University of Tennessee at Chattanooga in Partial Fulfillment of the Requirements of the Degree of Doctor of Philosophy in Computational Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

August 2014
ABSTRACT

Although the simulation of most broadband frequency responses are made under the assumption of constant electromagnetic material parameters, this is not a valid assumption for many materials found in nature. In this dissertation the time-accurate solution of the 3D time-domain Maxwell’s equations for dispersive materials in a Streamline Upwind/Petrov-Galerkin framework is investigated. For this purpose the permittivity associated with a material is expressed as a matrix, enabling the solution of anisotropic material models with multiple poles. Here diagonally isotropic models with up to 2 poles are investigated.

Near-field to far-field transformations are implemented to enable the solution of open boundary problems such as radiation patterns and radar cross sections. A unified perfectly matched layer absorbing boundary layer is implemented to efficiently terminate the computational region. Numerical simulations of these equations are tightly coupled together and compared against a loosely coupled approach to improve efficiency.

An alternative diagonal stabilization matrix is proposed which is implemented and compared with a non-sparse stabilization matrix derived from the flux Jacobians. Along with this new stabilization parameter, scalability is improved by coupling the equations for the perfectly matched layer with those of Maxwell’s equations. Further efficiency gains are achieved by allowing for a variable number of equations to be solved throughout the domain.
DEDICATION

Dedicated to my friends and family
ACKNOWLEDGEMENTS

I would like to thank all the members of my committee namely, Dr. Sreenivas, Dr. Wang and Dr. Mathews for their guidance and support. I would also like to thank Dr. Newman, who played a significant part in the creation of this dissertation. I would like to express my sincere gratitude to my advisor Dr. W. Kyle Anderson for spending countless hours disseminating his knowledge and whose wisdom and patience played a pivotal role in the completion of this work. I would also like to thank Dr. Swafford for giving me an opportunity to be a part of this wonderful institution. Finally, I would like to thank all my friends and colleagues at the SimCenter for their friendship and the many interesting and insightful discussions we have had over the years.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>DEDICATION</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xi</td>
</tr>
<tr>
<td><strong>CHAPTER</strong></td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. MAXWELL’S EQUATIONS AND SOLUTION METHODOLOGY</td>
<td>6</td>
</tr>
<tr>
<td>2.1. Governing equations and finite-element implementation</td>
<td>6</td>
</tr>
<tr>
<td>2.2. Shape functions and quadrature for FEM</td>
<td>10</td>
</tr>
<tr>
<td>2.3. Implicit time-stepping scheme</td>
<td>12</td>
</tr>
<tr>
<td>2.4. Linearization of the residual and the linear system solver</td>
<td>14</td>
</tr>
<tr>
<td>2.5. Boundary conditions</td>
<td>15</td>
</tr>
<tr>
<td>2.5.1. Perfect electric conductor</td>
<td>15</td>
</tr>
<tr>
<td>2.5.2. Silver-Muller</td>
<td>16</td>
</tr>
<tr>
<td>2.5.3. Farfield</td>
<td>17</td>
</tr>
<tr>
<td>2.6. Total Field Scattered Field (TFSF) formulation</td>
<td>19</td>
</tr>
<tr>
<td>2.7. Source waveforms</td>
<td>20</td>
</tr>
<tr>
<td>3. PERFECTLY MATCHED LAYERS</td>
<td>25</td>
</tr>
<tr>
<td>3.1. Introduction</td>
<td>25</td>
</tr>
<tr>
<td>3.2. PML formulation</td>
<td>28</td>
</tr>
<tr>
<td>3.3. PML implementation</td>
<td>31</td>
</tr>
<tr>
<td>4. NEAR-FIELD TO FAR-FIELD TRANSFORMATION</td>
<td>35</td>
</tr>
</tbody>
</table>
### LIST OF TABLES

8.1 Runtime comparison of blocked and 'unrolled' equations ........................................ 83
8.2 Runtime (ms) of Jacobian based \([\tau]\) and diagonal \([\tau]\) ........................................ 84
8.3 Material properties for the Debye model listed by pole number ................................. 95
8.4 Material properties for the Lorentz model listed by pole number ............................... 99
8.5 Material properties for the multilayer single pole Debye sphere model ..................... 103
8.6 Debye material properties for human brain ............................................................. 119
8.7 Runtime statistics of Fortran code vs CUDA kernel ................................................. 127
8.8 Average execution times for CUDA kernel vs Fortran code .................................... 128
### LIST OF FIGURES

2.1 Linear parent tetrahedron ................................................. 13
2.2 Quadratic parent tetrahedron ............................................. 13
2.3 Illustration of left and right sides in a Riemann solver ............. 18
2.4 TFSF illustrated by two regions ........................................... 20
2.5 (a) Time-domain waveform (b) discrete Fourier transform; frequency = 500Mhz . . 21
2.6 (a) Time-domain waveform (b) Fourier transform; frequency = 150Mhz, width = 100Mhz, m = 12 .................................................. 23
2.7 (a) Time-domain waveform (b) Fourier transform; frequency = 150Mhz, width = 100Mhz, m = 3 .................................................. 23
2.8 (a) Time-domain waveform (b) Fourier transform; frequency = 500Mhz, width = 100Mhz, m = 3 .................................................. 24
2.9 (a) Time-domain waveform (b) Fourier transform; frequency = 1.50Ghz, width = 100Mhz, m = 3 .................................................. 24
3.1 Coordinate stretching from [14] ............................................. 27
3.2 Variation of $\sigma_x$ along the x direction for linear, quadratic and quartic variation . . 33
3.3 PML layers in the x,y,z directions for layer thickness = 0.5, $\sigma$ = 75 in all directions 34
4.1 Closed contour for the surface equivalence theorem ..................... 43
7.1 Proposed algorithm for porting to the GPU ............................. 81
8.1 Gaussian wave propagation in the z direction - PEC sphere .......... 85
8.2 Bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz .......................... 86
8.3 Approximate diagonal stabilization vs Jacobian based stabilization - bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz .......................... 88

8.4 Approximate diagonal stabilization vs Jacobian based stabilization - bistatic RCS for PEC sphere of radius = 0.6m at frequency = 220 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz .......................... 89

8.5 PI results compared against HFSS using bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz .......................... 90

8.6 Gaussian wave of center frequency = 150 Mhz and bandwidth = 100 Mhz; propagation in the z direction in the presence of dielectric sphere with $\epsilon_r = 3$ ........ 92

8.7 Variation of $D_x$ component of plane wave propagating in z direction in the presence of dielectric sphere of $\epsilon_r = 3$ at frequency = 150 Mhz .......................... 93

8.8 Bistatic RCS for a frequency-independent dielectric sphere with $\epsilon_r = 3$ of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz .......................... 94

8.9 Gaussian wave propagation in the z direction in the presence of a dispersive sphere as a Debye single pole model .......................... 96

8.10 Bistatic RCS for a Debye model single pole sphere using $\epsilon_\infty = 2.88, \Delta \epsilon = 2.0$, $\tau_d = 0.1194$ at frequency = 150 Mhz .......................... 97

8.11 Bistatic RCS for a Debye model 2 pole sphere using $\epsilon_\infty = 2.88, \Delta \epsilon_1 = 2.0, \tau_{d1} = 0.1194$, $\Delta \epsilon_2 = 2.5, \tau_{d2} = 0.3$ at frequency = 150 Mhz .......................... 98

8.12 Gaussian wave propagation in the z direction in the presence of a dispersive sphere modeled using a Lorentz single pole model .......................... 100

8.13 Bistatic RCS for a Lorentz model single pole sphere using $\epsilon_\infty = 2.88, \Delta \epsilon = 2.0$, $\delta_p = 2.5$, $\omega_p = 0.1194$ at frequency = 150 Mhz .......................... 101

8.14 Bistatic RCS for a Lorentz model 2 pole sphere using $\epsilon_\infty = 2.88, \Delta \epsilon_1 = 2.0, \delta_{p1} = 2.5$, $\omega_{p1} = 0.1194$, $\Delta \epsilon_2 = 2.0, \delta_{p2} = 2.5$, $\omega_{p2} = 0.1194$ at frequency = 150 Mhz .......................... 102

8.15 Gaussian wave propagation in the z direction in the presence of a dispersive 2 layer sphere using Debye single pole models .......................... 104
8.16 Bistatic RCS for a 2 layer sphere with single pole Debye models. Outer layer $\epsilon_{\infty 1} = 2.88, \Delta \epsilon_1 = 2.0$, $\tau_{d1} = 0.1194$, and inner layer $\epsilon_{\infty 2} = 2.0, \Delta \epsilon_2 = 2.5$, $\tau_{d2} = 0.3$, at frequency $= 150 \text{ Mhz}$ ................................................. 105

8.17 Database created using hyperbolic sine clustering toward the ends, characteristic length $d = 0.62\text{m}$ ................................................................. 108

8.18 Coarse mesh for PEC NASA Almond with characteristic length $d = 0.62\text{m}$ ....... 109

8.19 Coarse mesh for PEC NASA Almond with characteristic length $d = 0.62\text{m}$, oriented in the z direction and in the x-y plane ................................. 110

8.20 Refined mesh for PEC NASA Almond with characteristic length $d = 0.62\text{m}$, oriented in the z direction and in the x-y plane ................................. 111

8.21 Gaussian wave propagation of center frequency $= 150 \text{ Mhz}$ and bandwidth $= 100 \text{ Mhz}$ in the z direction in the presence of a NASA Almond with a PEC boundary condition .................................................. 112

8.22 Bistatic RCS for a PEC NASA Almond with characteristic length $d = 0.62\text{m}$ frequency $= 150 \text{ Mhz}$ using a Gaussian pulse of center frequency $= 150 \text{ Mhz}$ and bandwidth $= 100 \text{ Mhz}$, time-step $= 0.0025\text{s}$ run for 5600 time-steps ............. 113

8.23 Gaussian wave propagation of center frequency $= 500 \text{ Mhz}$ and bandwidth $= 200 \text{ Mhz}$ in the z direction, in the presence of a NASA Almond with a PEC boundary condition .................................................. 114

8.24 Bistatic RCS for a PEC NASA Almond with characteristic length $= 0.62\text{m}$ frequency $= 500 \text{ Mhz}$ using a Gaussian pulse of center frequency $= 500 \text{ Mhz}$ and bandwidth $= 200 \text{ Mhz}$, time-step $= 0.00125\text{s}$ run for 11200 time-steps ............. 115

8.25 Illustration of the effect of temporal truncation error due to time-step. Bistatic RCS at $\phi = 36$ for a PEC NASA Almond with characteristic length $= 0.62\text{m}$, frequency $= 500 \text{ Mhz}$ using 3 different time-steps .................................................. 116

8.26 NASA Almond mesh in HFSS .................................................. 117

8.27 NASA Almond geometry in HFSS .................................................. 118

8.28 Brain mesh with 4 layers .................................................. 120

8.29 CSF layer of brain mesh .................................................. 121

8.30 White matter of brain mesh .................................................. 122

xiii
8.31 Aspect ratio for brain mesh ................................................................. 123
8.32 Maximum included angle for brain mesh ................................................. 124
8.33 Simulation of a 4 layer brain mesh at 500 Mhz ....................................... 125
8.34 Simulation of a 4 layer brain mesh at 500 Mhz ....................................... 126
CHAPTER 1
INTRODUCTION

For many years researchers have sought the high fidelity solution of electromagnetic problems using computational simulation of Maxwell’s equations while attempting to satisfy the twin goals of efficiency and efficacy. The solution of Maxwell’s equations is vital to problems such as antenna design, waveguide propagation and radar signature computation. Maxwell’s equations model wave phenomena and represent a linear hyperbolic system of equations for most problems. Although this permits closed-form solutions for simpler geometries, most meaningful problems require significant computational effort. While experimental methods are still invaluable, the ability to numerically simulate these complex problems results in significant cost savings.

One of the most popular methods in electromagnetic simulation is the finite-difference time-domain (FDTD) method proposed by Yee [1]. Here the computational region is discretized into uniform cuboids in three dimensional space where the electric fields are stored at the center of the faces and the magnetic fields are stored at the edges. Such an interleaved approach was chosen as opposed to collocating the electric and magnetic field vectors because the staggered approach can remove oscillations that often occur when using a central-difference scheme on an unstaggered grid [2]. Also, the staggered scheme reduces the total numerical phase velocity error compared to the unstaggered scheme [2]. Along with being a fully explicit method [3] it is also able to satisfy the requirements of tangential continuity of field intensities across material interfaces and provide divergence-free solution fields for a charge-free domain [4]. Despite the success and popularity of the FDTD method, it suffers from time-step constraints that are a direct result of the fact that this is an explicit method. As such, time-steps have to be chosen based on stability considerations and
Another approach that has gained traction as a result of its use in fluid dynamics is the finite-volume method (FVM). Finite-volume methods have been applied to the solution of Maxwell’s equations where the computational region is discretized into non-overlapping control volumes. Here the Maxwell’s curl equations are cast in a divergence form wherein the volume integral terms are converted to surface integrals allowing information flow to be treated as an exchange of flux fields through the control volume surface. Such methods have been implemented [5] but the resulting schemes are only second-order accurate which may be unacceptable for problems that involve higher frequencies [4]. While it is possible to resolve complicated geometries using a finite-volume method this approach requires a large stencil for higher-order discretizations resulting in complex data structures and larger memory requirements [4].

The finite-element method provides a means to accurately model geometries and obtain high solution accuracy with a nearest-neighbor stencil using higher-order polynomial basis functions. Also, using an implicit scheme allows time-steps to be dictated by physics instead of numerics. The discontinuous-Galerkin (DG) method derives its name from the fact that the field variables are discontinuous across element boundaries which results in mass matrices that are decoupled [6]. The Petrov-Galerkin (PG) method assumes continuous field variables across elements resulting in a globally coupled system of equations that needs to be solved iteratively. While both discontinuous-Galerkin and Petrov-Galerkin methods are used for solving electromagnetic problems, it has been shown in [4] that on a given mesh the approximate degrees of freedom for a DG scheme is about 24 times for a linear element and about 7 times for a quadratic element compared to the PG scheme. The PG method has a computational cost advantage when the fields are represented using cubic elements or less [4]. For this reason the Petrov-Galerkin method is utilized in this work.

In many implementations the Maxwell’s equations are posed as a single second-order wave equation in one of the field variables while the secondary variable is obtained during post processing. However this method suffers from the fact that the order of accuracy of the secondary
variable is one order less than the primary variable. In the present work Maxwell’s curl equations, i.e. Faraday’s law of induction and Ampere’s circuital law are expressed, in conservation form and solved in the time domain resulting in a system of six equations at each node. To derive a stable finite-element method, the resulting system of equations is multiplied by a suitably chosen weight function and integrated over the volume of an element. The choice of the weight function used in the current work provides stability for the computation. Because Maxwell’s equations represent wave propagation or convection, unstabilized Galerkin methods fail to damp out numerical oscillations for these types of problems [7, 8, 9, 10]. In this work the Streamline Upwind/Petrov-Galerkin (SUPG) scheme is used which provides stabilization along the direction of wave propagation. The stabilization matrix, often simply referred to as $[\tau]$, is computed from the flux Jacobians [11, 4].

In many implementations for solving Maxwell’s equations, the frequency dependence of the material parameters can be ignored and constant parameters are assumed. However this is an inaccurate model for many materials, and must be properly accounted for in simulations that compute broadband frequency responses. This frequency-dependent behavior is particularly important in the modeling of materials, such as biological tissues, where multipole models are often required over a large range of frequencies. Also metamaterials, which are a class of structures that have negative effective permeabilities and permittivities, are a topic of active research because of their ability to have a negative refractive index. These are frequency-dependent materials that have found use in the creation of super lenses, generation of cloaking devices and high gain antennas. Incorporating the frequency dependence into the curl equations involves modifying the constitutive equations, however this modification can be absorbed into the permittivity tensor. In this work an isotropic diagonal tensor is used, although the approach used poses no such restrictions and is suited for solving for fully anisotropic material models. An auxiliary differential equation approach is used to incorporate the polarization terms for a particular model.

The solution of common open boundary problems, such as the computation of the radiation
pattern of an antenna or scattering cross sections of an enclosed object, requires farfield information. Extending the computational region to the farfield is often impractical. The Near-to-Far-Field (NTFF) transformation theory [3], based on Huygens surface equivalence theorem [12], provides a means to truncate the computational region to the nearfield and still obtain the farfield information. The computation of the NTFF transformation requires the computation of Fourier transforms at each time-step. To optimize memory use and computational efficiency these are only computed and stored on the closed contour where the NTFF integrals are computed.

The simulation of electromagnetic problems often requires that different regions of the domain be modeled with different numbers of variables. This presents a problem when using a solver that uses a fixed block size abstraction. To build an efficient and scalable framework for these types of problems the system is “unrolled” to solve for each variable separately. Here, separate data structures are created to account for the loss of block abstraction.

In this implementation a perfectly matched layer (PML) absorbing region is used to terminate the computational region. PMLs were originally designed by Berenger [13] to model a lossy region that obeys Maxwell’s equations and would damp incident waves while not creating reflections. Moreover, they can be formulated to work with all angles of incidence. In this work a version of the PML proposed by Johnson [14] is used.

In the Petrov-Galerkin formulation, the cost associated with the computation of the stability matrix for a dispersive region can be significant [15]. This term involves an eigendecomposition followed by a singular value decomposition to compute a pseudoinverse because the intermediate matrix is rank deficient when polarization terms are included. Also, for matrices of dimension greater than 12 it is quite difficult to obtain symbolic eigendecompositions. Even at a block size of 12 the resulting matrices are quite cumbersome to incorporate into a code framework. To circumvent these issues an alternative diagonal formulation for the stabilization matrix is proposed that eliminates the eigendecomposition and renders the matrix inversion trivial.

While Maxwell’s equations can be solved in either the time domain or the frequency domain,
the need to model transient phenomena and obtain broadband frequency responses necessitate
the use of the time-domain approach. This is implemented through a time marching scheme to
obtain the time-accurate solution of the Maxwell’s equations. Fourier transforms of these solution
variables are computed through accumulated summation at each time-step, hence there is a need
to accurately resolve the field values through appropriate spatial and temporal sampling. The
desired frequency response can then be obtained during post-processing from the computed Fourier
transforms.

Graphics Processing Units (GPUs) have become quite popular as general purpose comput-
tational devices for high performance applications. However, most of the work in this field is
targeted at problems such as linear algebra routines that have regular memory access patterns. It is
not clear how such an architecture would fare at problems such as implicit FEM code where such
properties cannot be exploited. A residual routine is ported to an NVIDIA GPU using the Compute
Unified Device Architecture (CUDA), and it is compared against its Fortran implementation. The
challenges associated with this are discussed and some best practices suggested.

The suitability of this implementation as a scalable framework for solving open boundary and
multi-pole dispersive model problems is investigated. Accuracy of the solution using the original
Jacobian based $[\tau]$ and the proposed diagonal $[\tau]$ is investigated for dispersive and non-dispersive
models. Finally, an attempt is made to study the suitability of the GPU architecture to unstructured
problems and optimization strategies are suggested for the same.
CHAPTER 2
MAXWELL’S EQUATIONS AND SOLUTION METHODOLOGY

2.1 Governing equations and finite-element implementation

The time dependent Maxwell’s curl equations [4] are given by

\[ \nabla \times E = -\frac{\partial B}{\partial t} \] (2.1)
\[ \nabla \times H = \frac{\partial D}{\partial t} + J \] (2.2)
\[ \nabla \cdot B = 0 \] (2.3)
\[ \nabla \cdot D = \rho_c \] (2.4)

where \( E \) and \( H \) are electric and magnetic field intensities whereas \( D \) and \( B \) are electric and magnetic flux densities. These are related by the constitutive equations as

\[ D = \epsilon E \] (2.5)
\[ B = \mu H \] (2.6)

where \( \epsilon \) is the permittivity and \( \mu \) is the permeability of the medium. The flux densities are the fundamental variables in this formulation. In Eq. (2.4), \( \rho_c \) represents the charges enclosed by a closed surface and \( J \) in Eq. (2.2) represents the current density. Equations (2.1) and (2.2) are a system of six equations in six unknowns that can be written in a divergence form given by

\[ \frac{\partial Q}{\partial t} + \nabla \cdot F(Q) = 0 \] (2.7)
where \( Q \) is the list of electric (\( D \)) and magnetic (\( B \)) flux densities

\[
Q = (D_x, D_y, D_z, B_x, B_y, B_z) \tag{2.8}
\]

\[
F = \hat{\mathbf{i}} f + \hat{\mathbf{j}} g + \hat{\mathbf{k}} h \tag{2.9}
\]

\[
f = (0, H_z, -H_y, 0, -E_z, E_y) \tag{2.10}
\]

\[
g = (-H_z, 0, H_x, E_z, 0, -E_x) \tag{2.11}
\]

\[
h = (H_y, -H_x, 0, -E_y, E_x, 0) \tag{2.12}
\]

The solution to the governing equations is obtained using a Petrov-Galerkin scheme, which is a weighted residual method that can be written in the following form

\[
\int_{\Omega} \int_{\Omega} [\phi] \left( \frac{\partial Q}{\partial t} + \nabla \cdot F(Q) \right) d\Omega = 0 \tag{2.13}
\]

where \( \Omega \) is the space of the computational region that comprises the interior nodes as well as surface or boundary nodes. In a Bubnov-Galerkin system \([\phi]\) would be taken from the space of test functions or shape functions [10]. Such an approach becomes unstable for a hyperbolic system of equations such as the Maxwell’s equations on coarse grids because it does not provide sufficient damping or dissipation of the errors. This is equivalent to solving the system of equations in the finite-difference method using a central-difference scheme. The Streamline Upwind/Petrov Galerkin method (SUPG) [7] mitigates this problem through upwinding in the direction of the wave propagation. The added dissipation stabilizes the finite-element method. In SUPG \([\phi]\) is a weight function given as

\[
[\phi] = N[I] + \left( \frac{\partial N}{\partial x} [A] + \frac{\partial N}{\partial y} [B] + \frac{\partial N}{\partial z} [C] \right) [\tau] \tag{2.14}
\]
where

\[ N = \sum_{i}^{n} N_{i} c_{i} \quad (2.15) \]

Here \( N_{i} \) represents a basis function, \( n \) is the number of basis functions, \( c_{i} \) is an arbitrary constant, and \([A], [B], [C]\) are flux Jacobian matrices. The \([\tau]\) matrix, which controls the amount of dissipation that is added, is given by [11, 4]

\[ [\tau]^{-1} = \sum_{k=1}^{n} \left| \frac{\partial N_{k}}{\partial x} [A] + \frac{\partial N_{k}}{\partial y} [B] + \frac{\partial N_{k}}{\partial z} [C] \right| \quad (2.16) \]

The residual equation can be expanded using Eq. (2.14) as

\[
\iint_{\Omega} \left( N[I] + [P] \right) \left( \frac{\partial Q}{\partial t} + \nabla \cdot F \right) d\Omega = 0 \quad (2.17)
\]

\[
\iint_{\Omega} \left( N \frac{\partial Q}{\partial t} + N \nabla \cdot F + [P] \frac{\partial Q}{\partial t} + [P] \nabla \cdot F \right) d\Omega = 0 \quad (2.18)
\]

where \([P]\) is written as

\[ [P] = \left( \frac{\partial N}{\partial x} [A] + \frac{\partial N}{\partial y} [B] + \frac{\partial N}{\partial z} [C] \right) [\tau] \quad (2.19) \]

Expanding Eq. (2.17) using Green’s theorem and denoting the boundary of the computational region by \( \Gamma \)

\[
\iint_{\Omega} \left( N \frac{\partial Q}{\partial t} + [P] \frac{\partial Q}{\partial t} + [P] \nabla \cdot F \right) d\Omega + \int_{\Gamma} NF \cdot \hat{n} d\Gamma - \iint_{\Omega} F \cdot \nabla N = 0 \quad (2.20)
\]

Rearranging these terms, the weak form is obtained as

\[
\iint_{\Omega} \left( N \frac{\partial Q}{\partial t} - F \cdot \nabla N \right) d\Omega + \iint_{\Omega} [P] \left( \frac{\partial Q}{\partial t} + \nabla \cdot F \right) d\Omega + \int_{\Gamma} NF \cdot \hat{n} d\Gamma = 0 \quad (2.21)
\]
Because field variables are continuous across element boundaries, the surface integral is only required to be computed on the boundaries of the computational region and along material discontinuities where jumps exist in the tangential flux densities. To accurately resolve these jumps in field values, duplicate nodes are created at the interface and a flux-splitting approach based on a Riemann solver is taken [4]. Here, the flux through the interface can be written as

$$F(q_l, q_r) \cdot \hat{n} = \frac{1}{2} \left( F(q_l) + F(q_r) - [\tilde{T}][\tilde{\Lambda}][\tilde{T}]^{-1}[M]\Delta q \right)$$  \hspace{1cm} (2.22)

Here the tilde over the variables indicate that they are computed using simple averaged values based on the material properties and field values on either side of the interface. $[\tilde{T}]$ and $[\tilde{\Lambda}]$ are the matrices of right eigenvectors and eigenvalues of the term given by

$$[\tilde{T}][\tilde{\Lambda}][\tilde{T}]^{-1} = \left| \frac{\partial N_k}{\partial x} [\tilde{A}] + \frac{\partial N_k}{\partial y} [\tilde{B}] + \frac{\partial N_k}{\partial z} [\tilde{C}] \right|$$  \hspace{1cm} (2.23)

Also $\Delta q$, which represents the difference in field intensities instead of the flux densities, is used in evaluating this Riemann solver. This is due to the fact that it is not possible to satisfy the jumps in terms of the flux densities exactly whereas this can be accomplished by using the difference in field intensities and factoring out the matrix $[M]$, which is given as

$$[M] = \left[ \frac{\partial Q}{\partial q} \right]$$  \hspace{1cm} (2.24)

Additionally, boundary conditions are applied by weakly enforcing them through the flux term in the surface integral.
2.2 Shape functions and quadrature for FEM

In the finite-element method, solution variables are available at each time-step at node locations whereas values within the element are obtained by interpolating using appropriate shape functions. These functions must be designed in such a way that when the mesh is refined the approximate solution computed through FEM converges to the exact solution. In this work first-order and second-order shape functions based on Lagrange polynomials are used. First-order Lagrange shape functions are linear (P1) whereas second-order shape functions are quadratic (P2). Furthermore, the shape functions over the element are continuous across edges of neighboring elements. Although quadratic elements offer increased accuracy they have the adverse effect of increased problem size and decreasing sparsity in the linearization matrix. It has been observed in [4] that meshes with P2 elements have roughly ten times as many degrees of freedom compared to meshes with P1 elements. Nevertheless, for high fidelity solutions where the error should not exceed a certain threshold, the third-order accurate solution provided by P2 elements justify their cost and require less computational effort than the second-order accurate solution methods to obtain similar levels of accuracy.

Shape functions are obtained by mapping triangular and tetrahedral elements to parent elements in non-dimensional natural coordinates \((\xi, \eta, \rho)\) that extend between zero and one [16]. For a linear tetrahedron with 4 nodes, such as that shown in Fig. (2.1), the shape functions can be written as

\[
N(\xi, \eta, \rho) = \begin{pmatrix}
1 - \xi - \eta - \rho \\
\xi \\
\eta \\
\rho
\end{pmatrix}
\]  

(2.25)
Similarly, for a quadratic tetrahedron with 10 nodes, as that shown in Fig. (2.2), the shape functions can be written as

\[
N(\xi, \eta, \rho) = \begin{pmatrix}
1 - 3(\xi + \eta + \rho) + 2(\xi^2 + \eta^2 + \rho^2) + 4(\xi\eta + \eta\rho + \rho\xi) \\
2\xi(\xi - 0.5) \\
2\eta(\eta - 0.5) \\
2\rho(\rho - 0.5) \\
4\xi(1 - (\xi + \eta + \rho)) \\
4\xi\eta \\
4\eta(1 - (\xi + \eta + \rho)) \\
4\rho(1 - (\xi + \eta + \rho)) \\
4\xi\rho \\
4\eta\rho
\end{pmatrix}
\]  
(2.26)

The interpolant at any point \((x, y, z)\) in an element can then be written as

\[
U(x, y, z) = \sum_{i=1}^{n\text{nodes}} u_i N_i
\]  
(2.27)

where \(n\text{nodes}\) here is the number of basis functions associated with the element and \(u_i\) is the value of the function at each node. Higher-order shape functions can be computed using the procedure described in [17]. The integral of any function in a spatially discretized element is evaluated in the non-dimensional natural coordinate space \((\xi, \eta, \rho)\). Numerical quadrature rules are used to estimate the integral by computing the sum of solution values at Gauss quadrature points multiplied by appropriate weights [18]. Any function can be integrated over the tetrahedron as

\[
\iint_{\Omega} f(x, y, z) d\Omega \approx \sum_{i=1}^{n\text{gauss}} f_i(x(\xi, \eta, \rho), y(\xi, \eta, \rho), z(\xi, \eta, \rho)) W_i J
\]  
(2.28)
Here \( n_{gauss} \) is the number of Gauss points and \( W_i \) are the quadrature weights, such as those tabulated in [18]. \( J \) is the Jacobian of the element in physical space. Note that when evaluating volume integrals whose integrand is evaluated using polynomials of order \( P \), the number of Gauss points is chosen so that polynomials of order \( 2P \) can be integrated exactly. On surfaces, quadrature formulas are used that can integrate polynomials of order \( 2P + 1 \) [4].

2.3 Implicit time-stepping scheme

An implicit time-stepping scheme is used that alleviates the problem associated with explicit schemes of having to use extremely small time-step values required to maintain stability. This problem often occurs in meshes with widely varying grid spacing. With the use of an implicit time-stepping scheme, a time-step can be used that more adequately represents the time scale of the problem. The residual at each time-step can be expressed as

\[
\mathbf{R}^{n+1} = \iiint_{\Omega} \left( N \frac{\partial \mathbf{Q}}{\partial t} - (\mathbf{F} \cdot \nabla N)^{n+1} \right) d\Omega + \iint_{\Omega} \left( [P]^{n+1} \left( \frac{\partial \mathbf{Q}}{\partial t} + [\nabla \cdot \mathbf{F}]^{n+1} \right) \right) d\Omega + \int_{\Gamma} \mathbf{N}^{n+1} \cdot \mathbf{n} d\Gamma = 0
\]

(2.29)

After discretization, the solution is advanced to the next time-step using Newton’s method given by

\[
\left[ \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right]^{n+1} \Delta \mathbf{Q}^{n+1} = -\mathbf{R}^{n+1}
\]

(2.30)

Note that because of the linearity of the equations only one Newton step is required.
Figure 2.1 Linear parent tetrahedron

Figure 2.2 Quadratic parent tetrahedron
2.4 Linearization of the residual and the linear system solver

For materials with fixed properties, the linearization matrix \( \frac{\partial R}{\partial Q} \) is a sparse block matrix with a block size corresponding to the number of equations solved for at each node, and the number of rows is equivalent to the number of nodes in the mesh. When the number of equations to be solved at each mesh point varies throughout the mesh it is more appropriate to state the problem size in terms of the total number of solution variables within the entire computational domain instead of grouping them into blocks. This is precisely what is done in an ‘unrolled’ linearization matrix where no block abstraction exists. Because Maxwell’s equations are linear, exact computation of the linearization matrix can be achieved using a low order finite difference formula. This is accomplished by perturbing each unknown by a small value \( \delta x \), computing the resulting residual and subtracting the residual computed without perturbations.

\[
\frac{\partial R}{\partial Q} = \frac{R(Q + \delta x) - R(Q)}{\delta x} \tag{2.31}
\]

Due to the sparsity of the matrix, the entries are stored using compressed row storage for efficiency [19]. The level of sparseness depends on the order of the spatial element. Higher order elements decrease the sparsity of the matrix by introducing more off-diagonal terms, which may also have an adverse effect on the diagonal dominance of the matrix. To solve the linear system, the Generalized Minimum Residual method (GMRES) [19] is used. In practice, GMRES is used in conjunction with a preconditioner to accelerate convergence, thereby limiting the number of search directions. The preconditioner needs to be as efficient as possible while still providing an effective approximation to the matrix inverse. A preconditioner based on an incomplete LU factorization of the original system is used in this work. For electromagnetic problems ILU(0) [19] has been found to work well. Because of the linearity of the system, both the linearization and the preconditioner needs to be computed only once at the beginning of the time-marching process.
2.5 Boundary conditions

Depending on the material and the nature of the propagating wave, three boundary conditions are used in the current work. These are weakly enforced through the flux terms in the surface integral in (2.21). The flux normal to the surface is written as

\[
F \cdot \hat{n} = \begin{pmatrix}
-n_y H_z + n_z H_y \\
n_x H_z - n_z H_x \\
-n_y H_z + n_z H_y \\
n_y E_z - n_z E_y \\
-n_x E_z + n_z E_x \\
n_y E_z - n_z E_y
\end{pmatrix}
\] (2.32)

This flux can be modified appropriately depending on the required boundary condition. The modified fluxes for each of the three boundary conditions is given below.

2.5.1 Perfect electric conductor

When the material is a pure conductor, or one that has a high value of conductivity with very little resistance, one can apply the perfect electric conductor (PEC) boundary condition. This is based on the notion that the electric field inside the conductor is zero, combined with the fact that the tangential electric field is continuous across an interface. The two conditions can be reconciled to give the following condition

\[
E_t = E \times \hat{n} = 0
\] (2.33)
where $E_t$ is the tangential electric field. It must be kept in mind that this is fundamentally an approximation that ignores the phenomenon of skin effect [3] and may not be valid at all frequencies. The flux normal to a PEC surface can be obtained by applying Eq.(2.33) to Eq. (2.32). The last three terms in Eq. (2.32) correspond to the components of the term $E \times \hat{n}$ which are all zero for a PEC. Hence the flux on the surface can be written as

$$\mathbf{F} \cdot \hat{n} = \begin{pmatrix} -n_y H_z + n_z H_y \\ n_x H_z - n_z H_x \\ -n_y H_z + n_z H_y \\ 0 \\ 0 \\ 0 \end{pmatrix}$$ (2.34)

2.5.2 Silver-Muller

The Silver-Muller [20, 21] boundary condition is an absorbing boundary condition used to terminate radiating or guided waves that are incident normal to the boundary. For the case of normal waves this can be written as

$$(E - B \times \hat{n}) \times \hat{n} = 0$$ (2.35)
Equation (2.35) can be expanded as

\[ E \times \hat{n} = (B \times \hat{n}) \times \hat{n} \]  

(2.36)

\[ (E \times \hat{n})_x = i \left[ (B_z x_n - B_x z_n) z_n - (B_y y_n - B_y x_n) y_n \right] \]  

(2.37)

\[ (E \times \hat{n})_y = j \left[ (B_y y_n - B_y x_n) x_n - (B_y z_n - B_z y_n) z_n \right] \]  

(2.38)

\[ (E \times \hat{n})_z = k \left[ (B_y z_n - B_z y_n) y_n - (B_z y_n - B_y x_n) x_n \right] \]  

(2.39)

Hence this is weakly enforced through the flux as

\[
F \cdot \hat{n} = \begin{pmatrix}
-n_z H_z + n_z H_y \\
n_z H_z - n_z H_z \\
-n_y H_z + n_y H_z \\
(B_z x_n - B_x z_n) z_n - (B_y y_n - B_y x_n) y_n \\
(B_x y_n - B_y x_n) x_n - (B_y z_n - B_z y_n) z_n \\
(B_y z_n - B_z y_n) y_n - (B_z y_n - B_y x_n) x_n
\end{pmatrix}
\]  

(2.40)

2.5.3 Farfield

Farfield boundary conditions are imposed to terminate an absorbing layer or a boundary that is sufficiently far enough away from the source. These are applied through a Riemann solver on the boundary surface elements, where the left and right sides are as illustrated in Fig. (2.3). The values on the left side in the Riemann solver, denoted as \( q_l \), are obtained from within the domain whereas the values on the right side, denoted by \( q_r \), are obtained from outside the domain. For an absorbing layer, as described in Chapter 3, the variables on the left and right sides of the interface represent scattered field variables, which are described in the next section. Note that on the right side of
the interface, these values are all zero as shown in Eq. (2.41). The farfield boundary condition is applied using the Riemann solver given in Eq. (2.22) where the eigenvalues and eigenvectors are computed using averaged values obtained from $q_l$ and $q_r$. The material properties for the left and the right states are the same as that of the absorbing layer.

![Figure 2.3 Illustration of left and right sides in a Riemann solver](image)

$$q_r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$ (2.41)
2.6 Total Field Scattered Field (TFSF) formulation

The Total Field Scattered Field (TFSF) formulation is based on the principle of superposition of Maxwell’s equations. The electric and magnetic fields at any point in the mesh can be decomposed as

\[ E_{\text{total}} = E_{\text{inc}} + E_{\text{scat}} \]  
\[ H_{\text{total}} = H_{\text{inc}} + H_{\text{scat}} \]  

Here, \( E_{\text{inc}} \) and \( H_{\text{inc}} \) are the values of the incident fields imposed in the computational field. Usually this is applied at a driving port, which is an interface with duplicate faces specifically designed for this purpose [4]. \( E_{\text{scat}} \) and \( H_{\text{scat}} \) are the values of the scattered fields, which result from the interaction of the incident wave with objects in the computational space. In the presence of a scattering object the total field can be computed as the sum of these two fields. This is illustrated using a rectangular closed contour \( S \) in Fig. (2.4) where in region \( R_1 \) the solution variables are the total fields whereas in the outer region, \( R_2 \), only the scattered fields are stored. To maintain consistency at the interface of the two regions, the incident fields need to be added to the solution in the \( R_2 \) region when resolving jump conditions using a Riemann solver for computing the residual for nodes that are along the interface and within the primary field \( S \). This procedure also allows a way to apply a driving source wave and confine it to region \( R_1 \) while being transparent to any outgoing scattered waves [3]. Such a scheme also avoids the subtractive cancellation errors associated with a pure scattered calculation [3].
2.7 Source waveforms

Excitations, or driving source waveforms, are chosen to include only the desired frequencies of interest and should ideally be activated and deactivated smoothly to avoid introducing high frequency components into the solution space. Two popular forms of driving sources are the sine wave and the cosine modulated Gaussian pulse [22]. If the response at a single frequency is desired a sine wave can be used but care must be taken to avoid introducing spurious high frequency components caused by instant initiation of the waveform [3]. The sinusoidal source wave of frequency 500Mhz and its discrete Fourier transform, is shown below in Fig. (2.5).

A Gaussian pulse can be written in the time domain as [22]

$$g(t) = e^{-t^2/\alpha^2}$$

(2.44)
where \( \alpha \) is a parameter that controls the width of the Gaussian pulse in the time domain. One of the advantages of the Gaussian pulse is that its Fourier transform is also a Gaussian waveform. This is beneficial because it implies that this source waveform is limited in both the time and frequency domains. The Fourier transform of the Gaussian pulse waveform given by Eq. (2.44) can be written as

\[
G(\omega) = \alpha \sqrt{\pi} e^{-\alpha^2 \omega^2/4}
\]  \hspace{1cm} (2.45)

Another advantage of the Gaussian pulse is that the gradual increase in amplitude eliminates spurious high frequency components. Note that care must be taken when generating meshes to ensure sufficient mesh density along the direction of wave propagation to adequately resolve all the frequency components in the Gaussian pulse [23]. This can be accomplished by noting that the
highest frequency resolvable by a mesh is given as

\[ f_{\text{max}} = \frac{c}{n\Delta x} \]  

(2.46)

where \( \Delta x \) is the mesh spacing and \( n \) is a parameter that determines the number of cells per wavelength. According to the Nyquist theorem [23] the number of cells per wavelength, given by \( n \), needs to be more than two, although in practice a number closer to ten is typically used for second-order accurate methods. For FDTD methods, the number of cells per wavelength is usually selected to resolve the maximum frequency. This frequency is selected to be the one that corresponds to 10 percent of the peak amplitude in a Gaussian pulse [23]. Solving for \( n \) using Eq. (2.45) the following estimate for \( \alpha \) is obtained

\[ \alpha = \frac{\sqrt{2.3}}{\pi f_{\text{max}}} \]  

(2.47)

For the purpose of obtaining a frequency band response centered at a desired frequency a cosine modulated Gaussian pulse is used. It is given in the time domain by the following equations

\[ V = \cos(\omega(t - t_0))e^{-(t-t_0)^2/\alpha} \]  

(2.48)

\[ t_0 = \sqrt{m\alpha} \]  

(2.49)

The parameters \( t_0 \) and \( m \) can be used to control the delay of the waveform [23]. Figure (2.6a) shows a Gaussian pulse of 150 Mhz followed by its Fourier transform in Fig. (2.6b), when \( m \) is set to 12. Figure (2.7a) shows the Gaussian pulse in the time and Fig. (2.7b) illustrates its spectral content in the frequency domain, when the parameter \( m \) is set to 3. As further examples, Fig. (2.8) and Fig. (2.9) depict waveforms with center frequencies at 500Mhz and 1.5Ghz respectively. It must be noted that the higher frequency sources require greater number of mesh points to appropriately resolve the faster oscillations.
Figure 2.6 (a) Time-domain waveform (b) Fourier transform; frequency = 150Mhz, width = 100Mhz, m = 12

Figure 2.7 (a) Time-domain waveform (b) Fourier transform; frequency = 150Mhz, width = 100Mhz, m = 3
Figure 2.8 (a) Time-domain waveform (b) Fourier transform; frequency = 500Mhz, width = 100Mhz, m = 3

Figure 2.9 (a) Time-domain waveform (b) Fourier transform; frequency = 1.50Ghz, width = 100Mhz, m = 3
CHAPTER 3
PERFECTLY MATCHED LAYERS

3.1 Introduction

One of the problems associated with the numerical solution of a PDE, or a system of PDEs, is the termination of the computational region associated with it. The difficulty arises from the need to do so without introducing artifacts or unwanted reflections into the computation. It is also desirable to keep the computational region to a minimum to reduce the time and resources required to compute such a solution. Although there are problems, such as those that are periodic, that present a natural way to terminate the computational region, most problems that model wave propagation require approximate methods, based on reasonable theoretical assumptions, to do so. Such problems that exhibit wave-like phenomena have solutions that either oscillate or gradually decay as they propagate. Attempts to terminate a region for this type of problem using a hard-wall boundary condition will result in reflections from this boundary. The approach often used for such problems is to terminate the region with an absorbing layer that provides an effective gradual damping of the waves.

The idea of an absorbing boundary layer was first proposed by Berenger [13], until which time an absorbing boundary condition was used to truncate the computational space. Absorbing boundary conditions suffered from several limitations, namely the need for normal incidence at the absorbing boundary. However, Berenger proposed adding a layer of cells around the desired solution space consisting of a special absorbing medium that would exponentially attenuate the incoming waves. Although transitions from one material to another typically create reflections,
the medium, as proposed by Berenger, was designed to keep reflections to a minimum at the interface. Berenger’s approach was based on splitting the wave solutions into the sum of two artificial field components, referred to in the literature as the split-field PML. A more commonly used approach is the uniaxial PML (UPML) which expresses the PML region as having modified anisotropic material properties [14]. An alternate approach to this problem was to derive the PML as a complex coordinate stretching [24, 14]. In this approach Maxwell’s equations are analytically continued into the complex spatial coordinates where the fields decay exponentially. The waves exiting the domain are expressed by the superposition of planewaves as

\[ w(x, t) = \sum_{k=1}^{n} W_{k,\omega} e^{i(k\cdot x - \omega t)} \]  

(3.1)

where \( w(x, t) \) is the amplitude of the planewave as a function of its spatial coordinates and time given as a sum of \( n \) waves of constant amplitudes \( W_{k,\omega} \). Here \( k \) is the wavevector and \( \omega \) is the angular frequency associated with each component.

Equation (3.1) is an analytic function that is evaluated for real values of \( x \), however this can be analytically continued to the complex domain. For example Fig. (3.1) demonstrates the difference in \( w(x, t) \) when \( x \) is purely real or when it has a non zero imaginary component. The top part of Fig. (3.1) depicts the solution for purely real values of \( x \) whereas the lower portion of the figure shows a linearly growing imaginary component for values of \( x \) greater than 5. Next to this, the corresponding damped wave as a function of the real part of the spatial coordinate \( x \) is illustrated. The damping of the wave when imaginary components are introduced is because the spatial part of Eq. (3.1) now looks like the following

\[ e^{ik((Re(x)+iIm(x))} = e^{jRe(x)}e^{-kIm(x)} \]  

(3.2)
Figure 3.1 Coordinate stretching from [14]
From the above equation it is obvious that for a value of \( k \) greater than 0 and an increasing imaginary part of \( x \) the region provides an exponential damping, effectively acting as an absorbing region.

### 3.2 PML formulation

The Maxwell’s curl equations can be expressed in the time domain as

\[
\nabla \times \mathbf{E}(x,t) = -\mu \frac{\partial \mathbf{H}(x,t)}{\partial t} \tag{3.3}
\]

\[
\nabla \times \mathbf{H}(x,t) = \epsilon \frac{\partial \mathbf{E}(x,t)}{\partial t} + \mathbf{J}(x,t) \tag{3.4}
\]

where the constitutive equations given below have been used to expand \( \mathbf{D} \) and \( \mathbf{B} \).

\[
\mathbf{D} = \epsilon \mathbf{E} \tag{3.5}
\]

\[
\mathbf{B} = \mu \mathbf{H} \tag{3.6}
\]

The Fourier transform can be applied to the terms in equations above

\[
\tilde{\mathbf{E}}(x,\omega) = \int_{-\infty}^{\infty} \mathbf{E}(x,t)e^{-i\omega t} dt \tag{3.7}
\]

\[
\frac{\partial \tilde{\mathbf{H}}}{\partial t}(x,\omega) = \int_{-\infty}^{\infty} \frac{\partial \mathbf{H}(x,t)}{\partial t}e^{-i\omega t} dt \tag{3.8}
\]

\[
\tilde{\mathbf{H}}(x,\omega) = \int_{-\infty}^{\infty} \mathbf{H}(x,t)e^{-i\omega t} dt \tag{3.9}
\]

\[
\frac{\partial \tilde{\mathbf{E}}}{\partial t}(x,\omega) = \int_{-\infty}^{\infty} \frac{\partial \mathbf{E}(x,t)}{\partial t}e^{-i\omega t} dt \tag{3.10}
\]
where the $\tilde{E}(x, \omega), \frac{\partial \tilde{H}}{\partial t}(x, \omega), \tilde{H}(x, \omega), \frac{\partial \tilde{E}}{\partial t}(x, \omega)$ are Fourier transforms of the respective quantities in the time domain. Evaluating the time derivatives of the Fourier transform as given by Eqs. (3.8) and (3.10) results in the following equations

$$
\frac{\partial \tilde{H}}{\partial t}(x, \omega) = i\omega \tilde{H}(x, \omega) \quad (3.11)
$$

$$
\frac{\partial \tilde{E}}{\partial t}(x, \omega) = i\omega \tilde{E}(x, \omega) \quad (3.12)
$$

The curl equations can be rewritten in the frequency domain as

$$
\nabla \times \tilde{E}(x, \omega) = -i\omega\mu \tilde{H}(x, \omega) \quad (3.13)
$$

$$
\nabla \times \tilde{H}(x, \omega) = i\omega\epsilon \tilde{E}(x, \omega) + \tilde{J}(x, \omega) \quad (3.14)
$$

At this point, these two equations are in a form that facilitates the application of complex coordinate stretching [25]. The material tensor transformation as given in [25, 14] is shown in Eqs. (3.15) and (3.16) below

$$
\tilde{\epsilon} = \frac{[J] \epsilon(\omega) [J]^T}{\det[J]} \quad (3.15)
$$

$$
\tilde{\mu} = \frac{[J] \mu(\omega) [J]^T}{\det[J]} \quad (3.16)
$$

where $\epsilon(\omega)$ and $\mu(\omega)$ represent the frequency dependent material tensors permittivity and permeability respectively. $[J]$ is a diagonal matrix of PML with damping coefficients given by

$$
[J] = \begin{bmatrix}
Sx^{-1} & 0 & 0 \\
0 & Sy^{-1} & 0 \\
0 & 0 & Sz^{-1}
\end{bmatrix} \quad (3.17)
$$
According to [14], application of the material tensor transformations given by Eq. (3.17) results in the following PML auxiliary differential equations

\[
\frac{\partial(K_x U_x)}{\partial t} + \sigma_x U_x + \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} = 0 \tag{3.18}
\]

\[
\frac{\partial(K_y U_y)}{\partial t} + \sigma_y U_y + \frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x} = 0 \tag{3.19}
\]

\[
\frac{\partial(K_z U_z)}{\partial t} + \sigma_z U_z + \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} = 0 \tag{3.20}
\]

\[
\frac{\partial(K_x V_x)}{\partial t} + \sigma_x V_x = \frac{\partial(\mu K_x E_x)}{\partial t} + \sigma_z \varepsilon E_x \tag{3.21}
\]

\[
\frac{\partial(K_y V_y)}{\partial t} + \sigma_y V_y = \frac{\partial(\mu K_y E_y)}{\partial t} + \sigma_x \varepsilon E_y \tag{3.22}
\]

\[
\frac{\partial(K_z V_z)}{\partial t} + \sigma_z V_z = \frac{\partial(\mu K_z E_z)}{\partial t} + \sigma_y \varepsilon E_z \tag{3.23}
\]

The above six equations are in terms of the PML variables \(U_x, U_y, U_z, V_x, V_y, V_z\). \(\sigma_x, \sigma_y\) and \(\sigma_z\) are the PML damping factors in the x, y and z directions respectively. These variables are the set of unknowns to be solved for in the PML. The Maxwell equations corresponding to Ampere’s law take the form as shown below

\[
\frac{K_x K_z \partial(\mu H_x)}{\partial t} + \frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z} + \frac{K_y}{K_x} \sigma_x \mu H_x - \frac{K_y}{K_x} \sigma_x U_x + \sigma_x U_x = 0 \tag{3.24}
\]

\[
\frac{K_x K_y \partial(\mu H_y)}{\partial t} + \frac{\partial E_z}{\partial z} - \frac{\partial E_z}{\partial x} + \frac{K_z}{K_y} \sigma_y \mu H_y - \frac{K_z}{K_y} \sigma_y U_y + \sigma_y U_y = 0 \tag{3.25}
\]

\[
\frac{K_x K_z \partial(\mu H_z)}{\partial t} + \frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y} + \frac{K_y}{K_z} \sigma_z \mu H_z - \frac{K_y}{K_z} \sigma_z U_z + \sigma_z U_z = 0 \tag{3.26}
\]
Similarly, equations corresponding to Faraday’s law can be rewritten as

\[
\begin{align*}
\frac{K_y K_z}{K_x} \frac{\partial (\varepsilon E_x)}{\partial t} + \frac{\partial H_y}{\partial z} - \frac{\partial H_z}{\partial y} + \frac{K_y}{K_x} \sigma_z \varepsilon E_x - \frac{K_y}{K_x} \sigma_x V_x + \sigma_y V_x &= 0 \quad (3.27) \\
\frac{K_z K_x}{K_y} \frac{\partial (\varepsilon E_y)}{\partial t} + \frac{\partial H_z}{\partial x} - \frac{\partial H_x}{\partial z} + \frac{K_z}{K_y} \sigma_x \varepsilon E_y - \frac{K_z}{K_y} \sigma_y V_y + \sigma_x V_y &= 0 \quad (3.28) \\
\frac{K_x K_z}{K_y} \frac{\partial (\varepsilon E_z)}{\partial t} + \frac{\partial H_x}{\partial y} - \frac{\partial H_y}{\partial x} + \frac{K_x}{K_y} \sigma_y \varepsilon E_z - \frac{K_x}{K_y} \sigma_z V_z + \sigma_x V_z &= 0 \quad (3.29)
\end{align*}
\]

It must be noted that, if \( K_x, K_y \) and \( K_z \) are set to 1, the above equations Eq. (3.24 - 3.29) are essentially the Maxwell’s equations with a source term added to them.

### 3.3 PML implementation

In [14] the damping factor is proportional to the term \( k/\omega \), which is a constant for a non-dispersive medium. Here \( k \) is the wave vector and \( \omega \) is the angular frequency of the wave. This implies that the attenuation in the region is independent of wave frequency. All frequencies are attenuated at the same rate, which is significant for a broadband source wave such as a Gaussian pulse. It must be noted here that even though the attenuation is independent of the frequency it is still dependent on the angle of incidence of the incoming waves. Since the PML attenuates the waves, the absorbing region can be terminated with a hard wall boundary condition such as a Dirichlet boundary condition [14]. Care must be taken to put this boundary condition at a large enough distance from the PML interface. As long as the PML is thick enough and the Dirichlet boundary condition far enough away, any residual reflection would get attenuated on the way back. Even though a PML is reflectionless at infinite resolution, at finite mesh spacing the discretization effects can cause reflections at the interface of the PML and computational region. In order to minimize these reflections the value of \( \sigma \) can be slowly increased using a power law so that \( \sigma \) is
zero at the interface and subsequently rises until it reaches the maximum desired value of $\sigma_{\text{max}}$ in each direction. The components of $\sigma$ in each direction for a node in the PML can be written as

$$\sigma_x(\text{node}) = \left[ \frac{x_{\text{node}} - x_{\text{ref}}}{L_x \text{pml}} \right]^n \sigma_{x_{\text{max}}},$$  

$$\sigma_y(\text{node}) = \left[ \frac{y_{\text{node}} - y_{\text{ref}}}{L_y \text{pml}} \right]^n \sigma_{y_{\text{max}}},$$  

$$\sigma_z(\text{node}) = \left[ \frac{z_{\text{node}} - z_{\text{ref}}}{L_z \text{pml}} \right]^n \sigma_{z_{\text{max}}},$$

where $\sigma_x$, $\sigma_y$ and $\sigma_z$ are the damping factors in the x, y and z directions respectively. In Eq. (3.30) $x_{\text{ref}}$ gives the reference coordinate where the PML layer begins and $L_x \text{pml}$ is the length of the PML in the x direction. $\sigma_{x_{\text{max}}}$ is the maximum value of the damping factor $\sigma$ desired in the x direction and $n$ is a value that indicates how $\sigma_x$ varies in the x direction. Equations (3.31) and (3.31) shows similar variation for $\sigma_y$ and $\sigma_z$ in the y and z direction respectively. Figure (3.2) illustrates the linear, quadratic and quartic variation of $\sigma_x$ along the x direction for a PML of thickness 0.5. Values between 50 to 100 for $\sigma_{x_{\text{max}}}$, $\sigma_{y_{\text{max}}}$ and $\sigma_{z_{\text{max}}}$ were found to work well for most cases. Shown in Fig. (3.3) is the progression of the damping factor $\sigma$ in the PML layers, where each component $\sigma_x$, $\sigma_y$ and $\sigma_z$ is set to a value of 75. Each one of the components of $\sigma$ is varied using a power law of order 4.
Figure 3.2 Variation of $\sigma_x$ along the x direction for linear, quadratic and quartic variation
Figure 3.3 PML layers in the x,y,z directions for layer thickness = 0.5, $\sigma = 75$ in all directions
CHAPTER 4
NEAR-FIELD TO FAR-FIELD TRANSFORMATION

4.1 Introduction

Most problems in electromagnetic theory can be broadly categorized as either open boundary problems or closed boundary problems. In general problems that involve a guided wave are closed boundary problems while classical radiation and scattering problems are considered to be open boundary problems. Open boundary problems that involve far-field calculations are fundamentally limited by the need to have a large computational region to observe and measure the effects of the entity that is being studied. However it is possible to use near-field data to obtain a high-fidelity estimate of the far-field solution using a systematic near-field to far-field transformation theorem based on the Huygen’s surface equivalence theorem [12, 3]. This allows the extent of the domain to be significantly reduced, which directly translates to savings in run-times and memory resources.

4.2 Radar cross sections

One of the most common uses of scattering equations is for the estimation of Radar Cross Sections (RCS) or echo area. This is defined as “the area intercepting the amount of power that when scattered isotropically produces at the receiver or observer the power density that is equal to
the density scattered by the target” [12]. For a two dimensional object this can be written as

\[ \sigma_{2D} = \lim_{r \to \infty} 2\pi r \frac{|P^s|}{|P^i|} \]  

(4.1)

This can also be written in two other forms based on the electric (E) and magnetic (H) field intensities.

\[ \sigma_{2D} = \lim_{r \to \infty} 2\pi r \frac{|E^s|}{|E^i|} \]  

(4.2)

\[ \sigma_{2D} = \lim_{r \to \infty} 2\pi r \frac{|H^s|}{|H^i|} \]  

(4.3)

In three dimensions a similar set of equations can be written as

\[ \sigma_{3D} = \lim_{r \to \infty} 4\pi r^2 \frac{|P^s|}{|P^i|} \]  

(4.4)

\[ \sigma_{3D} = \lim_{r \to \infty} 4\pi r^2 \frac{|E^s|}{|E^i|} \]  

(4.5)

\[ \sigma_{3D} = \lim_{r \to \infty} 4\pi r^2 \frac{|H^s|}{|H^i|} \]  

(4.6)

In Eqs. (4.1) and (4.4) \( P^s \) and \( P^i \) are the scattered and incident power densities while \( E^s, E^i, H^s, H^i \) are the corresponding electric and magnetic scattered and incident field intensities, respectively. It must be noted that the dimension of the two dimensional RCS is length while that of the three dimensional RCS is area. When the transmitter and the receiver are co-located the resulting RCS is called monostatic and when they differ in position it is referred to as bistatic RCS. Cross sections can be plotted at a fixed spatial position or at a specified frequency over a range of spatial coordinates. When it is plotted over spatial coordinates the resulting diagram is called an RCS pattern.
4.3 Scattering from plane waves

In the Total Field Scattered Field (TFSF) formulation, waves scattered from a 3D object can be written in terms of the incident and scattered field contributions

\[ E' = E^i + E^s \]  \hspace{1cm} (4.7)
\[ H' = H^i + H^s \]  \hspace{1cm} (4.8)

Here the incident fields \( E^i \) and \( H^i \) represent the fields that would exist in the absence of any scattering objects. Scattered fields \( E^s \) and \( H^s \) are reflections that occur when the incident wave interacts with an object in the computational region. The total field at any point in the computational region, given by \( E' \) and \( H' \), is the sum of these two components. It must be noted that in the PML region only scattered field components exist. The scattered and total fields will be computed for canonical objects using an analytical formulation and later will be generalized for arbitrary three dimensional objects using a Near-Field to Far-field transformation (NTFF). Scattering from a plane wave is often illustrated using a PEC sphere due to its symmetry, which enables it to be used as a reference scatterer for more complicated geometries. Following the derivations given in [12], consider a plane wave polarized in the positive \( x \) direction and travelling in the positive \( z \) direction given by

\[ E^i = \hat{a}_x E_0 e^{-j\beta z} \]  \hspace{1cm} (4.9)
where $E_0$ is the peak amplitude and $\beta$ is the magnitude of the wave vector of the sinusoidal variation. In spherical coordinates

$$\tilde{\mathbf{E}}^i = \hat{a}_r E^i_r + \hat{a}_\theta E^i_\theta + \hat{a}_\phi E^i_\phi$$

(4.10)

The spherical components of the incident wave can be written in terms of its Cartesian coordinates as

$$\tilde{E}^i_r = E^i_x \sin(\theta) \cos(\phi)$$

(4.11)

$$\tilde{E}^i_\theta = E^i_x \cos(\theta) \cos(\phi)$$

(4.12)

$$\tilde{E}^i_\phi = -E^i_x \sin(\phi)$$

(4.13)

Eq. (4.11), (4.12) and (4.13) can be expanded using Eq. (4.9) as

$$\tilde{E}^i_r = E^i_0 \sin(\theta) \cos(\phi)e^{-i\beta r \cos(\theta)}$$

(4.14)

$$\tilde{E}^i_\theta = E^i_0 \cos(\theta) \cos(\phi)e^{-i\beta r \cos(\theta)}$$

(4.15)

$$\tilde{E}^i_\phi = -E^i_0 \sin(\phi)e^{-i\beta r \cos(\theta)}$$

(4.16)

These spherical wave components can be rewritten using the wave transformations given in Eq. (4.17) below

$$E^+ = e^{-i\beta r \cos(\theta)} = \sum_{n=0}^{\infty} a_n j_n(\beta r) P_n \cos(\theta)$$

(4.17)
where $\tilde{E}_x^+$ indicates a wave propagating in the positive x direction. Here $j_n$ represents the spherical Bessel function of order n and $P_n$ is the Legendre polynomial of order n.

$$a_n = i^{-n}(2n + 1)$$  \hspace{1cm} (4.18)

Eq. (4.14), (4.15) and (4.16) can be rewritten using the transformation [12] above to give

$$\tilde{E}_r^i = E_0 \frac{cos(\phi)}{j_\beta r} \sum_{n=0}^{\infty} i^{-n}(2n + 1) j_n(\beta r) \frac{\partial}{\partial \theta} P_n \cos(\theta)$$  \hspace{1cm} (4.19)

$$\tilde{E}_\theta^i = E_0 \cos(\phi) \sum_{n=0}^{\infty} i^{-n}(2n + 1) j_n(\beta r) P_n \cos(\theta)$$  \hspace{1cm} (4.20)

$$\tilde{E}_\phi^i = -E_0 \sin(\phi) \sum_{n=0}^{\infty} i^{-n}(2n + 1) j_n(\beta r) P_n \cos(\theta)$$  \hspace{1cm} (4.21)

A few key relationships can be used to greatly simplify the above equations [12]. These are shown in Eq. (4.22 - 4.24) below.

$$j_n(\beta r) = \frac{1}{\beta r} \hat{J}_n(\beta r)$$  \hspace{1cm} (4.22)

$$\frac{\partial P_n}{\partial \theta} = P_n^1(\cos \theta)$$  \hspace{1cm} (4.23)

$$P_0^1 = 0$$  \hspace{1cm} (4.24)

Here, $\hat{J}_n$ is a modified Bessel function that is related to the spherical Bessel function $j_n$ by Eq. (4.22). The incident and scattered fields from a sphere of radius $r_{sph}$ can be expressed as a superposition of $TE'$ and $TM'$ waves [12] to give the following equations for the scattered waves.
\[ \vec{E}_i = \frac{1}{i\mu\omega} \left( \frac{\partial^2}{\partial r^2} + \beta^2 \right) \vec{A}_i^r \]  
\[ \vec{E}_\theta = \frac{1}{i\mu\omega} \frac{\partial^2 \vec{A}_i^r}{\partial r \partial \theta} - \frac{1}{er \sin(\theta)} \frac{\partial \vec{F}_i^\theta}{\partial \phi} \]  
\[ \vec{E}_\phi = \frac{1}{i\mu\omega \sin(\theta)} \frac{\partial^2 \vec{A}_i^r}{\partial r \partial \phi} + \frac{1}{er} \frac{\partial \vec{F}_i^\phi}{\partial \theta} \]  
\[ \vec{H}_i^r = \frac{1}{i\mu\omega} \left( \frac{\partial^2}{\partial r^2} + \beta^2 \right) \vec{F}_i^r \]  
\[ \vec{H}_\theta = \frac{1}{i\mu\omega} \frac{\partial^2 \vec{F}_i^r}{\partial r \partial \theta} + \frac{1}{\mu r \sin(\theta)} \frac{\partial \vec{A}_i^r}{\partial \phi} \]  
\[ \vec{H}_\phi = \frac{1}{i\mu\omega \sin(\theta)} \frac{\partial^2 \vec{F}_i^r}{\partial r \partial \phi} - \frac{1}{\mu r} \frac{\partial \vec{A}_i^r}{\partial \theta} \]  

where \( \vec{A}_i^r \) and \( \vec{F}_i^r \) are given by

\[ \vec{A}_i^r = E_0 \frac{\cos \phi}{\omega} \sum_{n=0}^{\infty} a_n \hat{J}_n(\beta r)P_n^1 \cos(\theta) \]  
\[ \vec{F}_i^r = E_0 \frac{\sin \phi}{\omega \eta} \sum_{n=0}^{\infty} a_n \hat{J}_n(\beta r)P_n^1 \cos(\theta) \]

and the scattered wave fields can be given by replacing \( A_i^r \) and \( F_i^r \) by \( \vec{A}_s^r \) and \( \vec{F}_s^r \) where these are given by

\[ \vec{A}_s^r = E_0 \frac{\cos \phi}{\omega} \sum_{n=0}^{\infty} b_n \hat{H}_n^2(\beta r)P_n^1 \cos(\theta) \]  
\[ \vec{F}_s^r = E_0 \frac{\sin \phi}{\omega \eta} \sum_{n=0}^{\infty} c_n \hat{H}_n^2(\beta r)P_n^1 \cos(\theta) \]
In the above equation $\hat{H}_n^2$ is a modified Hankel function [12] of the second kind. In all the above equations the coefficients $a_n$, $b_n$ and $c_n$ are given by

\[
a_n = i^{-n} \frac{2n + 1}{n(n + 1)} \quad (4.35)
\]

\[
b_n = -a_n \hat{J}_n(\beta r_{sph}) / \hat{H}_n^2(\beta r_{sph}) \quad (4.36)
\]

\[
c_n = -a_n \hat{J}_n(\beta r_{sph}) / \hat{H}_n^2(\beta r_{sph}) \quad (4.37)
\]

The primed variables in the Bessel functions indicate derivatives with respect to its coefficients. In the far field, which is assumed as being at $\beta r \rightarrow \infty$, $E_i^s = 0$ and hence the scattered fields, are the sum of the $\theta$ and $\phi$ components. The bistatic RCS can be written using the scattered fields at $\beta r \rightarrow \infty$

\[
RCS_{bistatic} = \lim_{r \to \infty} \frac{4\pi r^2 |\hat{E}_{s}\|^2}{|\hat{E}_i|^2} = \frac{\lambda^2}{\pi} \left( \cos^2 \phi |A_{\theta}|^2 + \sin^2 \phi |A_{\phi}|^2 \right) \quad (4.38)
\]

where

\[
|\tilde{A}_\theta|^2 = \left| \sum_{n=1}^{\infty} j^n \left[ b_n \sin(\theta) P_{n}^{1} \cos(\theta) - c_n \frac{P_{n}^{1} \cos(\theta)}{\sin(\theta)} \right] \right|^2 \quad (4.39)
\]

\[
|\tilde{A}_\phi|^2 = \left| \sum_{n=1}^{\infty} j^n \left[ -c_n \sin(\theta) P_{n}^{1} \cos(\theta) + b_n \frac{P_{n}^{1} \cos(\theta)}{\sin(\theta)} \right] \right|^2 \quad (4.40)
\]

Eq. (4.38) can be used to compute an analytical bistatic RCS and validate the numerical solutions later obtained.
4.4 Surface equivalence theorem

The surface equivalence theorem is based on the principle that the electric and magnetic fields tangential to a closed contour that surrounds a radiating structure can be used to determine the electric and magnetic far fields. This is based on the principle in electromagnetic theory [26] which states that actual sources such as antennas or other radiating sources can be replaced by an equivalent source on a contour surrounding the radiator. According to [26] this is done by the application of suitable electric and magnetic current densities that are appropriately chosen to satisfy the boundary conditions. Also applying these current densities over the selected contour renders the fields inside zero. The fields outside are equivalent to that produced by the original source. This transformation is exact, however it requires integration over the entire closed contour.

Fig. (4.1) below illustrates the case of an electromagnetic wave impinging on an arbitrary structure. In Fig. (4.1a) it is assumed that the fields given by $E_1$ and $H_1$ filling the entire region is due to the physical electric and magnetic current sources $J$ and $M$ that exists on the surface of the structure. In Fig. (4.1b) it is seen that $J$ and $M$ are removed and are replaced with new electric fields $E_1$ and $H_1$ inside the closed contour $S$. However, the intent is to compute the field $E_1$ and $H_1$ outside $S$. Now for these fields to satisfy the electromagnetic boundary conditions on the closed contour there must exist non-physical currents that flow tangentially along the closed contour $S$.

\[
\begin{align*}
J_s &= \hat{n} \times (H_1 - H) \\
M_s &= -\hat{n} \times (E_1 - E)
\end{align*}
\]  

(4.41) 

(4.42)

where $\hat{n}$ is the unit outward normal vector to $S$. The virtual electric and magnetic currents given in Eq. (4.41) are responsible for the original fields both within and in the extended region outside $S$. Since the quantities of interest are the field variables outside of $S$ or at far-fields, the fields within $S$
Figure 4.1 Closed contour for the surface equivalence theorem
can be assumed to be zero. As such the original problem reduces to one with the following current
densities on the closed contour, as shown in Fig. (4.1c).

\[
\begin{align*}
\mathbf{J}_s &= \hat{n} \times (\mathbf{H}_1) \quad (4.43) \\
\mathbf{M}_s &= -\hat{n} \times (\mathbf{E}_1) \quad (4.44)
\end{align*}
\]

4.5 3D Near-Field to Far-field transformation (NTFF)

Based on the surface equivalence theorem, the 3D NTFF transformation can be derived for
an arbitrary 3D object surrounded by a closed three dimensional contour [3]. One of the easiest
structures to build for this purpose is a cuboid. The phasor electric and magnetic currents are
computed by Fourier transforming the \( \mathbf{E} \) and \( \mathbf{H} \) fields over the surface of this cuboid. The
equivalent currents thus obtained are then integrated with the free-space Green’s function to obtain
the desired far-field values. For this purpose vector potentials can be defined as

\[
\begin{align*}
\tilde{\mathbf{A}} &= \frac{\mu_0}{4\pi} \int_S \mathbf{J}_s \frac{e^{-ikr}}{R} ds' \approx \mu_0 \frac{e^{-ikr}}{4\pi r} \tilde{\mathbf{N}} \quad (4.45) \\
\tilde{\mathbf{F}} &= \frac{\epsilon_0}{4\pi} \int_S \mathbf{M}_s \frac{e^{-ikr}}{R} ds' \approx \epsilon_0 \frac{e^{-ikr}}{4\pi r} \tilde{\mathbf{L}} \quad (4.46)
\end{align*}
\]

where \( \tilde{\mathbf{N}} \) and \( \tilde{\mathbf{L}} \) are given by

\[
\begin{align*}
\tilde{\mathbf{N}} &= \int_S \mathbf{J}_s e^{-ikr'\cos \phi} ds' \quad (4.47) \\
\tilde{\mathbf{L}} &= \int_S \mathbf{M}_s e^{-ikr'\cos \phi} ds' \quad (4.48)
\end{align*}
\]
where \( r \) is an observation point and \( r' \) is the location of the source. \( \psi \) is the angle between these two points. \( R \) is defined as

\[
R = r - r' \tag{4.49}
\]

The phasor fields \( \mathbf{E} \) and \( \mathbf{H} \) that can be derived from Eq. (4.45) can be given by

\[
\mathbf{E} = -i\omega(\mathbf{A} + \frac{1}{k^2} \nabla(\nabla \cdot \mathbf{A})) - \frac{1}{\varepsilon_0} \nabla \times \mathbf{F} \tag{4.50}
\]

\[
\mathbf{H} = -i\omega(\mathbf{F} + \frac{1}{k^2} \nabla(\nabla \cdot \mathbf{F})) - \frac{1}{\varepsilon_0} \nabla \times \mathbf{A} \tag{4.51}
\]

The above equations can be expanded and written in terms of the spherical components. Ignoring terms that decay at the rate of \( \frac{1}{r^2} \) and radial components that are negligible relative to the \( \theta \) and \( \phi \) components results in the following equations

\[
\tilde{E}_r = 0 \tag{4.52}
\]

\[
\tilde{E}_\theta = -i\omega(\tilde{A}_\theta + \eta_0 \tilde{F}_\phi) = -ik \frac{e^{-ikr}}{4\pi r} (\tilde{L}_\phi + \eta_0 \tilde{N}_\theta) \tag{4.53}
\]

\[
\tilde{E}_\phi = -i\omega(\tilde{A}_\phi - \eta_0 \tilde{F}_\theta) = ik \frac{e^{-ikr}}{4\pi r} (\tilde{N}_\phi - \eta_0 \tilde{N}_\theta) \tag{4.54}
\]

Similar equations can be written for the \( H \) terms

\[
\tilde{H}_r = 0 \tag{4.55}
\]

\[
\tilde{H}_\theta = -i\omega(\tilde{A}_\phi - \eta_0 \tilde{F}_\phi)/\eta_0 = +ik \frac{e^{-ikr}}{4\pi r} (\tilde{N}_\phi - \eta_0 \tilde{N}_\theta/\eta_0) \tag{4.56}
\]

\[
\tilde{H}_\phi = -i\omega(\tilde{A}_\theta + \eta_0 \tilde{F}_\phi)/\eta_0 = -ik \frac{e^{-ikr}}{4\pi r} (\tilde{N}_\phi + \eta_0 \tilde{N}_\theta/\eta_0) \tag{4.57}
\]
where $\eta_0$ is the intrinsic impedance of free space. In the above equations the terms $\vec{N}$ and $\vec{L}$ are given by the following in Cartesian coordinates

\[
\vec{N} = \iiint_S \left( \hat{x}J_x + \hat{y}J_y + \hat{z}J_z \right) e^{+ikr'\cos\psi} ds' \quad (4.58)
\]

\[
\vec{L} = \iiint_S \left( \hat{x}M_x + \hat{y}M_y + \hat{z}M_z \right) e^{+ikr'\cos\psi} ds' \quad (4.59)
\]

The $\theta$ and $\phi$ components of $\vec{N}$ and $\vec{L}$ can be written as

\[
\vec{N}_\theta = \iiint_S \left( J_x\cos(\theta)\cos(\phi) + J_y\cos(\theta)\sin(\phi) - J_z\sin(\theta) \right) e^{+ikr'\cos\psi} ds' \quad (4.60)
\]

\[
\vec{N}_\phi = \iiint_S \left( -J_x\sin(\phi) + J_y\cos(\phi) \right) e^{+ikr'\cos\psi} ds' \quad (4.61)
\]

\[
\vec{L}_\theta = \iiint_S \left( M_x\cos(\theta)\cos(\phi) + M_y\cos(\theta)\sin(\phi) - M_z\sin(\theta) \right) e^{+ikr'\cos\psi} ds' \quad (4.62)
\]

\[
\vec{L}_\phi = \iiint_S \left( M_x\sin(\phi) + M_y\cos(\phi) \right) e^{+ikr'\cos\psi} ds' \quad (4.63)
\]

It is necessary in many cases to determine the radiated power in the far-field region. This has significance in classical problems such as determining antenna gain and RCS. The time-averaged Poynting vector $\mathbf{P}_{\text{avg}}$, which represents the average power transmitted, can be used for this purpose

\[
\mathbf{P}_{\text{avg}} = \frac{1}{2}(\mathbf{E} \times \mathbf{H}) \quad (4.64)
\]

Based on Parseval’s theorem [3], which states that the power computed in the frequency domain equals the power computed in the time domain, the scattered power can be written in spherical
coordinates as

\[ \tilde{P}_{\text{scat}} = \frac{1}{2}(\tilde{E}_\theta \tilde{H}^*_\phi) - \frac{1}{2}(\tilde{E}_\phi \tilde{H}^*_\theta) \] (4.65)

Using Eq. (4.52), (4.55), (4.60) and (4.64) the above equation can be rewritten as

\[ \tilde{P}_{\text{scat}} = \frac{k^2}{32 \pi^2 \eta_0 r^2} (|\tilde{L}_\phi + \eta_0 \tilde{N}_\theta|^2 + |\tilde{L}_\theta - \eta_0 \tilde{N}_\phi|^2) \] (4.66)

The bistatic RCS can therefore be computed as

\[ RCS = \frac{4 \pi r^2 |P_{\text{scat}}|}{|P_{\text{inc}}|} = \frac{k^2}{8 \pi \eta_0 P_{\text{inc}}} (|\tilde{L}_\phi + \eta_0 \tilde{N}_\theta|^2 + |\tilde{L}_\theta - \eta_0 \tilde{N}_\phi|^2) \] (4.67)

where \( P_{\text{inc}} \) is the power of the incident wave.
CHAPTER 5
DISPERSE MATERIALS AND APPROXIMATE STABILIZATION MATRIX

5.1 Introduction to dispersive materials

For many problems it is assumed that the material properties, such as permittivity ($\epsilon$) and permeability ($\mu$) are independent of frequency. However, such an approximation is no longer valid in many broadband applications where these parameters vary significantly over the frequency range of interest. In addition, the assumption of fixed material properties is in direct violation of the Kramers-Kronig relations [27], which enforce causality.

A scalar fixed value for permittivity and permeability implies a uniform and instantaneous response to a an applied field [27]. In this case the constitutive relationship between the electric field and the electric flux density is given by

$$D = \epsilon E$$ (5.1)

where $\epsilon$ is independent of frequency. To fully understand and model physical systems one must take into account the anisotropic, nonlinear and dispersive character of the material under investigation. In such a scenario the material properties are represented as second order tensors [27, 28]. Existence of off-diagonal terms indicate the dependence of the electric field on orthogonal flux density components. When $\epsilon$ depends on the electric field the material is said to be nonlinear. When the permittivity tensor is a function of spatial orientation the material is referred to as being anisotropic. Dispersive materials can be classified as those where the material properties are a
function of frequency. Frequency independence in a dielectric implies an instantaneous response to an external applied field [27]. In the frequency domain the constitutive equation can be written as

$$D(\omega) = \epsilon(\omega)E(\omega) \quad (5.2)$$

### 5.2 Constitutive equations for dispersive materials

The electric and magnetic flux density are related to the electric and magnetic fields through the following constitutive equations

$$D = \epsilon_0 E + P \quad (5.3)$$
$$H = \frac{B}{\mu_0} - M \quad (5.4)$$

where $P$ and $M$ represent the electric and magnetic dipole moments respectively. $E$ and $H$ are the electric and magnetic field intensities. $D$ and $B$ are electric and magnetic flux densities. Dielectric materials have bound charges, which in the presence of an external electric field, gets deformed from its equilibrium position. This results in a net dipole moment in the material which is represented by $P$. It must be noted that for a dielectric, in the absence of an electric field, there exists dipoles which are in equilibrium throughout the material. Hence, in such a case the net polarization $P$ is equal to zero. In the frequency domain the two polarization vectors can be related to the fields as

$$\tilde{P}(\omega) = \epsilon_0 \tilde{\chi}_e(\omega) \tilde{E}(\omega) \quad (5.5)$$
$$\tilde{M}(\omega) = \tilde{\chi}_m(\omega) \tilde{H}(\omega) \quad (5.6)$$
Here $\chi_e$ and $\chi_m$ are the electric and magnetic susceptibilities. In this work only materials with electric susceptibilities will be investigated. The equation for frequency-dependent permittivity can thus be written as

$$\tilde{\epsilon}(\omega) = \epsilon_0 \tilde{\epsilon}_r(\omega) = \epsilon_0 (\epsilon_\infty + \tilde{\chi}_e(\omega)) \quad (5.7)$$

In the above equation $\epsilon_\infty$ accounts for the permittivity at high frequency when the frequency-dependent susceptibility $\tilde{\chi}_e(\omega)$ goes to zero. It is also noteworthy that the relative permittivity $\tilde{\epsilon}(\omega)$ can be split into a real and imaginary part

$$\tilde{\epsilon}(\omega) = \tilde{\epsilon}'(\omega) + i\tilde{\epsilon}''(\omega) \quad (5.8)$$

The dielectric constant, also known as relative permittivity, is the real part of this equation divided by the permittivity of free space and the imaginary part contributes to the frequency-dependent conductivity. These relationships can be expressed by the following equations

$$\tilde{\epsilon}_r(\omega) = \frac{\tilde{\epsilon}'(\omega)}{\epsilon_0} \quad (5.9)$$
$$\tilde{\sigma}(\omega) = \omega \tilde{\epsilon}''(\omega) \quad (5.10)$$

5.3 Material models

A diagonal anisotropic susceptibility matrix is assumed for the material models, where for the frequency-dependent materials the susceptibility matrix is a function of frequency. Two frequency-dependent models and one frequency-independent anisotropic material models will be discussed
below. All of them take the general form of

\[
\tilde{P} = [\tilde{\chi}] \tilde{E}
\]  

(5.11)

or written in matrix form

\[
\begin{bmatrix}
\tilde{P}_x \\
\tilde{P}_y \\
\tilde{P}_z
\end{bmatrix} =
\begin{bmatrix}
\chi_x & 0 & 0 \\
0 & \chi_y & 0 \\
0 & 0 & \chi_z
\end{bmatrix}
\begin{bmatrix}
\tilde{E}_x \\
\tilde{E}_y \\
\tilde{E}_z
\end{bmatrix}
\]  

(5.12)

where \( \chi_j = f(\omega) \), where \( j \in \{x, y, z\} \) and \( f(\omega) \) is a function of frequency, defined by each model given below.

5.3.1 Debye material model

A Debye material model is similar to an RC circuit where the polarization in the material is analogous to the voltage that builds up across the capacitor [27]. At a constant source the capacitor has a voltage built up across it, however as the frequency is increased this voltage gradually decreases as the impedance of the capacitor decreases. As the frequency approaches infinity the capacitor is short circuited and the voltage across it drops to zero. This is similar to the polarization induced in the dielectric modeled by a Debye model, where at frequencies that approach infinity the polarization tends to zero, leaving behind only the residual high frequency term. The frequency-dependent relative permittivity for a Debye material model can be fundamentally written as

\[
\tilde{\varepsilon}_r(\omega) = \varepsilon_\infty + \frac{\varepsilon_s - \varepsilon_\infty}{1 + i\omega\tau_d} + \sigma_s
\]  

(5.13)

Here \( \varepsilon_\infty \) is the residual high frequency permittivity as \( \omega \to \infty \) and \( \varepsilon_s \) is the static permittivity or the permittivity at zero or DC frequency. \( \tau_d \) is a parameter that has the dimension of time and is
related to the resonant frequency of the dielectric. $\tau_d$ is not to be confused with the stabilization matrix $[\tau]$ used in the Streamline Upwind/Petrov Galerkin method. $\sigma_s$ is the static conductivity or the conductivity at DC frequency. It is common to write $\epsilon_s - \epsilon_\infty$ as $\Delta \epsilon$. Also, in literature, it is often the case that the $\sigma_s$ parameter is absorbed into the term $\frac{\epsilon_s - \epsilon_\infty}{1 + i\omega \tau}$. For a multiple pole formulation Eq. (5.13) can be rewritten as

$$\tilde{\epsilon}_r(\omega) = \epsilon_\infty + \sum_{i=1}^{n} \frac{\Delta \epsilon_i}{1 + i\omega \tau_d} + \sum_{i=1}^{n} \sigma_s$$  \hspace{1cm} (5.14)

Now referring back to the constitutive equation connecting the electric flux density and the electric field

$$\tilde{\mathbf{D}} = \epsilon_0 \epsilon_r \tilde{\mathbf{E}}$$  \hspace{1cm} (5.15)

$$\Rightarrow \tilde{\mathbf{D}} = \epsilon_0 \epsilon_\infty \tilde{\mathbf{E}} + \epsilon_0 \tilde{\chi} \tilde{\mathbf{E}}$$  \hspace{1cm} (5.16)

$$\Rightarrow \tilde{\mathbf{D}} = \epsilon_0 \epsilon_\infty \tilde{\mathbf{E}} + \tilde{\mathbf{P}}$$  \hspace{1cm} (5.17)

where $\tilde{\chi}$ is the susceptibility of the material. For a material model with diagonal anisotropy this is simply

$$\tilde{\chi} = \begin{bmatrix}
\frac{\Delta \epsilon_x}{1 + i\omega \tau_{d_x}} & 0 & 0 \\
0 & \frac{\Delta \epsilon_y}{1 + i\omega \tau_{d_y}} & 0 \\
0 & 0 & \frac{\Delta \epsilon_z}{1 + i\omega \tau_{d_z}}
\end{bmatrix}$$  \hspace{1cm} (5.18)

Each component of $\tilde{P}_j$ can be written as

$$\tilde{P}_j = \epsilon_0 \frac{\Delta \epsilon_j}{1 + i\omega \tau_{d_j}} \tilde{E}_j$$  \hspace{1cm} (5.19)
where $\tilde{P}_j \in \{\tilde{P}_x, \tilde{P}_y, \tilde{P}_z\}$. Eq. (5.19) can be rewritten as

$$\left(1 + i\omega \tau_{d_j}\right) \tilde{P}_j - \varepsilon_0 \Delta \varepsilon_j \tilde{E}_j = 0$$

(5.20)

Converting this into the time-domain form results in the following equation

$$\frac{\partial P_j}{\partial t} + \frac{(P_j - \varepsilon_0 \Delta \varepsilon_j E_j)}{\tau_{d_j}} = 0$$

(5.21)

### 5.3.2 Lorentz material model

In contrast with the Debye model the Lorentz material model is based on a second order mechanical model that is used to model the motion of charges [27]. This is analogous to the motion of a spring with an applied force. This can be written as a balance equation of the form

$$M \frac{\partial^2 x}{\partial t^2} = Q E(t) - Mg \frac{\partial x}{\partial t} - Mkx$$

(5.22)

In the above equation $M$ is the mass of the charge, $Q$ is the amount of charge, $g$ is the damping coefficient, $E$ is the applied electric field and $x$ is the displacement of the charge. The right side of the equation consists of a driving force given by $Q E(t)$, a damping force given by $Mg \frac{\partial x}{\partial t}$ and a restoring force is given by $Mkx$. If this equation is now converted to the frequency domain

$$-M\omega^2 \hat{x}(\omega) + iMg\omega \hat{x}(\omega) + Mk\hat{x}(\omega) = Q\hat{E}(\omega)$$

(5.23)

$$\hat{x}(\omega) = \frac{Q}{M(K + ig\omega - \omega^2)} \hat{E}(\omega)$$

(5.24)
Polarization is directly proportional to $Q\mathbf{x}$, hence it can be written as

$$\tilde{P} = NQ \tilde{\mathbf{x}} \quad (5.25)$$

where $N$ is a constant. Using Eq. (5.24) and Eq. (5.25) the polarization vector can be written as

$$\tilde{P}(\omega) = \epsilon_0 \frac{NQ^2}{M\epsilon_0 (K + ig\omega - \omega^2)} \tilde{E}(\omega) \quad (5.26)$$

Polarization is related to the electric field using

$$\tilde{P} = \epsilon_0 \tilde{\chi} \tilde{E} \quad (5.27)$$

Using Eq. (5.27) susceptibility can be written as

$$\tilde{\chi}(\omega) = \frac{NQ^2}{M\epsilon_0 (K + ig\omega - \omega^2)} \quad (5.28)$$

Similar to the Debye model a frequency-dependent susceptibility matrix $\chi$ [27] can be written from Eq. (5.28) as

$$\tilde{\chi} = \begin{bmatrix}
\Delta\epsilon_x \omega_x^2 \\
\omega_x^2 + 2i\omega\delta_x - \omega^2
\end{bmatrix}
\begin{bmatrix}
0 & 0 \\
0 & \Delta\epsilon_y \omega_y^2
\end{bmatrix}
\begin{bmatrix}
0 & 0 \\
\omega_y^2 + 2i\omega\delta_y - \omega^2
\end{bmatrix}
\begin{bmatrix}
\Delta\epsilon_z \omega_z^2 \\
\omega_z^2 + 2i\omega\delta_z - \omega^2
\end{bmatrix} \quad (5.29)$$

The general form of the $\tilde{P}_j$ component is

$$\tilde{P}_j = \epsilon_0 \frac{\Delta\epsilon_j \omega_j^2}{\omega_j^2 + 2i\omega\delta_j - \omega^2} \tilde{E}_j \quad (5.30)$$
Converting this to time domain

\[
\frac{\partial P_j}{\partial t} + \frac{1}{2\delta_j} \frac{\partial^2 P_j}{\partial t^2} + \frac{\omega_j^2}{2\delta_j} P_j - \frac{\epsilon_0 \Delta \epsilon_j \omega_j^2 E_j}{2\delta_j} = 0 \tag{5.31}
\]

where \( P_j \in \{P_x, P_y, P_z\} \)

5.3.3 Anisotropic frequency-independent dielectric

This model is used for dielectrics that are modeled as being independent of frequency. This is useful for narrowband responses when the susceptibility and therefore the permittivity can be considered to be a constant without loss of accuracy. The susceptibility matrix can be written as

\[
\tilde{\chi} = \begin{bmatrix}
\chi_{xx} & 0 & 0 \\
0 & \chi_{yy} & 0 \\
0 & 0 & \chi_{zz}
\end{bmatrix} \tag{5.32}
\]

In the time domain this can be written as

\[
P_x = \epsilon_0 (\chi_{xx} E_x) \tag{5.33}
\]
\[
P_y = \epsilon_0 (\chi_{yy} E_y) \tag{5.34}
\]
\[
P_z = \epsilon_0 (\chi_{zz} E_z) \tag{5.35}
\]

These equations can be implemented in code as

\[
\frac{\partial P_j}{\partial t} - \epsilon_0 \chi_j \frac{\partial E_j}{\partial t} = 0 \tag{5.36}
\]

where \( P_j \in \{P_x, P_y, P_z\} \)
5.4 Auxiliary Differential Equation (ADE) approach to the dispersive equations

There are two ADE approaches that can be taken for incorporating the dispersive media equations into the Maxwell’s system of equations [15]. Consider the curl equations in the frequency domain

\[ \nabla \times \tilde{E} = -i \omega \tilde{B} \] (5.37)
\[ \nabla \times \tilde{H} = i \omega \tilde{D} \] (5.38)

Also the constitutive equation for the electric flux density can be written as

\[ \tilde{D} = \varepsilon_0 \varepsilon_\infty \tilde{E} + \varepsilon_0 \tilde{\chi} \tilde{E} \] (5.39)

The ADE can be illustrated by considering the susceptibility function for a Debye material model as shown below

\[ \tilde{\chi} = \frac{\Delta \varepsilon}{1 + i \omega \tau_d} \] (5.40)

\( \tilde{D} \) can be rewritten using Eq. (5.39) and Eq. (5.38), which gives the following

\[ \nabla \times \tilde{H} = i \omega \left( \varepsilon_0 \varepsilon_\infty \tilde{E} + \varepsilon_0 \tilde{\chi} \tilde{E} \right) \] (5.41)
\[ \Rightarrow \nabla \times \tilde{H} = i \omega \varepsilon_0 \varepsilon_\infty \tilde{E} + \tilde{J} \] (5.42)

where \( \tilde{J} \) is given by

\[ \tilde{J} = i \omega \varepsilon_0 \frac{\Delta \varepsilon}{1 + i \omega \tau_d} \tilde{E} \] (5.43)
Eq. (5.42) and Eq. (5.43) can be converted to time domain to obtain the following

\[ \nabla \times \mathbf{H} = \epsilon_0 \epsilon_\infty \frac{\partial \mathbf{E}}{\partial t} + \mathbf{J} \]  
(5.44)

\[ \mathbf{J} + \tau \frac{\partial \mathbf{J}}{\partial t} = \epsilon_0 \Delta \epsilon \frac{\partial \mathbf{E}}{\partial t} \]  
(5.45)

If the model is represented by more than one pole then Eq. (5.41) and Eq. (5.42) can be expanded to get

\[ \nabla \times \tilde{\mathbf{H}} = i \omega \left( \epsilon_0 \epsilon_\infty \tilde{\mathbf{E}} + \epsilon_0 \sum_{p=1}^{n} \tilde{\chi}_p \tilde{\mathbf{E}} \right) \]  
(5.46)

\[ \Rightarrow \nabla \times \tilde{\mathbf{H}} = i \omega \epsilon_0 \epsilon_\infty \tilde{\mathbf{E}} + \sum_{p=1}^{n} \tilde{\mathbf{J}} \]  
(5.47)

These frequency domain equations can converted to the time domain to get

\[ \nabla \times \mathbf{H} = \epsilon_0 \epsilon_\infty \frac{\partial \mathbf{E}}{\partial t} + \sum_{p=1}^{n} \mathbf{J}_p \]  
(5.48)

and for each pole an equation similar to Eq. (5.45) can be written

\[ \mathbf{J}_p + \tau \frac{\partial \mathbf{J}_p}{\partial t} = \epsilon_0 \Delta \epsilon_p \frac{\partial \mathbf{E}}{\partial t} \]  
(5.49)

However, in this approach, the eigensystem is independent of the polarization terms since they appear as source terms in Eq. (5.48). Therefore, the effects of polarization are not accounted for in the stabilization matrix, which is derived from the eigensystem [15]. An alternative to this approach was recommended in [15] for Petrov-Galerkin schemes where \( \tilde{\mathbf{E}} \) in Faraday’s law, given in Eq. (5.37), is expanded as

\[ \tilde{\mathbf{E}} = \frac{\mathbf{D} - \mathbf{P}}{\epsilon_0 \epsilon_\infty} \]  
(5.50)
Substituting the above equation for $\tilde{E}$ into Eq. (5.37) and converting to time domain results in the following

$$\nabla \times \frac{(D - P)}{\epsilon_0 \epsilon_{\infty}} = -\frac{\partial B}{\partial t} \quad (5.51)$$

If $P$ has contributions from multiple poles then it can be written as a sum

$$P = \sum_{n=1}^{n_{poles}} P_n \quad (5.52)$$

where each $P_n$ can be written in frequency domain as

$$\tilde{P}_n = \epsilon_0 \frac{\Delta \epsilon_n}{1 + i\omega \tau_{d_n}} \tilde{E} \quad (5.53)$$

and in time domain these equations become the ADEs given by

$$P_n + \tau_{d_n} \frac{\partial P_n}{\partial t} = \epsilon_0 \Delta \epsilon_n E \quad (5.54)$$

5.5 Modified flux Jacobian for dispersive materials

To illustrate the structure of the flux Jacobian and eventually the eigendecomposition of the modified matrix $[\tilde{A}]$, a single pole anisotropic material model is used. This results in a total of 9 unknowns, due to the 3 polarization terms added, to be solved for the dielectric material. The full set of equations can be rewritten as shown below in Eq. (5.55) and Eq. (5.56).

$$\frac{\partial Q}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} + \frac{\partial h}{\partial z} + S = 0 \quad (5.55)$$
In Eq. (5.56) the terms $P_{sx}, P_{sy}, P_{sz}$ are source terms associated with $P_x, P_y, P_z$. These source terms depend on the model and the model parameters associated with it. For a single pole Debye model they can be written as

$$P_{sx} = (P_x - \Delta \epsilon_x E_x) / \tau_x$$  \hspace{1cm} (5.57)

$$P_{sy} = (P_y - \Delta \epsilon_y E_y) / \tau_y$$  \hspace{1cm} (5.58)

$$P_{sz} = (P_z - \Delta \epsilon_z E_z) / \tau_z$$  \hspace{1cm} (5.59)

Even though the spatial derivative terms look similar to a non-dispersive formulation it must be pointed out that the electric field intensities in the flux contributions for equations in the rows corresponding to $B_x, B_y$ and $B_z$ now take the form

$$E_x = \frac{D_x - P_x}{\epsilon_0 \epsilon_{xx}}$$  \hspace{1cm} (5.60)

$$E_y = \frac{D_y - P_y}{\epsilon_0 \epsilon_{yy}}$$  \hspace{1cm} (5.61)

$$E_z = \frac{D_z - P_z}{\epsilon_0 \epsilon_{zz}}$$  \hspace{1cm} (5.62)
The flux $\mathbf{F}$ can be written as

$$\mathbf{F} = \mathbf{f} \hat{i} + \mathbf{g} \hat{j} + \mathbf{h} \hat{k} \quad (5.63)$$

$\mathbf{F} \cdot \hat{n}$ can be computed as shown in Eq. (5.64) below, where $\hat{n}$ is the unit normal with components given by $k_x, k_y$ and $k_z$

$$\mathbf{F} \cdot \hat{n} = \begin{bmatrix}
-H_z k_y + H_y k_z \\
H_z k_x - H_x k_z \\
-H_y k_x + H_x k_y \\
E_z k_y - E_y k_z \\
-E_x k_x + E_y k_y \\
E_y k_z - E_x k_z \\
0 \\
0 \\
0
\end{bmatrix} \quad (5.64)$$

The flux Jacobian is defined as

$$[\ddot{A}] = \frac{\partial \mathbf{F} \cdot \hat{n}}{\partial \mathbf{Q}} \quad (5.65)$$

$[\ddot{A}]$ can be written using Eq. (5.64) as shown below
Note here that the abbreviations \( \epsilon_x, \epsilon_y, \epsilon_z \) are used to represent \( \epsilon_0 \epsilon_{\infty x}, \epsilon_0 \epsilon_{\infty y}, \epsilon_0 \epsilon_{\infty z} \) respectively.

Because the inverse of the stabilization matrix \([\tau]\) is computed as

\[
[\tau]^{-1} = \sum_{k=1}^{n} \left| \partial N_k / \partial x [A] + \partial N_k / \partial y [B] + \partial N_k / \partial z [C] \right| \]  

(5.67)

where \( n \) is the number of node points in the element, \([\bar{A}]_k\) can be written as

\[
[\bar{A}]_k = \frac{\partial N_k}{\partial x} [A]_k + \frac{\partial N_k}{\partial y} [B]_k + \frac{\partial N_k}{\partial z} [C]_k \]  

(5.68)

and \([\tau]^{-1}\) can be written as

\[
[\tau]^{-1} = \sum_{k=1}^{n} |[\bar{A}]_k| \]  

(5.69)
$|\tilde{A}|_k$ can be diagonalized and $|\tilde{A}|_k$ can be written as

$$|\tilde{A}|_k = [T]|\Lambda||T|^{-1}$$

(5.70)

In Eq. (5.70) above $|\Lambda|$ is a diagonal matrix of eigenvalues and $[T]$ and $[T]^{-1}$ are the matrices of right and left eigenvectors for $|\tilde{A}|_k$. The eigenvalues are given by

$$\text{diag} (|\Lambda|) = \begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
\sqrt{k_x^2 + k_y^2 + k_z^2} / \epsilon \mu \\
\sqrt{k_x^2 + k_y^2 + k_z^2} / \epsilon \mu \\
-\sqrt{k_x^2 + k_y^2 + k_z^2} / \epsilon \mu \\
-\sqrt{k_x^2 + k_y^2 + k_z^2} / \epsilon \mu
\end{bmatrix}$$

(5.71)

The eigenvectors can be given from the columns of $[T]$ as
\[
\begin{array}{cccccc}
\frac{k_x}{k_z} & 0 & 1 & 0 & -k_x & -\frac{e k_z}{\sqrt{\epsilon l \mu}} \\
\frac{k_y}{k_z} & 0 & 0 & 1 & -k_y & -\frac{e k_z}{(k_x^2 + k_y^2) \sqrt{\epsilon l \mu}} \\
1 & 0 & 0 & 0 & 0 & -\frac{k_x l \mu}{k_y} \\
0 & \frac{k_x}{k_z} & 0 & 0 & 0 & -\frac{k_y}{k_x} \\
0 & \frac{k_x}{k_z} & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

(5.72)

where \( l \) is given as

\[
l = k_x^2 + k_y^2 + k_z^2
\]

(5.73)

The matrix \( |\bar{A}| \) can be computed using Eq. (5.70), Eq. (5.71) and Eq. (5.72) as
Here \( \epsilon \) corresponds to \( \epsilon_\infty \) of the dispersive material. It is noted that \([T], ||[\Lambda]|\) and \([T]^{-1}\) are not numerically computed to evaluate \([\bar{A}]_k\) but rather \([\bar{A}]_k\) is evaluated from a symbolic matrix which is a product of \([T], ||[\Lambda]|\) and \([T]^{-1}\). However the above matrix is rank deficient and a singular value decomposition needs to be computed to obtain the pseudo-inverse. This is an expensive process and the computation cost (both run time and operation count) scales with the matrix size as \(O(n^3)\) [29] where \(n\) is the size of the matrix. It must be pointed out that each additional pole in an anisotropic dispersive model increases the block size by at least 3. As such, a fixed block size code based on the above stabilization matrix poses a problem as far as scalability is concerned.

It is interesting to note the structure of this matrix \([\bar{A}]_k\) when polarization terms are added. For every 'p' polarization equations that are added, the last p rows of this matrix are always zero. This directly translates to \([\bar{A}]_k\) having non zero values only in the first \((n - p)\) rows. Referring back to the residual equation given in Eq. (2.21), it can be seen that, since these are matrix multiplications, the size of the block has an impact on scalability. However, because the bottom p rows are zero, at least some of the inner loop indices do not increase with problem size in the residual routines.
5.6 Approximate diagonal \([\tau]\)

An exact computation of \([\tau]\) requires matrix inversion and in the case of dispersive or frequency dependent material models, where additional equations for the polarization terms need to be solved, a singular value decomposition (SVD) needs to be performed to compute the pseudoinverse of \([\tau]^{-1}\). This is necessitated due to the fact that the resulting matrix for \([\tau]^{-1}\) is rank deficient. Evaluation of SVD is a computationally expensive process, being almost five times as slow as a matrix inversion. It must also be pointed out that this cost increases [29] with the number of poles that are incorporated in the model since the size of the block that is subject to an SVD is equal to the number of equations to be solved for that model. Because the Jacobian-based stabilization matrix is expensive to implement, an approximate \([\tau]\) matrix inspired by the Lax Friedrich flux [30] is proposed. This formulation is designed with scalability and speed in mind for problems that are computationally demanding such as simulations of multipole dispersive models. The proposed form of the approximate diagonal matrix \([\tau]\) for a region in vacuum can be written using a modified matrix \(\tilde{A}_k\) as

\[
[\tau] = \left[ \sum_{k=1}^{\text{modes}} [\tilde{A}_k] \right]^{-1} \tag{5.74}
\]

where \(\tilde{A}_k\) now takes the form

\[
[\tilde{A}_k] = [T][\Lambda_{\text{diag}}][T^{-1}] \tag{5.75}
\]

In the above equation \(\Lambda_{\text{diag}}\) has the maximum eigenvalue from Eq. (5.71) on all diagonal terms. This ensures maximum damping for all wave components. \(\tilde{A}_k\) can be simplified to give
\[ [\tilde{A}]_k = \begin{bmatrix}
\sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 & 0 \\
0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 \\
0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 \\
0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 \\
0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 \\
0 & 0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}}
\end{bmatrix} \]

(5.76)

where

\[ L_k = \left( \frac{\partial N_k}{\partial x} \right)^2 + \left( \frac{\partial N_k}{\partial y} \right)^2 + \left( \frac{\partial N_k}{\partial z} \right)^2 \]

(5.77)

Here the term \( \sqrt{\frac{L_k}{\epsilon \mu}} \) is the maximum eigenvalue of the term \([T][\Lambda][T^{-1}]\), given in Eq. (5.71). For a dispersive material there are diagonal terms only on the first 6 rows, as shown below.
\[
[\bar{A}]_k = \begin{pmatrix}
\sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \sqrt{\frac{L_k}{\epsilon \mu}} \\
\end{pmatrix}
\]

(5.78)

Hence the inverse of the stabilization matrix is obtained by summing up the principal submatrix of \([\bar{A}]_k\) of size 6, over the nodes. The resultant diagonal matrix is then trivially inverted to obtain the 6x6 stabilization matrix \([\tau]\) for all material models.
CHAPTER 6
UNROLLED AND COUPLED SET OF PML-MAXWELL EQUATIONS

6.1 Introduction

Maxwell’s equations are fundamentally a linear set of equations with the exception of cases where the polarization term depends nonlinearly on the electric fields. Nonlinear behavior of Maxwell’s equations can be observed in photonic crystals [31], however in this present work it is assumed that the polarization terms vary linearly with the electric fields. An uncoupled or loosely coupled approach for the PML and Maxwell system of equations does not take advantage of the linearity of these systems of equations since multiple sub-iterations are needed to converge the solution for these systems to machine precision. However, coupling these systems together avoids this problem and only a single Newton iteration is required to converge the resulting coupled system.

6.2 Loosely coupled/ staggered PML-Maxwell equations

Decoupled systems are almost a necessity in multiphysics code for both steady state and transient simulations. Algorithms can take advantage of such decoupling by designing methods that are suited for each subsystem being solved [32]. Additionally this often places less demands on memory compared to the fully coupled approach [32]. From the perspective of code maintenance this approach is particularly attractive since this offers a certain modularity. Single physics codes have had significant time and effort invested in their development which make the case
for decoupled algorithms. Assuming the coupling between the subsystems is weak, this approach may end up being faster than the coupled algorithm [32]. However, for a linear system of equations the resulting sub-iterations can adversely affect execution speed. The Maxwell and PML equations can be written separately in vector form as

\[
\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} + \frac{\partial \mathbf{h}}{\partial z} + S_{\text{maxwell–pml}} = 0
\]

(6.1)

\[
\frac{\partial \mathbf{Q}_{\text{pml}}}{\partial t} + S_{\text{pml}} = 0
\]

(6.2)

where \( \mathbf{Q} \) and \( \mathbf{Q}_{\text{pml}} \) are vectors of unknowns for the Maxwell’s equations and PML equations respectively. \( \mathbf{f}, \mathbf{g} \) and \( \mathbf{h} \) are the flux vectors for Maxwell’s equations. \( S_{\text{pml}} \) is the source term associated with the PML equations. Note here that the term \( S_{\text{maxwell–pml}} \) represent the source terms that couple the PML and Maxwell equations. The residuals for the these two systems can be written as

\[
R(\mathbf{Q}^{n+1}) = \frac{\partial \mathbf{Q}^{n+1}}{\partial t} + \frac{\partial \mathbf{f}^{n+1}}{\partial x} + \frac{\partial \mathbf{g}^{n+1}}{\partial y} + \frac{\partial \mathbf{h}^{n+1}}{\partial z} + S_{\text{maxwell–pml}}^{n+1}
\]

(6.3)

\[
R(\mathbf{Q}_{\text{pml}}^{n+1}) = \frac{\partial \mathbf{Q}_{\text{pml}}^{n+1}}{\partial t} + S_{\text{pml}}^{n+1}
\]

(6.4)
For each node the linearization contribution per node can be written as

\[
\frac{\partial R(Q)}{\partial Q} = \begin{bmatrix}
\frac{\partial R(D_x)}{\partial D_x} & \frac{\partial R(D_x)}{\partial D_y} & \frac{\partial R(D_x)}{\partial D_z} & \frac{\partial R(D_x)}{\partial B_x} & \frac{\partial R(D_x)}{\partial B_y} & \frac{\partial R(D_x)}{\partial B_z} \\
\frac{\partial R(D_y)}{\partial D_x} & \frac{\partial R(D_y)}{\partial D_y} & \frac{\partial R(D_y)}{\partial D_z} & \frac{\partial R(D_y)}{\partial B_x} & \frac{\partial R(D_y)}{\partial B_y} & \frac{\partial R(D_y)}{\partial B_z} \\
\frac{\partial R(D_z)}{\partial D_x} & \frac{\partial R(D_z)}{\partial D_y} & \frac{\partial R(D_z)}{\partial D_z} & \frac{\partial R(D_z)}{\partial B_x} & \frac{\partial R(D_z)}{\partial B_y} & \frac{\partial R(D_z)}{\partial B_z} \\
\frac{\partial R(B_x)}{\partial D_x} & \frac{\partial R(B_x)}{\partial D_y} & \frac{\partial R(B_x)}{\partial D_z} & \frac{\partial R(B_x)}{\partial B_x} & \frac{\partial R(B_x)}{\partial B_y} & \frac{\partial R(B_x)}{\partial B_z} \\
\frac{\partial R(B_y)}{\partial D_x} & \frac{\partial R(B_y)}{\partial D_y} & \frac{\partial R(B_y)}{\partial D_z} & \frac{\partial R(B_y)}{\partial B_x} & \frac{\partial R(B_y)}{\partial B_y} & \frac{\partial R(B_y)}{\partial B_z} \\
\frac{\partial R(B_z)}{\partial D_x} & \frac{\partial R(B_z)}{\partial D_y} & \frac{\partial R(B_z)}{\partial D_z} & \frac{\partial R(B_z)}{\partial B_x} & \frac{\partial R(B_z)}{\partial B_y} & \frac{\partial R(B_z)}{\partial B_z}
\end{bmatrix}
\]

(6.5)
The PML system of equations can be written by

\[
\frac{\partial \mathbf{R}(\mathbf{Q})}{\partial \mathbf{Q}_{\text{pml}}} =
\begin{bmatrix}
\frac{\partial R(U_x)}{\partial U_x} & \frac{\partial R(U_x)}{\partial U_y} & \frac{\partial R(U_x)}{\partial U_z} & \frac{\partial R(U_x)}{\partial V_x} & \frac{\partial R(U_x)}{\partial V_y} & \frac{\partial R(U_x)}{\partial V_z} \\
\frac{\partial R(U_y)}{\partial U_x} & \frac{\partial R(U_y)}{\partial U_y} & \frac{\partial R(U_y)}{\partial U_z} & \frac{\partial R(U_y)}{\partial V_x} & \frac{\partial R(U_y)}{\partial V_y} & \frac{\partial R(U_y)}{\partial V_z} \\
\frac{\partial R(U_z)}{\partial U_x} & \frac{\partial R(U_z)}{\partial U_y} & \frac{\partial R(U_z)}{\partial U_z} & \frac{\partial R(U_z)}{\partial V_x} & \frac{\partial R(U_z)}{\partial V_y} & \frac{\partial R(U_z)}{\partial V_z} \\
\frac{\partial R(V_x)}{\partial U_x} & \frac{\partial R(V_x)}{\partial U_y} & \frac{\partial R(V_x)}{\partial U_z} & \frac{\partial R(V_x)}{\partial V_x} & \frac{\partial R(V_x)}{\partial V_y} & \frac{\partial R(V_x)}{\partial V_z} \\
\frac{\partial R(V_y)}{\partial U_x} & \frac{\partial R(V_y)}{\partial U_y} & \frac{\partial R(V_y)}{\partial U_z} & \frac{\partial R(V_y)}{\partial V_x} & \frac{\partial R(V_y)}{\partial V_y} & \frac{\partial R(V_y)}{\partial V_z} \\
\frac{\partial R(V_z)}{\partial U_x} & \frac{\partial R(V_z)}{\partial U_y} & \frac{\partial R(V_z)}{\partial U_z} & \frac{\partial R(V_z)}{\partial V_x} & \frac{\partial R(V_z)}{\partial V_y} & \frac{\partial R(V_z)}{\partial V_z}
\end{bmatrix}
\]

(6.6)

It can be seen that, in this case, the block sizes for both systems are 6 which can be solved separately in a staggered or loosely coupled fashion.

6.3 Tightly coupled/ staggered PML-Maxwell equations

The use of sub-iterations in the solution of Maxwell’s equations is necessitated due to the fact that PMLs are used to terminate the computational region, the equations for which are solved in a staggered manner. In a non-dispersive medium the linear nature of Maxwell’s equations suggests that the system of equations converge in a single step. A staggered solution procedure usually results in two or more sub-iterations for each time-step. Execution times have been found to speed
up by a factor of about 2.5 when the two systems are tightly coupled. The benefits are subject to
the size of the PML and the number of distinct regions in the computational mesh. The process
of deciding whether to solve a pair of system of equations in a staggered or coupled manner rests
ultimately on the nature of the problem at hand. In a tightly coupled system of equations the
Maxwell’s and PML equations are recast together in an implicit form. This permits a linearization,
with a block size of 12, to be written as
The above set of equations can be solved implicitly with one Newton iteration. A disadvantage, however, is that a much larger system of equations are solved at each iteration, thereby potentially requiring greater memory resources.
6.4 Unrolled system of equations

Two approaches are available to solve for a variable number of unknowns at each node. One method involves allocating a block size that equals the maximum number of unknowns at any point in the computational region. Looking at the case of scattering problems where a large portion of the computational region is comprised of free space, it is easy to see why this might be inefficient. If there exists a dispersive material that is modeled with multiple poles, a universal block size that equals the number of unknowns in the dispersive material gets allocated at every node in the mesh. The second approach is to eliminate the block abstraction as it pertains to solving a linear system. Doing so results in an 'unrolled' system of equations which avoids the overhead of excessive memory usage and additional computational costs that arise as a result of it. This is accomplished by assigning each region in the mesh a 'node_q' value, which is the number of equations to be solved for the nodes in that region. Therefore care must be taken in the mesh generation process to uniquely identify such nodes through volume tags. While it is obvious that additional data structures are necessary in the mesh management and linear algebra subroutines, the message passing routine also needs to be amended because the domain decomposition tool provides no information regarding the unrolled nodes. This data needs to be generated from the file that specifies the number of equations to be solved per region. The structure of the linearization matrix can be illustrated with the help of two 'blocked' rows that correspond to a node in the PML region, given by \( \text{node}_{\text{pml}} \), and a node in free space, denoted by \( \text{node}_{\text{free}} \). Additionally, \( \text{node}_{\text{pml,nbr}} \) and \( \text{node}_{\text{free,nbr}} \) denote the neighboring nodes of \( \text{node}_{\text{pml}} \) and \( \text{node}_{\text{free}} \), respectively. It must be noted that the considered free space node does not belong to a duplicate node pair. The free space node has 6 unknowns while the PML node has 12. The structure of the two rows in the sparse matrix can be illustrated as
A similar pattern can be observed for surface nodes. However, nodes that are part of a material interface, or are part of a duplicate node pair, may have different block size sub-matrices. An example of such a case would be a duplicate node pair found at the PML/air interface. It can be seen that for a node \textit{node}_a and its neighbor \textit{node}_{anbr}, where \textit{node}_a and \textit{node}_{anbr} may belong to either the PML or air, there are two specific cases that can arise for the row that corresponds to \textit{node}_a. These cases are

\[
\begin{align*}
\text{number of equations in node}_a & \leq \text{number of equations in node}_{anbr} \\
\text{number of equations in node}_a & > \text{number of equations in node}_{anbr}
\end{align*}
\]

In the first case the sub-matrix corresponding to the off-diagonal belonging to \textit{node}_{anbr}, on the row corresponding to \textit{node}_a in the sparse linearization matrix, has the same size as the sub-matrix of \textit{node}_a.
In the second case the sub-matrix corresponding to the off-diagonal belonging to \( \text{node}_{\text{anbr}} \), for the row corresponding to \( \text{node}_a \) in the sparse linearization matrix takes the form given by

\[
\begin{bmatrix}
\text{node}_a & \text{node}_{\text{anbr}} \\
\vdots & \\
\vdots & \\
[6 \times 6] & [6 \times 6] \\
\vdots & \\
\vdots & \\
\end{bmatrix} \Leftarrow \text{row corresponding to node}_a
\]

In the second case the sub-matrix has the dimension of the \( \text{node}_{\text{anbr}} \). This implies a sub-matrix size that is determined as \( \min(\text{dim}(\text{node}_a, \text{dim}(\text{node}_{\text{anbr}}))) \). While this applies to PML/air interfaces, the inclusion of polarization terms requires a modification in the way sub-matrix block sizes are determined for rows that involve duplicate node neighbors. This is done to take advantage of the fact that the dimensions of the linearization sub-matrix are \((n-p,n)\) where \(n\) is the total number of equations that can include the Maxwell, polarization and PML terms. The last \(p\) rows are populated by zero values, hence they have no contribution to the sub-matrix. If an air/dielectric interface is considered with 6 and 9 variables respectively, where the node in the air region is represented by
node_{air} and the node in the dielectric by node_{d}, the rows in the linearization matrix for each of these nodes can be expressed as

\[
\begin{bmatrix}
\text{node}_{air} & \text{node}_{d} \\
\vdots & \vdots \\
[6 \times 6] & [6 \times 9] \\
\vdots & \vdots \\
\end{bmatrix}
\]
\[\Leftarrow \text{row corresponding to node}_{air}\]

\[
\begin{bmatrix}
\text{node}_{air} & \text{node}_{d} \\
\vdots & \vdots \\
\vdots & \vdots \\
[6 \times 6] & [9 \times 9] \\
\vdots & \vdots \\
\end{bmatrix}
\]
\[\Leftarrow \text{row corresponding to node}_{d}\]

The above conditions enable the creation of a new 'unrolled' linearization matrix that is based on the knowledge of the material properties associated with a node. Such an approach allows the solution of problems that involve materials with differing number of unknowns without being constrained by a fixed block size. The savings can be substantial, particularly if the dispersive region is relatively small compared to the rest of the computational region.
7.1 Introduction

In the late 1990’s scientists began to adopt Graphics Processing Units (GPUs) for scientific applications, with particular emphasis in the fields of electromagnetics and medical imaging. However, because GPUs had initially been designed as dedicated fixed function graphics pipelines, doing so necessitated mapping the scientific applications to resemble pixel shaders [33]. For accelerating scientific codes on a GPU, the sequential portion of the application usually executes on the CPU, which is mostly optimized for non computational tasks such as branching [10] while the computationally intensive sections are dispatched to the GPU kernel.

7.2 Methodology

CUDA allows the application developer to write device code in C functions known as kernels. A kernel differs from a regular function in that it is executed by many GPU threads in a Single-Instruction Multiple Data (SIMD) fashion. An EM solver presents ample opportunities for parallelization as the mesh elements can be spatially partitioned and allocated to the compute resources. Because a finite-element code tends to be quite cumbersome compared to its alternative, the FDTD method, there exists opportunities and hurdles for a CUDA implementation. No predetermined memory access pattern exists for the nodes, thus adapting an unstructured solver to take advantage of the GPU architecture can be challenging. In accordance with Amdahl’s
law the code is profiled to determine the most time intensive subroutines that can benefit from acceleration. As a result, the residual calculation has been determined to dominate the solver runtime implying that high priority is placed on porting these calculations to the GPU. These routines have consequently been rewritten in C to execute as a kernel on the GPU. Because the majority of the code is written in Fortran, wrapper code has been developed to provide the interface with the CUDA kernel. This involves an additional data transfer, copying from Fortran code to C wrapper and from there to the kernel. Compatible data structures are created in C for the purpose of message passing from Fortran. Every effort has been made to maintain flexibility in the code and make it as generic as possible.

Mesh associated parameters that are read only are stored in shared memory, however the residuals are stored in global memory. Access conflicts for a node residual can result in delays that can also severely affect performance. There are two ways to deal with this problem. One option involves renumbering the nodes to minimize conflicts between threads, although this still does not eliminate contention at nodes shared by different threads. This method uses atomic operations to ensure thread safe operation, but it is an operation that can severely affect performance. In the second method, memory locations for the residual are allocated for the nodes associated with each mesh element assigned to a thread, regardless of whether or not the same node is accessed in another thread. Each thread writes its residual contribution to a separate memory location associated with a node and the final residual at each node is accumulated using a reduction operator across the threads. This comes at the price of increased memory usage, however all memory accesses are uncontested. As a result peak performance can be obtained as long as there is sufficient memory in the GPU global memory to store the duplicated residual contributions. The second approach is chosen in this work to minimize access delays.

The residual computation for the CUDA kernel is distributed approximately equally over the blocks to maximize the occupancy of each block. It must be pointed out that determining the ideal load conditions is not a trivial task as this usually involves finding the right balance between the
number of blocks, threads and nodes per thread. Although there are hard limits on most of these parameters, identifying the optimal set of parameters is often dependent on the solver. This is a result of the fact that each application tends to have differing memory access patterns, memory usage, processing needs etc. It is noted that for this study both the baseline Fortran code and CUDA enabled version use single precision arithmetic.

In the flowchart in Fig. (7.1) the boxes in yellow are code sections that are relevant to the porting. Before the beginning of the iteration, data related to the mesh is copied from the main memory space to the GPU memory space. In this implementation only the residual routines are ported onto the GPU. The green boxes indicate Fortran code sections or routines whose function is to pack data into data structures that can be passed to C wrapper code. The blue boxes are C wrapper code routines that receives packed data from Fortran. The yellow boxes represent CUDA kernels that are invoked from the C wrapper. The boxes in orange can be implemented in a similar fashion, however this is left for future work.

CUDA needs to be initialized with a call from the Fortran solver code to a dummy C function, since the first CUDA invocation incurs a startup initialization overhead. This overhead is proportional to the size of the data allocated, hence a large memory allocation can slow down code execution. From the Fortran code, in the driver routine the function ‘initcuda’ is called to initialize CUDA. This subroutine makes a single memory allocation for a single integer, which also implicitly initializes the GPU.

The solver parameters are copied from the global memory to shared memory variables. Global memory has a latency of about 400 to 800 cycles [34], depending on the instruction, while the shared memory is on-chip and has a latency of about 20 cycles. Considering that these parameters will be repeatedly accessed in an iteration loop, significant performance improvements can be achieved by doing so. To maintain data integrity and to provide synchronization, only the first thread makes the data copy. Since the solver parameters reside in shared memory space, all the threads within the block have access to these variables. A ”syncthread” command is then issued
to ensure data integrity. This is due to the fact that the order in which the threads are spawned is indeterminate and threads within a block have no other synchronization method.

Figure 7.1 Proposed algorithm for porting to the GPU

### 7.3 Performance tuning

The GPU implementation was tested on two meshes consisting of tetrahedral elements. Runtimes were obtained and compared against the Fortran code using a mesh with 2930 nodes while scalability was tested using a mesh with 18676 nodes. Optimal performance was obtained for both meshes with at least 128 threads. However increasing the number of threads per block decreases the mesh elements proportionately and thread blocks end up not being fully
utilized. Additionally blocks have an associated context switch time, and memory accesses in an unstructured solver can lead to bank conflicts. One approach to mitigate this problem is to renumber memory nodes for improving memory coalescing. L1 cache can be increased to 48KB at compile time by adding flags `-Xptxas -dlcm=ca` to the compile line. The `CudaDeviceSetCacheConfig` command can be used to set the preference for shared memory or L1 cache. Threads in a single block are executed on a single multiprocessor and attention must be paid to the size of the kernel. In the architecture, the Fermi register limit is 63 and large kernel sizes can cause register spillage. Automatic variables assigned to registers should be reused as much as possible. If needed, or in the event of register spillage, the number of registers can be increased using `-maxregcount`. However, it is possible to obtain better performance with smaller kernels. As opposed to general purpose processors, better performance is obtained by following the principle of 'recompute as opposed to pre-compute' in order to preserve registers. A common problem that may afflict a physics code is the presence of large arrays in a kernel. Large arrays that are present in shared memory are moved into local memory. Inspection of the generated ptx file is recommended to ensure that this has not happened. Most computational kernels can be classified as being either compute bound or memory bound. For compute bound problems the key is to maximize thread count while maintaining the required amount of shared memory and registers. For memory bound problems one should attempt to reduce memory latency using fast memory and caching strategies.
CHAPTER 8
RESULTS

8.1 Runtime comparison

Runtimes were obtained when the system of equations were coupled and unrolled. Table (8.1) illustrates the comparison of these runtimes against the blocked and uncoupled case using a mesh of 13597 node points and 72480 tetrahedral elements. Results were obtained for linear elements and with no optimization enabled. Also, the benefits of the approximate diagonal stabilization matrix, from the perspective of computational time, was investigated using a single pole Debye model and a PEC sphere which has 9 and 6 equations associated with it respectively. The code was run for 2 time-steps using a single processor and it was profiled using gprof. The time spent in the subroutines for the computation of the stabilization term was noted. Table (8.2) compares the runtimes of the Jacobian based stabilization matrix with the diagonal stabilization matrix.

<table>
<thead>
<tr>
<th>No of proc.</th>
<th>Time-steps</th>
<th>Blocked runtime(s)</th>
<th>Unrolled and coupled runtime(s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>10</td>
<td>196.73</td>
<td>165.47</td>
<td>1.19</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
<td>1278.52</td>
<td>592.0</td>
<td>2.16</td>
</tr>
<tr>
<td>96</td>
<td>500</td>
<td>4539.29</td>
<td>1802.1</td>
<td>2.52</td>
</tr>
</tbody>
</table>

Table 8.1 Runtime comparison of blocked and 'unrolled' equations
Table 8.2 Runtime (ms) of Jacobian based $[\tau]$ and diagonal $[\tau]$

<table>
<thead>
<tr>
<th>Material</th>
<th>Jacobian based $\tau$ (Total time)</th>
<th>approximate $\tau$(Total time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debye</td>
<td>213.09 (1041.59)</td>
<td>27.90 (479.85)</td>
</tr>
<tr>
<td>PEC</td>
<td>70.69 (323.99)</td>
<td>10.12 (147.04)</td>
</tr>
</tbody>
</table>

8.2 Bistatic RCS for PEC sphere

Bistatic RCS of a PEC sphere of radius 0.6m was computed by propagating a Gaussian wave, with $E_x$ and $B_y$ components, in the z direction. Figure (8.1) illustrates wave propagation in a plane perpendicular to the x axis. A time-step of 0.01s with a second order BDF2 temporal scheme was used for marching this wave in time. This time-step was chosen to accurately resolve the frequency content of the source wave, while ensuring that the temporal truncation error does not corrupt the solution. Since this is essentially a direct numerical simulation, the smaller time-steps are needed to resolve larger frequencies. Quadratic or P2 elements were used for the computation of the bistatic RCS, which results in a solution accuracy of order 3. It is necessary to have sufficient number of mesh points per wavelength to appropriately sample the wave in space. Duplicate faces were created between each PML volume since each PML region is treated as a non-material block. This is done so that each PML block can have distinct damping parameters. The mesh was discretized using 54996 tetrahedral elements and 10846 node points. Bistatic RCS is used to verify all the test cases below. The plots illustrate the variation of the bistatic RCS, for a fixed $\phi$, over $\theta$ which ranges from 0 to 180 degrees. Bistatic RCS is plotted over $\theta$, for $\phi$ corresponding to 36, 45 and 90 degrees in Fig. (8.2). The obtained solutions are compared against the solutions from HFSS [35], which is a frequency domain solver for electromagnetic problems.
Figure 8.1 Gaussian wave propagation in the z direction - PEC sphere
Figure 8.2 Bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz
The bistatic RCS is computed for the same case using the approximate diagonal \([\tau]\) and compared against the original Jacobian based stabilization matrix in Fig. (8.3) and they are found to be in good agreement. These results are additionally compared at a higher frequency of 220 Mhz in Fig. (8.4). This demonstrates that a Gaussian pulse may be used to obtain results over a broad range of frequencies. Results computed using linear or P1 elements were also compared against those obtained from HFSS and plotted in Fig. (8.5). It is seen that there are significant differences in these two solutions, therefore it is clear that the solution accuracy provided by P1 elements is not sufficient for this case. It should be noted that all results shown are obtained using quadratic or P2 elements unless otherwise stated.
Figure 8.3 Approximate diagonal stabilization vs Jacobian based stabilization - bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz
Figure 8.4 Approximate diagonal stabilization vs Jacobian based stabilization - bistatic RCS for PEC sphere of radius = 0.6m at frequency = 220 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz
Figure 8.5 P1 results compared against HFSS using bistatic RCS for PEC sphere of radius = 0.6m at frequency = 150 Mhz using a Gaussian pulse of center frequency = 150 Mhz and bandwidth = 100 Mhz
8.3 Bistatic RCS for dielectric sphere

Instead of a PEC sphere, the bistatic RCS is computed and plotted for a dielectric sphere of radius 0.6m at a frequency of 150 Mhz. The mesh used for this purpose is made up of 13597 nodes and 72480 tetrahedral elements. Figure (8.6) shows a slice plane normal to the x-axis and passing through the center of the dielectric given by the coordinates (0,0,0). Note that the driving wave starts at $z = -1.0$ and propagates in the z direction towards $z = 1.0$ where the PML layer begins. Figure (8.7) illustrates the variation of the $D_x$ component for a plane wave propagating in the positive z direction. Figure (8.8) illustrates the bistatic RCS for a dielectric sphere with $\epsilon_r$ set to 3, which is compared against the solution obtained from the frequency domain solver HFSS and they are found to be in good agreement.
Figure 8.6 Gaussian wave of center frequency = 150 Mhz and bandwidth = 100 Mhz; propagation in the z direction in the presence of dielectric sphere with $\varepsilon_r = 3$
Figure 8.7 Variation of $D_x$ component of plane wave propagating in $z$ direction in the presence of dielectric sphere of $\epsilon_r = 3$ at frequency = 150 Mhz
Figure 8.8 Bistatic RCS for a frequency-independent dielectric sphere with $\varepsilon_r = 3$ of radius $= 0.6m$ at frequency $= 150$ Mhz using a Gaussian pulse of center frequency $= 150$ Mhz and bandwidth $= 100$ Mhz
8.4 Bistatic RCS for Debye sphere

Scattering from a sphere of radius 0.6m, with frequency-dependent dielectric properties modeled using an isotropic Debye model is displayed in Fig. (8.9). Table (8.3) illustrates the material properties associated with this model. The bistatic RCS at 150 Mhz is computed, using a mesh with 13597 nodes and 72480 tetrahedral elements, and plotted at 150 Mhz, using a single pole Fig. (8.10) and two pole models Fig. (8.11). These results are compared with the solution obtained from HFSS. Both single pole and two pole solutions are found to be in good agreement with the corresponding solutions obtained from HFSS. The slight discrepancy in the two pole solution can possibly be attributed to the limited precision in the material parameters used by HFSS.

Table 8.3 Material properties for the Debye model listed by pole number

<table>
<thead>
<tr>
<th>Pole number</th>
<th>$\epsilon_\infty$</th>
<th>$\Delta \epsilon$</th>
<th>$\tau_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.88</td>
<td>2.0</td>
<td>0.1194</td>
</tr>
<tr>
<td>2</td>
<td>2.88</td>
<td>2.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Figure 8.9 Gaussian wave propagation in the z direction in the presence of a dispersive sphere as a Debye single pole model
Figure 8.10 Bistatic RCS for a Debye model single pole sphere using $\varepsilon_\infty = 2.88, \Delta \epsilon = 2.0, \tau_d = 0.1194$ at frequency = 150 Mhz
Figure 8.11 Bistatic RCS for a Debye model 2 pole sphere using $\varepsilon_\infty = 2.88, \Delta \varepsilon_1 = 2.0, \tau_{d1} = 0.1194, \Delta \varepsilon_2 = 2.5, \tau_{d2} = 0.3$ at frequency = 150 Mhz
8.5 Bistatic RCS for Lorentz sphere

Scattering from a sphere of radius 0.6m, with frequency-dependent dielectric properties, modeled using an isotropic Lorentz model is displayed in Fig. (8.12). Table (8.4) illustrates the material properties associated with this model. The bistatic RCS is computed and plotted at 150 Mhz, using a single pole Fig. (8.13) and two pole models Fig.(8.14). The mesh used for this simulation consists of 13597 nodes and 72480 tetrahedral elements. These results are compared with the solution obtained from HFSS. Both single pole and two pole solutions are found to be in good agreement with the corresponding solutions obtained from HFSS. The slight discrepancy in the single pole solution can possibly be attributed to the limited precision in the material parameters used by HFSS.

Table 8.4 Material properties for the Lorentz model listed by pole number

<table>
<thead>
<tr>
<th>Pole number</th>
<th>$\epsilon_\infty$</th>
<th>$\delta\epsilon$</th>
<th>$\Delta_p$</th>
<th>$\omega_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.88</td>
<td>2.0</td>
<td>2.5</td>
<td>0.1194</td>
</tr>
<tr>
<td>2</td>
<td>2.88</td>
<td>2.0</td>
<td>2.5</td>
<td>0.1194</td>
</tr>
</tbody>
</table>
Figure 8.12 Gaussian wave propagation in the z direction in the presence of a dispersive sphere modeled using a Lorentz single pole model
Figure 8.13 Bistatic RCS for a Lorentz model single pole sphere using $\epsilon_{\infty} = 2.88, \Delta \epsilon = 2.0$, $\delta_p = 2.5$, $\omega_p = 0.1194$ at frequency $= 150$ Mhz
Figure 8.14 Bistatic RCS for a Lorentz model 2 pole sphere using $\varepsilon_\infty = 2.88$, $\Delta\varepsilon_1 = 2.0$, $\delta_{\rho 1} = 2.5$, $\omega_{\rho 1} = 0.1194$, $\Delta\varepsilon_2 = 2.0$, $\delta_{\rho 2} = 2.5$, $\omega_{\rho 2} = 0.1194$ at frequency = 150 Mhz
8.6 Bistatic RCS for multilayer Debye sphere

Scattering from a multilayer sphere with frequency-dependent dielectric properties modeled using isotropic single pole Debye models is displayed in Fig. (8.15). Table (8.5) illustrates the material properties associated with this model listed by layer. The mesh used for this simulation consists of 15596 nodes and 86874 tetrahedral elements. The sphere consists of 2 layers with inner radius 0.12m and outer radius 0.6m. These concentric layers have distinct material properties. The bistatic RCS is computed and plotted at 150 Mhz, using a single pole Fig. (8.16) and compared with the results obtained from HFSS and they are found to be in good agreement.

Table 8.5 Material properties for the multilayer single pole Debye sphere model

<table>
<thead>
<tr>
<th>Layer</th>
<th>$\epsilon_{\infty}$</th>
<th>$\Delta\epsilon$</th>
<th>$\tau_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer</td>
<td>2.88</td>
<td>2.0</td>
<td>0.1194</td>
</tr>
<tr>
<td>Inner</td>
<td>2.88</td>
<td>2.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Figure 8.15 Gaussian wave propagation in the z direction in the presence of a dispersive 2 layer sphere using Debye single pole models
Figure 8.16 Bistatic RCS for a 2 layer sphere with single pole Debye models. Outer layer \( \epsilon_{\infty_1} = 2.88, \Delta \epsilon_1 = 2.0, \tau_{d_1} = 0.1194 \), and inner layer \( \epsilon_{\infty_2} = 2.0, \Delta \epsilon_2 = 2.5, \tau_{d_2} = 0.3 \), at frequency = 150 Mhz
8.7 Bistatic RCS for NASA Almond

Radar cross sections are computed for a NASA Almond [36] with characteristic length \( d = 0.62 \) m at 0.15 Ghz and 0.5 Ghz. A PEC boundary condition is applied to the surface of the geometry. A NASA Almond is a three dimensional structure that can be divided into two halves given by an ellipsoid and a half elliptic ogive. The parametric definition of the ellipsoid is given by

\[
x = d t \\
y = 0.193333d \left( \sqrt{1 - \left( \frac{t}{0.416667} \right)^2} \cos(\psi) \right) \\
z = 0.064444d \left( \sqrt{1 - \left( \frac{t}{0.416667} \right)^2} \sin(\psi) \right)
\]

and the half elliptic ogive is given by

\[
x = d t \\
y = 4.833450d \left( \sqrt{1 - \left( \frac{t}{2.083350} \right)^2} - 0.96 \right) \cos(\psi) \\
z = 1.611148d \left( \sqrt{1 - \left( \frac{t}{2.083350} \right)^2} - 0.96 \right) \sin(\psi)
\]

This definition was used to parametrically create a database which was imported into the mesh generation tool Pointwise. To faithfully reproduce this geometry in the database, hyperbolic sine clustering was used to refine the edges towards the geometric singularity in the ogive and to appropriately capture the curvature of the ellipsoid Fig. (8.17). In Pointwise care was taken to ensure that there are an adequate number of points at both ends of the geometry, as shown in Fig. (8.18), Fig. (8.19) and Fig. (8.20). The coarser mesh in Figs. (8.18) and (8.19) consists of 34465
nodes whereas the refined mesh in Fig. (8.20) consists of 57101 nodes. Figure (8.21) illustrates a Gaussian wave of center frequency 150 Mhz propagating toward the NASA Almond in the z direction. Results of the refined mesh at 150 Mhz are shown in Fig. (8.22) using a time-step of 0.0025s. Effects of the singularity can be seen in the discrepancy between the computed solution and the HFSS solution. To explore this further, a simulation was run at a frequency of 500 Mhz using a time-step of 0.00125s for 11200 time-steps as illustrated in Fig. (8.23). Bistatic RCS computed at this frequency is plotted in Fig. (8.24). There are two reasons for the differences in the computed solution and the one obtained from HFSS. One is the temporal truncation error introduced by the time-step used in the time-domain solver. This is investigated by plotting the bistatic RCS at 500 Mhz for $\phi = 36$ in Fig. (8.25). Three different time-steps, given by 0.005s, 0.0025s and 0.00125s are used for this purpose. It is seen that the difference between the two solutions decreases as time-step is decreased. This is to be expected as the frequency is increased, since the electrical length of the geometry increases with frequency. Another possible source of error is the geometry used by HFSS. In Fig. (8.26) and Fig. (8.27) it can be seen that the imported geometry used by HFSS is faceted as opposed to the smoother high fidelity representation seen in Fig. (8.17) and Fig. (8.20).
Figure 8.17 Database created using hyperbolic sine clustering toward the ends, characteristic length $d = 0.62\text{m}$
Figure 8.18 Coarse mesh for PEC NASA Almond with characteristic length \( d = 0.62 \text{m} \)
Figure 8.19 Coarse mesh for PEC NASA Almond with characteristic length $d = 0.62\text{m}$, oriented in the $z$ direction and in the $x$-y plane
Figure 8.20 Refined mesh for PEC NASA Almond with characteristic length $d = 0.62$ m, oriented in the $z$ direction and in the $x$-y plane
Figure 8.21 Gaussian wave propagation of center frequency $= 150$ Mhz and bandwidth $= 100$ Mhz in the z direction in the presence of a NASA Almond with a PEC boundary condition
Figure 8.22 Bistatic RCS for a PEC NASA Almond with characteristic length $d = 0.62\text{m}$ frequency $= 150\ \text{Mhz}$ using a Gaussian pulse of center frequency $= 150\ \text{Mhz}$ and bandwidth $= 100\ \text{Mhz}$, time-step $= 0.0025\text{s}$ run for 5600 time-steps
Figure 8.23 Gaussian wave propagation of center frequency = 500 Mhz and bandwidth = 200 Mhz in the z direction, in the presence of a NASA Almond with a PEC boundary condition
Figure 8.24 Bistatic RCS for a PEC NASA Almond with characteristic length = 0.62m, frequency = 500 Mhz using a Gaussian pulse of center frequency = 500 Mhz and bandwidth = 200 Mhz, time-step = 0.00125s run for 11200 time-steps
Figure 8.25 Illustration of the effect of temporal truncation error due to time-step. Bistatic RCS at $\phi = 36$ for a PEC NASA Almond with characteristic length = 0.62m, frequency = 500 Mhz using 3 different time-steps.
Figure 8.26 NASA Almond mesh in HFSS
Figure 8.27 NASA Almond geometry in HFSS
8.8 Simulation of a four layer human brain

Simulations were also run using the three dimensional model of the human brain [37], that consists of four layers. These layers represent the scalp, cerebrospinal fluid (CSF), gray matter and white matter. While the mesh is available as a download in a ’mat’ file format, it was not ideally suited for numerical simulation. The mesh was converted into a facet file format from which a volume mesh was generated that was of sufficient quality. A single pole Debye model, the material properties of which was obtained from [22], was used to simulate the regions of the brain. Table (8.6) lists the dielectric properties of the brain, as used in the simulation. Figure (8.28) show the outer layer of the brain mesh, while Fig. (8.29) and Fig. (8.30) show the CSF and the white matter respectively. To illustrate the quality of the mesh, the aspect ratio and the maximum included angle in the mesh are shown in Fig. (8.31) and Fig. (8.32) respectively. The $D_x$ component of plane wave propagation at 500 Mhz through the brain is shown in Fig. (8.33) and Fig. (8.34).

Table 8.6 Debye material properties for human brain

<table>
<thead>
<tr>
<th>Layer No</th>
<th>Name</th>
<th>$\varepsilon \rho s_{\infty}$</th>
<th>$\Delta \varepsilon$</th>
<th>$\tau_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scalp</td>
<td>12.73</td>
<td>3.545</td>
<td>0.25413</td>
</tr>
<tr>
<td>2</td>
<td>CSF</td>
<td>5</td>
<td>73.8</td>
<td>0.00294</td>
</tr>
<tr>
<td>3</td>
<td>Gray matter</td>
<td>52.99</td>
<td>52.88</td>
<td>0.4032</td>
</tr>
<tr>
<td>4</td>
<td>White matter</td>
<td>38.99</td>
<td>31.71</td>
<td>0.3645</td>
</tr>
</tbody>
</table>
Figure 8.28 Brain mesh with 4 layers
Figure 8.29 CSF layer of brain mesh
Figure 8.30 White matter of brain mesh
Figure 8.31 Aspect ratio for brain mesh
Figure 8.32 Maximum included angle for brain mesh
Figure 8.33 Simulation of a 4 layer brain mesh at 500 Mhz
Figure 8.34 Simulation of a 4 layer brain mesh at 500 Mhz
8.9 GPU results and runtime comparisons

The GPU implementation has been tested on a single GTX470 desktop graphics card from NVIDIA using a CUDA compute capability version 2.0 and a CUDA runtime driver version 3.20. The results obtained using the GPU are compared to those obtained using the original code. The graphics card has 14 multiprocessors (clocked at 1.22 Ghz), each comprising 32 cores for a total of 448 cores. Wall clock times are recorded during each run for both the original Fortran version and for the code ported to the GPU. Table (8.7) below compares the execution times on Fortran with a GPU implementation using a 2930 node tetrahedral mesh.

<table>
<thead>
<tr>
<th>Total threads</th>
<th>Blocks</th>
<th>Threads per block</th>
<th>Mesh elements</th>
<th>Kernel runtime</th>
<th>Fortran runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>800</td>
<td>25</td>
<td>32</td>
<td>140</td>
<td>0.679498</td>
<td>3.249644</td>
</tr>
<tr>
<td>1600</td>
<td>25</td>
<td>64</td>
<td>70</td>
<td>0.383718</td>
<td>3.250823</td>
</tr>
<tr>
<td>1600</td>
<td>50</td>
<td>32</td>
<td>70</td>
<td>0.617468</td>
<td>3.269889</td>
</tr>
<tr>
<td>3200</td>
<td>50</td>
<td>64</td>
<td>35</td>
<td>0.620021</td>
<td>3.256026</td>
</tr>
<tr>
<td>3200</td>
<td>25</td>
<td>128</td>
<td>35</td>
<td>0.404993</td>
<td>3.300707</td>
</tr>
<tr>
<td>2560</td>
<td>20</td>
<td>128</td>
<td>40</td>
<td>0.279704</td>
<td>3.255644</td>
</tr>
<tr>
<td>5120</td>
<td>20</td>
<td>256</td>
<td>20</td>
<td>0.301782</td>
<td>3.250507</td>
</tr>
<tr>
<td>6400</td>
<td>50</td>
<td>128</td>
<td>18</td>
<td>0.617136</td>
<td>3.250560</td>
</tr>
</tbody>
</table>

It can be seen from Table (8.7) that load balancing is key to performance on the GPU. In the Fermi architecture each streaming processor can have up to 8 active blocks and 48 active warps (or 1536 active threads). The maximum number of threads per block is 1024 and the warp size is 32. Ideally at least 128 threads are recommended for optimum performance, however increasing the number of threads per block decreases the number of mesh elements proportionately; as a result the blocks are not fully utilized. Blocks have an associated context-switch cost as a result of having to
save registers and shared memory, hence increasing block size indiscriminately will, in fact, slow
down execution. Threads in a single block are executed on a single multiprocessor and thus large
block sizes can cause shared memory (software-managed data cache) register spillage. This may
cause some variables to be stored in the global memory located off-chip. There is no determinate
means to ensure that arrays assigned to shared memory will necessarily reside there. Those that
are deemed too large are moved into the global memory, which can result in severe performance
degradation. Inspection of the generated ptx file is recommended to ensure that such an event has
not occurred.

Table (8.8) compares average execution times of Fortran code and CUDA kernel (optimal
parameters and load conditions) for two different meshes to ensure scalability. Note that the CUDA
runtimes include memory transfers to the wrapper and the CUDA kernel since this metric provides
a more realistic estimate of achievable runtimes in real world applications. Runtimes for the larger
mesh indicate that the algorithm scales well with the increased load. In fact the speedup increased
from a factor of 9 for the smaller mesh to slightly over 11 for the larger mesh, which is consistent
with the notion that GPU startup overhead and data transfer latencies are amortized by larger
workloads. The size of the mesh that can be used is only limited by the size of the GPU global
memory, which in this case is 1.2 GB.

<table>
<thead>
<tr>
<th>Mesh</th>
<th>Fortran execution time</th>
<th>CUDA kernel execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unstructured mesh with 2930 nodes and 14181 tetr</td>
<td>0.45s</td>
<td>0.05s</td>
</tr>
<tr>
<td>ahedron elements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unstructured mesh with 18676 nodes and 99428 tetr</td>
<td>3.25s</td>
<td>0.28s</td>
</tr>
<tr>
<td>ahedron elements</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A scalable method for the time-accurate solution of electromagnetic problems, using stabilized finite-element methods, that involve multipole dispersive materials has been proposed and implemented. For this purpose, Debye and Lorentz multipole material models were implemented, which involved the solution of ADEs for each polarization term. The ability to represent materials using high-fidelity multipole models is crucial when their material properties vary significantly over the frequency range of interest. An example where this is observed is the simulation of biological tissues and organs. The time-domain solution obtained as a result of scattering from a sphere, using both single and multipole models, was compared with the solution from a commercial frequency-domain electromagnetic solver HFSS. The results were found to be in good agreement.

Since additional equations needed to be solved at a region with dispersive material properties, an efficient way to deal with different block sizes was needed. The proposed solution involved unrolling the system of equations so that a fixed block size, corresponding to the region with the most number of equations, was no longer necessary. The solution runtimes were compared for fixed block size cases and an 'unrolled' system of equations for different materials. Performance benefits were observed when the system of equations was 'unrolled' for both dispersive and non-dispersive materials. Additionally, the 'unrolled' method was found to scale well when the solution region contained materials with a large number of unknowns.

Furthermore, it was demonstrated how a full Jacobian based stabilized Petrov-Galerkin method can be inefficient for problems involving dispersive materials. To mitigate this problem, an alternative diagonal stabilization matrix that was computationally simpler compared to a rigorous
Jacobian based stabilization matrix was proposed. This approach enabled the solution of problems with high-fidelity material models within a reasonable amount of time. The solution obtained using the diagonal stabilization matrix was then compared against the solution obtained using a full Jacobian based stabilization matrix for problems involving dispersive and non-dispersive materials, and these were found to be in good agreement. Runtimes using both approaches were noted for a non-dispersive problem that included a PEC material. Runtimes were also computed for dispersive material models using a single pole Debye material.

To compute farfield information, from a radiating source or as a result of scattering from arbitrary objects, a Near-to-Far-Field (NTFF) transformation based on the Huygen’s surface equivalence theorem is implemented. This has been utilized, for example, to compute the radiation pattern of antennas and radar cross sections of arbitrary geometries. Scattering by plane waves, from a PEC sphere, was used to verify the implementation of NTFF. The computed bistatic radar cross section was compared with the analytical exact solution based on Bessel functions. These results were also compared with the solution obtained from HFSS and were found to be in good agreement. Using a Gaussian pulse, the ability to compute accurate solutions over a broad range of frequencies was demonstrated. Bistatic radar cross sections were computed for a NASA Almond to explore the effects of a geometric singularity on the time-domain solution. The variation of the solution error with time-step sizes was studied, and the need for small time-steps in such cases established.

A Perfectly Matched Layer (PML) absorbing boundary layer was implemented to terminate the computational region. The PML, based on the formulation by Johnson [14], serves to damp out waves without the appearance of undesired reflections. This was accomplished by solving for an additional set of six equations, corresponding to the PML variables, in the absorbing boundary layer. Although these equations were originally solved in a staggered manner, which necessitated multiple sub-iterations at each time-step, they were later fully coupled with the six Maxwell’s equations. In doing so, the solution of the joint linear system of equations in a single Newton
iterate was possible.

The suitability of the GPU architecture for the solution of unstructured finite-element problems was investigated by porting a portion of the residual routine to CUDA. Runtimes of the CUDA kernel were compared against the Fortran implementation for varying block sizes and threads. Since unstructured solvers have no predetermined memory access patterns, a load partitioning strategy was recommended that eliminates access contention across threads, at a memory location in the kernel. The limitations of this approach were discussed and various recommendations were made for obtaining optimal performance in similar problems.

Future work involves the investigation of the dispersive material capability for problems such as the simulation of metamaterials and high frequency electronic circuit design. These are problems that involve frequency-dependent materials and require an efficient means to obtain accurate solutions over a broad range of frequencies. Another possible research area is the solution of inverse scattering problems in medical imaging, where it has been shown that malignant tissues can be identified based on their scattering response. Furthermore, the remainder of the time-domain code, including the linear algebra routines, should be ported to CUDA. Additionally, it needs to be investigated whether such an unstructured finite-element code can benefit from heterogeneous architectures for computation.
REFERENCES


VITA

Srijith Rajamohan obtained his bachelors in Electronics and Communications Engineering from the Cochin University of Science and Technology in 2005. He then proceeded to obtain his Masters in Electrical Engineering at the Pennsylvania State University, where he was interested in high performance computing and hardware accelerators. After graduating with his Masters in 2009, he decided to pursue his Phd in Computational Engineering at the SimCenter, University of Tennessee at Chattanooga and earned his degree in 2014.