COMPUTATIONAL DESIGN FOR ELECTROMAGNETIC SIMULATIONS

By

Ryan Steven Glasby

Approved:

W. Kyle Anderson Professor (Director of Thesis)	David L. Whitfield Director of Graduate School of Computational Engineering (Committee Member)
Lafayette K. Taylor Research Professor (Committee Member)	Daniel G. Hyams Associate Professor (Committee Member)
John V. Matthews, III Assistant Professor of Mathematics (Committee Member)	Will Sutton Dean of College of Engineering and Computer Science

A. Jerald Ainsworth
Dean of the Graduate School

COMPUTATIONAL DESIGN FOR ELECTROMAGNETIC SIMULATIONS

By

Ryan Steven Glasby

A Thesis
Submitted to the Faculty of the
University of Tennessee at Chattanooga
in Partial Fulfillment of the Requirements
for the Doctoral Degree in
Computational Engineering

The University of Tennessee at Chattanooga
Chattanooga, Tennessee

August 2011

ABSTRACT

An automatic computational procedure has been developed to efficiently and accurately design the shape of complicated electromagnetic objects. These electromagnetic objects can be simulated for operation at high frequencies (~10 GHz), and can be comprised of dissimilar materials. The automated design procedure consists of linking together an original electromagnetic field simulation tool, an original adjoint routine for obtaining sensitivity derivatives, and an original grid-smoothing tool with an existing optimization package. The electromagnetic field simulation software employs a temporally and spatially higher-order accurate Streamline Upwind/Petrov-Galerkin finite-element method that numerically solves Maxwell's equations in the time domain using implicit time stepping. The software for computing sensitivity derivatives employs a reverse-mode time-accurate discrete adjoint methodology that is formulated to automatically maintain consistency with the electromagnetic field simulation software. Grid smoothing is achieved using a spatially higher-order accurate Galerkin finite-element method that generates a numerical solution to the linear elastic equations. All computational solutions to the linear systems present in each software tool are obtained using the Generalized Minimum Residual algorithm with block diagonal preconditioning. Each software tool is implemented using a parallel processing paradigm and is therefore capable of being executed on a distributed memory supercomputer.

The order of accuracy of the electromagnetic field simulation software has been determined by using comparisons with exact solutions. The field software's results were compared to the exact

solution of a rectangular resonant cavity. In all cases, the order properties of the field software exceed theoretical expectations when linear, quadratic, and cubic tetrahedral elements are employed to discretize the field.

To demonstrate the consistency of the adjoint-based sensitivity derivates with those obtained directly from the field solver, derivatives have been extracted from the field software using a complex variable technique. The sensitivity derivatives from the reverse-mode time-accurate discrete adjoint method were then compared and demonstrated to agree to at least seven decimal places.

As a demonstration of the assembled technologies, the optimization procedure successfully and efficiently modified the shape of two electromagnetic objects to reduce a specified cost function. A dielectric cube, under the influence of a propagating plane wave, was repositioned within a larger free space volume so that the field variables on the surface of the cube match desired values at a specified time. A similar demonstration case has also been conducted to modify the shape of a dielectric ellipsoid, under the same conditions as the cube.

DEDICATION

Dedicated to Alicia and Scarlett

ACKNOWLEDGEMENTS

The author expresses his sincere appreciation to his principal advisor, committee, and to the rest of the SimCenter staff and students. First of all, I give sincere gratitude to Dr. W. Kyle Anderson who spent endless hours disseminating his incredible wealth of knowledge on the topic of this dissertation. I also give expressed appreciation to the other members of my committee, namely, Dr. Whitfield, Dr. Taylor, Dr. Hyams, and Dr. Matthews who have passed on much of their vast knowledge of computational engineering through their superb teaching styles. I thank the SimCenter staff and students for their support and interesting discussions. I thank Dr. Briley and Dr. Karman for passing along their expertise of viscous flow fields and grid generation. I also thank Dr. Burdyshaw for passing along his proficiency and experience base in computational design methodologies. Lastly, I especially thank Mr. Wally Edmondson whose know-how in systems administration in unparalleled.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

A	Derivative of flux in the x-direction with solution vector
$\mathcal{A}$	Term in residual equation
$\bar{A}$	A multiplied by τ
a_o	Inverse of wave impedance
a, b, c	Ellipsoid length, width, and height
α	Coefficient in linear elastic equations
B	Derivative of flux in the y-direction with solution vector
$\mathcal{B}$	Term in residual equation
$\bar{B}$	B multiplied by τ
$\vec{B}$	Magnetic flux density
β	Design variable
$\beta_x, \beta_y, \beta_z$	Mode wave numbers for rectangular resonant cavity
C	Derivative of flux in the z-direction with solution vector
$\mathcal{C}$	Term in residual equation
$\bar{C}$	C multiplied by τ
c	Speed of light
c_r	Relative speed of light
$\vec{D}$	Electric flux density

$\vec{\vec{D}}$	Ratio of electric flux density and inverse of wave impedance
$\mathcal{D}$	Term in residual equation
d	Displacement
$\vec{E}$	Electric field
ε	Permittivity
ε_r	Relative permittivity
$\vec{F}$	Flux in the x-direction
$\vec{F}_{LE}$	Flux in the x-direction for the linear elastic equations
$\vec{G}$	Flux in the y-direction
$\vec{G}_{LE}$	Flux in the y-direction for the linear elastic equations
Γ	Element surface
$\vec{H}$	Magnetic field, flux in the z-direction
$\vec{H}_{LE}$	Flux in the z-direction for the linear elastic equations
I	Cost function
J	Electric current density
$\mathcal{J}$	Element Jacobian
L	Characteristic length
$\vec{l}$	Left Eigenvector
$\vec{\lambda}$	Eigenvalue vector
λ_1	Adjoint variable
λ_2	Adjoint variable
Λ	Adjoint variable

M	Derivative of solution vector with respect to field variables for jump condition
M_{sub}	Submatrix for the linear elastic equations
μ	Permeability
μ_r	Relative permeability
N_{nodes}	Number of nodes contained within an element
N_i	Shape function/Lagrange polynomial
n_x, n_y, n_z	Cartesian unit normal vectors
$\bar{n}_x, \bar{n}_y, \bar{n}_z$	Cartesian non-unit normal vectors
Ω	Element volume
ω	Frequency
$\vec{q}$	Solution vector
R	Residual
$\vec{r}$	Right Eigenvector
$\vec{S}$	Source terms
σ	Conductivity
σ^*	Conductivity multiplied by characteristic length
T	Right Eigenvector matrix
T^{-1}	Left Eigenvector matrix
t^*, t	Nondimensionalized time
τ	Stabilization matrix
Δt	Nondimensional time increment
θ	Coefficient in linear elastic equations

W_i	Gaussian weight
w	Weighting function
$\overrightarrow{w}$	Field variables for jump condition
x^*	Nondimensionalized position
x, y, z	Nondimensionalized Cartesian coordinate directions
ξ, η, ρ	Parent element coordinate directions

CHAPTER I

INTRODUCTION

James Clerk Maxwell corrected Ampere's law and combined it with Faraday's law and Gauss's law in the early 1860's, which gave rise to Maxwell's equations [1]. These equations, which link electricity and magnetism, were derived from experimental observations. Faraday's law states that a time varying magnetic field induces an electric field, and Ampere's law with Maxwell's correction, states that an electric current and/or a time varying electric field can generate a magnetic field [1]. Assuming the material properties of the field do not depend on the electromagnetic field quantities, Maxwell's equations are a set of linear hyperbolic differential equations. Since the equation set is linear, numerous analytic solutions exist. These analytic solutions have been instrumental to the generation of numerous simple electromagnetic devices. However, analytic methods do not exist to solve complex electromagnetic problems. Traditionally, complex electromagnetic problems have been solved in an experimental laboratory. The scattering parameters of non-radiating devices can be experimentally obtained by a network analyzer, and the radiation pattern of an antenna can be experimentally obtained by a field scanner within an anechoic chamber. Design of complex electromagnetic devices has been based on experimental knowledge. However, costs to gain experimental knowledge is very high; therefore, since the 1960's, scientific researchers have been trying to obtain solutions to Maxwell's equations using numerical simulations.

Maxwell's equations can be cast in the time domain or the frequency domain. The time domain solution of the electromagnetic field can be transferred to the frequency domain by a Fourier transform. The time domain solution can capture all frequencies within a range, but the frequency domain solution has to be generated for each frequency as determined by a pulse within a pre-specified range of interest. Because of the numerical work associated with obtaining solutions in the frequency domain, for the current study, the time-domain formulation of Maxwell's equations is used.

The generation of approximate solutions to Maxwell's equations in the time domain has many difficulties associated with it. The equation set is hyperbolic, and an approximation method that employs upwinding is essential for numerical stability. Complicated electromagnetic devices are often operated at frequencies on the order of ten gigahertz (GHz) where the wavelength in air is approximately one inch. For many applications, the physical size of the relevant device is such that the wavelengths are disproportionately small, thereby making simulations very costly due to the large number of grid points required to adequately resolve all waves. Higher-order finite-element approximation methods can significantly reduce the number of grid points because waveforms can be accurately represented with fewer grid points. This is accomplished by distributing the data within the computational element with higher than first order (linear) polynomials. Even with higher-order elements, some problems are so large that they will not fit within the memory of a single computer, and it can take a significant amount of wall-clock time to generate a meaningful solution. To alleviate this problem, the computational domain is typically divided into numerous partitions which are distributed amongst multiple processes to significantly reduce wall-clock time. An electromagnetic field can contain multiple materials, and at the interface of each material a jump condition is necessary to properly

approximate the electromagnetic field quantities. Two of the eigenvalues of Maxwell's equations are identically zero, and the others are on the order of the speed of light. This has the propensity to make the matrix that represents the linear systems stiff. The use of higher order elements makes the linear system less diagonally dominant which also makes it more challenging to solve. A powerful, memory intensive linear system solver coupled with a preconditioning algorithm can be utilized to generate the time accurate approximate solution of the electromagnetic field.

Historically computer generated approximate solutions to Maxwell's equations utilize the finite-difference time-domain (FDTD) methodology of Yee [2]. For the FDTD methodology the spatial and temporal derivatives are approximated with finite-difference approximations. This leads to the volume of the three dimensional field being discretized with hexahedral elements. These elements are unable to accurately represent curved surfaces, and actually approximate curves with a stair step estimate. Also large computational stencils are needed to approximate the field with higher order accuracy. The FDTD methodology can solve for the electromagnetic field across dissimilar materials, but the stability of the method is constrained by the time step utilized [2].

Another way to approximate the solution to Maxwell's equations is to employ the finite-element time-domain (FETD) methodology. The FETD method approximates Maxwell's equations with higher-order accuracy both spatially and temporally, and is capable of conforming to curved surfaces by discretizing the volume of the three-dimensional field with isoparametric tetrahedral elements. This is accomplished by prescribing the field values within each computational cell with a polynomial distribution. The finite element method is thoroughly discussed by Hughes [3] and Zienkiewicz [4]. The finite element method solves Maxwell's equations in weak form by multiplying the governing equations by a weighting function and

integrating over the volume. The most popular implementation of this method for electromagnetic field simulations is to apply it to a second order wave equation for either the electric or magnetic field variables [5]. The other field variables are obtained in a post-processing step that involves numerical differentiation of the primary variables, and is therefore one order of accuracy less. Since the spatial derivatives of this equation are Laplacian operators, a Galerkin finite element method can generate a solution of the electric or magnetic field with numerical stability. However, one way to generate a higher-order solution for both the electric and magnetic fields that is made up of multiple materials is to utilize a stabilized finite-element method. The two stabilized finite-element methods available are the streamline upwind/Petrov-Galerkin (SU/PG) and the Discontinuous Galerkin (DG) approaches. The SU/PG approach stabilizes the algorithm by adding an artificial stream-wise dissipation term to the weighting function [6] to effectively add a degree of upwinding to the algorithm. The DG approach stabilizes the algorithm by assuming that the field variables for each cell are discontinuous from the field variables for adjacent cells. A Riemann solver [7] is used at the boundary between adjacent cells to obtain the solution to the electromagnetic field. In order to implement this approach, the storage requirements correspond to that obtained by representing the field variables in each tetrahedron independently, without sharing data between elements. For a three-dimensional field discretized with tetrahedra, the number of unknowns is approximately 24.0 times greater for linear elements than a SU/PG scheme, is approximately 7.5 times greater for quadratic elements, and is approximately 7.06 times greater for cubic elements [8]. This fact is a major drawback for the DG method because it takes significantly more computational resources to generate a numerical solution.

Traditionally, radiating and non-radiating electromagnetic objects have been designed from principles learned from analytic solutions to Maxwell's equations and from experimental knowledge gained from experience. The use of automatic computational shape design optimization is a novel alternative for designing electromagnetic objects. Automatic computational shape design couples electromagnetic field simulation software, a numerical routine that obtains sensitivity derivatives, and an optimization package to generate the optimal shape of an object that minimizes a cost function. The sensitivity derivatives are the derivatives of the cost function with respect to the design variables. The sensitivity derivatives can be obtained by either forward or reverse mode methods. The forward mode methods are either Taylor-series approximations or direct-differentiation methods for generating the sensitivity derivatives, and the reverse-mode methods are adjoint based methods. Jameson conducted the ground-breaking practical application of the reverse-mode adjoint based sensitivity analysis in 1988 [9], and applied this technique to aerodynamic optimization. The forward mode methods are inefficient for problems with multiple design variables because a linear system has to be solved for each design variable. Reverse mode methods do not have this inefficiency, and are discussed further in the next paragraph.

Two types of reverse mode adjoint methods exist for computing the shape-sensitivity derivatives. They are the continuous-adjoint and the discrete-adjoint methods. The continuous-adjoint approach takes derivatives of the governing differential equation set with respect to the design variables before the equation set is discretized [10]. A new differentiated equation set is generated and the sensitivity derivates are solved for numerically. However, if the cost function is changed, the process to compute the sensitivity derivatives is repeated to reflect the change. For the discrete-adjoint approach, the discretized field simulation software is differentiated and

the sensitivity derivates can be numerically obtained. The accuracy of these derivatives is directly dependent on the implementation of the field solver, but modifications to the cost function can be easily applied. An example of the implementation of the discrete adjoint method for steady-state problems is shown in [11], and an example of the implementation for time-dependent problems is shown in [12].

In regards to computing adjoint based sensitivity derivatives for electromagnetic problems, the first effort is attributed to Director and Rohrer in 1969 [13] and [14]. They conducted sensitivity analysis for networks, and derived a sensitivity expression based on Tellegen's theorem [15] and [16]. More recently, Sabbagh, Bakr, and Nikolova used the adjoint network method to conduct sensitivity analysis of the scattering parameters of microwave filters in 2005 [17]. They used the full-wave mode-matching technique to simulate an original network, generated sensitivities of scattering parameters with respect to design parameters, and applied the adjoint network method to the generalized scattering matrices of different filter components.

Kang, Chung, Cheon, and Jung implemented a 2-D numerical algorithm to reconstruct the complex permittivity profile of unknown scatterers in 2002 [18]. They simulated the electromagnetic field with the FDTD method, and computed the sensitivity derivates with a continuous adjoint approach. The adjoint variables are solved with the FDTD method, and they used a steepest descent method for optimization. With this methodology, they were able to successfully reconstruct the dielectric constant and the electric conductivity of a 2-D object.

Chung and Cheon partnered with Park and Hahn and developed a continuous adjoint FDTD approach for shape design [19] in 2000. They applied this approach to design the shape of a K $-$ α band unilateral fin line to obtain the broad-band transition taper shape. The unilateral fin line lies within a rectangular waveguide. Their optimization procedure produced, after 18 design

cycles, a K $-\alpha$ band unilateral fin line transition shape that is similar to exponential taper. In 2001, they used this approach to optimize the design of a two-dimensional parallel-plate waveguide antenna [20]. They applied 40 design cycles and found that reflected energy decreased over a broadband of frequencies.

Rickard, Georgieva, and Tam implemented absorbing boundary conditions (ABC's) for adjoint problems with a backwards time variable using the FDTD method in 2003 [21]. They found that the form of the ABC's for the adjoint backwards time problem is the same as for the original forward time problem, but that the sign before the spatial derivatives is opposite. The ABC's for the forward time problem were originally derived by Berenger in [22]. Rickard, Georgieva, and Tam's method was verified by comparing the reflections generated from solving the reverse-time adjoint problem for a microstrip line as compared to a forward-time method. They concluded that the reflections for the forward and backward time schemes are identical.

Chung, Ryu, Cheon, Park, and Hahn coupled the FETD method with design sensitivity analysis using the adjoint-variable method to obtain the optimal design of microwave devices in 2001 [23]. They took the curl of Maxwell's equations, and transformed them into the second-order wave equation. They then applied the FETD method to the second-order wave equation, and solved for the electric field that is assumed to be made up of one material property. They applied the adjoint-variable method to modify the design of a waveguide with a two-dimensional T-junction shape in 27 design cycles to obtain better performance over a broadband of frequencies.

Akcelik, Biros, Ghattas, Keyes, Ko, Lee, and Ng implemented a continuous-adjoint approach to design the shape of end caps of a low-loss cavity for the International Linear Collider in 2005 [24]. They numerically computed solutions to the Maxwell eigenvalues problem

which generates the magnetic field in the frequency domain with a finite-element method. In addition to solving the Maxwell eigenvalues problem, they had to solve an additional set of partial differential equations – an adjoint eigenvalue problem – for each function of interest. According to [24], each additional set of partial differential equations has "somewhat different structure from the original Maxwell eigenvalues problem and may require different discretizations, solvers, and preconditioners." They implemented their numerical software in the parallel processing paradigm with an effort to run on a large number of processors. They parameterized the shape of the end cells with analytic expressions and optimized the shape with regards to their cost function. Their goal for their specific application was to minimize a cost function that would improve the trapped energy in the end cell, and they do that by 58% while maintaining their constraint with 5 design cycles.

Georgieva, Glavic, Bakr, and Bandler implemented what they called a "feasible adjoint sensitivity technique" (FAST) for electromagnetic design optimization in the frequency domain in 2002 [25]. Their objective was to develop a versatile technique to extract sensitivities from any frequency domain solver, regardless of its discretization scheme. They attached FAST to a full-wave method of moments frequency-domain analysis tool and optimized the shape of a Yagi-Uda array and a regular patch antenna.

Nair and Webb implemented a higher-order finite-element method to numerically compute solutions to Maxwell's equations in the frequency domain, and simulated microwave devices over a frequency band in 2010 [26]. They employed an adaptive optimization procedure, called direct optimization [27], to locally increase accuracy by increasing mesh points and increasing the order of the finite elements. They chose a cost function that is directly related to scattering parameters. In 2001, Webb described a way to compute the design sensitivities using

high-order tetrahedral vector elements [28]. They simulated a 2-port rectangular waveguide with an E-plane bend over a frequency band of 8.25 GHz- 13.25 GHz with their higher-order adaptive optimizer procedure, and achieved their desired cost function five times faster than when an adaptive optimizer procedure that does not utilize higher-order elements was employed.

Toivanen, Makinen, Rahola, Jarvenpaa, and Yla-Oijala implemented a gradient based shape optimization scheme for ultra-wideband antennas in 2010 [29]. They used a discrete-adjoint approach to compute the sensitivity derivates. They computed derivatives of their field simulation software with automatic differentiation. Their field simulation software solves the electric field integral equation with the method of moments approach in the frequency domain. They noted that the method of moments approach generates a linear system that is a dense complex valued system. They parameterized the boundary of the antenna with B-splines. They used a radial basis function interpolation scheme to deform the mesh which does not have to solve an additional set of partial differential equations. Their cost function is the absolute value of the S_{11} scattering parameter squared, and their frequency band is 3-10 GHz. They spanned this band with 30 frequency sweeps, and optimized the antenna shape so that S_{11} is below -12 dB over the whole frequency band. Their optimized antenna shape can be generated from two initial configurations, and they concluded that their shape optimization procedure does not depend on initial shape.

With past computational explorations in mind, the objective of the current study is to develop simulation software to accurately approximate time-domain electromagnetic fields surrounding and within large complicated electromagnetic structures, and to develop numerical techniques to automatically optimize the shape of complicated electromagnetic structures. The time-domain electromagnetic fields are approximated with the spatially and temporally higher

order accurate SU/PG finite element scheme, and the shape sensitivity derivatives that are the crux of shape design optimization are computed with the time-accurate discrete-adjoint method. The algorithms are written in the message passing paradigm, and are capable of being executed on a distributed memory supercomputer. Implicit time stepping is employed, and a time step that is based on the physics of the problem is used.

CHAPTER II

ELECTROMAGNETIC FIELD SIMULATION SOFTWARE METHODOLOGY AND IMPLEMENTATION

2.1 Electromagnetic Field Simulation Software Formulation

The three-dimensional Maxwell's equations are numerically computed using the Streamline Upwind/Petrov-Galerkin finite element method in the time domain. This methodology provides a framework to numerically approximate Maxwell's equations that is higher order spatially and temporally, and is numerically stable on unstructured grids. Generally, complicated objects can be more accurately and more easily discretized on unstructured grids. Implicit backward difference time stepping is employed for this scheme. This, for the cost of solving a linear system at every time step, allows the use of a time step that is based on the physics of the problem instead of stability considerations. The field software can be executed on a distributed memory super-computer.

2.2 Governing Equations

The governing equations for the field simulation software are the six equations that make up Ampere's law with Maxwell's correction and Faraday's law. The solution of these equations generates the time history of the electric and magnetic fields within a volume. The six equations are a coupled set of linear, hyperbolic partial differential equations.

Maxwell's equation set in differential form [1]:

$$\nabla \times \vec{H} = J + \frac{\partial \vec{D}}{\partial t} \tag{2.1}$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.2}$$

$$\vec{D} = \varepsilon \vec{E} \tag{2.3}$$

$$\vec{B} = \mu \vec{H} \tag{2.4}$$

$$J = \sigma \vec{E} \tag{2.5}$$

$$\varepsilon = \varepsilon_r \varepsilon_o, \varepsilon_o \approx 8.854187817 \times 10^{-12} \ \frac{Farads}{Meter} \tag{2.6}$$

$$\mu = \mu_r \mu_o, \mu_o = 4\pi \times 10^{-7} \ \frac{Henries}{Meter} \tag{2.7}$$

$$c = \frac{1}{\sqrt{\varepsilon \mu}} \tag{2.8}$$

$$a_o = \sqrt{\frac{\varepsilon_o}{\mu_o}} \tag{2.9}$$

$$\vec{\bar{D}} = \frac{\vec{D}}{a_o} \tag{2.10}$$

Equations 2.1 and 2.2 are transformed to be solved for $\vec{\bar{D}}$ and $\vec{B}$ by defining the following quantities and by applying Equations 2.3, 2.4, and 2.10:

$$\vec{x}^* = \frac{\vec{x}}{L}, t^* = \frac{tc_o}{L}, \sigma^* = \sigma L \tag{2.11}$$

The * is dropped for convenience, and the governing equations (Equations 2.1 and 2.2) are solved in conservative, differential form as follows:

$$\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} + \frac{\partial \vec{H}}{\partial z} + \vec{S} = 0 \tag{2.12}$$

$$\vec{q}=\begin{bmatrix}\overline{D}_x\\ \overline{D}_y\\ \overline{D}_z\\ B_x\\ B_y\\ B_z\end{bmatrix},\ \vec{F}=\begin{bmatrix}0\\ B_z/\mu_r\\ -B_y/\mu_r\\ 0\\ -\overline{D}_z/\varepsilon_r\\ \overline{D}_y/\varepsilon_r\end{bmatrix},\ \vec{G}=\begin{bmatrix}-B_z/\mu_r\\ 0\\ B_x/\mu_r\\ \overline{D}_z/\varepsilon_r\\ 0\\ -\overline{D}_x/\varepsilon_r\end{bmatrix},\ \vec{H}=\begin{bmatrix}B_y/\mu_r\\ -B_x/\mu_r\\ 0\\ -\overline{D}_y/\varepsilon_r\\ \overline{D}_x/\varepsilon_r\\ 0\end{bmatrix},\ \vec{S}=\begin{bmatrix}\frac{\sigma}{\varepsilon_r a_o}\overline{D}_x\\ \frac{\sigma}{\varepsilon_r a_o}\overline{D}_y\\ \frac{\sigma}{\varepsilon_r a_o}\overline{D}_z\\ 0\\ 0\\ 0\end{bmatrix} \tag{2.13}$$

2.3 Finite Element Formulation

The first step of the finite element formulation is to consider the governing equations in weak form [3]:

$$\iiint_{\Omega} w\left(\frac{\partial\vec{q}}{\partial t}+\frac{\partial\vec{F}}{\partial x}+\frac{\partial\vec{G}}{\partial y}+\frac{\partial\vec{H}}{\partial z}+\vec{S}\right)=0 \tag{2.14}$$

The weighting function w is represented in Equation 2.15.

$$w=\sum_{i=1}^{N_{nodes}} N_i(\xi,\eta,\rho)d_i \tag{2.15}$$

In the above equation, $N(\xi,\eta,\rho)$ is the set of shape functions for a three-dimensional element, d_i is arbitrary, and N_{nodes} is the number of degrees of freedom (number of nodes) in the element. The shape functions are Lagrange polynomials. The application of Green's theorem to Equation 2.14 transforms the equation to Galerkin form. The Galerkin form of the governing equation is as follows:

$$\iiint_{\Omega} w\left(\frac{\partial \vec{q}}{\partial t}+\vec{S}\right)\partial\Omega+\iint_{\Gamma} w\left(\vec{F}n_x+\vec{G}n_y+\vec{H}n_z\right)\partial\Gamma-$$
$$\iiint_{\Omega}\left(\frac{\partial w}{\partial x}\vec{F}+\frac{\partial w}{\partial y}\vec{G}+\frac{\partial w}{\partial z}\vec{H}\right)\partial\Omega=0 \tag{2.16}$$

Solving this hyperbolic equation set in Galerkin form is unstable for coarse grids because it does not provide sufficient dissipation to prevent the field variables from oscillating, and is equivalent to solving the equation set with central differencing. A generally stable way to discretize a hyperbolic equation is to use upwinding. In order to stabilize the finite element approximation method, the Streamline Upwind/Petrov-Galerkin method is utilized [6]. The Petrov-Galerkin method adds a stream-wise upwind stabilization term to the weighting function [6], and is shown in Equation 2.17 as $\left(\frac{\partial w}{\partial x}A+\frac{\partial w}{\partial y}B+\frac{\partial w}{\partial z}C\right)\tau$. The Petrov-Galerkin weak form of the governing equations is:

$$\iiint_{\Omega}\left[w+\left(\frac{\partial w}{\partial x}A+\frac{\partial w}{\partial y}B+\frac{\partial w}{\partial z}C\right)\tau\right]\left[\frac{\partial \vec{q}}{\partial t}+\frac{\partial \vec{F}}{\partial x}+\frac{\partial \vec{G}}{\partial y}+\frac{\partial \vec{H}}{\partial z}+\vec{S}\right]=0 \tag{2.17}$$

After applying Green's theorem, this statement becomes:

$$\iiint_{\Omega}\left[w+\left(\frac{\partial w}{\partial x}A+\frac{\partial w}{\partial y}B+\frac{\partial w}{\partial z}C\right)\tau\right]\left(\frac{\partial \vec{q}}{\partial t}+\vec{S}\right)\partial\Omega+$$
$$\iint_{\Gamma} w\left(\vec{F}n_x+\vec{G}n_y+\vec{H}n_z\right)\partial\Gamma-\iiint_{\Omega}\left(\frac{\partial w}{\partial x}\vec{F}+\frac{\partial w}{\partial y}\vec{G}+\frac{\partial w}{\partial z}\vec{H}\right)\partial\Omega+ \tag{2.18}$$
$$\iiint_{\Omega}\left[\left(\frac{\partial w}{\partial x}A+\frac{\partial w}{\partial y}B+\frac{\partial w}{\partial z}C\right)\tau\right]\left[\frac{\partial \vec{F}}{\partial x}+\frac{\partial \vec{G}}{\partial y}+\frac{\partial \vec{H}}{\partial z}\right]\partial\Omega=0$$

$$[A]=\left[\partial\vec{F}/\partial\vec{q}\right],[B]=\left[\partial\vec{G}/\partial\vec{q}\right],[C]=\left[\partial\vec{H}/\partial\vec{q}\right] \tag{2.19}$$

The matrices $A, B,$ and C are as follows:

$$A=\begin{bmatrix}0&0&0&0&0&0\\0&0&0&0&0&\frac{1}{\mu_r}\\0&0&0&0&\frac{-1}{\mu_r}&0\\0&0&0&0&0&0\\0&0&\frac{-1}{\varepsilon_r}&0&0&0\\0&\frac{1}{\varepsilon_r}&0&0&0&0\end{bmatrix} \tag{2.20}$$

$$B=\begin{bmatrix}0&0&0&0&0&\frac{-1}{\mu_r}\\0&0&0&0&0&0\\0&0&0&\frac{1}{\mu_r}&0&0\\0&0&\frac{1}{\varepsilon_r}&0&0&0\\0&0&0&0&0&0\\\frac{-1}{\varepsilon_r}&0&0&0&0&0\end{bmatrix} \tag{2.21}$$

$$C=\begin{bmatrix}0&0&0&0&\frac{1}{\mu_r}&0\\0&0&0&\frac{-1}{\mu_r}&0&0\\0&0&0&0&0&0\\0&\frac{-1}{\varepsilon_r}&0&0&0&0\\\frac{1}{\varepsilon_r}&0&0&0&0&0\\0&0&0&0&0&0\end{bmatrix} \tag{2.22}$$

2.4 Parent Element: Gaussian Quadrature, Shape Functions and Derivatives, Element Jacobian

The tetrahedra present within the computational grid are mapped to parent tetrahedra for integration. The linear parent tetrahedron has the coordinates in non-dimensional (ξ, η, ρ) space of (0,0,0), (1,0,0), (0,1,0), and (0,0,1). There are specified quadrature rules for integrating a

function over a parent tetrahedron which are traditionally tabulated such that the weights sum to one. A quadrature table is available in [30]. A function can be integrated over a tetrahedron with the following relation:

$$\iiint_{\Omega} f(x,y,z)d\Omega = \frac{1}{6}\sum_{i=1}^{N_{gauss}} f_i(x(\xi,\eta,\rho), y(\xi,\eta,\rho), z(\xi,\eta,\rho))\, W_i \mathcal{J} \tag{2.23}$$

For this thesis, in the above equation (ξ,η,ρ) are Gauss points given in a table in [31], W_i are Gauss weights given in the same table, and $\mathcal{J}$ is the Jacobian. As long as the quadrature scheme is of higher order accuracy than the polynomial in the integrand, the integration is exact to machine precision. However, if higher order curved elements are employed the Jacobian is also a polynomial, and the integrand is made up of a rational polynomial. For the parent tetrahedron, the shape functions $N(\xi,\eta,\rho)$ are Lagrange polynomials written in terms of the isoparametric coordinates (ξ,η,ρ) [4]. The number of shape functions matches the number of nodes in the element. A linear tetrahedral element has four nodes, a quadratic tetrahedral element has ten nodes, and a cubic tetrahedral element has twenty nodes. Each of these elements is shown in the following Figures 2.1, 2.2, and 2.3 [4].

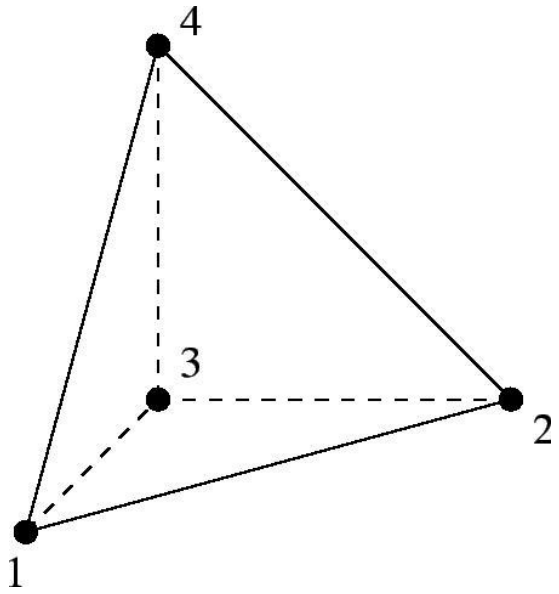

Figure 2.1 Linear Tetrahedron

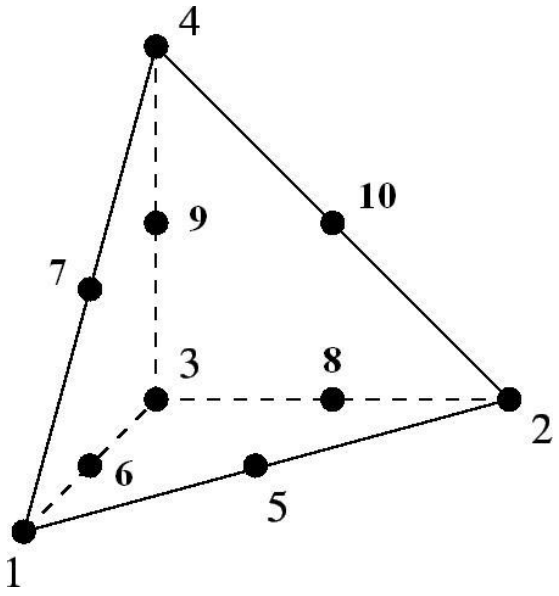

Figure 2.2 Quadratic Tetrahedron

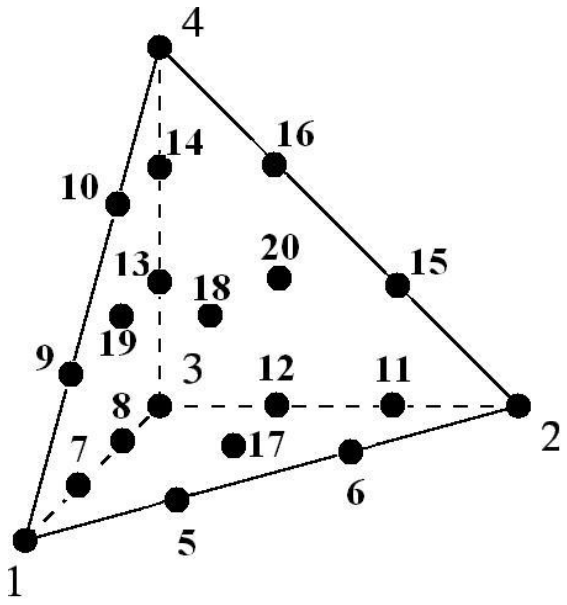

Figure 2.3 Cubic Tetrahedron

The Lagrangian shape functions and their derivatives for the isoparametric linear tetrahedron are [4]:

$$N(\xi,\eta,\rho) = \begin{bmatrix} 1-\xi-\eta-\rho \\ \xi \\ \eta \\ \rho \end{bmatrix} \tag{2.24}$$

$$\frac{\partial N(\xi,\eta,\rho)}{\partial \xi} = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.25}$$

$$\frac{\partial N(\xi,\eta,\rho)}{\partial \eta} = \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{2.26}$$

$$\frac{\partial N(\xi,\eta,\rho)}{\partial \rho} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.27}$$

The shape functions for the higher order tetrahedra can be found in [4]. For a given element, if information is known at the nodes it can be computed for the element by summing the multiplication of the value of the variable by the value of the shape function over all the nodes in the element at a given Gauss point [3]. For instance, the values of $x, y,$ and z for the element are evaluated by computing Equations 2.28-2.30.

$$x = \sum_{i=1}^{N_{nodes}} N_i x_i \tag{2.28}$$

$$y = \sum_{i=1}^{N_{nodes}} N_i y_i \tag{2.29}$$

$$z = \sum_{i=1}^{N_{nodes}} N_i z_i \tag{2.30}$$

In the above equations, $x_i, y_i,$ and z_i are the values at the nodes. In order to compute the Jacobian of the element, the following matrix must be computed [3]:

$$\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \rho} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} & \frac{\partial y}{\partial \rho} \\ \frac{\partial z}{\partial \xi} & \frac{\partial z}{\partial \eta} & \frac{\partial z}{\partial \rho} \end{bmatrix} \tag{2.31}$$

Each variable in the above matrix is computed as follows [3]:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} x_i \tag{2.32}$$

$$\frac{\partial x}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} x_i \tag{2.33}$$

$$\frac{\partial x}{\partial \rho} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \rho} x_i \tag{2.34}$$

$$\frac{\partial y}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} y_i \tag{2.35}$$

$$\frac{\partial y}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} y_i \tag{2.36}$$

$$\frac{\partial y}{\partial \rho} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \rho} y_i \tag{2.37}$$

$$\frac{\partial z}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} z_i \tag{2.38}$$

$$\frac{\partial z}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} z_i \tag{2.39}$$

$$\frac{\partial z}{\partial \rho} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \rho} z_i \tag{2.40}$$

The Jacobian is defined as [3]:

$$\mathcal{J} = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial z}{\partial \rho} + \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \rho} + \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \rho} - \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \rho} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\frac{\partial z}{\partial \rho} - \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \rho} \tag{2.41}$$

In order to compute $N_x, N_y,$ and N_z at each Gauss point of the element Equation 2.31 is inverted to generate the following matrix:

$$\begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} & \frac{\partial \xi}{\partial z} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} & \frac{\partial \eta}{\partial z} \\ \frac{\partial \rho}{\partial x} & \frac{\partial \rho}{\partial y} & \frac{\partial \rho}{\partial z} \end{bmatrix} = \frac{1}{\mathcal{J}} \begin{bmatrix} \frac{\partial z}{\partial \rho}\frac{\partial y}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \rho} & \frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \rho} - \frac{\partial z}{\partial \rho}\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \rho}\frac{\partial x}{\partial \eta} - \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \rho} \\ \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \rho} - \frac{\partial z}{\partial \rho}\frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \rho}\frac{\partial x}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \rho} & \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \rho} - \frac{\partial y}{\partial \rho}\frac{\partial x}{\partial \xi} \\ \frac{\partial z}{\partial \eta}\frac{\partial y}{\partial \xi} - \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta} - \frac{\partial z}{\partial \eta}\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta} \end{bmatrix} \tag{2.42}$$

Once Equation 2.42 is evaluated the values of $N_x, N_y,$ and N_z are computed at a given node that is interpolated to a Gauss point of the element as follows [3]:

$$\frac{\partial N_i}{\partial x} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial x} + \frac{\partial N_i}{\partial \rho}\frac{\partial \rho}{\partial x} \tag{2.43}$$

$$\frac{\partial N_i}{\partial y} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial y} + \frac{\partial N_i}{\partial \rho}\frac{\partial \rho}{\partial y} \tag{2.44}$$

$$\frac{\partial N_i}{\partial z} = \frac{\partial N_i}{\partial \xi}\frac{\partial \xi}{\partial z} + \frac{\partial N_i}{\partial \eta}\frac{\partial \eta}{\partial z} + \frac{\partial N_i}{\partial \rho}\frac{\partial \rho}{\partial z} \tag{2.45}$$

2.5 Derivation of Stabilization Matrix

The element based six by six stabilization matrix τ has dimensions of time and is defined as [31]:

$$\tau = \left(\sum_{i=1}^{N_{nodes}} \left|\frac{\partial N_i}{\partial x_j} A_j\right|\right)^{-1}, j = 1, 2, 3 \tag{2.46}$$

For a linear element, the stabilization matrix is of the form:

$$\tau = \left(\begin{matrix} \left|\frac{\partial N_1}{\partial x}A + \frac{\partial N_1}{\partial y}B + \frac{\partial N_1}{\partial z}C\right| + \left|\frac{\partial N_2}{\partial x}A + \frac{\partial N_2}{\partial y}B + \frac{\partial N_2}{\partial z}C\right| + \\ \left|\frac{\partial N_3}{\partial x}A + \frac{\partial N_3}{\partial y}B + \frac{\partial N_3}{\partial z}C\right| + \left|\frac{\partial N_4}{\partial x}A + \frac{\partial N_4}{\partial y}B + \frac{\partial N_4}{\partial z}C\right| \end{matrix}\right)^{-1} \tag{2.47}$$

Since the above equation contains the absolute value function, each term in the above equation of the form $\left|\frac{\partial N}{\partial x}A+\frac{\partial N}{\partial y}B+\frac{\partial N}{\partial z}C\right|$ is quantified by evaluating the eigensystem. In the following equations the derivatives of the shape function are written in a condensed form ($\frac{\partial N}{\partial x}$ is N_x). A representative term $\left|N_xA + N_yB + N_zC\right|$ from the stabilization matrix (Equation 2.47) is evaluated as follows:

$$\left|\frac{\partial N}{\partial x}A + \frac{\partial N}{\partial y}B + \frac{\partial N}{\partial z}C\right| =$$

$$\begin{vmatrix} 0 & 0 & 0 & 0 & {N_z}/{\mu_r} & {-N_y}/{\mu_r} \\ 0 & 0 & 0 & {-N_z}/{\mu_r} & 0 & {N_x}/{\mu_r} \\ 0 & 0 & 0 & {N_y}/{\mu_r} & {-N_x}/{\mu_r} & 0 \\ 0 & {-N_z}/{\varepsilon_r} & {N_y}/{\varepsilon_r} & 0 & 0 & 0 \\ {N_z}/{\varepsilon_r} & 0 & {-N_x}/{\varepsilon_r} & 0 & 0 & 0 \\ {-N_y}/{\varepsilon_r} & {N_x}/{\varepsilon_r} & 0 & 0 & 0 & 0 \end{vmatrix} = T|\lambda|T^{-1} \qquad (2.48)$$

In the above equation, T is the matrix of right eigenvectors, λ is the matrix of the eigenvalues, and T^{-1} is the matrix of left eigenvectors. The eigenvalues are derived from

$det\left(\frac{\partial N}{\partial x}A + \frac{\partial N}{\partial y}B + \frac{\partial N}{\partial z}C - \lambda I\right) = 0$ as follows:

$$det\begin{pmatrix} -\lambda & 0 & 0 & 0 & N_z/\mu_r & -N_y/\mu_r \\ 0 & -\lambda & 0 & -N_z/\mu_r & 0 & N_x/\mu_r \\ 0 & 0 & -\lambda & N_y/\mu_r & -N_x/\mu_r & 0 \\ 0 & -N_z/\varepsilon_r & N_y/\varepsilon_r & -\lambda & 0 & 0 \\ N_z/\varepsilon_r & 0 & -N_x/\varepsilon_r & 0 & -\lambda & 0 \\ -N_y/\varepsilon_r & N_x/\varepsilon_r & 0 & 0 & 0 & -\lambda \end{pmatrix} = 0 \tag{2.49}$$

$$\lambda = \begin{bmatrix} 0 \\ C_r\sqrt{N_x^2 + N_y^2 + N_z^2} \\ -C_r\sqrt{N_x^2 + N_y^2 + N_z^2} \\ 0 \\ C_r\sqrt{N_x^2 + N_y^2 + N_z^2} \\ -C_r\sqrt{N_x^2 + N_y^2 + N_z^2} \end{bmatrix} \tag{2.50}$$

The right eigenvectors $\vec{r}$ that make up the right eigenvector matrix T are defined as follows:

$$\left(\left(N_xA + N_yB + N_zC\right) - \lambda\vec{I}\right)\vec{r} = 0 \tag{2.51}$$

Each one dimensional array, $\vec{r}$, is six values long. However, there are only four distinct equations for six unknowns. Hence, two of the values of $\vec{r}$ are specified. The right eigenvector matrix is:

$$T = [\vec{r}_1 \quad \vec{r}_2 \quad \vec{r}_3 \quad \vec{r}_4 \quad \vec{r}_5 \quad \vec{r}_6] \tag{2.52}$$

$$\vec{r}_1 = \begin{bmatrix} N_x \\ N_y \\ N_z \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{r}_2 = \begin{bmatrix} N_x N_y \\ -N_x^2 - N_z^2 \\ N_y N_z \\ N_z \mu_r c_r \sqrt{N_x^2 + N_y^2 + N_z^2} \\ 0 \\ -N_x \mu_r c_r \sqrt{N_x^2 + N_y^2 + N_z^2} \end{bmatrix}, \vec{r}_3 = \begin{bmatrix} N_x N_y \\ -N_x^2 - N_z^2 \\ N_y N_z \\ -N_z \mu_r c_r \sqrt{N_x^2 + N_y^2 + N_z^2} \\ 0 \\ N_x \mu_r c_r \sqrt{N_x^2 + N_y^2 + N_z^2} \end{bmatrix} \quad (2.53)$$

$$\vec{r}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ N_x \\ N_y \\ N_z \end{bmatrix}, \vec{r}_5 = \begin{bmatrix} \frac{N_y}{\mu_r c_r \sqrt{N_x^2+N_y^2+N_z^2}} \\ \frac{-N_x}{\mu_r c_r \sqrt{N_x^2+N_y^2+N_z^2}} \\ 0 \\ \frac{N_x N_z}{(N_x^2+N_y^2+N_z^2)} \\ \frac{N_y N_z}{(N_x^2+N_y^2+N_z^2)} \\ \frac{N_z^2}{(N_x^2+N_y^2+N_z^2)} - 1 \end{bmatrix}, \vec{r}_6 = \begin{bmatrix} \frac{N_y}{\mu_r c_r \sqrt{N_x^2+N_y^2+N_z^2}} \\ \frac{-N_x}{\mu_r c_r \sqrt{N_x^2+N_y^2+N_z^2}} \\ 0 \\ \frac{-N_x N_z}{(N_x^2+N_y^2+N_z^2)} \\ \frac{-N_y N_z}{(N_x^2+N_y^2+N_z^2)} \\ 1 - \frac{N_z^2}{(N_x^2+N_y^2+N_z^2)} \end{bmatrix} \quad (2.54)$$

The left eigenvectors $\vec{l}$ that make up the left eigenvector matrix T^{-1} are defined as follows:

$$\left(\left(N_x A + N_y B + N_z C \right)^T - \lambda \vec{I} \right) \vec{l} = 0 \quad (2.55)$$

The left eigenvector matrix is:

$$T^{-1} = \begin{bmatrix} \vec{l}_1^T \\ \vec{l}_2^T \\ \vec{l}_3^T \\ \vec{l}_4^T \\ \vec{l}_5^T \\ \vec{l}_6^T \end{bmatrix} \quad (2.56)$$

In the following equations, $(N^2) = \left(N_x^2 + N_y^2 + N_z^2 \right)$.

$$\vec{l}_1^T = \left[\frac{N_x}{(N^2)} \quad \frac{N_y}{(N^2)} \quad \frac{N_z}{(N^2)} \quad 0 \quad 0 \quad 0\right] \tag{2.57}$$

$$\vec{l}_2^T = \left[\frac{-N_x}{2N_y(N^2)} \quad \frac{-1}{2(N^2)} \quad \frac{N_x^2+N_y^2}{2N_yN_z(N^2)} \quad \frac{1}{2c_r\mu_rN_z\sqrt{(N^2)}} \quad \frac{-N_x}{2c_r\mu_rN_yN_z\sqrt{(N^2)}} \quad 0\right] \tag{2.58}$$

$$\vec{l}_3^T = \left[\frac{-N_x}{2N_y(N^2)} \quad \frac{-1}{2(N^2)} \quad \frac{N_x^2+N_y^2}{2N_yN_z(N^2)} \quad \frac{-1}{2c_r\mu_rN_z\sqrt{(N^2)}} \quad \frac{N_x}{2c_r\mu_rN_yN_z\sqrt{(N^2)}} \quad 0\right] \tag{2.59}$$

$$\vec{l}_4^T = \left[0 \quad 0 \quad 0 \quad \frac{N_x}{(N^2)} \quad \frac{N_y}{(N^2)} \quad \frac{N_z}{(N^2)}\right] \tag{2.60}$$

$$\vec{l}_5^T = \left[\frac{\mu_rc_r\sqrt{(N^2)}}{2N_y} \quad 0 \quad \frac{-N_x\mu_rc_r\sqrt{(N^2)}}{2N_yN_z} \quad \frac{-N_x}{2N_z} \quad \frac{N_x^2+N_z^2}{2N_yN_z} \quad \frac{-1}{2}\right] \tag{2.61}$$

$$\vec{l}_6^T = \left[\frac{\mu_rc_r\sqrt{(N^2)}}{2N_y} \quad 0 \quad \frac{-N_x\mu_rc_r\sqrt{(N^2)}}{2N_yN_z} \quad \frac{N_x}{2N_z} \quad \frac{-N_x^2-N_z^2}{2N_yN_z} \quad \frac{1}{2}\right] \tag{2.62}$$

The representative matrix $\left|N_xA + N_yB + N_zC\right|$ or $T|\lambda|T^{-1}$ is shown in Equation 2.63.

$$\begin{bmatrix}
\frac{C_r(N_y^2+N_z^2)}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_xN_y}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_xN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & 0 & 0 & 0 \\
\frac{-C_rN_xN_y}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{C_r(N_x^2+N_z^2)}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_yN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & 0 & 0 & 0 \\
\frac{-C_rN_xN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_yN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{C_r(N_x^2+N_y^2)}{\sqrt{N_x^2+N_y^2+N_z^2}} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{C_r(N_y^2+N_z^2)}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_xN_y}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_xN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} \\
0 & 0 & 0 & \frac{-C_rN_xN_y}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{C_r(N_x^2+N_z^2)}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_yN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} \\
0 & 0 & 0 & \frac{-C_rN_xN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{-C_rN_yN_z}{\sqrt{N_x^2+N_y^2+N_z^2}} & \frac{C_r(N_x^2+N_y^2)}{\sqrt{N_x^2+N_y^2+N_z^2}}
\end{bmatrix} \tag{2.63}$$

The stabilization matrix τ is the inverse of the summation of the above equation at each node for a given Gauss point. The three by three nonzero matrices in Equation 2.63 are denoted as Ψ, and are inverted analytically with Kramer's rule as follows [32]:

$$\Psi = \begin{bmatrix} \psi_{11} & \psi_{12} & \psi_{13} \\ \psi_{21} & \psi_{22} & \psi_{23} \\ \psi_{31} & \psi_{32} & \psi_{33} \end{bmatrix} \tag{2.64}$$

$$\Psi^{-1} = \frac{1}{|\Psi|} \begin{bmatrix} \begin{vmatrix} \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{33} \end{vmatrix} & \begin{vmatrix} \psi_{13} & \psi_{12} \\ \psi_{33} & \psi_{32} \end{vmatrix} & \begin{vmatrix} \psi_{12} & \psi_{13} \\ \psi_{22} & \psi_{23} \end{vmatrix} \\ \begin{vmatrix} \psi_{23} & \psi_{21} \\ \psi_{33} & \psi_{31} \end{vmatrix} & \begin{vmatrix} \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{33} \end{vmatrix} & \begin{vmatrix} \psi_{22} & \psi_{23} \\ \psi_{32} & \psi_{33} \end{vmatrix} \\ \begin{vmatrix} \psi_{21} & \psi_{22} \\ \psi_{31} & \psi_{32} \end{vmatrix} & \begin{vmatrix} \psi_{12} & \psi_{11} \\ \psi_{32} & \psi_{31} \end{vmatrix} & \begin{vmatrix} \psi_{11} & \psi_{12} \\ \psi_{21} & \psi_{22} \end{vmatrix} \end{bmatrix} \tag{2.65}$$

2.6 Finite Element Implementation

The finite element method is implemented by applying Equation 2.18 to each element within the discretized electromagnetic field. This procedure generates a residual vector that is number of nodes long, and has six equations at each node. Equation 2.18 can be broken up into four terms ($\mathcal{A}, \mathcal{B}, \mathcal{C}$, and $\mathcal{D}$) as follows:

$$\mathcal{A} = \iiint_{\Omega} \left[w + \left(\frac{\partial w}{\partial x} A + \frac{\partial w}{\partial y} B + \frac{\partial w}{\partial z} C \right) \tau \right] \left(\frac{\partial \vec{q}}{\partial t} + \vec{S} \right) \partial\Omega \tag{2.66}$$

$$\mathcal{B} = - \iiint_{\Omega} \left(\frac{\partial w}{\partial x} \vec{F} + \frac{\partial w}{\partial y} \vec{G} + \frac{\partial w}{\partial z} \vec{H} \right) \partial\Omega \tag{2.67}$$

$$\mathcal{C} = \iiint_{\Omega} \left[w + \left(\frac{\partial w}{\partial x} A + \frac{\partial w}{\partial y} B + \frac{\partial w}{\partial z} C \right) \tau \right] \left[\frac{\partial \vec{F}}{\partial x} + \frac{\partial \vec{G}}{\partial y} + \frac{\partial \vec{H}}{\partial z} \right] \partial\Omega \tag{2.68}$$

$$\mathcal{D} = \iint_{\Gamma} w \left(\vec{F} n_x + \vec{G} n_y + \vec{H} n_z \right) \partial\Gamma \tag{2.69}$$

To evaluate the integrals in Equations 2.66-2.68, the $\vec{q}$ values, which are stored at the nodes of an element, are interpolated to the Gauss points by the following procedure [3]:

$$\vec{q} = \sum_{i=1}^{N_{nodes}} N_i \overrightarrow{q_i} \tag{2.70}$$

$\vec{F}, \vec{G}$, and $\vec{H}$ are computed at the Gauss points in the following manner:

$$\vec{F} = [A]\vec{q} \tag{2.71}$$

$$\vec{G} = [B]\vec{q} \tag{2.72}$$

$$\vec{H} = [C]\vec{q} \tag{2.73}$$

The derivatives of $\vec{q}$ with respect to $x, y,$ and z are computed at a Gauss point as follows [3]:

$$\frac{\partial \vec{q}}{\partial x} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial x} \overrightarrow{q_i} \tag{2.74}$$

$$\frac{\partial \vec{q}}{\partial y} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial y} \overrightarrow{q_i} \tag{2.75}$$

$$\frac{\partial \vec{q}}{\partial z} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial z} \overrightarrow{q_i} \tag{2.76}$$

The derivatives of $\vec{F}, \vec{G},$ and $\vec{H}$ with respect to $x, y,$ and z are computed at a Gauss point as follows:

$$\frac{\partial \vec{F}}{\partial x} = [A]\frac{\partial \vec{q}}{\partial x} \tag{2.77}$$

$$\frac{\partial \vec{G}}{\partial y} = [B]\frac{\partial \vec{q}}{\partial y} \tag{2.78}$$

$$\frac{\partial \vec{H}}{\partial z} = [C]\frac{\partial \vec{q}}{\partial z} \tag{2.79}$$

The integrand for a given node, $i,$ within the element has six equations (Maxwell's equations), and is made up of four terms. The integrand for terms $\mathcal{A} - \mathcal{C}$ (Equations 2.66-2.68) is shown in Equations 2.80-2.82 for the Maxwell equation $j.$ For convenience, the matrix A times τ is denoted $\bar{A}$, the matrix B times τ is denoted $\bar{B}$, and the matrix C times τ is denoted $\bar{C}$.

$$\mathcal{A}_i^j = N_i \frac{\partial \overrightarrow{q_J}}{\partial t} + \left(\frac{\partial N_i}{\partial x} \bar{A}(j,1) + \frac{\partial N_i}{\partial y} \bar{B}(j,1) + \frac{\partial N_i}{\partial z} \bar{C}(j,1) \right) \frac{\partial \overline{D_x}}{\partial t}$$

$$+ \left(\frac{\partial N_i}{\partial x} \bar{A}(j,2) + \frac{\partial N_i}{\partial y} \bar{B}(j,2) + \frac{\partial N_i}{\partial z} \bar{C}(j,2) \right) \frac{\partial \overline{D_y}}{\partial t}$$

$$+ \left(\frac{\partial N_i}{\partial x} \bar{A}(j,3) + \frac{\partial N_i}{\partial y} \bar{B}(j,3) + \frac{\partial N_i}{\partial z} \bar{C}(j,3) \right) \frac{\partial \overline{D_z}}{\partial t}$$

$$+ \left(\frac{\partial N_i}{\partial x} \bar{A}(j,4) + \frac{\partial N_i}{\partial y} \bar{B}(j,4) + \frac{\partial N_i}{\partial z} \bar{C}(j,4) \right) \frac{\partial B_x}{\partial t}$$

$$+ \left(\frac{\partial N_i}{\partial x} \bar{A}(j,5) + \frac{\partial N_i}{\partial y} \bar{B}(j,5) + \frac{\partial N_i}{\partial z} \bar{C}(j,5) \right) \frac{\partial B_y}{\partial t}$$

$$+ \left(\frac{\partial N_i}{\partial x} \bar{A}(j,6) + \frac{\partial N_i}{\partial y} \bar{B}(j,6) + \frac{\partial N_i}{\partial z} \bar{C}(j,6) \right) \frac{\partial B_z}{\partial t}$$

$$+ \left(N_i + \frac{\partial N_i}{\partial x} \bar{A}(j,j) + \frac{\partial N_i}{\partial y} \bar{B}(j,j) + \frac{\partial N_i}{\partial z} \bar{C}(j,j) \right) S_j \tag{2.80}$$

$$\mathcal{B}_i^j = -\left(\frac{\partial N_i}{\partial x} F_j + \frac{\partial N_i}{\partial y} G_j + \frac{\partial N_i}{\partial z} H_j \right) \tag{2.81}$$

$$C_i^j = \left(\frac{\partial N_i}{\partial x}\bar{A}(j,1) + \frac{\partial N_i}{\partial y}\bar{B}(j,1) + \frac{\partial N_i}{\partial z}\bar{C}(j,1)\right)\left(\frac{\partial F_1}{\partial x} + \frac{\partial G_1}{\partial y} + \frac{\partial H_1}{\partial z}\right)$$
$$+\left(\frac{\partial N_i}{\partial x}\bar{A}(j,2) + \frac{\partial N_i}{\partial y}\bar{B}(j,2) + \frac{\partial N_i}{\partial z}\bar{C}(j,2)\right)\left(\frac{\partial F_2}{\partial x} + \frac{\partial G_2}{\partial y} + \frac{\partial H_2}{\partial z}\right)$$
$$+\left(\frac{\partial N_i}{\partial x}\bar{A}(j,3) + \frac{\partial N_i}{\partial y}\bar{B}(j,3) + \frac{\partial N_i}{\partial z}\bar{C}(j,3)\right)\left(\frac{\partial F_3}{\partial x} + \frac{\partial G_3}{\partial y} + \frac{\partial H_3}{\partial z}\right)$$
$$+\left(\frac{\partial N_i}{\partial x}\bar{A}(j,4) + \frac{\partial N_i}{\partial y}\bar{B}(j,4) + \frac{\partial N_i}{\partial z}\bar{C}(j,4)\right)\left(\frac{\partial F_4}{\partial x} + \frac{\partial G_4}{\partial y} + \frac{\partial H_4}{\partial z}\right)$$
$$+\left(\frac{\partial N_i}{\partial x}\bar{A}(j,5) + \frac{\partial N_i}{\partial y}\bar{B}(j,5) + \frac{\partial N_i}{\partial z}\bar{C}(j,5)\right)\left(\frac{\partial F_5}{\partial x} + \frac{\partial G_5}{\partial y} + \frac{\partial H_5}{\partial z}\right)$$
$$+\left(\frac{\partial N_i}{\partial x}\bar{A}(j,6) + \frac{\partial N_i}{\partial y}\bar{B}(j,6) + \frac{\partial N_i}{\partial z}\bar{C}(j,6)\right)\left(\frac{\partial F_6}{\partial x} + \frac{\partial G_6}{\partial y} + \frac{\partial H_6}{\partial z}\right) \tag{2.82}$$

The derivative of the $\vec{q}$ values with respect to time is approximated with a stable second order backward finite difference equation as follows [33]:

$$\frac{\partial \vec{q}}{\partial t} = \frac{3\vec{q}^{n+1} - 4\vec{q}^{n} + \vec{q}^{n-1}}{2\Delta t} + O(\Delta t^2) \tag{2.83}$$

2.7 Boundary Conditions

The boundary conditions are applied by modifying the fluxes when evaluating Equation 2.69. Three types of boundary conditions are implemented for this solution procedure. These are perfect electric conducting (PEC) walls, material jump conditions, and Dirichlet boundary conditions.

Triangular face elements are used to describe the boundaries. Similar to the tetrahedra, the triangles are mapped to parent space. The linear isoparametric triangle has the coordinates in non-dimensional (ξ, η) space of (0,0), (1,0), and (0,1). As with the isoparametric tetrahedra, Gaussian quadrature rules are employed for the integration of a function over a triangle. The quadrature table is available in [34]. A function can be integrated over a triangle with the following relation:

$$\iint_{\Gamma} f(x,y,z)\,\partial\Gamma = \frac{1}{2}\sum_{i=1}^{N_{gauss}} f_i(x(\xi,\eta), y(\xi,\eta), z(\xi,\eta))\, W_i \mathcal{J} \tag{2.84}$$

In Equation 2.84, (ξ_i, η_i) are Gauss points, W_i are associated Gauss weights, and $\mathcal{J}$ is the Jacobian. For the isoparametric triangles, the shape functions $N(\xi, \eta)$ are Lagrange polynomials written in terms of the parental coordinates (ξ, η). In order to obtain the element unit normal vector and Jacobian, the terms $\frac{\partial x}{\partial \xi}$, $\frac{\partial x}{\partial \eta}$, $\frac{\partial y}{\partial \xi}$, $\frac{\partial y}{\partial \eta}$, $\frac{\partial z}{\partial \xi}$ and $\frac{\partial z}{\partial \eta}$ are computed. These terms are computed as follows [3]:

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} x_i \tag{2.85}$$

$$\frac{\partial x}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} x_i \tag{2.86}$$

$$\frac{\partial y}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} y_i \tag{2.87}$$

$$\frac{\partial y}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} y_i \tag{2.88}$$

$$\frac{\partial z}{\partial \xi} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \xi} z_i \tag{2.89}$$

$$\frac{\partial z}{\partial \eta} = \sum_{i=1}^{N_{nodes}} \frac{\partial N_i}{\partial \eta} z_i \tag{2.90}$$

The components of the non-normalized element normal vector are [35]:

$$\overline{n_x} = \frac{\partial z}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial z}{\partial \eta}, \quad \overline{n_y} = \frac{\partial x}{\partial \xi}\frac{\partial z}{\partial \eta} - \frac{\partial z}{\partial \xi}\frac{\partial x}{\partial \eta}, \quad \overline{n_z} = \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta} - \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} \tag{2.91}$$

The Jacobian, $\mathcal{J}$, is the square root of the sum of the squares of the non-normalized element normal vector which is shown as:

$$\mathcal{J} = \sqrt{\overline{n_x}^2 + \overline{n_y}^2 + \overline{n_z}^2} \tag{2.92}$$

The normalized element normal vector is the non-normalized element normal vector divided by the Jacobian which is shown as:

$$n_x = \frac{\overline{n_x}}{\mathcal{J}}, \quad n_y = \frac{\overline{n_y}}{\mathcal{J}}, \quad n_z = \frac{\overline{n_z}}{\mathcal{J}} \tag{2.93}$$

In Equation 2.69 $\vec{F}, \vec{G}$, and $\vec{H}$ are computed from Equations 2.71-2.73.

For a PEC boundary, $\nabla \times \vec{E} = 0$ and $\vec{B}$ is solved for [1]. This is enforced by solving Equation 2.69 with $\vec{F}, \vec{G}$, and $\vec{H}$ as:

$$\vec{F} = \begin{bmatrix} 0 \\ B_z/\mu_r \\ -B_y/\mu_r \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{G} = \begin{bmatrix} -B_z/\mu_r \\ 0 \\ B_x/\mu_r \\ 0 \\ 0 \\ 0 \end{bmatrix}, \vec{H} = \begin{bmatrix} B_y/\mu_r \\ -B_x/\mu_r \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.94}$$

For the material jump boundary condition, Equation 2.69 is solved by using Roe's scheme, which is a flux-difference splitting scheme that is designed to solve the Riemann problem. Roe's

scheme is implemented by Equation 2.95 [36]. Anderson et al. [8] applied an identical procedure to evaluate the material jump boundary condition.

$$\vec{F}n_x + \vec{G}n_y + \vec{H}n_z = \hat{F} = \frac{1}{2}\left(\hat{F}(\overrightarrow{w}_L) + \hat{F}(\overrightarrow{w}_R) - \left|n_x\tilde{A} + n_y\tilde{B} + n_z\tilde{C}\right|\widetilde{M}\Delta\overrightarrow{w}\right) \tag{2.95}$$

In order to implement this scheme each node on the boundary is duplicated and denoted as either a left state (L) or a right state (R). The average of the left and right states is denoted by $\sim$. In order to compute $\left|n_x\tilde{A} + n_y\tilde{B} + n_z\tilde{C}\right|$ the eigensystem derived before (Equation 2.63) is employed with two modifications. The first is the derivatives of the shape functions $N_x, N_y,$ and N_z are replaced with the normalized unit normal vector components $n_x, n_y,$ and n_z, and the second is the reference speed of light c_r is replaced with $\tilde{c_r}$. The terms $\overrightarrow{w}$ and $\widetilde{M}$ are computed as follows:

$$\overrightarrow{w} = \begin{bmatrix} \frac{c_o\overline{D_x}}{\varepsilon_r} \\ \frac{c_o\overline{D_y}}{\varepsilon_r} \\ \frac{c_o\overline{D_z}}{\varepsilon_r} \\ \frac{B_x}{\mu_r\mu_o} \\ \frac{B_y}{\mu_r\mu_o} \\ \frac{B_z}{\mu_r\mu_o} \end{bmatrix}, \widetilde{M} = \frac{\partial\vec{q}}{\partial\overrightarrow{w}} = \begin{bmatrix} \frac{\varepsilon_r}{c_o} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\varepsilon_r}{c_o} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\varepsilon_r}{c_o} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu_r\mu_o & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu_r\mu_o & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu_r\mu_o \end{bmatrix} \tag{2.96}$$

Dirichlet boundary conditions are implemented by setting the $\vec{q}$ value at each node on the boundary. Since $\vec{q}$ is set explicitly, it is not necessary to solve Equation 2.69.

2.8 Implicit Time Stepping

Implicit time stepping with Newton's method [33] is employed to obtain the time history of the electromagnetic field. Implicit time stepping allows for the use of a time step that depends on the physics of the problem. Implicit time stepping schemes are more numerically stable than explicit time stepping schemes. For an explicit time stepping scheme, the numerical stability of the scheme is governed by the time step, and generally the time step has to be less than or equal to the grid spacing divided by the wave speed. This is a serious limitation because in an unstructured grid topology the grid spacing can vary drastically, and the time step for the whole field must be in direct relation to the smallest tetrahedron. However, if an implicit time stepping scheme is employed, the time step does not depend on the grid spacing for numerical stability.

The unsteady residual is linearized with Newton's method, and an implicit time stepping algorithm is developed. This algorithm is robust enough to handle nonlinear problems, but since Maxwell's equations are linear, only one Newton step is needed at each time step. The first step in the Newton linearized implicit time stepping algorithm is to introduce a Newton iteration index, s, and calculate an iterative sequence, $s = 1, 2, \ldots$ until it satisfies the following relation:

$$\iiint_{\Omega}\left[w+\left(\frac{\partial w}{\partial x}A^{n+1,s}+\frac{\partial w}{\partial y}B^{n+1,s}+\frac{\partial w}{\partial z}C^{n+1,s}\right)\tau\right]\left(\frac{\partial \vec{q}}{\partial t}+\vec{S}^{n+1,s}\right)\partial\Omega+$$

$$\iint_{\Gamma} w\left(\vec{F}^{n+1,s}n_x+\vec{G}^{n+1,s}n_y+\vec{H}^{n+1,s}n_z\right)\partial\Gamma-$$

$$\iiint_{\Omega}\left(\frac{\partial w}{\partial x}\vec{F}^{n+1,s}+\frac{\partial w}{\partial y}\vec{G}^{n+1,s}+\frac{\partial w}{\partial z}\vec{H}^{n+1,s}\right)\partial\Omega+$$

$$\iiint_{\Omega}\left[\frac{\partial w}{\partial x}A^{n+1,s}+\frac{\partial w}{\partial y}B^{n+1,s}+\frac{\partial w}{\partial z}C^{n+1,s}\right]\tau\left[\frac{\partial \vec{F}^{n+1,s}}{\partial x}+\frac{\partial \vec{G}^{n+1,s}}{\partial y}+\frac{\partial \vec{H}^{n+1,s}}{\partial z}\right]\partial\Omega=0\equiv \mathrm{R}^{\mathrm{n+1,s}} \tag{2.97}$$

Next, introduce a Newton linearization about $\vec{q}^{n+1,s}$ (Taylor series expansion of $R^{n+1,s}$), and the unsteady residual becomes:

$$R^{n+1,s+1} = R^{n+1,s} + \left(\frac{\partial R}{\partial \vec{q}}\right)^{n+1,s} \Delta_s \vec{q}^{n+1} + O(\Delta_s \vec{q})^2 \tag{2.98}$$

$$\Delta_s \vec{q}^{n+1} \equiv \vec{q}^{n+1,s+1} - \vec{q}^{n+1,s} \tag{2.99}$$

The left hand side of Equation 2.98 is set to zero, and the algorithm is:

$$\left[\frac{\partial R}{\partial \vec{q}}\right]^{n+1,s} \Delta_s \vec{q}^{n+1} = -R^{n+1,s} \tag{2.100}$$

The solution of Equation 2.100 is the perturbation of the time accurate electromagnetic field from one time step to the next, and the procedure to solve Equation 2.100 is discussed in the next two sections.

2.9 Linearization Matrix

In Equation 2.100 the term $\left[\frac{\partial R}{\partial \vec{q}}\right]$ is the linearization matrix [33]. The linearization matrix is a sparse block matrix that is the square of the number of nodes in the overall computational mesh. Each block in the overall sparse matrix is six by six. The derivatives of the unsteady residual with respect to the field variables are computed with a complex Taylor series expansion (CTSE) [37]. The CTSE for a function f is shown in Equation 2.101.

$$f(x + i\Delta x) = f(x) + i\Delta x \frac{\partial f(x)}{\partial x} - \frac{\Delta x^2}{2!} \frac{\partial^2 f(x)}{\partial x^2} + \cdots \tag{2.101}$$

The derivative of the function with respect to x is computed by dividing the imaginary part of the complex perturbed function by the perturbation. The derivative of the function $f(x)$ with respect to x is shown in Equation 2.102.

$$\frac{\partial f(x)}{\partial x} = \frac{Im\left(f(x+i\Delta x)\right)}{\Delta x} + O(\Delta x^2) \tag{2.102}$$

The derivative evaluated in Equation 2.102 is second order accurate for the cost of one function evaluation. Because of Equation 2.102 is not subject to subtractive cancellation errors, and it can be computed with an exceptionally small Δx. The value of Δx is chosen as the square root of machine zero, which means that the accuracy of the evaluated derivative is of the order of machine zero. Filling the linearization matrix can be accomplished in a few nested steps. The first step is to loop through all of the nodes in the overall computational mesh. While looping through all of the nodes, at each node perturb $\vec{q}$ in the complex plane by machine zero. After the node is perturbed, loop through the tetrahedra connected to that node, and compute the nodal values of the residual at each tetrahedron. For each tetrahedron, add the nodal value of the derivative of the residual (imaginary part of the residual divided by the perturbation) to the proper place in the linearization matrix. The proper place in the linearization matrix is the perturbed node's row and the corresponding appropriate column.

The linearization matrix is large and only the nonzero components are stored in memory. Instead of the linearization matrix being stored as a two dimensional array dimensioned to be the square of the number of nodes in the overall computational mesh, the linearization matrix is stored as a one dimensional array that is the number of nonzero components long. However, it should be noted that each entry in the one-dimensional array represents a six-by-six matrix to

accommodate all six equations. Compressed row storage is used to access the specific instances of the one dimensional array.

2.10 Linear System Solver

At each time step the linear system, Equation 2.100, is solved. The solution of Equation 2.100 is a follows:

$$\Delta\vec{q} = \left[\frac{\partial R}{\partial \vec{q}}\right]^{-1}(-R) \tag{2.103}$$

Since the linearization matrix is large, it cannot be efficiently inverted directly, and an iterative solver must be employed. Typical iterative solvers are Jacobi and Gauss-Seidel, which both converge quickly for strongly diagonally-dominant systems and have limited memory requirements. However, these methods only converge if the matrix on the left hand side of the linear system is diagonally dominant. The Petrov-Galerkin solution procedure can have non-diagonally dominant matrices if higher-order elements or large time steps are employed. Higher order elements provide a higher-order solution at the cost of adding more nonzero off-diagonal columns to the rows of the linearization matrix. Hence, a linear system solver that is robust enough to solve a non-diagonally dominant system must be employed. For this solution procedure, the Generalized Minimal RESidual method (GMRES) [38] is utilized. This method uses the Arnoldi iteration to find the approximate solution vector in a Krylov subspace with minimal iterative linear system residual. The GMRES method yields the exact solution to machine accuracy when the dimension of the Krylov subspace equals the dimension of the system or one of the search directions becomes linearly dependent on one of the previous ones. However, each Krylov search vector is the same size as the overall solution vector stored at each

node, and storing each Krylov search vector is not practical due to computational memory constraints. In order to speed converge of the GMRES scheme and limit the number of search directions, a diagonal preconditioner is used. The diagonal preconditioner loosely approximates the inverse of the matrix, and takes minimal computational effort to solve. A direct-solve LU elimination method is used to solve the block six by six matrix problems in conjunction with the diagonal preconditioner.

2.11 Procedure to Execute the Software on a Distributed Memory Supercomputer

The solution procedure outlined in the previous sections has been implemented for a parallel distributed memory computing environment. The implemented procedure uses the message passing paradigm to pass data between processes.

The adapted procedure involves the following steps: decompose the computational grid, build sub-domains, generate send/receive lists, package data to be sent, send/receive data, unpack received data, and put it in its proper place in memory [39]. The unstructured mesh is decomposed with the METIS library [40]. METIS efficiently partitions an unstructured mesh and works to decrease the ratio of communication to computation or surface area to volume ratio for each sub-domain. METIS outputs a partition array that tells each node which domain it belongs to. Once the partition array is generated, subdomains are built. Subdomains are volumetric portions of the overall computational mesh that each process is assigned to work on. Each sub-domain is made up of a portion of the overall number of tetrahedra and boundary triangles, and is built from the nodes it owns. If a tetrahedron contains nodes that belong to difference processes, the tetrahedron belongs to multiple processes. For instance, Figure 2.4 shows a tetrahedron that is split over two processes.

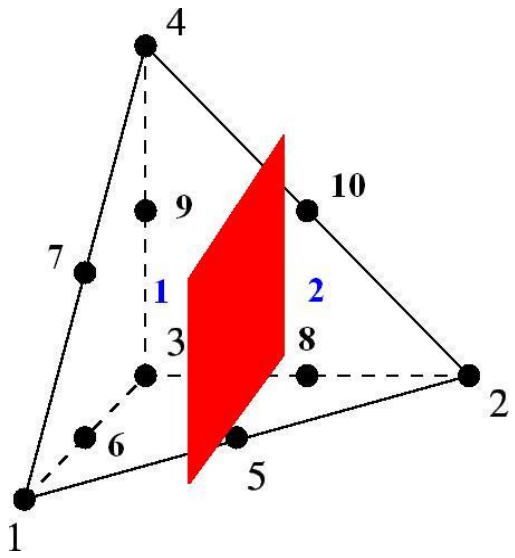

Figure 2.4 Split Quadratic Tetrahedron over Processes 1 and 2

In Figure 2.4, the quadratic tetrahedron is split over processes 1 and 2. Nodes 1, 3, 4, 6, 7, and 9 are owned by process 1, and nodes 2, 5, 8, and 10 are owned by process 2. The tetrahedron belongs to the sub-domain for processes 1 and 2. The nodes that are not owned by each process are phantom nodes, and their information must be passed. In order for information to be passed, send and receive lists must be generated. A send list for a given process is made up of two pieces of information. These pieces are the node numbers of the current process that are phantom nodes on another process, and the other process to which nodal information needs to be sent. A receive list for a given process is made up of the inverse of the information present in the send list. A receive list states each node on a current process that is a phantom node, and from where each phantom node receives its information. Once the send and receive lists are generated, each process packs up its owned data so that each process only sends and receives data once. The data is sent and received by using the message passing interface (MPI) library [41]. Each send and

receive call is nonblocking, which means that each process can compute on parts of the sub-domain that are not dependent on passed information while the send/receive process takes place. Once the data is received by a given process it is unpacked and put into its proper place in memory.

At each time step, messages have to be sent between sub-domains to update the solution vector at the phantom nodes for each subdomain before the residual and linearization matrix are computed. A matrix vector product between the linearization matrix and the solution vector on a given sub-domain would be as follows:

$$\begin{bmatrix} x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \neq \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.5 Product of Linearization Matrix and Solution Vector before Message Passing

In Figure 2.5 the $x's$ are known quantities for the given sub-domain and the 0's are unknown quantities for the sub-domain. The information denoted by 0's in the solution vector must be passed from the subdomain that owns that data, and Figure 2.6 shows the correct matrix-vector product implementation.

$$\begin{bmatrix} x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & x & x & x & x \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & x & x & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \\ x \end{bmatrix} = \begin{bmatrix} x \\ x \\ x \\ x \\ x \\ x \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 2.6 Product of Linearization Matrix and Solution Vector after Message Passing

CHAPTER III

TIME ACCURATE SHAPE SENSITIVITY ANALYSIS AND DESIGN

A shape design cycle is implemented that couples the electromagnetic field simulation software, time accurate adjoint based method for computing the sensitivity derivatives, software that smoothes the computational mesh by computing the linear elastic equations, and optimization package. The shape design cycle modifies the shape of an electromagnetic object such that a cost function is minimized. The subsequent methods for computing the sensitivity derivatives and the design cycle are discussed in the following sections.

3.1 Forward and Reverse Modes

Forward mode and reverse mode are the two methodologies used to compute time accurate shape sensitivity derivatives [11]. The time accurate shape sensitivity derivatives are used to design an object with a minimized cost function. The shape sensitivity derivatives are the derivatives of the cost function with respect to design variables that describe the shape of the object in computational space. The forward mode methodology can be used to compute the shape sensitivity derivatives with the following methods: finite difference method, complex Taylor series expansion, and direct differentiation [11]. The reverse mode methodology uses an adjoint method to approximate the shape sensitivity derivatives [11]. Each of these methods is discussed below where the cost function is denoted as I, and the design variables are denoted as $\beta's$.

3.2 Forward Mode, Finite-Difference Method

The first derivative of the cost function with respect to the design variables can be approximated with the central-difference method given as:

$$\frac{\partial I(\beta)}{\partial \beta} = \frac{I(\beta+\Delta\beta)-I(\beta-\Delta\beta)}{2\Delta\beta} + O(\Delta\beta^2) \tag{3.1}$$

The central-difference method is subject to subtractive cancellation, and should be used with caution especially when employing small perturbations. When subtractive cancellation is present, the truncation error increases as $\Delta\beta$ decreases. Subtractive cancellation is present when $\Delta\beta$ is reduced to a point that the computer cannot discriminate between the terms in the numerator of Equation 3.1. This method is not practical if multiple $\beta's$ are used to design an object because it would have to be computed for each design variable.

3.3 Forward Mode, Complex Taylor Series Expansion (CTSE)

Another way to approximate the shape sensitivity derivatives via forward mode is to employ the CTSE method. The CTSE method is discussed in Section 2.9. Even though this method is not subject to subtractive cancellation errors, it again is not practical for shape design if the object is described by a large number of $\beta's$. However, because of the high level of accuracy of this methodology, it can be used as a comparison tool with direct differentiation and the discrete adjoint method.

3.4 Forward Mode, Direct Differentiation

The last way to approximate the shape sensitivity derivatives via forward mode is to utilize direct differentiation. Here, the derivative of the cost function with respect to the k^{th}

design variable is computed by summing the derivatives from all time steps, and the sum is over N_k time steps that influence the cost function.

$$\frac{\partial I}{\partial \beta_k} = \sum_{i=1}^{N_t} \left\{ \frac{\partial I}{\partial \overrightarrow{q_i}} \frac{\partial \overrightarrow{q_i}}{\partial \beta_k} + \frac{\partial I}{\partial \vec{x}} \frac{\partial \vec{x}}{\partial \beta_k} \right\} \tag{3.2}$$

In Equation 3.2 the arrays $\frac{\partial I}{\partial \overrightarrow{q_i}}$ and $\frac{\partial I}{\partial \vec{x}}$ are computed by the CTSE method described in section 2.9. The array, $\frac{\partial \overrightarrow{q_i}}{\partial \beta_k}$ is second order accurate in time, and is evaluated at a time step i by Equation 3.3.

$$\frac{\partial \overrightarrow{q_i}}{\partial \beta_k} = -\left[\frac{\partial R_i}{\partial \overrightarrow{q_i}}\right]^{-1} \left\{ \frac{\partial R_i}{\partial \beta_k} + \left[\frac{\partial R_i}{\partial \vec{q}_{i-1}}\right] \frac{\partial \vec{q}_{i-1}}{\partial \beta_k} + \left[\frac{\partial R_i}{\partial \vec{q}_{i-2}}\right] \frac{\partial \vec{q}_{i-2}}{\partial \beta_k} \right\} \tag{3.3}$$

In Equation 3.3, the derivative of the unsteady residual (Equation 2.97) with respect to the design variables, $\frac{\partial R_i}{\partial \beta_k}$, is evaluated as follows:

$$\frac{\partial R_i}{\partial \beta_k} = \left[\frac{\partial R_i}{\partial \vec{x}}\right] \frac{\partial \vec{x}}{\partial \beta_k} \tag{3.4}$$

The matrix $\left[\frac{\partial R_i}{\partial \vec{x}}\right]$ is generated the same way the linearization matrix is generated in section 2.9 except that $\vec{x}$ is perturbed instead of $\vec{q}$. Also, $\left[\frac{\partial R_i}{\partial \vec{q}_{i-1}}\right]$ and $\left[\frac{\partial R_i}{\partial \vec{q}_{i-2}}\right]$ are computed in a similar fashion. The mesh sensitivity arrays $\frac{\partial \vec{x}}{\partial \beta_k}$ are computed by an auxiliary solver that computes the linear elastic equations, and they are imported into the direct method solver. The linear system shown in Equation 3.3 is solved with the GMRES linear system solver described in section 2.10, and the direct method solver can be executed on a distributed memory supercomputer as described in section 2.11. As with the other forward mode methods, the direct method is not practical for shape design problems with a large number of $\beta's$. For each β Equation 3.3 would

have to be computed with a linear system solver. However, the direct method is an excellent comparison tool for the reverse mode discrete adjoint method, and, as will be seen in the next section, some of the arrays and matrices needed for the direct method are needed for the discrete adjoint method.

3.5 Reverse Mode, Discrete-Adjoint Method

The reverse mode discrete adjoint method is optimal for computing the shape sensitivity derivatives when there are a large number of design variables. This methodology does not necessitate solving a linear system for each design variable considered. This is accomplished by summing the contributions of $\frac{\partial I}{\partial \beta_k}$ in reverse order from the last time step to the first time step and not explicitly solving for $\frac{\partial \overrightarrow{q_i}}{\partial \beta_k}$ at the current time step. At each step, $\frac{\partial \overrightarrow{q_i}}{\partial \beta_k}$ is expanded by Equation 3.3. Equation 3.2 is rewritten as Equation 3.5 to show this change.

$$\frac{\partial I}{\partial \beta_k} = -\sum_{i=N_t}^{1}\left(\frac{\partial I}{\partial \vec{q}_i}\left\{\left[\frac{\partial R_i}{\partial \vec{q}_i}\right]^{-1}\left(\frac{\partial R_i}{\partial \beta_k}+\left[\frac{\partial R_i}{\partial \vec{q}_{i-1}}\right]\frac{\partial \vec{q}_{i-1}}{\partial \beta_k}+\left[\frac{\partial R_i}{\partial \vec{q}_{i-2}}\right]\frac{\partial \vec{q}_{i-2}}{\partial \beta_k}\right)\right\}+\frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k}\right) \tag{3.5}$$

Next, Equation 3.5 is transposed, and is shown as Equation 3.6.

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T = -\sum_{i=N_t}^{1}\left(\left\{\left(\frac{\partial R_i}{\partial \beta_k}+\frac{\partial \vec{q}_{i-1}}{\partial \beta_k}\left[\frac{\partial R_i}{\partial \vec{q}_{i-1}}\right]+\frac{\partial \vec{q}_{i-2}}{\partial \beta_k}\left[\frac{\partial R_i}{\partial \vec{q}_{i-2}}\right]\right)^T\left[\frac{\partial R_i}{\partial \vec{q}_i}\right]^{-T}\right\}\left(\frac{\partial I}{\partial \vec{q}_i}\right)^T+\frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k}\right) \tag{3.6}$$

An algorithm is developed to compute Equation 3.6 by expanding the first three terms of the summation ($i = N_t,\ i = N_t - 1,\ i = N_t - 2$) and grouping terms together. When $i = N_t$ Equation 3.6 becomes:

$$\Lambda_{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t}}\right)^T \tag{3.7}$$

$$\lambda_1{}^{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t-1}}\right]^T \Lambda_{N_t} \tag{3.8}$$

$$\lambda_2{}^{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t-2}}\right]^T \Lambda_{N_t} \tag{3.9}$$

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_{N_t}}{\partial \beta_k}\right)^T \Lambda_{N_t} - \left(\frac{\partial \vec{q}_{N_t-1}}{\partial \beta_k}\right)^T \lambda_1{}^{N_t} - \left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \lambda_2{}^{N_t} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k} \tag{3.10}$$

In Equation 3.10 the terms $-\left(\frac{\partial \vec{q}_{N_t-1}}{\partial \beta_k}\right)^T \lambda_1$ and $-\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \lambda_2$ are not evaluated at $i = N_t$, but are evaluated at $i = N_t - 1$. At $i = N_t - 1$ Equation 3.6 with the added terms from 3.10 becomes Equation 3.11. In order to derive Equation 3.11 the term $\left(\frac{\partial \vec{q}_{N_t-1}}{\partial \beta_k}\right)^T$ is expanded from Equation 3.3.

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_{N_t-1}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-1}}\right)^T -$$

$$\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-2}}\right]^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-1}}\right)^T -$$

$$\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-3}}\right]^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-1}}\right)^T +$$

$$\left(\frac{\partial R_{N_t-1}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \lambda_1{}^{N_t} +$$

$$\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-2}}\right]^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \lambda_1{}^{N_t} +$$

$$\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-3}}\right]^T \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \lambda_1{}^{N_t} -$$

$$\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T {\lambda_2}^{N_t} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k} \tag{3.11}$$

The following terms can be pulled out of Equation 3.11:

$$\Lambda_{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \left\{\left(\frac{\partial I}{\partial \vec{q}_{N_t-1}}\right) - {\lambda_1}^{N_t}\right\}^T \tag{3.12}$$

$${\lambda_1}^{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-2}}\right]^T \Lambda_{N_t-1} \tag{3.13}$$

$${\lambda_2}^{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-3}}\right]^T \Lambda_{N_t-1} \tag{3.14}$$

After applying Equations 3.12 – 3.14, Equation 3.11 becomes Equation 3.15.

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T += -\left(\frac{\partial R_{N_t-1}}{\partial \beta_k}\right)^T \Lambda_{N_t-1} - \left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \left\{{\lambda_1}^{N_t-1} + {\lambda_2}^{N_t}\right\} - \left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T {\lambda_2}^{N_t-1} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k} \tag{3.15}$$

In Equation 3.15 the terms $-\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T \left\{{\lambda_1}^{N_t-1} + {\lambda_2}^{N_t}\right\}$ and $-\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T {\lambda_2}^{N_t-1}$ are not evaluated at time step $N_t - 1$, but are added to the contribution of $\left(\frac{\partial I}{\partial \beta_k}\right)^T$ at the next time step $N_t - 2$. At time step $N_t - 2$ the term $\left(\frac{\partial \vec{q}_{N_t-2}}{\partial \beta_k}\right)^T$ is expanded from Equation 3.3, and the contribution to $\left(\frac{\partial I}{\partial \beta_k}\right)^T$ from the time step $N_t - 2$ is shown in Equation 3.16.

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T += -\left(\frac{\partial R_{N_t-2}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-2}}\right)^T -$$

$$\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-3}}\right]^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-2}}\right)^T -$$

$$\left(\frac{\partial \vec{q}_{N_t-4}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-4}}\right]^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t-2}}\right)^T +$$

$$\left(\frac{\partial R_{N_t-2}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left\{{\lambda_1}^{N_t-1} + {\lambda_2}^{N_t}\right\} +$$

$$\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-3}}\right]^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left\{{\lambda_1}^{N_t-1} + {\lambda_2}^{N_t}\right\} +$$

$$\left(\frac{\partial \vec{q}_{N_t-4}}{\partial \beta_k}\right)^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-4}}\right]^T \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left\{{\lambda_1}^{N_t-1} + {\lambda_2}^{N_t}\right\} -$$

$$\left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T {\lambda_2}^{N_t-1} + \frac{\partial I}{\partial \vec{x}} \frac{\partial \vec{x}}{\partial \beta_k} \tag{3.16}$$

The following terms can be pulled out of Equation 3.16:

$$\Lambda_{N_t-2} = \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-2}}\right]^{-T} \left\{\left(\frac{\partial I}{\partial \vec{q}_{N_t-2}}\right) - {\lambda_1}^{N_t-1} - {\lambda_2}^{N_t}\right\}^T \tag{3.17}$$

$${\lambda_1}^{N_t-2} = \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-3}}\right]^T \Lambda_{N_t-2} \tag{3.18}$$

$${\lambda_2}^{N_t-2} = \left[\frac{\partial R_{N_t-2}}{\partial \vec{q}_{N_t-4}}\right]^T \Lambda_{N_t-2} \tag{3.19}$$

After applying Equations 3.17 - 3.19, Equation 3.16 becomes Equation 3.20.

$$\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_{N_t-2}}{\partial \beta_k}\right)^T \Lambda_{N_t-2} - \left(\frac{\partial \vec{q}_{N_t-3}}{\partial \beta_k}\right)^T \left\{{\lambda_1}^{N_t-2} + {\lambda_2}^{N_t-1}\right\} -$$

$$\left(\frac{\partial \vec{q}_{N_t-4}}{\partial \beta_k}\right)^T {\lambda_2}^{N_t-2} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k} \tag{3.20}$$

Equations 3.10, 3.15 and 3.20 provide the basis for the algorithm that is capable of solving the sensitivity derivatives with the reverse mode discrete adjoint method. The second order accurate in time algorithm is as follows:

1. $i = N_t$
2. $\Lambda_{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t}}\right]^{-T} \left(\frac{\partial I}{\partial \vec{q}_{N_t}}\right)^T$
3. $\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_{N_t}}{\partial \beta_k}\right)^T \Lambda_{N_t} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k}$
4. ${\lambda_1}^{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t-1}}\right]^T \Lambda_{N_t}$
5. ${\lambda_2}^{N_t} = \left[\frac{\partial R_{N_t}}{\partial \vec{q}_{N_t-2}}\right]^T \Lambda_{N_t}$
6. $i = N_t - 1$
7. $\Lambda_{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-1}}\right]^{-T} \left\{\left(\frac{\partial I}{\partial \vec{q}_{N_t-1}}\right) - {\lambda_1}^{N_t}\right\}^T$
8. $\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_{N_t-1}}{\partial \beta_k}\right)^T \Lambda_{N_t-1} + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k}$
9. ${\lambda_1}^{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-2}}\right]^T \Lambda_{N_t-1}$
10. ${\lambda_2}^{N_t-1} = \left[\frac{\partial R_{N_t-1}}{\partial \vec{q}_{N_t-3}}\right]^T \Lambda_{N_t-1}$

11. $i = i - 1$

12. $\Lambda_i = \left[\frac{\partial R_i}{\partial \vec{q}_i}\right]^{-T} \left\{\left(\frac{\partial I}{\partial \vec{q}_i}\right) - {\lambda_1}^{i+1} - {\lambda_2}^{i+2}\right\}^T$

13. $\left(\frac{\partial I}{\partial \beta_k}\right)^T \mathrel{+}= -\left(\frac{\partial R_i}{\partial \beta_k}\right)^T \Lambda_i + \frac{\partial I}{\partial \vec{x}}\frac{\partial \vec{x}}{\partial \beta_k}$

14. ${\lambda_1}^i = \left[\frac{\partial R_i}{\partial \vec{q}_{i-1}}\right]^T \Lambda_i$

15. ${\lambda_2}^i = \left[\frac{\partial R_i}{\partial \vec{q}_{i-2}}\right]^T \Lambda_i$

16. Go back to 11 until $i = 1$

When $i = 1$, $\left(\frac{\partial \vec{q}_{i-1}}{\partial \beta_k}\right)^T$ and $\left(\frac{\partial \vec{q}_{i-2}}{\partial \beta_k}\right)^T$ both equal zero, and the time accurate sensitivity derivatives are computed without having to solve a linear system for each design variable. This algorithm is implemented for a parallel computing environment as discussed in section 2.11. At each step the transpose of the linearization matrix is computed, and the linear system that computes Λ_i is solved with the GMRES algorithm discussed in section 2.10.

3.6 Software to Generate Higher-Order Numerical Solutions to the Linear Elastic Equations

To design the shape of an electromagnetic object, the computational mesh volume surrounding the object must be deformed during the design cycle to accommodate changes in the geometry. A methodology for obtaining a numerical solution to the linear elastic equations has been developed to smooth the mesh during each step of the design cycle [42]. Because of the linearity of the linear elastic equations, mesh sensitivity derivatives can be calculated by replacing the displacement at the boundaries with the sensitivity derivatives of the surface points.

The linear elastic equations are as follows (see e.g. [42]):

$$\frac{\partial}{\partial x}\left[\alpha_{11}\frac{\partial \Delta x}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{12}\frac{\partial \Delta x}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{13}\frac{\partial \Delta x}{\partial z}\right]+$$

$$\frac{\partial}{\partial x}\left[\theta_{11}\left(\frac{\partial \Delta y}{\partial y}+\frac{\partial \Delta z}{\partial z}\right)\right]+\frac{\partial}{\partial y}\left[\theta_{12}\frac{\partial \Delta y}{\partial x}\right]+\frac{\partial}{\partial z}\left[\theta_{13}\frac{\partial \Delta z}{\partial x}\right]=0$$

$$\frac{\partial}{\partial x}\left[\alpha_{21}\frac{\partial \Delta y}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{22}\frac{\partial \Delta y}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{23}\frac{\partial \Delta y}{\partial z}\right]+$$

$$\frac{\partial}{\partial x}\left[\theta_{21}\frac{\partial \Delta x}{\partial y}\right]+\frac{\partial}{\partial y}\left[\theta_{22}\left(\frac{\partial \Delta x}{\partial x}+\frac{\partial \Delta z}{\partial z}\right)\right]+\frac{\partial}{\partial z}\left[\theta_{23}\frac{\partial \Delta z}{\partial y}\right]=0$$

$$\frac{\partial}{\partial x}\left[\alpha_{31}\frac{\partial \Delta z}{\partial x}\right]+\frac{\partial}{\partial y}\left[\alpha_{32}\frac{\partial \Delta z}{\partial y}\right]+\frac{\partial}{\partial z}\left[\alpha_{33}\frac{\partial \Delta z}{\partial z}\right]+$$

$$\frac{\partial}{\partial x}\left[\theta_{31}\frac{\partial \Delta x}{\partial z}\right]+\frac{\partial}{\partial y}\left[\theta_{32}\frac{\partial \Delta y}{\partial z}\right]+\frac{\partial}{\partial z}\left[\theta_{33}\left(\frac{\partial \Delta x}{\partial x}+\frac{\partial \Delta y}{\partial y}\right)\right]=0 \tag{3.21}$$

In Equation 3.21 $\Delta x, \Delta y,$ and Δz are the displacements along the $x, y,$ and z coordinates. The linear elastic solver updates the coordinates at each node via Equation 3.22.

$$\vec{x}^{new}=\vec{x}^{old}+\Delta\vec{x} \tag{3.22}$$

The $\alpha's$ and $\theta's$ are defined as follows:

$$\alpha_{11}=\alpha_{22}=\alpha_{33}=\frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \tag{3.23}$$

$$\alpha_{12}=\alpha_{13}=\alpha_{21}=\alpha_{23}=\alpha_{31}=\alpha_{32}=\frac{E}{2(1+\nu)} \tag{3.24}$$

$$\theta_{11}=\theta_{22}=\theta_{33}=\frac{E\nu}{(1+\nu)(1-2\nu)} \tag{3.25}$$

$$\theta_{12}=\theta_{13}=\theta_{21}=\theta_{23}=\theta_{31}=\theta_{32}=\frac{E}{2(1+\nu)} \tag{3.26}$$

In Equations 3.23-3.26, E is Young's modulus of elasticity, and ν is Poisson's ratio. For this implementation, Young's modulus for each tetrahedron is the tetrahedron's aspect ratio divided by its volume and $\nu = 0.2$ [43].

The linear elastic equations are solved with the Galerkin finite element scheme. The equations are rewritten as follows:

$$\frac{\partial \vec{F}_{LE}}{\partial x} + \frac{\partial \vec{G}_{LE}}{\partial y} + \frac{\partial \vec{H}_{LE}}{\partial z} = 0 \tag{3.27}$$

The $\vec{q}_{LE}, \vec{F}_{LE}, \vec{G}_{LE},$ and $\vec{H}_{LE}$ vectors for the linear elastic equations are as follows:

$$\vec{q}_{LE} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{3.28}$$

$$\vec{F}_{LE} = \begin{bmatrix} \alpha_{11} \frac{\partial \Delta x}{\partial x} + \theta_{11} \left(\frac{\partial \Delta y}{\partial y} + \frac{\partial \Delta z}{\partial z} \right) \\ \alpha_{21} \frac{\partial \Delta y}{\partial x} + \theta_{21} \frac{\partial \Delta x}{\partial y} \\ \alpha_{31} \frac{\partial \Delta z}{\partial x} + \theta_{31} \frac{\partial \Delta x}{\partial z} \end{bmatrix} \tag{3.29}$$

$$\vec{G}_{LE} = \begin{bmatrix} \alpha_{12} \frac{\partial \Delta x}{\partial y} + \theta_{12} \frac{\partial \Delta y}{\partial x} \\ \alpha_{22} \frac{\partial \Delta y}{\partial y} + \theta_{22} \left(\frac{\partial \Delta x}{\partial x} + \frac{\partial \Delta z}{\partial z} \right) \\ \alpha_{32} \frac{\partial \Delta z}{\partial y} + \theta_{32} \frac{\partial \Delta y}{\partial z} \end{bmatrix} \tag{3.30}$$

$$\vec{H}_{LE} = \begin{bmatrix} \alpha_{13} \frac{\partial \Delta x}{\partial z} + \theta_{13} \frac{\partial \Delta z}{\partial x} \\ \alpha_{23} \frac{\partial \Delta y}{\partial z} + \theta_{23} \frac{\partial \Delta z}{\partial y} \\ \alpha_{33} \frac{\partial \Delta z}{\partial z} + \theta_{33} \left(\frac{\partial \Delta x}{\partial x} + \frac{\partial \Delta y}{\partial y} \right) \end{bmatrix} \tag{3.31}$$

The governing equations in weak form are shown in Equation 3.32.

$$\iiint_{\Omega} w\left(\frac{\partial \vec{F}_{LE}}{\partial x}+\frac{\partial \vec{G}_{LE}}{\partial y}+\frac{\partial \vec{H}_{LE}}{\partial z}\right)\partial\Omega = 0 \tag{3.32}$$

The Galerkin form of the equations is shown in Equation 3.33.

$$\iint_{\Gamma} w\left(\vec{F}_{LE}n_x+\vec{G}_{LE}n_y+\vec{H}_{LE}n_z\right)\partial\Gamma - \iiint_{\Omega}\left(\frac{\partial \mathrm{w}}{\partial \mathrm{x}}\vec{F}_{LE}+\frac{\partial \mathrm{w}}{\partial \mathrm{y}}\vec{G}_{LE}+\frac{\partial \mathrm{w}}{\partial \mathrm{z}}\vec{H}_{LE}\right)\partial\Omega = 0 \tag{3.33}$$

The function within each element is integrated with Gaussian quadrature as described in section 2.4. The element shape functions, shape function derivatives, and Jacobians are computed for each element as also described in section 2.4. The boundary conditions for the linear elastic equations are Dirichlet which means that the first term in Equation 3.33 does not have to be evaluated. The linear elastic equations are solved in matrix form as follows:

$$[M][\vec{q}_{LE}] = [RHS] \tag{3.34}$$

The matrix M is sparse and is composed of three by three blocks. For a given element, the second term of Equation 3.33 can be described as an element sub-matrix. If linear elements are employed, the element sub-matrix has the following form:

$$M_{sub} = \begin{bmatrix} \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \\ \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \\ \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \\ \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} & \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \end{bmatrix}$$

Figure 3.1 Form of the Linear Element (4 nodes) Sub-matrix

The integrand for each three by three block in Figure 3.1 is as follows (where M_{ij} denotes the i^{th} row and j^{th} column of the element submatrix):

$$M_{sub_{11}} = -\left(\frac{\partial N_i}{\partial x}\alpha_{11}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\alpha_{12}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\alpha_{13}\frac{\partial N_j}{\partial z}\right) \tag{3.35}$$

$$M_{sub_{12}} = -\left(\frac{\partial N_i}{\partial x}\theta_{11}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial y}\theta_{12}\frac{\partial N_j}{\partial x}\right) \tag{3.36}$$

$$M_{sub_{13}} = -\left(\frac{\partial N_i}{\partial x}\theta_{11}\frac{\partial N_j}{\partial z} + \frac{\partial N_i}{\partial z}\theta_{13}\frac{\partial N_j}{\partial x}\right) \tag{3.37}$$

$$M_{sub_{21}} = -\left(\frac{\partial N_i}{\partial x}\theta_{21}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial y}\theta_{22}\frac{\partial N_j}{\partial x}\right) \tag{3.38}$$

$$M_{sub_{22}} = -\left(\frac{\partial N_i}{\partial x}\alpha_{21}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\alpha_{22}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\alpha_{23}\frac{\partial N_j}{\partial z}\right) \tag{3.39}$$

$$M_{sub_{23}} = -\left(\frac{\partial N_i}{\partial y}\theta_{22}\frac{\partial N_j}{\partial z} + \frac{\partial N_i}{\partial z}\theta_{23}\frac{\partial N_j}{\partial y}\right) \tag{3.40}$$

$$M_{sub_{31}} = -\left(\frac{\partial N_i}{\partial x}\theta_{31}\frac{\partial N_j}{\partial z} + \frac{\partial N_i}{\partial z}\theta_{33}\frac{\partial N_j}{\partial x}\right) \tag{3.41}$$

$$M_{sub_{32}} = -\left(\frac{\partial N_i}{\partial y}\theta_{32}\frac{\partial N_j}{\partial z} + \frac{\partial N_i}{\partial z}\theta_{33}\frac{\partial N_j}{\partial y}\right) \tag{3.42}$$

$$M_{sub_{33}} = -\left(\frac{\partial N_i}{\partial x}\alpha_{31}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\alpha_{32}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\alpha_{33}\frac{\partial N_j}{\partial z}\right) \tag{3.43}$$

The submatrix is used to fill the global matrix by adding the blocks in each row of the sub-matrix to the corresponding row's columns in the global matrix. The linear elastic solver is implemented for computations on a distributed memory supercomputer, and the linear system (Equation 3.34) is solved by the procedures discussed in sections 2.11 and 2.10. In order to smooth a grid with the linear elastic equations, Equation 3.34 is solved with the $\Delta\vec{x}$ values on the boundary specified. The linear elastic equations are used also to generate the mesh sensitivity derivatives, $\frac{\partial \vec{x}}{\partial \beta_k}$. This is accomplished by replacing the $\vec{q}_{LE}$ with mesh sensitivities, and setting the mesh sensitivity boundary value to one for a boundary that is being shape optimized and to zero for a stationary boundary.

3.7 Design Optimization

The shape of an electromagnetic object is optimized by modifying the shape until a cost function is minimized. This task is accomplished by following this procedure:

1. Execute the field solver to a specified time step, and compute the cost function.
2. Execute the linear elastic solver to generate the mesh sensitivity derivatives.
3. Execute the time accurate adjoint sensitivity analysis solver to compute the sensitivity derivatives.
4. Import the cost function and the sensitivity derivatives to the PORT [43] optimization library, and the PORT optimization library outputs the modifications to the shape of the body in the form of $\Delta\vec{x}$ values for the surface.
5. Execute the linear elastic solver to modify the shape of the object and smooth the mesh.
6. Go back to 1 until the cost function is minimized.

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Electromagnetic Field Simulation Software Accuracy

The field software has been implemented with linear, quadratic, and cubic elements. The field software should theoretically provide an answer that is second order accurate when linear elements are employed, third order accurate when quadratic elements are employed, and fourth order accurate when cubic elements are employed. To attain order of accuracy, the L_2 norm of the error between the numerical solution and the exact solution is computed for multiple computational grid sizes. The order of accuracy is the slope of the line generated from evaluating the error for multiple grid sizes on a log-log plot. The exact solution is the electromagnetic field within a rectangular resonant cavity [1], which is shown in Equations 4.1-4.6:

$$\overline{D}_x = -\frac{\omega\varepsilon_r}{c_o}\frac{\beta_y}{\beta_x^2+\beta_y^2}A_o cos(\beta_x x) sin\left(\beta_y y\right) sin(\beta_z z) sin(\omega t) \tag{4.1}$$

$$\overline{D}_y = \frac{\omega\varepsilon_r}{c_o}\frac{\beta_x}{\beta_x^2+\beta_y^2}A_o sin(\beta_x x) cos\left(\beta_y y\right) sin(\beta_z z) sin(\omega t) \tag{4.2}$$

$$\overline{D}_z = 0 \tag{4.3}$$

$$B_x = -\frac{\beta_x\beta_z}{\beta_x^2+\beta_y^2}A_o sin(\beta_x x) cos\left(\beta_y y\right) cos(\beta_z z) cos(\omega t) \tag{4.4}$$

$$B_y = -\frac{\beta_y\beta_z}{\beta_x^2+\beta_y^2}A_o cos(\beta_x x) sin\left(\beta_y y\right) cos(\beta_z z) cos(\omega t) \tag{4.5}$$

$$B_z = A_o cos(\beta_x x) cos\left(\beta_y y\right) sin(\beta_z z) cos(\omega t) \tag{4.6}$$

In the above equations, the following parameters are defined as:

$$\beta_x = \frac{m\pi}{x_{length}}, \beta_y = \frac{n\pi}{y_{length}}, and\ \beta_z = \frac{p\pi}{z_{length}} \tag{4.7}$$

$$\beta_r = \sqrt{\beta_x^2 + \beta_y^2 + \beta_z^2} \tag{4.8}$$

$$\omega = c_r c_o \beta_r \tag{4.9}$$

The grid and the simulated field contours with quadratic elements for a rectangular resonant cavity after one period are shown in Figures 4.1-4.6. For this case, $m = n = p = 10$ and the nondimensional lengths of each edge of the cavity ($x_{length}, y_{length}, z_{length}$) are equal to 1.0. Each of the walls of the cavity are assumed to be PEC.

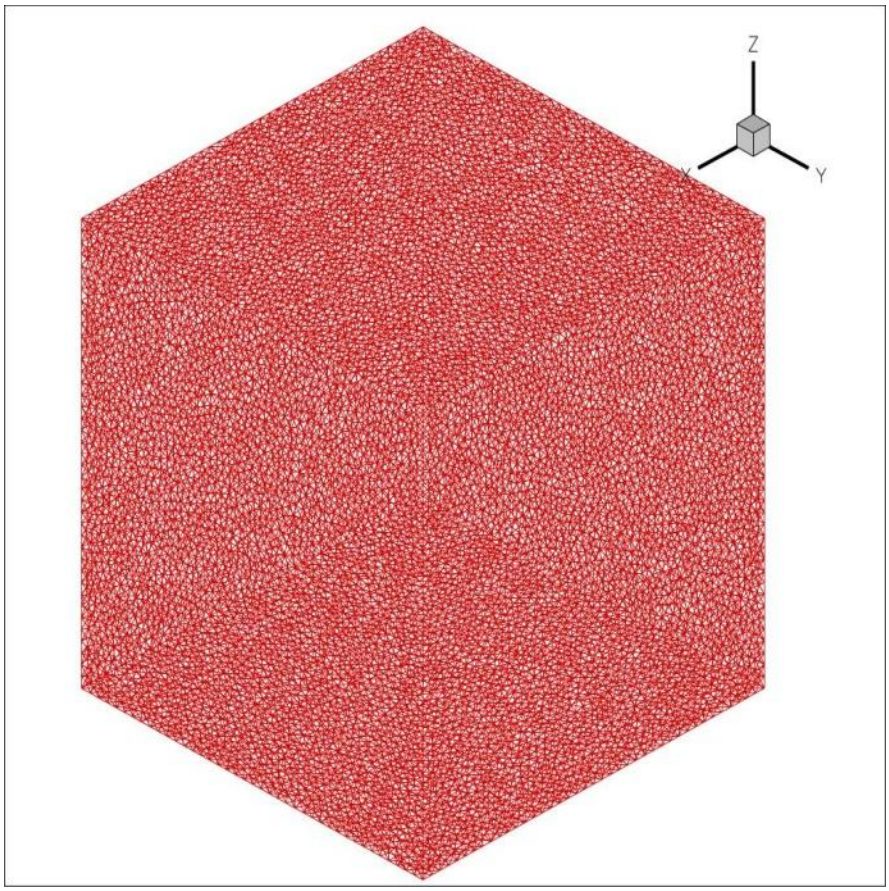

Figure 4.1 Rectangular Resonant Cavity Computational Grid Discretized with Tetrahedra

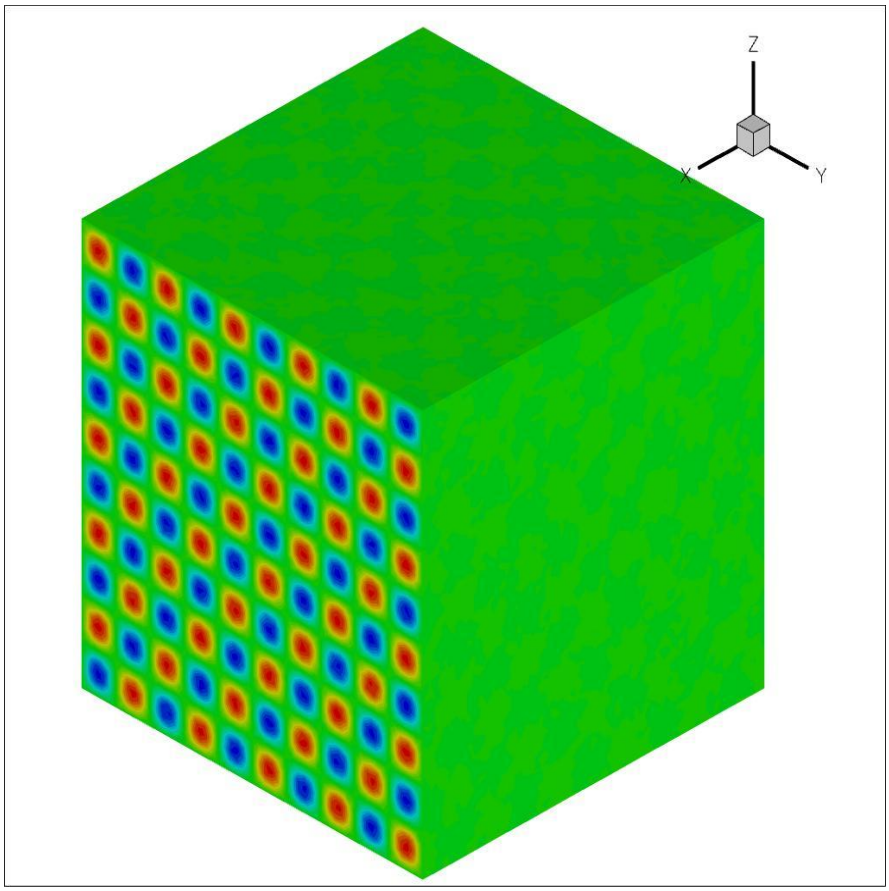

Figure 4.2 Rectangular Resonant Cavity, $\overline{D}_x$ Contours, Quadratic Elements

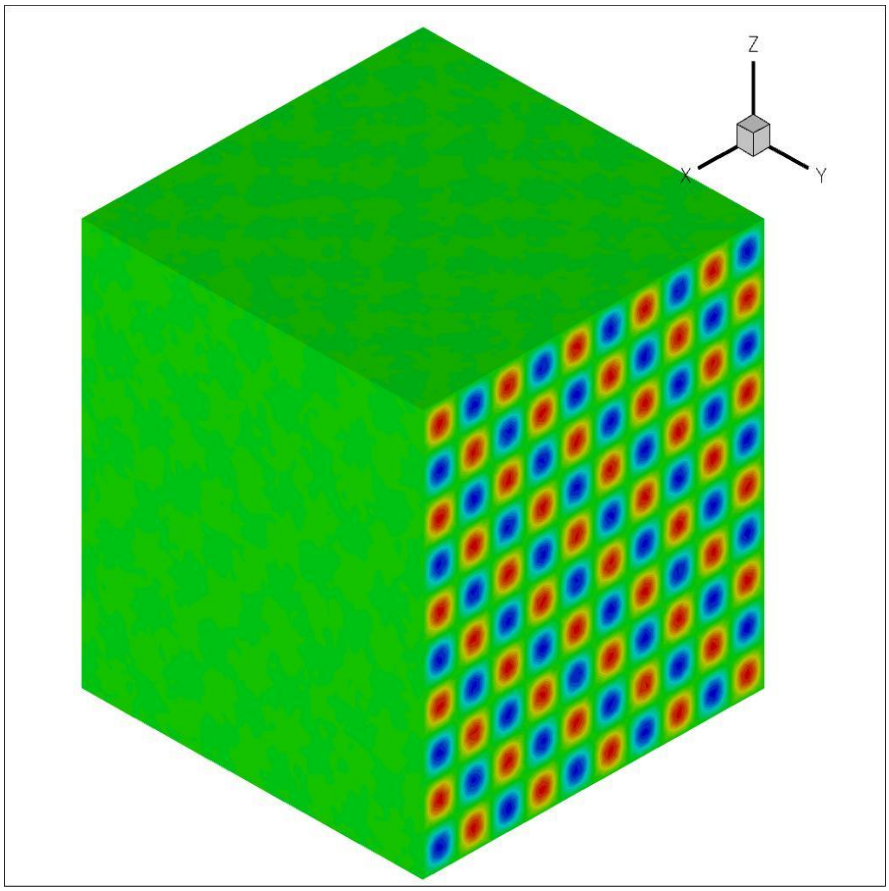

Figure 4.3 Rectangular Resonant Cavity, $\overline{D}_y$ Contours, Quadratic Elements

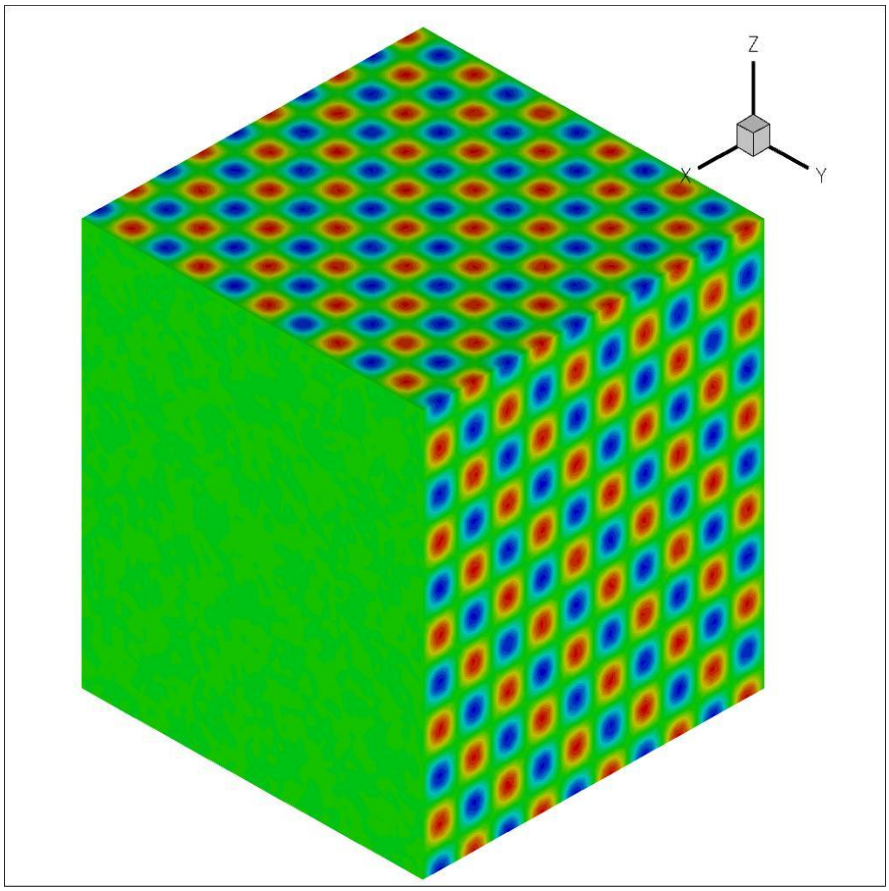

Figure 4.4 Rectangular Resonant Cavity, B_x Contours, Quadratic Elements

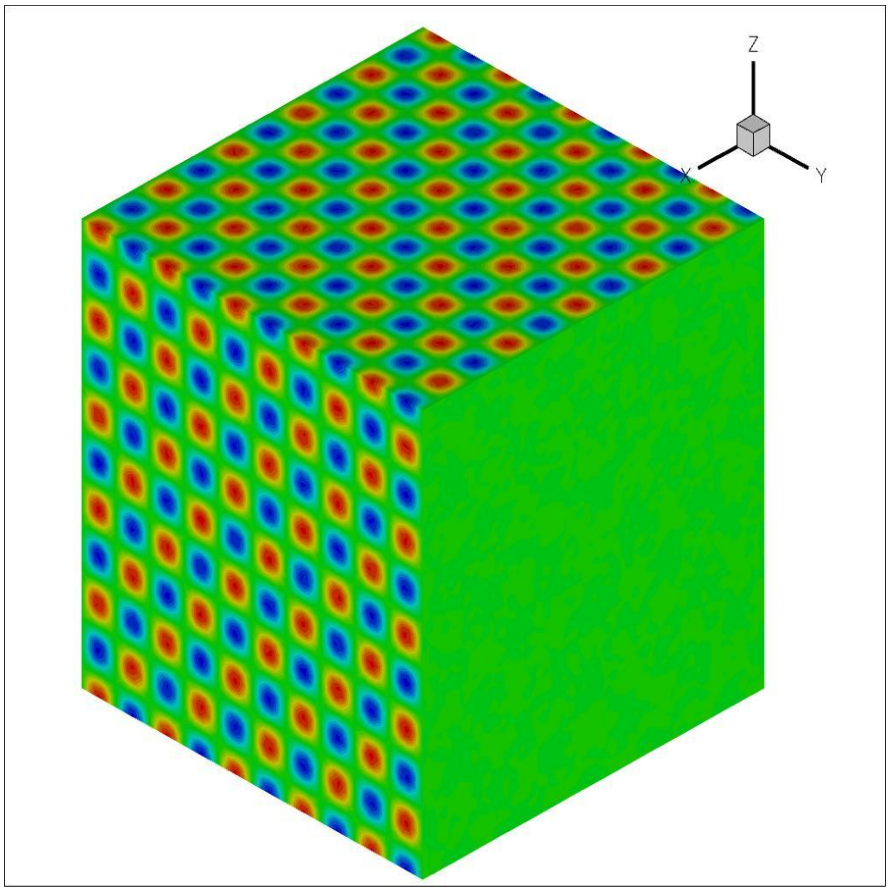

Figure 4.5 Rectangular Resonant Cavity, B_y Contours, Quadratic Elements

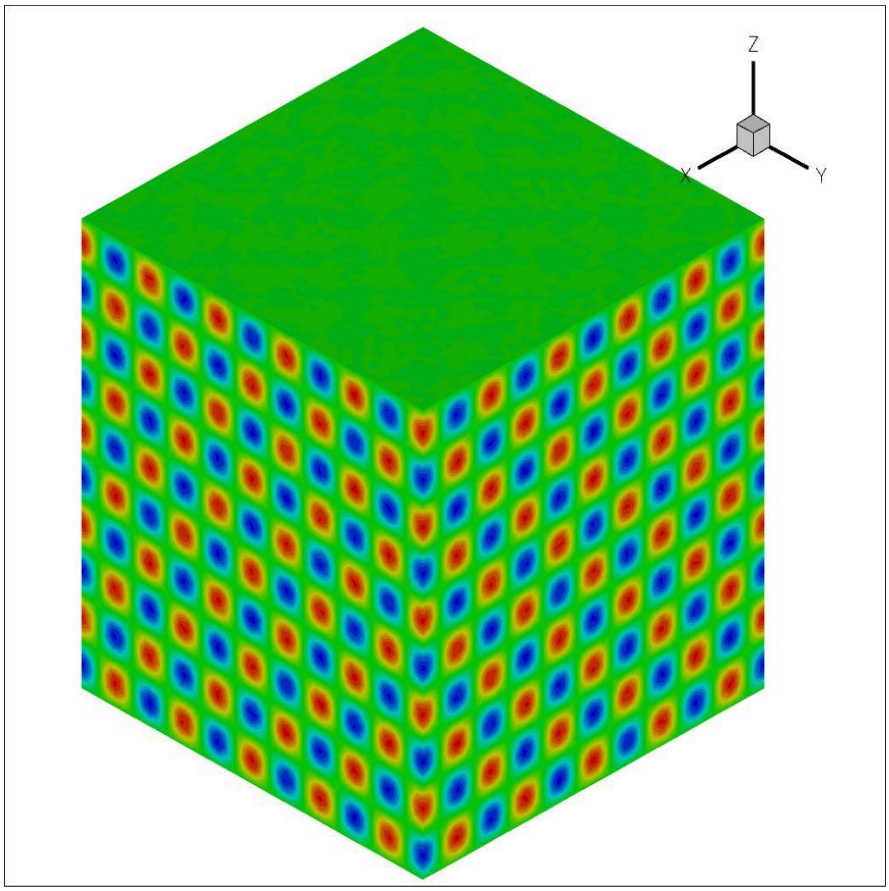

Figure 4.6 Rectangular Resonant Cavity, B_z Contours, Quadratic Elements

Figure 4.7 shows the order of accuracy of the field solver when linear, quadratic, and cubic elements are employed to discretize the field.

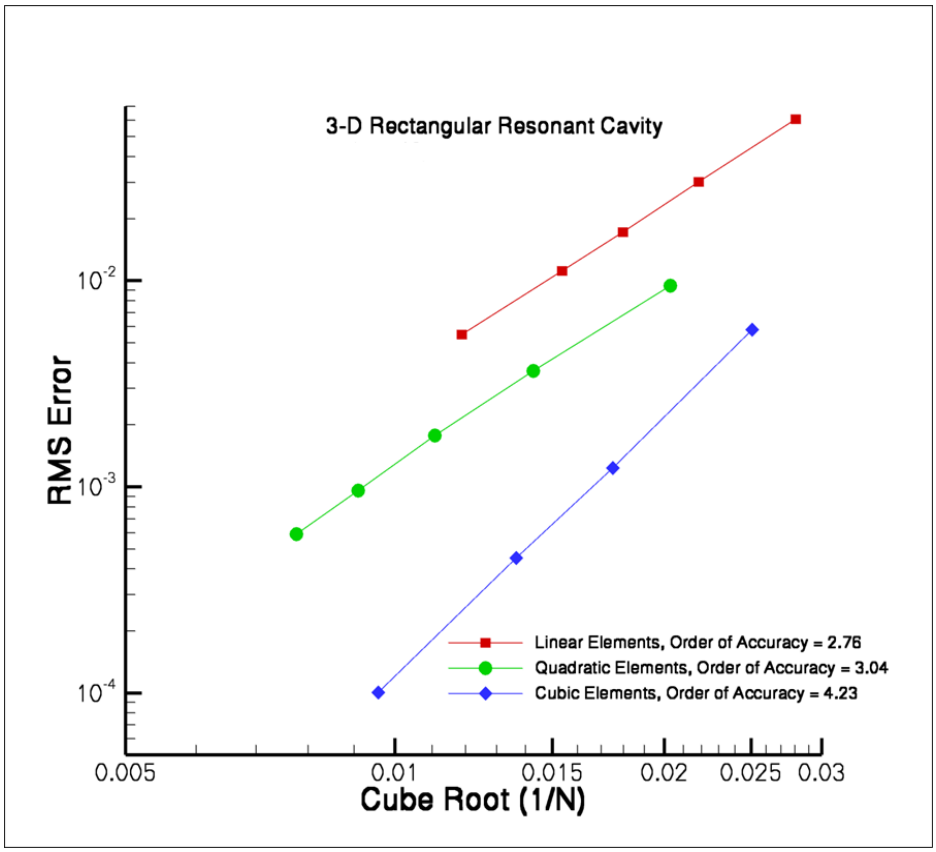

Figure 4.7 Order of Accuracy Study

From Figure 4.7 it is seen that the order of accuracy when linear elements are employed is 2.76, when quadratic elements are employed is 3.04, and when cubic elements are employed is 4.23. When linear elements are employed the order of accuracy is greater than theoretically expected, but when quadratic and cubic elements are employed the order of accuracy is what is theoretically expected.

4.2 Field Simulation Software Timing Comparison

A timing study has been conducted that compares the wall-clock time spent to generate a solution that has a specific RMS error when linear, quadratic, and cubic elements are employed. For this comparison, the specific RMS error is approximately equal to 0.04 after ten steps with $\Delta t = 0.001$. For the baseline case (linear elements), the average time for each of the 64 processes employed to generate this solution is approximately 56.193 seconds, when quadratic elements are employed, the average time per process is approximately 48.040 seconds, and when cubic elements are employed, the average time per process is approximately 33.139 seconds. The computational mesh when linear elements are employed is made up of 595,725 nodes, when quadratic elements are employed is made up of 203,541 nodes, and when cubic elements are employed is made up of 63,519 nodes. When linear elements are employed 4 Gauss points are utilized to integrate over the tetrahedral volume elements, 36 search directions are necessary to drive the residual of the linear system to machine zero at every time step, and the linearization matrix is made up of 8,706,389 non-zeroes. When quadratic elements are employed 16 Gauss points are utilized, 61 search directions are necessary, and the linearization matrix is made up of 4,925,278 non-zeroes. When cubic elements are employed 29 Gauss points are utilized, 65 search directions are necessary, and the linearization matrix is made up of 2,934,931 non-zeroes. This study shows that when higher-order elements are employed to discretize the field, less computational effort is needed to generate a solution at a pre-specified level of accuracy. However, the benefits of employing higher-order elements are somewhat detracted because of the following reasons: as the order of the elements is increased the order of the integration routine has to be increased correspondingly, their linearization matrices are made up of a

relatively large number of non-zeroes, and a comparatively large number of search directions are needed to drive the residual of the linear system to machine zero at every time step.

4.3 Field Simulation Software Applications

Since the order of accuracy study shows that the software is capable of accurately simulating electromagnetic fields, the software can be applied to more complicated problems. The field software can attain the scattering profile of a PEC sphere (grid shown in Figure 4.8, and solution shown in Figure 4.9) or a notional business jet (solution shown in Figure 4.10). Quadratic tetrahedral elements are utilized to discretize both of these electromagnetic fields. For each of these simulations a TE^x plane wave collides with a PEC body. Time dependent field variables corresponding to a TE^x plane wave are enforced in the far field, while the sphere and the business jet are assumed to be PEC. The analytic field equations for a TE^x plane wave are as follows [44]:

$$\overline{D}_x = 0 \tag{4.10}$$

$$\overline{D}_y = \frac{\varepsilon_r}{c_o} A_o cos(\omega t - kx) \tag{4.11}$$

$$\overline{D}_z = 0 \tag{4.12}$$

$$B_x = 0 \tag{4.13}$$

$$B_y = 0 \tag{4.14}$$

$$B_z = \mu \sqrt{\frac{\varepsilon_o}{\mu_o}} A_o cos(\omega t - kx) \tag{4.15}$$

$$\omega = kc \tag{4.16}$$

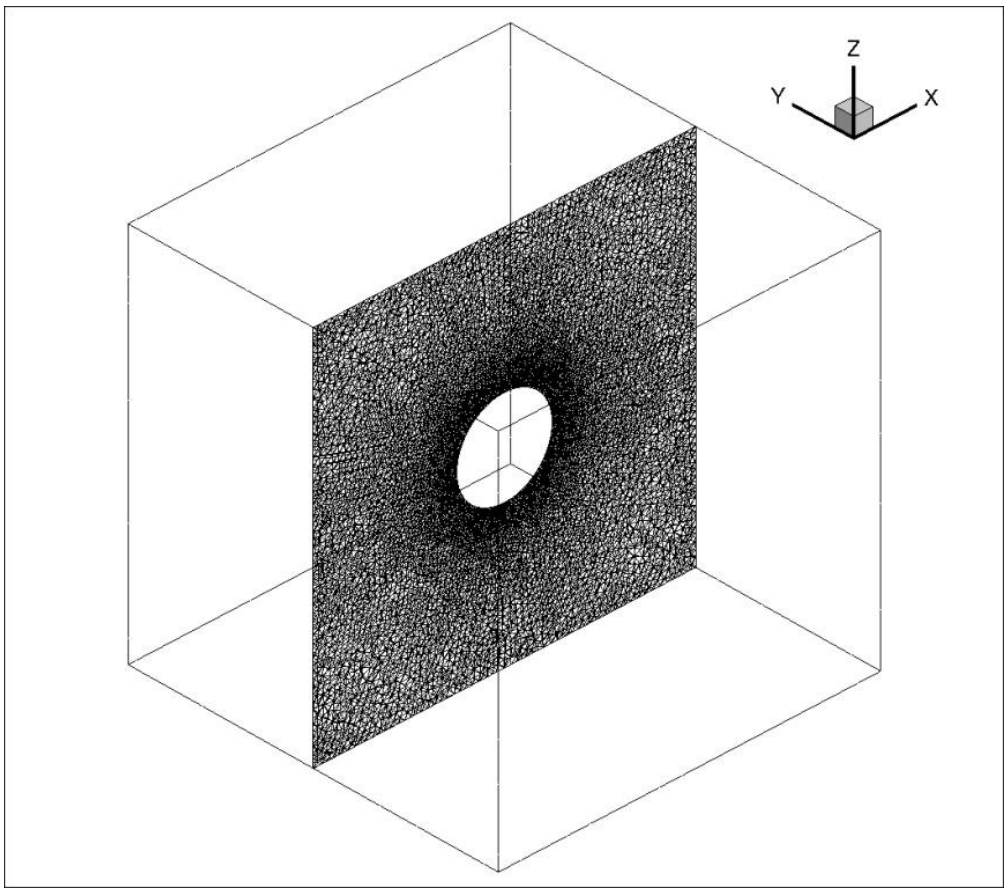

Figure 4.8 Computational Grid for Electromagnetic Scattering from a Sphere

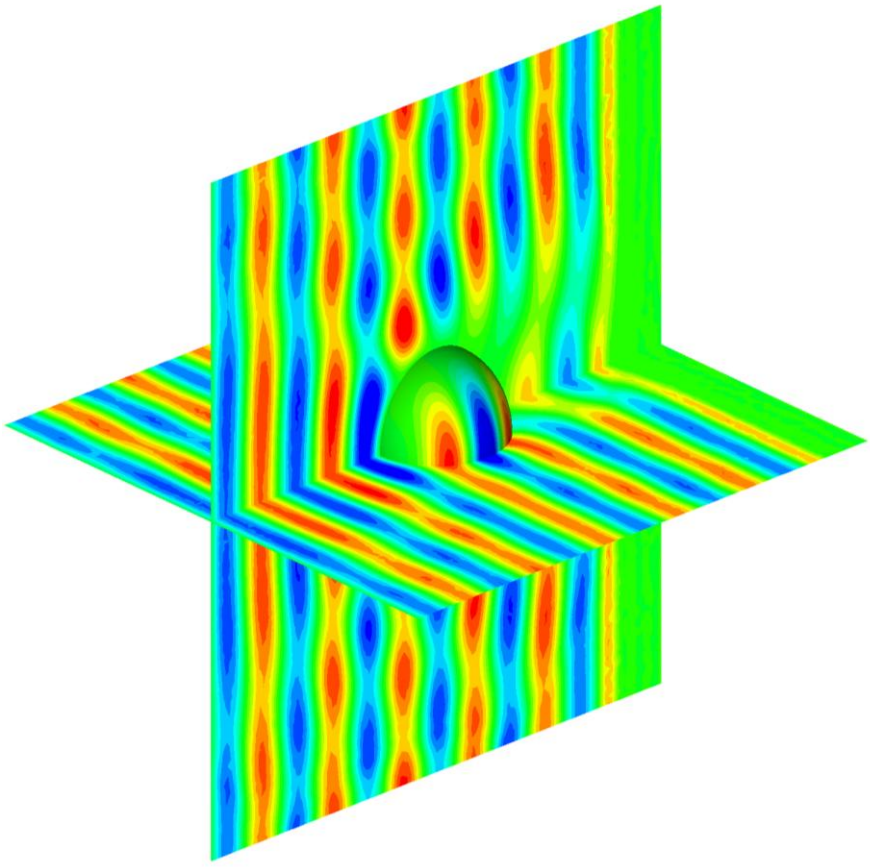

Figure 4.9 Electromagnetic Scattering from a Sphere

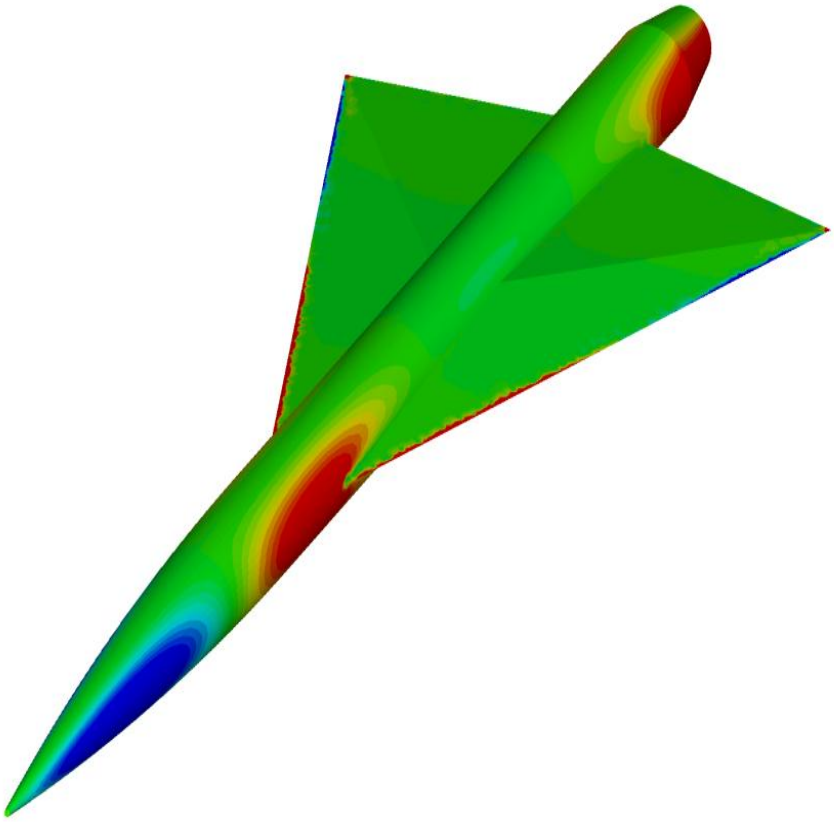

Figure 4.10 Electromagnetic Scattering from a Notional Business Jet

The field simulation software is capable of generating a higher order approximation for a field where the geometries are comprised of dissimilar materials. The following case is the simulation of a TE^x plane wave impinging on a cube with relative permittivity (ε_r) and permeability (μ_r) of 5.0 and 2.0 respectively. The computational mesh for this case is shown in Figure 4.11. Figures 4.12 and 4.13 show the field contours for the $\overline{D}_y$ and B_z fields generated with quadratic elements. The jump condition discussed in section 2.7 is employed at the boundary between the dielectric cube and free space. According to the theory, at the face of the cube that is perpendicular to the incoming TE^x plane wave the jump in $\overline{D}_y$ equals the ratio of relative permittivities between the dielectric material and free space, and the jump in B_z equals the ratio of relative permeabilities between the dielectric material and free space. The accuracy of the jump condition implemented relies on the values of these ratios, and it can be assessed in a post processing step. The post processing step shows that at the center of the face of the cube that is perpendicular to the incoming TE^x plane wave at a distance that is 0.0001 before and aft the face of the cube the interpolated ratio of $\overline{D}_y$ is approximately 5.05, and the interpolated ratio of B_z is approximately 2.02. Also, according to theory, the B_z field contour lines extending from the top and bottom faces of the cube into free space are continuous. Figures 4.12 and 4.13 and the post processing step show that the computed field closely matches the analytical solution; therefore, the jump boundary condition is a proper boundary condition between dissimilar materials.

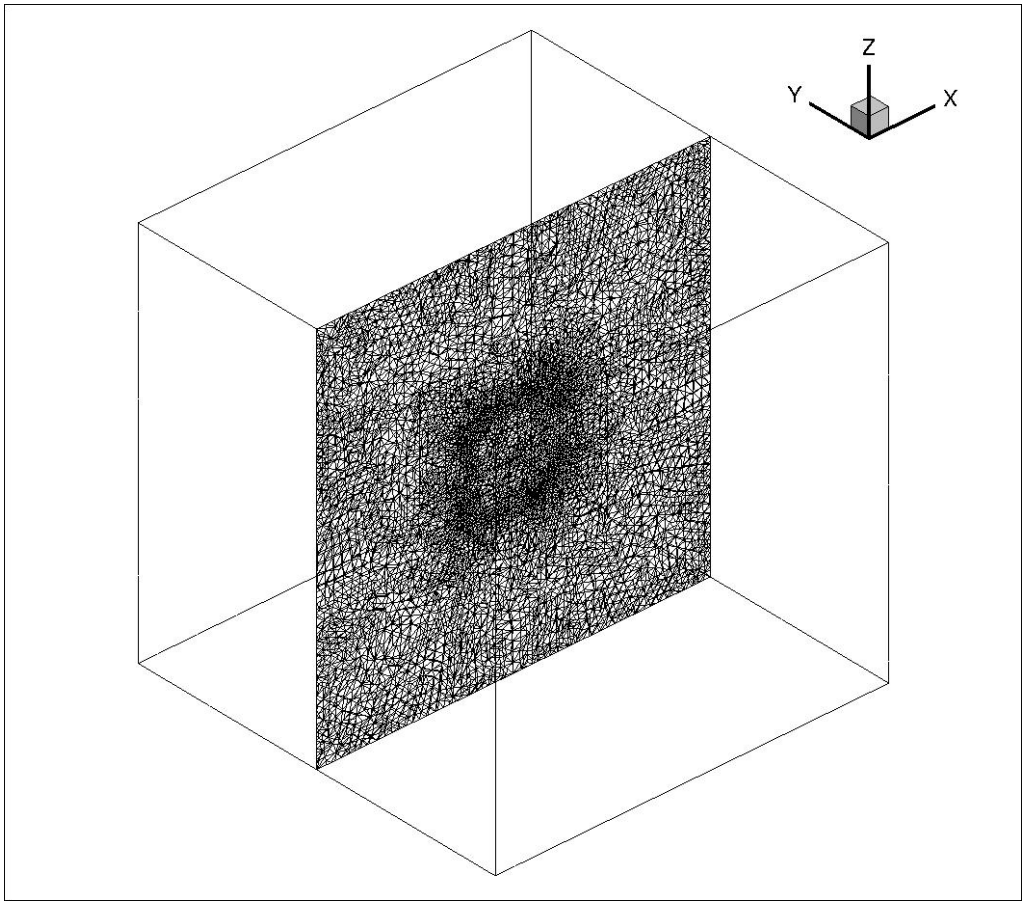

Figure 4.11 Computational Grid for Dielectric Cube Case

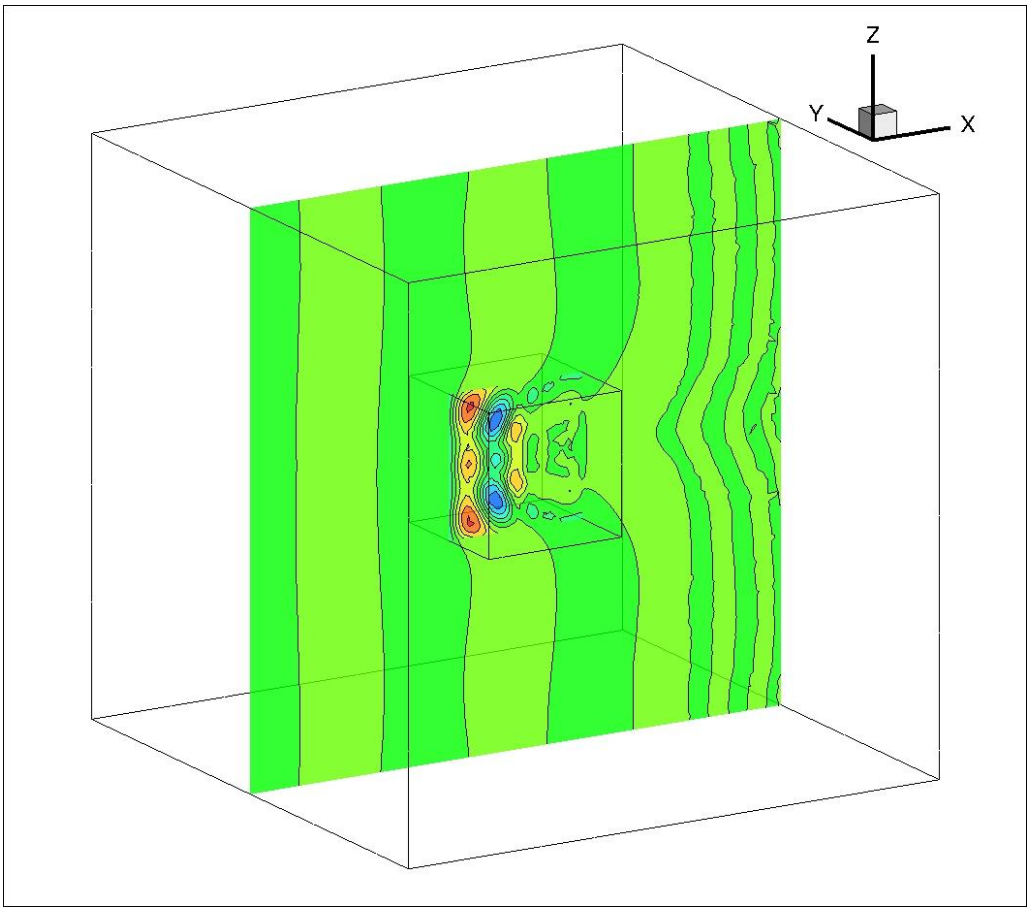

Figure 4.12 $\bar{D}_y$ Contours for Dielectric Cube Case

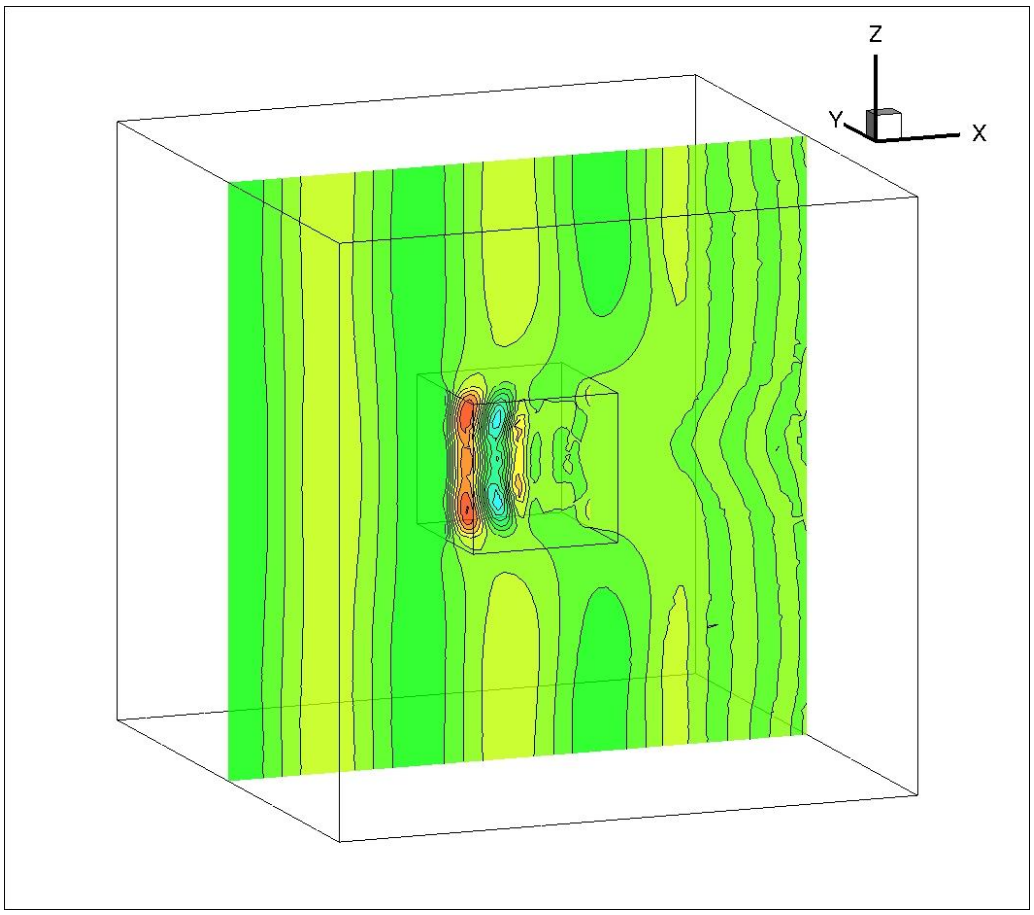

Figure 4.13 B_z Contours for Dielectric Cube Case

The following case is the simulation of a TE^x plane wave impinging on an ellipsoid with relative permittivity (ε_r) and permeability (μ_r) of 5.0 and 2.0 respectively. The equation for the ellipsoid is as follows:

$$\frac{x-0.5}{a}+\frac{y-0.5}{b}+\frac{z-0.5}{c}=1 \tag{4.17}$$

For this case, $a = 0.125$, $b = 0.05$, and $c = 0.125$. The computational mesh for this case is shown in Figure 4.14. Figures 4.15 and 4.16 show the field contours for the $\overline{D}_y$ and B_z fields generated with linear elements after the wave-front has propagated downstream of the ellipsoid. The jump condition discussed in Section 2.7 is employed at the boundary between the dielectric ellipsoid and free space.

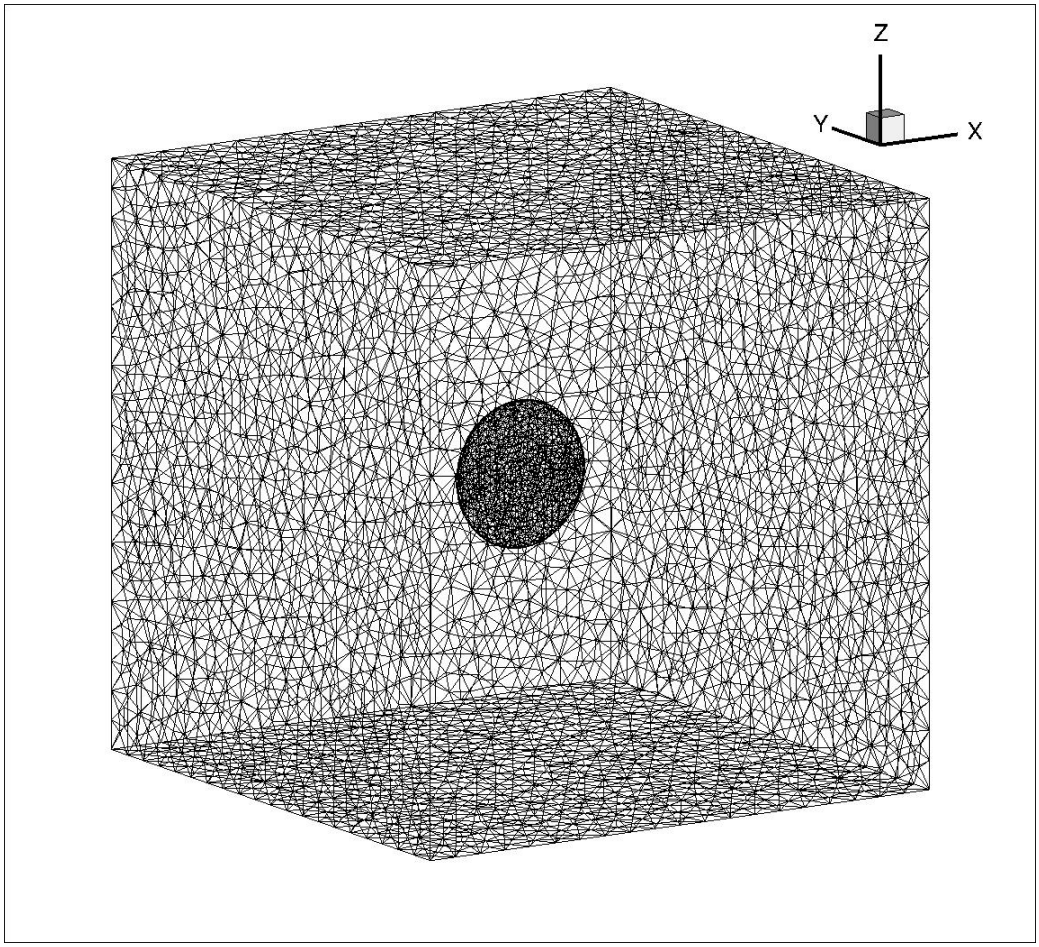

Figure 4.14 Computational Grid for Dielectric Ellipsoid Case

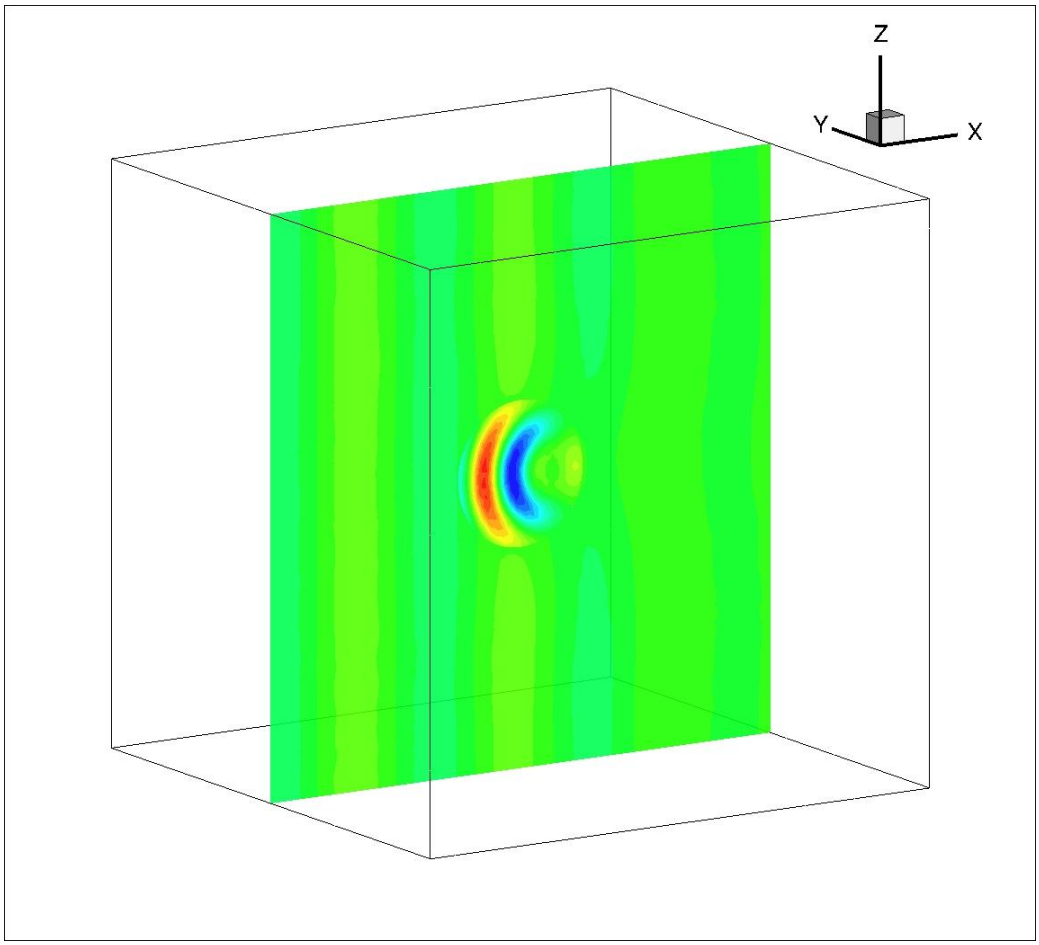

Figure 4.15 $\overline{D}_y$ Contours for Dielectric Ellipsoid Case

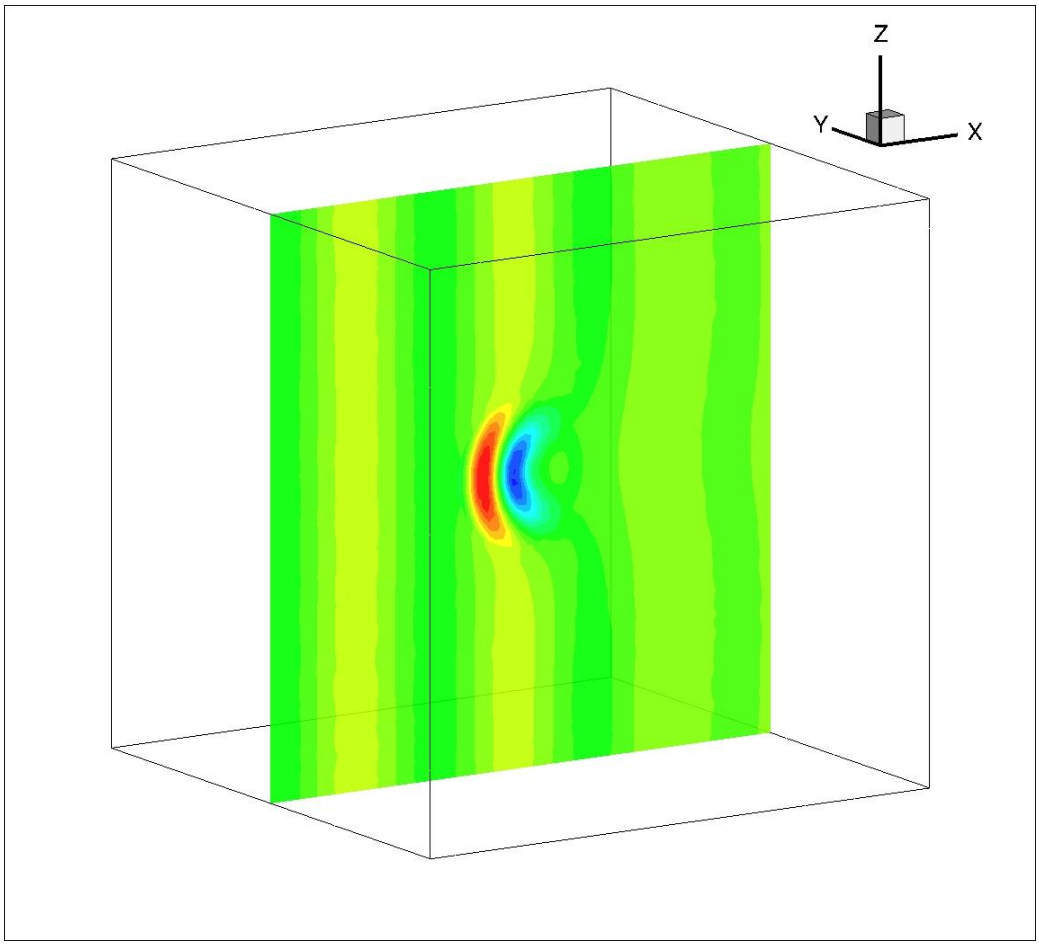

Figure 4.16 B_z Contours for Dielectric Ellipsoid Case

4.4 Verification of Shape Sensitivity Derivatives for a Dielectric Cube

For the case of the TE^x plane wave propagating through a volume that is made up of free space and a cube of dielectric material, the time accurate sensitivity derivatives are computed with three methods to verify the correctness of implementation. The cost function is selected to provide an opportunity to optimize the shape or location of the dielectric cube. For this verification study, the cost function is the normal component of $\overline{D}$ field the integrated over the surface of the cube.

$$I = \iint_{\Gamma} \left(\vec{\overline{D}} \cdot \vec{n} \right) \partial \Gamma \tag{4.18}$$

For this case, there is one design variable β_1 that simultaneously controls 500 nodes on the surface of the cube. For the forward mode complex Taylor series expansion, the $x, y,$ and z values of the 500 nodes are perturbed in the complex plane by machine epsilon. For the forward mode direct differentiation method and the reverse mode adjoint method the mesh sensitivity derivatives $\frac{\partial x}{\partial \beta}, \frac{\partial y}{\partial \beta},$ and $\frac{\partial z}{\partial \beta}$ for the 500 nodes are prescribed a value of machine epsilon. The sensitivity derivative is computed after 500 time steps with a nondimensionalized Δt equal to 0.001. After 500 time steps at a nondimensionalized Δt equal to 0.001, the wave front has propagated to the center of the cube. Table 4.1 shows the comparison of the sensitivity derivative generated from the complex Taylor series approach, the direct differentiation approach, and the discrete adjoint approach.

Table 4.1 Comparison of Sensitivity Derivatives Obtained using the Complex-Variable Approach, Direct Differentiation, and the Adjoint Method for a Dielectric Cube

Approach	$\frac{\partial I}{\partial \beta_1}$**, Linear Elements**
CTSE	-**3.831764**909581178E-002
Direct Differentiation	-**3.831764**939491016E-002
Adjoint	-**3.831764**939452845E-002

Table 4.1 shows that the direct and adjoint approaches match the CTSE approach to eight decimal places when linear elements are employed. This verifies that the adjoint approach is implemented correctly and can be used to reposition a dielectric cube.

4.5 Verification of Shape Sensitivity Derivatives for a Dielectric Ellipsoid

A similar verification study was conducted for the case of the TE^x plane wave propagating through a volume that is made up of free space and an ellipsoid of dielectric material. The cost function is shown in Equation 4.18. There is also one design variable for this case, and it is a from Equation 4.17. For the forward mode CTSE, a is perturbed in the complex plane by machine epsilon. That change in the shape of the surface is propagated through the mesh from the usage of software that solves the linear elastic equations in complex variable form. For the direct differentiation method and the adjoint method, the mesh sensitivity derivatives are computed with the software that solves the linear elastic equations. The sensitivity derivative is computed after 10 time steps with a non-dimensionalized Δt equal to 0.001. The sensitivity derivative is computed with linear, quadratic, and cubic elements employed, and the results are shown in Table 4.2.

Table 4.2 Comparison of Sensitivity Derivatives Obtained using the Complex-Variable Approach, Direct Differentiation, and the Adjoint Method for a Dielectric Ellipsoid

Approach	$\frac{\partial I}{\partial \beta_1}$**, Linear Elements**
CTSE	**1.245565**539259534E-010
Direct Differentiation	**1.2455649**04407576E-010
Adjoint	**1.2455649**04185009E-010
Approach	$\frac{\partial I}{\partial \beta_1}$**, Quadratic Elements**
CTSE	**5.092297**186846612E-013
Direct Differentiation	**5.0922965**42609352E-013
Adjoint	**5.0922966**55362702E-013
Approach	$\frac{\partial I}{\partial \beta_1}$**, Cubic Elements**
CTSE	**-6.399429**387536681E-009
Direct Differentiation	**-6.399428**369219613E-009
Adjoint	**-6.399429**126136743E-009

Table 4.2 shows that the direct and adjoint approaches match the CTSE approach to seven decimal places when linear, quadratic, and cubic elements are employed. This verifies that the adjoint approach is implemented correctly and can be used to modify the shape of a dielectric ellipsoid.

4.6 Shape Design Optimization Applications

The shape design optimization process discussed in Section 3.7 is applied to determine the placement of a dielectric cube within a volume of free space discretized with linear elements so that the electric flux density of the surface corresponds to the electric flux density computed at the same time but in a different position. The cost function is:

$$I = \iint_{\Gamma} \left(\left(\vec{\overline{D}} - \vec{\overline{D}}^{*} \right) \cdot \vec{n} \right) \partial \Gamma \tag{4.19}$$

In Equation 4.19, $\vec{\overline{D}}$ is the electric flux density on the surface at the current step in the design cycle, and $\vec{\overline{D}}^{*}$ is the target electric flux density on the surface.

There is one design variable, β_1, and it is the surface of the dielectric cube. The outer volume of free space is a cube that has a non-dimensional length, width, and height of 1.0, and the dielectric cube has a non-dimensional length, width, and height of 0.25. The "*" location of the dielectric cube is at the center of the volume of free space. When the dielectric cube is at the "*" location it spans from 0.375 – 0.625 in the $x, y,$ and z directions. The dielectric cube has an ε_r equal to 5.0 and an μ_r equal to 2.0, and the field contours for the "*" location are shown in Figures 4.12 and 4.13. The purpose of the design optimization routine is to move the dielectric cube from a starting location to the "*" location. Initially, the cube is displaced -0.1 in the x direction from the "*" location. When the dielectric cube is at the starting location it spans from 0.275 – 0.525 in the x direction, and from 0.375 – 0.625 in the y and z directions. Table 4.3 shows the cost function, sensitivity derivative, and x location of the start of the dielectric cube at each step of the design cycle as output by the PORT optimization library. The cost function is generated with the electromagnetic field simulation software, and the sensitivity derivative is

generated with the time accurate discrete adjoint solver. The cost function and the sensitivity derivative are generated after 800 time steps with a nondimensional time step of 0.001. At each time step the unsteady residual from the field solver and the adjoint variable from the adjoint solver are below machine zero. The x location of the start of the dielectric cube is computed with the Port optimization library with the cost function and sensitivity derivative as inputs.

Table 4.3 Design Cycle for the Positioning of a Dielectric Cube

Cycle #	Cost Function, I	Sensitivity Derivative, $\frac{\partial I}{\partial \beta_1}$	Cube starting location, x
1	6.622883818E-004	-1.033402E-003	0.2760334
2	6.604531451E-004	-1.308094E-003	0.2891143
3	6.036303621E-004	-4.436365E-003	0.395
4	2.970814484E-005	1.776709E-003	0.36472
5	9.987271703E-006	-1.378919E-003	0.37795
6	3.014413579E-008	3.616012E-004	0.375203
7	2.055952198E-008	4.026902E-005	0.3748585
8	2.053529086E-008	-1.652856E-006	0.3748721
9	2.053529086E-008	6.795675E-009	0.3748721

Table 4.3 shows that after 9 design cycles the shape design optimization routine has moved the dielectric cube to the "*" location. Figures 4.17 and 4.18 graphically show the movement of the dielectric cube during the design cycle. In Figures 4.17 and 4.18 the initial position of the

dielectric cube is shown in red, the position after 3 cycles is shown in black, and the final position is shown in blue.

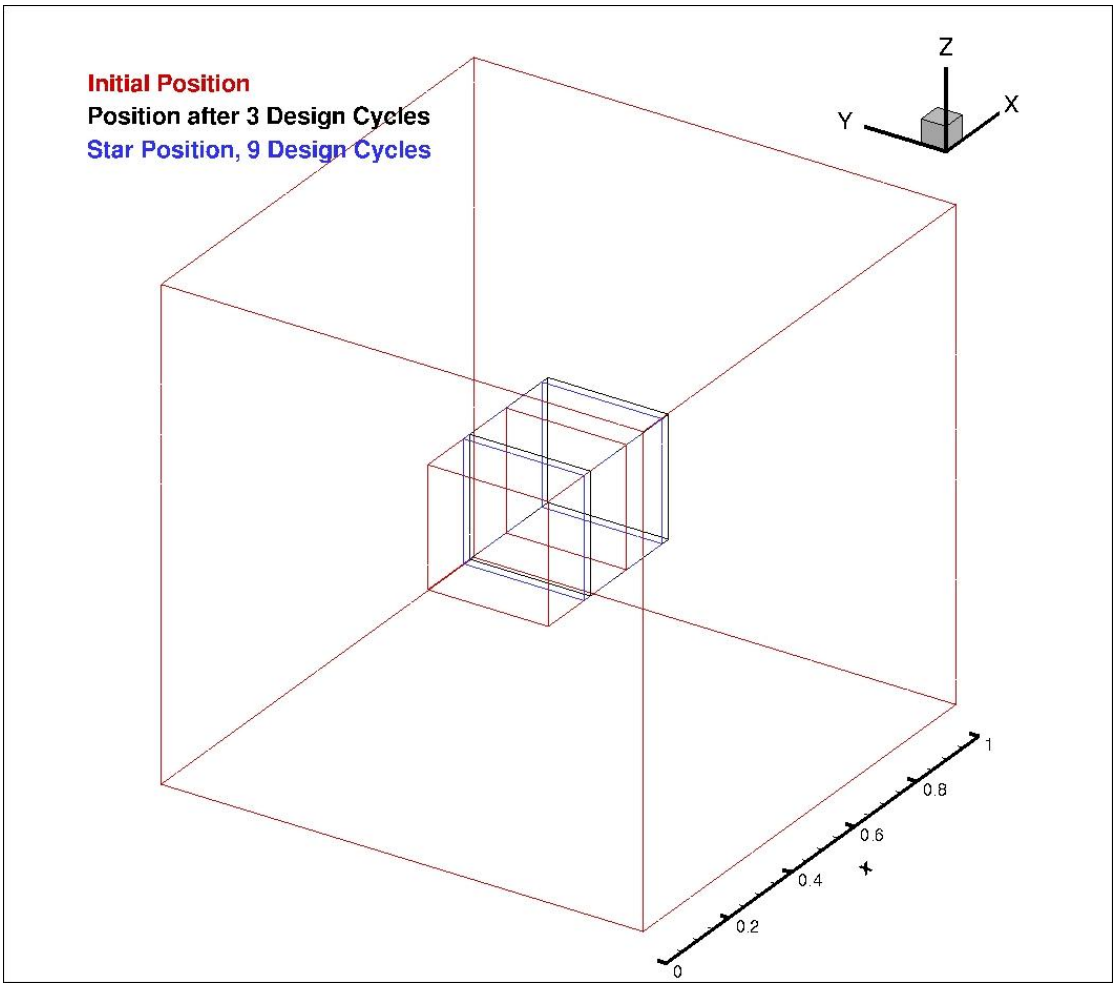

Figure 4.17 Movement of the Dielectric Cube during the Design Cycle (3D view)

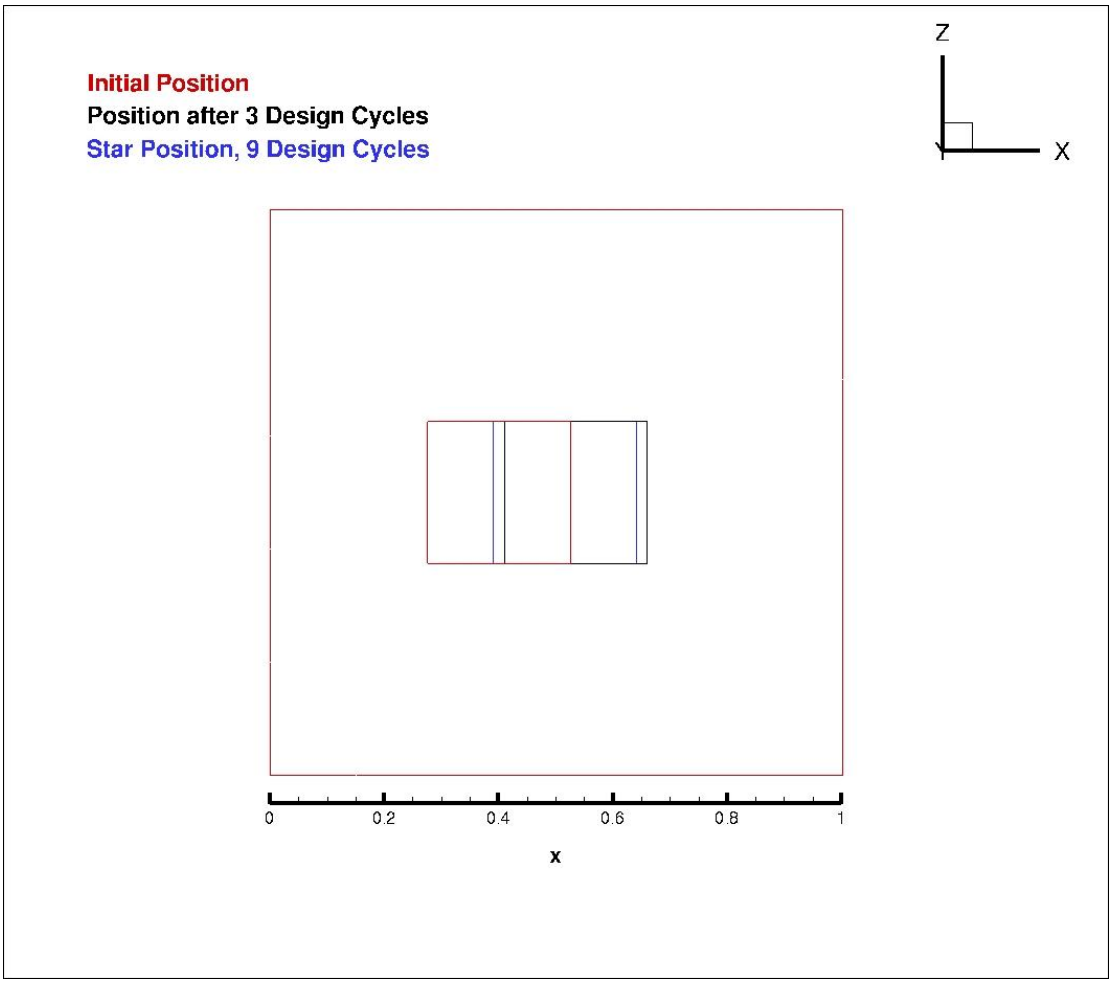

Figure 4.18 Movement of the Dielectric Cube during the Design Cycle (2D view)

The shape design optimization process is also applied to determine the shape of a dielectric ellipsoid within a volume of free space discretized with linear elements so that the electric flux density of the surface corresponds to the electric flux density computed at the same time but in a different position. The cost function is shown in Equation 4.19.

There are three design variables, $\beta_1 = a, \beta_2 = b$, and $\beta_3 = c$. Initially, $a = 0.125, b = 0.05,$ and $c = 0.125$. The "*" shape is a sphere where $a, b,$ and c are all equal to 0.1. The computational mesh and the field contours for the initial shape are shown in Figures 4.14-4.16. Table 4.4 shows the cost function, sensitivity derivatives, and value of the total change of the design variables at each step of the design cycle as output by the PORT optimization library. The cost function is generated with the electromagnetic field simulation software, and the sensitivity derivatives are generated with the time accurate discrete adjoint method. The cost function and the sensitivity derivatives are generated after 800 time steps with a non-dimensional time step of 0.001. The design variables are computed with the PORT optimization library with the cost function and sensitivity derivative as inputs.

Table 4.4 Design Cycle for the Shape Design of a Dielectric Ellipsoid

Cycle #	I	$\frac{\partial I}{\partial \beta_1}$	Total Δa	$\frac{\partial I}{\partial \beta_2}$	Total Δb	$\frac{\partial I}{\partial \beta_3}$	Total Δc
1	4.29087E-4	6.09286E-3	0	-5.9749E-3	0	2.835579E-3	0
2	3.23724E-4	5.29928E-3	-6.09286E-3	-5.84851E-3	5.9749E-3	2.153622E-3	-2.835579E-3
3	2.18947E-6	-5.0770E-4	-3.00000E-2	5.81361E-4	4.9351E-2	1.54296E-4	-1.930157E-2
4	1.26281E-6	1.49737E-4	-2.80751E-2	-2.98902E-4	4.5971E-2	1.4319E-4	-1.834881E-2
5	8.66984E-7	-7.1478E-5	-2.86343E-2	3.25383E-7	4.7189E-2	1.3886E-4	-1.884832E-2
6	8.16032E-7	-8.1779E-5	-2.85694E-2	1.68155E-5	4.7332E-2	1.34527E-4	-1.905373E-2
7	3.01985E-7	-1.0085E-4	-2.73548E-2	7.92124E-5	4.8632E-2	7.7104E-5	-2.16354E-2
8	1.94025E-8	-5.0392E-5	-2.56484E-2	5.72931E-5	4.9847E-2	9.98986E-6	-2.46875E-2
9	2.15284E-9	-1.6505E-5	-2.52923E-2	1.89020E-5	4.9902E-2	-8.7488E-7	-2.51363E-2

Table 4.4 shows that after 9 design cycles, the cost function is minimized and $a \approx 0.099707, b \approx 0.099902$, and $c \approx 0.0998637$. This means that the shape design process morphed the ellipsoid to the "*" shape, which is the shape of a sphere. Figures 4.19-4.21 show the computational mesh and the field contours for the final design outputted from the design cycle.

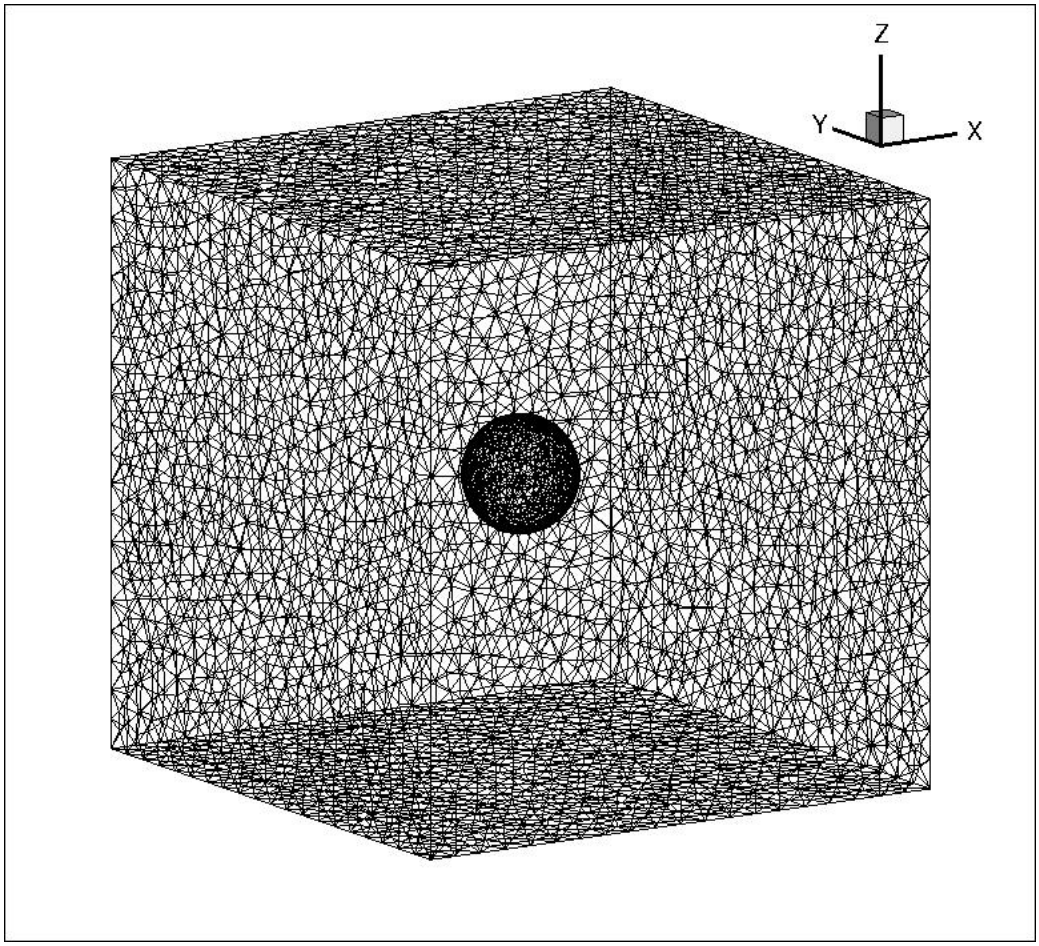

Figure 4.19 Computational Grid for Dielectric Sphere Case

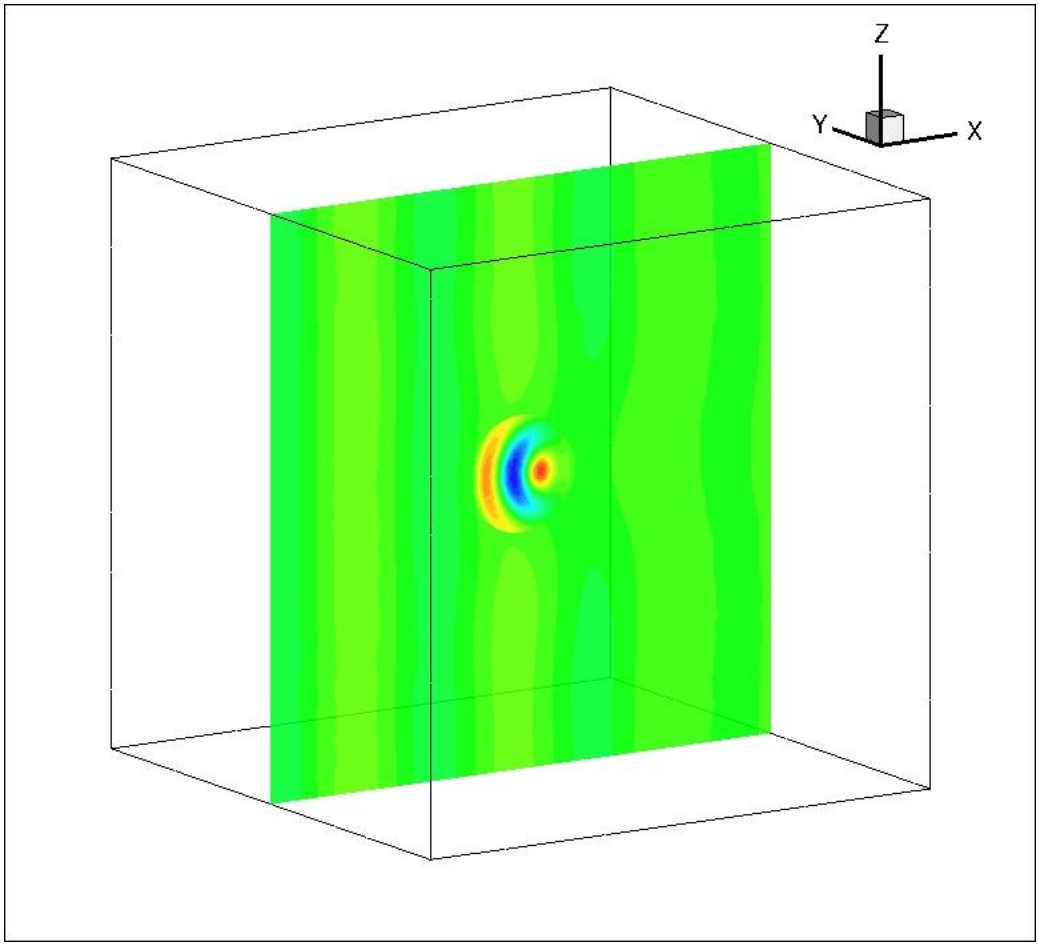

Figure 4.20 $\overline{D}_y$ Contours for Dielectric Sphere Case

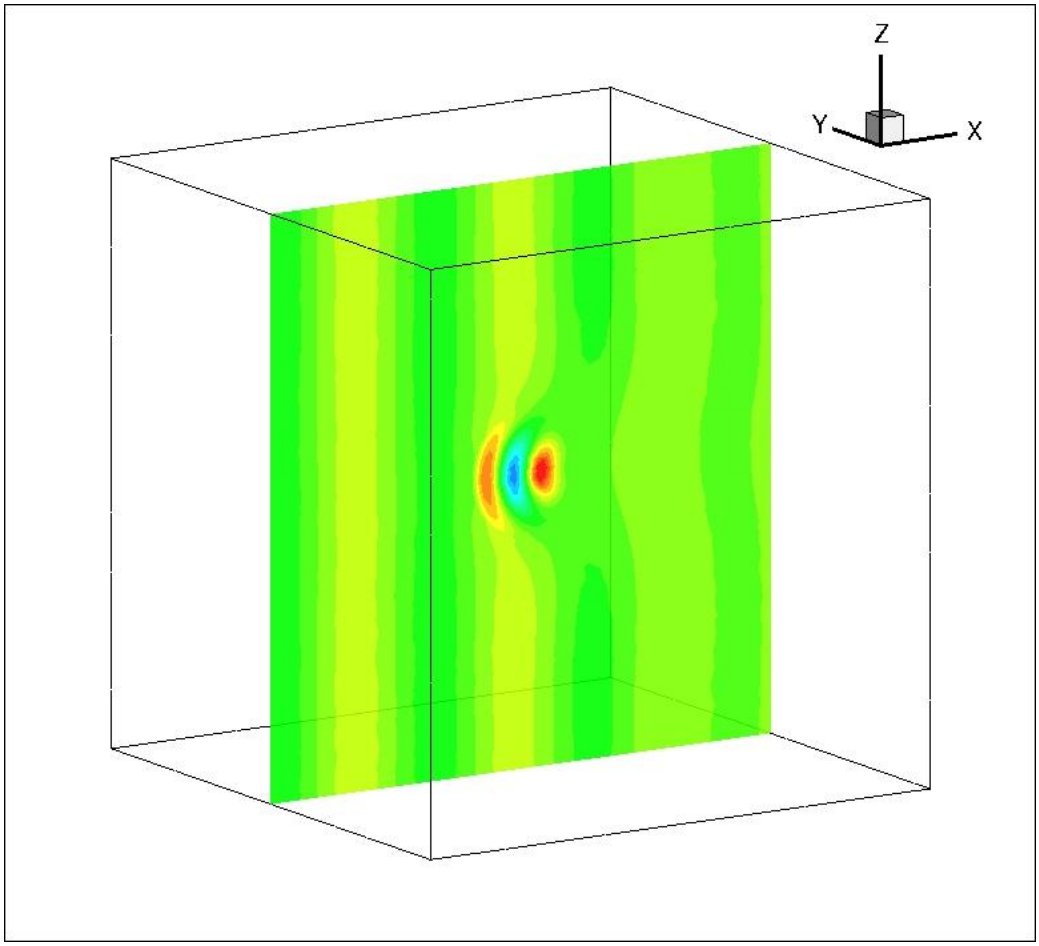

Figure 4.21 B_z Contours for Dielectric Sphere Case

CHAPTER V

CONCLUSION

A novel approach to the optimization of the shape of large electromagnetic structures has been implemented. Through the course of this implementation, a temporally and spatially numerical electromagnetic field simulation software, a software that uses the reverse-mode time-accurate discrete-adjoint method to compute the sensitivity derivatives, and a software that generates a higher-order numerical solution to the linear elastic equations have been developed from scratch and coupled together with an optimization library in order to automatically design the shape of large electromagnetic objects. The pieces of software developed have been written in a parallel message passing paradigm, and are capable of being executed on a distributed memory supercomputer. Because of this, larger complicated electromagnetic objects can be stored in memory and designed at a faster rate.

The electromagnetic field simulation software developed can accurately and efficiently approximate the electromagnetic field inside of and outside of complicated, large, 3D electromagnetic objects. Many complicated electromagnetic systems are operated at very high frequencies (~10 GHz), and higher-order elements can be utilized to lessen the grid requirements necessary to approximate an electromagnetic wave at high frequencies. To that end, the higher order accurate SU/PG method is utilized to simulate the electromagnetic field in a volume that is discretized with tetrahedra. The usage of tetrahedra to discretize the field leads to easier mesh

generation, and the ability to properly discretize the surface of curved electromagnetic objects. The mesh generation process discretizes the volume of the field with linear tetrahedra, and special care is taken to add the extra edge and face nodes used for spatial higher-order accuracy in a fast and efficient manner. This is accomplished by generating edge to element maps that for a given node added to an edge, gives a list of elements that own that edge and need to be notified of the added node. Implicit time stepping is utilized, and it has two benefits. The first is the ability to use a time step that depends on the physics of the problem, rather than a time step used that governs the numerical stability of the solver. The second is the linearization matrix generated for the implicit time stepping routine is also used in the software that computes the sensitivity derivatives with the time-accurate discrete-adjoint method. A special boundary condition, called a jump condition, is employed to allow for the simulation of a field that is made up of dissimilar materials. Through the usage of a grid study the field software's accuracy has been verified to match or exceed theoretical orders of accuracy for the elements employed. The grid study was the comparison of the computed solution, generated with linear, quadratic, and cubic elements, of the electromagnetic field within a rectangular resonant cavity with the exact solution with multiple grid sizes. The field software also properly simulated the propagation of a 3D wave through dissimilar materials, and the scattering of a 3D wave from large complicated PEC objects.

The sensitivity derivatives are computed with a reverse-mode time-accurate discrete-adjoint method that can accurately generate shape sensitivity derivatives of large, curved, 3D electromagnetic objects. The time accurate sensitivity derivatives are computed and can be used to quickly design an object that operates at a wide band of frequencies. The time accurate sensitivity derivatives have to be computed once for a frequency range while frequency domain

sensitivity derivatives would have to be computed for each frequency in the range. Because the discrete-adjoint method is used to compute the sensitivity derivatives, the software can be quickly adapted to generate the shape sensitivity derivatives of multiple cost functions, and is capable of generating the sensitivity derivatives for a large set of design variables without having to solve a linear system for each design variable considered. The sensitivity derivatives software is automatically concurrent with the field software because it generates the derivatives that make up its linear systems by the complex Taylor series expansion method. The sensitivity derivatives generated from the reverse-mode discrete-adjoint method have been verified to match the sensitivity derivatives generated from the forward-mode complex Taylor series expansion method and direct method to at least seven decimal places.

The software that generates a higher-order numerical solution to the 3D linear elastic equations can smooth a mesh that is subject to large deformations from the design process, and can generate the mesh sensitivity derivatives that are used by the sensitivity derivatives software to generate the overall shape sensitivity derivatives. The motion of the higher order edge and face nodes is solved for, rather than simply interpolating their motion. The linear elastic equations software has been used to smooth meshes with deformations up to 10% of the overall length of the mesh. After the smoothing process is complete, the mesh generated has well formed elements and is capable of being used by the other software to generate the electromagnetic field and the sensitivity derivatives.

The shape design optimization procedure has been implemented to automatically place a dielectric cube and modify the shape of a dielectric ellipsoid. Both objects are within a larger volume of free space, and are subjected to the propagation of an unsteady plane wave. The cost function is defined as dielectric object's surface integration of the current electric flux density

field minus the electric flux density field of the predetermined location/shape dotted with the unit normal vector of the surface. For each step of the design cycle, the cost function is generated with the field simulation software, the sensitivity derivatives are generated with the time accurate discrete adjoint method, and the mesh is smoothed by software that generates a numerical solution to the linear elastic equations. The dielectric cube was automatically placed by the optimization procedure to its predetermined location in nine design cycles, and the dielectric ellipsoid was automatically morphed by the optimization procedure to its predetermined shape in nine design cycles as well. The execution of this optimization procedure shows that it is capable of automatically designing in the time domain any large, curved 3D electromagnetic shape, including one that operates at high frequencies or is made of dissimilar materials, in an efficient manner.

BIBLIOGRAPHY

1. Balanis, C. A., *Advanced Engineering Electromagnetics*, John Wiley & Sons, New York, 1989.

2. Yee, K. S., "Numerical Solution of Initial Boundary Value Problem Involving Maxwell's Equations in Isotropic Media," *IEEE Transactions on Antennas and Propagation*, Vol. 14, No. 3, pp. 302-307, 1966.

3. Hughes, T. J. R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice-Hall, Inc., New Jersey, 1987.

4. Zienkiewicz, O. C., *The Finite Element Method*, McGraw-Hill Book Company (UK) Limited, London, 1977.

5. Jin, J., *The Finite Element Method in Electromagnetics*, John Wiley & Sons, 2nd Ed., New York, 2002.

6. Hughes, T. J. R. and Mallet, M., "A New Finite Element Formulation for Computational Fluid Dynamics III. The Generalized Streamline Operator for Multidimensional Advective-Diffusive Systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 58, No. 3, pp. 305-328, 1986.

7. Venkatakrishnan, V., Allmaras, S. R., Johnson, F. T., and Kamenetskii, D. S., "Higher Order Schemes for the Compressible Navier-Stokes Equations," *AIAA 2003-3987, Presented at the 16th AIAA Computational Fluid Dynamics Conference, Orlando, FL, June 23-26*, 2003.

8. Anderson, W. K., Wang, L., Kapakia, S., Tanis, C., and Hilbert B., "Petrov-Galerkin and Discontinuous-Galerkin Methods for Time-Domain and Frequency-Domain Electromagnetic Simulations," UTC-CECS-SimCenter-2011-01, February 2011.

9. Jameson, A., "Aerodynamic Design Via Control Theory," *J. Sci. Computing*, Vol. 3, pp. 233-260, 1988.

10. Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grid with a Continuous Adjoint Formulation," *35th Aerospace Science Meeting & Exhibit, AIAA 97-0643*, 1997.

11. Burdyshaw, C. E., "Achieving Automatic Concurrency Between Computational Field Solvers and Adjoint Sensitivity Codes," Ph. D. Thesis, University of Tennessee at Chattanooga, May 2006.

12. Li, S. and Petzold, L., "Adjoint Sensitivity Analysis for Time-Dependent Partial Differential Equations with Adaptive Mesh Refinement," *J. of Computational Physics*, Vol. 198, pp. 310-325, 2004.

13. Director, S. W. and Rohrer, R. A., "The Generalized Adjoint Network and Network Sensitivities," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 318-323, Aug. 1969.
14. Director, S. W. and Rohrer, R. A., "Automated Network Design-The Frequency-Domain Case," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 330-337, Aug. 1969.

15. Tellegen, B. D. H., "A General Network Theorem with Applications," *Philips Res. Rep.*, vol. 7, pp. 259-269, 1959.

16. Penfield, Jr., P., Spence, R., Duinker, S., "A Generalized Form of Tellegen's Theorem," *IEEE Trans. Circuit Theory*, vol. CT-17, pp. 302-305, Aug. 1970.

17. Sabbagh, M. A., Bakr M. H., and Nikolova, N., K., "Sensitivity Analysis of the Scattering Parameters of Microwave Filters using the Adjoint Network Method," *Int. J. RF and Microwave CAE 16: 596-606*, 2006.

18. Kang, N., Chung, Y., Cheon, C., and Jung, H., "A New 2-D Image Reconstruction Algorithm Based on FDTD and Design Sensitivity Analysis," *IEEE MTT-S Digest*, pp. 1143-1146, 2002.

19. Chung, Y., Cheon, C., Park, I., and Hahn, S., "Optimal Shape Design of Microwave Device Using FDTD and Design Sensitivity Analysis," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 48, No. 12, pp. 2289-2296, 2000.

20. Chung, Y., Cheon, C., Park, I., Hahn, S., "Optimal Design Method for Microwave Device Using Time Domain Method and Design Sensitivity Analysis-Part II: FDTD Case," *IEEE Transactions on Magnetics*, Vol. 37, No. 5, pp. 3255-3259, 2001.

21. Rickard, Y., Georgieva, N., and Tam, H., "Absorbing Boundary Conditions for Adjoint Problems in the Design Sensitivity Analysis with the FDTD Method," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 51, No. 2, pp. 526-529, 2000.

22. Berenger, J., "A Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *J. Computational Phys*., vol. 114, pp. 185-200, Oct. 1994.

23. Chung, Y., Cheon, C., Park, I., Hahn, S., "Optimal Design Method for Microwave Device Using Time Domain Method and Design Sensitivity Analysis-Part I: FETD Case," *IEEE Transactions on Magnetics*, Vol. 37, No. 5, pp. 3289-3293, 2001.

24. Akcelik, V., Biros, G., Ghattas, O., Keyes, D., Ko, K., Lee, L., and Ng, E. G., "Adjoint Methods for Electromagnetic Shape Optimization of the Low-Loss Cavity for the International Linear Collider," *Journal of Physics: Conference Series 16*, pp. 435-445, 2005.

25. Georgieva, N., Glavic, S., Bakr, M., and Bandler, J., "Feasible Adjoint Sensitivity Technique for EM Design Optimization," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 50, No. 12, pp. 2751-2758, 2001.

26. Nair, D. and Webb, J., "Adaptive Optimization of Microwave Devices over a Frequency Band," *Electromagnetics*, Vol. 30, pp. 177-189, 2010.

27. Nair, D. and Webb, J., "Optimization of Microwave Devices Using 3-D Finite Elements and the Design Sensitivity of the Frequency Response," *IEEE Transactions on Magnetics*, Vol. 39, No. 3, pp. 1325-1328, 2003.

28. Webb, J., "Design Sensitivities Using High-Order Tetrahedral Vector Elements," *IEEE Transactions on Magnetics*, Vol. 37, No. 5, pp. 3600-3603, 2001.

29. Toivanen, J. I., Makinen, R. A. E., Rahola, J., Jarvenpaa, S., and Yla-Oijala, P., "Gradient-Based Shape Optimization of Ultra-Wideband Antennas Parameterized Using Splines," *IET Microwaves, Antennas & Propagation*, Vol. 4, Is. 9, pp. 1406-1414, 2010.

30. Jinyun, Y., "Symmetric Gaussian Quadrature Formulae for Tetrahedral Regions," *Computer Methods in Applied Mechanics and Engineering*, Vol. 43, pp. 349-353, 1984.

31. Barth, T. J., "Numerical Methods for Gasdynamic Systems on Unstructured Meshes," *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, Vol. 5, Springer, pp. 195-285, 1998.

32. http://mathworld.wolfram.com/MatrixInverse.html .

33. Tannehill, J. C., Anderson, D. A., and Pletcher, R. H., *Computational Fluid Mechanics and Heat Transfer*, 2nd Ed., Taylor & Francis, Philadelphia, 1997.

34. Cowper, G. R., "Gaussian Quadrature Formulas for Triangles," *International Journal of Numerical Methods in Engineering*, Vol. 7, pp. 405-408, 1973.

35. Bonhaus, D. L., "A Higher Order Accurate Finite Element Method for Viscous Compressible Flows," Ph. D. Thesis, Virginia Polytechnic Institute and State University, November 1998.

36. Roe, P. L., "The Use of the Riemann Problem in Finite-Difference Schemes," *Lect. Notes Phys.*, Vol. 141, Springer-Verlag, New York, pp. 354-359, 1980.

37. Newman, J. C., Anderson, W. K., and Whitfield, D. L., "Multidisciplinary Sensitivity Derivatives Using Complex Variables," MSSU-COE-ERC-98-08, June 1998.

38. Saad, Y. and Schultz, M. H., "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, pp. 856-869, 1986.

39. Hyams, D. G., "An Investigation of Parallel Implicit Solution Algorithms for Incompressible Flows on Unstructured Topologies," Ph. D. Thesis, Mississippi State University, May 2000.

40. http://glaros.dtc.umn.edu/gkhome/views/metis .

41. Message Passing Interface Forum, "MPI: A Message Passing Interface Standard," *Technical Report UT-CS-94-230*, 1994.

42. Karman, S. L., "Unstructured Viscous Layer Insertion Using Linear-Elastic Smoothing," *AIAA Journal*, Vol. 45, No. 1, pp. 168-180, 2007.

43. http://www.netlib.org/port/ .

44. Orfanidis, S. J., *Electromagnetic Waves and Antennas*, Rutgers University, www.ece.rutgers.edu/~orfanidi/ewa , 1999-2010.

VITA

Ryan S. Glasby was born January 12, 1981 in Boise, Idaho. He graduated from Maumee High School in May of 1999. He received a B.S. and M.S. in Aeronautical and Astronautical Engineering from The Ohio State University in June of 2003 and March of 2006. In January of 2006, he began working on his Ph. D. in computational engineering at the UTC SimCenter: National Center for Computational Engineering. In February of 2011, he began working at Arnold Engineering and Development Center in the Computational Simulation Modeling and Analysis Section. Ryan currently works at AEDC.