Multiple Region Finite-Difference Time-Domain Modeling of Duct Cavities

by

Beijia Zhang

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2004

Author ....
Department of Electrical Engineering and Computer Science
December 12, 2004

Certified by ....
Robert T. Atkins
Lincoln Lab, Associate Group Leader
Thesis Supervisor

Certified by ....
Jin Au Kong
Professor
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Multiple Region Finite-Difference Time-Domain Modeling of Duct Cavities

by

Beijia Zhang

Submitted to the Department of Electrical Engineering and Computer Science
on January 30, 2004, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Engineering

Abstract

Although many radar cross section prediction techniques exist, none have proven to be completely satisfactory when applied to large cavities. Exact numerical techniques can accurately predict RCS, but are too computationally expensive to be used for many cavity geometries. High frequency techniques are computationally efficient but often are inaccurate in predicting the RCS of cavities. This inaccuracy becomes particularly apparent when the wideband range resolved signature is desired. To overcome these limitations, this thesis investigates the possibility of modeling large duct cavities in a piecewise manner using a finite-difference time-domain approach, modified to successively model individual subsections of the cavity. This change improves the computational efficiency of FD-TD while maintaining a high level of accuracy.

Thesis Supervisor: Robert T. Atkins
Title: Lincoln Lab, Associate Group Leader

Thesis Supervisor: Jin Au Kong
Title: Professor

Acknowledgments

This thesis is the result of more than three years of study and work, and is a significant milestone in my education. However, it would not have been possible without the support of many people.

I would first like to acknowledge Dr. Robert Atkins at MIT Lincoln Laboratory for taking me in as an inexperienced sophomore and helping me establish my independent research experience. His willingness to go above and beyond the call of duty was appreciated more than words can express. And his dedication to research is truly admirable. I can only hope I might someday be as respected in my profession as he is.

Secondly, I would like to thank Professor Jin Au Kong for being a dynamic teacher whose grasp of his material was truly deep and unshakable. I hope working with him in the future will help me acquire even just a small portion of his confidence and abilities in the world of electromagnetics.

Thirdly, I would like to thank the soon to be Dr. Joe Pacheco, Jr. who was always patient when I needed help and whose work this entire thesis is based upon. I suppose I can only corrupt a quote from Newton and say that if I have come so far, it is because I have stood on the shoulders of giants.

Furthermore, I would like to thank the past and present members of Groups 49 and 45 at Lincoln Lab for their support. I would particularly like to thank Raymond Quenneville and Dr. Douglas Koltenuk for their input and advice. Also, a special thanks to Dr. Paula Pomianowski, and Dr. James Kelly who helped me build the foundations of my research in previous years.

My parents have proven to be a constant source of comfort and strength during difficult times. They deserve special acknowledgement for their time and efforts in rearing me and making sure I made the most of myself. Also, their ability to deal with a cranky, snappish, and crochety daughter is nearly superhuman. My extended family also deserve acknowledgement for their concern and help.

I would like to thank Mrs. Voula Steinberg who is in a better place. She was a truly inspirational math teacher who first believed in me and believed I could attend MIT despite the fact that I was a nobody at a no-name school in the middle of nowhere. She tracked my academic progress until the month before she departed, and I like to think she still might be watching me now. I would like to thank my first physics teacher, Mrs. Margaret Loehr, for making things about electricity and magnetism interesting and creating that first spark, so to speak.

I would also like to thank Aya Shirai for being a friend and always reminding me to dare to break out of the mold because there are still those who accept people who different.

I would like to thank the doctors who have helped to keep my mother well enough to see me walk during commencement once again.

I would like to thank the Zeiseger Sports and Fitness Center which is going to help me lose the 10 pounds I gained while finishing this thesis.

I would like to thank sushi which produced a Zen-like quality in me every time I ate it. It kept me well and sane. A bit of heaven on a bite of vinegared rice–may I always afford to eat sea urchin roe.

In closing, I would like to thank a greater force that is beyond the domain of Maxwell's Equations and beyond the measure of science. It goes by different names in many religions and beliefs. And despite the confusion our mortal minds might have about it, I believe in it and hope it will continue to protect and bless me in the future so my life will not be in vain.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Target Radar Cross Section

Approached for the detection and identification of airborne, space-borne, or land-moving targets often employ the use of radar sensing. In these cases, prior knowledge of the targets' electromagnetic characteristics is essential in analyzing system performance and in designing signal processing and identification algorithms. Radar cross section (RCS) quantifies the behavior of the radar energy incident on and scattered from a given target. Specifically, Radar cross section, σ, is defined as,

$$\sigma(\phi, \theta) \equiv \lim_{R \to \infty} 4\pi R^2 \frac{|E_s(R, \phi, \theta)|^2}{|E_i(R, \phi, \theta)|^2}, \tag{1.1}$$

where E_i is the incident electric field, and E_s is the scattered electric field.

Because of the importance of target signature in radar sensing problems, RCS estimation for complex targets remains an area of significant research interest. A target's RCS can be obtained by using either direct measurement or computer simulation. Direct measurement requires a radar measurement facility as well as the availability of the desired target. Thus, this method can be expensive and impractical. Computer simulation, however, allows for RCS estimation using only information about the physical characteristics of the target. Because of this advantage, various numerical techniques to predict RCS have been developed. The combination of this diverse set of techniques, and continually improving computational resources has allowed RCS prediction to mature in many areas.

One area where prediction techniques remain limited, however, is the modeling of large cavities. Cavity structures can be an important contributor to the overall RCS of targets. For example, the inlet and or engine structure on aircraft can trap radar energy and scatter it strongly. The RCS of cavity structures, such as the one in this example, is often difficult to predict through current computer simulation techniques. The behavior of electromagnetic waves within a cavity can be complex, and the existing analytical and numerical techniques are either inaccurate or too computationally expensive to apply. This cavity problem is the focus of this thesis. Section 1.2.3 describes this problem at greater depth, and Section 1.4.1 presents a possible solution. Before these discussions, however, the next section describes the available prediction methods, and their limitations, in more detail.

1.2 RCS Prediction Methods

1.2.1 High Frequency Approximation Techniques

High frequency techniques involve physical optics (PO), geometrical optics (GO),the physical theory of diffraction (PTD), and the geometrical theory of diffraction (GTD). When the target and its features are large compared to the wavelength of incident radar source, a combination of these methods can be used to approximate the interaction between the target and the electromagnetic waves. Geometrical optics uses ray-tracing to model the target scattering, in particular the reflection off of the target and into the direction of the receiver [28]. GO alone treats specular scattering from targets, but not diffraction effects. Physical optics similarly calculates the reflection from the target surface but does that by approximating the surface currents. A smooth target surface is assumed, and the tangential magnetic field on the surface is approximated as twice the tangential component of the incident magnetic field in the illuminated region. From this approximation, the surface currents and the scattering can be derived [32]. Diffraction effects are calculated in PTD and GTD approaches by approximating the features of the target as combinations of wedges, straight edges, and corners and using asymptotic solutions for these geometries to predict the scattering from increment lengths of the edges [14].

1.2.2 Exact Numerical Techniques

Exact numerical techniques involve brute force numerical solutions to Maxwell's Equations. Method-of-Moments (MoM) solves Maxwell's equations in integral form. An integral equation is first developed for the unknown surface current. These surface currents are represented as a weighted series of basis functions. The integral equation is then tested with a series of testing functions to produce a matrix equation which can be solved for the unknown weights of the basis functions [32, 34]. Finite-Difference Time-Domain (FD-TD) in contrast solves Maxwell's Equations in differential form by discretizing both time and space, and solving the resulting difference equations using a marching in time technique [45]. FD-TD, both in three dimensions, and for the specific case of body-of-revolution geometries body-of-revolution, will be explored more in-depth in the following chapters.

1.2.3 Computational and Accuracy Concerns for Prediction Methods

High frequency methods are computationally efficient but often do not accurately predict cavity RCS. This inaccuracy is due to several factors. The high frequency approach produces an appoximate solution based on the idea that target elements scatter largely independently of each other. However, many portions of the target that are shadowed from the incident wave might be illuminated by specular reflection from other parts of the target. This is a problem unless ray-tracing is used. But even that is only an approximation of the possible multiple interactions between different parts of the cavity. Furthermore, surface waves are created when a component of the incident wave is tangential to a long surface on the target. These waves contribute to RCS when that surface is bounded by a discontinuity on the far end, causing a reflection.

Numerically exact methods provide high accuracy, but these techniques require too much computing power when modeling cavities of large electrical size. Method of Moments requires the surface current be sampled approximately every one-fifth of a wavelength or less. The resulting matrix problem becomes intractable for large objects since the required matrix inversion grows $\Omega(N^3)$, where N is the number of unknowns, which itself grows proportional to the square of the radar frequency of interest. Similarly, FD-TD requires that the entire computational domain be gridded with a lattice having a spatial increment Δ of approximately $\lambda/20$ to $\lambda/10$ for the highest frequency of interest. Since time is discretized, the FD-TD simulation must be run for enough time steps to allow electromagnetic energy to propagate across the target and for all interactions to finish.

Since space is also discretized, FD-TD must update every point in the grid for every time step. Therefore, FD-TD can be very computationally expensive. Traditional FD-TD approaches require large 3D arrays to store the lattice information and use considerable computer memory.

Even for particular Body of Revolution (BOR) geometries where the computer memory savings of a BOR FD-TD can be gained by using an essentially 2D FD-TD scheme–which will be briefly explained in the following chapter–memory limitations can still be an issue, and both traditional 3D and BOR FD-TD algorithms still require roughly the same amount of computational time. At present, computing power is such that FD-TD can only be applied to objects of moderate electrical size.

These accuracy and computational issues are prominent when applied to structures that contain cavities. For FD-TD, accuract becomes a concern. The interior of cavities creates multiple interactions between the the side walls. Each internal reflection causes the incident wave to become more spread out and less like a ray, making ray tracing inaccurate. Furthermore, the backwall of the cavity will reflect all surface waves that travel along the interior. The high frequency technique cannot model that behavior.

FD-TD also has problems. But these are computational rather than accuracy issues. Electromagnetic activity can be "retained" inside the cavity and still be present for a considerable amount of time after the initial excitation. Thus the FD-TD simulation must be extended for even more time steps to accurately model scattering from the interior of the cavity. For electrically small cavities, such as one of resonant size, FD-TD can provide a solution within a reasonable time frame. But for large cavities, the extended computational domain, and the additional time steps, make the FD-TD approach impractical. It is for this reason that developing better methods to predict the RCS of cavities is a current area of research.

1.3 Past Work

A number of past efforts have attempted to develop a cavity modeling technique that is computationally efficient, yet reasonably accurate. Most of these attempts have focused on creating hybrid techniques, which combine high frequency methods with exact numeric methods [5, 26, 4]. For example, a complex termination at the end of the cavity may be modeled by an exact technique but the rest of the cavity is modeled using a high frequency approach. Other methods combine integral and modal techniques [27, 35, 44]. But these hybrid techniques are often specialized for cavities with certain types of interior features and are still limited by CPU time requirements [31].

There also has been some development into using a specialized Finite Element Method (FEM) method that makes the memory requirements independent of the depth of the cavity by dividing the interior cavity into many thin layers. However, assembling the finite element equations require the use of Gaussian elimination, making the technique potentially computationally expensive for cavities with large apertures [30, 18, 4].

Some work has been done involving the idea of breaking up large cavities into segments. One proposed method works with electromagnetic fields in the spectral domain and converts the cavity into a stepped-waveguide model. The field spectra are propagated forward and backward along each waveguide section [37]. Another development borrows techniques from Microwave Network Theory: the cavity is divided into sections which are independently analyzed. Each division is represented by a generalized admittance matrix, and the aperture admittance is derived by cascading those matrices [43].

Some research has been conducted into exploiting spatial sparseness in FD-TD simulations: Johnson and Rahmat-Samii modeled the behavior of two scatterers separated by some distance by enclosing each scatterer with an FD-TD lattice such that each subregion is independent. The FD-TD problem domain is thus broken into the interior problem which uses FD-TD to solve for each sub-domain and an exterior problem which uses the Schelkunoff surface equivalence theorem to replace each scatterer by current sources [19]. The authors of that study found significant savings in computational time and memory. This division of the FD-TD computational domain into independent parts is related to the multiple region FD-TD method proposed in this paper. But the application to duct cavities does not require the formation of current sources since the subregions are not separated by space.

1.4 Background

1.4.1 Exploiting the Behavior within Duct Cavities

Current and past modeling techniques for large cavities do not, however, include breaking large cavities into segments within FD-TD and taking advantage of the behavior of electromagnetic waves within duct-like cavities. The scattering from the cavity can be thought of as consisting of two components. These components are:

Scattering from Cavity Termination Part of the energy of the pulse will move into the cavity from the opening to terminated end, and then back to the opening.

Scattering from Interior Features As the pulse propagates towards the termination of the cavity, part of the energy will be reflected by any features on the interior wall and scatter back directly towards the opening.

If the coupling between the cavity's interior features, and between these features and the cavity termination is weak, then it is possible the signature will be dominated by the direct scattering by each, and that the multiple interactions may be neglected. Under this assumption, if the cavity length is partitioned into segments, the activity that propagates into a segment is simply the activity that exited out of the neighboring segment, and the interaction between segments is local and first order in nature. Thus, one can model the entire cavity in a piecewise manner: one simulates the behavior of the electromagnetic waves in each segment and records the fields at both ends of the segment. Then this recorded data is used as an incident source for the neighboring segments.

1.4.2 Advantages of Partitioned Space

Application to FD-TD

Since FD-TD works in the time domain, it is suitable to implement the partitioned cavity technique within the FD-TD framework. FD-TD is also an exact method, which is capable of capturing the complex behavior of the electromagnetic energy within cavities. Normally this precision would make FD-TD computationally impractical for large cavities. A modified multiple region FD-TD potentially reduces these computational requirements significantly.

Savings in Memory

An important advantage of a multiple region FD-TD approach is that less memory is needed at any one time: the lattice information for only one segment needs to be kept in core memory. Though virtual memory is available in modern computers, this mechanism can cause the FD-TD program to become extremely slow. Thus, a computer with limited memory, which was previously incapable of running FD-TD for large objects without resorting to virtual memory, can run this partitioned form of FD-TD in the most efficient manner possible. This savings in memory is the same for both the smooth duct cavities and the cavities with features.

Savings in Time

Multiple region FD-TD provides savings in time through several methods. First, the elimination of the need for virtual memory prevents the slow downs associated with paging to disk. Secondly, the partitioned nature of the cavity allows for parallel computing. As soon as some data for the electromagnetic waves leaving through one cavity segment is recorded, a second computer can be used to start modeling the next cavity in parallel. Thirdly, for a large cavity with limited coupling between segments, the FD-TD simulation need only be performed for times for which energy remains in the segment. All segments of the cavity are not time stepped for the entire period energy remains in the cavity and a further savings in time is realized.

1.5 Thesis Work

This thesis describes a multiple region FD-TD algorithm, which more efficiently yet accurately models electromagnetic scattering from large duct cavities.

Chapter 2 provides an introduction to both 3D FD-TD and the Body of Revolution (BOR) variant of FD-TD, along with other pertinent supporting methods such as the Perfectly Matched Layer Boundary Condition (PML ABC).

Chapter 3 introduces the proposed modifications to realize a multiple region BOR FD-TD algorithm, which takes advantage of the behavior of the electromagnetic fields for the particular case of large, duct-like cavities.

Chapter 4 demonstrates the multiple region FD-TD approach. Results are calculated from simulations using a standard FD-TD algorithm, the multiple region FD-TD approach, and in a high frequency ray tracing technique. The results are shown to support the conclusion that multiple region FD-TD is able to produce results comparable to that of a standard FD-TD simulation while using less computational memory and computer time. Furthermore, the ability of these three different modeling methods to successfully produce accurate results depends on cavity size, cavity side-way shaping, and incident angle. These areas of validity are mapped out for each technique.

Chapter 5 will summarize this work, and provide suggestions for future development and applications of the multiple-region FD-TD approach.

Chapter 2

Finite-Difference Time-Domain Background

Understanding the multiple-region FD-TD method first requires a basic understanding of the standard FD-TD modeling technique. This section will introduce both the 3D FD-TD and the BOR FD-TD formulations along with the associated techniques to accurately predict RCS from specified targets.

2.1 3D FD-TD Algorithm

FD-TD is an exact numerical technique to solve Maxwell's Equations in differential form by discretizing them and expressing them as difference equations [45]. The FD-TD difference equations can also be derived from Maxwell's Equations in their integral form by discretizing space into cells and assuming the electric and magnetic fields are constant over each cell. However, only the derivation from the differential form will be demonstrated in this discussion.

Development of an FD-TD algorithm requires three elements: discretization of Maxwell's Equations, arranging electric and magnetic fields in a grid structure that discretizes space, and solving the discretized Maxwell's Equations using a time step solution that discretizes time.

2.1.1 Derivation of 3D FD-TD difference equations

Ampere and Faraday's law in their differential form for free space are given by,

$$\epsilon_0 \frac{\partial \vec{E}}{\partial t} = \nabla \times \vec{H} \tag{2.1}$$

$$\mu_0 \frac{\partial \vec{H}}{\partial t} = -\nabla \times \vec{E}. \tag{2.2}$$

These equations can be rewritten into six scalar equations which are,

$$\epsilon_0 \frac{\partial E_x}{\partial t} = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{2.3}$$

$$\epsilon_0 \frac{\partial E_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \tag{2.4}$$

$$\epsilon_0 \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{2.5}$$

$$\mu_0 \frac{\partial H_x}{\partial t} = \frac{\partial E_y}{\partial z} - \frac{\partial H_z}{\partial y} \tag{2.6}$$

$$\mu_0 \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z}. \tag{2.7}$$

$$\mu_0 \frac{\partial H_z}{\partial t} = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \tag{2.8}$$

These equations in turn can be discretized by using the central difference approximation which is given by Equation 2.9.

$$\frac{\partial f(\xi)}{\partial \xi} = \frac{f(\xi + \frac{\Delta\xi}{2}) - f(\xi - \frac{\Delta\xi}{2})}{\Delta\xi} \tag{2.9}$$

Thus, for example, Equation 2.3 can be written as,

$$\begin{aligned} \epsilon_0 \frac{E_x^{n+1/2}(i,j,k) - E_x^{n-1/2}(i,j,k)}{\Delta t} &= \frac{H_z^n(i,j+1/2,k) - H_z^n(i,j-1/2,k)}{\Delta} - \\ & \frac{H_y^n(i,j,k+1/2) - H_y^n(i,j,k-1/2)}{\Delta}. \end{aligned} \tag{2.10}$$

Where Δ refers to a step in space such that $\Delta \equiv \Delta x = \Delta y = \Delta z$. The superscript of

n refers to a step in time such that,

$$f(i\Delta x, j\Delta y, k\Delta z, n\Delta t) = f^n(i,j,k). \tag{2.11}$$

Note the use of 1/2 in the super and subscripts. This is a natural and desirable by-product of using the central difference approximation for first order derivatives. However, the arbitrary choice of deriving Equation 2.10 first sets up a situation where all magnetic fields will be given integer indices in time while all electric fields will have "half" indices. Furthermore, it also sets into place the integer indices and "half" indices for the fields in space. The selection of which fields will have integer indices and which will have "half" indices on the mesh is arbitrary but, as will become apparent in the following sections, one convention must be enforced for all the difference equations to be in agreement.

Equations similar to 2.10 can be generated for E_y, E_z, H_x, H_y, H_z. Furthermore, equation 2.10 can be rewritten as,

$$\begin{aligned} E_x^{n+1}(i+1/2,j,k) = {} & E_x^n(i+1/2,j,k) + \eta_0 \frac{\Delta\tau}{\Delta}[H_z^{n+1/2}(i+1/2,j+1/2,k) - \\ & -H_z^{n+1/2}(i+1/2,j-1/2,k) - H_y^{n+1/2}(i+1/2,j,k+1/2) + \\ & +H_y^{n+1/2}(i+1/2,j,k-1/2)] \end{aligned} \tag{2.12}$$

where τ is defined as

$$\Delta\tau = c\Delta t. \tag{2.13}$$

The other five equations are formed in a similar manner:

$$\begin{aligned} E_y^{n+1}(i,j+1/2,k) = {} & E_y^n(i,j+1/2,k) + \eta_0 \frac{\Delta\tau}{\Delta}[H_x^{n+1/2}(i,j+1/2,k+1/2) - \\ & -H_x^{n+1/2}(i,j-1/2,k-1/2) - H_z^{n+1/2}(i+1/2,j+1/2,k) + \\ & +H_z^{n+1/2}(i-1/2,j+1/2,k)] \end{aligned} \tag{2.14}$$

$$
\begin{aligned}
E_z^{n+1}(i,j,k+1/2) &= E_z^n(i,j,k+1/2) + \eta_0\frac{\Delta\tau}{\Delta}[H_y^{n+1/2}(i+1/2,j,k+1/2) - \\
&\quad -H_y^{n+1/2}(i-1/2,j,k+1/2) - H_x^{n+1/2}(i,j+1/2,k+1/2) + \\
&\quad +H_x^{n+1/2}(i,j-1/2,k+1/2)] \qquad (2.15)
\end{aligned}
$$

$$
\begin{aligned}
H_x^{n+1/2}(i,j+1/2,k+1/2) &= H_x^{n+1/2}(i,j+1/2,k+1/2) + \eta_0\frac{\Delta\tau}{\Delta}[E_y^n(i,j+1/2,k+1) - \\
&\quad -E_y^n(i,j+1/2,k) - E_z^n(i,j+1,k+1/2) + \\
&\quad +E_z^n(i,j,k-1/2)] \qquad (2.16)
\end{aligned}
$$

$$
\begin{aligned}
H_y^{n+1/2}(i+1/2,j,k+1/2) &= H_y^{n+1/2}(i+1/2,j,k+1/2) + \eta_0\frac{\Delta\tau}{\Delta}[E_z^n(i+1,j,k+1/2) - \\
&\quad -E_z^n(i,j,k+1/2) - E_x^n(i+1/2,j,k+1) + \\
&\quad +E_x^n(i+1/2,j,k)] \qquad (2.17)
\end{aligned}
$$

$$
\begin{aligned}
H_z^{n+1/2}(i+1/2,j+1/2,k) &= H_z^{n+1/2}(i+1/2,j+1/2,k) + \eta_0\frac{\Delta\tau}{\Delta}[E_x^n(i+1/2,j+1,k) - \\
&\quad -E_x^n(i+1/2,j,k) - E_y^n(i+1,j+1/2,k) + \\
&\quad +E_y^n(i,j+1/2,k)]. \qquad (2.18)
\end{aligned}
$$

The form of equation 2.12 suggests that each new value of E for the next time step can be generated from the previous value of E and the values of four neighboring H vectors which surround the E vector in space. Thus the temporal behavior of E and H in a region of interest can be calculated. FD-TD does precisely this operation: since E and H fields are offset from each other by 1/2 in both time and space, FD-TD can update all the values by alternating the calculation of electric and magnetic fields. This leapfrog action is commonly known as a "marching in time" approach [40].

2.2 FD-TD Lattice Structure

The region of interest in 3D FD-TD is usually discretized with an orthogonal grid, known as a Yee Lattice, which defines the locations of the six fields. One cube of the Yee lattice is show in Figure 2-1. As mentioned previously, E and H fields are offset from each other by $\Delta/2$ in space to produce an interleaved arrangement.

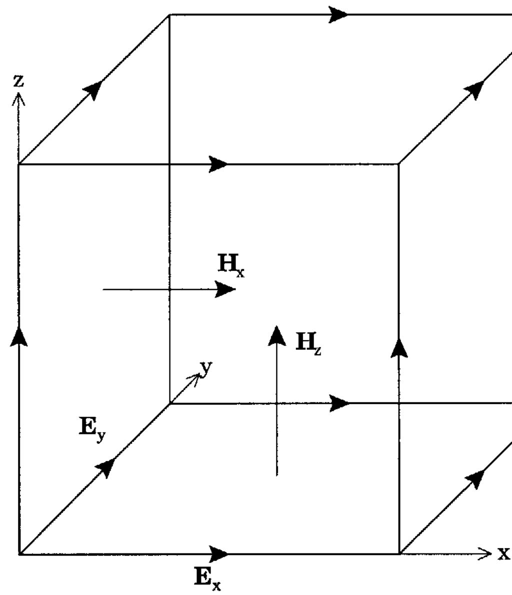

Figure 2-1: Field Quantities Represented Using Yee's Lattice.

2.3 BOR FD-TD

Body of Revolution (BOR) FD-TD allows for modeling of certain 3D targets using a 2D-like FD-TD approach. BOR FD-TD exploits rotational symmetry of the target by using a Fourier series to express the azimuthal (ϕ) dependence of the fields,

$$\vec{E} = \sum_{m=0}^{\infty} (\vec{e}_{m,u} \cos m\phi + \vec{e}_{m,v} \sin m\phi) \tag{2.19}$$

$$\vec{H} = \sum_{m=0}^{\infty} (\vec{h}_{m,u} \cos m\phi + \vec{h}_{m,v} \sin m\phi) \tag{2.20}$$

such that $\vec{e}_{m,u}, \vec{e}_{m,v}, \vec{h}_{m,u}$, and $\vec{h}_{m,v}$ are independent of ϕ. Each m is referred to as a "mode." The summation of modes cannot be carried out to indefinitely, but is often truncated by $m \geq k\rho_{max} + 1$, where k is the wavenumber of the highest frequency of the excitation, and ρ_{max} is the maximum radius of the modeled object.

The Fourier expansions can be substituted into Ampere's and Faraday's law to form the modal Maxwell's equations in cylindrical coordinates,

$$\pm\frac{m}{\rho}\hat{\phi} \times \vec{e}_{v,u} + \nabla \times \vec{e}_{u,v} = -\mu\frac{\partial}{\partial t}\vec{h}_{u,v} + \sigma^*\vec{h}_{u,v} \tag{2.21}$$

$$\pm\frac{m}{\rho}\hat{\phi} \times \vec{h}_{v,u} + \nabla \times \vec{h}_{u,v} = -\mu\frac{\partial}{\partial t}\vec{e}_{u,v} + \sigma\vec{e}_{u,v} \tag{2.22}$$

Expanding the cross products and curls, yields two sets of decoupled scalar equations,

$$\epsilon\frac{\partial}{\partial t}e_u^\rho + \sigma e_u^\rho = \frac{m}{\rho}h_v^z - \frac{\partial}{\partial z}h_u^\phi \tag{2.23}$$

$$\epsilon\frac{\partial}{\partial t}e_v^\phi + \sigma e_v^\phi = \frac{\partial}{\partial z}h_v^\rho - \frac{\partial}{\partial \rho}h_v^z \tag{2.24}$$

$$\epsilon\frac{\partial}{\partial t}e_u^z + \sigma e_u^z = -\frac{m}{\rho}h_v^\rho + \frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho h_u^\phi) \tag{2.25}$$

$$\mu\frac{\partial}{\partial t}h_v^\rho + \sigma^* h_v^\rho = \frac{m}{\rho}e_u^z + \frac{\partial}{\partial z}e_v^\phi \tag{2.26}$$

$$\mu\frac{\partial}{\partial t}h_u^\phi + \sigma^* h_u^\phi = -\frac{\partial}{\partial z}e_u^\rho + \frac{\partial}{\partial \rho}e_u^z \tag{2.27}$$

$$\mu\frac{\partial}{\partial t}h_v^z + \sigma^* h_v^z = -\frac{m}{\rho}e_u^\rho - \frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho e_v^\phi) \tag{2.28}$$

$$\epsilon\frac{\partial}{\partial t}e_v^\rho + \sigma e_v^\rho = -\frac{m}{\rho}h_u^z - \frac{\partial}{\partial z}h_v^\phi \tag{2.29}$$

$$\epsilon\frac{\partial}{\partial t}e_u^\phi + \sigma e_u^\phi = \frac{\partial}{\partial z}h_u^\rho - \frac{\partial}{\partial \rho}h_u^z \tag{2.30}$$

$$\epsilon\frac{\partial}{\partial t}e_v^z + \sigma e_v^z = \frac{m}{\rho}h_u^\rho + \frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho h_v^\phi) \tag{2.31}$$

$$\mu\frac{\partial}{\partial t}h_u^\rho + \sigma^* h_u^\rho = -\frac{m}{\rho}e_v^z + \frac{\partial}{\partial z}e_u^\phi \tag{2.32}$$

$$\mu\frac{\partial}{\partial t}h_v^\phi + \sigma^* h_v^\phi = -\frac{\partial}{\partial z}e_v^\rho + \frac{\partial}{\partial \rho}e_v^z \tag{2.33}$$

$$\mu\frac{\partial}{\partial t}h_u^z + \sigma^* h_u^z = \frac{m}{\rho}e_v^\rho - \frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho e_u^\phi) \tag{2.34}$$

These equations govern the twelve field components, but the two sets are interchangeable by replacing m by $-m$ and swapping v and u. Furthermore, only one set is being considered so the v and u subscripts will be dropped for the rest of the discussion, resulting in six field equations. In addition, the modeled object will be assumed to be in free space, so $\epsilon = \epsilon_0$, $\mu = \mu_0$, and $\sigma = \sigma^* = 0$.

2.3.1 BOR FD-TD Mesh Structure

As in 3D FD-TD, the E and H fields for BOR FD-TD are staggered in time and space, allowing for "marching in time" calculations. Figure 2-2 gives a schematic of the mesh structure for the BOR FD-TD fields where updates to each field are calculated from surrounding fields. Figure 2-3 illustrates the mesh structure of the BOR FD-TD fields as it would mathematically look in 3D.

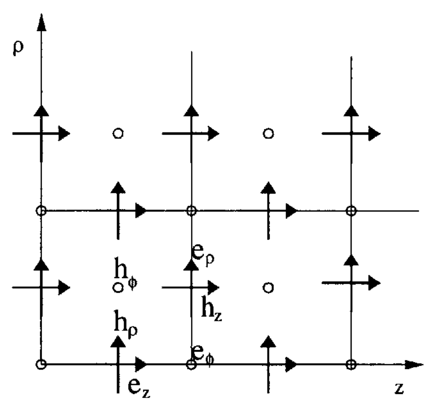

Figure 2-2: BOR 2D mesh showing interlocking cells and field vectors

To discretize the field components on this mesh in space and in time, the following notation will be used for any function of time and space:

$$f(i\Delta\rho, k\Delta z, n\Delta t) = f|_{i,k}^{n} \tag{2.35}$$

As discussed earlier, staggering the field components in time and space allows for a desirable "marching in time" algorithm. This can be shown in equation form by assigning either whole numbers or "half" numbers (1.5, 2.5, etc) to the indices for

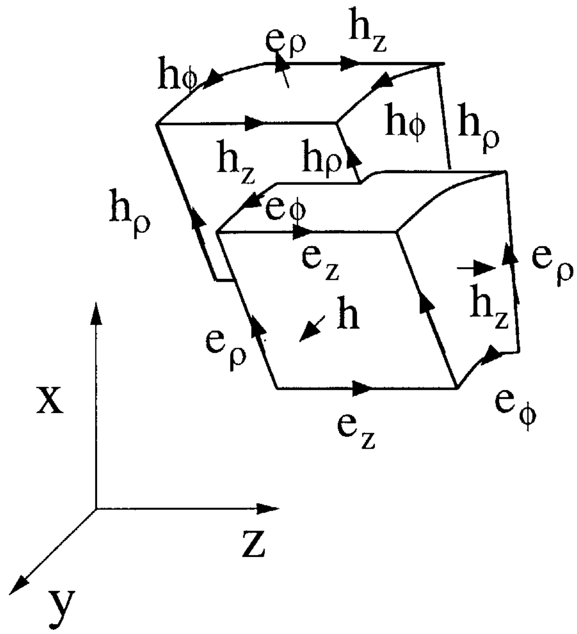

Figure 2-3: BOR 3D mesh showing interlocking cells and field vectors

space and time, and is a natural result of applying the central difference approximation for a first derivative as was done for 3D FD-TD. As shown in Figure 2-2, h_ρ and e_z lie directly on the mesh grid lines parallel to the z axis, and between mesh grid lines parallel to the ρ axis. This arrangement will be considered to have integer indices of i and k. So due to the staggering of the fields, h_z and e_ρ have "half" indices for i and z. Also, e_ϕ has a "half" index for z, and h_ϕ has a "half" index for i. Furthermore, all magnetic fields will be given integer indices in time while all electric fields will have "half" indices. As in the 3D FD-TD case, the choice of which fields will have integers for which indices is set in place by personal choice when deriving the first difference equation.

2.3.2 BOR Off-Axis Difference Equations

The central difference approximation is applied to the six field equations to yield FD-TD field update equations of a similar form to those found in traditional 3D FD-TD. For example,

$$
\begin{aligned}
e_\rho|_{i+1/2,k+1/2}^{n+1/2} &= e_\rho|_{i+1/2,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta z}\left(h_\phi|_{i+1/2,k}^{n} - h_\phi|_{i+1/2,k+1}^{n}\right) + \\
&+\eta_0 \frac{m\Delta\tau}{(i+1/2)\Delta\rho} h_z|_{i+1/2,k+1/2}^{n} \qquad (2.36)
\end{aligned}
$$

gives the update equation for the radial electric field. This equation is analogous to equation 2.12 for 3D FD-TD. The corresponding BOR equations for the remaining five 3D FD-TD equations (equations 2.14 to 2.18) are,

$$
\begin{aligned}
e_\phi|_{i,k+1/2}^{n+1/2} &= e_\phi|_{i,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta\rho}\left(h_z|_{i-1/2,k+1/2}^{n} - h_z|_{i+1/2,k+1/2}^{n}\right) + \\
&+\eta_0 \frac{\Delta\tau}{\Delta z}\left(h_\rho|_{i,k+1}^{n} - h_\rho|_{i,k}^{n}\right) \qquad (2.37)
\end{aligned}
$$

$$
\begin{aligned}
e_z|_{i,k}^{n+1/2} &= e_z|_{i,k}^{n-1/2} + \eta_0 \frac{(i+1/2)\Delta\tau}{i\Delta\rho} h_\phi|_{i+1/2,k}^{n} - \eta_0 \frac{(i-1/2)\Delta\tau}{i\Delta\rho} h_\phi|_{i-1/2,k}^{n} - \\
&-\eta_0 \frac{m\Delta\tau}{i\Delta\rho} h_\rho|_{i,k}^{n} \qquad (2.38)
\end{aligned}
$$

$$
h_\rho|_{i,k}^{n+1} = h_\rho|_{i,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta z}\left(e_\phi|_{i,k+1/2}^{n+1/2} - e_\phi|_{i,k-1/2}^{n+1/2}\right) + \frac{1}{eta_0}\frac{m\Delta\tau}{i\Delta\rho} e_z|_{i,k}^{n+1/2} \qquad (2.39)
$$

$$
\begin{aligned}
h_\phi|_{i+1/2,k}^{n+1} &= h_\phi|_{i+1/2,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta\rho}\left(e_z|_{i+1,k}^{n+1/2} - e_z|_{i,k}^{n+1/2}\right) + \\
&+\frac{1}{eta_0}\frac{\Delta\tau}{\Delta z}\left(e_\rho|_{i+1/2,k-1/2}^{n+1/2} - e_\rho|_{i+1/2,k+1/2}^{n+1/2}\right) \qquad (2.40)
\end{aligned}
$$

$$
\begin{aligned}
h_z|_{i+1/2,k+1/2}^{n+1} &= h_z|_{i+1/2,k+1/2}^{n} + \frac{1}{\eta_0}\frac{i\Delta\tau}{(i+1/2)\Delta\rho} e_\phi|_{i,k+1/2}^{n+1/2} - \frac{1}{\eta_0}\frac{(i+1)\Delta\tau}{(i+1/2)\Delta\rho} e_\phi|_{i+1,k+1/2}^{n+1/2} - \\
&-\frac{1}{\eta_0}\frac{m\Delta\tau}{(i+1/2)\Delta\rho} e_\rho|_{i+1/2,k+1/2}^{n+1/2}. \qquad (2.41)
\end{aligned}
$$

2.3.3 BOR On-Axis Difference Equations

One cannot use the previously presented difference equations to update the cells that lie directly on the axis of rotation (ie, on the z-axis) [10, 36]. As shown in Figure 2-2, e_z, e_ϕ, and h_ρ lie on the z-axis. Along the z axis, the $\hat{\rho}$ and $\hat{\phi}$ components are not defined. They may be approximated for any value of $z = z_0$ by using a value at $z = z_0$ and $\rho = \delta$ where δ is a small positive number. This approximation will also make the field component independent of ϕ.

Difference Equation for the On-Axis e_z Field

Solving for $e_z(\rho, \phi, z, t)$ on the z axis means solving for $e_z(\rho = 0, z, t)$ since it is independent of ϕ. We consider the value of $e_z(0, z, t)$ to be constant in the area bounded by a small loop of radius $\rho_0 = \Delta\rho/2$ where $\Delta\rho$ is the length of the grid cell in the ρ direction. This loop will be centered at $\rho = 0$ and perpendicular to the z axis. Ampere's Law 2.1 in integral form can be applied across this loop to produce,

$$\begin{aligned}\epsilon\frac{\partial}{\partial t}\int_0^{\rho_0}\int_0^{2\pi}&[e_{z,u}(0, z, t)\cos m\phi + e_{z,v}(0, z, t)\sin m\phi]\rho d\phi d\rho \\ &= \int_0^{2\pi}[h_{\phi,u}(\rho_0, z, t)\cos m\phi + h_{\phi,v}(\phi_0, z, t)\sin m\phi]\rho_0 d\phi. \end{aligned} \tag{2.42}$$

From the equations it can be observed that $e_z(\rho, \phi, z, t)$ is zero for non-zero values of m. For $m = 0$, the equation can be evaluated to produce,

$$\epsilon\pi\rho_0^2\frac{\partial}{\partial t}e_{z,u}(0, z, t) = 2\pi\rho_0 h_{\phi,u}(\rho_0, z, t). \tag{2.43}$$

The above equation can be discretized using the central difference approximation.

$$e_{z,u}|_{0,k}^{n+1/2} = e_{z,u}|_{0,k}^{n-1/2} + \frac{4\Delta t}{\epsilon\Delta\rho}h_{\phi,u}|_{1/2,k}^{n}, \tag{2.44}$$

The derivation for $e_{z,v}$ on the z axis produces an identical equation, so the final update equation for e_z is,

$$e_z|_{0,k}^{n+1/2} = e_z|_{0,k}^{n-1/2} + \frac{4\Delta t}{\epsilon\Delta\rho}h_\phi|_{1/2,k}^{n}, \tag{2.45}$$

Difference Equation for the On-Axis e_ϕ Field

The integral form of Ampere's Law is again used to find e_ϕ field along the z-axis. Ampere's law is calculated for a rectangular loop lying in the $\rho - z$ plane. This loop is shown in Figure 2-4.

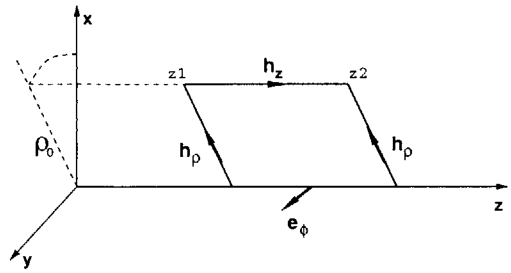

Figure 2-4: The contour used to calculate e_ϕ.

For mode $m = 1$, application of Ampere's law to the contour of Figure 2-4 produces,

$$\begin{aligned}
&\epsilon\frac{\partial}{\partial t}\int_{z_1}^{z_2}\int_0^{\rho_0}[e_{\phi,u}(0,z',t)\cos\phi + e_{\phi,v}(0,z',t)\sin\phi]d\phi dz \\
&\quad = \int_{z_1}^{z_2}[h_{z,u}(0,z',t)\cos\phi + h_{z,v}(0,z',t)\sin\phi]dz \\
&\quad + \int_0^{\rho_0}[h_{\rho,u}(0,z_2,t)\cos\phi + h_{\rho,v}(0,z_2,t)\sin\phi]d\rho \\
&\quad + \int_{z_1}^{z_2}[h_{z,u}(\rho_0,z',t)\cos\phi + h_{z,v}(\rho_0,z',t)\sin\phi]dz \\
&\quad + \int_{\rho_0}^{0}[h_{\rho,u}(0,z_1,t)\cos\phi + h_{\rho,v}(0,z_1,t)\sin\phi]d\rho
\end{aligned} \tag{2.46}$$

where $\rho_0 = \Delta\rho/2$, and $z' = z_1 + \Delta z/2$, which is really the z_0 of interest. When $\rho = 0$, h_z will also equal 0. The previous equation can be integrated and sine and cosine terms can be grouped to produce two equations,

$$\begin{aligned}
&\left[\epsilon\Delta z\frac{\Delta\rho}{2}\frac{\partial}{\partial t}e_{\phi,u}(0,z',t)\right]\cos\phi \\
&\quad = \left\{-\Delta h_{z,u}(\rho_0,z',t) + \frac{\Delta\rho}{2}[h_{\rho,u}(0,z_2,t) - h_{\rho,u}(0,z_1,t)]\right\}\cos\phi
\end{aligned} \tag{2.47}$$

$$\begin{aligned}
&\left[\epsilon\Delta z\frac{\Delta\rho}{2}\frac{\partial}{\partial t}e_{\phi,v}(0,z',t)\right]\sin\phi \\
&\quad = \left\{-\Delta h_{z,v}(\rho_0,z',t) + \frac{\Delta\rho}{2}[h_{\rho,u}(0,z_2,t) - h_{\rho,u}(0,z_1,t)]\right\}\sin\phi.
\end{aligned} \tag{2.48}$$

Solving for $e_{\phi,u}$ and $e_{\phi,v}$ from the above will produce two identical equations save for the u and v subscripts. Therefore, the on-axis e_ϕ at z_0 can be determined by,

$$\frac{\partial}{\partial t}e_\phi(0,z',t) = -\frac{2}{\epsilon\Delta\rho}h_z(\rho_0,z',t) + \frac{1}{\epsilon\Delta z}[h_\rho(0,z_2,t) + h_\rho(0,z_1,t). \tag{2.49}$$

The central difference approximation for first order derivatives can again be applied to produce the desired difference equation for the on-axis e_ϕ,

$$e_\phi|_{0,k+1/2}^{n+1/2} = e_\phi|_{0,k+1/2}^{n-1/2} - \frac{2\Delta t}{\epsilon\Delta\rho}h_z|_{1/2,k+1/2}^{n} + \frac{\Delta t}{\epsilon\Delta z}\left(h_\rho|_{0,k+1}^{n} - h_\rho|_{0,k}^{n}\right). \tag{2.50}$$

Difference Equation for the On-Axis h_ρ Field

h_ρ is non-zero only when $m = 1$. Discrete forms of equations 2.26 and 2.32 can be used to find the on-axis value of h_ρ by using the the value of e_z from the cell above as an approximation. This produces a set of difference equations,

$$h_{\rho,v}|_{0,k}^{n+1} = h_{\rho,v}|_{0}^{n} + \frac{\Delta t}{\mu\Delta\rho}e_{z,u}|_{1,k}^{n+1/2} + \frac{\Delta t}{\mu\Delta z}\left(e_{\phi,v}|_{0,k+1/2}^{n+1/2} - e_{\phi,v}|_{0,k-1/2}^{n+1/2}\right) \tag{2.51}$$

$$h_{\rho,u}|_{0,k}^{n+1} = h_{\rho,u}|_{0}^{n} - \frac{\Delta t}{\mu\Delta\rho}e_{z,v}|_{1,k}^{n+1/2} + \frac{\Delta t}{\mu\Delta z}\left(e_{\phi,u}|_{0,k+1/2}^{n+1/2} - e_{\phi,u}|_{0,k-1/2}^{n+1/2}\right). \tag{2.52}$$

2.4 Computational Domain

Another aspect of FD-TD programs is the division of the computational domain into total field and scattered field regions. The method of creating this division will be given in Section 2.6. Figure 2-5 summarizes the different regions within the lattice of a BOR FD-TD approach. The figure also serves as a two dimensional visualization of the 3D FD-TD for a "cut" along one axis of the 3D Yee lattice. This division lessens the burden on the absorbing boundary conditions at the ends of the computational domain. The absorbing boundary condition will be introduced in the next section.

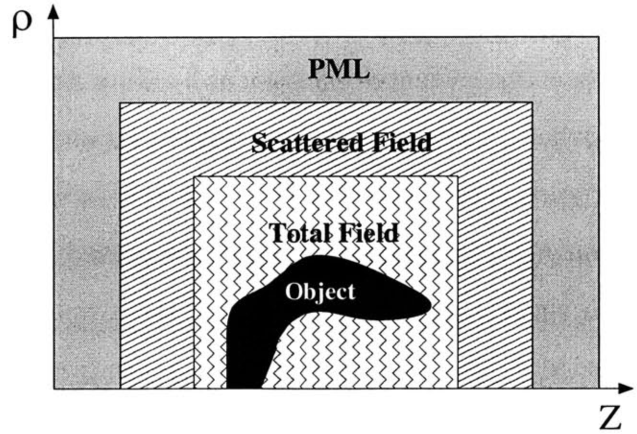

Figure 2-5: The different regions of a BOR FD-TD calculation domain. Note that the object will be rotationally symmetric along the z-axis.

2.5 Modeling Objects in the Computational Domain

2.5.1 Material Modeling

The perfect electric conductor (PEC) can be modeled with FD-TD. Since the boundary conditions for PEC require zero tangential electric fields, grid cells that correspond to PEC surfaces will have their electric fields set to zero during each update. For materials that are not PEC, the update equations must be altered to reflect the composition of the material. Most FD-TD programs do material modeling by tagging each cell in the Yee lattice and using alternative update equations–which take into account the behavior of the material–that correspond to the tag during the update. For example, modeling a PEC means resetting the tangential electric fields to zero during each update. The values of ϵ and μ can be altered to reflect any non-PEC materials on the target.

2.5.2 Geometry Modeling

In 3D FD-TD, an orthogonal Yee lattice is a natural fit for objects that have straight edges and sides. For objects that have curved surfaces which do not fit neatly within an orthogonal grid, the most simple FD-TD algorithms will approximate these by using a "staircase" to try to match the surface. More recently, conformal grids have been developed where the shape of the cells is adjusted to provide a better approximation to curved surfaces.

In BOR FD-TD, the geometry of the targets are assumed to be independent of ϕ. The object's surface in the 2D $z - \rho$ plane can be fitted by the staircase method or conformal grids as needed. The partitioned FD-TD method described by this paper uses the staircase method, although the partitioned approach is equally applicable with a conformal grid.

2.5.3 Berenger's Perfectly Matched Layer for Absorbing Boundary Conditions

The computational domain must be finite in extent. However, by considering the fields beyond the computational domain to be zero, one would have essentially created a PEC box surrounding the whole domain. To prevent unwanted reflections at the boundary, FD-TD must be run with an Absorbing Boundary Condition (ABC) to absorb incident waves and simulate free space beyond the computational domain. Engquist and Majda [15] proposed one type of ABC using a second order boundary condition,

$$\left[\frac{\partial^2}{\partial n \partial \tau} + \frac{\partial^2}{\partial \tau^2} - \frac{1}{2}\left(\frac{\partial^2}{\partial T_1^2} + \frac{\partial^2}{\partial T_2^2}\right)\right] w = 0 \tag{2.53}$$

where w is a field quantity which is tangential to the absorbing boundary, $\hat{n}$ is the normal direction of that boundary, $\hat{T}_1$ and $\hat{T}_2$ are the tangential directions, and τ is ct. This second-order absorbing boundary condition works well for waves which are incident at or close to normal to the boundary. But it works poorly for waves which are incident at grazing angles. Furthermore, it is impossible to implement the second order boundary condition at corners where the normal and tangential directions are not well defined. The corners would require a first order boundary condition.

Berenger's Perfectly Matched Layer (PML) is type of ABC that matches the impedance of free space and attenuates waves incident at any angle [6]. For this method, the outer boundary of the free space region is extended with several more lattice cells, as shown in Figure 2-5, which absorb the incident wave as it propagates into the region. But the PML region is matched to waves impinging at all angles to create a reflection-less boundary. This matching is accomplished through splitting the fields in the PML into two components to create an artificial non-Maxwellian space. This split will add the additional degrees of freedom necessary to absorb waves at any arbitrary angle of incidence.

PML for 3D FD-TD

In media with electric conductivity and magnetic loss, the Maxwell curl equations can be written as,

$$\epsilon_0 \frac{\partial E_x}{\partial t} + \sigma E_x = \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \tag{2.54}$$

$$\epsilon_0 \frac{\partial E_y}{\partial t} + \sigma E_y = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \tag{2.55}$$

$$\epsilon_0 \frac{\partial E_z}{\partial t} + \sigma E_z = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} \tag{2.56}$$

$$\mu_0 \frac{\partial H_x}{\partial t} + \sigma^* H_x = \frac{\partial E_y}{\partial z} - \frac{\partial H_z}{\partial y} \tag{2.57}$$

$$\mu_0 \frac{\partial H_y}{\partial t} + \sigma^* H_y = \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \tag{2.58}$$

$$\mu_0 \frac{\partial H_z}{\partial t} \sigma^* H_z = \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x} \tag{2.59}$$

where σ is the electric conductivity and σ^* is the magnetic conductivity. When,

$$\frac{\sigma}{\sigma^*} = \frac{\epsilon_0}{\mu_0} \tag{2.60}$$

the impedance of the medium is equal to that of free space. A wave that is normally incident on the boundary between this medium and free space will create no reflection. However a reflection will occur for non-normally incident waves, and thus this sort of medium provides little improvement over the second order ABC.

Berenger's improvement lay in splitting each field component into two quantities, each derived from only one spatial derivative term. For example, E_x fields calculated from differences of H_z in the $\hat{y}$ direction are denoted as E_{xy}, and E_x fields calculated from differences of H_y in the $\hat{z}$ direction are denoted as E_{xz}. E_{xy} and E_{xz} are updated independently of each other. The full set of 12 PML equations for 3D FD-TD are,

$$\epsilon_0 \frac{\partial E_{xy}}{\partial t} + \sigma_y E_{xy} = \frac{\partial (H_{zx} + H_{zy})}{\partial y} \quad (2.61)$$

$$\epsilon_0 \frac{\partial E_{xz}}{\partial t} + \sigma_z E_{xz} = -\frac{\partial (H_{yx} + H_{yz})}{\partial z} \quad (2.62)$$

$$\epsilon_0 \frac{\partial E_{yz}}{\partial t} + \sigma_z E_{yz} = \frac{\partial (H_{xy} + H_{xz})}{\partial z} \quad (2.63)$$

$$\epsilon_0 \frac{\partial E_{yx}}{\partial t} + \sigma_x E_{yx} = -\frac{\partial (H_{zx} + H_{zy})}{\partial x} \quad (2.64)$$

$$\epsilon_0 \frac{\partial E_{zx}}{\partial t} + \sigma_x E_{zx} = \frac{\partial (H_{yx} + H_{yz})}{\partial x} \quad (2.65)$$

$$\epsilon_0 \frac{\partial E_{zy}}{\partial t} + \sigma_y E_{zy} = -\frac{\partial (H_{xz} + H_{xy})}{\partial y} \quad (2.66)$$

$$\mu_0 \frac{\partial H_{xy}}{\partial t} + \sigma_y^* H_{xy} = -\frac{\partial (E_{zx} + E_{zy})}{\partial y} \quad (2.67)$$

$$\mu_0 \frac{\partial H_{xz}}{\partial t} + \sigma_z^* H_{xz} = \frac{\partial (E_{yx} + E_{yz})}{\partial z} \quad (2.68)$$

$$\mu_0 \frac{\partial H_{yz}}{\partial t} + \sigma_z^* H_{yz} = -\frac{\partial (E_{xy} + E_{xz})}{\partial z} \quad (2.69)$$

$$\mu_0 \frac{\partial H_{yx}}{\partial t} + \sigma_x^* H_{yx} = \frac{\partial (E_{zx} + E_{zy})}{\partial x} \quad (2.70)$$

$$\mu_0 \frac{\partial H_{zx}}{\partial t} + \sigma_x^* H_{zx} = -\frac{\partial (E_{yx} + E_{yz})}{\partial x} \quad (2.71)$$

$$\mu_0 \frac{\partial H_{zy}}{\partial t} + \sigma_y^* H_{zy} = \frac{\partial (E_{xz} + E_{xy})}{\partial y} \quad (2.72)$$

where, for example, σ_x denotes the electrical conductivity associated with $\hat{x}$ directed gradients in the magnetic field, and σ_x^* denotes the magnetic conductivity associated with the $\hat{x}$ directed gradients of the electric field. These equations will reduce to Maxwell's free space equations if $\sigma_x = \sigma_y = \sigma_z = \sigma_x^* = \sigma_y^* = \sigma_z^* = 0$. Furthermore, if $\sigma_x = \sigma_y = \sigma_z$ and $\sigma_x^* = \sigma_y^* = \sigma_z^*$, these equations will reduce to the equations for ordinary lossy media.

However, if $\sigma_x = \sigma_y = 0$ and $\sigma_x^* = \sigma_y^* = 0$, then field quantities arising from $\hat{z}$

directed gradients are attenuated. Also, when

$$\frac{\sigma_z}{\sigma_z^*} = \frac{\epsilon_0}{\mu_0}, \tag{2.73}$$

the impedance of the medium is matched to free space independent of the direction of propagation of the incident wave. Therefore, the artificial medium allows all waves to be absorbed without reflection. However, since the PML is is truncated, it is essentially backed by PEC. This PEC creates a wave that will reflect and propagate back into the computational domain. PML will attenuate this wave. The amount of attenuation is determined by the thickness of the PML and by its conductivities.

The loss factor of this medium is lower near the interface with free space to avoid possible minor spurious reflection from numerical errors and the effects of discretization. But as the wave propagates further into the PML, the loss can be increased. Different loss functions have been proposed, but good performance has been obtained from Berenger's proposed conductivity profile,

$$\sigma(\zeta) = \sigma_{\text{max}} \left[\frac{\zeta}{\delta}\right]^n \tag{2.74}$$

where δ is the total thickness of the PML and n is the order of the PML. Generally a second order PML has been found to work well.

PML for BOR FD-TD

PML equations can be applied to BOR FD-TD by using equations formulated through a stretched coordinates approach. This idea was formulated by Chew and Weedon ([11, 12]). First Maxwell's Equations are modified via a complex coordinate transform. This modification introduces additional degrees of freedom to allow for the lossy medium serving as PML to be reflection-less for all frequencies, polarizations, and angles of incidence. In the time harmonic $e^{-i\omega t}$ notation, Maxwell's Equations are,

$$\begin{aligned}
\nabla_\sigma \times \vec{E} &= i\omega\mu\vec{H} && (2.75)\\
\nabla_\sigma \times \vec{E} &= i\omega\mu\vec{H} && (2.76)\\
\nabla_\sigma \cdot \epsilon\vec{E} &= 0 && (2.77)\\
\nabla_\sigma \cdot \mu\vec{E} &= 0 && (2.78)
\end{aligned}$$

where

$$\nabla_\sigma = \hat{x}\frac{1}{s_x}\frac{\partial}{\partial x} + \hat{y}\frac{1}{s_y}\frac{\partial}{\partial y} + \hat{z}\frac{1}{s_z}\frac{\partial}{\partial z}. \tag{2.79}$$

In the previous equation, s_x, s_y, and s_z are the complex coordinate stretching variables. Using a change of variables,

$$\zeta \longrightarrow \tilde{\zeta} = \int_0^{\zeta} s_\zeta(\zeta')d\zeta' \tag{2.80}$$

where ζ represents x, y, or z, Maxwell's Equations for the PML can be given for a complex variable spatial domain. Using the same change of variables, ∇_σ becomes,

$$\nabla_\sigma \longrightarrow \tilde{\nabla} = \hat{x}\frac{\partial}{\partial \tilde{x}} + \hat{y}\frac{\partial}{\partial \tilde{y}} + \hat{z}\frac{\partial}{\partial \tilde{z}}. \tag{2.81}$$

and using the following equalities:

$$\begin{aligned}
\frac{\partial}{\partial \tilde{x}} &= \frac{1}{s_x}\frac{\partial}{\partial x} && (2.82)\\
\frac{\partial}{\partial \tilde{y}} &= \frac{1}{s_y}\frac{\partial}{\partial y} && (2.83)\\
\frac{\partial}{\partial \tilde{z}} &= \frac{1}{s_z}\frac{\partial}{\partial z}. && (2.84)
\end{aligned}$$

Maxwell's Equations can now be written as,

$$\begin{aligned}
\tilde{\nabla}_\sigma \times \vec{E} &= i\omega\mu\vec{H} && (2.85)\\
\tilde{\nabla}_\sigma \times \vec{E} &= i\omega\mu\vec{H} && (2.86)\\
\tilde{\nabla}_\sigma \cdot \epsilon\vec{E} &= 0 && (2.87)
\end{aligned}$$

$$\tilde{\nabla}_\sigma \cdot \mu \vec{E} = 0 \tag{2.88}$$

.

If $s_x = s_y = s_z = 1$, the transformed Maxwell's equations regress back into their original form. However, if

$$s_\zeta(\zeta') = 1 + \frac{i\sigma_\zeta(\zeta')}{\omega\epsilon} \tag{2.89}$$

the medium becomes lossy and non-Maxwellian. If s_ζ satisfy conditions similar to those that constrain σ_i of the PML equations for 3D FD-TD, then the interface between the PML and free space is reflection-less for all angles of incidence.

Obtaining the correct PML equations for BOR FD-TD is possible by generalizing this change of variable formulation for a cylindrical coordinate system. It will be necessary for the PML to absorb waves traveling in the ρ and z directions. Therefore, the following change of coordinates are used:

$$\tilde{z} = \int_0^z s_z(z')dz' = \int_0^z 1 + \frac{i\sigma_z(z')}{\omega\epsilon} dz' = z + \frac{i\Delta_z(z)}{\omega\epsilon} \tag{2.90}$$

$$\tilde{\rho} = \int_0^\rho s_\rho(\rho')d\rho' = \int_0^\rho 1 + \frac{i\sigma_\rho(\rho')}{\omega\epsilon} d\rho' = \rho + \frac{i\Delta_\rho(\rho)}{\omega\epsilon}. \tag{2.91}$$

For cylindrical coordinates the del operator becomes,

$$\tilde{\nabla} = \hat{\rho}\frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}} + \hat{\phi}\frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\phi}} + \hat{z}\frac{\partial}{\partial\tilde{z}}. \tag{2.92}$$

Expressions of the magnetic and electric fields as Fourier series (Equations 2.19 and 2.20) can be substituted into the new Maxwell's Equations (Equations 2.85 to 2.88) while applying the ∇ operator in cylindrical coordinates. This procedure will result in the equations for the fields inside of BOR PML in modal form,

$$\pm\frac{m}{\tilde{\rho}}\hat{\phi} \times \vec{e}_{v,u} + \tilde{\nabla} \times \vec{e}_{u,v} = i\omega\mu\vec{h}_{u,v} \tag{2.93}$$

$$\pm\frac{m}{\tilde{\rho}}\hat{\phi} \times \vec{h}_{v,u} + \tilde{\nabla} \times \vec{h}_{u,v} = -i\omega\epsilon\vec{e}_{u,v} \tag{2.94}$$

Expansion of the curls and cross products will produce two sets of equations,

$$-i\omega\epsilon e_u^\rho = \frac{m}{\tilde{\rho}}h_v^z - \frac{\partial}{\partial\tilde{z}}h_u^\phi \tag{2.95}$$

$$-i\omega\epsilon e_v^\phi = \frac{\partial}{\partial\tilde{z}}h_v^\rho - \frac{\partial}{\partial\tilde{\rho}}h_v^z \tag{2.96}$$

$$-i\omega\epsilon e_u^z = -\frac{m}{\tilde{\rho}}h_v^\rho + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}h_u^\phi) \tag{2.97}$$

$$-i\omega\mu h_v^\rho = \frac{m}{\tilde{\rho}}e_u^z - \frac{\partial}{\partial\tilde{z}}e_v^\phi \tag{2.98}$$

$$-i\omega\mu h_u^\phi = -\frac{\partial}{\partial\tilde{z}}e_u^\rho + \frac{\partial}{\partial\tilde{\rho}}e_u^z \tag{2.99}$$

$$-i\omega\mu h_v^z = -\frac{m}{\tilde{\rho}}e_u^\rho - \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}e_v^\phi) \tag{2.100}$$

$$-i\omega\epsilon e_v^\rho = -\frac{m}{\tilde{\rho}}h_u^z - \frac{\partial}{\partial\tilde{z}}h_v^\phi \tag{2.101}$$

$$-i\omega\epsilon e_u^\phi = \frac{\partial}{\partial\tilde{z}}h_u^\rho - \frac{\partial}{\partial\tilde{\rho}}h_u^z \tag{2.102}$$

$$-i\omega\epsilon e_v^z = \frac{m}{\tilde{\rho}}h_u^\rho + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}h_v^\phi) \tag{2.103}$$

$$-i\omega\mu h_u^\rho = -\frac{m}{\tilde{\tilde{\rho}}}e_v^z + \frac{\partial}{\partial\tilde{z}}e_u^\phi \tag{2.104}$$

$$-i\omega\mu h_v^\phi = -\frac{\partial}{\partial\tilde{z}}e_v^\rho + \frac{\partial}{\partial\tilde{\rho}}e_v^z \tag{2.105}$$

$$-i\omega\mu h_u^z = \frac{m}{\tilde{\rho}}e_v^\rho - \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}e_u^\phi). \tag{2.106}$$

As described earlier when the equations for the BOR FD-TD fields were derived, these two sets are independent and redundant. Thus they can be condensed into one set and have their v and u subscripts dropped:

$$-i\omega\epsilon e_\rho = \frac{m}{\tilde{\rho}}h_z - \frac{\partial}{\partial\tilde{z}}h_\phi \tag{2.107}$$

$$-i\omega\epsilon e_\phi = \frac{\partial}{\partial\tilde{z}}h_\rho - \frac{\partial}{\partial\tilde{\rho}}h_z \tag{2.108}$$

$$-i\omega\epsilon e_z = -\frac{m}{\tilde{\rho}}h_\rho + \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}h_\phi) \tag{2.109}$$

$$-i\omega\mu h_\rho = \frac{m}{\tilde{\rho}}e_z - \frac{\partial}{\partial\tilde{z}}e_\phi \tag{2.110}$$

$$-i\omega\mu h_{\phi} = -\frac{\partial}{\partial\tilde{z}}e_{\rho} + \frac{\partial}{\partial\tilde{\rho}}e_{z} \tag{2.111}$$

$$-i\omega\mu h_{z} = -\frac{m}{\tilde{\rho}}e_{\rho} - \frac{1}{\tilde{\rho}}\frac{\partial}{\partial\tilde{\rho}}(\tilde{\rho}e_{\phi}). \tag{2.112}$$

The above equations need to be discretized and put into a form that allows for time-stepping. This conversion is accomplished by splitting each field into two components, very much analogous to the splitting that was performed for the PML of 3D FD-TD. For example $e_{\rho} = e_{\rho z} + e_{\rho\phi}$ where $e_{\rho z}$ and $e_{\rho\phi}$ are defined by the equations,

$$-i\omega\epsilon s_{\phi}e_{\rho\phi} = \frac{m}{\rho}h_{z} - i\omega\epsilon s_{z}e_{\rho z} = \frac{\partial}{\partial z}h_{\phi} \tag{2.113}$$

For $e_p hi$, $e_{\phi} = e_{\phi z} + e_{\phi\rho}$ where $e_{\phi z}$ and $e_{\phi\rho}$ are defined by the equations,

$$-i\omega\epsilon s_{z}e_{\phi z} = \frac{\partial}{\partial z}h_{\rho} - i\omega\epsilon s_{\rho}e_{\phi\rho} = -\frac{\partial}{\partial\rho}h_{z}. \tag{2.114}$$

For e_z, the first derivative of Equation 2.109 must be expanded in order to properly split the field. Taking the derivative with respect to ρ,

$$-i\omega\epsilon e_{z} = \frac{\partial}{\partial r\tilde{h}o} + \frac{m}{\tilde{\rho}}h_{\rho} + \frac{1}{\tilde{\rho}}h_{\phi} \tag{2.115}$$

allows for e_z to be split into,

$$-i\omega\epsilon s_{\phi}e_{z\phi} = \frac{m}{\rho}h_{\rho} + \frac{1}{\rho}h_{\phi} \tag{2.116}$$

$$-i\omega\epsilon s_{\rho}e_{z\rho} = \frac{\partial}{\partial\rho}h_{\phi}. \tag{2.117}$$

The split h field terms are derived in a similar manner and are described by,

$$-i\omega\mu s_{\phi}h_{\rho\phi} = \frac{m}{\rho}e_{z} \tag{2.118}$$

$$i\omega\mu s_{z}h_{\rho z} = \frac{\partial}{\partial z_{\phi}} \tag{2.119}$$

$$i\omega\mu s_{z}h_{\phi z} = \frac{\partial}{\partial z}e_{\rho} \tag{2.120}$$

$$i\omega\mu s_\rho h_{\phi\rho} = -\frac{\partial}{\partial\rho}e_z \tag{2.121}$$

$$i\omega\mu s_\phi h_{z\phi} = -\frac{m}{\rho}e_\rho + \frac{1}{\rho}e_\phi \tag{2.122}$$

$$i\omega\mu s_\rho h_{z\rho} = \frac{\partial}{\partial\rho}e_\phi. \tag{2.123}$$

The set of PML equations is changed back from time harmonic form to the time domain to yield,

$$\epsilon\frac{\partial}{\partial t}e_{\rho z} + \sigma_z e_{\rho z} = -\frac{\partial}{\partial z}(h_{\phi z} + h_{\phi\rho}) \tag{2.124}$$

$$\epsilon\frac{\partial}{\partial t}e_{\rho\phi} + \sigma_\phi e_{\rho\phi} = \frac{m}{\rho}(h_{z\rho} + h_{z\phi}) \tag{2.125}$$

$$\epsilon\frac{\partial}{\partial t}e_{\phi z} + \sigma_z e_{\phi z} = -\frac{\partial}{\partial z}(h_{\rho z} + h_{\rho\phi}) \tag{2.126}$$

$$\epsilon\frac{\partial}{\partial t}e_{\phi\rho} + \sigma_\rho e_{\phi\rho} = -\frac{\partial}{\partial\rho}(h_{z\rho} + h_{z\phi}) \tag{2.127}$$

$$\epsilon\frac{\partial}{\partial t}e_{z\rho} + \sigma_\rho e_{z\rho} = -\frac{\partial}{\partial\rho}(h_{\phi z} + h_{\phi\rho}) \tag{2.128}$$

$$\epsilon\frac{\partial}{\partial t}e_{z\phi} + \sigma_\phi e_{z\phi} = -\frac{m}{\rho}(h_{\rho z} + h_{\rho\phi}) + \frac{1}{\rho}(h_{\phi z} + h_{\phi\rho}) \tag{2.129}$$

$$\mu\frac{\partial}{\partial t}h_{\rho z} + \sigma_z^* h_{\rho z} = \frac{\partial}{\partial z}(e_{\phi z} + e_{\phi\rho}) \tag{2.130}$$

$$\mu\frac{\partial}{\partial t}h_{\rho\phi} + \sigma_\phi^* h_{\rho\phi} = \frac{m}{\rho}(e_{z\rho} + e_{z\phi}) \tag{2.131}$$

$$\mu\frac{\partial}{\partial t}h_{\phi z} + \sigma_z^* h_{\phi z} = -\frac{\partial}{\partial z}(e_{\rho z} + e_{\rho\phi}) \tag{2.132}$$

$$\mu\frac{\partial}{\partial t}h_{\phi\rho} + \sigma_\rho^* h_{\phi\rho} = \frac{\partial}{\partial\rho}(e_{z\rho} + e_{z\phi}) \tag{2.133}$$

$$\mu\frac{\partial}{\partial t}h_{z\rho} + \sigma_\rho h_{z\rho} = -\frac{\partial}{\partial\rho}(e_{\phi z} + e_{\phi\rho}) \tag{2.134}$$

$$\mu\frac{\partial}{\partial t}h_{z\phi} + \sigma_\phi h_{z\phi} = -\frac{m}{\rho}(e_{\rho z} + e_{\rho\phi}) - \frac{1}{\rho}(e_{\phi z} + e_{\phi\rho}). \tag{2.135}$$

To discretize the PML equations, the central difference approximation can not be used to accurately represent rapidly decaying fields [40]. Instead, exponential time-stepping is used. The PML equations are treated as ordinary differential equations and are solved explicitly by finding a homogeneous and particular solution for each unknown. Using $e_{\rho z}$, as an example, the homogeneous solution is of the form,

$$e_{\rho z}^{hom.}(t) = Ce^{(\sigma_z/\epsilon)t} \tag{2.136}$$

with an unknown constant C. One can argue that the homogeneous solution arises from combined excitations over many previous time steps. At the previous time step, $t = (n - 1/2)\Delta t$, $e_\rho z$ is assumed to be known. There C can be expressed as,

$$\begin{aligned} e_{\rho z}^{hom.}(t = (n - 1/2)\Delta t) &= Ce^{-(\sigma_z/\epsilon)(n-1/2)\Delta t} = e_{\rho z}|^{n-1/2} \\ C &= e^{(\sigma_z/\epsilon)(n-1/2)\Delta t}e_{\rho,z}|^{n-1/2}. \end{aligned} \tag{2.137}$$

So at the next time step,

$$e_{\rho z}^{hom.}(t = (n + 1/2)\Delta t) = e^{(\sigma_z/\epsilon)(n-1/2)\Delta t}e_{\rho z}|^{n-1/2}e^{-(\sigma_z/\epsilon)(n+1/2)\Delta t} \tag{2.138}$$

$$= e_{\rho z}|^{n-1/2}e^{-(\sigma_z/\epsilon)\Delta t}. \tag{2.139}$$

The particular solution is of the form,

$$e_{\rho z}^{part.}(t') = -\frac{1}{\sigma_z}\frac{\partial(h_{\phi z} + h_{\phi\rho})}{\partial z} + Ke^{-sigma_z/\epsilon}t'. \tag{2.140}$$

It has already been established that the homogeneous solution accounts for contributions due to previous time steps. So the particular solution must arise form the h_ϕ field at the current time step. But all initial e fields are zero so K can be found using the following expression:

$$\begin{aligned} e_{\rho z}^{part.}(t' = 0) = 0 &= -\frac{1}{\sigma_z}\frac{\partial(h_{\phi z} + h_{\phi\rho})}{\partial z} + K \\ K &= \frac{1}{\sigma_z}\frac{\partial(h_{\phi z} + h_{\phi\rho})}{\partial z}. \end{aligned} \tag{2.141}$$

At the end of the time step, $t' = \Delta t$, the particular solution becomes,

$$e_{\rho z}^{part.}(t' = \Delta t) = \frac{e^{-(\sigma_z/\epsilon)\Delta t} - 1}{\sigma_z}\frac{\partial(h_{\phi z} + h_{\phi\rho})}{\partial z}. \tag{2.142}$$

Combining the particular and homogeneous solutions and discretizing the spatial derivative will give the desired discrete form,

$$\begin{aligned} e_{\rho z}|^{n+1/2}_{i+1/2,k+1/2} &= e^{-\sigma_z \Delta t/\epsilon} e_{\rho z}|^{n-1/2}_{i+1/2,k+1/2} + \frac{e^{-(\sigma_z/\epsilon)\Delta t} - 1}{\sigma_z \Delta z} \\ &\quad (h_{\phi,z}|^{n}_{i+1/2,k+1} + h_{\phi,\rho}|^{n}_{i+1/2,k+1} - \\ &\quad -h_{\phi,z}|^{n}_{i+1/2,k} - h_{\phi,\rho}|^{n}_{i+1/2,k}). \end{aligned} \tag{2.143}$$

The rest of the PML equations can be derived in a similar fashion.

2.6 Source Implementation

All initial fields within the FD-TD computational domain are zero. Excitation is created by adding quantities to these fields. Current sources can be introduced by adding a current density term, J, to the discretized Maxwell's Equations. A voltage source can be modeled by setting the electric field to V/Δ.

Usually for RCS calculations, a plane wave source is desired. The creation of this plane wave is what characterizes the difference between total field and scattered field in the calculation domain. Scattered field is defined as,

$$E_{\text{scat}} = E_{\text{total}} - E_{\text{inc}} \tag{2.144}$$

where E_{total} is the total field and E_{inc} is the incident field. This definition is enforced at the boundary between total and scattered field by adding in or subtracting out a correction term for the update equations on and next to this boundary.

This method is logical when one considers how the fields are calculated from adjacent field values: next to the boundary there are field values which lie in the total field region but are calculated from fields that lie in the scattered field region. Thus a correction term is added to the scattered field values when used to calculate the new value of the total field vectors. Similarly, next to the scattered/total field boundary there are scattered field values that are computed from vectors that lie within the total field. Thus a correction term is subtracted from the total field values when used to update scattered field vectors. Figures 2-6 and 2-7 depict the locations of the fields where the correction terms must be used to create a scattered-total field boundary in BOR FD-TD.

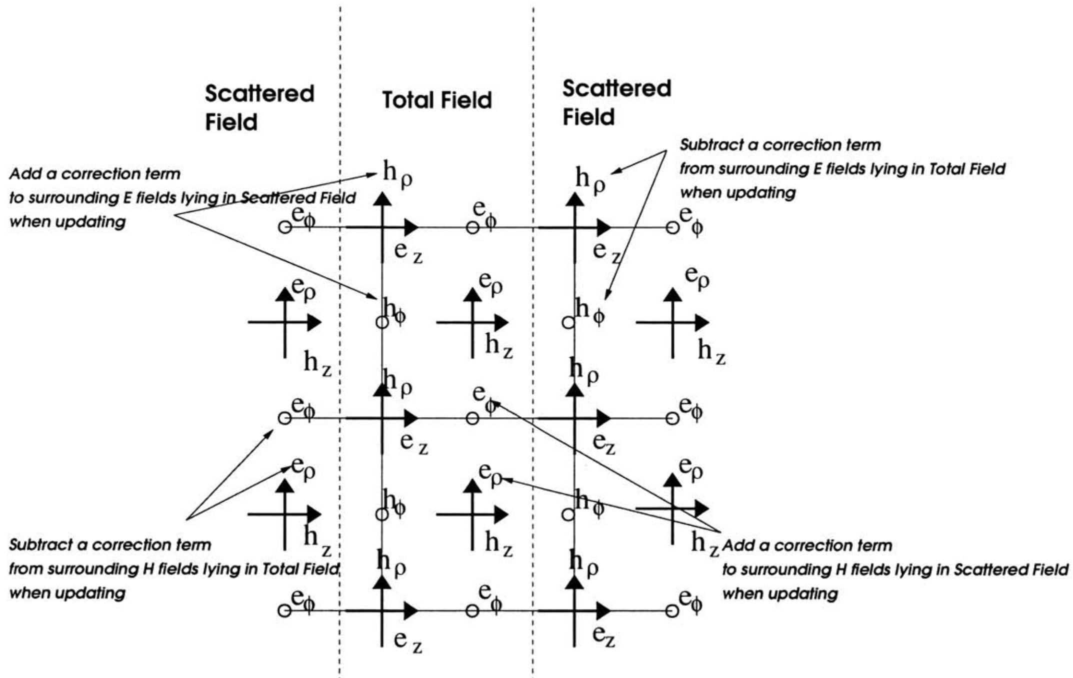

Figure 2-6: The BOR FD-TD fields for which a correction term must be added or subtracted during each update. These fields lie near the left and right boundaries between total and scattered field.

The correction terms are usually generated through some analytical expression to produce a wave at the desired incident angle and frequency. Since FD-TD is calculated in the time domain, a Gaussian pulse excitation is used to allow for multiple incident frequencies to be analyzed per trial. Most often the Gaussian pulse is modulated near the center frequency. This arrangement will concentrate the wave's power at the frequencies of interest. Afterwards, the calculated field quanties can be Fourier transformed to obtain the fields for a particular frequency.

For a body of revolution geometry in FD-TD, the incident fields can be given in

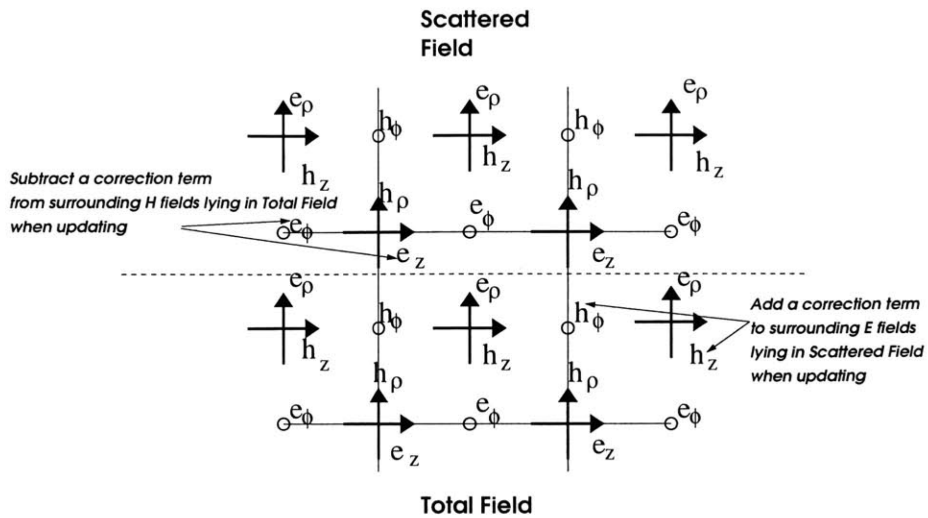

Figure 2-7: The BOR FD-TD fields for which a correction term must be added or subtracted during each update. These fields lie near the top boundary between total and scattered fields.

terms of horizontal and vertical polarization components,

$$\vec{E}_i = \left(E_h\hat{h} + E_v\hat{v}\right) P\left(t - \frac{\hat{k}_i \cdot \hat{r}}{c}\right) \tag{2.145}$$

$$\vec{H}_i = \frac{1}{\eta}\hat{k}_i \times \vec{E} = \frac{1}{\eta}\left(-E_h\hat{v} + E_v\hat{h}\right) P\left(t - \frac{\hat{k}_i \cdot \hat{r}}{c}\right) \tag{2.146}$$

$$\hat{r} = x\hat{x} + y\hat{y} + z\hat{z} \tag{2.147}$$

$$\hat{k}_i = -\hat{x}\sin\theta_i - \hat{z}\cos\theta_i \tag{2.148}$$

$$\hat{r} \cdot \hat{k}_i = -x\sin\theta_i - z\cos\theta_i = -\rho\cos\phi\sin\theta_i - z\cos\theta_i \tag{2.149}$$

$$\begin{aligned} \hat{h} &= \hat{x}\cos\theta_i - \hat{z}\sin\theta_i \\ &= r\hat{h}o\cos\theta_i\cos\phi - \hat{\phi}\cos\theta_i\sin\phi - \hat{z}\sin\theta_i \end{aligned} \tag{2.150}$$

$$\hat{v} = \hat{y} = \hat{\phi}\cos\phi + \hat{\rho}\sin\phi. \tag{2.151}$$

The modulated Gaussian pulse P, with a pulse width of σ and a modulation frequency of f, is defined as,

$$P(\tau) = e^{-\tau^2/2\sigma}\sin(2\pi f\tau). \tag{2.152}$$

For a BOR arrangement, the ϕ dependence must be represented with Fourier modes. Thus the expressions for the incident fields are decomposed into Fourier components. This produces,

$$e^{\rho}_{0,u} = \frac{1}{2\pi}\int_0^{2\pi} (E_h \cos\theta_i \cos\phi + E_v \sin\phi)\, P\left(t - \frac{\hat{k}_i \cdot \hat{r}}{c}\right) d\phi \tag{2.153}$$

$$e^{\rho}_{m,u} = \frac{1}{\pi}\int_0^{2\pi} (E_h \cos\theta_i \cos\phi + E_v \sin\phi)\, P\left(t - \frac{\hat{k}_i \cdot \hat{r}}{c}\right) \cos m\phi d\phi. \tag{2.154}$$

Usually a Gaussian quadrature technique is used to numerically compute these integrals.

Though an analytical form of the incident wave is available and the correction terms are usually generated on the fly in normal FD-TD programs, this is not the only method. For example, the correction terms could have been calculated far in advance and stored on disk. The correction terms corresponding to each time index are independent of the correction terms of other time indices. Furthermore, they are also independent of any activity within the computational domain. This degree of independence will permit the development of the multiple region FD-TD method as described in the next chapter.

2.7 Near to Far Field Transformation

Calculation of RCS requires information about the scattered fields in the far field. Huygens' principle is used to calculate the far field from the near field. The electric and magnetic fields outside a closed region containing the excitation sources can be determined from the tangential fields on the surface, S', of that region. The formulation of Huygens' principle in three dimensional free space, assuming time harmonic electromagnetic waves is,

$$\vec{E}(\vec{r}) = \oint_{S'} dS' \{ i\omega\mu \,\bar{\bar{G}}\,(\vec{r},\vec{r}') \cdot \hat{n} \times \vec{H}(\vec{r}') + \nabla\times \,\bar{\bar{G}}\,(\vec{r},\vec{r}') \cdot \hat{n} \times \vec{E}(\vec{r}')\} \tag{2.155}$$

$$\vec{H}(\vec{r}) = \oint_{S'} dS' \{ i\omega\mu \,\bar{\bar{G}}\,(\vec{r},\vec{r}') \cdot \hat{n} \times \vec{E}(\vec{r}') + \nabla\times \,\bar{\bar{G}}\,(\vec{r},\vec{r}') \cdot \hat{n} \times \vec{H}(\vec{r}')\} \tag{2.156}$$

where $\bar{\bar{G}}(\vec{r},\vec{r}')$ is the dyadic Green's function,

$$\bar{\bar{G}}(\vec{r},\vec{r}') = \left[\bar{\bar{I}} + \frac{1}{k^2}\nabla\nabla\right] \frac{e^{ik|\vec{r},\vec{r}'|}}{4\pi|\vec{r},\vec{r}'|}. \tag{2.157}$$

In the far field, ∇ is approximately $ik\hat{k}$ and $[\bar{\bar{I}} - \nabla\nabla]$ is $[\hat{\theta}\hat{\theta} + \hat{\phi}\hat{\phi}]$. Thus, equation 2.155 in the far field becomes,

$$\begin{aligned} \vec{E}(\vec{r}) &= \oint_{S'} dS' \Big\{ i\omega\mu[\hat{\theta}\hat{\theta}\hat{\phi}\hat{\phi}] \cdot \hat{n} \times \vec{H}(\vec{r}') + \\ &\quad + ik[\hat{\phi}\theta - \hat{\theta}\phi] \cdot \hat{n} \times \vec{E}(\vec{r}') \Big\} \frac{e^{ik|\vec{r},\vec{r}'|}}{4\pi|\vec{r},\vec{r}'|}. \end{aligned} \tag{2.158}$$

In 3D FD-TD, the Huygens' surface S' is normally a box that surrounds the entire total field domain and includes the boundary between the total field and scattered field. In BOR FD-TD, S' is usually a cylinder, implemented as the three sided partial outline of a rectangle within the computational domain.

2.8 Numerical Concerns for FD-TD

FD-TD requires the discretization of space into Δ of approximately $\lambda/20$ to $\lambda/10$ for the highest frequency of interest. Time is also discretized into Δt. For 3D FD-TD, the Courant-Friedrichs-Lewy stability criterion states that,

$$\Delta t_{2D} \leq \frac{1}{c\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}}} \tag{2.159}$$

where Δx, Δy, and Δz are the spatial increments. For BOR FD-TD to meet stability requirements, the time increment is dependent on both the spatial increment and mode number,

$$\Delta t_{BOR} \leq \frac{\Delta}{sc} \tag{2.160}$$

where $s \approx \max(\sqrt{2}, m+1)$ and is known as the "Courant stability factor." Though BOR FD-TD reduces the number of total update equations that need to be modified at any time, the stability requirement will create progressively smaller time steps for higher modes. This causes BOR FD-TD to update the equations for more points in time for higher modes.

Furthermore, the discretization of Maxwell's Equations using the central difference method is only an approximation. This imperfection will alter the phase velocity of the wave as it travels through the lattice. A free space wave should have its phase velocity, v_p equal to its group velocity, c. In the FD-TD mesh the phase velocity will be slightly smaller than the group velocity. And v_p will depend on both the frequency and direction of propagation. This aberration in phase velocity due to the mesh is known as numerical dispersion. This dispersion can be reduced by making $\Delta\tau = c\Delta t$ larger However, $c\Delta t$ has an upper limit to meet the stability requirement. Another way to minimize numerical dispersion is to reduce the spatial step size Δ. It is desirable for the step size to be small enough that the wavelength $\lambda \geq 10\Delta$ but in most applications Δ is chosen so that $\lambda \geq 20\Delta$.

2.9 Computational Expense of FD-TD

The stability requirements and the need to minimize numerical dispersion causes FD-TD programs to require both a large amount of memory and a long duration of time for simulations. Shown in Figure 2-8 is a chart that gives the approximate time and memory needed for a Sun Blade 1000 machine running BOR FD-TD to model a 3 meter deep and 1.5 meter wide cavity for a range of frequencies commonly used in radar analysis. As shown in the chart, at X-band the simulation would require several million years to complete. Also note that the calculation of the time requirements assumes that core memory is available. Given that several gigabytes of memory is needed at X-band, most computers would need to use virtual memory. This fact becomes more strongly evident when 3-D FD-TD instead of BOR FD-TD is used. As shown in Figure 2-9, 3-D FD-TD has a similiar computational time requirement but has a much greater memory requirement. As shown by both charts, both BOR FD-TD and FD-TD cannot be used to solve for electrically large cavities.

The multiple-region FD-TD method that will be introduced in the next section will help reduce some of the memory requirements. This method will also provide a possibility of reducing computational time by eliminating the need for virtual memory and creating a situation where parallel computing can be applied.

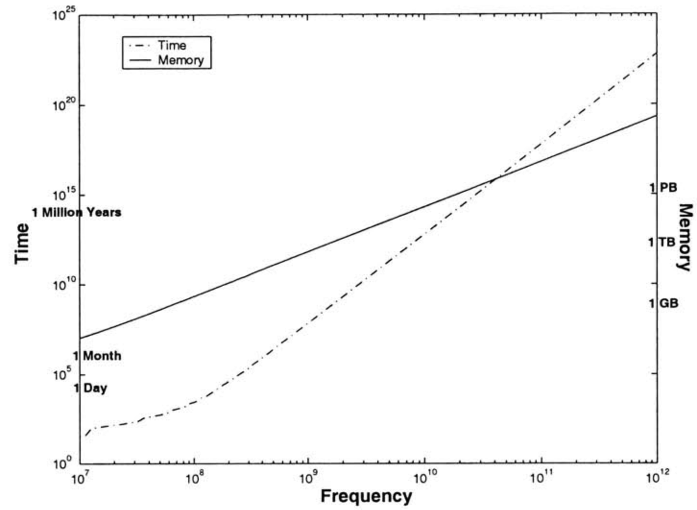

Figure 2-8: Computational demands of BOR FD-TD as estimated for a Sun Blade 1000 Computer.

2.10 Summary

Both the BOR FD-TD and 3D FD-TD algorithms were presented. The FD-TD method provides a means to model electromagnetic behavior in the time domain through the use of discretized Maxwell's Equations. The computational domain is truncated using a PML absorbing boundary condition. The distinction between scattered field and total field within that computational domain allows for plane wave sources to be implemented. Also presented were the stability requirements and numerical dispersion minimization requirement that place restrictions on the granularity at which time and space may be discretized within FD-TD. These requirements cause FD-TD to be computationally expensive, causing very long simulation times and very large computer memory needs.

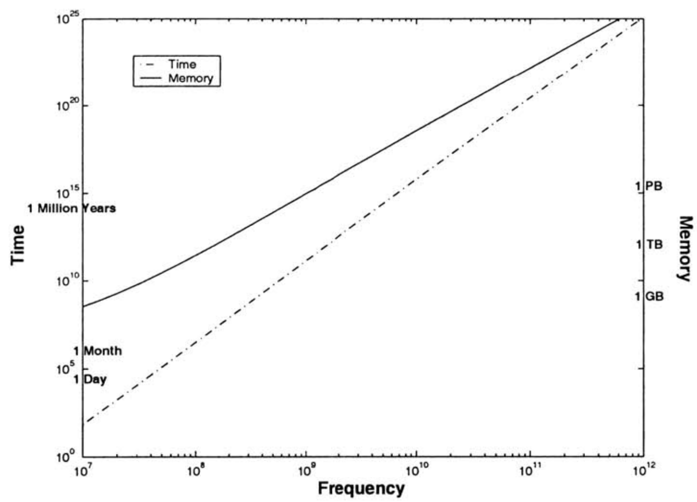

Figure 2-9: Computational demands of 3-D FD-TD as estimated for a Sun Blade 1000 Computer.

Chapter 3

RCS Prediction Using Partitioned Finite-Difference Time-Domain Method

Cavity geometries suitable for partitioning into multiple regions must meet certain requirements. The creation of cavity segments that can be modeled in a piece-wise manner requires the formulation of the inputs into each segment and of knowledge about the outputs of each section. This chapter discusses these issues and develops a partitioned FD-TD approach for duct cavities.

3.1 Theory and Justification for Partitioning

As stated earlier, the partitioned model should be valid for cavities where the energy travels mostly in an in and out fashion, and where coupling between interior features and the back wall is minimal. Examples of this type of cavity are shown in Figure 3-1. In that figure, 2-D cuts of two different body of revolution cavities embedded in a low RCS targets are shown along with the hypothesized paths that the incident waves will take. For future reference, the axis of rotation will be considered to be the z axis while the initial incident wave will approach the cavity in a $-\hat{z}$ direction. Waves propagating in the $-\hat{z}$ direction will be referred to as traveling in the "inward" direction, while the $+z$ direction will be considered the "outward" direction.

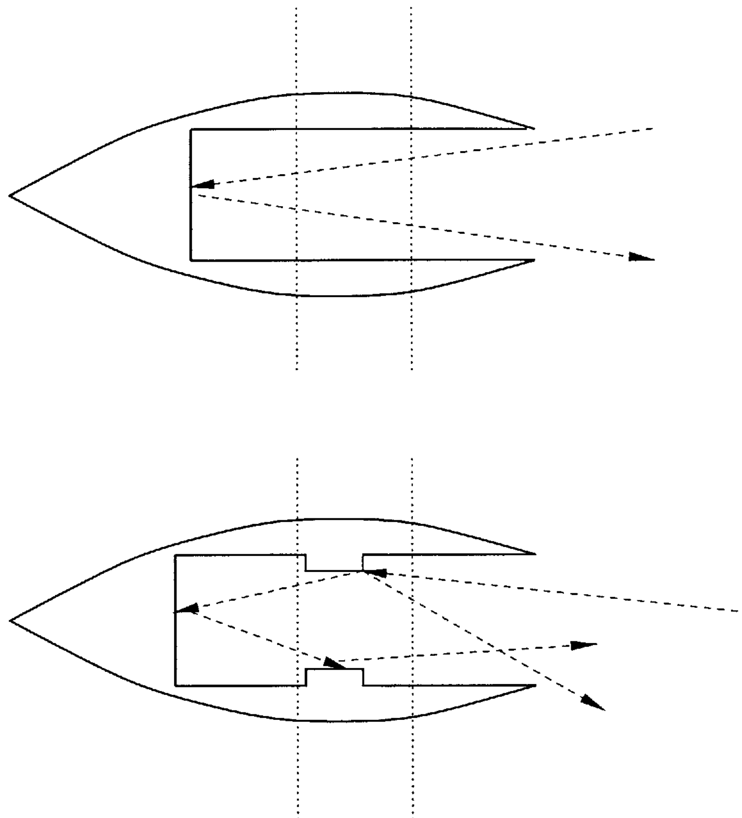

Figure 3-1: Directions of scattering that can be modeled using multiple region FD-TD.

Furthermore, each cavity in Figure 3-1 is divided into three segments with dotted lines. For the top cavity, the path of energy travels through each segment twice: once when it propagates inward in the $-\hat{z}$ direction, and once more when it propagates outward in the $+z$ direction. Thus, each segment needs to be modeled twice to capture both the inward and outward activity. Also, as the wave travels inward, the energy that propagate out through the left hand end of each segment must be known in order to find the correct excitation for the next neighboring segment that the wave travels to. Similarly, as the wave travels outward, the energy that propagates out through the right hand end of each segment must be known in order to find the correct excitation for the next neighboring segment that the wave travels to.

For the bottom cavity, again each segment needs to be modeled twice. However, note that the center segment has energy traveling in three paths: the inward incident energy, the outward propagating energy caused by the back wall reflection, and also outward propagating energy caused by reflection by features within that segment. Thus, for the center segment, two sets of data need to be known to correctly excite the neighboring segments to the right and left. This is the broader, more general characterization of the activity within the interior of duct cavities.

It is this assumption about the behavior of the incident wave as it enters and leaves each segment that allows for partitioning and piecewise modeling of duct cavities. Thus the concept of the multiple-region FD-TD method lies in recording the electromagnetic activity as energy leaves each segment, and then exciting neighboring segments with those recorded fields.

3.2 Partitioning and Classification of Cavity Segments

The implementation of multiple region FD-TD relies on categorizing each segment of the partitioned cavity as one of five cases.

Case 1 The first segment which includes the incident fields.

Case 2 Segments where the waves generally propagate from the opening toward the bottom of the cavity in the $-\hat{z}$ direction.

Case 3 Segment that includes the bottom of the cavity. Waves bounce and start traveling toward the mouth of the cavity in the $+\hat{z}$ direction.

Case 4 Segments where the waves generally propagate from the bottom of the cavity toward the opening of the cavity in the $+\hat{z}$ direction.

Case 5 A segment similar to Case 1 but where the waves now travel out of the cavity opening.

Figure 3-2 gives a visual summary of the cases. As shown in the figure, Case 1 and 5 share the same physical part of the cavity. Case 2 and 4 likewise share the same structure. Though the physical structure of modeled segments may be the same, these cases differ in how and where fields are recorded and artificially recreated within each segment. The cavity in Figure 3-2 is divided into three segments, thus creating only one instance of Case 2 and one instance of Case 4. Cavities that are divided into more than three segments will have multiple instances of Case 2 and Case 4. Cavities that are divided into two segments will not have any instances of Case 2 or Case 4.

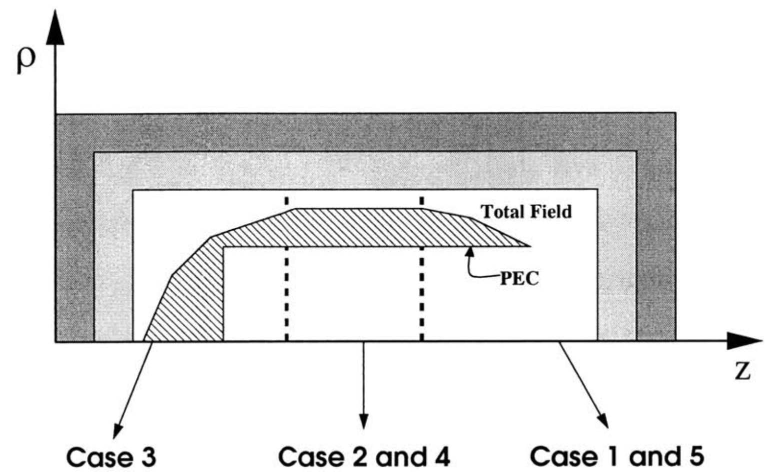

Figure 3-2: Partitioning the cavity into three segments with corresponding case numbers.

3.2.1 Case 1

Case 1 models the front portion of the cavity as a complete problem. Figure 3-3 contains a schematic for the computational domain of Case 1. That is, both the interior and exterior of the front portion of the cavity are modeled simultaneously. This arrangement will allow the MR FD-TD method to calculate the diffraction from the front edges of the cavity. The exterior of the cavity is surrounded by a scattered field layer and a PML layer, as in the normal unpartitioned FD-TD algorithm. The interior of the cavity is terminated with a layer of PML that is disconnected from the other PML that surrounds the exterior of the cavity. The reasoning for this arrangement will be made clear in the discusson below when the recording of the field activity for later retrieval and use is described.

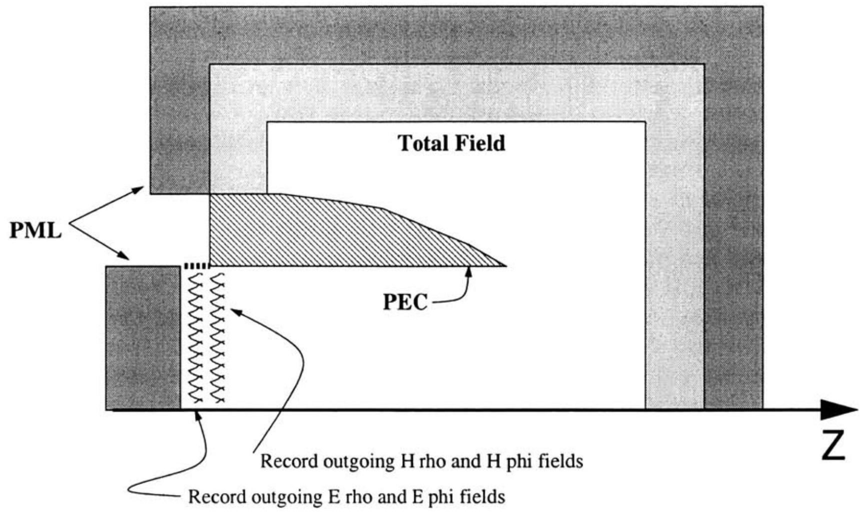

Figure 3-3: Schematic for the computational domain of Case 1.

Modeling of the Incident Wave

The incident wave for Case 1 is created in nearly the same manner as in the normal, unpartitioned FD-TD method. The proper electric and magnetic fields are subtracted at the scattered/total field boundaries as discussed in Figures 2-6 and 2-7. A small detail that deserves attention is that the incident field calculations need to be identical to those produced when the entire cavity is modeled in normal FD-TD. Usually the delay term and incident angle depend on the dimensions and orientation of the cavity. Thus, when modeling Case 1, prior knowledge about the exact dimensions of the whole cavity is needed to create the correct excitation. However, as shown in Figure 3-3, the exterior layer of scattered field does not extend completely around the entire cavity. No region of scattered field is created in the interior of the cavity at the end where total fields interacts with the PML. This end of the cavity should only see the electromagnetic activity that enters through the mouth of the cavity on the right hand side. This same activity will propagate further into the cavity and needs to be recorded in an unaltered form to create that effect. The lack of a scattered field region puts more stress on the PML, but the special PML used to absorb the interior activity is much thicker than the normal PML used for the rest of the problem.

Recording Fields

As shown in Figure 3-3, the fields near the boundary with the PML in the interior of the cavity will be recorded. By recording the fields at this location, one can capture the profile of the electromagnetic activity that will enter into the neighboring segment lying to the left of Case 1. The PML on the interior of the cavity will absorb the incident fields and allow the recorded fields to be free from artifacts of the artificial geometry created by the partitioning. Any scattering from segments further in the interior of the cavity will be handled by subsequent cases and can be modeled independently.

Note that the electric and magnetic fields are not recorded at the same z index. Rather, they are recorded at $\frac{1}{2}\delta$ apart. Also, the PEC of the interior of the cavity is artificially extended by one delta to accommodate this recording scheme. This extension is shown in Figure 3-3, and in subsequent figures with a heavy dotted line. The reasoning behind this setup will be made clear in Section 3.2.2 when the discussion will focus on replaying the recorded field activity into Case 2.

3.2.2 Case 2

Case 2 models the second segment of the cavity using only the interior surface. Figure 3-4 contains a schematic for the computational domain of Case 2. Unlike Case 1, or conventional FD-TD, Case 2 does not have an exterior layer of scattered field and PML. Since only the interior of the cavity needs to be modeled, and the interior surface is PEC, it is appropriate to ignore the exterior of the cavity and simply truncate the computational domain. The added advantages are a conservation of computer memory, and shorter simulation time when this technique is used. Note that for the sake of simplicity in the figure, the interior surface in the figure is made parallel to the z axis, so the PEC becomes a straight slab when the exterior surface is ignored. Cavities with various features on the interior can also be modeled using the multiple region FD-TD method. PML is placed at both ends of the cavity to allow for consistency when the fields must be recorded to be rebroadcast into neighbor segments of the cavity.

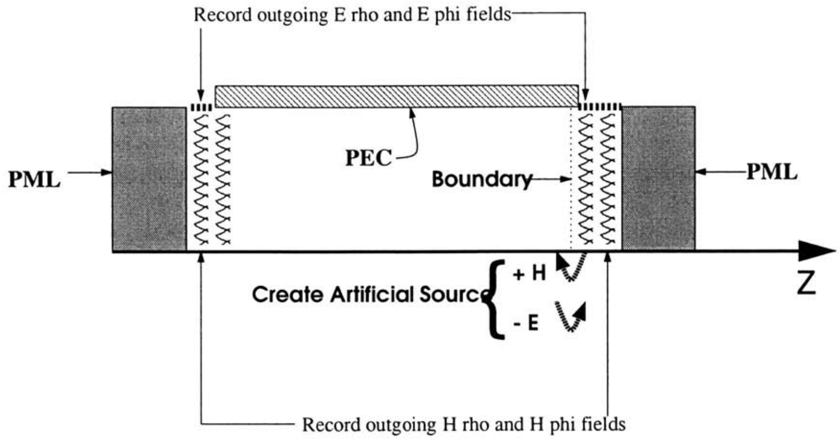

Figure 3-4: Schematic for the computational domain of Case 2.

Modeling of the Incident Wave

The data that was recorded from Case 1 is used to create an artificial source within Case 2 as shown in Figure 3-4. The creation of this source involves both adding in H fields when calculating E fields directly to the left of the boundary, and subtracting E fields when calculating H fields directly to the right of the boundary. This method arises naturally from the structure of the FD-TD lattice, as shown in Figure 3-5. Furthermore, this technique also ensures that the plane wave will propagate in only one direction. This approach is similar to the total and scattered fields arrangement to create plane waves in a conventional, unpartitioned FD-TD.

Recall the discussion in Section 2.6 regarding the creation of the plane wave source in the normal, unpartioned FD-TD algorithm. Though analytical expressions were developed to calculate the desired excitation, on-the-fly as needed, there is nothing to prevent obtaining those same fields through other means, and recording them in advance. Creating a plane wave source would simply mean loading the recorded fields into the lattice during the simulation. The creation of the artificial plane wave within Case 2 exactly follows this line of reasoning, using the output of Case 1 as input.

Figure 3-5 shows that creating the artificial source in Case 2 requires that the E and H fields from the previous segment of the cavity be recording from two neighboring lattice cells along the z axis rather than from the cells with the same z index. Thus, Case 1 was artificially extended in the $-\hat{z}$ direction to create a perfect match with the locations at which E and H must be altered in Case 2. Without this extension, there would be a half delta mismatch in the z direction. Although the difference of a half delta may not significantly affect the overall calculation of scattering and RCS, the creation of the extensions allow for a more correct, complete solution.

The altered update equations that correspond to Figure 3-5 are,

$$\begin{aligned} e_\rho|_{i+1/2,k+1/2}^{n+1/2} &= e_\rho|_{i+1/2,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta z}(h_\phi|_{i+1/2,k}^{n} - \\ &\quad -(h_\phi|_{i+1/2,k+1}^{n} + h_\phi^{recorded})) + \\ &\quad +\eta_0 \frac{m\Delta\tau}{(i+1/2)\Delta\rho} h_z|_{i+1/2,k+1/2}^{n} \end{aligned} \tag{3.1}$$

$$\begin{aligned} e_\phi|_{i,k+1/2}^{n+1/2} &= e_\phi|_{i,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta\rho}\left(h_z|_{i-1/2,k+1/2}^{n} - h_z|_{i+1/2,k+1/2}^{n}\right) + \\ &\quad +\eta_0 \frac{\Delta\tau}{\Delta z}\left((h_\rho|_{i,k+1}^{n} + h_\rho^{recorded}) - h_\rho|_{i,k}^{n}\right) \end{aligned} \tag{3.2}$$

$$\begin{aligned} h_\rho|_{i,k}^{n+1} &= h_\rho|_{i,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta z}\left(e_\phi|_{i,k+1/2}^{n+1/2} - (e_\phi|_{i,k-1/2}^{n+1/2} - e_\phi^{recorded})\right) \\ &\quad + \frac{1}{eta_0}\frac{m\Delta\tau}{i\Delta\rho} e_z|_{i,k}^{n+1/2} \end{aligned} \tag{3.3}$$

$$\begin{aligned} h_\phi|_{i+1/2,k}^{n+1} &= h_\phi|_{i+1/2,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta\rho}\left(e_z|_{i+1,k}^{n+1/2} - e_z|_{i,k}^{n+1/2}\right) + \\ &\quad + \frac{1}{eta_0}\frac{\Delta\tau}{\Delta z}\left((e_\rho|_{i+1/2,k-1/2}^{n+1/2} - e_\rho^{recorded}) - e_\rho|_{i+1/2,k+1/2}^{n+1/2}\right). \end{aligned} \tag{3.4}$$

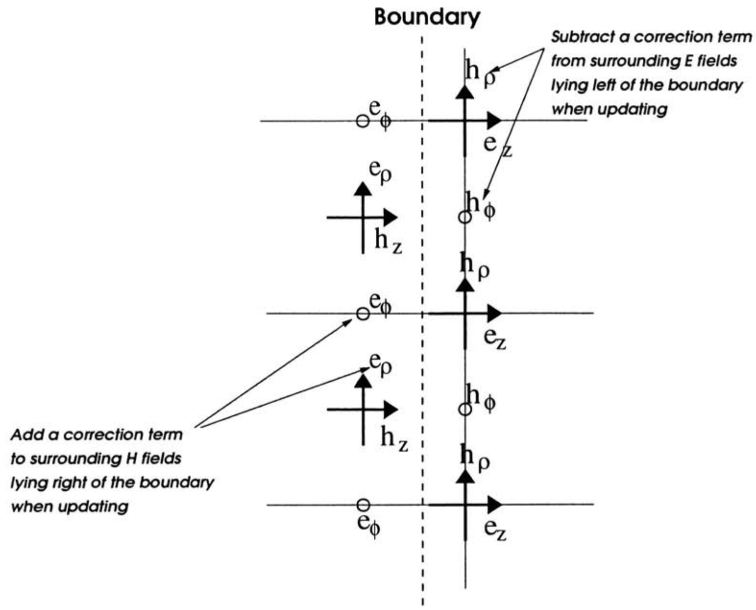

Figure 3-5: Schematic for the creation of artificial source in Case 2 and Case 3. Plane wave will propagate in the $-\hat{z}$ direction.

Recording Fields

Scattering from Case 2 can propagate in both the $-\hat{z}$ and $+\hat{z}$ directions. Therefore, the electromagnetic activity is recorded at both ends of the segment as indicated in Figure 3-4. However, the $+\hat{z}$ data must be recorded to the right of the boundary at which the artificial source is created. Recall that the excitation introduced in Case 2 does not propagate in the $+\hat{z}$ direction. Therefore, this arrangement will allow the recorded data to only contain the scattering information, and prevent any contamination from the incident pulse. As in Case 1, the E and H fields are recorded at one half delta apart to facilitate the creation of artificial sources in neighboring segments. Likewise, the rationale for the artificial extensions on both ends of Case 2 is the same as that given in the previous section for Case 1.

3.2.3 Case 3

Case 3 models the terminated end of the cavity using only the interior surface. Figure 3-4 contains a schematic for the computational domain of Case 3. As was done for Case 2, the exterior of the cavity is ignored and the computational domain is simply truncated. Since the bottom of the cavity is PEC, PML is only placed at one end to facilitate recording the scattered energy.

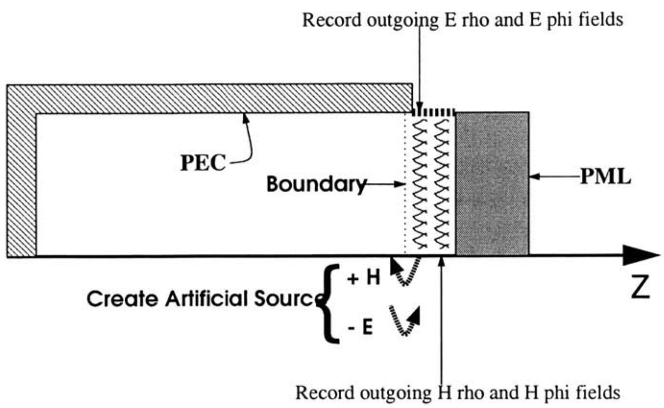

Figure 3-6: Schematic for the computational domain of Case 3.

Modeling Incident Wave

The creation of the artificial source in Case 3 follows the same technique as in Case 2. Figure 3-5, detailing the fields involved in creating source that travels in a $-\hat{z}$ direction, is applicable to Case 3 as well. Furthermore, the Case 2 equations for creating the incident wave (Equations 3.1 to 3.4) are applicable to Case 3 as well.

Recording Fields

Data is recorded in the same manner as Cases 1 and 2. However, the data must be recorded to the right of the boundary at which the artificial source is created. The artificial source propagates only in the $-\hat{z}$ direction in Case 3. Thus the recorded data will only contain the scattering resulting from reflection off of the terminated end and from the interior of the cavity, and not from the incident wave. The layer of PML to the right of the cells at which the fields are recorded, preventd any spurious reflections.

3.2.4 Case 4

Case 4 is complementary to Case 2 and shares the same geometry. Case 4 occurs after the main pulse has traveled into and out of Case 3. Thus the main pulse will now propagate in the $+\hat{z}$ direction. Figure 3-7 gives the schematic for the computational domain of Case 4.

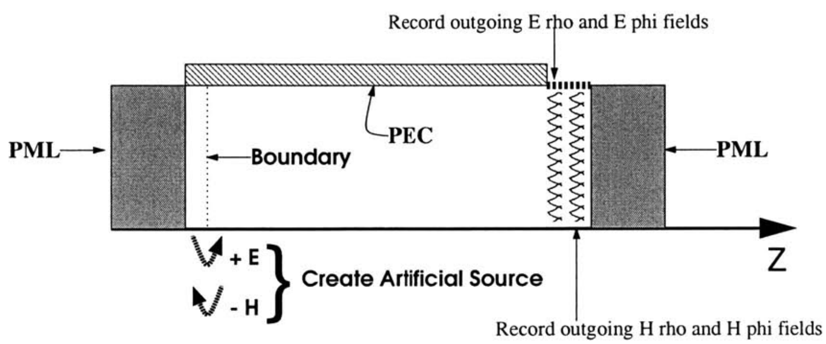

Figure 3-7: Schematic for the computational domain of Case 4.

Modeling Incident Wave

Whereas the artificial source was on the right hand end of the cavity segment for Case 2, the source is now placed on the left hand end for Case 4. Furthermore, due to the lattice structure, the creation of the artificial source is not the same as in Cases 2 or 3. Compare Figure 3-8, which is valid for Case 4 and Case 5, to Figure 3-5 which is valid for Case 2 and 3. Specifically, the equations that must be altered are,

$$\begin{aligned} e_\rho|_{i+1/2,k+1/2}^{n+1/2} &= e_\rho|_{i+1/2,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta z}(h_\phi|_{i+1/2,k}^{n} - \\ &\quad -(h_\phi|_{i+1/2,k+1}^{n} - h_\phi^{recorded})) + \\ &\quad +\eta_0 \frac{m\Delta\tau}{(i+1/2)\Delta\rho} h_z|_{i+1/2,k+1/2}^{n} \end{aligned} \tag{3.5}$$

$$\begin{aligned} e_\phi|_{i,k+1/2}^{n+1/2} &= e_\phi|_{i,k+1/2}^{n-1/2} + \eta_0 \frac{\Delta\tau}{\Delta\rho}\left(h_z|_{i-1/2,k+1/2}^{n} - h_z|_{i+1/2,k+1/2}^{n}\right) + \\ &\quad +\eta_0 \frac{\Delta\tau}{\Delta z}\left((h_\rho|_{i,k+1}^{n} - h_\rho^{recorded}) - h_\rho|_{i,k}^{n}\right) \end{aligned} \tag{3.6}$$

$$\begin{aligned} h_\rho|_{i,k}^{n+1} &= h_\rho|_{i,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta z}\left(e_\phi|_{i,k+1/2}^{n+1/2} - (e_\phi|_{i,k-1/2}^{n+1/2} + e_\phi^{recorded})\right) \\ &+ \frac{1}{eta_0}\frac{m\Delta\tau}{i\Delta\rho} e_z|_{i,k}^{n+1/2} \end{aligned} \tag{3.7}$$

$$\begin{aligned} h_\phi|_{i+1/2,k}^{n+1} &= h_\phi|_{i+1/2,k}^{n} + \frac{1}{\eta_0}\frac{\Delta\tau}{\Delta\rho}\left(e_z|_{i+1,k}^{n+1/2} - e_z|_{i,k}^{n+1/2}\right) + \\ &+ \frac{1}{eta_0}\frac{\Delta\tau}{\Delta z}\left((e_\rho|_{i+1/2,k-1/2}^{n+1/2} + e_\rho^{recorded}) - e_\rho|_{i+1/2,k+1/2}^{n+1/2}\right). \end{aligned} \tag{3.8}$$

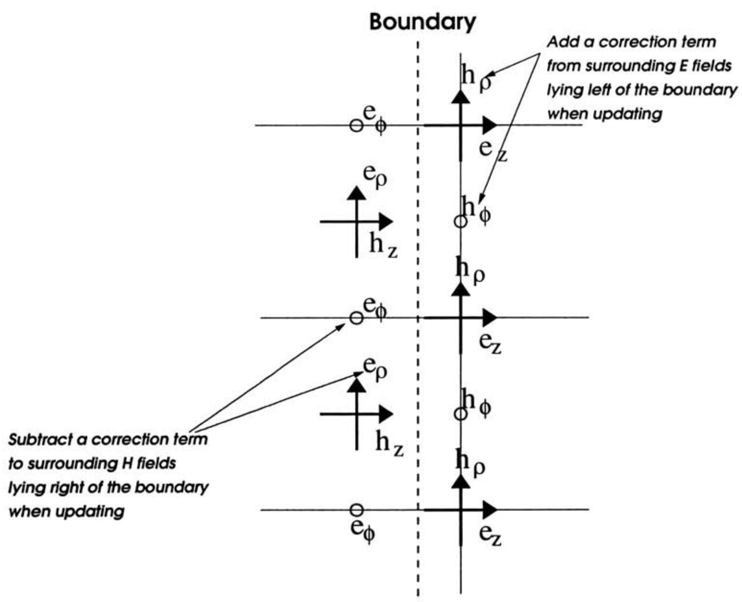

Figure 3-8: Schematic for the creation of artificial source in Case 4 and Case 5. Plane wave will propagate in the $+\hat{z}$ direction.

Recording Fields

The scattering information is recorded at the right hand end of Case 4. By creating the source on the left hand end and recording on the right hand end, one can model the main pulse as it propagates in the $+\hat{z}$ direction. However, another major component of the scattering that also propagates in the $+\hat{z}$ direction was created when the corresponding instance of Case 2 was modeled. Recall that fields were recorded at both ends of Case 2. Thus the fields that were recorded on the left hand end of Case 2 must be added to the fields that are recorded at the left hand end of Case 4. Otherwise, the source that will be used in subsequent instances of Case 4 or Case 5 will be incomplete. Figure 3-9 gives a visual interpretation of this approach. Also note that the artificial extensions placed in Case 2, and the locations where the E and H field were recorded, allow for an exact alignment with where the fields are recorded in Case 4.

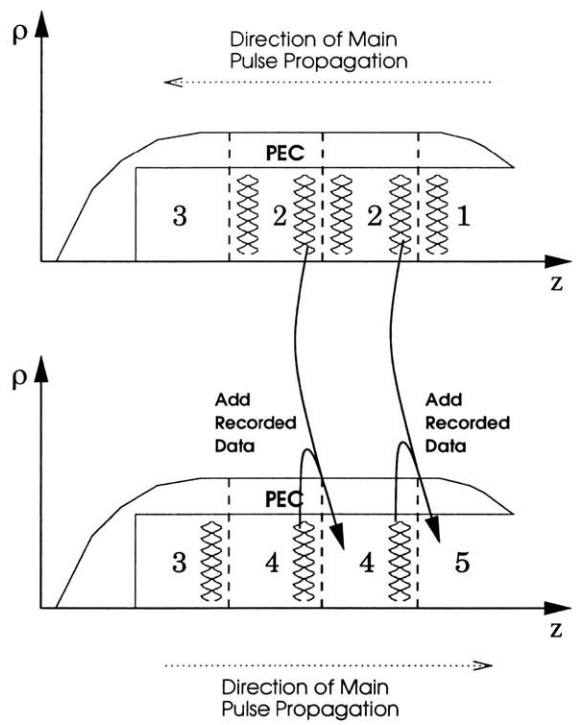

Figure 3-9: Schematic for creation of artificial source in Case 4 that includes scattering from instances of Case 2.

3.2.5 Case 5

Case 5 is complementary to Case 1, and shares the same geometry. Figure 3-10 gives the schematic for the computational domain of Case 5. Unlike Case 2, Case 3, and Case 4, the exterior of the cavity is of interest because the scattering from the lip of the cavity is of interest. Note that the major difference between the computational domains of Case 1 and Case 5 is the lack of a scattered/total field division. This arrangement is correct because the fields in Case 3 were recorded to the right of the boundary that created the artificial plane wave. The data that was recorded from Case 3 only captured the scattering phenomenon and not the original pulse. Thus, in a sense, the entire domain of Case 4 and the entire domain of Case 5, excluding PML, are all scattered field.

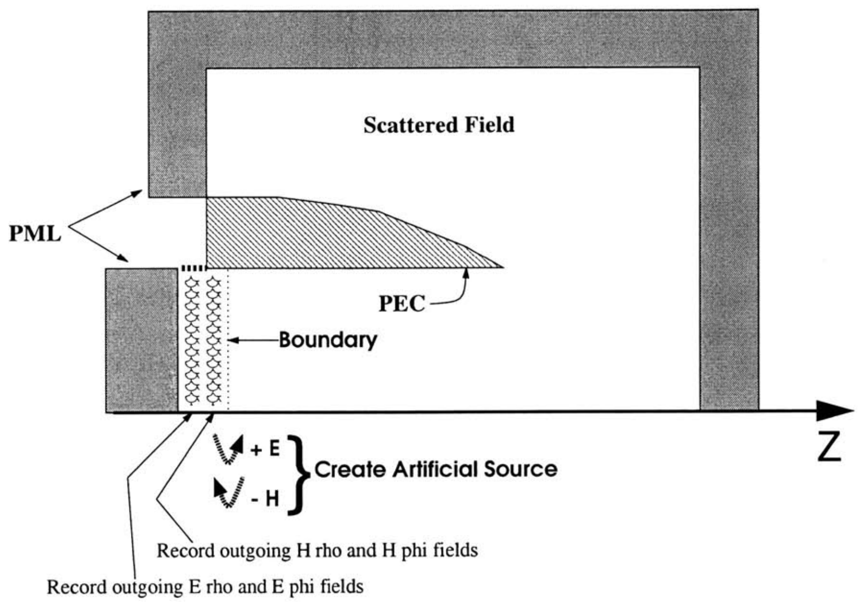

Figure 3-10: Schematic for the computational domain of Case 5.

Modeling Incident Wave

The excitation in Case 5 is created using the same technique as in Case 4. Equations 3.5 to 3.8 that characterize creating the incident wave into Case 4 are applicable to Case 5. Figure 3-8 detailing the fields involved in making this plane wave source is applicable to Case 5 as well.

Recording Fields

In Case 5, there is an option to record fields that lie between the PML and the location of the plane wave source. The use of this option will capture all scattered waves that propagate back in the $-\hat{z}$ direction. The use of the recorded fields will be discussed in Section 3.3.

3.3 Multiple Iterations

The artificial source that is introduced into Case 5 may interact with features within that segment of the cavity, or the opening of the cavity, to create scattering in the $-\hat{z}$ direction. Thus, it would be appropriate to record the fields at the left hand end of Case 5 to capture this scattering. Then the recorded data may be rebroadcast into Case 2 to Case 3 to Case 4 and back to Case 5 to model how it travels into and out of the cavity. This repeat will create what will be referred to as the second "iteration." The idea of iterations can be extended for third, fourth, or even more iterations by simply recording the fields at the left hand end of Case 5 and replaying that data into the rest of the segments each time. This is a form of back and forth scattering which multiple region FD-TD can deal to a limited degree.

Furthermore, the use of multiple iterations forces one to reconsider electromagnetic phenomenology within Case 4. In Case 4, fields are recorded at the right hand end of the cavity while the artificial plane wave source is placed on the left hand end, thus capturing the scattering that travel in the $+\hat{z}$ direction. However, there may be features within Case 4 that will cause some scattering in the $-\hat{z}$ direction. Thus, it would be appropriate to also record the fields on the left hand end of Case 4. Then this recorded data may be used in subsequent iterations: the data that is recorded at the left hand end of an instance of Case 2 will be combined with the data recorded on the left hand end of the corresponding instance of Case 4 during the previous iteration. The true computational domain of Case 4 has not been introduced until now for the sake of clarity. Figure 3-11 reflects the updated version of Case 4.

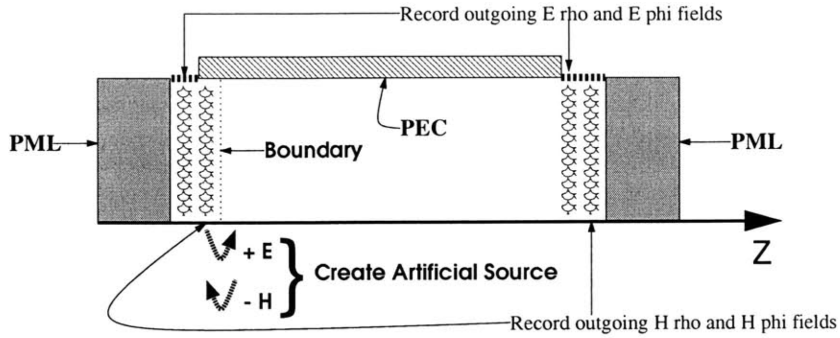

Figure 3-11: Schematic for the computational domain of Case 4.

3.4 Calculation of RCS

RCS is calculated from the fields on a Huygens Surface in Case 1 and Case 5 following the mathematics that were given in the previous chapter. The data on this surface is collected separately for Case 1 and Case 5, and, if multiple iterations are used, other instances of those cases. Then, due to the linearity of Maxwell's Equations, all this data is added to create the final field values. The Huygen's surface only includes the front end of the cavity and cuts into the PEC as shown in Figure 3-12 which uses Case 5 as an example. Note that the Huygens surface does not run through the PEC and into the interior of the cavity. Rather, the fields on the left hand side of the Huygens Surface for points with smaller z indices than the surface of the PEC will be assumed to be zero.

This type of abbreviated Huygen's surface is only appropriate when the exterior of the cavity has very low RCS. For all the cavity geometries tested, this is true. There is some minor noise associated with the discontinuity by cutting into the PEC, but as will be shown in Chapter 4, that contributes very little to the overall RCS.

Another aspect of implementing multiple region FD-TD that becomes important is conservation of hard disk space. Though multiple region FD-TD supplants the lack of memory by recording data onto a hard disk with–in an ideal world–an unlimited capacity, in practice some thought must be given to a thrifty use of disk space whenever possible. The RCS data which is collected separately for Case 1 and Case 5 may be especially large, and running out of disk space can be a real possibility. However,

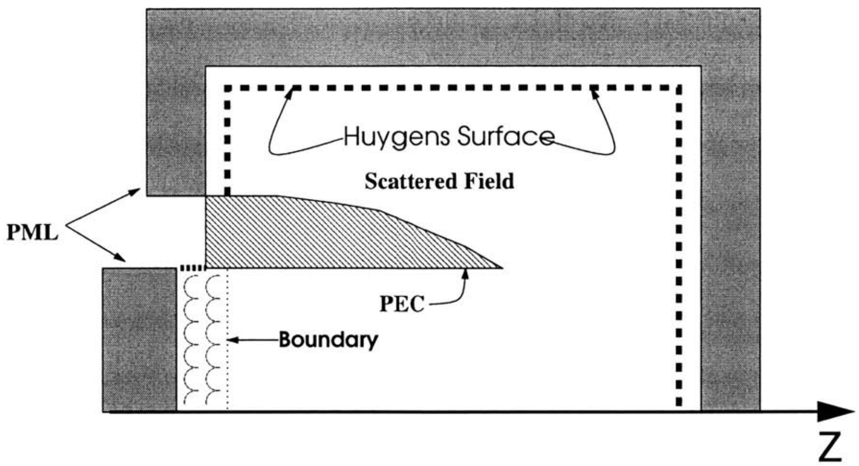

Figure 3-12: Schematic for the Huygens Surface in Case 5.

since the calculation of RCS eventually involves applying a Fourier Transform to the Huygen's surface data, the Fourier Transform can also be applied to Case 1 and Case 5 data separately before storage onto disk. This condenses the data considerably.

3.5 Extension to 3D FD-TD

So far the formulation for the multiple region FD-TD method has only been given in terms of BOR FD-TD. However, implementing the multiple region method in 3D FD-TD would follow a very similar development: the same five cases and the PML geometries can be used in a 3D arrangement. Furthermore, if the duct cavity lay on the z-axis, the fields that must be modified to create the input into each cavity are completely analogous to the BOR FD-TD fields: E_y, E_x, H_y, H_x are altered instead of e_ρ, e_ϕ, h_ρ, h_ϕ. The fields that are scattered from each segment will be recorded as a set of four two dimensional matrices instead of as a vectors at each time step.

3.6 Summary

The method to create a partitioned FD-TD program was presented. The creation of pseudo-incident waves and the recording of outgoing scattered fields from each segment of the cavity allow those segments to be modeled in a piecewise manner.

Furthermore, the use of multiple iterations help to account for any minor back and forth scattering between cavity partitions. Though the partitioning technique was given in terms of BOR FD-TD, the method is equally suitable to implement within 3D FD-TD.

Chapter 4

Results

4.1 Introduction

Though MR FD-TD is the focus of this thesis, it is only one of many possible approaches in a toolkit of all RCS modeling techniques. It is in the best interest of the researcher to choose the most optimal modeling technique for a given situation, while taking into consideration the available computational resources. Therefore, guidelines for selecting the best possible method would be very useful. To illustrate the need for such guidelines, we can compare the RCS of a cavity as predicted by three different modeling methods: conventional FD-TD, MR FD-TD, and a high frequency technique. Shown in Figures 4-1, 4-2, and 4-3 are the Inverse Synthetic Radar (ISAR) images of the RCS predicted by these three modeling methods.

ISAR images will be presented many times in this chapter. Therefore, a brief introduction to ISAR is needed. Like the better known Synthetic Aperture Radar (SAR) technique, ISAR produces a high resolution two dimensional image of the signature of a target. One dimension is "range" which is the measure of line-of-sight distance from the radar to target. The other dimension is cross range, perpendicular to the range. Resolution along this direction can be achieved by moving the radar to create a large antenna aperture, as done by the SAR method, or by assuming a fixed radar system and moving the target as done by ISAR. In this study, the body of revolution cavities are rotated on their center, half way down their axis of revolution. The rotation, discretized "look angles," maps to cross range while the range of frequencies map to down range. The result is a two dimensional image showing the areas of reflectivity in the target.

Figure 4-1 was generated by conventional FD-TD. Since it is an exact approach, this method should be reliably accurate. Of particular interest in this ISAR image is the amount of extended return. Extended return is what appears as areas of reflectivity far down range from the actual target. No physical part of the target exists at this location, As explained previously, it is due to the cavity interior emitting the energy that been delayed by multiple reflections from the side walls.

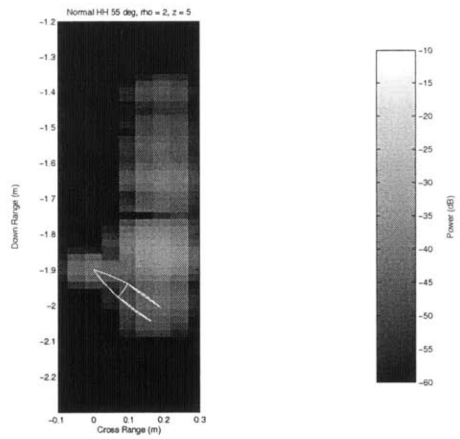

Figure 4-1: ISAR image for conventional FD-TD. For this image and all subsequent images, line-of-sight is upwards towards the cavity opening

Figure 4-2 was generated by Multiple Region FD-TD. This result is very similar to the one generated by conventional FD-TD. However, with just one example, there is not much of a guarantee that MR FD-TD will always generate results comparable to the conventional FD-TD prediction.

Figure 4-3 was generated by a High Frequency Technique. This result differs from the one generated by conventional FD-TD. Much of the extended return is missing from this image, appearing as one isolated spot instead of a long "tail." This isolated spot can be interpreted as a single pulse of reflected activity emanating from the cavity opening.

But with only one example it is premature to discredit high frequency techniques

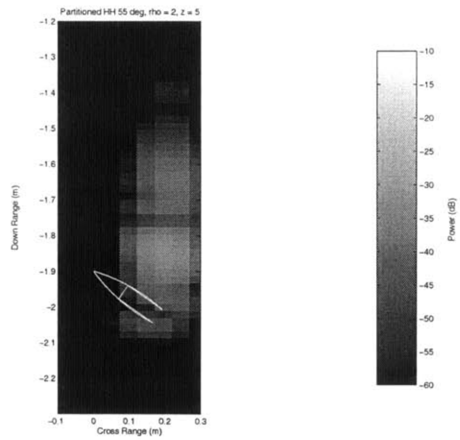

Figure 4-2: ISAR image for Multiple Region FD-TD.

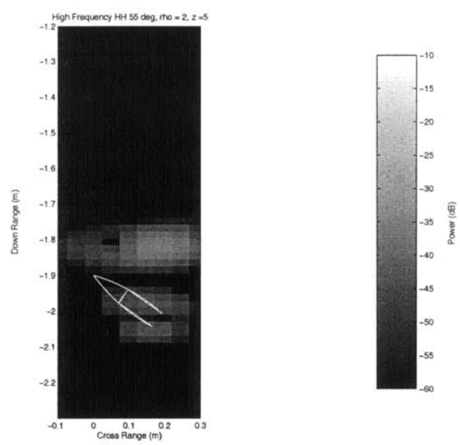

Figure 4-3: ISAR image for a High Frequency Technique.

altogether. For example, Figure 4-4 shows the ISAR image of the conventional FD-TD results for a differently shaped cavity. This cavity does not create much extended return.

Compare the Figure 4-4 with Figure 4-5 which is the ISAR image of the same cavity as modeled in a High Frequency Technique. Here the images are much less dissimilar.

These examples demonstrate that the high frequency technique is not always accurate in cavity modeling although it cannot be completely discredited. MR FD-TD may give more accurate predictions, but that accuracy may possibly be affected by cavity size, incident angle, polarization of the incident wave, and other factors. Prior

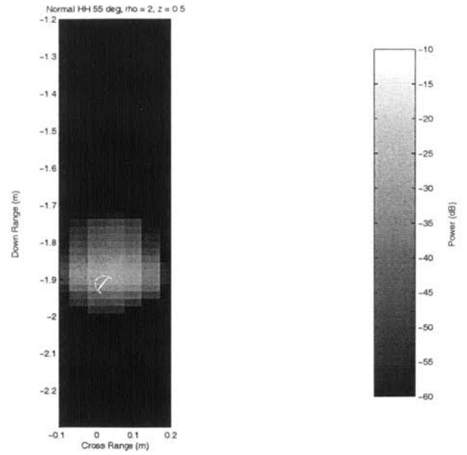

Figure 4-4: ISAR image for conventional FD-TD.

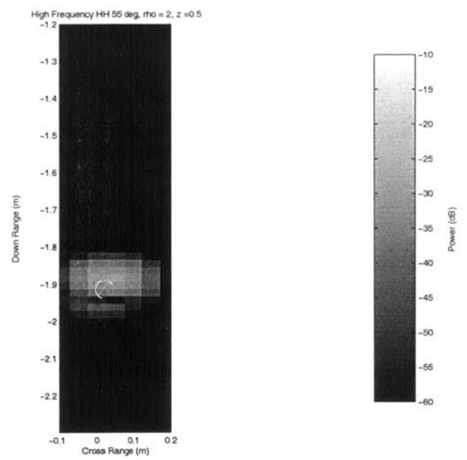

Figure 4-5: ISAR image for the high frequency method.

discussions have shown that conventional FD-TD is not always feasible. But the range of cavity sizes that are feasible has not been investigated.

The development of Multiple-Region FD-TD was undertaken with the idea that there exist classes of cavity geometries that cannot be accurately modeled with either conventional FD-TD or with the high frequency approach. MR FD-TD is meant to bridge the gap between exact approaches and high frequency approaches. Therefore, understanding where and how this gap occurs is key to showing the value of MR FD-TD and understanding its place in the tool-box of RCS modeling techniques.

4.1.1 Overview of the Study

Understanding where and how the gap between conventional FD-TD and the high frequency technique occurs requires a thorough investigation of each of those techniques when applied to cavities of different sizes and shapes. Furthermore, this same investigation must also be carried out for Multiple Region FD-TD to gain insight into its performance relative to the other modeling approaches.

First, straight-duct cavities of a range of sizes were systematically modeled for both polarizations and for different incident angles. This modeling was carried out in conventional FD-TD, multiple region FD-TD, and a high frequency technique.

The second portion of the investigation focused on duct cavities that did not have perfectly straight sideways. This half of the study determined the affect on RCS by changes in the cavity interior walls and the ability of the three prediction approaches to model the activity due to those changes.

4.2 Limits of Computation Feasibility and Validity

4.2.1 Conventional BOR FD-TD

Range of Validity

Given a lack of physical data, the results produced by the conventional unpartitioned FD-TD method will always be considered accurate. As stated earlier, the range of applicability is limited by the computational intractability of modeling large cavities. Therefore, it is necessary to investigate the range of computational feasibility of the conventional BOR FD-TD method.

Range of Computational Feasibility

The size of the target defines the amount of time and memory an FD-TD simulation would require. Time and memory requirements, in turn, demarcate the range of computational feasibility. Simple straight duct cavities of various sizes were used as benchmarks to define the limits of this range. These cavities were embedded in low RCS ogive shells since the electromagnetic activity of the cavity interior was of main interest. The ogive is defined by rotating an arc of a circle on its chord. For all cavities, the frequency of excitation was between 9 and 13 GHz. The FD-TD simulation was allowed to run for enough time steps to be equivalent to the amount of time needed for an electromagnetic wave to traverse a distance equal to 12 times the interior diagonal length of the cavity. This will ensure that the reflected energy, delayed by multiple interactions with the cavity side walls, will have enough time to exit the cavity.

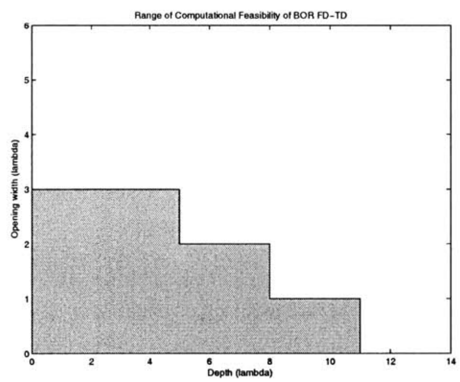

Figure 4-6: Range of feasibility for conventional FD-TD.

Figure 4-6 is a chart showing the range of computational feasibility. Any simulation that would not compile or took longer than two weeks to complete was regarded as not computationally feasible. The dimensions of the cavity opening and the depth are given in terms of the largest λ which was at roughly 3.3 centimeters for 9 GHz. A lack of memory prevented the modeling of a cavity with an opening diameter of 5 or more λ because the FD-TD program would not compile. Even with a 3 or 4 λ wide opening, the cavities were still limited in depth due to a combination of memory requirements and simulation time.

4.2.2 Multiple Region FD-TD

Range of Validity

The investigation of MR FD-TD started with finding the range of cavities sizes for which the approach gave accurate predictions. As was done for conventional FD-TD, simple straight duct cavities of various sizes were used as benchmarks to define this range of validity.

Accuracy is quantitatively determined by comparing the MR FD-TD results with the corresponding results generated by conventional FD-TD. This comparison is done by first converting RCS as a function of frequency into RCS as function of range. Then the correlation coefficient between the two sets of RCS data is calculated. A perfect match would generate a correlation coefficient of 1 while random noise would produce a coefficient close to 0. This method of determining accuracy will be used for all subsequent examples.

For each specific cavity geometry, four different simulations were conducted to cover both polarizations and two different incident angles at 55 and and 20 degrees. For all test cases, a representative setup of 3 segments (2 partitions or "cuts") was used. Tables 4.1, 4.2, 4.3, and 4.4 summarize the correlation scores between multiple region FD-TD and conventional FD-TD for all permutations of cavity size, polarization, and incident angle that were tested.

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.80112	0.87452	0.94522		
2 λ	0.75106	0.85549	0.77526	0.83379	
1 λ	0.70579	0.79701	0.84810	0.89839	0.86178
0.5 λ	0.71106	0.55229	0.50241	0.58239	0.56893

Table 4.1: Summary of the performance of multiple region FD-TD versus conventional FD-TD for straight duct cavity, HH polarization, 20 degrees incident angle.

It was found that MR FD-TD is not suitably accurate for cavities with openings smaller than 1 λ. Otherwise for cavities with openings of 1 λ or greater, multiple region FD-TD is always reasonably accurate. This accuracy is largely independent of

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.87433	0.94333	0.95522		
2 λ	0.84437	0.79439	0.74993	0.89576	
1 λ	0.85327	0.88805	0.88757	0.87787	0.88399
0.5 λ	0.77500	0.59425	0.23812	0.41655	0.47046

Table 4.2: Summary of the performance of multiple region FD-TD versus conventional FD-TD for straight duct cavity, VV polarization, 20 degrees incident angle.

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.81226	0.84042	0.85663		
2 λ	0.72011	0.7573	0.88116	0.92488	
1 λ	0.75000	0.87508	0.83839	0.86002	0.84300
0.5 λ	0.65875	0.65895	0.66809	0.65026	0.67616

Table 4.3: Summary of the performance of multiple region FD-TD versus conventional FD-TD for straight duct cavity, HH polarization, 55 degrees incident angle.

the polarization and angle of the incident wave.

Secondly, as shown earlier, the computed RCS can be used to generate ISAR images. ISAR images can provide a qualitative understanding of the accuracy of the RCS prediction. Figures 4-9 and 4-10 are a pair of ISAR images, showing the conventional FD-TD and MR FD-TD predictions for the same cavity structure. Being only 0.5λ long and 2λ wide, this cavity is not very deep and does not generate much extended return.

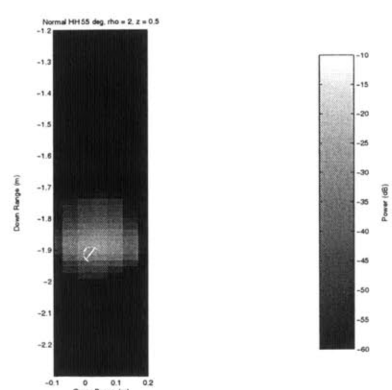

Figure 4-7: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.91618	0.92751	0.85340		
2 λ	0.84420	0.91670	0.89311	0.93615	
1 λ	0.75483	0.97009	0.86880	0.82515	0.83379
0.5 λ	0.61375	0.65885	0.66809	0.65850	0.67646

Table 4.4: Summary of the performance of multiple region FD-TD versus conventional FD-TD for straight duct cavity, VV polarization, 55 degrees incident angle.

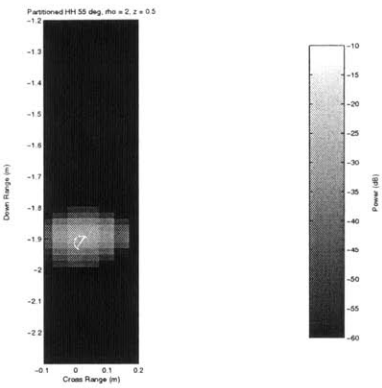

Figure 4-8: ISAR image for MR FD-TD using a 55 degree incident angle and HH polarization.

Figures 4-9 and 4-10 show the ISAR images for a much deeper cavity with a length of 2λ and a width fixed at the same 2λ seen in the previous example. The extra depth creates much more extended return, and MR FD-TD successfully models that activity.

MR FD-TD's ability to model extended return is further demonstrated by applying it to an even deeper cavity. Again the width is fixed at 2λ, but the length is increased to 5λ. The cavity in Figure 4-11 and 4-12 has an extended return that is much longer than the actual depth of the cavity. The length of the extended return is the same in both ISAR images although the part of the extended return farthest down range in the MR FD-TD image is very faint.

From the previous examples, it is tempting to conclude that the length of extended return is mostly determined by the depth of the cavity. However, that is not the case as shown in Figure 4-13. The cavity featured in this ISAR image has the same depth as the cavity in the previous two images at 5λ. However, the width of the cavity

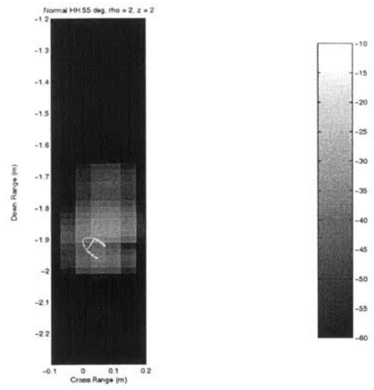

Figure 4-9: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

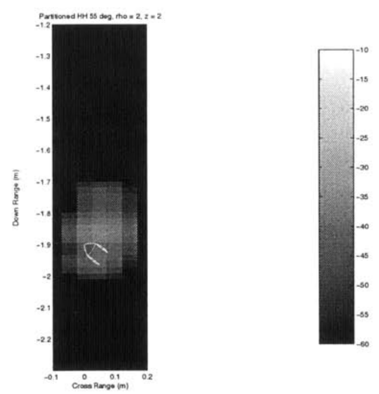

Figure 4-10: ISAR image for MR FD-TD using a 55 degree incident angle and HH polarization.

has been reduced to 1λ. Now, instead of a long extended return, there is only one isolated region of reflectivity that corresponds to the reflection from the interior of the cavity. Nevertheless, MR FD-TD still is able to accurately model this cavity as shown in Figure 4-14.

Furthermore, as will be shown in the discussion on the accuracy of the high frequency technique, a cavity with a large depth will also not necessarily have a long extended return if it also has a very large opening width.

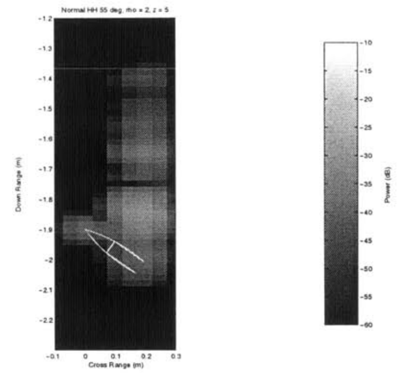

Figure 4-11: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

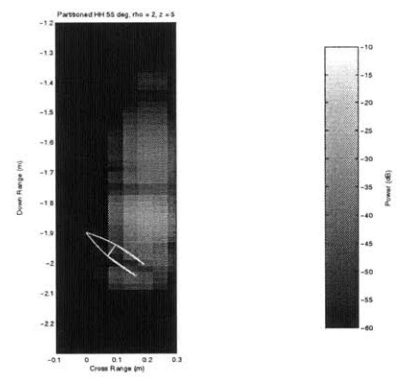

Figure 4-12: ISAR image for MR FD-TD using a 55 degree incident angle and HH polarization.

Range of Computational Feasibility

Figure 4-15 shows the range of cavity sizes where MR FD-TD is computationally feasible, using the same guidelines that were applied to conventional FD-TD. For a given cavity radius, the amount of memory needed is now independent of the depth of the cavity.

However computational time limits how deep the cavities can be. A narrow deep cavity can be partitioned and may not need much memory but the total number of time steps will be very high. Though a wider cavity uses more memory than a narrow one, this is not what limits cavity width. The limitation is due to the fact that wider cavities require more modes and smaller times steps. Therefore cavities modeled by MR FD-TD are only limited by computational time–not memory.

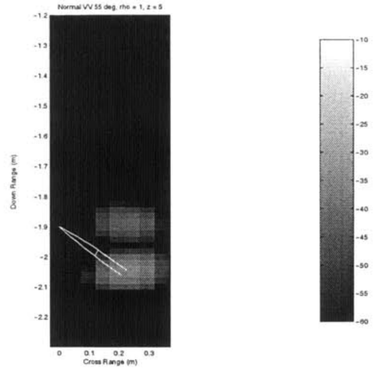

Figure 4-13: ISAR image for conventional FD-TD using a 55 degree incident angle and VV polarization.

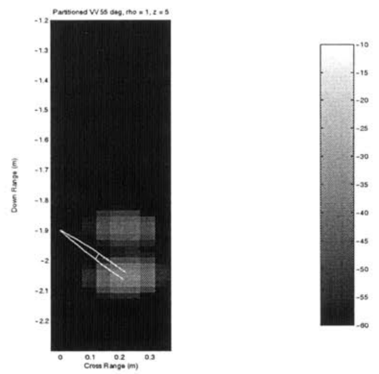

Figure 4-14: ISAR image for MR FD-TD using a 55 degree incident angle and VV polarization.

Effects of Using Fewer or More Partitions

Each partition in the cavity introduces an approximation into an otherwise exact method. This bit of inaccuracy is reflected in the fact that the use of more partitions creates a less accurate solution. The trend is shown in Table 4.2.2 of the correlation scores for one cavity modeled using different numbers of partitions. This cavity was 2 λ wide and 2 λ deep.

	Number of Segments				
	1 segment	2 segments	4 Segments	6 Segments	8 Segments
Correlation	0.8747	0.8555	0.8012	0.7311	0.6034

Table 4.5: Summary of the performance of multiple region FD-TD for various number of segments for the straight cavity.

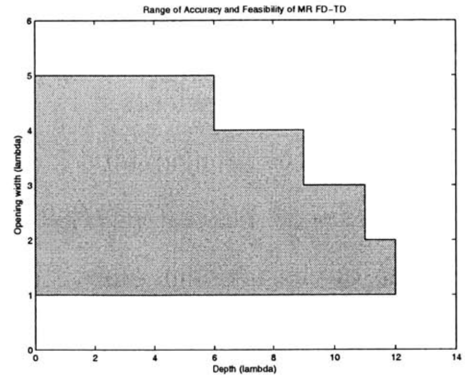

Figure 4-15: Range of feasibility for conventional FD-TD.

Effects of Incident Angle and Polarization

As shown by the tables of correlation scores, There is no significant difference between the correlation scores of 55 degrees and 20 degrees. It would seem likely that incident angle does not affect the accuracy of MR FD-TD so long as that the diffraction from the cavity opening and return from the cavity interior are the most dominant components of RCS. This is not the case for very large angles of incidence. Since the MR FD-TD approach ignores most of the exterior of the cavity, using an incident angle of 90 degrees would not produce accurate results. To confirm this conclusion, a 1λ wide and 5λ deep cavity was modeled for a range of angles using VV polarization. The correlation scores are shown in Table 4.2.2

	Incident Angle in Degrees					
	90	70	55	35	20	0
Correlation	0.4992	0.8238	0.8688	0.8621	0.8876	0.9023

Table 4.6: Summary of the performance of multiple region FD-TD for various angles of incidence

Furthermore, the correlation scores seem independent of the polarization of the incident wave. Therefore, polarization does not affect the accuracy of MR FD-TD.

Effects of Using Fewer or More Back and Forth Iterations

The RCS of a straight duct cavity as generated by the multiple region FD-TD method does not differ significantly when multiple back and forth iterations are used versus when no such iterations are used. This is true regardless of the size of the cavity, incident angle, and polarization of the incident pulse. This knowledge is significant because the additional calculations for extra iterations should be avoided whenever possible. Furthermore, this also shows that significant back and forth scattering–which would make extra iterations necessary–does not occur to an appreciable degree in straight duct cavities.

4.2.3 High Frequency Technique

Range of Validity

Tables 4.7, 4.8, 4.9, and 4.10 show the correlation of high frequency results with conventional BOR FD-TD. As was done for MR FD-TD, both polarizations and two angles of incidence were studied.

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.83546	0.63422	0.38910		
2 λ	0.74994	0.69003	0.49838	0.39801	
1 λ	0.68903	0.59039	0.38901	0.38972	0.32490
0.5 λ	0.45322	0.37825	0.23345	0.29839	0.22921

Table 4.7: Summary of the performance of multiple region FD-TD versus conventional FD-TD for straight duct cavity, HH polarization, 20 degrees incident angle.

It is important to note that results generated by conventional FD-TD are available only for a limited range of cavity sizes. Therefore the scope of correlation scores is bounded as well. However, the available correlation scores show a strong trend: the high frequency technique seems to be reasonably accurate for cavities with an opening of 2 λ or greater and with a depth smaller than the opening. This observation can be translated into Figure 4-16, showing the projected range of validity of the high

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.77344	0.55774	0.34678		
2 λ	0.69320	0.43677	0.35731	0.39054	
1 λ	0.58345	0.45466	0.36467	0.32565	0.23246
0.5 λ	0.39925	0.36667	0.34266	0.23467	0.19235

Table 4.8: Summary of the performance of a High Frequency Method versus conventional FD-TD for straight duct cavities, VV polarization, 20 degrees incident angle.

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.71003	0.46778	0.33456		
2 λ	0.64578	0.35783	0.24567	0.17357	
1 λ	0.33456	0.34501	0.26446	0.20341	0.16548
0.5 λ	0.23050	0.15400	0.14663	0.25634	0.16643

Table 4.9: Summary of the performance of a High Frequency Method versus conventional FD-TD for straight duct cavity, HH polarization, 55 degrees incident angle.

frequency technique.

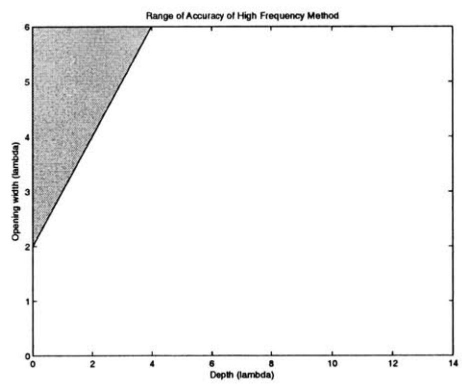

Figure 4-16: Range of validity for the high frequency technique.

In an effort to understand the phenomenology behind this trend, it is useful to study ISAR images of the high frequency method. Figures 4-17 and 4-18 show ISAR images of the RCS as predicted by conventional FD-TD and a high frequency technique. This cavity is 2λ wide and 0.5λ deep, having a depth that is much smaller than the width of the cavity. For the remainder of this thesis, cavities that have widths larger than their depths will be described as "shallow," regardless of the actual di-

	Depth				
Width	0.5 λ	2 λ	5 λ	8 λ	11 λ
3 λ	0.70351	0.45678	0.40246		
2 λ	0.63721	0.46421	0.26443	0.24312	
1 λ	0.34852	0.30562	0.13567	0.23416	0.20122
0.5 λ	0.29700	0.23563	0.20456	0.23356	0.12435

Table 4.10: Summary of the performance of a High Frequency Method versus conventional FD-TD for straight duct cavity, VV polarization, 55 degrees incident angle.

mension of their depth. As shown by the conventional FD-TD results, not much extended return is generated by this shallow cavity, and the high frequency technique does a reasonably good job of matching the exact technique.

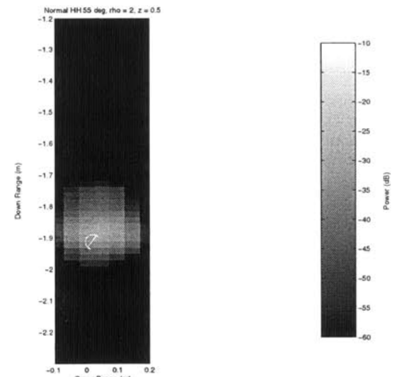

Figure 4-17: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

However, when the depth of the cavity is equal in length to the width, significant extended return is generated. The high frequency technique does not accurately model this phenomenon because it predicts two isolated areas of reflectivity as mapped in Figures 4-20. A logical explanation of these two distinct areas would be first a direct reflection from the rim of the cavity, and then a second delayed return from the interior of the cavity. This prediction differs from the conventional FD-TD prediction shown in Figure 4-19. This ISAR image shows that the cavity is continuously emitting energy and has an extended return that is more than 0.1 meters further down range than predicted by the high frequency technique. Also, compare the high frequency ISAR image with Figure 4-10 which had been introduced earlier in the discussion on

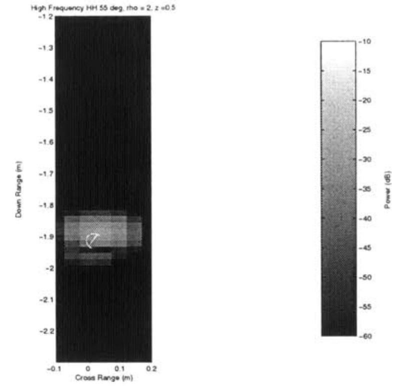

Figure 4-18: ISAR image for the high frequency technique using a 55 degree incident angle and HH polarization.

MR FD-TD which managed to correctly predict the length of the extended return of this particular cavity geometry.

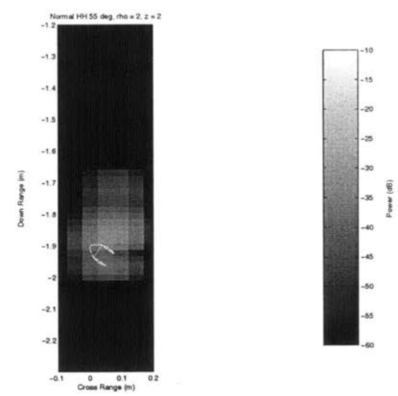

Figure 4-19: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

When the cavity depth is much greater than the cavity opening, it becomes more obvious that the high frequency method is not accurately modeling extended return. Figures 4-22 shows the ISAR image from the high frequency prediction. Again, it predicts the return coming from two groups: first from the rim of the cavity and then a single delayed return from the cavity interior. But, as shown in Figure 4-21, conventional FD-TD predicts a good deal of extended return, indicating a continuous and lengthy stream of energy emanating from the cavity opening. Also, compare the high frequency ISAR image with Figure 4-12 which had been introduced earlier in

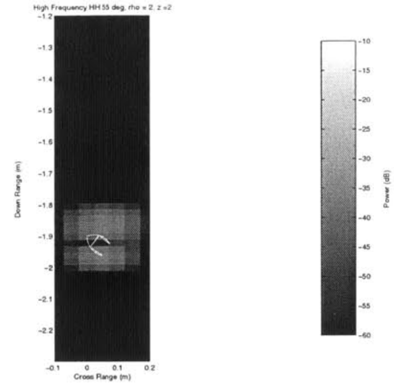

Figure 4-20: ISAR image for the high frequency method using a 55 degree incident angle and HH polarization.

the discussion on MR FD-TD which managed to correctly predict the length of the extended return.

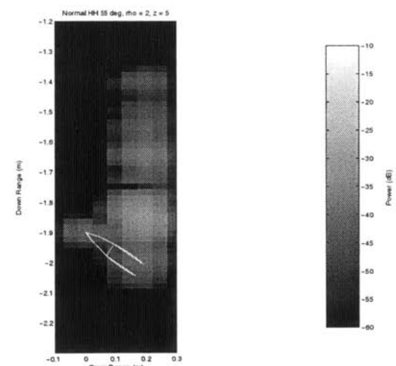

Figure 4-21: ISAR image for conventional FD-TD using a 55 degree incident angle and HH polarization.

The high frequency technique tends to predict the reflected energy from the cavity interior as arriving in a single pulse even though it may be spread out in time. However, when cavities are wider than they are deep, the return from the interior of the cavity does arrive like a single short pulse. These shallow cavities do not create the long "tail" of extended return. Furthermore, these cavities are also precisely the ones that created higher correlation scores for the high frequency technique. Thus, it can be concluded that the high frequency approach becomes a viable cavity modeling technique for shallow straight cavities because the expected extended return is mini-

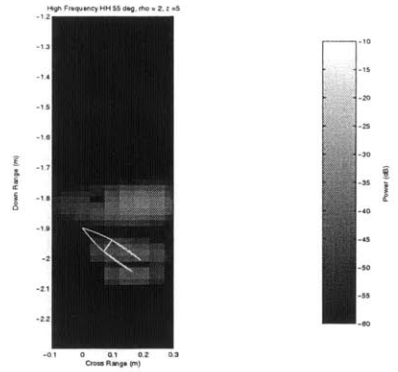

Figure 4-22: ISAR image for the high frequency method using a 55 degree incident angle and HH polarization.

mal. Note that these cavities must be shallow. Cavities with small openings may have minimal extended return yet the high frequency method will not provide an accurate prediction. Shown in Figure 4-23 is an ISAR image of the RCS of a 1λ wide by 5λ deep cavity as predicted by conventional FD-TD. There is not much extended return, having only a single isolated reflected pulse emanating from the cavity interior. The high frequency technique had previously been shown to be adequate in predicting this type of return. But as shown in Figure 4-24, the high frequency method does not predict any return from the cavity interior. Shallowness is a necessary feature of cavities that can be accurately modeled by the high frequency approach. Therefore, the range of validity of the high frequency method as derived from the tables of correlation scores is confirmed.

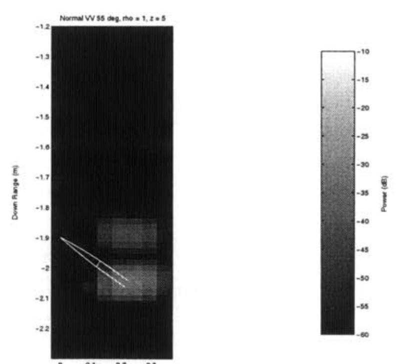

Figure 4-23: ISAR image for conventional FD-TD using a VV degree incident angle and HH polarization.

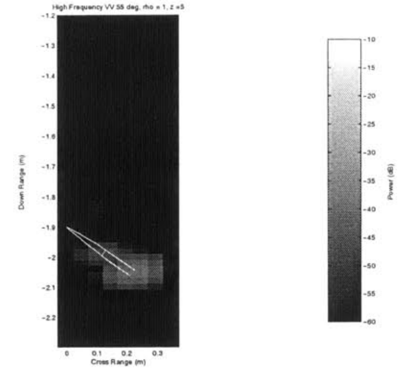

Figure 4-24: ISAR image for the high frequency technique using a 55 degree incident angle and VV polarization.

Effects of Incident Angle and Polarization

When an incident angle of 20 degrees is used, the high frequency method produces consistently higher correlation scores than when a 55 degree incident angle is used. The most important distinction between the results for 55 degrees and for 20 degrees is that much less extended return is seen at 20 degrees. At this angle, the radar mostly sees the bottom back wall of the cavity and there is minimal interaction with the side walls. As mentioned earlier, each interaction with the side walls of the cavity interior makes the incident wave less ray-like and more spread out. The ray-tracing component of the high frequency technique becomes less accurate. Reducing the number of reflections off of the side walls will increase the accuracy of the high frequency technique. This fact explains the improved predictions for simulations where the incident angle was 20 degrees. The impact incident angle has on accuracy is shown in Table 4.2.3 which gives the scores of a 2λ wide and 0.5λ deep cavity with a VV polarized incident wave.

	Incident Angle in Degrees					
	90	70	55	35	20	0
Correlation	0.8231	0.6438	0.63721	0.6589	0.69320	0.7239

Table 4.11: Summary of the performance of the high frequency method for various incident angles for the straight cavity.

Polarization did not affect the accuracy. There were no significant differences between the scores for the two different polarizations, and no general trends were found.

4.3 Electromagnetic Behavior in Outward Flared Cavities

The outward flared cavity has sloping sides so that the radius of the back wall is smaller than the radius of the opening. For all simulations, the same cavity geometry was used: 5λ deep, 2λ wide at the opening, and 1λ wide at the bottom back wall.

4.3.1 Extended Return

Figure 4-25 shows the prediction of conventional FD-TD for the outward flared cavity. Note that the amount of extended return is minimal: a compact area of reflectivity instead of a long tail. This is in marked contrast with Figure 4-11, introduced earlier, which was generated by a straight cavity with an equally wide opening and same depth.

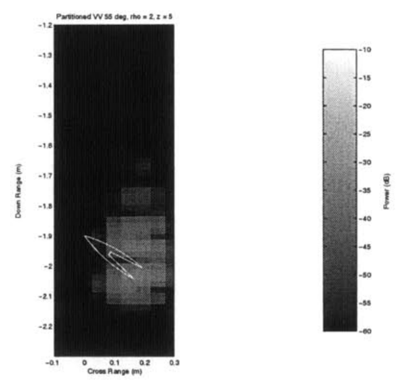

Figure 4-25: ISAR image for conventional FD-TD using a 55 degree incident angle and VV polarization.

Figure 4-26 shows the ISAR image for the RCS of the outward flared cavity as generated by the high frequency technique. This prediction lacks some of the extended return shown in the conventional FD-TD prediction. However, this cavity is still rather deep. One might note that this prediction is more accurate than it was for the 2λ wide and 5λ deep straight cavity presented earlier.

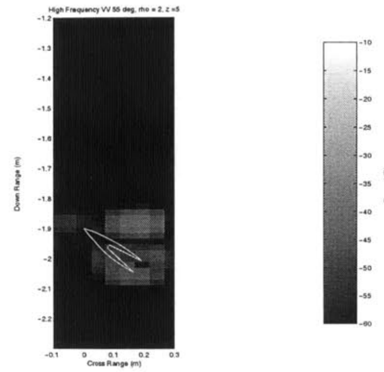

Figure 4-26: ISAR image for a high frequency technique using a 55 degree incident angle and VV polarization.

The outward flared cavity was also modeled by MR FD-TD. The ISAR image of the results are shown in Figure 4-27. MR FD-TD met expectations by giving a suitably accurate prediction.

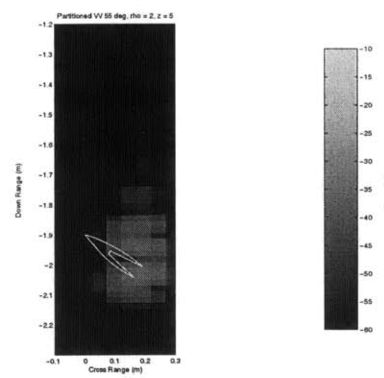

Figure 4-27: ISAR image for the MR FD-TD technique using a 55 degree incident angle and VV polarization.

The extended return of the outward flared cavity is much less than the straight cavity with the same size and depth. This effect occurs since the sloping allows energy to escape more readily as shown in part (b) of Figure 4-28. The sloping creates fewer interactions with the cavity side walls. From the previous conclusions about the relationship between the accuracy of the high frequency technique and extended return, one would expect the high frequency technique to be more accurate for cavities with more flaring. In the examples previous presented, the side walls were angled at about 5.7 degrees from horizontal. If the size of the back wall is reduced to a point–making the cavity interior into a cone–the angle is about 11.6 degrees. As shown in Table 4.12, the high frequency technique becomes more accurate. To make the angle of the flaring any larger would require shortening the cavity. Thus the increased accuracy of the high frequency technique must be attributed to both the flaring and the shallowness of the cavity.

	Angle of Flaring				
	0 degrees	5.7 degrees	11.6 degrees	20.0 degrees	31.3 degrees
Correlation	0.2430	0.4083	0.5620	0.6771	0.7731

Table 4.12: Summary of the performance of the high frequency method for various angles of flaring of interior cavity walls.

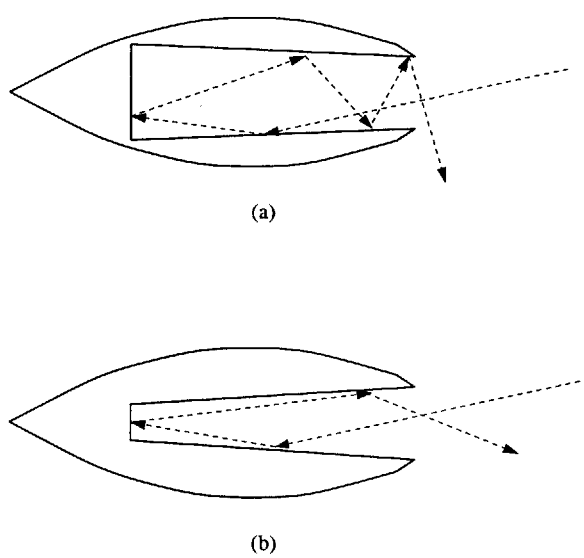

Figure 4-28: Diagram of ray-tracing for inward (a) and outward (b) flared cavities.

4.4 Electromagnetic Behavior in Inward Flared Cavities

The inward flared cavity has sloping sides so that the radius of the back wall at the bottom of the cavity is larger than the radius of the opening. For all simulations, the same cavity geometry was used: 5λ deep, 3λ wide at the bottom, and 2λ wide at the opening.

4.4.1 Extended Return

Figure 4-29 shows the prediction of conventional FD-TD for the inward flared cavity. Note that the amount of extended return is considerable. This extended return is longer than the extended return created by the straight cavity with the same cavity depth and width at the opening.

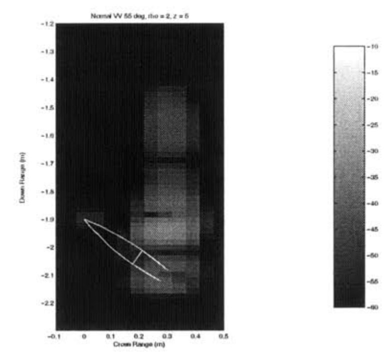

Figure 4-29: ISAR image for conventional FD-TD using a 55 degree incident angle and VV polarization.

Figure 4-30 shows the ISAR image for the inward flared cavity as generated by the high frequency technique. This method incorrectly predicts the return from the interior of the cavity as a single pulse. Furthermore, the correlation score is 0.2139, making the high frequency technique even less accurate than it was for the straight cavity of the same depth and opening width. Given the previous discussion on the inability of the high frequency technique to correctly predict long "tails" of extended return, this finding was expected.

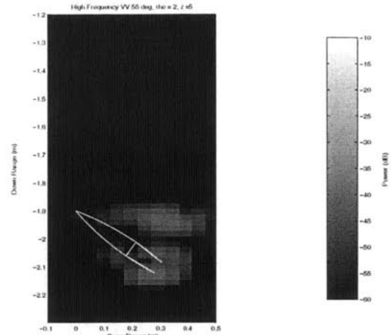

Figure 4-30: ISAR image for a high frequency technique using a 55 degree incident angle and VV polarization.

The inward flared cavity was also modeled by MR FD-TD. The ISAR image of the results are shown in Figure 4-31. Although the MR FD-TD results are somewhat comparable to the conventional FD-TD results, MR FD-TD was not able to capture a bit of extra extended return at the very end. This held true despite the use of extra iterations.

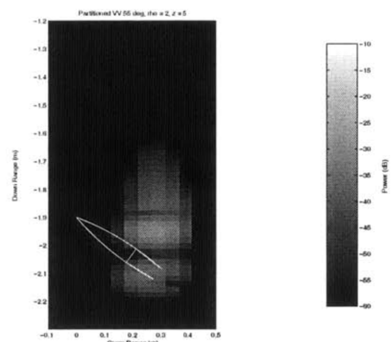

Figure 4-31: ISAR image for the MR FD-TD technique using a 55 degree incident angle and VV polarization.

The increased level of extended return in the inward flared makes sense since the sloping does not allows energy to escape readily as shown in part (a) of Figure 4-28. Energy has a tendency to remain trapped inside for a longer duration of time, thus creating more extended return. From the previous conclusions about the relationship between the accuracy of the high frequency technique and extended return, it should be expected that the high frequency approach would not provide an adequate prediction.

4.5 Electromagnetic Behavior in Cavities with Interior Features

The interior features of this cavity consist of a "bump" that protrudes out from the side wall at the half-way point between the opening and the cavity bottom. Since this cavity is a body of revolution, the bump translates into a ridge. The cavity interior in 2λ deep and 2λ wide at the opening.

4.5.1 Extended Return

Figure 4-32 shows the prediction of conventional FD-TD for the cavity with an interior feature. Note that the extended return appears as a bright spot much further down range from the other activity. The return from the straight cavity with the same sized depth and width did not have this extra pulse.

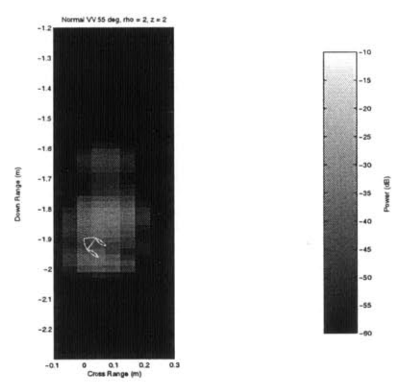

Figure 4-32: ISAR image for conventional FD-TD using a 55 degree incident angle and VV polarization.

Figure 4-33 shows the ISAR image for the cavity with an interior feature as generated by the high frequency technique. This method incorrectly predicts the return from the interior of the cavity. For this type of cavity, the high frequency technique is unable to predict the extra single pulse that emerges from the cavity after a delay. Though the high frequency method had previously been able to be fairly accurate for shallow cavities, all those cavities has featureless interior walls.

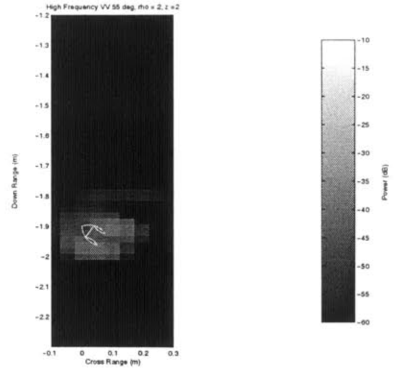

Figure 4-33: ISAR image for a high frequency technique using a 55 degree incident angle and VV polarization.

The cavity with an interior feature was also modeled by MR FD-TD. The ISAR image of the results are shown in Figure 4-34.

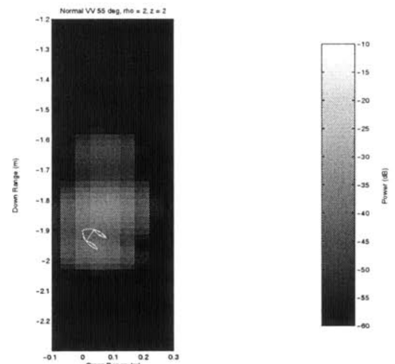

Figure 4-34: ISAR image for the MR FD-TD technique using a 55 degree incident angle and VV polarization.

MR FD-TD seems capable of correctly predicting the extended return, showing a single short pulse down range from all the other activity.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

This thesis investigated the possibility of applying a multiple region FD-TD approach to predict RCS for large, duct-like cavities. Furthermore, it sought to establish some understanding of the situations when this method is valid, and how it compares to other modeling approaches.

To gain that insight, it was necessary to understand how cavity signature in general was affected by the target geometry and relative angle and polarization of the radar antenna.

5.1.1 Range Validity of the Multiple Region Method

Multiple region FD-TD has been shown to be a comparable alternative for conventional FD-TD, provided that the cavity is 1λ or wider. In particular, the extended return predicted by conventional FD-TD is modeled accurately by the MR FD-TD given that this criterion is met. The range of validity is still limited by computational time. However, the overall range of cavity sizes where MR FD-TD is a tractable approach is larger than the range of conventional FD-TD. The angle of incidence, if smaller than about 70 degrees, does not affect the validity of MR FD-TD. Polarization has no impact.

5.1.2 Computational Savings

Multiple region FD-TD provides considerable computational savings over conventional unpartitioned FD-TD. These savings are summarized in Table 5.1. Mainly, partitioned FD-TD uses much less memory than conventional FD-TD. Where conventional FD-TD would require M amount of memory, partitioned FD-TD requires M/N, where N is the number of segments into which the cavity is divided. Using a larger number of segments allows for additional memory savings. But as shown in the last chapter, the level of accuracy generally decreased with an increase in the number of partitions.

This memory savings is advantageous because of many benefits. First, it allows the program to be run on machines that otherwise would not be able to support such a program. As mentioned in the prior chapter, programs using the conventional FD-TD approach often would not compile for larger cavities due a lack of memory.

Furthermore, as indicated in Table 5.1, the multiple region FD-TD approach allows for faster simulation times. Invoking virtual memory can be prevented because the memory demands of partitioned FD-TD can be reduced in most situations. This will avoid the slowness associated with continuously paging to virtual memory.

Another way multiple region FD-TD can decrease simulation time is through the use of parallel processing by calculating each segment on different machines. Lastly, if the cavity is very long, the FD-TD calculations only need to be carried out for the segments where there is activity. This reduces the overall number of calculations, thus reducing computational time. Therefore, the estimated time is $\leq T$ for MR FD-TD, when compared to T for conventional FD-TD. These savings may not hold true for smaller cavities that have a good deal of back and forth scattering in their interiors. Such cavities require the use of multiple iterations, causing extra calculations to be carried out. The time needed to do the extra calculations may outweigh any advantages of partitioning unless the cavity is very large and would otherwise require the use of virtual memory when modeled in conventional FD-TD.

	Conventional FD-TD	Partitioned FD-TD
Memory	M	M/N
Time	T	$\leq T$

Table 5.1: Summary of the savings of multiple region FD-TD over conventional unpartitioned FD-TD.

5.1.3 Range Validity of the High Frequency Technique

The high frequency method can produce reasonably accurate results for shallow straight or outward flared cavities that lack interior features. Cavities are considered shallow if the width of the opening is greater than the depth. Cavities that are deeper than they are wide create too much extended return. The high frequency technique has difficulty modeling this extended return. The high frequency technique is ill-suited for modeling cavities with interior features, despite the fact that they may not create long "tails" of extended return. And lastly, the high frequency method is not accurate for cavities with small openings at 1λ or less.

5.1.4 Range Feasibility of Conventional FD-TD

It has been shown that conventional FD-TD is a viable option for only a very limited range of cavity sizes. However, it may be the only option for cavities with extremely narrow ($\ll 1\lambda$) openings and for small cavities with lots of interior features. None of the other modeling approaches investigated in this thesis could produce comparable results for those classes of cavity geometries.

5.2 Future Work

5.2.1 Application to Different Cavity Profiles

There are an infinite number of different cavity geometries with which multiple region FD-TD can be tested, and this thesis could not explore all of them. Of great interest are cavities which can retain energy or create back and forth scattering since there are fewer approaches that correctly predict the RCS, and which are computationally feasible. Also of interest are cavities with interior features that create back and forth scattering. Back and forth scattering in particular is still somewhat difficult for MR FD-TD and the high frequency approach to model. Conventional FD-TD, in constrast, is computationally limited by cavity size.

5.2.2 Extension to Other Forms of FD-TD

The work for this thesis used the body of revolution version of FD-TD to implement the multiple region approach. However, as mentioned before, all the arguments and equations given in terms of BOR FD-TD are easily and readily adaptable to a 3D FD-TD environment.

The multiple region FD-TD program uses a staircase case approximation for targets. It may be possible to adapt the technique to FD-TD programs that use a conformal grid to better model targets. Modeling materials other than PEC and free space would only require small changes in the current update equations.

5.2.3 Comparison to Other Modeling Techniques

Though the multiple region FD-TD method has been shown to be accurate for cavity geometries where high frequency techniques fail, it may be enlightening to compare the results with other results obtained through some of the hybridized techniques to solve larger targets. It would also be interesting to compare the efficiency of MR FD-TD versus those approaches. As mentioned earlier, MR FD-TD is meant to be an addition to the tool-box of possible RCS modeling methods. But conventional FD-TD and the high frequency technique are not the only other methods in that box so these comparisons would be useful. Further information on of how all the prediction methods relate to one another would allow an analyst to choose the best possible modeling technique for a given situation.

5.2.4 Incorporation Parallel Computing

Since the cavity is modeled in a piecewise manner, multiple region FD-TD becomes a suitable candidate for distributed computing: each segment can be modeled on separate machines. These simulations can be done in tandem because as the fields at the first time step are calculated for one segment, the fields from the edge of that segment can be used to start the simulation for the neighboring segment and so forth. It is expected that parallel computing could appreciably expand the range of cavity sizes that are feasible to model with the MR FD-TD technique.

5.2.5 Supporting Other Computational Methods

Multiple region FD-TD can also be incorporated into other codes to predict RCS for cavities that have duct-like segments along their length. Efficient high frequency techniques can be used to model portions of the target while multiple region FD-TD can be applied to more problematic areas within the structure. For example, a very wide shallow cavity may lead into a narrow duct that has some very complicated termination at the end. The very wide shallow portion can be modeled with the high-frequency method. The duct portion can be modeled with MR FD-TD, and the termination can be modeled with conventional FD-TD or any other exact technique.

Appendix A

MR FD-TD FORTRAN Source Code

The MR FD-TD program models electromagnetic propagation through cavity segments and calculates the radar cross section when appropriate. The user must specify the "case" of each cavity segment and must provide the geometry of the segment. For Case 1 segments, the user must specify the desired incident wave.

A.1 Main FD-TD Algorithm

The main FD-TD algorithm contains the update equations. Furthermore, as appropriate for Cases 2, 3, and 4, it will read in data recorded from previous cavity segments to form the source. It will record data at the ends of the cavity segment for all cases.

```
************************************************************************
* BOR-FDTD CODE:                                              *
*      This programs calculates the scattering pattern of a         *
*      incident plane wave on a body of revolution.  The user         *
*      input the two dimensional shape of the BOR and incident          *
*      wave parameters.                                          *
*                                                              *
************************************************************************

      program bor_fdtd

      implicit none
      include 'common.f'
```

```
c 1/4/03 Made into a global variable since other outputs
c depend on the menu choice
c       integer menu_choice

      dbase = 'data'

 10   write(6,*)
      write(6,*) 'BOR FDTD Options'
      write(6,*) '1 = FDTD,WRITE FIELDS'
      write(6,*) '2 = RCS calculation'
      write(6,*) '3 = SEGMENTER'
      write(6,*) '4 = this space for rent'
      write(6, '(''*Enter option: '', $)')
      read(5,*) menu_choice

      write(6,*) 'Segment to the direct right of Case 3? Y=1 N=0'
      read(5,*) before3

      if (menu_choice.lt.1.OR.menu_choice.gt.4) goto 10

      if (menu_choice.eq.1) then
         call get_rcs_out_ranges(.FALSE.)
         call get_primary_input
         call init_fields
c        call init_freq
         call fdtd_loop(.FALSE.)
         if (case_id.eq.1) then
            call write_values
         end if
      else if (menu_choice.eq.2) then
         call get_rcs_out_ranges(.FALSE.)
         call get_primary_input
         call read_parms
         call read_values
         call write_out_all_parms
c next line is new
         call init_freq
         call read_phasors
         call calc_rcs
      else if (menu_choice.eq.3) then
         call get_rcs_out_ranges(.FALSE.)
         call get_primary_input
         call write_geometry
      else if (menu_choice.eq.4) then
      end if

      end

c**********************************************************************
c GET_PRIMARY_INPUT gets info from user about geomfile name, incident
c   wave, duration of simulation, out file names, etc
c**********************************************************************

      SUBROUTINE get_primary_input

      implicit none
      include 'common.f'

      integer conf_stair, totsteps, movie_test, mode_index, round,
     1     x1,x2,y1,y2, polarization
      real*8 dt_out, width, TIME_TO_DELAY, cost, sint
```

```
      real*8 theta_1,theta_2,theta_3,theta_4

C**** get Geometry and data filename
      write(6,'(''*Enter geometry file name: '',$)')
      read(5,*) fnamein

      write(6,'(''*Interior of the cavity contain features?: '',$)')
      write(6,*) '1. Cavity WITHOUT features'
      write(6,*) '2. Cavity WITH features'
      write(6,'(''*Enter your choice: '',$)')
      read(5,*) features


cBZ 8/01/02 get case_id
      write(6, '("*Enter case_id number:",$)')
      read(5,*) case_id

CBZ 10/22/02 FORCE IT TO BE AN ARTIFICIAL TIME/SPACE STEP if not
c   the first step!!

c      start_time = 1 if case_id = 1
      read(5,*) start_time

c      Needs z offset for gquad
c      default value should be zero!
      read(5,*) absolute_start
      read(5,*) absolute_end
      z_offset = absolute_start - 1


cBZ 9/23/02 get max_height
      write(6, '("*Enter the maximum rho (height) value:",$)')
      read(5,*) max_height

cBZ 8/6/03
      write(6, '("*Enter the maximum z (length) value:",$)')
      read(5,*) max_length



      write(6,'(''*Store for movie? (1=Y,2=N): '',$)')
      read(5,*) movie_test
      store_movie = movie_test.eq.1
      if (store_movie) then
        write(6,'(''*Movie header name: '',$)')
        read(5,*) mhname
        write(6,'(''*Movie file name: '',$)')
        read(5,*) mfname
        write(6,'(''*Number of time steps between each frame: '',$)')
        read(5,*) movie_step
        write(6,*) 'Field ids: er=1,ez=2,ephi=3,hr=4,hz=5,hphi=6'
        write(6,'(''*Enter id of field to store: '',$)')
        read(5,*) movie_num
        movie_type = 1
      end if

c####################
c####################
      call setup_staircase
      call setup_scat
c###########################
```

```
c#####################

c**** Calculate sigma_max so that reflections are 40 dB down
      sigma_max = 70*3/eta/40./0.434294481903/(PMLDEPTH*dz)
c      write(6,*) 'sigma_max = ',sigma_max
c      write(6,*) 'Enter sigma max'
c      read(5,*) sigma_max

      if (calc_bist) then
        write(6,'(''*Enter incident angle theta in degrees: '',$)')
        read(5,*) inc_ang
      end if

 33   write(6,'(''*Select polarization (1=HORZ,2=VERT): '',$)')
      read(5,*) polarization
      if (polarization.eq.1) then
        Ehg = 1.0
        Evg = 0.0
      else if (polarization.eq.2) then
        Ehg = 0.0
        Evg = 1.0
      else
        goto 33
      end if

 32   write(6,'(''*Enter duration of simulation (ns): '',$)')
      read(5,*) sim_duration
      if (sim_duration.lt.0.5) then
        print *,'Simulation must last longer than 0.5 ns.'
        goto 32
      end if

C**** Modulation of Gaussian Pulse (1-on 0-off)
c 50  write(6,'("*Modulate incident wave? (1=Y,0=N): ",$)')
c      read(5,*) modulate
c      if (modulate.gt.1.OR.modulate.lt.0) goto 50
c      if (modulate.eq.1) then
c        write(6,'("*Enter modulation frequency: ",$)')
c        read(5,*) modfreq
c      else
c        modfreq = -1
c      end if

C**** Convert incident angle to radians
      inc_ang=(inc_ang/180)*pi

      if (abs(inc_ang-pi).lt.tole.OR.abs(inc_ang).lt.tole) then
        mode_start = 1
        mode_end = 1
      else
c         modes = int(obj_height*2*pi*high_freq/c+1)
c Need to keep # modes constant for all segments
        modes = int(max_height*2*pi*high_freq/c+1)
        write(6,*) 'Estimated modes required: ',modes
        write(6,'(''*Enter start mode: '', $)')
        read(5,*) mode_start
        write(6,'(''*Enter end mode: '', $)')
        read(5,*) mode_end
      end if

c*********Always go with estimated number of modes
```

```
      mode_start = 0
c     mode_end = modes debugging
      mode_end = modes




      if (abs(Ehg-1).lt.tole) then
        eqset_start = 2
        eqset_end = 2
      else if (abs(Evg-1).lt.tole) then
        eqset_start = 1
        eqset_end =1
      else
        eqset_start = 1
        eqset_end = 2
      end if

C**** Standard Dev and Wave Delay Calculations
c       if (modulate.eq.0) then
c         sdev=5.0*dt_out(1)
c       else
c         sdev=(1.0/modfreq/4.0)
c       end if
c       width = sdev*sqrt(10.0)

c**** calculate pulse width to cover desired bandwidth

c**** amplitude function is exp(-2.3 (t/width)**2 )
c**** so that the amplitude function is "nonzero" for a duration of
c**** approximately 2*width seconds.

c**** width defined so that the function value is 10% of the maximum
c**** at the edge of the width, i.e. exp(-2.3) = 0.1

c**** Magnitude of Fourier transform of amplitude function is:
c**** exp(-(1/2.3) * (freq*pi*width)**2) which corresponds to a
c**** bandwidth of approximately 4.6 / (pi*width)

c**** It is therefore ideal to choose modulation frequency to be
c**** the center frequency.


      write(6,*) 'starting here'
      modulate = 1
      modfreq = (high_freq+low_freq)/2.0

      write(6,*) modfreq,modfreq

c      width = min(4.6/(pi*(max(high_freq-low_freq,0.5e9))),25*dt)
      width = 4.6/(pi*(max(high_freq-low_freq,0.5e9)))

      write(6,*) 'width=',width
      sdev = width / sqrt(2.3d0)

c      write(6,*)
c      write(6,*) 'width/dt = ', width/dt

c**** if width is not larger than 7*dt, you may want to define a smaller
c**** time step, which implies a smaller step size in order to avoid
c**** numerical dispersion effects.
```

```
c***  calculate time delay

 10  if (inc_ang.ge.(2*pi)) then
        inc_ang = inc_ang-2*pi
        goto 10
     end if

 20  if (inc_ang.lt.0) then
        inc_ang = inc_ang+2*pi
        goto 20
     end if
     cost = cos(inc_ang)
     sint = sin(inc_ang)

     print *, rcsz1, rcsz2, mheight

c     x1 = rcsz1
c     x2 = rcsz2

c     x1 = rcsz1 + z_offset
     x1 = rcsz1 - 1
     x2 = rcsz2 + zoffset - 1

     y1 = 1
     y2 = mheight

c***** determine the time delay so that wave arrives at the target at
c***** around time step 100-150.

     TIME_TO_DELAY = 100*dt_out(mode_start)
     theta_1 = atan2(y2*1.0,(x2-x1)*1.0)
     theta_2 = atan2(y2*4.0,(x2-x1)*1.0)
     theta_3 = atan2(y2*4.0,(x1-x2)*1.0)
     theta_4 = atan2(y2*1.0,(x1-x2)*1.0)

     if (inc_ang.ge.0.AND.inc_ang.lt.theta_1) then
       gd = (x2*dz*cost+0*dz*sint)/c + TIME_TO_DELAY + 2*sdev
     elseif (inc_ang.ge.theta_1.AND.inc_ang.lt.theta_2) then
       gd = (x2*dz*cost+y2*dz*sint)/c + TIME_TO_DELAY + 2*sdev
     elseif (inc_ang.ge.theta_2.AND.inc_ang.lt.theta_3) then
       gd = ((x1+x2)/2*dz*cost+y2*dz*sint)/c + TIME_TO_DELAY + 2*sdev
     elseif (inc_ang.ge.theta_3.AND.inc_ang.lt.theta_4) then
       gd = (x1*dz*cost+y2*dz*sint)/c + TIME_TO_DELAY + 2*sdev
     else
       gd = (x1*dz*cost+0*dz*sint)/c + TIME_TO_DELAY + 2*sdev
     end if

ccMake gd the same as the whole case

c     gd = 1.3297587838153371*1e-9

     totsteps = 0
     do 80 mode_index = mode_start,mode_end
       dt = dt_out(mode_index)
       totsteps = totsteps + round(sim_duration*1e-9/dt)
 80  continue

     if (store_movie) call setup_movie(totsteps)
```

```
      RETURN
      END

c***********************************************************************
c GET_RCS_OUT_RANGES gets info from user about what angles and freqs
c     to calc the RCS for.
c***********************************************************************

      SUBROUTINE get_rcs_out_ranges(skip_fd)

      implicit none
      include 'common.f'

      integer nang, fi, fi2, mono_bi
      real*8 mono_ang, ma, tempfreqlist(1:MAX_FREQS)
      logical skip_fd

      if (.NOT.skip_fd) then
        write(6,'(''*Enter lowest frequency of interest: '',$)')
        read(5,*) low_freq
        write(6,'(''*Enter highest frequency of interest: '',$)')
        read(5,*) high_freq

        if (abs(low_freq-high_freq).gt.tole) then

 10       write(6,'(''*Enter the number of frequencies: '',$)')
          read(5,*) num_freqs

          if (num_freqs.gt.MAX_FREQS) then
            write(6,*) 'Error. Number of freqs must be ',
     1           'less than ', MAX_FREQS, ' or raise ',
     2           'MAX_FREQS parmaeter'
            write(6,*)
            goto 10
          end if

          minf = 1
          maxf = num_freqs
          dfreq = (high_freq-low_freq)/(num_freqs-1.0)

          do 20 fi = minf, maxf
            freqlist(fi,1) = low_freq + dfreq*(fi-1.0)
c************* Define type as normal RCS freq
            freqlist(fi,2) = 0
            tempfreqlist(fi) = freqlist(fi,1)
 20       continue

          stepf = 1
        else
          freqlist(1,1) = low_freq
c********** Define type as normal RCS freq
          freqlist(1,2) = 0
          tempfreqlist(1) = freqlist(1,1)
          num_freqs = 1
          minf = 1
          maxf = 1
          stepf = 1
        end if
      end if

 100  write(6,*)
```

```
      write(6,*) '1. Calculate bistatic RCS vs angle for given freqs'
      write(6,*) '2. Estimate monostatic RCS vs angle for given freqs'
      write(6,'(''*Enter your choice: '',$)')
      read(5,*) mono_bi
      if (mono_bi.ne.1.AND.mono_bi.ne.2) then
        goto 100
      else
        calc_bist = mono_bi.eq.1
      end if

      if (calc_bist) then
        write(6,*) 'Bistatic RCS angles (in degrees)'
        write(6,'(''*Enter initial and final phi: '',$,$)')
        read(5,*) low_phi,high_phi

        if (abs(low_phi-high_phi).lt.tole) then
          dphi = high_phi-low_phi+1.0
        else
          write(6,'(''*Enter number of angles: '',$)')
          read(5,*) nang
          dphi = (high_phi-low_phi)/ dble(nang-1.0)
        end if

        write(6,'(''*Enter initial and final theta: '',$,$)')
        read(5,*) low_theta,high_theta

        if (abs(low_theta-high_theta).lt.tole) then
          dtheta = high_theta-low_theta+1.0
        else
          write(6,'(''*Enter number of angles: '',$)')
          read(5,*) nang
          dtheta = (high_theta-low_theta)/ dble(nang-1.0)
        end if
      else
        write(6,'(''*Enter incident angle theta in degrees: '',$)')
        read(5,*) inc_ang

        write(6,*) 'Monostatic RCS angles (in degrees)'
        write(6,'(''*Enter fixed phi angle: '',$)')
        read(5,*) low_phi
        high_phi = low_phi
        dphi = 1.0

        write(6,*) 'Note, monostatic angle range = inc_ang (+/-) ',
     1       'max_ang'
        write(6,'(''*Enter max angle: '',$,$)')
        read(5,*) mono_ang
        mono_ang = abs(mono_ang)
        low_theta = inc_ang-mono_ang
        high_theta = inc_ang+mono_ang

        if (abs(low_theta-high_theta).lt.tole) then
          dtheta = high_theta-low_theta+1.0
        else
          write(6,'(''*Enter number of angles (must be odd): '',$)')
          read(5,*) nang
          if (dble(nang/2).eq.dble(nang)/2.0) then
            write(6,*) 'Increasing nang to ', nang+1
            nang = nang+1
          end if
          dtheta = (high_theta-low_theta)/ dble(nang-1.0)
```

```
      end if
      mono_nang = nang
c******* Determine freqs that need to be calculated.

c******* Total frequencies needed num_freqs*(nang+1)/2
      if (num_freqs*(nang+1)/2.gt.MAX_FREQS) then
        write(6,*) 'MAX_FREQS error'
        pause
      end if

      fi2 = 1
      do 30 fi=1,num_freqs
c********** Update freqlist components so that they are considered
c********** for use in monostatic calculations
        mono_freq_ind(fi) = fi2
        do 40 ma = 0,mono_ang,dtheta
          freqlist(fi2,1) = tempfreqlist(fi)*cos(ma/180*pi)
          freqlist(fi2,2) = 1
          fi2 = fi2+1
40      continue
30    continue
      minf = 1
      maxf = num_freqs*(nang+1)/2
    end if


    RETURN
    END


c**********************************************************************
c MEMORY_CHECK checks if enough memory has been allocated and reports
c   all errors stored in error buffer.
c**********************************************************************

    SUBROUTINE memory_check

    implicit none
    include 'common.f'

    integer i, id

    if ((2*mheight+rcsz2-rcsz1-1).gt.mxdp) then
      print *,'error not enough memory for RCS components'
      print *,'set the parameter mxdp higher than',
   1       2*mheight+rcsz2-rcsz1-1
      enough_memory = .FALSE.
    end if

    if (nm.gt.mode_start) then
      write(6,*)
      print *,'nm =',nm,' is greater than the starting mode'
      print *,'number', mode_start, '. Adjust the nm parameter'
      enough_memory = .FALSE.
    end if

    if (mm.lt.mode_end) then
      write(6,*)
      print *,'mm =',mm,' is less than the ending mode'
      print *,'number', mode_end, '. Adjust the mm parameter'
      enough_memory = .FALSE.
```

```fortran
      end if

      if (errorcount.gt.0) then
         write(6,*) '********************************************'
         write(6,*) 'Insufficient memory to begin simulation. The'
         write(6,*) 'following parameter(s) in the common.f file'
         write(6,*) 'need to be adjusted:'

         do 10 i=1,errorcount
            id = errors(i)
            write(6,*)
            if (id.eq.NODE_ERROR) then
               write(6,*) 'Set MAX_NODES to at least',total_nodes
            else if (id.eq.MAX_Z_ERROR) then
               write(6,*) 'Set MAX_Z_CELLS to at least',maxz
            else if (id.eq.MAX_R_ERROR) then
               write(6,*) 'Set MAX_R_CELLS to at least',maxr
            else if (id.eq.MAX_STAIR_ERROR) then
               write(6,*) 'Set MAX_STAIR_NODES to at least',
     1              stair_node_count
            else if (id.eq.MAX_RCS_ERROR) then
               write(6,*) 'Set MAX_RCS_NODES to at least',
     1              2*mheight+(rcsz1-rcsz2)
            end if

 10      continue
         write(6,*) '********************************************'
         stop

      end if

      RETURN
      END

c***********************************************************************
c WRITE_OUT_ALL_PARMS outputs to a file all important parameters used
c    in running the simulation
c***********************************************************************

      SUBROUTINE write_out_all_parms

      implicit none
      include 'common.f'

      integer totsteps, mode_index, round, i, j
      real*8 dt_out

      open(unit=9,file='bor.out',status='unknown',form='formatted')

 89   format('Scat. field end points (',I4,',',I4,'),(',I4,',',I4,')')
      write(9,89) xscat_sp,1,maxz-xscat_sp,int(obj_height/dz)+ytot_sp

cBZ 9/30/02 ################
 99   format('mxr = ',F8.4,' obj_height = ',F8.4)
      write(9,99) mxr,obj_height

 100  format('max_height = ',F8.4,' dz = ',F8.4)
      write(9,100) max_height, dz

C     *** WRITE TO fnameout ***
```

```
write(9,11)
write(9,17) (high_freq/1E9), (low_freq/1E9)
write(9,11)
if (modulate.eq.1) then
  write(9,31) modfreq/1.0E9
else
  write(9,31) -1
end if
write(9,11)
write(9,26) len,obj_height
write(9,18) maxz,maxr,dz
write(9,11)
write(9,19) sigma_max
write(9,21) movie_num
write(9,22) movie_type
write(9,23) mheight,rcsz1
write(9,24) rcsz2, 2*mheight+rcsz2-rcsz1-1

write(9,*)
write(9,*) '         sdev =  ',sdev
write(9,*) '      inc_ang =  ',inc_ang/pi*180,' (deg)'
write(9,*) '           gd =  ',gd,' (sec)'
write(9,*)

write(9,*) ' Simulation Duration (ns) = ',sim_duration

totsteps = 0
do 80 mode_index = mode_start,mode_end
  dt = dt_out(mode_index)
  N = round(sim_duration*1e-9/dt)
  totsteps = totsteps + N
  write(9,36) mode_index,dt,N
80 continue
36 format(2X,'Mode = ',I2,2X'dt = ',E12.7,2X,'N time steps =',I6)
write(9,*) ' Total Steps to run = ',totsteps

write(9,11)
write(9,34) eqset_start, eqset_end
34 format(2X,'Running eqset ',I1,' through ',I1)
write(9,*)
if (abs(Ehg-1.0).lt.tole) then
  write(9,*) 'HH RCS calculated'
else
  write(9,*) 'VV RCS calculated'
end if

if (calc_bist) then
  write(9,*) 'Bistatic RCS calculated'
else
  write(9,*) 'Estimated Monostatic RCS calculated'
end if

write(9,11)
if (.NOT.use_conformal) then
  write(9,*) ' Staircase gridding used for ',fnamein,' geomfile'
else
  write(9,*) ' Conformal gridding used for ', fnamein, 'geomfile'
end if
write(9,25)

write(9,27) xtot_sp, ytot_sp
```

```
      write(9,28) xscat_sp, yscat_sp
      write(9,29) xhuy_sp, yhuy_sp
      write(9,30) xall_sp, yall_sp

 27   format('      xtot_sp = ',I8,'        ytot_sp = ',I8)
 28   format('     xscat_sp = ',I8,'       yscat_sp = ',I8)
 29   format('      xhuy_sp = ',I8,'        yhuy_sp = ',I8)
 30   format('      xall_sp = ',I8,'        yall_sp = ',I8)

      write(9,*) 'case_id = ', case_id
      write(9,*) 'features = ', features
      write(9,*) 'mode_no = ', mode_no
      write(9,*) 'end_playback = ', end_playback
      write(9,*) 'flag = ', flag
      write(9,*) 'quit_flag = ', quit_flag
      write(9,*) 'before3 = ', before3
      write(9,*) 'start_time ', start_time
      write(9,*) 'end_time = ', end_time
      write(9,*) 'start_mem_rec = ', start_mem_rec
      write(9,*) 'er_max = ', er_max
      write(9,*) 'er_mem = ', er_mem
      write(9,*) 'max_height = ', max_height
      write(9,*) 'mxr = ', mxr
      write(9,*) 'z_offset = ', z_offset
      write(9,*) 'absolute_start = ', absolute_start
      write(9,*) 'absolute_end = ', absolute_end
      write(9,*) 'rcsz_start = ', rcsz_start
      write(9,*) 'rcsz_end = ', rcsz_end
      write(9,*) 'rcsz = ', rcsz
      write(9,*) 'rcsr = ', rcsr
      write(9,*) 'x_start_tot = ', x_start_tot
      write(9,*) 'x_end_tot = ', x_end_tot
      write(9,*) 'upper_edgetot = ', upper_edgetot
      write(9,*) 'upper_edgescat = ', upper_edgescat
      write(9,*) 'upper_edgehuy = ', upper_edgehuy
      write(9,*) 'lower_edgetot = ', lower_edgetot
      write(9,*) 'lower_edgescat = ', lower_edgescat
      write(9,*) 'lower_edgeleft = ', lower_edgeleft
      write(9,*) 'lower_edgeright = ', lower_edgeright
      write(9,*) 'x_opening = ', x_opening
      write(9,*) 'y_opening = ', y_opening
      write(9,*) 'right_x = ', right_x
      write(9,*) 'right_y = ', right_y
      write(9,*) 'left_x = ', left_x
      write(9,*) 'left_y = ', left_y
      write(9,*) 'high_y = ', high_y
      write(9,*) 'high_x = ', high_x
      write(9,*) 'chuck = ', chuck
      write(9,*) 'max_length = ', max_length
      write(9,*) 'maxztrue = ', maxztrue
      write(9,*) 'zoffset = ', zoffset

      call plotb(ZB,RB,NP,51,41)

      do 82 i = 1, staircount
c        do 83 j = 1,3
         write(9,*) stair_zero(i,1),stair_zero(i,2),stair_zero(i,3)
c 83     continue
 82   continue
```

```
      close(unit=9)

C     *** FORMAT LINES ***

 09   format(I3)
 11   format('')
 17   format('High Freq (GHz) =',F6.2,3X,'Low Freq (GHz) = ',F6.2)
 31   format('Modulation Freq (GHz) (-1 = unmodulated)',F6.2)
 26   format(5X,'Length (m) = ',F4.2,4X,'Height (m) = ',F4.2)
 18   format(11X,'maxz = ',I5, 9X,'maxr = ',I4,9X,'dz = ',F8.7,' (m)')
 19   format('      sigma_max = ',F12.8)
 21   format('      movie_num = ',I12,'    (er=1, ez=2, ephi=3, hr=4, '
     1    ,'hz=5, hphi=6)')
 22   format('     movie_type = ',I12,'     (movie=1, wrtraw=2)')
 23   format('        mheight = ',I12,'        rcsz1 = ',I8)
 24   format('          rcsz2 = ',I12,'      NPInRCS = ',I8)

 25   format(/,'ADJUSTED DATA POINTS TO FIT FDTD GRID')



      RETURN
      END



c***********************************************************************
c WRITE_GEOMETRY: outputs to a file the important z-values per segment
c***********************************************************************

      SUBROUTINE write_geometry
      implicit none
      include 'common.f'
      open(unit=9,file='geom.info',status='unknown',form='formatted')
      write(9,*) rcsz_start
      write(9,*) rcsz_end
      write(9,*) maxz
      write(9,*) mheight
      close(unit=9)
      RETURN
      END



c***********************************************************************
c DT_OUT returns the required dt for stability based on mode number
c    and dz. Function used so that all dts in the program are calculated
c    in the same way.
c***********************************************************************

      REAL*8 FUNCTION dt_out(mode)

      implicit none
      include 'common.f'

      integer mode
```

```
c**** Taflove's stability criterion.
      dt_out=dz/((max(mode+1.0,1.45))*c)
      dt_out = 0.95*dt_out

c     dt_out = 0.90*dt_out

c***** Davidson stability criterion that only works for low order modes.
c     dt_out = 0.90*(dz/c)*(((mode+1.0)**2.0 + 2.8)/4 + 1.0)**(-0.5)

      RETURN
      END

c**********************************************************************
c the initialize routine
c**********************************************************************

      SUBROUTINE init_fields

      implicit none
      include 'common.f'
      integer k,i

      do 10 k=1,maxz
        do 20 i=1,maxr
          er(k,i)=0.0
          ez(k,i)=0.0
          ephi(k,i)=0.0
          hr(k,i)=0.0
          hz(k,i)=0.0
          hphi(k,i)=0.0
20      continue
10    continue

      do 30 k=1,pmldepth+1
        do 40 i=0,pmldepth+maxr+1
          erphil(k,i)=0.0
          erzl(k,i)=0.0

          ezphil(k,i)=0.0
          ezrl(k,i)=0.0

          ephirl(k,i)=0.0
          ephizl(k,i)=0.0

          hrphil(k,i)=0.0
          hrzl(k,i)=0.0

          hphirl(k,i)=0.0
          hphizl(k,i)=0.0

          hzphil(k,i)=0.0
          hzrl(k,i)=0.0
40      continue
30    continue


      do 31 k=1,pmldepth+1
        do 41 i=0,maxr+1
          erphilx(k,i)=0.0
          erzlx(k,i)=0.0
```

```
        ezphilx(k,i)=0.0
        ezrlx(k,i)=0.0

        ephirlx(k,i)=0.0
        ephizlx(k,i)=0.0

        hrphilx(k,i)=0.0
        hrzlx(k,i)=0.0

        hphirlx(k,i)=0.0
        hphizlx(k,i)=0.0

        hzphilx(k,i)=0.0
        hzrlx(k,i)=0.0
41      continue
31    continue


    do 50 k=1,pmldepth+1
      do 60 i=0,pmldepth+maxr+1
        erphir(k,i)=0.0
        erzr(k,i)=0.0

        ezrr(k,i)=0.0
        ezphir(k,i)=0.0

        ephizr(k,i)=0.0
        ephirr(k,i)=0.0

        hrphir(k,i)=0.0
        hrzr(k,i)=0.0

        hphizr(k,i)=0.0
        hphirr(k,i)=0.0

        hzphir(k,i)=0.0
        hzrr(k,i)=0.0
60      continue
50    continue

    do 90 k=1,maxz
      do 100 i=1,pmldepth+1
        erzt(k,i)=0.0
        erphit(k,i)=0.0

        ezphit(k,i)=0.0
        ezrt(k,i)=0.0

        ephizt(k,i)=0.0
        ephirt(k,i)=0.0

        hrphit(k,i)=0.0
        hrzt(k,i)=0.0

        hphirt(k,i)=0.0
        hphizt(k,i)=0.0

        hzphit(k,i)=0.0
        hzrt(k,i)=0.0
100     continue
90    continue
```

```
      return
      end

c*******************************************************************c
c FDTD Loop: loops through all time steps updating electric and
c    magnetic fields and call boundary condition routines to enforce
c    PEC BCs
c*******************************************************************c

      SUBROUTINE fdtd_loop(store_freqs)

      implicit none
      include 'common.f'

      integer k,i,m,eqset,ms,round,movie_frame

      logical store_freqs
      real*8 dt_out

c      print *,'debug 0'
c#########################
c#########################
      call memory_check
      call write_out_all_parms

c##########################
c##########################
      movie_frame = 0
      time = 0

      print *, "Starting simulation..."


cBZ****Recording membrane
      open(unit=9, file='er.out', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=10, file='ephi.out', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=11, file='hr.out', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=12, file='hphi.out', status='unknown',
     1     access='sequential', form='formatted')

cBZ****Playback membrane (will be at total field locations
c     where calculations require READ-NOT at the locations
c     where the read value is added into the vector: k+1 or k-1!)
      open(unit=13, file='er.in', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=14, file='ephi.in', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=15, file='hr.in', status='unknown',
     1     access='sequential', form='formatted')
      open(unit=16, file='hphi.in', status='unknown',
     1     access='sequential', form='formatted')

cBZ****Writes timing information
      open(unit=20, file='time.info', status='unknown',
     1     access='sequential', form='formatted')

cBZ****backscatter recording membrane for case2
```

```
      if ((case_id.eq.2).or.(case_id.eq.5).or.(case_id.eq.4)) then
         open(unit=21, file='erx.out', status='unknown',
     1        access='sequential', form='formatted')
         open(unit=22, file='ephix.out', status='unknown',
     1        access='sequential', form='formatted')
         open(unit=23, file='hrx.out', status='unknown',
     1        access='sequential', form='formatted')
         open(unit=24, file='hphix.out', status='unknown',
     1        access='sequential', form='formatted')
      end if


cBZ***Quit flags
      end_playback = 0
      quit_flag = 0
      flag = 0
      start_mem_rec = 0
      end_time = 0
      er_max = 0
      er_mem = 0


c######### Set RCS TOP breakpoint
      if ((case_id.eq.5).or.(case_id.eq.1)) then
         if ((rcsz1+17).ge.rcsz2) then
            pookie = 0
         else
            pookie = rcsz1 + 12
         end if
      end if


      do 5 m = mode_start, mode_end
         dt = dt_out(m)
         N = round(sim_duration*1e-9/dt)
         do 10 eqset=eqset_start,eqset_end
            ms=(-1)**(eqset+1)
            print *,"Mode=",m," Equation Set #",eqset

            do 20 time = start_time, N+start_time - 1
               print *,time,' of ', N+start_time-1
               do 30 k=1,maxz
                  do 40 i=1,maxr
                     call free_space_E(k,i,m,ms,use_conformal)
cBZ
c  I changed it so that start_time is the "absolute" time
c  we are working with.  start_mem_rec records the time step at
c  which we start writing to the membrane which controls the
c  NEXT start_time
 40               continue
 30            continue

               call pmlEeqn(m*ms,ms)

               if (use_conformal) then
c        call boundary_conditions(m,ms)
               else
                  call stair_boundary_conditions
               end if
```

```
cCBZ After free_space_E returns, we check flags
ccBZ Saving time-1 since on current iteration (time)
cc   we did not write.
c             if (quit_flag.eq.1) THEN
c                end_time = time - 1
ccBZ***save the ending time to the time.info file
c                write(20,*) end_time
ccBZ***Exit the loop (no need to do the H fields)
c                GO TO 21
c             END IF

cBZ####But if this is the last time step,
cBZ  and quit_flag hasn't been set, we must write
cBZ***the CURRENT time step since we did and will
c  write the field values on the membrane.
c  Then we allow the  H fields to finish
c  and let the loop end by itself.
            if  (time.EQ.N) THEN
               end_time = time
cBZ***save the ending time to the time.info file
               write(20,*) end_time
cBZ***DO NOT exit the loop
            END IF

            do 50 k=1,maxz
               do 60 i=1,maxr
                  call free_space_H(k,i,m,ms,use_conformal)
 60            continue
 50         continue

            call pmlHeqn(m*ms,ms)

            if (store_movie) then
               if (movie_frame.eq.movie_step) then
                  call movie(m,ms)
                  movie_frame = 0
               else
                  movie_frame = movie_frame + 1
               end if
            end if

            if ((case_id.eq.1).or.(case_id.eq.5)) then
               call update_dft(m, eqset)
            end if
20       continue
cBZ Outside the time loop
C and clears the data for next equation set and mode
 21        call init_fields
 10     continue
 5   continue

     if (case_id.eq.1) then
        call write_phasors
     else if (case_id.eq.5) then
        call read_phasorsx
        call write_phasors
     end if

cBZ closing. . . . .
cBZ-close recording membrane
     endfile(9)
```

```
      endfile(10)
      endfile(11)
      endfile(12)
      close(unit=9)
      close(unit=10)
      close(unit=11)
      close(unit=12)

cBZ-close playback membrane
      endfile(13)
      endfile(14)
      endfile(15)
      endfile(16)
      close(unit=13)
      close(unit=14)
      close(unit=15)
      close(unit=16)

cBZ-close timing output files
      endfile(20)
      close(unit=20)

cBZ-close backscatter backscatter recording membrane for case2
      if (case_id.eq.2) then
        endfile(21)
        endfile(22)
        endfile(23)
        endfile(24)
        close(unit=21)
        close(unit=22)
        close(unit=23)
        close(unit=24)
      end if



      return
      end


c****************************************************************c
c determines whether the grid cell (k,i) is a total or scattered
c field.
c****************************************************************c

      logical function inside(k,i)

      implicit none
      include 'common.f'
      integer k,i,t

      t=scattot(k,i)

C     *** total fields are 2-9,14-24 ***

      inside = (t.ge.2.AND.t.le.9)
      inside = (inside.OR.((t.ge.14).AND.(t.le.24)))

      return
      end
```

```
c********************************************************************c
c free_space_E contains the core update equations for calculating
c the free space E fields.
c********************************************************************c

      SUBROUTINE free_space_E(k,i,m,ms,conformal)

      implicit none
      include 'common.f'

      integer k,i,m,st,ms
      real*8 er_in,ephi_in,hr_in,hphi_in
      real*8 ephix, erx
      real*8 c1,c2,c3,c4,c5,cx
      real*8 gquad
      logical conformal

      st = scattot(k,i)

      if (i.ne.1) THEN
C **********Calculate E fields at time n+0.5

C ***************************Ez
         c1=(i+0.5-1.0)*dt/(eps*(i+0.0-1.0)*dz)
         c2=(i-0.5-1.0)*dt/(eps*(i+0.0-1.0)*dz)
         c3=(m+0.0)*dt/(eps*(i+0.0-1.0)*dz)

         c4=hphi(k,i-1)

cBZ********Top side in scattered field
         if ((st.eq.11).and.(case_id.eq.1)) then
           c4=c4-gquad(0.0,2*pi,ms*11,m,time*dt,(i-1)*dz,
     1       k*dz,inc_ang)
         end if
         ez(k,i)=ez(k,i)+(c1*hphi(k,i)-c2*c4+ms*c3*hr(k,i))/eta

C ***************************Ephi
         c1=dt/(eps*dz)
         c4=hz(k,i-1)

         if (k+1.gt.maxz) THEN
           c5=(hrzr(1,i)+hrphir(1,i))
         ELSE
           c5=hr(k+1,i)
         END IF

cBZ#####Right hand side in total field
         if (st.eq.6.OR.st.eq.7.OR.st.eq.8) then
           if (case_id.eq.1) then
cBZ****Incoming wave: FIRST
             c5 = c5+gquad(0.0,2*pi,
     1         ms*7,m,time*dt,i*dz,(k+1)*dz,
     1         inc_ang)
           else if (case_id.eq.2.OR.case_id.eq.3) then
cBZ****Incoming wave: second and third segments
             read (15,*, END=10) hr_in
             c5 = c5 + hr_in
           end if
10       end if
```

```
cBZ#####Left hand side in scattered field
c     Incoming wave
      if ((st.eq.1).and.(case_id.eq.1)) then
        c5 = c5 - gquad(0.0,2*pi,ms*7,m,time*dt,i*dz,
     1        (k+1)*dz,inc_ang)
      end if

      if ((st.eq.1).AND.((case_id.eq.5).or.(case_id.eq.4))) THEN
        read(15,*) hr_in
        c5 = c5 - hr_in
      END IF


cBZ#####Top side in scattered field
      if ((st.eq.11).and.(case_id.eq.1)) then
        c4=c4-gquad(0.0,2*pi,ms*9,m,time*dt,(i-1)*
     1    dz, k*dz,inc_ang)
      end if
      ephi(k,i)= ephi(k,i)+(c1*(c4-hz(k,i)+c5-hr(k,i)))/eta

cBZ...3/26/03...Shouldn't be necessary but let's do this as a test
      if ((case_id.eq.3).and.(k.eq.1)) then
        ephi(k,i) = 0
      end if


C ****************************Er
c ****************************

      if (k.eq.maxz) THEN
        c5=hphizr(1,i)+hphirr(1,i)
      ELSE
        c5=hphi(k+1,i)
      END IF

cBZ*****Right hand side in total field
c     Incoming wave
      if (st.eq.6.OR.st.eq.7.OR.st.eq.8) THEN
        if (case_id.eq.1) c5 = c5 + gquad(0.0,2*pi,ms*11,m,
     1        time*dt,i*dz,(k+1)*dz, inc_ang)

        if ((case_id.eq.2).OR.(case_id.eq.3)) THEN
          read(16,*,END=11) hphi_in
          c5 = c5 + hphi_in
        end if
 11   END IF


cBZ*****Left hand side scattered field
c***  Incoming wave
      if ((st.eq.1).AND.(case_id.eq.1)) THEN
        c5=c5-gquad(0.0,2*pi,ms*11,m,time*dt,i*dz,
     1        (k+1)*dz,inc_ang)
      END IF

cBZ*****Left hand side scattered field
c***  Incoming wave
      if ((st.eq.1).AND.((case_id.eq.5).or.(case_id.eq.4))) THEN
        read(16,*) hphi_in
        c5=c5 - hphi_in
```

```
        END IF


        c1=dt/(eps*dz)
        c2=(m*dt/eps)/((i+0.5-1.0)*dz)
        er(k,i)=er(k,i)+(c1*(hphi(k,i)-c5)-ms*c2*hz(k,i))/eta


cBZ...3/26/03...Shouldn't be necessary but let's do this as a test
        if ((case_id.eq.3).and.(k.eq.1)) then
          er(k,i) = 0
        end if

      ELSE
c######################################################################
C ********************************************************************* C
C ****************************On Axis Equations*********************** C
C ********************************************************************* C

C ****************************Ez
        c1=4*dt/(eps*dz)
        ez(k,i)=ez(k,i)+(c1*hphi(k,i))/eta
C*****************If the Fourier mode is not 0 ez(k,1) is zero.
cBZ*****Hmmmm....don't seem to worry about adding in incidents???
        if (m.ne.0) ez(k,i)=0.0

C ****************************Ephi

        c1=2*dt/(eps*dz)
        c2=dt/(eps*dz)
        if (k.eq.maxz) THEN
          c5=hrzr(1,i)+hrphir(1,i)
        ELSE
          c5=hr(k+1,i)
        END IF

cBZ########Right side total field, incoming FIRST CASE
        if ((st.eq.8).and.(case_id.eq.1)) THEN
          c5=c5+gquad(0.0,2*pi,ms*7,m,time*dt,i*dz,
     1         (k+1)*dz,inc_ang)
        end if

cBZ#######Right side total field, incoming
        if ((st.eq.8).and.((case_id.eq.2).OR.(case_id.eq.3))) THEN
          read(15,*, END=12) hr_in
          c5 = c5 + hr_in
 12     end if

cBZ########Left side scattered field, incoming
        if ((st.eq.1).and.(case_id.eq.1)) THEN
          c5=c5-gquad(0.0,2*pi,ms*7,m,time*dt,i*dz,
     1         (k+1)*dz,inc_ang)
        end if

        if ((st.eq.1).AND.((case_id.eq.5).or.(case_id.eq.4))) THEN
          read(15,*) hr_in
          c5 = c5 - hr_in
        END IF


        ephi(k,i) = ephi(k,i)+(-c1*hz(k,i)+c2*(c5-hr(k,i)))/eta
```

```
c     ******If the fourier mode !=1 then ephi(k,1) and hr(k,1) = zero
        if (m.ne.1) THEN
          ephi(k,i)=0.0
        END IF


cBZ...3/26/03...Shouldn't be necessary but let's do this as a test
        if ((case_id.eq.3).and.(k.eq.1)) then
          ephi(k,i) = 0
        end if



C ***************************Er
        if (k.eq.maxz) THEN
          c5=hphizr(1,i)+hphirr(1,i)
        ELSE
          c5=hphi(k+1,i)
        END IF

cBZ########Right side total field, incoming FIRST CASE
cBZ changed
        if ((st.eq.8).and.(case_id.eq.1)) THEN
          c5=c5+gquad(0.0,2*pi,ms*11,m,time*dt,i*dz,
     1       (k+1)*dz,inc_ang)
        end if


cBZ#######Right side total field, incoming
cBZ changed
        if ((st.eq.8).and.((case_id.eq.2).OR.
     1       (case_id.eq.3))) THEN
          read(16,*, END=13) hphi_in
          c5 = c5 + hphi_in
 13     end if

cBZ########Left side scattered field, incoming
        if ((st.eq.1).and.(case_id.eq.1)) THEN
          c5=c5-gquad(0.0,2*pi,ms*11,m,time*dt,i*dz,
     1       (k+1)*dz,inc_ang)
        end if

        if ((st.eq.1).AND.((case_id.eq.5).or.(case_id.eq.4))) THEN
          read(16,*) hphi_in
          c5=c5 - hphi_in
        END IF


        c1=c1/2.0
        c2=(m*dt/eps)/((i+0.5-1.0)*dz)
        er(k,i)=er(k,i)+(c1*(hphi(k,i)-c5)-ms*c2*hz(k,i))/eta

cBZ...3/26/03...Shouldn't be necessary but let's do this as a test
        if ((case_id.eq.3).and.(k.eq.1)) then
          er(k,i) = 0
        end if

      END IF
c end test of axis eq
```

```
ccDEBUG write out fields in case 3 where fields are added in!!!!
c      if ((case_id.eq.3).and.((st.eq.7).or.(st.eq.8))) then
c         write(25,*) er(k,i)
c      end if
cc

cBZ****writing to and closing output files
c IF AT A RECORDING CELL!!!!....
c Hmmm....case 5 should not have a recording cell...
      if ((((case_id.eq.1).and.((st.eq.22).or.(st.eq.16)))
     1     .or.((case_id.eq.1).and.((st.eq.23).or.(st.eq.24)))
     1     .or.((case_id.eq.2).and.((st.eq.22).or.(st.eq.16)))
     1     .or.((case_id.eq.2).and.((st.eq.23).or.(st.eq.24)))
     1     .or.((case_id.eq.2).and.(st.eq.12))
     1     .or.((case_id.eq.2).and.(st.eq.25))
     1     .or.((case_id.eq.3).and.(st.eq.12))
     1     .or.((case_id.eq.3).and.(st.eq.25))
     1     .or.((case_id.eq.4).and.((st.eq.17).or.(st.eq.18)))
     1     .or.((case_id.eq.4).and.((st.eq.26).or.(st.eq.27)))
     1     .or.((case_id.eq.4).and.(st.eq.23))
     1     .or.((case_id.eq.5).and.(st.eq.23)))
     1     .AND.(quit_flag.eq.0))
     1     THEN

c###########Controls the Start and Stop of Recording##########
c###############################################################

ccBZ Set flag if non-zero field
c          IF ((flag.EQ.0).AND.
c     1  (((abs(er(k,i))+abs(ephi(k,i)) + abs(ez(k,i))
c     1        + abs(hr(k,i))+ abs(hphi(k,i))+ abs(hz(k,i))).GT.0.0).OR.
c     1        (time.GE.(start_time+0)))) THEN
c             flag = 1
c             start_mem_rec = time
cc So we note the starting time in the time.info file
c             write(20,*) start_mem_rec
c          END IF

c          IF (flag.EQ.1) THEN
ccBZ If we have started recording,
ccBZ we check for maximum value
c                If (er_max.LT.ABS(er(k,i))) THEN
c                   er_max = ABS(er(k,i))
c                END IF
ccBZ And takes time average of the field
c                er_mem = ABS((er_mem*(time - start_mem_rec) +
c     1                   ABS(er(k,i)))/((time - start_mem_rec)+1))
c          END IF

ccBZ This will stop the recording and simulation for this mode
cc      once the fields get low.  CURRENT values will not be
cc      recorded!
c          If ((flag.EQ.1).AND.(er_mem.LT.(er_max *0)).AND.
c     1        ((start_mem_rec + 58).LT.time)) THEN
c             quit_flag = 1
c          END IF


CCCCFor now we start at 1 and end at N
cBZ Set flag if non-zero field
```

```
      IF ((flag.EQ.0).AND.
     1    (time.EQ.1)) THEN
        flag = 1
        start_mem_rec = time
c So we note the starting time in the time.info file
        write(20,*) start_mem_rec
      END IF


c#############################################################
c#############################################################

cBZ**** We have begun recording......
      if ((flag.eq.1).AND.(quit_flag.eq.0)) THEN

        if ((case_id.eq.2).and.(st.eq.12)) then
c Backscatter for case 2
          write(21,*) er(k,i)
          write(22,*) ephi(k,i)

        else if (((case_id.eq.1).or.(case_id.eq.2))
     1         .and.((st.eq.23).or.(st.eq.24))) THEN
c Forward scatter for case 1,2,
          write(10,*) ephi(k,i)
          write(9,*) er(k,i)

        else if ((case_id.eq.3).and.(st.eq.12)) then
c "Forward scatter" for case 3
          write(10,*) ephi(k,i)
          write(9,*) er(k,i)

        else if ((case_id.eq.4).and.
     1         ((st.eq.17).or.(st.eq.18))) THEN
c "Forward scatter" for case 4
          write(10,*) ephi(k,i)
          write(9,*) er(k,i)

        else if ((case_id.eq.4).and.
     1         (st.eq.23)) THEN
c Backscatter for case 4
          write(21,*) er(k,i)
          write(22,*) ephi(k,i)

        else if ((case_id.eq.5).and.
     1         (st.eq.23)) THEN
c Backscatter for case 5
          write(21,*) er(k,i)
          write(22,*) ephi(k,i)
        end if

      END IF
    END IF


    return
    end

c********************************************************************c
c free_space_H contains the core update equations for calculating
c the free space H fields.
c********************************************************************c
```

```
      SUBROUTINE free_space_H(k,i,m,ms,conformal)

      implicit none
      include 'common.f'

      integer k,i,m,st,ms
      real*8 er_in,ephi_in,hr_in,hphi_in
      real*8 erx, ephix
      real*8 c1,c2,c3,c4,c5,gquad
      logical conformal

      st=scattot(k,i)


      if (i.ne.1) THEN

C ***************************Hr
        c1=dt/(mu*dz)
        c2=(m*dt)/(mu*(i+0.0-1.0)*dz)


        IF (k.eq.1) THEN
          if (case_id.ne.3) then
            if (((case_id.eq.1).or.(case_id.eq.5))
     1           .and.(i.lt.left_y)) then
              c5=ephizlx(1,i)+ephirlx(1,i)
            else
              c5=ephizl(1,i)+ephirl(1,i)
            end if
          else
            c5=0
          end if
        ELSE
          c5=ephi(k-1,i)
        END IF


cBZ-use these eq only!!!
cBZ---left total field
        if ((st.eq.2.OR.st.eq.3).and.
     1     ((case_id.eq.4).or.(case_id.eq.5))) THEN
          read(14,*,END=14) ephi_in
          c5 = c5 + ephi_in
 14     END IF
c BZ added below 3/24/03
        if ((st.eq.3.OR.st.eq.4).and.
     1     (case_id.eq.1)) THEN
          c5=c5+gquad(0.0,2*pi,
     1       ms*2,m,time*dt,i*dz,
     1       (k-1)*dz,inc_ang)
      END IF


cBZ---right scattered field
        if ((st.eq.12).and.(case_id.eq.1))
     1     c5=c5-gquad(0.0,2*pi,ms*2,m,time*dt,i*dz,
     1     (k-1)*dz,inc_ang)

cBZ---pseudo-right scattered field for case 2,3
        if ((st.eq.12).and.((case_id.eq.3)
```

```
     1      .or.(case_id.eq.2))) then
          read(14,*) ephix
          c5 = c5 - ephix
        end if


        hr(k,i)=hr(k,i)+eta*(c1*(ephi(k,i)-c5)-ms*c2*ez(k,i))

C ***************************Hphi
C ******* only calculate if not a boundary cell as defined by
C ******* the conform_grid1 array

        c1=dt/(mu*dz)

        IF (k.eq.1) THEN
          if (case_id.ne.3) then
            if (((case_id.eq.1).or.(case_id.eq.5))
     1          .and.(i.lt.left_y)) then
              c5=erzlx(1,i)+erphilx(1,i)
            else
              c5=erzl(1,i)+erphil(1,i)
            end if
          else
            c5=0
          end if
        ELSE
          c5=er(k-1,i)
        END IF

        if (i.eq.maxr) THEN
          if ((case_id.eq.1).or.(case_id.eq.5)) then
            c4= ezrt(k,1) + ezphit(k,1)
          else
c Create PEC in PML for all cases except for 1,5
            c4 = 0
          end if
        ELSE
          c4=ez(k,i+1)
        END IF

cBZ****top total field: Do NOT add anything
c     since we have nothing recorded to add in
c     except for initial case
        if ((st.eq.4.or.st.eq.5.or.st.eq.6).AND.
     1      (case_id.eq.1))
     1     c4 = c4 + gquad(0.0,2*pi,
     1     ms*6,m,time*dt,(i+1)*dz,k*dz,
     1     inc_ang)

cBZ****left total field
        if ((st.eq.2.or.st.eq.3).AND.
     1      ((case_id.eq.4).OR.(case_id.eq.5))) THEN
          read(13,*, END=15) er_in
          c5 = c5 + er_in
 15     END IF

        if ((st.eq.3.or.st.eq.4).AND.
     1      (case_id.eq.1)) THEN
          c5=c5+gquad(0.0,2*pi,ms*4,m,time*dt,i*dz,
     1         (k-1)*dz,inc_ang)
        END IF
```

```
cBZ****right scattered field
      if ((st.eq.12).and.(case_id.eq.1))
   1     c5=c5 - gquad(0.0,2*pi,ms*4,m,time*dt,i*dz,
   1     (k-1)*dz,inc_ang)

cBZ---pseudo-right scattered field for case 3
      if ((st.eq.12).and.((case_id.eq.3)
   1     .or.(case_id.eq.2))) then
        read(13,*) erx
        c5 = c5 - erx
      end if

      hphi(k,i) = hphi(k,i)+eta*(c1*(c4-ez(k,i)+c5-er(k,i)))

C ***************************Hz

      c1=((i+0.0-1.0)*dt/mu)/((i+0.5-1.0)*dz)
      c2=((i+1.0-1.0)*dt/mu)/((i+0.5-1.0)*dz)
      c3=(m*dt/mu)/((i+0.5-1.0)*dz)
      if (i.eq.maxr) THEN
        if ((case_id.eq.1).or.(case_id.eq.5)) then
          c4=ephirt(k,1)+ephizt(k,1)
        else
          c4 = 0
        end if
      ELSE
        c4=ephi(k,i+1)
      END IF

cBZ****NO ADDING IN unless first segment
      if ((st.eq.4.or.st.eq.5.or.st.eq.6).AND.
   1     (case_id.eq.1))
   1     c4=c4+gquad(0.0,2*pi,
   1     ms*2,m,time*dt,(i+1)*dz,k*dz,
   1     inc_ang)

      hz(k,i)=hz(k,i)+eta*(c1*ephi(k,i)-c2*c4+ms*c3*er(k,i))

    ELSE
C********************************************************************** C
C ****************************On Axis Equations*********************** C
C ********************************************************************** C

C***************************Hr
c     if (k.eq.9) print *,k,i,ephi(k,i),ephi(k,i+1)
      c1=dt/(mu*dz)

      IF (k.eq.1) THEN
        if (case_id.ne.3) then
          if (((case_id.eq.1).or.(case_id.eq.5))
   1          .and.(i.lt.left_y)) then
            c5=ephizlx(1,i)+ephirlx(1,i)
          else
            c5=ephizl(1,i)+ephirl(1,i)
          end if
        else
          c5=0
        end if
      ELSE
```

```fortran
           c5=ephi(k-1,i)
         END IF



cBZ***Bottom left total field
         if ((st.eq.2).and.((case_id.eq.4).OR.
     1       (case_id.eq.5))) THEN
           read(14,*, END=16) ephi_in
           c5 = c5 + ephi_in
 16      END IF

cBZ***Bottom right scattered field
         if ((st.eq.12).and.(case_id.eq.1))
     1       c5= c5 - gquad(0.0,2*pi,ms*2,m,time*dt,i*dz,
     1       (k-1)*dz,inc_ang)

cBZ---pseudo-right scattered field for case 3
         if ((st.eq.12).and.((case_id.eq.3)
     1       .or.(case_id.eq.2))) then
           read(14,*) ephix
           c5 = c5 - ephix
         end if

         hr(k,i)=hr(k,i)+eta*(-ms*c1*ez(k,i+1)+c1*(ephi(k,i)-c5))

c******If the fourier mode !=1 then ephi(k,1) and hr(k,1) = zero
         if (m.ne.1) THEN
           hr(k,i)=0.0
         END IF

C ***************************Hphi


         c1=dt/(mu*dz)

         IF (k.eq.1) THEN
           if (case_id.ne.3) then
             if (((case_id.eq.1).or.(case_id.eq.5))
     1           .and.(i.lt.left_y)) then
               c5=erzlx(1,i)+erphilx(1,i)
             else
               c5=erzl(1,i)+erphil(1,i)
             end if
             else
             c5=0
           end if
         ELSE
           c5=er(k-1,i)
         END IF


         if (i.eq.maxr) THEN
           if ((case_id.eq.1).or.(case_id.eq.5)) then
             c4=ezrt(k,1)+ezphit(k,1)
           else
             c4=0
           end if
         ELSE
           c4=ez(k,i+1)
         END IF
```

```
c**** Top scattered
      if ((st.eq.4.or.st.eq.5.or.st.eq.6).AND.
   1      (case_id.eq.1))
   1      c4=c4+gquad(0.0,2*pi,
   1      ms*6,m,time*dt,(i+1)*dz,k*dz,
   1      inc_ang)

c****Lower left hand corner total field
      if ((st.eq.2).AND.((case_id.eq.4).OR.
   1      (case_id.eq.5))) THEN
        read(13,*, END=17) er_in
        c5= c5 + er_in
 17   END IF

c****Lower right hand corner scattered field
      if ((st.eq.12).and.(case_id.eq.1))
   1      c5=c5 - gquad(0.0,2*pi,ms*4,m,time*dt,i*dz,
   1      (k-1)*dz,inc_ang)

cBZ---pseudo-right scattered field for case 2,3
      if ((st.eq.12).and.((case_id.eq.3)
   1      .or.(case_id.eq.2))) then
        read(13,*) erx
        c5 = c5 - erx
      end if

      hphi(k,i)=hphi(k,i)+eta*(c1*(c4-ez(k,i)+c5-er(k,i)))

C ****************************Hz

c     print *,k,i
      c1=((i+0.0-1.0)*dt/mu)/((i+0.5-1.0)*dz)
      c2=((i+1.0-1.0)*dt/mu)/((i+0.5-1.0)*dz)
      c3=(m*dt/mu)/((i+0.5-1.0)*dz)

      if (i.eq.maxr) THEN
        if ((case_id.eq.1).or.(case_id.eq.5)) then
          c4=ephirt(k,1)+ephizt(k,1)
        else
          c4 = 0
        end if
      ELSE
        c4=ephi(k,i+1)
      END IF

cBZ****top total field: only add in during first segment
      if ((st.eq.4.or.st.eq.5.or.st.eq.6).AND.
   1      (case_id.eq.1))
   1      c4=c4+gquad(0.0,2*
   1      pi,ms*2,m,time*dt,(i+1)*dz,k*dz,
   1      inc_ang)

      hz(k,i)=hz(k,i)+eta*(c1*ephi(k,i)-c2*c4+ms*c3*er(k,i))
    END IF


    if ((((case_id.eq.1).and.((st.eq.22).or.(st.eq.16)))
   1    .or.((case_id.eq.1).and.((st.eq.23).or.(st.eq.24)))
   1    .or.((case_id.eq.2).and.((st.eq.22).or.(st.eq.16)))
   1    .or.((case_id.eq.2).and.((st.eq.23).or.(st.eq.24)))
   1    .or.((case_id.eq.2).and.(st.eq.12))
```

```
     1   .or.((case_id.eq.2).and.(st.eq.25))
     1   .or.((case_id.eq.3).and.(st.eq.12))
     1   .or.((case_id.eq.3).and.(st.eq.25))
     1   .or.((case_id.eq.4).and.((st.eq.17).or.(st.eq.18)))
     1   .or.((case_id.eq.4).and.((st.eq.26).or.(st.eq.27)))
     1   .or.((case_id.eq.4).and.(st.eq.1))
     1   .or.((case_id.eq.5).and.(st.eq.1)))
     1   .AND.(quit_flag.eq.0))
     1   THEN

cBZ flag must be on to write
        if ((flag.eq.1).AND.(quit_flag.eq.0)) THEN
          if ((case_id.eq.2).and.(st.eq.25)) then
c case2 backscatter
            write(23,*) hr(k,i)
            write(24,*) hphi(k,i)

          else if (((case_id.eq.2).or.(case_id.eq.1))
     1         .and.((st.eq.22).or.(st.eq.16))) then
c case 1, 2 forward scatter
            write(11,*) hr(k,i)
            write(12,*) hphi(k,i)

          else if ((case_id.eq.3).and.(st.eq.25)) then
c case 3 forward scatter
            write(11,*) hr(k,i)
            write(12,*) hphi(k,i)

          else if ((case_id.eq.4)
     1         .and.((st.eq.26).or.(st.eq.27))) then
c case 4 forward scatter
            write(11,*) hr(k,i)
            write(12,*) hphi(k,i)

          else if ((case_id.eq.4)
     1         .and.(st.eq.1)) then
c case 4 back scatter
            write(23,*) hr(k,i)
            write(24,*) hphi(k,i)

          else if ((case_id.eq.5)
     1         .and.(st.eq.1)) then
c case 5 back scatter
            write(23,*) hr(k,i)
            write(24,*) hphi(k,i)
          end if

        END IF
      END IF


      return
      end

C**********************************************************************C
C     Write out numerical values for each point in the bitmap          C
C**********************************************************************C

      SUBROUTINE matlab
      include 'common.f'
      integer i,k
```

```
      open(unit=81,file='matlab.dat',status='unknown',form='formatted')

      do 10 i=pmldepth,1,-1
        do 20 k=pmldepth,1,-1
          write(81,*) hphirl(k,i+maxr)+hphizl(k,i+maxr)
20      continue
        do 30 k=1,maxz
          write(81,*) hphirt(k,i)+hphizt(k,i)
30      continue
        do 40 k=1,pmldepth
          write(81,*) hphirr(k,i+maxr)+hphizr(k,i+maxr)
40      continue
10    continue

      do 50 i=maxr,1,-1
        do 60 k=pmldepth,1,-1
          write(81,*) hphirl(k,i)+hphizl(k,i)
60      continue
        do 70 k=1,maxz
          write(81,*) hphi(k,i)
70      continue
        do 80 k=1,pmldepth
          write(81,*) hphirr(k,i)+hphizr(k,i)
80      continue
50    continue

      return
      end
```

A.2 Geometry

This portion of the program defines the BOR mesh and flags the mesh as appropriate for each cavity segment. It also resets the electric and magnetic field values on this mesh to enforce the PEC of the geometry.

```
c***************************************************************c
      integer function round(x)
```

```
      real*8 x, dec

      dec = int(x)-x

      if (abs(dec).gt.(0.5d0)) then
        round = int(x)+1
      else
        round = int(x)
      end if

      return
      end


c***********************************************************************
c SETUP_STAIRCASE setups all the parameters needed to run the simulation
c   including a staircasing algorithm for representing the target.
c***********************************************************************

      SUBROUTINE setup_staircase

      implicit none
      include 'common.f'

      real*8 xstair(1:MAX_STAIR_NODES),
     1    ystair(1:MAX_STAIR_NODES), xcomp, ycomp,
     1    dx, dy

      real*8 zstep, radius

      integer xdir, ydir, x1, x2, y1, y2, round,
     1    defaults, index

      real*8 max_x_nodeint, max_y_nodeint,
     1    min_x_nodeint, min_y_nodeint

      real*8 max_x_node, max_y_node, min_x_node, min_y_node,
     1    slope, offset, dist_to_line, xnodes(1:MAX_NODES),
     2    ynodes(1:MAX_NODES), delta, current_x, current_y

      write(6,*) 'Setting up geometry. . .'

      write(6,'(''*Accept spacing defaults [Y=1,N=2]: '',$)')
      read(5,*) defaults

      if (defaults.eq.1) then
        xtot_sp=10
        ytot_sp=10

        xscat_sp=15
        yscat_sp=15

        xscatplay_sp=1
        xextend_sp=1

        xhuy_sp=2
        yhuy_sp=2

cBZ 12/13/02-Don't use this, makes code less
c readable!
c         xall_sp = xtot_sp+xscat_sp
```

```
c        yall_sp = ytot_sp+yscat_sp

      else
        write(6,'(''*Enter xtot_sp [10]: '',$)')
        read(5,*) xtot_sp
        write(6,'(''*Enter ytot_sp [10]: '',$)')
        read(5,*) ytot_sp

        write(6,'(''*Enter xscat_sp [15]: '',$)')
        read(5,*) xscat_sp
        write(6,'(''*Enter yscat_sp [15]: '',$)')
        read(5,*) yscat_sp

        write(6,'(''*Enter xhuy_sp [2]: '',$)')
        read(5,*) xhuy_sp
        write(6,'(''*Enter yhuy_sp [2]: '',$)')
        read(5,*) yhuy_sp

        xall_sp = xtot_sp+xscat_sp
        yall_sp = ytot_sp+yscat_sp
      end if

      errorcount = 0

c**** Read geometry file in.
      open(unit=10,file=fnamein,status='unknown',form='formatted')
      read(10,*) dz
      delta = dz
      read(10,*) total_nodes
      NP = total_nodes
      if (total_nodes.gt.MAX_NODES) then
        errorcount = errorcount+1
        errors(errorcount) = NODE_ERROR
        call memory_check
      end if

      do 10 index=1,total_nodes
        read(10,*) xnodes(index), ynodes(index)
 10   continue
      close(unit=10)



C**** Scale, position, and round object

      max_x_node = xnodes(total_nodes)/delta
      max_y_node = ynodes(total_nodes)/delta

      min_x_node = xnodes(1)/delta
      min_y_node = ynodes(1)/delta

      do 20 index=1,total_nodes

        xnodes(index) = xnodes(index)/delta
        if (xnodes(index).gt.max_x_node) then
          max_x_node = xnodes(index)
        end if
        if (xnodes(index).lt.min_x_node) then
          min_x_node = xnodes(index)
        end if
```

```
        ynodes(index) = ynodes(index)/delta
        if (ynodes(index).gt.max_y_node) then
          max_y_node = ynodes(index)
        end if
        if (ynodes(index).lt.min_y_node) then
          min_y_node = ynodes(index)
        end if

 20   continue


cBZ CONCERNS PLACEMENT OF STRUCTURE!  NOT PLACEMENT OF PML!
c LEFT ALIGNMENT

      if (case_id.eq.1) then
c     Beginning segment at opening:
c     PEC left justified, touches left PEC
c     Scat fields exist only on exterior of cavity

        do 30 index=1,total_nodes
c     Structure starts at z = 2, touching PML on LHS
c     with artificial extension of one delta on LHS
          xnodes(index) = round(xnodes(index) - min_x_node)
     1              + 2
c     Indices into lattice cannot start at zero
          ynodes(index) = round(ynodes(index)) + 1.d0
          RB(index) = ynodes(index)
          ZB(index) = xnodes(index)
 30     continue

      else if (case_id.eq.2) then
c     Propagating down the cavity: general case
c     "Incident fields" placed on RHS
c     at Scattered/Total field boundary
c     PEC touches PML on LHS

        do 31 index=1,total_nodes
c     Structure starts at z=2, touching PML on LHS
c     with artificial extension of one delta on LHS
          xnodes(index) = round(xnodes(index) - min_x_node)
     1          + 2
          ynodes(index) = round(ynodes(index)) + 1.d0
          RB(index) = ynodes(index)
          ZB(index) = xnodes(index)
 31     continue


      else if (case_id.eq.3) then
c     Bottom of the cavity
c     "Incident fields" at the Scat/Tot boundary on right hand side.
c     NO artificial extension on LHS
        do 32 index=1,total_nodes
            xnodes(index) = round(xnodes(index) - min_x_node)
     1          + 1
          ynodes(index) = round(ynodes(index)) + 1.d0
          RB(index) = ynodes(index)
          ZB(index) = xnodes(index)
 32     continue
```

```
      else if (case_id.eq.4) then
c     Propagating out of the cavity: general case
c     "Incident fields" placed on the LHS.
c     NO artificial extension on LHS
        do 33 index=1,total_nodes
c    Structure starts at z=2 with artificial extension of 1 delta lhs
           xnodes(index) = round(xnodes(index) - min_x_node)
     1          + 2
            ynodes(index) = round(ynodes(index)) + 1.d0
            RB(index) = ynodes(index)
            ZB(index) = xnodes(index)
 33     continue


      else if (case_id.eq.5) then
c     Ending segment at opening:
c     Regular surrounding fields
c     "Incident fields" on the left boundary.
c     PEC touches scattered field on left side
        do 34 index=1,total_nodes
c    Structure starts at z = 2 with artificial extension of 1 delta
            xnodes(index) = round(xnodes(index) - min_x_node)
     1          + 2
cBZ changed from xscat_sp to xscatplay_sp 12/31/02
            ynodes(index) = round(ynodes(index)) + 1.d0
            RB(index) = ynodes(index)
            ZB(index) = xnodes(index)
 34     continue
      else
        print *, 'Error in segment ID number.'
        pause
      end if


c**** Estimate total number of staircase nodes needed.
      dx = 0
      dy = 0
      do 500 index=1,total_nodes-1
        dx = dx + int(abs(xnodes(index)-xnodes(index+1)))
        dy = dy + int(abs(ynodes(index)-ynodes(index+1)))
 500  continue
c**** extra point needed for first point
      dy=dy+1

      if (2*(dx+dy)-1.gt.MAX_STAIR_NODES) then
        stair_node_count = 2*(dx+dy)-1
        errorcount = errorcount+1
        errors(errorcount) = MAX_STAIR_ERROR
      end if

c**** Generate a staircase model by digitizing each line segment.

      stair_node_count = 1
      xstair(stair_node_count) = xnodes(1)
      ystair(stair_node_count) = ynodes(1)

      do 40 index=1,total_nodes-1
        current_x = xnodes(index)
        current_y = ynodes(index)

 100    stair_node_count = stair_node_count
```

```
      if (abs(current_x-xnodes(index+1)).gt.tole.OR.
  1      abs(current_y-ynodes(index+1)).gt.tole) then

        xcomp = xnodes(index+1)-current_x
        ycomp = ynodes(index+1)-current_y

        if (xcomp.ne.0) then
          xdir = int(abs(xcomp)/xcomp)
        else
          xdir = 0
        end if
        if (ycomp.ne.0) then
          ydir = int(abs(ycomp)/ycomp)
        else
          ydir = 0
        end if


        stair_node_count = stair_node_count + 1

        if (xdir.ne.0.AND.ydir.ne.0) then
          slope = (ynodes(index+1)-ynodes(index)) /
  &              (xnodes(index+1)-xnodes(index))
          offset = ynodes(index)-slope*(xnodes(index))

          if (dist_to_line(-slope,1.0d0,offset,
  &              dble(current_x+xdir),dble(current_y)).lt.
  &              dist_to_line(-slope,1.0d0,offset,
  2              dble(current_x),dble(current_y+ydir))) then

            xstair(stair_node_count) = current_x+xdir
            ystair(stair_node_count) = current_y
            current_x = current_x+xdir
          else
            xstair(stair_node_count) = current_x
            ystair(stair_node_count) = current_y+ydir
            current_y = current_y+ydir
          end if
        else
          xstair(stair_node_count) = current_x+xdir
          ystair(stair_node_count) = current_y+ydir
          current_x = current_x+xdir
          current_y = current_y+ydir
        end if

        goto 100
      end if
40  continue

    if ((dx+dy).ne.stair_node_count) then
      write(6,*) 'estimate = ', dx+dy
      write(6,*) 'actual = ', stair_node_count
    end if


c**** now figure out which fields to set to zero.

cBZ 12/13/02 For cases [2,4], we only need the interior
c   cavity surface. These datapoints run in the -z
c   direction. So to extend the cavity by a lattice cube delta z,
```

```
c   we merely need to repeat the data for
c   either the first or last point

      if ((case_id.eq.2).OR.(case_id.eq.3).OR.(case_id.eq.4)) then
c   case 2,3,4 artificially extend to right by two so leave
c . first four indices of stair_zero free
        staircount = 5
c case 1,5 is extended by one to the LEFT but
c the points start on the outer surface.
c So we still have to also leave the
c first two indices free
c Also, we need to manually set "first" ez
c so we need the first THREE  indices free
      else if ((case_id.eq.1).or.(case_id.eq.5)) then
        staircount = 4
      else
        staircount = 1
      end if


      do 90 index = 1,stair_node_count-1
        xcomp = xstair(index+1)-xstair(index)
        ycomp = ystair(index+1)-ystair(index)
        if (ycomp.gt.tole.AND.abs(xcomp).lt.tole) then
c VERT up
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = ephif
          staircount = staircount+1
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = erf
          staircount = staircount+1
        else if (ycomp.lt.tole.AND.abs(xcomp).lt.tole) then
c VERT down
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = ephif
          staircount = staircount+1
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))-1
          stair_zero(staircount,3) = erf
          staircount = staircount+1
        else if (xcomp.gt.0.AND.abs(ycomp).lt.tole) then
c HORZ to right
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = ephif
          staircount = staircount+1
          stair_zero(staircount,1) = int(xstair(index))+1
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = ezf
          staircount = staircount+1
        else if (xcomp.lt.0.AND.abs(ycomp).lt.tole) then
c HORZ to left
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
          stair_zero(staircount,3) = ephif
          staircount = staircount+1
          stair_zero(staircount,1) = int(xstair(index))
          stair_zero(staircount,2) = int(ystair(index))
```

```
          stair_zero(staircount,3) = ezf
          staircount = staircount+1
        else
          print *,'error in determing staircase type. '
          print *,' (z,x) = ', stair_zero(index,1),
     2        stair_zero(index,2), index, xcomp, ycomp
          stair_zero(index,3) = ephif
          pause
        end if
 90   continue

c**** Complete the last zero field
      stair_zero(staircount,1) = int(xstair(stair_node_count))
      stair_zero(staircount,2) = int(ystair(stair_node_count))
      stair_zero(staircount,3) = ephif

cc INTERIOR RIGHT by TWO
      if ((case_id.eq.3).or.(case_id.eq.2).or.(case_id.eq.4)) then
c     extend to right by two, staircount = 1
c     the first point is the rightmost
c     (Of course, assuming the points run from
c     opening to the shorted end).
c     So we set the first four indices
        stair_zero(1,1) = stair_zero(5,1) + 2
        stair_zero(1,2) = stair_zero(5,2)
        stair_zero(1,3) = ephif
        stair_zero(2,1) = stair_zero(5,1) + 2
        stair_zero(2,2) = stair_zero(5,2)
        stair_zero(2,3) = ezf
        stair_zero(3,1) = stair_zero(5,1) + 1
        stair_zero(3,2) = stair_zero(5,2)
        stair_zero(3,3) = ephif
        stair_zero(4,1) = stair_zero(5,1) + 1
        stair_zero(4,2) = stair_zero(5,2)
        stair_zero(4,3) = ezf
      end if

c EXTERIOR LEFT by ONE
      if ((case_id.eq.1).or.(case_id.eq.5)) then
c     extend to LEFT (exterior points)
c     the first point when it is case 1
        stair_zero(1,1) = stair_zero(4,1) - 1
        stair_zero(1,2) = stair_zero(4,2)
        stair_zero(1,3) = ephif
        stair_zero(2,1) = stair_zero(4,1) - 1
        stair_zero(2,2) = stair_zero(4,2)
        stair_zero(2,3) = ezf
c manually set "first" ez
        stair_zero(3,1) = stair_zero(4,1)
        stair_zero(3,2) = stair_zero(4,2)
        stair_zero(3,3) = ezf
      end if

cINTERIOR LEFT by ONE
      if ((case_id.eq.1).or.(case_id.eq.5).or.
     $    (case_id.eq.2).or.(case_id.eq.4)) then
c  but first must manually set "last" ez
        stair_zero(staircount+1,1) = stair_zero(staircount,1)
        stair_zero(staircount+1,2) = stair_zero(staircount,2)
        stair_zero(staircount+1,3) = ezf
c     extend to LEFT by one delta (interior)
```

```
        stair_zero(staircount+2,1) = stair_zero(staircount,1) - 1
        stair_zero(staircount+2,2) = stair_zero(staircount,2)
        stair_zero(staircount+2,3) = ephif
        stair_zero(staircount+3,1) = stair_zero(staircount,1) - 1
        stair_zero(staircount+3,2) = stair_zero(staircount,2)
        stair_zero(staircount+3,3) = ezf

        staircount = staircount + 3
      end if

      chuck =  stair_zero(1,2)

      stair_node_count = staircount

c**** Find the highest y of stair_zero
      high_y = stair_zero(1,2)
      do 999 index = 2,staircount
        if (stair_zero(index,2).gt.high_y) then
          high_y = stair_zero(index,2)
        end if
 999  continue

c**** Find the highest x of stair_zero
      high_x = stair_zero(1,1)
      right_y = stair_zero(1,2)
      do 888 index = 2,staircount
        if (stair_zero(index,1).gt.high_x) then
          high_x = stair_zero(index,1)
cBZ 1/6/03 set here!
          right_y = stair_zero(index,2)
c     Also serves to find the opening coords for cases 5,1
          x_opening = stair_zero(index,1)
          y_opening = stair_zero(index,2) - 1
        end if
 888    continue



c#############
c   SET MAXZ
c#############
      if (case_id.eq.1) then
        maxz = xscat_sp + xtot_sp + high_x
      else if (case_id.eq.2) then
        maxz = high_x
      else if (case_id.eq.3) then
        maxz = high_x
      else if (case_id.eq.4) then
        maxz = high_x
c keep case 5 exactly the same as case 1
      else if (case_id.eq.5) then
        maxz = xscat_sp + xtot_sp + high_x
      end if



c*** To calculate correct incident angle, must calculate
c    maxz as if modeling entire cavity and get the offset
      if (case_id.eq.1) then
        max_length = max_length/delta
        maxztrue = round(max_length)
```

```
        maxztrue = maxztrue + 2.0*(xtot_sp+xscat_sp)
        zoffset = maxztrue - maxz
      end if

c#############
c   SET MAXR
c#############
      if (case_id.eq.1) then
        maxr = (ytot_sp + yscat_sp + max_height/delta)
      else if (case_id.eq.2) then
        maxr = high_y
      else if (case_id.eq.3) then
        maxr = high_y
      else if (case_id.eq.4) then
        maxr = high_y
      else if (case_id.eq.5) then
        maxr = (ytot_sp + yscat_sp + max_height/delta)
      end if

      len = delta*(max_x_node - min_x_node)
      obj_height = delta*(max_y_node)

      if (maxz.gt.MAX_Z_CELLS) then
        errorcount = errorcount + 1
        errors(errorcount) = MAX_Z_ERROR
      end if

      if (maxr.gt.MAX_R_CELLS) then
        errorcount = errorcount+1
        errors(errorcount) = MAX_R_ERROR
      end if



      if (case_id.eq.1) then
c**** For case 1, find y-value of lower edge of
c     cavity where it touches PML.
c     This should be the last point where x=1, if not we've
c     got problems. . .
        lower_edgetot = stair_zero(staircount,2)
        if (stair_zero(staircount,1).ne.1) then
            print *, "last x  =  ",stair_zero(staircount,1)
          pause
        end if

        upper_edgetot = stair_zero(2,2)
        upper_edgescat = stair_zero(1,2)

        do 998 index = 1,staircount-1
c Now find the upper edge of the cavity where it
c crosses the tot/scat boundary
          if ((stair_zero(index,1).eq.xscat_sp).and.
     1         (stair_zero(index+1,1).eq.xscat_sp+1)) then
            upper_edgescat = stair_zero(index,2)
            upper_edgetot = stair_zero(index+1,2)
            GO TO 998
          end if
 998    continue

        do 889 index = 1,staircount,1
c Now find the upper edge of the cavity where it
c crosses the Huygens surface
```

```
c rcsz1 = xscat_sp - xhuy+sp + 1
c and then
c rcsz1 = rcsz1+ 1 to account for the extension

          if ((stair_zero(index,1).eq.15).and.
     $        (stair_zero(index+1,1).eq.16)) then
c    $         (xscat_sp - xhuy_sp + 1 + 1)) then
            upper_edgehuy = stair_zero(index,2)
            GO TO 889
          end if
 889    continue

      end if

      if ((case_id.eq.2).or.(case_id.eq.3)) then
c**** For cases 2 and 3, find y-value of lower edges of
c     cavity on both sides.
        lower_edgeright = stair_zero(1,2)
        lower_edgeleft = stair_zero(staircount,2)
        if ((stair_zero(1,1).ne.maxz).or.
     1      (stair_zero(staircount,1).ne.1)) then
          print *, "first x  =  ",stair_zero(1,1)
          print *, "maxz  =  ",maxz
          print *, "last x  =  ",stair_zero(staircount,1)
          pause
        end if
      end if

      if (case_id.eq.4) then
c**** For case 4, find y-value of lower edge of
c     cavity where it crosses total/scat field on LHS.
c     This should be the last point where x = 1, if not we've
c     got problems...
        lower_edgeright = stair_zero(1,2)
        lower_edgeleft = stair_zero(staircount,2)
        if ((stair_zero(staircount,1).ne.1).or.
     1      (stair_zero(1,1).ne.maxz)) then
          print *, "last x  =  ",stair_zero(staircount,1)
          pause
        end if
      end if


      if (case_id.eq.5) then
c**** For case 5, find y-value of lower edge of
c     cavity where it crosses scat/tot
        lower_edgetot = stair_zero(2,2)
        lower_edgescat = stair_zero(1,2)
c We are moving in a "backwards" direction
c     to look at the interior of the cavity
        do 997 index = staircount,1,-1
          if ((stair_zero(index,1).eq.xscatplay_sp+1).and.
     1      (stair_zero(index-1,1).eq.xscatplay_sp+2)) then
            lower_edgescat = stair_zero(index,2)
            lower_edgetot = stair_zero(index-1,2)
            GO TO 997
          end if
 997    continue

        do 887 index = 1,staircount,1
c Now find the upper edge of the cavity where it
```

```
c crosses the Huygens surface
c rcsz1 = xscat_sp - xhuy+sp + 1
c           if (stair_zero(index,1).eq.(xscat_sp - xhuy_sp + 1)) then
          if ((stair_zero(index,1).eq.14).and.
     $          (stair_zero(index+1,1).eq.15)) then
            upper_edgehuy = stair_zero(index,2)
            GO TO 887
          end if
 887    continue

      end if


cBZ 1/5/03 set the coordinates for the rightmost
c and leftmost coordinates
c  <-<-
       right_x = high_x
c        right_y was set when we found high_x
       left_x =  stair_zero(staircount,1)
       left_y =  stair_zero(staircount,2)


      open(unit=10,file='stairnew.dat',status='unknown',
     1    form='formatted')

      do 1000 index=1,stair_node_count
        write(10,*) stair_zero(index,1), stair_zero(index,2),
     1        stair_zero(index,3)
 1000 continue
      close(unit=10)


      RETURN
      END

c**********************************************************************
c STAIR_BOUNDARY_CONDITIONS sets all the appropriate fields in the
c   staircase model to zero.
c**********************************************************************

      SUBROUTINE stair_boundary_conditions

      implicit none
      include 'common.f'

      integer index

      do 10 index = 1,stair_node_count
        if (stair_zero(index,3).eq.ezf) then
          ez(stair_zero(index,1),stair_zero(index,2)) = 0.0
        else if (stair_zero(index,3).eq.ephif) then
          ephi(stair_zero(index,1),stair_zero(index,2)) = 0.0
        else if (stair_zero(index,3).eq.erf) then
          er(stair_zero(index,1),stair_zero(index,2)) = 0.0
        else
          print *,'unknown stair_zero type'
          print *,stair_zero(index,1),stair_zero(index,2),
     1          stair_zero(index,3)
          pause
        end if
 10   continue
```

```
      RETURN
      END


c**********************************************************************
c REAL FUNCTION DIST_TO_LINE returns the perpendicular distance from a
c point in space (x,y) to a line that is of the form Ax+By=C
c**********************************************************************

      REAL*8 FUNCTION dist_to_line(A,B,C,x,y)

      implicit none
      real*8 A,B,C,x,y

      dist_to_line = abs((A*x+B*y-C)/sqrt(A**2.0d0 + B**2.0d0))

      RETURN
      END


c***************************************************************c
c setups cells for scattered/total field calculations.         c
c see picture in notes for numbering scheme.                   c
c***************************************************************c

      subroutine setup_scat

      implicit none
      include 'common.f'

cBZ added x0 and y0
      integer k,i,x1,y1,x2,y2, x0, y0
      integer downleftx, downlefty, downrightx, downrighty
      integer upleftx, uplefty, uprightx, uprighty

cBZ 9/30/02 This SHOULD be okay!!!???!!!
      mxr=int(obj_height/dz)

      do 10 k=1,maxz
        do 20 i=1,maxr
          scattot(k,i)=15
 20     continue
 10   continue

c**********************************
      if (case_id.eq.1) then

c***** We define rcsz Huygen's surface
c          rcsz1 = 1 Can not =1 since
c         running the Huygen's surface into
c         PEC
         rcsz1 = xscat_sp - xhuy_sp + 1
c          rcsz1 = high_x - 40 + 1
c       But since case 1 is extended by one
c       to the left, we add + 1
c       to match case 5
c          maxr = ytot_sp + yscat_sp + high_y
         rcsz2 = maxz - xscat_sp + xhuy_sp
         rcsz2 = rcsz2 + 1
```

```
      mheight = maxr - yscat_sp + yhuy_sp

c*****scat/tot rcs box region definers
cBZ x0 and y0 refer to the most lower left hand corner
c****Where scat/tot fields are depends on case
      x_start_tot = 1
cBZ yes
      x_end_tot = maxz - xscat_sp


      do 101 k = xscat_sp + 1 + xextend_sp, maxz - xscat_sp - 1
        do 111 i = upper_edgescat+1, maxr-yscat_sp-1
          scattot(k,i) = 14
111     continue
101   continue

      do 100 k = 1, maxz - xscat_sp - 1
        do 110 i = 1, upper_edgescat
          scattot(k,i) = 14
110     continue
100   continue


      scattot(1,1) = 24

      scattot(xextend_sp+1,1) = 16

      scattot(maxz-xscat_sp,1) = 8
      scattot(xscat_sp + xextend_sp + 1, maxr - yscat_sp) = 4
      scattot(maxz - xscat_sp, maxr - yscat_sp) = 6

      i = 1

      do 30 k = xextend_sp+2, maxz - xscat_sp - 1
        scattot(k,i) = 9
30    continue


      i = maxr - yscat_sp
      do 40 k = xscat_sp + xextend_sp + 2, maxz-xscat_sp - 1
        scattot(k,i) = 5
40    continue

      k = xscat_sp + xextend_sp + 1
      do 50 i = upper_edgetot + 1, maxr - yscat_sp - 1
        scattot(k,i) = 3
50    continue


      k = xextend_sp + 1

      do 51 i = 2, lower_edgetot - 1
        scattot(k,i) = 22
51    continue


      k = 1
      do 511 i = 2, lower_edgetot - 1
        scattot(k,i) = 23
511   continue
```

```
      k = maxz - xscat_sp
      do 60 i = 2, maxr - yscat_sp - 1
        scattot(k,i) = 7
60    continue

      k = xscat_sp + xextend_sp
      do 70 i = upper_edgescat + 1, maxr - yscat_sp
        scattot(k,i) = 1
70    continue

      k = maxz - xscat_sp + 1
      do 80 i = 1, maxr - yscat_sp
        scattot(k,i) = 12
80    continue

      i = maxr - yscat_sp + 1
      do 90 k = xscat_sp + xextend_sp + 1, maxz-xscat_sp
        scattot(k,i) = 11
90    continue

    else if (case_id.eq.2) then
      x_start_tot = 1
      x_end_tot = maxz - xscatplay_sp


      do 102 k = 1, maxz
        do 112 i = 1, maxr
          scattot(k,i) = 14
112     continue
102   continue


      scattot(1,1) = 24

      scattot(1+xextend_sp,1) = 16


      scattot(maxz - 2, 1) = 8



      i = 1

      do 32 k = xextend_sp+2, maxz - 3
        scattot(k,i) = 9
32    continue



      k = xextend_sp + 1

      do 52 i = 2, lower_edgeleft - 1
        scattot(k,i) = 22
52    continue

      k = 1
      do 512 i = 2, lower_edgeleft - 1
        scattot(k,i) = 23
512   continue
```

```
      k = maxz - 2
      do 62 i = 2, lower_edgeright - 1
        scattot(k,i) = 7
62    continue

      k = maxz - 1
      do 82 i = 1, lower_edgeright - 1
        scattot(k,i) = 12
82    continue

      k = maxz
      do 182 i = 1, lower_edgeright - 1
        scattot(k,i) = 25
182   continue


    else if (case_id.eq.3) then

      x_start_tot = 1
      x_end_tot = maxz - xscatplay_sp - 1


      do 103 k = 1, maxz
        do 113 i = 1, maxr
          scattot(k,i) = 14
113     continue
103   continue

      scattot(maxz - xscatplay_sp - 1, 1) = 8

      i = 1
      do 33 k = 1, maxz - xscatplay_sp - 2
        scattot(k,i) = 9
33    continue

      k = maxz - xscatplay_sp - 1
      do 63 i = 2, lower_edgeright - 1
        scattot(k,i) = 7
63    continue

c RECORD at 12, add in at 7 & 8
      k = maxz - xscatplay_sp + 1 - 1
      do 83 i = 1, lower_edgeright - 1
        scattot(k,i) = 12
83    continue

      k = maxz - xscatplay_sp + 1
      do 93 i = 1, lower_edgeright - 1
        scattot(k,i) = 25
93    continue


    else if (case_id.eq.4) then

      x_start_tot = 3
      x_end_tot = maxz

      do 104 k = 1, maxz
        do 114 i = 1, maxr
```

```
            scattot(k,i) = 14
 114      continue
 104    continue

        scattot(3,1) = 2
        scattot(maxz-1, 1) = 18
        scattot(maxz, 1) = 27

        i = 1
        do 34 k = 4, maxz - 2
          scattot(k,i) = 9
 34     continue

        k = 3
        do 44 i = 2, lower_edgeleft - 1
          scattot(k,i) = 3
 44     continue

        k = maxz - 1
        do 64 i = 2, lower_edgeright - 1
          scattot(k,i) = 17
 64     continue

        k = maxz
        do 164 i = 2, lower_edgeright - 1
          scattot(k,i) = 26
 164    continue

        k = 2
        do 84 i = 1, lower_edgeleft - 1
          scattot(k,i) = 1
 84     continue

        k = 1
        do 184 i = 1, lower_edgeleft - 1
          scattot(k,i) = 23
 184    continue

      else if (case_id.eq.5) then

c***** We define rcsz Huygen's surface
c This will be same as in case 1
        rcsz1 = xscat_sp - xhuy_sp + 1
        rcsz2 = maxz - xscat_sp + xhuy_sp
        rcsz2 = rcsz2 + 1

        mheight = maxr - yscat_sp + yhuy_sp

        x_start_tot = xscatplay_sp + 1
        x_end_tot = maxz


        do 105 k = 1, maxz
          do 115 i = 1, maxr
            scattot(k,i) = 14
 115      continue
 105    continue

        scattot(3,1) = 2
        scattot(maxz,1) = 18
        scattot(1,maxr) = 20
```

```
        scattot(maxz,maxr) = 19

        i = 1
        do 35 k = 4, maxz - 1
          scattot(k,i) = 9
35      continue

        k = maxz
        do 45 i = 2, maxr-1
          scattot(k,i) = 17
45      continue

        i = maxr
        do 55 k = 2, maxz - 1
          scattot(k,i) = 21
55      continue

        k = 3
        do 65 i = 2, lower_edgetot - 1
          scattot(k,i) = 3
65      continue

c NEEDs to be fixed-use the last point next to pml
        k = 1
        do 66 i = lower_edgetot, maxr - 1
          scattot(k,i) = 22
66      continue

        k = 2
        do 85 i = 1, lower_edgetot - 1
          scattot(k,i) = 1
85      continue

        k = 1
        do 86 i = 1, lower_edgetot - 1
          scattot(k,i) = 23
86      continue



      end if

      if ((case_id.EQ.1).OR.(case_id.EQ.5)) then
        rcsz_start = rcsz1
        rcsz_end = rcsz2
      else if ((case_id.EQ.2).OR.(case_id.EQ.4)) then
        rcsz_start = x_start_tot
        rcsz_end = x_end_tot
      else if (case_id.EQ.3) then
        rcsz_start = 1
        rcsz_end = x_end_tot
      end if

c**********************************
C WRITE OUT THE CONNECTION GEOMETRIES
      open(unit=9,file='connect.info',status='unknown',form='formatted')
      write(9,*) lower_edgetot
      write(9,*) lower_edgescat
      write(9,*) upper_edgetot
      write(9,*) upper_edgescat
```

```
      write(9,*) lower_edgeright
      write(9,*) lower_edgeleft
      close(unit=9)
c**********************************

c$$$c**********************************
c$$$c Write out the scattot setup
c$$$      open(unit=9,file='scattot.info',status='unknown',form='formatted')
c$$$      do 301 k=1,maxz
c$$$         do 201 i=1,maxr
c$$$            write(9,*) scattot(k,i)
c$$$ 201     continue
c$$$ 301  continue
c$$$      close(unit=9)
c$$$c**********************************
      return
      end
```

A.3 RCS Calculations

This portion of the program performs a discrete Fourier transform to calculate RCS.

```
C*********************************************************************C
C Performs the dft on the fly.There are 12 field values per grid per  C
C mode cell that will be stored (i.e. eru, erv, ephiu, ephiv, etc.)  C
C They are stored in the complex arrays feru, ferv, fephiu, fphiv,   C
C etc.  Since there are only six arrays at any given time holding    C
C field values (i.e. er, ephi, ez, hr, hphi, hz) the subroutine      C
C updates the appropiate complex arrays based on the input variables C
C mode (what Fourier is being calculated) and eqset (which equation  C
C set is being used).                                                 C
C                                                                     C
C Equation set 1 contains erv, ephiu, ezv, hru, hzu, hphiv            C
C Equation set 2 contains eru, ephiv, ezu, hrv, hzv, hphiu            C
C                                                                     C
C Adjacent field values are averaged in order to approximate their   C
C values along the lattice points (k,i) (Note: hr and ez are never   C
C averaged since they lie on the lattice points)                      C
C                                                                     C
C*********************************************************************C

      SUBROUTINE update_dft(mode,eqset)

      implicit none
      include 'common.f'

      integer k, i, j, mode, eqset
      real*8 temp, tempfreq
      complex*16, parameter :: zim = (0.0d0,1.d0)

      if (eqset.eq.1) THEN
        k=rcsz1
C      ***loop cycles through first mheight-1 points, left side of box
        do 10 i=1,mheight-1
```

```
C     ******loop cycles through all frequencies of interest.
         do 11 j=minf,maxf,stepf
c           tempfreq = low_freq+dfreq*(j+0.0)
           tempfreq = freqlist(j,1)

           ferv(mode,i,j)= 0

           fhzu(mode,i,j)= 0

           fephiu(mode,i,j)= 0

           fhru(mode,i,j)= 0

           fezv(mode,i,j)= 0

           fhphiv(mode,i,j)= 0
 11      continue
 10    continue

       i=mheight
C     ***loop cycles through mheight,mheight+z2-z1 points, top of box
       do 20 k=rcsz1,rcsz2
C     ******loop cycles through all frequencies of interest.
         do 21 j=minf,maxf,stepf
c           tempfreq = low_freq+dfreq*(j+0.0)
           tempfreq = freqlist(j,1)

           temp=(er(k,i)+er(k,i-1))/2.0
           if (k.le.pookie) then
           temp = 0
           end if
           ferv(mode,mheight+k-rcsz1,j)=ferv(mode,mheight+k-rcsz1,
     1        j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt


           temp=(hz(k,i)+hz(k-1,i))/2.0
           if (k.le.pookie) then
           temp = 0
           end if
           fhzu(mode,mheight+k-rcsz1,j)=fhzu(mode,mheight+k-rcsz1,
     1        j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt


           temp=(hphi(k,i)+hphi(k,i-1))/2.0
           if (k.le.pookie) then
           temp = 0
           end if
           fhphiv(mode,mheight+k-rcsz1,j)=fhphiv(mode,mheight+k-
     1        rcsz1,j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt


           temp=hr(k,i)
           if (k.le.pookie) then
           temp = 0
           end if
           fhru(mode,mheight+k-rcsz1,j)=fhru(mode,mheight+k-rcsz1,
     1        j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt


           temp=ez(k,i)
           if (k.le.pookie) then
```

```
            temp = 0
            end if
            fezv(mode,mheight+k-rcsz1,j)=fezv(mode,mheight+k-rcsz1,
     1          j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt


            temp=(ephi(k,i)+ephi(k-1,i))/2.0
            if (k.le.pookie) then
            temp = 0
            end if
            fephiu(mode,mheight+k-rcsz1,j)=fephiu(mode,mheight+k-
     1          rcsz1,j)+temp*exp(2*pi*zim*tempfreq*dt*time)*dt
 21       continue
 20     continue

        k=rcsz2
C     ***loop cycles through last mheight-1 points, right side of box
        do 30 i=1,mheight-1
C     ******loop cycles through all frequencies of interest.
          do 31 j=minf,maxf,stepf
c             tempfreq = low_freq+dfreq*(j+0.0)
            tempfreq = freqlist(j,1)

            if (i.eq.1) then
              temp = er(k,i)
            else
              temp = (er(k,i)+er(k,i-1))/2.0
            end if
            ferv(mode,2*mheight-i+rcsz2-rcsz1,j)=ferv(mode,2*
     1          mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*zim*
     2          tempfreq*dt*time)*dt

            temp=(hz(k,i)+hz(k-1,i))/2.0
            fhzu(mode,2*mheight-i+rcsz2-rcsz1,j)=fhzu(mode,2*
     1          mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*zim*
     2          tempfreq*dt*time)*dt

            temp=(ephi(k,i)+ephi(k-1,i))/2.0
            fephiu(mode,2*mheight-i+rcsz2-rcsz1,j)=fephiu(mode
     1          ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2          zim*tempfreq*dt*time)*dt

            temp=hr(k,i)
            fhru(mode,2*mheight-i+rcsz2-rcsz1,j)=fhru(mode
     1          ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2          zim*tempfreq*dt*time)*dt

            temp=ez(k,i)
            fezv(mode,2*mheight-i+rcsz2-rcsz1,j)=fezv(mode
     1          ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2          zim*tempfreq*dt*time)*dt

            if (i.eq.1) then
              temp = hphi(k,i)
            else
              temp=(hphi(k,i)+hphi(k,i-1))/2.0
            end if
            fhphiv(mode,2*mheight-i+rcsz2-rcsz1,j)=fhphiv(mode
     1          ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2          zim*tempfreq*dt*time)*dt
```

```
31        continue
30      continue

      ELSE
C****Eqset number 2

        k=rcsz1
C     ***loop cycles through first mheight-1 points, left side of box
        do 110 i=1,mheight-1
C     ******loop cycles through all frequencies of interest.
          do 111 j=minf,maxf,stepf
c             tempfreq = low_freq+dfreq*(j+0.0)
            tempfreq = freqlist(j,1)

            feru(mode,i,j)= 0
            fhzv(mode,i,j)= 0
            fephiv(mode,i,j)= 0
            fhrv(mode,i,j)= 0
            fezu(mode,i,j)= 0
            fhphiu(mode,i,j)=0

111       continue
110     continue

        i=mheight
C     ***loop cycles through mheight,mheight+z2-z1 points, top of box
        do 120 k=rcsz1,rcsz2
C     ******loop cycles through all frequencies of interest.
          do 121 j=minf,maxf,stepf
c             tempfreq = low_freq+dfreq*(j+0.0)
            tempfreq = freqlist(j,1)

            temp=(er(k,i)+er(k,i-1))/2.0
c             if (k.eq.rcsz1) write(81,*) temp
            if (k.le.pookie) then
              temp = 0
            end if
            feru(mode,mheight+k-rcsz1,j)=feru(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt

            temp=(hz(k,i)+hz(k-1,i))/2.0
            if (k.le.pookie) then
              temp = 0
            end if
            fhzv(mode,mheight+k-rcsz1,j)=fhzv(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt

            temp=(hphi(k,i)+hphi(k,i-1))/2.0
            if (k.le.pookie) then
              temp = 0
            end if
            fhphiu(mode,mheight+k-rcsz1,j)=fhphiu(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt

            temp=hr(k,i)
            if (k.le.pookie) then
              temp = 0
            end if
```

```
            fhrv(mode,mheight+k-rcsz1,j)=fhrv(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt

            temp=ez(k,i)
            if (k.le.pookie) then
              temp = 0
            end if
            fezu(mode,mheight+k-rcsz1,j)=fezu(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt

            temp=(ephi(k,i)+ephi(k-1,i))/2.0
            if (k.le.pookie) then
              temp = 0
            end if
            fephiv(mode,mheight+k-rcsz1,j)=fephiv(mode,mheight
     1         +k-rcsz1,j)+temp*exp(2*pi*zim*tempfreq*
     2         dt*time)*dt
 121      continue
 120    continue

        k=rcsz2
C     ***loop cycles through last mheight-1 points, right side of box
        do 130 i=1,mheight-1
C     ******loop cycles through all frequencies of interest.
          do 131 j=minf,maxf,stepf
c             tempfreq = low_freq+dfreq*(j+0.0)
            tempfreq = freqlist(j,1)

            if (i.eq.1) then
              temp = er(k,i)
            else
              temp = (er(k,i)+er(k,i-1))/2.0
            end if
            feru(mode,2*mheight-i+rcsz2-rcsz1,j)=feru(mode,2*
     1         mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*zim*
     2         tempfreq*dt*time)*dt

            temp=(hz(k,i)+hz(k-1,i))/2.0
            fhzv(mode,2*mheight-i+rcsz2-rcsz1,j)=fhzv(mode,2*
     1         mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*zim*
     2         tempfreq*dt*time)*dt

            temp=(ephi(k,i)+ephi(k-1,i))/2.0
            fephiv(mode,2*mheight-i+rcsz2-rcsz1,j)=fephiv(mode
     1         ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2         zim*tempfreq*dt*time)*dt

            temp=hr(k,i)
            fhrv(mode,2*mheight-i+rcsz2-rcsz1,j)=fhrv(mode
     1         ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2         zim*tempfreq*dt*time)*dt

            temp=ez(k,i)
            fezu(mode,2*mheight-i+rcsz2-rcsz1,j)=fezu(mode
     1         ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2         zim*tempfreq*dt*time)*dt

            if (i.eq.1) then
              temp = hphi(k,i)
```

```
            else
               temp=(hphi(k,i)+hphi(k,i-1))/2.0
            end if
            fhphiu(mode,2*mheight-i+rcsz2-rcsz1,j)=fhphiu(mode
     1           ,2*mheight-i+rcsz2-rcsz1,j)+temp*exp(2*pi*
     2           zim*tempfreq*dt*time)*dt

 131       continue
 130     continue

      END IF

      return
      end


C***************************************************************C
C write out phasor values to a file.                            C
C***************************************************************C

      SUBROUTINE write_phasors

      implicit none
      include 'common.f'

C*****pm: the current mode being written out.

      integer pm,i,k,fi
      complex*16 temp
      complex*16, parameter :: zim = (0.0d0,1.d0)

      write(6,*) 'Writing out frequency data...'
      if (case_id.eq.1) then
        open(unit=9,file='fdata/info1.dat',status='unknown',
     1       form='formatted')
      else if (case_id.eq.5) then
        open(unit=9,file='fdata/info5.dat',status='unknown',
     1       form='formatted')
      end if

      write(9,*) dt
      write(9,*) dz
      write(9,*) N
      write(9,*) inc_ang
      write(9,*) gd
      write(9,*) sdev
      write(9,*) rcsz1
      write(9,*) rcsz2
      write(9,*) mheight
      write(9,*) mode_start
      write(9,*) mode_end
      write(9,*) modulate
      write(9,*) modfreq
      write(9,*) num_freqs
      do 130 fi=minf,maxf
        write(9,*) freqlist(fi,1), freqlist(fi,2)
 130  continue
      close(unit=9)

 100  format(F12.8, ' ', F12.8)
```

```
      open(unit=9,file='fdata/feru.dat',status='unknown',
     1     form='formatted')

      do 10 pm = mode_start,mode_end
        do 20 i = 1,2*mheight+rcsz2-rcsz1-1
          do 30 k = minf,maxf,stepf
            temp = feru(pm,i,k)
            write(9, *) dble(temp), aimag(temp)
30        continue
20      continue
10    continue
      close(unit=9)

      open(unit=9,file='fdata/ferv.dat',status='unknown',
     1     form='formatted')

      do 101 pm = mode_start,mode_end
        do 201 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 301 k = minf,maxf,stepf
            temp = ferv(pm,i,k)
            write(9, *) dble(temp), aimag(temp)
301       continue
201     continue
101   continue
      close(unit=9)



      open(unit=9,file='fdata/fezu.dat',status='unknown',
     1     form='formatted')

      do 102 pm = mode_start,mode_end
        do 202 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 302 k = minf,maxf,stepf
            temp = fezu(pm,i,k)
            write(9, *) dble(temp), aimag(temp)
302       continue
202     continue
102   continue
      close(unit=9)


      open(unit=9,file='fdata/fezv.dat',status='unknown',
     1     form='formatted')

      do 103 pm = mode_start,mode_end
        do 203 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 303 k = minf,maxf,stepf
            temp = fezv(pm,i,k)
            write(9, *) dble(temp), aimag(temp)
303       continue
203     continue
103   continue
      close(unit=9)


      open(unit=9,file='fdata/fephiu.dat',status='unknown',
     1     form='formatted')

      do 104 pm = mode_start,mode_end
```

```
      do 204 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 304 k = minf,maxf,stepf
          temp = fephiu(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
304     continue
204   continue
104 continue
    close(unit=9)


      open(unit=9,file='fdata/fephiv.dat',status='unknown',
   1       form='formatted')

    do 105 pm = mode_start,mode_end
      do 205 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 305 k = minf,maxf,stepf
          temp = fephiv(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
305     continue
205   continue
105 continue
    close(unit=9)


      open(unit=9,file='fdata/fhru.dat',status='unknown',
   1       form='formatted')


    do 106 pm = mode_start,mode_end
      do 206 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 306 k = minf,maxf,stepf
          temp = fhru(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
306     continue
206   continue
106 continue
    close(unit=9)


      open(unit=9,file='fdata/fhrv.dat',status='unknown',
   1       form='formatted')

    do 107 pm = mode_start,mode_end
      do 207 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 307 k = minf,maxf,stepf
          temp = fhrv(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
307     continue
207   continue
107 continue
    close(unit=9)


      open(unit=9,file='fdata/fhzu.dat',status='unknown',
   1       form='formatted')


    do 108 pm = mode_start,mode_end
      do 208 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 308 k = minf,maxf,stepf
```

```
          temp = fhzu(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
308     continue
208   continue
108 continue
    close(unit=9)


      open(unit=9,file='fdata/fhzv.dat',status='unknown',
   1       form='formatted')


    do 109 pm = mode_start,mode_end
      do 209 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 309 k = minf,maxf,stepf
          temp = fhzv(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
309     continue
209   continue
109 continue
    close(unit=9)



      open(unit=9,file='fdata/fhphiu.dat',status='unknown',
   1    form='formatted')

    do 110 pm = mode_start,mode_end
      do 210 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 310 k = minf,maxf,stepf
          temp = fhphiu(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
310     continue
210   continue
110 continue
    close(unit=9)


      open(unit=9,file='fdata/fhphiv.dat',status='unknown',
   1    form='formatted')


    do 111 pm = mode_start,mode_end
      do 211 i = 1,2*mheight + rcsz2 - rcsz1 - 1
        do 311 k = minf,maxf,stepf
          temp = fhphiv(pm,i,k)
          write(9, *) dble(temp), aimag(temp)
311     continue
211   continue
111 continue
    close(unit=9)

    print *, 'Finished writing out phasors'
    return
    end
C***************************************************************C
C read out phasor values from a file to keep running sum       C
C***************************************************************C

    SUBROUTINE read_phasorsx
```

```
      implicit none
      include 'common.f'

C*****pm: the current mode being written out.

      integer pm,i,k,fi
      real*8 tempr, tempi, temprx, tempix
      complex*16, parameter :: zim = (0.0d0,1.d0)


      write(6,*) 'Reading in frequency data...'

      minf = 0
      maxf = int((high_freq-low_freq)/dfreq)
      stepf = 1

      print *,low_freq,high_freq,dfreq
      print *,minf,maxf,stepf


      print *, 'reading in freq data'


      open(unit=9,file='fdata/feru.dat',status='old',
     1     form='formatted')
      do 10 pm = mode_start,mode_end
         do 20 i = 1,2*mheight+rcsz2-rcsz1-1
            do 30 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               feru(pm,i,k) = feru(pm,i,k) + tempr + zim*tempi
 30         continue
 20      continue
 10   continue
      close(unit=9)


      open(unit=9,file='fdata/ferv.dat',status='old',
     1     form='formatted')
      do 101 pm = mode_start,mode_end
         do 201 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 301 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               ferv(pm,i,k) = ferv(pm,i,k)+ tempr + zim*tempi
 301        continue
 201     continue
 101  continue
      close(unit=9)


      open(unit=9,file='fdata/fezu.dat',status='old',
     1     form='formatted')
      do 102 pm = mode_start,mode_end
         do 202 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 302 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               fezu(pm,i,k) = fezu(pm,i,k)+ tempr + zim*tempi
 302        continue
 202     continue
 102  continue
      close(unit=9)
```

```
      open(unit=9,file='fdata/fezv.dat',status='old',
     1     form='formatted')
      do 103 pm = mode_start,mode_end
        do 203 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 303 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fezv(pm,i,k) = fezv(pm,i,k)+ tempr + zim*tempi
303       continue
203     continue
103   continue
      close(unit=9)


      open(unit=9,file='fdata/fephiu.dat',status='old',
     1     form='formatted')
      do 104 pm = mode_start,mode_end
        do 204 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 304 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fephiu(pm,i,k) = fephiu(pm,i,k) + tempr + zim*tempi
304       continue
204     continue
104   continue
      close(unit=9)


      open(unit=9,file='fdata/fephiv.dat',status='old',
     1     form='formatted')
      do 105 pm = mode_start,mode_end
        do 205 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 305 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fephiv(pm,i,k) = fephiv(pm,i,k) + tempr + zim*tempi
305       continue
205     continue
105   continue
      close(unit=9)


      open(unit=9,file='fdata/fhru.dat',status='old',
     1     form='formatted')
      do 106 pm = mode_start,mode_end
        do 206 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 306 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhru(pm,i,k) = fhru(pm,i,k) + tempr + zim*tempi
306       continue
206     continue
106   continue
      close(unit=9)


      open(unit=9,file='fdata/fhrv.dat',status='old',
     1     form='formatted')
      do 107 pm = mode_start,mode_end
        do 207 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 307 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhrv(pm,i,k) = fhrv(pm,i,k) + tempr + zim*tempi
307       continue
```

```
207      continue
107   continue
      close(unit=9)


      open(unit=9,file='fdata/fhzu.dat',status='old',
     1     form='formatted')
      do 108 pm = mode_start,mode_end
         do 208 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 308 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               fhzu(pm,i,k) = fhzu(pm,i,k) + tempr + zim*tempi
308         continue
208      continue
108   continue
      close(unit=9)


      open(unit=9,file='fdata/fhzv.dat',status='old',
     1     form='formatted')
      do 109 pm = mode_start,mode_end
         do 209 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 309 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               fhzv(pm,i,k) = fhzv(pm,i,k) + tempr + zim*tempi
309         continue
209      continue
109   continue
      close(unit=9)


      open(unit=9,file='fdata/fhphiu.dat',status='old',
     1     form='formatted')
      do 110 pm = mode_start,mode_end
         do 210 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 310 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               fhphiu(pm,i,k) = fhphiu(pm,i,k) + tempr + zim*tempi
310         continue
210      continue
110   continue
      close(unit=9)


      open(unit=9,file='fdata/fhphiv.dat',status='old',
     1     form='formatted')
      do 111 pm = mode_start,mode_end
         do 211 i = 1,2*mheight + rcsz2 - rcsz1 - 1
            do 311 k = minf,maxf,stepf
               read(9, *) tempr, tempi
               fhphiv(pm,i,k) = fhphiv(pm,i,k) + tempr + zim*tempi
311         continue
211      continue
111   continue
      close(unit=9)


      print *, 'Finished reading in old phasors for running sum'
      stepf = 1

      return
```

```
      end

C**************************************************************C
C read out phasor values from a file.                          C
C**************************************************************C

      SUBROUTINE read_phasors

      implicit none
      include 'common.f'

C*****pm: the current mode being written out.

      integer pm,i,k,fi
      real*8 tempr, tempi, temprx, tempix
      complex*16, parameter :: zim = (0.0d0,1.d0)


      write(6,*) 'Reading in frequency data...'

      minf = 0
      maxf = int((high_freq-low_freq)/dfreq)
      stepf = 1

      print *,low_freq,high_freq,dfreq
      print *,minf,maxf,stepf


      print *, 'reading in freq data'


      open(unit=9,file='fdata/feru.dat',status='old',
     1     form='formatted')
      do 10 pm = mode_start,mode_end
        do 20 i = 1,2*mheight+rcsz2-rcsz1-1
          do 30 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            feru(pm,i,k) = tempr + zim*tempi
30        continue
20      continue
10    continue
      close(unit=9)


      open(unit=9,file='fdata/ferv.dat',status='old',
     1     form='formatted')
      do 101 pm = mode_start,mode_end
        do 201 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 301 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            ferv(pm,i,k) = tempr + zim*tempi
301       continue
201     continue
101   continue
      close(unit=9)


      open(unit=9,file='fdata/fezu.dat',status='old',
     1     form='formatted')
      do 102 pm = mode_start,mode_end
        do 202 i = 1,2*mheight + rcsz2 - rcsz1 - 1
```

```
      do 302 k = minf,maxf,stepf
        read(9, *) tempr, tempi
        fezu(pm,i,k) = tempr + zim*tempi
302   continue
202   continue
102   continue
      close(unit=9)


      open(unit=9,file='fdata/fezv.dat',status='old',
     1     form='formatted')

      do 103 pm = mode_start,mode_end
        do 203 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 303 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fezv(pm,i,k) = tempr + zim*tempi
303       continue
203     continue
103   continue
      close(unit=9)


      open(unit=9,file='fdata/fephiu.dat',status='old',
     1     form='formatted')
      do 104 pm = mode_start,mode_end
        do 204 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 304 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fephiu(pm,i,k) = tempr + zim*tempi
304       continue
204     continue
104   continue
      close(unit=9)


      open(unit=9,file='fdata/fephiv.dat',status='old',
     1     form='formatted')
      do 105 pm = mode_start,mode_end
        do 205 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 305 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fephiv(pm,i,k) = tempr + zim*tempi
305       continue
205     continue
105   continue
      close(unit=9)


      open(unit=9,file='fdata/fhru.dat',status='old',
     1     form='formatted')
      do 106 pm = mode_start,mode_end
        do 206 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 306 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhru(pm,i,k) = tempr + zim*tempi
306       continue
206     continue
106   continue
      close(unit=9)
```

```
      open(unit=9,file='fdata/fhrv.dat',status='old',
     1     form='formatted')
      do 107 pm = mode_start,mode_end
        do 207 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 307 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhrv(pm,i,k) = tempr + zim*tempi
307       continue
207     continue
107   continue
      close(unit=9)


      open(unit=9,file='fdata/fhzu.dat',status='old',
     1     form='formatted')
      do 108 pm = mode_start,mode_end
        do 208 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 308 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhzu(pm,i,k) = tempr + zim*tempi
308       continue
208     continue
108   continue
      close(unit=9)
      close(unit=10)

      open(unit=9,file='fdata/fhzv.dat',status='old',
     1     form='formatted')
      do 109 pm = mode_start,mode_end
        do 209 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 309 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhzv(pm,i,k) = tempr + zim*tempi
309       continue
209     continue
109   continue
      close(unit=9)


      open(unit=9,file='fdata/fhphiu.dat',status='old',
     1     form='formatted')
      do 110 pm = mode_start,mode_end
        do 210 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 310 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhphiu(pm,i,k) = tempr + zim*tempi
310       continue
210     continue
110   continue
      close(unit=9)


      open(unit=9,file='fdata/fhphiv.dat',status='old',
     1     form='formatted')
      do 111 pm = mode_start,mode_end
        do 211 i = 1,2*mheight + rcsz2 - rcsz1 - 1
          do 311 k = minf,maxf,stepf
            read(9, *) tempr, tempi
            fhphiv(pm,i,k) = tempr + zim*tempi
311       continue
```

```
211      continue
111   continue
      close(unit=9)


      print *, 'Currently you are calculating the RCS at',maxf-minf+1


      stepf = 1


      return
      end


C***************************************************************C
C initialize all frequency field values to zero                 C
C***************************************************************C

      SUBROUTINE init_freq

      implicit none
      include 'common.f'

      integer m,k,i

      do 10 m=mode_start,mode_end
        do 20 k=1,mxdp
          do 30 i=1,MAX_FREQS
            fephiu(m,k,i)=0.0
            fephiv(m,k,i)=0.0
            feru(m,k,i)=0.0
            ferv(m,k,i)=0.0
            fezu(m,k,i)=0.0
            fezv(m,k,i)=0.0
            fhphiu(m,k,i)=0.0
            fhphiv(m,k,i)=0.0
            fhru(m,k,i)=0.0
            fhrv(m,k,i)=0.0
            fhzu(m,k,i)=0.0
            fhzv(m,k,i)=0.0
30        continue
20      continue
10    continue

      return
      end


CBZ 10/16/02 Severely modified

C***************************************************************C
C write out necessary values to calculate RCS to a file.        C
c DO THIS ONLY ONCE FOR THE FIRST MODE!                         c
C***************************************************************C

      SUBROUTINE write_values

      implicit none
      include 'common.f'

C*****pm: the current mode being written out.
```

```
      integer pm,i,k,fi
      complex*16 temp

      write(6,*) 'Writing out necessary data...'
      open(unit=9,file='rcs.info',status='unknown',
     1     form='formatted')
c variable
      write(9,*) dt
      write(9,*) dz
      write(9,*) low_freq
      write(9,*) high_freq
      write(9,*) dfreq
      write(9,*) inc_ang
c variable
      write(9,*) gd
      write(9,*) sdev
      write(9,*) modulate
      write(9,*) modfreq
      write(9,*) num_freqs
      do 130 fi=minf,maxf
        write(9,*) freqlist(fi,1), freqlist(fi,2)
 130  continue
      close(unit=9)
      return
      end


CBZ 10/24/02 Modified
C***************************************************************C
C reads in geom and time parameters from a BOR file.            C
C***************************************************************C

      SUBROUTINE read_values

      implicit none
      include 'common.f'
C*****pm: the current mode being written out.
      integer pm,i,k,fi
      real*8 tempr, tempi

      write(6,*) 'Reading in frequency data...'

      open(unit=9,file='rcs.info',status='old',
     1     form='formatted')
      read(9,*) dt
      read(9,*) dz
      read(9,*) low_freq
      read(9,*) high_freq
      read(9,*) dfreq
      read(9,*) inc_ang
      read(9,*) gd
      read(9,*) sdev
      read(9,*) modulate
      read(9,*) modfreq
      read(9,*) num_freqs
      do 130 fi=1,num_freqs
        read(9,*) freqlist(fi,1), freqlist(fi,2)
 130  continue
      close(unit=9)
```

```
      minf = 0
      maxf = int((high_freq-low_freq)/dfreq)
      stepf = 1

      print *,low_freq,high_freq,dfreq
      print *,minf,maxf,stepf

      enough_memory = .TRUE.

      if (nm.gt.mode_start) then
        write(6,*)
        print *,'nm =',nm,' is greater than the starting mode'
        print *,'number', mode_start, '.  Adjust the nm parameter'
        enough_memory = .FALSE.
      end if

      if (mm.lt.mode_end) then
        write(6,*)
        print *,'mm =',mm,' is less than the ending mode'
        print *,'number', mode_end, '.  Adjust the mm parameter'
        print *,'in the common.f file'
        enough_memory = .FALSE.
      end if

      if ((maxf-minf+1).gt.MAX_FREQS) then
        print *, 'too many frequencies, lower number of freq'
        print *, 'from ', maxf-minf+1, ' to less than ',MAX_FREQS
        print *, 'or increase MAX_FREQS variable in the common.f file.'
        enough_memory = .FALSE.
      end if

      if ((2*mheight+rcsz2-rcsz1-1).gt.mxdp) then
        print *,'error not enough memory for RCS components'
        print *,'set the parameter mxdp higher than',
     1       2*mheight+rcsz2-rcsz1-1
        enough_memory = .FALSE.
      end if

      if (.NOT.enough_memory) then
        print *,'Not enough memory, must allocate more by altering'
        print *,'parms in common.f file'
        stop
      end if

      return
      end


CBZ 10/24/02 Modified
C****************************************************************C
C read out matlab generated geometry parameters from a file.    C
C****************************************************************C

      SUBROUTINE read_parms

      implicit none
      include 'common.f'
      write(6,*) 'Reading in Matlab generated geometry data...'

      open(unit=9,file='geom.data',status='old',
     1    form='formatted')
```

```
      read(9,*) N
      read(9,*) rcsz1
      read(9,*) rcsz2
      read(9,*) rcsz
      read(9,*) mheight
c     Use the default calculations-not this
c      read(9,*) mode_start
c      read(9,*) mode_end
      close(unit=9)
      write(6,*) 'DONE Reading in geom.data. . .'
      return
      end

C**************************************************************C
C Calculate far-field E and H fields using Huygens' Principle   C
C**************************************************************C

      subroutine calc_rcs

      implicit none
      include 'common.f'

      real*8 besselj, kwave, rho, kps, cz, RCS, RCSDB
      real*8 obs_phi, obs_theta, eincsq, temp, targ
      real*8 cosp, sinp, cost, sint, sinmp, cosmp, PDIV, tempfreq
      integer pt_rB, pt_rB0, pt_z1, pt_z2, pt_rC, pt_rC0,t,phase_z
      integer pt_index, freq_index, mode_index

      real*8 dp_kwave, dutheta, tempang, dp_obs_theta
      real*8 out_freq, kps_tole

      complex*16 Escat_theta_A, Escat_theta_B, Escat_theta_C,
     1     einc(1:MAX_FREQS),
     1     At, Escat_phi_A, Escat_phi_B, Escat_phi_C, Ap, A, uniti, I1,
     2     I3, I5, c1, c2, c3, c4, c5, RCSc, RCScold, eincc
      complex*16 ferup, fervp, fephiup, fephivp, fezup, fezvp,
     1     fhrup, fhrvp, fhphiup, fhphivp, fhzup, fhzvp

      character filnam*1024, frmt*30
      integer ilen

      parameter(PDIV=1.0,uniti=(0.0d0,1.0d0),kps_tole=1.0e-7)

      write(6,*) 'Calculating RCS. . .'

      ilen = index(dbase,' ') - 1
      write(frmt,'(a2,i4,a4)') '(a',ilen,',a8)'
      write(filnam,frmt)dbase,'/rcs.dat'

      open(unit=9,file='rcs.dat',status='unknown',form='formatted')
c      open(unit=10,file='rcsold.dat',status='unknown',form='formatted')

c      open(unit=12,file='scat.dat',status='unknown',form='formatted')

C*****Some reference points to define
C*****pt_z1 index of first point of integral A
C*****pt_z2 index of last point of integral A
C*****pt_rB index of first point of integral B -left side
C*****pt_rB0 index of last point of integral B -left side
C*****pt_rC index of first point of integral C -right side
C*****pt_rC0 index of last point of integral C -right side
```

```
      pt_rB = 1
      pt_rB0 = mheight
      pt_z1 = mheight
      pt_z2 = mheight + rcsz2 - rcsz1
C*****low point (i.e. right side botom corner)
      pt_rC = 2*mheight + rcsz2 - rcsz1 - 1
C*****high point (i.e. right side top corner)
      pt_rC0 = mheight + rcsz2 - rcsz1


      print *,pt_rB,pt_rB0,pt_z1,pt_z2,pt_rC,pt_rC0

      do 1 freq_index = 1,MAX_FREQS
        einc(freq_index) = 0.0
 1    continue

C*****Calculate DFT of incident field for RCS calculation.
      do 5 t = 1,N
        targ = (t*dt-gd)+(rcsz1*dz*cos(inc_ang)+10*dz*sin(inc_ang))/c
        temp=((Ehg**2)+(Evg**2))*(1/(sqrt(2*pi)))*exp(-(targ**2.0)/
     1       ((sdev)**2.0))*((sin(2*pi*modfreq*targ))*modulate+
     2       abs(modulate-1))*5.0

        do 8 freq_index=minf,maxf,stepf
c       do 8 freq_index=1,num_freqs
c           tempfreq = low_freq + freq_index*dfreq
          tempfreq = freqlist(freq_index,1)
c           print *,tempfreq
          einc(freq_index)=einc(freq_index)+temp*exp(2*pi*uniti*
     1         tempfreq*dt*t)*dt
 8      continue
 5    continue

      do 10 freq_index=minf,maxf,stepf
c      do 10 freq_index=1,num_freqs
c        print *,minf,maxf,freq_index
        eincc = einc(freq_index)
        eincsq = (abs(einc(freq_index)))**2.0
c        kwave = ((low_freq+freq_index*dfreq)/c)*(2*pi)
        kwave = (freqlist(freq_index,1)/c)*(2*pi)
        if (calc_bist) then
          dp_kwave = kwave
          dutheta = dtheta
        else
c           print *,freq_index,mono_nang,int((freq_index-1)/
c    1           ((mono_nang+1)/2))+1
          dp_kwave = (freqlist(mono_freq_ind(int((freq_index-1)/
     1         ((mono_nang+1)/2))+1),1)/c)*(2*pi)
          tempang = dtheta*(freq_index-mono_freq_ind(int((freq_index
     1         -1)/((mono_nang+1)/2))+1))
          low_theta = dble(inc_ang/pi*180-tempang*2)
          high_theta = dble(inc_ang/pi*180+tempang*2)
          dutheta = high_theta-low_theta
          if (abs(dutheta).lt.eps) dutheta = 1.0
c           print *,dtheta,tempang,low_theta,high_theta,
c    1           (inc_ang/pi*180+low_theta)/2.,
c    2           (inc_ang/pi*180+high_theta)/2.
        end if

        do 20 obs_phi=low_phi,high_phi,dphi
```

```
      sinp = sin(obs_phi/180*pi)
      cosp = cos(obs_phi/180*pi)

      do 30 obs_theta=low_theta,high_theta,dutheta
        if (calc_bist) then
          dp_obs_theta = obs_theta
        else
          dp_obs_theta = (inc_ang/pi*180+obs_theta)/2.0
        end if
        sint = sin(obs_theta/180*pi)
        cost = cos(obs_theta/180*pi)

C*************Initialize integral values

        Escat_theta_A = 0.0
        Escat_phi_A = 0.0
        Escat_theta_B = 0.0
        Escat_phi_B = 0.0
        Escat_theta_C = 0.0
        Escat_phi_C = 0.0

        do 40 mode_index = nm,mm
          sinmp = sin(mode_index*obs_phi/180*pi)
          cosmp = cos(mode_index*obs_phi/180*pi)

C****************Three different integrals to evaluate
          c3 = 2*pi*exp(uniti*mode_index*1.5*pi)
          c4 = 2*pi*exp(uniti*(mode_index+1)*1.5*pi)

C****************Integral A: z1 -> z2 -center integral at r0
          rho = (mheight - 1) * dz
          kps = kwave * rho * sint

            if (abs(kps).lt.kps_tole) then
              if (mode_index.eq.1) then
                I1=0.0
                I3=pi
                I5=pi
              else
                I1=0.0
                I3=0.0
                I5=0.0
              end if
              if (mode_index.eq.0) then
                I1=2*pi
              end if
            else
              c2 = 2.0*pi*uniti*mode_index/kps
              c5 = c2*exp(uniti*mode_index*1.5*pi)
              I1 = c3*besselj(kps,mode_index)
              I3 = c4*besselj(kps,mode_index+1)+c5*
     1             besselj(kps,mode_index)
              I5 = c5*besselj(kps,mode_index)
            end if

          do 50 pt_index = pt_z1,pt_z2
            ferup = feru(mode_index,pt_index,freq_index)*cosmp
     1           +ferv(mode_index,pt_index,freq_index)*sinmp
            fervp = ferv(mode_index,pt_index,freq_index)*cosmp
     1           -feru(mode_index,pt_index,freq_index)*sinmp
            fezup = fezu(mode_index,pt_index,freq_index)*cosmp
```

```
     1          +fezv(mode_index,pt_index,freq_index)*sinmp
              fezvp = fezv(mode_index,pt_index,freq_index)*cosmp
     1          -fezu(mode_index,pt_index,freq_index)*sinmp
              fephiup = fephiu(mode_index,pt_index,freq_index)*
     1          cosmp+fephiv(mode_index,pt_index,freq_index)*
     1          sinmp
              fephivp = fephiv(mode_index,pt_index,freq_index)*
     1          cosmp-fephiu(mode_index,pt_index,freq_index)*
     1          sinmp
              fhrup = fhru(mode_index,pt_index,freq_index)*cosmp
     1          +fhrv(mode_index,pt_index,freq_index)*sinmp
              fhrvp = fhrv(mode_index,pt_index,freq_index)*cosmp
     1          -fhru(mode_index,pt_index,freq_index)*sinmp
              fhzup = fhzu(mode_index,pt_index,freq_index)*cosmp
     1          +fhzv(mode_index,pt_index,freq_index)*sinmp
              fhzvp = fhzv(mode_index,pt_index,freq_index)*cosmp
     1          -fhzu(mode_index,pt_index,freq_index)*sinmp
              fhphiup = fhphiu(mode_index,pt_index,freq_index)*
     1          cosmp+fhphiv(mode_index,pt_index,freq_index)*
     1          sinmp
              fhphivp = fhphiv(mode_index,pt_index,freq_index)*
     1          cosmp-fhphiu(mode_index,pt_index,freq_index)*
     1          sinmp
              do 55 phase_z = 0,(int(PDIV)-1)
                cz = (rcsz1+pt_index-pt_z1)*dz+phase_z*dz/PDIV
                c1 = exp(-uniti*kwave*cz*cost)

                Escat_theta_A = (dz/PDIV)*rho*c1*(-sint*fhphiup
     1              *c3*besselj(kps, mode_index)+fezup*I3+
     2              cost*fhzvp*I5)+Escat_theta_A

                Escat_phi_A = (dz/PDIV)*rho*c1*(-fhzup*I3-sint*
     1              fephiup*c3*besselj(kps,mode_index)+cost*
     2              fezvp*I5)+Escat_phi_A

55              continue
50            continue
C*********************************************************

C*****************Integral B: 0 -> r0 -left integral at z1

              cz = rcsz1*dz
              c1 = exp(-uniti*kwave*cz*cost)

              do 60 pt_index = pt_rB, pt_rB0
                ferup = feru(mode_index,pt_index,freq_index)*cosmp
     1              +ferv(mode_index,pt_index,freq_index)*sinmp
                fervp = ferv(mode_index,pt_index,freq_index)*cosmp
     1              -feru(mode_index,pt_index,freq_index)*sinmp
                fezup = fezu(mode_index,pt_index,freq_index)*cosmp
     1              +fezv(mode_index,pt_index,freq_index)*sinmp
                fezvp = fezv(mode_index,pt_index,freq_index)*cosmp
     1              -fezu(mode_index,pt_index,freq_index)*sinmp
                fephiup = fephiu(mode_index,pt_index,freq_index)*
     1              cosmp+fephiv(mode_index,pt_index,freq_index)*
     1              sinmp
                fephivp = fephiv(mode_index,pt_index,freq_index)*
     1              cosmp-fephiu(mode_index,pt_index,freq_index)*
     1              sinmp
                fhrup = fhru(mode_index,pt_index,freq_index)*cosmp
     1              +fhrv(mode_index,pt_index,freq_index)*sinmp
```

```
            fhrvp = fhrv(mode_index,pt_index,freq_index)*cosmp
     1          -fhru(mode_index,pt_index,freq_index)*sinmp
            fhzup = fhzu(mode_index,pt_index,freq_index)*cosmp
     1          +fhzv(mode_index,pt_index,freq_index)*sinmp
            fhzvp = fhzv(mode_index,pt_index,freq_index)*cosmp
     1          -fhzu(mode_index,pt_index,freq_index)*sinmp
            fhphiup = fhphiu(mode_index,pt_index,freq_index)*
     1          cosmp+fhphiv(mode_index,pt_index,freq_index)*
     1          sinmp
            fhphivp = fhphiv(mode_index,pt_index,freq_index)*
     1          cosmp-fhphiu(mode_index,pt_index,freq_index)*
     1          sinmp

            rho = (pt_index-1)*dz
            kps = kwave * rho * sint
c            print *,obs_theta,kps,sint

            if (abs(kps).lt.kps_tole) then
              if (mode_index.eq.1) then
                I1=0.0
                I3=pi
                I5=pi
              else
                I1=0.0
                I3=0.0
                I5=0.0
              end if
              if (mode_index.eq.0) then
                I1=2*pi
              end if
            else
              c2 = 2.0*pi*uniti*mode_index/kps
              c5 = c2*exp(uniti*mode_index*1.5*pi)
              I1 = c3*besselj(kps,mode_index)
              I3 = c4*besselj(kps,mode_index+1)+c5*
     1            besselj(kps,mode_index)
              I5 = c5*besselj(kps,mode_index)
            end if

            Escat_theta_B = -dz*rho*c1*(-cost*fhphiup*I3-ferup
     1          *I3-cost*fhrvp*I5+fephivp*I5)+Escat_theta_B

            Escat_phi_B = -dz*rho*c1*((fhrup-cost*fephiup)*I3+
     1          (-fhphivp-cost*fervp)*I5)+Escat_phi_B

 60       continue
C*********************************************************

C****************Integral C: 0 -> r0 -right integral at z2

          cz = rcsz2*dz
          c1 = exp(-uniti*kwave*cz*cost)

          do 70 pt_index = pt_rC0, pt_rC
            ferup = feru(mode_index,pt_index,freq_index)*cosmp
     1          +ferv(mode_index,pt_index,freq_index)*sinmp
            fervp = ferv(mode_index,pt_index,freq_index)*cosmp
     1          -feru(mode_index,pt_index,freq_index)*sinmp
            fezup = fezu(mode_index,pt_index,freq_index)*cosmp
     1          +fezv(mode_index,pt_index,freq_index)*sinmp
            fezvp = fezv(mode_index,pt_index,freq_index)*cosmp
```

```
     1            -fezu(mode_index,pt_index,freq_index)*sinmp
              fephiup = fephiu(mode_index,pt_index,freq_index)*
     1            cosmp+fephiv(mode_index,pt_index,freq_index)*
     1            sinmp
              fephivp = fephiv(mode_index,pt_index,freq_index)*
     1            cosmp-fephiu(mode_index,pt_index,freq_index)*
     1            sinmp
              fhrup = fhru(mode_index,pt_index,freq_index)*cosmp
     1            +fhrv(mode_index,pt_index,freq_index)*sinmp
              fhrvp = fhrv(mode_index,pt_index,freq_index)*cosmp
     1            -fhru(mode_index,pt_index,freq_index)*sinmp
              fhzup = fhzu(mode_index,pt_index,freq_index)*cosmp
     1            +fhzv(mode_index,pt_index,freq_index)*sinmp
              fhzvp = fhzv(mode_index,pt_index,freq_index)*cosmp
     1            -fhzu(mode_index,pt_index,freq_index)*sinmp
              fhphiup = fhphiu(mode_index,pt_index,freq_index)*
     1            cosmp+fhphiv(mode_index,pt_index,freq_index)*
     1            sinmp
              fhphivp = fhphiv(mode_index,pt_index,freq_index)*
     1            cosmp-fhphiu(mode_index,pt_index,freq_index)*
     1            sinmp
              rho = (pt_rC-pt_index)*dz
              kps = kwave * rho * sint

              if (abs(kps).lt.kps_tole) then
                if (mode_index.eq.1) then
                  I1=0.0
                  I3=pi
                  I5=pi
                else
                  I1=0.0
                  I3=0.0
                  I5=0.0
                end if
                if (mode_index.eq.0) then
                  I1=2*pi
                end if
              else
                c2 = 2.0*pi*uniti*mode_index/kps
                c5 = c2*exp(uniti*mode_index*1.5*pi)
                I1 = c3*besselj(kps,mode_index)
                I3 = c4*besselj(kps,mode_index+1)+c5*
     1              besselj(kps,mode_index)
                I5 = c5*besselj(kps,mode_index)
              end if

              Escat_theta_C = dz*rho*c1*(-cost*fhphiup*I3-ferup*
     1            I3-cost*fhrvp*I5+fephivp*I5)+Escat_theta_C

              Escat_phi_C = dz*rho*c1*((fhrup-cost*fephiup)*I3+
     3            (-fhphivp-cost*fervp)*I5)+Escat_phi_C

70          continue
C**********************************************************

40        continue

          At = Escat_theta_A + Escat_theta_B + Escat_theta_C

          Ap = Escat_phi_A + Escat_phi_B + Escat_phi_C
```

```
          A = At*((cost*cost*cosp+sint*sint)*Ehg+(cost*sinp)*Evg)
     1          +Ap*((-cost*sinp)*Ehg+cosp*Evg)

          RCScold = ((kwave**2)*(A**2))/(4.0*pi*eincc**2)
c          RCSc = kwave*A/(4.0*pi*eincc)
          RCSc = (kwave*A)/(sqrt(4.0*pi)*eincc)

          RCS = ((kwave**2)*((abs(A))**2))/(4.0*pi*eincsq)
          if (abs(RCS).lt.1e-7) then
            RCSDB = -200.0
          else
            RCSDB = 10*LOG10(RCS)
          end if

c          write(10,*) dp_kwave,obs_phi,dp_obs_theta,RCSDB,
c    1          abs(RCScold),atan2(imag(RCScold),dble(RCScold))

c          out_freq = (dp_kwave/(2*pi))*c
          out_freq = freqlist(freq_index,1)

          write(9,*) out_freq, dble(RCSc), imag(RCSc)

 30       continue
 20     continue
 10   continue

 99   format(F25.15,'   ',F25.15)

      close(unit=9)
c      close(unit=10)
c      close(unit=12)
      return
      end
```

A.4 PML Calculations

Berenger's Perfectly Matched Layer is implemented in this portion of the program.

```
c***************************************************************c
c PML Equations: right, left, top                               c
c***************************************************************c
c E fields                                                      c
c***************************************************************c

      subroutine pmlEeqn(m,ms)

      implicit none
      include 'common.f'

      integer k,i,m,axis,ms
      real*8 c1,c2,c3,c4,c5,c6
      real*8 sigma_r,sigma_z
      axis=1
C *************Calculate Erz fields*********************

c     real region interface
```

```fortran
      c1=dt/(eps*dz)
      do 10 i=1,pmldepth
        do 20 k=1,maxz-1

C ****CENTER TOP REGION
          erzt(k,i)=erzt(k,i)+(1/eta)*(c1*(hphizt(k,i)+hphirt(k,i)
     1         -hphizt(k+1,i)-hphirt(k+1,i)))
20      continue

        erzt(maxz,i)=erzt(maxz,i)+(1/eta)*(c1*(hphizt(maxz,i)+hphirt
     1      (maxz,i)-hphizr(1,i+maxr)-hphirr(1,i+maxr)))
10    continue

      do 30 i=1,pmldepth+maxr
        do 40 k=1,pmldepth
          sigma_z=sigma_max*((k+0.0)/pmldepth)**2.0
          c1=exp(-sigma_z*dt/eps)
          c2=(c1-1.0)/(sigma_z*dz)

C **********Right Side*********
          erzr(k,i)=c1*erzr(k,i)+(1/eta)*(-c2*(hphizr(k,i)+hphirr
     1       (k,i)-hphizr(k+1,i)-hphirr(k+1,i)))
C **********Left Side**********reminder:k=right.left=pmldepth.1
          if (k.eq.1) THEN
            if (i.gt.maxr) THEN
              c3=hphizt(1,i-maxr)+hphirt(1,i-maxr)
            ELSE
              c3=hphi(1,i)
            END IF
            erzl(k,i)=c1*erzl(k,i)+(1/eta)*(-c2*(hphizl(k,i)+hphirl
     1         (k,i)-c3))
          ELSE
            erzl(k,i)=c1*erzl(k,i)+(1/eta)*(-c2*(hphizl(k,i)+hphirl
     1         (k,i)-hphizl(k-1,i)-hphirl(k-1,i)))
          END IF

C ****Left Side: interior of case1
          if (((case_id.eq.1).or.(case_id.eq.5))
     1       .and.(i.le.left_y)) then
c     if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (k.eq.1) THEN
              if (case_id.eq.1) then
                c3=hphi(1,i)
              else if (case_id.eq.5) then
                c3=hphi(1,i)
              end if
              erzlx(k,i)=c1*erzlx(k,i)+(1/eta)
     1           *(-c2*(hphizlx(k,i)+hphirlx(k,i)-c3))
            ELSE
              erzlx(k,i)=c1*erzlx(k,i)+(1/eta)
     1           *(-c2*(hphizlx(k,i)+hphirlx(k,i) -
     1           hphizlx(k-1,i)-hphirlx(k-1,i)))
            END IF
          end if


40      continue
30    continue

C **************Calculate Erphi fields**********************c
```

```
C ****CENTER BOTTOM & TOP REGIONS

      do 41 k=1,maxz
        do 42 i=1,pmldepth
          c4=(m*dt/eps)/((i+0.5-1.0+maxr)*dz)
          erphit(k,i)=erphit(k,i)-(1.0/eta)*(c4*(hzrt(k,i)+
     1         hzphit(k,i)))
 42     continue
 41   continue

C ****RIGHT & LEFT SIDES

      do 46 k=1,pmldepth
        do 47 i=1,pmldepth+maxr
          c4=(m*dt/eps)/((i+0.5-1.0)*dz)
          erphil(k,i)=erphil(k,i)-(c4*(hzrl(k,i)+hzphil(k,i)))/eta
          erphir(k,i)=erphir(k,i)-(c4*(hzrr(k,i)+hzphir(k,i)))/eta

          if (((case_id.eq.1).or.(case_id.eq.5))
     1        .and.(i.le.left_y)) then
c          if ((case_id.eq.1).or.(case_id.eq.5)) then
            erphilx(k,i)=erphilx(k,i)-
     1           (c4*(hzrlx(k,i)+hzphilx(k,i)))/eta
          end if



 47     continue
 46   continue

C **************Calculate Ephiz fields**********************c

C ****CENTER BOTTOM & TOP REGIONS

      c1=dt/(eps*dz)
      do 50 i=1,pmldepth
        do 60 k=1,maxz-1
          ephizt(k,i)=ephizt(k,i)+(c1*(hrzt(k+1,i)+hrphit(k+1,i)-
     1         hrzt(k,i)-hrphit(k,i)))/eta
 60     continue

        ephizt(maxz,i)=ephizt(maxz,i)+(c1*(hrzr(1,i+maxr)+hrphir
     1       (1,i+maxr)-hrzt(k,i)-hrphit(k,i)))/eta
 50   continue

      do 70 k=1,pmldepth
        sigma_z=sigma_max*((k+0.0+0.5)/pmldepth)**2.0
        c1=exp(-sigma_z*dt/eps)
        c2=(c1-1.0)/(sigma_z*dz)
c        print *,'sigma_z',c1,c2
        do 80 i=1,pmldepth+maxr

C **********Right Side*********
          if (abs(m).ne.1.AND.i.eq.axis) THEN
            ephizr(k,i)=0.0
          ELSE
            ephizr(k,i)=c1*ephizr(k,i)-(c2*(hrzr(k+1,i)+hrphir
     1           (k+1,i)-hrzr(k,i)-hrphir(k,i)))/eta
          END IF
```

```
      if (((case_id.eq.2).or.(case_id.eq.4).or.(case_id.eq.3))
     1       .and.(i.eq.right_y)) then
c     create artificial PEC in the PML RIGHT
        ephizr(k,i)=0.0
      end if


C **********Left Side**********

      if (k.eq.1) THEN
        if (i.gt.maxr) THEN
          c3=hrzt(1,i-maxr)+hrphit(1,i-maxr)
        ELSE
          c3=hr(1,i)
        END IF

        if (abs(m).ne.1.AND.i.eq.axis) THEN
          ephizl(k,i)=0.0
        ELSE
          ephizl(k,i)=c1*ephizl(k,i)-(c2*(c3-hrzl(k,i)-
     1          hrphil(k,i)))/eta
        END IF

      ELSE
        if (abs(m).ne.1.AND.i.eq.axis) THEN
          ephizl(k,i)=0.0
        ELSE
          ephizl(k,i)=c1*ephizl(k,i)-(c2*(hrzl(k-1,i)+
     1          hrphil(k-1,i)-hrzl(k,i)-hrphil(k,i)))/eta
        END IF

      END IF


c     Interior PML:
      if (((case_id.eq.1).or.(case_id.eq.5))
     1       .and.(i.le.left_y)) then
c       if ((case_id.eq.1).or.(case_id.eq.5)) then
        if (k.eq.1) THEN
          if (case_id.eq.1) then
            c3=hr(1,i)
          else if (case_id.eq.5) then
            c3=hr(1,i)
          end if

          if (abs(m).ne.1.AND.i.eq.axis) THEN
            ephizlx(k,i)=0.0
          ELSE
            ephizlx(k,i)=c1*ephizlx(k,i)-(c2*(c3-hrzlx(k,i)-
     1            hrphilx(k,i)))/eta
          END IF
        ELSE
          if (abs(m).ne.1.AND.i.eq.axis) THEN
            ephizlx(k,i)=0.0
          ELSE
            ephizlx(k,i)=c1*ephizlx(k,i)-(c2*(hrzlx(k-1,i)+
     1            hrphilx(k-1,i)-hrzlx(k,i)-hrphilx(k,i)))/eta
          END IF
        END IF
      end if
```

```
          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c create artificial PEC in the PML LEFT
            ephizlx(k,i)=0.0
          end if

          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.high_y)) then
c create artificial PEC in the PML LEFT
            ephizl(k,i)=0.0
          end if

          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c create artificial PEC in the PML LEFT
            ephizl(k,i)=0.0
          end if

          if (((case_id.eq.2).or.(case_id.eq.4)).and.
     1        (i.eq.left_y)) then
c create artificial PEC in the PML LEFT
            ephizl(k,i)=0.0
          end if

 80     continue
 70   continue

C **************Calculate Ephir fields**********************c

C ****Sigma_r region

      do 90 i=1,pmldepth
          sigma_r=sigma_max*((i+0.0)/pmldepth)**2.0
          c1=exp(-sigma_r*dt/eps)
          c2=(c1-1.0)/(sigma_r*dz)

c top center region
        do 100 k=1,maxz
          if (i.eq.1) THEN
            ephirt(k,i)=c1*ephirt(k,i)-(c2*(hz(k,maxr)-hzrt
     1         (k,i)-hzphit(k,i)))/eta
          ELSE
            ephirt(k,i)=c1*ephirt(k,i)-(c2*(hzrt(k,i-1)+hzphit
     1         (k,i-1)-hzrt(k,i)-hzphit(k,i)))/eta
          END IF
 100    continue

c right and left top regions
        do 110 k=1,pmldepth
          ephirr(k,i+maxr)=c1*ephirr(k,i+maxr)-(c2*(hzrr
     1        (k,i-1+maxr)+hzphir(k,i-1+maxr)-hzrr(k,i+maxr)-
     2        hzphir(k,i+maxr)))/eta
          ephirl(k,i+maxr)=c1*ephirl(k,i+maxr)-(c2*(hzrl
     1        (k,i-1+maxr)+hzphil(k,i-1+maxr)-hzrl(k,i+maxr)-
     2        hzphil(k,i+maxr)))/eta
 110    continue
 90   continue

C ****Right and Left Center Regions (no sigmas!)
```

```
      c5=dt/(eps*dz)
      do 120 k=1,pmldepth
        do 130 i=1,maxr
          if (i.eq.1) THEN
            c4=0.0
            c3=0.0
          ELSE
            c4=hzrr(k,i-1)+hzphir(k,i-1)
            c3=hzrl(k,i-1)+hzphil(k,i-1)
          END IF

          if (i.eq.axis) THEN
            if (abs(m).ne.1) THEN
              ephirr(k,i)=0.0
              ephirl(k,i)=0.0
            ELSE
              c6=2*dt/(eps*dz)
              ephirr(k,i)=ephirr(k,i)-(c6*(hzphir(k,i)+
     1            hzrr(k,i)))/eta
              ephirl(k,i)=ephirl(k,i)-(c6*(hzphil(k,i)+
     1            hzrl(k,i)))/eta
            END IF
          ELSE
            ephirr(k,i)=ephirr(k,i)+(c5*(c4-hzrr(k,i)-
     1          hzphir(k,i)))/eta
            ephirl(k,i)=ephirl(k,i)+(c5*(c3-hzrl(k,i)-
     1          hzphil(k,i)))/eta
          END IF



          if (((case_id.eq.1).or.(case_id.eq.5))
     1          .and.(i.le.left_y)) then
c           if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (i.eq.1) THEN
              c3=0.0
            ELSE
              c3=hzrlx(k,i-1)+hzphilx(k,i-1)
            END IF

            if (i.eq.axis) THEN
              if (abs(m).ne.1) THEN
                ephirlx(k,i)=0.0
              ELSE
                c6=2*dt/(eps*dz)
                ephirlx(k,i)=ephirlx(k,i)-(c6*(hzphilx(k,i)+
     1              hzrlx(k,i)))/eta
              END IF
            ELSE
              ephirlx(k,i)=ephirlx(k,i)+(c5*(c3-hzrlx(k,i)-
     1          hzphilx(k,i)))/eta
            END IF
          end if


          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c     create artificial PEC in the PML LEFT
            ephirlx(k,i)=0.0
          end if
```

```
         if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.high_y)) then
c    create artificial PEC in the PML LEFT
           ephirl(k,i)=0.0
         end if

         if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c    create artificial PEC in the PML LEFT
           ephirl(k,i)=0.0
         end if

         if (((case_id.eq.2).or.(case_id.eq.4)).and.
     1        (i.eq.left_y)) then
c    create artificial PEC in the PML LEFT
           ephirl(k,i)=0.0
         end if

         if (((case_id.eq.2).or.(case_id.eq.4).or.(case_id.eq.3))
     1        .and.(i.eq.right_y)) then
c    create artificial PEC in the PML RIGHT
           ephirr(k,i)=0.0
         end if
 130    continue
 120  continue

C **************Calculate Ezr fields**********************c

C ****Calculate TOP(right,left,center) REGIONS, ie sigma_r regions

      do 140 i=1,pmldepth
        sigma_r=sigma_max*((i+0.0)/pmldepth)**2.0
        c1=exp(-sigma_r*dt/eps)
        c2=(c1-1.0)/(sigma_r*dz)/(i+maxr-1.0)

c middle region

        do 150 k=1,maxz
          if (i.eq.1) THEN
            ezrt(k,i)=c1*ezrt(k,i)-(c2/eta)*((i-0.5+maxr)*
     1         (hphizt(k,i)+hphirt(k,i))-(i-1.5+maxr)*hphi(k,maxr))
          ELSE
            ezrt(k,i)=c1*ezrt(k,i)-(c2/eta)*((i-0.5+maxr)*
     1         (hphizt(k,i)+hphirt(k,i))-(i+maxr-1.5)*
     2         (hphizt(k,i-1)+hphirt(k,i-1)))
          END IF

 150    continue

        do 160 k=1,pmldepth
          ezrl(k,i+maxr)=c1*ezrl(k,i+maxr)-(c2/eta)*((i-0.5+maxr)*
     1       (hphizl(k,i+maxr)+hphirl(k,i+maxr))-(i+maxr-1.5)*
     2       (hphizl(k,i-1+maxr)+hphirl(k,i-1+maxr)))

          ezrr(k,i+maxr)=c1*ezrr(k,i+maxr)-(c2/eta)*((i-0.5+maxr)*
     1       (hphizr(k,i+maxr)+hphirr(k,i+maxr))-(i+maxr-1.5)*
     2       (hphizr(k,i-1+maxr)+hphirr(k,i-1+maxr)))

 160    continue
 140  continue
```

```
C ****Right and Left Center Regions (no sigmas!)

      do 125 i=1,maxr
        do 135 k=1,pmldepth
          if (i.eq.axis) THEN
            if (abs(m).eq.0) THEN
              c4=4*dt/(eps*dz)
              ezrl(k,i)=ezrl(k,i)+(c4/eta)*(hphirl(k,i)+hphizl(k,i))
              ezrr(k,i)=ezrr(k,i)+(c4/eta)*(hphirr(k,i)+hphizr(k,i))
            ELSE
              ezrl(k,i)=0.0
              ezrr(k,i)=0.0
            END IF
          ELSE
            c5=dt*(i+0.5-1.0)/((i+0.0-1.0)*dz*eps)
            c6=dt*(i-0.5-1.0)/((i+0.0-1.0)*dz*eps)
            ezrl(k,i)=ezrl(k,i)+(c5/eta)*(hphirl(k,i)+hphizl(k,i))
     1          -(c6/eta)*(hphirl(k,i-1)+hphizl(k,i-1))
            ezrr(k,i)=ezrr(k,i)+(c5/eta)*(hphirr(k,i)+hphizr(k,i))
     1          -(c6/eta)*(hphirr(k,i-1)+hphizr(k,i-1))
          END IF


          if (((case_id.eq.1).or.(case_id.eq.5))
     1        .and.(i.le.left_y)) then
c           if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (i.eq.axis) THEN
              if (abs(m).eq.0) THEN
                c4=4*dt/(eps*dz)
                ezrlx(k,i)=ezrlx(k,i)+(c4/eta)*
     1               (hphirlx(k,i)+hphizlx(k,i))
              ELSE
                ezrlx(k,i)=0.0
              END IF

            ELSE
              c5=dt*(i+0.5-1.0)/((i+0.0-1.0)*dz*eps)
              c6=dt*(i-0.5-1.0)/((i+0.0-1.0)*dz*eps)
              ezrlx(k,i)=ezrlx(k,i)+(c5/eta)*
     1           (hphirlx(k,i)+hphizlx(k,i))
     1           -(c6/eta)*(hphirlx(k,i-1)+hphizlx(k,i-1))
            END IF
          end if


          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c     create artificial PEC in the PML LEFT
            ezrlx(k,i)=0.0
          end if

          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.high_y)) then
c     create artificial PEC in the PML LEFT
            ezrl(k,i)=0.0
          end if

          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1        (i.eq.left_y)) then
c     create artificial PEC in the PML LEFT
```

```
            ezrl(k,i)=0.0
          end if

          if (((case_id.eq.2).or.(case_id.eq.4)).and.
     1         (i.eq.left_y)) then
c     create artificial PEC in the PML LEFT
            ezrl(k,i)=0.0
          end if

          if (((case_id.eq.2).or.(case_id.eq.4).or.(case_id.eq.3))
     1         .and.(i.eq.right_y)) then
c     create artificial PEC in the PML RIGHT
            ezrr(k,i)=0.0
          end if

 135    continue
 125  continue

C **************Calculate Ezphi fields**********************c

C ***TOP PML

      do 170 i=1,pmldepth
        c1=m*dt/(eps*(i+0.0+maxr-1.0)*dz)
        do 180 k=1,maxz
          ezphit(k,i)=ezphit(k,i)+(c1/eta)*(hrphit(k,i)+hrzt(k,i))
 180    continue
 170  continue

C ***Right/Left PML

      do 190 i=1,pmldepth+maxr
        do 200 k=1,pmldepth
          if (i.eq.axis) THEN
            ezphir(k,i)=0.0
            ezphil(k,i)=0.0
          ELSE
            c1=m*dt/(eps*(i+0.0-1.0)*dz)
            ezphir(k,i)=ezphir(k,i)+(c1/eta)*(hrphir(k,i)+hrzr(k,i))
            ezphil(k,i)=ezphil(k,i)+(c1/eta)*(hrphil(k,i)+hrzl(k,i))
          END IF

           if (((case_id.eq.1).or.(case_id.eq.5))
     1         .and.(i.le.left_y)) then
c          if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (i.eq.axis) THEN
              ezphilx(k,i)=0.0
            ELSE
              c1=m*dt/(eps*(i+0.0-1.0)*dz)
              ezphilx(k,i)=ezphilx(k,i)+(c1/eta)*
     1             (hrphilx(k,i)+hrzlx(k,i))
            END IF
          end if


          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1         (i.eq.left_y)) then
c     create artificial PEC in the PML
            ezphilx(k,i)=0.0
          end if
```

```
          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1          (i.eq.high_y)) then
c       create artificial PEC in the PML LEFT
            ezphil(k,i)=0.0
          end if

          if (((case_id.eq.1).or.(case_id.eq.5)).and.
     1          (i.eq.left_y)) then
c       create artificial PEC in the PML
            ezphil(k,i)=0.0
          end if


          if (((case_id.eq.2).or.(case_id.eq.4)).and.
     1          (i.eq.left_y)) then
c       create artificial PEC in the PML LEFT
            ezphil(k,i)=0.0
          end if

          if (((case_id.eq.2).or.(case_id.eq.4).or.(case_id.eq.3))
     1          .and.(i.eq.right_y)) then
c       create artificial PEC in the PML RIGHT
            ezphir(k,i)=0.0
          end if


 200    continue
 190  continue

      return
      end

c****************************************************************c
c H fields                                                      c
c****************************************************************c

      subroutine pmlHeqn(m,ms)

      implicit none
      include 'common.f'

      integer k,i,m,axis,ms
      real*8 c1,c2,c3,c4,c5,c6
      real*8 sigma_r,sigma_rs,sigma_z,sigma_zs
      axis=1

C **************Calculate Hrz fields**********************c

C ****The Right & Left Reigions of PML
      do 210 k=1,pmldepth
        sigma_z=sigma_max*((k+0.0)/pmldepth)**2.0
        sigma_zs=sigma_z*(mu/eps)
        c1=exp(-sigma_zs*dt/mu)
        c2=(c1-1.0)/(sigma_zs*dz)
c         print *,'sigma_zs',c1,c2

        do 220 i=1,pmldepth+maxr
          if (k.eq.1) THEN
            if (i.gt.maxr) THEN
              hrzr(k,i)=c1*hrzr(k,i)-eta*c2*(ephizr(k,i)+ephirr
```

```
     1                (k,i)-ephirt(maxz,i-maxr)-ephizt(maxz,i-maxr))
            ELSE
              if (i.eq.axis.AND.abs(m).ne.1) THEN
                hrzr(k,i)=0.0
              ELSE
                hrzr(k,i)=c1*hrzr(k,i)-eta*c2*(ephizr(k,i)+ephirr
     1                (k,i)-ephi(maxz,i))
              END IF
            END IF
          ELSE
            if (i.eq.axis.AND.abs(m).ne.1) THEN
              hrzr(k,i)=0.0
            ELSE
              hrzr(k,i)=c1*hrzr(k,i)-eta*c2*(ephizr(k,i)+ephirr
     1             (k,i)-ephizr(k-1,i)-ephirr(k-1,i))
            END IF
          END IF

          if (i.eq.axis.AND.abs(m).ne.1) THEN
            hrzl(k,i)=0.0
          ELSE
            hrzl(k,i)=c1*hrzl(k,i)-eta*c2*(ephizl(k,i)+ephirl(k,i)-
     1           ephizl(k+1,i)-ephirl(k+1,i))
          END IF


          if (((case_id.eq.1).or.(case_id.eq.5))
     1         .and.(i.le.left_y)) then
c          if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (i.eq.axis.AND.abs(m).ne.1) THEN
              hrzlx(k,i)=0.0
            ELSE
              hrzlx(k,i)=c1*hrzlx(k,i)-eta*c2*(ephizlx(k,i)
     1             +ephirlx(k,i)-ephizlx(k+1,i)-ephirlx(k+1,i))
            end if
          END IF

 220    continue
 210  continue

C ****The Up/Down Center Region PML

      c5=dt/(mu*dz)

      do 230 k=1,maxz
        do 240 i=1,pmldepth
          if (k.eq.1) THEN
            hrzt(k,i)=hrzt(k,i)+eta*c5*(ephizt(k,i)+ephirt(k,i)-
     1           ephizl(1,i+maxr)-ephirl(1,i+maxr))
          ELSE
            hrzt(k,i)=hrzt(k,i)+eta*c5*(ephizt(k,i)+ephirt(k,i)-
     1           ephizt(k-1,i)-ephirt(k-1,i))
          END IF

 240    continue
 230  continue

C **************Calculate Hrphi fields**********************c

C ****The Right/Left Regions PML
```

```
      do 250 i=1,maxr+pmldepth
        do 260 k=1,pmldepth
          if (i.ne.axis) THEN
            c1=m*dt/(mu*(i+0.0-1.0)*dz)
            hrphir(k,i)=hrphir(k,i)-eta*c1*(ezphir(k,i)+ezrr(k,i))
            hrphil(k,i)=hrphil(k,i)-eta*c1*(ezphil(k,i)+ezrl(k,i))
          ELSE
            if (abs(m).ne.1) THEN
              hrphir(k,i)=0.0
              hrphil(k,i)=0.0
            ELSE
              c6=dt/(mu*dz)
              hrphir(k,i)=hrphir(k,i)+eta*ms*c6*(ezphir(k,i+1)+
     1              ezrr(k,i+1))
              hrphil(k,i)=hrphil(k,i)+eta*ms*c6*(ezphil(k,i+1)+
     1              ezrl(k,i+1))
            END IF
          END IF


          if (((case_id.eq.1).or.(case_id.eq.5))
     1          .and.(i.le.left_y)) then
c          if ((case_id.eq.1).or.(case_id.eq.5)) then
            if (i.ne.axis) THEN
              c1=m*dt/(mu*(i+0.0-1.0)*dz)
              hrphilx(k,i)=hrphilx(k,i)-eta*c1*
     1              (ezphilx(k,i)+ezrlx(k,i))
            ELSE
              if (abs(m).ne.1) THEN
                hrphilx(k,i)=0.0
              ELSE
                c6=dt/(mu*dz)
                hrphilx(k,i)=hrphilx(k,i)+eta*ms*c6*
     1              (ezphilx(k,i+1)+ezrlx(k,i+1))
              END IF
            END IF
          END IF

260     continue
250   continue

C ****The Up/Down Regions PML

      do 270 i=1,pmldepth
        c1=m*dt/(mu*(i+maxr+0.0-1.0)*dz)
        do 280 k=1,maxz
          hrphit(k,i)=hrphit(k,i)-eta*c1*(ezphit(k,i)+ezrt(k,i))
280     continue
270   continue

C **************Calculate Hphiz fields**********************c

C ****The Right/Left PML

      do 290 k=1,pmldepth
        sigma_z=sigma_max*((k+0.0)/pmldepth)**2.0
        sigma_zs=sigma_z*(mu/eps)
        c1=exp(-sigma_zs*dt/mu)
        c2=eta*(c1-1.0)/(sigma_zs*dz)

        do 300 i=1,pmldepth+maxr
```

```
        if (k.eq.1) THEN
          if (i.gt.maxr) THEN
            hphizr(k,i)=c1*hphizr(k,i)-c2*(erzt(maxz,i-maxr)+
     1          erphit(maxz,i-maxr)-erzr(k,i)-erphir(k,i))
          ELSE
            hphizr(k,i)=c1*hphizr(k,i)-c2*(er(maxz,i)
     1          -erzr(k,i)-erphir(k,i))
          END IF
        ELSE
          hphizr(k,i)=c1*hphizr(k,i)-c2*(erzr(k-1,i)+erphir(k-1,i)
     1        -erzr(k,i)-erphir(k,i))
        END IF

        hphizl(k,i)=c1*hphizl(k,i)-c2*(erzl(k+1,i)+erphil(k+1,i)
     1      -erzl(k,i)-erphil(k,i))


        if (((case_id.eq.1).or.(case_id.eq.5))
     1      .and.(i.le.left_y)) then
c        if ((case_id.eq.1).or.(case_id.eq.5)) then
          hphizlx(k,i)=c1*hphizlx(k,i)-c2*(erzlx(k+1,i)
     1        +erphilx(k+1,i)-erzlx(k,i)-erphilx(k,i))
        end if


 300    continue
 290  continue

C ****The Up/Down PML

      c3=eta*dt/(mu*dz)

      do 310 k=1,maxz
        do 320 i=1,pmldepth
          if (k.eq.1) THEN
            hphizt(k,i)=hphizt(k,i)+c3*(erphil(1,i+maxr)+
     1          erzl(1,i+maxr)-erphit(k,i)-erzt(k,i))
          ELSE
            hphizt(k,i)=hphizt(k,i)+c3*(erphit(k-1,i)+erzt(k-1,i)-
     1          erphit(k,i)-erzt(k,i))
          END IF
 320    continue
 310  continue

C **************Calculate Hphir fields**********************c

C ****Bottom/Top (sigma_r) Regions PML

      do 330 i=1,pmldepth
        sigma_r=sigma_max*((i+0.0+0.5)/pmldepth)**2.0
        sigma_rs=sigma_r*(mu/eps)
        c1=exp(-sigma_rs*dt/mu)
        c2=eta*(c1-1.0)/(sigma_rs*dz)

C *******Center Top Region
        do 340 k=1,maxz
          hphirt(k,i)=c1*hphirt(k,i)-c2*(ezrt(k,i+1)+ezphit(k,i+1)-
     1        ezrt(k,i)-ezphit(k,i))
 340    continue

c ****right/left corners
```

```
          do 350 k=1,pmldepth
             hphirr(k,i+maxr)=c1*hphirr(k,i+maxr)-c2*(ezrr(k,i+1+maxr)+
     1           ezphir(k,i+1+maxr)-ezrr(k,i+maxr)-ezphir(k,i+maxr))
             hphirl(k,i+maxr)=c1*hphirl(k,i+maxr)-c2*(ezrl(k,i+1+maxr)+
     1           ezphil(k,i+1+maxr)-ezrl(k,i+maxr)-ezphil(k,i+maxr))
 350      continue
 330    continue

C ****Right/Left Center Regions (no sigmas!!)

          c4=eta*dt/(mu*dz)
          do 360 k=1,pmldepth
            do 370 i=1,maxr
               hphirr(k,i)=hphirr(k,i)+c4*(ezrr(k,i+1)+ezphir(k,i+1)
     1              -ezrr(k,i)-ezphir(k,i))

               hphirl(k,i)=hphirl(k,i)+c4*(ezrl(k,i+1)+ezphil(k,i+1)
     1              -ezrl(k,i)-ezphil(k,i))

               if (((case_id.eq.1).or.(case_id.eq.5))
     1              .and.(i.le.left_y)) then
c               if ((case_id.eq.1).or.(case_id.eq.5)) then
                 hphirlx(k,i)=hphirlx(k,i)+c4*(ezrlx(k,i+1)
     1                +ezphilx(k,i+1)-ezrlx(k,i)-ezphilx(k,i))
               end if

 370        continue
 360      continue

C **************Calculate Hzr fields**********************c

        do 380 i=1,pmldepth
          sigma_r=sigma_max*((i+0.0+0.5)/pmldepth)**2.0
          sigma_rs=sigma_r*(mu/eps)
          c1=exp(-sigma_rs*dt/mu)
          c2=eta*(c1-1.0)/(sigma_rs*dz)
          c3=c2/(i+maxr+0.5-1.0)

C *******Middle Top/Bottom Region
          do 390 k=1,maxz
             hzrt(k,i)=c1*hzrt(k,i)+c3*((i+maxr+0.0)*(ephizt(k,i+1)+
     1           ephirt(k,i+1))-(i+maxr-1.0)*(ephizt(k,i)+ephirt(k,i)))
 390      continue

c *******right/left corners
          do 400 k=1,pmldepth
             hzrr(k,i+maxr)=c1*hzrr(k,i+maxr)+c3*((i+maxr+0.0)*
     1           (ephizr(k,i+maxr+1)+ephirr(k,i+maxr+1))-(i+maxr-1.0)*
     1           (ephizr(k,i+maxr)+ephirr(k,i+maxr)))

             hzrl(k,i+maxr)=c1*hzrl(k,i+maxr)+c3*((i+maxr+0.0)*
     1           (ephizl(k,i+maxr+1)+ephirl(k,i+maxr+1))-(i+maxr-1.0)*
     2           (ephizl(k,i+maxr)+ephirl(k,i+maxr)))

 400      continue
 380    continue

C ****Right/Left Center Regions (no sigmas!!)

        do 410 i=1,maxr
          c5=eta*(i+0.0-1.0)*dt/(mu*(i+0.5-1.0)*dz)
```

```
         c6=eta*(i+1.0-1.0)*dt/(mu*(i+0.5-1.0)*dz)
         do 420 k=1,pmldepth
           hzrl(k,i)=hzrl(k,i)+c5*(ephizl(k,i)+ephirl(k,i))-c6*
     1          (ephizl(k,i+1)+ephirl(k,i+1))
           hzrr(k,i)=hzrr(k,i)+c5*(ephizr(k,i)+ephirr(k,i))-c6*
     1          (ephizr(k,i+1)+ephirr(k,i+1))

           if (((case_id.eq.1).or.(case_id.eq.5))
     1          .and.(i.le.left_y)) then
c           if ((case_id.eq.1).or.(case_id.eq.5)) then
             hzrlx(k,i)=hzrlx(k,i)+c5*(ephizlx(k,i)+ephirlx(k,i))-c6*
     1            (ephizlx(k,i+1)+ephirlx(k,i+1))
           end if


 420     continue
 410   continue

C **************Calculate Hzphi fields**********************c

C ****Top/Bottom PML Regions

       do 430 i=1,pmldepth
         c1=m*dt/(mu*dz)
         c2=eta*c1/(i+maxr+0.5-1.0)

         do 440 k=1,maxz
           hzphit(k,i)=hzphit(k,i)+c2*(erphit(k,i)+erzt(k,i))
 440     continue
 430   continue

C ****Right/Left PML Regions

       do 450 i=1,pmldepth+maxr
         c1=eta*m*dt/(mu*dz*(i+0.5-1.0))
         do 460 k=1,pmldepth
           hzphir(k,i)=hzphir(k,i)+c1*(erphir(k,i)+erzr(k,i))
           hzphil(k,i)=hzphil(k,i)+c1*(erphil(k,i)+erzl(k,i))
           if (((case_id.eq.1).or.(case_id.eq.5))
     1          .and.(i.le.left_y)) then
c           if ((case_id.eq.1).or.(case_id.eq.5)) then
             hzphilx(k,i)=hzphilx(k,i)+c1*
     1            (erphilx(k,i)+erzlx(k,i))
           end if

 460     continue
 450   continue

c     if ((case_id.eq.1).or.(case_id.eq.5)) then
c       do 470  i=1,maxr
c         do 480 k=1,pmldepth
c           if (i.eq.mheight) then
c               write(41,*) erzl(k,i) + erphil(k,i)
c               write(41,*) ephizl(k,i) + ephirl(k,i)
c               write(41,*) ezrl(k,i) + ezphil(k,i)
c               write(41,*) hrzl(k,i) + hrphil(k,i)
c               write(41,*) hphizl(k,i) + hphirl(k,i)
c               write(41,*) hzrl(k,i) + hzphil(k,i)
c               write(41,*) erzl(k,i-1) + erphil(k,i-1)
c               write(41,*) hphizl(k,i-1) + hphirl(k,i-1)
c           end if
```

```
c 480       continue
c 470     continue
c       end if

      return
      end
```

A.5 Gaussian Quadrature for Incident Wave

This portion of the program uses the Gaussian quadrature to calculate an integral. From this calculation, the program obtains the coefficients for Fourier series to form the incident plane wave.

```
c******************************************************************c
c Calculates an numerical integral using Gaussian Quadrature      c
c in order to determines the coef of the Fourier series for       c
c the incident plane wave.                                        c
c Intno: -4-Ermu;  2-Ephimu; -6-Ezmu; 4-Ermv; -2-Ephimv; 6-Ezmv c
c         7-Hrmu; -11-Hphimu; 9-Hzmu; -7-Hrmv; 11-Hphimv; -9-Hzmv c
c******************************************************************c

      real*8 function Gquad(a,b,IntNo,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 a,b,t,r,zg,theta,Intgrl,value,y,z,weight
      real*8 Ermu,Ephimu,Ezmu,Ermv,Ephimv,Ezmv,cossq,z20
      real*8 Hrmu,Hphimu,Hzmu,Hrmv,Hphimv,Hzmv,sinsq
      real*8 weight20, dx, e1, e2, h, mid, steps
      integer j,IntNo,m,AIN,i

      dimension z(10), weight(10), z20(20), weight20(20)
      DATA (z(j), j=1,10)/-.9739065285,-.8650633667,-.6794095683,
     1     -.4333953941, -.1488743390,.1488743390,.4333953941,
     2     .6794095683, .8650633667,.9739065285/

      DATA (weight(j), j=1,10)
     1     /.0666713443,.1494513492,.2190863625,.2692667193,
     2      .2955242247,.295524247,.2692667193,.2190863625,
     3      .1494513492,.0666713443/

      DATA (z20(j), j=1,20)
     1     /-0.99312859919241, -0.96397192726078,
     2      -0.91223442826796, -0.83911697181213,
     3      -0.74633190646476, -0.63605368072468,
     4      -0.51086700195146, -0.37370608871528,
     5      -0.22778585114165, -0.07652652113350,
     6       0.07652652113350,  0.22778585114165,
     7       0.37370608871528,  0.51086700195146,
     8       0.63605368072468,  0.74633190646476,
```

```
     9      0.83911697181213,   0.91223442826796,
     1      0.96397192726078,   0.99312859919241/


      DATA (weight20(j), j=1,20)
     1    /0.01761400713536,
     1    0.04060142981029, 0.06267204829089,
     2    0.08327674159386, 0.10193011980641,
     3    0.11819453199405, 0.13168863844930,
     4    0.14209610916487, 0.14917298630417,
     5    0.15275338717117, 0.15275338723120,
     6    0.14917298659407, 0.14209610937519,
     7    0.13168863843930, 0.11819453196154,
     8    0.10193011980823, 0.08327674160932,
     9    0.06267204829828, 0.04060142982019,
     1    0.01761400714091/


cBZ offsets
      if (case_id.eq.1) then
        zg = zg + zoffset*dz
      else if (case_id.eq.5) then
        zg = zg + 0*dz
      end if

C*****Expressions to account for "real*8" distance from orgin
C*****of field values.  It calculates field distances for 1/2 lattice
C*****points, and since "grid" i=1,maxr <=> "real" i=0,(maxr-1)*dr

      r=r-dz

      AIN = abs(IntNo)
      if (AIN.eq.4.OR.AIN.eq.9.OR.AIN.eq.11)
     1    r=r+dz/(2.0)
      if (AIN.eq.4.OR.AIN.eq.9.OR.AIN.eq.2)
     1    zg=zg+dz/(2.0)
      if (AIN.eq.7.OR.AIN.eq.9.OR.AIN.eq.11)
     1    t = t-dt
c    1    t = t
      if (AIN.eq.4.OR.AIN.eq.6.OR.AIN.eq.2)
     1    t = t-dt/2.0
c    1    t = t+dt/2.0


C*****************************************************************
C* Integration by 20-point Gauss-Legendre quadrature.  A and B  *
C* are the limits of integration, and FUNC is the user-supplied *
C* function to be integrated.  The result is returned in INTGRL *
c* Incident waves of mode m are divided in m+1 regions which    *
c* are each computed by 20-point gquad.                         *
C*****************************************************************

      INTGRL = 0.0
      dx = (b-a)/(m+1)
      do 5 steps = 1,(m+1)
        e1 = a + (steps-1)*dx
        e2 = a + steps*dx
        h = (e2-e1)/2
        mid = (e1+e2)/2

        do 10 I = 1,20
```

```
      Y = z20(I)*h + mid
      if (IntNo.eq.2) value = Ephimu(Y,m,t,r,zg,theta)
      if (IntNo.eq.4) value = Ermv(Y,m,t,r,zg,theta)
      if (IntNo.eq.6) value = Ezmv(Y,m,t,r,zg,theta)
      if (IntNo.eq.7) value = Hrmu(Y,m,t,r,zg,theta)
      if (IntNo.eq.9) value = Hzmu(Y,m,t,r,zg,theta)
      if (IntNo.eq.11) value = Hphimv(Y,m,t,r,zg,theta)

      if (IntNo.eq.-2) value = Ephimv(Y,m,t,r,zg,theta)
      if (IntNo.eq.-4) value = Ermu(Y,m,t,r,zg,theta)
      if (IntNo.eq.-6) value = Ezmu(Y,m,t,r,zg,theta)
      if (IntNo.eq.-7) value = Hrmv(Y,m,t,r,zg,theta)
      if (IntNo.eq.-9) value = Hzmv(Y,m,t,r,zg,theta)
      if (IntNo.eq.-11) value = Hphimu(Y,m,t,r,zg,theta)

      if (IntNo.eq.45) value = cossq(Y)
      if (IntNo.eq.46) value = sinsq(Y)

      INTGRL = INTGRL + h*weight20(I)*value
10    continue
5   continue

    if (m.eq.0) THEN
      gquad = INTGRL*5.0/2.0
    ELSE
      gquad = INTGRL*5.0
    END IF

c     if (abs(gquad).gt.1e-6) print *,gquad,IntNo,m,t,r,zg,theta

c     if (abs(gquad).gt.1) then
c       print *, IntNo, t, r, zg, sdev, theta, m
c     end if


    RETURN
    END

c***************************************************************c
c All the incident wave functions to be integrated by Gaussian
c Quadrature.
c***************************************************************c

    real*8 function cossq(phi)

    implicit none
    include 'common.f'

    real*8 phi

    cossq = cos(6*phi)*cos(phi)*exp(-(3+cos(phi))**2.0)*100

    return
    end

C*********************************************************
    real*8 function sinsq(phi)

    implicit none
    include 'common.f'
```

```
      real*8 phi

      sinsq = (1/pi)*(sin(phi))**2.0

      return
      end
C*************************************************************

      real*8 function Ermu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ermu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(Ehg*cos(phi)*cos
     1     (theta)+Evg*sin(phi))*exp(-(((t-gd)+((zg*cos(theta)
     2     +r*sin(theta)*cos(phi))/c))**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c*************************************************************c

      real*8 function Ermv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ermv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(Ehg*cos(phi)*cos(theta)
     1     +Evg*sin(phi))*exp(-(((t-gd)+((zg*cos(theta)+r*sin(theta)
     2     *cos(phi))/c))**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c*************************************************************c

      real*8 function Ephimu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ephimu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(-Ehg*sin(phi)*cos(theta)
     1     +Evg*cos(phi))*exp(-(((t-gd)+((zg*cos(theta)+r*sin(theta)
     2     *cos(phi))/c))**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
```

```
      end

c**************************************************************c

      real*8 function Ephimv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ephimv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(-Ehg*sin(phi)*cos(theta)
     1     +Evg*cos(phi))*exp(-(((t-gd)+((zg*cos(theta)+r*sin(theta)
     2     *cos(phi))/c))**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c**************************************************************c

      real*8 function Ezmu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ezmu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(-Ehg*sin(theta))*
     1     exp((-((t-gd)+(zg*cos(theta)+r*sin(theta)*cos
     2     (phi))/c)**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c**************************************************************c

      real*8 function Ezmv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Ezmv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(-Ehg*sin(theta))*
     1     exp((-((t-gd)+(zg*cos(theta)+r*sin(theta)*cos(phi))
     2     /c)**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c**************************************************************c
```

```
      real*8 function Hrmu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Hrmu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(Evg*cos(theta)*
     1     cos(phi)-Ehg*sin(phi))*exp((-((t-gd)+(zg*cos(theta)+r*
     2     sin(theta)*cos(phi))/c)**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c*****************************************************************c

      real*8 function Hrmv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Hrmv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(Evg*cos(theta)*
     1     cos(phi)-Ehg*sin(phi))*exp((-((t-gd)+(zg*cos(theta)+r*
     2     sin(theta)*cos(phi))/c)**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c*****************************************************************c

      real*8 function Hphimu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Hphimu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(-Evg*cos(theta)*
     1     sin(phi)-Ehg*cos(phi))*exp((-((t-gd)+(zg*cos(theta)+r*
     2     sin(theta)*cos(phi))/c)**2)/(sdev**2))*
     3     ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4     cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c*****************************************************************c

      real*8 function Hphimv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'
```

```
      real*8 phi,t,r,zg,theta
      integer m

      Hphimv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(-Evg*cos(theta)*
     1    sin(phi)-Ehg*cos(phi))*exp((-((t-gd)+(zg*cos(theta)+r*
     2    sin(theta)*cos(phi))/c)**2)/(sdev**2))*
     3    ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4    cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c**************************************************************c

      real*8 function Hzmu(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Hzmu=(1/(pi*sqrt(2*pi)))*cos(m*phi)*(-Evg*sin(theta))*
     1    exp((-((t-gd)+(zg*cos(theta)+r*sin(theta)*cos(phi))
     2    /c)**2)/(sdev**2))*
     3    ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4    cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end

c**************************************************************c

      real*8 function Hzmv(phi,m,t,r,zg,theta)

      implicit none
      include 'common.f'

      real*8 phi,t,r,zg,theta
      integer m

      Hzmv=(1/(pi*sqrt(2*pi)))*sin(m*phi)*(-Evg*sin(theta))*
     1    exp((-((t-gd)+(zg*cos(theta)+r*sin(theta)*cos(phi))
     2    /c)**2)/(sdev**2))*
     3    ((sin(2*pi*modfreq*((t-gd)+((zg*cos(theta)+r*sin(theta)*
     4    cos(phi))/c))))*modulate+abs(modulate-1))

      return
      end
```

A.6 Memory Allocation

The memory allocation along with global variables and constants are specified in this portion of the program.

```
c*************************************************************c
c This is the common file for the BOR program. It contains    c
c all the global variables and constants used in the program  c
c*************************************************************c

      integer mz, mr, maxpt, MAX_FREQS, mm, mxdp, nm, MAXCP,
     1    MAX_STAIR_NODES, MAX_Z_CELLS, MAX_R_CELLS, MAX_RCS_NODES,
     2    MAX_NODES

C***** ADJUSTABLE PARAMETERS TO ALLOCATE MEMORY NEEDED

      parameter(mz = 1050)
      parameter(MAX_Z_CELLS = mz)
      parameter(mr = 426)
      parameter(MAX_R_CELLS = mr)
      parameter(maxpt = 1800)
      parameter(nm = 0)
      parameter(mm = 30)
      parameter(mxdp = 2000)
      parameter(MAX_FREQS = 140)
      parameter(MAXCP = 4*maxpt)
      parameter(MAX_STAIR_NODES=2021)
      parameter(MAX_RCS_NODES=mxdp)
      parameter(MAX_NODES=maxpt)

C*****  DO NOT CHANGE BELOW ************************************

      real*8 sigma_max,dz,freq,len,tole, dt, sdev
      real*8 Ehg,Evg,gd,modfreq,maxf_v,inc_ang,obj_height
      real*8 low_freq,high_freq,dfreq,sim_duration

      real*8 eta, mu, eps, c, pi

      logical enough_memory

cBZ 08/01/02 added below********************************
      integer menu_choice
      integer case_id
      integer features
      integer mode_no
      integer end_playback
      integer flag, quit_flag, before3
      integer start_time, end_time, start_mem_rec
      real*8  er_max, er_mem
      real*8  max_height, max_length, mxr
      integer z_offset, absolute_start, absolute_end
      integer rcsz_start, rcsz_end
      integer rcsz, rcsr
      integer x_start_tot, x_end_tot
      integer upper_edgetot,upper_edgescat, upper_edgehuy
      integer lower_edgetot, lower_edgescat
      integer lower_edgeleft, lower_edgeright
      integer x_opening, y_opening
      integer right_x, right_y
      integer left_x, left_y, pookie
      integer high_y, high_x, chuck, zoffset, maxztrue

cBZ 10/11/02 cells to store field data for RCS calculation
c$$$      real*8 er_top(1:mz), ez_top(1:mz)
c$$$      real*8 ephi_top(1:mz), hr_top(1:mz)
c$$$      real*8 hz_top(1:mz), hphi_top(1:mz)
```

```
c$$$   real*8 er_topx(1:mz), hphi_topx(1:mz)
c$$$
c$$$   real*8 er_left(1:mr), ez_left(1:mr)
c$$$   real*8 ephi_left(1:mr), hr_left(1:mr)
c$$$   real*8 hz_left(1:mr), hphi_left(1:mr)
c$$$   real*8 ephi_leftx(1:mr), hz_leftx(1:mr)
c$$$
c$$$   real*8 er_right(1:mr), ez_right(1:mr)
c$$$   real*8 ephi_right(1:mr), hr_right(1:mr)
c$$$   real*8 hz_right(1:mr), hphi_right(1:mr)
c$$$   real*8 ephi_rightx(1:mr), hz_rightx(1:mr)

cBZ*****************************************************

      integer N, time, pmldepth, NP, maxz, maxr,modes,ps
      integer movie_num,movie_type,nframe,gquad_count,mheight
      integer modulate,rcsz1,rcsz2,minf,maxf,stepf
      integer eqset_start, eqset_end, mode_start, mode_end

c*******# cells in total field region
      integer xtot_sp, ytot_sp

c*******# cells in scattered field region
      integer xscat_sp, yscat_sp

c*******# cells between total fields and Huygens' surface
      integer xhuy_sp, yhuy_sp

c*******# cells from object to PML region
      integer xall_sp, yall_sp, xscatplay_sp, xextend_sp

c*******xall_sp = xtot_sp+xscat_sp
c*******yall_sp = ytot_sp+yscat_sp

      character base*80
      character*72 fnamein, dnamefdata, mhname, mfname,
     1   dbase

      parameter(c=2.99792458d8, mu=1.25663706144D-6)
      parameter(pi=3.1415926535d0, eps=8.8541874D-12,eta=376.73031d0)
      parameter(pmldepth=15)

      parameter(tole=1d-12)

C*****  Geometry readin routine parameters and variables.

C*****  RB,ZB: translated points; RBa, ZBa: original data points
      real*8 RBa(1:maxpt), ZBa(maxpt)
      real*8 RBt(1:maxpt), ZBt(maxpt)
      real*8 RB(maxpt), ZB(maxpt)

C*****  Parameters giving starting position of the target.
      integer start_z,end_z,end_r
      parameter(start_z = 40, end_z=mz-40,end_r=mr-40)

      integer accessk, accessi, accesst
      parameter(accessk=1,accessi=2,accesst=3)

      integer actype, acl1, acl2, acA, acNPi, ack, aci
      integer acz, acr, YES, NO, YES_RIGHT, YES_LEFT
      parameter(actype=1,acl1=2,acl2=3,acA=4,acNPi=5)
```

```
      parameter(ack=6, aci=7, acz=8, acr=9)
      parameter(YES=1, NO=0, YES_RIGHT=2, YES_LEFT=3)

      integer conform_grid1(1:mz,1:mr)
      real*8 conform_list(1:9,MAXCP)
      integer borrow_list(1:4,MAXCP), listcount
      integer ezt, ezb, erl, err
      parameter(ezt=1,ezb=3,erl=4, err=2)

      integer parallel, perp
      parameter(parallel=2,perp=1)

C*****  Variables for conformal Hz field.
C*****  using accessk, accessi, accesst

      integer conform_hz(1:3,MAXCP), EQZERO_HZ, STRETCH_HZ, SC_HZ,
     1   hzcount, conform_hz1(1:mz,1:mr)

      real*8 conform_hz_length(MAXCP)
      parameter(EQZERO_HZ=1, STRETCH_HZ=2, SC_HZ=3)

C*****  Variables and parameters for conformal Hr field.

      integer conform_hr(1:3,MAXCP), EQZERO_HR, SRIGHT_HR, SLEFT_HR,
     1   hrcount, conform_hr1(1:mz,1:mr), SLEFT_HR_DC, SRIGHT_HR_DC,
     2   SRIGHT_HR_IC, SLEFT_HR_IC

      real*8 conform_hr_length(MAXCP)
      parameter(EQZERO_HR=1, SRIGHT_HR=2, SLEFT_HR=3, SRIGHT_HR_DC=4,
     1   SLEFT_HR_DC=5, SRIGHT_HR_IC=6, SLEFT_HR_IC=7)

      integer erf,ezf,ephif,hrf,hzf,hphif,hzfo,hrfo,ezsc,ersc
      parameter(erf=1,ezf=2,ephif=3,hrf=4,hzf=5,hphif=6,hzfo=7)
      parameter(hrfo=8, ezsc=9,ersc=10)

      integer ephi_conform1(1:mz,1:mr), ephicount, conform_ephi(1:3,
     1   MAXCP)

      integer ez_conform1(1:mz,1:mr), er_conform1(1:mz,1:mr)

      integer staircase(6:7,1:MAXCP),staircount
      integer total_nodes, stair_node_count,
     1   stair_zero(1:MAX_STAIR_NODES,1:3)

      integer movie_step
      logical store_movie, use_conformal, use_stair2

      integer errorcount, errors(10),
     1   NODE_ERROR, MAX_Z_ERROR, MAX_R_ERROR, MAX_STAIR_ERROR,
     2   MAX_RCS_ERROR

      parameter(NODE_ERROR=1, MAX_Z_ERROR=2, MAX_R_ERROR=3,
     1   MAX_STAIR_ERROR=4, MAX_RCS_ERROR=5)

C ******************************************************************

C*****  Cells in the free space region

      real*8 er(1:mz,1:mr), ez(1:mz,1:mr)
      real*8 ephi(1:mz,1:mr), hr(1:mz,1:mr)
      real*8 hz(1:mz,1:mr), hphi(1:mz,1:mr)
```

```
C ******Note: the array scattot indicates whether the cell is in a
C ******      scattering field points dictated by picture. see chart in
C ******      README file. Tot Fields: 2-9, 14; Scat Fields: 1,11,12,15

      integer scattot(1:mz, 1:mr)

C ******Cells in left Region of PML (includes top-bottom left corners)

      real*8   erzl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephizl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   ezrl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hrzl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphizl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hzrl(1:pmldepth+1,0:pmldepth+mr+1)

      real*8 erphil(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephirl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ezphil(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hrphil(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphirl(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hzphil(1:pmldepth+1,0:pmldepth+mr+1)

C ******Case 1 Cells in left Region of PML (includes top-bottom left corners)
c to do outer problem simulation -> need interior PML

      real*8   erzlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephizlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   ezrlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hrzlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphizlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hzrlx(1:pmldepth+1,0:pmldepth+mr+1)

      real*8 erphilx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephirlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ezphilx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hrphilx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphirlx(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hzphilx(1:pmldepth+1,0:pmldepth+mr+1)



C ******Cells in the right Region of PML (incl. top-bot right corners)

      real*8   erzr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephizr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   ezrr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hrzr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphizr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8   hzrr(1:pmldepth+1,0:pmldepth+mr+1)

      real*8 erphir(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ephirr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 ezphir(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hrphir(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hphirr(1:pmldepth+1,0:pmldepth+mr+1)
      real*8 hzphir(1:pmldepth+1,0:pmldepth+mr+1)

C ******Cells in the top Region of PML (no corners)

      real*8   erzt(1:mz,1:pmldepth+1)
```

```
      real*8 ephizt(1:mz,1:pmldepth+1)
      real*8   ezrt(1:mz,1:pmldepth+1)
      real*8   hrzt(1:mz,1:pmldepth+1)
      real*8 hphizt(1:mz,1:pmldepth+1)
      real*8   hzrt(1:mz,1:pmldepth+1)

      real*8 erphit(1:mz,1:pmldepth+1)
      real*8 ephirt(1:mz,1:pmldepth+1)
      real*8 ezphit(1:mz,1:pmldepth+1)
      real*8 hrphit(1:mz,1:pmldepth+1)
      real*8 hphirt(1:mz,1:pmldepth+1)
      real*8 hzphit(1:mz,1:pmldepth+1)

C ******Frequency components
C ****** mxf = maximum number of frequencies to store.
C ******  mm = maximum number of modes to store.
C ******mxdp = maximum number of points to calculate far-field with.

      real*8 low_phi, high_phi, dphi, low_theta, high_theta, dtheta

      integer num_freqs
      complex*16 feru(nm:mm,1:mxdp,1:MAX_FREQS),
     1 ferv(nm:mm,1:mxdp,1:MAX_FREQS),
     2 fephiu(nm:mm,1:mxdp,1:MAX_FREQS),
     3 fephiv(nm:mm,1:mxdp,1:MAX_FREQS),
     4 fezu(nm:mm,1:mxdp,1:MAX_FREQS),
     5 fezv(nm:mm,1:mxdp,1:MAX_FREQS),
     6 fhru(nm:mm,1:mxdp,1:MAX_FREQS),
     7 fhrv(nm:mm,1:mxdp,1:MAX_FREQS),
     8 fhphiu(nm:mm,1:mxdp,1:MAX_FREQS),
     9 fhphiv(nm:mm,1:mxdp,1:MAX_FREQS),
     1 fhzu(nm:mm,1:mxdp,1:MAX_FREQS),
     2 fhzv(nm:mm,1:mxdp,1:MAX_FREQS)
      real*8 freqlist(1:MAX_FREQS,1:2)
c****** gives the starting index in freqlist of extra freqs for
c****** use in approximating the monostatic RCS
      integer mono_freq_ind(1:MAX_FREQS), mono_nang
      logical calc_bist

C ******Common Block

      common/A/ sigma_max,dz,freq,len, dt, sdev,
     1 Ehg,Evg,gd,modfreq,maxf_v,inc_ang,obj_height,
     2 low_freq,high_freq,dfreq,sim_duration

      common/B/ menu_choice, case_id, features, mode_no, end_playback,
     1 flag, quit_flag, before3, start_time, start_mem_rec,
     2 end_time, er_max, er_mem, max_height, max_length, mxr,
     3 z_offset,absolute_start,absolute_end,rcsz_start, rcsz_end,
     4 rcsz, rcsr, x_start_tot, x_end_tot,
     5 upper_edgetot,upper_edgescat, upper_edgehuy,
     5 lower_edgetot,lower_edgescat,
     5 lower_edgeleft, lower_edgeright,
     6 x_opening, y_opening,
     7 right_x, right_y, left_x, left_y, pookie,
     8 high_y, high_x, chuck, zoffset, maxztrue

c$$$  common/C/ er_top, ez_top, ephi_top,
c$$$ 1 hr_top, hz_top, hphi_top,
c$$$ 2 er_topx, hphi_topx
c$$$
```

```
c$$$  common/D/ er_left, ez_left, ephi_left,
c$$$  1   hr_left, hz_left, hphi_left,
c$$$  2   ephi_leftx, hz_leftx
c$$$
c$$$  common/E/ er_right, ez_right, ephi_right,
c$$$  1   hr_right, hz_right, hphi_right,
c$$$  2   ephi_rightx, hz_rightx


      common/CA/ N, time, NP, maxz, maxr,modes,ps
      common/CB/ movie_num,movie_type,nframe,gquad_count,mheight
      common/CC/ modulate,rcsz1,rcsz2,minf,maxf,stepf
      common/CD/ eqset_start, eqset_end, mode_start, mode_end
      common/CE/ xtot_sp, ytot_sp, xscat_sp, yscat_sp,
      1   xhuy_sp, yhuy_sp, xall_sp, yall_sp, xscatplay_sp, xextend_sp

      common/CD/ RBa, ZBa, RBt, ZBt, RB, ZB


      common/DA/ conform_grid1
      common/DB/ conform_list
      common/DC/ borrow_list
      common/DD/ listcount, hzcount, hrcount, ephicount
      common/DE/ conform_hz, conform_hr, conform_ephi
      common/DF/ conform_hz1, conform_hr1, ephi_conform1,
      1   ez_conform1, er_conform1
      common/DG/ conform_hz_length, conform_hr_length

      common/EA/ staircase
      common/EB/ stair_zero
      common/EC/ staircount, total_nodes, stair_node_count

      common/FA/ errors, movie_step, errorcount
      common/FB/ enough_memory, store_movie, use_conformal,
      1   use_stair2
      common/FC/ base
      common/FD/ fnamein, dnamefdata, mhname, mfname, dbase


      common/GA/ er, ez, ephi, hr, hz, hphi
      common/GB/ scattot

      common/HA/ erzl, ephizl, ezrl, hrzl, hphizl, hzrl
      common/HB/ erphil, ephirl, ezphil, hrphil, hphirl, hzphil,
      1   erzlx, ephizlx, ezrlx, hrzlx, hphizlx, hzrlx,
      2   erphilx, ephirlx, ezphilx, hrphilx, hphirlx, hzphilx

      common/HC/ erzr, ephizr, ezrr, hrzr, hphizr, hzrr
      common/HD/ erphir, ephirr, ezphir, hrphir, hphirr, hzphir
      common/HE/ erzt, ephizt, ezrt, hrzt, hphizt, hzrt
      common/HF/ erphit, ephirt, ezphit, hrphit, hphirt, hzphit

      common/IA/ low_phi, high_phi, dphi, low_theta,
      1   high_theta, dtheta

      common/JA/ num_freqs, mono_nang, calc_bist
      common/JB/ feru, ferv, fephiu, fephiv, fezu, fezv, fhru,
      1   fhrv, fhphiu, fhphiv, fhzu, fhzv
      common/JC/ freqlist
      common/JD/ mono_freq_ind
```

Bibliography

[1] A. Altintas, P.H. Pathak, and M.C. Liang. A selective model scheme for the analysis of EM coupling into or radiate from large open-ended waveguide cavities. *IEEE Transactions on Antennas and Propagation*, 36(1):84–96, January 2002.

[2] Veeraraghava Anantha and Allen Taflove. Efficient modeling of infinite scatterers using a generalized total-field/scattered-field FDTD boundary partially embedded within PML. *IEEE Transactions on Antennas and Propagation*, 50(10):1337–1349, October 2002.

[3] Morgens G. Andreasen. Scattering from bodies of revolution. *IEEE Transactions on Antennas and Propagation*, 13:303–310, March 1965.

[4] Pierre Baldensperger, Jian Liu, and Jian-Ming Jin. A hybrid SBR/FE-BI technique for computing the RCS of electrically large objects with deep cavities. In *2001 IEEE International Symposium*, volume 4, pages 726–729. Antennas and Propagation Society, July 2001.

[5] Andre Barka, Paul Soudais, and Dominique Volpert. Scattering from 3-D cavities with a plug and play numerical scheme combining IE, PDE, and modal techniques. *IEEE Transactions on Antennas and Propagation*, 48(5):704–712, May 2000.

[6] J.P. Berenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114:185–200, 1994.

[7] Charles L. Britt. Solution of electromagnetic scattering problems using time domain techniques. *IEEE Transactions on Antennas and Propagation*, 37(9):1181–1191, September 1989.

[8] R. Burkholder and P.H. Pathak. Analysis of EM penetration into and scattering by electrically large open waveguidge cavities using gaussian beam shooting. *Proceeding of the IEEE*, 79(10):1401–1411, October 1991.

[9] M. Celuch-Marcysiak and W. Gwarek. On the nature of solutions produced by finite difference schemes in time domain. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 12:23–40, 1999.

[10] Yinchao Chen, Raj Mittra, and Paul Harms. Finite-difference time-domain algorithm for solving Maxwell's equations in rotationally symmetric geometries. *IEEE Trans. Microwave Theory Tech.*, 44(6):832–839, June 1996.

[11] W.C. Chew, J.M. Jin, and E. Michielssen. Complex coordinate stretching as a generalize absorbing boundary condition. a perfectly matched layer for the absorption of electromagnetic waves. *Microwave and Optical Technology Letters*, 15(6):363–369, August 1997.

[12] W.C. Chew and William H. Weedon. A 3-D perfectly matched medium from modified Maxwell's equation in stretched coordinates. *Microwave and Optical Technology Letters*, 7(13):599–604, September 1994.

[13] J. W. Crispin Jr. and A.L. Maffet. Radar cross section estimation for simple shapes. *Proceeding of the IEEE*, 53(8):833–848, August 1965.

[14] J. W. Crispin Jr. and K. M. Siegel. *Methods of Radar Cross-Section Analysis.* Academic Press Inc, 1968.

[15] Bjorn Engquist and Andrew Majda. Absorbing boundary conditions for the numerical simulation of waves. *Mathematics of Computation*, 31(139):629–651, July 1977.

[16] Guo-Xin Fan and Qing Huo Liu. A PML-FDTD algorithm for simulating plasma covered cavity-backed slot antennas. *Microwave and Optical Technology Letters*, 19(4):258–262, November 1998.

[17] Roger F. Harrington. Matrix methods for field problems. *Proceedings of the IEEE*, 55(2):139–149, February 1967.

[18] J. M. Jin. Electromagnetic scattering from large, deep, and arbitarily-shaped cavities. *Electromagnetics*, 18(1):3–34, 1998.

[19] J. Michael Johnson and Yahya Rahmat-Samii. A multiple-region finite-difference-time-domain method. *Microwave and Optical Technology Letters*, 14(2):101–105, February 1997.

[20] Thomas G. Jugens and Allen Taflove. Three-dimensional contour FDTD modeling of scattering from single and multiple bodies. *IEEE Transactions on Antennas and Propagation*, 41(12):1703–1708, December 1993.

[21] Thomas G. Jugens, Allen Taflove, Korada Umashankar, and Thomas G. Moore. Finite-difference time-domain modeling of curved surfaces. *IEEE Transactions on Antennas and Propagation*, 40(4):357–366, April 1992.

[22] Thomas Jurgens. A broadband absorbing boundary condition for the FDTD modeling of circular waveguides. In *IEEE MTT-S International Microwave Symposium Digest*, volume 1, pages 35–37, New York, NY, USA, 1995.

[23] R. Kastner and R. Mittra. A spectral-iteration technique for analyzing scattering from arbitrary bodies. *IEEE Transactions on Antennas and Propagation*, 31(5), May 2002.

[24] Eugene F. Knott, John. F. Shaeffer, and Michael T. Tuley. *Radar Cross-Section: its Prediction, Measurement and Reduction.* Artech House, 1985.

[25] J. A. Kong. *Electromagnetic Wave Theory.* John Wiley and Sons, New York, 1986.

[26] Robert Lee and Tse-Tong Chia. Analysis of electromagnetic scattering from a cavity with a complex termination by means of a hybrid ray-FDTD method. *IEEE Transactions on Antennas and Propagation*, 41(11):1560–1569, November 1993.

[27] Hao Ling. RCS of waveguide cavities: A hybrid boundary integral/modal approach. *IEEE Transactions on Antennas and Propagation*, 38(9):1413–1420, September 1990.

[28] Hao Ling, Ri-Chee Chou, and Shung-Wu Lee. Shooting and bouncing rays: Calculating the RCS of an arbitarily shaped cavity. *IEEE Transactions on Antennas and Propagation*, 37(2):194–205, February 1989.

[29] Hao Ling, Shung-Wu Lee, and Ri-Chee Chou. High frequency RCS of open cavities with rectangular and circular cross sections. *IEEE Transactions on Antennas and Propagation*, 37(5):648–654, May 1989.

[30] Jian Liu and Jian-Ming Jin. A special higher-order finite element method for scattering by large cavities. In *2000 IEEE International Symposium*, volume 2, pages 1168–1171. Antennas and Propagation Society, July 2000.

[31] Jian Liu et al. Computation of radar cross section of jet engine inlets. *Microwave and Optical Technology Letters*, 33(5):322–325, June 1994.

[32] Andrew Lewis Maffett. *Topics for a Statistical Description of Radar Cross Section.* Wiley Series in Remote Sensing. Wiley, 1989.

[33] Louis N. Medgyesi-Mitschang and Dau-Sing Wang. Hybrid solutions for scattering from perfectly conducting bodies of revolution. *IEEE Transactions on Antennas and Propagation*, 31(4):570–583, July 1983.

[34] Joe Pacheco Jr. Finite difference techniques for body of revolution radar cross section. Master's thesis, Massachusetts Institute of Technology, May 2000.

[35] P. H. Pathak and R Burkholder. Modal, ray and beam techniques for analyzing the EM scattering by open-ended waveguide cavities. *IEEE Transactions on Antennas and Propagation*, 37(5):635–647, May 1989.

[36] Dennis Prather and Shouyuan Shi. Formulation and application of the finite-difference time-domain method for the analysis of axially symmetric diffractive optical elements. *Journal of the Optical Society of America*, 16(5):1131–1142, May 1999.

[37] Juan Rius, Angel Lozano, Lluis Jofre, and Angel Cardama. Spectral iterative algorithm for RCS computation in electrically large or intermediate perfectly conducting cavities. *IEEE Transactions on Antennas and Propagation*, 42(6):790–797, June 1994.

[38] K. L. Shlager and J. B. Schneider. A selective survey of the finite-difference time-domain literature. *IEEE Antennas and Propagation Magazine*, 37(4):39–56, 1995.

[39] Dennis M. Sullivan. *Electromagnetic Simulation Using the FDTD Method.* IEEE, 1st edition, 2000.

[40] Allen Taflove. *The Finite-Difference Time-Domain.* Artech House, 1995.

[41] Gary A. Thiele and T. H. Newhouse. A hybrid technique for combining moment methods with a geometrical theory of diffraction. *IEEE Transactions on Antennas and Propagation*, 23:62–69, 1975.

[42] Derek Warren Truesdale. Finite-difference time-domain analysis of horn antenna scattering. Master's thesis, Massachusetts Institute of Technology, May 1998.

[43] T. Wang and H. Ling. Electromagnetic scattering from three-dimensional cavities via a connection scheme. *IEEE Transactions on Antennas and Propagation*, 39(10):1505–1513, October 1991.

[44] Zhonggui Xiang and Tse-Tong Chia. A hybrid BEM/WTM approach for analysis of the EM scattering from large open-ended cavities. *IEEE Transactions on Antennas and Propagation*, 49(2):165–172, February 2001.

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