Fast Solution of Large-Body Problems Using Domain Decomposition and Null-Field Generation in the Method of Moments

Except where reference is made to the work of others, the work described in this dissertation is my own or was done in collaboration with my advisory committee. This dissertation does not include proprietary or classified information.

Tyler N. Killian

Certificate of Approval:

Michael E. Baginski Associate Professor Electrical Engineering	Sadasiva M. Rao, Chair Professor Electrical Engineering
Lloyd S. Riggs Professor Electrical Engineering	Tin-Yau Tam Professor Mathematics

George T. Flowers
Dean
Graduate School

Fast Solution of Large-Body Problems Using Domain Decomposition and Null-Field Generation in the Method of Moments

Tyler N. Killian

A Dissertation

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Doctor of Philosophy

Auburn, Alabama
December 18, 2009

Fast Solution of Large-Body Problems Using Domain Decomposition and Null-Field Generation in the Method of Moments

Tyler N. Killian

Signature of Author

Date of Graduation

Vita

Tyler N. Killian, son of John and Robin Killian, was born on March 9, 1983, in Gadsden, Alabama. He attended high school at Sand Rock High School in Sand Rock, Alabama and graduated in 2001. After high school he attended Auburn University where he graduated Summa Cum Laude with a Bachelor's degree in electrical engineering in 2005. In the fall semester 2005, he entered graduate school at Auburn University and obtained the Master's degree in electrical engineering in the summer of 2008. The following semester he entered the Ph.D. program in electrical engineering at Auburn University.

Dissertation Abstract

Fast Solution of Large-Body Problems Using Domain Decomposition and Null-Field Generation in the Method of Moments

Tyler N. Killian

Doctor of Philosophy, December 18, 2009
(M.S., Auburn University, 2008)
(B.S., Auburn University, 2005)

104 Typed Pages

Directed by Sadasiva M. Rao

In this work, a new Method of Moments (MoM) solution procedure for calculating electromagnetic scattering and radiation by electrically large conducting bodies is presented. By using domain-decomposition, conducting structures are divided into several disjoint pieces. By replacing basis functions on each piece of the structure with specially designed functions, null fields may be produced on surrounding areas thereby decoupling sections of the geometry. Also, the geometrical divisions induce a partitioning on the overall system matrix. By creating these null fields, the blocks in the system matrix with the largest element values are eliminated. The result is a block-diagonally-dominant moment matrix that can be used in an iterative procedure for rapid convergence. Furthermore, due to the nature of the algorithm, the solution procedure can be divided cleanly among multiple processors for extra savings in CPU resources. Finally, since an iterative procedure is employed, the large memory requirements typical in MoM problems can be effectively sidestepped.

Acknowledgments

To the author's parents, he expresses his appreciation for their love and encouragement throughout life.

To his advisor, Dr. Sadasiva M. Rao, he thanks for his patience and top-notch guidance. His expertise has been invaluable for this work.

To Dr. Michael Baginski, he thanks for his encouragement and interesting technical discussions.

To the Naval Research Laboratory, he expresses his appreciation for the financial support that made this work possible as well as for their helpful technical suggestions.

To the NASA Langley Research Center, he thanks for the GSRP fellowship and travel funding that allowed him to present his research at technical conferences.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliography follows van Leunen's *A Handbook for Scholars*.

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

Chapter 1

Introduction and Overview

In this chapter, we will introduce a basic overview of the technique as well as the motivation behind it. Furthermore, the strengths and weaknesses of current and previous techniques will be highlighted. The general concepts used in the technique will be given as well as the simple steps used to achieve accurate currents for large-body problems. Furthermore, this chapter acts as an outline for applying the method to the various formulations in the following chapters.

1.1 Motivation

The Method of Moments [1] is an accurate method used in electromagnetics to solve for the electric currents induced on an object due to an incident field. The object of interest is discretized for representation on a computer. Usually, this is either done with a triangular mesh for three-dimensional objects or line segments for two-dimensional objects. Since the Method of Moments is a Boundary Element Method (BEM), then only the object under study needs to be discretized. Once this is done, a set of N basis functions are defined on the object to represent the desired currents. Finally, a system matrix representing the interaction between each pair of basis functions is constructed. This matrix can be inverted to solve the linear system relating the currents to the incident field. Filling the matrix typically requires $O(N^2)$ operations while the inversion procedure requires $O(N^3)$ operations. For large-body problems, these steps represent a problem both in terms of storage and solution

speed. In order to address these issues, we need a method that can solve the problem piecemeal to reduce storage and also a solver requiring much fewer operations than full matrix inversion or LU decomposition.

1.2 Current Methods

There are currently a variety of methods used to combat the aforementioned issues with the Method of Moments. Brute force methods, such as using supercomputers, attempt to directly solve the linear system using multiple processors [10]. Furthermore, they can solve large systems by storing the system matrix on hard disk. However, this method scales very poorly. Parallel LU decomposition cannot be fully parallelized and therefore the efficient gains become smaller with each additional processor. Furthermore, large problems can require hundreds of gigabytes of storage if the full system matrix is generated. Another method is the use of entire domain basis functions. Here, functions are defined over larger parts of the geometry. Using this technique, fewer functions can be used on the overall geometry. The downside is that these functions can only be defined for canonical shapes such as squares, circles, etc. If the geometry cannot be represented in terms of these shapes, the functions cannot be utilized. Another popular method is the Fast Multipole Method (FMM) [8]. Here, the geometry is broken up and basis functions are grouped together. By using wave translation, each set of functions is allowed to radiate "as a group" thereby allowing us to avoid calculating all the individual elements in the system matrix when using an iterative solver such as the conjugate gradient method. By accelerating the vector products in the iterative solver, large problems have been solved [9]. Since the method approximates parts of the system matrix, it may not be accurate when solving for precise current values on a scatterer. Furthermore, the iterative procedure is typically

the conjugate gradient solver [12], which can display erratic or slow convergence and numerical problems due to precision round-off. An additional method is to generate characteristic basis functions, where entire domain functions are generated at run time [11]. Here, the structure is broken up into pieces. For each piece, an incident wave is applied at various angles and the resulting currents are found for each case. Each current solution then becomes a basis function. A singular value decomposition is then used to find the minimal set of basis functions from this collection that is necessary to represent the currents on that part of the structure. Once this has been done for each piece of the structure, all the functions are collected and a new system matrix is generated and solved using these new functions. If there are very few basis functions, the new linear system can be solved very efficiently. However, since each piece is considered separately, there is no guarantee that the new basis functions can properly account for the interaction with the rest of the structure. Furthermore, for complex shapes, there is no guarantee that the number of unknowns will be much less than when sub-domain functions are used. Also, research has been done in trying to create a sparse system matrix [16],[17]. In [16], a special set of basis functions is used where the functions produce a highly directional field pattern with small sidelobes. Test functions in line with the sidelobes will then produce small matrix elements. The elements are then simply dropped since they are small in magnitude. Although this is an interesting idea, it appears that the system matrix becomes ill-conditioned for the new basis functions. Furthermore, the solution is still sensitive to these matrix elements, irregardless of how small they are and therefore they cannot be dropped. In [17], a set of entire domain functions are developed with coefficients that are solved for at runtime. Although the number of unknowns remains the same, the new functions generate a highly sparse, banded matrix. This system may then be solved without matrix inversion. Unfortunately, the functions are not easily extended

to three-dimensional problems. Finally, attempts have been made to diagonalize the system matrix by a basis change to fully computed entire-domain basis functions [7]. In this work, we use the concepts introduced in [6] and [7] for decoupling various parts of the structure. However, we do not attempt to form a full eigenfunction solution over the entire structure as that is not computationally efficient. Instead, we decouple grouped basis functions according to a nearest neighbor criteria. By applying these concepts in a different way, we are able to form a much more efficient solution.

1.3 Concepts

In this section, the basic concepts used in the method will be covered. First, it is imperative that sub-domain functions be used in the original MoM formulation. The original MoM matrix will form the basis for the method. Here, coupling between functions depends directly on their spatial separation. Typically, the coupling falls off as $\frac{1}{R}$ or $\frac{1}{R^2}$ where R is the distance between two functions. Thus, the largest elements in the system matrix are for those functions which are closest together spatially. Our strategy will be to eliminate those elements in the system matrix which are largest. In order to accomplish this, first functions will be grouped spatially using a nearest neighbor criteria. The system matrix can then be rearranged such that elements in the same group are in the same partitions in the matrix. Next, we take each group one at a time. Adjacent groups within a given radius (usually a few wavelengths) are considered "near-field groups", while those outside are called "far-field groups." We wish to decouple our chosen group from those groups which are in the near-field region. We generate a new set of basis functions on the group to replace the sub-domain functions. These functions consist of a linear combination of the original sub-domain functions with the added criteria that they produce a null field on the

groups in the near-field. This corresponds to creating zeros in the system matrix thereby eliminating the largest elements. Once this procedure has been carried out for each group, we are left with a new system matrix with partitions corresponding to the constructed groups. Furthermore, this final matrix will be block-diagonally-dominant due to the elimination of near-field groups. This matrix will converge very quickly in an iterative scheme making use of its structure, such as the Gauss-Seidel scheme. Computing the new basis essentially results in solving for the near-field interaction while the iterative procedure solves for the far-field interactions. Furthermore, the iterative procedure will take very few iterations allowing us to speed up the solution procedure. So, by making use of the physics of the problem, we can solve the near-field and far-field problems separately enabling us to maintain accuracy where it is most needed and gain speed where accuracy is not as essential. Finally, each process along the way can be subdivided cleanly so that the method is amenable to parallel processing.

1.4 General Procedure

The procedure begins with a standard MoM formulation using sub-domain basis functions. Sub-domain basis functions are defined only over a small portion of the geometry and are typically on the order of a tenth of a wavelength in size. This requirement ensures that when two basis functions are separated by a large distance, the corresponding matrix elements representing the coupling between the functions will be small in magnitude. Next, the functions are placed into multiple disjoint groups. These groups may be decided by the geometry of the problem. For example, for an antenna array, the functions on each array element may be a group in the system. Otherwise, we may simply group elements which are electrically close to one

another. The functions within each group are then renumbered if necessary so that these groups effectively partition the system matrix so that each group will correspond to one self-block along the diagonal. Once the groups have been properly formed, we select a "source" group and then begin creating a new set of basis functions for that group. We choose "test" points (points where the new basis functions will create null fields) corresponding to those elements in the MoM matrix which are the largest. Since our original basis functions are sub-domain functions, the largest elements in the matrix will correspond to functions which are geometrically close to one another on the structure. Consequently, we establish a nearest neighbor criteria where groups that are within a predetermined radius (typically a few wavelengths) are included as test groups. Since these groups are close to the source, the coupling between these elements and the source elements will be significant and the corresponding matrix elements will be large. Once the test points have been chosen, we form a new set of basis functions to replace the source group functions. Each new basis function is a linear combination of a sub-domain function from the source group, which is given a coefficient of one, and all the functions from the test groups, each of which is assigned a thus far undetermined coefficient. To solve for these unknown coefficients, or weights, for a source group of size K, we form K linear systems:

$$\begin{bmatrix} Z_{t_1t_1} & Z_{t_1t_2} & \cdots & Z_{t_1t_n} \\ Z_{t_2t_1} & Z_{t_2t_2} & & \\ \vdots & & \ddots & \\ Z_{t_nt_1} & & & Z_{t_nt_n} \end{bmatrix} \begin{bmatrix} \alpha_{1,j} \\ \vdots \\ \\ \alpha_{n,j} \end{bmatrix} = - \begin{bmatrix} Z_{t_1s_j} \\ \vdots \\ \\ Z_{t_ns_j} \end{bmatrix} \tag{1.1}$$

where $j = 1, \ldots, K$ and there are n testing locations and the $Z_{m,n}$ elements come from the MoM system matrix. Also, the column vectors $[\alpha_{i,j}]$ with $i = 1, \ldots, n$ represent

the desired coefficients for forming the new basis functions. Notice that the matrix to be inverted is just a principal submatrix of the original MoM matrix with the indices of the chosen test functions. Also, since there are n unknowns and n equations, this system can be solved exactly if the matrix is nonsingular. Since the matrix effectively represents solving a smaller method of moments problem, the matrix will be nonsingular. Furthermore, as we move from one new source basis to another (as j changes) in the same source group, only the right hand side of Eq. 1.1 is modified. Therefore, for each source group, it is only necessary to invert one matrix. Finally, in most situations, the number of test points is larger than the number of functions in the source group. In this case, solving Eq. 1.1 can be made more efficient by using LU decomposition and then using forward/backward substitution for multiple right-hand sides.

Next, we let f_i for $i = 1, \ldots, K$ be the original sub-domain functions and g_i for $i = 1, \ldots, K$ the new set of functions. The new functions are given by:

$$\begin{bmatrix} g_{s_1} \\ g_{s_2} \\ \vdots \\ g_{s_K} \end{bmatrix} = \begin{bmatrix} f_{s_1} \\ f_{s_2} \\ \vdots \\ f_{s_K} \end{bmatrix} + \begin{bmatrix} \alpha_{1,1} & \alpha_{2,1} & \cdots & \alpha_{n,1} \\ \alpha_{1,2} & \alpha_{2,2} & & \\ \vdots & & \ddots & \\ \alpha_{1,K} & & & \alpha_{n,K} \end{bmatrix} \begin{bmatrix} f_{t_1} \\ f_{t_2} \\ \vdots \\ f_{t_n} \end{bmatrix} \tag{1.2}$$

Note that the $[\alpha]$ column vectors in Eq. 1.1 become rows in the matrix seen in Eq. 1.2. Finally, this process is carried out for each group in the structure.

By changing our source functions, we create a new system matrix $\tilde{Z}$. This can be represented formally as:

$$\tilde{Z} = ZR \tag{1.3}$$

where Z is the original MoM system matrix for the sub-domain functions and R is a sparse matrix representing a basis change from the sub-domain functions to the new, null-field producing functions. R is constructed as follows. Note, each column represents a new basis function. Since each new function is a linear combination of the original functions we have

$$g_i = \sum_{j=1}^{N} \beta_j f_j \tag{1.4}$$

where N is the number of unknowns. Column i in R will then be $[\beta_1, \beta_2, \ldots, \beta_N]^T$. Using our procedure, the replaced source function is always assigned a weight of 1 when the new functions are created. This means R will have ones along the diagonal. Furthermore, test points are only chosen at locations near the source, not over the entire structure. For points not chosen, we simply place a zero in R. Consequently, R will be a sparse matrix, becoming more sparse as the system size increases. As an example, suppose we have a system size of $N = 5$ with sub-domain functions f_1, f_2, f_3, f_4, f_5. In this case, Z, $\tilde{Z}$, and R are all 5×5 matrices. Now suppose we wish to replace f_2 with a function $g_2 = f_2 + \alpha_1 f_1 + \alpha_2 f_3 + \alpha_3 f_5$ while leaving f_1, f_3, f_4, f_5 untouched. In this case we have

$$R = \begin{bmatrix} 1 & \alpha_1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \alpha_3 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

where f_1, f_3, f_4, f_5 are transformed into themselves and f_2 is transformed into g_2. Note that although the subdomain functions were created with the purpose of representing the electric current, they still form a true basis in that they may represent

any function, including other basis functions. In this way, any basis function can be constructed from the original sub-domain functions without having to re-derive a new mathematical formulation from scratch for the new functions.

Using this procedure, we may transform all of the original basis functions (using only one R matrix), and thereby create a new system matrix. If we have M groups and our original MoM matrix is partitioned as

$$\begin{bmatrix} Z_{G_1,G_1} & Z_{G_1,G_2} & \cdots & Z_{G_1,G_M} \\ Z_{G_2,G_1} & Z_{G_2,G_2} & & \\ \vdots & & \ddots & \\ Z_{G_M,G_1} & & & Z_{G_M,G_M} \end{bmatrix} \begin{bmatrix} I_{G_1} \\ I_{G_2} \\ \vdots \\ I_{G_M} \end{bmatrix} = \begin{bmatrix} V_{G_1} \\ V_{G_2} \\ \vdots \\ V_{G_M} \end{bmatrix} \tag{1.6}$$

where the groups G_i for $i = 1, \ldots, M$ may vary in size, then the new system matrix will look similar to the following:

$$\tilde{Z} = \begin{bmatrix} \tilde{Z}_{G_1,G_1} & [0] & \tilde{Z}_{G_1,G_3} & \tilde{Z}_{G_1,G_4} & \cdots & [0] \\ [0] & \tilde{Z}_{G_2,G_2} & & & & \\ \tilde{Z}_{G_3,G_1} & [0] & \tilde{Z}_{G_3,G_3} & & & \\ \tilde{Z}_{G_4,G_1} & \tilde{Z}_{G_4,G_2} & & \tilde{Z}_{G_4,G_4} & & \\ \vdots & & & & \ddots & [0] \\ [0] & & & & & \tilde{Z}_{G_n,G_n} \end{bmatrix} \begin{bmatrix} \tilde{I}_{G_1} \\ \tilde{I}_{G_2} \\ \vdots \\ \tilde{I}_{G_M} \end{bmatrix} = \begin{bmatrix} V_{G_1} \\ V_{G_2} \\ \vdots \\ V_{G_M} \end{bmatrix} \tag{1.7}$$

An important observation here is that certain partitions have been replaced with zeros due to our choice of basis functions. Furthermore, the change of basis for a group has the effect of creating zeros in that group's block-column. Also, these zeros produce nulls at locations which are close geometrically, but need not be close in the system matrix. In fact, we do not have a banded matrix in general. Finally, solving

this system gives us the coefficients for the new basis functions, not the original sub-domain functions. Our original system is transformed as follows:

$$\begin{aligned} ZI = V &\Rightarrow ZRR^{-1}I = V \\ &\Rightarrow (ZR)(R^{-1}I) = V \\ &\Rightarrow \tilde{Z}\tilde{I} = V \end{aligned} \tag{1.8}$$

Once the new system has been solved, we can obtain the coefficients for the original functions with

$$I = R\tilde{I} \tag{1.9}$$

Since all the largest terms in the system matrix have been eliminated, the newly created matrix is block-diagonally-dominant. Roughly, this means that the elements of the diagonal blocks are much larger than the off-diagonal elements. We begin solving this system by inverting the diagonal blocks and solving for currents as if the non-diagonal blocks were zero:

$$\begin{bmatrix} \tilde{Z}_{G_1} & 0 & \cdots & 0 \\ 0 & \tilde{Z}_{G_2} & & \\ \vdots & & \ddots & \\ 0 & & & \tilde{Z}_{G_M} \end{bmatrix} \begin{bmatrix} \tilde{I}_{G_1} \\ \\ \vdots \\ \tilde{I}_{G_M} \end{bmatrix} = \begin{bmatrix} V_{G_1} \\ \\ \vdots \\ V_{G_M} \end{bmatrix} \tag{1.10}$$

In effect, what we are doing is solving the system by taking into account only the near-field interaction between elements. Since this type of interaction is very complex and involves multiple reflections, we must solve this part of the problem exactly. Generally, this solution will be fairly accurate overall and will lead to a more quickly converging solution when used as the initial guess in an iterative scheme. An additional iterative

scheme may be useful because the off-diagonal blocks are not necessarily zero and therefore it may be necessary to go through an additional step to further enhance the accuracy of our solution.

Next, we take the initial current guess and use it as the starting point for the Block-Gauss-Seidel iterative scheme as described in Appendix A. Since our matrix $\tilde{Z}$ is block-diagonally-dominant and our iterative solver favors this type of matrix, it will quickly converge to a solution. Typically, this requires only a few iterations. Furthermore, this scheme has an interesting physical interpretation. To solve for a new current value $\tilde{I}_{G_j}^{k+1}$, we use

$$\tilde{I}_{G_j}^{k+1} = \tilde{Z}_{G_j,G_j}^{-1} \left[V_{G_j} - \sum_{\substack{i=1 \\ i \neq j}}^{M} \tilde{Z}_{G_j,G_i} \tilde{I}_{G_i}^{k} \right] \tag{1.11}$$

where there are M groups and $\tilde{I}_{G_i}^{k}$ is the previously found solution for $\tilde{I}_{G_i}$. The summation in Eq. 1.11 represents the total field on group j due to the currents on the rest of the structure. This means that Eq. 1.11 represents a modification in current values at location j due to reflections on the structure. Therefore, our initial current guess represents currents due to the incident wave plus all reflections within each group. Further iteration represents additional reflections between the groups. Also, note that since each group usually radiates in all directions, subsequent reflections will be progressively smaller in magnitude as energy is radiated into the surrounding space. Consequently, groups that are spatially separated by large distances will require very few reflections to obtain convergence.

To summarize, we have five basic steps:

1. Group Formation - The geometry of the problem is divided up into disjoint groups. Each group consists of a set of sub-domain basis functions.

2. Create new basis functions - For each group, a new set of basis functions is constructed - one for each member of the group. For a given basis function in the group, the new function will consist of that sub-domain function plus a linear combination of the source functions at all the locations along the structure where null-fields shall be produced. The function inside the group is given a coefficient of one, while those outside (where nulls are produced) are given unknown coefficients or weights. These weights can be solved for exactly by a matrix inversion. Furthermore, only one matrix inversion is necessary for each group.

3. Create new system matrix $\tilde{Z}$ - Once the new functions are generated, a new system matrix is generated. Since the newly generated functions are linear combinations of the original sub-domain functions, the elements of the new system matrix are combinations of those from the sub-domain system matrix. Also, the matrices can be partitioned such that the blocks correspond to the basis groups found in step 1. The new matrix will be block-diagonally-dominant since all the largest elements (due to near-field coupling) have been eliminated. These null blocks are generated by the specially constructed basis functions.

4. Construct initial guess - We construct an initial guess in order to accelerate the convergence of the iterative procedure. Here, we invert the diagonal blocks of the new matrix and solve for the currents on the structure. This amounts to finding the currents due to the incident wave excitation and the most dominant near-field interactions on the structure. Since each group interacts weakly with the ignored groups, this solution will be fairly accurate but can be made as accurate as desired by further iteration.

5. Iterate for accuracy - The initial guess represents an approximate solution, but one can iterate further to obtain more accuracy if necessary. Our iteration scheme is the block adaptation of the Gauss-Seidel solver. Since the matrix is block-diagonally-dominant and the solver is based on this property, it will converge very quickly. Therefore, very few iterations will be necessary to get a sufficiently accurate solution.

Notice that the method makes no mention of any specific MoM formulation. In fact, as long as the basis functions are sub-domain functions, the MoM system matrix will have a diagonally strong structure. Furthermore, the functions can be properly grouped and the above procedure may be applied. Throughout the remaining chapters, the procedure will be applied to a variety of formulations, including both two-dimensional and three-dimensional problems.

1.5 Simple example

Here a simple example will be used to illustrate and clarify the concepts discussed in the previous sections. In the example, there are 2 two-dimensional strips, each carrying three pulse functions as shown in Figure 1.1. We will place the functions on each strip into 2 separate groups: $G_1 = \{f_1, f_2, f_3\}$ and $G_2 = \{f_4, f_5, f_6\}$.

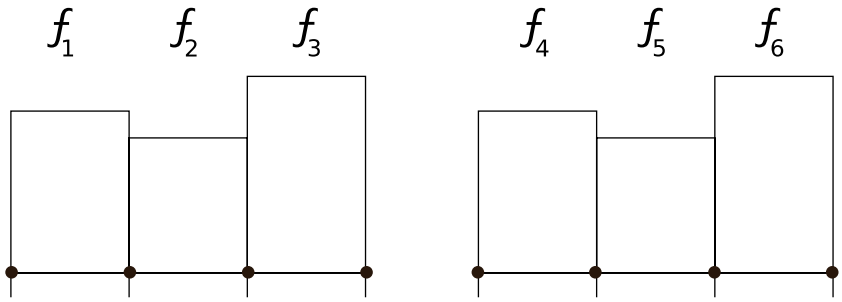

Figure 1.1: Pulse functions on two 2D strips.

Here, the geometry provides us with a convenient group selection. For an arbitrary case, we simply create groups with functions that are geometrically close to one another on the given structure.

Our choice of groups induces a natural partitioning on the MoM matrix. The original sub-domain system matrix is given by:

$$
\begin{aligned}
Z &= \left[\begin{array}{cc}
\left[\begin{array}{ccc} Z_{1,1} & Z_{1,2} & Z_{1,3} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} \end{array} \right] &
\left[\begin{array}{ccc} Z_{1,4} & Z_{1,5} & Z_{1,6} \\ Z_{2,4} & Z_{2,5} & Z_{2,6} \\ Z_{3,4} & Z_{3,5} & Z_{3,6} \end{array} \right] \\
\left[\begin{array}{ccc} Z_{4,1} & Z_{4,2} & Z_{4,3} \\ Z_{5,1} & Z_{5,2} & Z_{5,3} \\ Z_{6,1} & Z_{6,2} & Z_{6,3} \end{array} \right] &
\left[\begin{array}{ccc} Z_{4,4} & Z_{4,5} & Z_{4,6} \\ Z_{5,4} & Z_{5,5} & Z_{5,6} \\ Z_{6,4} & Z_{6,5} & Z_{6,6} \end{array} \right]
\end{array} \right] \\
&= \left[\begin{array}{cc} [Z_{G_1G_1}] & [Z_{G_1G_2}] \\ [Z_{G_2G_1}] & [Z_{G_2G_2}] \end{array} \right]
\end{aligned}
\tag{1.12}
$$

where $Z_{G_iG_j}$ is the partition consisting of all the sub-domain MoM matrix elements with sources from group G_j and test points on group G_i. In order for Z to be block-diagonally dominant, the elements in $Z_{G_1G_1}$ and $Z_{G_2G_2}$ should be much larger in magnitude than the elements in $Z_{G_1G_2}$ and $Z_{G_2G_1}$. To eliminate $Z_{G_2G_1}$, we create three new entire-domain basis functions to replace the sub-domain functions on Group 1, which we refer to as the source group. These will be given by:

$$
\begin{aligned}
g_1 &= f_1 + \alpha_1 f_4 + \beta_1 f_5 + \gamma_1 f_6 \\
g_2 &= f_2 + \alpha_2 f_4 + \beta_2 f_5 + \gamma_2 f_6 \\
g_3 &= f_3 + \alpha_3 f_4 + \beta_3 f_5 + \gamma_3 f_6
\end{aligned}
\tag{1.13}
$$

The weights α_i, β_i, and γ_i are chosen such that their net effect is to produce null fields at every point on the second strip. We will refer to Group 2 as the test group. To give an example of how we find the coefficients for the new basis functions, we will solve for g_1. Mathematically, we can represent the null-field generation using the following criteria:

$$\begin{aligned} Z_{4,1} + \alpha_1 Z_{4,4} + \beta_1 Z_{4,5} + \gamma_1 Z_{4,6} &= 0 \\ Z_{5,1} + \alpha_1 Z_{5,4} + \beta_1 Z_{5,5} + \gamma_1 Z_{5,6} &= 0 \\ Z_{6,1} + \alpha_1 Z_{6,4} + \beta_1 Z_{6,5} + \gamma_1 Z_{6,6} &= 0 \end{aligned} \tag{1.14}$$

We can obtain the coefficients α_1, β_1, and γ_1 with the following relationship:

$$\begin{bmatrix} \alpha_1 \\ \beta_1 \\ \gamma_1 \end{bmatrix} = - \begin{bmatrix} Z_{4,4} & Z_{4,5} & Z_{4,6} \\ Z_{5,4} & Z_{5,5} & Z_{5,6} \\ Z_{6,4} & Z_{6,5} & Z_{6,6} \end{bmatrix}^{-1} \begin{bmatrix} Z_{4,1} \\ Z_{5,1} \\ Z_{6,1} \end{bmatrix} \tag{1.15}$$

From this relationship, we can make a couple of important observations. First, the matrix to be inverted is identical to the system matrix for the standard MoM problem involving only the segments 4, 5, and 6. Since this problem is well-defined, the inverse must exist and we can therefore solve for the coefficients exactly. Furthermore, if we try to solve for g_2 or g_3, we will use the same matrix, but will multiply it by a different column vector. Therefore, only one matrix must be inverted to solve for the coefficients for each group.

Once we solve Eq. 1.15, we obtain α_1, β_1, and γ_1 and thereby generate the new basis function g_1. In a similar fashion, we can form g_2 and g_3. These functions replace f_1, f_2 and f_3. Likewise, we can replace f_4, f_5, and f_6 with new functions g_4, g_5, and g_6

to produce nulls on the first strip thereby eliminating $Z_{G_1G_2}$. We form a new system matrix $\tilde{Z}$ by a source basis change to the newly formed $g_i's$. The elements in $\tilde{Z}$ are then simply linear combinations of the original elements of Z. For example, if we wish to generate the element $\tilde{Z}_{2,1}$, we compute

$$\tilde{Z}_{2,1} = Z_{2,1} + \alpha_1 Z_{2,4} + \beta_1 Z_{2,5} + \gamma_1 Z_{2,6} \tag{1.16}$$

In matrix notation we create $\tilde{Z}$ with the following basis transformation:

$$\begin{bmatrix} \tilde{Z}_{1,1} & \tilde{Z}_{1,2} & \tilde{Z}_{1,3} & \tilde{Z}_{1,4} & \tilde{Z}_{1,5} & \tilde{Z}_{1,6} \\ \tilde{Z}_{2,1} & \tilde{Z}_{2,2} & \tilde{Z}_{2,3} & \tilde{Z}_{2,4} & \tilde{Z}_{2,5} & \tilde{Z}_{2,6} \\ \tilde{Z}_{3,1} & \tilde{Z}_{3,2} & \tilde{Z}_{3,3} & \tilde{Z}_{3,4} & \tilde{Z}_{3,5} & \tilde{Z}_{3,6} \\ \tilde{Z}_{4,1} & \tilde{Z}_{4,2} & \tilde{Z}_{4,3} & \tilde{Z}_{4,4} & \tilde{Z}_{4,5} & \tilde{Z}_{4,6} \\ \tilde{Z}_{5,1} & \tilde{Z}_{5,2} & \tilde{Z}_{5,3} & \tilde{Z}_{5,4} & \tilde{Z}_{5,5} & \tilde{Z}_{5,6} \\ \tilde{Z}_{6,1} & \tilde{Z}_{6,2} & \tilde{Z}_{6,3} & \tilde{Z}_{6,4} & \tilde{Z}_{6,5} & \tilde{Z}_{6,6} \end{bmatrix} = \begin{bmatrix} Z_{1,1} & Z_{1,2} & Z_{1,3} & Z_{1,4} & Z_{1,5} & Z_{1,6} \\ Z_{2,1} & Z_{2,2} & Z_{2,3} & Z_{2,4} & Z_{2,5} & Z_{2,6} \\ Z_{3,1} & Z_{3,2} & Z_{3,3} & Z_{3,4} & Z_{3,5} & Z_{3,6} \\ Z_{4,1} & Z_{4,2} & Z_{4,3} & Z_{4,4} & Z_{4,5} & Z_{4,6} \\ Z_{5,1} & Z_{5,2} & Z_{5,3} & Z_{5,4} & Z_{5,5} & Z_{5,6} \\ Z_{6,1} & Z_{6,2} & Z_{6,3} & Z_{6,4} & Z_{6,5} & Z_{6,6} \end{bmatrix}$$
$$\times \begin{bmatrix} 1 & 0 & 0 & \alpha_4 & \alpha_5 & \alpha_6 \\ 0 & 1 & 0 & \beta_4 & \beta_5 & \beta_6 \\ 0 & 0 & 1 & \gamma_4 & \gamma_5 & \gamma_6 \\ \alpha_1 & \alpha_2 & \alpha_3 & 1 & 0 & 0 \\ \beta_1 & \beta_2 & \beta_3 & 0 & 1 & 0 \\ \gamma_1 & \gamma_2 & \gamma_3 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

Note that we only need to store the lower left and upper right hand blocks since the remaining partitions are simply identity matrices. Also, note that we only need to store coefficients for locations where we want to produce null fields. In larger problems, we only produce nulls on near-field elements, not the entire structure.

Thus, if we have $\tilde{Z} = ZR$, then R will be very sparse and we will only need a minor amount of storage.

Our final partitioned matrix is:

$$\tilde{Z} = \begin{bmatrix} \tilde{Z}_{G_1G_1} & 0 \\ 0 & \tilde{Z}_{G_2G_2} \end{bmatrix} \tag{1.18}$$

where the off-diagonal blocks are exactly zero by design.

In this example, the system may be solved exactly by simply inverting the sub-blocks $\tilde{Z}_{G_1G_1}$ and $\tilde{Z}_{G_2G_2}$. In a problem where there are several groups, it is unnecessary to produce nulls everywhere outside of a source group. Only the test points in the near-field neighborhood must have nulls since their corresponding matrix elements are the most significant. Additional test points from the remaining groups may be picked up sparsely if one so chooses, although this appears to be unnecessary and adds complexity to the algorithm. The only requirement is that the group blocks corresponding to the self terms dominate the terms in their respective columns and rows. Since $\tilde{Z}$ can be made highly block-diagonally dominant, the system will converge quickly when used in an iterative scheme.

1.6 Parallel Processing

Finally, since each part of the solution procedure can be further divided into computationally separate pieces, the method is highly amenable to parallel processing. In this section, we will discuss how each part of the algorithm may be implemented in a parallel scheme.

In order to construct the new system matrix $\tilde{Z}$, we must first generate the elements of the original matrix Z. Each one of the matrix elements may be computed

completely independently from the other elements. Furthermore, the elements may be distributed evenly across each of the processors allowing for equal work load. Therefore, this part of the algorithm tends to scale linearly with the number of processors used. In the present work, only one block-row of Z was stored in memory at a time. The row was broken up according to the group choice and then the column blocks corresponding to the groups were divided amongst the processors.

When solving for the coefficients used to construct the new basis functions, each source group can be considered separately. Each processor may be assigned a set of groups, each of which requires a single matrix inversion. If the groups are similar in size, they may simply be distributed evenly across the processors. If they are uneven, the processors can each get a different number of groups in order to balance the load. Here, dynamic scheduling may be used so that when a CPU becomes available, it simply advances to the next group and begins solving for weights. In this type of situation, if one CPU becomes occupied with a large set of coefficients, the remaining CPUs can continue to work on smaller groups. Good load balancing can be achieved in this way. For example, one CPU can solve for the coefficients on one group with N null-field points, while another can solve eight groups, each of which has N/2 null-field points. Here, each group solution carries a complexity of $O(N^3)$ where N is the number of null-field points that must be produced for that source group.

Typically, one iteration of the solver takes roughly the same amount of time as does one matrix-vector multiply with the same dimensions as the system size. Therefore, the iteration procedure generally does not require parallel processing since it is a very fast procedure and requires only a few iterations. In this work, this procedure was not made parallel. However, it is possible to create a parallel version if necessary. If the Gauss-Seidel version is used, then as we move from one group to the next solving for currents, we may take each group and assign subsets of the basis

functions to each processor. Each processor can then form the summations given in Eq. A.3. The inverse in Eq. A.3 may then be done by a single CPU. Alternatively, one could use a block-Jacobi iterative solver, which is essentially the same as the Gauss-Seidel solver with the exception that once a set of currents are found for a group, they are not used until the next iteration. The Gauss-Seidel version uses those new current values immediately, leading to faster convergence. In a parallel scheme, the block-Jacobi solver would have the benefit that each group of currents could be found independently and thus the algorithm is slightly more parallel than the Gauss-Seidel version.

1.7 Efficiency

Here we compare the efficiency of the method to a full LU decomposition solution. We assume that we have divided a structure with N basis functions into K groups, each with M unknowns. Furthermore, we assume that each source group produces a total of P null field points. There are three major sources that determine the amount of required memory. First, we must store the coefficients for each group. There are M sets of P coefficients for every source group. Therefore, we must store $K \times M \times P$ complex numbers for the coefficients. Next, we need enough scratch space to solve for the coefficients. This requires enough room to store a $P \times P$ matrix, or P^2 complex values. Also, we must store a block-row of the partitioned matrix for use in the iterative scheme. This requires storing $M \times N$ complex values. So we have a required storage of approximately $KMP + P^2 + MN$ complex values. For the full solution, we must store the entire matrix for a total of N^2 complex values. Note that we have $K \ll N$, $M \ll N$, and $P \ll N$ so that we require much less storage than does storing the entire matrix. For example, suppose that we have $N = 100000$, $K = 400$,

$M = 250$, and $P = 2000$. Then we have $\frac{KMP+P^2+MN}{N^2} = 0.024$. So, in this case, we require less than 3 percent of the storage needed for storing the full matrix.

We can also compare the necessary CPU requirements for the two methods. For a LU decomposition, the number of operations are on the order of N^3. For our method, there are a few essential steps that determine the characteristic CPU scaling of the solution. First, since the number of groups scales linearly with N, then the required operations for finding the coefficients does so as well. Next, creating the new system matrix is equivalent to multiplying the sub-domain system matrix by a sparse matrix. Normally, matrix multiplication is an $O(N^3)$ process, but since one of the matices is highly sparse, this step will scale as N^2. Finally, the algorithm requires generating the original system matrix and this is an N^2 procedure. Therefore, our method is an order of magnitude faster than a full solution. The numerical examples given in this work demonstrate this type of behavior in terms of actual wall clock time.

Chapter 2

Two-dimensional conducting cylinders

In this chapter, the procedure is applied to arbitrarily shaped two-dimensional conducting cylinders, where the cylinders are infinite along the z-axis. First, we develop the integral equations for the electric and magnetic field formulations for both TE and TM polarizations. Next, we show how those are used to construct the combined-field formulation to avoid problems with internal resonances for closed bodies. Furthermore, we discuss how the geometry is decomposed into groups so that the method may be properly applied. Also, we display the technique used to generate null fields so that groups located near one another on the structure may be mathematically decoupled from each other in order to generate a block-diagonally-dominant matrix. Finally, some example problems and results are given to show the effectiveness of the technique.

2.1 Electric-Field Integral Equation - EFIE

In this section, we present the Electric Field Integral Equation for both Transverse Electric (TE) and Transverse Magnetic (TM) incident waves. Also, we represent the induced current in terms of pulse expansion functions and develop the matrix components when testing with pulse functions. Applying the boundary condition that the total tangential electric field must be zero at the surface of the conductor, the

following integral equation for the TM case may be derived [9]:

$$\frac{k\eta}{4}\int_C J_z(\rho')H_o^{(2)}(k|\rho-\rho'|)dl' = E_z^i(\rho') \tag{2.1}$$

where ρ is an observation point and ρ' is a source point along the contour C. Note there is only a z-component of current. We expand this current in terms of pulse functions:

$$J_z(\rho) = \sum_{n=1}^{N} \alpha_n P_n(\rho) \tag{2.2}$$

where the α_n's are the desired unknown coefficients. If we represent the contour C as a collection of line segments (edges), then the pulses are defined as shown in Figure 2.1:

$$P_n(\rho) = \begin{cases} 1 & \rho \in edge\, n \\ 0 & otherwise \end{cases} \tag{2.3}$$

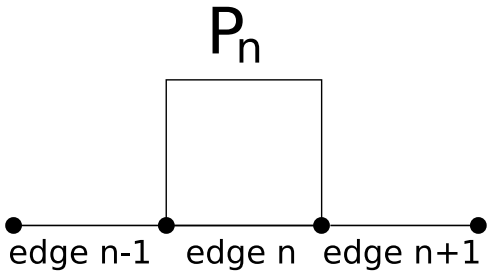

Figure 2.1: Pulse function for two-dimensional TM polarization.

Expanding Eq. 2.1 we arrive at:

$$\frac{k\eta}{4}\sum_{n=1}^{N}\alpha_n\int_C P_n H_o^{(2)}(k|\rho-\rho'|)dl' = E_z^i(\rho) \tag{2.4}$$

Using the Galerkin Method and testing each side of Eq. 2.4 using the inner product

$$\langle \mathbf{a}, \mathbf{b}\rangle = \int_C \mathbf{a}\cdot\mathbf{b}dl \tag{2.5}$$

we obtain the entries for the system matrix:

$$\begin{aligned} Z_{m,n} &= \frac{k\eta}{4}\int_C P_m \int_C P_n H_o^{(2)}(k|\rho-\rho'|)dl' \\ &= \frac{k\eta l_m}{4}\int_C H_o^{(2)}(k|\rho_{ct}^m-\rho'|)dl' \end{aligned} \tag{2.6}$$

where ρ_{ct}^m is the centroid and l_m is the length of the mth edge. The final integral can be evaluated by Gaussian Quadrature. The excitation vector is given by:

$$\begin{aligned} V_m &= \int_C P_m E_z^i(\rho)dl \\ &= l_m E_z^i(\rho_{ct}^m) \end{aligned} \tag{2.7}$$

The TE case begins with the following integral equation [9]:

$$\begin{aligned} \mathbf{E}(\rho)\cdot\hat{a}_l &= \frac{k\eta}{4}\int_C J_l(\rho')H_o^{(2)}(k|\rho-\rho'|)(\hat{a}_l'\cdot\hat{a}_l)dl' \\ &+ \frac{\eta}{4k}\frac{d}{dl}\int_C \frac{dJ_l(\rho')}{dl'}H_o^{(2)}(k|\rho-\rho'|)dl' \end{aligned} \tag{2.8}$$

where C is the contour of the object. Again, as in the TM case, we represent the current in terms of pulse functions, except in this case each function is defined from the midpoint of one segment to the midpoint of an adjacent segment. Also, rather than represent the derivative on J_l in Eq. 2.8 in terms of delta functions, we define the charge in terms of pulse doublets. If P_n, shown in Figure 2.2, is defined as:

$$P_n = \begin{cases} 1 & \rho \in (ct_n, ct_{n+1}) \\ 0 & otherwise \end{cases} \tag{2.9}$$

where ct_n and ct_{n+1} are the centroids of the nth and $n+1$th edges, respectively, then we can define the charge as:

$$\frac{dP_n}{dl} = \begin{cases} \frac{1}{l_n} & \rho \in edge\ n \\ \frac{-1}{l_{n+1}} & \rho \in edge\ n+1 \\ 0 & otherwise \end{cases} \tag{2.10}$$

where l_n and l_{n+1} are the lengths of the nth and $n+1$th edges.

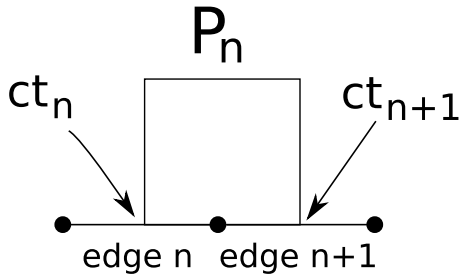

Figure 2.2: Pulse function for two-dimensional TE polarization.

Applying an expansion/testing procedure similar to the TM case we obtain the MoM matrix components:

$$\begin{aligned} Z_{m,n} &= \frac{k\eta}{4}\int_C P_m \int_C P_n H_o^{(2)}(k|\rho-\rho'|)(\hat{a}_l' \cdot \hat{a}_l)dl'dl \\ &+ \frac{\eta}{4k}\int_C P_m \frac{d}{dl}\int_C \frac{dP_n}{dl'} H_o^{(2)}(k|\rho-\rho'|)dl'dl \end{aligned} \tag{2.11}$$

To evaluate the integrals, we can do the following:

$$\begin{aligned} &\int_C P_m \int_C P_n H_o^{(2)}(k|\rho-\rho'|)(\hat{a}_l' \cdot \hat{a}_l)dl'dl \\ &= \frac{l_m}{2}\int_C H_o^{(2)}(k|\rho_{m-1/4}-\rho'|)(\hat{a}_l' \cdot \hat{a}_{l_m})dl' \\ &+ \frac{l_{m+1}}{2}\int_C H_o^{(2)}(k|\rho_{m+1/4}-\rho'|)(\hat{a}_l' \cdot \hat{a}_{l_{m+1}})dl' \end{aligned} \tag{2.12}$$

and

$$\begin{aligned}
&\frac{d}{dl}\int_C \frac{dJ_l(\rho')}{dl'} H_o^{(2)}(k|\rho-\rho'|)dl' \\
&= -\int_C \frac{dP_n}{dl'} H_o^{(2)}(k|\rho_{m+1/2}-\rho'|)dl' \\
&\quad + \int_C \frac{dP_n}{dl'} H_o^{(2)}(k|\rho_{m-1/2}-\rho'|)dl'
\end{aligned} \tag{2.13}$$

where the testing quantities and locations can be seen in Figure 2.3.

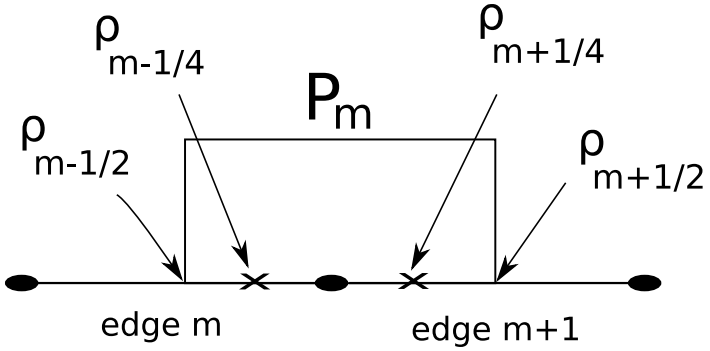

Figure 2.3: Two-dimensional TE polarization testing locations.

The incident field vector is given by:

$$\begin{aligned}
V_m &= \int_C P_m \mathbf{E}^i(\rho)\cdot\hat{a}_l dl \\
&= \frac{l_m}{2}\mathbf{E}^i(\rho_{m-1/4})\cdot\hat{a}_{l_m} + \frac{l_{m+1}}{2}\mathbf{E}^i(\rho_{m+1/4})\cdot\hat{a}_{l_m}
\end{aligned} \tag{2.14}$$

2.2 Magnetic-Field Integral Equation - MFIE

By applying the boundary condition on the magnetic field, we can arrive at another formulation for both the 2D TE and TM polarizations. First, the boundary condition on the magnetic field is:

$$\mathbf{J}(\rho) = \hat{n}\times\mathbf{H}^t(\rho),\ \rho\in C \tag{2.15}$$

where $\mathbf{J}$ is the total current, $\mathbf{H}^t$ is the total magnetic field, and C is the contour of the conductor. The MFIE only applies to closed bodies and thus $\hat{n}$ may be defined as a vector pointing to the outside of the object of interest. The integral equation for the TM polarization case is [9]:

$$\mathbf{H}^i(\rho) \cdot \hat{a}_l = \frac{\mathbf{J}_z(\rho)}{2} + \frac{jk}{4} \oint_C \mathbf{J}_z(\rho') \left(\hat{n} \cdot \frac{\rho - \rho'}{|\rho - \rho'|} \right) H_1^{(2)}(k|\rho - \rho'|) dl' \tag{2.16}$$

where the deleted integral is evaluated everywhere except $\rho = \rho'$. Expanding and testing with pulse functions we obtain the matrix components:

$$Z_{m,n} = \int_C \frac{P_m P_n}{2} dl = \frac{l_m}{2}, \text{ for } m = n \tag{2.17}$$

$$\begin{aligned} Z_{m,n} &= \frac{jk}{4} \int_C P_m \int_C P_n \left(\hat{n} \cdot \frac{\rho - \rho'}{|\rho - \rho'|} \right) H_1^{(2)}(k|\rho - \rho'|) dl' \\ &= \frac{jkl_m}{4} \int_C \left(\hat{n} \cdot \frac{\rho_{ct}^m - \rho'}{|\rho_{ct}^m - \rho'|} \right) H_1^{(2)}(k|\rho - \rho'|) dl', \text{ for } m \neq n \end{aligned} \tag{2.18}$$

where ρ_{ct}^m is the centroid of edge m and the remaining integral may be evaluated using Gaussian Quadrature. The excitation vector is given by:

$$\int_C P_m \mathbf{H}^i(\rho) \cdot \hat{a}_l = l_m \mathbf{H}^i(\rho_{ct}^m) \cdot \hat{a}_{l_m} \tag{2.19}$$

We can also write an integral equation for TE polarization [9]:

$$-\mathbf{H}_z^i(\rho) = \frac{\mathbf{J}_l(\rho)}{2} + \frac{jk}{4} \oint_C \mathbf{J}_l(\rho') \left(\hat{n}' \cdot \frac{\rho - \rho'}{|\rho - \rho'|} \right) H_1^{(2)}(k|\rho - \rho'|) dl' \tag{2.20}$$

Again, the deleted integral is not evaluated at $\rho = \rho'$. Furthermore, as with the EFIE TE case, we expand and test with pulse functions extending from the centroid of one

edge to the centroid of the adjacent edge. After applying the expansion and testing procedure, we arrive at the following MoM matrix components:

$$Z_{m,n} = \int_C \frac{P_m P_n}{2} dl = \frac{1}{2}\left[\frac{l_m}{2} + \frac{l_{m+1}}{2}\right], \text{ for } m = n \tag{2.21}$$

$$\begin{aligned} Z_{m,n} &= \frac{jk}{4}\int_C P_m \int_C P_n \left(\hat{n}' \cdot \frac{\rho - \rho'}{|\rho - \rho'|}\right) H_1^{(2)}(k|\rho - \rho'|)dl' \\ &= \frac{jk}{4}\left[\frac{l_m}{2}\int_C \left(\hat{n}' \cdot \frac{\rho_{m-1/4} - \rho'}{|\rho_{m-1/4} - \rho'|}\right) H_1^{(2)}(k|\rho_{m-1/4} - \rho'|)dl' \right. \\ &+ \left. \frac{l_{m+1}}{2}\int_C \left(\hat{n}' \cdot \frac{\rho_{m+1/4} - \rho'}{|\rho_{m+1/4} - \rho'|}\right) H_1^{(2)}(k|\rho_{m+1/4} - \rho'|)dl'\right], \text{ for } m \neq n \end{aligned} \tag{2.22}$$

For the excitation vector we have:

$$\begin{aligned} V_m &= -\int_C P_m H_z^i(\rho) dl \\ &= -\left[\frac{l_m}{2} H_z(\rho_{m-1/4}) + \frac{l_{m+1}}{2} H_z(\rho_{m+1/4})\right] \end{aligned} \tag{2.23}$$

The testing locations are the same as those given in Figure 2.3. In the next section, we will combine these formulations to eliminate any matrix conditioning problems that may occur due to internal resonances for closed bodies.

2.3 Combined-Field Integral Equation - CFIE

To avoid any problems due to internal resonances for closed bodies, we may use the Combined Field Integral Equation [4] given by

$$-\frac{\gamma}{\eta}\hat{n} \times \hat{n} \times \mathbf{E} + (1 - \gamma)\hat{n} \times \mathbf{H} = 0 \tag{2.24}$$

which can be implemented as

$$\begin{aligned} Z_{CFIE} &= \gamma Z_{EFIE} + \eta(1-\gamma) Z_{MFIE} \\ V_{CFIE} &= \gamma V_{EFIE} + \eta(1-\gamma) V_{MFIE} \end{aligned} \tag{2.25}$$

where $0 \leq \gamma \leq 1$ is a constant depending on the problem and $\eta = \sqrt{\frac{\mu}{\epsilon}}$ is the impedance of the medium. For open body problems, we let $\gamma = 1$ and for closed bodies γ is typically 0.5. Note the same equations may be used for both TM and TE polarizations.

2.4 Element Grouping

For the two-dimensional case, groups are formed along the contour of the cylinder as shown in Figure 2.4. Basis functions along the rim are placed into disjoint sets of roughly equal size. Typically, the group size should be on the order of a few wavelengths. For example, between 2 and 6 wavelengths. The larger the group, the better, as long as the group sizes are not large with respect to the overall geometry. Also, for two-dimensional geometries, a given near-field radius will not enclose many unknowns and so larger group sizes are acceptable. The separation distance between two groups is taken to be the distance from the center of one group to the center of the other. For a given source group, nulls are created on groups within the near-field region (typically a couple of wavelengths). Usually, this means that the two groups adjacent to the source group are eliminated. However, if a cross-section of the cylinder is sufficiently thin, it may be necessary to eliminate more than the two adjacent groups. This can be seen in Figure 2.5.

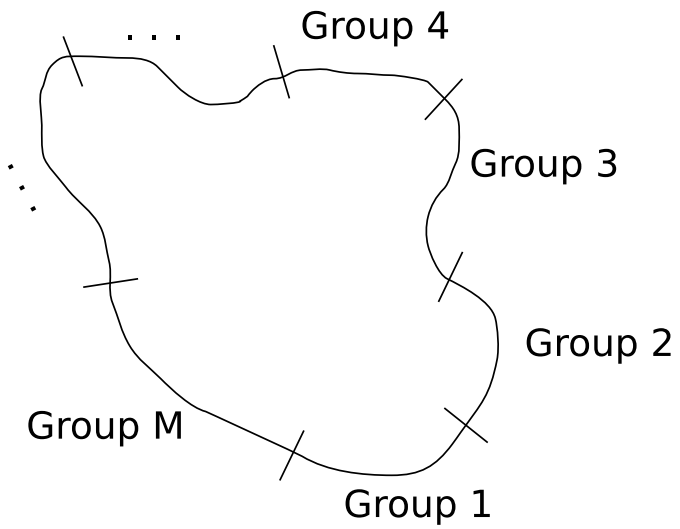

Figure 2.4: Element grouping for two-dimensional case.

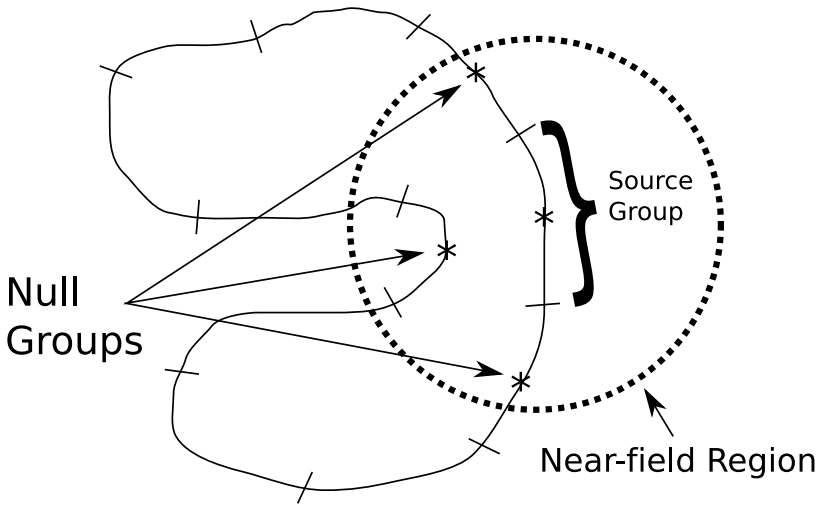

Figure 2.5: Elimination of groups within near-field.

2.5 Numerical Results

The following two-dimensional examples using both TM and TE polarization show the effectiveness of the technique. All closed body solutions utilize the combined-field formulation while open bodies use the electric field formulation for each polarization. Also, both codes use pulse functions for both the source basis and test basis. Finally, the block form of the Gauss-Seidel iterative solver has been used.

Consider the square cylinder with 50λ sides illuminated by a plane wave as shown in the Figure 2.6. The total contour length for this case is 200λ. Figures 2.7 and 2.8 show the real and imaginary currents on Face #1 for an incident wave with TM polarization. The currents for Face #2 are given in Figures 2.9 and 2.10. The TE case is given in Figures 2.11 and 2.12 for Face #1 and Figures 2.13 and 2.14 for Face #2. The amplitude of the incident magnetic field H_o is 1.0 Amps/m for each polarization. The angle of incidence for both cases is 35^0 with respect to x-axis. The contour of the cylinder is divided into 2000 divisions using a 10 divisions per wavelength criterion. Further, the basis functions are collected into 40 groups with 50 basis functions per group. For each source group, the testing is carried out on the two adjacent groups. The initial guess, obtained by computing currents only from the individual groups, is refined with one iteration following the Gauss-Seidel procedure. The currents are shown to be in agreement with the standard MoM solution.

For the next example, consider a highly complex shaped contour with concavity as shown in Figure 2.15. The total circumference is 100λ. The incident wave is at 45^0 with respect to the x-axis with $H_o = 1.0$. There are 25 groups with 40 sub-domain basis functions in each group. Figures 2.16 and 2.17 show the real and imaginary currents for the TM case. Figures 2.18 and 2.19 show the currents for the TE case.

The numerical results, after one iteration, are compared with the standard MoM solution and show good agreement for each case.

Next, we consider the large two-dimensional strip array shown in Figure 2.20. Each element is 2λ in length and there are 100 collinear elements in the array spaced 0.1λ apart. Each element is a group with 20 segments per group resulting in 100 groups. Testing for a given source group is done on the two adjacent elements. If the element is at either end of the array, then testing is done only on the single adjacent element. The incident wave is normal to the array and has magnitude $H_o = 1.0$. Figure 2.21 shows the bistatic radar cross section with respect to the azimuthal angle for the TM case. The TE case is given in Figure 2.22. The results are compared with the conventional MoM solution and good agreement is evident in each case.

Next, we show the real-time results for a 700λ circumference two-dimensional circular cylinder using the new procedure and compare with the conventional method. The scattering case is solved for a 600 MHz incident wave at 180^0 with respect to the x-axis with $H_o = 1.0$ and TM polarization. The cross section of the cylinder is divided into 7000 edges using a 10 divisions per wavelength criterion. For the conventional MoM solution, the matrix fill and execution times are 12 minutes and 189 minutes, respectively. For the new procedure, 7000 unknowns are divided into 70 groups of 100 unknowns each. In the following table, we provide the computational time for each step involved. The processor for the test machine is a 2.4 GHz Pentium 4.

1	Matrix fill	12 minutes 20 seconds
2	Construction of new basis	16.1 seconds
3	Construction of new system matrix	15.7 seconds
4	Iterative solution (single iteration)	15.8 seconds
5	New method total solution time (Add 1,2,3, and 4)	13 minutes 8 seconds

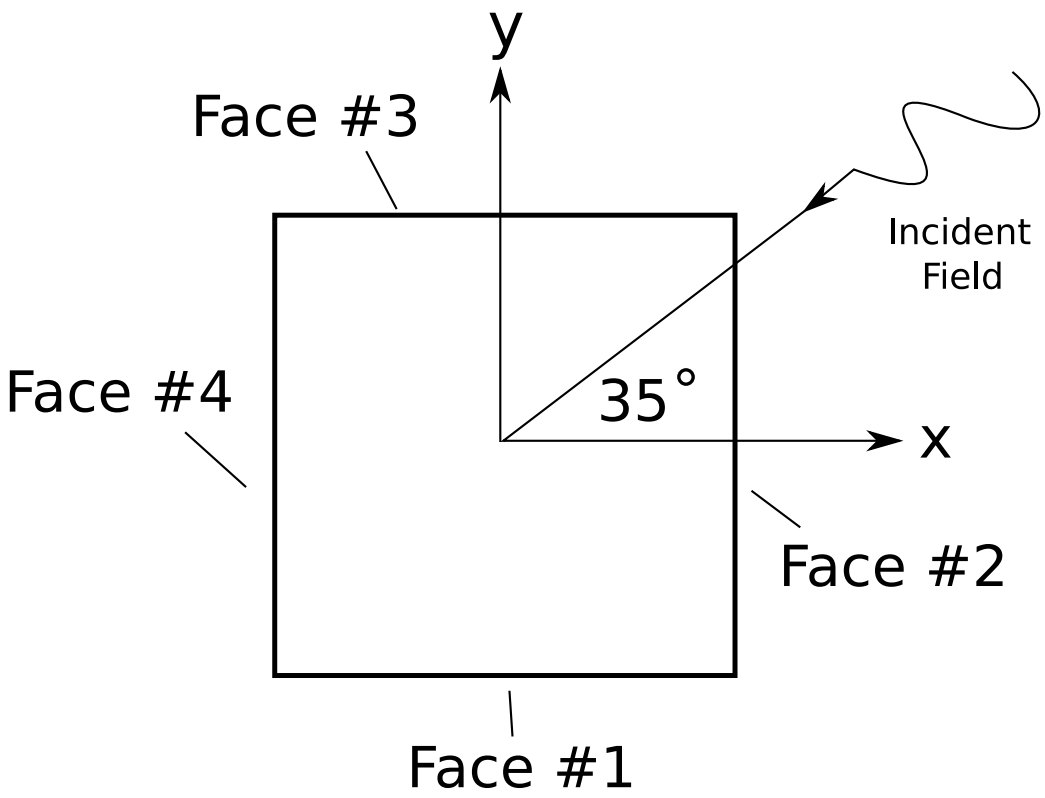

Figure 2.6: Cross section of the 50λ square cylinder.

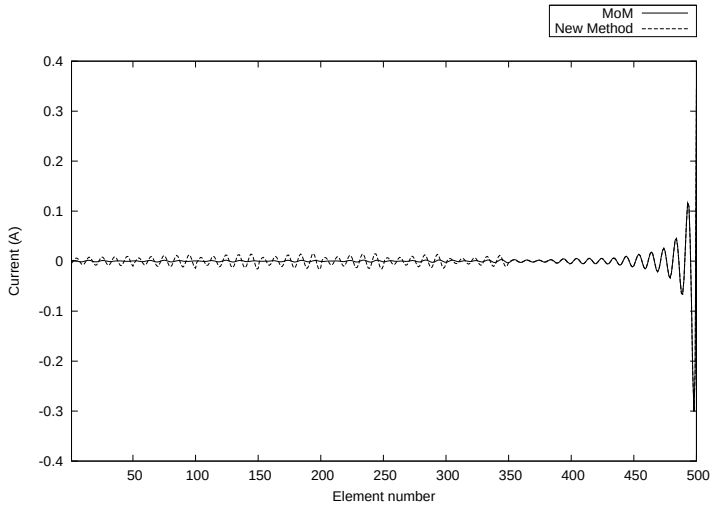

Figure 2.7: Real currents in the shadow region (Face#1) for square cylinder with 50λ sides with TM incident wave $H_o = 1$ at 35^0.

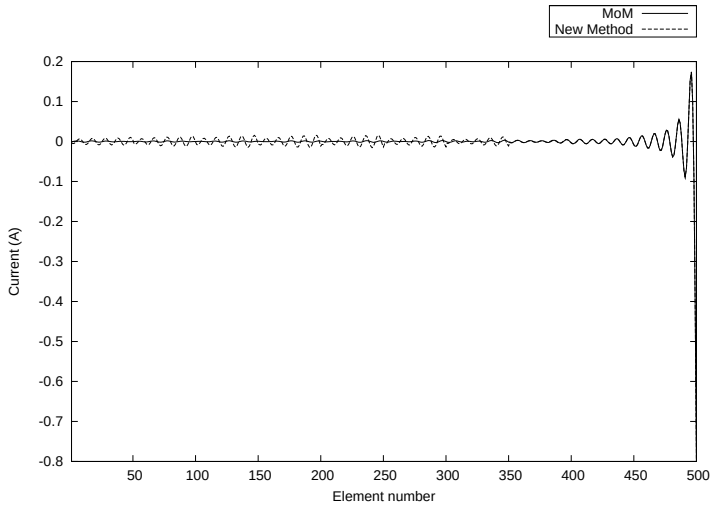

Figure 2.8: Imaginary currents in the shadow region (Face#1) for square cylinder with 50λ sides with TM incident wave $H_o = 1$ at 35^0.

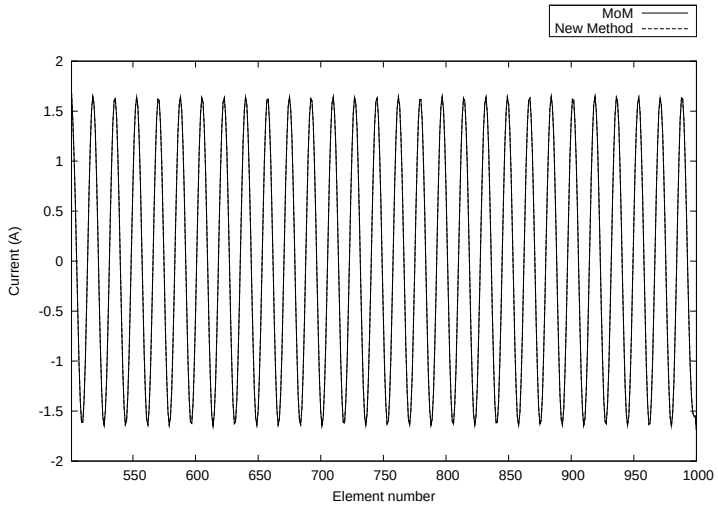

Figure 2.9: Real currents in the shadow region (Face#2) for square cylinder with 50λ sides with TM incident wave $H_o = 1$ at 35^0.

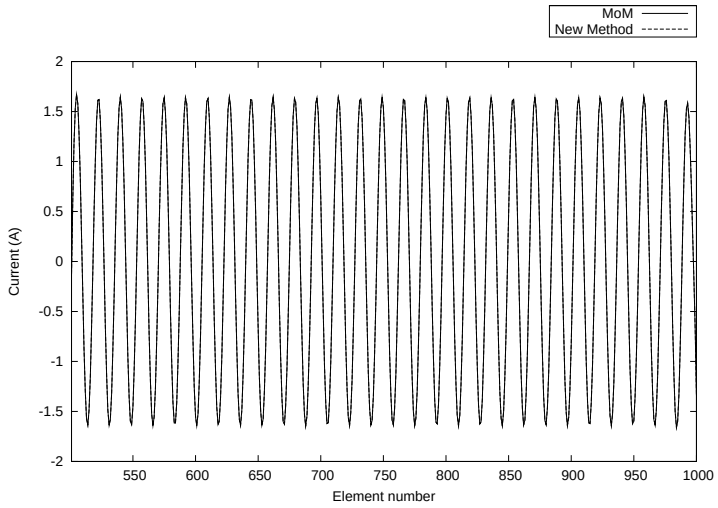

Figure 2.10: Imaginary currents in the shadow region (Face#2) for square cylinder with 50λ sides with TM incident wave $H_o = 1$ at 35^0.

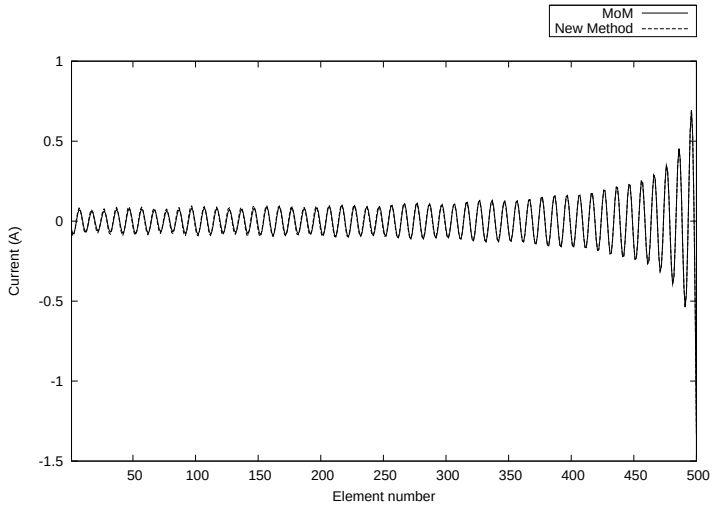

Figure 2.11: Real currents in the shadow region (Face#1) for square cylinder with 50λ sides with TE incident wave $H_o = 1$ at 35^0.

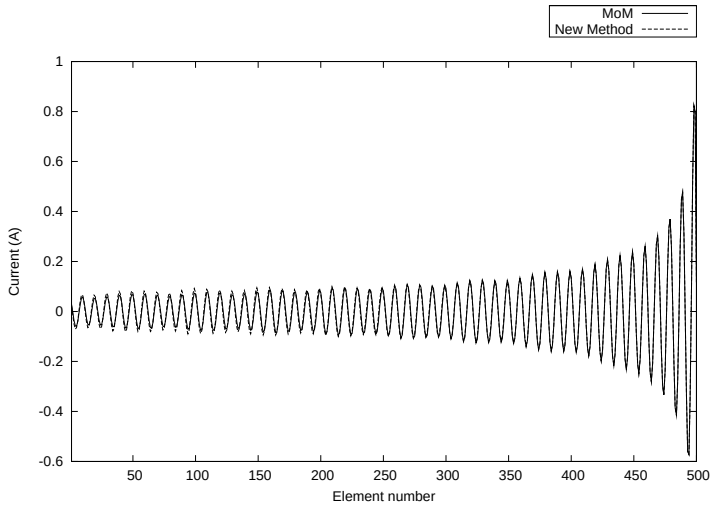

Figure 2.12: Imaginary currents in the shadow region (Face#1) for square cylinder with 50λ sides with TE incident wave $H_o = 1$ at 35^0.

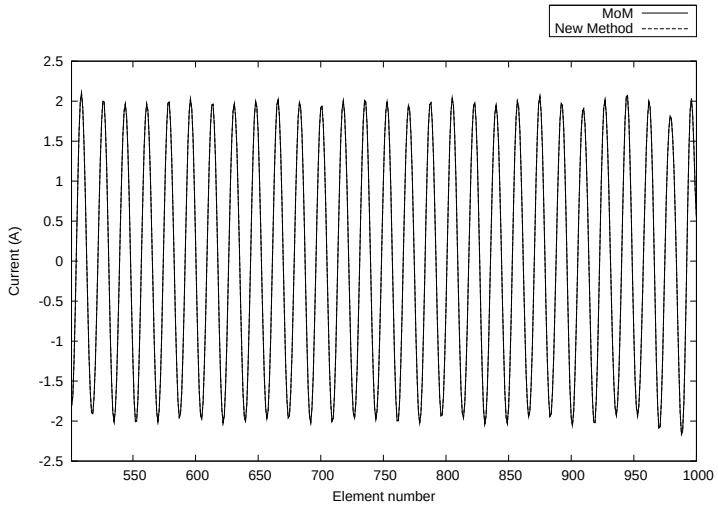

Figure 2.13: Real currents in the shadow region (Face#2) for square cylinder with 50λ sides with TE incident wave $H_o = 1$ at 35^0.

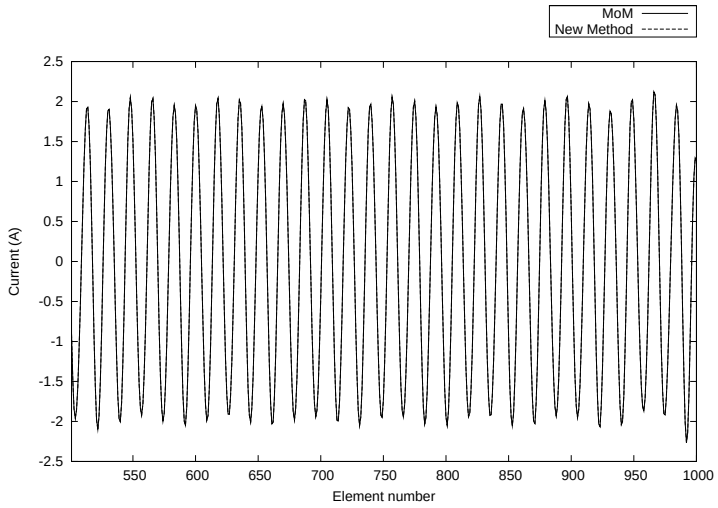

Figure 2.14: Imaginary currents in the shadow region (Face#2) for square cylinder with 50λ sides with TE incident wave $H_o = 1$ at 35^0.

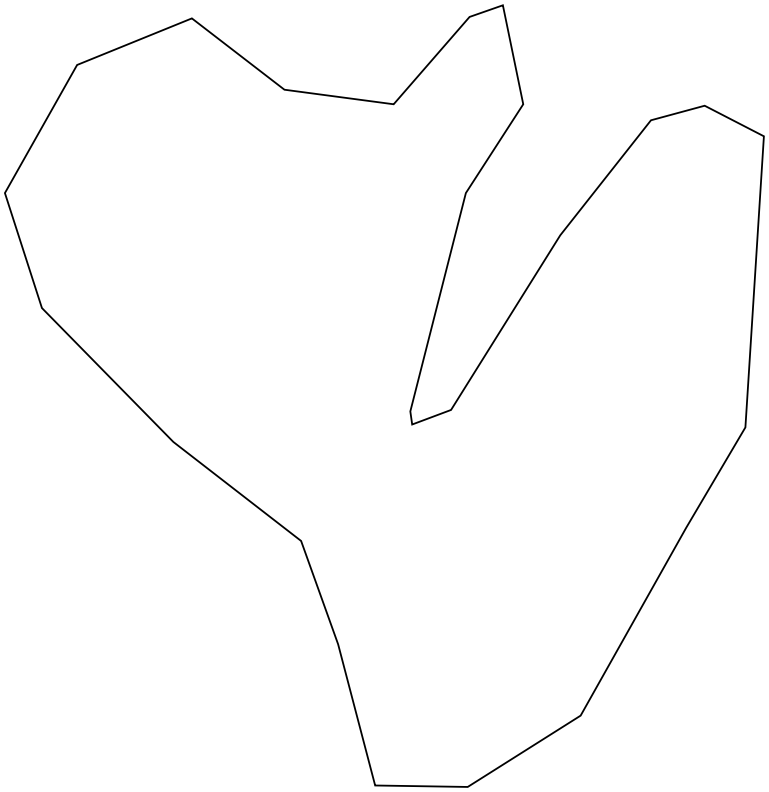

Figure 2.15: Object with complex shape and concavity.

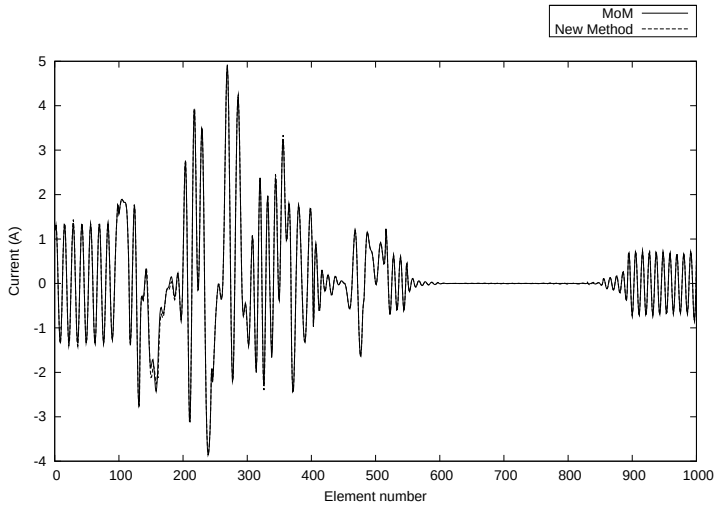

Figure 2.16: Real currents for object with concavity with 100λ total circumference and TM incident wave with $H_o = 1$ at 45^0. (a) Real part and (b) Imaginary part.

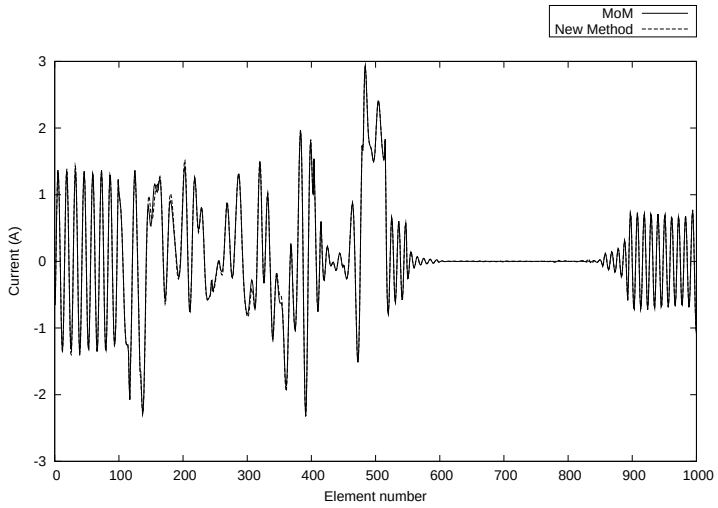

Figure 2.17: Imaginary currents for object with concavity with 100λ total circumference and TM incident wave with $H_o = 1$ at 45^0. (a) Real part and (b) Imaginary part.

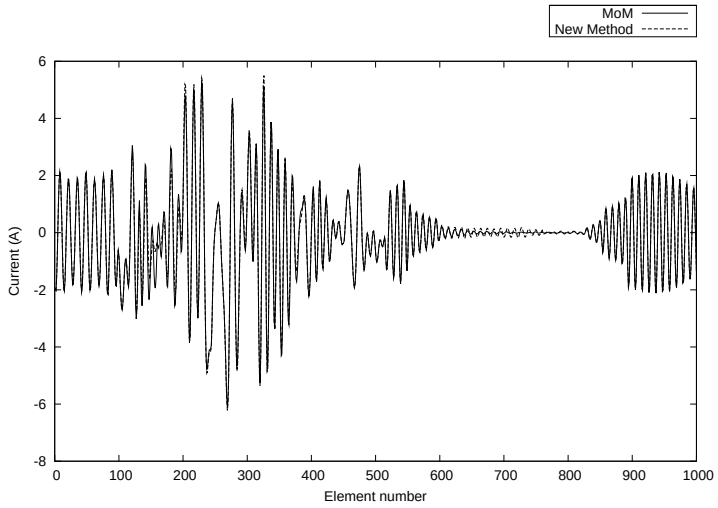

Figure 2.18: Real currents for object with concavity with 100λ total circumference and TE incident wave with $H_o = 1$ at 45^0. (a) Real part and (b) Imaginary part.

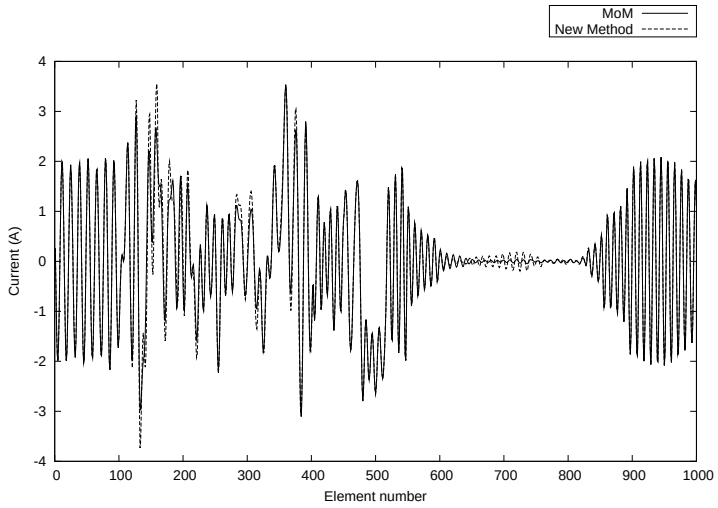

Figure 2.19: Imaginary currents for object with concavity with 100λ total circumference and TE incident wave with $H_o = 1$ at 45^0. (a) Real part and (b) Imaginary part.

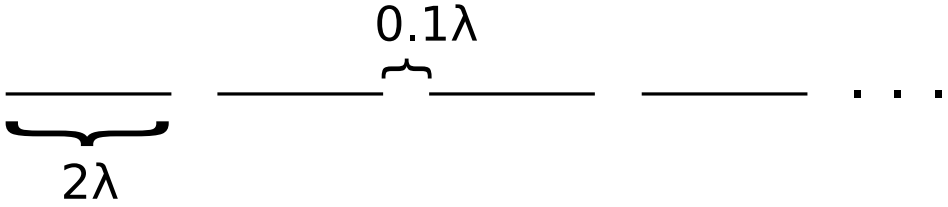

Figure 2.20: Geometry for two-dimensional strip array. Each array element is 2λ in length and they are separated by 0.1λ.

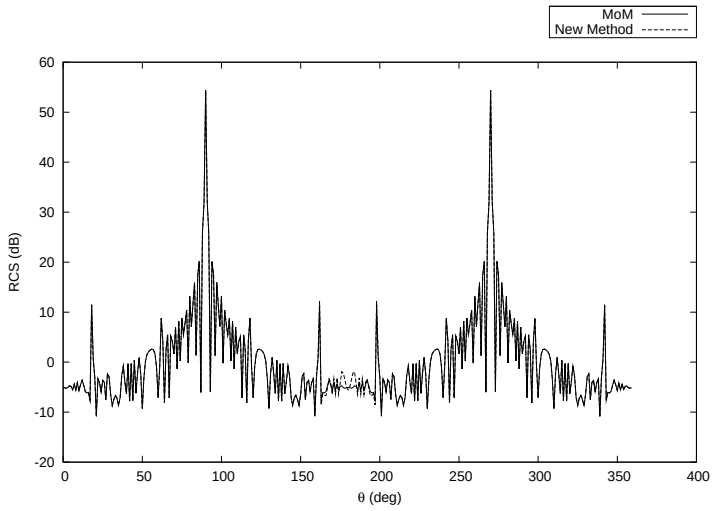

Figure 2.21: RCS for 2D strip array with 100 2λ elements. The incident wave is normal to the array with $H_o = 1$ and has TM polarization.

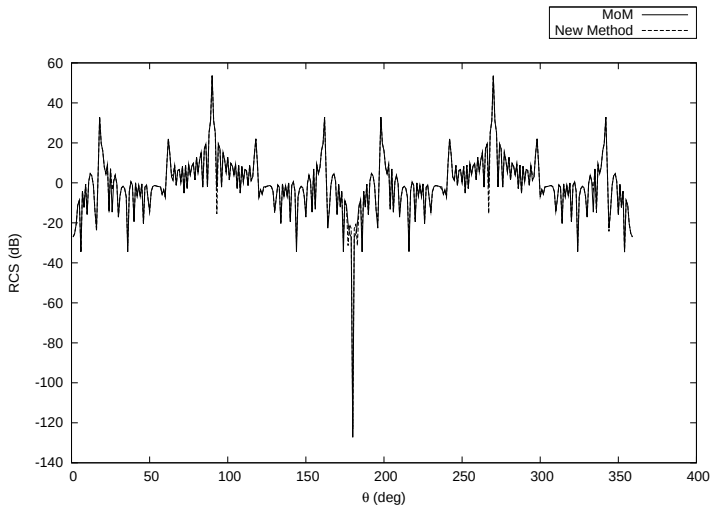

Figure 2.22: RCS for 2D strip array with 100 2λ elements. The incident wave is normal to the array with $H_o = 1$ and has TE polarization.

2.6 Conclusions

In this chapter, the method was applied to 2D arbitrary conducting cylinders. Here, open and closed bodies were considered with both transverse electric and transverse magnetic incident waves. Also, we considered scattering cases from canonical shapes as well as complex shapes. Accurate and efficient results were obtained in each case. Furthermore, it was shown that for 2D cases, the method virtually eliminates the solution time and therefore the overall solution is limited by the matrix fill time. Also, the groups for two-dimensional cases may be formed arbitrarily or may be decided by the geometry of the object. Furthermore, the groups should ideally be a few wavelengths in length and should include only a few wavelengths in radius as a near-field designation.

Chapter 3

Arbitrarily shaped three-dimensional conductors

In this chapter, we apply the method to arbitrarily shaped three-dimensional conductors. First the integral equations are presented for both the electric field and magnetic field boundary conditions. Furthermore, each equation is expanded with set of sub-domain basis functions in preparation for applying the method. Also, the combined-field formulation is presented to avoid any internal resonance problems associated with closed bodies. Next, we discuss how the algorithm is applied in the general case. We determine how groups should be formed and to what groups the null-field procedure should be applied. The decoupling procedure is illustrated and we discuss how to handle near-field and far-field interactions on an arbitrary three-dimensional body. Finally, we apply the algorithm to some challenging scattering problems to illustrate the accuracy and effectiveness of the method.

3.1 Electric Field Integral Equation - EFIE

The Electric Field Integral Equation (EFIE) for Perfect Electrical Conductors (PECs) is developed as follows. First, by definition of a PEC, the total tangential electric field at the surface of the conductor is identically zero:

$$\mathbf{E}^i_{tan}(\mathbf{r}) + \mathbf{E}^s_{tan}(\mathbf{r}) = 0,\ \mathbf{r} \in S_c \tag{3.1}$$

or

$$\mathbf{E}^i_{tan}(\mathbf{r}) = -\mathbf{E}^s_{tan}(\mathbf{r}),\ \mathbf{r} \in S_c \tag{3.2}$$

where $\mathbf{E}^i_{tan}$ and $\mathbf{E}^s_{tan}$ are the incident and scattered fields, respectively and S_c is the conductor surface.

The scattered field at a point $\mathbf{r}$ is related to the currents on the conductor by the following equation:

$$\mathbf{E}^s(\mathbf{r}) = -j\omega\mathbf{A}(\mathbf{r}) - \nabla\Phi(\mathbf{r}) \tag{3.3}$$

where $\mathbf{A}$ and Φ are the vector and magnetic potentials given by:

$$\mathbf{A}(\mathbf{r}) = \mu \int_{S_c} \mathbf{J}(\mathbf{r}')G(\mathbf{r},\mathbf{r}')dS' \tag{3.4}$$

$$\Phi(\mathbf{r}) = \frac{-1}{j\omega\epsilon} \int_{S_c} \nabla \cdot \mathbf{J}(\mathbf{r}')G(\mathbf{r},\mathbf{r}')dS' \tag{3.5}$$

with the Green's function given by

$$G(\mathbf{r},\mathbf{r}') = \frac{e^{-jkR}}{4\pi R}, R = |\mathbf{r}-\mathbf{r}'| \tag{3.6}$$

Substituting for the scattered field we get the desired integral equation:

$$\mathbf{E}^i(\mathbf{r})_{tan} = [j\omega\mathbf{A}(\mathbf{r}) + \nabla\Phi(\mathbf{r})]_{tan} \tag{3.7}$$

Next, we discretize the geometry of the object of interest using a triangular surface mesh. Then we define a set of basis functions over the surface to approximate the current on the PEC. Let $\mathbf{f}_n$ for $n = 1, ..., N$ be a basis defined on the surface of the conductor. The total current on the conductor is then given by:

$$\mathbf{J}(\mathbf{r}) = \sum_{n=1}^{N} \alpha_n \mathbf{f}_n(\mathbf{r}) \tag{3.8}$$

Here we use the RWG basis functions as given in [2]. Each function is defined over a pair of triangles as:

$$\mathbf{f}_n(\mathbf{r}) = \begin{cases} \frac{l_n}{2A_n^+}\rho_n^+, & \mathbf{r} \in T_n^+ \\ \frac{l_n}{2A_n^-}\rho_n^-, & \mathbf{r} \in T_n^- \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

where A_n^+ and A_n^- are the areas of the positive and negative side triangles, T_n^+ and T_n^-, respectively.

The quantities in Eq. 3.9 are shown in Figure 3.1.

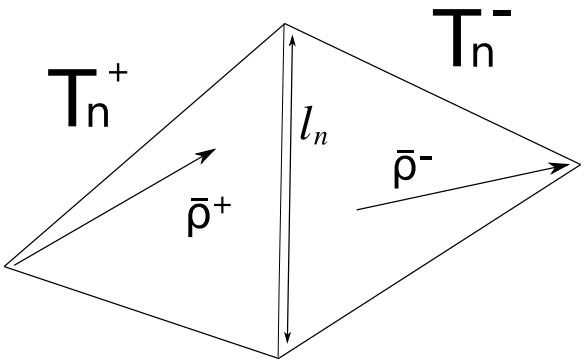

Figure 3.1: Quantities for RWG basis functions.

Substituting Eq. 3.9 into Eqs. 3.4 and 3.5 we get

$$\mathbf{A}(\mathbf{r}) = \sum_{n=1}^{N} \alpha_n \mathbf{A}_n(\mathbf{r}) \tag{3.10}$$

and

$$\Phi(\mathbf{r}) = \sum_{n=1}^{N} \alpha_n \Phi_n(\mathbf{r}) \tag{3.11}$$

where

$$\mathbf{A}_n(\mathbf{r}) = \mu \int_{S_c} \mathbf{f}_n(\mathbf{r}')G(\mathbf{r}, \mathbf{r}')dS' \tag{3.12}$$

$$\Phi_n(\mathbf{r}) = \frac{-1}{j\omega\epsilon}\int_{S_c} \nabla \cdot \mathbf{f}_n(\mathbf{r}')G(\mathbf{r},\mathbf{r}')dS' \tag{3.13}$$

Since we have N unknowns, we must generate N linearly independent equations. We do this by performing a testing procedure on each side of Eq. 3.7,using the inner product

$$\langle \mathbf{f}, \mathbf{g} \rangle \equiv \int_{S_c} \mathbf{f} \cdot \mathbf{g} dS \tag{3.14}$$

and the Galerkin procedure, we test each side of Eq. 3.7 using the previously defined RWG functions. We thereby generate a linear system

$$ZI = V \tag{3.15}$$

which can be solved to find the coefficients α_n. The testing procedure provides us with the following quantities:

$$Z_{m,n} = j\omega \langle \mathbf{A}_n, \mathbf{f}_m \rangle + \langle \nabla\Phi_n, \mathbf{f}_m \rangle \tag{3.16}$$

as well as

$$V_m = \left\langle \mathbf{E}^i, \mathbf{f}_m \right\rangle \tag{3.17}$$

The potential terms can be evaluated at the centroids of the testing triangles:

$$\langle \mathbf{A}_n, \mathbf{f}_m \rangle = \frac{l_m}{2}\left[\mathbf{A}_n(\mathbf{r}_m^{ct+}) \cdot \rho_m^{ct+} + \mathbf{A}_n(\mathbf{r}_m^{ct-}) \cdot \rho_m^{ct-}\right] \tag{3.18}$$

$$\left\langle \nabla\Phi_{c/p/q,n}, \mathbf{f}_m \right\rangle = -l_m\left[\Phi_{c/p/q,n}(\mathbf{r}_m^{c+}) - \Phi_{c/p/q,n}(\mathbf{r}_m^{c-})\right] \tag{3.19}$$

and the excitation vector is given by:

$$\left\langle \mathbf{E}^i, \mathbf{f}_m \right\rangle = \frac{l_m}{2}\left[\mathbf{E}^i(\mathbf{r}_m^{ct+}) \cdot \rho_m^{ct+} + \mathbf{E}^i(\mathbf{r}_m^{ct-}) \cdot \rho_m^{ct-}\right] \tag{3.20}$$

Finally, we solve for the currents $I_m = \alpha_m$.

3.2 Magnetic Field Integral Equation - MFIE

The Magnetic Field Integral Equation (MFIE) for PECs is developed as follows. This method is only applicable to closed bodies since we must assume the fields inside are zero. First, we apply the boundary conditions on the magnetic field at the surface of the conductor. The magnetic field is related to the current on the structure with

$$\mathbf{J}(\mathbf{r}) = \hat{n} \times (\mathbf{H}^i(\mathbf{r}) + \mathbf{H}^s(\mathbf{r})),\ \mathbf{r} \in S_c \tag{3.21}$$

where $\hat{n}$ is a unit vector normal to the surface of the conductor and pointing outward, $\mathbf{H}^i$ is the incident wave, and $\mathbf{H}^s$ is the scattered field due to the currents on the conductor. It is shown in [5] that for $\mathbf{r}$ not on an edge, we may write the scattered portion as

$$\begin{aligned} \hat{n} \times \mathbf{H}^s(\mathbf{r}) &= \lim_{\mathbf{r} \to S_c} \hat{n} \times \nabla \times \mathbf{A}(\mathbf{r}) \\ &= \frac{\mathbf{J}}{2}(\mathbf{r}) + \hat{n} \times \oint_{S_c} \mathbf{J}(\mathbf{r}') \times \nabla' G(\mathbf{r}, \mathbf{r}') dS',\ \mathbf{r} \in S_c \end{aligned} \tag{3.22}$$

where $G(\mathbf{r}, \mathbf{r}') = \frac{e^{-jkR}}{4\pi R}$, $R = |\mathbf{r} - \mathbf{r}'|$, and $\mathbf{r}$ approaches the surface from outside the object. The deleted integral is evaluated everywhere except $\mathbf{r} = \mathbf{r}'$. Substituting

Eq. 3.22 into Eq. 3.21 we arrive at the integral equation

$$\hat{n} \times \mathbf{H}^i(\mathbf{r}) = \frac{\mathbf{J}}{2}(\mathbf{r}) - \hat{n} \times \oint_{S_c} \mathbf{J}(\mathbf{r}') \times \nabla' G(\mathbf{r}, \mathbf{r}') dS', \ \mathbf{r} \in S_c \tag{3.23}$$

As with the EFIE in Eq. 3.8, we may expand the current using RWG functions. Also, applying the Galerkin procedure using the inner product in Eq. 3.14, we arrive at a linear system $ZI = V$ with the following elements:

$$\begin{aligned} Z_{m,n} &= \frac{1}{2} < \mathbf{f}_m, \mathbf{J} > -l_m \left[\frac{1}{2A_m^+} \int_{S_c} \rho^+ \cdot \hat{n}_m^+ \times \oint_{S_c} \mathbf{f}_n \times (\nabla' G)_m^+ dS' dS \right. \\ &\quad \left. + \frac{1}{2A_m^-} \int_{S_c} \rho^- \cdot \hat{n}_m^- \times \oint_{S_c} \mathbf{f}_n \times (\nabla' G)_m^- dS' dS \right] \end{aligned} \tag{3.24}$$

and

$$V_m = l_m \left[\frac{1}{2A_m^+} \int_{S_c} \rho^+ \cdot \hat{n}_m^+ \times \mathbf{H}^i(\mathbf{r}_m^+) dS \right. \tag{3.25}$$

$$\left. + \frac{1}{2A_m^-} \int_{S_c} \rho^- \cdot \hat{n}_m^- \times \mathbf{H}^i(\mathbf{r}_m^-) dS \right] \tag{3.26}$$

The integrals may be evaluated as in [3]. Again, the deleted integral is evaluated everywhere except $\mathbf{r} = \mathbf{r}'$. Solving this system provides us with the coefficients for the basis defined on the structure.

3.3 Combined Field Integral Equation - CFIE

To avoid any problems due to internal resonances for closed bodies, we may use the Combined Field Integral Equation [4] given by

$$-\frac{\gamma}{\eta} \hat{n} \times \hat{n} \times \mathbf{E} + (1 - \gamma) \hat{n} \times \mathbf{H} = 0 \tag{3.27}$$

which can be implemented as

$$Z_{CFIE} = \gamma Z_{EFIE} + \eta(1-\gamma) Z_{MFIE} \tag{3.28}$$

$$V_{CFIE} = \gamma V_{EFIE} + \eta(1-\gamma) V_{MFIE} \tag{3.29}$$

where $0 \leq \gamma \leq 1$ is a constant depending on the problem and $\eta = \sqrt{\frac{\mu}{\epsilon}}$ is the impedance of the medium. For open body problems, we let $\gamma = 1$ and for closed bodies γ is typically 0.5.

3.4 Element Grouping

In this section, we discuss the grouping procedure for the arbitrary three dimensional conducting case. In order to account for the near-field interactions, we use the null-producing procedure on those groups that are within a given perimeter. As shown in Figure 3.2, we take each group as a source. All those groups within a given near-field radius will be selected for the decoupling procedure. Each basis function in a given group will be replaced with a linear combination of that basis function, with a coefficient of 1, and all the basis functions in the null groups, which are assigned unknown coefficients. By requiring that the new basis functions produce null fields everywhere on the selected groups, we generate zeros within the new system matrix. Once this has been carried out for all the groups, we may solve for the currents on the structure by first inverting the self-block-matrices to account for near-field interactions, and then solve for the remaining current contributions via the Gauss-Seidel iterative procedure. Note also that the coefficients for all the groups can be solved for independently from one another. In other words, parallel processing may be used to solve for weights throughout the structure simultaneously. Furthermore, since each set of group coefficients may be solved for separately, a parallel procedure shows a high

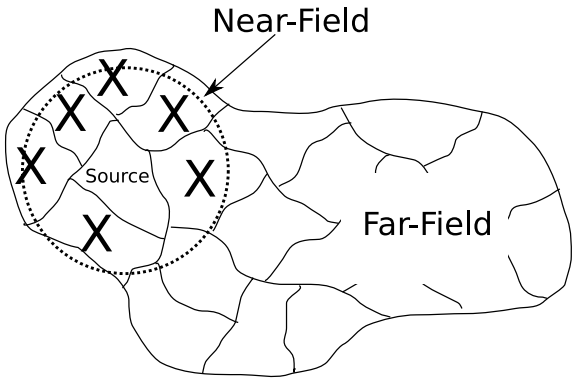

Figure 3.2: Grouping for arbitrary 3d case. X's represent null groups.

degree of parallel efficiency irregardless of how many processors are used, provided that the number of processors does not exceed the number of groups.

3.5 Numerical Results

In this section, we present some numerical results to verify the accuracy of the algorithm. First, using the CFIE formulation, we show a bistatic RCS calculation for a 5λ radius sphere, where λ is the wavelength for the given frequency, as seen in Figure 3.3. In this case, we have 92,550 unknowns. The structure is divided into 314 groups, each roughly $1\lambda^2$ in area. For each source group, all groups within a 2λ distance were included as near-field groups. Figure 3.4 shows the bistatic RCS for 1 and 2 iterations as well as the RCS when computed with a MoM BOR (Body of Revolution) code. The incident field $\mathbf{E} = \eta\hat{a}_\theta$ is incident at an angle of $(\theta, \phi) = (45^o, 0^o)$ with a frequency of 300 MHz. The RCS is calculated for the x-z plane, and is shown in Figure 3.4. Since there is very little difference between the plots, a single iteration suffices in this case. Also, note that, for this code, the initial guess given in Sec. 1.4 is implemented as the first iteration of the solver. Thus, the first iteration only includes near-field interactions. A good way to determine the quality of our solution is to calculate the

following:

$$r = V - ZI \tag{3.30}$$

where V is the excitation vector, Z is the standard moment matrix (sub-domain functions), and I is the current vector for the sub-domain functions. The total error is then given by:

$$Total\ error = \sum_{i}^{N} |r_i| \tag{3.31}$$

where N is the number of unknowns for the system. The average error per term is then given by:

$$Avg.\ error\ per\ term = \frac{Total\ error}{N} \tag{3.32}$$

For the case of the sphere, the average error per term is .096 for 1 iteration, while it is only .0142 for two iterations. So, the spherical case compares very well with a standard Method of Moments solution. The RCS was also compared to a MoM BOR (Body of Revolution) code and shows excellent agreement. Also, when computed on 8 3.0 GHz AMD Opteron CPUs, the solution took approximately 25 wall clock hours and less than 4.5 GB of storage. This includes the time taken for a serialized RCS calculation on a single CPU (around 2 hours). Storing the full matrix in memory would require approximately 64 GB of storage. Furthermore, an LU decomposition would require approximately 19 days and 11 hours to solve on 8 CPUs assuming a perfectly parallel solver. Also, the weight generation step shows proper scaling when using multiple CPUs. For a single CPU, the weight generation procedure takes 49 hours. For 4 CPUs, this step took 9 hours and 14 mins. It required only 4 hours, 28 mins for 8 CPUs.

For the next example, we compute the bistatic RCS of a $12m \times 12m$ square plate with 42883 unknowns as seen in Figure 3.5. Here, we have an open structure and

therefore utilize only the EFIE. The plate is excited with an incident wave $\mathbf{E} = \eta\hat{a}_\theta$ at an angle of $(\theta, \phi) = (45^o, 0^o)$ with a frequency of 300 MHz. The RCS is computed for the x-z plane. The groups were formed by dividing the plate into a square grid of 144 groups, each roughly $1\lambda^2$ in size and having on average 298 unknowns. Using a 2λ near-field criteria, the solver went through two iterations and the RCS for each is shown in Figure 3.6. The average error for the first iteration is 0.612 while the average error for the second is 0.232. Also, when computed on 8 3.0 GHz AMD Opteron CPUs, the solution took approximately 2 hours 45 minutes wall clock time and less than 2gb GB of storage. Storing the full matrix in memory would require 13.7 GB of memory. Furthermore, computing a LU decomposition on 8 CPUs would take approximately 1 day and 22 hours assuming the solver is fully parallel.

As a final example, we also calculate the bistatic RCS of a French Mirage fighter jet. The geometry may be seen in Figure 3.7. For this case, we have an open structure and therefore use the EFIE only. From end to end, the jet is approximately 33λ long. Across the wings, it is approximately 23λ in width. Also, from the bottom to the highest point on the plane (vertical fin), it is about 10λ high. The total surface area is $550\lambda^2$. There are 159293 unknowns and so a full LU decomposition in this case would be computationally expensive both in terms of memory and CPU time. In fact, the required storage for a full LU decomposition would be 189 GB. Furthermore, the LU decomposition itself would take approximately 99 days 5 hours assuming the solver was fully parallel. For this case, we formed the groups by placing the aircraft inside a cubical grid with each cell having dimensions of $1\lambda \times 1\lambda \times 1\lambda$. All functions within a given cell were grouped together. There are 702 groups with an average group size is 226 functions and all groups within 2λ of a given source group were decoupled with null field points. Using an incident wave $\mathbf{E} = \eta\hat{a}_\theta$ at an incident angle of $(\theta, \phi) = (45^o, 0^o)$ with a frequency of 750 MHz, we calculate the bistatic RCS for

the x-z plane as shown in Figure 3.8. After running 2 iterations, the average error per term for the first iteration is 0.224 while the error for the second iteration is 0.218. The error demonstrates that our solution agrees very well with the standard MoM solution and more iterations may be performed for further accuracy if necessary. Also, note that the grouping scheme was generated automatically and thus it is possible to automate the entire process for an arbitrary three-dimensional conductor.

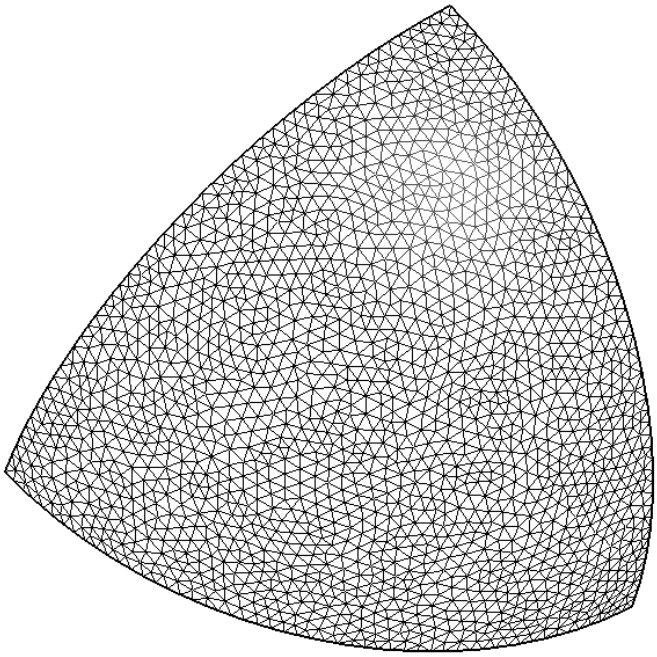

Figure 3.3: Triangular mesh for one octant of 5λ radius sphere.

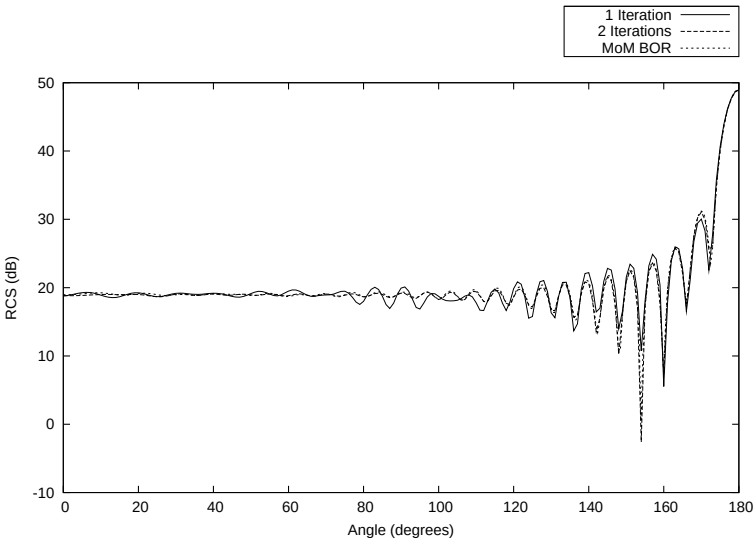

Figure 3.4: Bistatic RCS for 5λ radius sphere. First and second iterations and MoM BOR code.

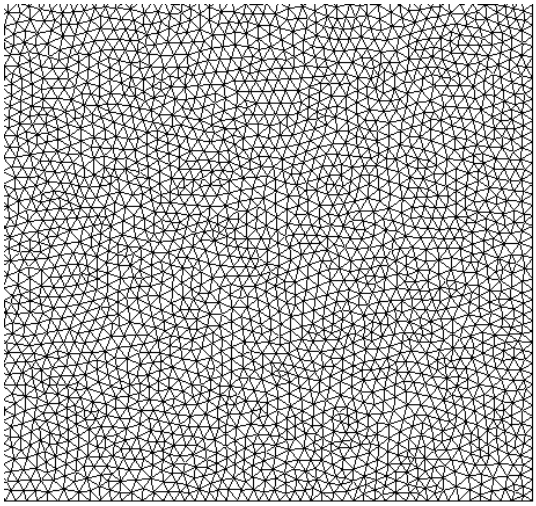

Figure 3.5: Triangular mesh for section of $12\lambda \times 12\lambda$ square plate.

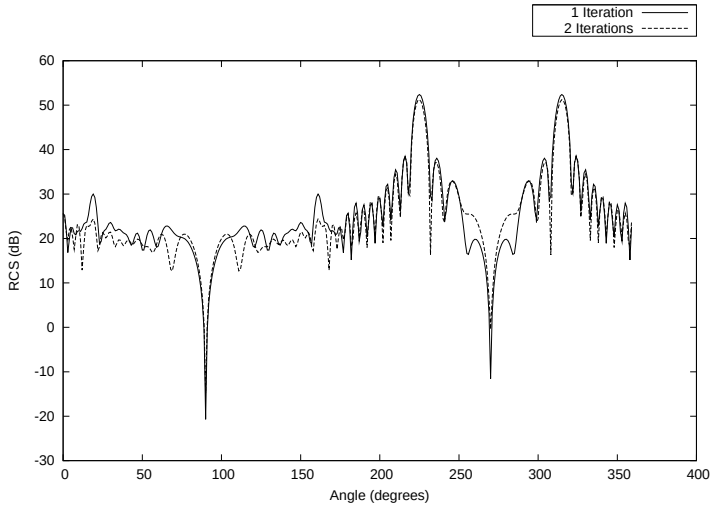

Figure 3.6: Bistatic RCS for $12\lambda \times 12\lambda$ square plate. The first and second iterations are shown.

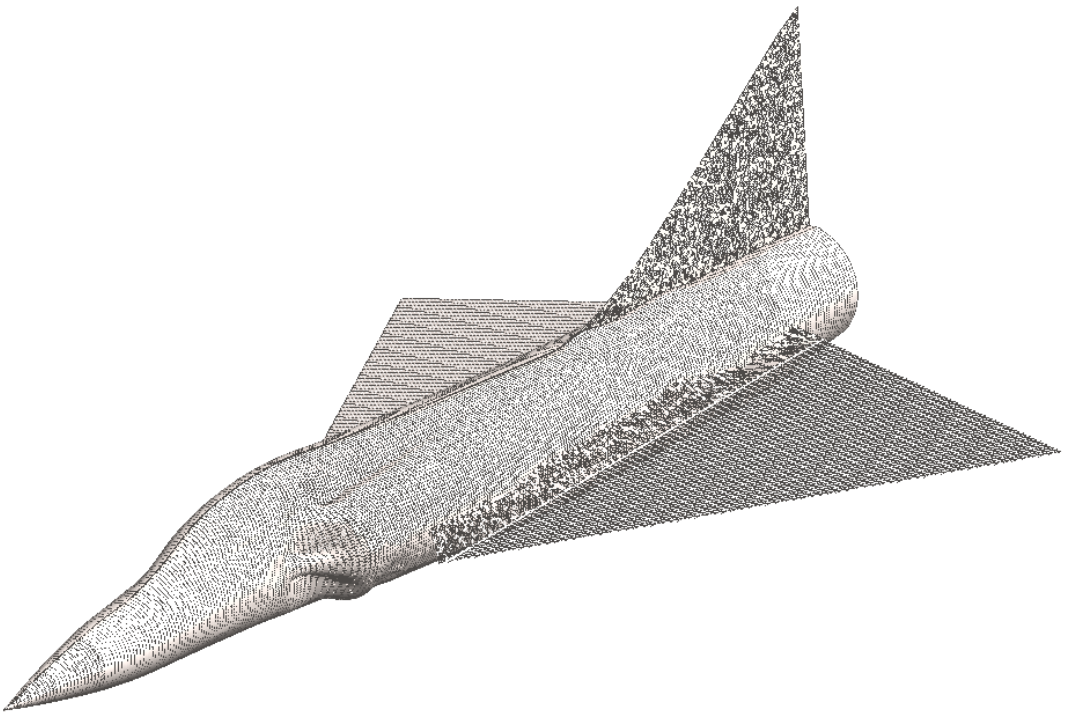

Figure 3.7: Triangular mesh for French Mirage aircraft geometry.

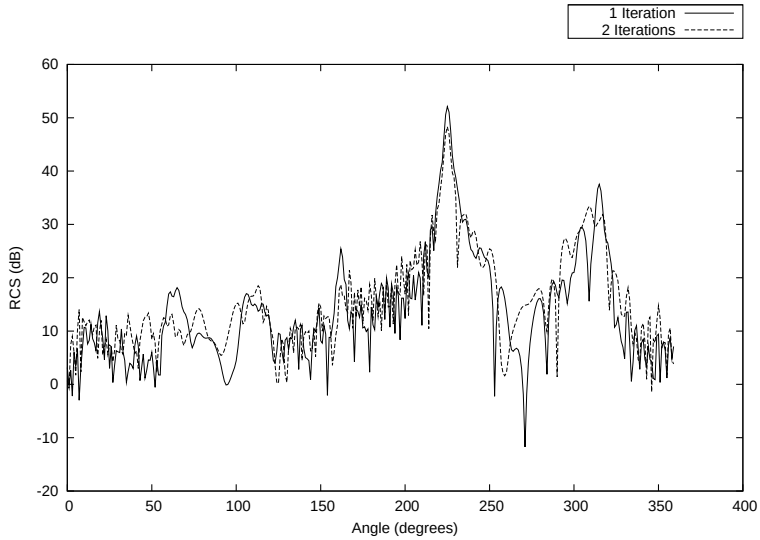

Figure 3.8: Bistatic RCS for aircraft, first and second iterations

3.6 Conclusions

In this chapter, the method was applied to arbitrary three-dimensional conductors. First, we developed the integral equations for the electric field, magnetic field, and combined field formulations. The electric field formulation may be used for open bodies while the combined field may be used for closed bodies in order to avoid problems with internal resonances. Next, we discussed how basis functions should be grouped together using a domain decomposition technique. Furthermore, a near-field region was defined where all the functions within a certain radius of a source group were eliminated by developing a new set of basis functions. These functions were defined in such a way that the net effect of the parts of each function was to produce null fields everywhere on the structure within the near-field radius. This procedure decouples all strongly interacting groups and allows us to produce a strongly block-diagonally-dominant matrix that will converge very quickly within an iterative scheme such as the block Gauss Seidel method. Furthermore, some challenging example problems were simulated to show the effectiveness of the method. The results agree very well with the standard Method of Moments solution using sub-domain basis functions. Furthermore, we can conclude that a good rule of thumb for the size of each group is roughly $1\lambda^2$. Also, the near-field criteria should generally be set at approximately a 2λ radius. A larger radius leads to a more quickly converging iterative solution, but could result in difficulty in solving for the group coefficients. Finally, the numerical results demonstrated that the technique can be made highly parallel and the solution time decreases in direct proportion to the number of processors used.

Chapter 4

Three-dimensional conductor periodic arrays

In this chapter, we use our method to solve a special case of arbitrary 3D conductors. Here, we solve for the case of finite conducting periodic arrays. The motivation behind this special case is to demonstrate that the geometry behind certain problems may be utilized to further enhance the solution efficiency. Although each of the array elements may be any arbitrary three-dimensional surface, the arrays themselves are periodic only in two dimensions as shown in Figure 4.1. However, the concepts applied here may easily be extended to cases where there is periodicity in three dimensions.

4.1 Integral Equation Formulation

For the finite periodic array, we use the same integral formulation as was used in the arbitrary 3D case. In the case where each array element is an open body, we use the EFIE only. If each element is a closed body, we use the combined field formulation as seen in Section 3.3 with $\alpha = 0.5$. This allows us to avoid any issues arising due to internal resonances within each element.

4.2 Element Grouping

Although nothing prevents each array element from being subdivided into groups, we are assuming here that each element forms one and only one group. Null fields are produced on those groups that are horizontally and vertically adjacent to a given source group. This can be seen in Figure 4.2.

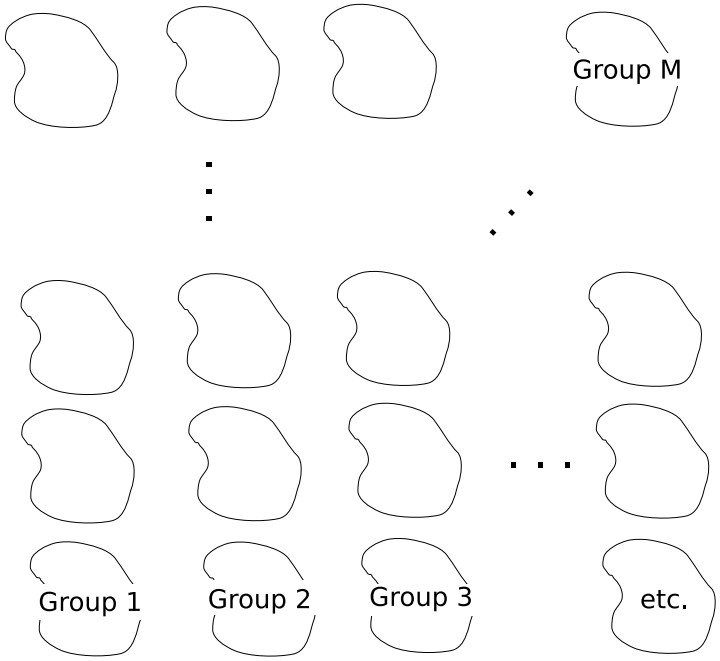

Figure 4.1: Finite periodic conducting array periodic in two dimensions.

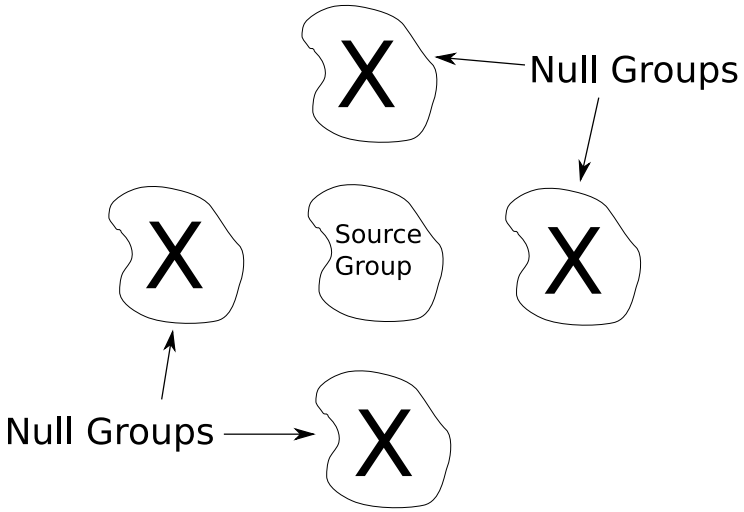

Figure 4.2: Elimination of groups within near-field for finite periodic array.

For groups on the edges and corners of the array, there will only be three and two adjacent groups, respectively. The finite periodic array is a special case in the general method because the weights used to produce the null fields are largely redundant. In fact, for two-dimensional arrays, one must solve for at most 9 sets of weights. These are each of the four corners, one element along each of the four sides, and one element internal to the structure. Once these weights have been computed, they can easily be copied to the other groups in the system.

It greatly simplifies the programming and is computationally inexpensive to compute all the corners separately. Note that while the physical structure may show symmetry for edge groups and corner groups, this symmetry may not exist numerically due to the meshing of each element and the basis functions defined there. So, in general, one cannot solve for the coefficients at one corner of the structure and then directly copy those weights to the other three corners. The same reasoning applies to elements along the edges of the array.

4.3 Numerical Results

Here we present a bistatic RCS calculation for a finite periodic array. Each element is a 0.5λ x 0.5λ square plate, each expanded with 64 RWG basis functions. There are 2500 plates arranged on a 50 x 50 grid for a total of 160000 unknowns. Each plate is spaced 0.75λ from each of the adjacent plates. The array is excited with a 300MHz incident wave with $E_\theta = 120\pi$ and $E_\phi = 0$ at an angle of $\theta = 45^o$ and $\phi = 0^o$. Finally, the bistatic RCS is calculated for the x-z plane. Figure 4.3 displays the geometry for the problem. Figure 4.4 shows a comparison between the first and second iterations of the Gauss-Seidel scheme. Note that there is a negligible difference between the RCS values. When compared to the standard Method of

Moments solution, the average error per current term (either real or imaginary part) for 1 iteration was 0.288 while it was 0.122 for 2 iterations. Here the error per term is calculated by first obtaining the error vector:

$$r = V - ZI \tag{4.1}$$

where V is the excitation vector, Z is the standard moment matrix (sub-domain functions), and I is the current vector for the sub-domain functions. The total error is then given by:

$$Total\ error = \sum_{i}^{N} |r_i| \tag{4.2}$$

where N is the number of unknowns for the system. The average error per term is then given by:

$$Avg.\ error\ per\ term = \frac{Total\ error}{N} \tag{4.3}$$

When computed on 8 3.0 GHz AMD Opteron CPUs, the solution took approximately 19.5 wall clock hours and less than 500 MB of storage. This includes the time taken for a serialized RCS calculation on a single CPU (around 3 hours). A full LU decomposition of this problem would require 190 GB of storage and would have a solution time of approximately 100 days and 13 hours assuming the solver is fully parallel. Also, note that, for this code, the initial guess given in Sec. 1.4 is implemented as the first iteration of the solver. Thus, for this case, the simulation shows that the currents are almost entirely determined by the near-field interactions on the structure.

4.4 Conclusions

In this section, a special case of the 3D conductor formulation was considered. Here, a finite array of elements periodic in 2 dimensions was constructed and the bistatic RCS was calculated. There are two essential reasons for simulating this special case. First, the problem lends itself very well to the iterative algorithm. Since there is spacing between the elements, there is already significant decoupling between different parts of the structure. So, this type of structure is well-suited for the algorithm. Furthermore, due to the geometry, the decoupling weights of many groups are redundant. This significantly reduces the amount of necessary computation. After calculating at most 9 different sets of weights, the remaining groups may be decoupled by simply copying the weights throughout the structure. Finally, a typical problem of a square plate periodic array was simulated and the results were shown to compare very well with the standard Method of Moments solution. Furthermore, only 1 or 2 iterations were necessary and therefore the solution time was shown to be much more efficient than when using a standard LU decomposition or conjugate gradient solution.

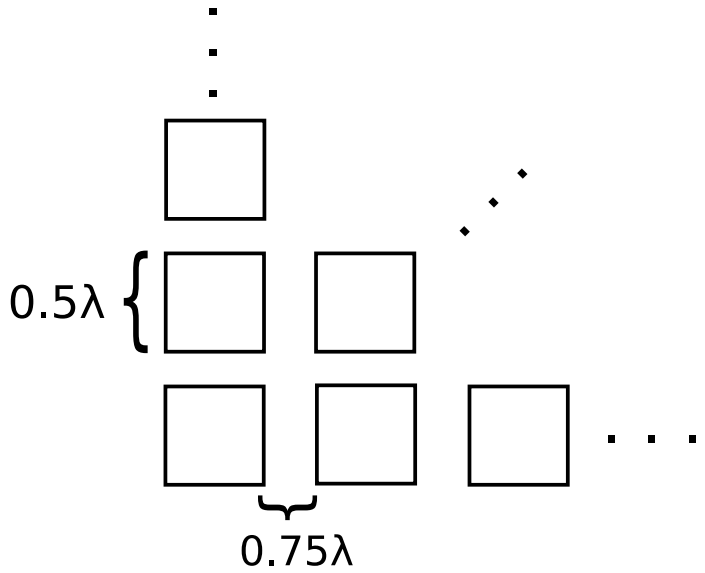

Figure 4.3: Geometry for 50 x 50 $0.5\lambda^2$ plate finite periodic array.

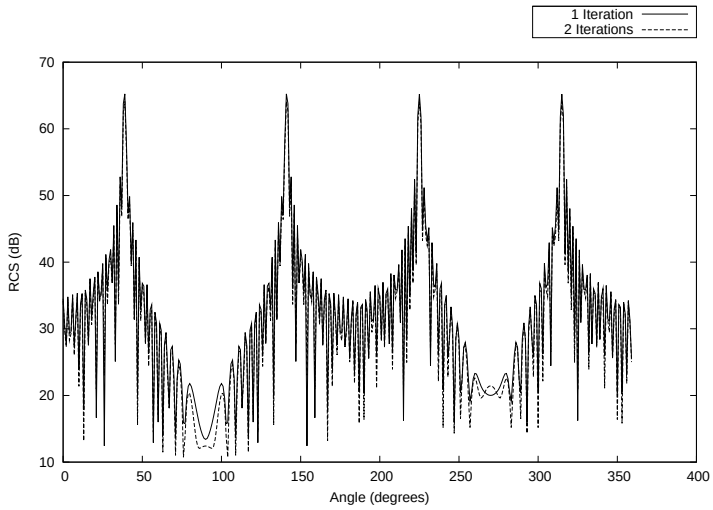

Figure 4.4: Bistatic RCS result after 1 and 2 iterations.

Chapter 5

Conclusions

In this work, we applied a domain decomposition technique where any arbitrary conducting structure may be divided into an arbitrary number of groups. Next, a basis transformation allowed us to decouple groups adjacent to one another on the structure. This was accomplished using the criteria that each group should radiate null fields at locations on the structure chosen by a nearest neighbor criteria. A new block-diagonally-dominant system matrix was constructed from the matrix corresponding to a set of sub-domain basis functions. Using this new matrix, we were able to apply a linear iterative solver capable of converging within a few iterations. Since a full LU decomposition solver was not used and iterative solvers require very little memory, we were able to avoid both CPU time and memory size constraints usually found in the standard Method of Moments. Finally, this method has been outlined in a very general format and may therefore be applied to any existing Method of Moments code formulated with sub-domain basis functions. Various geometries in two and three dimensions were simulated to illustrate the successful application of the method to arbitrary conducting structures.

5.1 Scaling and Parallel Efficiency

In this section, we discuss the scaling as well as the parallel efficiency of each part of the algorithm. Referring to Sec. 1.4, we see that there are five major steps to the algorithm. The scaling and efficiency of each step is as follows:

1. Group Formation - This is part of the preprocessing stage. All groups are formed prior to running the code and so this is not considered in the scaling behavior of the algorithm.

2. Create new basis functions - Although scaling here is dependent upon the geometry and nearest neighbor criteria, we may still construct a rule of thumb. If we assume that all the groups are roughly the same size for all geometries, the scaling will be $O(N)$. Furthermore, the coefficients/weights for each source group may be solved for independently. Therefore, the groups may be evenly distributed across multiple processors and the efficiency will be near 100%.

3. Create new system matrix $\tilde{Z}$ - This step is represented by the operation $\tilde{Z} = ZR$. For dense matrices, matrix multiplication is typically an $O(N^3)$ operation. However, in this case, R is highly sparse, so forming the new system matrix is an $O(N^2)$ process. For parallel operation, we can view the matrix multiplication as forming columns of $\tilde{Z}$. Since the columns may be evenly distributed to each of the processors, this is step also will have a parallel efficiency of approximately 100%.

4. Construct initial guess - This step consists of inverting the diagonal blocks of $\tilde{Z}$. Assuming group sizes are roughly the same, the number of blocks requiring inversion grows linearly with the number of unknowns. Therefore, the scaling for this step is $O(N)$. Furthermore, the blocks may be inverted in any order and may be evenly distributed across multiple processors. Therefore, the parallel efficiency will be near 100%. Note, however, that because the groups are generally small with respect to the overall structure, this step, in terms of absolute time, will most likely be negligible in execution time.

5. Iterate for accuracy - Here, each iteration is roughly equivalent to a matrix-vector product in terms of scaling. Furthermore, very few iterations are required. Therefore, this scales as $O(N^2)$. Furthermore, when viewing this as equivalent to a matrix-vector product, we can consider it as a collection of dot products - one for each row of the matrix dotted with the vector. These dot products may be evenly distributed over multiple processors and therefore the parallel efficiency will be near 100%.

The graphs in Figures 5.1 and 5.2 demonstrate the parallel efficiency of the arbitrary three-dimensional code. The test problem was a linear array of cubes, each one has dimensions of $0.2m \times 0.2m \times 0.2m$ and they are each spaced $0.8m$ apart. Each cube has 72 unknowns and there are 40 cubes for a total of 2880 unknowns. Nulls were placed on adjacent boxes so that the two cubes on the ends of the array generated 72 null points and those within the array generated 144. Using different numbers of CPUs, the entire program was timed to display the parallel efficiency of the overall algorithm. This involves generating the new basis coefficients, creating the new system matrix, a single matrix fill, and one iteration of the block Gauss-Seidel solver. Two test machines were used. The first has 6 CPUs, each of which is a 64-bit 1.4 GHz Intel Itanium 2 processor. Here, the times were taken for 1, 2, 4, and 6 CPUs. For each case, we plot the ratio of the time for a single processor to that of the time for the given case. The results are shown in Figure 5.1. The second machine has 8 2.3 GHz AMD Opterons. For this machine, 1, 2, 4, 6, and 8 processors were used. The results for this test are given in Figure 5.2. The ideal ratio for each case is also given. This test demonstrates that the overall algorithm is almost ideally parallel. This is difficult to achieve in standard methods due to typical solvers such as LU decomposition.

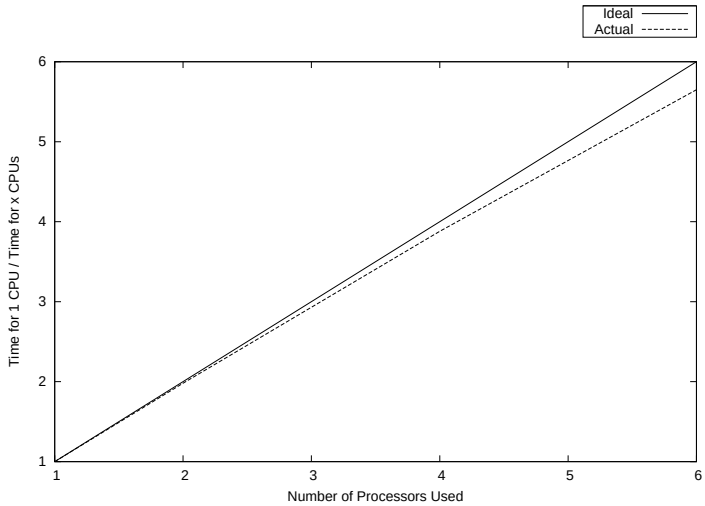

Figure 5.1: Ratio of single CPU time to 1, 2, 4, and 6 processors for machine one.

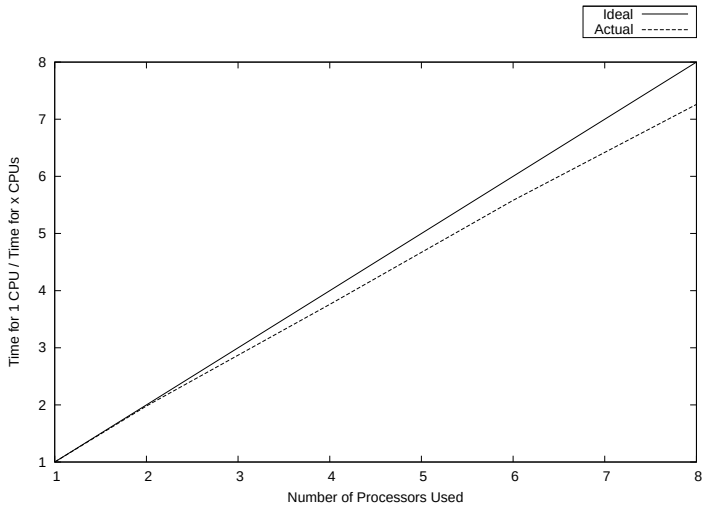

Figure 5.2: Ratio of single CPU time to 1, 2, 4, 6, and 8 processors for machine two.

5.2 Element Grouping

Element grouping should begin with considering the geometry of the object to be simulated. If there are parts of the object which are likely to be highly coupled, either due to their proximity or other reasons, then the newly created basis functions for each part should produce null-fields on the other part. Furthermore, any type of symmetry or periodicity may be utilized in order to make the solution procedure more efficient. For example, the coefficients for each element in a periodic array are largely redundant and thus several matrix inversions may be avoided as the weights may simply be copied throughout the structure. For arbitrarily shaped objects, the process can be fully automated. The groups may be formed by simply creating a cubical grid enclosing the entire object and then merely grouping together all those basis functions that happen to fall into a common cell. A cell size of approximately $1\lambda \times 1\lambda \times 1\lambda$ appears to work well. Also, a good rule of thumb for the near-field criteria in this case is 2λ. Finally, using these criteria generally leads to a rapid convergence of only a few iterations with the iterative solver.

5.3 Further Research

Further research includes applying the method to various moment formulations. Typical moment codes include support for wire and dielectric structures. This method should apply to these codes very easily. Furthermore, it is often desirable to compute multiple right hand sides as well as various frequencies. Multiple right hand sides can probably be solved simultaneously in the iterative solver while still maintaining a reasonable efficiency. Multiple frequencies require resolving the entire system multiple times. More research needs to be done in this area, such as possibly reusing the weights across multiple frequencies, in order to solve high bandwidth problems. Also,

there is potential for optimizing the various parts of the algorithm. For example, finding redundant structures within the geometry allows for faster weight generation. Also, since the algorithm is generally limited by the matrix fill time, it would be beneficial to find ways of speeding up this step. For example, using less rigorous integrations when functions are separated by large distances. Furthermore, Graphics Processing Units (GPUs) may be applied here since the matrix fill may easily be made parallel [10] as well as the other parts of the algorithm. For shared memory, multiple CPU architectures, a parallel LU decomposition would lower the memory requirements. In the current implementation, each processor needs its own scratch space for computing the new basis coefficients. A parallel LU decomposition would require scratch space for only a single matrix. The group coefficients could then be found one group at a time while maintaining the speed of a parallel code. Also, the algorithm shows tremendous potential for distributed computing architectures. It would be extremely effective in very large problems if hundreds of processors could be utilized to solve a single problem. Finally, the concepts here may offer a more efficient solution in the time domain as well.

Bibliography

[1] R.F. Harrington, *Field Computation by Moment Methods Classic Reissue*. New York: IEEE Press, 1993.

[2] S.M. Rao, D.R. Wilton, and A.W. Glisson, "Electromagnetic scattering by surfaces or arbitrary shape," *IEEE Trans. Antennas Propag.*, Vol. 30, pg. 409-418, May 1982.

[3] S.M. Rao, "Electromagnetic Scattering and Radiation of Arbitrarily-Shaped Surfaces by Triangular Patch Modeling," Ph.D. Dissertation University of Mississippi, Aug. 1980.

[4] J.R. Mautz, R.F. Harrington, "H-field, E-field, and combined-field solutions for conducting bodies of revolution," Archiv fuer Elektronik und Uebertragungstechnik. Vol. 32, pg. 157-164. Apr. 1978.

[5] J. Van Bladel, *Electromagnetic Fields*. New York: McGraw-Hill, 1964.

[6] M. L. Waller, "Development and Application of Adaptive Basis Functions to Generate a Diagonal Moment Method Matrix for Electromagnetic Field Problems," Ph.D. Dissertation Auburn University, Aug. 2001.

[7] M. L. Waller, Rao S.M., "Application of adaptive basis functions for a diagonal moment matrix solution of arbitrarily shaped three-dimensional conducting body problems," *IEEE Trans. Antennas Propag.*, Vol. 50, Iss. 10, pg. 1445- 1452, Oct. 2002

[8] V. Rokhlin, "Rapid solution of integral equations of scattering theory in two dimensions," Journal of Computational Physics, Vol. 86, Iss. 2, pg. 414-439, 1990.

[9] W. Gibson, *The Method of Moments in Electromagnetics*. Chapman and Hall/CRC: Taylor and Francis Group, 2008

[10] Y. Zhang, M. Taylor, T. Sarkar, A. De, M. Yuan, H. Moon, C. Liang, "Parallel in-core and out-of-core solution of electrically large problems using the RWG basis functions," IEEE Ant. and Prop. Magazine, Vol. 50, No. 5, Oct. 2008.

[11] V. V. S. Prakash, R. Mittra "Characteristic basis function method: A new technique for efficient solution of method of moments matrix equations," Microwave and Optical Technology Letters, Vol. 36, Iss. 2, pg. 95-100.

[12] T. Sarkar, S. M. Rao, "The application of the conjugate gradient method for the solution of electromagnetic scattering from arbitrarily oriented wire antennas," Antennas and Propagation, IEEE Transactions on , Vol. 32, No. 4, pg. 398-403, Apr 1984

[13] J. R. Westlake, *A Handbook of Numerical Inversion and Solution of Linear Equations*. New York, Wiley, 1968.

[14] "CUDA Zone – The resource for CUDA developers," NVIDIA, 23 July 2009 http://www.nvidia.com/cuda.

[15] "NVIDIA CUDA Programming Guide Version 2.1," NVIDIA Dec. 2008, available at http://www.nvidia.com/cuda.

[16] "The impedance matrix localization (IML) method for moment-method calculations," IEEE Ant. and Prop. Magazine, Vol. 32, No. 5, pg. 18-30.

[17] "A New Technique to Generate a Sparse Matrix Using the Method of Moments for Electromagnetic Scattering Problems," Microwave and Optical Technology Letters Vol. 19, No. 4, Nov. 1998.

Appendices

Appendix A

Block Gauss-Seidel Iterative Solver

The Gauss-Seidel iterative scheme [13] for solving linear systems may be used to construct an analogous method for partitioned systems. The Gauss-Seidel solver is a straightforward algorithm for solving a linear system:

$$Ax = b \tag{A.1}$$

If there are M unknowns in the vector x, we obtain a set of solutions x_m^k for m = 1,2,...,M for each iteration step k. For the first iteration (k=0) we may simply put in a "guess" for the solution. A simple choice is to let $x = 0$. Subsequent solutions may be obtained with the following summation:

$$x_m^{k+1} = \frac{1}{A_{m,m}} \left(b_m - \sum_{\substack{i=1 \\ i \neq m}}^{N} A_{m,i}\, x_i^k \right) \tag{A.2}$$

where m = 1,2,...,M but the x_m values may be calculated in any order.

If we have a partitioned system rather than a typical linear system, we may construct a similar algorithm:

$$[x]_m^{k+1} = [A]_{m,m}^{-1} \left([b]_m - \sum_{\substack{i=1 \\ i \neq m}}^{N} [A]_{m,i} [x]_i^k \right) \tag{A.3}$$

where each $[x]_m$ and $[b]_m$ are subvectors of x and b, respectively, and each $[A]_{m,n}$ is a submatrix of A. This we will call the Block Gauss Seidel method. Also, note that,

according to the summation formula in Eq. A.3, the calculated values $[x]_m^{k+1}$ are not used until the subsequent iteration. In this case, the algorithm is generally referred to as the Jacobi algorithm. However, in this work, the values of $[x]_m^{k+1}$ are reused as soon as they are available. This leads to faster convergence and is generally referred to as the Gauss-Seidel algorithm [13].

Appendix B

Radar Cross Section (RCS)

In this section, we present the mathematical steps involved in calculating the radar cross section (RCS) for the numerical results given in the previous chapters. Note that before any of the following steps may be applied, one must first obtain the electric currents for the given structure.

For two-dimensional problems, the following definition was used for the RCS:

$$\sigma = 2\pi\rho\frac{|\mathbf{E}^s|^2}{|\mathbf{E}^i|^2} \tag{B.1}$$

where $\mathbf{E}^i$ is the incident field and $\mathbf{E}^s$ is the scattered field calculated in the far-field in the direction of

$$\hat{\rho} = \cos\theta\hat{x} + \sin\theta\hat{y} \tag{B.2}$$

The scattered electric field in the far-field region is given by

$$\mathbf{E}^s = -j\omega\mathbf{A} \tag{B.3}$$

where $\mathbf{A}$ is the magnetic vector potential and $\omega = 2\pi f$ with f as the frequency for the problem. For the two-dimensional examples, this quantity is

$$\mathbf{E}^s = -\frac{k\eta}{4}\int_C \mathbf{J}(\rho')H_o^{(2)}(k|\rho - \rho'|)dC' \tag{B.4}$$

Using the far-field approximation for the kernel we have as $x \to \infty$,

$$H_o^{(2)}(x) \to \sqrt{\frac{2j}{\pi x}} e^{-jx} \tag{B.5}$$

Also, since we are in the far-field, we can approximate $|\rho - \rho'|$ as

$$|\rho - \rho'| = \frac{(\rho - \rho') \cdot (\rho - \rho')}{|\rho - \rho'|} \approx \rho - \hat{\rho} \cdot \rho' \tag{B.6}$$

Substituting B.5 and B.6 into B.4 we arrive at

$$\mathbf{E}^s = -k\eta \sqrt{\frac{j}{8k\pi}} \frac{e^{-jk\rho}}{\sqrt{\rho}} \int_C \mathbf{J}(\rho') e^{jk(\hat{\rho} \cdot \rho')} dC' \tag{B.7}$$

Note this equation is valid for both TM and TE incidence. The polarization of $\mathbf{J}$ will change depending upon the incident wave polarization as will the way the integration is performed since it depends upon the current expansion functions.

We may derive the three-dimensional case in a similar fashion. Here, the RCS definition is given by

$$\sigma = \lim_{r \to \infty} 4\pi r^2 \frac{|\mathbf{E}^s|^2}{|\mathbf{E}^i|^2} \tag{B.8}$$

where $\mathbf{E}^i$ is the incident field and $\mathbf{E}^s$ is the scattered field calculated in the far-field in the direction of

$$\hat{\mathbf{r}} = \sin\theta \cos\phi \hat{\mathbf{x}} + \sin\theta \sin\phi \hat{\mathbf{y}} + \cos\theta \hat{\mathbf{z}} \tag{B.9}$$

The electric field in the far-field region is given by:

$$\mathbf{E}^s = -j\omega \mathbf{A} = -j\omega\mu \int \mathbf{J} \frac{e^{-jkR}}{4\pi R} dv' \tag{B.10}$$

If we let $\mathbf{r}'$ be the integration coordinates, then as $r \to \infty$ we can approximate R as

$$R = \frac{(\mathbf{r} - \mathbf{r}') \cdot (\mathbf{r} - \mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \approx \mathbf{r} - \hat{\mathbf{r}} \cdot \mathbf{r}' \tag{B.11}$$

giving us

$$\mathbf{E}^s \approx \frac{-j\omega\mu}{4\pi r} e^{-jkr} \int \mathbf{J} e^{jk\hat{\mathbf{r}}\cdot\mathbf{r}'} dv' \tag{B.12}$$

For RWG current expansion functions, we can evaluate the integral as

$$\mathbf{E}^s = \frac{-j\omega\mu}{8\pi r} e^{-jkr} \sum_{n=1}^{N_c} \alpha_n l_n \left[\rho_n^{c+} e^{jk\hat{\mathbf{r}}\cdot\mathbf{r}_n^{c+}} + \rho_n^{c-} e^{jk\hat{\mathbf{r}}\cdot\mathbf{r}_n^{c-}} \right] \tag{B.13}$$

where the α_n's are the coefficients for the expansion functions. From these equations, we may obtain the radar cross section of any arbitrary conductor in either two or three dimensions.

Appendix C

Matrix Partitioning and Parallel Processing

A topic closely related to this work is inverting a partitioned matrix. Typically, matrix inversion is an $O(N^3)$ process for full matrices. Furthermore, it is difficult to create a parallel matrix inversion scheme. However, using the basic concepts present in this work, it may be possible to partition a matrix and, by trading submatrix inverses for matrix multiplies, create a highly parallel solution scheme. Consider the partitioned matrix:

$$Z = \begin{bmatrix} [Z_{1,1}] & [Z_{1,2}] \\ [Z_{2,1}] & [Z_{2,2}] \end{bmatrix} \tag{C.1}$$

Also suppose that we interpret this as representing a MoM system matrix for two bodies where $Z_{i,j}$ is the submatrix for expansion functions on body j and testing functions on body i. If we replace the basis functions on body 1 with functions producing nulls on body 2 and vice versa, then we can completely decouple the bodies and then solve each system independently. In terms of matrix operations, we wish to produce a matrix R with the same partitioning scheme as Z with the following property:

$$\begin{aligned} \tilde{Z} &= \begin{bmatrix} [Z_{1,1}] & [Z_{1,2}] \\ [Z_{2,1}] & [Z_{2,2}] \end{bmatrix} \begin{bmatrix} [R_{1,1}] & [R_{1,2}] \\ [R_{2,1}] & [R_{2,2}] \end{bmatrix} \\ &= \begin{bmatrix} \left[\tilde{Z}_{1,1}\right] & [0] \\ [0] & \left[\tilde{Z}_{2,2}\right] \end{bmatrix} \end{aligned} \tag{C.2}$$

where $\tilde{Z}$ is a new system matrix resulting from a change of basis. Furthermore, we will force the diagonal blocks of R to be identity matrices:

$$[R_{1,1}] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{C.3}$$

$$[R_{2,2}] = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{C.4}$$

Using the fact that the off-diagonal blocks of $\tilde{Z}$ must be zero we arrive at the following relationships:

$$Z_{2,1} + Z_{2,2}R_{2,1} = 0 \Rightarrow R_{2,1} = -Z_{2,2}^{-1}Z_{2,1} \tag{C.5}$$

$$Z_{1,1}R_{1,2} + Z_{1,2} = 0 \Rightarrow R_{1,2} = -Z_{1,1}^{-1}Z_{1,2} \tag{C.6}$$

We can then construct the diagonal blocks of $\tilde{Z}$ with the following equations:

$$\tilde{Z}_{1,1} = Z_{1,1} + Z_{1,2}R_{2,1} \tag{C.7}$$

$$\tilde{Z}_{2,2} = Z_{2,2} + Z_{2,1}R_{1,2} \tag{C.8}$$

Then, we can invert $\tilde{Z}$ by simply inverting the diagonal blocks:

$$\tilde{Z}^{-1} = \begin{bmatrix} \left[\tilde{Z}_{1,1}\right]^{-1} & [0] \\ [0] & \left[\tilde{Z}_{2,2}\right]^{-1} \end{bmatrix} \tag{C.9}$$

Also, note that

$$\tilde{Z}^{-1} = R^{-1}Z^{-1} \tag{C.10}$$

So we can get the inverse of Z by simply multiplying $\tilde{Z}^{-1}$ by R:

$$\begin{aligned} Z^{-1} &= R\tilde{Z}^{-1} \\ &= \begin{bmatrix} [I] & [R_{1,2}] \\ [R_{2,1}] & [I] \end{bmatrix} \begin{bmatrix} \left[\tilde{Z}_{1,1}^{-1}\right] & [0] \\ [0] & \left[\tilde{Z}_{2,2}^{-1}\right] \end{bmatrix} \qquad \text{(C.11)} \\ &= \begin{bmatrix} \left[\tilde{Z}_{1,1}^{-1}\right] & \left[R_{1,2}\tilde{Z}_{2,2}^{-1}\right] \\ \left[R_{2,1}\tilde{Z}_{1,1}^{-1}\right] & \left[\tilde{Z}_{2,2}^{-1}\right] \end{bmatrix} \qquad \text{(C.12)} \end{aligned}$$

Substituting the values found in Eqs. C.5 to C.8 we can get the inverse in terms of the partitions for the original system matrix Z:

$$Z^{-1} = \begin{bmatrix} \left[\left(Z_{1,1} - Z_{1,2}Z_{2,2}^{-1}Z_{2,1}\right)^{-1}\right] & \left[-Z_{1,1}Z_{1,2}\left(Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2}\right)^{-1}\right] \\ \left[-Z_{2,2}Z_{2,1}\left(Z_{1,1} - Z_{1,2}Z_{2,2}^{-1}Z_{2,1}\right)^{-1}\right] & \left[\left(Z_{2,2} - Z_{2,1}Z_{1,1}^{-1}Z_{1,2}\right)^{-1}\right] \end{bmatrix} \tag{C.13}$$

Viewing it this way, we can see that there are now 4 partition inverses, 4 partition multiplies (if done in the proper order), and 2 matrix subtractions. Now suppose Z is of dimensions $N \times N$ with N an even number and each of the partitions has dimensions $\frac{N}{2} \times \frac{N}{2}$. Then, taking all 4 partition inverses will result in about half the number of operations as taking the full matrix inverse since $4\left(\frac{N}{2}\right)^3 = \frac{N^3}{2}$. Although this part of the algorithm cannot be made fully parallel, it is possible to substantially accelerate it with multiple processors [10]. The remaining operations are only matrix multiplies and subtractions. In these two cases, one can achieve up to 100% efficiency when doing parallel processing.

Now consider a case where we have an $N \times N$ matrix with $N = 2^m$ for some integer m. We can partition our matrix into 2×2 partitions:

$$Z = \begin{bmatrix} [2 \times 2] & [2 \times 2] & \dots & [2 \times 2] \\ [2 \times 2] & [2 \times 2] & & \\ \vdots & & \ddots & \\ [2 \times 2] & & & [2 \times 2] \end{bmatrix} \tag{C.14}$$

We apply the partitioned inverse scheme recursively:

$$Z = \begin{bmatrix} [2 \times 2] & [2 \times (N-2)] \\ [(N-2) \times 2] & [(N-2) \times (N-2)] \end{bmatrix} \tag{C.15}$$

We can use this step until we reach a point where we either a) already have the needed partition inverses we need or b) must invert a 2×2 partition. For case b, the inverse may be performed analytically and each element may be calculated in parallel. Furthermore, as mentioned previously, all remaining operations may be performed in parallel. So, this strategy may offer a highly parallel alternative for computing a full matrix inverse. A full inverse is important in a linear system when the number of right hand sides is larger than the dimensions of the system. In this case, it is more efficient to compute a full matrix inverse rather than performing an LU decomposition and then solving for multiple right hand sides.

The following Fortran 90 code implements the above procedure and performs an in-place matrix inversion for a matrix have dimensions $N \times N$ where $N = 2^m$ for some integer m.

```
program fm_sol
  use fm_lib
  implicit none
  integer :: m, n, nb, dm
  real :: myre1, myre2, mylg
  complex :: j = (0,1)
  complex, dimension(:,:), allocatable :: Zsi

  nb = 512 ! matrix dimension

  ! Fill matrix Zsi with random real and
  ! imaginary values between -5.0 and 5.0
  allocate(Zsi(nb,nb))
  do m = 1,nb
    do n = 1,nb
      call random_number(myre1)
      call random_number(myre2)
      Zsi(m,n) = (10.0*myre1-5.0) + j * (10.0*myre2-5.0)
    end do
  end do
  mylg = (log(real(nb))/log(2.0))
  dm = nint(mylg) ! number of (recursive) levels
  call pwtin(Zsi) ! Perform matrix inverse
end program fm_sol
```

```
module fm_lib

contains

subroutine tby2i(Zs)
  implicit none
  complex, dimension(2,2), intent(inout) :: Zs
  complex :: mydet, mytmp

  mydet = Zs(1,1)*Zs(2,2) - Zs(1,2)*Zs(2,1)
  mytmp = Zs(1,1)
  Zs(1,1) = Zs(2,2)
  Zs(2,2) = mytmp
  Zs(1,2) = -Zs(1,2)
  Zs(2,1) = -Zs(2,1)
  Zs(1,1) = Zs(1,1)/mydet
  Zs(1,2) = Zs(1,2)/mydet
  Zs(2,1) = Zs(2,1)/mydet
  Zs(2,2) = Zs(2,2)/mydet
end subroutine tby2i
```

```
! In-place matrix multiply
! Replaces Za
subroutine ipmml(Za,Zb)
  implicit none
  complex, dimension(:,:), intent(inout) :: Za
  complex, dimension(:,:), intent(in) :: Zb
  integer :: m,n,p,mc,nc,pc
  complex, dimension(size(Za,2)) :: Rt

  mc = size(Za,1)
  nc = size(Zb,2)
  pc = size(Za,2)

  do m = 1,mc
    Rt = 0.0
    do n = 1,nc
      do p = 1,pc
        Rt(n) = Rt(n) + Za(m,p)*Zb(p,n)
      end do
    end do
    Za(m,1:nc) = Rt
  end do
end subroutine ipmml
```

```
! In-place matrix multiply
! Replaces Zb
subroutine ipmmr(Za,Zb)
  implicit none
  complex, dimension(:,:), intent(in) :: Za
  complex, dimension(:,:), intent(inout) :: Zb
  integer :: m,n,p,mc,nc,pc
  complex, dimension(size(Zb,1)) :: Rt

  mc = size(Za,1)
  nc = size(Zb,2)
  pc = size(Za,2)

  do n = 1,nc
    Rt = 0.0
    do m = 1,mc
      do p = 1,pc
        Rt(m) = Rt(m) + Za(m,p)*Zb(p,n)
      end do
    end do
    Zb(1:mc,n) = Rt
  end do
end subroutine ipmmr
```

```
! Compute Za - Zb*Zc replacing Za
subroutine ipsmm(Za,Zb,Zc)
  implicit none
  complex, dimension(:,:), intent(inout) :: Za
  complex, dimension(:,:), intent(in) :: Zb, Zc
  integer :: m,n,p,mc,nc,pc
  complex, dimension(size(Za,2)) :: Rt

  mc = size(Zb,1)
  nc = size(Zc,2)
  pc = size(Zb,2)

  do m = 1,mc
    Rt = 0.0
    do n = 1,nc
      do p = 1,pc
        Rt(n) = Rt(n) + Zb(m,p)*Zc(p,n)
      end do
      Za(m,n) = Za(m,n) - Rt(n)
    end do
  end do
end subroutine ipsmm
```

```
! Compute Za + Zb*Zc replacing Za
subroutine ipamm(Za,Zb,Zc)
  implicit none
  complex, dimension(:,:), intent(inout) :: Za
  complex, dimension(:,:), intent(in) :: Zb, Zc
  integer :: m,n,p,mc,nc,pc
  complex, dimension(size(Za,2)) :: Rt

  mc = size(Zb,1)
  nc = size(Zc,2)
  pc = size(Zb,2)

  do m = 1,mc
    Rt = 0.0
    do n = 1,nc
      do p = 1,pc
        Rt(n) = Rt(n) + Zb(m,p)*Zc(p,n)
      end do
      Za(m,n) = Za(m,n) + Rt(n)
    end do
  end do
end subroutine ipamm
```

```
recursive subroutine pwtin(Za)
        implicit none
        complex, dimension(:,:), target, intent(inout) :: Za
        integer :: m,n,nb
        complex, dimension(:,:), pointer :: pa, pb, pc, pd

  nb = size(Za,1)

  pa => Za(1:nb/2,1:nb/2)
  pb => Za(1:nb/2,nb/2+1:nb)
  pc => Za(nb/2+1:nb,1:nb/2)
  pd => Za(nb/2+1:nb,nb/2+1:nb)

  if (nb==2) then
    call tby2i(Za)
    return
  end if

  call pwtin(pd)
  call ipmml(pb,pd)
  call ipsmm(pa,pb,pc)
  call pwtin(pa)
  call ipmmr(pd,pc)
  call ipmmr(pa,pb)
  call ipamm(pd,pc,pb)
```

```
    call ipmml(pc,pa)

    do m = 1,size(pb,1)
      do n = 1,size(pb,2)
        pb(m,n) = -pb(m,n)
      end do
    end do

    do m = 1,size(pc,1)
      do n = 1,size(pc,2)
          pc(m,n) = -pc(m,n)
      end do
    end do
  end subroutine pwtin
end module fm_lib
```

Appendix D

GPU Acceleration

GPUs (Graphics Processing Units) may contain hundreds of cores and are able to execute up to thousands of threads simultaneously [14]. Although the codes used in this work have not been accelerated with GPU hardware, this has the potential to speed up the overall procedure significantly. In fact, the major limiting factor in this work has been the matrix fill time for the MoM system matrix. Luckily, filling this matrix can be done in a highly parallel manner. Each element may be calculated independently from all the other elements in the matrix. Furthermore, they may be calculated in any order. Thus, this procedure is embarrassingly parallel and is suitable for implementation on a GPU.

To test this idea, we have implemented a standard two-dimensional EFIE formulation with TM polarization as seen in Chapter 2. Here, each thread is assigned to a single element of the system matrix. Threads working in parallel may then quickly fill the matrix. In order to solve the linear system, we used a conjugate gradient iterative solver. Iterative solvers are more amenable to parallel processing than are direct solutions. In this algorithm, there are a number of matrix-vector multiplies and these determine how quickly the solver executes. To implement this procedure on the GPU, we assigned each thread to a row in the system matrix. In each matrix-vector multiply, each thread performs one dot product between its assigned row and the vector by which the matrix is multiplied.

Next, we have tested this code on both a 2.67 GHz Core i7 CPU and a NVIDIA Tesla C1060 GPU and compared the results for unknowns numbering from 1000 up

to 22000 (the upper memory limit for the GPU). The ratio of the execution time for the CPU to that of the GPU is shown in Figure D.1. Note that, for each case, there is a roughly linear gain in speed as the number of unknowns increases. This implies that the growth rate for the GPU execution time is an order of magnitude less than that of the CPU.

There is also a fluctuation in the graph for the matrix fill time. This could be due to a number of factors. First, when programming this routine, the threads must be assigned in multiples of 32 for maximum performance (an artifact of the NVIDIA cards) [15]. However, the matrix cannot always be perfectly divided into partitions of this size. To remedy this situation, the implementation used here will fill the largest submatrix of the system matrix that may be partitioned in this way. After that, the remaining terms are calculated separately in smaller blocks. This introduces latency that may behave in an unpredictable manner. Furthermore, there may be differences in how the compilers translate each of the codes. A separate compiler must be used for the CPU and GPU and they may optimize operations differently. Also, the speed at which different operations (multiplication, addition, etc.) are handled on each device may vary as the GPU has a complex cache/memory architecture. Finally, the variables for the GPU are in single precision (a hardware limitation) while those on the CPU are in double. This affects how the operations are performed as well as how the code is compiled and optimized.

The speed of the gradient solver also improves as the number of unknowns increases, although much more slowly than does the matrix fill. This is due to the fact that the solver has only been parallelized along one dimension of the matrix, while the matrix has been parallelized along two. Still, the solver is almost ten times faster than the CPU for large problems.

Overall, it appears that this technique would greatly accelerate the matrix fill and therefore the the algorithm presented in this work. Furthermore, inexpensive programmable GPUs are currently available for desktop machines and can also be found in many supercomputers making this a practical option.

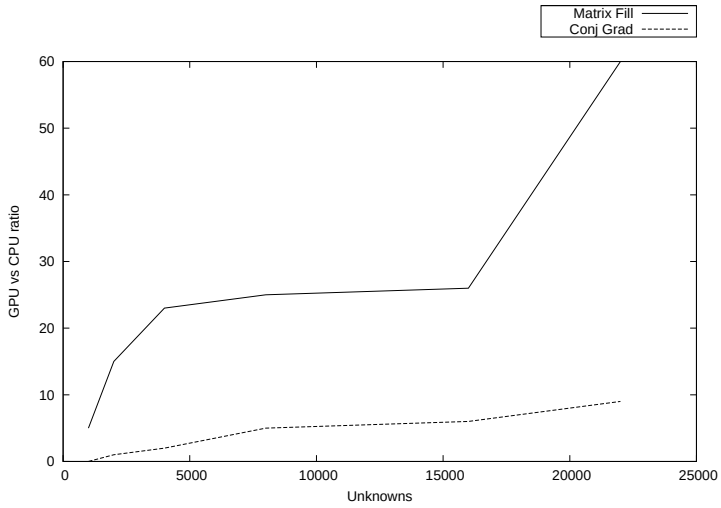

Figure D.1: Ratio of CPU to GPU execution times for matrix fill and conjugate gradient solver.