

https://dspace.org/

The development of a new preconditioner by modifying the simply sparse compression matrix to solve electromagnetic method of moments problems

Dreyer, Renier Lambertus

2010-02-12T12:26:00Z
http://hdl.handle.net/10539/7508

Downloaded from a DSpace repository. https://dspace.org/

The Development of a New Preconditioner by Modifying the Simply Sparse Compression Matrix to Solve Electromagnetic Method of Moments Problems.

Renier Lambertus Dreyer

A thesis submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Doctor of Philosophy.

Johannesburg, June 2009

Declaration

I declare that this thesis is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Doctor of Philosophy in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other university.

Signed this 20th day of October, 2009

_____

Renier Lambertus Dreyer.

Abstract

The aim of this research was to improve the matrix solution methods for SuperNEC MoM problems, which is an electromagnetic simulation software package used to model antennas, and develop a new preconditioner for the iterative method BICGSTAB(L).
This was achieved by firstly implementing the ATLAS BLAS library optimised for a specific computer architecture. The ATLAS code primarily makes use of code generation to build and optimise applications. Comparisons show that the matrix solution times using LU decomposition optimised by ATLAS is improved by between 4.1 and 4.6 times, providing a good coding platform from which to compare other techniques.
Secondly the BICGSTAB iterative solution method in SuperNEC was improved by making use of an alternative algorithm BICGSTAB(L). Systems of equations that converged slowly or not at all using BICGSTAB, converged more quickly when using BICGSTAB(L) with L set to 4, despite the high condition numbers in the coefficient matrices.
Thirdly a domain decomposition method, Simply Sparse, was characterised. Investigations showed that Simply Sparse is a good compression technique for SuperNEC MoM matrices. The custom Simply Sparse solver also solves large matrix problems more quickly than LU decomposition and scales well with increased problem sizes. LU decomposition is still however quicker for problems smaller than 7000 unknowns as the overheads in compressing the coefficient matrices dominate the Simply Sparse method for small problems.

Lastly a new preconditioner for BICGSTAB(L) was developed using a modified form of the Simply Sparse matrix. This was achieved by considering the Simply Sparse matrix to be equivalent to the full coefficient matrix $[A]$. The largest 1% to 2% of the Simply Sparse elements was selected to form the basis of the preconditioning matrix. These elements were further modified by multiplying them by a large constant i.e. 1×10^{7}. The system of equations was then solved using BICGSTAB(L) with L set to 4.

The new preconditioned BICGSTAB(L) algorithm is quicker than both LU decomposition and the custom Simply Sparse solution method for problems larger than 5000 unknowns.

This thesis is dedicated to my parents,

Clarence and Riekie Dreyer,

my brothers,

Deon and Etienne Dreyer,

and my Grandparents

Bertus and Esmè Du Preez,

Thabo (1913 – 1982) and Millie Dreyer (1919 – 2007)

Acknowledgements

I wish to thank and acknowledge:

My Supervisor, Prof. Alan Clark for his patience and willingness to discuss any of my questions.

Dr Francis Canning, for his help and advice in using Simply Sparse and explaining the theory behind it.

Dr. Derek Nitch, for providing insight into the workings of SuperNEC and iterative techniques in general.

Dr. Andre Fourie, for helping me see the bigger picture.

Contents

Declaration ii
Abstract iii
Acknowledgements v
Contents i
List of Figures iv
List of Tables v
List of Symbols vi
1. Introduction 1
1.1. Aim 1
1.2. Importance of the Problem 1
1.3. Limitations 2
1.4. Data Collection Techniques 3
1.5. Overview of the Thesis 3
2. Background 5
2.1. LU Decomposition in SuperNEC 5
2.1.1. Solution of the matrix equation 5
2.2. Iterative Methods and Preconditioners 6
2.2.1. Iterative Methods 6
2.2.2. Preconditioners 9
2.3. Domain Decomposition Methods 10
3. Code Optimisation for Specific Computer Hardware using ATLAS 12
3.1. Introduction 12
3.2. ATLAS summarised 12
3.3. Method of Optimisation 13
3.4. Timing Results for various SuperNEC Problems 14
3.5. Summary 19
4. BICGSTAB(L) method for SuperNEC Problems 21
4.1. Introduction 21
4.2. BICGSTAB(L) introduced 21
4.3. Conjugate Gradient Method 22
4.4. Bi-Conjugate Gradient Method 23
4.5. Conjugate Gradient Squared Method (CGS) 24
4.6. Bi-Conjugate Gradient Stabilized Method 24
4.7. Bi-Conjugate Gradient Stabilized L Method 25
4.8. BICGSTAB in SuperNEC 26
4.9. Results for various Problems in SuperNEC 27
4.10. Chapter Summary 30

5. Simply Sparse 31
5.1. Introduction 31
5.2. Simply Sparse Summarised 31
5.3. Simply Sparse Experimental Results 34
5.4. Chapter Summary 35
6. Modified Simply Sparse Matrix as a Preconditioner 36
6.1. Various Solution Methods Investigated 36
6.1.1. ILU (Incomplete LU factorization) 36
6.1.2. Block Diagonal 36
6.1.3. SIM/SOR 37
6.2. Development of a New Preconditioning Matrix for BICGSTAB(L) 38
6.2.1. Concept 38
6.3. Results 40
6.3.1. Matrix Creation Times 40
6.3.2. Matrix Solution Times 41
6.3.3. Total Times 43
6.4. Chapter Summary 45
7. Conclusion 46
7.1. ATLAS 46
7.2. BICGSTAB(L) 47
7.3. Simply Sparse 47
7.4. Original Contribution (Preconditioning Technique) 47
7.5. Recommendation for Further Investigation 48
References 49
Appendix A: SuperNEC Theory 52
A1. Introduction 52
A2. The Integral Equation for Free Space 52
A3. The electric field integral equation (EFIE) 52
A4. Numerical Solution 54
A5. Current expansion on wires 56
A6. Evaluation of the fields 63
A7. The matrix equation for current 70
A8. Solution of the matrix equation 71
Appendix B: Absolute Error Plots for Preconditioned BICGSTAB(L) problems 76
B1. Introduction 76
B2. Results 76
B3. Conclusion 79
Appendix C: Preconditioning Multiplier Study 80

C1. Introduction 80
C2. Reasoning 80
C3. Experimental results 80
C4. Conclusion 81
Appendix D: Matrix Block Diagonal Inclusion Study 82
D1. Introduction 82
D2. Reasoning 82
D3. Experimental Results 82
D4. Conclusion 83
Appendix E: Interaction Distance Preconditioning with BICGSTAB(L) 84
E1. Introduction 84
E2. Experimental Results 84
E3. Conclusion 85
Bibliography 86

List of Figures

Figure 1: Flow Chart to Choose a Suitable Solution Method.....7
Figure 2: Pattern of a Simply Sparse Matrix. i.e. the location of non-zero elements are plotted. (Matrix order = 8007).....11
Figure 3: SuperNEC numerical model of the Hawk aircraft. 3269 segments.....15
Figure 4: Numerical model of a cube and dipole antenna illustrating small and large problem sizes. 807 segments to 8007 segments.....16
Figure 5: Numerical model of a cube and dipole antenna illustrating small and large problem sizes. 1207 segment to 5106 segments.....16
Figure 6: Various sections of the Hawk aircraft ranging from 579 segments to 4977 segments.....17
Figure 7: Factorisation and total times for the Hawk aircraft.....17
Figure 8: Factorisation and total times for the Box configuration.....18
Figure 9: Factorisation and total times for the Plate configuration.....18
Figure 10: Factorisation times compared for the Hawk, Box and Plate configuration showing problem type has no effect.....19
Figure 11: Comparison of SuperNEC problems where the ratio of the imaginary and real eigen values are shown.....27
Figure 12: Convergence rates for a typical SuperNEC problem, varying L from 1 to 4 in BICGSTAB(L). The condition number for this problem is 43.....29
Figure 13: Simple Sparse Matrix T.....33
Figure 14: Simply Sparse Matrix T permutation of the row and columns.....33
Figure 15: Comparing LU Decomposition and Simply Sparse (807 to 8007 unknowns).34
Figure 16: Graphic representation of a Block Diagonal Inclusion Method. 5 Blocks included on the left and 20 blocks on the right.....37
Figure 17: Various density matrices. From top left to bottom right they are 30%, 20%, 10%, 5%, 2% and 1%.....39
Figure 18: Comparison of Matrix Creation Times.....41
Figure 19: Comparison of Matrix Factorization/Solution times.....42
Figure 20: Comparing Total Run Times (i.e. Matrix Creation Time + Solution Time)....43
Figure 21: Absolute Error for a Problem with 5000 Unknowns.....44

List of Tables

Table 1: The average cost per Krylov dimension 25
Table 2: Convergence of BICGSTAB for different problems with various condition numbers 26
Table 3: Varying L for different size problems with a range of condition numbers 28
Table 4: Varying L for a problem with a large condition number (Hawk Aircraft) 29
Table 5: Varying the Preconditioning Multiplier for a problem with 807 unknowns (Simulation stopped if iterations were greater than 10) 39
Table 6: Comparing Preconditioned BICGSTAB(L) with optimised LU decomposition 42

List of Symbols

Frequency in Mega Hertz	MHz
Time	t
Number of unknowns (i.e. the order of the matrix)	N
Coefficient matrix (i.e. impedance matrix)	$[A]$
Vector of unknown currents	$[x]$
Vector of applied voltages	$[b]$
Lower triangular matrix	$[L]$
Upper triangular matrix	$[U]$
Constants	a, c, B
Iteration number	k, i
Complex vector	z
Coefficient matrix approximation	$[M]$
Diagonal matrix	$[D]$
SSOR parametric constant	ω
Constant for BICGSTAB(L)	L
BICGSTAB polynomial stabilising coefficient	ω_k
Residual	r
Search vector	p
Iterative method approximation to $[x]$	x_0
Unitary matrix	$[V]$, $[W]$
Transformed impedance matrix	$[T]$

1. Introduction

1.1. Aim

The aim of this research is to improve the matrix solutions methods used by the electromagnetics (EM) software simulation package, SuperNEC [1], and in the process develop a new preconditioner for the iterative method BICGSTAB(L) [2].

This is achieved by:

1. Investigating and improving the current LU decomposition subroutines by optimising the software code for specific computer architectures using the Automatically Tuned Linear Algebra Subprogram Software (ATLAS) [3].
2. Investigate and improve on the current iterative method Bi Conjugate Gradient Stabilized (BICGSTAB) [4, p24] within SuperNEC by making use of a modified method BICGSTAB(L) .
3. Characterising the domain decomposition method, Simply Sparse [5], by solving various EM problems.
4. Develop a new preconditioner for BICGSTAB(L) by modifying and manipulating the Simply Sparse matrices, making sure the code has been optimised with ATLAS.

1.2. Importance of the Problem

The motivation behind conducting the research stems from the fact that although considerable work has been done on solving very large problems (more than 10 000 unknowns) with methods such as Simply Sparse [5], work still needs to be done to solve smaller problems (5 000 to 10 000 unknowns) more quickly with limited resources. These unknowns are the currents on the wire segments used to model structures in SuperNEC [1]. To accurately model the structure the wire segments need to be a tenth of a wavelength at the frequency of interest. For example, modelling a Hawk aircraft at 20MHz would require in the order of a few hundred wire segments, whereas modelling the same structure at 400MHz would require many thousands of segments.

This becomes critical when problems have multiple frequencies and where the EM structures might need to be optimised over a frequency range.

A good example of such a problem would be a numerical model of a Yagi-Uda antenna mounted on a platform, such as a ship. The design of the antenna can be optimised using techniques such as genetic algorithm optimisers. This would typically require upward of 40 generations (or simulations), with 20 chromosomes (or variables) simulated at a minimum of 10 frequencies, yielding 8000 matrices. Even with the advances in computational power, this example illustrates the need to solve matrices, small and large, as quickly as possible.

1.3. Limitations

There are a number of components that need to be looked at when evaluating how well a solution method performs. This is illustrated by the following expression [6, p1]:

$$t \propto aN^2 + cN^3 \tag{1}$$

Where

t is the time to solve a problem

aN^2 is the time taken to create a matrix with N unknowns

cN^3 is the time taken to solve a matrix with N unknowns

As the term cN^3 dominates the expression with increase in the size of N, this is what is concentrated on to improve solution methods. This research therefore also looks at the time taken to solve matrix equations, and does not look at optimising the time taken to create the matrix.

Although the creation time is not investigated, it is taken into account when comparing techniques, as different techniques have different overhead requirements to create a matrix.

It should be noted that fast solvers such as Simply Sparse scale differently. The cN^3 component becomes $N\log(N)$ but the creation of the matrix then has a large coefficient a, making the fast solver only practical for large problems.

There are also a number of different solution techniques and fast solvers that are not investigated, primarily because these are proprietary codes and do not exist within SuperNEC.

1.4. Data Collection Techniques

Data was acquired from a number of sources. This included timing data from SuperNEC, Matlab [7] and Simply Sparse. Different computing platforms, such as Core 2 Duo, Pentium IV, Pentium 3 and Celerons were used. Only data from the same platform was compared to avoid hardware architectural differences.

Different EM problems were constructed to investigate the solution techniques.

The problems primarily had to increase in size from as small as a few hundred unknowns to 8,000 unknowns. A secondary factor, which was investigated in the iterative solvers, was the condition number of the matrices. This was investigated by using real world problems, such as aircraft and helicopters for different degrees of ill-conditioned matrices, as well as idealised problems such as a plate and cube.

1.5. Overview of the Thesis

Chapter 2 provides background for the thesis:

- The LU decomposition method used in SuperNEC is outlined.
- The BICGSTAB method is explored and compared with other methods.
- Domain decomposition methods are introduced with an emphasis on Simply Sparse.

Chapter 3 looks at software code optimization for specific computer hardware using ATLAS:

- The basic ATLAS method is summarised.
- Timing results for various EM problems in SuperNEC are presented, showing the improvements ATLAS makes.

Chapter 4 presents the improved performance of BICGSTAB(L) over BICGSTAB using EM MoM problems.

- The BICGSTAB(L) method is summarised.
- Timing results for various EM problems in SuperNEC are presented, showing the improvement in performance.
- The L parameter within BICGSTAB(L) is investigated and a general value determined for all problem types.

Chapter 5 investigates the performance of Simply Sparse.

- The Simply Sparse method is summarised.
- Timing results for various EM problems are presented, showing the improvement in performance.

Chapter 6 presents the new preconditioner created from Simply Sparse matrices, and the improved performance over existing techniques within SuperNEC.

- Other methods that were investigated are highlighted.
- The new preconditioning method is outlined.
- Timing results and comparisons for various problem sizes are presented.

Chapter 7 summarises the findings, presenting the conclusions and recommendations for further work.

2. Background

Developing a new preconditioner for a system of linear equations can best be described in the words of Yousef Saad. [8, p 264].

> *"Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science. Theoretical results are rare and some methods work surprisingly well, often despite expectations"*

This thesis primarily addresses the "*art*" of creating a preconditioner by testing potential preconditioners through experimentation. There is however a substantial amount of background or "*science*" needed to understand preconditioners. The sections in this chapter introduce the basic concepts of matrix manipulation used in this thesis. This can be broken down into:

- Direct solution of matrices with LU decomposition in SuperNEC
- Iterative solution methods and preconditioners used in BICGSTAB
- Domain decomposition techniques such as Simply Sparse.

2.1. LU Decomposition in SuperNEC

Below is a modified excerpt from the SuperNEC technical reference manual [9], outlining the LU decomposition method. The solution to the system of equations finds the currents on the wire segments used to model structures within SuperNEC.

For further information on how SuperNEC creates the matrices using the Method of Moments (MoM) [10] approach please refer to Appendix A.

2.1.1. Solution of the matrix equation

The matrices to be solved within SuperNEC are of the form:

$$[A][x] = [b] \tag{2}$$

and is solved by Gaussian elimination .

Where $[A]$ is a square matrix and $[x]$ and $[b]$ are single column vectors.

The basic step is factorisation of the matrix $[A]$ into the product of an upper triangular matrix $[U]$ and a lower triangular matrix $[L]$ where

$$[A] = [L][U] \tag{3}$$

The matrix equation is then

$$[L][U][x] = [b] \tag{4}$$

from which the solution, $[x]$, is computed in two steps as

$$[L][F] = [b] \tag{5}$$

and

$$[U][x] = [F] \tag{6}$$

Equation (5) is first solved for [F] by forward substitution, and equation (6) is then solved for $[x]$ by backward substitution.

The major computational effort is factoring $[A]$ into $[L]$ and $[U]$. This takes approximately $\frac{1}{3}N^3$ multiplication steps for a matrix of order N compared to N^3 for inversion of $[A]$ by the Gauss-Jordan method.

2.2. Iterative Methods and Preconditioners

2.2.1. Iterative Methods

There are a number of factors to be considered when solving a system of equations. The three that could be considered the most important of these are memory, speed and accuracy. Iterative methods use successive approximations to obtain more accurate solutions to a linear system at each step [4, p5]. Iterative methods hope to solve the linear systems in questions more quickly than standard techniques, such as LU decomposition, to within the required accuracy using limited memory. Whereas LU decomposition provides an exact solution to the system of equations, iterative methods provide an approximation.

There are a wide variety of iterative techniques, from stationary techniques such as Jacobi, Gauss-Seidel, SOR and SSOR to non-stationary methods such as conjugate gradient (CG), minimum residual (MINRES), generalised minimum residual (GMRES), bi-conjugate gradient (BICG), quasi minimal residual (QMR), conjugate gradient squared (CGS), bi-conjugate gradient stabilised (BICGSTAB) and Chebyshev iterations [4, pp5-7]. Stationary iterative methods can be expressed as:

$$x^{(k)} = Bx^{(k-1)} + c \tag{7}$$

Where neither B nor c depend on the iteration count k [4, p7].

In non-stationary iterative methods the constants B and c are dependent on k.

Each of these techniques are suited to a specific type of problem. A flowchart suggesting the best method for a specific problem is shown in Figure 1 [4, p1].

Matrices generated by SuperNEC are typically non-symmetric, with no transpose readily available where storage is at a premium.

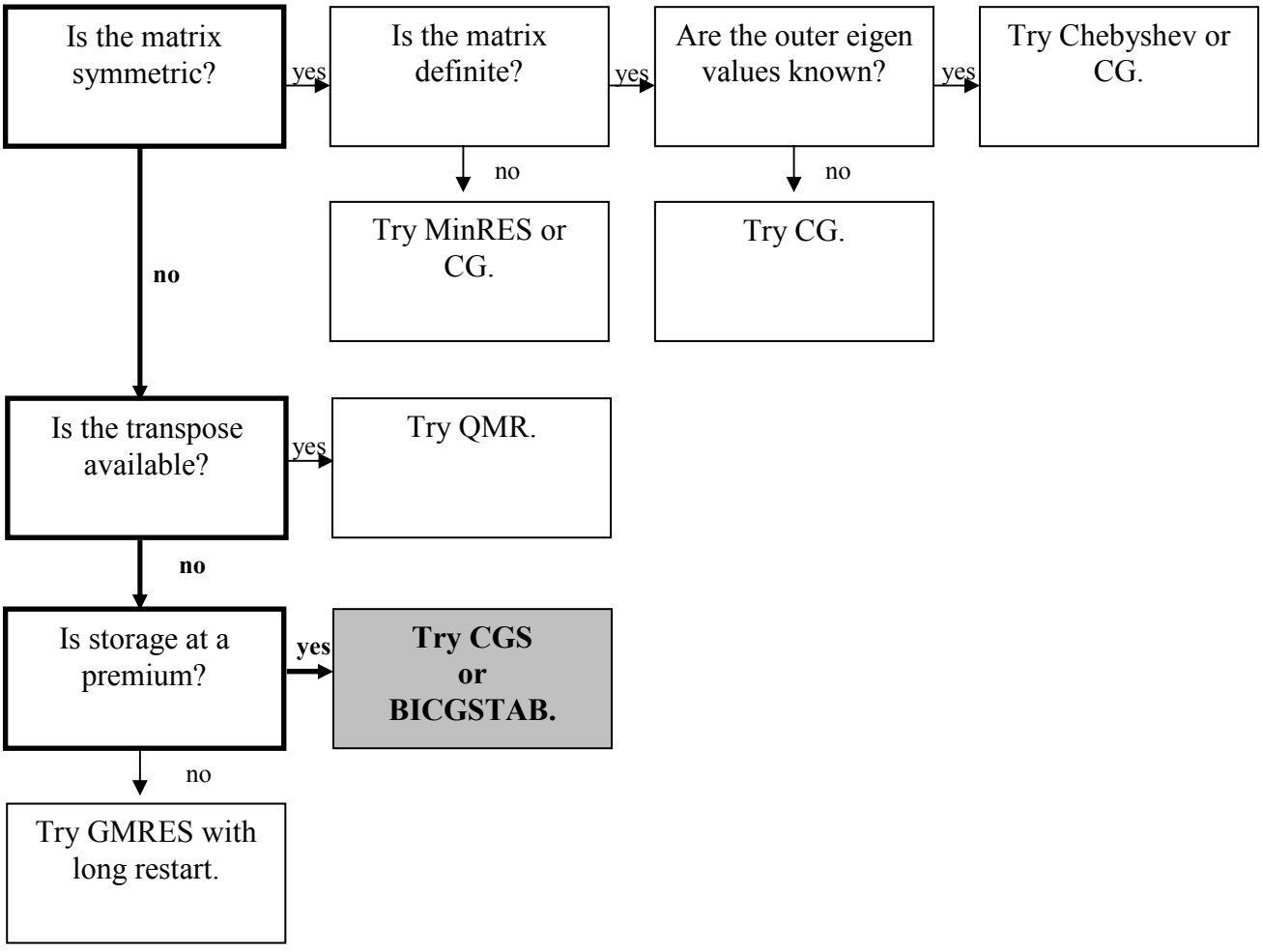


Figure 1: Flow Chart to Choose a Suitable Solution Method

Figure 1 shows that a good solution technique for SuperNEC problems would be CGS or BICGSTAB.

Some of these methods are summarised from [4] below.

Conjugate Gradient (CG).

The conjugate gradient method derives its name from the fact that it generates a sequence of conjugate (or orthogonal) vectors. These vectors are the residuals of the iterations. They are also the gradients of a quadratic functional, the minimization of which is equivalent to solving the linear system. CG is an extremely effective method when the coefficient matrix is symmetric positive definite, since storage for only a limited number of vectors is required.

An $N \times N$ complex symmetric matrix A is positive definite if $z^* Az > 0$ for all non-zero complex vectors z, where z^* is the conjugate transpose of z.

Bi-Conjugate Gradient (BICG).

The Bi-conjugate Gradient method generates two CG-like sequences of vectors, one based on a system with the original coefficient matrix $[A]$, and one on $[A]^T$. Instead of orthogonalizing each sequence, they are made mutually orthogonal, or "bi-orthogonal". This method, like CG, uses limited storage. It is useful when the matrix is non-symmetric and nonsingular; however, convergence may be irregular, and there is a possibility that the method will break down. BICG requires a multiplication with the coefficient matrix and with its transpose at each iteration.

Conjugate Gradient Squared (CGS).

The Conjugate Gradient Squared method is a variant of BICG that applies the updating operations for the $[A]$ sequence and the $[A]^T$ sequences both to the same vectors. Ideally, this would double the convergence rate, but in practice convergence may be much more irregular than for BICG. A practical advantage is that the method does not need the multiplications with the transpose of the coefficient matrix.

Bi-Conjugate Gradient Stabilized (BICGSTAB).

The BICGSTAB method is a variant of BICG, like CGS, but using different updates for the $[A]^T$ sequence in order to obtain smoother convergence than CGS.

2.2.2. Preconditioners

A preconditioner is any form of implicit or explicit modification of an original linear system which makes it "easier" to solve by a given method [8, p264]

For example, if a matrix $[M]$ approximates the coefficient matrix $[A]$ in some way, the transformed system:

$$[M]^{-1}[A][x] = [M]^{-1}[b] \tag{8}$$

has the same solution to the original system $[A][x] = [b]$, but the properties of the coefficient matrix $[M]^{-1}[A]$ may be more favourable to solve quickly [4, p35].

Using a preconditioner in an iterative method does however add additional overhead to the computational method used. The cost trade-off of constructing, and applying, the preconditioner needs to be outweighed by the improved rate of convergence for that iterative method.

There are a number of different types of traditional preconditioners.

Some are:

SSOR (excerpt from [4, p37]):

The SSOR preconditioner like the Jacobi preconditioner, can be derived from the coefficient matrix without any work. If the original, symmetric, matrix is decomposed as

$$[A] = [D] + [L] + [L]^T \tag{9}$$

in its diagonal $[D]$, lower $[L]$, and upper $[U]$ triangular part, the SSOR matrix is defined as

$$[M] = ([D] + [L])[D]^{-1}([D] + [L])^T \tag{10}$$

or, parameterized by ω

$$[M](\omega) = \frac{1}{2-\omega}\left(\frac{1}{\omega}[D] + [L]\right)\left(\frac{1}{\omega}[D]\right)^{-1}\left(\frac{1}{\omega}[D] + [L]\right)^T \tag{11}$$

The optimal value of the ω parameter will reduce the number of iterations to a lower order.
In practice, however, the information needed to calculate the optimal ω is prohibitively expensive. SSOR has been implemented in SuperNEC by Derek Nitch [11].

Jacobi:

One of the simplest preconditioner is the Point Jacobi, consisting of the diagonal of the matrix only. In terms of SuperNEC, the use of such a preconditioner would be investigated as the self impedance, or diagonal elements, of the wire segments could be considered to be the dominant components.

A variation on the Point Jacobi is the Block Jacobi, where variables can be separated into blocks or groups and manipulated separately [4, p37]. Jacobi Preconditioners require little storage, but more advanced preconditioners may perform better.

ILU (Incomplete LU factorisation):

ILU is based on the incomplete factorisation of the coefficient matrix A. The incomplete factorisation ignores non-zero fill elements during factorisation when those elements in the original matrix are zero.

The creation of these matrices are however non trivial. ILU may also break down even if the factorisation of the full matrix exists. [4, p38] The computational overhead of factorising the matrix may also be prohibitive.

Certain strategies exist that may improve the performance of ILU. These strategies look at different levels of fill in, i.e. what is included and what is excluded also known as drop tolerances. Elements may be excluded for a defined tolerance.

2.3. Domain Decomposition Methods

Domain decompositions methods revolve around the principle of divide and conquer. The partitioning of the problem, many times, stems from the actual nature of the physical problem, where a natural split occurs, and not just due to a numerical manipulation. These sub-regions might even be solved with different equations. The sub-problems may also be easier to work with as they are rectangular [8, p383].

Simply Sparse can be considered to be a domain decomposition technique, combined with matrix compression. The physical problem is divided into radiating regions and the currents flowing on the structure from each of those regions can be computed more easily. Figure 2 is an example of what the Simply Sparse matrix looks like. Note that the matrix is sparse, where white space indicates the elements have a magnitude of 0.

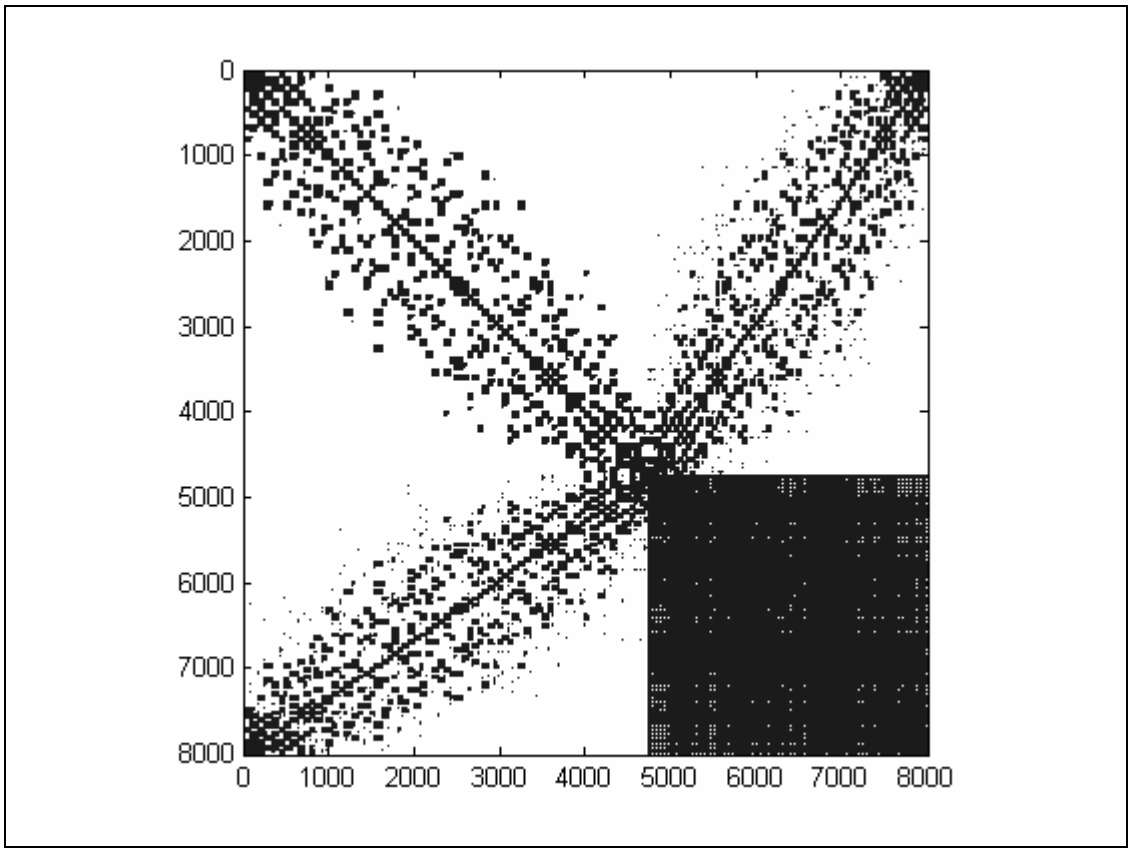


Figure 2: Pattern of a Simply Sparse Matrix. i.e. the location of non-zero elements are plotted. (Matrix order = 8007)

3. Code Optimisation for Specific Computer Hardware using ATLAS

In speeding up any software code, before algorithmic enhancements are applied, the existing code should be the optimisation on the hardware available for testing. This provides a good baseline for measurement, eliminating computer architecture as a variable when later looking at developing new or improved algorithms.

This chapter summarises the aim and methodology behind such an approach called the ATLAS project as discussed in [3] and presents timing results for a number of SuperNEC problems using un-optimised software libraries as compared to the optimised ATLAS libraries.

3.1. Introduction

Computational hardware is continuously evolving, making it necessary to constantly re-evaluate the programming paradigms used to create high performance computing libraries. These libraries are used extensively by the scientific and computational communities to simulate real world environments, where improved performance allows for greater accuracy and detailed results. A number of research methods and paradigms used to bridge the gap between hardware and software performance are being developed by various research groups. These include coding techniques to enhance object-orientated code such as Expression Templates [12], Temporary Base Class Idioms [13], and methods to make use of multiple cores such as the message passing interface (MPI) within parallel programming paradigms. A different approach is to automatically tune the software for a specific hardware architecture. Using code generators, or scripts, reduces the time required to tune software from months, in the case of hand-optimised methods, to hours. Extensive research has been conducted using this approach in the Automatically Tuned Linear Algebra Software (ATLAS) project using the Automated Empirical Optimisation of Software (AEOS) paradigm.

3.2. ATLAS summarised

Linear Algebra routines are widely used in the computational sciences. The performance of these linear algebra applications is often what limits scientists modelling more

complex problems, which in turn is what determines how closely reality can be modelled. This emphasises the need for efficient routines that take advantage of the hardware available.

When new hardware comes on the market, programmers usually optimise their code by hand. This is a slow process, threatening to make the code obsolete as new hardware is released every few months.

ATLAS implements a new paradigm to optimising linear algebra software, bringing the development cycles of hardware and software more inline with one another. The paradigm used in the ATLAS project (AEOS, Automated Empirical Optimisation of Software) focuses on using empirical techniques to provide portable high performance libraries.

The present emphasis of ATLAS is on the Basic Linear Algebra Subprograms (BLAS) [14,15,16]. Some degree of support is provided for all levels of the BLAS implementation and release of ATLAS (version 3.4.1) supports a small number of LAPACK [Linear Algebra PACKage] routines [17].

3.3. Method of Optimisation

The ATLAS project primarily makes use of code generation to build applications. This process involves search scripts and timing mechanisms to find the best method of performing a given operation for specific computer architecture.

The optimisation process can be broken down into the following basic operations [3, p5]:

- The performance critical routines need to be isolated
- A method of adapting the routines to different environments
- Robust context sensitive timers
- A search mechanism, to search for the optimum method, needs to be put in place

There are, in general, two methods for adapting routines to different environments. These are parameterised adaptation and code generation. Parameterised adaptation involves varying parameters, specific to a machine, within the source code. This can be done at compile time and have no effect on the runtime performance of the software. The parameterised approach is limited in that it cannot optimise for certain underlying architectural characteristics such as cache size. This can only be done by modifying the source code. Code generation is more flexible as it is a program that generates source

code, taking into account a number of parameters while able to modify the source code specific to a given architecture.

AEOS packages adapts to both hardware and software architectures [3, p7], matrix-vector operations, and level 3 BLAS, which performs matrix-matrix operations.

The performance enhancements are particularly relevant to SuperNEC as SuperNEC makes use of the BLAS to solve the equation

$$[A][x] = [b] \tag{12}$$

The different BLAS levels are optimised as follows [3, p9]:

Level 1: No memory reuse is possible in level 1 while small speedups are achieved from floating point usage

Level 2: Memory blocking allows reuse of vector operands but not, in general, matrix operands. This reduces vector operands from order N^2 to order N. Irreducible matrix costs, however, causes the memory load to remain at order N^2. Only a modest speedup can be achieved.

Level 3: Orders of magnitude speedup can be achieved in level 3 as operations can be blocked so that the fetch cost of order N^3 become order N^2. Triple nested loops used here are almost always too complex for compilers to optimise without hints from the programmer.

There are however a few limits to the ATLAS approach. These include the need for an adequate ANSI C compiler, hierarchical memory and a increase in compile time. These drawbacks are small compared to the performance improvements that can be achieved.

3.4. Timing Results for various SuperNEC Problems

Three complex structures, namely a Hawk aircraft shown in figure 3, a dipole above a cube shown in figure 4 and a dipole above a gridded plate in figure 5 were modelled using SuperNEC. The box and plate configuration were modelled at different frequencies, keeping their physical dimensions constant, to obtain varying size problems. A different approach was used for the Hawk aircraft. The Hawk aircraft was cut into sections. The first section was that of the nose, the second included more of the body of the aircraft

until the last section included most of the aircraft shown in figure 6. Using this approach to construct test case problems ensures the problems have different eigen values with different matrix condition numbers.

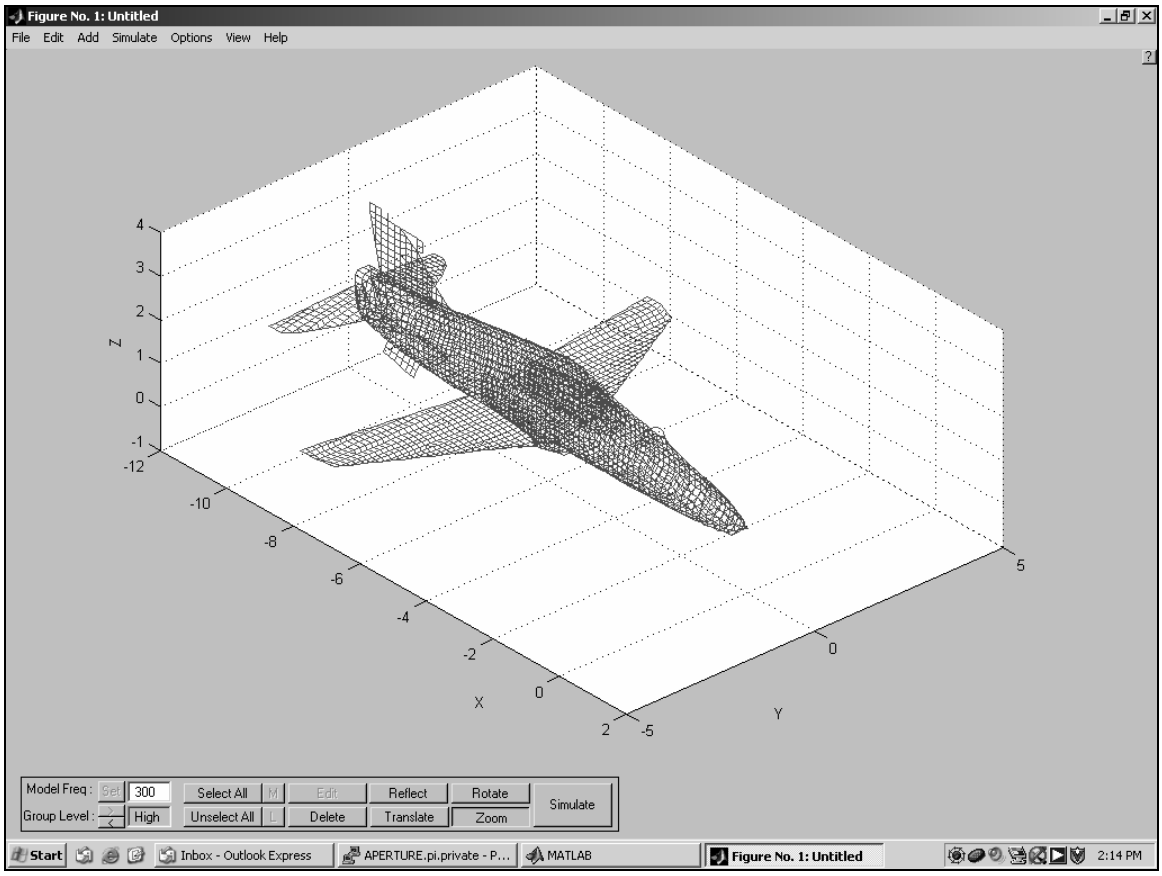


Figure 3: SuperNEC numerical model of the Hawk aircraft. 3269 segments

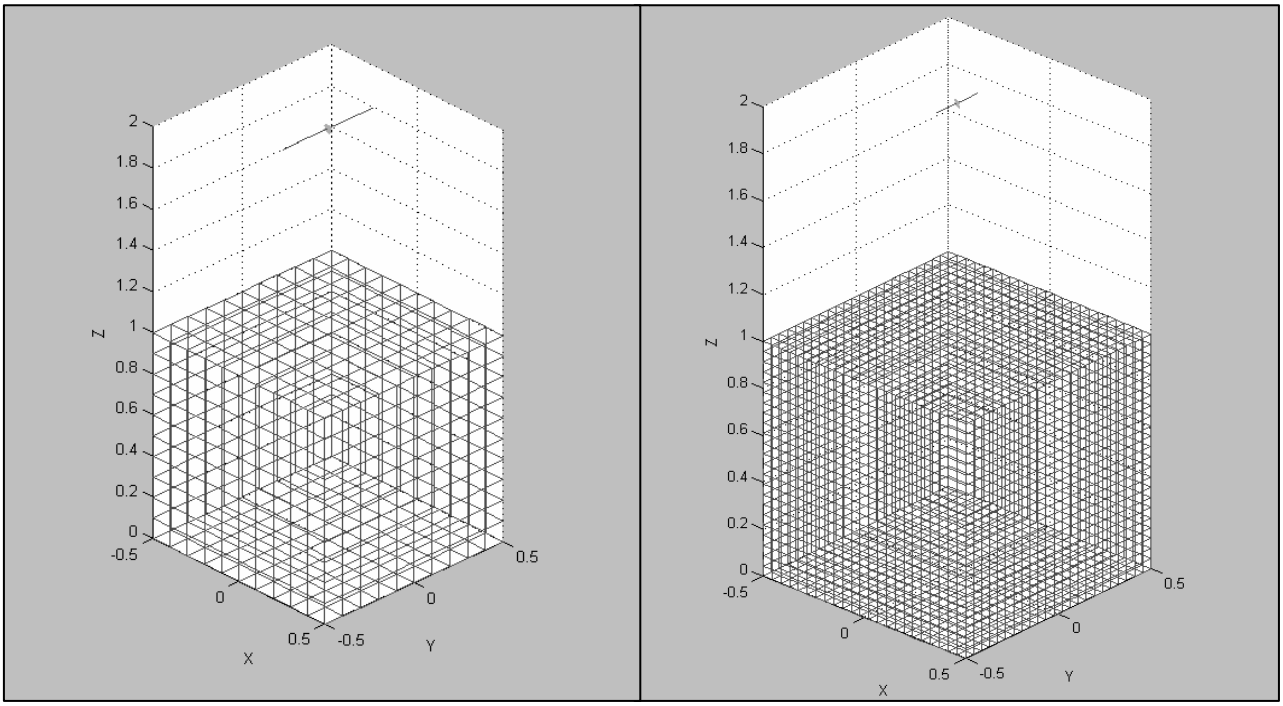


Figure 4: Numerical model of a cube and dipole antenna illustrating small and large problem sizes. 807 segments to 8007 segments

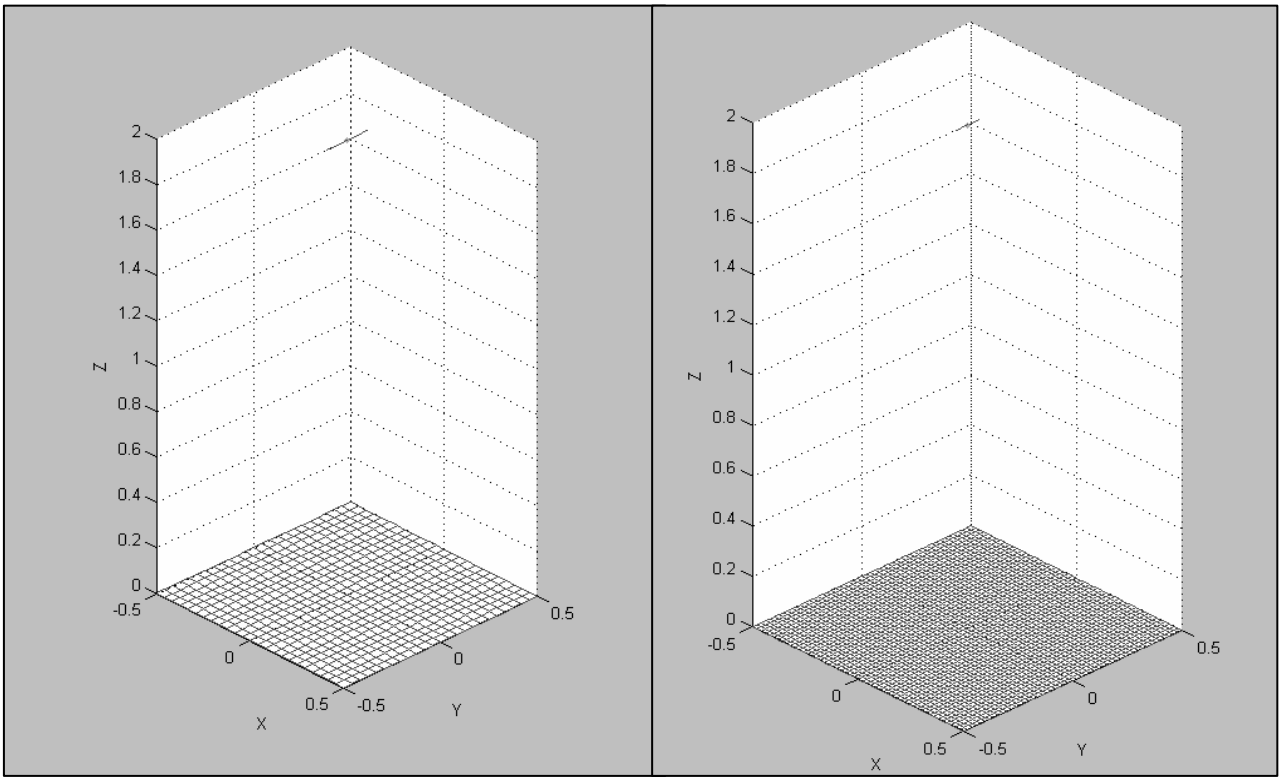


Figure 5: Numerical model of a cube and dipole antenna illustrating small and large problem sizes. 1207 segment to 5106 segments.

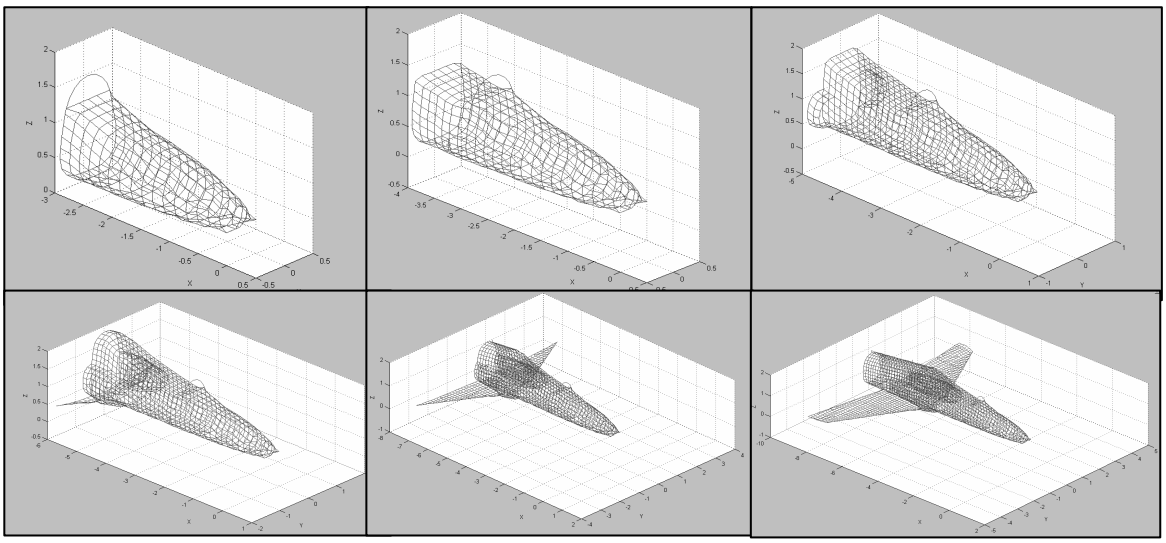


Figure 6: Various sections of the Hawk aircraft ranging from 579 segments to 4977 segments.

The configurations presented above were simulated with the un-optimised SuperNEC engine and an optimised engine for a Pentium 4 machine. The total run time of a problem can be generally broken up into fill time (the time taken to create the matrices) and factorisation time (the time taken to solve the system)

A graph comparing the factorisation times and total times of the Hawk aircraft for various problem sizes is presented in figure 7. Similarly figure 8 and figure 9 present the factorisation times and total times for the optimised and un-optimised SuperNEC engines for the plate and box configurations.

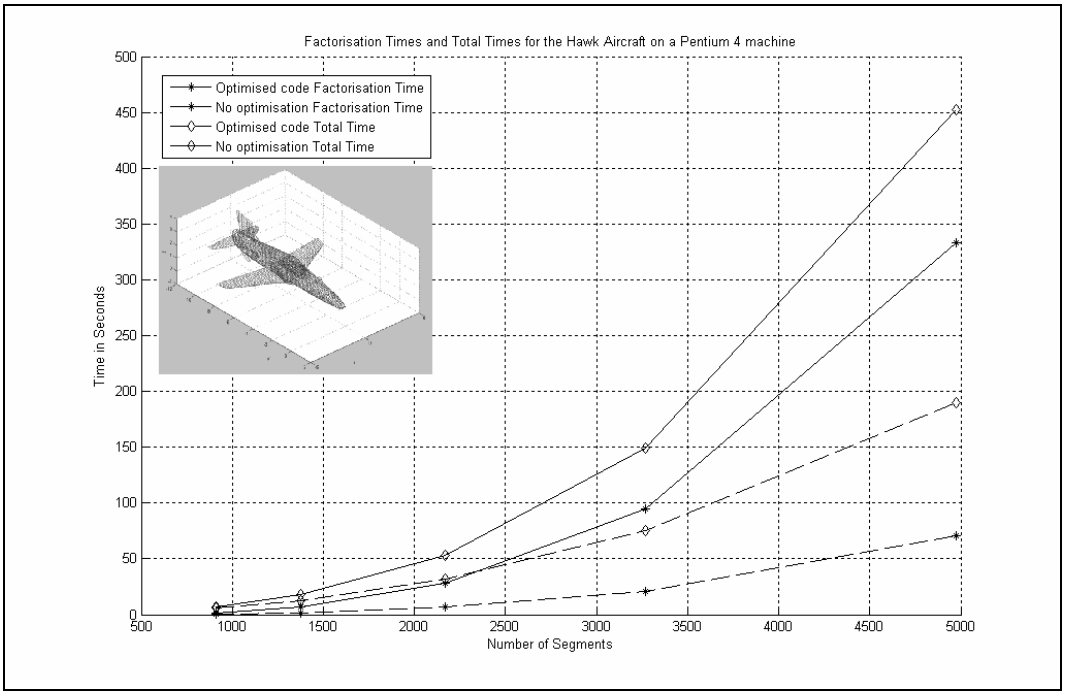


Figure 7: Factorisation and total times for the Hawk aircraft.

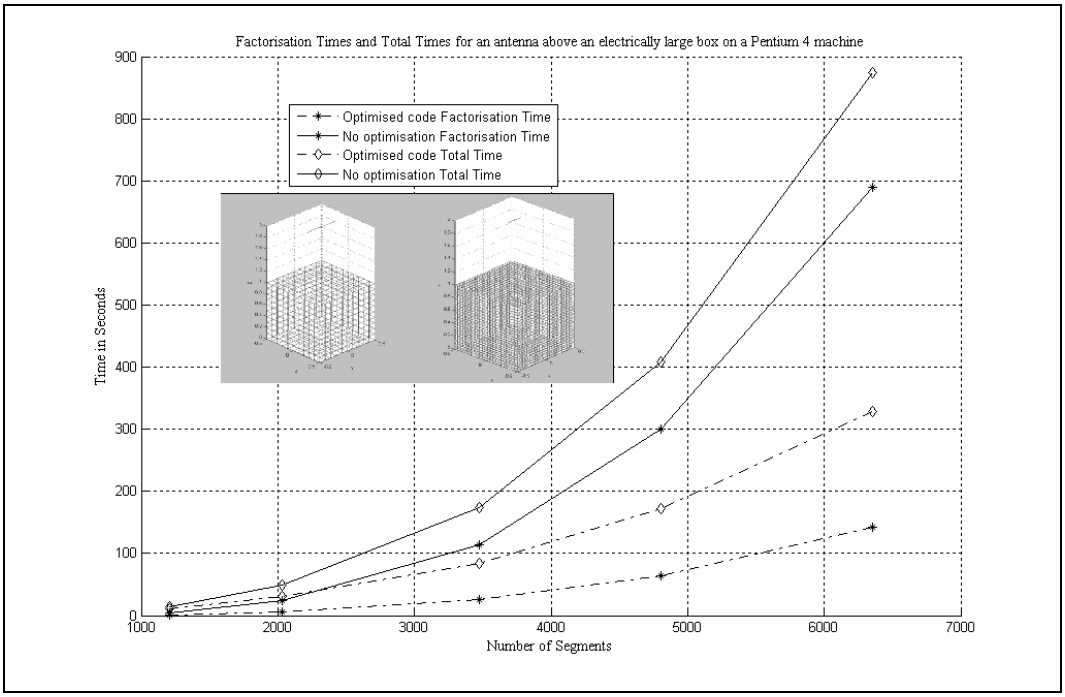


Figure 8: Factorisation and total times for the Box configuration.

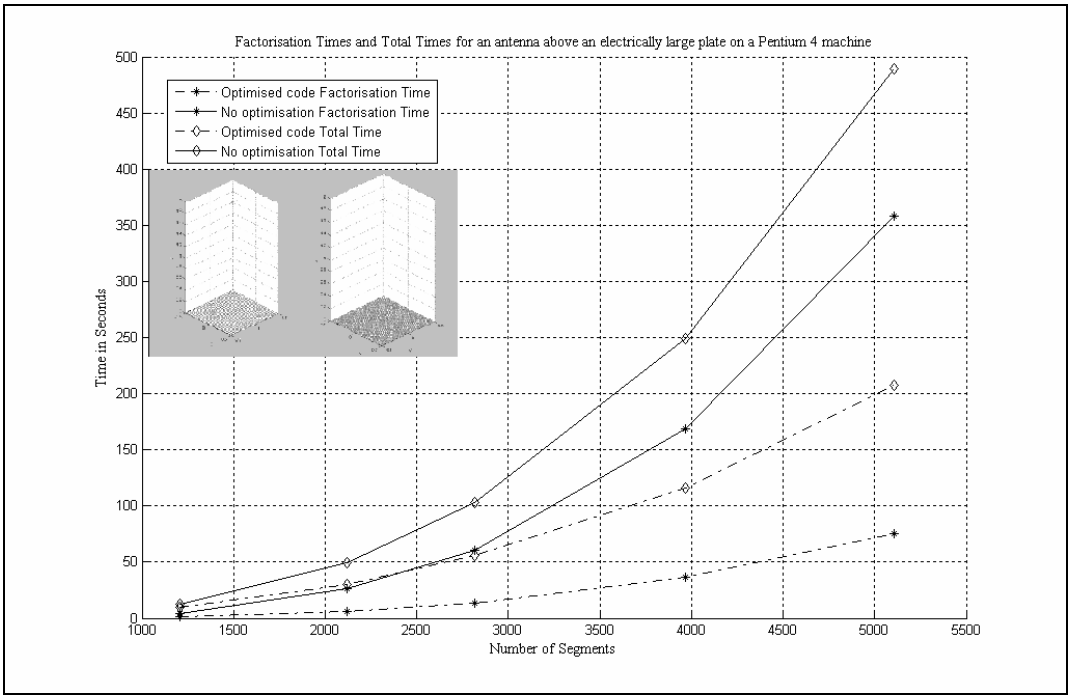


Figure 9: Factorisation and total times for the Plate configuration.

The timing plots above show that the factorisation times are sped up approximately by between 4.1 times to 4.6 times that of the un-optimised engine. Figure 7, 8 and 9 reinforce one another as they show the same improvement despite problem size.

It is interesting to note that problem type has no noticeable effect of the factorisation time, which is expected. This is illustrated by plotting the factorisation times for the various problems studied in figure 10. This is not necessarily the case for iterative methods.

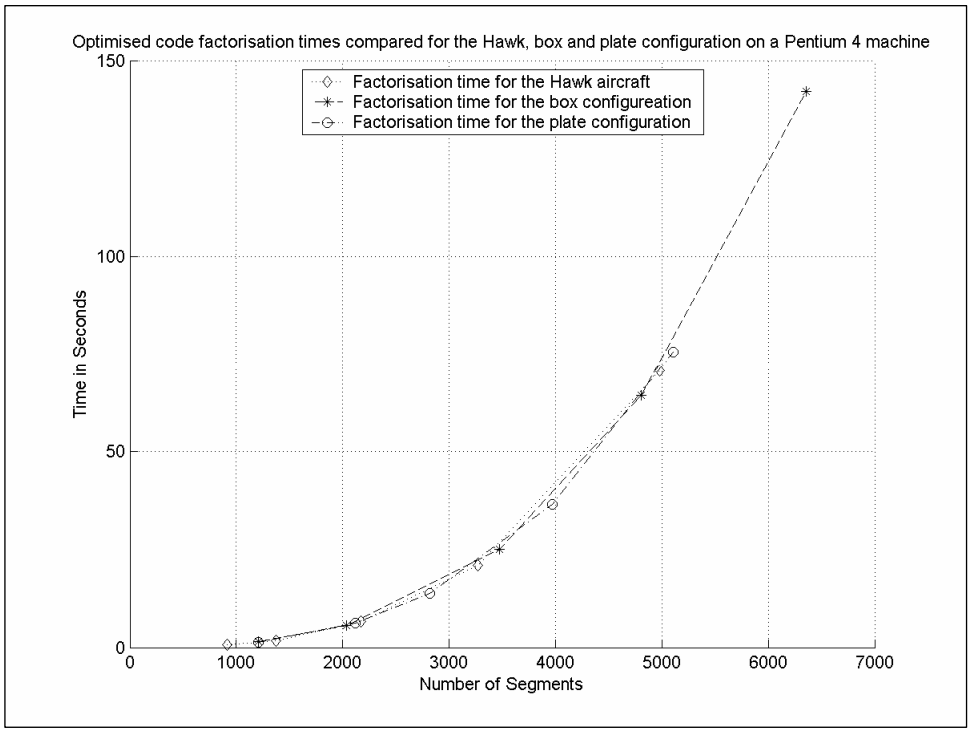


Figure 10: Factorisation times compared for the Hawk, Box and Plate configuration showing problem type has no effect.

3.5. Summary

Due to the rapid improvement of computational hardware, new methods have been developed to create and maintain high performance software by the ATLAS project. ATLAS achieves this by automatically tuning software code through varying parameters and by using programs to write source code. The ATLAS project primarily makes use of code generation to build applications.

The optimisation process can be broken down into the following operations [3, p5]:

- The performance critical routines need to be isolated
- A method of adapting the routines to different environments
- Robust context sensitive timers
- A search mechanism, to search for the optimum method, needs to be put in place

Three complex structures, namely a Hawk aircraft, a dipole above a cube and a dipole above a gridded plate were modelled using SuperNEC for comparison purposes. The comparisons show that the factorisation times are sped up by between 4.1 times to 4.6 times that of the un-optimised engine. This speed up in factorisation time outweighs the overheads and complexity associated with the compilation of the optimised libraries.

By optimising the linear algebra source code used in SuperNEC, a baseline for measurement purposes has been created. Using the optimised linear algebra libraries as a standard library, various methods such as BICGSTAB(L) and Simply Sparse can now be compared more accurately.

4. BICGSTAB(L) method for SuperNEC Problems

4.1. Introduction

Due to the advancement of computational power, computational electromagnetics has over the past few years been exposed to new solution techniques, which otherwise would not have been considered, for large complex problems. The problems are typically of the form

$$[A][x] = [b] \tag{13}$$

in most MoM based packages, where $[A]$ is a non-singular non-symmetric $N \times N$ matrix and $[b]$ is some given N vector. Such systems can be solved by direct methods, such as LU decomposition or by various iterative methods or approximations. One such non-stationary iterative method, employed by SuperNEC, is the Bi-Conjugate Gradient Stabilised (BICGSTAB) algorithm [18, p1463]. BICGSTAB developed from the Conjugate Gradient Method (CG), which was extended to the Bi-Conjugate Gradient Method (BICG) and was later modified to the BICGSTAB method. A modified algorithm to BICGSTAB, BICGSTAB(L) developed by G. Sleijpen et al [2], is used here to solve various electromagnetic problems in SuperNEC and the results are compared with BICGSTAB. All code used makes use of optimised ATLAS libraries as discussed in chapter 3.

Sections 4.2 to 4.7 summarises the work presented in [2]

4.2. BICGSTAB(L) introduced

The BICGSTAB(L) algorithm was developed by Sleijpen, G et al in response to the poor performance of BICGSTAB when the stabilising polynomial's coefficient (ω_k) is close to zero which may cause stagnation. It was found to most likely occur when $[A]$ is real and has non-real eigen values with an imaginary part that is large relative to the real part [2, p12]. The development of BICGSTAB(L) from the conjugate Gradient method to BICGSTAB is summarised.

4.3. Conjugate Gradient Method

The Conjugate Gradient method is one of the oldest and best known non-stationary iterative methods. This method generates successive approximations to the solution (iterations), residuals corresponding to the iterates and search directions to update the iterates [4, p12].

Residuals can be thought of as follows. Suppose you have an equation

$$f(x) = b \tag{14}$$

The a residual would be the result of

$$f(x_0) - b \tag{15}$$

Where x_0 is the approximation. An iterative technique used is close to convergence when the residual is close to zero.

The pseudo code of the CG method is shown below.

$$
\begin{aligned}
&for \;\; i = 1,2,3 \\
&p_{i-1} = r^{(i-1)^T} r^{(i-1)} \\
&if \;\; i = 1 \\
&\quad p^{(1)} = r^{(0)} \\
&else \\
&\quad B_{i-1} = p_{i-1} / p_{i-2} \\
&\quad p^{(i)} = r^{(i-1)} + B_{i-1} p^{(i-1)} \\
&endif \\
&q^{(i)} = Ap^{(i)} \\
&\alpha_i = p_{i-1} / p^{(i)^T} q^{(i)} \\
&x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)} \\
&r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}
\end{aligned}
$$

Where

$$
\begin{aligned}
&x = iterates \\
&r = residuals \\
&p = search\ \ vector \\
&B_i = r^{(i)^T} r^{(i)} / r^{(i-1)^T} r^{(i-1)} \\
&q^{(i)} = Ap^{(i)} \\
&\alpha_i = r^{(i)^T} r^{(i)} / p^{(i)^T} Ap^{(i)}
\end{aligned}
$$

The conjugate gradient method is applicable to symmetric positive definite systems and the speed of convergence depends on the condition number.

4.4. Bi-Conjugate Gradient Method

The following section summarises the Bi-Conjugate Gradient Method as discussed in [4, p19].

Conjugate gradient method is not suitable for non symmetric systems. This is due to the residual vectors which cannot be made orthogonal with short recurrences. The Bi-Conjugate Gradient method replaces the orthogonal sequence of residuals with two mutually orthogonal sequences. This comes with a price of no longer providing minimisation.

The pseudo code in section 4.3 would be altered as follows to implement the BICG method.

Two sequences of residuals would be updated instead of one.

$$r^{(i)} = r^{(i-1)} - \alpha_i A p^{(i)}, \quad \tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i A^T \tilde{p}^{(i)} \tag{16}$$

The two search directions would be

$$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}, \quad \tilde{p}^{(i)} = \tilde{r}^{(i-1)} + \beta_{i-1} \tilde{p}^{(i-1)} \tag{17}$$

Where

$$\alpha_i = \frac{\tilde{r}^{(i-1)^T} r^{(i-1)}}{\tilde{p}^{(i)^T} A p^{(i)}}, \qquad \beta i = \frac{\tilde{r}^{(i)^T} r^{(i)}}{\tilde{r}^{(i-1)^T} r^{(i-1)}}$$

ensuring bi-orthogonal relations.

The Bi-CG method is applicable to non-symmetric matrices. It is not applicable where the matrix is given as an operator and the transpose of the matrix is not available.

4.5. Conjugate Gradient Squared Method (CGS)

This section summarises the work discussed in [4, p21].

In BICG, the residual vector, $r^{(i)}$ can be regarded as the product of $r^{(0)}$ and the ith degree polynomial in A, that is

$$r^{(i)} = P_i(A)r^{(0)} \tag{18}$$

This same polynomial satisfies $\tilde{r}^{(i)} = P_i(A)\tilde{r}^{(0)}$ so that

$$pi = (\tilde{r}^{(i)}, r^{(i)}) = (P_i(A^T)\tilde{r}^{(0)}, P_i(A)r^{(0)}) = (\tilde{r}^{(i)}, P_i^2(A)r^{(0)}) \tag{19}$$

This suggests that if $P_i(A)$ reduces $r^{(0)}$ to a smaller vector $r^{(i)}$, then it might be advantageous to apply this "contraction" operation twice, and compute $P_i^2(A)r^{(0)}$. It also shows that the iteration coefficients can still be recovered from these vectors, and relatively easy to find the corresponding approximations for x.

4.6. Bi-Conjugate Gradient Stabilized Method

This section summarises the work discussed in [4, p24].

BICGSTAB method was developed to solve non-symmetric linear systems while avoiding the irregular convergence patterns of CGS.

The BICGSTAB method can be interpreted as the product id BICG and repeatedly applied GMRES. GMRES (Generalized Minimal Residual) method, which generates a sequence of orthogonal vectors where all previously computed vectors in the orthogonal sequence are retained, is applied to non-symmetric systems. Instead of computing the CGS sequence $i \mapsto P_i^2(A)r^{(0)}$, BICGSTAB computes $i \mapsto Q_i(A)P_i(A)r^{(0)}$ where Q_i is an i^{th} degree polynomial describing the steepest decent update. The full BICGSTAB algorithm is outlined in [4].

The BICGSTAB method will however not converge if the GMRES step stagnates. To avoid this and problems associated with the BICG algorithm, alternative algorithms can be employed. One such algorithm is BICGSTAB(2) developed by Gutknecht [19] and a more general approach is suggested by Sliejpen which leads us on to BICGSTAB(L).

4.7. Bi-Conjugate Gradient Stabilized L Method

This section summarises the work discussed in [2].

The minimum residual polynomial introduced to achieve smoother convergence of BICGSTAB, $1-\omega_k$, to a large extent allows many problems to converge smoothly and often more quickly than BICG and CGS. If ω_k is close to zero, stagnation or breakdown may occur. This is likely to occur if $[A]$ is real and has non real eigen values with the imaginary part much larger than the real part. A second degree minimum residual polynomial is expected to perform better under these circumstances. Gutknecht [19] introduced such a second degree polynomial, BICGSTAB(2). In many cases this improves the algorithm but may still suffer from breakdown in cases where BICGSTAB stagnates or breaks down. Sleijpen introduces a generalisation of BICGSTAB by introducing a polynomial of degree L, hence the name BICGSTAB(L), which minimises near breakdown. Sleijpin suggests that even degree polynomials are well suited for problems with complex eigen pairs. Table 1 is a comparison of the computational and memory costs of the various methods.

Table 1: The average cost per Krylov dimension

METHOD	Computational Cost			Memory Requirement
	MV	AXPY	DOT	
CG	1	3.25	1	7
BICG	2	6.5	2	7
BICGSTAB	1	3	2	7
BICGSTAB2	1	5.5	2.75	10
BICGSTAB(L)	1	0.75(L+3)	0.25(L+7)	2L+5

The computational cost indicates the large vector operations needed to increase the Krylov subspace by one. The memory requirement indicates the maximum number of n-vectors that have to be stored during computation.

MV stands for Matrix Vector Multiply

AXPY stands for $y = ax + y$

DOT stands for Dot Product

4.8. BICGSTAB in SuperNEC

SuperNEC has been used over a number of years to solve large complex EM problems. In an effort to obtain solutions more quickly, various methods have been employed. These methods included iterative methods such as BICGSTAB. Solutions do however not always converge. It was seen that convergence was proportional to the condition number of the impedance matrix being solved. Table 2 illustrates the convergence of various sized problems with BICGSTAB with different condition numbers.

Table 2: Convergence of BICGSTAB for different problems with various condition numbers

Problem	Matrix Size	Condition Number	Number of Iterations
Hawk Aircraft	4977	5.39×10^4	No Convergence
Cube with antenna on top	4806	271	No Convergence
Cube with antenna on top	2032	25.3	57
Cube with antenna on top	590	15	22

Furthermore it was noticed that the matrices that did not converge had eigen values where the imaginary part was large relative to the real part. It is suggested by Sleijpen that this may be the reason why some matrices stagnate, although this typically occurs with matrices where $[A]$ is real. Figure 11 shows that the imaginary component is large compared to the real component of the eigen values for a SuperNEC problem that does not converge where $[A]$ is complex.

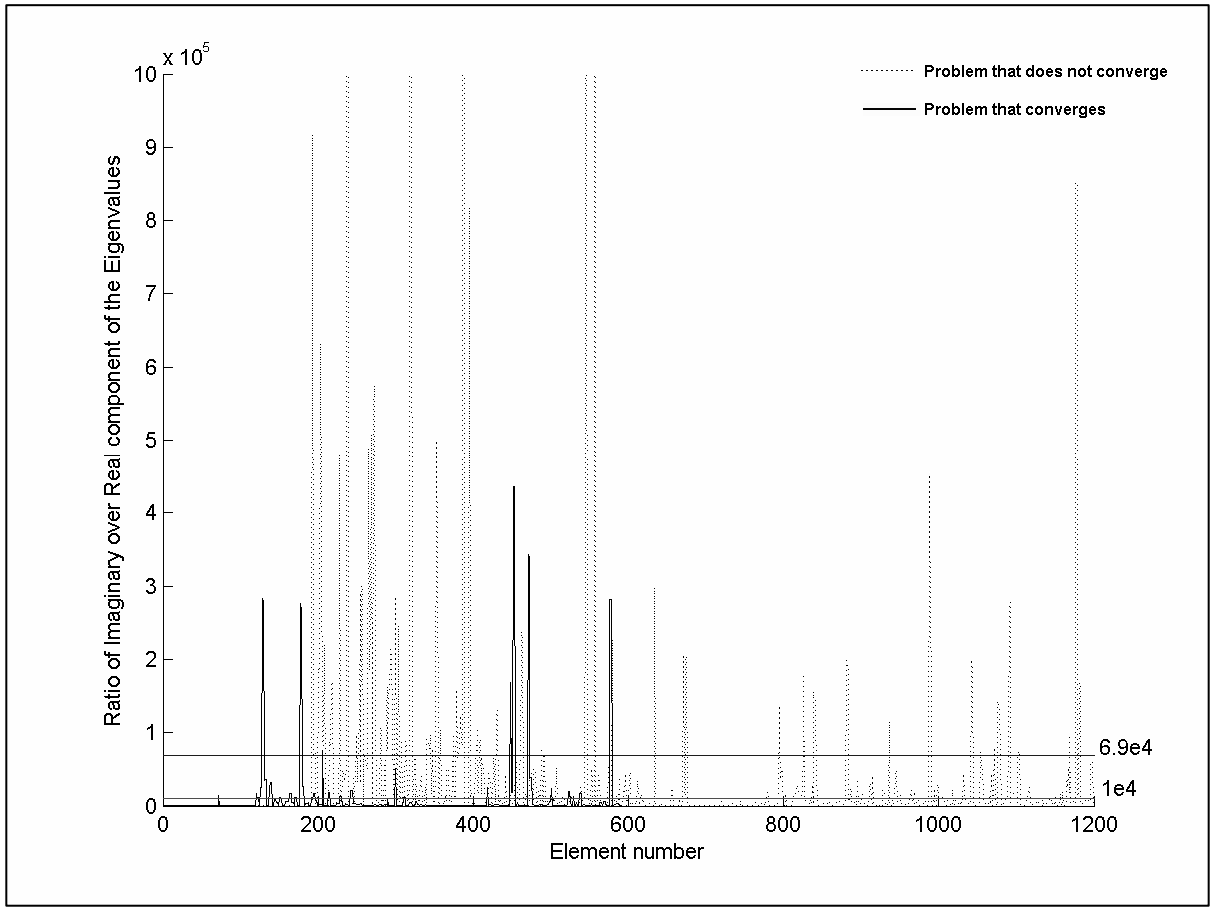


Figure 11: Comparison of SuperNEC problems where the ratio of the imaginary and real eigen values are shown.

In figure 11, the average value of the ratio that does not converge is $6{,}9 \times 10^4$ compared to an average of 1×10^4 for a problem that does converge.

4.9. Results for various Problems in SuperNEC

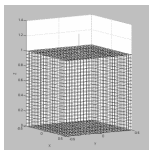
A number of problems with large condition numbers were analyzed using BICGSTAB(L). L was varied from 1 to 5 to investigate which value would generally be best to use with SuperNEC problems, and whether convergence can be achieved.

For comparative purposes, the SuperNEC matrices were solved using the cgstab.m [20] routine in MATLAB on a Pentium 4 machine with 1 Gigabyte of memory.

Table 3 below shows the results of various problem sizes with poor condition numbers.

In general, setting L to 4 (as suggested by Sleijpen) allows these problems to converge, and in some cases more quickly when L is set to any other value, especially with large condition numbers.

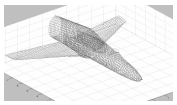
Table 3: Varying L for different size problems with a range of condition numbers



Size	L	Time	Iterations	Condition number
590	1	5.035	22	15.3403
	2	5.17	36	
	3	5.09	30	
	4	5	24	
1203	1	0	Stagnates	90.157
	2	22.1	52	
	3	20.6	60	
	4	20.2	44	
	5	20.9	70	
2032	1	56.9	57	25.2629
	2	56	44	
	3	54	27	
	4	54.5	20	
	5	57.4	65	
3473	1	155.2	23	43.83
	2	169.9	74	
	3	160	51	
	4	164.4	72	
	5	158.4	40	
4806	1	0	Stagnates	271
	2	351.7	184	
	3	364.7	222	
	4	338.3	148	
	5	345.3	150	

Table 4 shows a problem with a large condition number $\left(5.39\times10^{4}\right)$. With L set to 1 the iterative solution stagnates. Setting L to values of 2, 3 and 4 allowed the iterative technique to converge.

Table 4: Varying L for a problem with a large condition number (Hawk Aircraft)



Problem Size	L	Time (s)	Iterations	Condition number
4977	1	0	Stagnates	5.39E+04
	2	310	64	
	3	365	210	
	4	374	236	

It should be noted that the stop criterion for the iterative solutions presented above made use of the default stop criterion used by Sleijpen. That is

$$Norm(A * x - b) < 1e^{-8} \tag{20}$$

Figure 12 shows the various rates of convergence for a problem using BICGSTAB(L) for various values of L. L set to 1 converged the slowest. L set to 3 and 4 converged at a similar rate. It should be noted that L set to 1 is equivalent to BICGSTAB.

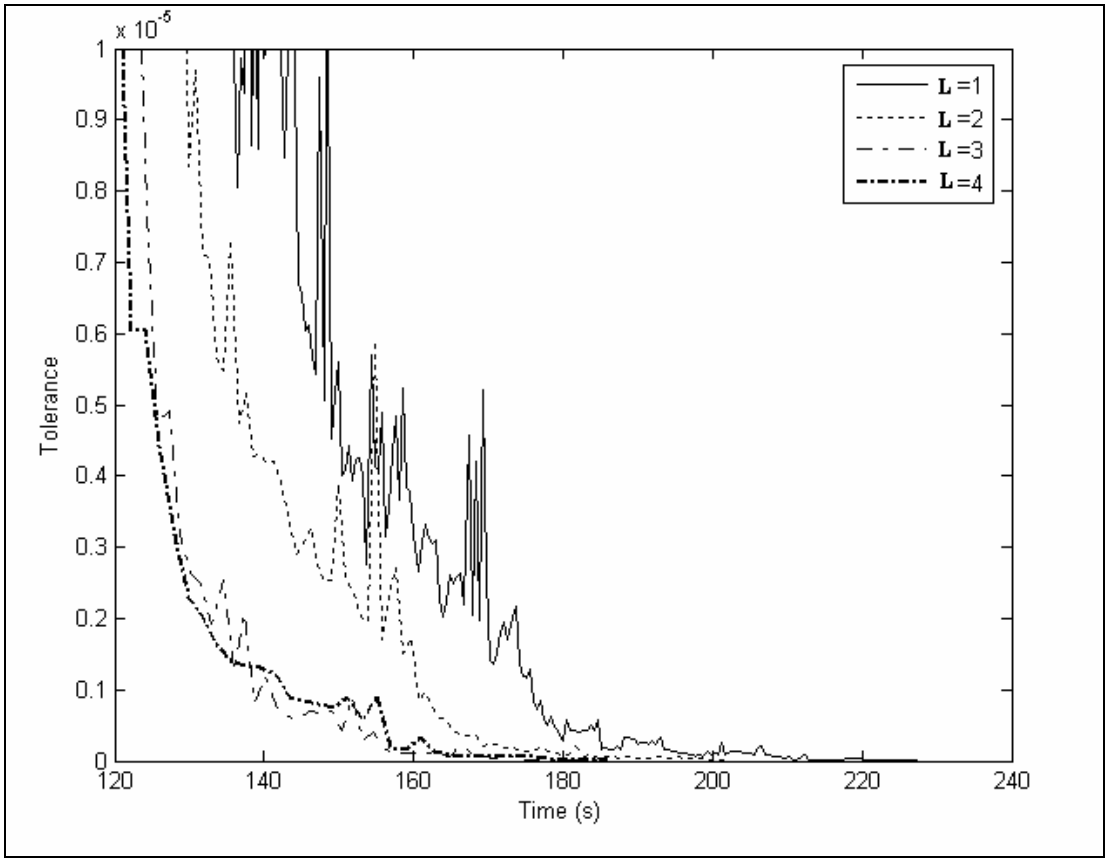


Figure 12: Convergence rates for a typical SuperNEC problem, varying L from 1 to 4 in BICGSTAB(L). The condition number for this problem is 43.

4.10. Chapter Summary

BICGSTAB(L) and its development from CG, BICG, CGS and BICGSTAB has been outlined. BICGSTAB(L) aims to improve the convergence rate of problems solved by the BICGSTAB method despite the properties of the eigen values and coefficient matrix.

Experimental results performed using SuperNEC problems showed that setting L to 2 is sometime more beneficial than setting L to 4, which is confirmed in [2, p13].

In general, however, setting L to 4 as a default value is best. The convergence rate with L set to 4 is in most cases the fastest and usually ensures a solution, even with a high condition number.

BICGSTAB(L) is, however, only a method to solve a system of equations. It does not attempt to look at the actual physical properties of the structure to improve solution times such as domain decomposition methods. The next section looks at such a method, called Simply Sparse.

5. Simply Sparse

5.1. Introduction

Simply Sparse was developed by Dr Francis Canning as a general compression and solution method for MoM Programs [5]. This chapter partly summarizes the paper written and presented at the 2002 IEEE APS symposium. Experimental results of Simply Sparse are also presented.

5.2. Simply Sparse Summarised

Simply Sparse is a new method that transforms the coefficient matrix into a sparse matrix. The sparse matrix that is generated can be inverted with sparse direct methods, such as sparse LU factorization or solved using iterative techniques such as BICGSTAB(L).

Wavelet methods generally require matching the wavelet functions to the wire grid that defines the SuperNEC structure. The Simply Sparse technique, on the other hand, implicitly uses information about the wire grid. The actual details of the grid are not used, but the information used is about how a source, or wire segment, radiates to the far field. This implicitly carries information about the grid and it also carries information about the integral equation and other global properties of the wire grid structure.

An optimization procedure is used to develop a new basis function, providing the optimum amount of sparseness. This optimization is done on separate regions of the wire grid.

If we consider a frequency domain method of moments problem with a wave number k with ten wavelets per wavelength along a one dimensional surface. Approximately two functions per wavelength along the surface will have a centre frequency smaller (in magnitude) than k. The other eight functions will approximately have a frequency greater (in magnitude) than k. These eight functions per wavelength will radiate weakly

[5, p234].

For a two dimensional surface the results are even better. Only π functions per square wavelength radiate strongly while the other 97 functions radiate weakly. Functions that radiate weakly generate small matrix elements when tested at a distant location. The maximum sparseness achievable is approximately a factor 1000.

The Simply Sparse method does better due to its optimization procedure which directly considers the radiated field. In an N by N method of moment matrix numerical results have shown that the typical N^2 storage required for the coefficient matrix, also known as the impedance matrix, is reduced to approximately $1000N$. This results is valid for N greater than one thousand and less than one million.

An example of how Simply Sparse is used in MoM based programs is now shown. If you consider a group of basis functions in a MoM problem, they should come from one physical region, so they are somewhat close together. For M basis functions in this region choose M directions throughout 3 dimensional space (e.g. M points on a sphere) These M directions can be described as M vectors, each going from the centre of the sphere to one of the M points on the sphere. After choosing a point y in the region (say the centre), the functions $F(m,n)$ and $G(m,n)$ are defined. The functions G and F are the two perpendicular components of the electric field where n is the basis functions and m the direction from the centre of the region at a distance of one million wavelengths. Each function gives a complex number.

For $F(m,n)$ and $G(m,n)$ where $m,n=1$ to M each gives a M by M matrix of complex numbers. Concatenating these matrices gives a $2M$ by M matrix$[Z]$. The singular value decomposition gives:

$$[Z]=[W][D][V]^h \tag{21}$$

The singular values are the real valued diagonal elements of D. Matrices W and V are unitary. The matrix V gives a linear transformation of this region with M basis functions to M new basis functions. The new basis functions radiates to the far field with a strength given by the associated value in D.

The values in D are at first roughly the same size before they rapidly decrease in size. The point at which they decrease corresponds roughly to the Nyquist sampling criterion.

This applies to a certain region within the problem. All of the unknowns may now be partitioned into regions in a similar way. For each region a matrix V, as in equation 21, is computed. A block diagonal matrix D^R may be assembled with these matrices V giving its diagonal blocks. Similarly a block diagonal matrix D^L may be assembled for the testing functions. The MoM impedance matrix $[A]$ may be transformed into a new matrix T by:

$$[T] = [D]^L [A][D]^R \tag{22}$$

T is sparse with structure shown in figure 13. Five physical regions give T a five by five block structure. A permutation of this transforms the structure to that shown in figure 14.

It is possible to apply sparse LU factorization, and other techniques, to the sparse matrix as shown in either figure 13 or figure 14.

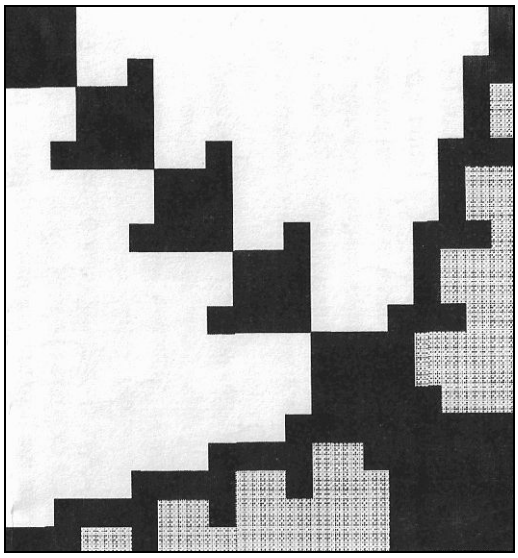


Figure 13: Simple Sparse Matrix T [5, p237]

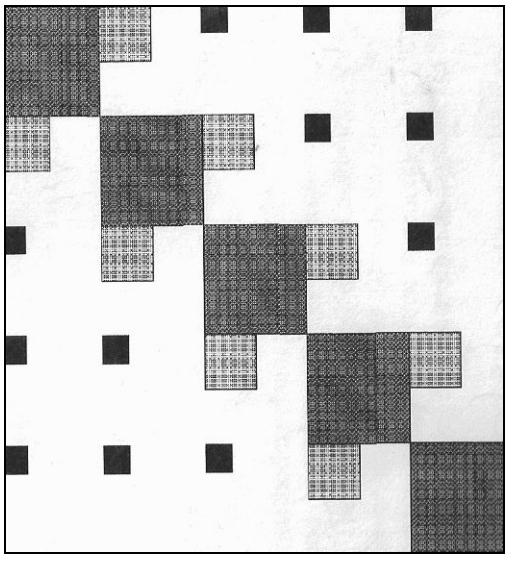


Figure 14: Simply Sparse Matrix T permutation of the row and columns [5, p237]

5.3. Simply Sparse Experimental Results

A number of different size problems were simulated using Simply Sparse, SuperNEC and NEC2 [21]. It was noted that Simply Sparse performed poorly when using the SuperNEC matrix fill routines. This was due to the object orientated nature of SuperNEC fill routine. Every time Simply Sparse requested a portion of the matrix, the request and data had to pass through many code objects, degrading the performance. SuperNEC was however based on the NEC2 code. Where SuperNEC is object orientated, NEC2 is procedural. This provides a better method for comparison, making use of the same theory, but without the object orientated overhead.

The solutions in figure 15 makes use of LU decomposition to solve the matrix, while Simply Sparse makes use of NEC2 to fill the matrix and then compresses it. Simply Sparse uses a custom solver for the solution. Note that these times are total times, which included the overheads to create the matrix structures.

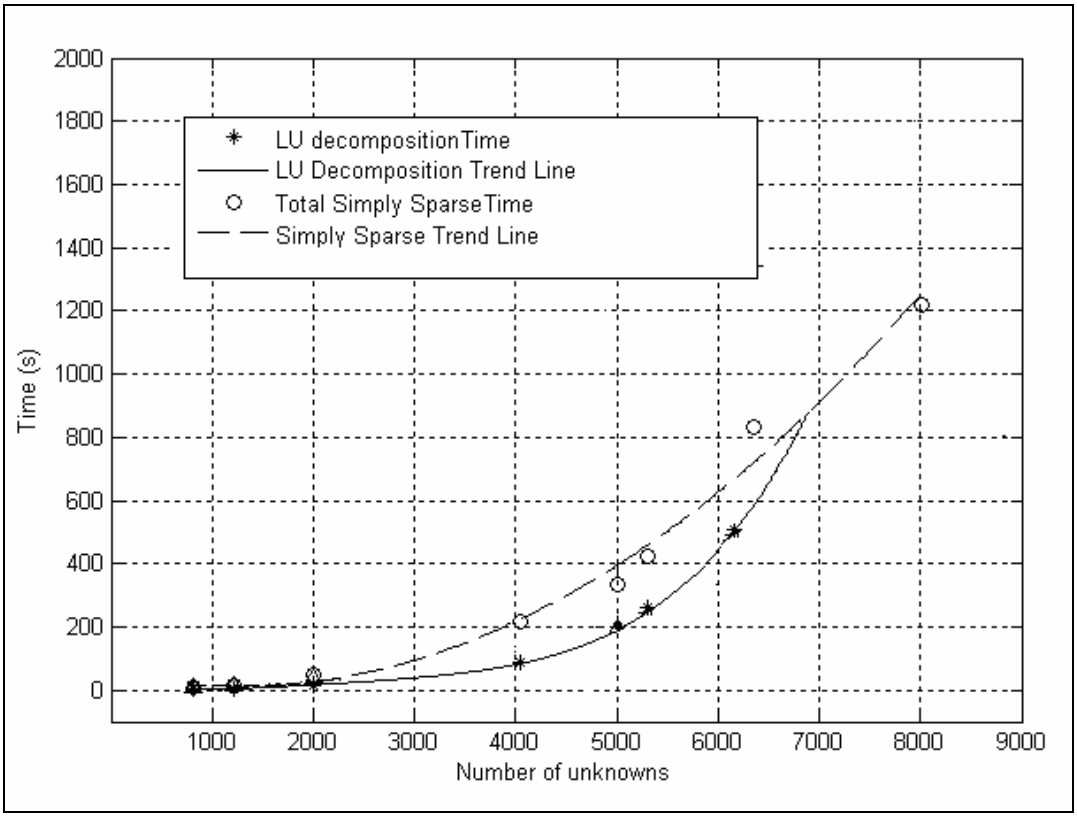


Figure 15: Comparing LU Decomposition and Simply Sparse (807 to 8007 unknowns)

Figure 15 shows that while Simply Sparse scales better as problem sizes increase, standard LU decomposition is still better for problems smaller than 7000 unknowns. Larger problems can also be solved with Simply Sparse as less memory is required.

5.4. Chapter Summary

From the theory and experimental data presented we see that Simply Sparse is a good compression technique for MoM matrices. Simply Sparse also solves large matrix problems more quickly and scales well with increased problem sizes. LU decomposition is still however quicker for problems smaller than 7000 unknowns as the overheads for Simply Sparse dominate.

This leads to the investigation of combining the matrix compression of Simply Sparse and the improved performance of BICGSTAB(L), shown in the previous chapter, to reduce solution times. This is discussed in chapter 6.

6. Modified Simply Sparse Matrix as a Preconditioner

This chapter looks at the development of a new preconditioner to BICGSTAB(L) by modifying the compressed Simply Sparse matrix. A comparison is made between current techniques, such as LU decomposition, Simply Sparse and the new preconditioning technique for the BICGSTAB(L) method. A brief summary of other methods investigated is also presented.

6.1. Various Solution Methods Investigated

A number of techniques were investigated to construct a preconditioner. A brief description of some of the techniques and their investigation is outlined here.

6.1.1. ILU (Incomplete LU factorization)

Chapter 2 gives a description of ILU. Stated again briefly, ILU is based on the incomplete factorisation of the coefficient matrix $[A]$. The incomplete factorisation ignores non-zero fill elements during factorisation when those elements in the original matrix are zero. Tolerance factors can be manipulated to vary which elements are included or excluded.

A preliminary investigation using ILU, using standard suggested tolerances [8, p275], yielded no convergence of any of the MoM problems investigated. Increasing the number of elements in the preconditioning matrix might have yielded some convergence, but as these matrices are relatively dense (more than 2%) their convergence rate would be slower than the existing solution methods in SuperNEC.

ILU, as a solution method, was therefore discounted.

6.1.2. Block Diagonal

Another approach investigated to construct a preconditioning matrix was to include blocks around the diagonal of the matrix. The investigation introduced more and more blocks in the search for a preconditioner. For example figure 16 shows a matrix on the left with 5 blocks around the diagonal included, while the matrix on the right included 20 blocks.

The reason for this approach stems from the construction of the problem to be solved. In these MoM problems, the main diagonal represents the self impedance of the elements and assumes these to be the most important. Matrix elements close to the diagonal should represent wire segments that are physically close to one another and have a strong mutual impedance value. This is however not always the case as, for example, a structure could fold in on itself, producing strong mutual impedance values further away from the diagonal.

Experimental results also showed no convergence using this kind of preconditioning matrix and was hence discounted as a solution technique. Please refer to Appendix D for further experimental data.

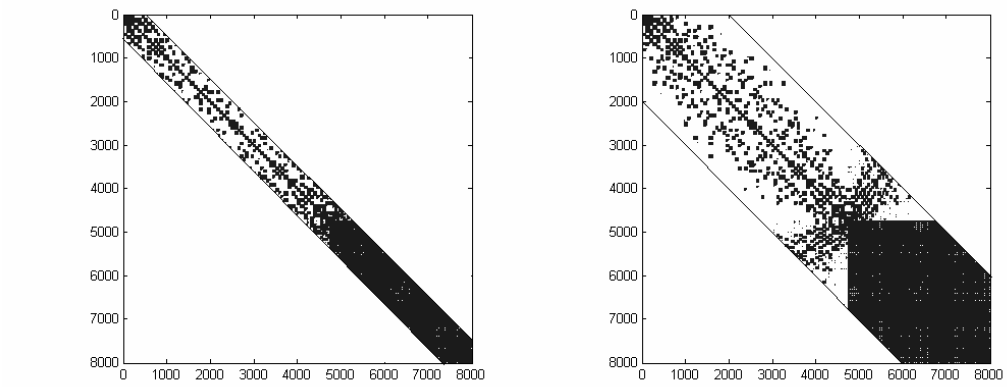


Figure 16: Graphic representation of a Block Diagonal Inclusion Method. 5 Blocks included on the left and 20 blocks on the right

6.1.3. SIM/SOR

SuperNEC includes a solution method called Sparse Iterative Method (SIM) combined with successive over relaxation (SOR). This method makes use of wire segments that are within a certain interaction range, with the hope of including only the most important data.

Investigating this technique showed that the SOR iterations were slow and did not converge. This is in all likelihood due to the high cost of finding an optimal ω. The method allows for a change over to BICGSTAB instead, but yielded either no or slow

convergence with the MoM problems investigated. For further experimental data on interaction distance as a preconditioner please refer to Appendix E.

The work done in [18] does show an improvement over the traditional BICGSTAB method. The poor performance of the BICGSTAB has however been addressed in chapter 4. The best results shown in [18] require 10 iterations. With preconditioner densities varying from 5% to 0.5% the total solution times would not perform better than LU decomposition for small problems and fast solvers like Simply Sparse for large problems.

6.2. Development of a New Preconditioning Matrix for BICGSTAB(L)

A new preconditioning method was developed for BICGSTAB(L) by modifying the compressed Simply Sparse matrix. The thinking behind the approach is described below.

6.2.1. Concept

The compressed Simply Sparse matrix contains all the important information to describe the matrix $[A]$ (or impedance matrix) sufficiently in the EM MoM system.

$$[A][x] = [b] \tag{23}$$

The Simply Sparse matrix then substitutes the full uncompressed impedance matrix $[A]$.

A preconditioner is created by including only the largest elements of the Simply Sparse matrix (defined by the magnitude of the complex number of the element in the matrix). The typical density is between 1% and 2%. As the Simply Sparse matrix is already available no significant effort is needed in creating the preconditioning matrix. The only requirement is that the preconditioning matrix is not singular to within the working accuracy of the software, i.e. it can be inverted.

Making the preconditioning matrix so sparse ensures that small elements are excluded. To emphasise the importance of the remaining elements, they are multiplied by a large constant, 1×10^7 (Not multiplying the preconditioning matrix by this constant results in slow convergence and a more dense preconditioning matrix is needed, i.e. 10%)

Appendix C shows in more detail how the value of the large constant was determined.

Table 5 shows the rate of convergence of a problem, changing the size of the constant multiplier used.

Table 5: Varying the Preconditioning Multiplier for a problem with 807 unknowns (Simulation stopped if iterations were greater than 10)

Multiplying factor	Iterations	Condition number of impedance matrix
100	>10	1380
1000	7	1380
10000	3	1380
100000	2	1380
1000000	2	1380
10000000	2	1380

Once the preconditioner has been constructed, it is used to solve the system of equations using BICGSTAB(L), which has been optimised with the ATLAS libraries.

A graphical representation of various density preconditioning matrices is shown in figure 17. These matrices represent a cube with an antenna above it.

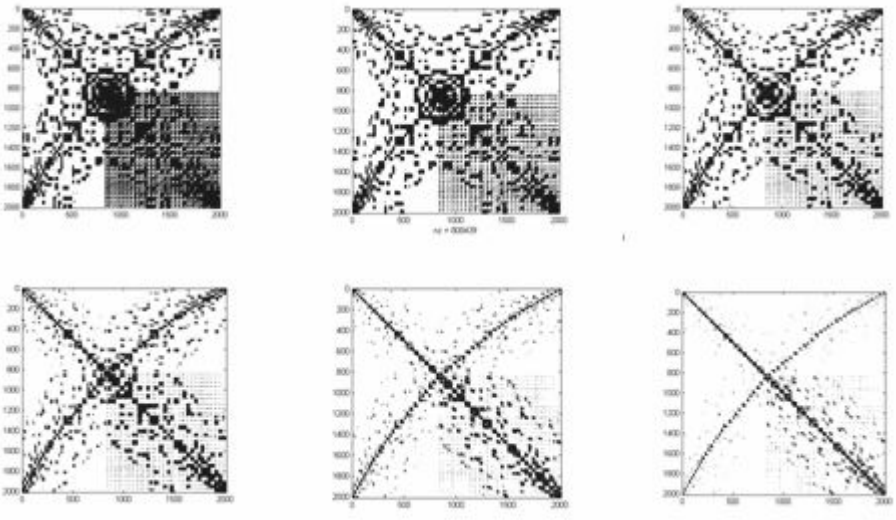


Figure 17: Various density matrices. From top left to bottom right they are 30%, 20%, 10%, 5%, 2% and 1%

6.3. Results

In comparing various solution methods and benchmarking data, the following has to be considered:

- Coding paradigms (i.e. procedural compared to object orientated)
- Matrix creation time
- System solve time

In order to compare algorithms accurately, Matlab was used as a standard platform for LU decomposition and BICGSTAB(L) methods. This ensures that the ATLAS libraries are used and that the same coding paradigm is used. Simply Sparse has a custom solver that is not implemented in Matlab, but also makes use of the ATLAS libraries.

6.3.1. Matrix Creation Times

There are two different methods used to create the various matrices. To create the full uncompressed matrix, the NEC2 code is used. The SuperNEC code is object orientated and suffers from poor performance when creating the matrix. This was seen when using a code profiler how multiple requests for different parts of the matrix were passed through many header structures before performing the task required. This is not seen in the procedural code that NEC2 was written in.

The Simply Sparse code is used to create the compressed matrix.

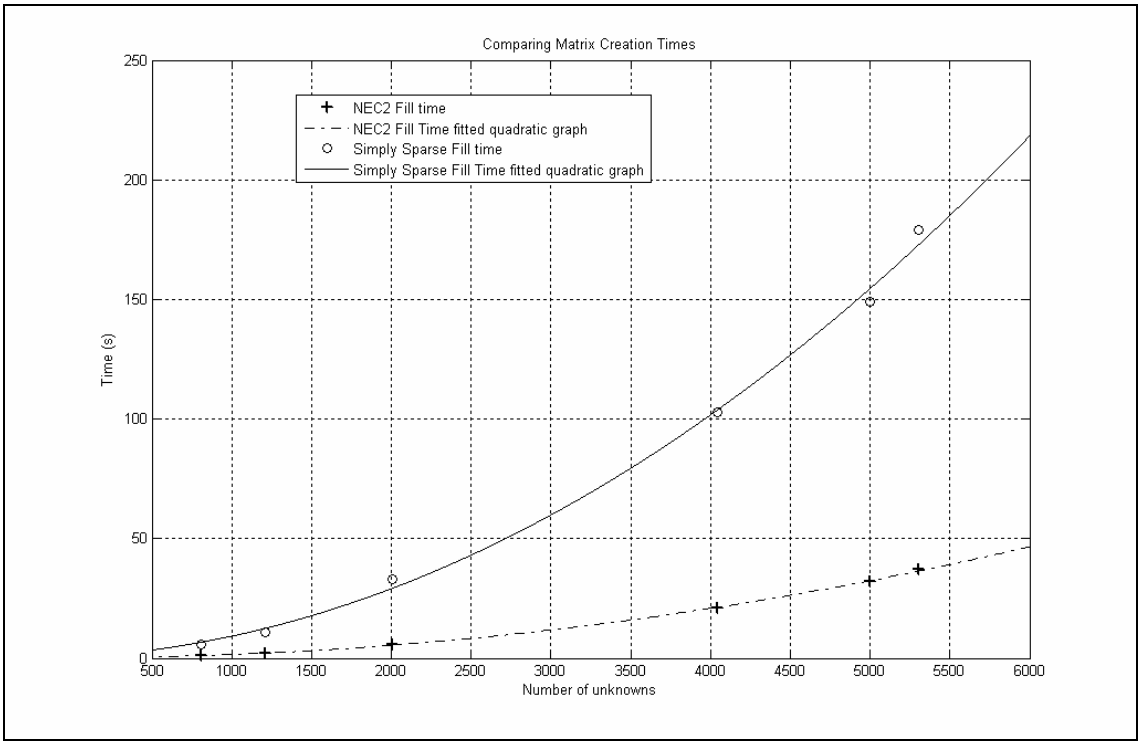


Figure 18: Comparison of Matrix Creation Times

Figure 18 shows that the overheads in creating the Simply Sparse matrix far outweigh that of the comparative NEC2 time. This is done in the hope of reducing the solve time significantly as the solve time dominates the overall solution time with increased problem sizes.

6.3.2. Matrix Solution Times

The solution time of the various techniques are shown in figure 19. The NEC2 code has not been optimised with ATLAS for LU decomposition, but is included here to show the performance improvement achieved by using the ATLAS libraries within Matlab for LU decomposition. It is interesting to note that the custom Simply Sparse solver only outperforms the optimised LU decomposition for problems larger that 5000 unknowns.

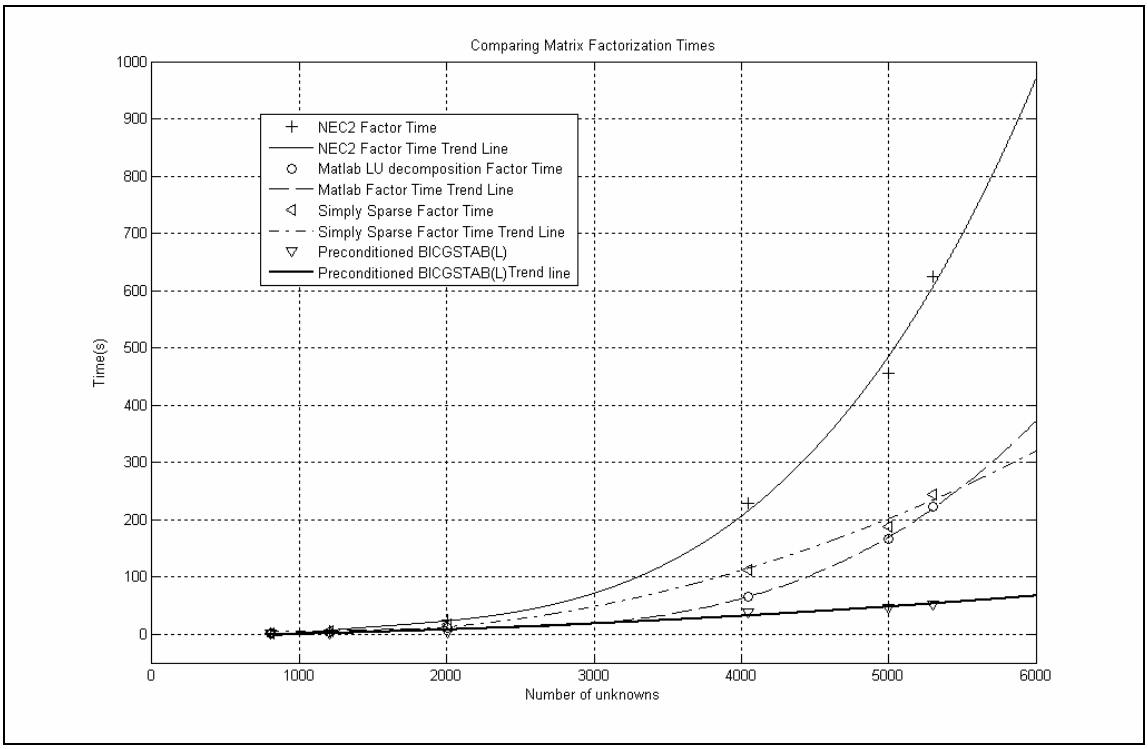


Figure 19: Comparison of Matrix Factorization/Solution times

Table 6: Comparing Preconditioned BICGSTAB(L) with optimised LU decomposition

Problem Size	**807**	**1207**	**2007**	**4043**	**5000**	**5303**	**6355**
LU Decomposition Solve Time (s)	1.07	2.53	11.08	66.06	166.6	258.9	Memory limit reached
Preconditioning Solve Time (s)	1.21	1.09	2.8	37.5	46.19	52.2	148.51
Iterations	2	2	2	3	2	2	2
Residual	3.67E-14	1.84E-09	1.75E-09	1.87E-08	9.66E-16	6.19E-09	6.36E-09
%	2.63	1.32	1.03	0.76	1.06	0.99	0.97

Table 6 shows more clearly the time taken to solve smaller problems. The preconditioning technique is at best just as quick as LU decomposition for smaller problems. This is expected as the overheads for solving smaller problems with BICGSTAB(L) dominates the solve time. It should be noted that the odd problem sizes are due to the use of artificial (cube and plate structures) and real world problems (Hawk aircraft).

6.3.3. Total Times

In order to compare solution techniques, the matrix creation time and solution time needs to be considered. This is shown in figure 20.

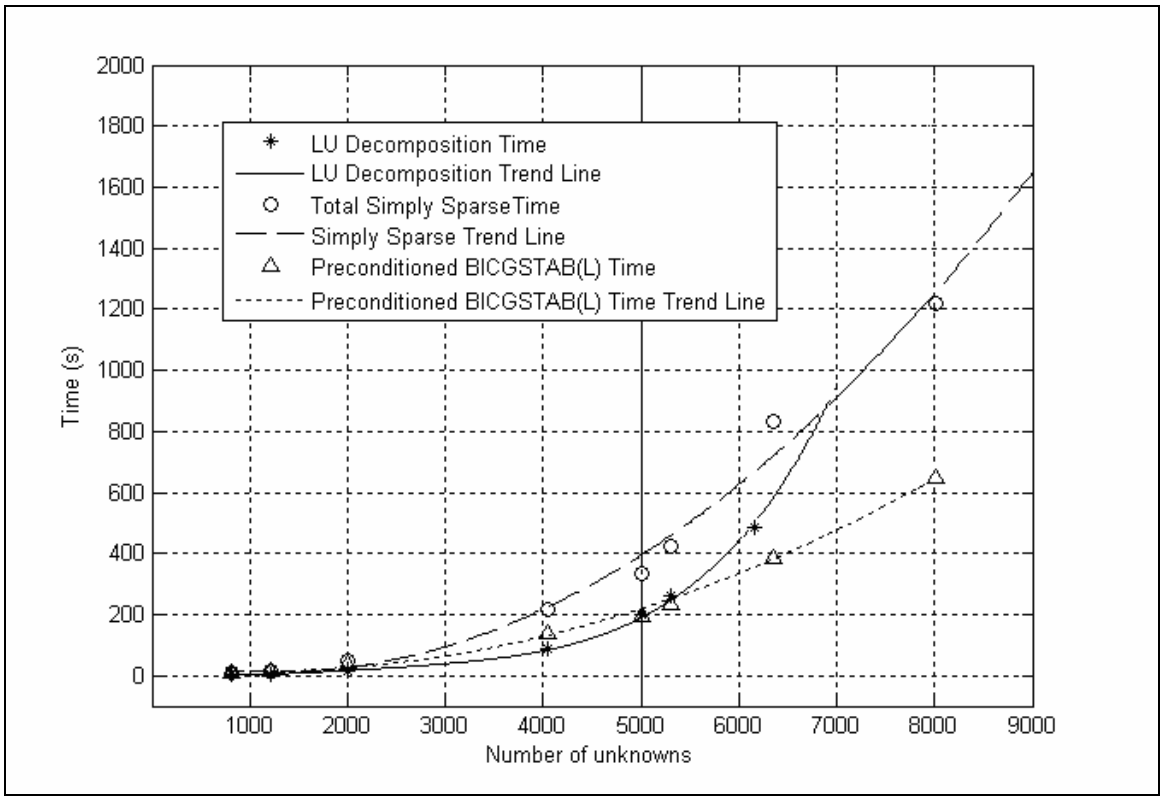


Figure 20: Comparing Total Run Times (i.e. Matrix Creation Time + Solution Time)

The stop criterion for the preconditioned BICGSTAB(L) method is for the residual to be less than 10^{-8}. The residual is defined as:

$$Norm([A][x]-[b]) < 10^{-8} \tag{24}$$

Figure 20 shows again the advantage of working with sparse matrices, as larger problems can be solved compared to full matrix inversion of LU decomposition.

The overheads in creating the matrix for both the BICGSTAB(L) preconditioning technique and the Simply Sparse technique make them impractical to solve small problems. The Simply Sparse technique only outperforms LU decomposition for problems larger than 7000 unknowns. The new preconditioning technique, however, outperforms both LU decomposition and the Simply Sparse technique for problems larger than 5000 unknowns.

Figure 21 shows the typical absolute error achieved with the preconditioning technique. Absolute error is defined as:

$$|xapprox - x| \tag{25}$$

For this example, it means that the current on each wire has an average error less than 0.5×10^{-3}. A more comprehensive set of plots can be found in Appendix B.

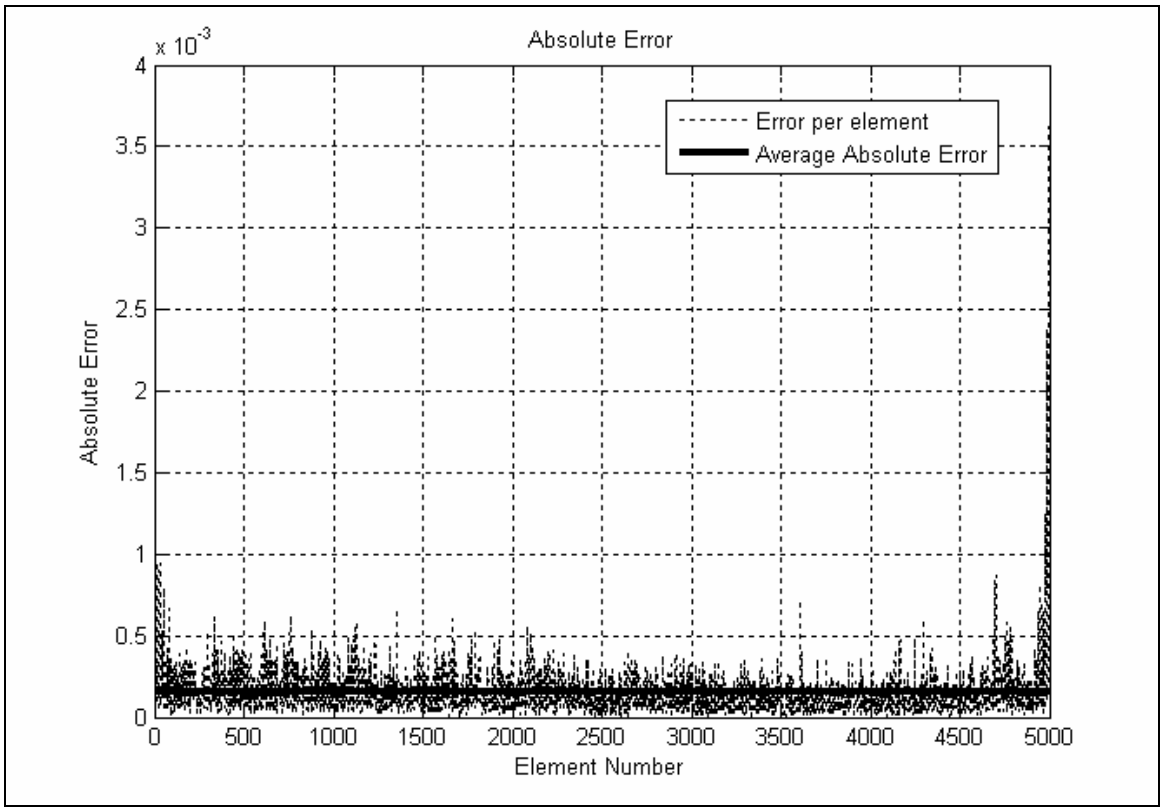


Figure 21: Absolute Error for a Problem with 5000 Unknowns

6.4. Chapter Summary

A new preconditioning technique has been developed that significantly reduces the number of iterations for BIGSTAB(L) to 2 or 3 iterations for SuperNEC problems. This is achieved regardless of the condition number of the coefficient matrix and hence problem type. The preconditioner is created by modifying the compressed Simply Sparse matrix to create a very sparse (between 1% and 2%) matrix and multiplying it by 10^7. The new technique has been shown to be quicker than LU decomposition and the custom Simply Sparse method for problems larger than 5000 unknowns.

7. Conclusion

The aim of this research was to improve the matrix solutions methods for EM MoM problems in SuperNEC, and develop a new preconditioner for the iterative method BICGSTAB(L) using the domain decomposition method, Simply Sparse, to construct the matrices.

This was achieved by:

- Investigating and improving the LU decomposition subroutines used by implementing the Automatically Tuned Linear Algebra Subprogram Software (ATLAS). The libraries optimised using ATLAS were used as a baseline in all other methods investigated.
- Investigating and improving on the current iterative method BICGSTAB by making use of a modified method BICGSTAB(L).
- Characterising the domain decomposition method Simply Sparse.
- Developing a new preconditioner for BICGSTAB(L) by modifying and manipulating the Simply Sparse matrices, where the code had been optimised with ATLAS.

7.1. ATLAS

The ATLAS code primarily makes use of code generation to build and optimise applications.
Three complex structures, namely a Hawk aircraft, a dipole above a cube and a dipole above a meshed plate were modelled using SuperNEC for comparison purposes. A range of problems with different matrix condition numbers was intentionally chosen to address poor solution times due to problem type. The comparisons show that the matrix solve times using LU decomposition optimised by ATLAS are improved by between 4.1 times to 4.6 times compared to that of the un-optimised code. This speed up in factorisation time outweighs the overheads and complexity associated with the compilation of the optimized libraries, which need to be done only once, taking hours instead of weeks of optimising code by hand

7.2. BICGSTAB(L)

The BICGSTAB(L) method was characterised using a number of problems while using the ATLAS libraries to optimise the code for the computer hardware it was run on. It was determined that in general setting the L parameter to 4 helps problems converge more quickly than setting it to 2 or 3, even with high condition numbers. Setting L to 1 reduces the code back to the original BICGSTAB. This is an improvement over the standard BICGSTAB method where problems do not always converge when the matrix condition number is high. BICGSTAB(L) is, however, only a solution method and does not take any advantage of the physical properties of the problems as a domain decomposition method might.

7.3. Simply Sparse

The domain decomposition method, Simply Sparse, was outlined, showing how the impedance matrix can be constructed, making use of the physical properties of the problem. The resulting matrices are also sparse matrices, containing only the essential information within the matrix to describe the problem.

From the experimental data presented it was shown that Simply Sparse is a good compression technique for SuperNEC MoM matrices. The custom Simply Sparse solver also solves large matrix problem more quickly than LU decomposition and scales well with increased problem sizes. LU decomposition is still however quicker for small problems as the overheads in compressing the matrices for Simply Sparse dominate.

7.4. Original Contribution (Preconditioning Technique)

A new preconditioning technique has been developed by making use of the compression techniques of Simply Sparse, the solution method of BICGSTAB(L) and the optimised BLAS libraries created by ATLAS.

The preconditioning technique looks to use the best properties of each of the methods presented and combine them in a unique way after applying a modification. BICGSTAB(L) was chosen as the solution method, as problems converge well despite the condition number of the matrices. L was set to 4 for optimal convergence as shown in

the characterisation of BICGSTAB(L). Simply Sparse was used to create a compressed impedance matrix using the physical properties of the problem.

The preconditioning matrix was constructed by setting to zero up to 99% of the largest elements within the Simply Sparse impedance matrix. Using just this matrix as the preconditioner did aid some problems in converging but not quicker than any other method. The preconditioning matrix was then modified to considerably improve the rate of convergence. This is done by multiplying the remaining elements by 10^7.

Other preconditioning methods, such as including more blocks around the diagonal, incomplete LU decomposition and SIM/SOR were investigated but with poor results, resulting in those methods being discarded.

Experimental results using the new preconditioner with BICGSTAB(L) and the Simply Sparse impedance matrix show that although the solve time for the new method is fast for even small problems, the overheads of creating the sparse matrices with Simply Sparse dominates. The new method is however quicker than both LU decomposition and the custom Simply Sparse solution method for Problems larger than 5000 unknowns. Problems typically converge within 2 or 3 iterations.

7.5. Recommendation for Further Investigation

Although this work looks at the most commonly used solution methods such as LU decomposition, BICGSTAB(L) and a domain decomposition technique, there are a large number of techniques that need to be characterised and compared to the new preconditioning method that was developed.

A preliminary investigation into ILU did not yield promising results. Further investigation into varying the many parameters of this method may yield better results.

Work also needs to be conducted to make the preconditioning matrix even more sparse by using greater precision methods. Increasing the precision might however also have a negative effect of increasing the solution time.

Further work should also be conducted to try and find another matrix compression method that does not have the overheads associated with Simply Sparse. The impedance matrix of SIM would be such a candidate. Reducing the overheads of constructing the impedance matrix would improve the overall speed of the preconditioning technique.

References

[1] Nitch D.C, “A serial and parallel design of NEC2 to demonstrate the advantages of the object-oriented paradigm in comparison with the procedural paradigm” PhD thesis, University of the Witwatersrand, Johannesburg, December 1992.

[2] Sleijpen G, (1993), “BICGSTAB(L) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum”, Electronic Transactions on Numerical Analysis Volume 1. pp. 11-32

[3] Whaley R, Petitet A, Dongarra J, (2000), “Automatic Empirical Optimization of Software and the ATLAS project” published electronically http://www.cs.utsa.edu/~whaley/papers/lawn147.ps

[4] Barrett R, Berry M, Chan T, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, Van der Vorst H, “Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods”, Society for Industrial and Applied Maths (SIAM) ISBN 0-89871-328-5, 1993. Also available: http://www.netlib.org/linalg/html_templates/Templates.html

[5] Canning F.X , Rogovin K, (2002),” Simply Sparse, A general Compression/Solution Method for MoM Programs.”, Antennas and Propagation Society International Symposium, 2002. IEEE Volume 2, Issue , pp 234 - 237

[6] Fourie A.P.C, Nitch D.C, “A fast Sparse Iterative Method (SIM) for Method of Moments” Antennas and Propagation Society International Symposium, 1994. AP-S. Digest**,** Volume 2, pp 1146-1149.

[7] The Mathworks, Inc., “Getting Started with Matlab”, http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.shtml

[8] Saad Y, (2000), “Iterative Methods for Sparse Linear Systems”, Second Edition, published electronically http://www-users.cs.umn.edu/~saad/books.html

[9] “SuperNEC MoM technical Reference Manual”, Version 2.9, Poynting Software (Pty) Ltd, South Africa.

[10] Harrington R.F, “Field Computation by Moment Methods” Macmillan, New York, 1968

[11] “SuperNEC GUI Input User Reference Manual”, Version 2.9, Poynting Software (Pty) Ltd, South Africa.

[12] Kirby R.C , “A new look at expression templates for matrix computation”, Computing in Science & Engineering, May-June 2003, Volume5, Issue 3, pp 66-70.

[13] Weiland T, “TBCI and URMEL - new Computer Codes for Wake Field and Cavity Mode Calculations”, IEEE Transactions on Nuclear Science, August 1983, Volume 30, Issue 4, pp 2489-2491

[14] Dongarra J, Du Croz J, Duff I, Hammarling S, ”A Ser of level 3 Basic Linear Algebra Subprograms”, ACM Transactions on Mathematical Software1990, pp 1-17

[15] Dongarra J, Du Croz J, Hammarling S, Hanson R, “Algorithm 656: An extended Set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs” ACM Transactions on Mathematical Software1998, pp18-32

[16] Dongarra J, Du Croz J, Hammarling S, Hanson R, “An Extended Set of FORTRAN Basic Linear Algebra Subprograms”, ACM Transactions on Mathematical Software1998, pp1-17

[17] Anderson E., Bai Z, Bischof C, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S, Sorensen D, “LAPACK Users’ Guide, Second Edition”, SIAM, Philadelphia, PA, 1995

[18] Clark A.R, Fourie A.P.C, Nitch D.C, “Stationary, non-stationary and hybrid iterative Method of Moments solutions schemes”, IEEE Transactions on Antennas and propagation, Vol. 49, No. 10, October 2001, pp 1462-1469

[19] Gutknecht M.H, “Variants of BICGSTAB for matrices with complex spectrum”, SIAM J. Sci. Compute September 1993, Vol 14, pp. 1020-1033

[20] Code obtained from http://www.math.uu.nl/people/sleijpen/CGSTAB_software/cgstab.m.gz

[21] Burke G.J, Poggio A.J, “Numerical Electromagnetic Code (NEC) –Method of Moments” tech doc 116, Naval Oceans Systems Centre, San Diego, CA, 1981

[22] Poggio A.J and Adams R.W, Approximations for Terms Related to the Kernel in Thin-Wire Integral Equations, UCRL-51985, Lawrence Livermore Laboratory, CA, December 19, 1975

[23] Harrington R.F, Field Computation by Moment Methods, MacMillan, New York, 1968

[24] Yeh Y.S and Mei K.K, “Theory of Conical Equiangular Spiral Antennas,” Part I - Numerical Techniques, IEEE Trans. Ant. And Prop., AP-15, 1967, p.634

[25] Neureuther A.R et al., “A Comparison of Numerical Methods for Thin Wire Antennas,” Presented at the 1968 Fall URSI Meeting, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1968

[26] Miller E.K et al., An Evaluation of Computer Program Using Integral Equations for the Electromagnetic Analysis of Thin Wire Structures, UCRL-75566, Lawrence Livermore Laboratory, CA, March 1974

[27] Wu T.T and King R.W.P “The Tapered Antenna and Its Application to the Junction Problem for Thin Wires,” IEEE Trans. Ant. And Prop., AP-24, No.1, pp.42-45, January 1976

[28] Poggio, A.J, Integral Representation for Fields Due to Sources on Open Surfaces with Applications to End Caps, UCRL-51723, Lawrence Livermore Laboratory, CA, December 16, 1974

[29] Ralston A, A First Course in Numerical Analysis, McGraw-Hill, New York, 1965

Appendix A: SuperNEC Theory

This appendix is an excerpt from the SuperNEC user reference manual [9].

It serves to provide the reader with information on how the impedance matrix is constructed, although not necessary for the understanding of the work presented in this thesis.

A1. Introduction

SuperNEC (SNEC) is an object-oriented version of the FORTRAN program NEC-2. This technical manual is therefore very similar to the NEC-2 manual. In fact, many sections have been copied verbatim from the NEC-2 manual. The reason this document has been produced (as opposed to referring the reader to any readily available NEC-2 manual) is because new theory will be added to SNEC and some features that exist in NEC-2 will not be implemented in SNEC. The easiest way of keeping an up to date reference of the theory implemented in SNEC is to start with an electronic version of the NEC-2 manual and modify the document as the SNEC code evolves. Many extensions have been made to SNEC in prototype form. These features include hybridisation with UTD, fast iterative solvers, MBPE amongst other features. When these extensions are formally incorporated into the SNEC program, then the theory behind the extension will be added to this manual.

A2. The Integral Equation for Free Space

The SNEC program uses an electric-field integral equation (EFIE) to model the electromagnetic response of general structures. The EFIE can be used to model both wire structure and surfaces. Surfaces are represented by wire grids and have had reasonable success for far-field quantities but with variable accuracy for surface fields. The EFIE and its derivation are outlined in the following sections.

A3. The electric field integral equation (EFIE)

The form of the EFIE used in NEC follows from an integral representation for the electric field of a volume current distribution $\mathbf{J}$.

$$\mathbf{E}(\mathbf{r}) = \frac{-j\eta}{4\pi k}\int_{v} \mathbf{J}(\mathbf{r}') \bullet \overline{\overline{G}}(\mathbf{r},\mathbf{r}')dV' \qquad \text{(A1)}$$

where

$$\overline{\overline{G}}(\mathbf{r},\mathbf{r}') = (k^2\bar{I} + \nabla\nabla)\, g(r,r')$$

$$g(\mathbf{r},\mathbf{r}') = e^{\frac{-jk|\mathbf{r}-\mathbf{r}'|}{|\mathbf{r}-\mathbf{r}'|}}$$

$$k = \omega\sqrt{\mu_0\varepsilon_0}$$

$$\eta = \sqrt{\frac{\mu_0}{\varepsilon_0}}$$

and the time convention is $e^{j\omega t}$.

$\bar{\mathrm{I}}$ is the identity dyad $(\hat{x}\hat{x} + \hat{y}\hat{y} + \hat{z}\hat{z})$. When the current distribution is limited to the surface of a perfectly conducting body, equation (A1) becomes:

$$\mathbf{E}(\mathbf{r}) = \frac{-j\eta}{4\pi k}\int_S \mathbf{J}_s(\mathbf{r}') \bullet \overline{\overline{G}}(\mathbf{r},\mathbf{r}')dA' \qquad \text{(A2)}$$

where

$\mathbf{J}_s$ is the surface current density.

The observation point $\mathbf{r}$ is restricted to be off the surface S so that $\mathbf{r} \neq \mathbf{r}'$.

If $\mathbf{r}$ approaches S as a limit, equation (A2) becomes:

$$\mathbf{E}(\mathbf{r}) = \frac{-j\eta}{4\pi k}\int_S \mathbf{J}_s(\mathbf{r}') \bullet \overline{\overline{G}}(\mathbf{r},\mathbf{r}')dA' \qquad \text{(A3)}$$

where the $\int_S$ now represents a principal value integral. It is a principle value integral since $g(\mathbf{r},\mathbf{r}')$ is now unbounded.

An integral equation for the current induced on S by an incident field $\mathbf{E}$ can be obtained from equation (A3) and the boundary condition for $r \in S$:

$$\hat{n}(\mathbf{r}) \times \left[\mathbf{E}^s(\mathbf{r}) + \mathbf{E}^I(\mathbf{r})\right] = 0 \qquad \text{(A4)}$$

where

$\hat{n}(\mathbf{r})$ is the unit normal vector of the surface at $\mathbf{r}$

$\mathbf{E}^s$ is the field due to the induced current $\mathbf{J}_S$.

Substituting equation (A3) for $\mathbf{E}^s$ yields the integral equation:

$$-\hat{n}(\mathbf{r}) \times \mathbf{E}^I(\mathbf{r}) = \frac{-j\eta}{4\pi k}\hat{n}(\mathbf{r}) \times \int_S \mathbf{J}_s(\mathbf{r}) \bullet (k^2\bar{\mathrm{I}} + \nabla\nabla)g(\mathbf{r},\mathbf{r}')dA' \qquad \text{(A5)}$$

The vector integral in equation (A5) can be reduced to a scalar integral equation when the conducting surface S is that of a cylindrical thin wire, thereby making the solution much easier. The assumptions applied for a thin wire, known as the thin-wire approximation, are as follows:

1. Transverse currents can be neglected relative to axial currents on the wire.
2. The circumferential variation in the axial current can be neglected.
3. The current can be represented by a filament on the wire axis.

4. The boundary condition on the electric field need be enforced in the axial direction only.

These widely used approximations are valid as long as the wire radius is much less than the wavelength and much less than the wire length. An alternate kernel for the EFIE, based on an extended thin-wire approximation in which condition 3 is relaxed, is also included in NEC for wires having too large a radius for the thin-wire approximation[22]. From assumptions 1, 2 and 3, the surface current $\mathbf{J}_S$ on a wire of radius can be replaced by a filamentary current I where

$$I(s)\hat{s} = 2\pi\mathbf{J}_s(\mathbf{r})$$

where

s = distance parameter along the wire axis at $\mathbf{r}$

$\hat{s}$ = unit vector tangent to the wire axis at $\mathbf{r}$

Equation (A5) then becomes

$$-\hat{n}(\mathbf{r})\times\mathbf{E}^I(\mathbf{r}) = \frac{-j\eta}{4\pi k}\hat{n}(\mathbf{r})\times\int_S \mathbf{I}(s')(k^2\hat{\mathrm{s}}'-\nabla\frac{\partial}{\partial s'})g(\mathbf{r},\mathbf{r}')ds' \qquad \text{(A6)}$$

where the integration is over the length of the wire. Enforcing the boundary condition in the axial direction reduces equation (A6) to the scalar equation,

$$-\hat{s}\bullet\mathbf{E}^I(\mathbf{r}) = \frac{-j\eta}{4\pi k}\int_S \mathbf{I}(s')(k^2\hat{\mathrm{s}}\bullet\hat{\mathrm{s}}'-\frac{\partial^2}{\partial s\partial s'})g(\mathbf{r},\mathbf{r}')ds' \qquad \text{(A7)}$$

Since $\mathbf{r}'$ is now the point at s' on the wire axis while $\mathbf{r}$ is a point at s on the wire surface $|\mathbf{r}-\mathbf{r}'| \geq a$ and the integrand is bounded.

A4. Numerical Solution

The integral equations (A7) is solved numerically in NEC by a form of the method of moments. An excellent general introduction to the method of moments can be found in R.F. Harrington's book, "Field Computation by Moment Methods"[23]. A brief outline of the method follows.

The method of moments applies to a general linear-operator equation,

$$Lf = e \qquad \text{(A8)}$$

where f is an unknown response, e is a known excitation, and L is a linear operator (an integral operator in the present case). The unknown function f may be expanded in a sum of basis functions, f_j, as:

$$f = \sum_{j=1}^{N}\alpha_j f_j \qquad \text{(A9)}$$

A set of equations for the coefficients α_j are then obtained by taking the inner product of equation (A9) with a set of weighting functions $\{w_i\}$

$$< w_i, Lf >=< w_i, e > \tag{A10}$$

Due to the linearity of *L,* substituting equation (A9) for *f* yields,

$$\sum_{j=1}^{N} \alpha_j < w_i, Lf_j >=< w_i, e > \tag{A11}$$

This equation can be written in matrix notation as:

$$[G][A] = [E] \tag{A12}$$

where

$$\begin{aligned} G_{ij} &=< w_i, Lf_j > \\ A_j &= \alpha_j \\ E_i &=< w_i, e > \end{aligned} \tag{A13}$$

The solution is then

$$[A] = [G]^{-1}[E] \tag{A14}$$

For the solution of equation (A7), the inner product is defined as

$$< f, g >= \int_S f(\mathbf{r}) g(\mathbf{r}) dA' \tag{A15}$$

where the integration is over the structure surface. Various choices are possible for the weighting functions $\{w_i\}$ and basis functions $\{f_i\}$. When $w_i = f_i$, the procedure is known as Galerkin's method. In NEC the basis and weight functions are different, $\{w_i\}$ being chosen as a set of delta functions

$$w_i(\mathbf{r}) = \delta(\mathbf{r} - \mathbf{r}_i) \tag{A16}$$

with $\{\mathbf{r}_i\}$ a set of points on the conducting surface. The result is a point sampling of the integral equations known as the collocation method of solution. Wires are divided into short straight segments with a sample point at the centre of each segment.

The choice of basis functions is very important for an efficient and accurate solution. In NEC, the support of $\{f_i\}$ is restricted to a localised subsection of the surface near $\mathbf{r}_i$. This choice simplifies the evaluation of the inner-product integral and ensures that the matrix $[G]$ will be well conditioned. For finite N, the sum of $\{f_j\}$ cannot exactly equal a general current distribution so the functions $\{f_i\}$ should be chosen as close as possible to the actual current distribution. The basis functions used in NEC are explained in the following sections.

A5. Current expansion on wires

Wires in SNEC are modelled by short straight segments with the current on each segment represented by three terms - a constant, a sine, and a cosine. This expansion was first used by Yeh and Mei [24] and has been shown to provide rapid solution convergence[25],[26]. It has the added advantage that the fields of the sinusoidal currents are easily evaluated in closed form. The amplitude of the constant, sine, and cosine terms are related such that their sum satisfies physical conditions on the local behaviour of current and charge at the segment ends. This differs from AMP where the current was extrapolated to the centres of the adjacent segments, resulting in discontinuities in current and charge at the segment ends. Matching at the segment ends improves the solution accuracy, especially at the multiple-wire junctions of unequal length segments where AMP extrapolated to an average length segment, often with inaccurate results.

The total current on segment number *j* in NEC has the form:

$$I_j(s) = A_j + B_j \sin k(s - s_j) + C_j \cos k(s - s_j), \left| s - s_j \right| < \frac{\Delta_j}{2} \tag{A17}$$

where s_j is the value of s at the centre of segment *j* and Δ_j is the length of segment *j*. Of the three unknown constants A_j, B_j and C_j two are eliminated by local conditions on the current leaving one constant, related to the current amplitude, to be determined by the matrix equation. The local conditions are applied to the current and to the linear charge density, *q*, which is related to the current by the equation of continuity:

$$\frac{\partial I}{\partial s} = -j\omega q \tag{A18}$$

At a junction of two segments with uniform radius, the obvious boundary conditions are that the current and charge are continuous at the junction. At a junction of two or more segments with unequal radii, the continuity of current is generalised to Kirchoff's current law that the sum of currents into the junction is zero. The total charge in the vicinity of the junction is assumed to distribute itself on individual wires according to the wire radii. T.T. Wu and R.W.P. King [27] have derived a condition for the linear charge density on a wire at a junction that neglects local coupling effects. Using their derivation, $\frac{\partial I}{\partial s}$, can be determined by:

$$\left.\frac{\partial I(s)}{\partial s}\right|_{s \text{ at junction}} = \frac{Q}{\ln\left(\frac{2}{ka}\right) - \gamma} \tag{A19}$$

where

$a =$ wire radius

$k = \frac{2\pi}{\lambda}$

$\gamma = 0.5772$

Q is related to the total charge in the vicinity of the junction and is constant for all wires at the junction.

At a free wire end, the current may be assumed to go to zero. On a wire of finite radius, however, the current can flow into the end cap and hence be nonzero at the wire end. In one study of this effect, a condition relating the current at the wire end to the current derivative was derived [28]. For a wire radius a, this condition is:

$$I(s)\Big|_{s \text{ at end}} = \frac{-(\hat{s} \bullet \hat{n}_c)}{k} \frac{J_1(ka_j)}{J_0(ka_j)} \frac{\partial I(s)}{\partial (s)}\Bigg|_{s \text{ at end}} \tag{A20}$$

where J_o and J_1 are Bessel functions of order 0 and 1 respectively. The unit vector $\hat{n}_c$ is normal to the end cap. Hence, $\hat{s} \bullet \hat{n}_c$ is +1 if the reference direction, $\hat{s}$, is toward the end, and -1 if $\hat{s}$ is away from the end.

Thus, for each segment two equations are obtained from the two ends. At free ends the following holds:

$$I_j\left(s_j \pm \frac{\Delta_j}{2}\right) = \frac{\pm 1}{k} \frac{J_1(ka_j)}{J_0(ka_j)} \frac{\partial I(s)}{\partial (s)}\Bigg|_{s=s_j \pm \frac{\Delta_{j2}}{2}} \tag{A21}$$

When the end of a segment connects to other segments, a second equation is obtained:

$$\frac{\partial I_j(s)}{\partial s}\Bigg|_{s_j \pm \frac{\Delta_j}{2}} = \frac{Q_j^{\pm}}{\ln\left(\frac{2}{ka}\right) - \gamma} \tag{A22}$$

Two additional unknowns Q_j^- and Q_j^+ are associated with the junctions but can be eliminated by Kirchoff's current equation at each junction. The boundary condition equations provide the additional equation-per-segment to completely determine the current function of equation (A17) for every segment.

To apply these condition, the current is expanded as a sum of basis functions chosen so that they satisfy the local conditions on current and charge in any linear combination. A typical set of basis functions and their sum on a four segment wire are shown in Figure A1. For a general segment *I* in Figure A2 the i^{th} basis function has a peak on segment *i* and extends onto every segment connected to *i*, going to zero with zero derivative at the outer ends of the connected segments.

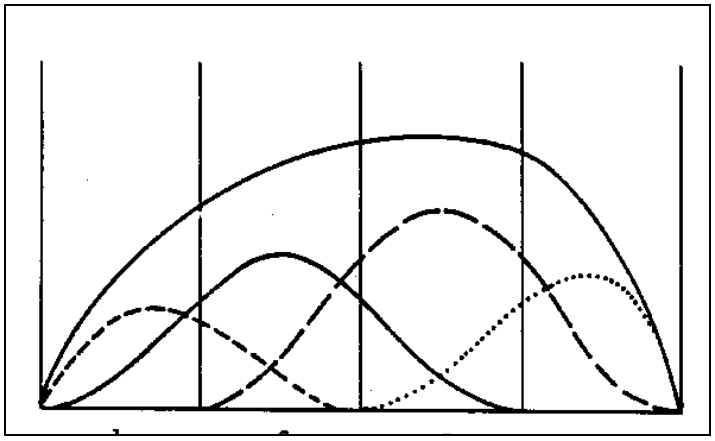


Figure A1: Current basis function and sim on a four segment wire.

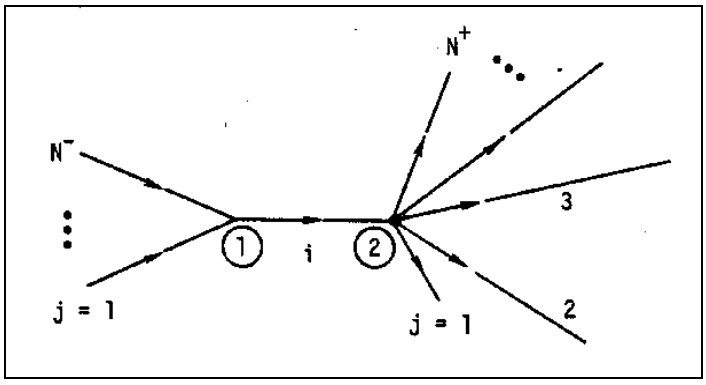


Figure A2: Segments covered by the i^{th} basis function.

The general definition of the i^{th} basis function is given below. For the junction and end conditions described above, the following definitions apply for the factors in the segment end conditions:

$$a_i^- = a_i^+ = \left[\ln\left(\frac{2}{ka_i}\right) - \gamma \right]^{-1} \tag{A23}$$

and

$$X_i = \frac{J_1(ka_i)}{J_0(ka_i)} \tag{A24}$$

The condition of zero current at a free end may be obtained by setting X_i to zero.

The portion of the i^{th} basis function on segment i is then:

$$f_i^0(s) = A_i^0 + B_i^0 \sin k(s - s_i) + C_i^0 \cos k(s - s_i), |s - s_i| < \frac{\Delta_i}{2} \tag{A25}$$

If $N^- \neq 0$ and $N^+ \neq 0$, end conditions are:

$$\frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i - \frac{\Delta_i}{2}} = a_i^- Q_i^- \tag{A26}$$

$$\frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i-\frac{\Delta_i}{2}} = a_i^+ Q_i^+ \tag{A27}$$

If $N^- = 0$ and $N^+ \neq 0$, end conditions are:

$$f_i^0\left(s_i - \frac{\Delta_i}{2}\right) = \frac{1}{k} X_i \frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i-\frac{\Delta_i}{2}} \tag{A28}$$

$$\frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i+\frac{\Delta_i}{2}} = a_i^+ Q_i^+ \tag{A29}$$

If $N^- \neq 0$ and $N^+ = 0$, end conditions are:

$$f_i^0\left(s_i + \frac{\Delta_i}{2}\right) = \frac{-1}{k} X_i \frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i+\frac{\Delta_i}{2}} \tag{A30}$$

$$\frac{\partial}{\partial s} f_i^0(s)\Big|_{s=s_i-\frac{\Delta_i}{2}} = a_i^- Q_i^- \tag{A31}$$

Over segments connected to end 1 of segment *i*, the i^{th} basis function is:

$$f_j^-(s) = A_j^- + B_j^- \sin k(s-s_j) + C_j^- \cos k(s-s_j), \left|s-s_j\right| < \frac{\Delta_j}{2},\ j = 1...N^- \tag{A32}$$

End conditions are:

$$f_j^-\left(s_j - \frac{\Delta_j}{2}\right) = 0 \tag{A33}$$

$$\frac{\partial}{\partial s} f_j^-(s)\Big|_{s=s_j-\frac{\Delta_j}{2}} = 0 \tag{A34}$$

$$\frac{\partial}{\partial s} f_j^-(s)\Big|_{s=s_j+\frac{\Delta_j}{2}} = a_j^+ Q_i^- \tag{A35}$$

Over segments connected to end 2 of segment *i*, the i^{th} basis function is:

$$f_j^+(s) = A_j^+ + B_j^+ \sin k(s-s_j) + C_j^+ \cos k(s-s_j), \left|s-s_j\right| < \frac{\Delta_j}{2},\ j = 1...N^+ \tag{A36}$$

End conditions are:

$$\frac{\partial}{\partial s} f_j^+(s)\Big|_{s=s_j-\frac{\Delta_j}{2}} = a_j^- Q_i^+ \tag{A37}$$

$$f_j^+\left(s_j - \frac{\Delta_j}{2}\right) = 0 \tag{A38}$$

$$\frac{\partial}{\partial s} f_j^+(s)\Big|_{s=s_j+\frac{\Delta_j}{2}} = 0 \tag{A39}$$

Equations (A25), (A32) and(A36), defining the complete basis function, involve $3(N^- + N^+ + 1)$ unknown constants. Of these, $3(N^- + N^+) + 2$ unknowns are eliminated by the end conditions in terms of Q_i^- and Q_i^+ which can then be determined from the two Kirchoff's current equations:

$$\sum_{j=1}^{N^-} f_j^-\left(s_j + \frac{\Delta_j}{2}\right) = f_i^0\left(s_i - \frac{\Delta_i}{2}\right) \tag{A40}$$

$$\sum_{j=1}^{N^+} f_j^+\left(s_j + \frac{\Delta_j}{2}\right) = f_i^0\left(s_i - \frac{\Delta_i}{2}\right) \tag{A41}$$

The complete basis function is then defined in terms of one unknown constant. In this case A_i^0 was set to -1 since the function amplitude is arbitrary, being determined by the boundary condition equations. The final result is given below:

$$A_j^- = \frac{a_j^+ Q_i^-}{\sin k\Delta_j} \tag{A42}$$

$$B_j^- = \frac{a_j^+ Q_i^-}{2\cos k\frac{\Delta_j}{2}} \tag{A43}$$

$$C_j^- = \frac{-a_j^+ Q_i^-}{2\sin k\frac{\Delta_j}{2}} \tag{A44}$$

$$A_j^+ = \frac{-a_j^- Q_i^+}{\sin k\Delta_j} \tag{A45}$$

$$B_j^+ = \frac{a_j^- Q_i^+}{2\cos k\dfrac{\Delta_j}{2}} \tag{A46}$$

$$C_j^+ = \frac{a_j^- Q_i^+}{2\sin k\dfrac{\Delta_j}{2}} \tag{A47}$$

For $N^- \neq 0$ and $N^+ \neq 0$:

$$A_i^0 = -1 \tag{A48}$$

$$B_i^0 = \left(a_i^- Q_i^- + a_i^+ Q_i^+\right)\frac{\sin k\dfrac{\Delta_i}{2}}{\sin k\Delta_i} \tag{A49}$$

$$C_i^0 = \left(a_i^- Q_i^- - a_i^+ Q_i^+\right)\frac{\cos k\dfrac{\Delta_i}{2}}{\sin k\Delta_i} \tag{A50}$$

$$Q_i^- = \frac{a_i^+\left(1-\cos k\Delta_i\right) - P_i^+ \sin k\Delta_i}{\left(P_i^- P_i^+ + a_i^- a_i^+\right)\sin k\Delta_i + \left(P_i^- P_i^+ - P_i^+ a_i^-\right)\cos k\Delta_i} \tag{A51}$$

$$Q_i^+ = \frac{a_i^-\left(\cos k\Delta_i - 1\right) - P_i^- \sin k\Delta_i}{\left(P_i^- P_i^+ + a_i^- a_i^+\right)\sin k\Delta_i + \left(P_i^- P_i^+ - P_i^+ a_i^-\right)\cos k\Delta_i} \tag{A52}$$

For $N^- = 0$ and $N^+ \neq 0$:

$$A_i^0 = -1 \tag{A53}$$

$$B_i^0 = \frac{\sin k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} + a_i^+ Q_i^+ \frac{\cos k\dfrac{\Delta_i}{2} - X_i \sin k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} \tag{A54}$$

$$C_i^0 = \frac{\cos k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} + a_i^+ Q_i^+ \frac{\sin k\dfrac{\Delta_i}{2} - X_i \cos k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} \tag{A55}$$

$$Q_i^+ = \frac{\cos k\Delta_i - 1 - X_i \sin k\Delta_i}{\left(a_i^+ + X_i P_i^+\right)\sin k\Delta_i + \left(a_i^+ X_i - P_i^+\right)\cos k\Delta_i} \tag{A56}$$

For $N^- \neq 0$ and $N^+ = 0$:

$$A_i^0 = -1 \tag{A57}$$

$$B_i^0 = \frac{-\sin k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} + a_i^- Q_i^- \frac{\cos k\dfrac{\Delta_i}{2} - X_i \sin k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} \tag{A58}$$

$$C_i^0 = \frac{\cos k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} - a_i^- Q_i^- \frac{\sin k\dfrac{\Delta_i}{2} - X_i \cos k\dfrac{\Delta_i}{2}}{\cos k\Delta_i - X_i \sin k\Delta_i} \tag{A59}$$

$$Q_i^- = \frac{1 - \cos k\Delta_i + X_i \sin k\Delta_i}{\left(a_i^- - X_i P_i^-\right)\sin k\Delta_i + \left(P_i^- + a_i^- X_i\right)\cos k\Delta_i} \tag{A60}$$

For all cases:

$$P_i^- = \sum_{j=1}^{N^-} \left(\frac{1 - \cos k\Delta_i}{\sin k\Delta_j} \right) a_j^+ \tag{A61}$$

$$P_i^+ = \sum_{j=1}^{N^+} \left(\frac{\cos k\Delta_j - 1}{\sin k\Delta_j} \right) a_j^- \tag{A62}$$

where the sum of P_i^- is over segments connected to end 1 of segment *i*, and the sum for P_i^+ is over segments connected to end 2. If $N^- = N^+ = 0$ the complete basis function is:

$$f_i^0 = \frac{\cos k(s - s_i)}{\cos k\dfrac{\Delta_i}{2} - X_i \sin \dfrac{\Delta_i}{2}} - 1 \tag{A63}$$

When a segment end is connected to a ground plane, the end condition on both the total current and the last basis function is:

$$\frac{\partial}{\partial s} I_j(s)\Big|_{s=s_j \pm \frac{\Delta_j}{2}} = 0 \tag{A64}$$

This replaces the zero current condition at a free end. This condition does not require a separate treatment, however, but is obtained by computing the last basis function as if the last segment is connected to its image segment on the other side of the surface.

It should be noted that in AMP, the basis function f_i has unit value at the centre of segment *i* and zero value at the centres of connected segments although it does extend onto the connected segments. As a result, the amplitude of f_i is the total current at the centre of segment *i* must be computed by summing the contributions of all basis functions extending onto segment *i*.

A6. Evaluation of the fields

The current on each wire segment has the form:

$$I_i(s) = A_j + B_j \sin k(s - s_j) + C_j \cos k(s - s_j), \left|s - s_j\right| < \frac{\Delta_i}{2} \tag{A65}$$

where

$$k = \omega\sqrt{\mu_0 \varepsilon_0}$$

Δ_i is the segment length

The solution requires the evaluation of the electric field at each segment due to this current. Three approximations of the integral equation are used: a thin-wire form for most cases, an extended thin-wire form for thick wires, and a current element approximation for large interaction distances. In each case, the evaluation of the field is greatly simplified by the use of formulas for the fields of the constant and sinusoidal current components.

The accuracy of the thin-wire approximation for a wire of radius *a* and length Δ depends on *ka* and $\frac{\Delta}{a}$. Studies have shown that the thin-wire approximation leads to errors of less than 1% for $\frac{\Delta}{a}$ greater than 8 [22]. Furthermore, in the numerical solution of the EFIE, the wire is divided into segments less than about 0.1λ in length to obtain an adequate representation of current distribution thus restricting *ka* to less than about 0.08. The extended thin-wire approximation is applicable to shorter and thicker segments, resulting in errors less than 1% for $\frac{\Delta}{a}$ greater than 2.

For the thin-wire kernel, the source current is approximated by a filament on the segment axis while the observation point is on the surface of the observation segment. The fields are evaluated with the source segment on the axis of a local cylindrical coordinate system as illustrated in Figure A3.

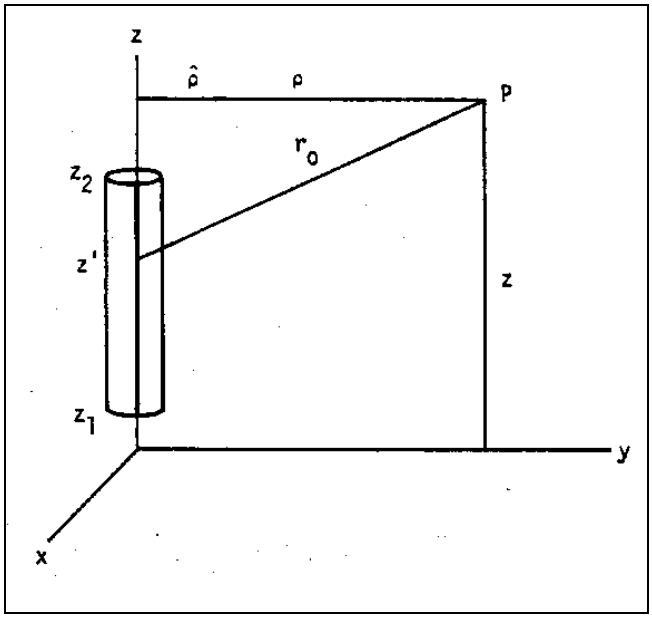


Figure A3: Current-filament geometry for the thin-wire kernel.

Using

$$G_0 = e^{\frac{-jkr_0}{r_0}} \tag{A66}$$

$$r_0 = \left[\rho^2 + (z - z')^2\right]^{\frac{1}{2}} \tag{A67}$$

the ρ and z components of the electric field at *P* due to a sinusoidal current filament of arbitrary phase

$$I = \sin(kz' - \theta_0), \quad z_1 < z' < z_2 \tag{A68}$$

are:

$$E_\rho^f(\rho, z) = \frac{-j\eta}{2k^2\lambda\rho}\left[(z'-z)I\frac{\partial G_0}{\partial z'} + IG_0 - (z'-z)G_0\frac{\partial I}{\partial z'}\right]_{z_1}^{z_2} \tag{A69}$$

$$E_z^f(\rho, z) = \frac{j\eta}{2k^2\lambda}\left[G_0\frac{\partial I}{\partial z'} - I\frac{\partial G_0}{\partial z'}\right]_{z_1}^{z_2} \tag{A70}$$

For a current that is constant over the length of the segment with strength *I*, the fields are:

$$E_\rho^f(\rho, z) = \frac{I}{\lambda}\frac{j\eta}{2k^2}\left[\frac{\partial G_0}{\partial \rho}\right]_{z_1}^{z_2} \tag{A71}$$

$$E_z^f(\rho,z)=\frac{I}{\lambda}\frac{j\eta}{2k^2}\left\{\left[\frac{\partial G_0}{\partial z'}\right]_{z_1}^{z_2}+k^2\int_{z_1}^{z_2}G_0\,dz\right\} \tag{A72}$$

These field expressions are exact for the specified currents. The integral over z' of G_0 is evaluated numerically in NEC.

Substituting sine and cosine currents and evaluating the derivatives yields the following equations for the fields. For

$$I=I_0\begin{pmatrix}\sin kz'\\ \cos kz'\end{pmatrix} \tag{A73}$$

$$E_\rho^f(\rho,z)=\frac{-I_0}{\lambda}\frac{j\eta}{2k^2\rho}G_0\left\{k(z-z')\begin{pmatrix}\cos kz'\\ -\sin kz'\end{pmatrix}+\left[1-(z-z')^2(1+jkr_0)\frac{1}{r_0^2}\right]\begin{pmatrix}\sin kz'\\ \cos kz'\end{pmatrix}\right\} \tag{A74}$$

$$E_z^f(\rho,z)=\frac{I_0}{\lambda}\frac{j\eta}{2k^2}G_0\left\{k\begin{pmatrix}\cos kz'\\ -\sin kz'\end{pmatrix}-(z-z')\ (1+jkr_0)\frac{1}{r_0^2}\begin{pmatrix}\sin kz'\\ \cos kz'\end{pmatrix}\right\}\Bigg|_{z_1}^{z_2} \tag{A75}$$

For a constant current of strength I_0:

$$E_\rho^f(\rho,z)=\frac{-I_0}{\lambda}\frac{j\eta\rho}{2k^2}\left[(1+jkr_0)\frac{G_0}{r_0^2}\right]_{z_1}^{z_2} \tag{A76}$$

$$E_z^f(\rho,z)=\frac{-I_0}{\lambda}\frac{j\eta}{2k^2}\left\{\left[(z-z')\ (1+jkr_0)\frac{G_0}{r_0^2}\right]_{z_1}^{z_2}+k^2\int_{z_1}^{z_2}G_0\,dz\right\} \tag{A77}$$

Despite the seemingly crude approximation, the thin-wire kernel does accurately represent the effect of wire radius for wires that are sufficiently thin. The accuracy range was studied by Poggio and Adams [22] where an extended thin-wire kernel was developed for wires that are too thick for the thin-wire approximation,

The derivation of the extended thin-wire kernel starts with the current on the surface of the source segment with surface density.

$$J(z')=\frac{I(z')}{2\pi a} \tag{A78}$$

where a is the radius of the source segment. The geometry for evaluation of the fields is shown in Figure A4. A current filament of strength $\frac{Id\phi}{2\pi}$ is integrated over ϕ with

$$\rho' = \left[\rho^2 + a^2 - 2a\rho\cos\phi\right]^{\frac{1}{2}} \tag{A79}$$

$$r = \left[\rho'^2 + (z - z')^2\right]^{\frac{1}{2}} \tag{A80}$$

Thus, the z component of the field of the current tube is

$$E_z^t(\rho, z) = \frac{1}{2\pi}\int_0^{2\pi} E_z^f(\rho', z)\,d\phi \tag{A81}$$

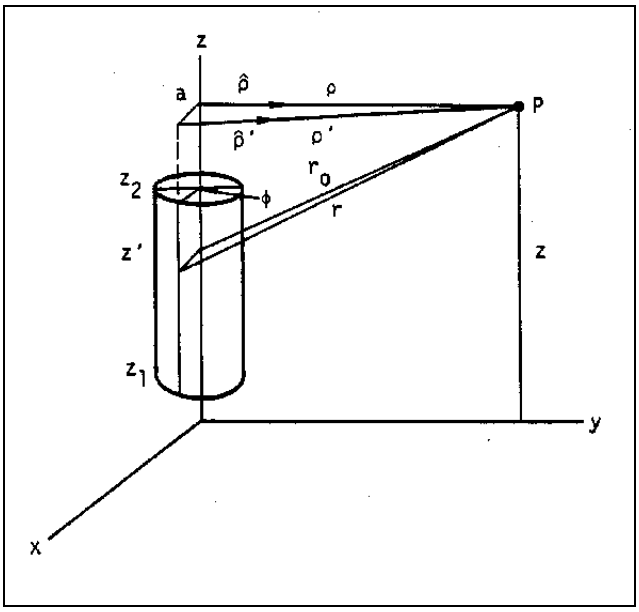


Figure A4: Current geometry for the extended thin-wire kernel.

For the ρ component of field, the change in the direction of $\hat{\rho}'$ must be considered. The field in the direction $\hat{\rho}$ is:

$$E_z^t(\rho, z) = \frac{1}{2\pi}\int_0^{2\pi} E_z^f(\rho', z)(\hat{\rho}\bullet\hat{\rho}')\,d\phi \tag{A82}$$

where

$$\hat{\rho}\bullet\hat{\rho}' = \frac{\rho - a\cos\phi}{\rho'} = \frac{\partial\rho'}{\partial\rho} \tag{A83}$$

The integral over ϕ in equations (A81) and (A82) cannot be evaluated in closed form. Poggio and Adams, however, have evaluated them as a series in powers of a^2 [83]. The first term in the series gives the thin-wire kernel. For the extended thin-wire kernel, the second term involving a^2 is retained with terms of order a^4 neglected. As with the thin-wire kernel, the field observation point is on the segment surface. Hence, when evaluating the field on the source segment, $\rho = a$.

The field equations with the extended thin-wire approximation are given below. For a sinusoidal current of equation (A68):

$$E_\rho(\rho,z)=\frac{-j\eta}{2k^2\lambda\rho}\left[(z'-z)I\frac{\partial G_0}{\partial z'}+IG_2-(z'-z)G_2\frac{\partial I}{\partial z'}\right]_{z_1}^{z_2} \tag{A84}$$

$$E_z(\rho,z)=\frac{j\eta}{2k^2\lambda}\left[G_1\frac{\partial I}{\partial z'}-I\frac{\partial G_1}{\partial z'}\right]_{z_1}^{z_2} \tag{A85}$$

For a constant current of strength I_0:

$$E_\rho(\rho,z)=\frac{I_0}{\lambda}\frac{j\eta}{2k^2}\left[\frac{\partial G_1}{\partial\rho}\right]_{z_1}^{z_2} \tag{A86}$$

$$E_z(\rho,z)=\frac{-I_0}{\lambda}\frac{j\eta}{2k^2}\left\{\left[\frac{\partial G_1}{\partial z'}\right]_{z_1}^{z_2}+k^2\left[1-\frac{(ka)^2}{4}\right]\int_{z_1}^{z_2}G_0dz-\frac{(ka)^2}{4}\left[\frac{\partial G_0}{\partial z'}\right]_{z_1}^{z_2}\right\} \tag{A87}$$

The term G_1 is the series approximation of:

$$G_1^t=\frac{1}{2\pi}\int_0^{2\pi}Gd\phi \tag{A88}$$

where

$$G=e^{\frac{-jkr}{r}} \tag{A89}$$

Neglecting terms of order a^4:

$$G_1=G_0\left\{1-\frac{a^2}{2r_0^2}(1+jkr_0)+\frac{a^2\rho^2}{4r_0^4}\left[3(1+jkr_0)-k^2r_0^2\right]\right\} \tag{A90}$$

$$\frac{\partial G_1}{\partial z'}=\frac{(z-z')}{r_0^2}G_0\left\{\begin{array}{l}(1+jkr_0)-\frac{a^2}{2r_0^2}\left[3(1+jkr_0)-k^2r_0^2\right]\\-\frac{a^2\rho^2}{4r_0^4}\left[jk^3r_0^3+6k^2r_0^2-15(1+jkr_0)\right]\end{array}\right\} \tag{A91}$$

$$\frac{\partial G_1}{\partial\rho}=\frac{\rho G_0}{r_0^2}\left\{\begin{array}{l}(1+jkr_0)-\frac{a^2}{r_0^2}\left[3(1+jkr_0)-k^2r_0^2\right]\\-\frac{a^2\rho^2}{4r_0^4}\left[jk^3r_0^3+6k^2r_0^2-15(1+jkr_0)\right]\end{array}\right\} \tag{A92}$$

The term G_2 is the series approximation of:

$$G_2^t = \frac{1}{2\pi}\int_0^{2\pi} \frac{\rho - a\cos\phi}{\rho'^2} G d\phi \tag{A93}$$

To order a^2:

$$G_2 = \frac{G_0}{\rho}\left\{1 + \frac{a^2\rho^2}{4r_0^4}\left[3(1+jkr_0) - k^2 r_0^2\right]\right\} \tag{A94}$$

$$\frac{\partial G_2}{\partial z'} = \frac{(z-z')}{\rho r_0^2} G_0 \left\{(1+jkr_0) - \frac{a^2\rho^2}{4r_0^4}\left[jk^3 r_0^3 + 6k^2 r_0^2 - 15(1+jkr_0)\right]\right\} \tag{A95}$$

Equation (A86) makes use of the relation:

$$(\hat{\rho}\bullet\hat{\rho}')\frac{\partial G}{\partial \rho'} = \frac{\partial G}{\partial \rho'}\frac{\partial \rho'}{\partial \rho} = \frac{\partial G}{\partial \rho} \tag{A96}$$

while equation (A87) follows from:

$$G_1 = \left[1 - \frac{(ka)^2}{4} - \frac{a^2}{4}\frac{\partial^2}{\partial z'^2}\right] G_0 \tag{A97}$$

When the observation point is within the wire $(\rho < a)$, a series expansion in ρ rather than a is used for G_0 and G_2. For G_1 this simply involves inter-changing ρ and a in equations (A90) and (A91). Then for $\rho < a$, with:

$$r_a = \left[a^2 + (z - z;)^2\right]^{\frac{1}{2}} \tag{A98}$$

$$G_a = e^{\frac{-jkr_a}{r_a}} \tag{A99}$$

the expressions for G_1, G_2 and their derivatives are:

$$G_1 = G_a\left\{1 - \frac{\rho^2}{2r_a^2}(1+jkr_a) + \frac{a^2\rho^2}{4r_a^4}\left[3(1+jkr_a) - k^2 r_a^2\right]\right\} \tag{A100}$$

$$\frac{\partial G_1}{\partial z'} = \frac{(z-z')}{r_a^2} G_a \left\{\begin{array}{l}(1+jkr_a) - \frac{\rho^2}{2r_a^2}\left[3(1+jkr_a) - k^2 r_a^2\right] \\ -\frac{a^2\rho^2}{4r_a^4}\left[jk^3 r_a^3 + 6k^2 r_a^2 - 15(1+jkr_a)\right]\end{array}\right\} \tag{A101}$$

$$\frac{\partial G_1}{\partial \rho} = \frac{-\rho}{r_a^2} G_a \left\{ (1 + jkr_a) - \frac{a^2}{2r_a^2} \left[3(1 + jkr_a) - k^2 r_a^2 \right] \right\} \tag{A102}$$

$$G_2 = -\frac{\rho}{2r_a^2} G_a (1 + jkr_a) \tag{A103}$$

$$\frac{\partial G_2}{\partial z'} = -\frac{(z - z')\rho}{2r_a^4} G_a \left\{ \left[3(1 + jkr_a) - k^2 r_a^2 \right] \right\} \tag{A104}$$

Special treatment of bends in wires is required when the extended thin-wire kernel is used. The problem stems from the cancellation of terms evaluated at $z1$ and $z2$ in the field when segments are part of a continuous wire. The current expansion in NEC results in a current having a continuous value and derivative along a wire without junctions. This ensures that for two adjacent segments on a straight wire, the contributions to the z component of electric field at $z2$ of the first segment exactly cancel the contributions from $z1$, representing the same point, for the second segment. For a straight wire of several segments, the only contributions to E_z with either the thin-wire or extended thin-wire kernel come from the two wire ends and the integral of G_0 along the wire. For the ρ component of field or either component at a bend, while there is not complete cancellation, there may be partial cancellation of large end contributions.

The cancellation of end terms makes necessary a consistent treatment of the current on both sides of a bend for accurate evaluation of the field. This is easily accomplished with the thin-wire kernel since the current filament on the wire axis is physically continuous around a bend. However, the current tube assumed for the extended thin-wire kernel cannot be continuous around its complete circumference at a bend. This was found to reduce the solution accuracy when the extended thin-wire kernel was used for bent wires.

To avoid this problem in NEC, the thin-wire form of the end terms in equation (A69) through (A72) is always used at a bend or change in radius. The extended thin-wire kernel is used only at segment ends where two parallel segments join, or at free ends. The switch from extended thin-wire form to the thin-wire form is made from one end of a segment to the other rather than between segments where the cancellation of terms is critical.

When segments are separated by a large distance, the interaction may be computed with sufficient accuracy by treating the segment current as an infinitesimal current element at the segment centre. In spherical co-ordinates, with the segment at the origin along the $\theta = 0$ axis, the electric field is:

$$E_r(r, \theta) = \frac{M\eta}{2\pi r^2} e^{-jkr} \left(1 - \frac{j}{kr} \right) \cos\theta \tag{A105}$$

$$E_\theta(r, \theta) = \frac{M\eta}{4\pi r^2} e^{-jkr} \left(1 + jkr - \frac{j}{kr} \right) \sin\theta \tag{A106}$$

The dipole moment *M* for a constant current *I* on a segment of length Δ_i is:

$$M = I\Delta_i \tag{A107}$$

For a current $I\cos[k(s-s_i)]$ with $|s-s_i| < \frac{\Delta_i}{2}$

$$M = \frac{2I}{k}\sin\left(\frac{k\Delta_i}{s}\right) \tag{A108}$$

while for a current $I\sin[k(s-s_i)]$

$$M = 0 \tag{A109}$$

Use of this approximation saves a significant amount of time in evaluating the interaction matrix elements for large structures. The minimum interaction distance at which it is used is selected by the user in NEC. A default distance of one wavelength is set, however.

For each of the three methods of computing the field at a segment due to the current on another segment, the field is evaluated on the surface of the observation segment. Rather than choosing a fixed point on the segment surface, the field is evaluated at the cylindrical coordinates (ρ', z) with the source segment at the origin. If the centre point on the axis of the observation segment is at (ρ, z), then:

$$\rho' = \left[\rho^2 + a_0^2\right]^{\frac{1}{2}} \tag{A110}$$

where a_0 is the radius of the observation segment. Also, the component of E_ρ tangent to the observation segment is computed as:

$$\mathbf{E}_\rho \bullet \hat{s} = (\hat{\rho} \bullet \hat{s})\frac{\rho}{\rho'}E_\rho \tag{A111}$$

Inclusion of the factor $\frac{\rho}{\rho'}$, which is the cosine of the angle between $\hat{\rho}$ and $\hat{\rho}'$, is necessary for accurate results at bends in thick wires.

A7. The matrix equation for current

For a structure having N_s wire segments, the order of the matrix in equation (A12) is N_s The matrix equation has the form:

$$[G][I] = [E] \tag{A112}$$

I is then the column vector of segment basis function amplitudes. The elements of *E* are the left-hand side of equation (A7) evaluated at the segment centres.

A matrix element G_{ij} in matrix G represents the electric field at the centre of segment *i* due to the *jth* segment basis function, centred on segment *j*.

A8. Solution of the matrix equation

The matrix equation

$$[G][I]=[E] \tag{A113}$$

is solved in NEC by Gauss elimination [29]. The basic step is factorisation of the matrix G into the product of an upper triangular matrix U and a lower triangle matrix L where

$$G=[L][U] \tag{A114}$$

The matrix equation is then

$$[L][U][I]=[E] \tag{A115}$$

from which the solution, *I*, is computed in two steps as

$$[L][F]=[E] \tag{A116}$$

and

$$[U][I]=[F] \tag{A117}$$

Equation (A116) is first solved for F by forward substitution, and equation (A117) is then solved for *I* by backward substitution.

The major computational effort is factoring *G* into *L* and *U*. This takes approximately $\frac{1}{3}N^3$ multiplication steps for a matrix of order *N* compared to N^3 for inversion of *G* by the Gauss-Jordan method. Solution of equations (A116) and (A117), making use of the triangular properties of *L* and *U*, takes approximately as many multiplications as would be required for multiplication of G^{-1} by the column vector *E*. The factored matrices *L* and *U* are saved in NEC since the solution for induced current may be repeated for a number of different excitations. This, then, requires only the repeated solution of equations (A116) and (A117).

Computation of the elements of the matrix *G* and solution of the matrix equation are the two most time-consuming steps in computing the response of a structure, often accounting for over 90% of the computation time. This may be reduced substantially by making use of symmetries of the structure, either symmetry about a plane, or symmetry under rotation.

In rotational symmetry, a structure having M sectors is unchanged when rotated by any multiple of *360/M* degrees. If the equations for all segments in the first sector are numbered first and followed by successive sectors in the same order, the matrix equation can be expanded in sub matrices in the form:

$$\begin{bmatrix} A_1 & A_2 & A_3 & \cdots & A_{M-1} & A_M \\ A_M & A_1 & A_2 & \cdots & A_{M-2} & A_{M-1} \\ A_{M-1} & A_M & A_1 & \cdots & A_{M-3} & A_{M-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_2 & A_3 & A_4 & \cdots & A_M & A_1 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_M \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \\ \vdots \\ E_M \end{bmatrix} \tag{A118}$$

If there are N_c equations in each sector, E_i and I_i are N_c element column vectors of the excitations and currents in sector *i*. A_i is a sub matrix of order N_c containing the interaction fields in sector 1 due to currents in sector *i*. Due to symmetry, this is the same as the fields in sector *k* due to currents in sector $i+k$, resulting in the repetition pattern shown. Thus, only matrix elements in the first row of sub matrices need be computed, reducing the time to fill the matrix by a factor of *1/M*.

The time to solve the matrix equation can also be reduced by expanding the excitation sub-vectors in a discrete Fourier series as:

$$E_i = \sum_{k=1}^{M} S_{ik} E_k, \; i = 1 \ldots M \tag{A119}$$

$$E_i = \frac{1}{M} \sum_{k=1}^{M} S_{ik}^{*} E_k, \; i = 1 \ldots M \tag{A120}$$

where

$$s_{ik} = e^{\frac{j2\pi(i-1)(k-1)}{M}} \tag{A121}$$

where

$$j = \sqrt{-1}$$

* indicates the conjugate of the complex number.

Examining a component in the expansion,

$$E = \begin{bmatrix} S_{1k} E_k \\ S_{2k} E_k \\ \\ \\ S_{Mk} E_k \end{bmatrix} \tag{A122}$$

it is seen that the excitation differs from sector to sector only by a uniform phase shift. This excitation of a rotationally symmetric structure results in a solution having the same form as the excitation, i.e.,

$$I = \begin{bmatrix} S_{1k} I_k \\ S_{2k} I_k \\ \\ \\ S_{Mk} I_k \end{bmatrix} \tag{A123}$$

It can be shown that this relation between solution and excitation holds for any matrix having the form of that in equation (A118). Substituting these components of E and I into equation (A118) yields the following matrix equation of order N_c relating I_k to E_k:

$$\left[S_{1k} A_1 + S_{2k} A_2 + \ldots + S_{Mk} A_M\right]\left[I_k\right] = S_{1k}\left[E_k\right] \tag{A124}$$

The solution for the total excitation is then obtained by an inverse transformation,

$$I_i = \sum_{k=1}^{M} S_{ik} I_k, \quad i = 1 \ldots M \tag{A125}$$

The solution procedure, then, is first to compute the M sub matrices A_i and Fourier-transform these to obtain

$$A_i = \sum_{k=1}^{M} S_{ik} A_k, \quad i = 1 \ldots M \tag{A126}$$

The matrices A_i of order N_c are then each factored into upper and lower triangular matrices by the Gauss elimination method. For each excitation vector, the transformed sub-vectors are then computed by equation (A120) and the transformed current sub-vectors are obtained by solving the M equations,

$$\left[A_i\right]\left[I_i\right] = \left[E_i\right] \tag{A127}$$

The total solution is then given by equation (A125).

The same procedure can be used for structures that have planes of symmetry. The Fourier transform is then replaced by even and odd excitations about each symmetry plane. All equations remain the same with the exception that the matrix S with elements S_{ij}, given by equation (A121), is replaced by the following matrices:

For one plane of symmetry:

$$S = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{A128}$$

For two orthogonal planes of symmetry:

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{A129}$$

and for three orthogonal symmetry planes:

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \tag{A130}$$

For either rotational or plane symmetry, the procedure requires factoring of M matrices of order N_c rather than one matrix of order MN_c. Each excitation then requires the solution of the M matrix equations. Since the time for factoring is approximately proportional to the cube of the matrix order and the time for solution is proportional to the square of the order, the symmetry results in a reduction of factor time by M^{-2} and in solution time by M^{-1}. The time to compute the transforms is generally small compared to the time for matrix operations since it is proportional to a lower power of N_c. Symmetry also reduces the number of locations required for matrix storage by M^{-1} since only the first row of sub matrices need be stored. The transformed matrices, A_i can replace the matrices A_i as they are computed.

NEC includes a provision to generate and factor an interaction matrix and save the result of a file. A later run, using the file, may add to the structure and solve the complete model without unnecessary repetition of calculations. This procedure is called the Numerical Green's Function (NGF) option since the effect is as if the free space Green's function in NEC were replaced by the Green's function for the structure on the file. The NGF is particularly useful for a large structure, such as a ship, on which various antennas will be added or modified. It also permits taking advantage of partial symmetry since a NGF file may be written for the symmetric part of a structure, taking advantage of the symmetry to reduce computation time. Unsymmetric parts can be added in a later run.

For the NGF solution the matrix is partitioned as

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \end{bmatrix} \tag{A131}$$

where A is the interaction matrix for the initial structure, D is the matrix for the added structure, and B and C represent mutual interactions. The current is computed as

$$I_2 = \left[D - CA^{-1}B\right]^{-1}\left[E_2 - CA^{-1}E_1\right] \tag{A132}$$

$$I_1 = A^{-1}E_1 - A^{-1}BI_2 \tag{A133}$$

after the factored matrix A has been read from the NGF file along with other necessary data.

Electrical connections between the new structure and the old (NGF) structure require special treatment. If a new wire connects to an old wire the current basis function for the old wire segment is changed by the modified condition at the junction. The old basis function is given zero amplitude by adding a new equation having all zeros except for a one in the column of the old basis function. A new column is added for the corrected basis function.

Appendix B: Absolute Error Plots for Preconditioned BICGSTAB(L) problems

B1. Introduction

The approximate solutions found using the new preconditioner with BICGSTAB(L) presented in this thesis needs to be checked against the full solutions of the system of equations. This is done by looking at the absolute error plots of each problem solved.

The absolute error is defined as:

$$|xapprox - x| \tag{B1}$$

B2. Results

The absolute error for each problem solved is plotted here. The plot looks at the absolute error for every unknown in the matrix. The average is also plotted.

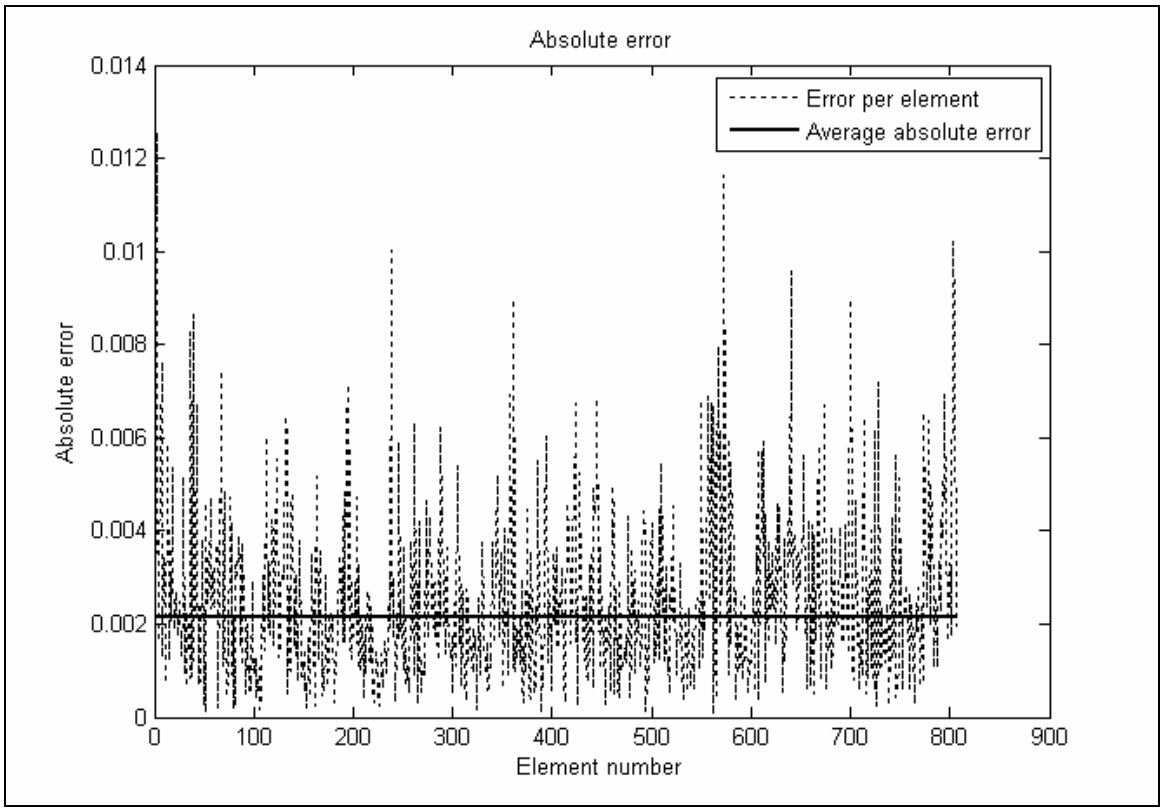


Figure B1: Absolute Error for the Problem with 807 Unknowns

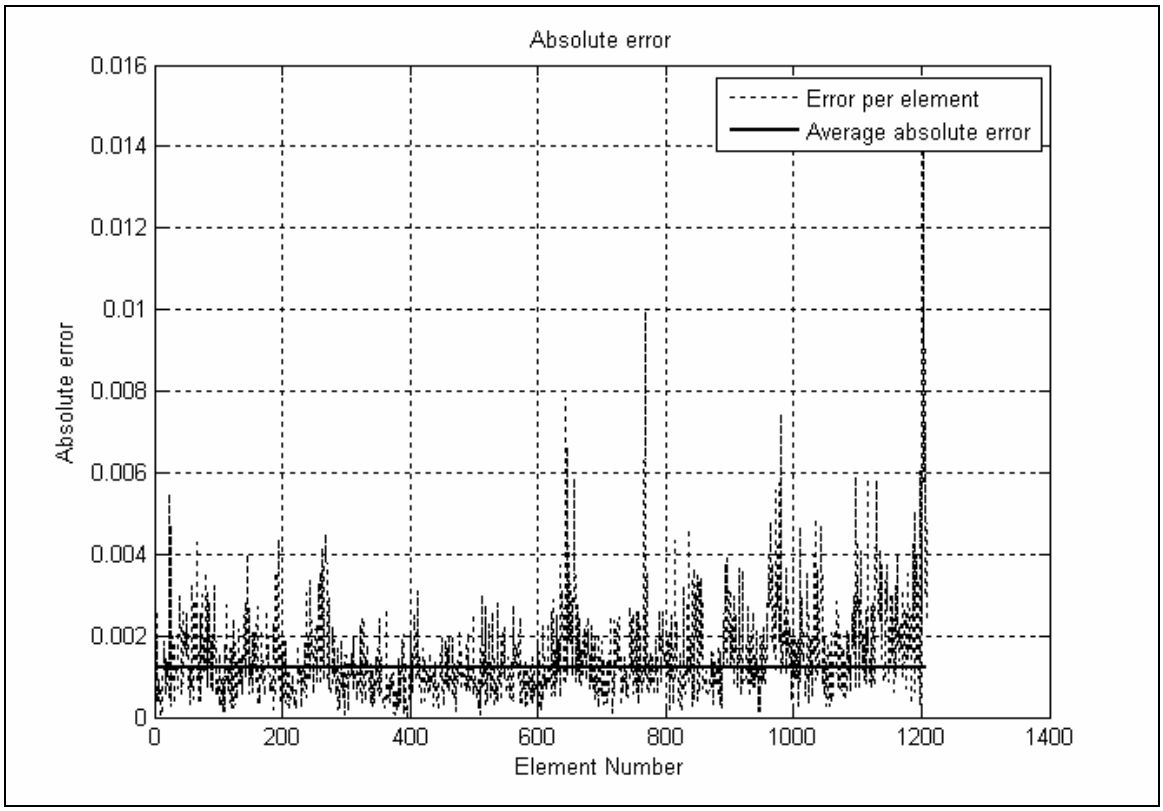


Figure B2: Absolute Error for the Problem with 1207 Unknowns

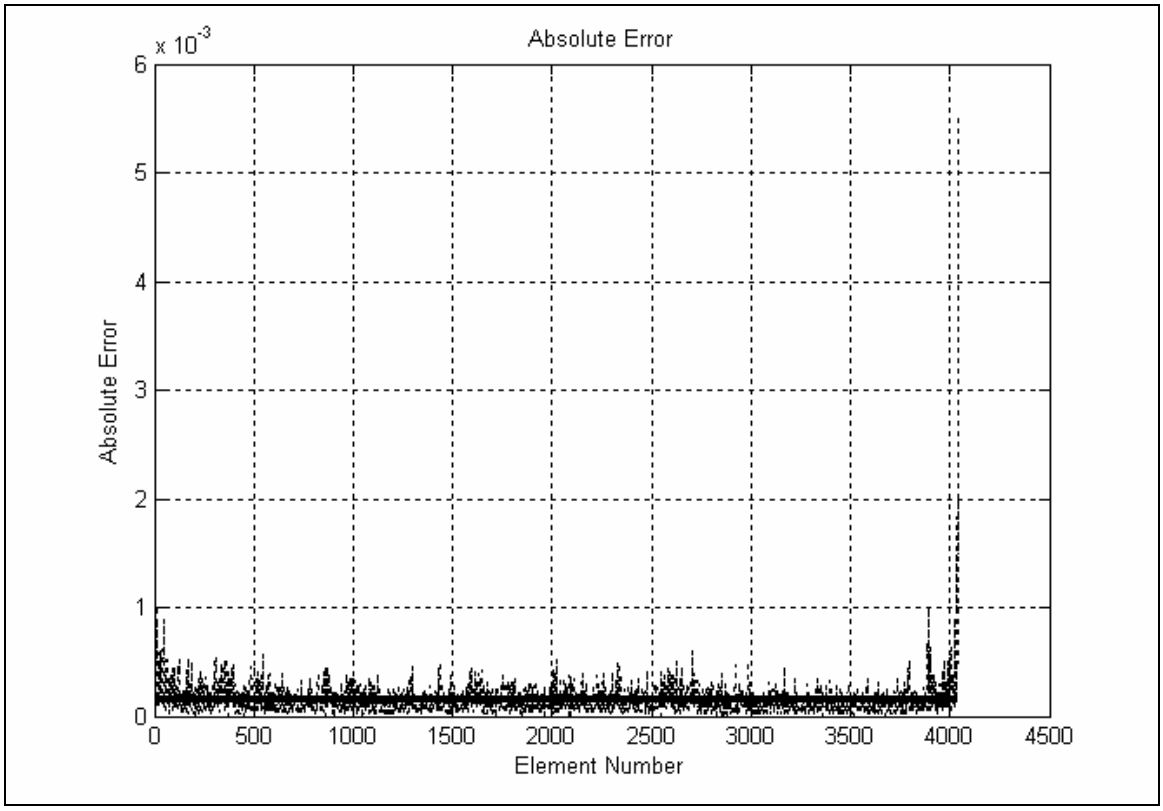


Figure B3: Absolute Error for the Problem with 4043 Unknowns

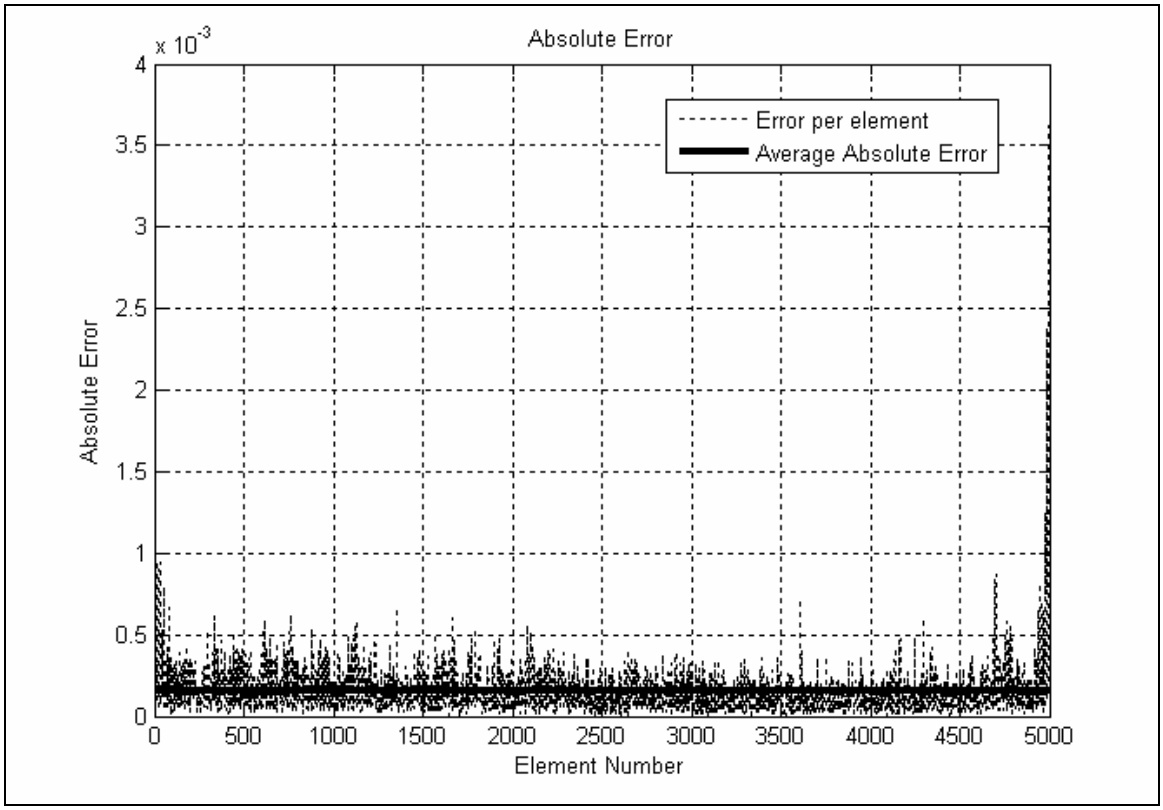


Figure B4: Absolute Error for the Problem with 5000 Unknowns

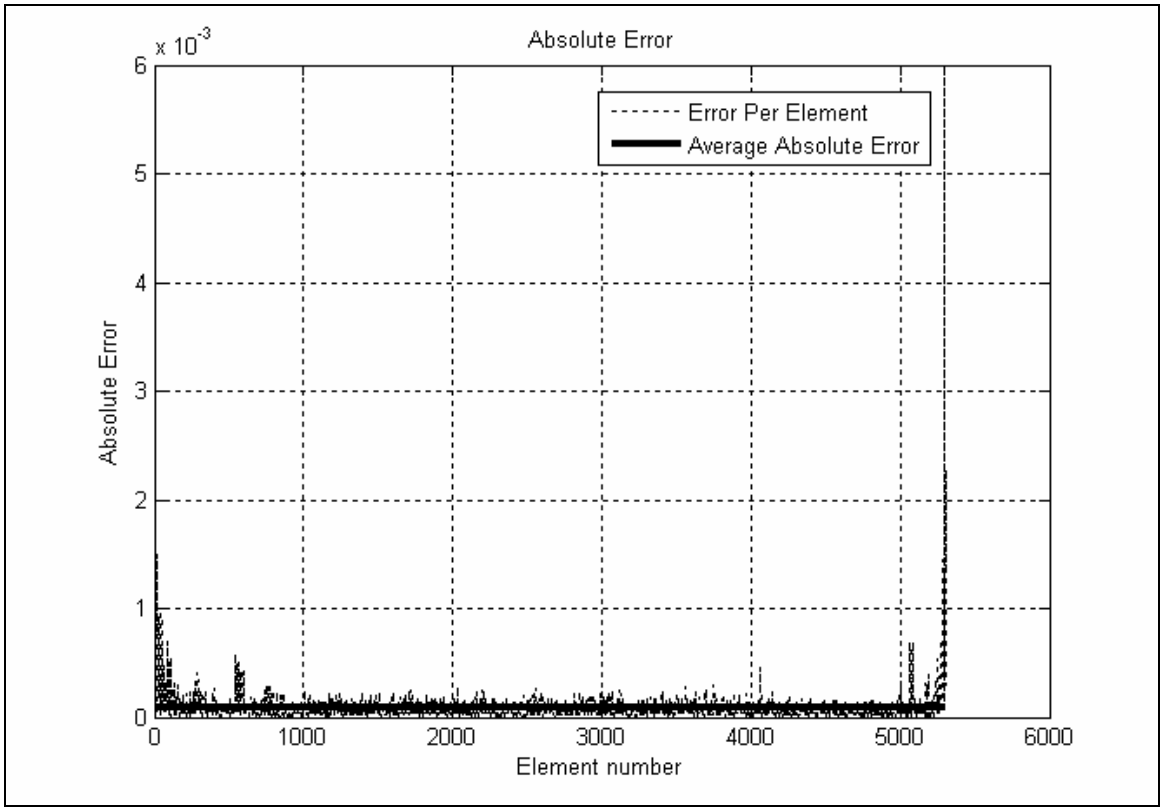


Figure B5: Absolute Error for the Problem with 5303 Unknowns

B3. Conclusion

From the absolute error plots of all the problems we can see that the approximation to the system of equations solved with the preconditioned BICGSTAB(L) method is within an acceptable range, i.e. of the order of 1×10^{-3}, as errors of this size have no significant impact on the results such as the radiation pattern and antenna impedance. If greater accuracy is required a further iteration can be done without significantly increasing the time taken to solve the system of equations. These plots therefore show that the preconditioning method developed finds the solutions to problems with an acceptably small error.

Appendix C: Preconditioning Multiplier Study

C1. Introduction

The Simply Sparse matrix and BICGSTAB(L) have been investigated in an attempt to speed up solution times to problems in chapters 4 and 5. This appendix looks at how the preconditioner presented in chapter 6 was obtained, thereby successfully combining the Simply Sparse matrix and BICGSTAB(L) to find solutions to the system of electromagnetic equations more quickly.

C2. Reasoning

In constructing a preconditioner many times small elements within the matrix are set to zero as it is thought their influence is negligible. Making use of a multiplication factor on the remaining elements hopes to emphasise the importance of larger elements within the matrix, thereby obtaining a solution with fewer iterations in BICGSTAB(L).

C3. Experimental results

Various size multiplication factors were used on different problems to determine the most suitable candidate to improve convergence. The preconditioning matrix is constructed from the Simply Sparse matrix by eliminating all but the largest, in terms of magnitude of the complex number, 1% of the elements. This sparse matrix is then multiplied by a factor, called the multiplication factor here. Convergence to the system of equations is said to have been achieved when the:

$$Norm(A * x - b) < 1e^{-8} \tag{C1}$$

Results carried out on three different problems is shown in tables C1, C2 and C3.

Table C1: Varying the Preconditioning Multiplier for a problem with 807 unknowns (Simulation stopped if iterations were greater than 10)

Multiplying factor	Iterations	Condition number of impedance matrix
100	>10	1380
1000	7	1380
10000	3	1380
100000	2	1380
1000000	2	1380
10000000	2	1380

Table C2: Varying the Preconditioning Multiplier for a problem with 2007 unknowns (Simulation stopped if iterations were greater than 50)

Multiplying factor	Iterations	Condition number of impedance matrix
10000	>50	2185
100000	21	2185
1000000	8	2185
10000000	2	2185

Table C3: Varying the Preconditioning Multiplier for a problem with 3607 unknowns (Simulations stopped if iterations were greater than 20)

Multiplying factor	Iterations	Condition number of impedance matrix
1000000	>20	4749
10000000	2	4749

C4. Conclusion

A number of different multiplication factors were tested to investigate which number would be suitable as a default value. In all cases rapid convergence, within 2 iterations, is achieved when the multiplication factor is set to 1×10^{7}, making it a suitable candidate as a default multiplication factor. Table C1 shows that rapid convergence may be achieved with a smaller multiplication factor but this is specific to one problem only.

Appendix D: Matrix Block Diagonal Inclusion Study

D1. Introduction

When constructing a preconditioner to an iterative method only enough information should be retained to allow rapid convergence. One such method is to include blocks from the Simply Sparse matrix that are close to the diagonal, referred to as diagonal block inclusion.

D2. Reasoning

In SuperNEC impedance matrices, the most important information is the self-impedance of wire elements (i.e. the impedance of the wire segment due to its interaction with itself). These elements are found on the diagonal of the matrix. Wire segments close to each other will have a stronger influence on each. These elements with a strong mutual impedance are typically located close to the diagonal of the impedance matrix. This study looks at the performance of a preconditioner with varying numbers of blocks included close to the diagonal from the Simply Sparse matrix.

This is illustrated in figure D1.

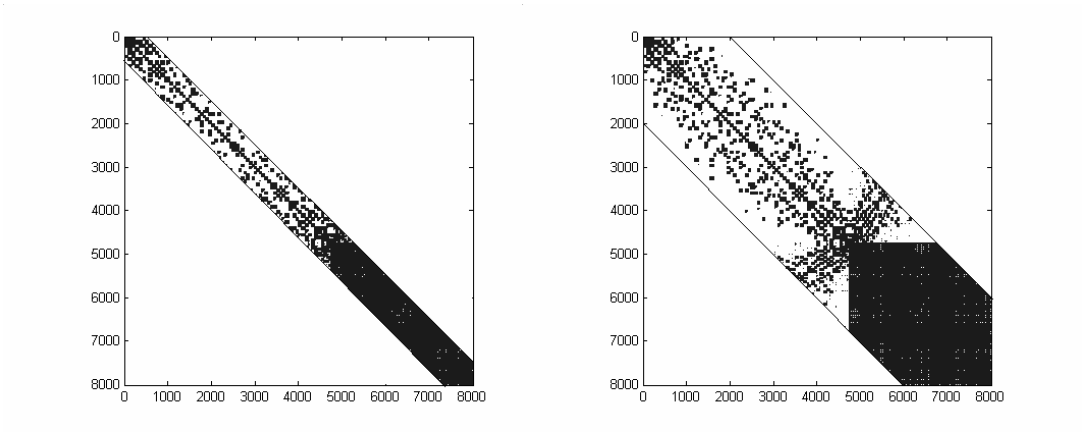


Figure D1: Graphic representation of a Block Diagonal Inclusion Method. 5 Blocks included on the left and 20 blocks on the right

D3. Experimental Results

In constructing a preconditioner, the matrix density can not be too large, as this would increase the time taken per iteration to invert it. Typically a matrix of the order of a few percent would be ideal. Table D1 shows a number of different problems with increasing number of blocks close to the diagonal. Iterations are halted if they exceed 200.

The stop criterion for this study was for the average tangential e-field error to be less than 0,001 V/m. The BICGSTAB method was used.

Table D1: Comparing the convergence rate of various problems by including more blocks on the preconditioning matrix.

Problem	Diagonal Blocks included	Tangential E-Field Error	Preconditioner Density (%)	Iterations
Large Truck	6	0,081	4,5	201
	8	0,0428	7,6	201
	10	0,0161	9,1	201
Helicopter	5	0,0009	7,91	62
	7	0,0009	10,5	164
	9	0,0021	11,58	201
Train	5	0,4757	1,84	201
	7	0,4449	2,77	201
	9	0,6785	3,49	201

D4. Conclusion

A preconditioner constructed with increasing number of elements close to the diagonal of the impedance matrix was constructed. The experimental results showed that convergence took place very slowly, if at all, despite increasing the number of elements within the preconditioner. This was applied to different problem types with the same results. The block inclusion method was therefore not considered a suitable preconditioner.

Appendix E: Interaction Distance Preconditioning with BICGSTAB(L)

E1. Introduction

When searching for an appropriate preconditioner for BICGSTAB(L), interaction distance was one of the methods considered. The interaction distance method looks to include only the influence of those wire segments that are within a certain distance of each other. It is reasoned that all other wire segments will interact weakly and can be ignored. For example, if the interaction distance is set to 0.5 wavelengths, then the preconditioning matrix will be filled with the values of the interactions between all segments in the structure that are 0.5 wavelengths apart or less. Interactions between segments that are further apart than 0.5 wavelengths are not included in the preconditioner.

E2. Experimental Results

When choosing an interaction distance it is important to choose a value that will include enough information, but create a preconditioning matrix that is still only a few percent sparse in order to invert it easily. The interaction distance shown in Table E2 was chosen to be 0,05 wavelengths. The matrices were more dense than required (in the order of 20%) which should have yielded fewer iterations. From chapter 4 we can see that setting L to 4 should give the best results. The stop criterion for this study was for the average tangential e-field error to be less than 0,001 V/m

Table E1: Experimental Results for interaction distance as a preconditioner to BICGSTAB(L)

Problem size	L	Iterations	Stop Criterion Tangential E-field Error
1203	1	33	0,001
2032	1	39	0,001
3473	1	33	0,001
6354	1	30	0,001
1203	2	54	0,001
2032	2	34	0,001
3473	2	20	0,001
4806	2	114	0,001
6354	2	46	0,001
1203	3	78	0,001
2032	3	90	0,001
3473	3	51	0,001
4806	3	183	0,001
6354	3	39	0,001
1203	4	84	0,001
2032	4	24	0,001
3473	4	56	0,001
4806	4	196	0,001
6354	4	32	0,001

E3. Conclusion

Table E1 shows that despite using relatively dense preconditioning matrices (of the order of 20%) the solutions to the problems converge slowly, i.e. 30 iterations or more. Each iteration is also slower than the preconditioner developed in this thesis as it has about ten times as many elements. Varying L does not seem to improve the convergence rate of any of the problems. Although the problems do converge, they do so more slowly than conventional methods. The interaction method was thus not considered a viable solution method.

Bibliography

Amestoy P.R, Duff I.S, L'Excellent J-Y, "MUMPS Multifrontal Massively Parallel Solver Version 2.0", Technical Report TR/PA/98/02, CERFACS, February 6, 1998

Anderson E., Bai Z, Bischof C, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Ostrouchov S, Sorensen D, "LAPACK Users' Guide, Second Edition", SIAM, Philadelphia, PA, 1995

Barrett R, Berry M, Chan T, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C, Van der Vorst H, "Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods", Society for Industrial and Applied Maths (SIAM) ISBN 0-89871-328-5, 1993. Also available:
http://www.netlib.org/linalg/html_templates/Templates.html

Blackford L.S, Choi J, Cleary A, Azevedo E.D, Demmel J, Dhillon I, Dongarra J, Hammarling S, Henry G, Petitet A, Stanley K,Walker D, Whayley R, "ScaLAPACK: A Linear Algebra Library for Message Passing Computers", Proceedings of SIAM Conference on Parallel Processing (1997)

Burke G.J, Poggio A.J, "Numerical Electromagnetic Code (NEC) –Method of Moments" tech doc 116, Naval Oceans Systems Centre, San Diego, CA, 1981

Canning F.X , Rogovin K, (2002)," Simply Sparse, A general Compression/Solution Method for MoM Programs.", Antennas and Propagation Society International Symposium, 2002. IEEE Volume 2, Issue , pp 234 - 237

Carr M., Bleszynski M., Volakis J.L., "A Near-Field Preconditioner and its Performance in Conjunction with the BiCGStab(ell) Solver", IEEE antennas & propagation magazine 2004, vol. 46, N°2, pp. 23-30

Chew, W.C, Jin, J, Lu, C, Michielssen, E, Song, M, "Fast Solution Methods in Electromagnetics", IEEE Transactions on Antennas and Propagation, Vol 45, No. 3, March 1997

Clark A.R, Fourie A.P.C, Nitch D.C, "Stationary, non-stationary and hybrid iterative Method of Moments solutions schemes", IEEE Transactions on Antennas and propagation, Vol. 49, No. 10, October 2001, pp 1462-1469

Dongarra J, Du Croz J, Duff I, Hammarling S, ”A Ser of level 3 Basic Linear Algebra Subprograms”, ACM Transactions on Mathematical Software1990, pp 1-17

Dongarra J, Du Croz J, Hammarling S, Hanson R, “Algorithm 656: An extended Set of Basic Linear Algebra Subprograms: Model Implementation and Test Programs” ACM Transactions on Mathematical Software1998, pp18-32

Dongarra J, Du Croz J, Hammarling S, Hanson R, “An Extended Set of FORTRAN Basic Linear Algebra Subprograms”, ACM Transactions on Mathematical Software1998, pp1-17

Dreyer, R.L, Clark, A.R, “Preliminary Results for Simply Sparse as a Preconditioner to SIM” Antennas and Propagation Society International Symposium, 2002. IEEE Volume 2, Issue , 2002 Page(s): 234 - 237

Duff I.S, Erisman A.M, Reid, J.K, “Direct Methods for Sparse Matrices”, Oxford Science Publications, (1986)

Fourie A.P.C, Nitch D.C., “Comparing the Spares Iterative Method (SIM) with the Banded Jacobi and Conjugate Gradient Techniques”, Antennas and Propagation Society International Symposium, 1994. AP-S. Digest Volume 2, Issue , 20-24 Jun 1994 pp1181 - 1184

Fourie A.P.C, Nitch, D.C, Clark, A.R, “A Sparse Iterative Method (SIM) for Method of Moments calculations”, Applied Computational Electromagnetics Society Journal March 1999, vol 14, pp 9-16

Fourie A.P.C, Nitch D.C, “A fast Sparse Iterative Method (SIM) for Method of Moments” Antennas and Propagation Society International Symposium, 1994. AP-S. Digest**,** Volume 2, pp 1146-1149.

Gutknecht M.H, “Variants of BICGSTAB for matrices with complex spectrum”, SIAM J. Sci. Compute September 1993, Vol 14, pp. 1020-1033

Hageman L.A, Young D.M, “Applied Iterative Methods”, Academic Press, 1981

Hanson R., Krogh F, Lawson C, “A Proposal for Standard Linear Algebra Subprograms”, ACM SIGNUM Newsletter, 8(16), 1973

Harrington R.F, “Field Computation by Moment Methods” Macmillan, New York, 1968

Kirby R.C , “A new look at expression templates for matrix computation”, Computing in Science & Engineering, May-June 2003, Volume5, Issue 3, pp 66-70.

Lawson, C, Hanson, R, Kincaid, D, Krogh, F, “Basic Linear Algebra Subprograms for FORTRAN usage”, ACM Transactions on Mathematical Software 1979, pp 308-323

Miller E.K et al., An Evaluation of Computer Program Using Integral Equations for the Electromagnetic Analysis of Thin Wire Structures, UCRL-75566, Lawrence Livermore Laboratory, CA, March 1974

Neureuther A.R et al., “A Comparison of Numerical Methods for Thin Wire Antennas,” Presented at the 1968 Fall URSI Meeting, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1968

Nitch D.C, “A serial and parallel design of NEC2 to demonstrate the advantages of the object-oriented paradigm in comparison with the procedural paradigm” PhD thesis, University of the Witwatersrand, Johannesburg, December 1992.

Poggio A.J and Adams R.W, Approximations for Terms Related to the Kernel in Thin-Wire Integral Equations, UCRL-51985, Lawrence Livermore Laboratory, CA, December 19, 1975

Poggio, A.J, Integral Representation for Fields Due to Sources on Open Surfaces with Applications to End Caps, UCRL-51723, Lawrence Livermore Laboratory, CA, December 16, 1974

Ralston A, A First Course in Numerical Analysis, McGraw-Hill, New York, 1965

Saad Y, (2000), “Iterative Methods for Sparse Linear Systems”, Second Edition, published electronically http://www-users.cs.umn.edu/~saad/books.html

Sarkar T, Siarkiewicz K, Stratton R, “Survey of Numerical Methods for Solution of Large System of Linear Equations for Electromagnetic Field Problems”, IEEE Transactions on Antennas and Propagation November 1981, Vol AP-29

Sleijpen G, (1993), “BICGSTAB(L) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum”, Electronic Transactions on Numerical Analysis Volume 1. pp. 11-32

“SuperNEC GUI Input User Reference Manual”, Version 2.9, Poynting Software (Pty) Ltd, South Africa.

“SuperNEC MoM technical Reference Manual”, Version 2.9, Poynting Software (Pty) Ltd, South Africa.

The Mathworks, Inc., “Getting Started with Matlab”, http://www.mathworks.com/access/helpdesk/help/techdoc/learn_matlab/learn_matlab.shtml

Van der Vorst H.A, “Iterative Krylov Methods for Large Linear Systems”, Cambridge University Press, 2003

Weiland T, “TBCI and URMEL - new Computer Codes for Wake Field and Cavity Mode Calculations”, IEEE Transactions on Nuclear Science, August 1983, Volume 30, Issue 4, pp 2489-2491

Whaley R, Petitet A, Dongarra J, (2000), “Automatic Empirical Optimization of Software and the ATLAS project” published electronically
http://www.cs.utsa.edu/~whaley/papers/lawn147.ps

Wu T.T and King R.W.P “The Tapered Antenna and Its Application to the Junction Problem for Thin Wires,” IEEE Trans. Ant. And Prop., AP-24, No.1, pp.42-45, January 1976

Yeh Y.S and Mei K.K, “Theory of Conical Equiangular Spiral Antennas,” Part I - Numerical Techniques, IEEE Trans. Ant. And Prop., AP-15, 1967, p.634