# Efficient Techniques for Inductance Extraction of Complex 3-D Geometries

by

Mattan Kamon

B.S. Engineering Science, Pennsylvania State University (1991)
M.A. Mathematics, Pennsylvania State University (1991)

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

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994



Author ..................................................................
Department of Electrical Engineering and Computer Science
January 28, 1994

Certified by ..................................................................
Jacob K. White
Associate Professor, Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ..................................................................
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

**Efficient Techniques for Inductance Extraction of Complex 3-D Geometries**

by

Mattan Kamon

Submitted to the Department of Electrical Engineering and Computer Science
on January 28, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science

## Abstract

This thesis describes the efficient computation of frequency dependent inductances and resistances for complex three-dimensional geometries of conductors. A mesh analysis equation formulation technique combined with a multipole-accelerated Generalized Minimal Residual (GMRES) matrix solution algorithm is used to compute the inductance and resistance matrices in nearly order $n$ time and memory where $n$ is the number of volume-filaments. Previous approaches have required order $n^3$ time and order $n^2$ memory. Results from examples are given to demonstrate that this multipole-accelerated algorithm can reduce required computation time and memory by more than an order of magnitude for realistic integrated circuit packaging problems.

Thesis Supervisor: Jacob K. White
Title: Associate Professor, Department of Electrical Engineering and Computer Science

## Acknowledgments

First and foremost, I'd like to thank my research advisor, Jacob White, for his guidance and enthusiasm. He is always patient and willing to spend the time to hear any of my questions or ideas.

I would also like to thank officemate Joel Phillips for many very specific and useful discussions and also as someone at whom I can just throw ideas.

For the "how will it be used" questions, Michael Tsuk has been very helpful and willing to answer real application questions.

Much of the research group cannot go without thanks. For instance, Songmin Kim for his discussions on integral formulations, Keith Nabors for multipole questions, and Ignacio McQuirk for his practical examples. Then there are our many "systems" people who have been willing to do such things as replace an operating system on a moment's notice to help me meet a deadline.

This work was supported by the Defense Advanced Research Projects Agency contract N00014-91-J-1698, the National Science Foundation contract (MIP-8858764 A02), a National Science Foundation fellowship, and grants from I.B.M. and Digital Equipment Corporation.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

A wide range of integrated circuit and packaging design problems require accurate estimates of the coupling inductances of complicated three-dimensional structures. The frequencies of interest generally require magnetoquasistatic analysis, and the most commonly used approach is to apply finite-difference or finite-element techniques to a differential problem formulation. However, finite-element techniques require that the entire 3-D volume be discretized, and for complex structures, the generation of such a volume discretization can become cumbersome and require prohibitive execution time. Instead, volume-element methods can be applied to solving integral formulations of the problem in which case only the interior of the conductors need be discretized. Unfortunately, volume-element methods generate dense matrix problems which, if solved directly, grow in computational cost like $n^3$ and in memory like $n^2$, where $n$ is the number of elements into which the problem is discretized.

This thesis describes a method for magnetoquasistatic analysis of complicated three-dimensional packages and interconnect. The method uses a standard volume-element discretization of an integral formulation from magnetoquasistatic analysis also known as the Partial Element Equivalent Circuit (PEEC) method [16]. In this work, however, the discretized equations are reformulated using a circuit analysis technique known as mesh analysis. The mesh formulation leads to a dense system of equations which is solved iteratively using a rapidly converging Krylov-subspace method known as Generalized Minimal Residual (GMRES). Finally, since the system of equations is dense, the matrix-vector product required at each iteration of GMRES is expensive and to reduce its cost, a multipole-accelerated algorithm is used. The combination of these techniques yields a packaging analysis program, named FastHenry, whose computational complexity and memory requirements grow linearly with the number of volume-elements required to discretize the conductors.

The next chapter gives background details about computing inductances using the PEEC method. It begins with the integral formulation derived from Maxwell's equations under the magnetoquasistatic assumption and continues with a description of the discretization of conductor volumes. The final section takes these ideas and discusses the use of nodal analysis which has been used in the past to give a system of equations which must be solved to extract the desired inductance parameters. Chapter 3 discusses the mesh formulation and its advantages over nodal analysis for solution by iterative methods. Chapter 4 describes the multipole-acceleration of the matrix vector products needed at each iteration of GMRES. A significant portion of this thesis has been the development of preconditioners to accelerate GMRES convergence as discussed in Chapter 5. Chapter 6 demonstrates the use of FastHenry and gives accuracy and efficiency results for a number of examples and conclusions are given in Chapter 7. Appendix A gives a brief description of the implementation in FastHenry and Appendix B is the User's Manual distributed with the FastHenry code.

# Chapter 2

# Background Formulation

Inductance extraction is the process of computing the complex frequency-dependent impedance matrix of a multiterminal system, such as an electrical package, under the magnetoquasistatic approximation [9]. For a problem of $k$ terminal pairs, let $Z_r(\omega) \in \mathbf{C}^{k\times k}$ denote this impedance matrix at frequency $\omega$. Then,

$$Z_r(\omega)\tilde{I}_s(\omega) = \tilde{V}_s(\omega), \tag{2.1}$$

where $\tilde{I}_s, \tilde{V}_s \in C^n$ are vectors of the terminal current and voltage phasors, respectively [3]. Note that column $i$ of of $Z_r$ can be computed by setting entry $i$ of $\tilde{I}_s$ to one, the rest to zero, and then computing the resulting voltage vector $\tilde{V}_s$. The $i^{th}$ column of $Z_r$ is then given by $\tilde{V}_s$.

For illustration, consider a geometry consisting of two input-output terminal pairs as shown in Fig. 2-1. For this problem,

$$Z_r(\omega) = R_r(\omega) + j\omega L_r(\omega) = \begin{bmatrix} R_{11}(\omega) + j\omega L_{11}(\omega) & R_{12}(\omega) + j\omega L_{12}(\omega) \\ R_{21}(\omega) + j\omega L_{21}(\omega) & R_{22}(\omega) + j\omega L_{22}(\omega) \end{bmatrix} \tag{2.2}$$

where $R_r$ is referred to as the resistance matrix, and $L_r$ the inductance matrix. Also, $L_{11}$ and $L_{22}$ are the self-inductances of the conductors, and $L_{12} = L_{21}$ is the mutual inductance.

As described above, column one of $Z_r$ can be computed by computing the voltages $\tilde{V}_{s1}$ and $\tilde{V}_{s2}$ resulting from setting $\tilde{I}_{s1} = 1$ and $\tilde{I}_{s2} = 0$. Note that instead, $Y_r = Z_r^{-1}$ could be computed in a similar manner by setting $\tilde{V}_s$ and computing $\tilde{I}_s$. The rest of this chapter describes the integral formulation and standard discretization used to compute the resultant voltages, $\tilde{V}_s$, required to compute $Z_r$, or currents, $\tilde{I}_s$, to compute $Y_r$.

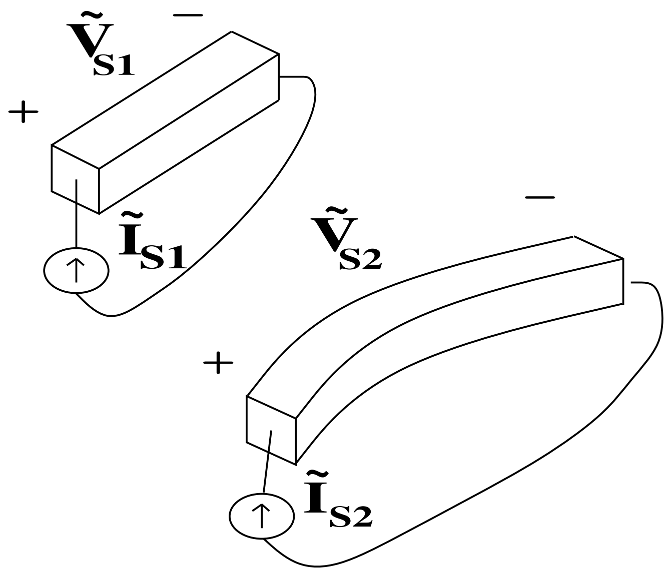


Figure 2-1: Two conductors, each with an input and output terminal.

## 2.1 Integral Equation

Several integral equation-based approaches have been used to derive the $Z_r$ associated with a given package or interconnect structure [15, 1]. These integral formulations are derived by assuming sinusoidal steady-state, and then applying the magnetoquasistatic assumption that the displacement current, $\epsilon\omega\boldsymbol{E}$, is negligible everywhere. Given this, begin with Maxwell's Equations in sinusoidal steady state,

$$\nabla \times \boldsymbol{E} = -jw\mu\boldsymbol{H} \tag{2.3}$$

$$\nabla \times \boldsymbol{H} = jw\epsilon\boldsymbol{E} + \boldsymbol{J} \tag{2.4}$$

$$\nabla \cdot (\epsilon\boldsymbol{E}) = \rho \tag{2.5}$$

$$\nabla \cdot (\mu\boldsymbol{H}) = 0 \tag{2.6}$$

where $\omega$ is the angular frequency. In addition, within the conductors, by Ohm's law,

$$\boldsymbol{J} = \sigma\boldsymbol{E} \tag{2.7}$$

where $\sigma$ is the conductivity. As stated above, the frequencies of interest will be considered small enough such that the displacement current, $j\omega\epsilon\boldsymbol{E}$, can be neglected in (2.4). This assumption is clearly justified within the conductors where the conductivity is large. Under this quasistatic assumption, the divergence of (2.4) gives current conservation,

$$\nabla \cdot \boldsymbol{J} = 0. \tag{2.8}$$

Note that for this work, we wish to allow for ideal current sources, and thus (2.8) becomes

$$\nabla \cdot \boldsymbol{J} = I_s(\boldsymbol{r}). \tag{2.9}$$

From this point, we wish to eliminate the field quantities, $\boldsymbol{E}$ and $\boldsymbol{H}$, in favor of the current density, $\boldsymbol{J}$, and applied voltages. From Gauss' Law of magnetic flux, (2.6), the magnetic flux can be written as

$$\mu \boldsymbol{H} = \nabla \times \boldsymbol{A} \tag{2.10}$$

where $\boldsymbol{A}$ is the vector potential. Applying this to (2.3),

$$\nabla \times (\boldsymbol{E} + jw\boldsymbol{A}) = 0. \tag{2.11}$$

This implies that there exists a scalar function, $\Phi$, such that

$$-\nabla\Phi = \boldsymbol{E} + jw\boldsymbol{A} \tag{2.12}$$

where $\Phi$ will be called the scalar potential. We require one final relation to relate the vector potential, $\boldsymbol{A}$ to the current density, $\boldsymbol{J}$. By use of (2.10) and by choosing the Coulomb gauge,

$$\nabla \cdot \boldsymbol{A} = 0. \tag{2.13}$$

Equation (2.4) then becomes

$$-\nabla^2 \boldsymbol{A} = \mu \boldsymbol{J} \tag{2.14}$$

and thus

$$\boldsymbol{A}(\boldsymbol{r}) = \frac{\mu}{4\pi} \int_{V'} \frac{\boldsymbol{J}(\boldsymbol{r}')}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dv' \tag{2.15}$$

where $V'$ is the volume of all conductors.

Substituting (2.15) and (2.7), into (2.12) yields the following integral equation:

$$\frac{\boldsymbol{J}(\boldsymbol{r})}{\sigma} + \frac{jw\mu}{4\pi} \int_{V'} \frac{\boldsymbol{J}(\boldsymbol{r}')}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dv' = -\nabla\Phi(\boldsymbol{r}). \tag{2.16}$$

Using equation (2.16) plus current conservation, (2.9), the conductor current density, $\boldsymbol{J}$, and scalar potential, $\Phi$, can be computed [1].

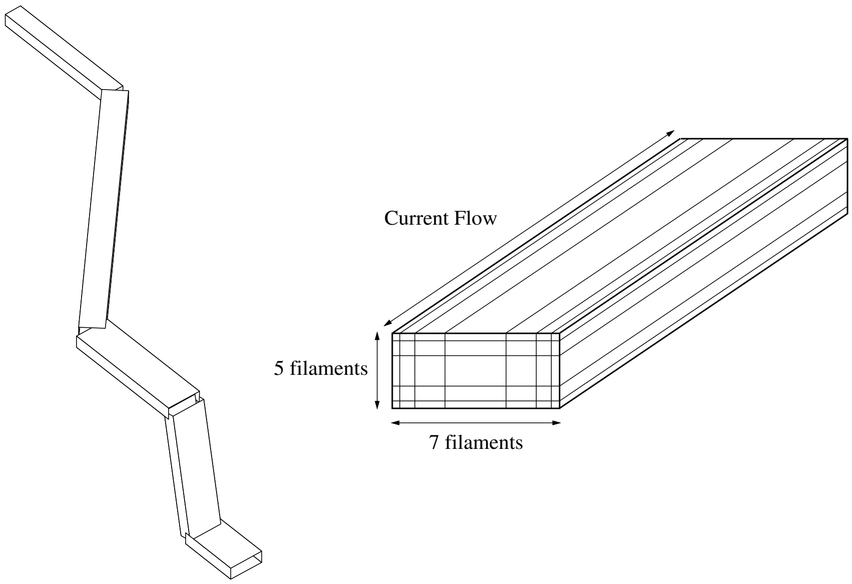


Figure 2-2: Single pin of a pin-connect divided into 5 sections, each of which is a bundle of 35 filaments.

### 2.1.1 Discretization

Given the magnetoquasistatic assumption, the current within a long thin conductor can be assumed to flow parallel to its surface, as there is no charge accumulation on the surface. Thus, for long thin structures such as pins of a package or connector, the conductor can be divided into *filaments* of rectangular cross-section inside which the current is assumed to flow along the length of the filament. In order to properly capture skin and proximity effects in these long, thin conductors, the cross section of the conductor can be divided into a bundle of parallel filaments as shown in Fig. 2-2. It is also possible to use the filament approach for planar structures, such as ground planes, where the current distribution is two-dimensional. In such cases, a grid of filaments can be used, as in Fig. 2-3. Once the conductors are discretized into filaments, the interconnection of the current filaments can be represented with a planar graph, where the $n$ nodes in the graph are associated with connection points between filaments, and the $b$ branches in the graph represent the filaments into which each conductor segment is discretized. An example graph, or circuit, for a single conductor example is shown in Fig. 2-4.

If the current density inside each filament is assumed to be constant, then the approximation to the unknown current distribution can then be written as

$$\boldsymbol{J}(\boldsymbol{r}) \approx \sum_{i=1}^{b} I_i w_i(\boldsymbol{r}) \mathbf{l_i} \tag{2.17}$$

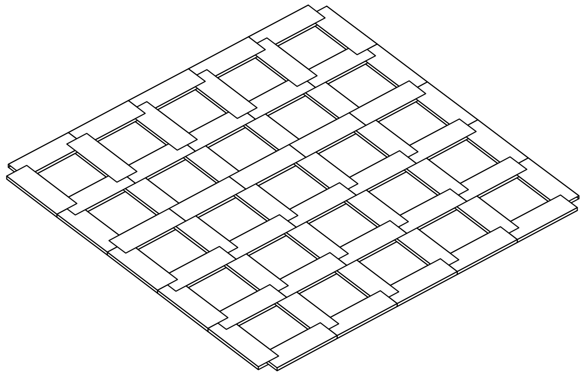

Figure 2-3: Discretization of a Ground Plane. Segments are drawn one-third actual width.

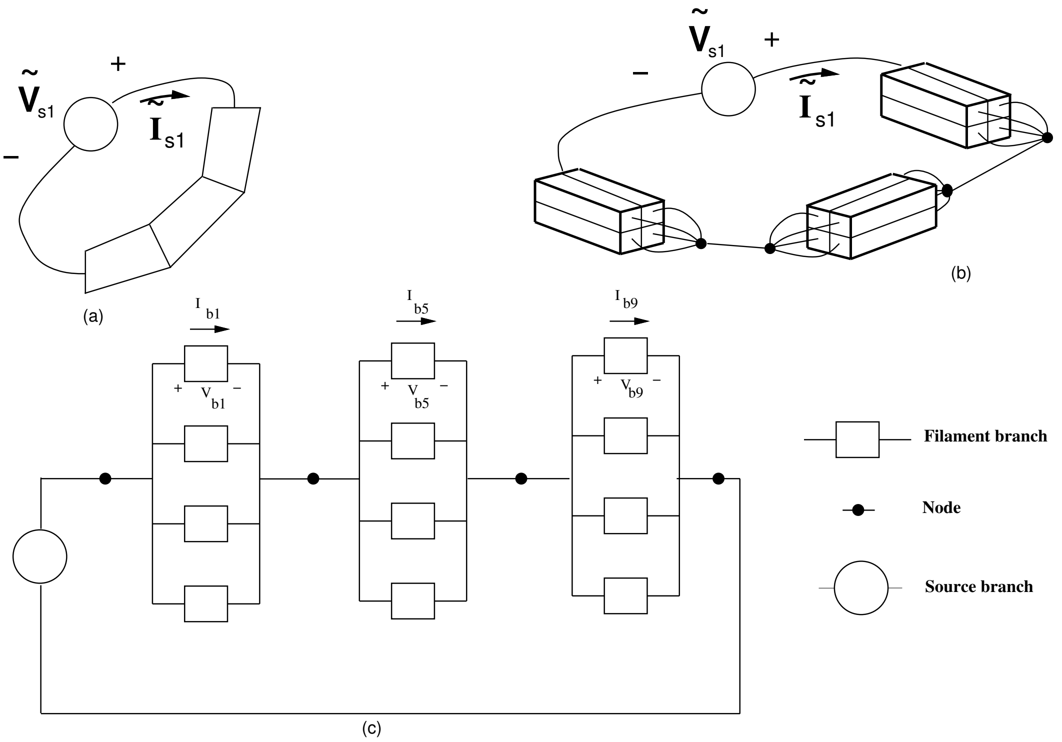


Figure 2-4: One conductor, (a) as piecewise-straight sections, (b) discretized into filaments, (c) modelled as a circuit.

where $I_i$ is the current inside filament $i$, $\mathbf{l_i}$ is a unit vector along the length of the filament and $w_i(\boldsymbol{r})$ is the weighting function which has a value of zero outside filament $i$, and $1/a_i$ inside, where $a_i$ is the cross sectional area. By defining the inner product of two vector functions, $\mathbf{a}$ and $\mathbf{b}$, by

$$(\mathbf{a}, \mathbf{b}) = \int_V \mathbf{a} \cdot \mathbf{b} \, dv \tag{2.18}$$

and following the method of moments [8], a system of $b$ equations can be generated by taking the inner product of each of the weighting functions with the vector integral equation, (2.16). This gives

$$\left(\frac{l_i}{\sigma a_i}\right) I_i + j\omega \sum_{j=1}^{b} \left( \frac{\mu}{4\pi a_i a_j} \int_{V_i} \int_{V_j'} \frac{\mathbf{l_i} \cdot \mathbf{l_j}}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dV' \, dV \right) I_j = \frac{1}{a_i} \int_{a_i} (\Phi_A - \Phi_B) \, dA \tag{2.19}$$

where $l_i$ is the length of filament $i$, $a_i$ is the cross section, $\Phi_A$ and $\Phi_B$ are the potentials on the filament end faces, and $V_i$ and $V_j'$ are the volumes of filaments $i$ and $j$, respectively. Note that the right hand side of (2.19) results from integrating $\nabla\Phi$ along the length of the filament, and is effectively the average potential on face $A$ minus the average on face $B$.

In matrix form, (2.19) becomes

$$(R + j\omega L) I_b = \tilde{\Phi}_A - \tilde{\Phi}_B \tag{2.20}$$

where $I_b \in \mathbf{C}^b$ is the vector of $b$ filament currents,

$$R_{ii} = \frac{l_i}{\sigma a_i} \tag{2.21}$$

is the $b \times b$ diagonal matrix of filament dc resistances,

$$L_{ij} = \frac{\mu}{4\pi a_i a_j} \int_{V_i} \int_{V_j'} \frac{\mathbf{l_i} \cdot \mathbf{l_j}}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dV' \, dV \tag{2.22}$$

is the $b \times b$ dense, symmetric positive definite matrix of partial inductances, and $\tilde{\Phi}_A$ and $\tilde{\Phi}_B$ are the averages of the potentials over the cross sections of the filament faces. Equation (2.20) can also be written as

$$Z I_b = V_b \tag{2.23}$$

where $Z = R + j\omega L \in \mathbf{C}^{b \times b}$ is called the branch impedance matrix and $V_b = \tilde{\Phi}_A - \tilde{\Phi}_B$ is the vector of branch voltages.

Note that one can view, for instance, filament $i$ as a resistor with resistance $R_{ii}$ in series with an inductor with self-inductance $L_{ii}$ and mutual inductance $L_{ij}$ with filament $j$. The circuit obtained from the graph described above is known as the Partial Element Equivalent Circuit [19, 15].

It is worth noting that in the example discretization shown in Fig. 2-4b and c, the ends of adjacent bundles of filaments are effectively shorted together at each node. This approximation is acceptable when the conductor is long and thin and thus the transverse current is negligible. When the conductor is not long and thin, it may be more appropriate to join the bundles together with a small grid of short filaments or even by a full 3-D discretization of filaments in the same manner that the ground plane in Fig. 2-3 is a 2-D discretization.

### 2.1.2 Nodal Analysis Formulation

Current conservation, (2.9), must be enforced at each of $n$ nodes where filaments connect. This can be written as

$$AI_b = I_s \tag{2.24}$$

where $A \in \mathbf{R}^{n \times b}$ is the branch incidence matrix and $I_s$ is the mostly zero vector of source currents at the node locations. Each row in $A$ corresponds to a filament connection node, and each column to a filament current. Column $i$ in $A$ has two nonzero entries: $-1$ in the row corresponding to the node from which filament $i$'s current leaves, and $+1$ in the row corresponding to the node into which filament $i$'s current enters.

Since $\nabla^2 \Phi = 0$, the branch voltages, $V_b$, can be derived from a set of node voltages, denoted $\tilde{\Phi}_n$, as in

$$A^t \tilde{\Phi}_n = V_b. \tag{2.25}$$

Combining (2.23), (2.24), and (2.25) yields

$$AZ^{-1}A^t \tilde{\Phi}_n = I_s. \tag{2.26}$$

Notice that column $i$ of $Z_r$ can now be computed by appropriately setting the source currents, $I_s$, that correspond to $\tilde{I}_{s_i}$ equal to one (unit current through conductor $i$), and then solving (2.26) to compute the node voltages, $\tilde{\Phi}_n$. The difference of appropriate node voltages gives the entries of $\tilde{V}_s$, the vector of voltages across each of the conductors.

In most programs, the dense matrix problem in (2.26) is solved with some form of Gaussian elimination or direct factorization. These programs avoid forming $Z^{-1}$ explicitly by reformulating (2.26) into the sparse tableau form [17],

$$\begin{bmatrix} Z & -A^t \\ A & 0 \end{bmatrix} \begin{bmatrix} I_b \\ \tilde{\Phi}_n \end{bmatrix} = \begin{bmatrix} 0 \\ I_s \end{bmatrix}. \tag{2.27}$$

Using direct factorization to solve (2.27) implies that the calculation grows at least as $b^3$, where again $b$ is the number of current filaments into which the system of conductors is discretized. For complicated packaging structures, $b$ can exceed ten thousand, and solving (2.27) with direct factorization will take days, even using a high performance scientific workstation.

# Chapter 3

# The Mesh-Based Approach

The obvious approach to trying to reduce the cost of solving (2.27) is to apply iterative methods. However, such methods converge slowly because (2.27) contains equations of two different types. Another approach is to reformulate the equations using mesh analysis, and then apply an iterative method.

This chapter first describes the reformulation using mesh analysis and then discusses the use of a Krylov-subspace iterative method known as GMRES (Generalized Minimal Residual). Finally, the eigenspectra for the systems generated in (2.27) are compared to those generated from mesh analysis as insight into the effectiveness of GMRES for solving both system.

## 3.1 Mesh Analysis

In mesh analysis [3], a mesh is any loop of branches in the graph which does not enclose any other branches. Also, the currents flowing around any mesh in the network are the unknowns, rather than node voltages. Mesh analysis is easiest to describe if it is assumed that sources generate explicit branches in the graph representing the discretized problem. Kirchoff's voltage law, which implies that the sum of branch voltages around each mesh in the network must be zero, is represented by

$$MV_b = V_s \tag{3.1}$$

where $V_b$ is the vector of voltages across each branch except for the source branches, $V_s$ is the mostly zero vector of source branch voltages, and $M \in \mathbf{R}^{m\times b}$ is the mesh matrix, where

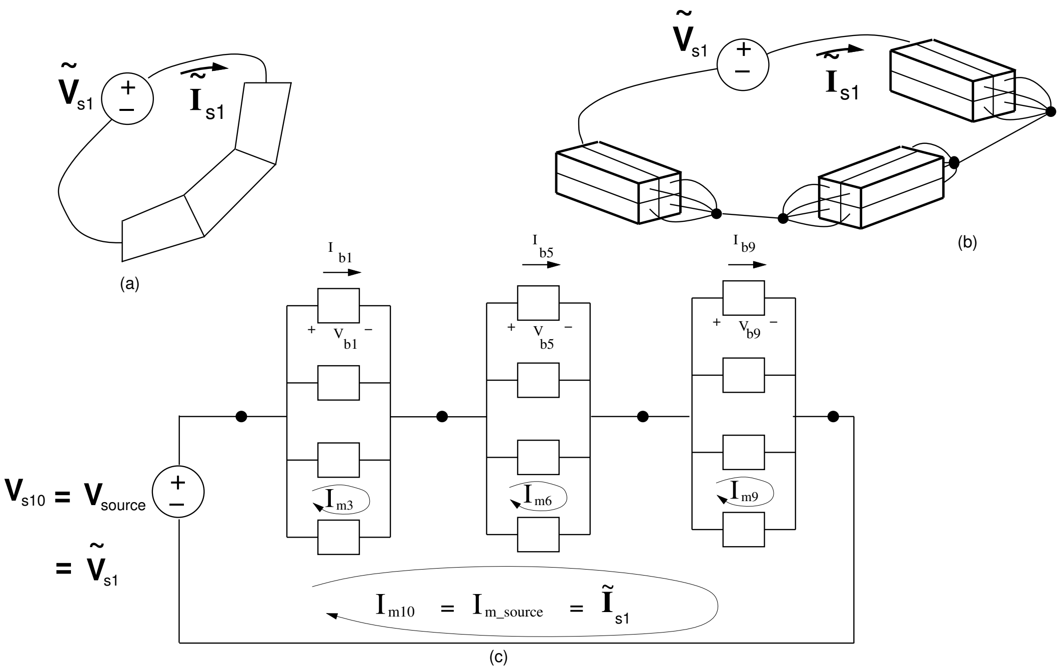


Figure 3-1: One conductor, (a) as piecewise-straight sections, (b) discretized into filaments, (c) modelled as a circuit.

$m = b' - n + 1$ is the number of meshes and $b'$ is the number of filaments branches plus the number of source branches.

The relationship between branch currents and branch voltages given in (2.23) still holds, and the mesh currents, that is, the currents around each mesh loop, satisfy

$$M^t I_m = I_b, \tag{3.2}$$

where $I_m \in \mathbf{C}^m$ is the vector of mesh currents. Note that each of the entries in the terminal current vector, $\tilde{I}_s$ from (2.1), will be identically equal to some entry in $I_m$. And similarly, each of the entries in the terminal voltage vector, $\tilde{V}_s$, will correspond to some entry in $V_s$. Fig. 3-1 illustrates the definitions of the above quantities for a single conductor example.

Combining (3.2) with (3.1) and (2.23) yields

$$MZM^t I_m = V_s. \tag{3.3}$$

The matrix $MZM^t$ is *easily* constructed directly [3]. To compute the $i^{th}$ column of the reduced admittance matrix, $Y_r = Z_r^{-1}$, solve (3.3) with a $V_s$ whose only nonzero entry corresponds to $\tilde{V}_{s_i}$, and then extract the entries of $I_m$ associated with the source branches.

*Algorithm 3.2.1 (GMRES Algorithm for $Ax = b$).*

guess $\boldsymbol{x}^0$
for $k = 0, 1, \ldots$ until converged {
   Compute the error, $\boldsymbol{r}^k = \boldsymbol{b} - A\boldsymbol{x}^k$
   Find $\boldsymbol{x}^{k+1}$ to minimize $\boldsymbol{r}^{k+1}$
      based on $\boldsymbol{x}^i$ and $\boldsymbol{r}^i$, $i = 0, \ldots, k$
}

## 3.2 Using an iterative solver

The standard approach to solving the complex linear system in (3.3) is Gaussian elimination, but the cost is $m^3$ operations. For this reason, inductance extraction of packages requiring more than a few thousand filaments is considered computationally intractable. To improve the situation, consider using a conjugate-residual style iterative method like GMRES [18]. Such methods have the general form given in Algorithm 3.2.1.

Note that the GMRES algorithm can be directly applied to solving (3.3) because the matrix $MZM^t$ is easily constructed explicitly. This is not the case for the nodal formulation, (2.26), as either the $Z$ matrix must first be inverted or the sparse tableau form in (2.27) must be used. The sparse tableau form is disadvantageous because it is a much larger system of equations and it is difficult to solve iteratively as described in the next section.

When applying the GMRES algorithm to solving (3.3), the cost of each iteration of the GMRES algorithm is at least order $m^2$ operations. This follows from the fact that evaluating $\boldsymbol{r}^k$ implies computing a matrix-vector product, where in this case the matrix is $MZM^t$ and is dense. Note also that forming $MZM^t$ explicitly requires order $m^2$ storage. As will be discussed in Chapter 4, multipole acceleration can be used to reduce the matrix-vector product cost and the required storage requirements to order $b$.

## 3.3 Nodal versus Mesh Analysis

The rate of convergence of GMRES for solving $Ax = b$ can be related to the eigenspectrum of the matrix $A$. From this fact, the convergence properties of nodal analysis versus mesh analysis can be compared by observing the eigenspectra of the matrices produced by each of these formulations.

It is known that at the $i^{th}$ iteration, the GMRES algorithm produces a residual $r^i =$

$b - Ax^i$ which satisfies

$$\|r^i\|_2 = \min_{p \in P_i} \|p^i(A) r^0\|_2 \tag{3.4}$$

where $P_i$ is the set of all $i^{th}$ degree polynomials, $p$, such that $p(0) = 1$ and $\|\cdot\|_2$ is the Euclidean norm [18]. If $A$ is diagonalizable then

$$A = V \Lambda V^{-1} \tag{3.5}$$

where $\Lambda$ is a diagonal matrix whose entries, $\lambda_1, \lambda_2, \ldots$, are the eigenvalues of $A$ and $V$ is the matrix whose columns are the eigenvectors of $A$. From substitution into (3.4),

$$\frac{\|r^i\|_2}{\|r^0\|_2} \leq \kappa(V) \min_{p \in P_i} \|p(\Lambda)\|_2, \tag{3.6}$$

$$\leq \kappa(V) \min_{p \in P_i} \max_{\lambda_i} |p(\lambda_i)| \tag{3.7}$$

where

$$\kappa(V) \equiv \|V\|_2 \; \|V^{-1}\|_2 = \frac{\lambda_{max}(V)}{\lambda_{min}(V)} \tag{3.8}$$

is the condition number of the matrix $V$ and $\lambda_{max}(V)$ and $\lambda_{min}(V)$ are the maximum and minimum eigenvalues of $V$, respectively.

Note that $\kappa(V) = 1$ if $A$ is normal, i.e. $AA^H = A^H A$ where $A^H$ is the conjugate-transpose of $A$. Thus from (3.7), we see that for normal $A$, the error at iteration $i$ of GMRES is dependent on how well an $i^{th}$ degree polynomial can fit the eigenvalues of $A$. From this, consider some insights for matrices with real eigenvalues:

- If the eigenvalues are spread over a large interval, convergence will be slow, while clustered eigenvalues lead to faster convergence.
- Due to the constraint that $p(0) = 1$, matrices with eigenspectra which have both negative and positive clusters of eigenvalues will converge slower than those with eigenvalues on only one side of the origin.
- Also because $p(0) = 1$, eigenvalues close to the origin slow convergence.

Consider now the spectra resulting from the nodal formulation, (2.27), and the mesh formulation, (3.3), for the printed circuit board example described later in Fig. 5-8. The printed circuit board is two thin metal sheets sandwiching 255 small copper lines.

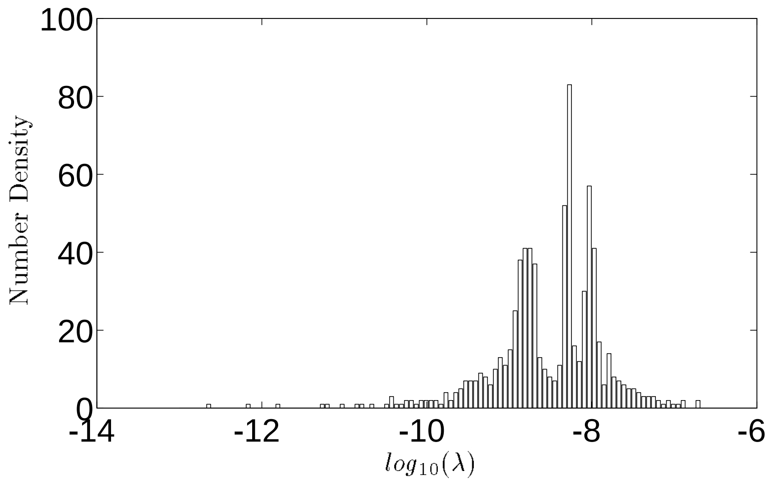


Figure 3-2: Eigenvalue spectrum of $MLM^t$ for a coarse discretization of the printed circuit board example

For inductance extraction problems, the high frequency ($\omega \to \infty$) limit gives the worst case convergence so consider a coarse discretization of this structure yielding a $751 \times 751$ $MLM^t$ matrix with a condition number $\kappa \approx 10^6$. Clearly, $MLM^t$ is always positive definite since $L$ is positive definite and is normal since $L$ is symmetric. Also, from the spectrum of $MLM^t$ shown in Fig. 3-2, one can see that, while most of the eigenvalues are in the interval $[10^{-10}, 10^{-7}]$, the remaining isolated eigenvalues are located toward the origin in the interval $[10^{-13}, 10^{-10}]$. Such small eigenvalues are not easily cancelled with the polynomial produced by the Krylov-subspace methods, so they will significantly slow convergence.

The high frequency limit for (2.27) can be written as

$$S = \begin{bmatrix} L & A^t \\ A & 0 \end{bmatrix} \tag{3.9}$$

where $S$ is now a real symmetric matrix. The eigenspectrum, $\lambda(S)$, is shown in Fig. 3-3. While there are fewer eigenvalues close to zero, a large portion of eigenvalues are negative. Note that for $S$, since each entry in $L \approx 10^{-9}$, the values of $L$ were scaled by $5 \times 10^8$ for the spectrum in Fig. 3-3 and the GMRES comparison of Fig. 3-4 described below.

Both $MLM^t$ and $S$ have an eigenstructure which does not yield fast convergence with GMRES as shown in Fig. 3-4. In this case, the mesh formulation, $MLM^t$, has no advantage over the nodal formulation, $S$. Significant improvement is possible, however, through *preconditioning* which involves solving an equivalent system with a more favorable eigen-

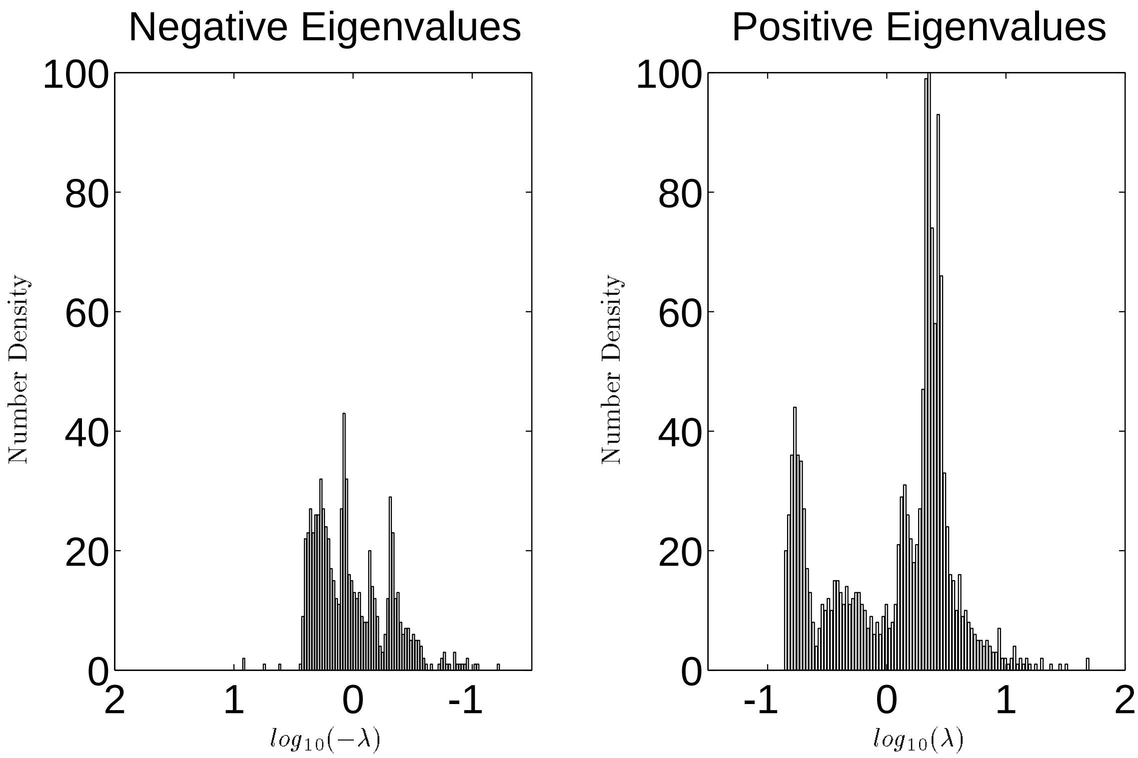


Figure 3-3: Eigenvalue spectrum of $S$, the sparse tableau formulation, for a coarse discretization of the printed circuit board example

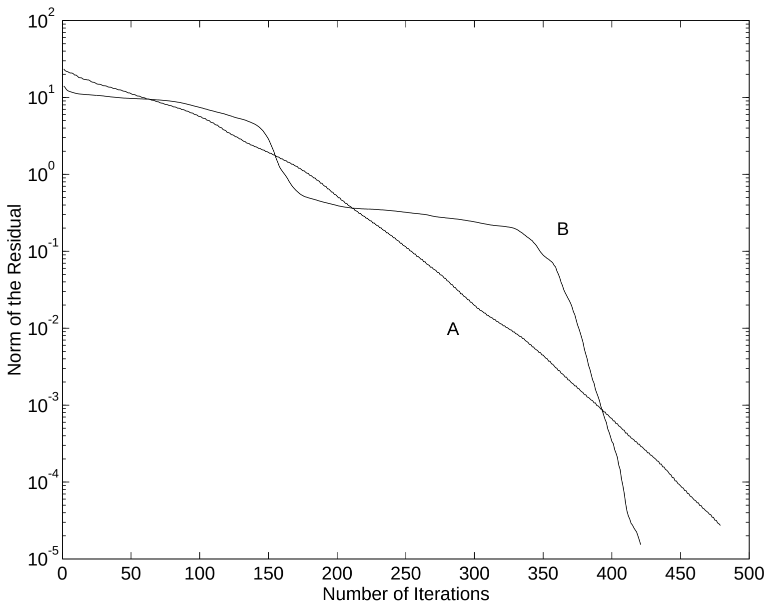


Figure 3-4: Convergence history of (A), the Sparse Tableau Formulation, and (B), the Mesh Formulation for a coarse discretization of the printed circuit board example.

structure. Preconditioning $MLM^t$ is discussed in Chapter 5 and the rest of this section discusses the difficulties in preconditioning the $S$ matrix.

In general, the GMRES iterative method applied to solving $Cx = b$ can be significantly accelerated by preconditioning if there is an easily computed good approximation to the inverse of $C$. We denote the approximation to $C^{-1}$ by $P$, in which case preconditioning the GMRES algorithm is equivalent to using GMRES to solve

$$CPy = b \tag{3.10}$$

for the unknown vector $y$. The solution is then computed with $x = Py$. Clearly, if $P$ is precisely $C^{-1}$, then (3.10) is trivial to solve, but then $P$ will be very expensive to compute.

Preconditioning the $S$ matrix is very difficult since it contains equations of two different types: those resulting from $V_b = ZI_b$, and those from current conservation, $AI_b = I_s$. While it is possible to approximate the inverse of $Z$, it is difficult to account for the effect of the $A$ matrix. In some sense, the $A$ matrix and its position in (2.27) are in fact responsible for the many negative eigenvalues. As the following theorems show, for the ideal case of $L = I$, where $I$ is the identity, $S$ has exactly $n$ negative eigenvalues, where $n$ is the number of rows of $A$. Note that $L = I$ corresponds to the low frequency limit of a system of conductors discretized into identical filaments.

*Theorem 3.3.1.* Let $C = \begin{bmatrix} I & B \\ A & 0 \end{bmatrix}$ and assume $AB$ is nonsingular and diagonalizable, where $I$ is the bxb identity matrix, $B \in \mathbf{R}^{bxn}$, $A \in \mathbf{R}^{nxb}$ and $n < b$.

Then $b-n$ eigenvalues of $C$ are one, and the other $2n$ eigenvalues of $C$ satisfy $\lambda(\lambda-1) = \lambda'$ where $\lambda'$ is an eigenvalue of $AB$.

*Proof.* If $AB$ is diagonalizable, for each eigenvector of $AB$ with eigenvalue $\lambda'$, there exist two eigenvectors of $C$ with associated eigenvalues satisfying $\lambda(\lambda-1) = \lambda'$. To show this, let $x_2$ be an eigenvector of $AB$ and let $\lambda$ be such that $\lambda(\lambda-1) = \lambda'$. Then, $ABx_2 = \lambda(\lambda-1)x_2$. Since $AB$ is nonsingular, $\lambda \neq 1$, and letting $x_1 = Bx_2/(\lambda-1)$ gives

$$\begin{array}{rcl} Ix_1 \quad + \quad Bx_2 & = & \lambda x_1 \\ Ax_1 \qquad\qquad & = & \lambda x_2 \end{array} \tag{3.11}$$

and thus $\lambda \in \lambda(C)$ with eigenvector $x^t = [x_1^t \; x_2^t]$. $2n$ eigenvalue-eigenvector pairs are thus given by $\lambda = (1 \pm \sqrt{1+4\lambda'})/2$ and $x^t = [(Bx_2)^t/(\lambda-1) \quad x_2^t]$

Next, suppose $\lambda \in \lambda(C)$, but $\lambda \neq 1$ with eigenvector $x^t = [x_1^t \; x_2^t]$ where $x$ is different from the previous $2n$ eigenvectors. Since $Bx_2 = (\lambda-1)x_1$ we get

$$ABx_2 = \lambda(\lambda-1)x_2. \tag{3.12}$$

This is impossible since then $x_1$ must be an eigenvector of $AB$ and $x$ an eigenvector of one of the previous $2n$ eigenvalues. □

*Theorem 3.3.2.* The matrix $S$ with $Z = I$ has $n$ negative eigenvalues and $b$ positive eigenvalues.

*Proof.* Since $AA^t$ is positive definite, Thm 3.3.1 gives that the $n$ negative eigenvalues are given by

$$\lambda = (1 - \sqrt{1+4\lambda'})/2. \tag{3.13}$$

where $\lambda' \in \lambda(AA^t)$. Similarly, another $n$ are positive, and the remaining $b-n$ are equal to one. □

Even if $Z^{-1}$ could be computed exactly, preconditioning using only $Z^{-1}$ as in

$$P = \begin{bmatrix} L^{-1} & 0 \\ 0 & I \end{bmatrix} \tag{3.14}$$

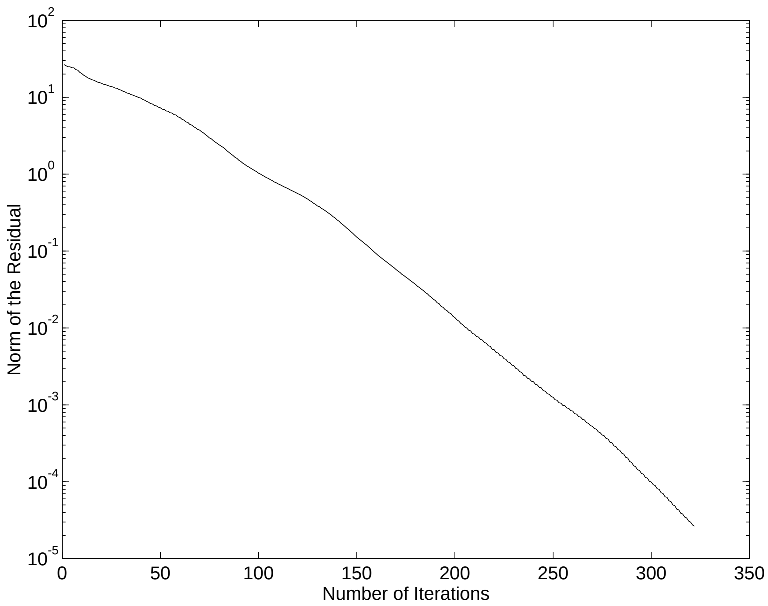


Figure 3-5: Convergence history for the Sparse Tableau Formulation using $L^{-1}$ as a preconditioner

for the high frequency limit, does not improve the performance of GMRES in part due to the fact that there are still $n$ negative eigenvalues. This can be shown by a similar argument as in Theorem 3.3.2 with

$$SP = \begin{bmatrix} I & A^t \\ AL^{-1} & 0 \end{bmatrix}. \tag{3.15}$$

Note that $AL^{-1}A^t$ is positive definite since $L$ is positive definite.

In addition, $SP$ is no longer normal, and for the printed circuit board example above, $\kappa(V) \approx 10^4$ where $V$ is the eigenvector matrix of $SP$. The matrix, in fact, does not show much improvement in convergence rate over the unpreconditioned system as shown in Fig. 3-5.

The $MZM^t$ matrix does not suffer from the above difficulties because it has equations of one type, and for the high frequency case, $MLM^t$ is positive definite. For this reason, this thesis focuses on using the Mesh Formulation for inductance extraction. In Chapter 5, various techniques are explored for preconditioning $MLM^t$ to significantly accelerate GMRES convergence.

# Chapter 4

# The Multipole Approach

As discussed in Section 3.2, the dominant cost of each iteration of GMRES results from computing the matrix-vector product, $(MZM^t)I_m^k$ where $I_m^k$ is the approximated solution at the $k^{th}$ iteration. This operation requires order $m^2$ operations and order $m^2$ storage to store the $MZM^t$ matrix. It is possible to approximately compute $MZM^tI_m^k$ in order $b$ operations using a hierarchical multipole algorithm for electrostatic analysis [6]. Such algorithms also avoid explicitly forming $MZM^t$, and so reduce the memory required to order $b$. This chapter describes how a multipole algorithm can be used for inductance extraction under the mesh formulation.

## 4.1 The Electrostatic Analogy

To show how a multipole algorithm can be applied to computing $MZM^tI_m^k$, consider expanding the matrix-vector product by separating $Z$ into its real and imaginary parts,

$$MZM^tI_m^k = MRM^tI_m^k + j\omega MLM^tI_m^k. \tag{4.1}$$

The $MRM^tI_m^k$ term can be computed in order $m$ operations because $R$ is the diagonal matrix derived from the filament resistances, and $M$ is the sparse mesh matrix with order $m$ nonzero elements. Forming $MLM^tI_m^k$ is more expensive, requiring order $m^2$ operations as $L$ is dense. From (2.19) and (2.22) it is clear that entry $i$ of the portion of the product, $LM^tI_m^k$, or equivalently $LI_b$, is

$$(LI_b)_i = \sum_{j=1}^{b}\left(\frac{\mu}{4\pi a_i a_j}\int_{V_i}\int_{V_j'}\frac{\mathbf{l_i}\cdot\mathbf{l_j}}{\|\boldsymbol{r}-\boldsymbol{r}'\|}dV'\,dV\right)I_j. \tag{4.2}$$

In terms of the vector potential,

$$(LI_b)_i = \frac{1}{a_i} \int_{V_i} \boldsymbol{A}(\boldsymbol{r}) \cdot \mathbf{l_i} dV. \tag{4.3}$$

Equation 4.3 is verified since substituting (2.17) in (2.15) gives

$$\boldsymbol{A}(\boldsymbol{r}) = \frac{\mu}{4\pi} \sum_{j=1}^{b} \left( \int_{V_j'} \frac{\mathbf{l_j}}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dV' \right) \frac{I_j}{a_j}. \tag{4.4}$$

The above decomposition shows that $LI_b$ can be evaluated by integrating the vector potential, $\boldsymbol{A}$, over each filament [12]. Also, from (4.4), each component of the vector potential can be considered a scalar electrostatic potential generated by a collection of charges. That is, for $p \in \{1, 2, 3\}$, the $p^{th}$ component of $\boldsymbol{A}(\boldsymbol{r})$, denoted $\psi_p(\boldsymbol{r}) \in \mathbf{C}$, is a scalar potential given by

$$\psi_p(\boldsymbol{r}) = \frac{\mu}{4\pi} \sum_{j=1}^{b} \left( \int_{V_j'} \frac{(\mathbf{l_j})_p}{\|\boldsymbol{r} - \boldsymbol{r}'\|} dV' \right) \frac{I_j}{a_j}. \tag{4.5}$$

and therefore $(I_j/a_j)(\mathbf{l_j})_p$ can be interpreted as a charge density due to filament $j$.

The electrostatic analogy implies that $LI_b$ can be computed by combining the results of evaluating the electrostatic potential along $b$ filaments due to $b$ filament charges *for three separate sets of filament charges.* It is the evaluation of these electrostatic potentials which can be accelerated with the hierarchical multipole algorithm [6]. That is, the electrostatic potential due to $b$ charges can be evaluated at $b$ points in order $b$ operations using the hierarchical multipole algorithm. Therefore, by using the multipole algorithm three times, $LI_b$ can be computed in order $b$ operations.

## 4.2 The Hierarchical Multipole Algorithm

A complete description of the fast multipole algorithm is quite lengthy, and can be found in [6], or in the context of 3-D capacitance extraction, in [13, 14]. To see roughly what the algorithm exploits to achieve its efficiency consider the two configurations given in Figs. 4-1 and 4-2, depicted in 2-D for simplicity. In either figure, the obvious approach to determining the electrostatic potential at the $n_1$ evaluation points from the $n_2$ point-charges involves $n_1 * n_2$ operations: at each of the $n_1$ evaluation points one simply sums the contribution to the potential from $n_2$ charges.

An accurate approximation for the potentials for the case of Fig. 4-1 can be computed in far fewer operations using *multipole expansions*, which exploit the fact that $r >> R$

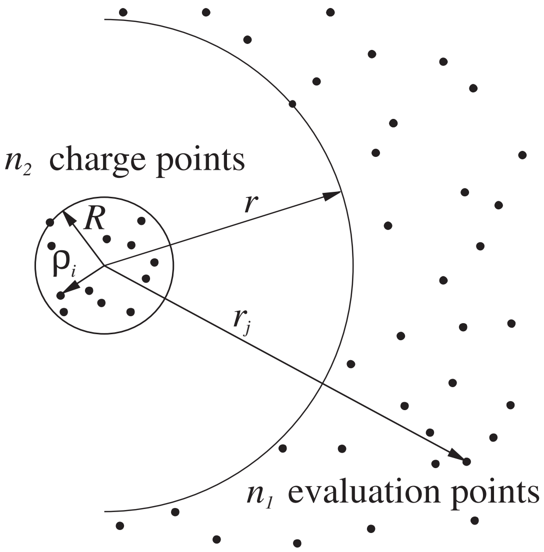


Figure 4-1: The evaluation point potentials are approximated with a multipole expansion.

(defined in Fig. 4-1). That is, the details of the distribution of the charges in the inner circle of radius $R$ in Fig. 4-1 do not strongly affect the potentials at the evaluation points outside the outer circle of radius $r$.

It is also possible to compute an accurate approximation for the potentials at the evaluation points in the inner circle of Fig. 4-2 in far fewer than $n_1 * n_2$ operations using *local expansions*, which again exploit the fact that $r >> R$ (as in Fig. 4-2). In this second case, what can be ignored are the details of the evaluation point distribution.

This brief description of the hierarchical multipole algorithm is only intended to make clear that the algorithm's efficiency stems from coalescing charges and evaluation points using multipole and local expansions. A few points about the algorithm's application to computing $LI_b$ should be considered however. When filaments are very near each other, that is, $R \approx r$, a multipole expansion representation would lead to excessive error, so the interaction is evaluated directly using (2.22). Direct evaluations are also used for small groups of distant filaments when the computation required to build the multipole and local expansions exceeds the direct evaluation cost, thus making the algorithm adaptive. Therefore, when the hierarchical multipole algorithm is used to compute $LI_b$, the evaluation is typically a combination of three sets of multipole and local expansion evaluations for the three components of the vector potential, along with a single set of nearby-filament direct evaluations.

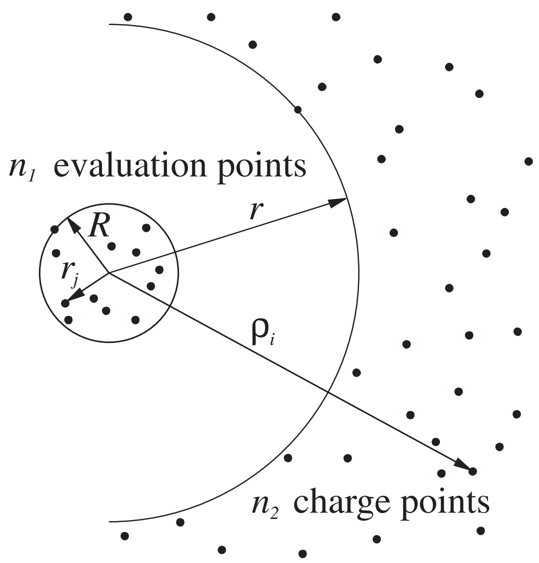


Figure 4-2: The evaluation point potentials are approximated with a local expansion.

# Chapter 5

# Accelerating Iteration Convergence

In Section 3.3 it was observed that the GMRES iterative method converged slowly when applied to (3.3). In general, however, the GMRES iterative method applied to solving (3.3) can be significantly accelerated by *preconditioning* if there is an easily computed good approximation to the inverse of $MZM^t$. We denote the approximation to $(MZM^t)^{-1}$ by $P$, in which case preconditioning the GMRES algorithm is equivalent to using GMRES to solve

$$(MZM^t)Px = V_s \tag{5.1}$$

for the unknown vector $x$. The mesh currents are then computed with $I_m = Px$. Clearly, if $P$ is precisely $(MZM^t)^{-1}$, then (5.1) is trivial to solve, but then $P$ will be very expensive to compute. This chapter describes the efforts of this work to develop preconditioners for $MZM^t$.

## 5.1 Local Inversion

An easily computed good approximation to $(MZM^t)^{-1}$ can be constructed by noting that the most tightly coupled meshes are ones which are physically close. To exploit this observation, for each mesh $i$, the submatrix of $MZM^t$ corresponding to all meshes near mesh $i$ is inverted directly. Then, the row of the inverted submatrix associated with mesh $i$ becomes the $i^{th}$ row of $P$. This is illustrated in Fig. 5-1, where the submatrix is drawn as a block

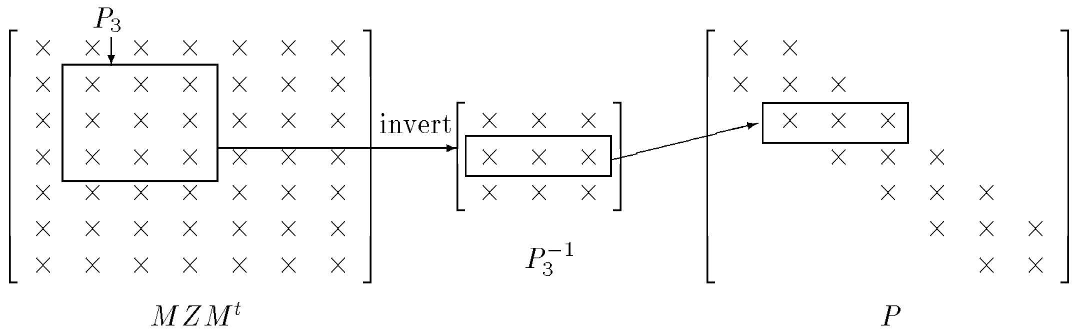


Figure 5-1: The steps leading to the third row of the preconditioner $P$ ("$\times$" denotes a non-zero element). Note that for illustration, $P_3$ is drawn as a block along the diagonal.

along the diagonal for illustration. We refer to this preconditioner as a "local-inversion" preconditioner, because it is formed by inverting physically localized problems.

This preconditioner works well for pin-connect and other similar structures for which most of the meshes are small and thus what is 'local' is obvious. The fact that most of the meshes are small can be observed from from Fig. 3-1 by noticing that most of the meshes, such as those associated with $I_{m3}$, $I_{m6}$, and $I_{m9}$, are small and consist of only two physically close filaments. Comparatively, there are relatively few large meshes, such as $I_{m10}$, each which result from the presence of an external source and include many filaments. The many filaments which are included in each external source mesh span much of the physical problem domain and therefore much of the problem can be physically close to these large meshes. For this reason, the large meshes associated with sources cannot be included in the preconditioner, otherwise excessively large subproblems would be inverted directly. Since there are relatively few of these large meshes in a pin-connect structure and they are physically separate (only one per pin), not including the large meshes when forming the preconditioner does not significantly slow convergence.

For large ground-plane problems, with possibly hundreds of external sources, the performance of local-inversion is severely degraded. As for pin-connects, many of the meshes are small; in this case, most meshes include four filaments (See Fig. 2-3). Each external source, however, requires the formation of a large mesh traversing the ground plane between its two contact points as shown in Fig. 5-2. If there are hundreds of these meshes, in which case many of them will be physically close or even possibly partially overlapping, local-inversion becomes ineffective since it cannot include large meshes.

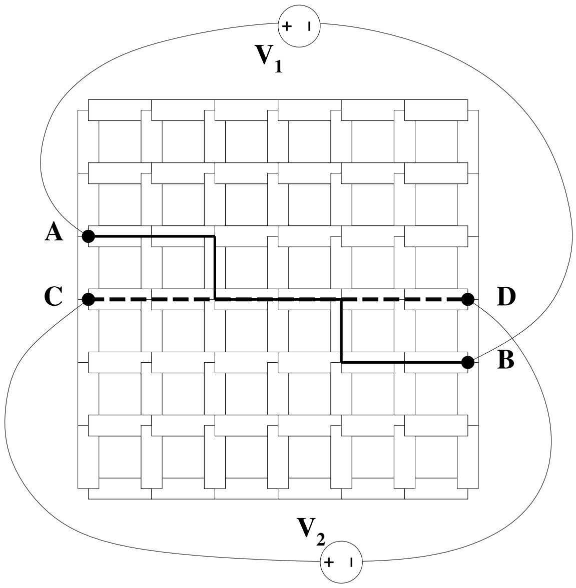


Figure 5-2: Two ground plane meshes due to external sources. One source is connected between points $A$ and $B$ and the other between $C$ and $D$.

## 5.2 Sparsified Preconditioners

Other approaches to preconditioning which might help account for large mesh interactions involve somehow approximately factoring $MZM^t$ or directly factoring an approximation to $MZM^t$. The idea is then to first compute a sparse factorization

$$P^{-1} = LU \tag{5.2}$$

and then at each iteration, $Px$ will be computed by solving $LUy = x$ with forward and backward substitution.

Approximately factoring $MZM^t$ using an approach like incomplete LU factorization is ineffective however, because the diagonals of $MZM^t$ are not necessarily greater than the sum of the off-diagonals and therefore ignored terms can become more significant. Another approach is to "sparsify" $MZM^t$, possibly based on the sparsity pattern of $MRM^t$, and then directly factor the "sparsified" matrix to construct a preconditioner.

Fig. 5-3 shows the results of applying the local-inversion preconditioner, the sparsity-based preconditioner, plus an example of the sparsified-L class of preconditioners to be discussed in the next section. The $MZM^t$ matrix is $751 \times 751$ and corresponds to the printed-circuit board example of Fig. 5-8 in the high frequency limit, that is, as $\omega \to \infty$. The high frequency case is chosen because it has been found to demonstrate the worst case convergence for the sparsified preconditioners, as discussed in the next section. A point worth noting here is that for these sparsified preconditioners in the low frequency limit, $\omega \to 0$, we have that $P^{-1}$, $MZM^t \to MRM^t$ and therefore $(MZM^t)P \to I$. The local-inversion preconditioner shows approximately the same convergence behavior at low and high frequency.

From Fig. 5-3 it is apparent that local-inversion and sparsity-based preconditioning only slightly accelerated convergence over no preconditioning. Note that this example includes approximately 300 large meshes.

## 5.3 Positive definite sparsifications of L

To develop a better preconditioner, instead of sparsifying $MZM^t$ as described above, consider sparsifying the partial inductance matrix, $L$, and then generating the preconditioner by directly factoring the sparse result, $P^{-1} = M(R + j\omega L_s)M^t$, where $L_s$ is the sparsified partial inductance matrix. We call this class of preconditioners "sparsified-L." Note that for the low frequency limit, the whole problem would be factored exactly since $R$ is diagonal. This section shows that this type of preconditioner is effective for a large class of problems and that $L_s$ must be chosen symmetric positive definite for this type of preconditioner to be effective.

To motivate the discussion of this section, consider the following two choices for $L_s$: the sparsest approach to choosing $L_s$ would be to take the diagonal of $L$, or consider sparsifying $L$ based on the magnitude of the elements by zeroing all terms except those that satisfy $L_{ij}^2 > \epsilon |L_{ii}L_{jj}|$, for some $\epsilon$. Fig. 5-4 compares this threshold sparsification to preconditioning choosing $L_s$ as the diagonal of $L$. Clearly, choosing $L_s$ to be the diagonal of $L$ produced the better preconditioner, yet many more terms were included in $L_s$ for the threshold sparsification.

### 5.3.1 The High Frequency Limit

As $\omega \to \infty$, the preconditioned matrix reduces to $(MLM^t)(ML_sM^t)^{-1}$. In what follows, it will be shown that $L_s$ should be chosen to be symmetric positive definite for this sparsified-L class of preconditioners to be effective.

*Lemma 5.3.1.* The product of real symmetric positive definite matrices has positive eigenvalues.

*Proof.* Let $A$ and $B$ be real symmetric positive definite matrices, then $A^{-1/2}$ and $B^{-1/2}$ exist and are also symmetric positive definite. $AB$ has the same eigenvalues as

$$D = A^{-1/2} A B A^{1/2} = A^{1/2} B A^{1/2} = (B^{1/2} A^{1/2})^t (B^{1/2} A^{1/2}) = C^t C. \tag{5.3}$$

Then, for any vector $x$, $x^t D x = (Cx)^t (Cx) = y^t y > 0$, where $y = Cx$. Therefore, $D$ has positive eigenvalues. □

*Theorem 5.3.2.* If $L_s$ is symmetric positive definite, then the preconditioned system, $(MLM^t)(ML_sM^t)^{-1}$ has positive eigenvalues.

*Proof.* For any $x$, let $y = M^t x$. Then $x^t(MLM^t)x = y^t L y > 0$ since $L$ is positive definite. Following a similar argument for $ML_sM^t$ and using Lemma 5.3.1, the theorem is proved. □

Consider again the example of $L_s$ as the threshold sparsification of $L$, that is, form $L_s$ by zeroing all terms except those that satisfy $L_{ij}^2 > \epsilon |L_{ii} L_{jj}|$, for some $\epsilon$. In this case, $L_s$ is not necessarily positive definite. Fig. 5-4 compared this preconditioner for $\epsilon = 0.1$ to the preconditioner formed by taking $L_s$ to be only the diagonal of $L$, which is obviously positive definite. Clearly, using the threshold sparsification preconditioner results in slower convergence than using the diagonal-of-L preconditioner. This can be explained by examining the spectra of the preconditioned matrices in Fig. 5-5. For both cases, the eigenvalues seem similarly clustered, except the threshold preconditioned matrix has a large cluster of negative eigenvalues while the diagonal-of-L preconditioned matrix is positive definite.

Theorem 5.3.2 leads to the result that the condition number of preconditioned system in the high frequency limit is bounded *independent* of the mesh matrix, $M$.

*Theorem 5.3.3.* If $L_s$ is positive definite, then

$$\kappa[(MLM^t)(ML_sM^t)^{-1}] \leq \kappa(LL_s^{-1})$$

where, for a matrix with positive eigenvalues $A$, the condition number $\kappa(A)$ is defined as $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ , $\lambda_{\min}(A)$ are the maximum and minimum eigenvalues, respectively.

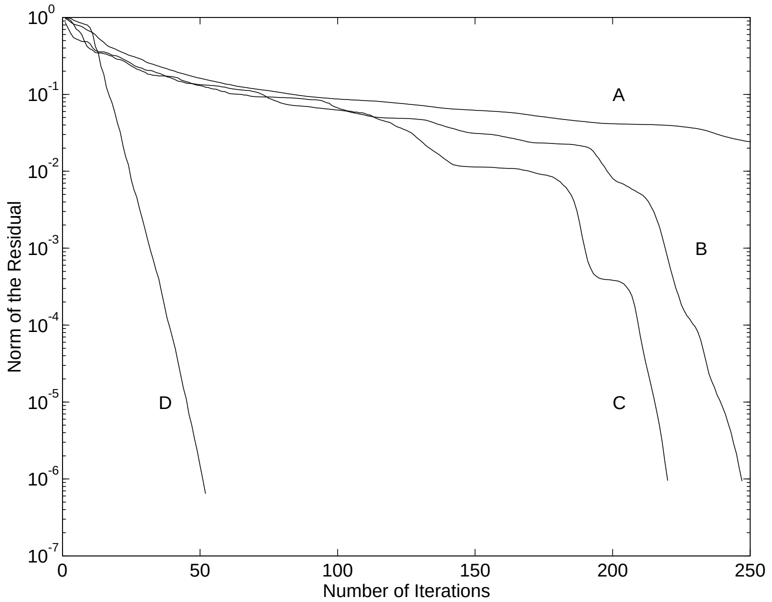


Figure 5-3: Convergence of GMRES applied to PCB example with no preconditioning (A), sparsity-based preconditioning (B), local-inversion preconditioning (C), and sparsified-$L$ preconditioning using the diagonal of $L$ (D)

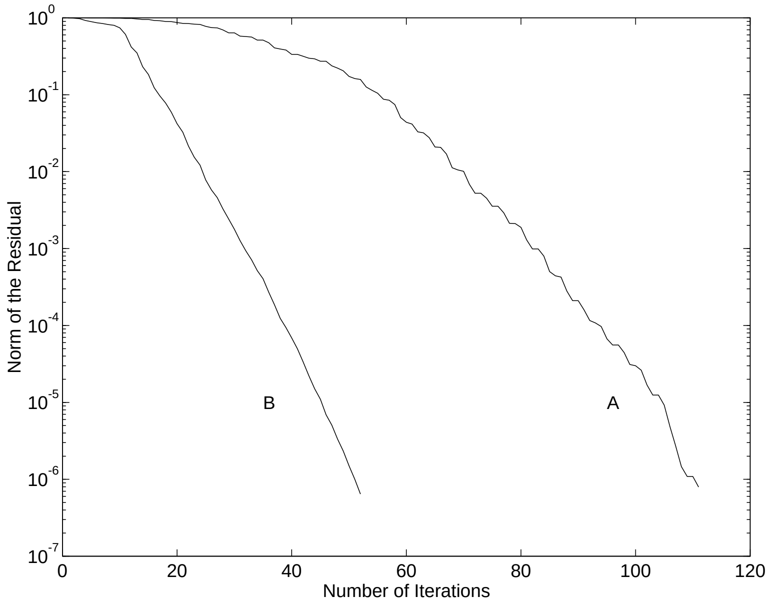


Figure 5-4: Convergence of GMRES applied to PCB example with threshold preconditioning for $\epsilon = 10^{-1}$ (A), and diagonal-of-$L$ preconditioning (B)

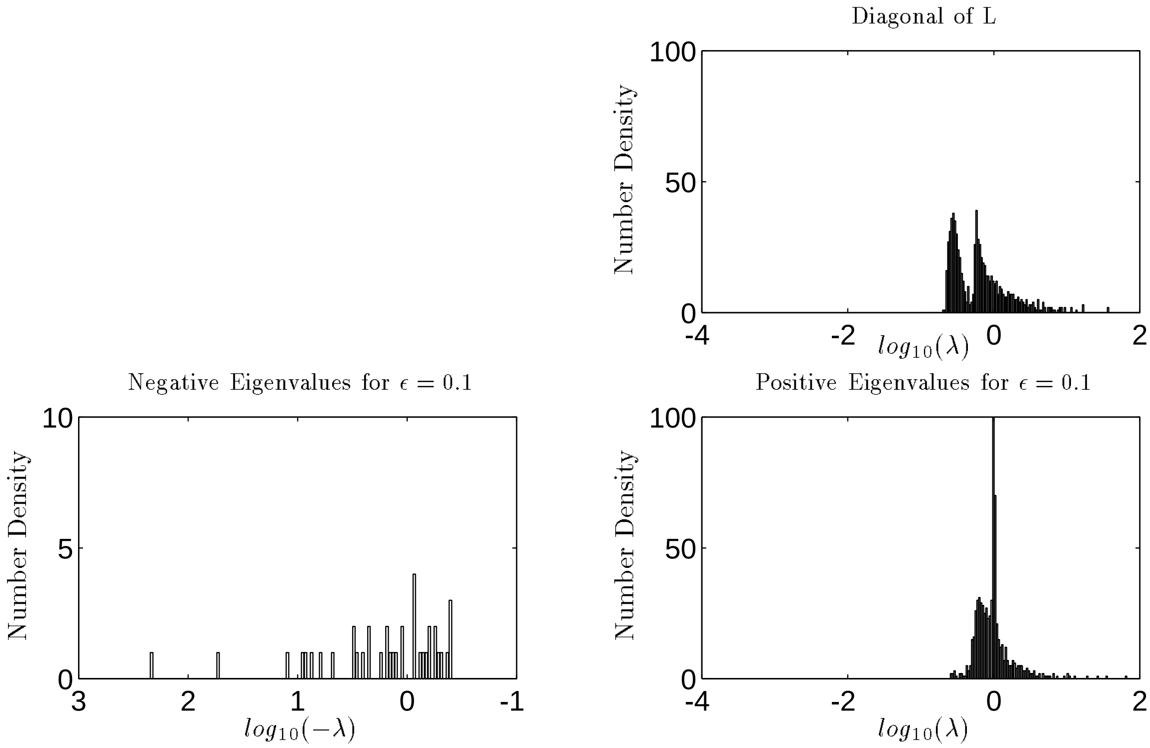


Figure 5-5: Eigenvalues for the positive definite diagonal-of-$L$ preconditioned system and the indefinite threshold preconditioned system for $\epsilon = 0.1$.

*Proof.* For the matrix $A = MLM^t$ with preconditioner $P^{-1} = ML_sM^t$, and with the maximum eigenvalue $\lambda_{\max}(AP)$, there is some $y$ such that $APy = \lambda_{\max}y$. Then for $x = Py$, the generalized eigenvalue problem $MLM^tx = \lambda_{\max}ML_sM^tx$ is satisfied. Therefore,

$$\lambda_{\max}(AP) = \frac{x^tMLM^tx}{x^tML_sM^tx}$$

and so there is also a vector $u = M^tx$ such that

$$\lambda_{\max}(AP) = \frac{u^tLu}{u^tL_su}$$

and so

$$\lambda_{\max}(AP) \leq \max_u \frac{u^tLu}{u^tL_su}$$

By a similar argument,

$$\lambda_{\min}(AP) \geq \min_v \frac{v^tLv}{v^tL_sv}$$

and thus

$$\kappa[MLM^t(ML_sM^t)^{-1}] \leq \kappa(LL_s^{-1})$$

□

The above two theorems lead to the conclusion that one should focus on choosing positive definite $L_s$ matrices. As described above, the sparsest approach would be to take the diagonal of $L$. Another approach is to divide physical space into disjoint regions and then to include in $L_s$ only the principal submatrices of $L$ corresponding to the groups of filaments contained inside each region. Thus, a filament will be included in exactly one region and by appropriately numbering the branches, $L_s$ can be written as a block diagonal matrix. An immediate approach is for each region, or block, to consist of the set of parallel filaments in a single section of conductor. Each block will then be no larger than the number of filaments in a section. For the simple one conductor example shown in Fig. 3-1, $L_s$ would consist of three $4 \times 4$ blocks. Another choice is to uniformly divide space into cubes and have each block consists of filaments within a cube. This cube-block method is easy to implement since the cube information is needed for implementation of the multipole algorithm. Note also that the cube-block preconditioner is denser than the section-block or diagonal-of-L preconditioners.

*Theorem 5.3.4.* $L_s$ for the cube-block and section-block preconditioners is positive definite.

*Proof.* The set of eigenvalues of a block diagonal matrix is the union of the sets of eigenvalues from each block. Since $L$ is symmetric positive definite, so are all of its principal submatrices (See, for instance [11], p. 397). Given the block diagonals of $L_s$ are principal submatrices of $L$, the theorem is proved. $\square$

### 5.3.2 The General Case

Under certain conditions the bound on GMRES convergence in the limit as $\omega \to \infty$ holds for all $\omega$.

*Theorem 5.3.5.* Given a problem discretized with filaments of the same size, and assuming that the GMRES algorithm uses the diagonal-$L$ preconditioner, the residual at iteration $k$, $r^k = b - \tilde{Z}_m(\omega)x^k$, where $\tilde{Z}_m(w)$ is the preconditioned $Z_m = MZM^t$, satisfies

$$\frac{||r^k||}{||r^0||} \leq 2\left[\frac{\sqrt{\kappa}-1}{\sqrt{\kappa+1}}\right]^k \quad , \tag{5.4}$$

where $\kappa = \kappa(LL_s^{-1}) = \kappa(L)$, independent of frequency. The following observation and short lemma will be used to prove Theorem 5.3.5.

*Observation 5.3.6.* If all filaments are the same size, the matrices $R$ and $L$ are constant along the main diagonal. The preconditioner $P$ constructed from the main diagonal of $L$ is $P^{-1} = (r + j\omega l)MM^t$, where $r$ and $l$ are the diagonal elements of $R$ and $L$ respectively.

*Lemma 5.3.7.* Given $Z_m = M(rI + j\omega L)M^t$, and $P^{-1} = M(r + i\omega l)M^t$, the preconditioned matrix

$$\tilde{Z}_m = P^{\frac{1}{2}} Z_m P^{\frac{1}{2}} \tag{5.5}$$

is of the form

$$\tilde{Z}_m = Ce^{j\theta}[T + j\sigma I] \tag{5.6}$$

where $T = T^t$ is real-symmetric and $C, \sigma, \theta \in \mathbf{R}$.

*Proof.* As $MM^t$ is symmetric positive definite, $P$ may be factored as $P = P^{\frac{1}{2}}P^{\frac{1}{2}}$, with

$$P^{\frac{1}{2}} = \frac{1}{\sqrt{r + j\omega l}}(MM^t)^{-\frac{1}{2}}. \tag{5.7}$$

Also since $MM^t$ is symmetric positive definite, the preconditioner can be applied so that the preconditioned system is given by Eq. 5.5 (See, for instance [5]). Combining Eq. 5.7 with Eq. 5.5,

$$P^{\frac{1}{2}} Z_m P^{\frac{1}{2}} = \left[ \frac{r}{j\omega} I + (MM^t)^{-\frac{1}{2}} MLM^t (MM^t)^{-\frac{1}{2}} \right] \frac{j\omega}{r + j\omega l} \tag{5.8}$$

or

$$P^{\frac{1}{2}} Z_m P^{\frac{1}{2}} = \frac{\omega}{\sqrt{r^2 + \omega^2 l^2}} e^{j\theta} \left[ j\sigma I + T \right] \tag{5.9}$$

with $\theta = \tan^{-1} \frac{r}{\omega l}$, $\sigma = -r/\omega$ and

$$T = (MM^t)^{-\frac{1}{2}} MLM^t (MM^t)^{-\frac{1}{2}}. \tag{5.10}$$

$MLM^t$ is positive definite since $L$ is, and therefore $T$ is symmetric positive definite. □

We are now ready to prove Theorem 5.3.5.

*Proof.* From Lemma 5.3.7, $\tilde{Z}_m$ is of the form $T + j\sigma I$, $T$ symmetric positive definite. Therefore, using Theorem 4 in [4], the computed iterates $x^k$ satisfy

$$\frac{||b - Ax^k||}{||b - Ax^0||} \leq \frac{2}{R^k + 1/R^k} \leq \frac{2}{R^k} \tag{5.11}$$

with $R = c(\omega) + \sqrt{c(\omega)^2 - 1}$, and where

$$c(\omega) = \frac{\sqrt{\lambda_{\max}(T)^2 + \sigma^2} + \sqrt{\lambda_{\min}(T)^2 + \sigma^2}}{\lambda_{\max}(T) - \lambda_{\min}(T)}. \tag{5.12}$$

Since

$$c(\omega) < c_\infty \equiv \frac{\lambda_{\max}(T) + \lambda_{\min}(T)}{\lambda_{\max}(T) - \lambda_{\min}(T)} = \frac{\kappa(T) + 1}{\kappa(T) - 1} \tag{5.13}$$

Then

$$R \leq c_\infty + \sqrt{c_\infty^2 - 1} = \frac{\sqrt{\kappa(T)} + 1}{\sqrt{\kappa(T)} - 1} \leq \frac{\sqrt{\kappa(L)} + 1}{\sqrt{\kappa(L)} - 1} \tag{5.14}$$

which combined with Eq. 5.11 proves the theorem. □

*Remark.* The preconditioned matrix $\tilde{Z}_m$ is normal and its eigenvalues lie on a line in the complex plane. To show this let $C = \frac{\omega}{\sqrt{r^2+\omega^2 l^2}}$. $\tilde{Z}_m$ is normal since $\tilde{Z}_m \tilde{Z}_m^H = C^2(T^2 + \sigma^2 I) = \tilde{Z_m}^H \tilde{Z}_m$. The eigenvalues $\lambda(\tilde{Z}_m) = Ce^{j\theta}(j\sigma + \lambda(T))$ clearly lie on a line, as the $\lambda(T)$ are real.

As a demonstration with non-uniform filaments, consider the printed circuit board example described below, between the low and high frequency limits. The required number of GMRES iterations monotonically and asymptotically increase toward the high frequency limit (See Fig. 5-6).

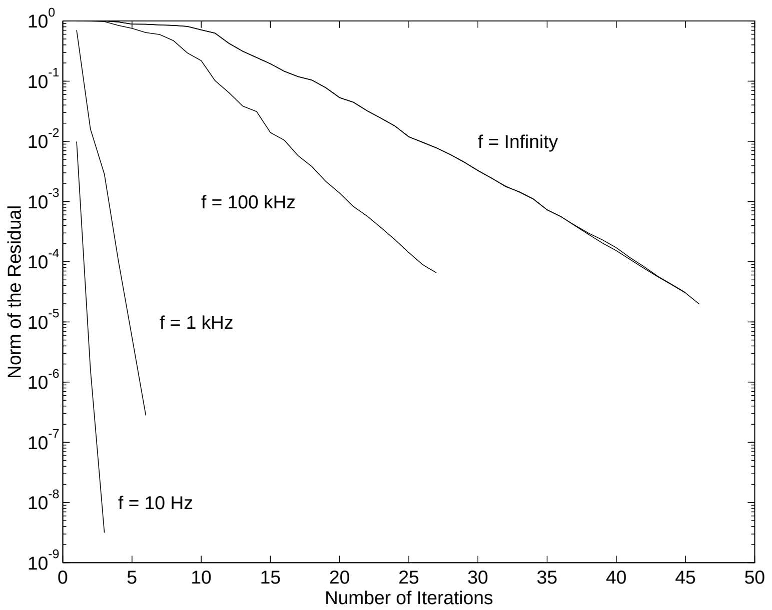


Figure 5-6: Convergence of GMRES using the sparsified-L preconditioner on printed circuit board package at various frequencies

### 5.3.3 Performance of Sparsified-L Preconditioners

To compare the relative merits of the diagonal-of-L, cube-block, and section-block preconditioners, consider the two industrial examples in Figs. 5-7 and 5-8. Fig. 5-7 is thirty-five pins of a 68-pin cerquad package and Fig. 5-8 is a portion of a printed circuit board (PCB) that would be placed underneath a Pin-Grid-Array package. The PCB example consists of two resistive reference planes sandwiching 255 copper lines. Each plane in the PCB has 53 external contacts not shown in the figure. For this experiment, the cerquad package was discretized into 3488 filaments which corresponds to 3305 meshes and each reference plane in the PCB was discretized into a $60 \times 60$ grid of meshes giving a total 7501 meshes including the copper lines. The GMRES error in the solution at high frequency as a function of iteration is plotted in Fig. 5-9 for the cerquad example, and in Fig. 5-10 for the PCB example. Note that the section-block and diagonal-of-L preconditioners are identical for the PCB example since there is only one filament per conductor section. As the figures clearly show, the block diagonal preconditioners are an improvement over the diagonal-of-L and local-inversion preconditioners. It is worth noting that for the cerquad package example, the number of non-zero elements in the factored cube-block preconditioner is 43 times that for the diagonal-of-L preconditioner, possibly prohibiting its use for larger problems. Also,

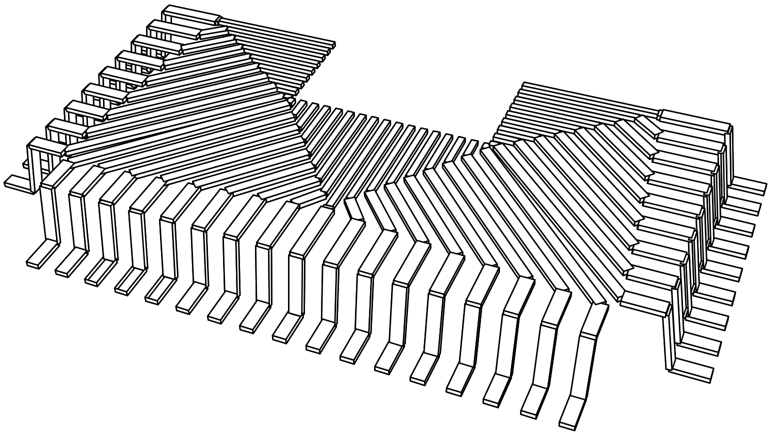

Figure 5-7: Half of a cerquad package. Thirty-five pins shown.

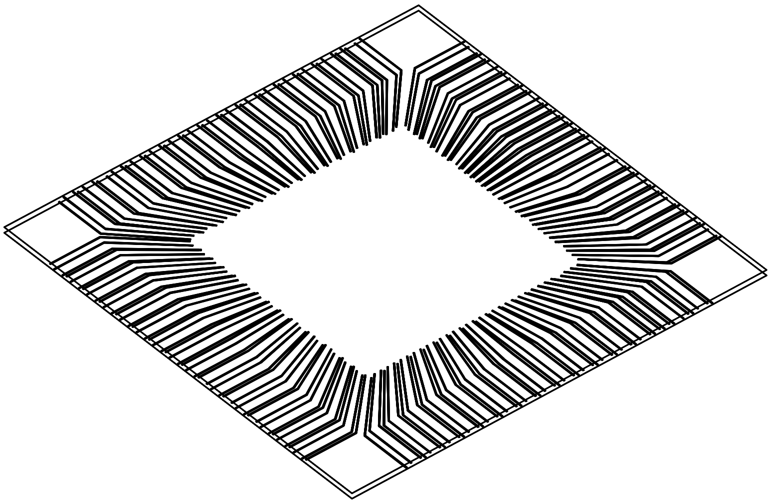

Figure 5-8: A portion of a printed circuit board directly underneath a PGA package. Two resistive reference planes sandwiching 255 copper lines. Only the outline of the planes is drawn.

for the pin-connect example, unlike the PCB example, local-inversion preconditioning did better than diagonal-of-$L$. This behavior can be expected since there are only 35 large meshes which must be excluded from the local-inversion preconditioner.

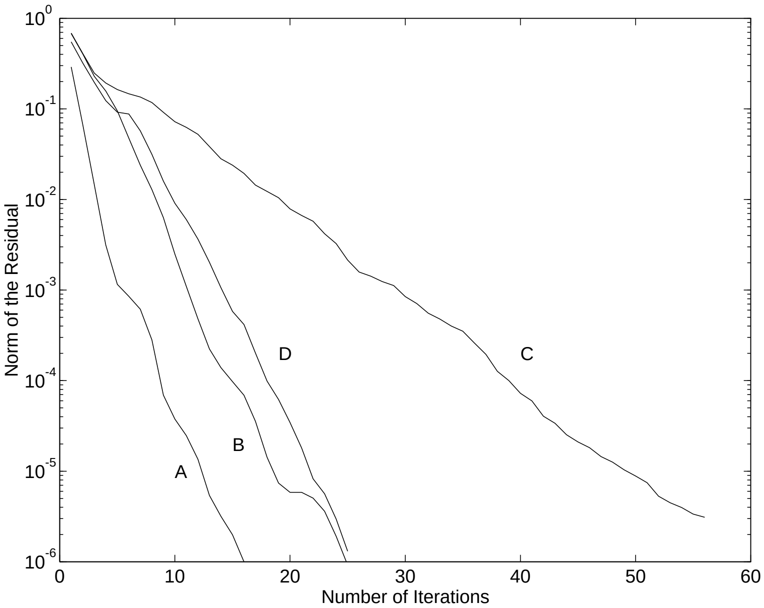


Figure 5-9: Convergence of GMRES applied to the cerquad example with cube-block preconditioning (A), section-block preconditioning (B), diagonal-of-L preconditioning (C), and local inversion precondtioning (D).

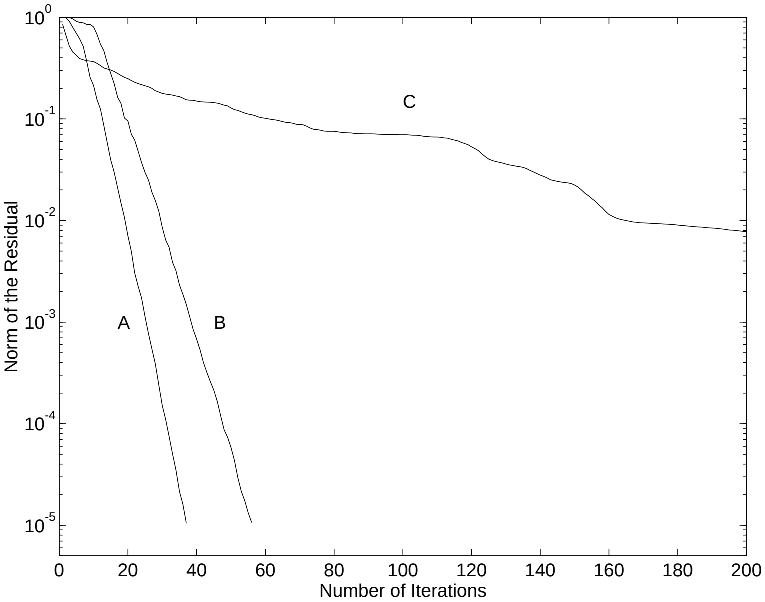


Figure 5-10: Convergence of GMRES applied to the PCB example with cube-block preconditioning (A), and diagonal-of-L preconditioning (B), and local-inversion preconditioning (C).

# Chapter 6

# Algorithm Results

In this section we demonstrate the accuracy, utility, and computational efficiency of the multipole-accelerated version of FastHenry. For an accuracy comparison, we consider the portion of a 68-pin package, shown previously in Fig. 5-7. Each pin consists of eight to ten conductor sections. We discretized each section into $2 \times 2$ filaments. This generated a problem with 1368 branches for which $MZM^t$ is a $1061 \times 1061$ dense matrix. Note, using only four filaments per section is hardly sufficient to model the skin effect, though with the coarse discretization, the problem is small enough to make possible an accuracy comparison between direct factorization, GMRES, and multipole-accelerated GMRES.

For the example package, the mutual inductance between pins 1 and 2 (labeled clockwise from the right) is much larger than the mutual inductance between pins 1 and 18 which are perpendicular to each other except for their vertical sections. To show that the approximations used by the hierarchical multipole algorithm are sufficiently well-controlled that small coupling inductances are computed accurately, consider the results in Table 6.1. The mutual inductance between pins 1 and 18 is more than two orders of magnitude smaller than the mutual inductance between pins 1 and 2, yet the solution computed using the multipole-accelerated algorithm is still within one percent of the solution computed using direct factorization.

As an example of the utility of frequency dependent inductance extraction possible with FastHenry, consider the two cases of computing the mutual inductance between a pair of PC board traces over a resistive ground plane, as shown in Fig. 6-1, and the same pair of traces over a divided ground plane, as in Fig. 6-2. The traces have their return paths through the

| pin pair | direct | gmres | multipole |
|---|---|---|---|
| 1 to 2 | 5.31870e+00 | 5.31867e+00 | 5.31403e+00 |
| 1 to 18 | 3.68292e-02 | 3.68223e-02 | 3.71027e-02 |

Table 6.1: Comparison of the accuracy of the computed inductance matrix entries between direct factorization, GMRES with explicit matrix-vector products, and the multipole-accelerated GMRES algorithm.

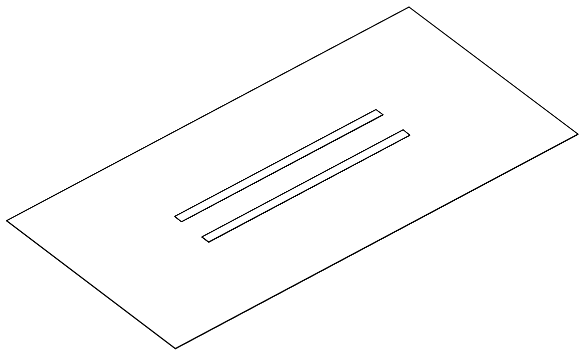

Figure 6-1: Two Traces over a Solid Ground Plane. The return path for the traces is through the plane. Traces are widened for illustration.

ground plane. For the divided plane case, the two portions are electrically connected with short resistive 'tethers' toward the outer edges as shown. The traces are 8 mils wide, 1 mil thick, 8 mils above the 1 mil thick ground plane, and their center to center distance is 32 mils.

If a current source is connected to one of the traces, current will flow down the trace and return through the plane. For the solid plane case, the current in the plane with a DC

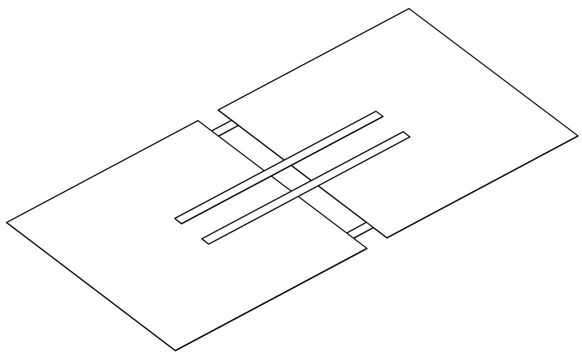

Figure 6-2: Two Traces over a Divided Ground Plane. The return path for the traces is through the plane. The divided portions are connected together toward the edges as shown. Traces are widened for illustration.

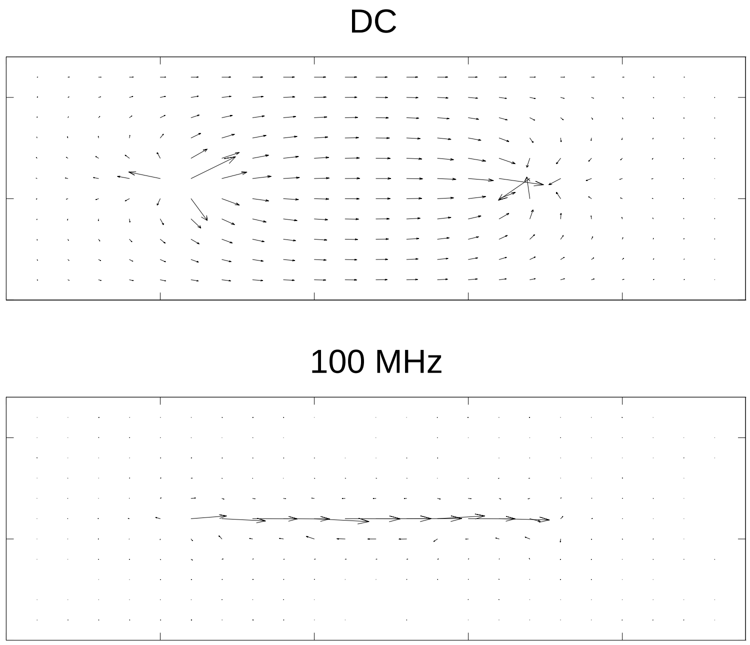


Figure 6-3: Current Distribution in Solid Ground Plane at DC and high frequency

source produces a current distribution pattern which spreads to fill the width of the plane. Similarly in the divided plane case, the current spreads throughout most of the plane, but narrows as it crosses the tethers. The situation at a high frequency is quite different. For the solid plane, the ground plane return current is concentrated directly underneath the trace, but for the divided plane the current leaves the path underneath the trace to cross the tethers (See Figs. 6-3 and 6-4).

This difference has a marked effect on the mutual inductance between the traces as the frequency rises. For the solid plane, as the frequency rises, the current gathers underneath the trace and the mutual inductance drops by two orders of magnitude, however for the divided plane, little decrease is observed with frequency (See Fig. 6-5).

To demonstrate the computational efficiency of FastHenry, we successively refined a coarse discretization of the ground plane of the example shown in Fig. 6-1. As the discretization of the plane is refined, the size of the problem will grow quickly, making the memory and CPU time advantage of the multipole-accelerated, preconditioned GMRES algorithm apparent (see Figs. 6-6 and 6-7). As the graphs clearly indicate, the cost of direct factorization grows like $m^3$, the cost of explicit GMRES grows as $m^2$, but the cost

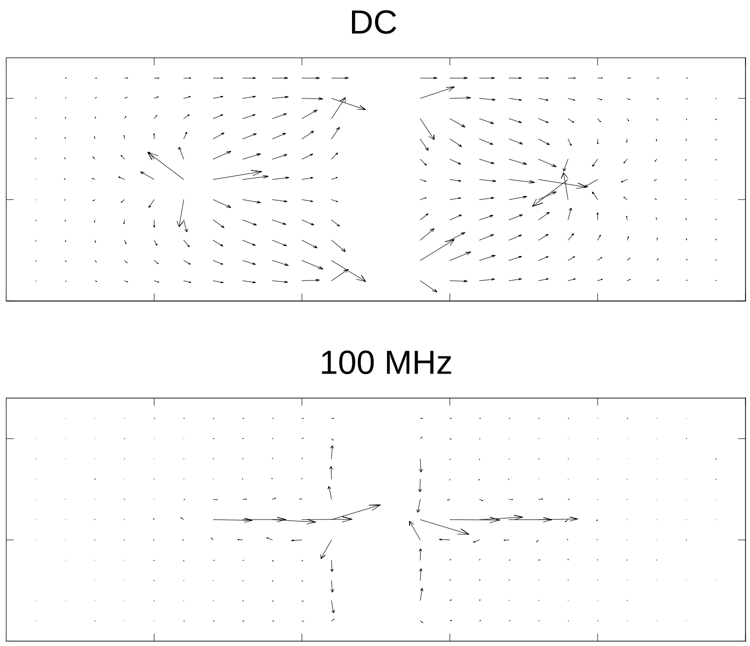


Figure 6-4: Current Distribution in Divided Ground Plane at DC and high frequency

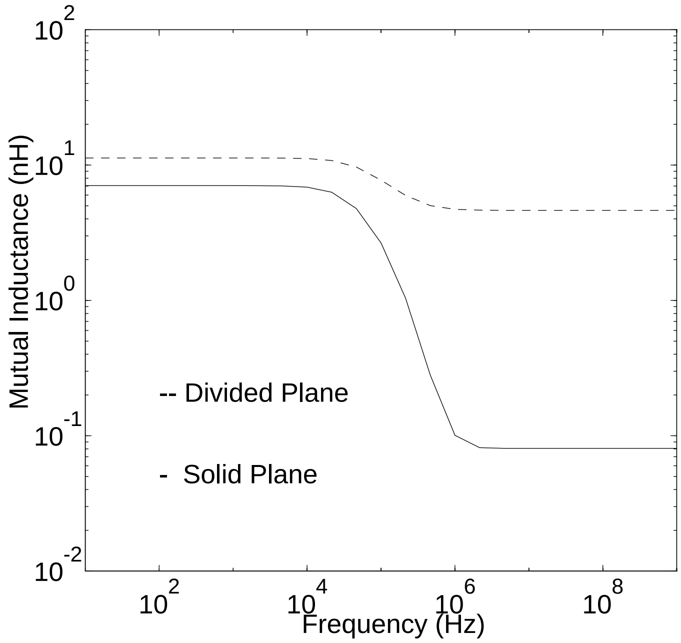


Figure 6-5: Mutual Inductance Between Traces

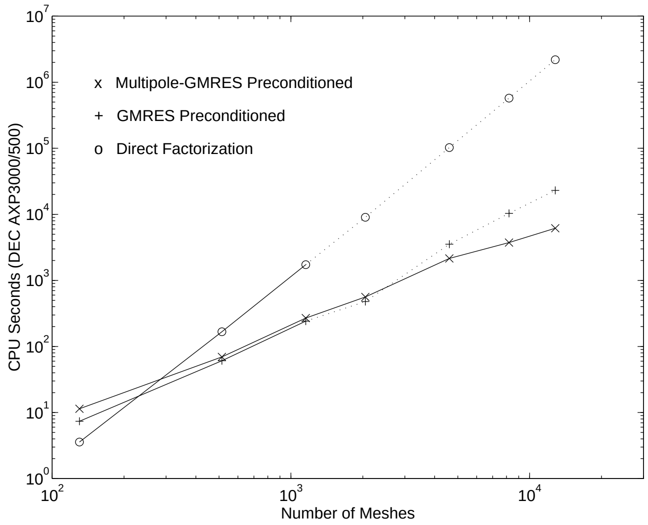


Figure 6-6: Comparison of the CPU time to compute the reduced inductance matrix for two traces over a solid plane using direct factorization, GMRES, and GMRES with with multipole acceleration.

of multipole-accelerated GMRES grows only linearly with $m$. In addition, the memory requirement for multipole-accelerated GMRES algorithm grows linearly with $m$, but grows like $m^2$ for either explicit GMRES or direct factorization. In particular, for a $12,802$ mesh problem, the multipole accelerated algorithm is more than two orders of magnitude faster than direct factorization, and uses an order of magnitude less time and memory than explicit GMRES. Note that the dotted lines in Figs. 6-6 and 6-7 indicate extrapolated values due to excessive memory requirements.

A significantly more complex problem and one that uses the sparsified-L preconditioner is the high frequency analysis of a portion of a PCB described previously and shown in Fig. 5-8. To properly model the current flow in the two reference planes surrounding the copper lines, the planes must be finely discretized. Here again, as the discretization is refined, the cost of direct factorization grows like $m^3$, the cost of explicit GMRES grows as $m^2$, but the cost of multipole-accelerated GMRES grows only linearly with $m$ as shown in Fig. 6-8. For this PCB example, the associated impedance matrix is 18x18, while the pair of traces over plane example has only a 2x2 impedance matrix. Thus, nine times as many GMRES solutions are required to compute the PCB example's impedance matrix. Even so, for a $10,000$ mesh problem, the multipole-accelerated GMRES algorithm is still over an order of magnitude faster in computation time.

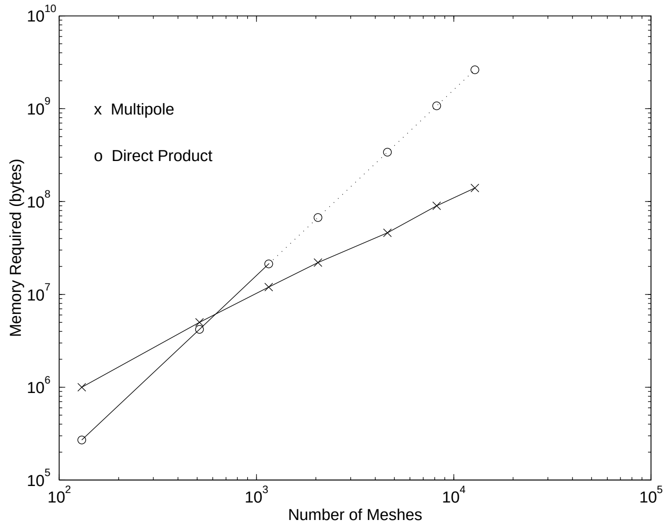


Figure 6-7: Comparison of the memory required using explicit matrix-vector products and using the multipole algorithm.

From Table 6.2 it can be observed that the time to compute the preconditioners is negligible compared to the total execution time, although for larger problems, the time required to compute the cube-block preconditioner may become significant. Also, the required number of iterations for either of the preconditioners does not grow rapidly with problem size. Finally, our analyses based on examining the spectra alone can be empirically justified by noting that for the discretizations shown in Table 6.2, $\kappa(V) < 20$, where $V$ is the matrix of eigenvectors of the preconditioned system.

| Precond. type | Size of $MZM^t$ | Precond. factor time | Total execution time | Total number of iterations | Average # of iters. per solve |
|---|---|---|---|---|---|
| diagonal-L | 751x751 | 0.26 | 450.46 | 729 | 41 |
| cube-block | 751x751 | 6.07 | 254.35 | 374 | 21 |
| diagonal-L | 1099x1099 | 0.81 | 1042.57 | 760 | 42 |
| cube-block | 1099x1099 | 11.86 | 755.12 | 518 | 29 |
| diagonal-L | 2101x2101 | 3.43 | 1901.58 | 760 | 42 |
| cube-block | 2101x2101 | 44.91 | 1381.15 | 502 | 28 |
| diagonal-L | 4351x4351 | 15.87 | 5522.79 | 842 | 47 |
| cube-block | 4351x4351 | 174.13 | 4609.96 | 641 | 36 |
| diagonal-L | 7501x7501 | 46.24 | 8894.92 | 883 | 49 |
| cube-block | 7501x7501 | 452.11 | 7309.18 | 635 | 35 |

Table 6.2: Execution times and iteration counts for diagonal-of-L and cube-block preconditioning of the printed circuit board example. Times are in CPU seconds for the DEC AXP3000/500.

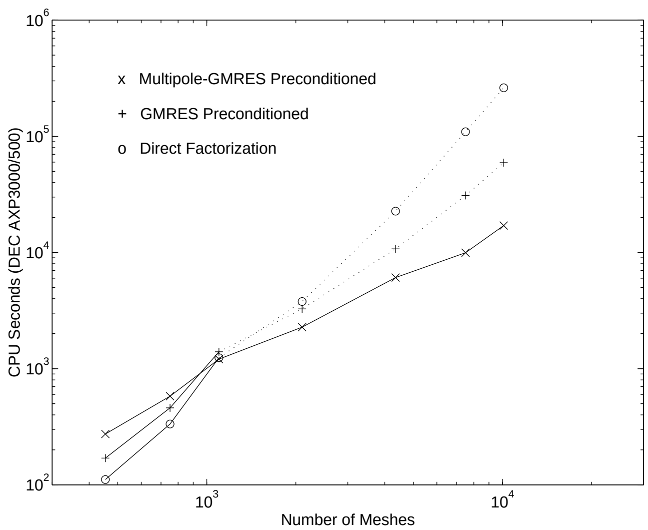


Figure 6-8: Comparison of the CPU time to compute the reduced inductance matrix for the PCB package using direct factorization, GMRES, and GMRES with with multipole acceleration.

# Chapter 7

# Conclusions

In this thesis, it is shown that 3-D inductance extraction can be substantially accelerated using the hierarchical multipole algorithm. The multipole-accelerated inductance extraction program, FastHenry, was shown to be more than two orders of magnitude faster than direct factorization when used to extract the inductance matrix for realistic packaging examples. In addition, the multipole-accelerated algorithm uses an order of magnitude less time and memory than the explicit GMRES algorithm. Finally, the sparsified preconditioner insures rapid convergence even for very irregular problems, making FastHenry a robust program.

# Appendix A

# FastHenry Implementation

The following is a pseudo-code description of the implementation of FastHenry as a computer program. The aim is to aid in putting together the ideas of this thesis in order to understand the partially commented FastHenry C code which is available via anonymous ftp from rle-vlsi.mit.edu.

The variable `sys` used below refers to a structure which contains pointers to many of the structures and matrices needed by most functions. `seg-list` the a list of all segments, `fil-list` is the list of all filaments.

```
main(argc, argv)
{
/* read the geometry specified in the input file. See Appendix B */
readGeom(sys);

/* divide each section, or segment, into filaments based on nhinc, nwinc. See Chapter 2 */
for each seg in seg-list {
   assignFil(seg);
   }

/* Call routines similar to FastCap to compute multipole expansions with each filament
   corresponding to a FastCap 'charge'. See Chapter 4 */
SetupMulti(sys);

/* Find user defined meshes, from algorithm in [2], pages 280-284. For the mesh formulation,
   the user may have created loops in the specified geometry. This finds those loops in this
   user created graph*/
make_trees(sys);
```

```
/* If user has put holes in any reference planes, must make one mesh for each of these holes
   */
find_hole_meshes(sys);

/* Create the Mesh Matrix M (see below and also Chapter 3) */
fillM(sys);

/* Fill the R and L matrices (see below and also Chapter 2) */
fillZ(sys);

for each frequency point {

   /* form preconditioner for this frequency (Chapter 5)*/
   indPrecond(sys,frequency);

   /* Do one solve with GMRES for each terminal pair specified in geometry */
   for each ext in terminal-pair-list {
      /* fill the right hand side corresponding to this ext turned on */
      fill_b(ext,b);

      /* call gmres to solve for solution, x. (see below and Chapter 3) */
      gmres(sys,b,x);

      /* multiply x by preconditioner one last time and put result in x */
      multPrecond(sys,x,y);
      x = y;

      /* extract the appropriate entries in x to go into one column of reduced admittance
        matrix, Y */
      extractYcol(Y, x, ext, terminal-pair-list);
      }

   /* save Y to disk */
   dump_to_Ycond(Y);

   /* invert Y to get Zc */
   invert(Y,Zc);

   /* save Zc to disk */
   cx_dumpMat_totextfile(Zc, "Zc.mat");
   }
}

/* this function creates the M matrix. Actually, it creates Mlist, a
```

```
list of meshes */
fillM(sys)
{

   /* make a mesh out of user created graph loops (found in make_trees() ) */
   for each loop {
      make_mesh_from_path(loop,Mlist);
      }

   /* Make meshes resulting from a segment being divided into many filaments (two fils
     per mesh) */
   for each seg in seg-list {
      make_fil_meshes(seg,Mlist);
      }

   /* make meshes resulting from discretizing ground plane. (four fils per mesh) */
   for each plane in plane-list {
      makeMlist(Mlist, plane);
      }
}

/* this computes the diagonal R matrix (resistance of each fil) and the dense L matrix (par-
  tial inductance matrix). This is only called if the matrix-vector products are done directly,
  however the mutual() function is called for multipole products when inside SetupMulti()
  for the interactions which are to be done directly */
fillZ(sys)
{
   for each filament, fili, in fil-list {
      R[fili] = resistance(fili);
      for each filament, filj, in fil-list {
         if (fili == filj)
            L[fili][fili] = selfterm(fili);
         else
            /* compute mutual inductance between two fils. See [7] and [10]. */
            L[fili][filj] = mutual(fili, filj);
         }
      }
}

/* This computes the solution x with right hand side b and implicit matrix A. This pseudo-
  code version is similar to Algorithm 3.2.1.*/
gmres(sys,b,x)
{
   /* assume initial guess, x, is zero. norm() is the Euclidean norm of a vector. */
   r0 = r = b;
```

```
   k = 0;
   while (norm(r)/norm(r0) > tolerance) and (k < max_iterations) {
      k = k + 1;

      /* compute matrix-vector product with mulitpole algorithm (see below)*/
      SetupComputePsi(sys,x,p);
      r = b - p;

      /* update x as described in [18]. */
      x = update_x();
      }
}

/* use multipole algorithm to compute Ax. Return value in p */
SetupComputePsi(sys,x,p)
{

   /* apply preconditioner, result in y */
   multPrecond(sys,x,y);

   /* Call multipole algorithm for each of the three components of the vector potential.
     (MLMt)y */
   p = 0;
   for i = 1 to 3 {
      p = p + ComputePsi(y,i,sys);
      }

   /* add in the contribution due to (MRMt)y */
   p = p + do_real_part(sys,y);
}
```

# Appendix B

# FastHenry User's Manual

**FastHenry USER'S GUIDE**

M. Kamon C. Smithhisler J. White

Research Laboratory of Electronics
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 U.S.A.

28 June 1993

This work was supported by Defense Advanced Research Projects Agency contract N00014-91-J-1698, a National Science Foundation Graduate Fellowship, and grants from IBM and Digital Equipment Corporation.

This manual describes FastHenry, a three-dimensional inductance extraction program. FastHenry computes the frequency dependent self and mutual inductances between conductors of complex shape. The algorithm used in FastHenry is an acceleration of the mesh formulation approach. The linear system resulting from the mesh formulation is solved using a generalized minimal residual algorithm with a fast multipole algorithm to efficiently compute the iterates.

This manual is divided into two sections. The first section explains how to prepare input files for FastHenry. The input files contain the description of the conductor geometries. The second section shows how to run the program. It mainly explains how to modify the default settings assumed by FastHenry. Also included in this section is output from an example run.

Information on compiling FastHenry, obtaining the FastHenry source code and corresponding about FastHenry is given in Appendix B.3.

## B.1 How to Prepare Input Files

This section of the manual describes how to prepare input files for FastHenry. The input files specify the <u>discretization</u> of <u>conductor volumes</u> into <u>filaments</u>. The input file specifies each conductor as a sequence of straight <u>segments</u>, or elements, connected together at <u>nodes</u>. Each segment has a finite conductivity and its shape is a cylinder of rectangular cross section of some width and height. A node is simply a point in 3-space. The cross section of each segment can then be broken into a number of parallel, thin filaments, each of which will be assumed to carry a uniform cross section of current along its length. The first part of this section describes the file format through a simple example. A detailed description is in the second part, and more complex examples can be found later in this manual.

### B.1.1 A Simple Example

The following is an input file which calculates the loop inductance of four segments nearly tracing the perimeter of a square:

```
**This is the title line. It will always be ignored**.
* Everything is case INsensitive
* An asterisk starts a comment line.

* The following line names the length units for the rest of the file
.Units MM

* Make z=0 the default z coordinate and copper the default conductivity.
* Note that the conductivity is in units 1/(mm*Ohms), not 1/(m*Ohms).
.Default z=0 sigma=5.8e4

* The nodes of a square (z=0 is the default)
N1 x=0 y=0
N2 x=1 y=0
N3 x=1 y=1
N4 x=0 y=1
N5 x=0 y=0.01
```

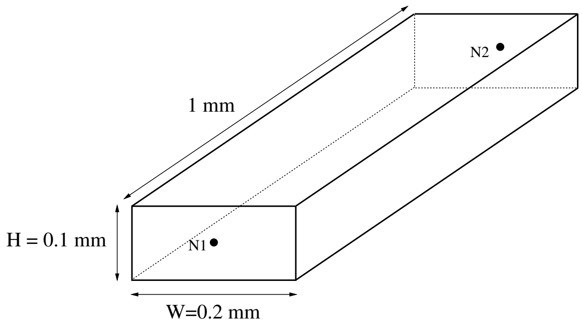


Figure B-1: Example Segment for Sample Input File

```
* The segments connecting the nodes
E1 N1 N2 w=0.2 h=0.1
E2 N2 N3 w=0.2 h=0.1
E3 N3 N4 w=0.2 h=0.1
E4 N4 N5 w=0.2 h=0.1

* define one 'port' of the network
.external N1 N5

* Frequency range of interest.
.freq fmin=1e4 fmax=1e8 ndec=1

* All input files must end with:
.end
```

As described in the comments, `.Units MM` defines all coordinates and lengths to be in millimeters. All lines with an `N` in the first column define nodes, and all lines starting with `E` define segments. In particular, the line

```
E1 N1 N2 w=0.2 h=0.1
```

defines segment E1 to extend from node `N1` to `N2` and have a width of 0.2mm and height of 0.1 mm as drawn in Figure B-1. If the $n \times n$ impedance matrix, $Z(\omega)$, for an $n$-conductor problem is thought of as the parameters describing an $n$-port network, then the line

```
.external N1 N5
```

defines `N1` and `N5` as one port of the network. In this example, only one port is specified, so the output will be a $1 \times 1$ matrix containing the value of the impedance looking into this one port.

FastHenry calculates $Z(\omega)$ at the discrete frequencies described by the line

```
.freq fmin=1e4 fmax=1e8 ndec=1
```

where `fmin` and `fmax` are the minimum and maximum frequencies of interest, and `ndec` is the number of desired frequency points per decade. In this case, $Z(\omega)$ will be calculated at $10^4, 10^5, 10^6, 10^7$, and $10^8$ Hz. All input files must end with `.end`.

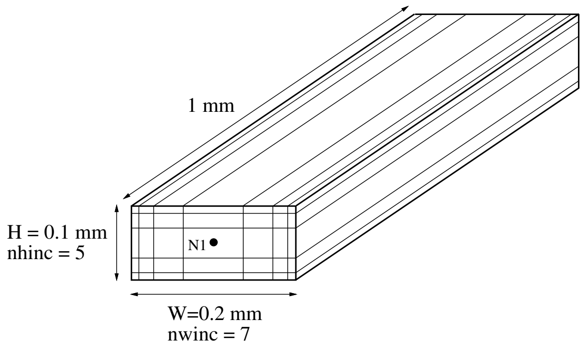


Figure B-2: Segment discretized into 35 filaments

In the above example, FastHenry created one filament per segment since no discretization of the segments into filaments was specified. In order to properly model non-uniform cross sectional current due to skin and proximity effects, a finer discretization must be used. Finer filaments are easily specified in the segment definition. For example, replacing the definition for `E1` with

```
E1 N1 N2 w=0.2 h=0.1 nhinc=5 nwinc=7
```

specifies that `E1` is to be broken up into thirty-five filaments: five along its height (`nhinc=5`) and seven along its width (`nwinc=7`). See Figure B-2.

### B.1.2 Input File Syntax

The previous section described many of the basics required for an input file. This section gives a more complete and detailed description of the input file format and should serve as a reference.

Some general facts about file syntax:

- Lines are processed sequentially
- Upper and lower case are not distinguished.
- Lines are restricted to 1000 characters but can be continued with a "+" as the first character in subsequent lines. Intervening "*" lines are not allowed.
- "*" marks a comment line.
- The first line in the file is considered the title line and is ignored. It is recommended that this line start with an "*" for future compatibility.
- The file must end with the `.End` keyword.

In general, each line in the input file will either define some geometrical object, such as a node or segment, or it will specify some program parameter. All input lines that define geometrical objects begin with a letter defining their type, and then some unique alphanumeric string. For instance, all node definitions begin with the letter `N`. This sets object lines apart from parameter specification lines which begin with a period, ".".

The remainder of this section will describe all possible input lines. In the following description, all arguments enclosed in '[ ]' indicate an optional argument. If not included on the input line, the actual value used for this argument will be either the program default, or the user default defined by the `.Default` keyword (described below).

### Node Definitions

```
Syntax:  Nstr [x = x_val] [y = y_val] [z = z_val]
```

This defines a node called `Nstr` where `str` is any alphanumeric string. The first character on the line must begin with an `N` for this to be interpreted as a node definition. The node will have location $(x\_val, y\_val, z\_val)$ where each coordinate has units defined by the `.Units` keyword.

Any of the coordinates can be omitted assuming that a default value has been previously specified with the `.Default` keyword. Otherwise, an error will occur and the program will exit.

### Segment Definitions

```
Syntax:  Estr node1 node2 [w = value] [h = value] [sigma, rho = value]
                [wx = value wy = value wz = value]
                [nhinc = value] [nwinc = value]
```

This defines a segment called `Estr` where `str` is any alphanumeric string. The first character on the line must begin with the letter `E` for this to be interpreted as a segment definition. The segment will extend from node `node1` to node `node2` where these are previously defined node names. `h` and `w` are the segment height and width. Either `sigma`, the conductivity, or `rho`, the resistivity, can be specified for the segment.

Discretization of the segment into multiple, parallel thin filaments is specified with the `nhinc` and `nwinc` arguments. `nhinc` specifies the number of filaments in the height direction, and `nwinc`, the number in the width direction. Both must be integers. See Figure B-2.

To specify the orientation of the cross section, `wx`, `wy`, and `wz` represent any vector pointing along the width of the cross section. If these are omitted, the width vector is assumed to lie in x-y plane perpendicular to the length. If the length lies along the z-axis, then the width is assumed along the x-axis.

`h` and `w` can be omitted provided they are assigned a default value in a previous `.Default` line.

`nhinc`, `nwinc`, and `sigma` or `rho` can be omitted, and if not previously given a default value, then 1, 1, and the conductivity of copper, respectively, are used as default values.

Note that the nodes used to define the nodes must be nodes defined under **Node Definitions** described above and cannot be ground plane nodes. To connect to the ground plane, the user must instead create a new node at the desired location as described in **Node Definitions** and then use the `.Equiv` keyword to equivalence the ground plane node and the new node.

### `.Units` keyword

```
Syntax:  .Units unit-name
```

This specifies the units to be used for all subsequent coordinates and lengths until the end of file or another `.Units` specification is encountered. Allowed units are meters, centimeters, millimeters, micrometers, inches, and mils with `unit-name` specified as `m, cm, mm, um, in, mils`, respectively.

Note that this keyword affects the expected units for the conductivity and resistivity.

### `.Default` keyword

```
Syntax:  .Default [x = value] [y = value] [z = value] [w = value]
                  [h = value] [sigma, rho = value]
                  [nhinc = value] [nwinc = value]
```

This keyword specifies default values to be used for subsequent object definitions. A certain default value is used until the end of the file, or until it is superseded by another `.Default` line changing that value.

### `.External` keyword

```
Syntax:  .External node1 node2
```

This keyword specifies node `node1` and node `node2` as a terminal pair or port whose impedance parameters should be calculated for the output impedance matrix. If an input file includes $n$ `.External` lines, then the impedance matrix will be an $n \times n$ complex matrix.

This keyword effectively places a voltage source between these nodes and will later use the current through that source to determine an entry in the admittance matrix. Note that it is up to the user to insure that there are NO loops of only voltage sources. Also, a voltage source with no possible return path will always have zero current through it producing a row of zeros in the admittance matrix. The output impedance matrix will thus be nonsense.

### `.Freq` keyword

```
Syntax:  .Freq fmin=value fmax=value [ndec = value]
```

This keyword specifies the frequency range of interest. `fmin` and `fmax` are the minimum and maximum frequencies of interest, and `ndec` is the number of desired frequency points per decade. FastHenry must perform the entire solution process for each frequency.

Note that `ndec` need not be an integer. For instance,

```
.freq fmin=1e3 fmax=1e7 ndec=0.5
```

will have FastHenry calculate impedance matrices for $f = 10^3, 10^5$, and $10^7$ Hz.

If `fmin` is zero, FastHenry will run only the DC case regardless of the value of `fmax`.

### `.Equiv` keyword

```
Syntax:  .Equiv node1 node2 node3 node4 ...
```

This keyword specifies that nodes `node1, node2, node3, node4,...` are to be considered electrically equivalent yet maintain their separate spatial coordinates. It basically 'shorts' all these nodes together. If any of the node names are not previously defined, then they become pseudonyms for those nodes in the list which are defined.

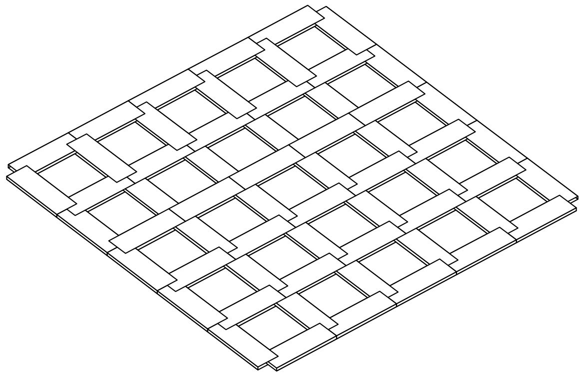

Figure B-3: Discretization of a Ground Plane. Segments are one-third actual width.

### `.End` keyword

```
Syntax:  .End
```

This keyword specifies the end of the file. All subsequent lines are ignored. This line must end the file.

### Ground Plane definitions

```
Syntax:  Gstr x1=value y1=value z1=value x2=value y2=value z2=value
              x3=value y3=value z3=value
              thick=value seg1=value seg2=value [sigma, rho = value]
              [nhinc=value] [relx=value] [rely=value] [relz=value]
              [Nstr1     (x_val,y_val,z_val) ]
              [Nstr2     (x_val,y_val,z_val) ]
              [Nstr3     (x_val,y_val,z_val) ].....
              [hole <hole-type> (val1,val2,....)]
              [hole <hole-type> (val1,val2,....)].....
```

This defines a ground plane of finite extent and conductivity called `Gstr` where `str` is any alphanumeric string. The first character on the line must begin with the letter `G` for this to be interpreted as a ground plane definition. The three locations $(x1, y1, z1), (x2, y2, z2), (x3, y3, z3)$ mark three of the four corners of the plane in either clockwise or counterclockwise order. The code will determine the fourth corner assuming the first three are corners of a parallelogram. Actually, the plane must be a rectangle, but this condition may not be flagged until later in the code. The thickness of the plane is specified with the `thick` argument and the conductivity with either the `sigma` or `rho` argument.

The ground plane is approximated by first laying down a grid of nodes on the plane, and then connecting, with a segment, every node to its adjacent nodes excluding diagonally adjacent nodes. Each segment is given a height equal to the specified thickness of the plane and width equal to the node spacing in order to completely fill the space between segments. Figure B-3 shows a sample ground plane with segments that are one-third normal width for illustration.

`seg1` and `seg2` specify the number of segments along each edge of the plane. `seg1` will be the number of segments along the edge from $(x1, y1, z1)$ to $(x2, y2, z2)$ and `seg2`, the number along the edge from $(x2, y2, z2)$ to $(x3, y3, z3)$. Thus the total number of nodes created will be (seg1+1)*(seg2+1), and the total number of segments created will be $(\texttt{seg1}+1)*\texttt{seg2}+\texttt{seg1}*(\texttt{seg2}+1)$ .

`nhinc` can be used to specify a number of filaments for discretization of each segment along the thickness. This could be used for modelling nonuniform current along the thickness. If omitted, the value of 1 is assumed, regardless of the `.Default` setting.

In general, since the ground plane nodes are generated internally, there is no way to refer to them later in the input file. The exceptions to this are the nodes explicitly referenced in the ground plane definition. The argument

```
Nstr1     (x_val,y_val,z_val),
```

where `str1` is an alphanumeric string, will cause all subsequent references to `Nstr1` to refer to the node in the plane closest to the point $(x\_val, y\_val, z\_val)$. Note that no spaces are allowed between the '()'. This referencing is accomplished, in effect, by doing the equivalent of

```
.Equiv Nstr1 <internal-node-name>
```

where `<internal-node-name>` is the internal node name of the nearest ground plane node.

If one or more of `relx`, `rely`, and `relz` are specified, then the above node referencing instead chooses the node closest to $(x\_val+relx, y\_val+rely, z\_val+relz)$. In other words, `relx`, `rely`, and `relz` default to 0 if not specified.

A coarse ground plane may cause two different references to refer to the same ground plane node. FastHenry will warn of such an event, but it is not an error condition.

Holes can be specified in the plane with

```
hole <hole-type> (val1,val2,val3,...)
```

where `<hole-type>` is the hole type and the `valn`'s make the list of arguments to be sent to the hole generating function. Holes are generated by first removing ground plane nodes, and then removing all segments connected to those nodes. The following describes the available hole generating functions:

```
hole point (x,y,z)
```

removes the node nearest to the point (x,y,z).

```
hole rect (x1,y1,z1,x2,y2,z2)
```

removes a rectangular region whose opposite corners are the nodes in the plane nearest (x1,y1,z1) and (x2,y2,z2).

```
hole circle (x,y,z,r)
```

removes the nodes contained within the circle of radius `r` centered at (x,y,z).

```
hole user1 (val1,val2,...)
```

calls the user defined function hole_user1() to remove nodes. user1 - user7 are available. The user can add the functions to the source file hole.c contained in the release. See the functions hole_rect(), hole_point(), and hole_circle() to see examples of the format for writing user hole functions.

Any shaped hole can be formed with a combination of hole directives. A few exceptions exist, however. Forming a hole that isolates or nearly isolates a section of the plane is not allowed. FastHenry warns of this with:

```
Warning: Multiple boundaries found around one hole region
  possibly due to an isolated or nearly isolated region of conductor.
  This may lead to no unique solution.
```

Holes may not be produced as expected if the discretization of the plane is coarse. It is recommended that the plane be viewed by using the options '-f simple -g on' options to generate a fastcap file which can be used to generate a postscript image of the plane.

See example `hole.inp`.

*Warning*: Fast convergence for ground planes has not been fully accomplished. At the time of release, some ground planes with many closely coupled conductors converged slowly using the iterative algorithm. This may also happen when nhinc $> 1$. If you have practical examples for which this happens, please send them to fasthenry@rle-vlsi.mit.edu.

## B.2 Running FastHenry

The basic form of the FastHenry program command line is

```
fasthenry [<input file>] [<Options>]
```

Usually only the `input file`, as described in the previous section, is specified. For example, the command

```
fasthenry pin-connect.inp
```

runs `fasthenry` on the example pin connect structure.

Information about the input file and other FastHenry information are sent to the standard output. The impedance matrices for the frequencies specified in the input file will be placed in the file `Zc.mat`. The source file `ReadOutput.c` is a sample program for reading the output file for postprocessing.

### B.2.1 Command Line Options

This section describes using the command line options for changing the defaults settings. All arguments are case insensitive.

`-s` {`ludecomp | iterative`} - Specifies the matrix solution method used to solve the linear system arising from the discretization. `iterative` uses the GMRES iterative algorithm and `ludecomp` uses LU decomposition with back substitution. In general, GMRES is faster, however some speed up may be obtained using LU decomposition for problems with fewer than 1000 filaments. `iterative` is the default.

`-m` {`direct | multi`} - Specifies the method to use to perform the matrix-vector product for the iterative algorithm. `direct` forms the full matrix and performs the product directly. `multi` uses the multipole algorithm to approximate the matrix-vector product. For larger problems, the multipole algorithm can save both computation time and memory. `multi` is the default.

`-p` {`on | off`} - Determines whether or not to precondition the matrix to accelerate iteration convergence. `on` is the default.

`-o n` - Specifies $n$ as the order of multipole expansions. Default is 2.

`-l` {`n | auto`} - Specifies $n$ as the number of partitioning levels for the multipole algorithm. `auto` chooses the level automatically and is the default.

`-f {off | simple | refined | both}` - Specifies the type of FastCap generic file to make for visualization ONLY. `off` will produce no file and is the default. `simple` will produce a file named `fastcapfile` from the segments defined in the input file. `refined` produces a similar file, named `fastcapfile2`, but using the segments produced by either user refinement with `-i` or required refinement necessary for accuracy of the multipole algorithm. `both` produces both files. One FastCap 'panel' is created for each of the four sides along the length of the each segment. Ground planes are handled differently. See the `-g` option below. `fastcap -mfastcapfile` should produce the appropriate postscript file. See the FastCap manual for more details.

`-g {on | off}` - controls appearance of the ground plane when using the `-f` option. `on` will draw all the overlapping segments of the ground plane. Note that this may take a long time to run through fastcap to generate a postscript image. `off` is the default and only four panels are produced for each plane, one for each of the edges. Thus only the outline of each plane is drawn. This makes generation of the postscript image faster and also the planes are transparent. No holes are visible however.

`-a {on | off}` - `on` allows the multipole algorithm to automatically refine the structure as is necessary to maintain accuracy in the approximation. The structure will be refined whether or not the multipole algorithm is used. `off` prevents refinement and will produce a warning if the multipole algorithm is used and prevented from necessary refinement. `on` is the default. Note that this is NOT refinement to reduce discretization error. This must be done by the user. See the `-i` option.

`-i n` - Specifies $n$ as the level for initial refinement. This option allows the user to refine the structure if the input file is too coarse. It will divide each segment of the geometry into multiple segments so that no segment has a length greater than $\frac{1}{2^n}$ times the length of the smallest cube which contains the whole structure. The default is 0 (no refinement).

`-d {on | off | mrl | mzmt | grids | meshes}` - dump certain internal matrices to files. The format of the file is specified with the `-k` option. `on` dumps all files, `off` dumps none and is the default. `mrl` dumps the M, R, and L matrices. `mzmt` dumps the $MZM^t$ matrix. `grids` dumps matrices for viewing the current distribution inside each ground plane. It only dumps in matlab format. The same is true for `meshes` which is a matlab file for viewing the meshes chosen by FastHenry. For information on using `grids` send mail to fasthenry@rle-vlsi.mit.edu.

`-k {matlab | text | both}` - Specifies type of file to dump with the `-d` option. `matlab` dumps the files as MRL.mat and MZMt.mat and are in a format readable by matlab. `text` saves the files M.dat, L.dat, R.dat, and MZMt.dat as text. `both` saves files in both formats.

`-t rtol, -b atol` - Specifies the tolerance for iteration error. FastHenry calculates each column of the impedance matrix separately. The iterative algorithm will stop iterating when both the real and imaginary part of each element, $x_k$, of the current column being calculated satisfies

$$|x_k^{i-1} - x_k^i| < rtol * (|x_k^i| + atol * (max_j |x_j^i|)) \tag{B.1}$$

where $i$ is the iteration number. The defaults are $rtol = 10^{-3}$ and $atol = 10^{-2}$.

`-c n` - n = maximum number of iterations to perform. Overrides the default of 200.

`-D {on | off}` - Controls the printing of debugging information. `off` is the default. `on` will cause FastHenry to print more detailed information about the automatic partitioning level selection, memory consumption, preconditioner calculation, and convergence of the iterates.

### B.2.2 Example Run

With the release are example structures `30pin.inp`, `pin-connect.inp`, `onebargp.inp`, `hole.inp`, `broken.inp`, and `together.inp`. Here is a sample run of FastHenry for the 30 pin connector example, `30pin`. Comments describing the output appear along the right margin in addition to the `FastHenry` command line option which would change the described setting.

```
prompt % fasthenry 30pin
Solution technique: ITERATIVE                  <-- use GMRES (-s)
Matrix vector product method: MULTIPOLE        <-- and multipole algorithm (-m)
  Order of expansion: 2                        <-- order of multipole (-o)
Preconditioner: ON                             <-- GMRES preconditioner (-p)
Error tolerance: 0.001                         <-- relative tol (-t, -b)
Reading from file: 30pin
Title:
**30 pin, right angle connector**              <-- first line from input file

all lengths multiplied by 0.001 to convert to meters
Total number of filaments before multipole refine: 290
Total number of filaments after multipole refine: 455      <-- multipole needed
                                                               to refine (-a)
Multipole Summary
  Expansion order: 2                                  <--  -o option
  Number of partitioning levels: 3                    <--  -l option
  Total number of filaments: 455
Percentage of multiplies done by multipole: 100%
Scanning graph to find fundamental circuits...        <-- find meshes
Number of Groundplanes : 0                            <-- some data about
Number of filaments: 455                                  input geometry
Number of segments:  455
Number of nodes:     605
Number of conductors:30
Number of meshes:    60
          ----from tree:   60              <-- fundamental circuits from graph
          ----from planes: 0
Number of real nodes:455
filling M...
filling R and L...
Total Memory allocated: 1985 kilobytes          <-- Total memory consumed
Frequency = 10000
Forming overlapped preconditioner       <-- form preconditioner for this freq.
conductor 0 from node npin4_5_1         <-- do 1st column of admittance matrix
Calling gmres...
1 2 3 4 5 6 7 8 9 10 11                 <-- Iterations until convergence
conductor 1 from node npin4_4_1         <-- Begin next column
Calling gmres...
1 2 3 4 5 6 7 8 9 10 11
conductor 2 from node npin4_3_1
Calling gmres...
1 2 3 4 5 6 7 8 9 10 11
conductor 3 from node npin4_2_1
 .
 .
```

```
 .
conductor 28 from node npin0_1_1
Calling gmres...
1 2 3 4 5 6 7 8 9 10
conductor 29 from node npin0_0_1
Calling gmres...
1 2 3 4 5 6 7 8 9 10

All impedance matrices dumped to file Zc.mat    <-- Where the results are

Times:  Read in stuff   1.51                    <-- some execution time info
        Multipole setup 11.94
        Scanning graph  0.01
        Form A M and Z  0.06
        form M'ZM       0
        Form precond    1.36
        GMRES time      75.92
   Total:               90.8
90.980u 0.640s 1:36.62 94.8% 321+2448k 0+0io 4pf+0w
prompt %
```

### B.2.3 Processing the Output

The file Zc.mat is a text file containing the impedance matrices for the frequencies requested in the input file. The file ReadOutput.c is an example program for reading the text file for whatever processing is necessary. It contains the function ReadZc() which reads from a file and returns a linked list in which each element of the list contains an impedance matrix and its corresponding frequency. See the source file for more details. This function can be extracted and included in whatever program the user desires.

The function main() is provided as an example use of ReadZc(). For each of the matrices, it divides the imaginary part by the frequency to give the matrix $R + jL$ and then dumps the result to the standard output.

ReadOutput.c can be compiled by typing

```
cc -o ReadOutput ReadOutput.c
```

Here is a sample of its output after processing Zc.mat produced by running fasthenry on example file onebargp.inp:

```
prompt % ReadOutput Zc.mat
Not part of any matrix: Row 2 :  nodein  to  nodeout
Not part of any matrix: Row 1 :  nb1  to  nbout
Reading Frequency 10000
Reading Frequency 100000
Reading Frequency 1e+06
Reading Frequency 1e+07
Reading Frequency 1e+08
freq = 1e+08
Row 0: 0.00112838+3.50478e-08j -2.21062e-05-7.0003e-09j
Row 1: -2.23118e-05-6.99564e-09j 4.83217e-05+2.02958e-08j
freq = 1e+07
Row 0: 0.00112837+3.50478e-08j -2.21055e-05-7.0003e-09j
Row 1: -2.2311e-05-6.99564e-09j 4.83217e-05+2.02958e-08j
```

```
freq = 1e+06
Row 0: 0.0011272+3.50501e-08j -2.20335e-05-7.00045e-09j
Row 1: -2.22379e-05-6.99578e-09j 4.83168e-05+2.02958e-08j
freq = 100000
Row 0: 0.00106093+3.51782e-08j -1.7938e-05-7.00794e-09j
Row 1: -1.80765e-05-7.00341e-09j 4.80425e-05+2.02964e-08j
freq = 10000
Row 0: 0.000955377+3.58555e-08j -1.25489e-05-7.01913e-09j
Row 1: -1.25828e-05-7.01518e-09j 4.75474e-05+2.03045e-08j
prompt %
```

### B.2.4 Other Examples

This example shows how to run `fasthenry` on example `30pin.inp` with user refinement at level 4 so that no segment will be larger than 1/16 of the largest dimension of the structure. It will also have FastHenry dump the unrefined structure to a FastCap readable file for visualization:

```
fasthenry 30pin.inp -i 4 -f simple
```

Run `fasthenry` on example `onebargp.inp` still using the iterative algorithm, but without the multipole algorithm for the matrix vector product:

```
fasthenry onebargp.inp -m direct
```

## B.3 Compiling FastHenry

A tar file containing the source files for `fasthenry` and this guide may be obtained on tape by sending a written request to

Prof. Jacob White
Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Room 36-880
Cambridge, MA 02139 U.S.A.

This address may also be used for general correspondence regarding `fasthenry`, although electronic mail may be sent to `fasthenry-bug@rle-vlsi.mit.edu`, for bug reports, and to `fasthenry@rle-vlsi.mit.edu`, for questions or comments, if it is more convenient.

The tar file has the form

```
fasthenry-1.0-13Jan93.tar.Z
```

and yields a one level directory when untarred with the commands

```
uncompress fasthenry-1.0-13Jan93.tar.Z
tar xvf fasthenry-1.0-13Jan93.tar
```

It will create a directory called `fasthenry` and which contains all the C source files, the LaTeX files for this manual, and the example files.

### B.3.1 Compilation Procedure

FastHenry is compiled by changing to the `fasthenry` directory, and typing

```
make
```

to create the executable `fasthenry`. This will use the file `Makefile` to make `fasthenry`. Certain DEC compilers run out of space during the compile. If either this is the case or Matlab files will be dumped, replace the file `Makefile` with a copy of `Makefile.dec`. To compile on a Silicon Graphics Workstation, change the first line of `Makefile` to

```
CFLAGS = -O -DFOUR -DOLDPRE -Olimit 2000 -cckr
```

### B.3.2 Producing this Guide

In the `fasthenry` directory, type

```
latex manual.tex
```

and then again,

```
latex manual.tex
```

to get the references correct. The manual will be the file `manual.dvi`.

# Bibliography


[1] A. C. Cangellaris, J. L. Prince, and L. P. Vakanas. Frequency-dependent inductance and resistance calculation for three-dimensional structures in high-speed interconnect systems. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, 13(1):154–159, March 1990.

[2] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, N.J., 1974.

[3] C. A. Desoer and E. S. Kuh. *Basic Circuit Theory*. McGraw-Hill, New York, 1969.

[4] R. Freund. On conjugate-gradient type methos and polynomial preconditioners for a class of complex non-hermitian matrices. *Numer. Math.*, 57:285–312, 1990.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, second edition, 1989.

[6] L. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. M.I.T. Press, Cambridge, Massachusetts, 1988.

[7] F. W. Grover. *Inductance Calculations, Working Formulas and Tables*. Dover Publications, New York, 1962.

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

[9] H. A. Haus and J. R. Melcher. *Electromagnetic Fields and Energy*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[10] C. Hoer and Carl Love. Exact inductance equations for rectangular conductors with applications to more complicated geometries. *J. Res. Natl. Bureau Standards*, 69C:127–137, 1965.

[11] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985.

[12] M. Kamon, M. J. Tsuk, and J. White. Fasthenry, a multipole-accelerated 3-d inductance extraction program. In *Proceedings of the ACM/IEEE Design Automation Conference*, Dallas, June 1993.

[13] K. Nabors and J. White. Fastcap: A multipole accelerated 3-D capacitance extraction program. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, November 1991.

[14] K. Nabors and J. White. Fast capacitance extraction of general three-dimensional structures. *IEEE Trans. on Microwave Theory and Techniques*, June 1992.

[15] A. E. Ruehli. Inductance calculations in a complex integrated circuit environment. *IBM J. Res. Develop.*, 16:470–481, September 1972.

[16] A. E. Ruehli. Survey of computer-aided electrical analysis of integrated circuit interconnections. *IBM Journal of Research and Development*, 23(6):626–639, November 1979.

[17] A. E. Ruehli and P. A. Brennan. Efficient capacitance calculations for three-dimensional multiconductor systems. *IEEE Transactions on Microwave Theory and Techniques*, 21(2):76–82, February 1973.

[18] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.

[19] W. T. Weeks, L. L. Wu, M. F. McAllister, and A. Singh. Resistive and inductive skin effect in rectangular conductors. *IBM Journal of Res. and Develop.*, 23(6):652–660, November 1979.