EFFICIENT NUMERICAL METHODS FOR CAPACITANCE EXTRACTION BASED ON BOUNDARY ELEMENT METHOD

A Dissertation

by

SHU YAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Computer Engineering

EFFICIENT NUMERICAL METHODS FOR CAPACITANCE EXTRACTION BASED ON BOUNDARY ELEMENT METHOD

A Dissertation

by

SHU YAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Weiping Shi
	Vivek Sarin
Committee Members,	Hank Walker
	Krzysztof Arkadiusz Michalski
Head of Department,	Costas Georghiades

December 2005

Major Subject: Computer Engineering

ABSTRACT

Efficient Numerical Methods for Capacitance Extraction

Based on Boundary Element Method. (December 2005)

Shu Yan, B.S., Tsinghua University;

M.S., Tsinghua University

Co–Chairs of Advisory Committee: Dr. Weiping Shi
Dr. Vivek Sarin

Fast and accurate solvers for capacitance extraction are needed by the VLSI industry in order to achieve good design quality in feasible time. With the development of technology, this demand is increasing dramatically. Three-dimensional capacitance extraction algorithms are desired due to their high accuracy. However, the present 3D algorithms are slow and thus their application is limited. In this dissertation, we present several novel techniques to significantly speed up capacitance extraction algorithms based on boundary element methods (BEM) and to compute the capacitance extraction in the presence of floating dummy conductors.

We propose the PHiCap algorithm, which is based on a hierarchical refinement algorithm and the wavelet transform. Unlike traditional algorithms which result in dense linear systems, PHiCap converts the coefficient matrix in capacitance extraction problems to a sparse linear system. PHiCap solves the sparse linear system iteratively, with much faster convergence, using an efficient preconditioning technique. We also propose a variant of PHiCap in which the capacitances are solved for directly from a very small linear system. This small system is derived from the original large linear system by reordering the wavelet basis functions and computing an approximate LU factorization. We named the algorithm RedCap. To our knowledge, RedCap is the first capacitance extraction algorithm based on BEM that uses a direct method to

solve a reduced linear system.

In the presence of floating dummy conductors, the equivalent capacitances among regular conductors are required. For floating dummy conductors, the potential is unknown and the total charge is zero. We embed these requirements into the extraction linear system. Thus, the equivalent capacitance matrix is solved directly. The number of system solves needed is equal to the number of regular conductors.

Based on a sensitivity analysis, we propose the selective coefficient enhancement method for increasing the accuracy of selected coupling or self-capacitances with only a small increase in the overall computation time. This method is desirable for applications, such as crosstalk and signal integrity analysis, where the coupling capacitances between some conductors needs high accuracy. We also propose the variable order multipole method which enhances the overall accuracy without raising the overall multipole expansion order. Finally, we apply the multigrid method to capacitance extraction to solve the linear system faster.

We present experimental results to show that the techniques are significantly more efficient in comparison to existing techniques.

To my dearest parents and grandpa.

ACKNOWLEDGMENTS

I am thankful to many people for the completion of the thesis and the entire PhD journey. The first person I wish to thank is my supervisor, Prof. Weiping Shi, who has led me into the area of computer engineering and guided me through the study and research in the past five years. I also want to express my deepest gratefulness to Prof. Vivek Sarin, who has inspired me to continue the study and research. Both professors have patiently taught me how to become a good researcher, by kindly correcting me, constantly encouraging me, and always being very cooperative. I believe I would continue to benefit from the spirits and wisdom they have demonstrated in the rest of my life. Besides, I would thank Prof. Walker and Prof. Michalski for being in my committee and being very supportive for my study. I also appreciate Prof. Jianguo Li, Prof. Peng Li and Prof. Jiang Hu for the valuable advice and help.

Looking back on these five tough but rewarding years, I feel very lucky to have spent them with the computer engineering group. All those brain-storming discussions, group meetings, hardworking days and nights, together with all the laughters and gatherings, will all remain part of my best memories. I sincerely wish all our group members to have a very successful career and a very happy life!

My gang of friends is my biggest fortune. All of them are so important in my life and have given me the most valuable help and support at one time or another. It is hard to imagine what life would be without them.

I am in great debt to my beloved parents, for the unchangeable confidence and patience they have for me, and for the unconditional love and support they have given me at all the ups and downs in my life. I wish I could have finished the PhD study sooner, so that my dearest grandpa could share the happiness. I want to give my special thanks to him for teaching me to be brave and optimistic.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

Many aspects of VLSI design, such as interconnect delay estimation and signal integrity analysis require understanding the electromagnetic properties of complex structures [1]. Consequently, accurate parasitic extraction is critical to the analysis and design of VLSI chips, packaging and MEMS.

Capacitance is one of the important parasitics. Most existing capacitance extraction methods fall into two categories: library look-up, where the layout is divided into sections and matched against a pre-characterized library to derive the parasitic value, and field solver, where the electromagnetic field is computed to derive the capacitance. Although the library methods are faster, they are applicable only to regular structures such as interconnects on ICs. As the need for modeling large and complex package structures increases, the demand for fast and accurate 3D capacitance extraction tools is increasing.

The capacitance of an m-conductor geometry is summarized by an $m \times m$ capacitance matrix $\mathbf{C}$. To determine the j-th column of the capacitance matrix, we compute the surface charges produced on each conductor by raising conductor j to unit potential while grounding the other conductors. Then C_{ij} is numerically equal to the charge on conductor i. This procedure is repeated m times to compute all columns of $\mathbf{C}$.

The journal model is *IEEE Transactions on Automatic Control.*

A. Boundary element method

The Boundary Element Method (BEM) [2] solves field problems by solving an equivalent source problem. In the case of electric fields it solves for equivalent charge, while in the case of magnetic fields it solves for equivalent currents. BEM uses an integral formulation of Maxwell's Equations, which allow for highly accurate field calculations. BEM is the basis of many capacitance extraction algorithms [3, 4, 5, 6, 7]. Our work is based on BEM as well.

For conductors in uniform media, each of the m potential problems can be solved using an equivalent free-space formulation where the conductor-dielectric interface is replaced by a layer of unknown charge density σ. The charge layer satisfies the integral equation

$$\psi(x) = \int_{surfaces} \sigma(x') \frac{1}{4\pi\epsilon_0 \|x - x'\|} d\alpha', \qquad x \in surfaces, \tag{1.1}$$

where $\psi(x)$ is the known conductor surface potential, $d\alpha'$ is the incremental conductor surface area, $x' \in d\alpha'$, and $\|x - x'\|$ is the Euclidean distance between x and x'. A Galerkin scheme is often used to solve (1.1) numerically for σ. In this approach, the conductor surfaces are divided into n small panels, $A_1, \ldots, A_n$, and a dense linear system is constructed:

$$\mathbf{P_{cc}} \mathbf{q_c} = \mathbf{v_c}, \tag{1.2}$$

where $\mathbf{q_c} \in \Re^n$ is the unknown vector of conductor panel charges, $\mathbf{v_c} \in \Re^n$ is the vector of known conductor panel potentials, and $\mathbf{P_{cc}} \in \Re^{n \times n}$ is the potential coefficient matrix. Each entry of $\mathbf{P_{cc}}$ is defined as

$$p_{ij} = \frac{1}{a_i} \frac{1}{a_j} \int_{A_i} \int_{A_j} \frac{1}{4\pi\epsilon_0 \|x_i - x_j\|} \, d\alpha_j d\alpha_i \tag{1.3}$$

for panels A_i and A_j, where a_i and a_j are panel areas.

To solve the problem with multiple dielectrics, we apply the equivalent charge

approach [8, 9], which solves the equivalent field problem in free-space with the same boundary condition as for the original problem in multiple dielectrics. It uses the free-space Green's function in conjunction with total charge on the conductor surfaces and polarization charge on the dielectric-dielectric interfaces. Thus, the potential produced is given by

$$\psi(x) = \int_{S_c} \sigma_c(x') \frac{1}{4\pi\epsilon_0 \|x - x'\|} d\alpha' + \int_{S_d} \sigma_d(x') \frac{1}{4\pi\epsilon_0 \|x - x'\|} d\alpha', \tag{1.4}$$

where σ_c and σ_d are the charge densities on the conductor surfaces S_c and the dielectric-dielectric interfaces S_d, respectively. The interface condition

$$\epsilon_a \frac{\partial \psi_a(x)}{\partial n_a} - \epsilon_b \frac{\partial \psi_b(x)}{\partial n_a} = 0 \tag{1.5}$$

should be satisfied at any point x on the dielectric-dielectric interface. Here, as shown in Fig. 1, ϵ_a and ϵ_b are the permittivities of the two adjacent regions a and b, n_a is the normal to the dielectric-dielectric interface at x pointing to dielectric a, and ψ_a and ψ_b are the potentials at x in the dielectrics a and b, respectively.

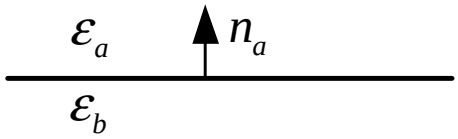


Fig. 1. Notations of the dielectric-dielectric interface.

Using the same numerical approach as that for the uniform case, we transform (1.4) and (1.5) into the linear system

$$\begin{bmatrix} \mathbf{P_{cc}} & \mathbf{P_{cd}} \\ \mathbf{E_{dc}} & \mathbf{E_{dd}} \end{bmatrix} \begin{bmatrix} \mathbf{q_c} \\ \mathbf{q_d} \end{bmatrix} = \begin{bmatrix} \mathbf{v_c} \\ 0 \end{bmatrix}, \tag{1.6}$$

where $\mathbf{q_c}$ and $\mathbf{q_d}$ are the vectors of charges on the conductor panels and dielectric-

dielectric interface panels, respectively, and $\mathbf{v_c}$ is the vector of potential on conductor panels. The entries of $\mathbf{P_{cc}}$ and $\mathbf{P_{cd}}$ are defined in (1.3). The entries of $\mathbf{E_{dc}}$ and off-diagonal entries of $\mathbf{E_{dd}}$ are defined as

$$e_{ij} = (\epsilon_a - \epsilon_b)\frac{\partial}{\partial n_a}\frac{1}{a_i}\frac{1}{a_j}\int_{A_i}\int_{A_j}\frac{1}{4\pi\epsilon_0\|x_i - x_j\|}\,d\alpha_j d\alpha_i, \tag{1.7}$$

and the diagonal entries of $\mathbf{E_{dd}}$ are defined as

$$e_{ii} = (\epsilon_a + \epsilon_b)\frac{1}{2a_i\epsilon_0}.$$

BEM applied to the capacitance extraction problem gives rise to dense linear systems (1.2) and (1.6). We reformulate the two linear systems as follows.

$$\mathbf{Pq} = \mathbf{v}. \tag{1.8}$$

The matrix $\mathbf{P}$ is symmetric for uniform dielectric and unsymmetric for multiple dielectrics.

According to the above discussion, the BEM involves the following three steps.

1. Panel discretization
2. Calculating $\mathbf{P}$
3. Solving $\mathbf{Pq} = \mathbf{v}$

The three steps influence the speed and accuracy of the extraction algorithms.

Another important family of field solvers uses Finite Element Method (FEM) and Finite Difference Method (FDM) to solve differential formulations. Those methods discretize the space into small regions and convert the problem into sparse linear systems. However, since the whole space are involved, the dimension of the linear system is much larger than the linear system from BEM.

B. Previous work

The dense linear system (1.8) is often solved by iterative techniques such as GMRES (Generalized Minimal Residual) and CG (Conjugate Gradient) methods. Each iteration requires the product of the dense coefficient matrix of order n with a vector, which takes $O(n^2)$ time and $O(n^2)$ memory.

Great efforts have been made to accelerate computation and reduce memory requirements of the matrix-vector product. By exploiting the fast decaying nature of the Green's function, the matrix-vector products can be computed efficiently. For example, based on the fast multipole approximation [10], FastCap [4] reduces the complexity to $O(n)$. Based on the hierarchical refinement, the HiCap algorithm [3] also reduces the complexity to $O(n)$. With the regular gridding of the space and Fast Fourier Transformation, the pre-corrected FFT method [6] solves the problem in $O(n \log n)$ time. Based on singular value decomposition, the IES^3 method [5] approximate the linear system using its dominant singular values and corresponding vectors. The complexity is reduced to $O(n \log n)$. However, solving the linear system is still a formidable task because of the slow rate of convergence of the iterative solver, especially when n is very large.

Unlike the methods mentioned above that solve a dense system, there are other methods that solve a sparse system. In [11], the BEM in (1.2) was represented and approximated in a wavelet basis. In [12] and [13], two different wavelet bases are proposed. In each case, the linear system is constructed directly and sparsified using thresholding techniques. Iterative methods were used to solve the sparse linear system. Compared with the BEMs without any acceleration, these wavelet-based methods are significantly faster. However, a comparison of their efficiency with that of other extraction methods with acceleration is not reported. In addition, the nu-

merical quadrature involved in these methods is complicate and needs more analysis as pointed out by the author [12].

In addition to the methods using iterative solvers, there are attempts to solve the linear system of BEM without iterations. In [14] and [15], sparse representations of the linear systems are constructed using low rank approximation of the off diagonal blocks. These sparse representations are solved by LU factorization. The time complexity of those algorithms is $O(n^\gamma)$, where γ ranges from 2.20 to 2.89, which is slower than most iterative methods.

C. Our motivation

In this dissertation, we try to find faster capacitance extraction algorithms while maintaining or improving the accuracy.

First, to avoid the slow convergence of solving the dense linear system, we propose the PHiCap method, a **P**reconditioned **Hi**erarchical refinement method for **Cap**acitance extraction, which transforms the dense linear system of HiCap into a sparse linear system in wavelet basis. The sparse linear system is solved with faster convergence using efficient preconditioners based on incomplete LU or Cholesky factorizations. Unlike the wavelet approximation methods in [11, 12, 13], PHiCap first approximates the problem using the multipole approach, and then applies an accurate wavelet transformation. Therefore, PHiCap combines both the fast multipole and the wavelet method.

We exploit the advantage of sparsification further by **Red**ucing the sparse linear system of PHiCap into a small sub-system, which is solved for **Cap**acitance by Gaussian elimination (RedCap). RedCap does not require any iterative method, and avoids solving the original large system. The time complexity of RedCap depends on

the sparse transformation, which takes $O(n \log n)$ time if we use the transformation in PHiCap. Compared to other iteration-free algorithms such as [14] and [15], RedCap is more efficient in terms of complexity. To the best of our knowledge, RedCap is the most efficient iteration-free method for capacitance extraction.

In the presence of floating dummy conductors, the equivalent capacitances among regular conductors are required. For floating dummy conductors, the potential is unknown and the total charge is zero. We embed these requirements into the extraction linear system. Thus, the equivalent capacitance matrix is solved directly. The number of system solves needed is equal to the number of regular conductors.

In the panel discretization step, approximation error is introduced due to the assumption of uniform charge distribution on each panel. This is the dominant approximation error of BEM. Although fine discretization reduces error, it also results in large linear systems and unacceptable computation time. We propose a multigrid-like scheme to acquire a good initial solution based on coarse discretization. This helps to preserve high accuracy while reducing the computation cost greatly.

When calculating $\mathbf{P}$, approximation error is introduced due to the numerical integration of (1.3) or (1.7) and truncation of multipole expansion. To reduce error without significant increase in computation time, we propose the variable order multipole scheme and the selective coefficient enhancement scheme.

D. Organization of the dissertation

The rest of the dissertation will present our work on fast capacitance extraction. Chapter II will introduce the HiCap [3] algorithm and the preconditioned iterative solver, which form the basis of our work. The PHiCap algorithm [16], the RedCap algorithm [17] will be explained in Chapter III and Chapter IV, respectively. In

Chapter V, the fast extraction algorithm handling floating dummy conductors will be introduced. The selective coefficient enhancement algorithm, variable order multipole and multigrid methods will be described in the subsequent chapters. Finally, we conclude our study in Chapter IX.

The major contributions of the dissertation include several novel algorithms that improve the existing capacitance extraction methods in terms of running time and accuracy.

CHAPTER II

PRELIMINARIES

In this chapter, we introduce the two techniques used in the dissertation: the HiCap algorithm [3] and the preconditioned iterative linear solver [19].

A. HiCap algorithm

The HiCap algorithm is a capacitance extraction algorithm based on adaptive hierarchical refinement, which exploits the far field approximation of electrostatic field. The field due to a cluster of charges at some distance can be approximated with a single term. As a result, instead of $O(n^2)$ nonzero entries which denote the interactions between all pairs of charges, the linear system from HiCap has $O(n)$ block entries. Consequently, the HiCap algorithm reduces the complexity of matrix-vector product to $O(n)$. Experimental results show that HiCap is significantly faster and more memory efficient compared to FastCap [4] and QuickCap [7].

1. Potential matrix approximation

First, we introduce the hierarchical refinement used in HiCap. The following procedure `Refine` describes the recursive refinement procedure that subdivides a large panel into a hierarchy of small panels, and builds a hierarchical representation of the potential coefficient matrix.

```
Refine(Panel Ai, Panel Aj)
{
  Pij = PotentialEstimate(Ai, Aj);
  Ri = longest side of Ai;
```

```
    Rj = longest side of Aj;

    if ((Pij*Ri < Peps) && (Pij*Rj < Peps))
      RecordInteraction(Ai, Aj);
    else if (Ri > Rj) {
      Subdivide(Ai);
      Refine(Ai.left, Aj);
      Refine(Ai.right, Aj);
    } else {
      Subdivide(Aj);
      Refine(Ai, Aj.left);
      Refine(Ai, Aj.right);
    }
  }
```

Procedure `PotentialEstimate` returns an estimate of the potential coefficient for two panels defined in (1.3) and(1.7). If the estimated coefficient is less than the user provided error bound `Peps` (P_ϵ), then the panels are allowed to interact at this level. The recursion is terminated and the interaction is recorded between the two panels by procedure `RecordInteraction`. However, if the estimate is greater than $P\epsilon$, then the estimate may not be accurate. In this case, the panel with the larger area, say `Ai`, is to be subdivided into `Ai.left` and `Ai.right`. The procedure `Refine` is called recursively. Procedure `Subdivide` subdivides a panel into two small panels. The subdivision hierarchy is stored in a binary tree where each node has two pointers, `left` and `right`, pointing to the two small panels.

Since a panel may be refined against many other panels, the actual subdivision of a panel may have occurred previously. When this happens, `Subdivide` uses the same subdivision.

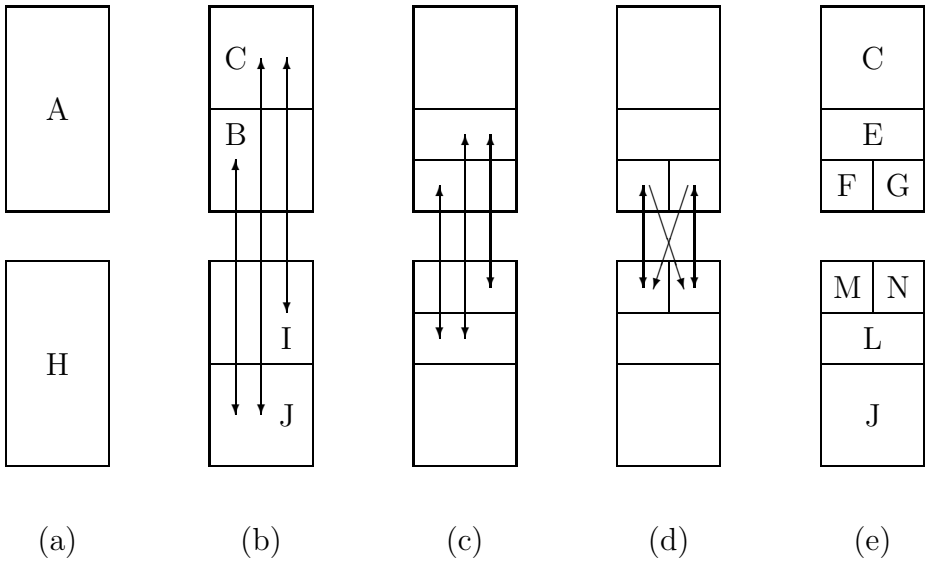


Fig. 2. Partition conductor surfaces into panels.

Fig. 2 shows the refine process of two conductor surfaces. Given two conductor surfaces A and H in (a), assume the estimated coefficient between A and H is greater than the user provided error bound P_ϵ, so A is subdivided into $B \cup C$, and then H is subdivided into $I \cup J$ in (b). Now assume the estimates between BJ, CI and CJ are less than P_ϵ, but estimate BI is greater than P_ϵ. Then we record interactions BJ, CI and CJ at this level, and further subdivide panels B and I. The final panels are shown in (e). The self-potential coefficient P_{ii} is computed at this time.

Fig. 3 shows the hierarchical data structure [18] produced by `Refine`, and associated potential coefficients produced by `RecordInteraction`. The panels are stored as nodes in the tree, and the coefficients are stored as links between the nodes. The value of each coefficient is stored as a floating-point number associated with the link.

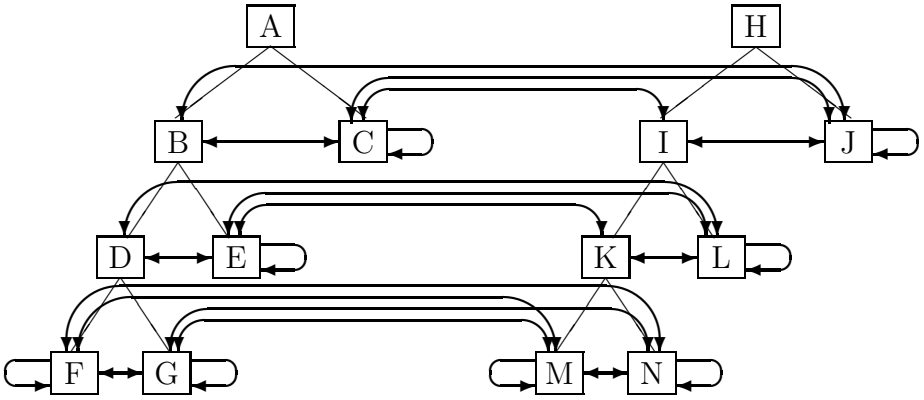


Fig. 3. Hierarchical data structure and potential coefficients.

Each tree represents one conductor surface, each non-leaf node represents one panel further subdivided, and each leaf node represents one panel not further subdivided. The union of all the leaf nodes completely cover the surfaces of the conductors. Each horizontal link represents one pair of potential coefficients defined in (1.3) and (1.7). Each self-link represents one self-potential coefficient.

Fig. 4 shows the block matrix represented by the links of Fig. 3. Each block entry represents one interaction between panels. Note that there are total 8 panels for the two conductors, so an explicit representation of the coefficient matrix would require 64 entries. However, the block matrix has only 40 entries. Furthermore, if we use uniform grid discretization, then there would be a total of 16 panels, and 256 entries.

It has been shown that under a very general condition, the matrix of the HiCap algorithm contains $O(n)$ interactions [3]. The accuracy is controllable via P_ϵ.

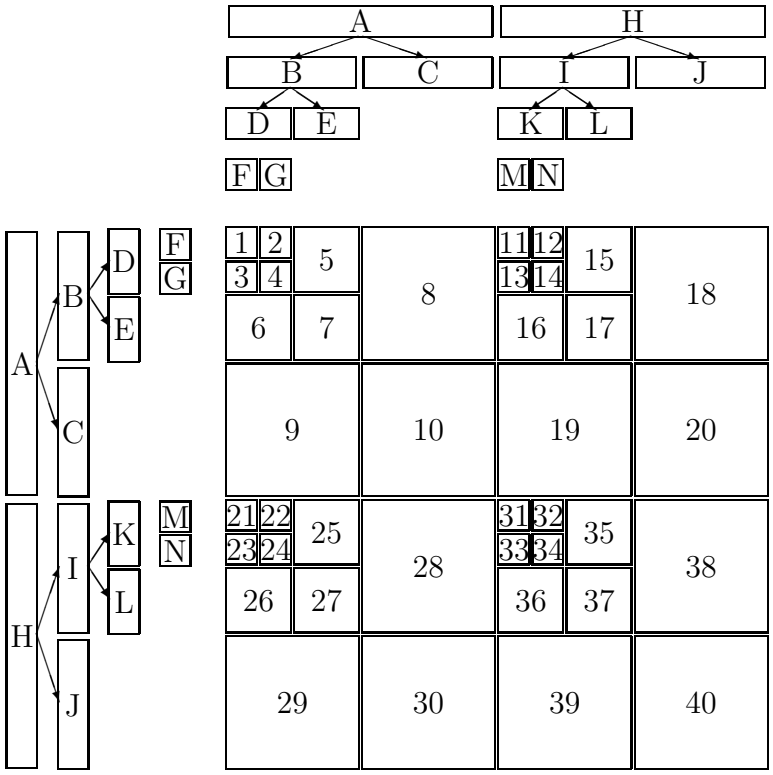


Fig. 4. Potential coefficient matrix with block entries.

2. Matrix-vector multiplication

Matrix-vector multiplication is required by the iterative solver. Given the potential matrix approximated as explained in section 1, matrix-vector multiplication includes the three steps.

The first step computes the charge of all panels in the tree. The charge of a leaf panel `Ai` is given by `Qi` from **q**. The charge of an non-leaf panel is the sum of the charge of its children panels. This calculation can be done in a single depth-first traversal of the tree, propagating the charge upward. In other words, to compute the charge for each panel, the charges of its children panels are computed first, and then the charge of the panel equals the sum of the charge of its children panels. The time

to compute the charge on all panels is linear in the number of panels in the tree. The following procedure `AddCharge` describes this step.

```
AddCharge(Panel Ai)
{
  if (Ai is leaf)
    Ai.charge = Qi;
  else {
    AddCharge(Ai.left);
    AddCharge(Ai.right);
    Ai.charge = Ai.left->charge + Ai.right->charge;
  }
}
```

The second step computes the potential on each panel A_i due to its interacting panels. This can be computed by adding the product of potential coefficient P_{ij} with charge at A_j, for every A_j that has interaction with A_i. The time to compute the charge on all nodes is linear in terms of the number of links in the tree. The following procedure `CollectPotential` describes this step.

```
CollectPotential(Panel Ai)
{
  for all Aj such that AiAj has interaction {
    Ai.potential = Ai.potential + Aj.charge*Pij;
    if (Ai is not leaf) {
      CollectPotential(Ai.left);
      CollectPotential(Ai.right);
    }
```

```
    }
  }
```

The third step distributes the potential from the non-leaf nodes to the leaf nodes. This is done by another depth first traversal of the tree that propagates potential down to the leaf nodes. Each non-leaf node adds its accumulated potential to its children's potential, recursively. The time complexity of this step is linear in the number of nodes in the tree. The following procedure `DistributePotential` describes this step.

```
DistributePotential(Panel Ai)
{
  if (Ai is not leaf) {
    Ai.left->potential = Ai.left->potential + Ai.potential;
    Ai.right->potential = Ai.right->potential + Ai.potential;
    DistributePotential(Ai.left);
    DistributePotential(Ai.right);
  }
}
```

The total time for the matrix-vector product is linear in the number of nodes and links. It is well known that for any binary tree with n leaves, there are exactly $n-1$ non-leaf nodes. Therefore, the time is $O(n)$, where n is the number of leaf panels.

3. Solving the linear system

The Generalized Minimum Residual (GMRES) method or the Conjugate Gradient (CG) method can be used to solve the linear system. According to the section 2, the

complexity of the matrix-vector multiplication is $O(n)$, which is a major improvement over the methods that do not use far field approximation. From the experiments in [3], it is clear that HiCap is much more efficient than other methods. However, since the linear system is dense even with the hierarchical refinement approximation, it is hard to obtain effective preconditioners. Without preconditioning, the iterative solvers converge slowly.

4. Summary

The HiCap algorithm approximates the far field effects by adaptively refining the conductor surfaces. The linear system is dense with $O(n)$ block entries. As a result, the complexity of the matrix-vector multiplication at each iteration is reduced to $O(n)$. The rate of convergence is slow when the problem size is large. However, since the linear system is dense, efficient preconditioners are hard to construct.

B. Preconditioned iterative solver

Iterative solvers are commonly used to solve the large-scale equations. The computational cost of the iterative solvers is decided by both the cost of matrix-vector multiplication and the number of iterations required for convergence [19].

As explained earlier in chapter I and the previous section, great efforts have been made to reduce the complexity of the matrix-vector product. The capacitance extraction algorithm such as FastCap [4], HiCap [3], pFFT [6] and IES3 [5] successfully reduce the complexity of matrix-vector product from $O(n^2)$ to $O(n \log n)$ or $O(n)$. However, most of the algorithms do not apply efficient preconditioners to accelerate the convergence of the iterative solver.

For the linear systems, especially those with poor condition number, a precon-

ditioning matrix $\mathbf{Q}$ is desired to accelerate the convergence rate. We use (2.1) to represent the general linear system

$$\mathbf{Ax} = \mathbf{b}. \tag{2.1}$$

Consider the left-preconditioning system

$$\mathbf{QAx} = \mathbf{Qb}. \tag{2.2}$$

Ideally, $\mathbf{QA}$ should be as close to the identity matrix as possible.

For the dense linear systems, it is hard to find a good preconditioner $\mathbf{Q}$. However, for the sparse linear systems, the incomplete Cholesky (for symmetric linear systems) or LU (for unsymmetric linear systems) factors are very efficient preconditioners.

Let $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ be the incomplete triangular factors of $\mathbf{A}$. Since $\hat{\mathbf{L}}\hat{\mathbf{U}}$ is close to $\mathbf{A}$, $\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}$ can be used a preconditioner. The corresponding preconditioned linear system is

$$\hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{Ax} = \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1}\mathbf{b}. \tag{2.3}$$

In this dissertation, we use the incomplete LU or incomplete Cholesky factorization as preconditioners to solve the sparse linear system of PHiCap.

CHAPTER III

PHICAP ALGORITHM

A. Introduction

In the previous chapter, we introduced the HiCap algorithm which is based on hierarchical refinement. The corresponding linear system has $O(n)$ block entries, instead of $O(n^2)$ entries. Consequently, the matrix-vector multiplication is accelerated to $O(n)$. In spite of the hierarchical refinement approximation, the linear system is dense. As a result, effective preconditioners are hard to obtain and the iterative solvers converge slowly.

Unlike the methods that solve a dense system, such as fast multipole approximation [4, 3], pre-corrected FFT method [6, 20] or singular value decomposition method [5], there are other methods that solve a sparse system. In [11], the BEM in (1) was represented and approximated in a wavelet basis. In [12] and [13], two different wavelet bases are proposed. In each case, the linear system is constructed directly and sparsified using thresholding techniques. Iterative methods were used to solve the sparse linear system. Compared with the BEMs without any acceleration, these wavelet-based methods are significantly faster. However, a comparison of their efficiency with that of other extraction methods with acceleration is not reported. In addition, the numerical quadrature involved in these methods is complicate and needs more analysis as pointed out by the author [12].

In this chapter, we present PHiCap, a **P**reconditioned **Hi**erarchical algorithm for

---

*

Capacitance extraction. Starting from the HiCap algorithm introduced in the previous chapter, we propose a wavelet transformation that transforms the dense linear system with $O(n)$ block entries from HiCap into a sparse linear system with $O(n)$ nonzero entries. To solve the sparse linear system, we use preconditioned GMRES or CG iterative methods (see, e.g., [21]). Incomplete LU factorization or incomplete Cholesky factorization is used as a preconditioner. The rate of convergence of the iterative methods increases dramatically by using these preconditioners.

B. Algorithm outline

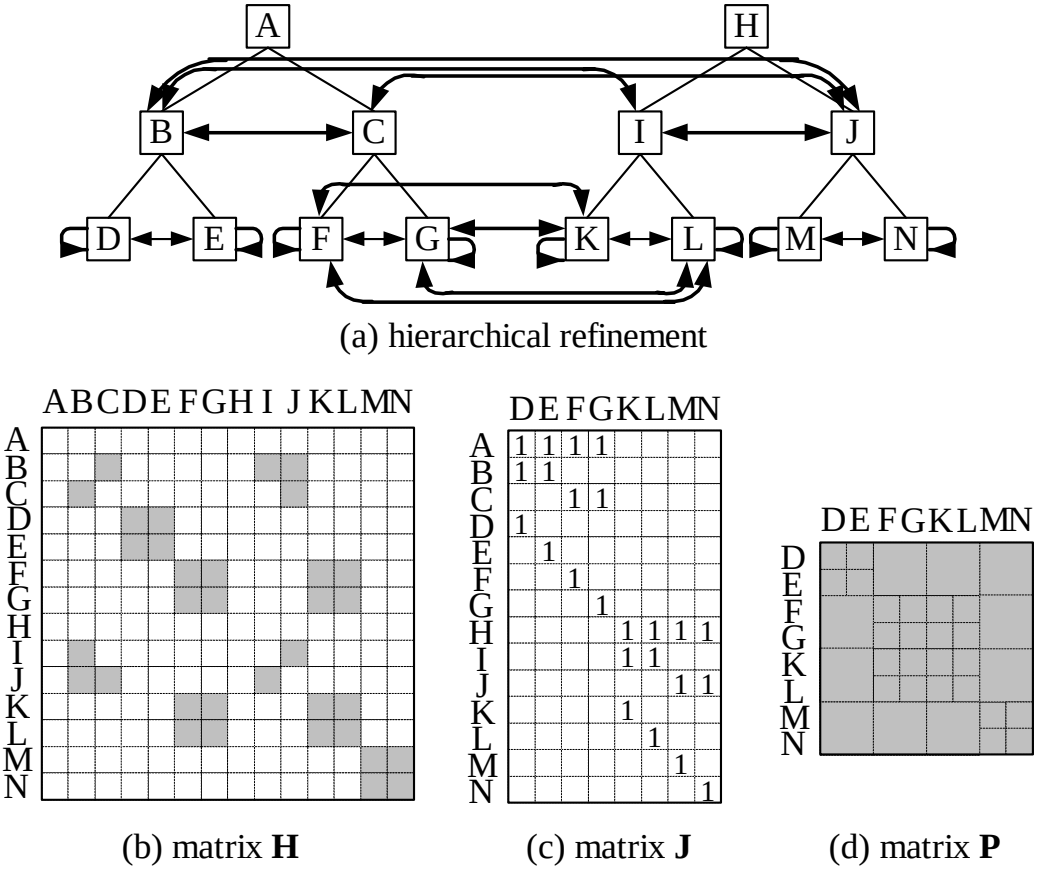
The PHiCap Algorithm

1. Compute the factorization $\mathbf{P} = \mathbf{J^T H J}$.
2. Transform the dense system $\mathbf{Pq} = \mathbf{v}$ to an equivalent sparse system $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$.
3. Compute an incomplete factorization preconditioner for $\tilde{\mathbf{P}}$.
4. Solve $\tilde{\mathbf{P}}\tilde{\mathbf{q}} = \tilde{\mathbf{v}}$ using preconditioned CG or GMRES method.
5. Compute capacitance.

In the following sections, we will explain each step in detail.

C. Factorization of $\mathbf{P}$ (Step 1)

We use the HiCap algorithm [3] as explained in section II.A to construct the hierarchical data structure. Fig. 5(a) shows an example of the hierarchical data structure. Now, we show that the hierarchical data structure can be used to obtain the factorization of the potential coefficient matrix $\mathbf{P}$.



(a) hierarchical refinement

(b) matrix **H**

(c) matrix **J**

(d) matrix **P**

Fig. 5. Hierarchical refinement of conductors and the matrix of coefficients.

Let N be the number of all panels in the hierarchy, $\mathbf{q}_N \in \Re^N$ be the vector of panel charges, $\mathbf{v}_N \in \Re^N$ be the vector of panel potentials due to the coefficients directly linked to each panel, and $\mathbf{H} \in \Re^{N\times N}$ be the matrix of all coefficients (Fig. 5(b)). Matrix $\mathbf{H}$ has $O(N)$ nonzero entries. Each nonzero entry represents a coefficient link in the hierarchy. Let $\mathbf{J} \in \mathbb{R}^{N\times n}$ be the matrix representing the tree structure, as shown in Fig. 5(c). Each row of $\mathbf{J}$ corresponds to a panel, either leaf or non-leaf, and each column corresponds to a leaf panel. Entry (i, j) of $\mathbf{J}$ is 1 if panel i contains the leaf panel j, and 0 otherwise.

In the HiCap algorithm, charge and potential of leaf panels are chosen as basis functions. Let n be the number of leaf panels, $\mathbf{q}$ and $\mathbf{v} \in \Re^n$ be the charge and potential vectors of the leaf panels, respectively.

Accordingly, each of the three steps of the matrix-vector multiplication, discussed in section II.A.2, can be represented by the following matrix operation.

1. AddCharge: $\mathbf{q}_N = \mathbf{Jq}$
2. CollectPotential: $\mathbf{v}_N = \mathbf{Hq}_N$
3. DistributePotential: $\mathbf{v} = \mathbf{J^T v}_N$

Combine the three step, we have

$$\mathbf{v} = \mathbf{J^T HJq}.$$

As a result, we can represent the coefficient matrix $\mathbf{P}$ using its factorization: $\mathbf{P} = \mathbf{J^T HJ}$. Here $\mathbf{P}$ is a dense matrix with $O(n)$ block entries (Fig. 5(d)).

D. Transforming the linear system (Step 2)

1. Overview

In this section, we transform the dense linear system (1.8) from HiCap into a sparse system using a set of appropriate basis.

In traditional BEM, such as HiCap [3], the basis consists of voltage and charge of each leaf panel. For any leaf panel A_i, q_i represents the charge and v_i represents the potential on A_i.

PHiCap uses a different basis, which is equivalent to Haar wavelets. For every pair of sibling panels, which could be either leaf or non-leaf, we arbitrarily name one as "older" and one as "younger". For each "older" sibling A_i, we define a variable $\tilde{q}_i$ to represent the charge difference between A_i and its "younger" sibling, and a variable $\tilde{v}_i$ to represent the potential difference between A_i and its "younger" sibling. Fig. 6 shows the new basis functions for the example in Fig. 5. In the new basis, the potential vector consists of entries for the potential difference between sibling panels, and entries for the potential of root panels. The charge vector consists of the charge difference between sibling panels along with the charge of the root panels.

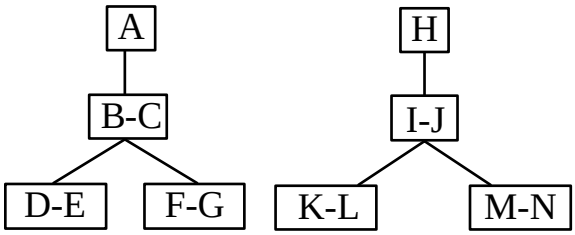


Fig. 6. Hierarchy of the basis functions of PHiCap.

Now, we briefly outline the sparse transformation. Details will be given in the following sub-sections. Our transformation is constructed based on the factorization $\mathbf{P} = \mathbf{J^T H J}$. Since $\text{rank}(\mathbf{J}) = n$, we can always construct an orthonormal transforma-

tion $\mathbf{F} \in \mathbb{R}^{N \times N}$ (described later in this section), such that

$$\mathbf{FJ} = \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix},$$

where $\mathbf{W} \in \mathbb{R}^{n \times n}$. Thus,

$$\begin{aligned} \mathbf{P} &= \mathbf{J^T H J} \\ &= \mathbf{J^T (F^T F) H (F^T F) J} \\ &= \mathbf{(FJ)^T \left(FHF^T\right) (FJ)} \\ &= \begin{bmatrix} \mathbf{W^T} & \mathbf{0} \end{bmatrix} \left(\mathbf{FHF^T}\right) \begin{bmatrix} \mathbf{W} \\ \mathbf{0} \end{bmatrix}, \end{aligned}$$

where $\mathbf{FHF^T}$ can be represented as

$$\mathbf{FHF^T} = \begin{bmatrix} \tilde{\mathbf{P}} & \times \\ \times & \times \end{bmatrix}.$$

Here, $\tilde{\mathbf{P}}$ is a sparse $n \times n$ matrix (we show this property later), and $\times$'s denote submatrices that do not contribute to $\mathbf{P}$. Since $\mathbf{P} = \mathbf{W^T \tilde{P} W}$, the dense linear system (1.8) is transformed to the sparse system

$$\tilde{\mathbf{P}} \tilde{\mathbf{q}} = \tilde{\mathbf{v}}, \tag{3.1}$$

where $\tilde{\mathbf{q}} = \mathbf{Wq}$ and $\tilde{\mathbf{v}} = \mathbf{W^{-T} v}$. The number of nonzero entries in the sparse matrix $\tilde{\mathbf{P}}$ is $O(n)$ (we show this property later).

2. Constructing **F**

We first introduce a basic transformation that is used to construct **F**. Consider the matrix

$$\hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} 1 & 1 \\ c_k & 0 \\ 0 & c_k \end{bmatrix},$$

where c_k is a constant that depends on the height k of a node in the tree. For leaf nodes, $k = 0$.

There exists an orthonormal matrix

$$\hat{\mathbf{F}}_\mathbf{k} = \begin{bmatrix} \frac{2}{\sqrt{2({c_k}^2+2)}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} & \frac{c_k}{\sqrt{2({c_k}^2+2)}} \\ \frac{c_k}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} & -\frac{1}{\sqrt{{c_k}^2+2}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix},$$

such that

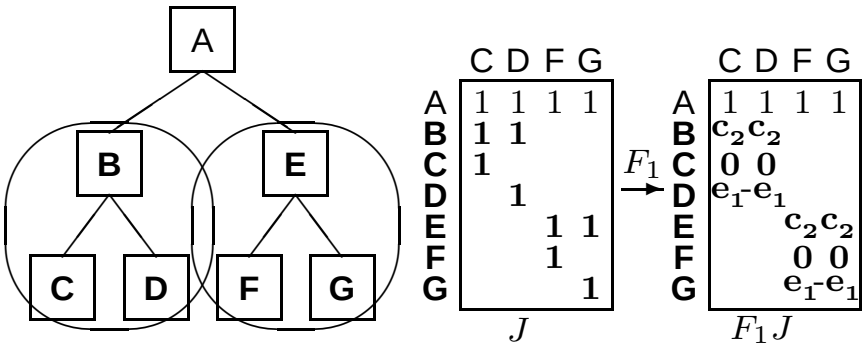
$$\hat{\mathbf{F}}_\mathbf{k}\hat{\mathbf{J}}_\mathbf{k} = \begin{bmatrix} c_{k+1} & c_{k+1} \\ 0 & 0 \\ e_k & -e_k \end{bmatrix}, \tag{3.2}$$

where

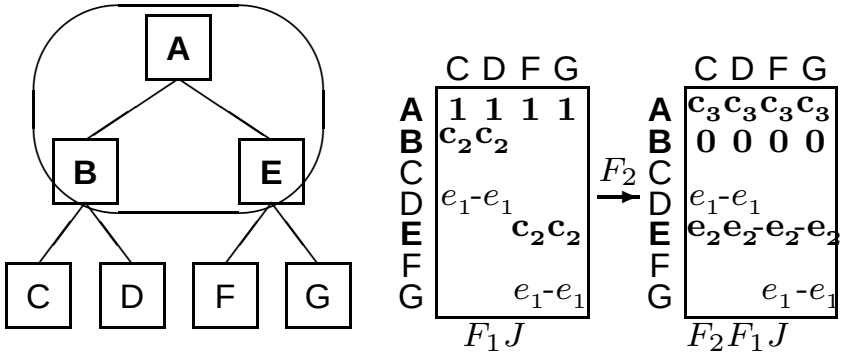
$$c_{k+1} = \sqrt{\frac{c_k^2+2}{2}}, \qquad e_k = \frac{c_k}{\sqrt{2}}, \qquad k \geq 1,$$

and $c_1 = 1$.

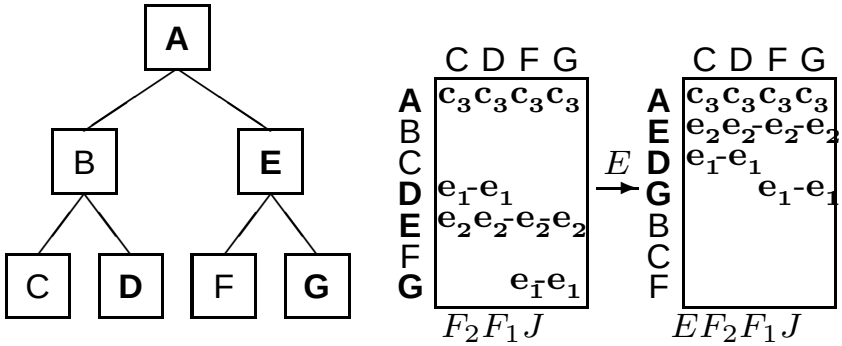
To simplify the discussion, we define an *element tree* as a tree with one root and two children. Given a hierarchical data structure and the corresponding matrix **J**, the transformation is done by a depth-first traversal of the corresponding tree, propagating the transformation upward to the root. Fig. 7 illustrates the procedure. Starting from height $k = 1$, as shown in Fig. 7(a), for each *element tree* rooted at height 1, i.e., trees (B, C, D) and (E, F, G), we can identify the corresponding $\hat{\mathbf{J}}_\mathbf{1}$



(a) Transformation with F_1



(b) Transformation with F_2



(c) Permuting rows with E

Fig. 7. Construction of transformation **F** for a tree of height 2.

blocks in $\mathbf{J}$. We construct $\mathbf{F_1}$ that transforms all $\hat{\mathbf{J}}_\mathbf{1}$ blocks to $\hat{\mathbf{F}}_\mathbf{1}\hat{\mathbf{J}}_\mathbf{1}$ blocks without changing anything else in $\mathbf{J}$. Next, as illustrated in Fig. 7(b), we identify the *element tree* at height $k = 2$, i.e., tree (A, B, E), and the corresponding $\hat{\mathbf{J}}_\mathbf{2}$ block in $\mathbf{F_1J}$. Note that the rows of A, B, and E have two instances of the $\hat{\mathbf{J}}_\mathbf{2}$ block in columns C and F and columns D and G, respectively. We construct $\mathbf{F_2}$ that transforms the $\hat{\mathbf{J}}_\mathbf{2}$ blocks to $\hat{\mathbf{F}}_\mathbf{2}\hat{\mathbf{J}}_\mathbf{2}$ blocks without changing anything else in $\mathbf{F_1J}$. In this way, the transformation is propagated to the root. Finally, as shown in Fig. 7(c), we move the nonzero rows to the top of the matrix using a permutation matrix $\mathbf{E}$. The overall transformation is given as $\mathbf{F} = \mathbf{EF_2F_1}$. It is easy to see that the nonzero rows of $\mathbf{FJ}$ correspond to the root node and nodes that are right children of other nodes. In other words, zero rows in $\mathbf{FJ}$ correspond to nodes that are left children of other nodes.

For a tree of height h, the transformation is

$$\mathbf{F} = \mathbf{EF_hF_{h-1}} \cdots \mathbf{F_2F_1},$$

where $\mathbf{F_k}$ is constructed according to the element trees at height k. Since $\mathbf{F_1}, \mathbf{F_2}, \ldots, \mathbf{F_h}$, and $\mathbf{E}$ are orthonormal, the transformation matrix $\mathbf{F}$ is orthonormal.

3. Computing $\mathbf{FHF^T}$

The matrix $\mathbf{H}$ is transformed into $\mathbf{FHF^T}$ by applying the transformations $\mathbf{F_k}$ as shown below

$$\mathbf{H_{k+1}} = \mathbf{F_kH_kF_k^T}, \qquad k = 1, 2, \ldots, h,$$

where $\mathbf{H_1} = \mathbf{H}$, and then by applying the permutation matrix $\mathbf{E}$:

$$\mathbf{FHF^T} = \mathbf{EH_{h+1}E^T}.$$

In the hierarchical data structure, this is done by a depth-first traversal of the tree, propagating the transformation upward, in a manner similar to the process of constructing the matrix **F**. For any tree rooted at R, the transformation is described below.

```
TreeTransform(Panel R)
{
  if (R is leaf)
    return;
  if (R has children)
  {
    TreeTransform(R.left);
    TreeTransform(R.right);
  }
  ElementTreeTransform(R);
}
```

In the code, R.left and R.right are the left and right children of panel R, respectively. Let R.row, R.left.row and R.right.row consist of all the links pointing to panel R, R.left and R.right, respectively, and let R.col, R.left.col and R.right.col consist of all the links starting from panel R, R.left and R.right, respectively. The addition and subtraction are vector operations, adding or subtracting the coefficients of the links panel-wise. After transformation, we reuse R.right to represent the new wavelet basis of the difference of R.right and R.left.

Fig. 8 shows the application of the element tree transform for the example in Fig. 5. We first transform the rows in **H** corresponding to panels B, D and E, where row

```
ElementTreeTransform(Panel R)
{
  // row transformation
  R.row = R.row + R.right.row + R.left.row;
  R.right.row = R.right.row - R.left.row;
  Delete R.left.row;

  // column transformation
  R.col = R.col + R.right.col + R.left.col;
  R.right.col = R.right.col - R.left.col;
  Delete R.left.col;
}
```

`B` is replaced by the summation of the three rows, row `E` is replaced by the difference of rows `D` and `E`, and row `D` is discarded (Fig. 8(a)). We then transform the columns in the updated matrix **H** corresponding to panels `B`, `D` and `E` in a similar manner (Fig. 8(b)). The matrix **H** before and after the row and column transformations of `ElementTreeTransform(B)` is shown in Fig 8(c).

The matrix **H** is stored in the hierarchical structure as in Fig. 5(a). The procedure for element trees transforms the basis and their coefficients by modifying the links in the hierarchy. From hierarchical refinement, each pair of sibling panels are linked to the similar group of panels. During transformation, the same amount of links to the right sibling panel are remained and the coefficient values are replaced by the corresponding coefficient difference of sibling panels. The links to the left sibling panel are deleted. At the mean time, the number of links to the parent panel is

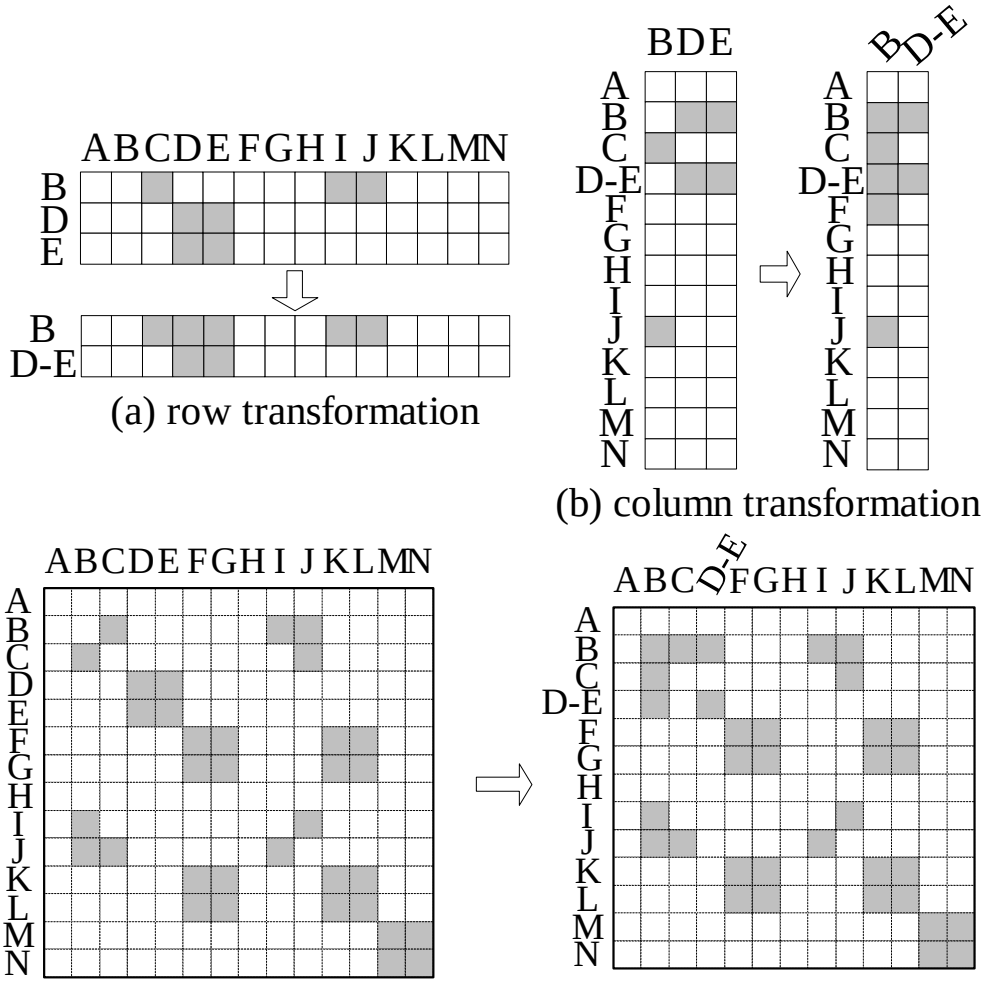


Fig. 8. Transformation for element tree `B`,`D`, and `E`.

increased by the same amount as that of link reduction to the left sibling panel. As a result, after transformation, the higher level panels have more coefficients, and the total number of coefficients remains about the same, which is $O(n)$. For the example in Fig. 5(a), the transformation results in the sparse matrix shown in Fig. 9(a). The transformed sparse linear system is given by (3.1), where $\tilde{q}$ and $\tilde{v}$ are the charge and potential vectors in the new basis, respectively.

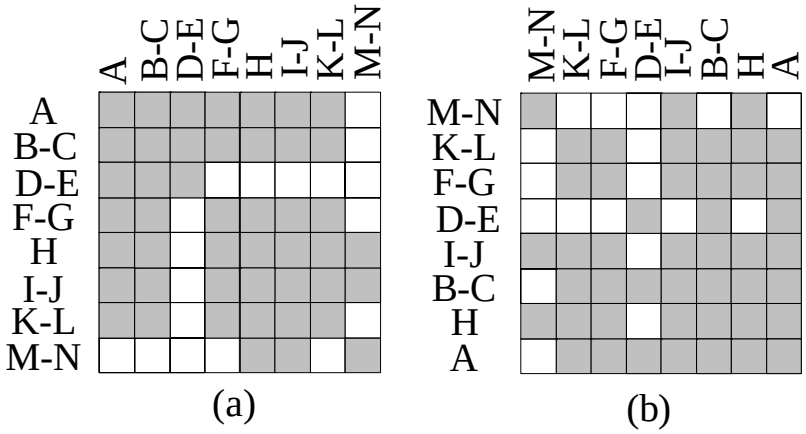


Fig. 9. The sparse pattern of $\tilde{\mathbf{P}}$. (a) with arbitrary ordering, and (b) with the proposed ordering according to the level of the new basis.

In the transformed matrix $\mathbf{FHF^T}$, we are concerned only with the submatrix $\tilde{\mathbf{P}}$ that contains the links among root nodes and right child nodes. The matrix $\tilde{\mathbf{P}}$ can be treated as a sparse matrix with the number of nonzeros that are comparable to the number of block entries in $\mathbf{P}$ (see Fig. 10).

4. Computing $\tilde{\mathbf{v}}$

The rows of the transformed matrix $\hat{\mathbf{F}}_\mathbf{k}\hat{\mathbf{J}}_\mathbf{k}$ are orthogonal. It follows that the rows of $\mathbf{W}$ are mutually orthogonal, and that $\mathbf{WW^T}$ is a diagonal matrix with values $2^k c_{k+1}^2$, where k is the height of the corresponding node in the tree, as defined in the previous

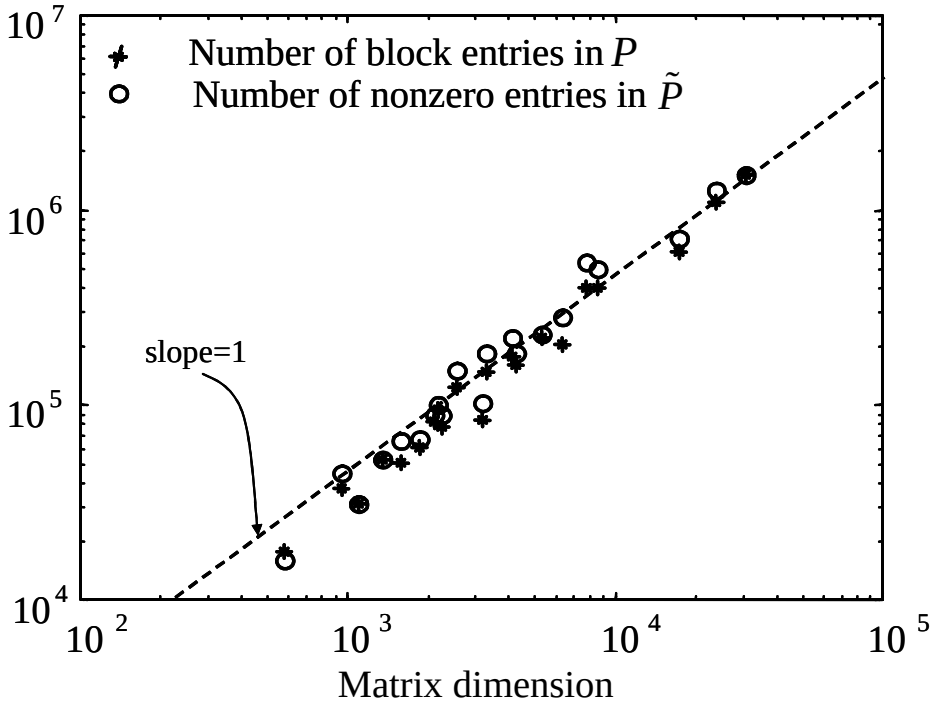


Fig. 10. The number of nonzero entries in $\tilde{\mathbf{P}}$ and the number of block entries in $\mathbf{P}$ are comparable, for various problem sizes.

subsections. This property is exploited when computing $\tilde{\mathbf{v}}$:

$$\tilde{\mathbf{v}} = \mathbf{W}^{-\mathbf{T}}\mathbf{v} = \left(\mathbf{W}\mathbf{W}^{\mathbf{T}}\right)^{-\mathbf{1}}\mathbf{W}\mathbf{v}.$$

Furthermore, the sum of entries in each row of $\mathbf{W}$ is zero for all nodes except the root. The sum of entries in rows corresponding to root nodes at height k is $2^k c_{k+1}$. As a result, $\tilde{\mathbf{v}}$ has nonzero entries of value c_{k+1}^{-1} in the locations corresponding to the roots of the conductor surfaces at unit potential.

Note: The preceding description is for a balanced tree in which all leaf nodes are at height 1. An unbalanced tree can be embedded into a balanced tree by adding additional dummy nodes, and the above procedure can be followed. An efficient implementation can be developed by avoiding the actual construction of dummy nodes.

E. Solving the transformed system (Steps 3-4)

For problems in uniform medium, the linear system (1.8) before transformation is symmetric. Correspondingly, the transformed sparse linear system (3.1) is symmetric. We use the incomplete Cholesky factorization with no fill [21] to compute the preconditioner. Preconditioned Conjugate Gradients method is used to solve the system.

For problems with multiple dielectrics, the linear system (1.8) before transformation is unsymmetric. Correspondingly, the transformed sparse linear system (3.1) is unsymmetric. The preconditioner is computed from an incomplete LU factorization with no fill [21]. We use right preconditioned GMRES method to solve the system.

F. Computing capacitance (Step 5)

Capacitance can be computed from $\tilde{\mathbf{q}}$ directly without computing $\mathbf{q}$. Recall that $\tilde{\mathbf{q}} = \mathbf{W}\mathbf{q}$, and that the rows of $\mathbf{W}$ corresponding to root nodes at height k have identical nonzero entries with value c_{k+1}. Thus, a root node entry in $\tilde{\mathbf{q}}$ is c_{k+1} times the sum of all the leaf panel charges in that tree. Capacitance can be computed by adding the root node entries of each conductor in $\tilde{\mathbf{q}}$ after scaling them by corresponding factors c_{k+1}^{-1}.

G. Complexity analysis

The complexity of constructing the factorization of $\mathbf{P}$ in Step 1 is $O(n)$ [3]. The transformation of the linear system in Step 2 usually takes $O(nh)$ time, where h is the height of the tree. Normally, $h = O(\log n)$. Since the number of nonzeros in $\tilde{\mathbf{P}}$ is $O(n)$, which was explained previously, the incomplete factorization can be done in $O(n)$ time. Each iteration requires a matrix-vector product with $\tilde{\mathbf{P}}$, and a solution

of systems with the lower and upper triangular factors of the preconditioner. Thus, Steps 3 and 4 take $O(n)$ time when the number of iterations is small. Capacitance can be computed in constant time. The overall complexity of this algorithm is normally $O(n \log n + mn)$, where m is the number of conductors.

H. Experimental results

We compare PHiCap with the following algorithms: FastCap with expansion order 2, FastCap with expansion order 1, and HiCap. Other methods, such as SVD [5] and pFFT [6], exhibit performance that is similar to FastCap. No benchmark experiments were reported for the geometric independent method [22]. In [3], HiCap algorithm can only solve problems in uniform media. To make the comparison complete, we extend HiCap to the multiple dielectrics case. The algorithms are executed on a Sun UltraSPARC Enterprise 4000. Unless otherwise noted, the iterations are terminated when the relative residual norm of the preconditioned system is reduced below 10^{-2}.

The first set of benchmarks are $k \times k$ bus crossing structures from [4], shown in Fig. 11. Each bus is scaled to 1 $meter \times 1$ $meter \times (2k+1)$ $meters$. The distance between the adjacent buses in the same layer is 1 $meter$ and the distance between the two bus layers is 2 $meter$. For the uniform dielectric cases, the permittivity is assumed to be ϵ_0. For the multiple dielectric cases, see Fig. 12, the medium surrounding the upper layer conductors has permittivity $3.9\epsilon_0$ and the medium surrounding the lower layer conductors has permittivity $7.5\epsilon_0$. The shaded box represents the interface of the two layers.

Tables I and II compare the four algorithms. PHiCap is the fastest and uses much less memory compared to FastCap. The current implementation of PHiCap uses more memory compared to HiCap because of the additional storage needed for

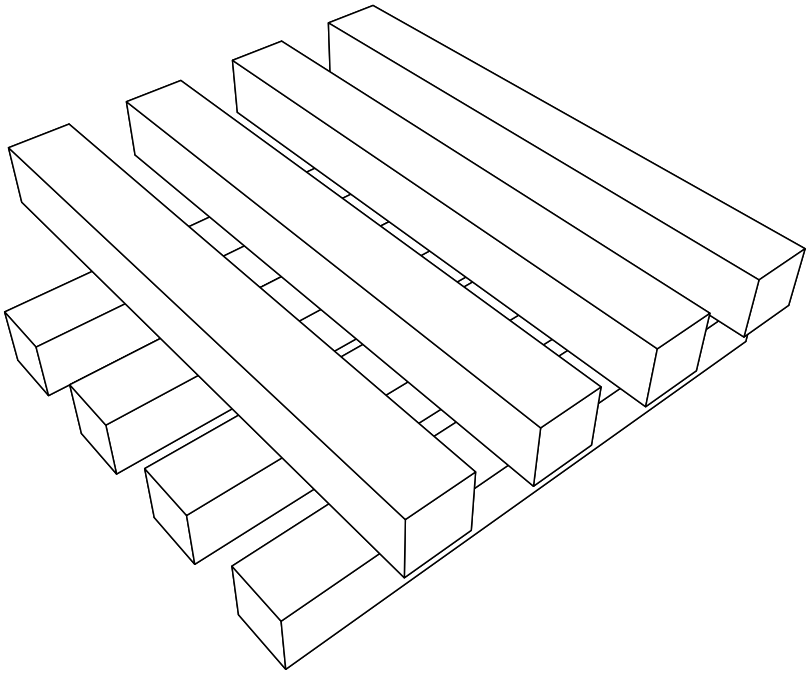


Fig. 11. 4×4 bus crossing benchmark in uniform dielectric.

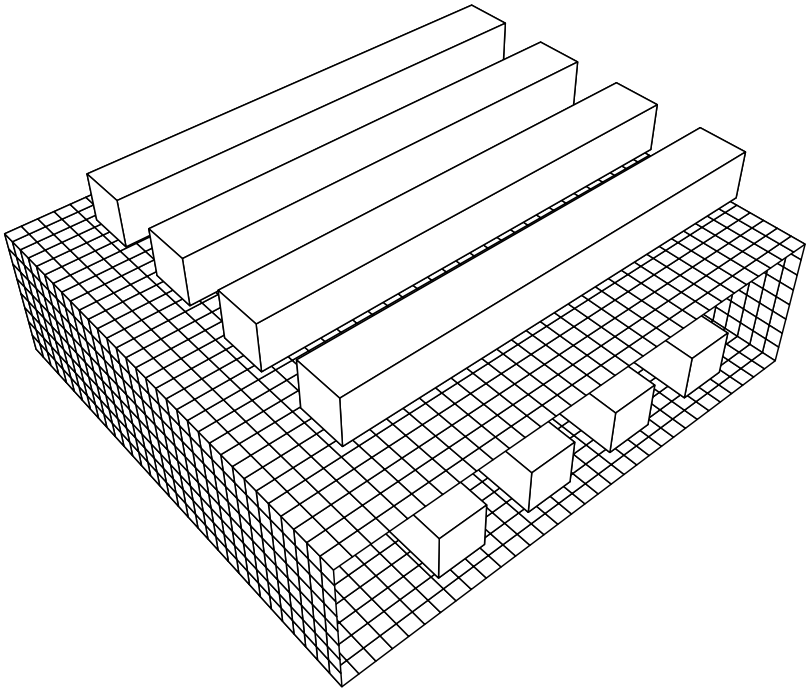


Fig. 12. 4×4 bus crossing benchmark in two layers of dielectrics (section view).

the transformed system $\tilde{\mathbf{P}}$. The storage requirement can be reduced by computing $\tilde{\mathbf{P}}$ directly, since it is not necessary to construct $\mathbf{H}$. In HiCap and PHiCap algorithms, the discretization threshold P_ϵ is chosen to be 0.008. This ensures that the relative error in the capacitance matrix computed by PHiCap is below 3%, which is acceptable in practice. The relative error in the capacitance matrix $\mathbf{C}'$, which is computed by HiCap or PHiCap algorithms, is defined as $\|\mathbf{C}-\mathbf{C}'\|_F/\|\mathbf{C}\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. As per standard practice, $\mathbf{C}$ is computed by FastCap with expansion order 2.

Table III shows the first and second rows of the capacitance matrix computed by PHiCap and FastCap. It is easy to see that for self-capacitances and significant coupling capacitances, where a coupling capacitance is considered significant if it is greater than 10% of the self capacitance, PHiCap's error is mostly within 3%, with respect to FastCap with expansion order 2. The error for the small coupling capacitances is sometimes large, which is acceptable since the small coupling capacitances have minor influence on the circuit performance. Fig. 13 shows the error distribution of the self capacitances and the significant coupling capacitances for the six benchmark examples in Table I and II.

The second set of benchmarks are complex industrial circuits containing 8 layers of dielectrics and 48, 68 and 116 conductors, respectively. The smallest case of 48 conductors is shown in Fig. 14. The results are in Table IV. FastCap cannot solve these examples because of prohibitive time and memory requirements. PHiCap displays near-optimal preconditioning in these experiments. Fig. 15 shows that the residual norm decreases rapidly for PHiCap. In contrast, the decrease is slower for HiCap. As a result, PHiCap requires much less time to solve the problem.

The third benchmark we studied is the parallel-plate problem. It is well known that this problem yields ill-conditioned systems when the two plates are very close to

Table I. Comparison of PHiCap, HiCap and FastCap for bus crossing benchmarks with uniform dielectric. Time is CPU seconds, iteration is average for solving one conductor, memory is MB, and error is with respect to FastCap (order=2).

	FastCap (order=2)	FastCap (order=1)	HiCap	PHiCap
	4×4 Bus with Uniform Dielectric			
Time	18.6	19	0.5	0.4
Iteration	8	14.9	9	3
Memory	25.7	16.7	0.7	1.0
Error	—	0.05%	2.1%	2.0%
Panel	2736	2736	1088	1088
	6×6 Bus with Uniform Dielectric			
Time	113.9	68.5	2.4	1.5
Iteration	14.4	14.5	11.9	3.2
Memory	62.5	40.3	1.9	2.9
Error	—	1.1%	2.1%	2.2%
Panel	5832	5832	3168	3168
	8×8 Bus with Uniform Dielectric			
Time	206	204	7.3	3.4
Iteration	12	21.9	13.0	3.9
Memory	112	67	3.1	4.9
Error	—	1.0%	3.1%	3.0%
Panel	10080	10080	4224	4224

Table II. Comparison of PHiCap, HiCap and FastCap for bus crossing benchmarks with multiple dielectrics. Time is CPU seconds, iteration is average for solving one conductor, memory is MB, and error is with respect to FastCap (order=2).

	FastCap (order=2)	FastCap (order=1)	HiCap	PHiCap
	4×4 Bus with Two Layer Dielectrics			
Time	63	36	1.7	2
Iteration	13	14	9	3
Memory	68	39	3.0	3.9
Error	—	0.6%	1.0%	1.0%
Panel	3456	3456	2120	2120
	6×6 Bus with Two Layer Dielectrics			
Time	162	104	10.4	5.8
Iteration	17.1	17	11.3	3
Memory	92	61	6.3	8.3
Error	—	0.5%	1.0%	1.2%
Panel	5448	5448	4120	4120
	8×8 Bus with Two Layer Dielectrics			
Time	324	197	32	15
Iteration	18	18	12.8	3
Memory	133	86.9	11.5	15.3
Error	—	0.0%	1.4%	1.4%
Panel	7968	7968	6784	6784

Table III. First two rows of capacitance matrix computed by PHiCap and FastCap (order=2) for 4×4 bus crossing benchmark with uniform dielectric.

	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}
FastCap	405.54	-137.54	-12.02	-8.07	-48.40	-40.26	-40.17	-48.48
PHiCap	405.78	-139.30	-18.89	0.10	-48.05	-39.77	-39.84	-46.60
	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}	C_{26}	C_{27}	C_{28}
FastCap	-137.54	468.23	-132.66	-11.89	-40.15	-32.59	-32.54	-40.20
PHiCap	-139.32	468.27	-129.49	-18.30	-39.96	-31.49	-31.21	-41.04

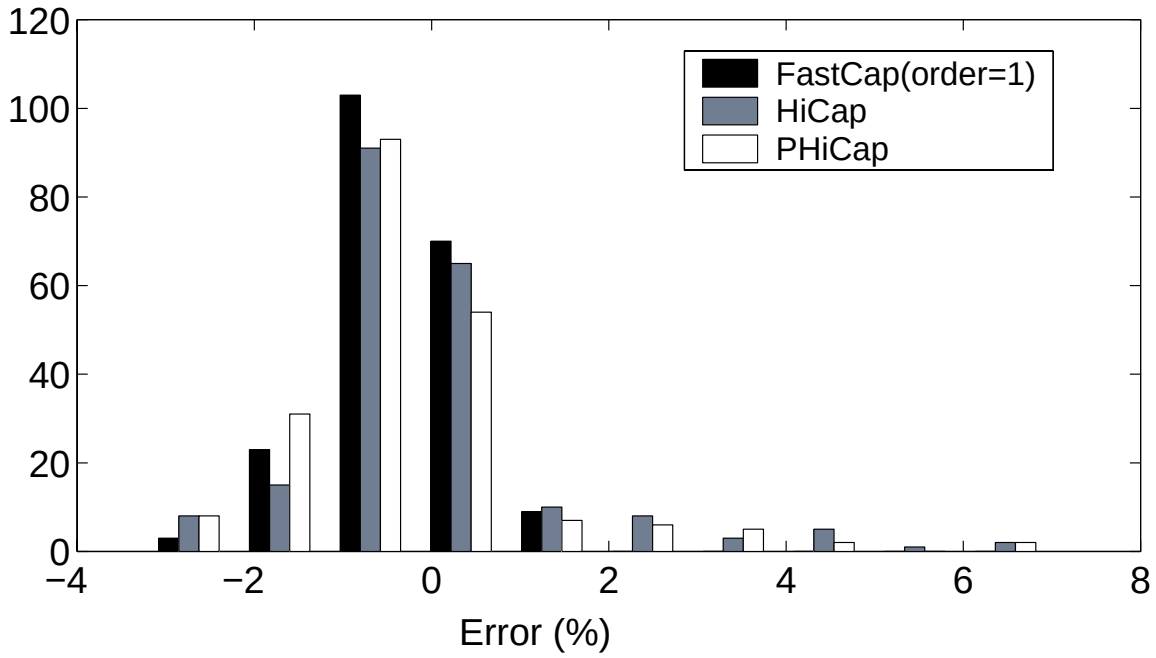


Fig. 13. Error distribution of self capacitance and significant coupling capacitance for the 6 examples in Table I and II.

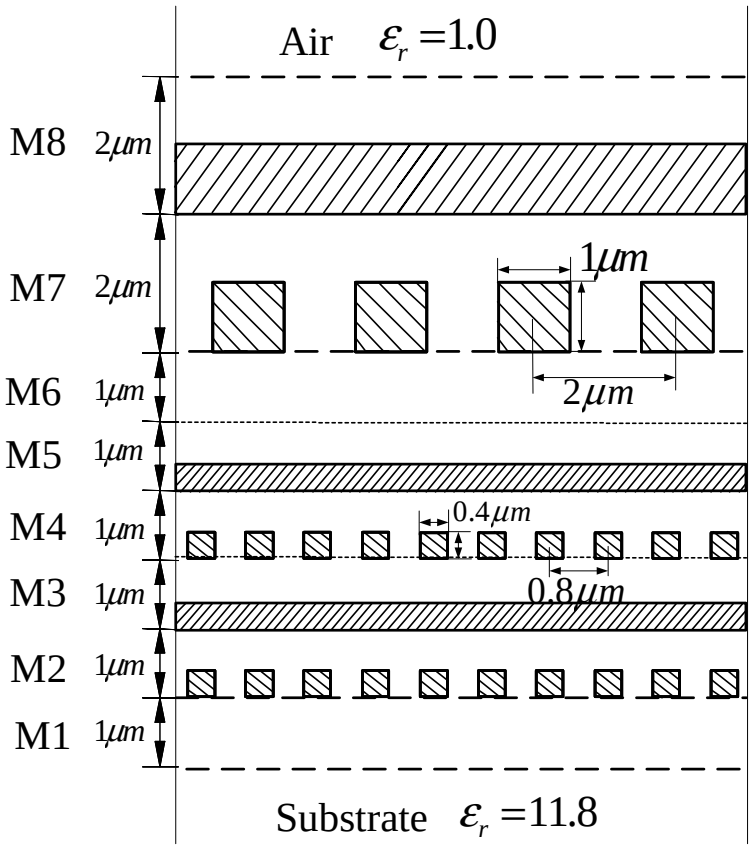


Fig. 14. Example with 48 metal conductors and 8 dielectric layers. Metal wires are shaded. Relative permittivity of M1 is 3.9, M2 through M6 is 2.5, and M7 and M8 is 7.0. Layers M2 through M5 have 10 conductors each whereas layers M7 and M8 have 4 conductors each.

Table IV. Comparison of PHiCap and HiCap for complex multiple dielectric problems shown in Fig. 14. Time is CPU seconds, iteration is average for solving one conductor, and memory is MB. FastCap was unable to solve these problems. Error is with respect to HiCap.

	48 conductors		68 conductors		116 conductors	
	HiCap	PHiCap	HiCap	PHiCap	HiCap	PHiCap
Time	533	122	3011	389	12930	2391
Iteration	18.7	2.8	25.3	3.0	36.8	5.1
Memory	43	59	115	161	406	570
Panel	19840	19840	42912	42912	138552	138552

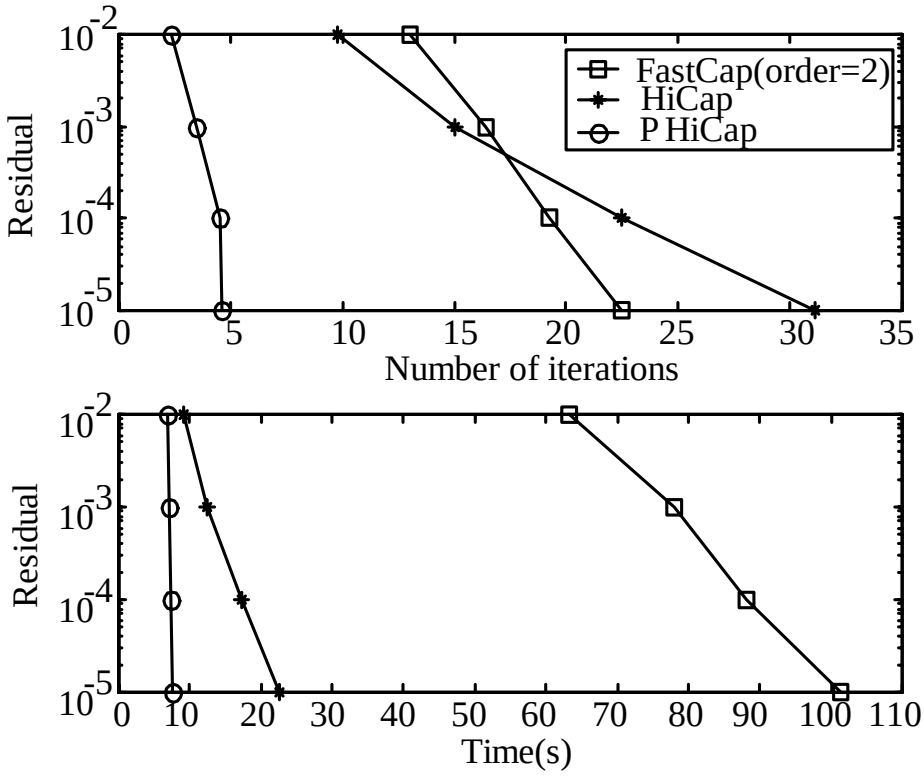


Fig. 15. Comparison of the rate of convergence of PHiCap.

each other. We consider the problem with plate size 10m × 10m and distance between two plates 0.1m. The results in Table V indicate that PHiCap performs very well on these problems too.

Table V. Comparison of PHiCap and HiCap for parallel plates of size 10m × 10m and distance 0.1m. Time is CPU seconds and iteration is average for solving one conductor.

	Coarse Discretization $P_\epsilon = 0.01$		Fine Discretization $P_\epsilon = 0.005$	
	Time	Iteration	Time	Iteration
HiCap	100.2	54	523.5	72.5
PHiCap	53.3	6	265.8	6

The multi-scale method [23] uses a similar idea to sparsify the dense matrix **P**. However, there are important differences between the multi-scale method and our method. The multi-scale method is based on high-order FMM, whereas our method is based on HiCap. It was shown that the hierarchical approach in HiCap is more efficient and kernel independent [3]. The multi-scale method uses a block diagonal preconditioner, while ours uses incomplete Cholesky or LU factorizations. In addition, the multi-scale method has been applied to uniform dielectric only. For the $k \times k$ bus crossing benchmarks, we compare the number of iterations needed by the two methods to reduce the residual norm below 10^{-9}. Table VI shows that the number of iterations required by PHiCap is less than the multi-scale method. The growth in iterations with the increase of the problem size is negligible for PHiCap. A more detailed comparison was not possible because only the number of iterations were reported in [23].

Table VI. Comparison of iteration numbers for multi-scale method and new algorithm. Experiments are for $k \times k$ bus crossing conductors. Convergence tolerance is 10^{-9}.

	1×1	2×2	4×4	6×6	8×8
Multiscale	12	17	18	18	18
New	7	14	13	14	16

I. Summary

This chapter introduces the PHiCap algorithm, which is a preconditioned hierarchical algorithm for capacitance extraction. PHiCap transforms the dense linear system into a sparse system and then solves it by a preconditioned iterative method. The sparse structure allows construction of inexpensive but highly effective preconditioners based on incomplete factorization techniques. The dense-to-sparse transformation used in PHiCap is applicable to multipole-based methods as well, where the linear system can be represented by a block matrix. Numerical experiments demonstrate the superiority of PHiCap over FastCap and HiCap in terms of the number of iterations of the solver and the overall running time.

Experiments on the $k \times k$ bus crossing benchmark show that PHiCap is up to 70 times faster than FastCap (order=2), is up to 60 times faster than FastCap (order=1), and is up to 2 times faster than HiCap. For complex industrial problems with multiple dielectrics, PHiCap is 4-8 times faster than HiCap. With the same computing resources, FastCap is not able to solve these problems due to the expensive memory and running time cost. The number of iterations used by PHiCap is also less than the multi-scale method [23].

CHAPTER IV

REDCAP ALGORITHM

A. Introduction

In the previous chapter, we proposed the PHiCap algorithm that transforms the dense linear system to the sparse linear system in wavelet basis. The incomplete LU and incomplete Cholesky factorization are very effective preconditioners for the sparse linear system. Experiments show that with preconditioning, the iterative solver converges within 5 iterations even for very large problems.

Inspired by this observation, we propose an iteration free approach in which we approximate the sparse linear system by its incomplete factorization and solve a reduced linear system. The **Red**uced linear system is solved for **Cap**acitance using forward/backward substitution directly without resorting to an iterative method. We name the algorithm RedCap. RedCap avoids solving the original large system.

The time complexity of RedCap depends on the sparse transformation, which takes $O(n \log n)$ time if we use the transformation in PHiCap. If we use the ICCAP algorithm [24], whose complexity is claimed to be $O(n)$, to generate the sparse linear system, the complexity is $O(n)$. For the sake of clarity, we explain the RedCap algorithm based on PHiCap that is introduced in the previous chapter.

RedCap is not the first attempt to solve the linear system of BEM without iterations. In [14] and [15], sparse representations of the linear systems are constructed using low rank approximation of the off diagonal blocks. These sparse representations are solved by LU factorization. The time complexity of those algorithms is $O(n^\gamma)$, where γ ranges from 2.20 to 2.89, which is much slower than RedCap. To the best of our knowledge, RedCap is the most efficient iteration-free method that solves a

reduced linear system for capacitance extraction.

We first outline the RedCap algorithm in section B. We explain the details of the RedCap algorithm in sections C and D, and present experimental results in section E. We summarize the algorithm in section F.

B. Algorithm outline

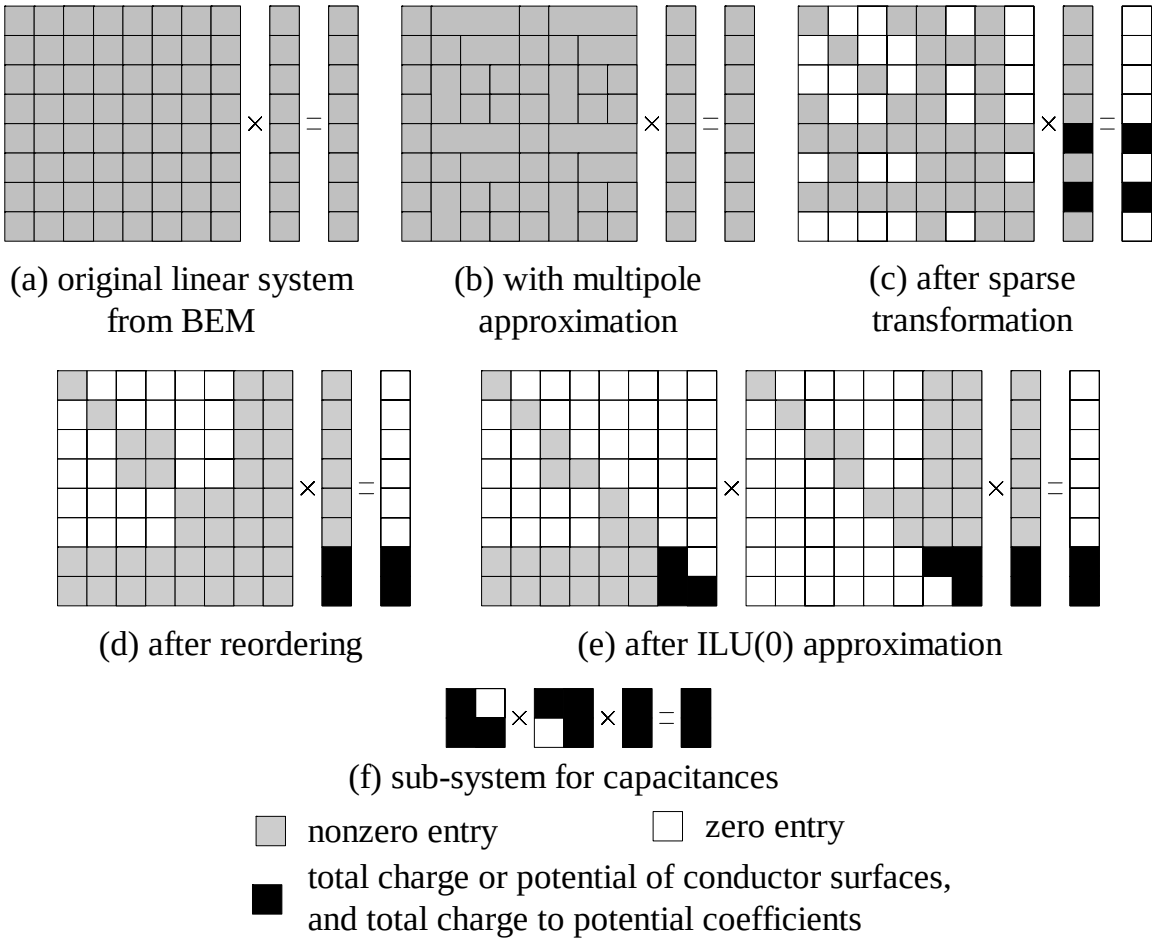


Fig. 16. The flow of the RedCap algorithm.

The outline of the RedCap algorithm is shown in Fig. 16. Fig. 16(a) is the dense $n \times n$ linear system from BEM, which is never constructed explicitly. Fig. 16(b) is the linear system approximated by hierarchical refinement approximation (HiCap

algorithm in chapter II). This linear system is still dense, but is represented by $O(n)$ block entries. In Fig. 16(c), the linear system is transformed into a sparse system (the PHiCap algorithm in chapter III). The unknown charge vector in the transformed system includes entries for the total charge on each conductor surface. These entries are sufficient to derive the capacitance. The entries of the right-hand side potential vector in the transformed system are zero except for those corresponding to the conductor surfaces. So far, the algorithm is the same as PHiCap, which is explained in detail in chapter III.

In Fig. 16(d), the matrix is permuted symmetrically to order the dense rows and columns at the end. Fewer fill-ins will be discarded in the following approximation step. The total surface charge entries of the charge vector and the nonzero entries of the potential vector are ordered at the end as well. We perform incomplete LU factorization with no fill-in (ILU(0)) to approximate the sparse linear system (see Fig. 16(e)). Since only the bottom part of the charge vector is needed to compute the capacitances, we identify the corresponding small sub-system in Fig. 16(f), and solve the sub-system for the total charges directly using Gaussian elimination.

In the algorithm, approximations are introduced only in the multipole approximation step and the ILU(0) step. From [3], we know that the multipole approximation is very accurate. We will show later that the ILU(0) step does not introduce significant error either since the reordering reduces the number of nonzeros that are discarded.

The first two steps (Fig. 16(b),(c)) of RedCap were discussed in previous chapters II and III. In the following sections, we will describe the rest steps, including the ordering (Fig. 16(d)), ILU (Fig. 16(e)) and solving the sub-system (Fig. 16(f)).

C. Ordering and ILU(0) approximation

Unlike exact LU factorization, the incomplete LU factorization maintains sparsity of the factors by dropping nonzero entries, called fill-in, in positions where the matrix has zero entries. Formally, incomplete LU factorization with no fill-in, written as ILU(0), replaces (3.1) with

$$\hat{\mathbf{L}}\hat{\mathbf{U}}\hat{\mathbf{q}} = \tilde{\mathbf{v}}, \tag{4.1}$$

where $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the incomplete lower and upper triangular factors of $\tilde{\mathbf{P}}$, respectively, and $\hat{\mathbf{q}}$ is an approximation of $\tilde{\mathbf{q}}$. Compared to the original linear system (3.1), error is introduced in the new system (4.1).

By ordering sparse rows before dense rows, the number of dropped fill-ins can be reduced. According to the discussion in chapter III, the links at lower level of the hierarchy are propagated and merged upward to the higher level panels. As a result, basis functions at higher levels in the tree have more coefficients. Therefore, the rows and columns of $\tilde{\mathbf{P}}$ corresponding to higher levels tend to be denser. We order the basis functions according to their levels in the tree. Low level basis functions are ordered before high level basis functions. As a result, the total charges and the potentials of the conductor surfaces are ordered at the bottom of $\tilde{\mathbf{q}}$ and $\tilde{\mathbf{v}}$, respectively.

Table VII compares the effect of this ordering. Using the proposed ordering, the exact LU factorization has fewer fill-ins, and therefore fewer fill-ins are dropped in ILU(0) approximation. Fig. 9(b) shows the reordered $\tilde{\mathbf{P}}$ matrix of the example in Fig. 5. Fig. 17 shows the sparsity pattern of $\tilde{\mathbf{P}}$ for the same 4×4 bus crossing example used in Table VII. We used a coarser discretization to illustrate the sparsity clearly.

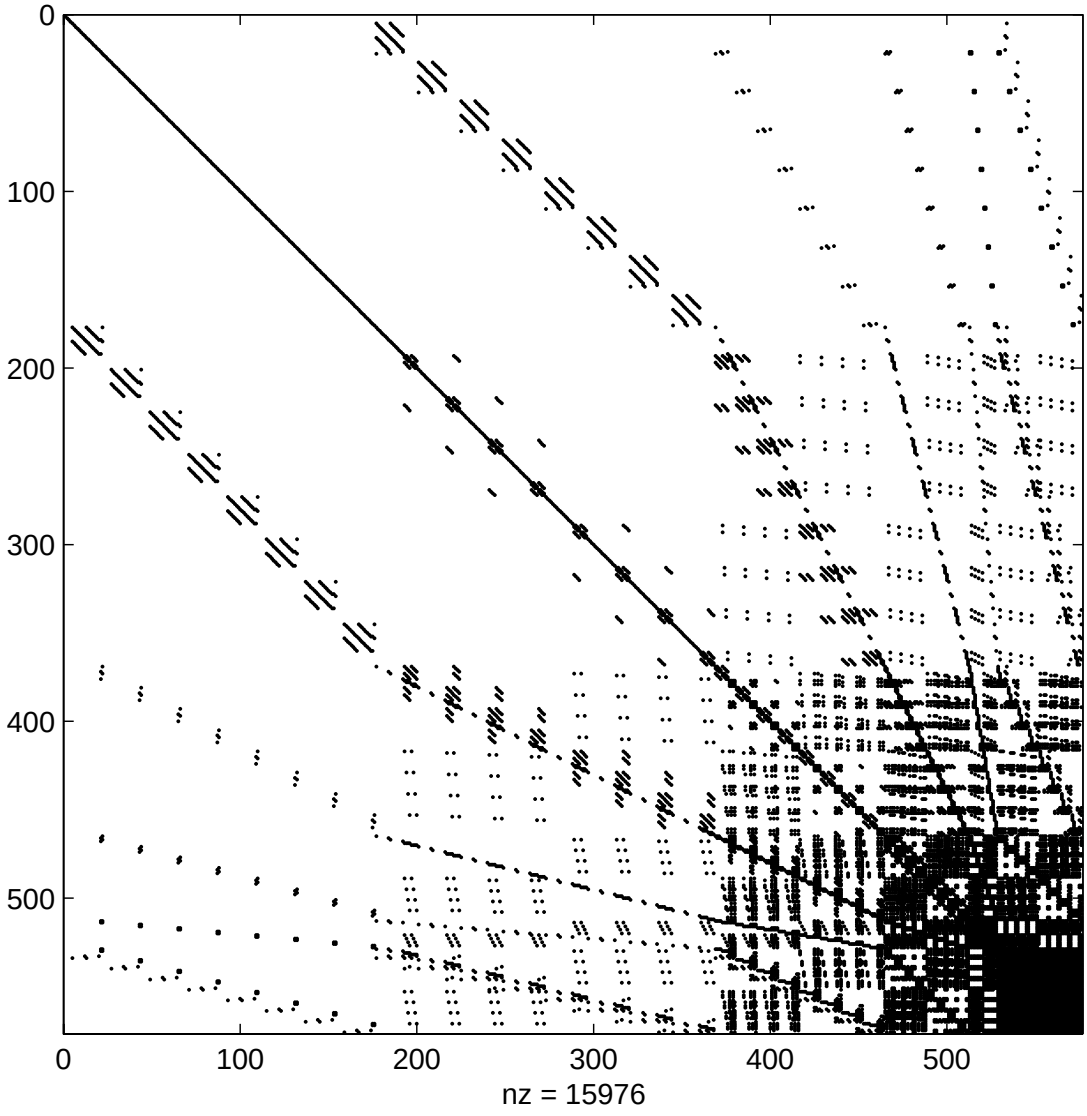


Fig. 17. Sparsity pattern of the transformed linear system after reordering the dense rows and columns at the end.

Table VII. Comparison of different orderings of $\tilde{\mathbf{P}}$ for 4×4 bus crossing benchmark in uniform dielectric. Since the linear system is symmetric, only the lower triangular part is reported. $\|\cdot\|$ is the Frobenius norm.

	Nonzeros in L	Nonzeros dropped in $\hat{\mathbf{L}}$	$\frac{\|\mathbf{L}-\hat{\mathbf{L}}\|}{\|\mathbf{L}\|}$
Arbitrary ordering	1592928	94.4%	12.0%
Proposed ordering	414938	78.6%	2.9%

D. Equivalent reduced system

Based on the ordering, we split $\hat{\mathbf{q}}$ and $\tilde{\mathbf{v}}$ into two parts:

$$\hat{\mathbf{q}} = \begin{bmatrix} \hat{\mathbf{q}}_\mathbf{1} \\ \hat{\mathbf{q}}_\mathbf{2} \end{bmatrix} \quad \text{and} \quad \tilde{\mathbf{v}} = \begin{bmatrix} \tilde{\mathbf{v}}_\mathbf{1} \\ \tilde{\mathbf{v}}_\mathbf{2} \end{bmatrix},$$

where $\hat{\mathbf{q}}_\mathbf{1}$ and $\tilde{\mathbf{v}}_\mathbf{1}$ correspond to the charge differences and potential differences between sibling panels, and $\hat{\mathbf{q}}_\mathbf{2}$ and $\tilde{\mathbf{v}}_\mathbf{2}$ correspond to the total charges and potentials on conductor surfaces. In the basis hierarchy, $\hat{\mathbf{q}}_\mathbf{2}$ and $\tilde{\mathbf{v}}_\mathbf{2}$ are basis functions for the roots, and $\hat{\mathbf{q}}_\mathbf{1}$ and $\tilde{\mathbf{v}}_\mathbf{1}$ are basis functions for non-root nodes. It is easy to see that $\tilde{\mathbf{v}}_\mathbf{1} = \mathbf{0}$. For the capacitance extraction purpose, we only need to compute $\hat{\mathbf{q}}_\mathbf{2}$.

We split $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ to conform to $\hat{\mathbf{q}}$ and $\tilde{\mathbf{v}}$:

$$\hat{\mathbf{L}} = \begin{bmatrix} \hat{\mathbf{L}}_{\mathbf{11}} & \mathbf{0} \\ \hat{\mathbf{L}}_{\mathbf{21}} & \hat{\mathbf{L}}_{\mathbf{22}} \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{U}} = \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{11}} & \hat{\mathbf{U}}_{\mathbf{12}} \\ \mathbf{0} & \hat{\mathbf{U}}_{\mathbf{22}} \end{bmatrix}.$$

From equation (4.1), we have

$$\hat{\mathbf{L}}_{\mathbf{11}} \left(\hat{\mathbf{U}}_{\mathbf{11}} \hat{\mathbf{q}}_{\mathbf{1}} + \hat{\mathbf{U}}_{\mathbf{12}} \hat{\mathbf{q}}_{\mathbf{2}} \right) = \mathbf{0}, \tag{4.2}$$

$$\hat{\mathbf{L}}_{\mathbf{21}} \left(\hat{\mathbf{U}}_{\mathbf{11}} \hat{\mathbf{q}}_{\mathbf{1}} + \hat{\mathbf{U}}_{\mathbf{12}} \hat{\mathbf{q}}_{\mathbf{2}} \right) + \hat{\mathbf{L}}_{\mathbf{22}} \hat{\mathbf{U}}_{\mathbf{22}} \hat{\mathbf{q}}_{\mathbf{2}} = \tilde{\mathbf{v}}_{\mathbf{2}}. \tag{4.3}$$

Since $\hat{\mathbf{L}}_{\mathbf{11}}$ is nonsingular, equation (4.2) implies

$$\hat{\mathbf{U}}_{\mathbf{11}} \hat{\mathbf{q}}_{\mathbf{1}} + \hat{\mathbf{U}}_{\mathbf{12}} \hat{\mathbf{q}}_{\mathbf{2}} = \mathbf{0}. \tag{4.4}$$

Combining equation (4.3) and (4.4), we obtain the following system for $\hat{\mathbf{q}}_{\mathbf{2}}$:

$$\hat{\mathbf{L}}_{\mathbf{22}} \hat{\mathbf{U}}_{\mathbf{22}} \hat{\mathbf{q}}_{\mathbf{2}} = \tilde{\mathbf{v}}_{\mathbf{2}}. \tag{4.5}$$

The dimension of sub-system (4.5) equals the number of conductor surfaces, which is $O(m)$. It is a small system compared to the original system (4.1) of dimension n. Furthermore, it can be solved in $O(m^2)$ time using its triangular factors $\hat{\mathbf{L}}_{\mathbf{22}}$ and $\hat{\mathbf{U}}_{\mathbf{22}}$. The vector $\hat{\mathbf{q}}_{\mathbf{2}}$ is sufficient to calculate the capacitances among conductors.

E. Experimental results

We compare RedCap with FastCap [4] and PHiCap [16]. Table VIII and Table IX report the experimental results on bus crossing examples. The uniform dielectric cases are standard benchmarks from [4] and the multi-layer dielectrics cases are from [16]. Fig. 11 and Fig. 12 are the 4×4 bus crossing examples in uniform and multi-layer dielectrics, respectively. The experiments were conducted on a Sun UltraSPARC Enterprise 4000. Time is cpu seconds needed to compute the entire capacitance matrix. Iteration is average for solving one conductor. Memory is MB. The relative error in the capacitance matrix $\mathbf{C}'$ is defined as $\|\mathbf{C}-\mathbf{C}'\|_F / \|\mathbf{C}\|_F$, where $\|\cdot\|_F$ denotes the Frobenius norm. The RedCap algorithm is much faster and more memory efficient

than FastCap. It uses less memory compared to PHiCap because the algorithm does not need to store iteration vectors. Additional saving is obtained by overwriting $\tilde{\mathbf{P}}$ with $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$.

Table VIII. Comparison of RedCap, PHiCap and FastCap for bus crossing benchmarks with uniform dielectric. Time is CPU seconds, iteration is average for solving one conductor, memory is MB, and error is with respect to FastCap (order=2).

		Time	Iter.	Mem.	Error	Dim.
4×4	FastCap	18.6	8.0	25.7	—	2736
	PHiCap	0.4	3.0	2.4	2.1%	1088
	RedCap	0.3	0	2.1	1.1%	48
6×6	FastCap	113.9	14.4	62.5	—	5832
	PHiCap	1.5	3.2	7.3	2.3%	3168
	RedCap	1.1	0	6.4	1.7%	72
8×8	FastCap	206.7	12.0	111.9	—	10080
	PHiCap	3.3	3.4	12.8	3.0%	4224
	RedCap	2.8	0	11.4	1.8%	96

Table X compares the RedCap algorithm with PHiCap for complex multiple dielectric problems shown in Fig. 14. FastCap was unable to solve these problems, because the running time cost and memory cost exceed the computation resource available to us. RedCap and PHiCap use similar set-up time. However, RedCap computes the solution in less than 1% of the time spent by PHiCap in the iterative solver.

Table X gives the error of the RedCap algorithm with respect to PHiCap. The overall error of the capacitance matrix is less than 2%. Fig. 18 shows the error dis-

Table IX. Comparison of RedCap, PHiCap and FastCap for bus crossing benchmarks with multiple dielectrics. Time is CPU seconds, iteration is average for solving one conductor, memory is MB, and error is with respect to FastCap (order=2).

		Time	Iter.	Mem.	Error	Dim.
4×4	FastCap	63.2	13.0	67.5	—	3456
	PHiCap	2.0	2.0	6.4	0.7%	2120
	RedCap.	1.5	0	5.6	0.7%	48
6×6	FastCap	162.1	17.1	91.5	—	5448
	PHiCap	5.7	3.0	14.0	1.3%	4120
	RedCap	3.4	0	12.4	1.4%	72
8×8	FastCap	329.1	18.0	133.3	—	7968
	PHiCap	14.2	3.0	26.3	1.4%	6784
	RedCap	6.9	0	23.5	1.5%	96

Table X. Comparison of PHiCap and HiCap for complex multiple dielectric problems shown in Fig. 14. Time is CPU seconds, iteration is average for solving one conductor, and memory is MB. FastCap was unable to solve these problems. Error is with respect to PHiCap.

	48 conductors		68 conductors	
	PHiCap	RedCap	PHiCap	RedCap
Dimension	19840	191	42912	655
Set-up time	34.5	34.0	150.7	153.0
Solving time	88.6	0.1	355.8	2.3
Total time	123.1	34.1	506.5	155.4
Iteration	2.8	0	3.0	0
Memory	98	64	310	210
Error	—	1.7%	—	1.5%

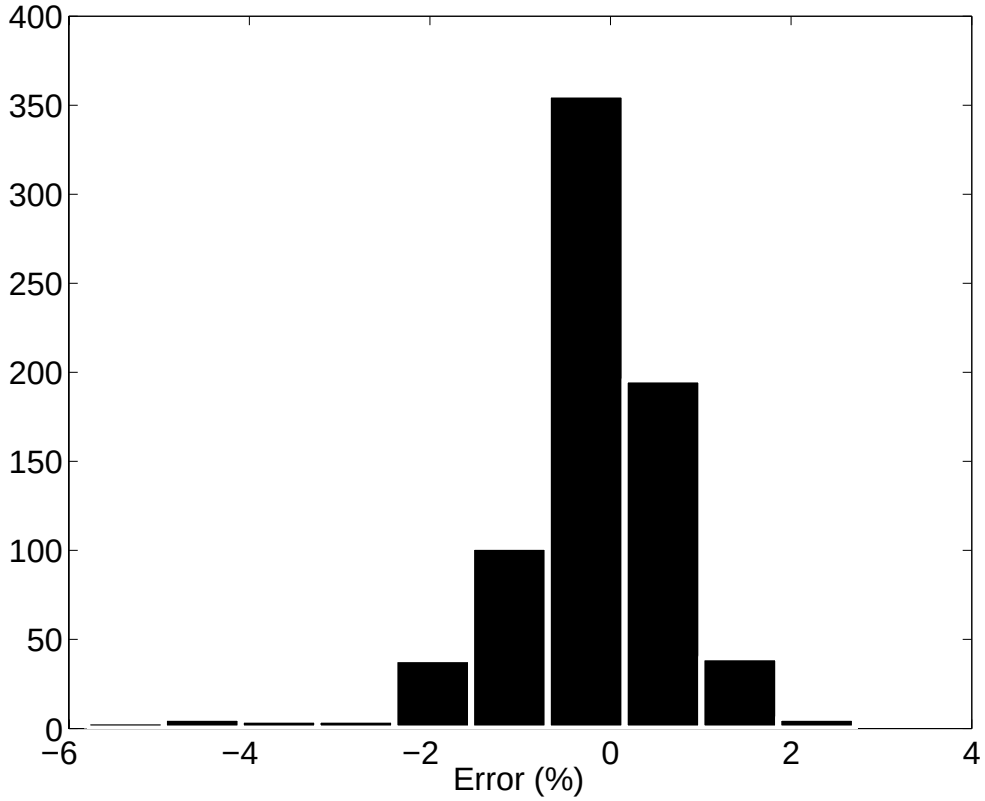


Fig. 18. Error distribution of RedCap with respect to PHiCap. Self-capacitances and significant coupling capacitances of all the experiments in Table VIII, IX and X are included. Coupling capacitances greater than 10% of self capacitances are considered significant.

tribution of individual capacitances. These experiments confirm that the incomplete LU factorization with the proposed ordering is indeed a good approximation of the original system. We are unable to compare the charge distribution error since the RedCap algorithm only computes the total charge on each conductor surface.

F. Summary

In this chapter, we presented the RedCap algorithm that solves a reduced linear system, instead of the large linear system from BEM, to derive the capacitances. The size of the reduced linear system is equal to the number of conductor surfaces, which is much smaller than the number of panels in the discretization of BEM. Approximations to the triangular factors of the reduced system are obtained directly via incomplete factorization without computing the system explicitly. The reduced system is solved directly using the approximate factors. To the best of our knowledge, the RedCap algorithm is the most efficient iteration-free method for capacitance extraction.

Numerical experiments on conductors embedded in uniform and multi-layer dielectric show that the RedCap algorithm is up to 100 times faster than FastCap [4] and is up to 4 times faster than PHiCap [16]. The RedCap algorithm uses two approximations: fast multipole approximation of the linear system and inexact factorization of the sparse system. Despite the two approximations involved, RedCap preserves very good accuracy. The error with respect to FastCap is within 2% for all the experiments we have done.

CHAPTER V

EXTRACTION WITH FLOATING DUMMY-FILLS

A. Introduction

To reduce the pattern-dependent process variations of the dielectric and metal thickness, floating dummy metals are often inserted [25]. Unlike regular conductors whose potential is known while charge distribution is unknown in the extraction process, the potential of a floating dummy conductor is unknown and its global surface charge is zero. In circuit simulation and integrity analysis, we need the equivalent capacitances among the regular conductors only.

Conventional capacitance extraction tools treat floating dummy conductors as regular ones, and compute the capacitances among both regular and dummy conductors. The capacitance matrix satisfies

$$\begin{bmatrix} \mathbf{C_{rr}} & \mathbf{C_{rf}} \\ \mathbf{C_{fr}} & \mathbf{C_{ff}} \end{bmatrix} \begin{bmatrix} \mathbf{V_r} \\ \mathbf{V_f} \end{bmatrix} = \begin{bmatrix} \mathbf{Q_r} \\ \mathbf{Q_f} \end{bmatrix},$$

where the subscript r represents regular conductors, and f represents floating dummy conductors. Since the potential of a floating conductor is an unknown constant, and its global surface charge is equal to zero, we have

$$\mathbf{Q_f} = 0.$$

Thus, the equivalent capacitance matrix is

$$\mathbf{C_{eq}} = \mathbf{C_{rr}} - \mathbf{C_{rf}}\mathbf{C_{ff}}^{-1}\mathbf{C_{fr}}.$$

Let m_r and m_f represent the number of regular and floating dummy conductors, respectively. Clearly, $m_r + m_f = m$. We have $\mathbf{C_{rr}} \in \mathbb{R}^{\mathbf{m_r} \times \mathbf{m_r}}$, $\mathbf{C_{rf}} \in \mathbb{R}^{\mathbf{m_r} \times \mathbf{m_f}}$,

$\mathbf{C_{fr}} \in \mathbb{R}^{\mathbf{m_f} \times \mathbf{m_r}}$, $\mathbf{C_{rr}} \in \mathbb{R}^{\mathbf{m_f} \times \mathbf{m_f}}$ and $\mathbf{C_{eq}} \in \mathbb{R}^{\mathbf{m_r} \times \mathbf{m_r}}$. This method needs $m_r + m_f$ linear solves, which results in very high computation cost when there are many dummy fill-ins. For the methods in ordinary basis, such as HiCap, the linear system is (1.8). For methods, such as PHiCap, that use the wavelet basis, the linear system is (3.1). For RedCap, the linear system is (4.5). Some methods [26] based on FDM/FEM were proposed, but none is known for the BEM method.

In this chapter, we propose a method to expand the linear system for extraction by embedding the floating dummy potential and global surface charge requirement into extraction. As a result, the equivalent capacitances among regular conductors can be solved through m_r, instead of $m_r + m_f$, system solves. This method is applicable to both the algorithm in ordinary basis, such as FastCap [4] and HiCap [3], and the algorithm in wavelet basis, such as PHiCap [16] and RedCap [17]. However, our studies show that the methods based on wavelet basis are much more efficient than the methods based on ordinary basis, and the method based on RedCap is the most efficient. The method based on standard basis are unattractive due to slow convergence.

B. Expanded extraction methods

With the presence of floating dummy-fills, the extraction requirement is different. For a floating dummy conductor, the global surface charge is zero and the potential is unknown. In this section, we embed this requirement for dummy-fills to existing capacitance extraction methods, including HiCap, PHiCap and RedCap.

1. Expanded HiCap

We rewrite the linear system (1.8) from HiCap by distinguishing the panels of regular conductors and those of floating dummy conductors.

$$\begin{bmatrix} \mathbf{P_{rr}} & \mathbf{P_{rf}} \\ \mathbf{P_{fr}} & \mathbf{P_{ff}} \end{bmatrix} \begin{bmatrix} \mathbf{q_r} \\ \mathbf{q_f} \end{bmatrix} = \begin{bmatrix} \mathbf{v_r} \\ \mathbf{v_f} \end{bmatrix}, \tag{5.1}$$

where the subscript r represents regular conductors, and f represents floating dummy conductors. Let n_r and n_f be the number of panels on regular conductors and floating dummy conductors, respectively. Vectors $\mathbf{q_f} \in \mathbb{R}^{\mathbf{n_f}}$ and $\mathbf{v_f} \in \mathbb{R}^{\mathbf{n_f}}$ are charge and potential on panels of floating dummy conductors, respectively.

Since the global surface charge of each floating dummy conductor is zero, we have

$$\begin{aligned} \mathbf{B^T q_f} &= \mathbf{0} \\ \mathbf{B V_f} &= \mathbf{v_f} \end{aligned}, \tag{5.2}$$

where $\mathbf{B} \in \mathbb{R}^{\mathbf{n_f} \times \mathbf{m_f}}$ is the incidence matrix denoting the relationship between floating dummy conductors and their panels. Each column of $\mathbf{B}$ corresponds to one floating dummy conductor, and each row of $\mathbf{B}$ corresponds to one of the panels of floating dummy conductors. Any entry B_{ij} of $\mathbf{B}$ is one if the i-th panel belongs to the j-th floating dummy conductor, otherwise B_{ij} is zero.

Embed (5.2) into (5.1), we have the following expanded linear system.

$$\begin{bmatrix} \mathbf{P_{rr}} & \mathbf{P_{rf}} & \mathbf{0} \\ \mathbf{P_{fr}} & \mathbf{P_{ff}} & \mathbf{-B} \\ \mathbf{0} & \mathbf{-B^T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q_r} \\ \mathbf{q_f} \\ \mathbf{V_f} \end{bmatrix} = \begin{bmatrix} \mathbf{v_r} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{5.3}$$

With m_f system solves of (5.3), we get the equivalent capacitance matrix $\mathbf{C_{eq}}$. However, since the expanded linear system is no longer symmetric positive definite, it is

difficult to construct efficient preconditioners, and the convergence of the system is very slow.

It is worth noting that the discussion in this section is applicable not only to HiCap, but also to other extraction methods, such as FastCap, which use the ordinary basis.

2. Expanded PHiCap

The linear system from PHiCap (3.1) is in wavelet basis. We use subscript r and f represent the regular conductors and the floating dummy conductors, respectively. We use subscript 1 to represent the wavelet basis for potential and charge difference of sibling panels, and 2 to represent the wavelet basis for potential and total charge of root panels. Thus, we rewrite (3.1) as

$$\begin{bmatrix} \tilde{\mathbf{P}}_{\mathbf{1},\mathbf{1}} & \tilde{\mathbf{P}}_{\mathbf{1},\mathbf{2r}} & \tilde{\mathbf{P}}_{\mathbf{1},\mathbf{2f}} \\ \tilde{\mathbf{P}}_{\mathbf{2r},\mathbf{1}} & \tilde{\mathbf{P}}_{\mathbf{2r},\mathbf{2r}} & \tilde{\mathbf{P}}_{\mathbf{2r},\mathbf{2f}} \\ \tilde{\mathbf{P}}_{\mathbf{2f},\mathbf{1}} & \tilde{\mathbf{P}}_{\mathbf{2f},\mathbf{2r}} & \tilde{\mathbf{P}}_{\mathbf{2f},\mathbf{2f}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_{\mathbf{1}} \\ \tilde{\mathbf{q}}_{\mathbf{2r}} \\ \tilde{\mathbf{q}}_{\mathbf{2f}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{v}}_{\mathbf{2r}} \\ \tilde{\mathbf{v}}_{\mathbf{2f}} \end{bmatrix}. \tag{5.4}$$

Let M_r and M_f be the number of surfaces of the regular conductors and floating dummy conductors, respectively. Let $M = M_r + M_f$, which is the number of surfaces of all conductors. Vectors $\tilde{\mathbf{q}}_{\mathbf{2r}} \in \mathbb{R}^{\mathbf{M_r}}$ and $\tilde{\mathbf{v}}_{\mathbf{2r}} \in \mathbb{R}^{\mathbf{M_r}}$ are the total surface charges and the surface potential of regular conductors, respectively.

Let $\tilde{\mathbf{B}} \in \mathbb{R}^{\mathbf{M_r} \times \mathbf{m_r}}$ denotes the relationship between dummy conductors and their surfaces (wavelet basis of roots). Each column of $\tilde{\mathbf{B}}$ corresponds to one floating dummy conductor, and each row of $\tilde{\mathbf{B}}$ corresponds to one of the surfaces of floating dummy conductors. Any entry $\tilde{B}_{ij}$ is one if the i-th surface belongs to the j-th floating dummy conductor, otherwise $\tilde{B}_{ij}$ is zero. Since the global surface charge of

each floating dummy conductor is zero, we have

$$\begin{aligned} \tilde{\mathbf{B}}^{\mathbf{T}}\tilde{\mathbf{q}}_{\mathbf{2f}} &= \mathbf{0} \\ \tilde{\mathbf{B}}\mathbf{V}_{\mathbf{f}} &= \tilde{\mathbf{v}}_{\mathbf{2f}} \end{aligned} . \tag{5.5}$$

Embed (5.5) to (5.4), we have

$$\begin{bmatrix} \tilde{\mathbf{P}}_{\mathbf{1,1}} & \tilde{\mathbf{P}}_{\mathbf{1,2r}} & \tilde{\mathbf{P}}_{\mathbf{1,2d}} & \mathbf{0} \\ \tilde{\mathbf{P}}_{\mathbf{2r,1}} & \tilde{\mathbf{P}}_{\mathbf{2r,2r}} & \tilde{\mathbf{P}}_{\mathbf{2r,2f}} & \mathbf{0} \\ \tilde{\mathbf{P}}_{\mathbf{2f,1}} & \tilde{\mathbf{P}}_{\mathbf{2f,2r}} & \tilde{\mathbf{P}}_{\mathbf{2f,2f}} & -\tilde{\mathbf{B}} \\ \mathbf{0} & \mathbf{0} & -\tilde{\mathbf{B}}^{\mathbf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_{\mathbf{1}} \\ \tilde{\mathbf{q}}_{\mathbf{2r}} \\ \tilde{\mathbf{q}}_{\mathbf{2f}} \\ \mathbf{V}_{\mathbf{f}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{v}}_{\mathbf{2r}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{5.6}$$

We construct the approximate factorization of (5.6)

$$\begin{bmatrix} \hat{\mathbf{L}} & \\ \mathbf{G}_{\mathbf{1}} & \mathbf{L}_{\mathbf{exp}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}} & \mathbf{G}_{\mathbf{2}} \\ & \mathbf{U}_{\mathbf{exp}} \end{bmatrix}, \tag{5.7}$$

where

$$\mathbf{G}_{\mathbf{1}}^{\mathbf{T}} = \hat{\mathbf{U}}^{-\mathbf{T}} \begin{bmatrix} \mathbf{0} \\ 0 \\ -\tilde{\mathbf{B}} \end{bmatrix}, \mathbf{G}_{\mathbf{2}} = \hat{\mathbf{L}}^{-\mathbf{1}} \begin{bmatrix} \mathbf{0} \\ 0 \\ -\tilde{\mathbf{B}} \end{bmatrix}.$$

Matrix $\hat{\mathbf{L}}$ and $\hat{\mathbf{U}}$ are the incomplete LU factors of $\tilde{\mathbf{P}}$ in (4.1), which we rewrite as

$$\begin{bmatrix} \hat{\mathbf{L}}_{\mathbf{1,1}} & & \\ \hat{\mathbf{L}}_{\mathbf{2r,1}} & \hat{\mathbf{L}}_{\mathbf{2r,2r}} & \\ \hat{\mathbf{L}}_{\mathbf{2f,1}} & \hat{\mathbf{L}}_{\mathbf{2f,2r}} & \hat{\mathbf{L}}_{\mathbf{2f,2f}} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{1,1}} & \hat{\mathbf{U}}_{\mathbf{1,2r}} & \hat{\mathbf{U}}_{\mathbf{1,2f}} \\ & \hat{\mathbf{U}}_{\mathbf{2r,2r}} & \hat{\mathbf{U}}_{\mathbf{2r,2f}} \\ & & \hat{\mathbf{U}}_{\mathbf{2f,2f}} \end{bmatrix}.$$

Matrix $\mathbf{L}_{\mathbf{exp}}$, $\mathbf{U}_{\mathbf{exp}}$, $\mathbf{G}_{\mathbf{1}}$ and $\mathbf{G}_{\mathbf{2}}$ are from the exact LU factorization of the expanded part, which implies

$$\mathbf{G}_{\mathbf{1}}\mathbf{G}_{\mathbf{2}} + \mathbf{L}_{\mathbf{exp}}\mathbf{U}_{\mathbf{exp}} = \mathbf{0}.$$

Thus, we have

$$\mathbf{L_{exp}U_{exp}} = -\mathbf{G_1G_2} = -\begin{bmatrix} \mathbf{0} & \mathbf{0} & \tilde{\mathbf{B}}^{\mathbf{T}} \end{bmatrix} \hat{\mathbf{U}}^{-1}\hat{\mathbf{L}}^{-1} \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \tilde{\mathbf{B}} \end{bmatrix}$$

$$= -\tilde{\mathbf{B}}^{\mathbf{T}}{\hat{\mathbf{U}}_{\mathbf{2f,2f}}}^{-1}\hat{\mathbf{L}}^{-1}_{\mathbf{2f,2f}}\tilde{\mathbf{B}},$$

We solve the linear system (5.6) iteratively using (5.7) as preconditioner. Similar to the previous section, with m_f system solves of (5.6), we get the equivalent capacitances among regular conductors. Though the condition number of the linear system (5.6) is worse compared with (3.1), the iterative solver preconditioned by (5.7) converges very fast.

3. Expanded RedCap

We approximate (5.6) using it is approximate factorization (5.7). The approximate linear system is

$$\begin{bmatrix} \hat{\mathbf{L}}_{\mathbf{1,1}} & & & \\ \hat{\mathbf{L}}_{\mathbf{2r,1}} & \hat{\mathbf{L}}_{\mathbf{2r,2r}} & & \\ \hat{\mathbf{L}}_{\mathbf{2f,1}} & \hat{\mathbf{L}}_{\mathbf{2f,2r}} & \hat{\mathbf{L}}_{\mathbf{2f,2f}} & \\ \mathbf{0} & \mathbf{0} & -\tilde{\mathbf{B}}^{\mathbf{T}}\hat{\mathbf{U}}^{-\mathbf{1}}_{\mathbf{2f,2f}} & \mathbf{L_{exp}} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{1,1}} & \hat{\mathbf{U}}_{\mathbf{1,2r}} & \hat{\mathbf{U}}_{\mathbf{1,2f}} & \mathbf{0} \\ & \hat{\mathbf{U}}_{\mathbf{2r,2r}} & \hat{\mathbf{U}}_{\mathbf{2r,2f}} & \mathbf{0} \\ & & \hat{\mathbf{U}}_{\mathbf{2f,2f}} & -\hat{\mathbf{L}}^{-\mathbf{1}}_{\mathbf{2f,2f}}\tilde{B} \\ & & & \mathbf{U_{exp}} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{q}}_{\mathbf{1}} \\ \tilde{\mathbf{q}}_{\mathbf{2r}} \\ \tilde{\mathbf{q}}_{\mathbf{2f}} \\ \mathbf{V_f} \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{v}}_{\mathbf{2r}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

We identify the small sub-system

$$\begin{bmatrix} \hat{\mathbf{L}}_{\mathbf{2r,2r}} & & \\ \hat{\mathbf{L}}_{\mathbf{2f,2r}} & \hat{\mathbf{L}}_{\mathbf{2f,2f}} & \\ \mathbf{0} & -\tilde{\mathbf{B}}^{\mathbf{T}}\hat{\mathbf{U}}^{-1}_{\mathbf{2f,2f}} & \mathbf{L_{exp}} \end{bmatrix} \cdot \begin{bmatrix} \hat{\mathbf{U}}_{\mathbf{2r,2r}} & \hat{\mathbf{U}}_{\mathbf{2r,2f}} & \mathbf{0} \\ & \hat{\mathbf{U}}_{\mathbf{2f,2f}} & -\hat{\mathbf{L}}^{-1}_{\mathbf{2f,2f}}\tilde{\mathbf{B}} \\ & & \mathbf{U_{exp}} \end{bmatrix} \cdot \begin{bmatrix} \hat{\mathbf{q}}_{\mathbf{2r}} \\ \hat{\mathbf{q}}_{\mathbf{2f}} \\ \mathbf{v_F} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{v}}_{\mathbf{2r}} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \tag{5.8}$$

and solve it m_r times for $\hat{\mathbf{q}}_{\mathbf{2r}}$ with the consideration of floating dummy conductors.

C. Experimental results

Experiments with floating dummy conductors are reported in Fig. 19. The four methods in Table XI are compared. Overall, the methods using wavelet bases, including the expanded PHiCap algorithm and the expanded RedCap algorithm, are much faster than the methods using the standard basis, such as the conventional method and the expanded HiCap algorithm. In Table XI, the running time of the expanded RedCap algorithm does not change much regardless of the number of dummy conductors. This is because the solving time takes a very small portion of the total running time. The expanded PHiCap converges fast, often within 3 iterations, when preconditioned using (5.7), regardless of the number of dummy conductors. For the expanded HiCap method, the iteration per solve increase greatly with the increase of dummy conductors, which means the condition number of system (5.3) deteriorates. The total iterations needed by the expanded HiCap method is even larger than the conventional method, although fewer system solves are needed for the expanded HiCap method.

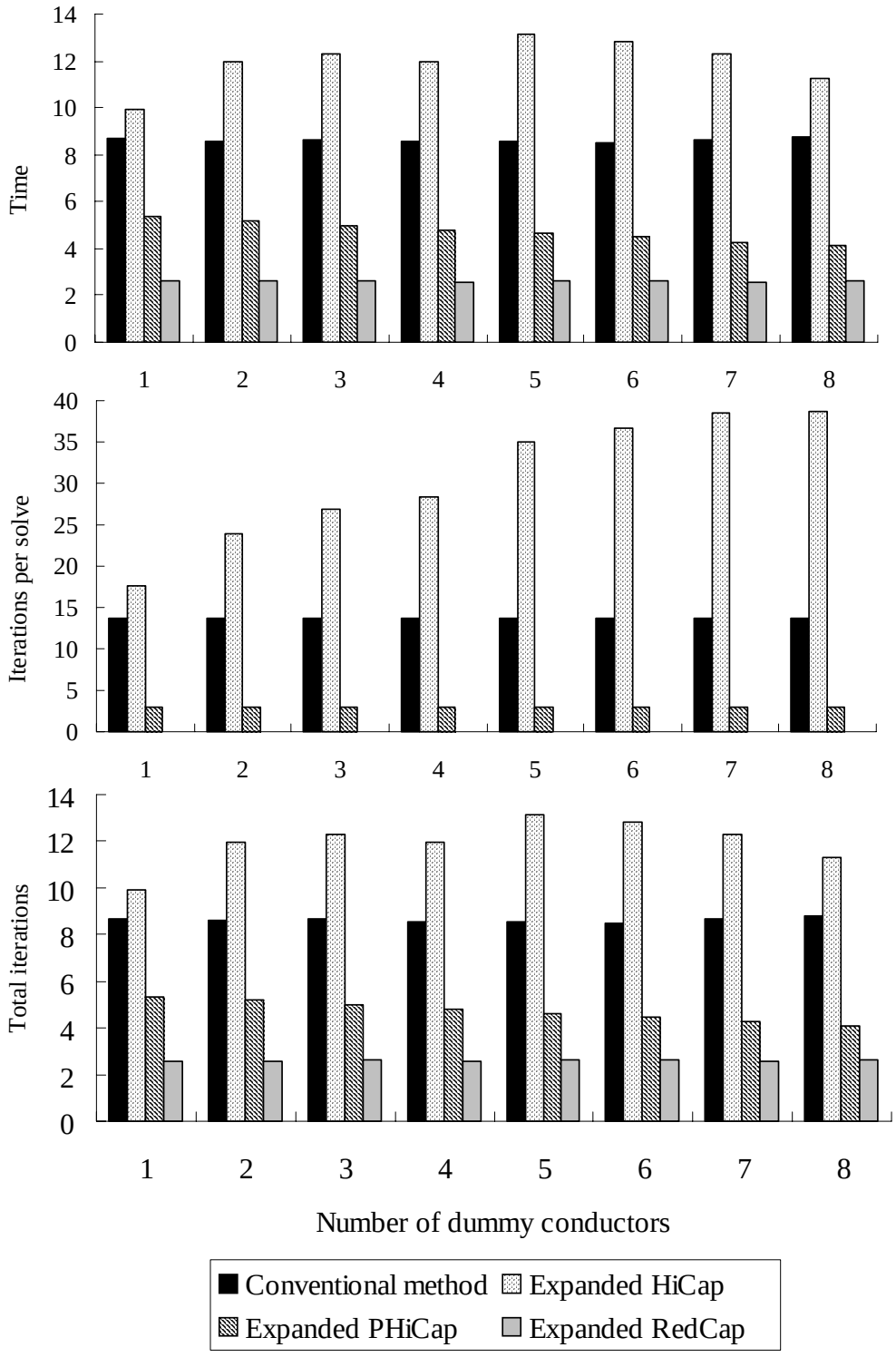


Fig. 19. Comparison of four methods with different number of dummy conductors in 8×8 bus crossing examples.

D. Summary

Table XI summarizes the behavior of four extraction algorithms considering the floating dummy conductors.

Table XI. Four extraction algorithms considering the floating dummy conductors. As defined earlier, n is the number of panels, m_r and m_f are the number of regular and floating dummy conductors, respectively.

	Conventional method (HiCap)	Expanded HiCap	Expanded PHiCap	Expanded RedCap
Linear system	(1.8)	(5.3)	(5.6)	(5.8)
Dimension	n	$n + O(m_f)$	$n + O(m_f)$	$O(m_r + m_f)$
No. of solves	$m_r + m_f$	m_r	m_r	m_r
Solver	iterative	iterative	iterative*	direct
Convergence	bad	worst	good	—
Efficiency**	bad	worst	good	best

* use (5.7) as preconditioner.

** refer to experimental results in Fig. 19.

In the presence of floating dummy-fills, the equivalent capacitances among non-dummy conductors are needed. In this chapter, we proposed a scheme to embed the global charge and potential conditions of floating dummy conductors into the extraction linear system, and compute the equivalent capacitances for regular conductors directly. This idea is applied to HiCap, PHiCap and RedCap. Experiments show that the methods based on wavelet basis are much more efficient than the methods based on standard basis and the method based on RedCap is most efficient. The method based on standard basis are unattractive due to the slow convergence.

CHAPTER VI

SELECTIVE COEFFICIENT ENHANCEMENT METHOD

A. Introduction

As discussed in chapter I, due to the numerical integration of (1.3) or (1.7) and the truncation of multipole expansions, error is introduced in the step of calculating $\mathbf{P}$. In this chapter, we present the sensitivity analysis for the accuracy of the capacitances over the accuracy of the potential coefficient p_{ij} or e_{ij}. Based on the sensitivity analysis, we propose a method which enhance the accuracy of selected coupling or self-capacitances by only increasing the accuracy of those potential coefficients that are critical.

The technique is desirable for such situations as crosstalk and signal integrity analysis, where only the capacitances between some pairs of conductors need to be computed with high accuracy.

B. Sensitivity analysis for the accuracy of $\mathbf{C}$

We first present a theory on the accuracy of selected entries of $\mathbf{C}$. Due to the error caused by the second step of BEM mentioned in section C of chapter I, instead of $\mathbf{Pq} = \mathbf{v}$, the linear system we actually solve is

$$\bar{\mathbf{P}}\bar{\mathbf{q}} = \mathbf{v}, \tag{6.1}$$

where $\bar{\mathbf{P}}$ is an approximation of $\mathbf{P}$ and $\bar{\mathbf{q}}$ is the corresponding solution. As a result, the capacitance matrix obtained is an approximation $\bar{\mathbf{C}}$ instead of $\mathbf{C}$.

Assume there are k_i panels on the i-th conductor, and the panels are numbered in ascending order from the first conductor to the m-th conductor. To calculate the j-th

column of $\bar{\mathbf{C}}$, we first solve (6.1) with $\mathbf{v} = \mathbf{v}^{(j)} = [0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0]^T$, where the first $k_1 + \cdots + k_{j-1}$ entries of $\mathbf{v}^{(j)}$ are 0, followed by k_j 1's and $n - (k_1 + \cdots + k_j)$ 0's. Then for $i = 1, 2, \ldots, m$,

$$\bar{C}_{ij} = \sum_{l=k_1+\cdots+k_{i-1}+1}^{k_1+\cdots+k_i} \bar{\mathbf{q}}_l^{(j)}, \tag{6.2}$$

where $\bar{\mathbf{q}}^{(j)}$ is the solution of (6.1) corresponding to the right-hand side $\mathbf{v} = \mathbf{v}^{(j)}$. Similarly, if $\mathbf{q}^{(j)}$ denotes the solution of (1.8) corresponding to the right-hand side $\mathbf{v} = \mathbf{v}^{(j)}$, then entry C_{ij} of the capacitance matrix is given by

$$C_{ij} = \sum_{l=k_1+\cdots+k_{i-1}+1}^{k_1+\cdots+k_i} \mathbf{q}_l^{(j)}. \tag{6.3}$$

Let $\mathbf{r}^{(l)}$ denote the l-th column of $\mathbf{P}^{-1}$, then we have the following result.

Lemma 1 *For every $j = 1, 2, \ldots, m$, vector $\mathbf{q}^{(j)}$ and the columns of $\mathbf{P}^{-1}$ satisfy the equation*

$$\mathbf{q}^{(j)} = \sum_{l=k_1+\cdots+k_{j-1}+1}^{k_1+\cdots+k_j} \mathbf{r}^{(l)}.$$

Proof. It follows from the definition of $\mathbf{r}^{(l)}$ that $\mathbf{P}\mathbf{r}^{(l)} = \mathbf{e}^{(l)}$ where $\mathbf{e}^{(l)}$ denotes the l-th column of the identity matrix. Therefore

$$\begin{aligned} \mathbf{P} \cdot \Big(\sum_{l=k_1+\cdots+k_{j-1}+1}^{k_1+\cdots+k_j} \mathbf{r}^{(l)} \Big) &= \sum_{l=k_1+\cdots+k_{j-1}+1}^{k_1+\cdots+k_j} \mathbf{P}\mathbf{r}^{(l)} \\ &= \sum_{l=k_1+\cdots+k_{j-1}+1}^{k_1+\cdots+k_j} \mathbf{e}^{(l)} \\ &= \mathbf{v}^{(j)}. \end{aligned}$$

On the other hand, $\mathbf{P}q^{(j)} = \mathbf{v}^{(j)}$, therefore

$$\mathbf{q}^{(j)} = \sum_{l=k_1+\cdots+k_{j-1}+1}^{k_1+\cdots+k_j} \mathbf{r}^{(l)}.$$

□

Let $\mathbf{E} = \bar{\mathbf{P}} - \mathbf{P}$ be the error matrix. Then the following Lemma tells the relationship between the error of $\bar{C}_{ij}$ and $\mathbf{E}$.

Lemma 2 *Let $\bar{C}_{ij}$ and C_{ij} be given by (6.2) and (6.3) respectively, then $C_{ij} - \bar{C}_{ij} = (\mathbf{q}^{(i)})^T \mathbf{E} \bar{\mathbf{q}}^{(j)}$.*

Proof. From $\bar{\mathbf{P}}\bar{\mathbf{q}}^{(j)} = \mathbf{v}^{(j)}$ and $\mathbf{P}\mathbf{q}^{(j)} = \mathbf{v}^{(j)}$, we have $\mathbf{P}^{-1}\bar{\mathbf{P}}\bar{\mathbf{q}}^{(j)} = \mathbf{P}^{-1}\mathbf{v}^{(j)} = \mathbf{q}^{(j)}$. In other words, $\mathbf{P}^{-1}(\mathbf{P} + \mathbf{E})\bar{\mathbf{q}}^{(j)} = \mathbf{q}^{(j)}$. Hence $\mathbf{q}^{(j)} - \bar{\mathbf{q}}^{(j)} = \mathbf{P}^{-1}\mathbf{E}\bar{\mathbf{q}}^{(j)}$, which implies $q_l^{(j)} - \bar{q}_l^{(j)} = (\mathbf{P}^{-1}\mathbf{E}\mathbf{q}^{(j)})_l = (\mathbf{r}^{(l)})^T \mathbf{E}\mathbf{q}^{(j)}$, for any $j = 1, 2, \ldots, m$ and $l = 1, 2, \ldots, n$. The last equality holds because the matrix $\mathbf{P}$ is symmetric. Now using Lemma 1, formulas (6.2), (6.3), we have

$$\begin{aligned} C_{ij} - \bar{C}_{ij} &= \sum_{l=k_1+\cdots+k_{i-1}+1}^{k_1+\cdots+k_i} (q_l^{(j)} - \bar{q}_l^{(j)}) \\ &= \sum_{l=k_1+\cdots+k_{i-1}+1}^{k_1+\cdots+k_i} (\mathbf{r}^{(l)})^T \mathbf{E}\mathbf{q}^{(j)} \\ &= (\mathbf{q}^{(i)})^T \mathbf{E}\bar{\mathbf{q}}^{(j)}. \end{aligned}$$

□

Lemma 3 *Let $\mathbf{d} = \bar{\mathbf{q}} - \mathbf{q}$, $Cond(\mathbf{P})$ be the condition number of $\mathbf{P}$, and assume $\|\mathbf{E}\|/\|\mathbf{P}\| \leq 1/Cond(\mathbf{P})$. Then*

$$\mathbf{d} = -(\mathbf{I} + \mathbf{P}^{-1}\mathbf{E})^{-1}\mathbf{P}^{-1}\mathbf{E}\mathbf{q}.$$

Proof. Since $\mathbf{v} = \mathbf{P}\mathbf{q}$ and $\mathbf{v} = \bar{\mathbf{P}}\bar{\mathbf{q}} = (\mathbf{P} + \mathbf{E})(\mathbf{q} + \mathbf{d})$, we have

$$(\mathbf{P} + \mathbf{E})\mathbf{d} = -\mathbf{E}\mathbf{q}.$$

Therefore

$$\mathbf{d} = -(\mathbf{P} + \mathbf{E})^{-1}\mathbf{E}\mathbf{q} = -(\mathbf{I} + \mathbf{P}^{-1}\mathbf{E})^{-1}\mathbf{P}^{-1}\mathbf{E}\mathbf{q},$$

where $(\mathbf{I}+\mathbf{P^{-1}E})^{-1}$ exists because of the assumption. □

The condition $\|\mathbf{E}\|/\|\mathbf{P}\| \leq 1/Cond(\mathbf{P})$ is usually satisfied in practice. It simply says that the relative error of $\bar{\mathbf{P}}$ is small in comparison with $1/Cond(\mathbf{P})$.

Theorem 1 *Under the notations and assumptions of previous Lemmas, we have the following error representation*

$$C_{ij} - \bar{C}_{ij} = (\mathbf{q}^{(i)})^T \mathbf{E} \mathbf{q}^{(j)} - O(\|\mathbf{E}\|^2).$$

Proof. From Lemma 2 and Lemma 3,

$$\begin{aligned} & C_{ij} - \bar{C}_{ij} \\ = \; & (\mathbf{q}^{(i)})^T \mathbf{E} \bar{\mathbf{q}}^{(j)} \\ = \; & (\mathbf{q}^{(i)})^T \mathbf{E} \mathbf{q}^{(j)} - (\mathbf{q}^{(i)})^T \mathbf{E} \left(\mathbf{I}+\mathbf{P^{-1}E}\right)^{-1} \mathbf{P^{-1}E} \mathbf{q}^{(j)} \\ = \; & (\mathbf{q}^{(i)})^T \mathbf{E} \mathbf{q}^{(j)} - O(\|\mathbf{E}\|^2). \end{aligned}$$

□

Corollary 1 *Let e_{kl} be any entry of $\mathbf{E}$, then*

$$\begin{aligned} & |C_{ij} - \bar{C}_{ij}| \\ \leq \; & |(\mathbf{q^{(i)}})^{\mathbf{T}}| \cdot |\mathbf{E}| \cdot |\mathbf{q^{(j)}}| + \mathbf{O}(\|\mathbf{E}\|^{\mathbf{2}}) \\ = \; & \sum_{k=1}^{n} \sum_{l=1}^{n} |q_k^{(i)}| \cdot |e_{kl}| \cdot |q_l^{(j)}| + O(\|\mathbf{E}\|^2) \\ = \; & \sum_{k=1}^{n} \sum_{l=1}^{n} (|q_k^{(i)}| \cdot |p_{kl}| \cdot |q_l^{(j)}|) \cdot \left|\frac{e_{kl}}{p_{kl}}\right| + O(\|\mathbf{E}\|^2). \end{aligned}$$

Proof. Follows immediately from Theorem 1. □

The relative errors $\left|\frac{e_{kl}}{p_{kl}}\right|$ for all pairs of k, l are usually of the same magnitude,

depending on how the coefficients are computed. Corollary 1 says that the relative error is magnified by a factor $|q_k^{(i)}| \cdot |p_{kl}| \cdot |q_l^{(j)}|$. Therefore for those $\bar{p}_{kl}$ with large corresponding factor $|q_k^{(i)}| \cdot |p_{kl}| \cdot |q_l^{(j)}|$, we should compute $\bar{p}_{kl}$ with high accuracy. This helps to reduce the error for $|C_{ij} - \bar{C}_{ij}|$.

C. Selective coefficient enhancement algorithm

Selective Coefficient Enhancement Scheme

Input: Conductors, integers i, j and threshold θ.

Output: Capacitance matrix $\mathbf{C}$, where C_{ij} is computed at high accuracy.

Phase I: Initial computation.

1: Calculate $\mathbf{P}$ with ordinary accuracy.

2: For each conductor l, solve $\mathbf{P}\mathbf{q}^{(l)} = \mathbf{v}^{(l)}$.

3: Compute matrix $\mathbf{C}$.

Phase II: Selective coefficient enhancement.

4: For each entry p_{kl} of $\mathbf{P}$, if $|q_k^{(i)} \cdot p_{kl} \cdot q_l^{(j)}| \geq \theta$,
recalculate p_{kl} using a more accurate method,
and let the new coefficient matrix be $\mathbf{P'}$.

5: Solve equation $\mathbf{P'}\mathbf{q}^{(i)} = \mathbf{v}^{(i)}$ using the solution obtained in step 2 as the initial value.

6: Compute C_{ij} using $\mathbf{q}^{(i)}$.

In circuit simulation, the aggressor net and the victim net are often known in advance. Therefore we need to compute the coupling capacitance between the aggressor and the victim with high accuracy. For this situation, we have the following 2-phase scheme based on the above theory. In the first phase, the capacitance matrix is computed with the coefficient matrix of ordinary accuracy. In the second phase, potential coefficients are selectively refined according to charges computed in the first phase. The selected coupling capacitance is then recalculated.

In the algorithm, i and j denote the capacitance entry C_{ij} that needs to be computed to high accuracy. Input variable threshold θ is a user supplied value that affects the final accuracy for C_{ij}.

D. Experimental results

The implemented algorithm is executed on a SUN UltraSPARC Enterprise 4000, and tested for the 8 bus example shown in Fig. 20. Note that in step 4, the naive way to check all entries p_{kl} of $\mathbf{P}$ takes $O(n^2)$ time. However, since we use the HiCap algorithm [3] (chapter II), which stores matrix $\mathbf{P}$ with $O(n)$ links, step 4 takes only $O(n)$ time.

Table XII gives the experimental results, where the traditional method is either "no enhancement", meaning that all coefficients are computed at ordinary accuracy, or "full enhancement", meaning that all coefficients are computed at high accuracy. The high accuracy method uses 3×3 Gaussian quadrature, while the ordinary accuracy method uses the single point. The conductors are labeled from 0 at one side to 7 at the other side. Only self capacitance C_{ii} or coupling capacitance C_{ij} between adjacent conductors are significant enough to be considered. The error of a capacitance entry $\bar{C}_{ij}$ is defined as $|\bar{C}_{ij} - C_{ij}|/|C_{ij}|$, where C_{ij} is the capacitance computed by full

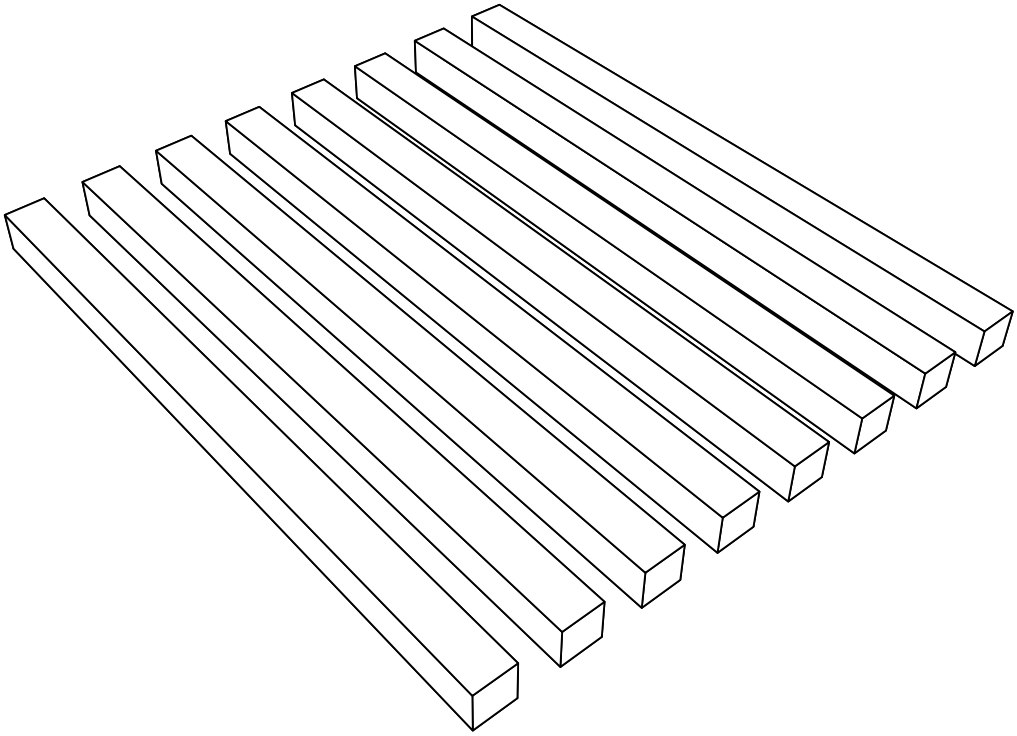


Fig. 20. The 8 bus example.

Table XII. Experimental results of selective capacitance enhancement algorithm. Time is CPU seconds. Error is with respect to full enhancement.

Capacitance		C_{00}	C_{01}	C_{10}	C_{11}	C_{12}	C_{21}	C_{22}	C_{23}	C_{32}	C_{33}
No	Error	2.62	3.94	3.99	3.61	4.56	4.41	3.53	4.21	4.15	3.57
Enhance.	Time	59.18									
	Error	0.88	0.79	0.77	0.61	0.95	0.89	0.92	0.54	0.62	0.94
Selective	Time	64.8	64.6	66.2	67.6	65.7	65.8	66.8	65.9	65.5	66.9
Enhance.	Time Increase	9.5	9.2	11.9	14.3	11.0	11.2	12.9	11.4	10.7	13.1
	Error	0									
Full	Time	83.25									
Enhance.	Time Increase	41.1									

enhancement method. That is why the full enhancement method has no error in the table. The "time increase" field is compared with the "no enhancement" method. Our GMRES reduces the two-norm residual to 1% from the initial residual.

The "no enhancement" method gives a less accurate capacitance matrix, while the "full enhancement" method consumes longer running time, compared with our selective coefficient enhancement scheme. Our method is suitable for applications where only a few entries of $\mathbf{C}$ need to be computed accurately. If we want to compute all entries of $\mathbf{C}$ accurately, then the full enhancement method is still faster.

E. Summary

Using the sensitivity analysis for linear systems, we present a theory that clarifies the relationship between the error of each entry of the capacitance matrix and the error of the potential coefficients. Based on the theory, we propose a technique which enhances the accuracy of user selected coupling or self-capacitances by increasing the accuracy of those potential coefficients which are critical. Experiments show that the accuracy for selected entries of the capacitance matrix can be improved greatly with a small increase in the overall computation time. This technique is very useful for accurate crosstalk and signal integrity analysis.

CHAPTER VII

VARIABLE ORDER MULTIPOLE METHOD

A. Introduction

We discussed the three steps of BEM and the corresponding three sources of error in chapter I. In the FMM, the potential due to a set of charges located within a region is approximated by a multipole series. This approximation causes error in the step of calculating $\mathbf{P}$, which eventually reduces the accuracy of capacitance solution.

In this chapter, starting from the error estimation of the multipole approximation, we propose the variable order multipole method that selects nodes with large unbalanced charge distribution for high order expansion, while keeps the expansion order of other nodes unchanged. Finally, we present the experiments showing the technique is effective and practical in improving the overall accuracy of the capacitance matrix.

B. Error estimation of the multipole approximation

For FMM, error is introduced by the far field approximation. The traditional error estimation of FMM [27] is expressed as a function of the radius of the sphere and the distance to the observation point, but not expressed as a function of the location and amount of charge. In the following, we present a better error estimation where the locations and values of charges are taken into account.

For simplicity, let us consider the hierarchical refinement algorithm of chapter II [3]. The hierarchical refinement algorithm can be viewed as the 0-th order FMM, and is much easier to describe than the general FMM [27].

Theorem 2 *Assume we partition a panel A_k into two small panels A_1 and A_2 of*

equal shape and size. Let the radius of the smallest sphere that contains A_k be r_s. Consider panel A_l of distance r from the center of the sphere, for some $r > r_s$. Then the error of potential at A_l due to using A_k with charge $(q_1 + q_2)$ to approximate A_1 and A_2 with charges q_1 and q_2 respectively, is about

$$\frac{|q_2 - q_1|}{2} \cdot \frac{r_s}{r} \cdot p_{kl}.$$

where p_{kl} is define in (3).

Proof. The potential at $x \in A_l$ due to the charge on panels A_1 and A_2, with uniform charge densities $\sigma_1 = q_1/area(A_1)$ and $\sigma_2 = q_2/area(A_2)$ is

$$\int_{x' \in A_1} \frac{\sigma_1}{4\pi\epsilon_0 \|x' - x\|} da' + \int_{x' \in A_2} \frac{\sigma_2}{4\pi\epsilon_0 \|x' - x\|} da'. \tag{7.1}$$

If we treat A_1 and A_2 as a single panel A with uniform charge density $(\sigma_1 + \sigma_2)/2$, then the potential at x will be

$$\int_{x' \in A} \frac{\sigma_1 + \sigma_2}{2} \frac{1}{4\pi\epsilon_0 \|x' - x\|} da'. \tag{7.2}$$

Assume without loss of generality $\sigma_2 \geq \sigma_1$, then the difference between (7.1) and (7.2) is

$$\begin{aligned}
& \frac{\sigma_2 - \sigma_1}{2} \frac{1}{4\pi\epsilon_0} \left(\int_{A_1} \frac{1}{\|x' - x\|} da' - \int_{A_2} \frac{1}{\|x' - x\|} da' \right) \\
\leq\ & \frac{\sigma_2 - \sigma_1}{2} \frac{1}{4\pi\epsilon_0} \int_{A_1} \left(\frac{1}{\|x' - x\|} - \frac{1}{\|x' - x\| + r_s} \right) da' \\
\leq\ & \frac{\sigma_2 - \sigma_1}{2} \frac{r_s}{r} \int_{A_1} \frac{1}{4\pi\epsilon_0 \|x' - x\|} da' \\
\approx\ & \frac{q_2 - q_1}{2} \cdot \frac{r_s}{r} \cdot p_{kl}.
\end{aligned}$$

□

The ratio r_s/r decreases as r increases, which is fully exploited in the FMM. Now,

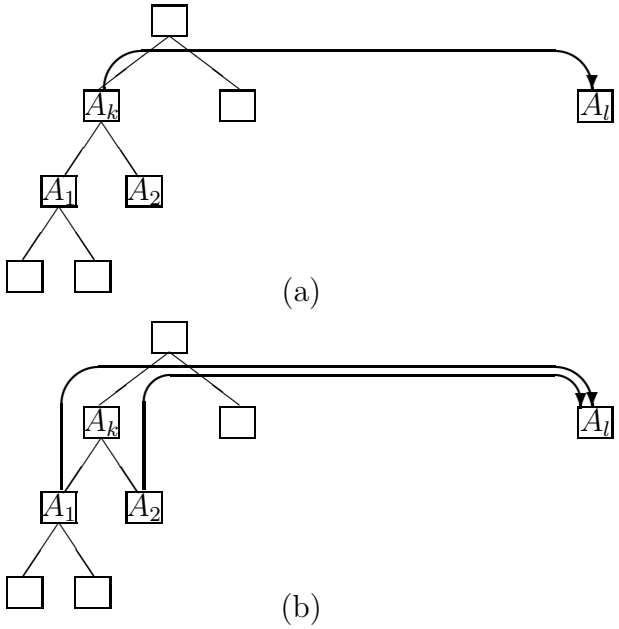


Fig. 21. In (a), assume $|q_2 - q_1| \cdot p_{kl}$ is less than a user supplied error bound. Therefore A_k interacts with A_l. In (b) assume otherwise. A_k passes the interaction down to A_1 and A_2.

we exploit the other factor $q_2 - q_1$, which has not been exploited before.

Previous study has illustrated the correlation between the error of potential and the error of charge distribution, see [28]. From this correlation, we expect to see a reduction of error in charge distribution if we reduce the error in potential. Our variable order multipole idea is thus derived from Theorem 2 and illustrated in Fig. 21. On the left side of Fig. 21, there is a hierarchy of panel discretization, where panel A_k is discretized as A_1 and A_2. In Fig. 21(a), assume $|q_2 - q_1| \cdot p_{kl} \leq \theta$, where θ is a user supplied threshold. Therefore the error given in Theorem 2 is small, and A_1 and A_2 are treated as one panel A_k when interacting with A_l. In Fig. 21(b), assume $|q_2 - q_1| \cdot p_{kl} > \theta$. Therefore A_1 and A_2 will interact with A_l directly. The interaction in Fig. 21(a) corresponds to 0-th order multipole, and the interactions in Fig. 21(b) can be viewed as (1/2)-th order multipole. (It is equivalent to (1/2)-th order multipole because of the amount of information computed. If we go down two levels, then it will be comparable to 1st order multipole.) The concept can be applied recursively for A_1 and A_2 respectively. As a result, panels high in the hierarchy tend to pass the interaction down more often than panels low in the hierarchy, and panels contain non-uniform charge tend to pass the interaction down more often than panels contain uniform charge distribution.

C. Variable order multipole algorithm

Variable Order Multipole Scheme

Input: Conductors and threshold θ.

Output: Capacitance matrix $\mathbf{C}$.

1: Build low order multipole structure $\mathbf{P}^1$.

2: For each conductor i

3: $t \leftarrow 1$.

4: Repeat

5: Run GMRES to solve $\mathbf{P}^t \cdot \mathbf{q} = \mathbf{v}$ for one iteration, and let the result be $\mathbf{q}^t$.

6: For each multipole coefficient p_{kl}^t,
if $\left(\left|\left(q_{l_{left}}^t - q_{l_{right}}^t\right) \cdot p_{kl}^t\right| > \theta\right)$
then use high order expansion for p_{kl}^{t+1},
else use p_{kl}^t for p_{kl}^{t+1}.

7: $t \leftarrow t + 1$.

8: Until GMRES converges.

9: Compute the i-th row of $\mathbf{C}$ from $\mathbf{q}^t$.

Based on the error estimation presented in the previous section, we propose the variable order multipole algorithm. Compared with the traditional FMM algorithms [4, 3] that improves the accuracy by increasing the expansion order for all nodes, our new scheme selects some nodes for high order expansion and leaves other nodes with ordinary expansion order. For nodes with large unbalanced charge distribution and large potential coefficient, i.e., large $\left|\left(q_{l_{left}}^t - q_{l_{right}}^t\right) \cdot p_{kl}^t\right|$, we use high order expansion.

Threshold θ is a user defined value that decides which coefficients are selected for high order expansion. In our implementation, high order expansion means going down the hierarchy as shown in Fig. 21. Note that in step 6, we also rely on the fact that our algorithm stores the potential coefficients matrix in a data structure of size $O(n)$ [3]. Therefore step 6 takes only $O(n)$ time.

D. Experimental results

Experimental result is shown in Fig. 22. The implemented algorithm is executed on a SUN UltraSPARC for the 4×4 conductor example used in [4] and [3]. The error of capacitance matrix $\bar{\mathbf{C}}$ is defined as $\|\bar{\mathbf{C}} - \mathbf{C}\|/\|\mathbf{C}\|$, where norm $\|\cdot\|$ is the Frobenius norm, and $\mathbf{C}$ is the accurate capacitance matrix computed by using direct method without FMM. Our GMRES reduces the two-norm residual to 1% from the initial residual.

In Fig. 22, point **A** is for the case where all nodes have ordinary expansion order and point **B** is for the case where all nodes have high expansion order. Both **A** and **B** are based on the traditional hierarchical method. Points between **A** and **B** are from variable order multipole method with various θ.

From the curve in Fig. 22, we find point **C** has comparable accuracy as **B** while uses much less time. This shows the advantage of the proposed scheme over traditional methods.

E. Summary

In this chapter, we first present the error estimation of the multipole approximation. Unlike the traditional error estimation of the multipole approximation, we consider the locations and values of the charges in our error estimation as well. Then, based on the new error estimation, we propose the variable order multipole method that selects nodes with large unbalanced charge distribution for high order expansion, while keeps the expansion order of other nodes unchanged. The experiments show that the variable order multipole method is effective and practical in improving the overall accuracy of the capacitance matrix.

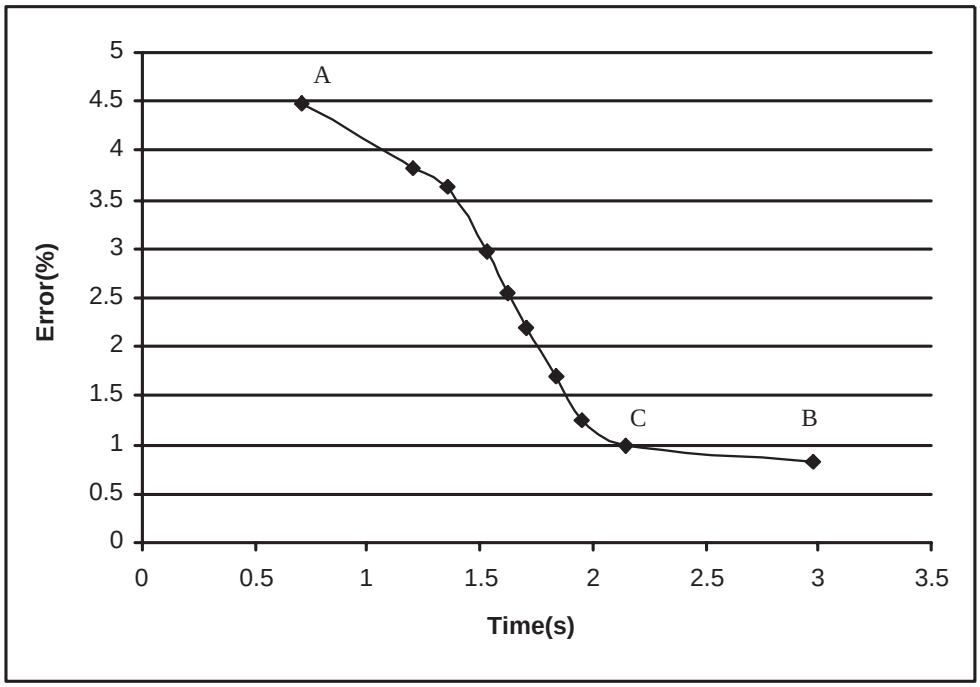


Fig. 22. Experimental result for variable order multipole scheme.

CHAPTER VIII

MULTIGRID METHOD

A. Introduction

According to the discussion in chapter I, in the discretization step, approximation error is introduced due to the assumption of uniform charge distribution on each panel. This error is the dominant approximation error of BEM. Ideally, very fine discretization is desirable. However, fine discretization results in large linear systems and unacceptable computation time. To reduce the time cost while preserving high accuracy, we propose a multigrid-like scheme to acquire a good initial solution from coarse discretization.

B. Multigrid method for capacitance extraction

The multigrid method is a fast linear iterative solver based on the multilevel or multi-scale paradigm [29]. The multigrid method uses several levels of refinement. The solution of each level is mapped to the next (finer) level and used as the initial value for solving the next (finer) level. Compared with the traditional method, the multigrid method may use the same number of iterations. However most systems solved by the multigrid method are small, thereby saving the total time. The multigrid method can be applied in combination with any of the common discretization techniques. In this section, we apply multigrid method to capacitance extraction. The algorithm is outlined below.

In this algorithm, charges on each coarse panel are evenly mapped to its fine panels. Different convergence criteria ϵ_{coarse} and ϵ_{fine} are used for coarse level and fine level respectively. Usually, small ϵ_{fine} is chosen to guarantee the accuracy of the

final solution. On the other hand, large ϵ_{coarse} is chosen because it provides sufficiently good initial solution at low computation cost.

Multigrid Scheme

Input: Conductors, convergence criteria

ϵ_{coarse} and ϵ_{fine}.

Output: Capacitance matrix $\mathbf{C}$.

1: Discretize conductor surfaces coarsely and build
   corresponding coarse level linear system
   $\mathbf{P_{coarse}} \cdot \mathbf{q_{coarse}} = \mathbf{v_{coarse}}$.
2: Further discretize conductor surfaces and build
   corresponding fine level linear system
   $\mathbf{P_{fine}} \cdot \mathbf{q_{fine}} = \mathbf{v_{fine}}$.
3: For each conductor i,
4: Solve coarse level linear system for $\mathbf{q_{coarse}}^{(i)}$
   with convergence criteria ϵ_{coarse}.
5: Map $\mathbf{q_{coarse}}^{(i)}$ to fine discretization $\mathbf{q^0_{fine}}^{(i)}$.
6: Solve fine level linear system for $\mathbf{q_{fine}}^{(i)}$
   using $\mathbf{q^0_{fine}}^{(i)}$ as the initial value,
   with convergence criteria ϵ_{fine}.
7: Compute i-th row of matrix $\mathbf{C}$ using $\mathbf{q_{fine}}^{(i)}$.

C. Experimental results

Fig. 23 is a comparison of multigrid method and traditional method for the 4×4 bus crossing example (Fig. 11). For the traditional method, the conductors are

divided into 17408 panels and the large system is solved directly with the convergence criteria $\epsilon = 0.01$. It takes 16 iterations to reduce the residual to less than 0.01. The computation time is 34.84 sec. For the multigrid method, the conductors are divided into 576 panels at the coarse level and further divided into 17408 panels at the fine level. With convergence criteria $\epsilon_{coarse} = 0.1$ and $\epsilon_{fine} = 0.01$, the number of iterations for coarse level and fine level are 5 and 11, respectively. Compared with the traditional method, multigrid method saves 5 iterations of solving a large system at the expense of 5 iterations of solving a small system. The total computation time is 26.72 sec, which is 23.2% less than that of the traditional method.

Our experience tells us that using more than two levels of refinement does not provide additional benefit. This is because for capacitance extraction, the total number of iterations is small. Therefore if there are three or more linear systems, then the number of iterations for the finest level will not decrease, while the overhead for setting up linear systems will increase.

D. Summary

In this chapter, we apply multigrid method for the first time to capacitance extraction. The method solves the linear system with fine discretization faster by using the solution of coarse discretization as a good initial solution for the linear solver. The experiments show the potential of the multigrid method.

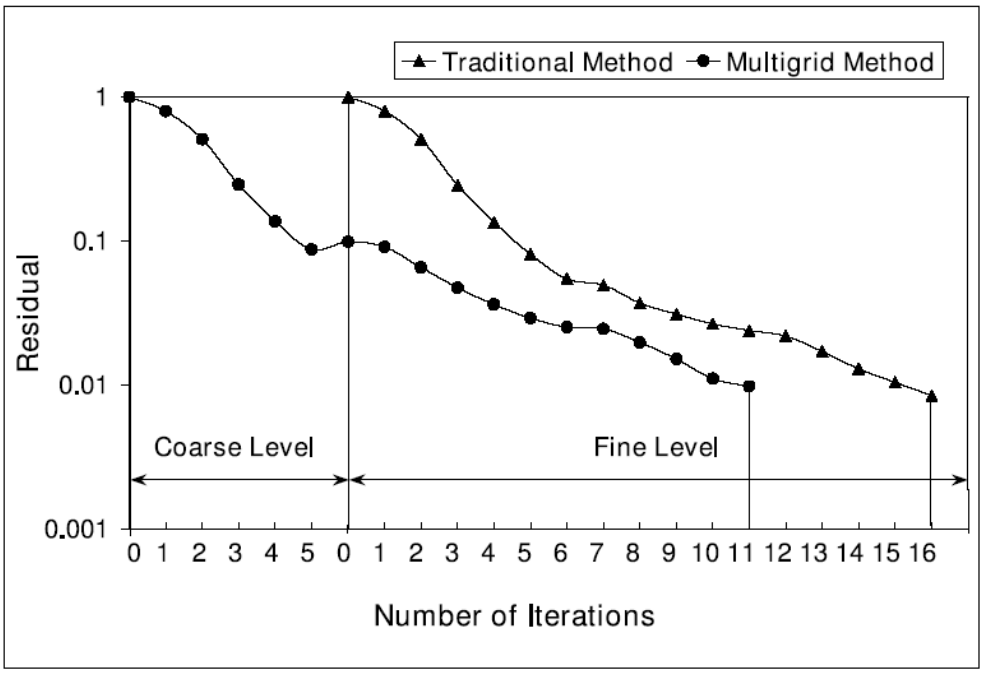


Fig. 23. Experimental result for multigrid scheme.

CHAPTER IX

CONCLUSIONS

In this dissertation, we presented several novel techniques based on BEM for the capacitance extraction problem. Our algorithms are significantly faster than existing techniques and have comparable accuracy.

We proposed the PHiCap algorithm, which is based on the hierarchical refinement algorithm and the wavelet transformation. Unlike traditional algorithms which result in dense linear systems, PHiCap reduces the capacitance extraction problem to a sparse linear system. PHiCap uses the efficient preconditioning techniques to accelerate the rate of convergence of the iterative method used to solve the sparse linear system. Experiments on benchmark problems in both uniform and multi-layer dielectric show that PHiCap is more efficient than FastCap and HiCap. Since the transformation step is accurate, the accuracy of PHiCap is the same as that of HiCap. It is worth noting that the dense-to-sparse transformation used in PHiCap is applicable to multipole-based methods as well, where the linear system can be represented by a block matrix.

We have also proposed an iteration-free method called RedCap for capacitance extraction. Starting from the sparse linear system of PHiCap, RedCap reorders the wavelet basis, so that the sparse linear system is approximated by its incomplete factorizations without much loss of accuracy. Then, a reduced linear system is identified and solved for the total charges of each conductor surface using forward and backward substitution. The capacitances are computed directly from the solution of the reduced linear system. To our knowledge, this is the first capacitance extraction algorithm based on BEM that reduces the problem size from $O(n)$ to $O(m)$, and solves the reduced problem using direct method. Numerical experiments show that

the RedCap algorithm is up to 100 times faster than FastCap [4] and is up to 4 times faster than PHiCap [16]. The error with respect to FastCap is within 2% for the experiments we have done.

We presented a theory based on the sensitivity analysis for linear systems that clarifies the relationship between the error of each entry of the capacitance matrix and the error of the potential coefficients. We proposed a technique that enhances the accuracy of user selected coupling or self-capacitances by increasing the accuracy of those potential coefficients which are critical. Experiments show that the accuracy for selected entries of the capacitance matrix can be improved greatly with a small increase in the overall computation time. This technique is very useful for accurate crosstalk and signal integrity analysis.

We also developed a more comprehensive error estimation procedure of the multipole approximation that consider the locations and values of the charges as well. Using this procedure, we proposed a variable order multipole method in chapter VII. This algorithm selects nodes with large unbalanced charge distribution for high order expansion, while keeps the expansion order of other nodes unchanged. The experiments show that the variable order multipole method is effective and practical in improving the overall accuracy of the capacitance matrix.

Finally, we applied the multigrid method to capacitance extraction problems. To the best of our knowledge, this is the first time multigrid has been used for the capacitance extraction problem. The method solves the linear system with fine discretization faster by using the solution of coarse discretization as a good initial solution for the linear solver. The experiments show the potential of the multigrid method.

REFERENCES

[1] E. Chiprout J. R. Phillips and D. D. Ling, "Efficient full-wave electromagnetic analysis via model-order reduction of fast integral transforms," in *Proc. of Design Automation Conference*, Las Vegas, NV, June 1996, pp. 377–382, ACM Press.

[2] P. K. Kythe, *An Introduction to Boundary Element Methods*, Boca Raton, FL, CRC Press, 1995.

[3] W. Shi, J. Liu, N. Kakani, and T. Yu, "A fast hierarchical algorithm for 3-d capacitance extraction," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 330–336, 2002.

[4] K. Nabors and J. White, "FastCap: A multipole accelerated 3-d capacitance extraction program," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 11, pp. 1447–1459, 1991.

[5] S. Kapur and D. E. Long, "IES^3: a fast integral equation solver for efficient 3-dimensional extraction," in *Proc. of International Conference on Computer-Aided Design*, San Jose, CA, Nov. 1997, pp. 448–455, IEEE Computer Society.

[6] J. R. Phillips and J. White, "A precorrected FFT method for capacitance extraction of complicated 3-d structures," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1059–1072, 1997.

[7] Y. L. Le Coz and R. B. Iverson, "A stochastic algorithm for high speed capacitance extraction in integrated circuits," *Solid State Electronis*, vol. 35, no. 7, pp. 1005–1012, 1992.

[8] S. M. Rao, T. K. Sarkar, and R. F. Harrington, "The electrostatic field of conducting bodies in multiple dielectric media," *IEEE Trans. on Microwave Theory and Techoniques*, vol. 32, pp. 1441–1448, 1984.

[9] K. Nabors and J. White, "Multipole-accelerated capacitance extraction algorithm for 3-d structures with multiple dielectrics," *IEEE Trans. on Circuits and Systems*, vol. 39, pp. 946–954, 1992.

[10] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, Cambridge, MA, MIT Press, 1988.

[11] N. Soveiko and M. Nakhla, "Efficient capacitance extraction computations in wavelet domain," *IEEE Trans. on Circuits and Systems – I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 684–701, 2000.

[12] R. Schneider M. Spasojevic and P. L. Levin, "On the creation of sparse boundary element matrices for two dimensional electrostatics problems using the orthogonal Haar wavelet," *IEEE Trans. on Dielectric and Electrical Insulation*, vol. 4, no. 6, pp. 249–258, 1997.

[13] M. Spasojevic P. L. Levin and R. Schneider, "Creation of sparse boundary element matrices for 2-d and axi-symmetric electrostatics problems using the bi-orthogonal Haar wavelet," *IEEE Trans. on Dielectric and Electrical Insulation*, vol. 5, no. 4, pp. 469–484, 1998.

[14] Francis X. Canning and Kevin Rogovin, "Fast direct solution of standard moment-method matrices," *IEEE Trans. on Antennas and Propagation Magazine*, vol. 40, no. 3, pp. 15–26, 1998.

[15] D. Gope and V. Jandhyala, "An iteration-free fast multilevel solver for dense

method of moment systems," in *Proc. of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, Boston, MA, Oct. 2001, pp. 177–180.

[16] S. Yan, V. Sarin, and W. Shi, "Sparse transformations and preconditioners for 3-d capacitance extraction," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1420–1426, 2002.

[17] S. Yan, V. Sarin, and W. Shi, "Fast capacitance extraction using inexact factorization," in *Proc. of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, Portland, OR, USA, Oct., 2004, pp. 285–288.

[18] C. E. Leiserson T. H. Corman and R. L. Rivest, *Introduction to Algorithms*, Cambridge, MA, MIT Press, 2nd edition, 1990.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, MD, Johns Hopkins University Press, 2nd edition, 1989.

[20] Z. Zhu, B. Song, and J. White, "Algorithms in FastImp: a fast and wideband impedance extraction program for complicated 3-d geometries," in *Proc. of Design Automation Conference*, Anaheim, CA, June 2003, pp. 712–717, ACM Press.

[21] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Philadelphia, PA, SIAM, 2nd edition, 2003.

[22] S. Kapur and D. E. Long, "Large-scale capacitance calculation," *Proc. of Design Automation Conference*, Anaheim, CA, June 2000, pp. 744–749, ACM Press.

[23] J. Tausch and J. White, "A multiscale method for fast capacitance extraction," in *Proc. of Design Automation Conference*, New Orleans, LA, June 1999, pp. 537–542, ACM Press.

[24] R. Jiang, Y. Chang, and C. C. Chen, "Iccap: a linear time sparse transformation and reordering algorithm for 3d bem capacitance extraction," in *Proc. of Design Automation Conference*, San Diego, CA, June 2005, pp. 163–166, ACM Press.

[25] B. E. Stine, D. S. Boning, J. E. Chung, L. Camilletti, F. Kruppa and et al., "The physical and electrical effect of metal-fill patterning practices for oxide chemical-mechanical polishing process," *IEEE Trans. on Electronic Devices*, vol. 45, no. 3, pp. 665–679, 1998.

[26] F. Charlet O. Cueto and A. Farcy, "An efficient algorithm for 3d interconnect capacitance extraction considering floating conductors," in *Proc. of the International Conferences on Simulation of Semiconductor Processes and Devices*, Kobe, Japan, pp. 98–101, 2000.

[27] L. Greengard, *The Rapid Evaluation of Potential Fields in Particle Systems*, Cambridge, MA, MIT Press, 1988.

[28] J. G. Korvink M. Bachtold, M. Emmenegger and H. Baltes, "An error indicator and automatic adaptive meshing for electrostatic boundary element simulations," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 12, pp. 1439–1446, 1997.

[29] V. E. Henson W. L. Briggs and S. McCormick, *A Multigrid Tutorial*, Philadelphia, PA, SIAM, 2nd edition, 2000.

VITA

Shu Yan was born in Lanzhou, Gansu province, China. She joined Tsinghua University, Beijing, China in 1994 and received the B.S. and M.S. degrees in electrical engineering in June 1998 and January 2001, respectively. She then joined the Department of Electrical and Computer Engineering of Texas A&M University, College Station, Texas and graduated with a Ph.D. Degree in December 2005. Her research has been focused on efficient numerical methods for parasitic extraction and modeling. She worked in the IBM research labs at Austin, Texas and Watson, New York in the summers of 2004 and 2005, respectively. Now she is working with Freescale Semiconductor, Inc. in Austin, Texas. She can be reached by email at babayu@gmail.com, or by mail care of Dr. Weiping Shi, Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas 77843.

The typist for this thesis was Shu Yan.