Efficient Integral Equation Based Algorithms for Parasitic Extraction of Interconnects with Smooth or Rough Surface

by

Zhenhai Zhu

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

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2004

Author . . . .
Department of Electrical Engineering and Computer Science
August 30, 2004

Certified by . . . .
Jacob K. White
Professor
Thesis Supervisor

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

Efficient Integral Equation Based Algorithms for Parasitic Extraction of Interconnects with Smooth or Rough Surface

by

Zhenhai Zhu

Abstract

This thesis describes a few efficient parasitic extraction algorithms based on integral equation methods. It has two parts. Part one describes the algorithms used in FastImp, a program for accurate analysis of wide-band electromagnetic effects in very complicated geometries of conductors. The program is based on a recently developed surface integral formulation and a Pre-corrected FFT accelerated iterative method, but includes a new piecewise quadrature panel integration scheme, a new scaling and preconditioning technique as well as a generalized grid interpolation and projection strategy. Computational results are given on a variety of integrated circuit interconnect structures to demonstrate that FastImp is robust and can accurately analyze very complicated geometries of conductors. Part two describes an efficient Stochastic Integral Equation (SIE) Method for computing the mean value and variance of the capacitance of interconnects with random surface roughness in $O(Nlog^2(N))$ time. An ensemble average Green's function is used to account for the surface roughness. A second-order correction scheme is used to improve the accuracy. A sparsification technique based on the Hierarchical Matrix method is proposed to significantly reduce the computational cost. The SIE method avoids the time-consuming Monte Carlo simulations and the discretization of rough surfaces. Numerical experiments show that the results of the new method agree very well with those of Monte Carlo simulations.

Thesis Supervisor: Jacob K. White
Title: Professor

Acknowledgments

First and primary thanks must go to Prof. Jacob White, who has given me many advices and strong support. He showed me what good research is all about. I will try my very best for the rest of my life to match up to that standard.

I would also like to thank Professor Luca Daniel and Duane Boning who are willing to act as readers for the dissertation. Over my tenure at MIT, Luca has given me many useful advices, technical and non-technical. Many of my presentation tricks are from Luca.

A number of people have contributed to this dissertation work.

Prof. Alper Demir (Koc University, Turkey) has collaborated very closely with me on the development of Stochastic Integral Equation Method. His advice, encouragement and guidance are extremely valuable to the development of the second part of this dissertation work. I feel truly privileged to work with Alper, a two-time ICCAD best paper winner and a Guillemin-Cauer Award winner.

Prof. Alan Edelman (MIT) has taught me two math classes, Numerical Linear Algebra (18.335) and Random Matrix Theory (18.996). Coincidentally these are the first and the last class I took at MIT. Without the skills and techniques I have learned from Alan, it is almost impossible for me to carry out the research reported in this dissertation.

Dr. Michael Tsuk (Ansoft) was the first people who brought to me the rough surface modeling problem. I had an enjoyable three-month summer intern at Compaq Alpha Server Platform Group under Michael's supervision.

Dr. Bjarne Buchmann developed a sparse matrix view of the Pre-corrected FFT method and wrote an early version of pfft code. These become the foundation for the code pfft++.

Dr. Ben Song wrote the interface for FFTW code in pfft++ and has helped me debug the pfft++ code. Ben also made very helpful suggestions to the panel integration and the scaling technique used in the impedance extraction code, FastImp.

I had many interesting discussions with Prof. Jingfang Huang when he was a postdoc at MIT in 2001 and enjoyed very much a trip to his research group at University of North Carolina at Chapel Hill in 2002.

I paid a visit to Intel's Technology CAD division (San Clara, CA) every time I went to

ICCAD conference at San Jose, CA. My interactions with Dr. Youngchul Park, Dr. Dan Jiao and other group members there are very useful to check if I am working on important problems and if my solutions are practical.

I should also thank Dr. Abe Elfadel (IBM T.J.Watson Research Center). We had interesting discussions on interconnect modeling and rough surface problem when he invited me over to T.J.Watson center to give a talk.

I want to thank Dr. Wolfgang Hackbusch and his students Dr. S. Borm and Dr. L.Grasedyck for developing the the Hierarchical Matrix method and making the implementation code HLib freely available. Some of the numerical experiments in part two of this dissertation were carried out based on the code HLib.

Thanks should also go to my group mates, Shihhsien Kuo and Jaydeep Bardhan (for applying pfft++ in bio-molecular simulation and optimization), David Willis (for extending pfft++ for higher-order basis), Carlos Pinto Coelho, Dimitry Vasilyev and Junghoon Lee (for discussions on Model-Order Reduction), Joe Kanapka (for discussions on sparsification techniques), Tom Klemas, Xin Hu and Anne Vithayathil. I really enjoyed my stay at Computational Prototyping Group at MIT.

And last but certainly not the least, I want to thank my family. I am grateful to my wife Jing Wang for her love, patience and constant support through thick and thin. I have taken a unnecessarily long journey to where I am now. Without Jing, this journey would have been unbearable. Having left China for 7 years by now, I recognize that it must have been very hard for my parents. Still they have always encouraged and supported me. With my own first born expecting in just a couple of months, I can appreciate my parents' love more and more everyday. Thank you Mom and Dad!

Contents

List of Figures

15

List of Tables

18

Chapter 1

Introduction

1.1 Motivation

The unwanted electromagnetic effects, or parasitic effects, are ubiquitous in electronic systems. They arise along with intentional electromagnetic effects and have caused concerns from the early main frame super-computer era [87] to the more recent personal computer age [21]. These unwanted effects are referred to as either electromagnetic interference (EMI) [25] or the problem of electromagnetic compatibility (EMC) [76]. In today's mixed signal designs, sensitive analog circuits and rapidly switching digital logic are typically collocated on a single integrated circuit. The parasitic effects, such as coupling, can create problems that are very difficult to find and eliminate. The difficulty is that these coupling problems are often caused by simultaneous interactions between a large number of conductors.

The goal of parasitic extraction is to model these interactions, more specifically, to determine the relation between the currents and voltages at the terminals of a set of conductors. This relation is usually characterized with circuit parameters or parasitic parameters, such as an impedance matrix [50]. For a n terminal-pair problem in the sinusoidal steady-state at frequency ω, the impedance matrix $Z(\omega) \in C^{n \times n}$ satisfies

$$\bar{I}(\omega) = [Z(\omega)]^{-1} \bar{V}(\omega), \tag{1.1}$$

where $\bar{I}(\omega), \bar{V}(\omega) \in C^n$ are vectors of terminal current and voltage, respectively [23]. From (1.1), we see that if we let $\bar{V}(\omega) = \bar{e}_i$, where $\bar{e}_i$ is the unit vector with the i-th entry being one and all other entries being zero, the calculated terminal current vector $\bar{I}_i(\omega)$ is just the i-th column of matrix $[Z(\omega)]^{-1}$. Therefore, we have

$$Z(\omega) = \left[\begin{array}{cccc} \bar{I}_1(\omega) & \bar{I}_2(\omega) & \cdots & \bar{I}_n(\omega) \end{array} \right]^{-1}. \qquad (1.2)$$

Computing terminal current is accomplished by the electromagnetic analysis of the set of conductors. This approach, however, only gives the frequency-dependent impedance at specified frequencies. In order to simulate parasitic effects in circuits with nonlinear devices, the so-called model-order reduction (MOR) techniques [31] are commonly used to find the time-domain compact circuit model for the set of conductors. To ensure the accuracy of the compact circuit model, the MOR techniques usually need the impedance matrix across a wide frequency range, typically from zero frequency to at least tens of giga Hertz (GHz). Hence there has been renewed emphasis on developing efficient parasitic extraction tools capable of wide-band electromagnetic analysis of very complicated 3D geometries of conductors.

In the area of electromagnetic analysis of complicated geometries of interconnect, most of the recently developed programs have been based on combining discretized integral formulations with accelerated iterative methods [71, 49, 80, 52, 2, 79]. Though these programs and techniques have been very effective, there is still no accelerated integral equation program capable of solving full Maxwell's equations in general 3D structures with lossy conductors which is accurate from zero frequency to microwave frequencies.

In addition, the 3D structures are typically modeled only as having smooth surfaces in the existing programs and techniques [71, 49, 52, 53, 2, 80, 115]. However, many of the fabrication processes used to generate both on- and off-chip interconnect will produce conductors with surface variations. Though these variations may not be truly random, they can often be accurately modeled by assuming random surface roughness with an appropriate spatial correlation. Experiments indicate that interconnect can easily have topological variations with peak to valley distances larger than five microns [97, 14]. Measurements

indicate that this surface roughness can increase high frequency resistance by as much as a factor of three[97], and analytical treatments of the surface roughness problem [69, 43] correlates well with these measurements. It has also been shown that capacitance is significantly increased by the surface roughness [112, 75].

To the best of our knowledge, very little work on numerical techniques specifically designed for 3D interconnects and packages with rough surfaces has been reported in the literature. Although it is possible to use existing programs to analyze conductors with rough surfaces, such approaches are slow for two reasons. First, the details in the random profile of rough surfaces requires a very fine discretization, and second, an ensemble of analyses must be performed to estimate the mean and variance of the extracted parameters. A straightforward approach to calculate the mean and variance is the Monte Carlo process. An ensemble of surface realizations are generated using a height probability distribution and the height spectral density. The governing equations are then solved for each realization. Even though the fast methods in [71, 49, 52, 53, 2, 80, 115] are quite efficient, the Monte Carlo approach is still rather inefficient because it typically involves many thousands of solves to get statistically reasonable mean and variance values.

1.2 Dissertation outline

This dissertation has two self-contained parts. The first part presents a collection of numerical techniques used in a wideband impedance extraction program called FastImp. The second part concerns a generic stochastic integral equation (SIE) method for modeling rough surface effects.

The main contributions in this dissertation are listed as the following:

1. A general and extensible fast integral equation solver, pfft++

   This solver is based on a generalized grid interpolation and projection strategy. It can handle rather general integral operators commonly seen in boundary element method and has been applied in bio-molecular simulation [59] and optimization [3] and aero-dynamics simulation [108], in addition to the computational electromagnetics reported in this dissertation. The code pfft++ is available at www.rle.mit.edu/cpg/research_codes.htm.

2. A wideband impedance extraction program, FastImp

   The program is based on a recently developed surface integral formulation, but includes a new piecewise quadrature panel integration scheme, a new scaling technique and a local preconditioning technique. The piecewise quadrature scheme fixes a low-frequency problem in the existing surface integral formulation and makes it widebanded. The scaling technique makes the convergence behavior of the system matrix from the discretized surface integral formulation almost independent of conductor feature size. The code FastImp is available at www.rle.mit.edu/cpg/research_codes.htm.

3. A fast stochastic integral equation (SIE) method

   The SIE method has a few intricate steps, including a second-order correction scheme and a combined sampling process based on hierarchical matrix method [33, 34, 5]. Though only the capacitance extraction is used in this dissertation to demonstrate that the SIE method can compute the mean value and variance of the capacitance of 3D interconnects in $O(Nlog^2(N))$ time, it is expected that the SIE method can also be utilized in the impedance extraction of 3D interconnects to model the rough surface effect.

This dissertation is organized in the following way.

In part one, after a literature review, we first derive the surface integral formulation (chapter 3). We then show how the piecewise quadrature scheme improves the accuracy of panel integration and that it solves the low frequency problem in [105] (chapter 4). A simple scaling technique and a local preconditioner are used in chapter 5 to improve the accuracy and memory efficiency of the surface integral formulation. In chapter 6, we explain the extensions needed to use the Pre-corrected FFT (pFFT) algorithm to accelerate the complicated integral operators in our surface formulation. A stand-alone general and extensible fast integral equation solver, pfft++, has been developed based on ideas in chapter 6. Combining pfft++ with ideas presented in chapter 4 and 5, we have developed a fast impedance extraction program, FastImp. Numerical experiments are used in chapter 7 to demonstrate the accuracy, speed and capacity of pfft++ and FastImp.

In part two, we first review some of the existing techniques and approaches for ana-

lyzing rough surface effects. We then describe a mathematical model for characterizing random rough surfaces in chapter 9. We explain the basic ideas of the Stochastic Integral Equation (SIE) Method in chapter 10. In chapter 11, we present a few ideas to efficiently compute the most time-consuming matrices used in SIE method and demonstrate that the computational complexity of SIE is $O(Nlog^2(N))$. The outline of the algorithm is given in chapter 12 and finally numerical experiments are used in chapter 13 to verify the accuracy and speed of fast SIE method.

Part I

Wideband Impedance Extraction of Interconnects With Smooth Surfaces

Chapter 2

Overview

Many integral formulations have been developed and can be generally categorized into four kinds according to the state variables used in these formulations.

1. Formulations using the field variables E and H have been used for decades to solve the radiation and scattering problems [37, 104, 82, 38] as well as eddy current problems [58, 86]. The well-known formulations include the electric field integral equation (EFIE) and magnetic field integral equation (MFIE) [13, 104], which are also known as Stratton-Chu's formulation [94], as well as the so-called PMCHW formulation [57, 110, 66].

2. Formulations using the current and charge as state variables, such as the mixed potential integral equation (MPIE) formulation [37, 70, 11, 10, 61].

3. Formulations using vector and scalar potentials as state variables are commonly used for solving eddy current problems [72].

4. Formulations using virtual sources, such as virtual current or charge, are also commonly used for solving eddy current problems [63, 47].

It is well-known that the EFIE formulation is not guaranteed to produce a unique solution at interior resonant frequencies for closed structures [104, 16]. Many remedies have been proposed [77]. But there still remain many unsolved problems.

The MPIE formulation has been extensively used for the analysis of microstrip structures [70, 11, 10, 61] and for arbitrary shaped conductors with only surface current [83]. It was recognized in [64] that MPIE has accuracy problem at low frequencies. The so-called loop/star and loop/tree basis functions were used to overcome this low-frequency problem [64, 111, 15, 85]. The MPIE formulation has also been used for the analysis of interconnects in VLSI or analog circuits. In this case, it is also known as the Partial Equivalent Element Circuit (PEEC) method [40]. Results of the Magneto-Quasi-Static (MQS) analysis in [49] and the Electro-Magneto-Quasi-Static (EMQS) analysis in [48] have clearly demonstrated that the PEEC method can produce accurate results across a wide frequency range, from zero to hundreds of giga hertz. However, unlike the microstrip structures, which are usually approximated by zero-thickness perfect or lossy conductors [70, 11, 10, 61], typical interconnect structures are lossy conductors with finite thickness. Because of the skin effect, analyzing them involves a frequency-dependent discretization of the interior of conductors and the substrate ground. At high frequencies, this kind of discretization usually renders the number of piecewise constant basis functions (also called filaments) to be prohibitively large [68, 100, 17, 18]. Recently, an entire-domain basis scheme has shown some promise to remedy the situation [20], but we have yet to see that it will eventually lead to a wideband fast Maxwell's equation solver for general 3D structures.

The motivation behind this dissertation is to find a numerically stable surface integral formulation, as such formulations avoid a frequency-dependent discretization of the interior of conductors and the substrate. The formulation should be capable of wideband analysis and it should also be easily accelerated by the well-established techniques, such as the fast multipole method [30, 29] and the pre-corrected FFT algorithm [80].

One recently developed surface integral formulation has shown promise [103, 105], but was plagued with numerical difficulties of poorly understood origin. It is shown in chapter 4 that one of that formulation's difficulties was related to inaccuracy in the approach used to evaluate integrals over discretization panels, and a more accurate approach based on an adapted piecewise quadrature scheme was proposed. In chapter 5, it is also shown that the condition number of the system matrix could be very large if the feature size of the structure is small, and a scaling technique is proposed to reduce the condition number. In

addition, a different preconditioner than the one used in [103] is proposed to improve the memory efficiency. With these issues being resolved, the formulation is indeed valid across wide frequency range and for all feature sizes. Now the formulation is acceleration-ready.

The Fast Multiple Method (FMM) [30, 29] has been used successfully in many applications, such as electrostatic analysis in FastCap [71] and others [2], magneto-quasi-static analysis in FastHenry [49], and fullwave analysis in the Fast Illinois Solver Code [93]. Though the algorithm is rather general, its most efficient variants are kernel-dependent. On the other hand, the pre-corrected FFT (pFFT) algorithm [78], which has been successfully used in many applications [80, 106, 59], is nearly kernel-independent but can be quite in-efficient for highly inhomogeneous problems. Since our surface integral formulation has a number of different kernels and the problem geometry is near-planar and densely packed, the pFFT algorithm seems better suited to our formulation. In addition, as a by-product of our work, we also developed a flexible and stand-alone fast integral equation solver using extensions of several of the sparse matrix based ideas in [9].

Combining the fast solver with the improved surface integral formulation, we have developed a fast impedance extraction program, FastImp. Experiments using several large examples show that FastImp can perform full 3D electromagnetic analysis of interconnect structures with millions of unknowns from zero frequency to hundreds of giga hertz.

Chapter 3

Derivation of the Surface Integral Formulation

Figure 3-1 is a general description of the 3D interconnect structures embedded in an isotropic and homogeneous medium. We assume that each conductor $V_i, i = 1, 2, ..., n$, is piecewise homogeneous and the homogeneous medium region is always denoted by V_0.

We will derive the surface integral formulation from a different viewpoint than the one used in [105]. This way, it is very easy to see its connections to the MPIE formulation and the EFIE formulation.
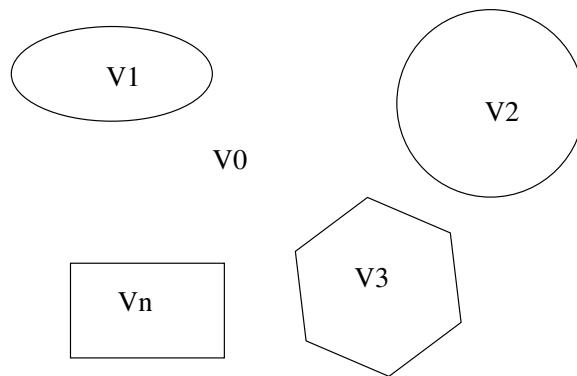
Figure 3-1: A general description of the 3D interconnect structures embedded in homogeneous medium

3.1 Governing equations

For domains of constant permittivity and permeability, the independent Maxwell's equations in time-harmonic form are [96]

$$\nabla \times \vec{E} = -j\omega\mu\vec{H} \tag{3.1}$$

$$\nabla \times \vec{H} = \vec{J} + j\omega\varepsilon\vec{E} \tag{3.2}$$

$$\nabla \cdot \vec{J} = -j\omega\rho \tag{3.3}$$

where $\vec{E}$ is the electric field, $\vec{H}$ is the magnetic field, $\vec{J}$ is the volume current density, ρ is the net charge density, and μ and ε are the permeability and permittivity, respectively. The constitutive equation for conductors is

$$\vec{J} = \sigma\vec{E}, \tag{3.4}$$

where σ is the conductivity. Equations (3.1) and (3.2) imply

$$\nabla \times \nabla \times \vec{E} - \omega^2\varepsilon\mu\vec{E} = -j\omega\mu\vec{J}. \tag{3.5}$$

Obviously equations (3.1)-(3.4) are equivalent to equations (3.1) and (3.3)-(3.5). In view of (3.2) and (3.4), we have

$$\nabla \cdot \nabla \times \vec{H} = (\sigma + j\omega\varepsilon)\nabla \cdot \vec{E} = 0,$$

where we have assumed homogeneity of σ and ε inside each conductor. Thus

$$\nabla \cdot \vec{E}(\vec{r}) = 0, \ \vec{r} \in V_i \tag{3.6}$$

where $\vec{r}$ is a point in the interior of conductor V_i. This means the net charge inside a homogeneous conductor is zero [81]. Hence equation (3.5) can be reduced to

$$(\nabla^2 + \omega^2\varepsilon\mu)\vec{E}(\vec{r}) = j\omega\mu\vec{J}(\vec{r}), \ \vec{r} \in V_i. \tag{3.7}$$

It should be noted that the permitivitty and permeability inside a conductor are assumed to be the same as those of the free space [36].

Equations (3.1), (3.4), (3.6) and (3.7) are the governing equations inside each conductor V_i, and equations (3.1)-(3.4) are the governing equations in the homogeneous medium V_0.

3.2 Boundary conditions

The surface of each conductor could be divided into two parts: contact surfaces and non-contact surfaces, as shown in figure 3-2. The contact is an artificially exposed surface. It is created primarily because we want to use the divide-and-conquer strategy to separate a block of 3D interconnect from other parts within a large chip. Here we use voltage source connected to the contacts as excitation and compute current through the contacts, from which the impedance can be easily calculated, as shown in (1.2). In this case, it is reasonable to assume that the total current through voltage source and the corresponding contacts is equal. Hence there should be no charge accumulation on the contacts. So equation (3.6) also holds true on the contacts.

Because of the nature of the commonly used strategy to decompose a large chip into many smaller blocks, the conductors connected to these contacts are usually long and thin signal lines. Hence it is reasonable to assume that the current goes into these contacts does not have the transversal components, i.e., $\hat{t} \cdot \vec{J} = 0$, where $\hat{t}$ is the unit tangential vector on the contacts. Using the constitutive equation in (3.4) implies

$$\hat{t}(\vec{r}) \cdot \vec{E}(\vec{r}) = \vec{E}_t(\vec{r}) = 0, \tag{3.8}$$

if $\vec{r}$ is on a contact. Equations (3.6) and (3.8) imply that if $\vec{r}$ is on a contact,

$$\hat{n}(\vec{r}) \cdot \frac{\partial \vec{E}(\vec{r})}{\partial n(\vec{r})} = \frac{\partial E_n(\vec{r})}{\partial n(\vec{r})} = 0, \tag{3.9}$$

where $\hat{n}(\vec{r})$ is the normal unit vector on the contact. Since equation (3.8) and (3.9) also imply (3.6), we will use (3.6) on the non-contacts only to avoid redundancy. On the other hand, since charge on a non-contact surface is not necessarily zero, in view of (3.3) and

31
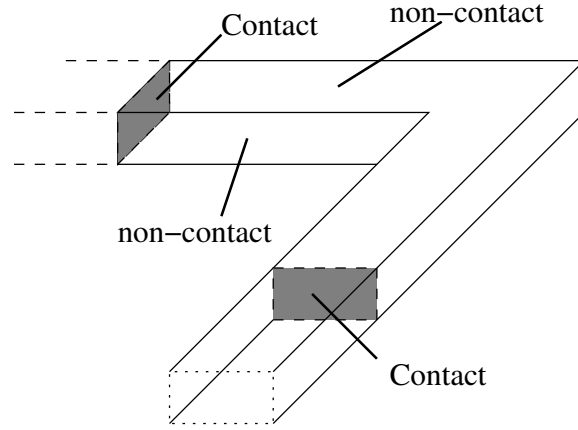
Figure 3-2: The surface of a 3D interconnect conductor

(3.4), the boundary condition when $\vec{r}$ is on a non-contact surface becomes [96]

$$\hat{n}(\vec{r}) \cdot \vec{E}(\vec{r}) = E_n(\vec{r}) = \frac{j\omega\rho(\vec{r})}{\sigma}, \tag{3.10}$$

where ρ is the surface charge density since the net charge inside a conductor is zero [81]. It should be noted that the position vector $\vec{r}$ in $\vec{E}(\vec{r})$ of (3.8) and (3.10) and in $\frac{\partial \vec{E}(\vec{r})}{\partial n}$ of (3.9) is only on the inner side of the conductor surfaces. It should also be noted that through out this paper $\vec{J}$ is always the actual volume current density.

3.3 Surface integral representation

Green's second identity can be used to construct a surface integral representation of the solution to equation (3.7) [13]

$$T\vec{E}(\vec{r}) = \int_{S_i} dS' \left(G_0(\vec{r},\vec{r'}) \frac{\partial \vec{E}(\vec{r'})}{\partial n(\vec{r'})} - \frac{\partial G_0(\vec{r},\vec{r'})}{\partial n(\vec{r'})} \vec{E}(\vec{r'}) \right) - j\omega\mu \int_{V_i} dV' G_0(\vec{r},\vec{r'}) \vec{J}(\vec{r'}) \tag{3.11}$$

where

$$G_0(\vec{r},\vec{r'}) = \frac{e^{jk_0|\vec{r}-\vec{r'}|}}{4\pi|\vec{r}-\vec{r'}|}, \quad k_0 = \omega\sqrt{\varepsilon\mu} , \tag{3.12}$$

$$T = \begin{cases} 1 & \text{if } \vec{r} \in V_i \\ 1/2 & \text{if } \vec{r} \in S_i \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

and S_i is the surface of conductor V_i. When $\vec{r} \in S_i$, the surface integral in (3.11) should be the principal value integral [35]. From (3.11) and (3.13) one can see that the integral representation in the right-hand side of (3.11) generates zero field when $\vec{r}$ is outside of V_i. If we write equation (3.11) for each conductor separately but let $\vec{r}$ be fixed on the inner side of the surface of a particular conductor V_k, and then sum these equations, we obtain

$$\begin{aligned} \frac{1}{2}\vec{E}(\vec{r}) = & \int_S dS' \left(G_0(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_0(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right) \\ & - j\omega\mu \int_V dV' G_0(\vec{r},\vec{r}') \vec{J}(\vec{r}'), \ \vec{r} \in S_k \end{aligned} \tag{3.14}$$

where $k = 1,2,...,n$, S is the union of all conductor surfaces and V is the union of all conductor regions.

Substituting (3.4) into (3.7) yields,

$$\nabla^2 \vec{E}(\vec{r}) + (\omega^2 \varepsilon\mu - j\omega\mu\sigma_i)\vec{E}(\vec{r}) = 0, \ \vec{r} \in V_i \tag{3.15}$$

where σ_i is the conductivity of the conductor V_i. Again, Green's second identity yields the surface integral representation of the solution to equation (3.15)

$$\frac{1}{2}\vec{E}(\vec{r}) = \int_{S_i} dS' \left(G_1(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_1(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right), \ \vec{r} \in S_i \tag{3.16}$$

where

$$G_1(\vec{r},\vec{r}') = \frac{e^{jk_1|\vec{r}-\vec{r}'|}}{4\pi|\vec{r}-\vec{r}'|}, \ k_1 = -\sqrt{\omega^2\varepsilon\mu - j\omega\mu\sigma_i} \ . \tag{3.17}$$

Since (3.14) and (3.16) are the formal solutions to the same equation in slightly different forms, they are obviously equivalent. We use both to simplify the derivation.

So far, only the formal solutions to equation (3.7) inside each conductor have been found. To find the formal solution to the governing equations in region V_0, the homoge-

33

neous medium, we turn to the MPIE. Now each conductor is treated as a volume current source. In the standard MPIE formulation [37], the electric field everywhere, including the interior of every conductor, is

$$\vec{E}(\vec{r}) = -j\omega\vec{A} - \nabla\phi(\vec{r}) = -j\omega\mu \int_V dV' G_0(\vec{r},\vec{r}')\vec{J}(\vec{r}') - \nabla\phi(\vec{r}) \tag{3.18}$$

where

$$\phi(\vec{r}) = \int_S dS' \frac{\rho(\vec{r}')}{\varepsilon} G_0(\vec{r},\vec{r}'). \tag{3.19}$$

Notice that the volume integral term in equation (3.18) is identical to the one in equation (3.14), we could use this fact to eliminate this undesirable volume integral term. Let $\vec{r} \in S_k$ in equation (3.18) and subtract it from equation (3.14), we then obtain

$$-\frac{1}{2}\vec{E}(\vec{r}) = \int_S dS' \left(G_0(\vec{r},\vec{r}')\frac{\partial\vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_0(\vec{r},\vec{r}')}{\partial n(\vec{r}')}\vec{E}(\vec{r}') \right) + \nabla\phi(\vec{r}),\ \vec{r} \in S_k \tag{3.20}$$

where $k = 1, 2, ..., n$. The integral representation (3.20) is no longer the formal solution to equation (3.7), hence it is no longer equivalent to (3.16). Now we have found the surface integral representation of the formal solutions to the governing equations inside conductors and homogeneous medium. It should be noted that the surface integrals in (3.14), (3.16) and (3.20) are all principal value integrals.

Unlike the standard MPIE, the Lorentz gauge $\nabla \cdot \vec{A} + j\omega\varepsilon\mu\phi = 0$ is not explicitly enforced in our formulation because it is implied by equations (3.18), (3.19) and (3.6), which are explicitly enforced. Now it is clear that had equation (3.5), instead of equations (3.6) and (3.7), been used as the governing equations, we would have to enforce Lorentz gauge. This would introduce the vector potential $\vec{A}$ and ultimately a volume integral term into our formulation. Since this volume term is different from the ones in (3.14) and (3.18), it may not be possible to eliminate this undesirable term using the same trick used to derive (3.20).

It is worth mentioning that equations (3.11) and (3.16) are very similar to the standard EFIE formulation [104]. There are a few equivalent forms of EFIE, the one closest to

34

equation (3.16) is

$$\frac{1}{2}\vec{E}(\vec{r}) = \int_{S_i} dS' \left(G_1(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_1(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right)$$
$$+ \int_{S_i} dS' \left(\hat{n}(\vec{r}') G_1(\vec{r},\vec{r}') (\nabla' \cdot \vec{E}(\vec{r}')) \right), \quad \vec{r} \in S_i. \tag{3.21}$$

And the EFIE equation closest to equation (3.11) is equation (3.21) with the addition of a volume integral term like the one in equation (3.11), with G_1 being replaced by G_0.

The standard EFIE is derived from the vector Helmholtz equation (3.5) using the Green's second identity in vector form, with equation (3.6) not explicitly enforced. However, as discussed before, equation (3.6) must be enforced in our formulation. This is why we have chosen equation (3.16) rather than equation (3.21), the standard EFIE.

3.4 Surface formulation

We follow the convention in the PEEC model, using the difference between ϕ on two contacts of the same conductor as the voltage excitation term [49, 48]. In light of this, we introduce one last equation, equation (3.29), into our formulation.

The boundary conditions and the surface integral representation of the solution to the Maxwell's equations are summarized as the following:

$$\frac{1}{2}\vec{E}(\vec{r}) = \int_{S_i} dS' \left(G_1(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_1(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right), \quad \vec{r} \in S_i \tag{3.22}$$

$$-\frac{1}{2}\vec{E}(\vec{r}) = \int_S dS' \left(G_0(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_0(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right) + \nabla\phi(\vec{r}), \quad \vec{r} \in S_{nc} \tag{3.23}$$

$$\phi(\vec{r}) = \int_S dS' \frac{\rho(\vec{r}')}{\varepsilon} G_0(\vec{r},\vec{r}'), \quad \vec{r} \in S \tag{3.24}$$

$$\nabla \cdot \vec{E}(\vec{r}) = 0, \quad \vec{r} \in S_{nc} \tag{3.25}$$

$$E_n(\vec{r}) = \frac{j\omega\rho(\vec{r})}{\sigma}, \quad \vec{r} \in S_{nc} \tag{3.26}$$

$$\hat{t}(\vec{r}) \cdot \vec{E}(\vec{r}) = 0, \quad \vec{r} \in S_c \tag{3.27}$$

$$\frac{\partial E_n(\vec{r})}{\partial n(\vec{r})} = 0, \quad \vec{r} \in S_c \tag{3.28}$$

$$\phi(\vec{r}) = constant, \quad \vec{r} \in S_c \tag{3.29}$$

where S_{nc} and S_c are the non-contact part and the contact part of conductor surface S, respectively. There are eight scalar state variables, E_x, E_y, E_z, $\frac{\partial Ex}{\partial n}$, $\frac{\partial Ey}{\partial n}$, $\frac{\partial Ez}{\partial n}$, ϕ and ρ. All components of $\vec{E}$ and $\frac{\partial \vec{E}}{\partial n}$ are defined only on the inner side of the conductor surfaces. The potential ϕ and the surface charge density ρ are continuous across the surface, so no differentiation between the inner and the outer side of the surface is needed for ϕ and ρ. It should be noted that equation (3.25) is essentially same as (3.6) because it is enforced on the inner side of the non-contact conductor surface, which is still in the interior of the conductors.

Equations (3.22)-(3.29) form a system of nine equations which involve unknowns on conductor surfaces: three scalar equations in (3.22), three scalar equations in (3.23) complemented by two scalar equations in (3.27), one scalar equation in (3.24), one scalar equation in (3.25) complemented by (3.29), and one scalar equation in (3.26) complemented by (3.28). Since there are only eight variables and nine equations, (3.22)-(3.29) may not have a solution. In [44] it was shown that the domain truncation shown in figure 3-2 combined with artificial boundary conditions (3.27)-(3.29) insures that there is no solution. The PEEC formulation has a similar problem and so equation (3.19) was not enforced for the interior of the conductor [50]. In our formulation, we discard one of the three scalar equations in (3.23).

In the local coordinate system ($\hat{t}_1$, $\hat{t}_2$, $\hat{n}$), where $\hat{t}_1$ and $\hat{t}_2$ are two orthogonal unit vectors tangential to the surface of conductor and $\hat{n}$ is the unit vector normal to the surface of conductor, the term $\nabla \phi$ in (3.23) can be written as $\hat{t}_1 \frac{\partial \phi}{\partial t_1} + \hat{t}_2 \frac{\partial \phi}{\partial t_2} + \hat{n} \frac{\partial \phi}{\partial n}$. Using ϕ on the surface, finite-differences can be used to compute $\frac{\partial \phi}{\partial t_1}$ and $\frac{\partial \phi}{\partial t_2}$, but not $\frac{\partial \phi}{\partial n}$. In light of the above observation, we have decided to enforce equation (3.23) only along two tangential directions in the local coordinate system and not to enforce it along the normal direction.

Equation (3.23) then becomes

$$
\begin{aligned}
-\hat{t}(\vec{r}) \cdot \frac{1}{2}\vec{E}(\vec{r}) &= \hat{t}(\vec{r}) \cdot \int_S dS' \left(G_0(\vec{r},\vec{r}') \frac{\partial \vec{E}(\vec{r}')}{\partial n(\vec{r}')} - \frac{\partial G_0(\vec{r},\vec{r}')}{\partial n(\vec{r}')} \vec{E}(\vec{r}') \right) \\
&\quad + \hat{t}(\vec{r}) \cdot \nabla\phi(\vec{r}), \quad \vec{r} \in S_{nc}.
\end{aligned}
\tag{3.30}
$$

This results in a system of eight scalar equations.

In summary, the surface integral formulation consists of equations (3.22) and (3.24)-(3.30). For the EMQS analysis, k_0 in equation (3.12) becomes zero and the term $\omega^2 \varepsilon \mu$ in equation (3.17) should be dropped [39]. But the number of state variables is unchanged. For the MQS analysis, on top of above simplification, the surface charge density ρ in equation (3.26) is assumed to be zero [39]. Hence it becomes redundant and is not used as a state variable, and equation (3.24) is eliminated. Hence the total number of scalar unknowns and equations for MQS analysis becomes seven. It should be noted that the two slightly different sets of equations, equations (3.22) and (3.24)-(3.30) for EMQS and fullwave analysis, and equations (3.22) and (3.25)-(3.30) for MQS analysis, are all widebanded, as will be shown in the numerical result section. In addition, our numerical experiments show that they all produce virtually identical impedance at very low frequencies but they behave very differently at high frequencies. These are consistent with electromagnetic theory. One does not need to switch between these different sets of equations to achieve wideband behavior.

3.5 Discretization of the formulation

In order to discretize the integral equations (3.22), (3.30) and (3.24), a piecewise constant centroid collocation scheme is used. The conductor surface is discretized into many flat panels. Seven unknowns are associated with each panel: E_x, E_y, E_z, $\frac{\partial E_x}{\partial n}$, $\frac{\partial E_y}{\partial n}$, $\frac{\partial E_z}{\partial n}$ and ρ. The scalar potential ϕ is associated with the panel vertexes, and $\nabla\phi$ in (3.30) is computed using finite-difference. With this setting, equations (3.26), (3.27), (3.28), and (3.29) become simple algebraic equations. But equation (3.25) deserves more attention.

Applying the integral form of equation (3.25) to the surface of an infinitely thin small
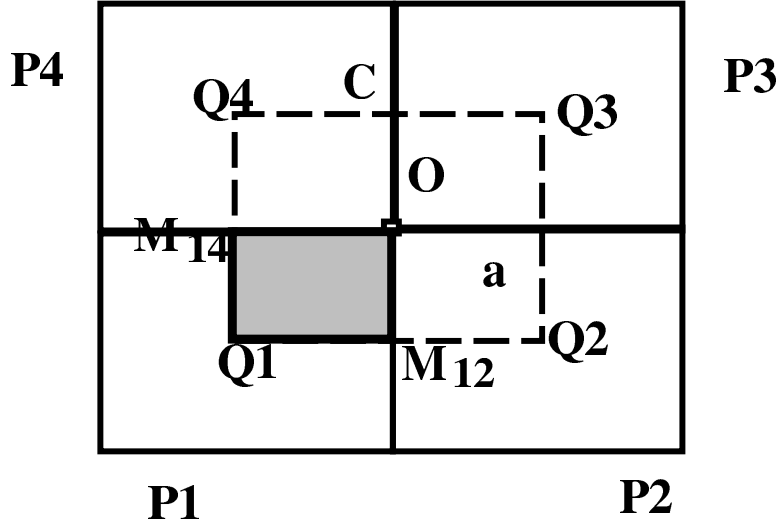
Figure 3-3: Dual panel

rectangular box beneath the conductor surface, we obtain

$$\int_C d\gamma \vec{E}_t(\gamma) \cdot (\hat{n}(\gamma) \times \hat{l}(\gamma)) - \int_a dS(\vec{r})\hat{n}(\vec{r}) \cdot \frac{\partial \vec{E}(\vec{r})}{\partial n(\vec{r})} = 0 \qquad (3.31)$$

where a is the top of the box, C is the periphery of a, $\hat{n}$ is the normal unit vector and $\hat{l}$ is the unit vector along C. Equation (3.31) is enforced on the so-called dual panel around each vertex, one dual panel $Q_1Q_2Q_3Q_4$ is shown in figure 3-3. Panel $Q_1Q_2Q_3Q_4$ is divided by the edges of regular panels into four sub-panels. The state variables $\vec{E}$ and $\frac{\partial \vec{E}}{\partial n}$ on sub-panel $OM_{12}Q_1M_{14}$, the shaded one in figure 3-3, is the same as those defined on panel P_1. The same holds true for other three sub-panels.

38

Now we can write the system matrix as

$$
\begin{bmatrix}
P_1 & 0 & 0 & D_1 & 0 & 0 & 0 & 0 & 0 \\
0 & P_1 & 0 & 0 & D_1 & 0 & 0 & 0 & 0 \\
0 & 0 & P_1 & 0 & 0 & D_1 & 0 & 0 & 0 \\
T_{1,x}P_0 & T_{1,y}P_0 & T_{1,z}P_0 & T_{1,x}D_0 & T_{1,y}D_0 & T_{1,z}D_0 & g_{11} & g_{12} & 0 \\
T_{2,x}P_0 & T_{2,y}P_0 & T_{2,z}P_0 & T_{2,x}D_0 & T_{2,y}D_0 & T_{2,z}D_0 & g_{21} & g_{21} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & 0 & P_0^1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & P_0^2 \\
-A_x & -A_y & -A_z & C_x & C_y & C_z & 0 & 0 & 0 \\
0 & 0 & 0 & N_x & N_y & N_z & 0 & 0 & \frac{-j\omega}{\sigma}I \\
0 & 0 & 0 & T_{1,x} & T_{1,y} & T_{1,z} & 0 & 0 & 0 \\
0 & 0 & 0 & T_{2,x} & T_{2,y} & T_{2,z} & 0 & 0 & 0 \\
N_x & N_y & N_z & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0
\end{bmatrix}
\begin{bmatrix}
\frac{\partial E_x}{\partial n} \\
\frac{\partial E_y}{\partial n} \\
\frac{\partial E_z}{\partial n} \\
E_x \\
E_y \\
E_z \\
\phi_{nc} \\
\phi_c \\
\rho
\end{bmatrix}
=
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \Phi_c
\end{bmatrix}
$$

$$(3.32)$$

where ϕ_{nc} and ϕ_c are the potential on the non-contacts and the contacts, respectively. The horizontal lines in the system matrix are used to mark the corresponding relation between row blocks and the equations (3.22) to (3.29). For example, the three rows in the first row block correspond to (3.22). Matrix I is the identity matrix, P_0 and P_1 are respectively the dense matrices corresponding to the single-layer integral with Green's function G_0 in (3.12) and G_1 in (3.17). The elements of matrix P_0 are

$$
P_0(i,j) = \int_{Panel_j} ds(\vec{r}') G_0(\vec{r}_i, \vec{r}') = \int_{Panel_j} ds(\vec{r}') \frac{e^{jk_0|\vec{r}_i-\vec{r}'|}}{4\pi|\vec{r}_i - \vec{r}'|}, \qquad (3.33)
$$

where $\vec{r}_i$ is the centroid of the i-th panel. Matrices P_0^1 and P_0^2 are sub-matrix blocks of P_0. The number of rows in P_0^1 and P_0^2 is the size of ϕ_{nc} and ϕ_c, respectively. D_0 and D_1 are respectively the dense matrices corresponding to the double-layer integral with Green's

function G_0 and G_1. The elements of matrix D_0 are

$$D_0(i,j) = \begin{cases} -\int_{Panel_j} ds(\vec{r}') \frac{\partial}{\partial n(\vec{r}')} \frac{e^{jk_0|\vec{r}_i-\vec{r}'|}}{4\pi|\vec{r}_i-\vec{r}'|} & i \neq j \\ \frac{1}{2} & i = j. \end{cases} \qquad (3.34)$$

Sparse matrices g_{11}, g_{12}, g_{21} and g_{22} represent the finite-difference approximation of $\nabla\phi$. Sparse matrices $T_{1,\alpha}$, $T_{2,\alpha}$ and N_α ($\alpha = x,y,z$) are the transfer matrices relating the local coordinate system (t_1, t_2 and n) to the global coordinate system (x,y and z). The nonzero elements of the sparse matrices A_x, A_y and A_z and the nonzero elements of the sparse matrices C_x, C_y and C_z are related to the dual panel discretization. And Φ_c, the known potential on the contact, is used as the excitation.

The structure of the system matrix in (3.32) can be used to show that the system is not singular even when the frequency is identically zero. It is straightforward to verify that at zero frequency the matrix block P_1 and D_1 are dense. Since the nonzero block P_1 in the first three columns are never in the same row, the first three columns are linearly independent. For the same reason, the fourth to sixth column are linearly independent from each other. Noticing that the nonzero matrix blocks in row 9 to 12 of the column 4 to 6 are never in the same row as the nonzero blocks in row 9 to 12 of the column 1 to 3, we conclude that the first six columns are linearly independent. Similarly, due to the nonzero block $-I\varepsilon$ in row 6 and 7, we can also conclude that the first eight columns are linearly independent. At zero frequency, the matrix block $-\frac{j\omega}{\sigma}I$ is zero. But because of the matrix block I in column 8, column 8 and 9 are linearly independent. Therefore, we can conclude that the system matrix is not singular even when the frequency is identically zero. This means that our surface integral formulation does not have the kind of low-frequency problem reported in [64]. Hence we do not need to use the so-called loop/star and loop/tree basis functions to discretize the formulation.

Chapter 4

Improving the accuracy of panel integration

Early experiments with the above formulation suggested that there was a low-frequency problem which was resolved using a linearization technique [105]. In this chapter we show that the formulation does not have difficulties at low frequencies, and that the early experimental results were due to inaccuracy in the approach to the panel integration, particularly the nearby interactions at low frequencies. We then propose a simple piecewise quadrature scheme to fix the problem.

4.1   Definition

After discretization, the integrals over conductor surface S or S_i are replaced by the summation of integrals over panels. These integrals are

$$I_1(\vec{r}) = \int_{P_i} dS' G(\vec{r}, \vec{r'}) = \int_{P_i} dS' \frac{e^{jk_0|\vec{r}-\vec{r'}|}}{4\pi|\vec{r}-\vec{r'}|} \tag{4.1}$$

$$I_2(\vec{r}) = \int_{P_i} dS' \frac{\partial G(\vec{r}, \vec{r'})}{\partial n(\vec{r'})} = \begin{cases} -\frac{1}{2} & \vec{r} \in P_i \\ \hat{n}(P_i) \cdot \int_{P_i} dS' \nabla_{\vec{r'}} G(\vec{r}, \vec{r'}) & otherwise \end{cases} \tag{4.2}$$

where P_i is the i-th panel, $\hat{n}(P_i)$ is the unit normal vector on the flat panel P_i, and $G(\vec{r},\vec{r'})$ is either $G_0(\vec{r},\vec{r'})$ or $G_1(\vec{r},\vec{r'})$ defined in (3.12) and (3.17). From the symmetry property of the Green's function, it follows that

$$\int_{P_i} dS' \nabla_{\vec{r'}} G(\vec{r},\vec{r'}) = -\nabla_{\vec{r}} \int_{P_i} dS' G(\vec{r},\vec{r'}) = -\nabla_{\vec{r}} I_1(\vec{r}) \tag{4.3}$$

when $\vec{r}$ is not on panel P_i. Therefore, to compute the integrals in equation (4.1) and (4.2), it is only necessary to compute $I_1(\vec{r})$ and $\frac{\partial I_1(\vec{r})}{\partial D}$, where D stands for x, y or z.

4.2   Decomposition

It is shown in [41] that any integration over a polygon is equal to the signed summation of the integration over a chosen set of triangles. The vertexes of these triangles are those of the polygon and the projection of the evaluation point onto the plane where the polygon lies, as shown in figure 4-1. To be more precise, let $f(\vec{r})$ be a general integrand, its integration over a polygon in figure 4-1 could be written as

$$\int_S d\vec{r} f(\vec{r}) = \sum_{i=1}^N s_i \int_{PV_iV_{i+1}} d\vec{r} f(\vec{r}) \tag{4.4}$$

where N is the number of vertexes, $V_{N+1} = V_1$, and $s_i = -1$ if V_iV_{i+1} is clockwise looking from the evaluation point E and $s_i = 1$ if otherwise. This idea was used in [105] to compute the integrals $I_1(\vec{r})$ and $\frac{\partial I_1(\vec{r})}{\partial D}$.

4.3   De-singularization and Reduction to 1-D integration

In a polar coordinate system, a triangle after the decomposition is shown in figure 4-2. Using the relation $R = \sqrt{r^2 + h^2}$ and $RdR = rdr$, the integrals I_1 and $\frac{\partial I_1}{\partial D}$ over this triangle could be rewritten in polar coordinates as

$$I_1 = \int_{\theta_A}^{\theta_B} d\theta \int_0^{r_1(\theta)} rdr \frac{e^{ikR}}{4\pi R}$$

42
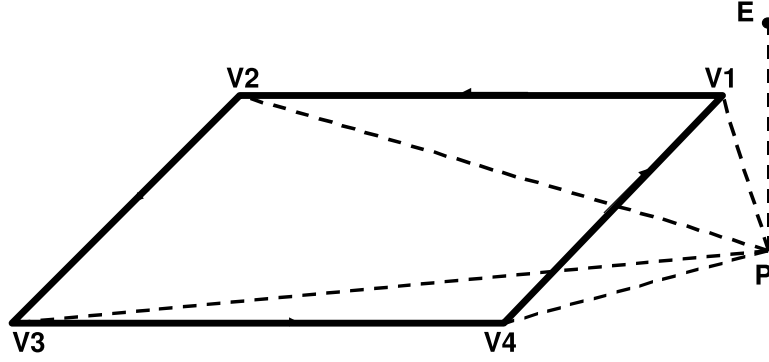
Figure 4-1: Decomposition of an integration over a polygon into several integrations over triangles. Point E is the evaluation point, point P is the projection of E on the plane. Vi $(i = 1, 2, ..., 4)$ are the vertexes of the panel.

$$= \int_{\theta_A}^{\theta_B} d\theta \int_h^{R_1(\theta)} dR \frac{e^{ikR}}{4\pi}$$

$$= \begin{cases} \int_{\theta_A}^{\theta_B} d\theta \frac{e^{ikR_1(\theta)} - e^{ikh}}{4\pi ik} & k \neq 0 \\ \int_{\theta_A}^{\theta_B} d\theta \frac{R_1(\theta) - h}{4\pi} & k = 0 \end{cases} \tag{4.5}$$

$$\frac{\partial I_1}{\partial D} = \int_{\theta_A}^{\theta_B} d\theta \left(\frac{e^{ikR_1(\theta)}}{4\pi} \frac{\partial R_1(\theta)}{\partial D} - \frac{e^{ikh}}{4\pi} \frac{\partial h}{\partial D} \right) \tag{4.6}$$

Now the singularity of the original kernels in I_1 and $\frac{\partial I_1}{\partial D}$ has been eliminated and the 2-D integrations have been reduced to 1-D integrations. The quadrature rule is used to compute the two 1-D integrations in equation (4.5) and (4.6). The shared rapid changing kernel in these two integrals is $f(\theta) = e^{ikR_1(\theta)}$, where $R_1(\theta) = \sqrt{d^2 sec^2(\theta) + h^2}$. When $d << AB$, $\theta_A \approx \frac{-\pi}{2}$ and $\theta_B \approx \frac{\pi}{2}$, and $f(\theta)$ changes rapidly over the interval. Many quadrature points must be used to achieve reasonable accuracy.

4.4 Piecewise Quadrature Scheme

A simple variable transformation and a piecewise quadrature scheme can be used to solve the above-mentioned problem. Let $x = dtan(\theta)$, it easily follows that $\frac{d\theta}{dx} = \frac{d}{r^2}$, where $r^2 = d^2 + x^2$. The rapidly changing part of I_1 and $\frac{\partial I_1}{\partial D}$ could be rewritten as

$$\int_{\theta_A}^{\theta_B} d\theta e^{ikR} = \int_{x_A}^{x_B} dx g(x), \tag{4.7}$$
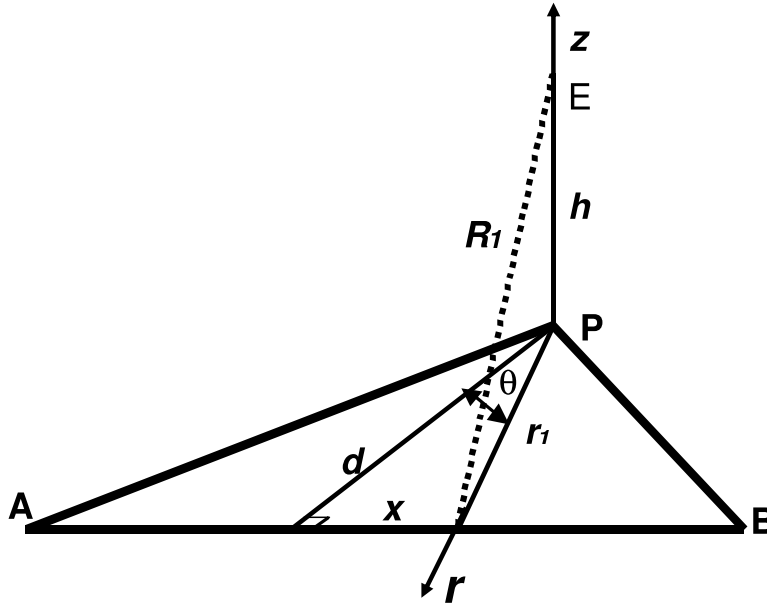
Figure 4-2: Triangle in polar coordinate system, d is the distance between point P and edge AB

where $g(x) = \frac{d}{d^2+x^2}e^{ik\sqrt{h^2+d^2+x^2}}$. The magnitude of the integrand $g(x)$ is shown in figure 4-3, where k is the wave number corresponding to the low frequency $f = 1$Hz in free space. Accurate evaluation requires many quadrature points because of the rapid variation about $x = 0$. However dividing the integration domain into two sub-domains at $x = 0$ and using a quadrature scheme for each subdomain dramatically reduces the needed number of quadrature points. The convergence behavior of the integration over the whole domain and over the two sub-domains is shown in figure 4-4. It is clear that the piecewise scheme uses many fewer quadrature points, or has higher accuracy if only a small number of quadrature points are used. As will be shown in the numerical result section, using the piecewise scheme has indeed fixed the low-frequency problem reported in [105].

Unfortunately, this is not appreciated in [105] and a small number (24) of quadrature points are used for the integration over the whole domain. Since the lower the frequency, the smaller the damping factor in complex wave number k_1 in (3.17), hence the higher the peak of the integrand $g(x)$, the formulation in [105] has a low frequency problem.
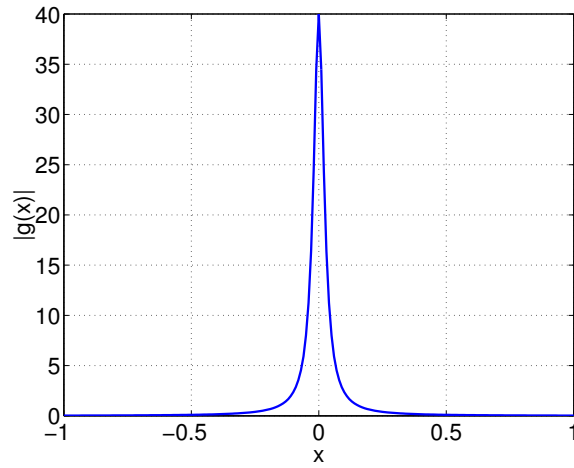
Figure 4-3: Distribution of the integrand
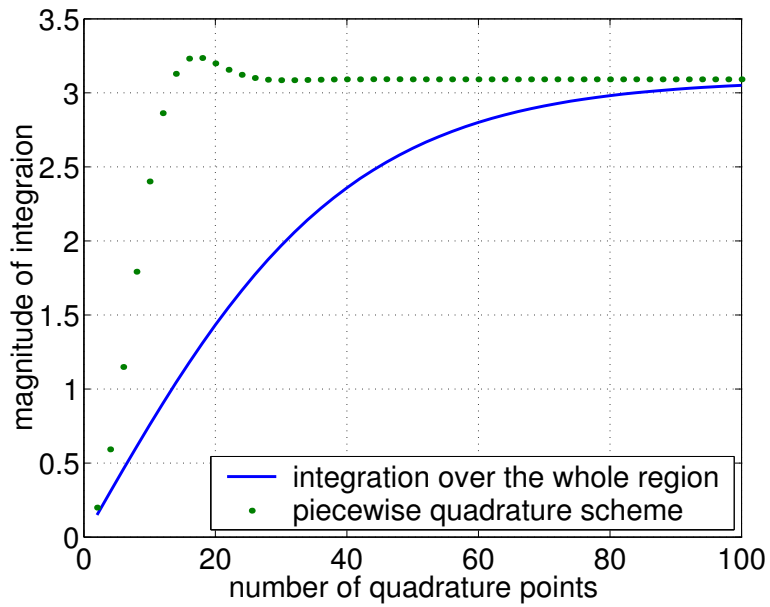
Figure 4-4: convergence behavior of different schemes

Chapter 5

Scaling and preconditioning

5.1 Scaling

The system in (3.32) will be solved iteratively, and therefore reducing the system's condition number will likely accelerate iteration convergence. As expressed in (3.32), the system has a condition number that rises rapidly as the problem geometry decreases. The difficulty is easily eliminated by scaling.

Suppose the average edge length of panels is $O(u)$, we will first estimate the scale of each matrix block in (3.32) in terms of u.

The elements of matrix P_0 are expressed in (3.33). Since $\vec{r}_i$ and $\vec{r}'$ are on the conductor surface, it is clear that $\frac{e^{jk_0|\vec{r}_i - \vec{r}'|}}{|\vec{r}_i - \vec{r}'|}$ in (3.33) is $O(1/u)$ and $ds(\vec{r}')$ is $O(u^2)$. Hence $P_0(i,j)$ is $O(u)$. And the same holds true for the elements in matrix P_1. Following a similar reasoning, $D_0(i,j)$ in (3.34) is $O(1)$, as are the elements in matrix D_1.

The dual panel discretization in (3.31) implies that the elements in matrices C_x, C_y and C_z are $O(u)$ and the elements in matrices A_x A_y and A_z are $O(u^2)$. And it is easy to check that the elements in the finite difference matrices g_{11}, g_{12}, g_{21} and g_{22} are $O(1/u)$.

Now it is clear that the scale in different matrix blocks in (3.32) could be different by many orders of magnitude if u is small. The huge difference in the scale could lead to large condition number. For example, the condition number could be as large as 10^{20} for micron feature sizes.

A simple scaling manipulation as the following can be used to remedy the situation:

scale the first three columns with $1/u$ and the seventh and eighth column with u, and then scale the sixth, seventh, eight and the last row with $1/u$, and scale the second to the last row with u. This manipulation can be written as

$$
\begin{bmatrix}
\frac{1}{u}P_1(u) & 0 & 0 & D_1(1) & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{u}P_1(u) & 0 & 0 & D_1(1) & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{u}P_1(u) & 0 & 0 & D_1(1) & 0 & 0 & 0 \\
\frac{1}{u}T_{1,x}P_0(u) & \frac{1}{u}T_{1,y}P_0(u) & \frac{1}{u}T_{1,z}P_0(u) & T_{1,x}D_0(1) & T_{1,y}D_0(1) & T_{1,z}D_0(1) & ug_{11}(\frac{1}{u}) & ug_{12}(\frac{1}{u}) & 0 \\
\frac{1}{u}T_{2,x}P_0(u) & \frac{1}{u}T_{2,y}P_0(u) & \frac{1}{u}T_{2,z}P_0(u) & T_{2,x}D_0(1) & T_{2,y}D_0(1) & T_{2,z}D_0(1) & ug_{21}(\frac{1}{u}) & ug_{22}(\frac{1}{u}) & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\varepsilon I & 0 & \frac{1}{u}P_0^1(u) \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -\varepsilon I & \frac{1}{u}P_0^2(u) \\
-\frac{A_x(u^2)}{u^2} & -\frac{A_y(u^2)}{u^2} & -\frac{A_z(u^2)}{u^2} & \frac{1}{u}C_x(u) & \frac{1}{u}C_y(u) & \frac{1}{u}C_z(u) & 0 & 0 & 0 \\
0 & 0 & 0 & N_x & N_y & N_z & 0 & 0 & \frac{-j\omega}{\sigma}I \\
0 & 0 & 0 & T_{1,x} & T_{1,y} & T_{1,z} & 0 & 0 & 0 \\
0 & 0 & 0 & T_{2,x} & T_{2,y} & T_{2,z} & 0 & 0 & 0 \\
N_x & N_y & N_z & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0
\end{bmatrix}
\quad (5.1)
$$

where the corresponding scale of each matrix block is also shown. It is easy to check that all matrix blocks are $O(1)$. Hence the new system matrix is much better conditioned.

The scaling factor u could be either the average panel size or the maximum panel size. From our experiments, either one could effectively reduce the condition number. It should be pointed out that the above scaling procedure is effective only when the panels do not vary significantly in size. Otherwise, a fixed scaling factor may not be sufficient. For example, we might have to use a separate scaling factor for each column of P_1 and P_0.

Empirical study of a simple straight wire is used here to verify the effectiveness of the scaling technique. The iterative solver GMRES [89] is used to solve the linear system matrix generated for different structure sizes. Comparison of the convergence behavior with or without the scaling is shown in figure 5-1. It is clear from this figure that the scaling has made the number of iterations almost independent of the structure size. In particular, the number of iterations has been reduced by a factor of five when the feature

48
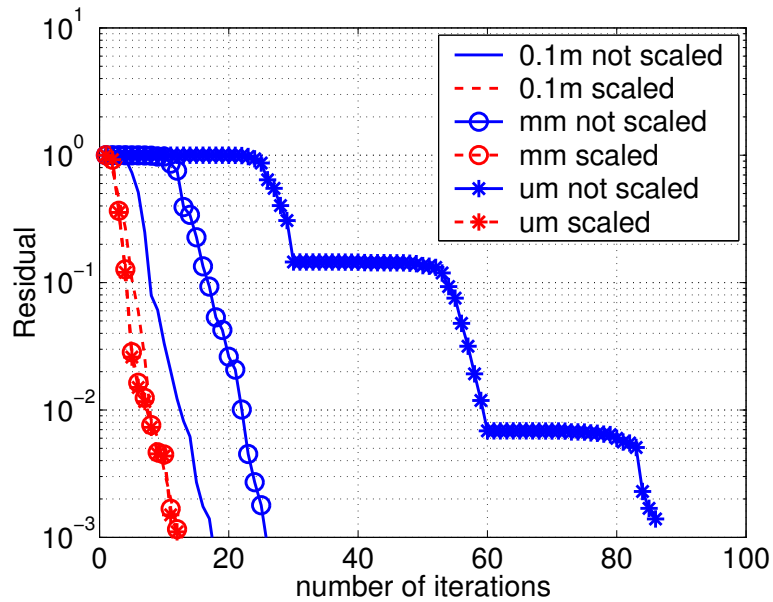
Figure 5-1: Convergence behavior of the iterative solver GMRES for different structure feature sizes with or without scaling. All the dashed (solid) lines are the cases with (without) scaling. The lines with circles (stars) are for the millimeter (micrometer) sized structures, and the lines without any mark are for the structures with feature size in the order of 0.1m.

size is in the order of micron. This confirms the analysis carried out before and verifies our scaling technique.

5.2   Preconditioning

A straightforward way of constructing a preconditioner for the system matrix like the one in (3.32) is to simply replace the dense matrix blocks corresponding to the integral operators with their diagonal elements and keep all other sparse matrix blocks. This method was used

in [103] to construct a preconditioner from the system matrix in (3.32), as shown in

$$
\left[\begin{array}{ccc|ccc|ccc}
P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 & 0 & 0 \\
0 & P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 & 0 \\
0 & 0 & P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 \\
\hline
T_{1,x}P_0^D & T_{1,y}P_0^D & T_{1,z}P_0^D & \frac{1}{2}T_{1,x} & \frac{1}{2}T_{1,y} & \frac{1}{2}T_{1,z} & g_{11} & g_{12} & 0 \\
T_{2,x}P_0^D & T_{2,y}P_0^D & T_{2,z}P_0^D & \frac{1}{2}T_{2,x} & \frac{1}{2}T_{2,y} & \frac{1}{2}T_{2,z} & g_{21} & g_{22} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & 0 & (P_0^1)^D \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & (P_0^2)^D \\
\hline
-A_x & -A_y & -A_z & C_x & C_y & C_z & 0 & 0 & 0 \\
0 & 0 & 0 & N_x & N_y & N_z & 0 & 0 & \frac{-j\omega}{\sigma}I \\
0 & 0 & 0 & T_{1,x} & T_{1,y} & T_{1,z} & 0 & 0 & 0 \\
0 & 0 & 0 & T_{2,x} & T_{2,y} & T_{2,z} & 0 & 0 & 0 \\
N_x & N_y & N_z & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0
\end{array}\right] \qquad (5.2)
$$

where the superscript D means that the matrix block is just the diagonal part of the corresponding block in (3.32). For example, $P_0^D = diag(P_0)$. Extensive numerical experiments have shown that this preconditioner significantly reduces the number of iterations. But for some structures the number of nonzeros in the preconditioner after the sparse LU factorization is still rather large. This is partially because some rows in the preconditioner before the LU factorization are not sparse enough.

As explained in section 3.2, the boundary conditions, equations (3.25)-(3.28), are enforced in the local coordinate system. And it is explained in section 3.4 that equation (3.30) has to be enforced in the local coordinate system. On the other hand, the vector unknowns $\vec{E}$ and $\frac{\partial \vec{E}}{\partial n}$ in (3.32) are defined in the global coordinate system. This inconsistency introduces a large number of nonzeros into (5.2). These nonzeros are mainly transformation between the local and the global coordinates. In addition, the diagonal elements of matrix blocks $T_{1,x}$ and $T_{2,y}$ could be zero. If only these two diagonal blocks are kept in row 4 and

50

5 of the matrix in (5.2),

$$
\begin{bmatrix}
T_{1,x}P_0^D & 0 & 0 & \frac{1}{2}T_{1,x} & 0 & 0 & g_{11} & g_{12} & 0 \\
0 & T_{2,y}P_0^D & 0 & 0 & \frac{1}{2}T_{2,y} & 0 & g_{21} & g_{22} & 0
\end{bmatrix},
\tag{5.3}
$$

some of the elements in $T_{1,x}$ and $T_{2,y}$ will almost certainly be zero, and the inevitable pivoting in LU factorization will constrain the ordering algorithm used to minimize fill-in's, resulting in very dense LU factors. In order to avoid the pivoting or at least minimize it, the off-diagonal blocks in row 4 and 5 of the preconditioner in (5.2) have to be kept. This accounts for a large number of nonzeros.

One way to reduce the number of nonzeros is to define all vector variables and enforce all vector equations in the local coordinate system. The resulting system matrix is

$$
\begin{bmatrix}
T_{11}P_1 & T_{12}P_1 & T_{13}P_1 & T_{11}D_1 & T_{12}D_1 & T_{13}D_1 & 0 & 0 & 0 \\
T_{21}P_1 & T_{22}P_1 & T_{23}P_1 & T_{21}D_1 & T_{22}D_1 & T_{23}D_1 & 0 & 0 & 0 \\
T_{31}P_1 & T_{32}P_1 & T_{33}P_1 & T_{31}D_1 & T_{32}D_1 & T_{33}D_1 & 0 & 0 & 0 \\
T_{11}P_0 & T_{12}P_0 & T_{13}P_0 & T_{11}D_0 & T_{12}D_0 & T_{13}D_0 & g_{11} & g_{12} & 0 \\
T_{21}P_0 & T_{22}P_0 & T_{23}P_0 & T_{21}D_0 & T_{22}D_0 & T_{23}D_0 & g_{21} & g_{22} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & 0 & P_0^1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & P_0^2 \\
0 & 0 & -A_n & C_{t_1} & C_{t_2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & \frac{-j\omega}{\sigma}I \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0
\end{bmatrix},
\tag{5.4}
$$

where matrices $T_{mn}(m,n = 1,2,3))$ are defined as $T_{11}(i,j) = \hat{t}_1^{(i)} \cdot \hat{t}_1^{(j)}$, $T_{12}(i,j) = \hat{t}_1^{(i)} \cdot \hat{t}_2^{(j)}$, $T_{13}(i,j) = \hat{t}_1^{(i)} \cdot \hat{n}^{(j)}$, and etc, and $(\hat{t}_1^{(i)}, \hat{t}_2^{(i)}, \hat{n}^{(i)})$ is the local coordinate system on the i-th panel. The new system matrix in (5.4) is different from the one in (3.32) by just a similarity transformation. Hence they all have the same condition number and lead to the same convergence behavior if an iterative solver is used. But the preconditioners constructed from

51

these different system matrices are significantly different, particularly in matrix sparsity.

An important advantage of defining variables and enforcing equations in the local coordinate system is that the diagonal elements of matrix blocks T_{11}, T_{22} and T_{33} are 1. Hence unlike the preconditioner in (5.2), all the off-diagonal blocks could be thrown away. Extracting the diagonal part of the diagonal matrix blocks and keeping the remaining sparse matrices in (5.4) yields a new preconditioner

$$
\begin{bmatrix}
P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 & 0 & 0 \\
0 & P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 & 0 \\
0 & 0 & P_1^D & 0 & 0 & -\frac{1}{2}I & 0 & 0 & 0 \\
P_0^D & 0 & 0 & \frac{1}{2}I & 0 & 0 & g_{11} & g_{12} & 0 \\
0 & P_0^D & 0 & 0 & \frac{1}{2}I & 0 & g_{21} & g_{22} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & 0 & (P_0^1)^D \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -I\varepsilon & (P_0^2)^D \\
0 & 0 & -A_n & C_{t_1} & C_{t_2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & I & 0 & 0 & \frac{-j\omega}{\sigma}I \\
0 & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I & 0 & 0 & 0 & 0 \\
0 & 0 & I & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & I & 0
\end{bmatrix}. \tag{5.5}
$$

Comparing to the preconditioner in (5.2), this preconditioner is much sparser. The density of most rows below the third row have been reduced by about one half.

To verify the effectiveness of the new and sparser preconditioner, we used it in the analysis of a four-turn spiral over a lossy substrate ground plane. The total number of unknowns is 72531. The performance of the preconditioners in the global and the local coordinate system is compared in table 5.1. As it is expected, the preconditioner in the local coordinate system before the LU factorization is indeed much sparser. But this advantage is somewhat offset by the fill-in's generated by LU factorization. Table 5.1 shows that the number of nonzeros in the local preconditioner after the LU factorization is about 20%

Table 5.1: Performance of preconditioners in the global and the local coordinate system

	local	global
number of nonzeros before LU	320376	547636
number of nonzeros after LU	1097973	1318222
number of fill-in's	777597	770586
number of GMRES iterations	15	15

fewer than that in the global preconditioner. This directly translates into a 20% saving in memory usage. It is worth noting that though more matrix blocks are thrown away in constructing the local preconditioner than the global preconditioner, both preconditioners lead to the same iteration count, as shown in table 5.1.

Chapter 6

Pre-corrected FFT algorithm

After discretization, the algebraic equations (3.26), (3.27), (3.28), (3.29) and (3.31) become sparse matrix equations. But integral equations (3.22), (3.24) and (3.30) become dense matrix equations. So solving the whole system using iterative methods still takes $O(N^2)$ operations, where N is the number of unknowns. Many fast algorithms avoid forming matrix A explicitly and compute the matrix vector product approximately, requiring only $O(N)$ or $O(Nlog(N))$ operations [29, 6, 78]. In this paper, we use the pre-corrected FFT algorithm to accelerate the dense matrix vector product corresponding to the discretized integral operators in (3.22), (3.24) and (3.30).

6.1   Mathematical Preliminaries

An abstract form of the kernels in (3.22), (3.24), and (3.30) is

$$K(\vec{r'},\vec{r}) = \mathcal{F}_1(\mathcal{F}_2(G(\vec{r'},\vec{r}))) \tag{6.1}$$

where $G(\vec{r'},\vec{r})$ is the Green's function, and the possible options for operator $\mathcal{F}_1(\cdot)$ and $\mathcal{F}_2(\cdot)$ are

$$\mathcal{F}_1(\cdot) = U(\cdot), \frac{d(\cdot)}{dx(\vec{r})}, \frac{d(\cdot)}{dy(\vec{r})}, \frac{d(\cdot)}{dz(\vec{r})}, \frac{d(\cdot)}{dn(\vec{r})}, \tag{6.2}$$

and

$$\mathcal{F}_2(\cdot) = U(\cdot), \frac{d(\cdot)}{dx(\vec{r'})}, \frac{d(\cdot)}{dy(\vec{r'})}, \frac{d(\cdot)}{dz(\vec{r'})}, \frac{d(\cdot)}{dn(\vec{r'})}, \tag{6.3}$$

and $U(\cdot)$ is the identity operator.

For the sake of clarity, we use a simple single-kernel integral equation

$$\int_S dS' K(\vec{r'}, \vec{r}) \rho(\vec{r'}) = f(\vec{r}), \quad \vec{r} \in S \tag{6.4}$$

to illustrate how the pFFT algorithm can be used to accelarate the operation of an integral operator. Function $f(\vec{r})$ is the known right hand side term. The procedure extends easily to the integral equations with multiple kernels, such as (3.22), (3.24), and (3.30).

The standard procedure to solve equation (6.4) numerically is to discretize it by means of projection [35] and solve the resultant linear system with an iterative method [88, 98], such as GMRES [89]. Let X be the infinite-dimensional functional space in which the exact solution of equation (6.4) lies, and assume that $B_n \subset X$ and $T_n \subset X$ are its subspaces with spans $\{b_j(\vec{r}), j = 1, 2, ..., n\}$ and $\{t_i(\vec{r}), i = 1, 2, ..., n\}$, where n is the dimension of both subspaces. In general, the solution of the equation (6.4) is not in subspace B_n. Therefore, the approximate solution

$$\rho_n(\vec{r}) = \sum_{j=1}^{n} \alpha_j b_j(\vec{r}) \in B_n \tag{6.5}$$

generates an error

$$e_n(\vec{r}) = \int_S dS' K(\vec{r'}, \vec{r}) \rho_n(\vec{r'}) - f(\vec{r}) = \phi(\vec{r}) - f(\vec{r}), \quad \vec{r} \in S \tag{6.6}$$

and the unknown expansion coefficients α_i could be computed by enforcing the projection of the error into T_n to vanish, i.e.,

$$< t_i(\vec{r}), e_n(\vec{r}) > = < t_i(\vec{r}), \phi(\vec{r}) > - < t_i(\vec{r}), f(\vec{r}) > = 0, \quad i = 1, 2, ..., n \tag{6.7}$$

or

$$\sum_{j=1}^{n} \alpha_j \int_{\Delta_i^t} dS t_i(\vec{r}) \int_{\Delta_j^b} dS' K(\vec{r'}, \vec{r}) b_j(\vec{r'}) = \int_{\Delta_i^t} dS t_i(\vec{r}) f(\vec{r}), \quad i = 1, 2, ..., n, \tag{6.8}$$

56
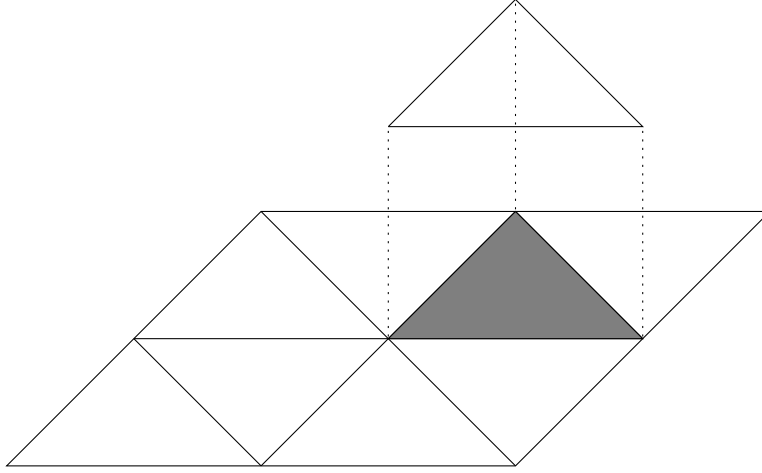
Figure 6-1: A piece-wise constant basis function, shaded area is its support

where Δ_i^t and Δ_j^b are the support of the basis fucntion $t_i(\vec{r})$ and $b_j(\vec{r})$, respectively. In matrix, equation (6.8) becomes

$$[A]\bar{\alpha} = \bar{f} \tag{6.9}$$

where

$$A_{i,j} = \int_{\Delta_i^t} dS t_i(\vec{r}) \int_{\Delta_j^b} dS' K(\vec{r'},\vec{r}) b_j(\vec{r'}) \tag{6.10}$$

The commonly used basis functions in B_n or T_n are low-order polynomials with local support [35]. Figure 6-1 shows a piece-wise constant basis function whose support is a panel. Figure 6-2 shows a vertex-based piece-wise linear basis function whose support is the union of a cluster of panels sharing the vertex with which the basis function is associated. When the i-th testing function is $t_i(\vec{r}) = \delta(\vec{r} - \vec{r_{c,i}})$, where $\vec{r}_{c,i}$ is the collocation point, the discretization method is called the collocation method. And when $B_n = T_n$, the discretization method is called the Galerkin's method.

6.2   Philosophical Preliminaries

Since forming matrix A and computing the matrix vector product in (6.9) all require $O(N^2)$ arithmetic operations, it is obvious that using an iterative method to solve equation (6.9) needs at least $O(N^2)$ time, where N is the size of the matrix A. This could be very expensive for large N. Many fast algorithms avoid forming matrix A explicitly and compute the
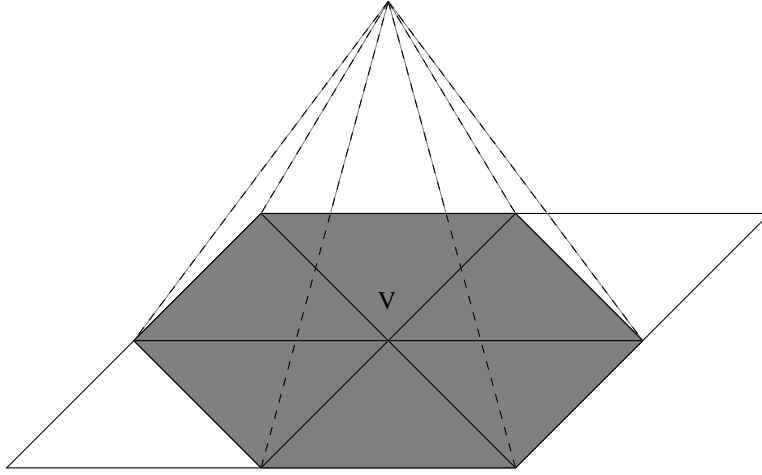
Figure 6-2: A piece-wise linear basis function associated with the vertex V, where the shaded area is its support

matrix vector product approximately, which only needs $O(N)$ or $O(Nlog(N))$ operations [29, 6, 78].

FFT-based methods are well-known [42, 90], but older algorithms required a regular discretization mesh, which is not always possible or optimal for 3D geometries. The Pre-corrected FFT (pFFT) algorithm was originally proposed in [80, 78], where the detailed steps to accelarate a single-layer integral operator were shown. It has also been extended to the case where higher order basis functions are used [24]. The basic idea of pFFT is to separate the potential computation into far-field part and near-field part. The far-field potential is computed by using the grid charges on a uniform 3D grid to represent charges on the panels. The near-field potential is compued directly. The algorithm has four steps: Projection, Convolution, Interpolation and Nearby interaction. The effect of this algorithm is to replace the matrix vector product $A\bar{\alpha}$ in equation (6.9) with $(D+IHP)\bar{\alpha}$, where D is the direct matrix that represents the nearby interaction, I is the interpolation matrix, H is the convolution matrix, and P is the projection matrix. Matrices D, I and P are sparse, hence their memory usage is $O(N_p)$, where N_p is the number of panels, and their product with a vector needs only $O(N_p)$ work. The matrix H is a multilevel Toeplitz matrix. Hence its memory usage is $O(N_g)$ and its product with a vector could be computed by using FFT in $O(N_glog(N_g))$ operations [26], where N_g is the number of grid points. Therefore, the overall computational complexity of $(D+IHP)\bar{\alpha}$ is $O(N_p)+O(N_glog(N_g))$.

Unlike [80, 78], we use polynomials in both interpolation and projection steps. Hence the interpolation matrix I and projection matrix P are completely independent of the Green's function $G(\vec{r}, \vec{r'})$ in equation (6.1). This makes it much easier to handle the complicated kernels $K(\vec{r'}, \vec{r})$ in (6.1). It also makes it straight forward to treat piecewise constant basis and high-order basis in either collocation or Galerkin's method in a unified framework. This is particularly important from implementation point of view.

6.3 Algorithm details

In this section, we will use a simple 2D example to show how to generate the four matrices, $[I]$, $[P]$, $[H]$ and $[D]$. Generalization of the procedure to the 3D cases is straight forward. The algorithm presented here is general enough such that the general integral operator in equation (6.4) discretized either by the collocation method or by the Galerkin's method using either piece-wise const element or high-order element could be handled in a unified framework.

6.3.1 Interpolation matrix

We start with the interpolation, the third and easiest step in the four-step pFFT algorithm.

Suppose the potential on the uniform grids has been computed through the first two steps, namely projection and convolution, we could use a simple polynomial interpolation scheme to compute the potential at any point within the region covered by the grids. Figure 6-3 shows a 2D 3×3 uniform grid, more points could be used to get more accurate results. The triangle inside the grid represents the local support Δ_i^t in equation (6.8). The simplest set of polynomial functions for the interpolation is $f_k(x, y) = x^i y^j, i, j = 0, 1, 2, k = 2i + j$. The potential at any point can be written as a linear combination of these polynomials,

$$\phi(x, y) = \sum_k c_k f_k(x, y) = \bar{f}^t(x, y) \bar{c} \tag{6.11}$$

where $\bar{c}$ is a column vector and t stands for transpose. Matching $\phi(x, y)$ in (6.11) with the

given potential at each grid point results in a set of linear equations. In matrix form, it is

$$[F]\bar{c} = \bar{\phi}_g \tag{6.12}$$

where the j-th row of the matrix $[F]$ is the set of polynomials $\bar{f}(x,y)$ evaluated at the jth grid point (x_j, y_j), and $\phi_{g,j}$ is the given potential at point (x_j, y_j). Solving for $\bar{c}$ and substituting it back into (6.11) yields

$$\phi(\vec{r}) = \phi(x,y) = \bar{f}^t(x,y)[F]^{-1}\bar{\phi}_g = \bar{D}_0^t(\bar{r})\bar{\phi}_g \tag{6.13}$$

It should be noted that matrix $[F]$ in (6.12) is only related to the distance between points in the uniform grid and the specific set of interpolation polynomials chosen in the algorithm. So the inverse of matrix $[F]$ is done only once. And since the size of the matrix is rather small (9×9 in this simple 2D case), computing its inverse is inexpensive. It is possible that the number of polynomials is not equal to the number of points in the interpolation grid. In this case the inverse becomes psuedo inverse, which is computed using the singular value decomposition (SVD) [98].

It easily follows that the derivative of the potential at a point $\bar{r}$ with respect to α is

$$\frac{d\phi(\bar{r})}{d\alpha} = \frac{d}{d\alpha}\bar{f}^t(\bar{r})[F]^{-1}\bar{\phi}_g = \bar{D}_\alpha^t(\bar{r})\bar{\phi}_g \tag{6.14}$$

where α stands for x or y. Hence the gradient of the potential at $\bar{r}$ is

$$\nabla\phi(\bar{r}) = (\hat{x}\bar{D}_x^t(\bar{r}) + \hat{y}\bar{D}_y^t(\bar{r}))\bar{\phi}_g \tag{6.15}$$

and the normal derivative of the potential at point $\bar{r}$ is

$$\frac{d\phi(\bar{r})}{dn} = \hat{n} \cdot \nabla\phi(\bar{r}) = (n_x\frac{d\bar{f}^t(\bar{r})}{dx} + n_y\frac{d\bar{f}^t(\bar{r})}{dy})[F]^{-1}\bar{\phi}_g = \bar{D}_n^t(\bar{r})\bar{\phi}_g \tag{6.16}$$

where n_x and n_y are the projection of the unit normal vector of the function support Δ_i^t along x and y direction. Using the notation in (6.2), equations (6.13), (6.14) and (6.16) could be

60

written as

$$\mathcal{F}_1(\phi(\vec{r})) = \bar{D}_\beta^t(\vec{r})\bar{\phi}_g \tag{6.17}$$

where $\bar{D}_\beta^t(\vec{r})$ stands for $\bar{D}_0^t(\vec{r})$, $\bar{D}_x^t(\vec{r})$, $\bar{D}_y^t(\vec{r})$ or $\bar{D}_n^t(\vec{r})$.

As described in section 6.1, we want to compute

$$\Psi_i = \int_{\Delta_i^t} dS\, \mathcal{F}_1(\phi(\vec{r}))t_i(\vec{r}), \quad i = 1, 2, .., N_t. \tag{6.18}$$

where N_t is the number of testing basis functions. Substituting (6.17) into (6.18) yields

$$\Psi_i = \int_{\Delta_i^t} dS\, t_i(\vec{r})\bar{D}_\beta^t(\vec{r})\bar{\phi}_g = (\bar{W}_\beta^{(i)})^t \bar{\phi}_g, \quad i = 1, 2, .., N_t, \tag{6.19}$$

where $\bar{W}_\beta^{(i)}$ stands for $\bar{W}_0^{(i)}$, $\bar{W}_x^{(i)}$ and $\bar{W}_y^{(i)}$. If the collocation method is used, then $\bar{W}_\beta^{(i)}$ in equation (6.19) could be simplified as

$$\bar{W}_\beta^{(i)} = \bar{D}_\beta(x_c, y_c), \quad i = 1, 2, .., N_t, \tag{6.20}$$

where (x_c, y_c) is the collocation point. When the piece-wise constant testing function is used, the support Δ_i^t is the panel associated with it, as shown in figure 6-1. When the linear testing function is used, Δ_i^t is a cluster of panels, as shown in figure 6-2. Apparently, computing elements of $\bar{W}_\beta^{(i)}$ for higher order basis functions could be more expensive because integrating over a cluster of panels needs more quadrature points than integrating over a single panel.

In matrix format, equation (6.19) becomes

$$\bar{\Psi} = [I]\bar{\phi}_g \tag{6.21}$$

where $[I]$ is an $N_t \times N_g$ matrix, and N_g is the number of grid points. To cover the local support of a basis function, only a small number of the interpolation grid points are needed, as shown in figure 6-3. Hence computing each Ψ_i through interpolation only involves potential at a few grid points. So each row of the interoplation matrix $[I]$ is rather sparse. The non-zero elements in the i-th row of the matrix $[I]$ are just the elements of the row

61
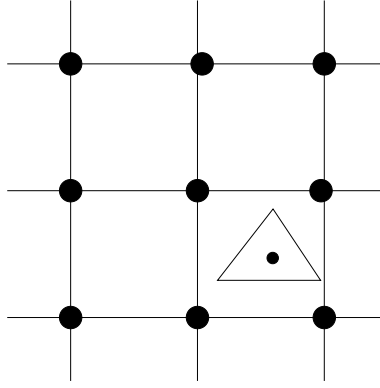
Figure 6-3: 2-D pictorial representation of the interpolation step

vector $(\bar{W}_\beta^{(i)})^t$ in (6.19) or (6.20).

6.3.2 Projection matrix

Figure 6-4 shows a 2D pictorial representation of the projection step. Similar to the previous section, a triangle is used to represent the support of a basis function. A 3×3 projection grid is assumed here and obviously more points could be used if the accuracy requirement is higher.

We start with a point charge ρ_p at point **S** on the triangle, shown in figure 6-4. The potential at point **E** due to this point charge is

$$\phi_E^{(1)} = \rho_p G(\vec{r}_s, \vec{r}_E). \tag{6.22}$$

The purpose of the projection is to find a set of grid charges $\bar{\rho}_g$ on the projection grid points such that they generate the same potentail at point **E**, i.e.,

$$\phi_E^{(2)} = \sum_i \rho_{g,i} G(\vec{r}_i, \vec{r}_E) = (\bar{\rho}_g)^t \bar{\phi}_g = \phi_E^{(1)} \tag{6.23}$$

where $\phi_{g,i} = G(\vec{r}_i, \vec{r}_E)$. We could use the same set of polynomials in (6.11) to expand the Green's function

$$G(\vec{r}, \vec{r}_E) = \sum_k f_k(\vec{r}) c_k = \bar{f}^t(\vec{r}) \bar{c}. \tag{6.24}$$

62

Matching both sides at each grid point $\vec{r}_i$ yields a linear system

$$[F]\bar{c} = \bar{\phi}_g, \tag{6.25}$$

where F is same as that in (6.12). Substituting the solution $\bar{c} = F^{-1}\bar{\phi}_g$ into (6.24) and evaluating it at point **S** yields

$$G(\vec{r}_s, \vec{r}_E) = \bar{f}^t(\vec{r}_s)F^{-1}\bar{\phi}_g. \tag{6.26}$$

In light of (6.22) and (6.23) we have

$$(\bar{\rho}_g)^t = \rho_p\bar{f}^t(\vec{r}_s)F^{-1}, \tag{6.27}$$

the projection charges for a point charge. A charge distribution $b_j(\vec{r})$ on the jth basis function support could be regarded as a linear combination of an infinite number of point charges. Equation (6.27) implies that the projection charges are linearly proportional to the point charge, hence it easily follows that the projection charges for the charge distribution $b_j(\vec{r})$ is

$$(\bar{\rho}_g^{(j)})^t = [\int_{\Delta_j^b} dS b_j(\vec{r})\bar{f}^t(\vec{r})][F]^{-1}. \tag{6.28}$$

If the piece-wise constant basis function is used, equation (6.28) becomes

$$(\bar{\rho}_g^{(j)})^t = [\int_{\Delta_j^b} dS \bar{f}^t(\vec{r})][F]^{-1}. \tag{6.29}$$

We usually have to use more than one basis function, as implied by equation (6.5). In this case, the total charge on each grid point is the accumulation of grid charge due to each basis function. Assuming there are N_b basis functions and N_g grid points, the relation between the total grid charges $\bar{Q}_g$ and the magnitude of basis functions $\bar{\alpha}$ in (6.5) is

$$\bar{Q}_g = \sum_{j=1}^{N_b} \alpha_j\bar{\rho}_g^{(j)} = [P]\bar{\alpha} \tag{6.30}$$

where $[P]$ is an $N_g \times N_b$ matrix. Due to the locality of the basis support, the projection

63

grid for each basis function has only a small number of points. Hence each column of the projection matrix $[P]$ is rather sparse. The non-zero elements in the j-th column of matrix $[P]$ are the elements of the column vector $\bar{\rho}_g^{(j)}$ in equation (6.28) or (6.29).

If the kernel has a differential operator inside the integral, the potential at point $\mathbf{E}$ due to a point charge is

$$\phi_E^{(1)} = \frac{\partial}{\partial \beta(\vec{r}_s)} [\rho_p G(\vec{r}_s, \vec{r}_E)] = \frac{\partial}{\partial \beta(\vec{r}_s)} [\rho_p \bar{f}^t(\vec{r}_s) F^{-1} \bar{\phi}_g]. \tag{6.31}$$

where β stands for x, y or n. We again want to find a set of grid charges $\bar{\sigma}_\beta$ on the projection grid points such that they generate the same potentail at point $\mathbf{E}$, i.e.,

$$\phi_E^{(2)} = \sum_i \sigma_{\beta,i} G(\vec{r}_i, \vec{r}_E) = (\bar{\sigma}_\beta)^t \bar{\phi}_g = \phi_E^{(1)} \tag{6.32}$$

Equations (6.31) and (6.32) imply that the projection charges are

$$(\bar{\sigma}_\beta)^t = \frac{\partial}{\partial \beta(\vec{r}_s)} [\rho_p \bar{f}^t(\vec{r}_s) F^{-1}]. \tag{6.33}$$

Similar to the single-layer operator case, the projection charges for a charge distribution $b_j(\vec{r})$ on the jth basis function support is

$$\left(\bar{\sigma}_\beta^{(j)}\right)^t = [\int_{\Delta_j^b} dS b_j(\vec{r}) \frac{\partial}{\partial \beta(\vec{r})} \bar{f}^t(\vec{r})][F]^{-1}. \tag{6.34}$$

The projection matrix for the kernel with a differential operator is structurely identical to the matrix $[P]$ in equation (6.30). The non-zero elements in the j-th column of the matrix are the elements of the column vector $\bar{\sigma}_\beta^{(j)}$ in equation (6.34).

6.3.3 Convolution matrix and fast convolution by FFT

By definition, the relation between the grid potential $\bar{\phi}_g$ in (6.21) and grid charge $\bar{Q}_g$ in (6.30) is

$$\phi_{g,j} = \sum_i G(\vec{r'}_i, \vec{r}_j) Q_{g,i} \tag{6.35}$$
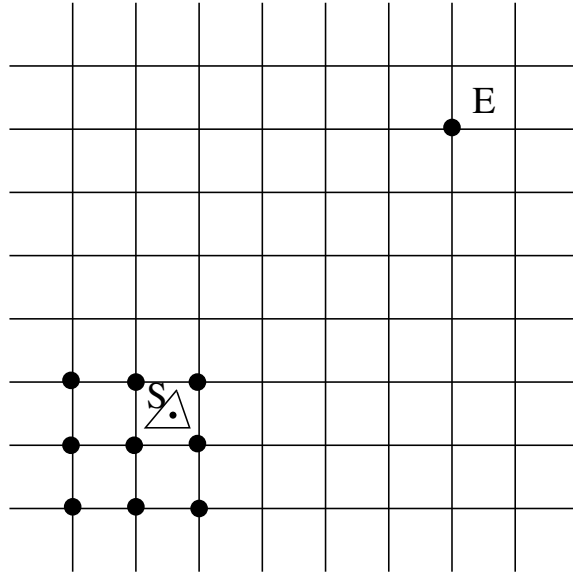
Figure 6-4: 2-D pictorial representation of the projection step

In matrix form, it is

$$\bar{\phi}_g = [H]\bar{Q}_g \tag{6.36}$$

where the matrix H is the so-call convolution matrix. Since the Green's function is position invariant and $\bar{\phi}_g$ and $\bar{Q}_g$ are defined on the same set of uniform grid, we have

$$H_{i,j} = G(\vec{r'}_i, \vec{r}_j) = G(\vec{r}_i, \vec{r}_j) = G(\vec{r}_i - \vec{r}_j, 0). \tag{6.37}$$

Matrix H is a multilevel Toeplitz matrix [26]. The number of levels is 2 and 3 for 2D cases and 3D cases, respectively. It is well-known that the storage of a Toeplitz matrix only needs $O(N)$ memory and a Toeplitz matrix vector product can be computed in $O(Nlog(N))$ operations using FFT [26], where N is the total number of grid points. It should be pointed out that convolution matrix H being a Toeplitz matrix is hinged upon the position invariance of the Green's function.

65

6.3.4 Direct matrix and pre-correction

Substituting equation (6.36) and (6.30) into (6.21) yields

$$\bar{\Psi} = [I][H][P]\bar{\alpha} \tag{6.38}$$

In view of (6.18), (6.7) and (6.9), this implies

$$A = [I][H][P]. \tag{6.39}$$

As pointed out in previous three sections, the sparse representation of matrix A in (6.39) reduces the memory usage and computing time for matrix vector product dramatically. Unfortunately, the calculations of the potential on th grid using (6.39) do not accurately approximate the nearby interaction. It is proposed in [78] that the nearby interaction should be computed directly and the inaccurate contributions from the use of grid should be removed. Figure 6-5 shows how the nearby neighboring basis supports are defined. The empty circle in middle of the solid dots are the center of the so-called direct stencil and the stencil size in figure 6-5 is 2. The shaded triangle represents the source, and the other empty triangles represent the targets where Ψ in equation (6.18) is to be evaluated. Only those triangles within the region covered by the direct stencil are considered to be nearby neighbors to the source. And the direct interaction between this list of nearby neighbors and the source is just $A_{i,j}$ defined in (6.10), where i is the index of the shaded triangle representing the source and $j \in \mathcal{N}_i$, the nearby neighbor set for the ith source. The pre-corrected direct matrix element is

$$D_{i,j} = A_{i,j} - (\bar{W}_\beta^{(i)})^t [H_L] \bar{\rho}_g^{(j)}, \quad j \in \mathcal{N}_i \tag{6.40}$$

where $(\bar{W}_\beta^{(i)})^t$ is defined in equation (6.19), $\bar{\rho}_g^{(j)}$ is defined in equation (6.28) and (6.34), and $[H_L]$ is a small convolution matrix (not to be confused with $[H]$ in (6.39)) that relates the potential on the grid points around basis support Δ_i^t and the charge on the grid points around basis support Δ_j^b. It is intuitive from figure 6-5 that $\mathcal{N}_i$ is a very small set. Hence the direct matrix D is very sparse and the sparsity of D is dependent upon the size of the direct stencil. Larger stencil size means more neighboring triangles in figure 6-5 and hence more
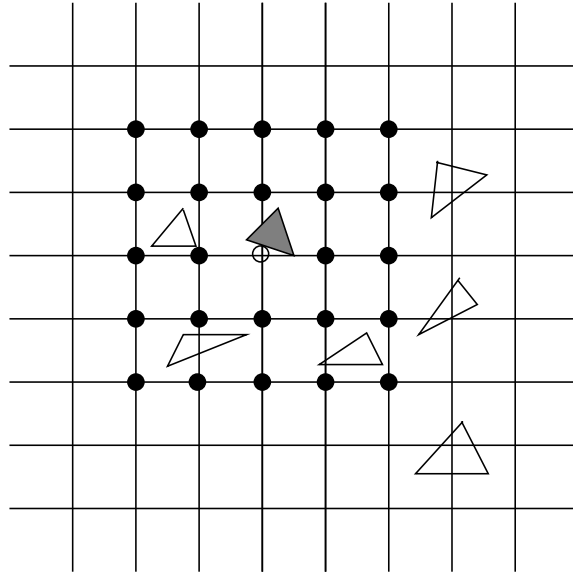
Figure 6-5: 2-D pictorial representation of the nearby interaction. Direct stencil size is 2.

computation in (6.40). It will be shown later in section 7.1 that the setup time of the pFFT algorithm is directly related to the direct stencil size.

Since matrix $[H_L]$ in (6.40) is rather small, the FFT does not speed up the computation much. However, there are other ways to reduce the operation count. Because the grid is uniform and the Green's function is position invariant, only a few matrices $[H_L]$ are unique. So we could pre-compute them once and use them to pre-correct all the nearby interactions in the direct matrix $[D]$.

6.3.5 A summary of the four matrices

In view of (6.38), (6.39) and (6.40), the matrix vector product is computed efficiently using

$$[A]\bar{\alpha} = ([D] + [I][H][P])\bar{\alpha}. \tag{6.41}$$

Sections 6.3.1 and 6.3.2 are summarized in table 6.1. It is clear by now that the interpolation matrix $[I]$ and the projection matrix $[P]$ are independent of the Green's function. Matrix $[I]$ is only related to the operator $\mathcal{F}_1$ and the testing functions. And matrix $[P]$ is only related to the operator $\mathcal{F}_2$ and the basis functions.

The direct matrix, however, is dependent upon all the above information. So we have

67

Table 6.1: Relation between operator pair and the interpolation matrix and the projection matrix

$\mathcal{F}_1$	$U(\cdot)$	$\frac{d(\cdot)}{dx'}, \frac{d(\cdot)}{dy'}$	$\frac{d(\cdot)}{dn'}$
interpolation	$\bar{W}_0^{(i)}$ in (6.19)	$\bar{W}_x^{(i)}, \bar{W}_y^{(i)}$ in (6.19)	$\bar{W}_n^{(i)}$ in (6.19)
$\mathcal{F}_2$	$U(\cdot)$	$\frac{d(\cdot)}{dx}, \frac{d(\cdot)}{dy}$	$\frac{d(\cdot)}{dn}$
projection	$\bar{\rho}_g^{(j)}$ in (6.28)	$\bar{\sigma}_x^{(j)}, \bar{\sigma}_y^{(j)}$ in (6.34)	$\bar{\sigma}_n^{(j)}$ in (6.34)

to set up one direct matrix for each $\mathcal{F}_1$ and $\mathcal{F}_2$ operator pair. The convolution matrix, on the other hand, is only related to the Green's function and the location of grid points. It is not related to $\mathcal{F}_1$ or $\mathcal{F}_2$. So we only need to set up one convolution matrix for each unique Green's function.

In addition, if the Galerkin's method is used, the basis function $b_j(\vec{r})$ in equation (6.28) or (6.34) is identical to the testing function $t_i(\vec{r})$ in equation (6.19). It is easy to check that $\bar{W}_0^{(i)} = \bar{\rho}_g^{(j)}$, $\bar{W}_x^{(i)} = \bar{\sigma}_x^{(j)}$, $\bar{W}_y^{(i)} = \bar{\sigma}_y^{(j)}$ and $\bar{W}_n^{(i)} = \bar{\sigma}_n^{(j)}$. This implies a duality relation

$$[I] = [P]^t. \tag{6.42}$$

6.4   Implementation

Base upon the algorithm described above, we have developed a C++ program called pfft++, using the generic programming technique [95, 56, 46]. The whole algorithm includes two major parts: forming the four matrices I, P, D and H, and computing the matrix vector product using (6.41). Since the matrices I and P are not related to the kernel, they are formed separately so that they could be used for different kernels. This is particularly useful when for example a Helmholtz equation is to be solved at various wave numbers or frequencies. A high level description of the implementation of the pfft++ is shown in algorithms 1, 2 and 3.

Using pfft++ to solve a single kernel integral equation such as (6.4) is straight forward. We could simply treat pfft++ as a black box that could perform the matrix vector product efficiently. After forming the four matrices by calling algorithms 1 and 2, algorithm 3 is

Algorithm 1: construct kernel Independent sparse matrices.
Input: source elements, target elements, differential operator pairs ($\mathcal{F}_1$, $\mathcal{F}_2$), projection stencil size, interpolation stencil size, direct stencil size
Output: interpolation matrix $[I]$ and projection matrix $[P]$
(1)   find the optimal grid size
(2)   setup grid and element association
(3)   setup interpolation stencil
(4)   setup projection stencil
(5)   setup direct stencil
(6)   form the interpolation matrix $[I]$ for each $\mathcal{F}_1$
(7)   form the projection matrix $[P]$ for each $\mathcal{F}_2$

Algorithm 2: construct kernel dependent sparse matrices.
Input: source elements, target elements, kernel, integration scheme, differential operator pairs ($\mathcal{F}_1$, $\mathcal{F}_2$)
Output: direct matrix $[D]$ and convolution matrix H
(1)   form the sparse representation of $[H]$
(2)   compute the FFT of $[H]$
(3)   form the direct matrix $[D]$ for each pair of ($\mathcal{F}_1$, $\mathcal{F}_2$)

to be called repeatedly in the inner loop of an iterative solver. To solve the integral equations with multiple kernels, we could simply repeat the above procedure for each integral operator individually.

6.5   Comparison to the original pFFT algorithm

The basic sparsification ideas used here are very similar to those in the original pre-corrected FFT algorithm [80]. The difference lies primarily in the ways the interpolation matrix and the projection matrix are generated. And this difference turns out to be important.

In the original pFFT algorithm [80, 78], the projection matrix and the interpolation matrix are all related to the Green's function or kernel. If one wants to solve a Helmholtz equation with different wave numbers or at different frequencies, these two matrices have to be re-generated for each frequency. As explained in section 6.4, the interpolation matrix and the projection matrix are only generated once in pfft++.

In the original pFFT algorithm, the convolution matrix is directly related to the kernel,

Algorithm 3: compute matrix vector product.
Input: vector x, differential operator pair $(\mathcal{F}_1, \mathcal{F}_2)$
Output: vector y
(1) find the index m of $[I]$ from $\mathcal{F}_1$
(2) find the index n of $[P]$ from $\mathcal{F}_2$
(3) find the index k of $[D]$ from operator pair $(\mathcal{F}_1, \mathcal{F}_2)$
(4) $y_1 = [P_m]x$
(5) $y_1 = fft(y_1)$
(6) $y_2 = [H]y_1$
(7) $y_2 = ifft(y_2)$
(8) $y_3 = [I_n]y_2$
(9) $y = y_3 + [D_k]x$

which includes the effect of the operator $\mathcal{F}_2$. The convolution matrix in this work is directly related to the Green's function, not the operator $\mathcal{F}_2$. To see why this difference is important, suppose we want to compute the double-layer integral

$$\int_S d\vec{r}' \frac{\partial G(\vec{r}, \vec{r}')}{\partial n(\vec{r}')} \rho(\vec{r}'). \tag{6.43}$$

Using the original pFFT algorithm, it has to be done as the following

$$\int_S d\vec{r}' [n_x \frac{\partial G(\vec{r}, \vec{r}')}{\partial x(\vec{r}')} + n_y \frac{\partial G(\vec{r}, \vec{r}')}{\partial y(\vec{r}')} + n_z \frac{\partial G(\vec{r}, \vec{r}')}{\partial z(\vec{r}')}] \rho(\vec{r}'). \tag{6.44}$$

This suggests that three convolution matrices $[H_x]$, $[H_y]$ and $[H_z]$ corresponding to $\frac{\partial G}{\partial x}$, $\frac{\partial G}{\partial y}$ and $\frac{\partial G}{\partial z}$ have to be generated and foreward FFT has to be performed for each of them. For each operation of the double-layer integral operator, $[H_x]\bar{\rho}$, $[H_y]\bar{\rho}$ and $[H_z]\bar{\rho}$ have to be carried out separately. As shown in section 6.3.3, pfft++ only needs one convolution matrix and hence only one convolution will be carried out in the matrix vector product step. This is a significant reduction in memory usage and CPU time.

Chapter 7

Numerical Results

7.1 Performance of pfft++

Base upon the algorithm described in chapter 6, we have developed pfft++, a flexible and extensible fast integral equation solver. The program pfft++ has been tested using random distributions on the surface of a sphere shown in figure 7-1. After discretizing the surface, the integral operator in equation (6.4) is turned into either the dense matrix $[A]$ in (6.9) or the sparse matrix representation in (6.41). We assume a random vector α and compute the matrix vector product in (6.9) directly as $y_1 = [A]\bar{\alpha}$. We then compute the matrix vector product using pfft++ as $y_2 = pfft(\bar{\alpha})$. The relative error in the pFFT approximation is

$$error = \left(\frac{\sum_{i=1}^{N}(y_{1,i} - y_{2,i})^2}{\sum_{i=1}^{N} y_{1,i}^2}\right)^{1/2}. \tag{7.1}$$

We first use a medium size example to demonstrate the trade-off between accuracy and CPU time and memory usage. We carried out the numerical experiment described above on a sphere discretized with 4800 panels. When the kernels are Laplace kernel and its normal derivative, the radius of the sphere is $R = 1m$. When the kernels are Helmholtz kernel and its normal derivative, the radius of the sphere is $R = 5.3cm$ so that the size of the panels is smaller than one tenth of a wave length at 10GHz. Using increasingly larger stencil size in projection and interpolation, the accuracy is expected to increase. Table 7.1 clearly shows that this expectation has been met, where p stands for the stencil size of
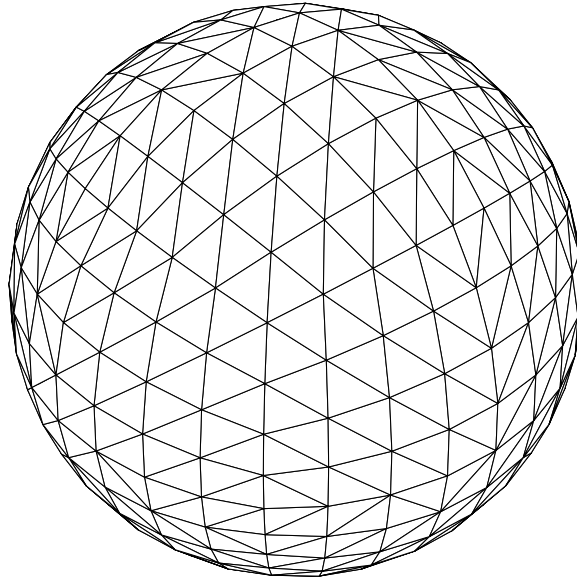
Figure 7-1: A sphere discretized with triangle panels.

both projection and interpolation. For instance, $p = 3$ means that a $3 \times 3 \times 3$ 3D grid is used as the projection and the interpolation stencil. With the increase of the stencil size, the computational resource is expected to increase as well. This is shown in table 7.2, 7.3 and 7.4. The CPU time and memory usage increase significantly with the increase of the stencil size. In particular, the setup time of pfft++ increases by 4 to 10 times when stencil size increases from 3 to 5 or from 5 to 7. Though we only show data for a medium size problem here, from our numerical experiments, the observation is also true for large examples. Foutunately, almost all engineering problems only require modest accuracy, 3 to 4 digits. At this level of accuracy, the computational cost of pfft++ is very reasonable.

Figure 7-2 shows the CPU time versus problem size for different kernels. The projection and the interpolation stencil size is 3 for all these cases. It is clear that the CPU time grows almost linearly with the problem size for all types of kernels. Though not shown here in plot, the memory usage of pfft++ also grows linearly with the problem size.

7.2   Performance of fastImp

Base upon the algorithms described in previous chapters we have developed FastImp, a fast impedance extraction program. In this section, we first use small examples to demon-

Table 7.1: Relative error in (7.1) for different projection and interpolation stencil sizes and different kernels

	$p = 3$	$p = 5$	$p = 7$
$\frac{1}{r}$	8.4×10^{-5}	1.3×10^{-6}	4.3×10^{-9}
$\frac{\partial}{\partial n}\frac{1}{r}$	8.5×10^{-3}	1.1×10^{-4}	8.4×10^{-7}
$\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	8.3×10^{-5}	1.3×10^{-6}	1.7×10^{-9}
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	6.0×10^{-3}	7.5×10^{-5}	5.9×10^{-7}
$\frac{e^{ikr}}{r}, kR = 11.1$	4.9×10^{-4}	1.1×10^{-5}	4.0×10^{-7}
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 11.1$	1.4×10^{-2}	2.8×10^{-4}	6.5×10^{-6}

Table 7.2: CPU time for forming I, P, D and H matrices in (6.41) for different projection and interpolation stencil sizes and different kernels, unit is second

	$p = 3$	$p = 5$	$p = 7$
$\frac{1}{r}$	3.76	39.48	305.61
$\frac{\partial}{\partial n}\frac{1}{r}$	4.28	45.93	326.47
$\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	55.66	249.01	1022.05
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	47.80	229.02	971.32
$\frac{e^{ikr}}{r}, kR = 11.1$	53.06	242.65	1082.36
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 11.1$	47.99	226.89	967.58

Table 7.3: CPU time for doing one matrix vector product for different projection and interpolation stencil sizes and different kernels, unit is second

	$p = 3$	$p = 5$	$p = 7$
$\frac{1}{r}$	0.07	0.11	0.17
$\frac{\partial}{\partial n}\frac{1}{r}$	0.07	0.11	0.17
$\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	0.20	0.33	0.64
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	0.20	0.33	0.61
$\frac{e^{ikr}}{r}, kR = 11.1$	0.19	0.32	0.63
$\frac{\partial}{\partial n}\frac{e^{ikr}}{r}, kR = 11.1$	0.19	0.33	0.65

Table 7.4: Memory usage for different projection and interpolation stencil sizes and different kernels, unit is Mb

	$p = 3$	$p = 5$	$p = 7$
$\frac{1}{r}$	10.75	35.18	87.94
$\frac{\partial}{\partial n} \frac{1}{r}$	10.75	35.18	87.94
$\frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	16.04	47.3	114.5
$\frac{\partial}{\partial n} \frac{e^{ikr}}{r}, kR = 1.11 \times 10^{-9}$	16.04	47.3	114.5
$\frac{e^{ikr}}{r}, kR = 11.1$	16.04	47.3	114.5
$\frac{\partial}{\partial n} \frac{e^{ikr}}{r}, kR = 11.1$	16.04	47.3	114.5
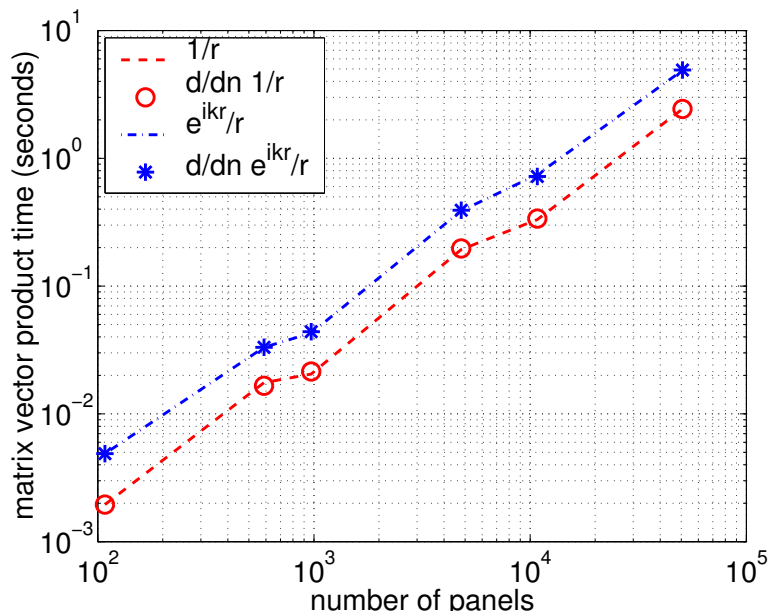
Figure 7-2: CPU time of one matrix-vector product versus the problem size, $kR = 11.1$ for Helmholtz kernel and its normal derivative where R is the radius of the sphere.

strate FastImp's accuracy. We then use a few practical examples to demonstrate FastImp's flexibility. And finally we use large examples to show FastImp's speed. If not specified explicitly, the calculations were carried out on a desk top computer with a Pentium IV micro-processor (1.2 GHz clock rate) and 1GB memory.

7.2.1 Accuracy

A ring example This ring example is used to verify the piecewise quadrature scheme proposed in chapter 4. We also intend to use this relatively small example to conduct the convergence test of FastImp. The third goal is to numerically show that with the help of a preconditioner the formulation behaves reasonably well across a wide frequency range.

The ring is $10mm$ in radius, with a square cross section of the size $0.5mm \times 0.5mm$. The conductivity is that of the copper, which is $5.8 \times 10^7 sm^{-1}$. In order to compare with results from the well-established FastHenry program [49], we have carried out the Magneto-Quasi-Static (MQS) analysis. The number of nonuniform filaments used by FastHenry is 960, 3840 and 15360, respectively. The incremental ratio of the nonuniform filaments is 1.3. The number of nonuniform panels used by FastImp is 992 and 2048, respectively. The resistance and inductance calculated by both codes are compared in figures 7-3 and 7-4, where the low frequency inductance calculated using the analytical formula in [32] is 48.89 nH. These two figures show that FastImp's results converges very well with the refinement of panel discretization. It should be noted that the inductance calculated with FastImp is very close to 48.89nH in the low frequency range, as shown in figure 7-4. This suggests that the piecewise quadrature scheme proposed in chapter 4 has indeed eliminated the low frequency problem reported in [105]. Also, at high frequencies, the resistance in figure 7-3 scales to the square root of frequency and the inductance in figure 7-4 drops a little. This suggests that the skin effect has been well captured.

It is worth mentioning that a large number of filaments has to be used by FastHenry in order to capture the skin effect at high frequencies. On the other hand, with a small and fixed number of panels, the skin effect has been well captured by FastImp. This clearly demonstrates the advantage of the surface integral formulation over the volume integral
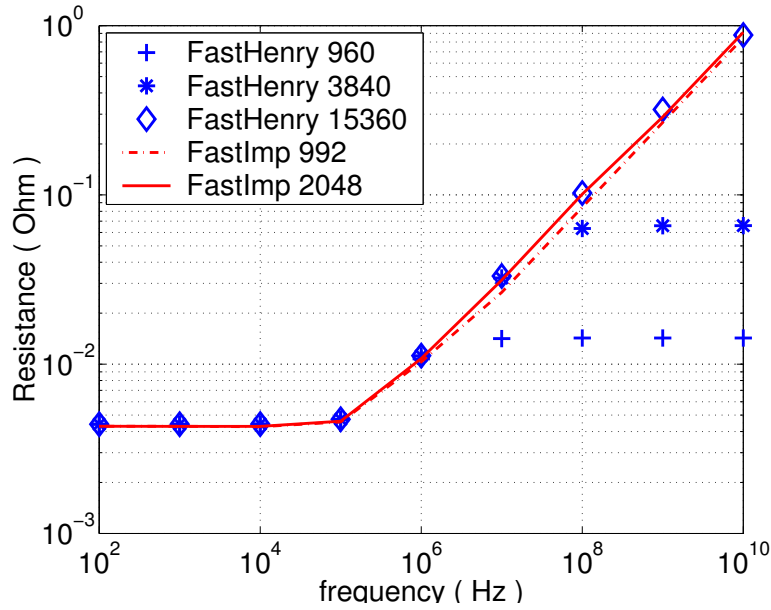
Figure 7-3: Resistance of a ring

formulation.

Figure 7-5 shows the number of GMRES iterations versus frequency for discretization with 992 and 2048 panels. Here the tolerance for GMRES is set to be 1×10^{-4}. Though low-frequency calculation takes more GMRES iterations, the number of iterations is still very reasonable, considering the number of unknowns is 6946 and 14338, respectively. This indicates that FastImp is indeed a wideband solver.

A shorted transmission line The length of the transmission line is $2cm$. The cross-section of each conductor is $50 \times 50um$, and the space between two conductors is $50um$. The conductivity of both conductors is again $5.8 \times 10^7 sm^{-1}$. One end of this transmission line is shorted, and the other end becomes a port.

In theory, this finite-length transmission line is a 3D structure. We have used FastImp to carry out 3D Magneto-Quasi-Static (MQS) analysis, 3D Electro-Magneto-Quasi-Static (EMQS) analysis and 3D fullwave analysis and calculated the input impedance at the open port. This finite-length transmission line could also be treated as a quasi 2D structure since its length is 400 times its width and thickness. In this case, the per-unit-length resistance R_0 and inductance L_0 are obtained by dividing the total resistance and inductance (from the

76
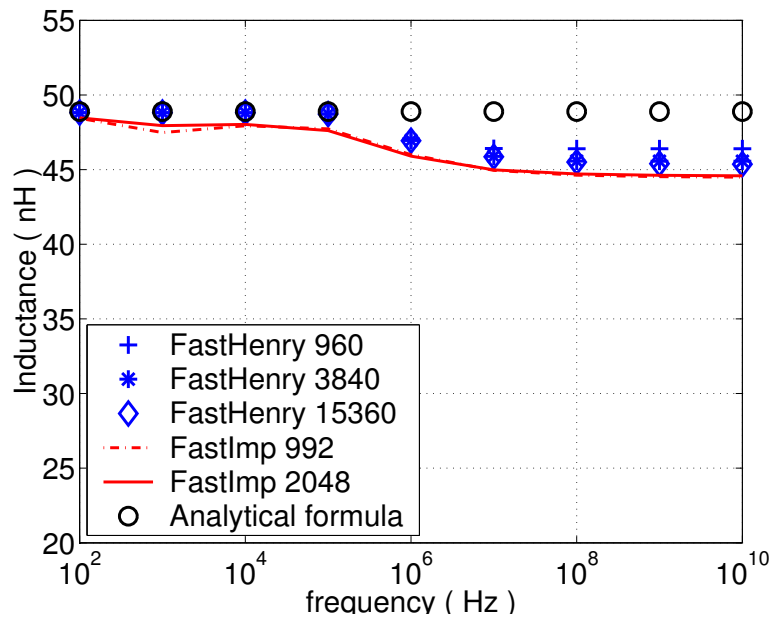
Figure 7-4: Inductance of a ring
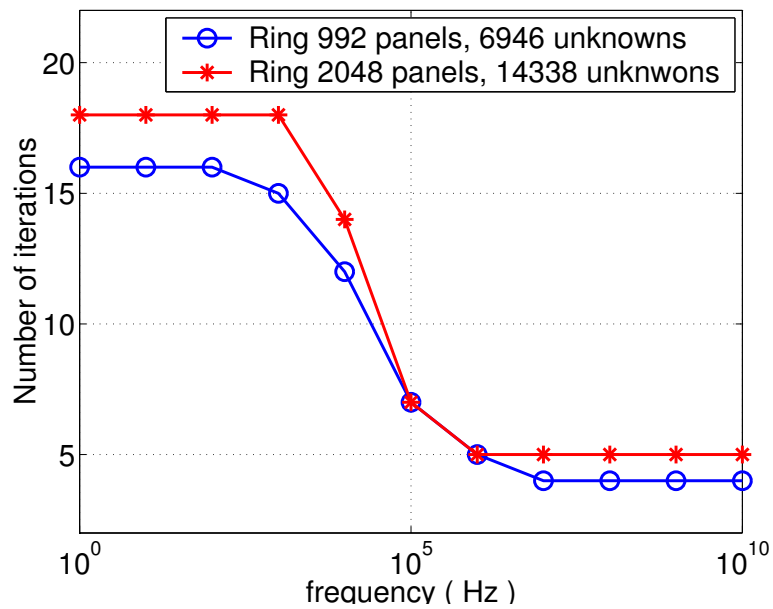
Figure 7-5: Number of GMRES iterations versus frequency

77

3D MQS analysis) by the length. And the per-unit-length capacitance C_0 is just

$$C_0 = \frac{1}{c^2 L_0},$$

where c is the speed of light. The behavior of a shorted 2D transmission line is well understood. We could calculate its input impedance from R_0, C_0 and L_0 [81]. This 2D analytic model is used as standard for EMQS results.

We have also used FastHenry to calculated the input impedance of this shorted transmission line. We used 20×20 filaments on the cross-section of each conductor. Since this rather fine discretization guarantees that the size of the filaments close to the conductor surface is smaller than the smallest skin depth in the frequency range we care about, FastHenry's results should be accurate.

In order to verify FastImp's MQS, EMQS and fullwave analysis, we have plotted figures 7-6 and 7-7, the magnitude and phase of the input impedance calculated using different methods. FastImp's MQS results are almost identical to FastHenry's results. FatsImp's EMQS results are almost indistinguishable from those of the 2D analytic model. As expected, the EMQS results are essentially the same as MQS results at lower frequencies (less than $1GHz$), and resonances appear at higher frequencies. Since the separation distance between two conductors of the transmission line is only $50um$, a small fraction of the shortest wave length at $10GHz$, the fullwave results are essentially same as the EMQS results.

To see the difference in EMQS and fullwave results, we deliberately used a larger separation between the two conductors of the shorted transmission line. Figures 7-8 and 7-9 show the the magnitude and phase of the input admittance calculated using EMQS and fullwave modes. Here the separation is $1cm$. Due to fullwave radiation loss at resonant frequencies, the magnitude of fullwave admittance is smaller than that of EMQS admittance at resonant frequencies, as shown in figure 7-8. For the same reason, the imaginary part of the fullwave admittance is less dominant than that of EMQS admittance. Hence the phase change at resonant frequencies for fullwave admittance is not as sharp as that of EMQS admittance, as shown in figure 7-9. Figures 7-8 and 7-9 are in good agreement with the experiments carried out in [50].
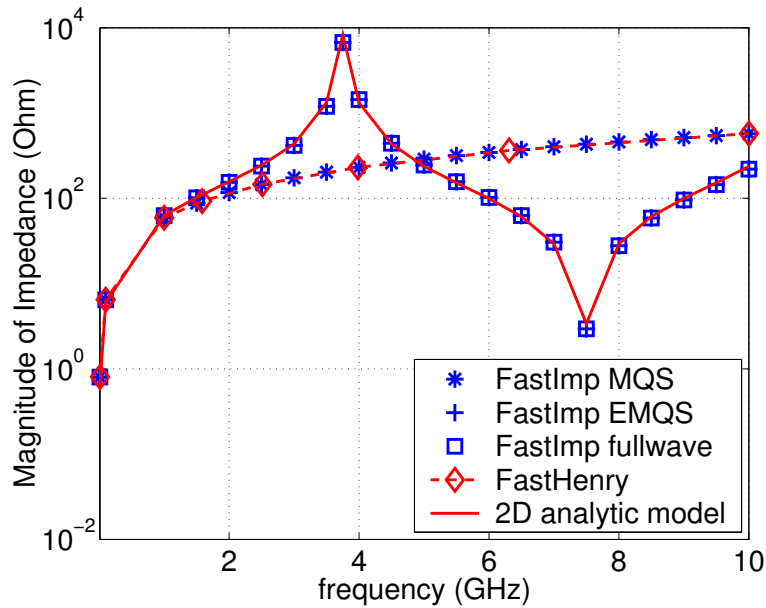
Figure 7-6: Magnitude of the impedance of a shorted transmission line, length is 2cm, separation is 50um

The comparison in figures 7-6, 7-7, 7-8 and 7-9 clearly demonstrates the accuracy of FastImp's MQS, EMQS and fullwave analysis.

7.2.2 Flexibility

A few practical structures are analyzed in this section to demonstrate FastImp's flexibility. The CPU time and memory usage for different examples are compared in table 7.5.

Multiple conductor crossover bus Figure 7-10 shows a multiple conductor bus with three-layer of identical conductors. Each layer has 10 conductors and the conductors on different layer are orthogonal to each other. The size of every conductor is $1 \times 1 \times 25$um. We only extracted one column of the impedance matrix (since this is a multiple port structure) at one frequency point $f = 1$GHz using the EMQS analysis. The CPU time and memory usage are shown in table 7.5.

Stacked spirals over ground The impedance matrix of two stacked 9-turn circular spirals over a lossy ground plane (shown in figure 7-11) and two stacked 8-turn rectangular spirals over a lossy ground plane (shown in figure 7-12) are extracted at one frequency
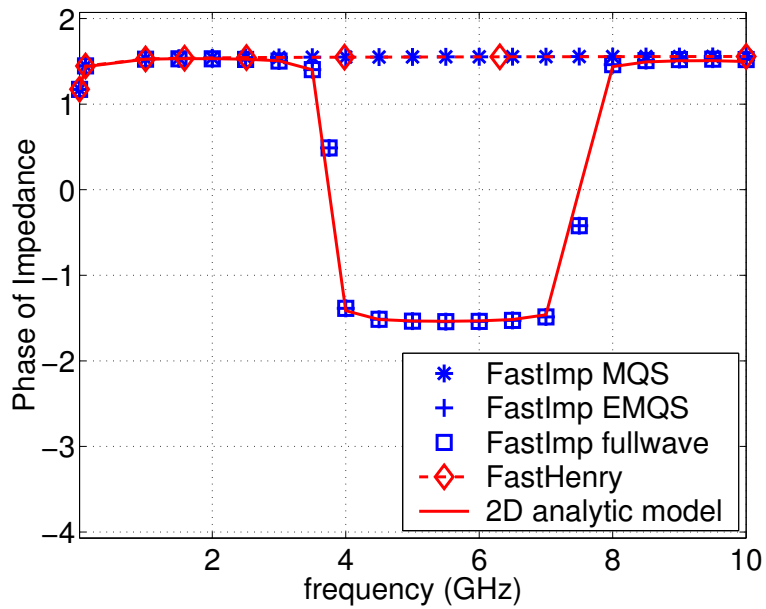
Figure 7-7: Phase of the impedance of a shorted transmission line, length is 2cm, separation is 50um
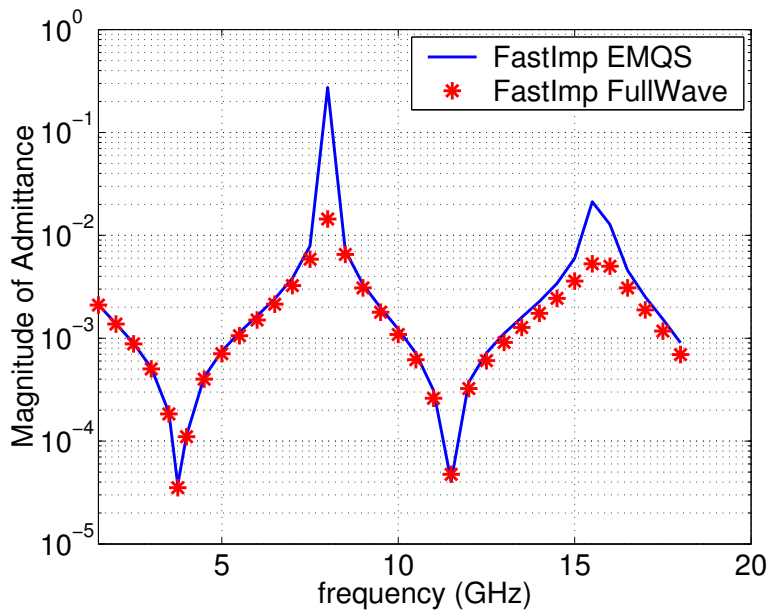
Figure 7-8: Magnitude of the admittance of a shorted transmission line, length is 2cm, separation is 1cm
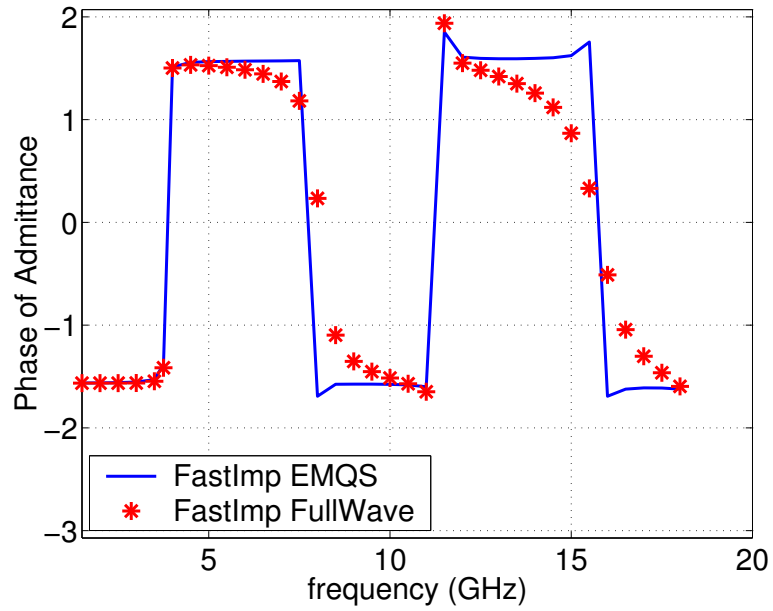
Figure 7-9: Phase of the admittance of a shorted transmission line, length is 2cm, separation is 1cm
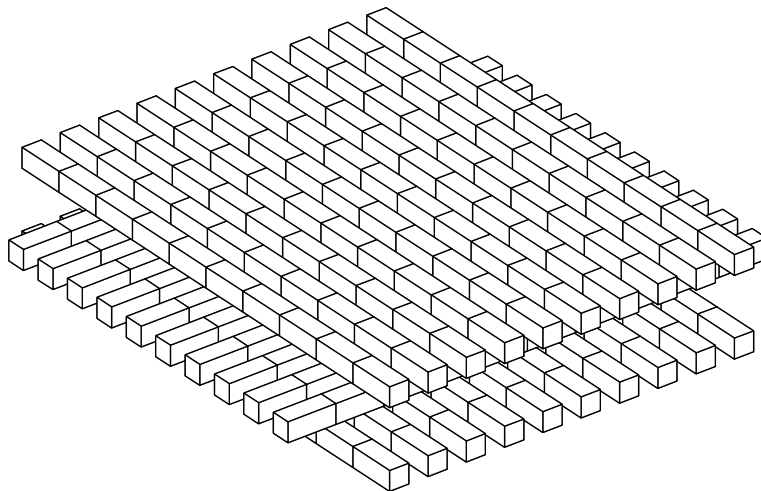
Figure 7-10: Multiple conductor bus
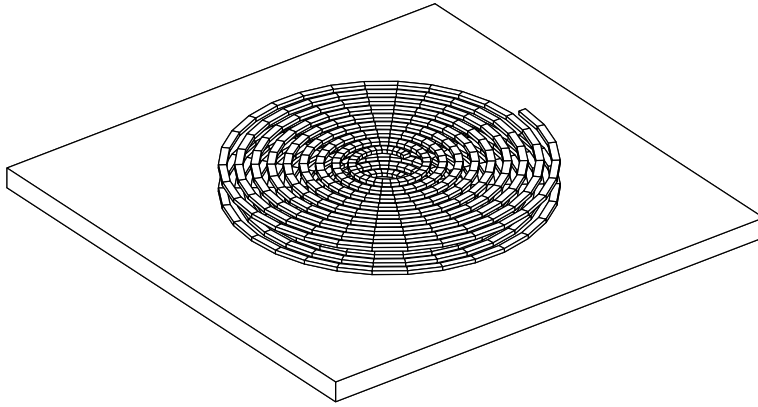
Figure 7-11: Stacked 9-turn circular spirals over ground
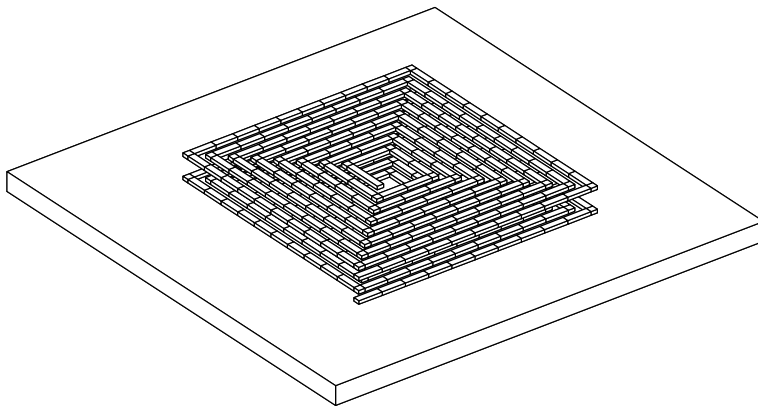
Figure 7-12: Stacked 8-turn rectangular spirals over ground

point $f = 1$GHz using the EMQS analysis. The CPU time and memory usage are shown in table 7.5.

7.2.3 speed

Large 3D structures   FastImp has been used to perform the EMQS analysis of two large structures shown in figure 7-14 and 7-15. Figure 7-14 shows a portion of an RF circuit, which includes five circular spiral inductors of various sizes and number of turns, and two 3D interconnect structures with straight wires and right-angle bends. Figure 7-15 shows a 16×8 array of 3-turn rectangular spirals. The discretization, detailed breakdown of CPU time and memory usage for the analysis of these two examples are shown in table 7.6, 7.7 and 7.8, respectively. The analysis of the 16×8 spiral array in figure 7-15 was carried out

Table 7.5: Comparison of CPU time and memory usage for various practical structures

	bus	circular spirals	rectangle spirals
#panels	18,540	15,194	18,520
#unknowns	148,380	121,558	148,166
FastImp	9min,340Mb	68min,642Mb	54min,749Mb
*iterative	160min,19GB	750min,72GB	590min,83GB
*standard	136days,19GB	100days,19GB	168days,22GB

* obtained by estimation or extrapolation.

Table 7.6: Discretization of the RF interconnect example and the 16x8 spiral array example

	RF interconnect	16x8 spiral array
number of panels	15,566	180,224
number of unknowns	124,574	1,442,048
number of grids	$32 \times 64 \times 16$	$256 \times 128 \times 8$
grid step	3.53um	1.91um

on a server with 32GB memory and one 64-bit Itanium micro-processor. This server is about 3 times slower than the desktop computer used for other examples.

Computational complexity of FastImp   We have used FastImp to analyze a series of similar structures with increasingly larger size. These structures are 1x1, 2x2, 4x4 and 8x8 spiral arrays. All elements in these arrays are 3-turn rectangular spirals. The CPU time versus number of spiral elements in the spiral arrays is shown in figure 7-13. The plot

Table 7.7: A detailed breakdown of the CPU time used by the RF interconnect example and the 16x8 spiral array example. Unit is second

	RF interconnect	16x8 array
P and I matrices	23.5	746
D and H matrices	1444	14353
form the preconditioner P_r	3.61	53
LU factorization of P_r	3.28	1927
GMRES (tol = 1×10^{-3})	369 (48 iter)	25168 (80 iter)
total	1820	42247

Table 7.8: A detailed breakdown of the memory usage for the RF interconnect example and the 16x8 spiral array example. Unit is GB

	RF interconnect	16x8 spiral array
direct matrices	0.593	5.54
projection matrices	0.025	0.39
interpolation matrices	0.015	0.23
convolution matrices	0.013	0.13
maps between grids and panels	0.026	0.70
preconditioner	0.114	2.76
GMRES	0.100	2.21
total	0.886	11.96

clearly indicates that the CPU time grows nearly linearly with the problem size.
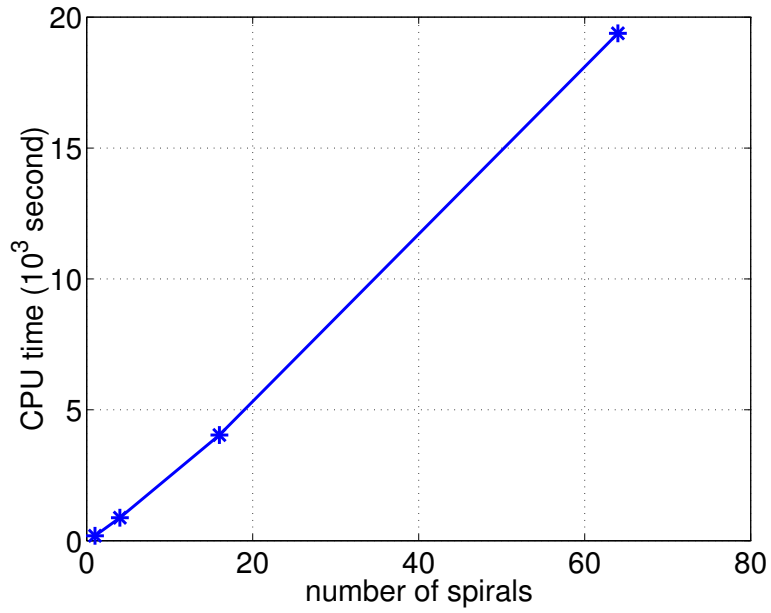
Figure 7-13: The CPU time vs. number of spirals in the spiral arrays
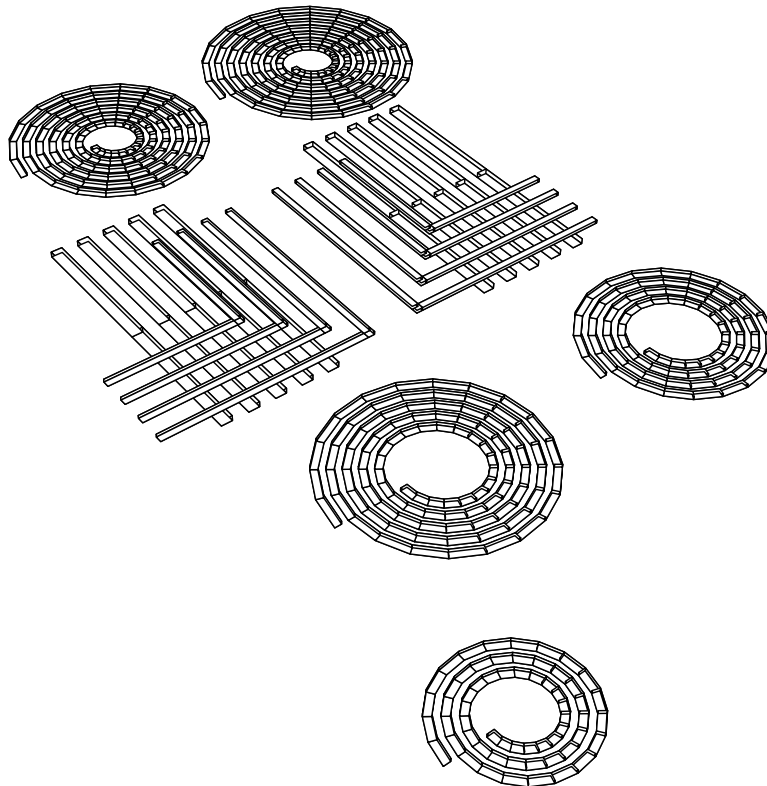
Figure 7-14: A portion of an RF circuit consisting of 5 circular spirals and two pieces of 3D interconnects with straight wires and right-angle bends
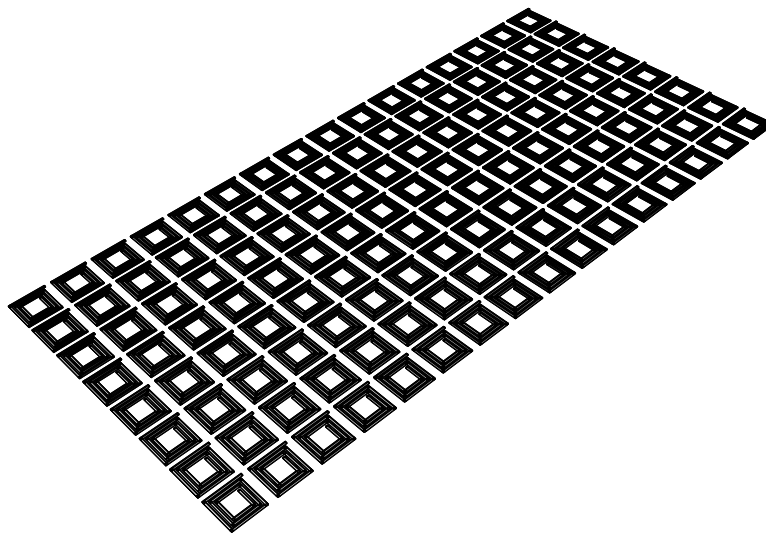
Figure 7-15: 16x8 3-turn rectangular spiral array

Part II

Stochastic Integral Equation Method and Its Application in Capacitance Extraction

Chapter 8

Overview

The effect of surface roughness in the context of electromagnetic radiation and scattering has been studied for at least three decades [45]. A recent extensive review has been made in [107] which cites 275 papers and books. Work on the analysis of rough surface effect falls roughly into two broad categories: approximate analytical techniques [112, 4] and numerical simulation techniques [69, 99]. Since the solutions of the approximate analytical approach are in the explicit analytical forms, it is possible to calculate the mean value and even the variance directly [4]. However, many assumptions have to be made in the approximate analytical techniques, hence their applications are limited. In the numerical simulation approach, the statistical nature of the rough surface model is commonly dealt with using computationally intensive Monte Carlo methods [99]. A few non-Monte-Carlo approaches have been proposed to handle surface roughness differently.

A surface impedance boundary condition was proposed in [43] to take into account the roughness effect. This strategy avoids the discretization of rough surfaces. But only 2D grooves with periodic roughness are analyzed in [43]. More importantly, since the impedance boundary condition depends on the profile of the 2D groove, this approach does not avoid the time-consuming Monte Carlo simulations.

A Stochastic Fourier Transform Approach (SFTA) was proposed in [7, 8] to address the classical problem of how to express the average of the product of an unknown and a known function. It exploits the form of the kernel in the Magnetic Field Integral Equation (MFIE) formulation and hence is not necessarily applicable to the formulations with

different kernels. More importantly, this approach only provides formal solutions of the average scattering field. It is not easy to use the standard numerical techniques to solve the governing equations. Hence it is unclear if the SFTA is computationally more efficient than the Monte Carlo approach.

An ensemble average Green's function idea was proposed in [101], where the mean scattering field in a 2D rough surface acoustic scattering problem was calculated directly without using the Monte Carlo process. In this case, the analytical solution is not readily available. The ensemble average was taken on both sides of the governing integral equation instead of on the analytical solution as in [4]. This leads to an integral equation of the mean scattering field defined only on the smooth surface with the surface roughness removed. Since only a 2D rough surface is considered in [101], it is possible to use analytical techniques such as the Laplace transformation to obtain the solution to the mean field integral equation. A crucial assumption, the uncorrelatedness between source and Green's function, is used in [101] to avoid the above-mentioned classical problem. The justification for this assumption in [101] is based on physical intuition.

In this dissertation, we extend the ensemble average Green's function idea in [101] to the numerical solution of stochastic integral equations for both 2D and 3D structures. We first demonstrate a mathematical interpretation of the uncorrelatedness assumption in [101] and show that it leads to inaccurate results when the surface roughness magnitude is large. We then propose a correction scheme to substantially improve the accuracy. We believe that this scheme strikes a good balance between the method in [101] and in [7, 8] to address the problem of how to express the average of the product of an unknown and a known function, i.e., it is sufficiently accurate and it is compatible with standard numerical techniques for solving integral equations. In addition, we have extended the ensemble average Green's function idea to the calculation of the variance. We use the relatively simple capacitance problems to show that it is possible to directly calculate the mean surface charge density, the mean and variance of the capacitance by just one solve, as oppose to many thousands of solves in Monte Carlo approach. Due to the close connection between the numerical techniques for capacitance extraction and impedance extraction, as demonstrated in [71, 49, 52, 53, 2, 80, 115], we believe that the new method, the Stochastic Integral Equation

(SIE) method, will also be a very useful tool in attacking the more complicated impedance extraction problems in the presence of conductor surface roughness.

Chapter 9

Mathematical Model for Random Rough Surfaces

The rough surface of a realistic interconnect conductor is neither periodic nor explicitly given. In practice they can only be described by their statistical properties. A random rough surface can be described mathematically as $h = F(\vec{r})$, where h is the surface height fluctuation with respect to a nominal smooth surface defined by a mean surface height, and $\vec{r}$ is the position vector on the nominal smooth surface. The function $F(\vec{r})$ is a single-value function of position vector $\vec{r}$, which means there is no overhangs on the surface. The height fluctuation is a stochastic process with respect to the position.

A key characterization of random rough surfaces is the height distribution function, $P_1(h)$. The term $P_1(h)dh$ is the probability of a surface height fluctuation between h and $h + dh$ at any point on the surface. Among many possible distributions, the most commonly used one is the Gaussian distribution [107, 99]

$$P_1(h) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{h^2}{2\sigma^2}), \tag{9.1}$$

where σ is the standard deviation. It describes the fluctuations of surface heights around an average surface height.

The height distribution function only describes the statistical properties of a random rough surface at individual points, it does not reflect the connection between random

heights at different points. So we also need a joint distribution probability density function. The commonly used one is the Gaussian joint distribution defined as [74]

$$P_2(h_1, h_2; \vec{r}_1, \vec{r}_2)) = \frac{exp\left(-\frac{h_1^2 - 2C(\vec{r}_1, \vec{r}_2)h_1 h_2 + h_2^2}{2\sigma^2(1 - C(\vec{r}_1, \vec{r}_2)^2)}\right)}{2\pi\sigma^2\sqrt{1 - C(\vec{r}_1, \vec{r}_2)^2}}, \quad (9.2)$$

where $C(\vec{r}_1, \vec{r}_2)$ is the auto-correlation function. We assume the random rough surface is translation invariant, i.e.,

$$C(\vec{r}_1, \vec{r}_2) = C(|\vec{r}_1 - \vec{r}_2|) = C(\xi). \quad (9.3)$$

The most commonly used auto-correlation function is the Gaussian correlation function [107, 99]

$$C(\xi) = exp\left(-\frac{\xi^2}{\eta^2}\right), \quad (9.4)$$

and the exponential correlation function

$$C(\xi) = exp\left(-\frac{|\xi|}{\eta}\right), \quad (9.5)$$

where η is correlation length, defined as the value of ξ at which

$$C(\eta) = \frac{1}{e}, \quad (9.6)$$

where e is Euler constant. Gaussian correlation function in (9.4) satisfies $\dot{C}(0) = 0$, which means that the surface points very close to each other are almost fully correlated and hence the surface profile, when zoomed in, is smooth. On the other hand, $\dot{C}(0) \neq 0$ holds for the exponential correlation function in (9.5). This implies that the surface profile behaves like a fractal, i.e., it stays rough no matter how close the surface is zoomed in. This is obviously not physical. Therefore, we believe that Gaussian correlation function is better suited in modeling real rough surfaces. Hence it will be used throughout this dissertation.

The height fluctuation defined by (9.1), (9.2) and (9.3) is a stationary Gaussian stochastic process [74]. From above description it is clear that this stochastic process is uniquely

determined by two parameters: the standard deviation σ and the correlation length η. Figure 9-1 shows a random surface profile generated from a stationary Gaussian stochastic process with the Gaussian correlation function in (9.4), where $\sigma = \eta = 0.05mm$.

Here we want to emphasize that the stochastic integral equation (SIE) method developed in this dissertation is not tied to the specific mathematical model for rough surface. A different model for rough surface simply means that different probability density functions like the ones in (9.1) and (9.2) will be used in calculating the ensemble average Green's function. But the basic elements of the SIE method remain the same. Likewise, different rough surface models can be used for different sides of the same conductor.

In order to facilitate the development of the Stochastic Integral Equation (SIE) method and to reduce the CPU time, we have made two assumptions about the rough surface models:

1. Rough surfaces have no over-hang

   Considering the origin of the surface roughness, it is without loss of generality to assume that the surface height fluctuation is always perpendicular to the nominal smooth surface, as shown in figure 9-1. In other words, the surface height fluctuation as a function of the location on the nominal smooth surface is always a single-valued function.

2. Small correlation length

   We assume that the correlation between height fluctuation at two points on the same rough surface decreases quickly with the increase of the distance between these two points. In the case where the rough surfaces are modeled by stationary stochastic processes, this assumption means that the correlation length η in (9.6) is relatively small compare to the feature size of the conductors. We believe that this assumption is reasonable for realistic rough surfaces. Hence the algorithm based on this assumption should be useful for practical applications.
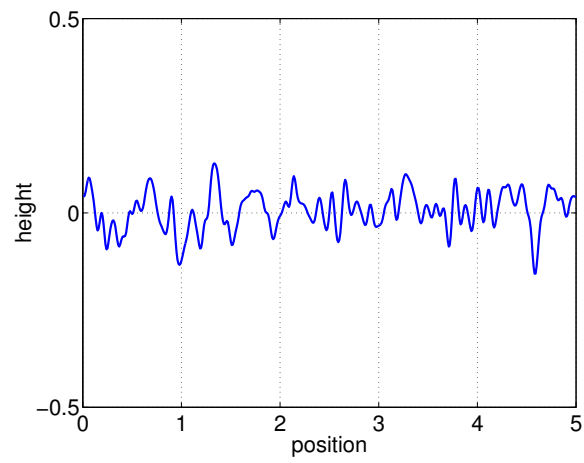
Figure 9-1: The profile of a random rough surface, standard deviation=0.05mm, correlation length=0.05mm

Chapter 10

Stochastic Integral Equation Method

For the sake of clarity, we use a simple 2D capacitance problem, a single conductor over a ground plane, to explain the basic ideas of the stochastic integral equation method. As will be made clear, this method can be readily extended to the multiple conductor case as well as to the 3D capacitance problems with little modification.

Figure 10-1 shows one conductor over an infinite ground plane, where the conductor is denoted as D_1 and the ground plane is denoted as D_0. Without loss of generality, we assume that the side walls of conductor D_1 (denoted as S_1 and S_3) are smooth, only the top and the bottom surfaces (denoted as S_2 and S_4) are rough. The position of the points on the top and the bottom surfaces are defined by

$$\begin{cases} y_1(x) = b + h_1(x) & (x, y_1(x)) \in S_2 \\ y_2(x) = a + h_2(x) & (x, y_2(x)) \in S_4 \end{cases} \tag{10.1}$$

where $h_1(x)$ and $h_1(x)$ are two independent surface height fluctuation functions with statistical characteristics defined by (9.1), (9.2) and (9.3), and with profiles like the one shown in figure 9-1, and a and b are the nominal position of the top and the bottom surface, respectively, as shown in figure 10-1. To facilitate the explanation in the following sections, we also define the smooth nominal surfaces $\tilde{S}_2$ and $\tilde{S}_4$ for rough surfaces S_2 and S_4 as $\{(x, y) \in \tilde{S}_2 | 0 \le x \le c, y = b\}$ and $\{(x, y) \in \tilde{S}_4 | 0 \le x \le c, y = a\}$, respectively.
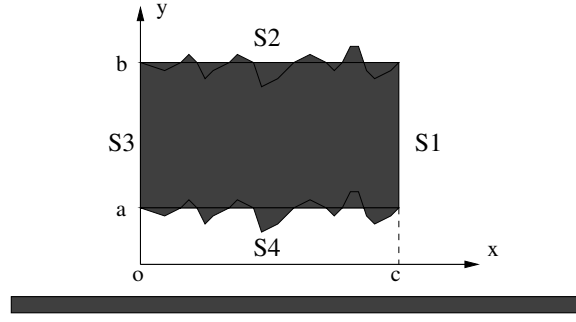
Figure 10-1: One conductor over a ground plane. The top and the bottom surfaces are rough.

10.1 Description of 2D capacitance problem

The 2D capacitance can be calculated by solving a 2D exterior Laplace problem defined as

$$
\begin{cases}
(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})\phi(x,y) = 0 & (x,y) \in D \\
\phi(x,y) = V_i & (x,y) \in \partial D_i \quad i = 0,1 \\
\phi(x,y) = 0 & (x,y) \to \infty
\end{cases}
\tag{10.2}
$$

where D denotes the region outside of conductors, ∂D_i refers to the surface of D_i and V_i is the given voltage on ∂D_i. To compute capacitance, we set $V_1 = 1$ and $V_0 = 0$. Equation (10.2) can be converted to the equivalent integral equation [71]

$$
\int_{\partial D_1} dl(x',y') \frac{\rho(x',y')}{\varepsilon_0} G(x',y';x,y) = V_1 = 1, \quad (x,y) \in \partial D_1,
\tag{10.3}
$$

where

$$
G(x',y';x,y) = \frac{1}{2\pi} \ln \sqrt{\frac{(x-x'')^2 + (y-y'')^2}{(x-x')^2 + (y-y')^2}},
\tag{10.4}
$$

and (x'',y'') is the image of (x',y') with respect to the ground plane. Here we have used the image theory ([81], pp. 48) to take into account the effect of ground plane. So in our calculation, we only need to discretize conductor surface ∂D_1. It should be noted that with the Green's function in (10.4) the boundary conditions at infinity and on the ground plane are satisfied automatically. Using the fact that the rough surface height is a function of position, and combining that with the standard change of variable identity for integrals

yields

$$\int_a^b \frac{\rho(x'=0,y')}{\varepsilon_0} G(x'=0,y';x,y)dy' \ +$$

$$\int_0^c \sqrt{1+(\frac{dy_1(x')}{dx'})^2} \frac{\rho(x',y_1(x'))}{\varepsilon_0} G(x',y_1(x');x,y)dx' \ +$$

$$\int_a^b \frac{\rho(x'=c,y')}{\varepsilon_0} G(x'=c,y';x,y)dy' \ +$$

$$\int_0^c \sqrt{1+(\frac{dy_2(x')}{dx'})^2} \frac{\rho(x',y_2(x'))}{\varepsilon_0} G(x',y_2(x');x,y)dx' \ = \ 1$$

$$(x,y) \in \partial D_1, \qquad (10.5)$$

where the first and the third terms are associated with the two smooth sides and the second and the fourth terms are associated with the rough top and bottom (see figure 10-1). Now define

$$\tilde{\rho}(x',y') = \begin{cases} \rho(x',y'), \quad (x',y') \in S_1,S_3 \\ \sqrt{1+(\frac{dy_1(x')}{dx'})^2}\rho(x',y_1(x')), \quad (x',y') \in \tilde{S}_2 \\ \sqrt{1+(\frac{dy_2(x')}{dx'})^2}\rho(x',y_2(x')), \quad (x',y') \in \tilde{S}_4, \end{cases} \qquad (10.6)$$

$$d\tilde{l}(x',y') = \begin{cases} dx', \quad (x',y') \in \tilde{S}_2,\tilde{S}_4 \\ dy', \quad (x',y') \in S_1,S_3, \end{cases} \qquad (10.7)$$

$$\hat{G}(x',y';x,y) = \begin{cases} G(x',y';x,y), \quad (x',y') \in S_1,S_3 \\ G(x',y_1(x');x,y), \quad (x',y') \in \tilde{S}_2 \\ G(x',y_2(x');x,y), \quad (x',y') \in \tilde{S}_4, \end{cases} \qquad (10.8)$$

then equation (10.5) can be written as

$$\int_{\partial \tilde{D}_1} \frac{\tilde{\rho}(x',y')}{\varepsilon_0} \hat{G}(x',y';x,y)d\tilde{l}(x',y') = 1, \quad (x,y) \in \partial D_1, \qquad (10.9)$$

where $\partial \tilde{D}_1$ is the nominal smooth surface. It should be pointed out that with the change of variable defined in (10.6) and (10.7), the unknown charge $\tilde{\rho}$ is define on the nominal smooth surface $\partial \tilde{D}_1$ and the integral domain of equation (10.9) becomes $\partial \tilde{D}_1$. This makes it much

99

easier to use the standard definition of stochastic integral [74] in the following sections. [1]
In view of (10.6) and (10.7), the self capacitance is

$$C = \int_{\partial D_1} \rho(x',y')dl(x',y') = \int_{\partial \tilde{D}_1} \tilde{\rho}(x',y')d\tilde{l}(x',y'). \tag{10.10}$$

It should be noted that the charge density distribution is a nonlinear function of the surface
point location, as shown in (10.3). This implies that the capacitance is also a nonlinear
function of the surface point location. Hence the mean capacitance is not equal to the
capacitance for the conductor with a nominal smooth surface. This should not be surprising
because the rough surface conductor's surface area is larger. As shown in [75] as well as in
our numerical result section, this difference is not negligible.

10.2   Basic ideas

Instead of solving equation (10.9) for many statistically independent realizations of the
rough surface and taking the ensemble average of charge density, we derive the integral
equation for the mean charge density directly. By solving this stochastic integral equation,
we can easily calculate the mean capacitance.

Taking the ensemble average on both sides of equation (10.9) yields

$$\int_{\partial \tilde{D}_1} d\tilde{l}(x',y') < \frac{\tilde{\rho}(x',y')}{\varepsilon_0} \tilde{G}(x',y';x,y) >= 1, \quad (x,y) \in \partial \tilde{D}_1 \tag{10.11}$$

where

$$\tilde{G}(x',y';x,y) = \begin{cases} \hat{G}(x',y';x,y), & (x,y) \in S_1, S_3 \\ \hat{G}(x',y';x,y_1(x)), & (x,y) \in \tilde{S}_2 \\ \hat{G}(x',y';x,y_2(x)), & (x,y) \in \tilde{S}_4, \end{cases} \tag{10.12}$$

and the angle brackets $<>$ stand for ensemble average. Assuming that the charge den-
sity distribution is uncorrelated to Green's function, as is done in [101], equation (10.11)

---

[1]The stochastic processes $y_1(x')$ and $y_2(x')$ in (10.6) are differentiable because the autocorrelation of $h_1(x)$
and $h_2(x)$ in (10.1), which is defined in (9.4), has derivative of order up to at least two [74].

becomes

$$\int_{\partial \tilde{D}_1} d\tilde{l}(x',y') \frac{< \tilde{\rho}(x',y') >}{\varepsilon_0} < \tilde{G}(x',y';x,y) >= 1, \quad (x,y) \in \partial \tilde{D}_1. \tag{10.13}$$

In section 10.6 we will explain the significance of this uncorrelatedness assumption and show that there is a way to compensate for the error introduced by this approximation. In view of (10.10) the mean self capacitance is

$$< C >= \int_{\partial \tilde{D}_1} d\tilde{l}(x',y') < \tilde{\rho}(x',y') > . \tag{10.14}$$

It is clear that $< \tilde{\rho}(x',y') >$ is exactly what we want to compute, and it is treated as the unknown variable.

It should be noted that the surface roughness is not explicit in the ensemble average Green's function any more, hence only the smooth reference surface $\partial \tilde{D}_1$ needs to be discretized and only one solve is needed to obtain the mean charge density. Equation (10.13) can be solved numerically using the standard methods [35].

10.3 Discretization of stochastic integral equation

We use piecewise constant basis function and the Galerkin method to discretize equation (10.13). The centroid collocation method can be regarded as a special case of Galerkin method with one quadrature point for the testing integral [35]. The discretized system for (10.13) is

$$[\bar{A}] < \tilde{\rho} >= \tilde{L}, \tag{10.15}$$

where

$$\bar{A}_{k,j} = \int_{\tilde{\Delta}_k} d\tilde{l}(x,y) \int_{\tilde{\Delta}_j} d\tilde{l}(x',y') < \tilde{G}(x',y';x,y) >, \tag{10.16}$$

$$\tilde{L}_k = \int_{\tilde{\Delta}_k} d\tilde{l}(x,y), \tag{10.17}$$

$< \tilde{G}(x',y';x,y) >$ is in one of the forms shown in appendix A, and $\tilde{\Delta}_k$ and $\tilde{\Delta}_j$ are the k-th and the j-th panel on the nominal smooth surface. Here a few representative forms, namely,

$K_1(x',y';x,y)$, $K_2(x',b;x,y)$, $K_6(x',b;x,b)$ and $K_8(x',a;x,b)$ are used to show how the panel integration with the ensemble average Green's function is computed.

Substituting (A.1) into (10.16) and considering (10.7), we obtain

$$\bar{A}_{k,j} = \int_{\tilde{\Delta}_k} dy \int_{\tilde{\Delta}_j} dy' K_1(x',y';x,y) = \int_{\tilde{\Delta}_k} dy \int_{\tilde{\Delta}_j} dy' G(x',y';x,y), \tag{10.18}$$

where $x' = 0$ or $x' = c$, and $x = 0$ or $x = c$. Substituting (A.2) into (10.16) and considering (10.7), we obtain

$$\begin{aligned}
\bar{A}_{k,j} &= \int_{\tilde{\Delta}_k} dy \int_{\tilde{\Delta}_j} dx' K_2(x',b;x,y) \\
&= \int_{\tilde{\Delta}_k} dy \int_{\tilde{\Delta}_j} dx' \int_{-\infty}^{+\infty} dh_1 P_1(h_1) G(x',b+h_1;x,y) \\
&= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{\tilde{\Delta}_k} dy \int_{\tilde{\Delta}_j} dx' G(x',b+h_1;x,y)
\end{aligned} \tag{10.19}$$

where $x = 0$ or $x = c$. Substitution of (A.7) into (10.16) leads to

$$\begin{aligned}
\bar{A}_{k,j} &= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' K_6(x',a;x,b) \\
&= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) G(x',a+h_1;x,b+h_2) \\
&= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' G(x',a+h_1;x,b+h_2).
\end{aligned}$$
$$\tag{10.20}$$

Substitution of (A.9) into (10.16) leads to

$$\begin{aligned}
\bar{A}_{k,j} &= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' K_8(x',b;x,b) \\
&= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1,h_2;x',x) G(x',b+h_1;x,b+h_2) \\
&\simeq \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1,h_2;x_j,x_k) \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_j} dx' G(x',b+h_1;x,b+h_2),
\end{aligned}$$
$$\tag{10.21}$$

where (x_j,b) and (x_k,b) are respectively the centroid of the j-th and the k-th panel on $\tilde{S}_2$. Since the correlation coefficient (9.3) is a function of the distance between evaluation point

and source point, $P_2(h_1,h_2;x',x)$ should be a function x' and x. Hence strictly speaking, one can not interchange the integration order of the ensemble average integral and panel integrals. Hence the last approximate equality in (10.21) deserves some explanation.

It is clear from (9.3) and (9.4) that $C(|x'-x|) < 0.001$ when $|x'-x| > 3\eta$. Therefore, in view of (9.2), when the distance between the j-th panel and the k-th panel is bigger than 3η, we have

$$P_2(h_1,h_2;x',x) \simeq P_1(h_1)P_1(h_2). \qquad (10.22)$$

Hence the right-hand side of the second equal sign in (10.21) is almost same as the right-hand side of the second equal sign in (10.20), and approximation in the last equality of (10.21) introduces very little error. Fortunately, the distance between most panels is bigger than 3η. When $|x'-x| < 3\eta$, the approximation introduced in the last equal sign of (10.21) becomes questionable. However, using $P_2(h_1,h_2;x',x) \simeq P_2(h_1,h_2;x_j,x_k)$ in (10.21) is tantamount to using a staircase piecewise constant function $C(|x_j-x_k|)$ with a step size equal to the j-th panel size along x' and a step size equal to the k-th panel size along x to approximate the continuous correlation function $C(|x'-x|)$ for $x' \in panel_j$ and $x \in panel_k$. This is certainly acceptable when the j-th and the k-th panel are small. As a rule-of-thumb, the panel size should be smaller than half of the correlation length η.

We use the standard Gauss Hermite quadrature to calculate the outer ensemble average integral in (10.19), (10.20) and (10.21) and the standard Gauss Legendre quadrature to compute the integral over the k-th panel in (10.18), (10.19), (10.20) and (10.21) and use the analytical formula in [109] to calculate the inner integral over the j-th panel. It should be pointed out that the advantage of exchanging integration order in (10.19), (10.20) and (10.21) is that the inner integral over the j-th panel with regular Green's function can be computed efficiently using methods developed for panel integration in boundary element method [109, 41, 114].

10.4   Conditioning of the system matrix

At first glance, the ensemble average Green's function seems to be less singular than the original Green's function due to the ensemble average integral in $\tilde{G}$, as shown in appendix A. Since the singularity of the Green's function is the key factor to keep the condition number of the system matrix from growing too fast with the refinement of the mesh [12], one might think the conditioning of the system matrix of the stochastic integral equation is worse than that of the corresponding smooth surface system matrix. In this section we show that the diagonal elements of the system matrix for both the stochastic integral equation and the regular integral equation are approximately equal. This implies that the order of the singularity of the ensemble average Green's function is the same as that of the original Green's function.

The diagonal elements in the system matrix is expressed as (10.18) or (10.21), where $k = j$. Since the panel integration in (10.18) is the same for both the ensemble average Green's function and the regular Green's function, the diagonal elements are obviously the same when both the source panel and the evaluation panel are on the smooth side walls S_1 or S_3. However, the panel integration in (10.21) is different from that of the smooth surface problem. Since the source panel and the evaluation panel coincide, the two-dimensional ensemble average integral in (10.21) degenerates to one-dimensional integral. Hence we have

$$
\begin{aligned}
\bar{A}_{k,k} &= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' K_8(x',b;x,b) \\
&= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' \int_{-\infty}^{+\infty} dh P_1(h) G(x',b+h;x,b+h) \\
&= \int_{-\infty}^{+\infty} dh \left\{ P_1(h) \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' G(x',b+h;x,b+h) \right\} \\
&= \int_{-\infty}^{+\infty} dh \left\{ P_1(h) \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' G(x',b;x,b) \right\} \\
&= \left\{ \int_{-\infty}^{+\infty} dh P_1(h) \right\} \left\{ \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' G(x',b;x,b) \right\} \\
&= \int_{\tilde{\Delta}_k} dx \int_{\tilde{\Delta}_k} dx' G(x',b;x,b),
\end{aligned}
\tag{10.23}
$$
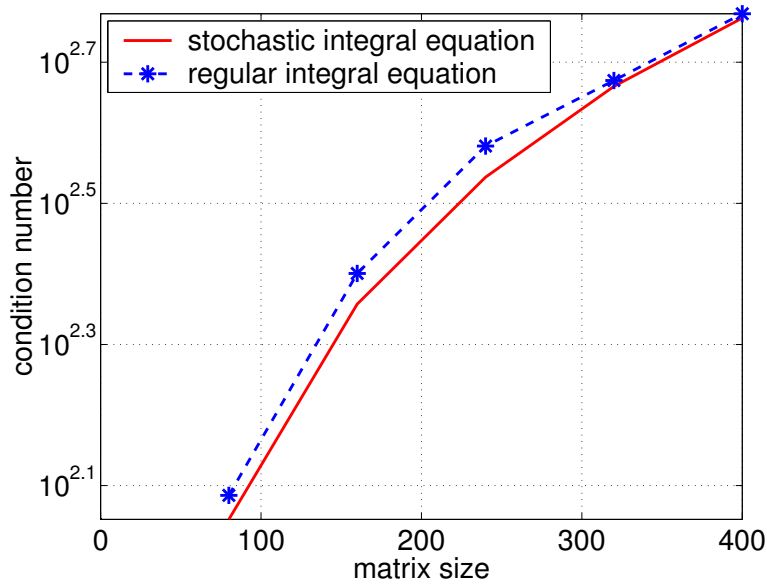
Figure 10-2: The condition number of the system matrix with different sizes, generated by stochastic integral equation method and the regular integral equation method

where the fourth equality is due to the translation invariance property of the Green's function and the last equality follows from the definition of probability distribution. Now we have shown that the diagonal elements in the system matrix generated from either the ensemble average Green's function or the regular Green's function on the smooth reference surface are approximately equal. Hence the singularity order of the ensemble average Green's function is approximately the same as that of the Green's function for the corresponding smooth surface problem. This implies that the condition number of the system matrix generated by the stochastic integral equation method is no worse than that of the system matrix for the corresponding smooth surface problem.

We use a simple numerical experiment to verify our reasoning. We use both the regular integral equation method and the stochastic one to set up the system matrix for the 2D capacitance problem of a single conductor over an infinite ground plane, as shown in figure 10-1. The condition number of the two system matrices is compared in figure 10-2. It is clear that the two matrices have roughly the same condition number for different matrix sizes.

10.5 Translation invariance of the ensemble average Green's function

An important property of the regular Green's functions for the capacitance extraction problems or the impedance extraction problems is that they are translation invariant. This makes it possible to use FFT-based methods like the Precorrected-FFT method [80, 115] to accelerate the time-consuming matrix vector products in an iterative linear system solver. In this section we show that the ensemble average does not change this property. Though we only use the Green's function for 2D capacitance problems to illustrate this, the extension to the Green's functions for 3D capacitance problems or impedance problems is straightforward.

We first want to establish that the various forms in appendix A can be written in more compact forms. As mentioned in chapter 9, we asume that the rough surfaces have no overhang, i.e., the surface height fluctuation is either along x direction or along y direction, but not along both directions. Denote the mean position of the source point and the evaluation point on the nominal smooth surface as (x'_0, y'_0) and (x_0, y_0), we have four possible forms for the ensemble average Green's function:

$$
\tilde{K}_1(x'_0, y'_0; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1, h_2; y'_0, y_0) G(x'_0 + h_1, y'_0; x_0 + h_2, y_0)
$$

$$(10.24)$$

$$
\tilde{K}_2(x'_0, y'_0; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1, h_2; y'_0, x_0) G(x'_0 + h_1, y'_0; x_0, y_0 + h_2)
$$

$$(10.25)$$

$$
\tilde{K}_3(x'_0, y'_0; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1, h_2; x'_0, y_0) G(x'_0, y'_0 + h_1; x_0 + h_2, y_0)
$$

$$(10.26)$$

$$
\tilde{K}_4(x'_0, y'_0; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1, h_2; x'_0, x_0) G(x'_0, y'_0 + h_1; x_0, y_0 + h_2).
$$

$$(10.27)$$

If the source point and the evaluation point are on different rough surfaces, then the correlation coefficient in (9.3) is zero. Since the source point and the evaluation point in (10.25) and (10.26) are randomly perturbed along different directions, they can not be on the same rough surface, due to our initial assumption. Hence these two equations can be simplified as

$$
\tilde{K}_2(x_0', y_0'; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) G(x_0' + h_1, y_0'; x_0, y_0 + h_2)
$$

$$(10.28)$$

$$
\tilde{K}_3(x_0', y_0'; x_0, y_0)
$$
$$
= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) G(x_0', y_0' + h_1; x_0 + h_2, y_0).
$$

$$(10.29)$$

It should be noted that if a rough surface degenerates to a smooth surface then the corresponding roughness distribution becomes $P_1(h) = \delta(h)$, where h is either h_1 or h_2 and the corresponding ensemble average integral becomes the evaluation of the integrand at $h = 0$. It can be checked that kernel in (A.1) is a special case of (10.24), kernels in (A.2) and (A.3) are special cases of (10.29), kernels in (A.4) and (A.5) are special cases of (10.28), and kernels in (A.7)-(A.10) are special cases of (10.27).

Now we have shown that all nine cases in appendix A can be written in one of the four compact forms in (10.24), (10.28), (10.29) and (10.27). Since the original Green's function is translation invariant, and the joint distribution $P_2(h_1, h_2; x_0', x_0)$ and $P_2(h_1, h_2; y_0', y_0)$ in (10.24) and (10.27) are also translation invariant in terms of (x_0', x_0) and (y_0', y_0), respectively, the translation invariance of $\tilde{K}_i(i = 1, 2, 3, 4)$ in (10.24), (10.28), (10.29) and (10.27) easily follows. Hence the ensemble average Green's function is translation invariant.

It should be noted that the compact forms in (10.24), (10.28), (10.29) and (10.27) also cover the cases where all four sides of a conductor are rough. Hence the conclusion made in this section is rather general. In addition, to get these general forms for 3D cases, we just

need to replace *x* and *y* in (10.24), (10.28), (10.29) and (10.27) with normal and tangent direction variable in the local coordinate system on 3D surface. The key is to make sure that the height fluctuation is along the direction normal to the 3D surface. Hence it is rather straightforward to extend the ensemble average Green's function and the Stochastic Integral Equation method to 3D cases.

10.6 Second order correction to the uncorrelatedness assumption

In this section we show that the solution of (10.15) is only a zero-th order approximation to the correct mean charge density. Hence we will call it $< \tilde{\rho}^{(0)} >$ in the remaining part of the paper and we have

$$< \tilde{\rho}^{(0)} >= \bar{A}^{-1}\tilde{L}. \tag{10.30}$$

As shown in appendix B, the discretization of (10.9) on each realization of the rough surface results in

$$[A]\tilde{\rho} = \tilde{L}, \tag{10.31}$$

or equivalently,

$$A\tilde{\rho} = (\bar{A}+A-\bar{A})\tilde{\rho} = \bar{A}[I+\bar{A}^{-1}(A-\bar{A})]\tilde{\rho} = \tilde{L}. \tag{10.32}$$

Therefore,

$$\tilde{\rho} = [I+\bar{A}^{-1}(A-\bar{A})]^{-1}\bar{A}^{-1}\tilde{L}. \tag{10.33}$$

Using the Taylor expansion in (10.33) and in view of (10.30), we obtain

$$\begin{aligned} \tilde{\rho} &\simeq [I-\bar{A}^{-1}(A-\bar{A})+\bar{A}^{-1}(A-\bar{A})\bar{A}^{-1}(A-\bar{A})] < \tilde{\rho}^{(0)} > \\ &= \tilde{\rho}^{(0)}+\tilde{\rho}^{(1)}+\tilde{\rho}^{(2)}, \end{aligned} \tag{10.34}$$

where

$$\tilde{\rho}^{(0)} =< \tilde{\rho}^{(0)} >, \tag{10.35}$$

108

$$\tilde{\rho}^{(1)} = -\bar{A}^{-1}(A - \bar{A}) < \tilde{\rho}^{(0)} >, \tag{10.36}$$

and

$$\tilde{\rho}^{(2)} = \bar{A}^{-1}(A - \bar{A})\bar{A}^{-1}(A - \bar{A}) < \tilde{\rho}^{(0)} > . \tag{10.37}$$

The approximation in (10.34) is due to the truncated Taylor expansion and the leading term in the truncation error is a third order term

$$\tilde{\rho}^{(3)} = \bar{A}^{-1}((A - \bar{A})\bar{A}^{-1})^2(A - \bar{A}) < \tilde{\rho}^{(0)} > . \tag{10.38}$$

Take the ensemble average on both side of (10.34), we obtain

$$< \tilde{\rho} > \ \simeq \ < \tilde{\rho}^{(0)} > + < \tilde{\rho}^{(2)} >, \tag{10.39}$$

where the elimination of the term $< \tilde{\rho}^{(1)} >$ is due to

$$< \tilde{\rho}^{(1)} > = -\bar{A}^{-1} < (A - \bar{A}) > < \tilde{\rho}^{(0)} > = 0. \tag{10.40}$$

Now it is clear that the uncorrelatedness assumption in section 10.2 only gives us the zero-th order term $< \tilde{\rho}^{(0)} >$ in (10.39). Hence its accuracy depends largely on the size of the deviation of each matrix A from $\bar{A}$. In other words, it depends on the magnitude of the surface roughness. [2] In the numerical result section we show that this is indeed the case and that the second order correction term improves the accuracy significantly.

The difficulty in (10.39) is that in order to obtain the correction term $< \tilde{\rho}^{(2)} >$ we need

---

[2] One slightly different perspective is as the following. Similar as in (10.32), we have

$$A = \bar{A} + A - \bar{A} = \bar{A}[I + \bar{A}^{-1}(A - \bar{A})]. \tag{10.41}$$

Taking matrix inverse on both sides, using the same Taylor expansion in (10.34) and then taking the ensemble average on both sides, we obtain

$$< A^{-1} > \simeq \bar{A}^{-1} + \bar{A}^{-1} < (A - \bar{A})\bar{A}^{-1}(A - \bar{A}) > \bar{A}^{-1}. \tag{10.42}$$

Therefore, the uncorrelatedness assumption in section 10.2 is equivalent to assuming $< A^{-1} > \simeq \bar{A}^{-1} = < A >^{-1}$, which is certainly questionable when the deviation of each matrix A from $\bar{A}$ is large.

to compute $< (A - \bar{A})\bar{A}^{-1}(A - \bar{A}) >$. In the following, we will use the Kronecker product $\otimes$ and its properties defined in appendix C to show that it can be computed efficiently.

As explained in section 10.1 and 10.2 both $< \tilde{\rho} >$ and $< \tilde{\rho}^{(0)} >$ are defined on the smooth reference surface. In view of (10.39), the average capacitance is

$$
\begin{aligned}
< C > &= \tilde{L}^T < \tilde{\rho} > \simeq \tilde{L}^T (< \tilde{\rho}^{(0)} > + < \tilde{\rho}^{(2)} >) \\
&= < C^{(0)} > + < C^{(2)} > .
\end{aligned} \tag{10.43}
$$

where

$$
\begin{aligned}
< C^{(2)} > &= \tilde{L}^T < \tilde{\rho}^{(2)} > \\
&= \tilde{L}^T \bar{A}^{-1} < (A - \bar{A})\bar{A}^{-1}(A - \bar{A}) > \bar{A}^{-1} \tilde{L} \\
&\simeq < \tilde{\rho}^{(0)} >^T < E \bar{A}^{-1} E > < \tilde{\rho}^{(0)} >
\end{aligned} \tag{10.44}
$$

and

$$
E = A - \bar{A}. \tag{10.45}
$$

The last approximate equality in (10.44) would be exact if $\bar{A}^T = \bar{A}$, i.e., if the Galerkin method is used to generate $\bar{A}$. If the collocation method is used, then some error is introduced. But numerical experience shows that this is usually not very significant. In view of the identities in appendix C, (10.44) can be writen as

$$
\begin{aligned}
< C^{(2)} > &\simeq < (< \tilde{\rho}^{(0)} >^T E)\bar{A}^{-1}(E < \tilde{\rho}^{(0)} >) > \\
&= < (< \tilde{\rho}^{(0)} >^T E^T) \otimes (< \tilde{\rho}^{(0)} >^T E) > vec(\bar{A}^{-1}) \\
&= (< \tilde{\rho}^{(0)} >^T \otimes < \tilde{\rho}^{(0)} >^T) < E^T \otimes E > vec(\bar{A}^{-1}) \\
&= (vec(B))^T vec(\bar{A}^{-1}) = trace(\bar{A}^{-T} B).
\end{aligned} \tag{10.46}
$$

where

$$
vec(B) = [(< \tilde{\rho}^{(0)} >^T \otimes < \tilde{\rho}^{(0)} >^T) < E^T \otimes E >]^T
$$

$$
\begin{aligned}
&= \; <E \otimes E^T> (<\tilde{\rho}^{(0)}> \otimes <\tilde{\rho}^{(0)}>) \\
&= \; [F](<\tilde{\rho}^{(0)}> \otimes <\tilde{\rho}^{(0)}>) \tag{10.47}
\end{aligned}
$$

$$
F = [F^{ij}]_{N \times N} = \; <
\begin{bmatrix}
E_{11}E^T & E_{12}E^T & \cdots & E_{1N}E^T \\
E_{21}E^T & E_{22}E^T & \cdots & E_{2N}E^T \\
\vdots & \vdots & \ddots & \vdots \\
E_{N1}E^T & E_{N2}E^T & \cdots & E_{NN}E^T
\end{bmatrix}
> \tag{10.48}
$$

$$
\begin{aligned}
F_{nm}^{ij} &= \; <E_{ij}E_{mn}> \\
&= \; <(A_{ij} - \bar{A}_{ij})(A_{mn} - \bar{A}_{mn})> \\
&= \; <A_{ij}A_{mn}> - \bar{A}_{ij}\bar{A}_{mn}. \tag{10.49}
\end{aligned}
$$

For the definition of *vec* operator, see (C.2) in appendix C. In equation (10.46), the second equality is due to (C.6), the third equality is due to (C.3) and the last equality is due to (C.8).

10.7 Variance of capacitance

The algorithm in previous sections only gives us the mean capacitance. In this section we show that the variance of the capacitance can be calculated by using the same basic ideas and the same ensemble average Green's function. More importantly, we show that no extra computation work is necessary to obtain the approximate value of the capacitance variance if the second-order correction in section 10.6 has already been used to calculate the mean capacitance.

In view of (10.10), we have

$$
\begin{aligned}
C^2 &= \int_{\partial \tilde{D}_1} d\tilde{l}(x,y) \tilde{\rho}(x,y) \int_{\partial \tilde{D}_1} d\tilde{l}(x',y') \tilde{\rho}(x',y') \\
&= \int_{\partial \tilde{D}_1} d\tilde{l}(x,y) \int_{\partial \tilde{D}_1} d\tilde{l}(x',y') \tilde{\rho}(x,y) \tilde{\rho}(x',y')
\end{aligned}
$$

and hence

$$< C^2 >= \int_{\partial \tilde{D}_1} d\tilde{l}(x,y) \int_{\partial \tilde{D}_1} d\tilde{l}(x',y') < \tilde{\rho}(x,y)\tilde{\rho}(x',y') > . \qquad (10.51)$$

The variance of the capacitance is

$$Var\{C\} =< C^2 > - < C >^2 . \qquad (10.52)$$

If the discretization scheme in section 10.3 is used, (10.51) becomes

$$< C^2 > \simeq \tilde{L}^T \begin{bmatrix} < \tilde{\rho}_1 \tilde{\rho}_1 > & < \tilde{\rho}_1 \tilde{\rho}_2 > & \cdots & < \tilde{\rho}_1 \tilde{\rho}_N > \\ < \tilde{\rho}_2 \tilde{\rho}_1 > & < \tilde{\rho}_2 \tilde{\rho}_2 > & \cdots & < \tilde{\rho}_2 \tilde{\rho}_N > \\ \vdots & \vdots & \ddots & \vdots \\ < \tilde{\rho}_N \tilde{\rho}_1 > & < \tilde{\rho}_N \tilde{\rho}_2 > & \cdots & < \tilde{\rho}_N \tilde{\rho}_N > \end{bmatrix} \tilde{L}$$
$$= \tilde{L}^T < \tilde{\rho}\tilde{\rho}^T > \tilde{L}, \qquad (10.53)$$

where N is the number of panels on the conductor surface $\partial \tilde{D}_1$. Substitution of (10.34) into (10.53) results in

$$< C^2 > \simeq \tilde{L}^T \sum_{i=0}^{N_1} \sum_{j=0}^{N_2} < \tilde{\rho}^{(i)} (\tilde{\rho}^{(j)})^T > \tilde{L}, \qquad (10.54)$$

where $N_1 \leq 2$ and $N_2 \leq 2$. In view of (10.43), we have

$$< C >^2 \simeq \sum_{i=0}^{N_1} \sum_{j=0}^{N_2} < C^{(i)} >< C^{(j)} >, \ < C^{(1)} >= 0. \qquad (10.55)$$

It is well-known that the accuracy of the double Taylor expansion in (10.54) and (10.55) depends not just on $N_1 + N_2$, but also on if all the terms with order below $N_1 + N_2$ are included. [3] In the following, we discuss various options and their effect on the accuracy of

---

[3]Suppose we have two polynomials $p_1(x) = 1 + x + x^2 + x^3$ and $p_2(y) = 1 + y + y^2 + y^3$, then $p(x,y) = p_1(x)p_2(y) = 1 + (x+y) + (xy + x^2 + y^2) + (x^2 y + xy^2 + x^3 + y^3) + (xy^3 + x^3 y + x^2 y^2) + (x^2 y^3 + x^3 y^2)$. If we truncate p(x,y) up to the third order terms, then we have $\tilde{p}(x,y) = 1 + (x+y) + (xy + x^2 + y^2) + (x^2 y + xy^2 + x^3 + y^3)$ and the truncation error is in the fourth order. However, if we truncate $p_1(x)$ and $p_2(y)$ to the second

$Var\{C\}$. To maitain the same order in truncation error, we use the same $N_1 + N_2$ in (10.54) and (10.55).

1. $N_1 + N_2 = 0$ in (10.54) and (10.55)

$$< C^2 > \quad \simeq \quad \tilde{L}^T < \tilde{\rho}^{(0)} (\tilde{\rho}^{(0)})^T > \tilde{L} = < C^{(0)} >^2 \qquad (10.56)$$

$$Var\{C\} = < C^2 > - < C^{(0)} >^2 \simeq < C^{(0)} >^2 - < C^{(0)} >^2 = 0. \qquad (10.57)$$

This obviously unreasonable result suggests that keeping just the zero-th order term in (10.54) is inadequate for capacitance variance calculation. This is consistent with the observation made after (10.39).

2. $N_1 + N_2 = 1$ in (10.54) and (10.55)

$$\begin{aligned} < C^2 > \quad &\simeq \quad \tilde{L}^T (< \tilde{\rho}^{(0)} > < \tilde{\rho}^{(0)} >^T + < \tilde{\rho}^{(0)} > < \tilde{\rho}^{(1)} >^T \\ &\quad + < \tilde{\rho}^{(1)} > < \tilde{\rho}^{(0)} >^T) \tilde{L} \\ &= \quad \tilde{L}^T < \tilde{\rho}^{(0)} > < \tilde{\rho}^{(0)} >^T \tilde{L} = < C^{(0)} >^2, \end{aligned} \qquad (10.58)$$

where the second equal sign is due to (10.40). So we have

$$Var\{C\} = < C^2 > - < C^{(0)} >^2 \simeq < C^{(0)} >^2 - < C^{(0)} >^2 = 0. \qquad (10.59)$$

Same as before, this result is not acceptable.

3. $N_1 + N_2 = 2$ in (10.54) and (10.55)

$$\begin{aligned} < C^2 > \quad &\simeq \quad \tilde{L}^T < \tilde{\rho}^{(0)} > (< \tilde{\rho}^{(0)}) >^T + < \tilde{\rho}^{(1)} >^T + < \tilde{\rho}^{(2)} >^T) \tilde{L} \\ &\quad + \tilde{L}^T (< \tilde{\rho}^{(1)} > < \tilde{\rho}^{(0)} >^T + < \tilde{\rho}^{(1)} (\tilde{\rho}^{(1)})^T >) \tilde{L} \\ &\quad + \tilde{L}^T < \tilde{\rho}^{(2)} > < \tilde{\rho}^{(0)} >^T \tilde{L} \end{aligned}$$

order, then we have $\hat{p}(x,y) = 1 + (x+y) + (xy + x^2 + y^2) + (x^2y + xy^2)$. Notice that the two terms x^3 and y^3 are missing in $\hat{p}(x,y)$, comparing to $\tilde{p}(x,y)$. This suggests that truncation error in $\hat{p}(x,y)$ is in the third order, one order lower than that of $\tilde{p}(x,y)$.

113

$$\begin{aligned}
&= \; <C^{(0)}><C>+\tilde{L}^T<\tilde{\rho}^{(1)}(\tilde{\rho}^{(1)})^T>\tilde{L} \\
&\quad + <C^{(2)}><C^{(0)}>, \tag{10.60}
\end{aligned}$$

where the second equal sign is due to (10.40) and we have also used (10.43). So we have

$$\begin{aligned}
Var\{C\} &= \; <C^2>-<C>^2 \\
&\simeq \; \tilde{L}^T<\tilde{\rho}^{(1)}(\tilde{\rho}^{(1)})^T>\tilde{L}. \tag{10.61}
\end{aligned}$$

4. $N_1 + N_2 = 3$ in (10.54) and (10.55) It is clear that the term $<\tilde{\rho}^{(0)}(\tilde{\rho}^{(3)})^T>$ and $<\tilde{\rho}^{(3)}(\tilde{\rho}^{(0)})^T>$ are missing in (10.54) and the term $<C^{(0)}><C^{(3)}>$ is missing in (10.55). Hence the accuracy is not necessarily better than (10.61).

5. $N_1 + N_2 = 4$ in (10.54) and (10.55) It is clear that the term $<\tilde{\rho}^{(0)}(\tilde{\rho}^{(3)})^T>$, $<\tilde{\rho}^{(0)}(\tilde{\rho}^{(4)})^T>$, $<\tilde{\rho}^{(1)}(\tilde{\rho}^{(3)})^T>$, $<\tilde{\rho}^{(3)}(\tilde{\rho}^{(1)})^T>$ and $<\tilde{\rho}^{(4)}(\tilde{\rho}^{(0)})^T>$ are missing in (10.54) and the terms $<C^{(0)}><C^{(3)}>$, $<C^{(0)}><C^{(4)}>$ and $<C^{(1)}><C^{(3)}>$ are missing in (10.55). Hence the accuracy is not necessarily better than (10.61).

Based upon the above discussion, we will use the scheme in (10.61) to calculate the capacitance variance. Hence we just need to focus on the calculation of the term $\tilde{L}^T<\tilde{\rho}^{(1)}(\tilde{\rho}^{(1)})^T>\tilde{L}$.

In view of (10.30), (10.36) and (10.45), we have

$$\begin{aligned}
&\tilde{L}^T<\tilde{\rho}^{(1)}(\tilde{\rho}^{(1)})^T>\tilde{L} \\
&\simeq \; <(<\tilde{\rho}^{(0)}>^T E<\tilde{\rho}^{(0)}>)(<\tilde{\rho}^{(0)}>^T E^T<\tilde{\rho}^{(0)}>)> \\
&= \; <(<\tilde{\rho}^{(0)}>^T E<\tilde{\rho}^{(0)}>)\otimes(<\tilde{\rho}^{(0)}>^T E^T<\tilde{\rho}^{(0)}>)> \\
&= \; (<\tilde{\rho}^{(0)}>^T\otimes<\tilde{\rho}^{(0)}>^T)<E\otimes E^T>(<\tilde{\rho}^{(0)}>\otimes<\tilde{\rho}^{(0)}>) \\
&\simeq \; (vec(B))^T(<\tilde{\rho}^{(0)}>\otimes<\tilde{\rho}^{(0)}>) \\
&= \; (<\tilde{\rho}^{(0)}>\otimes<\tilde{\rho}^{(0)}>)^T vec(B) \\
&= \; (<\tilde{\rho}^{(0)}>^T\otimes<\tilde{\rho}^{(0)}>^T)vec(B) \\
&= \; <\tilde{\rho}^{(0)}>^T B<\tilde{\rho}^{(0)}> \tag{10.62}
\end{aligned}$$

where the second equality is due to (C.6), the third equality is due to (C.3), the fourth equality is due to (10.47), the sixth equality is due to (C.4), and the last equality is due to (C.6). It should be pointed out again that the two approximate equality in (10.62) would be exact if $\bar{A}^T = \bar{A}$ and $E^T = E$, i.e., if the Galerkin method is used. Similar to the calculation of $< C^{(2)} >$ in (10.46), everything boils down to the calculation of $vec(B)$. Hence no extra computation work is necessary to obtain the approximate value of the capacitance variance once we have used the second-order correction scheme in section 10.6 to calculate the mean capacitance.

Chapter 11

Matrix Sparsification

It is obvious that matrix F in (10.48) has $O(N^4)$ entries, where N is the number of panels used to discretize the nominal smooth surface. Hence direct calculation of matrix B in (10.46) and subsequent $< C^{(2)} >$ in (10.46) and $Var\{C\}$ in (10.61) needs $O(N^4)$ work. This makes it very time-consuming to use the stochastic IE method in chapter 10 to analyze even a simple 2D structure.

In order to reduce the CPU time, we exploit the small correlation length assumption made in chapter 9. In this chapter, we first show that matrix F can be approximated by a sparse matrix with $O(N^3)$ entries. We then show that the sparsification techniques in [54, 52, 33, 34] can be used to obtain a data-sparse representation of the matrices B and $\bar{A}^{-1}$ in (10.46) in $O(Nlog^2N)$ time. These sparse representations are then used to compute $trace(A^{-T}B)$ in (10.46) and $< \tilde{\rho}^{(0)} >^T B < \tilde{\rho}^{(0)} >$ in (10.62) in $O(NlogN)$ time. Therefore the overall cost of computing $< C^{(2)} >$ in (10.46) and $Var\{C\}$ in (10.61) are substantially reduced from $O(N^4)$ to $O(Nlog^2N)$.

11.1   Sparsification of the matrix F

As mentioned in section 10.5, without loss of generality we can assume that the surface height fluctuation is either along x direction or along y direction, but not along both directions. Hence the surface roughness can show up in the Green's function in four different forms, as shown in equation (10.24)-(10.27). There are four panels involved in the expres-

117

sion for F_{mn}^{ij} in (10.49). Here we focus on the case where these panels are all on the same rough side S_2 in figure 10-1, i.e., the height fluctuation of all four panels is along y direction and their nominal position is the same. Other cases can be treated similarly.

In view of (B.5), the system matrix entry A_{ij} and A_{mn} are

$$
\begin{aligned}
A_{ij} &= \int_{\tilde{\Delta}_i} d\tilde{l}(x,y) \int_{\tilde{\Delta}_j} d\tilde{l}(x',y') \tilde{G}(x',y';x,y) \\
&= \int_{\tilde{\Delta}_i} dx_i \int_{\tilde{\Delta}_j} dx_j G(x_j,b+h_j;x_i,b+h_i) \quad\quad\quad (11.1)
\end{aligned}
$$

and

$$
\begin{aligned}
A_{mn} &= \int_{\tilde{\Delta}_m} d\tilde{l}(x,y) \int_{\tilde{\Delta}_n} d\tilde{l}(x',y') \tilde{G}(x',y';x,y) \\
&= \int_{\tilde{\Delta}_m} dx_m \int_{\tilde{\Delta}_n} dx_n G(x_n,b+h_n;x_m,b+h_m) \quad\quad\quad (11.2)
\end{aligned}
$$

where h_i,h_j,h_m and h_n are the height fluctuation of panel i,j,m and n. [1] The second equality in (11.1) and (11.2) is due to (10.7), (10.8) and (10.12). Hence we have

$$
\begin{aligned}
< A_{ij}A_{mn} > &= \int_{\tilde{\Delta}_i} dx_i \int_{\tilde{\Delta}_j} dx_j \int_{\tilde{\Delta}_m} dx_m \int_{\tilde{\Delta}_n} dx_n \\
&\quad \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \int_{-\infty}^{+\infty} dh_n \\
&\quad P_4(h_i,h_j,h_m,h_n;x_i,x_j,x_m,x_n) \\
&\quad G(x_j,b+h_j;x_i,b+h_i)G(x_n,b+h_n;x_m,b+h_m) \\
&\simeq \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \int_{-\infty}^{+\infty} dh_n \\
&\quad P_4(h_i,h_j,h_m,h_n;\bar{x}_i,\bar{x}_j,\bar{x}_m,\bar{x}_n)A_{ij}A_{mn}, \quad\quad\quad (11.3)
\end{aligned}
$$

where $(\bar{x}_i,b),(\bar{x}_j,b),(\bar{x}_m,b),(\bar{x}_n,b)$ are the centroids of panel i,j,m and n. The approximate equality is due to the change of order in panel integration and ensemble average integration,

---

[1] Here we use piecewise constant shape function to approximate the rough surface profile. The piecewise linear or rooftop shape function can also be used. But since each linear shape function involves two random variables (height fluctuation of the two ends of the panel), the ensemble average in (11.3) will involve an eight-fold integral in the worst-case scenario, i.e., when all four panels are different. This would be computationally too expensive.

same as in (10.21). The multiple dimension joint Gaussian distribution function P_4 is

$$P_4(h_i,h_j,h_m,h_n;\bar{x}_i,\bar{x}_j,\bar{x}_m,\bar{x}_n) = \frac{1}{(2\pi)^2\sigma^4\sqrt{d_4}}\exp(\frac{-\bar{h}^T[D]^{-1}\bar{h}}{2\sigma^2}), \qquad (11.4)$$

where σ is the standard deviation,

$$\bar{h} = \begin{bmatrix} h_i \\ h_j \\ h_m \\ h_n \end{bmatrix}, \qquad (11.5)$$

$$[D] = \begin{bmatrix} 1 & C(|\bar{x}_i - \bar{x}_j|) & C(|\bar{x}_i - \bar{x}_m|) & C(|\bar{x}_i - \bar{x}_n|) \\ C(|\bar{x}_j - \bar{x}_i|) & 1 & C(|\bar{x}_j - \bar{x}_m|) & C(|\bar{x}_j - \bar{x}_n|) \\ C(|\bar{x}_m - \bar{x}_i|) & C(|\bar{x}_m - \bar{x}_j|) & 1 & C(|\bar{x}_m - \bar{x}_n|) \\ C(|\bar{x}_n - \bar{x}_i|) & C(|\bar{x}_n - \bar{x}_j|) & C(|\bar{x}_n - \bar{x}_m|) & 1 \end{bmatrix}, \qquad (11.6)$$

and $d_4 = det(D)$. The entries in $[D]$ are the correlation coefficient defined in (9.4).

To facilitate the following discussion, we define respectively set $near(i)$ and $far(i)$ as

$$near(i) = \{panel_k| \, |\bar{x}_i - \bar{x}_k| \le 3\eta\} \qquad (11.7)$$

and

$$far(i) = \{panel_k| \, |\bar{x}_i - \bar{x}_k| > 3\eta\} \qquad (11.8)$$

where $\bar{x}_i$ and $\bar{x}_k$ are respectively the centroids of panel i and k, and η is the correlation length. It is straightforward to show that

$$j \in near(i) \Rightarrow i \in near(j) \qquad (11.9)$$

and

$$j \in far(i) \Rightarrow i \in far(j). \qquad (11.10)$$

119

In addition, it is also easy to show that

$$n \in near(j), i \in far(j) \Rightarrow i \in far(n) \tag{11.11}$$

is almost always true.

Equation (9.4) suggests that the correlation coefficient drops exponentially fast with the distance between panels, i.e.

$$j \in far\{i\} \Rightarrow |x_i - x_j| \geq 3\eta$$
$$\Rightarrow \quad C(|\bar{x}_i - \bar{x}_j|) \leq 1.24 \times 10^{-4} \Rightarrow C(|\bar{x}_i - \bar{x}_j|) \simeq 0. \tag{11.12}$$

Therefore, if

$$\begin{cases} m \in far\{i\} \\ n \in far\{i\} \\ m \in far\{j\} \\ m \in far\{j\} \end{cases} \tag{11.13}$$

then

$$\begin{cases} C(|\bar{x}_i - \bar{x}_m|) \simeq 0 \\ C(|\bar{x}_i - \bar{x}_n|) \simeq 0 \\ C(|\bar{x}_j - \bar{x}_m|) \simeq 0 \\ C(|\bar{x}_j - \bar{x}_n|) \simeq 0 \end{cases}, \tag{11.14}$$

and matrix D becomes

$$[D] \simeq \begin{bmatrix} 1 & C(|\bar{x}_i - \bar{x}_j|) & 0 & 0 \\ C(|\bar{x}_j - \bar{x}_i|) & 1 & 0 & 0 \\ 0 & 0 & 1 & C(|\bar{x}_m - \bar{x}_n|) \\ 0 & 0 & C(|\bar{x}_n - \bar{x}_m|) & 1 \end{bmatrix}. \tag{11.15}$$

120

In this case it is straightforward to show that

$$P_4(h_i, h_j, h_m, h_n; \bar{x}_i, \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_2(h_i, h_j; \bar{x}_i, \bar{x}_j) P_2(h_m, h_n; \bar{x}_m, \bar{x}_n) \qquad (11.16)$$

where the joint distribution function P_2 is defined in (9.2). In view of (11.3), we have $< A_{ij}A_{mn} > \simeq \bar{A}_{ij}\bar{A}_{mn}$ and hence $F_{nm}^{ij} \simeq 0$ immediately follows from (10.49). It should be pointed out that (11.14) would be exact if panel pair i and j and panel pair m and n are on different side of the same conductor surface or on different conductor surfaces. For structures with multiple conductors, this is expected to be the majority case. Hence the error introduced by this scheme is quite insignificant.

We assume that a rough surface segment of 3η long contains at most p panels. Since η is independent of the total number of panels, so is p. The approximation in (11.13)-(11.15) implies that there are at most $4p + 2$ non-zero rows and columns in each sparsified F^{ij}. A typical sparsity pattern of the sparsified matrix F^{ij} is shown in figure 11-1, where it is assumed that the indexes of the spatially close panels are also close to each other. The natural panel ordering for 2D structures always satisfies this. However, a panel ordering for 3D surfaces may not. But this does not change the fact that matrix F^{ij} can be approximated by a sparse matrix. Now it is clear that the sparsified matrix F^{ij} has $O(N)$ non-zero entries and the total number of non-zero entries in the sparsified matrix F is $O(N^3)$.

11.2   Characteristics of matrix F and matrix B

Since the sparsified matrix F has $O(N^3)$ entries, direct computing $vec(B)$ or matrix B in (10.46) still needs $O(N^3)$ operations. In this section we conduct a detailed analysis of the characteristics of matrix F and B, and show that matrix B is hierarchically low rank. This makes it possible to use an idea similar to the hierarchical SVD in [54, 52] and the so-call hierarchical matrix (H-matrix) in [33, 34, 5] to construct a sparse representation of matrix B.
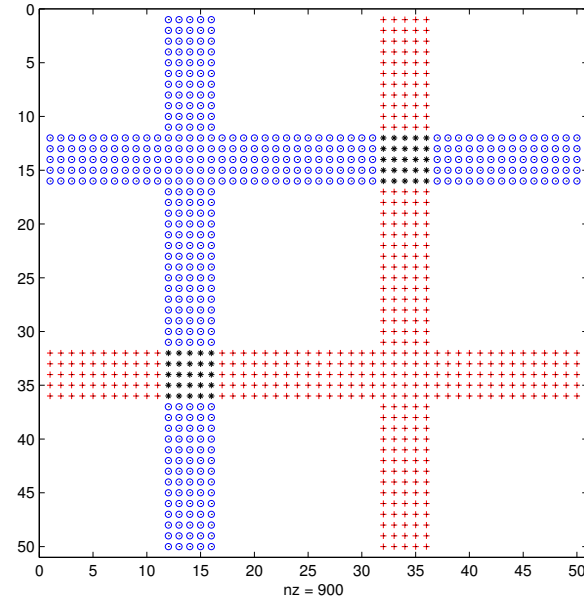
Figure 11-1: Typical sparsity pattern of the sparsified matrix F^{ij}. Here the total number of panels is 50, $i = 34$ and $j = 14$, and a rough surface segment of 3η long contains $p = 2$ panels. The nonzero entries are categorized into three regiones marked by $+$ (region I), o (region II) and $*$ (region III), respectively.

11.2.1 Characteristics of matrix F

The nonzero entries in figure 11-1 are categorized into three regions marked by $+$ (region I), o (region II) and $*$ (region III), respectively. In view of (11.13), the row and column index pair (m,n) in region I, II and III of matrix F^{ij} satisfies

1. Region I: $\{\, m \in near(i) \text{ and } n \in far(j) \,\}$ or $\{\, m \in far(j) \text{ and } n \in near(i) \,\}$

2. Region II: $\{\, m \in near(j) \text{ and } n \in far(i) \,\}$ or $\{\, m \in far(i) \text{ and } n \in near(j) \,\}$

3. Region III: $\{\, m \in near(j) \text{ and } n \in near(i) \,\}$ or $\{\, m \in near(i) \text{ and } n \in near(j) \,\}$

For a fixed panel i, it is obvious that most other panels belong to the set $far(i)$. Using property (11.10) and (11.11), one can show that the regions I and II in the majority of the matrices F^{ij} satisfy

1. Region I: $j \in far(i)$, $j \in far(m)$ and $j \in far(n)$

2. Region II: $i \in far(j)$, $i \in far(m)$ and $i \in far(n)$

122

Following the same reasoning from (11.13)-(11.16), we immediately obtain

$$P_4(h_i, h_j, h_m, h_n; \bar{x}_i, \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_3(h_i, h_m, h_n; \bar{x}_i, \bar{x}_m, \bar{x}_n) P_1(h_j) \qquad (11.17)$$

in region I, and

$$P_4(h_i, h_j, h_m, h_n; \bar{x}_i, \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_3(h_j, h_m, h_n; \bar{x}_j, \bar{x}_m, \bar{x}_n) P_1(h_i) \qquad (11.18)$$

in region II, where the definition of the joint distribution function P_3 is similar to that of the function P_4 in (11.4). Substituting (11.17) and (11.18) into (11.3) and (10.49) results in

$$
\begin{aligned}
F_{mn}^{ij} \simeq{}& \int_{-\infty}^{+\infty} dh_j P_1(h_j) \\
& \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_m \int_{-\infty}^{+\infty} dh_n \\
& P_3(h_i, h_m, h_n; \bar{x}_i, \bar{x}_m, \bar{x}_n) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}
\end{aligned}
\qquad (11.19)
$$

for the entries in region I and

$$
\begin{aligned}
F_{mn}^{ij} \simeq{}& \int_{-\infty}^{+\infty} dh_i P_1(h_i) \\
& \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \int_{-\infty}^{+\infty} dh_n \\
& P_3(h_j, h_m, h_n; \bar{x}_j, \bar{x}_m, \bar{x}_n) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}
\end{aligned}
\qquad (11.20)
$$

for the entries in region II, respectively.

In view of (10.47) and (10.48), each column in matrix B is

$$
\begin{aligned}
b_i &= \sum_{j=1}^{N} F^{ij} < \rho^{(0)} >< \rho_j^{(0)} > \\
&= \begin{bmatrix} f^{i1} & f^{i2} & \cdots & f^{iN} \end{bmatrix} < \rho^{(0)} >, \\
&= [M^{(i)}] < \rho^{(0)} >
\end{aligned}
\qquad (11.21)
$$

where

$$f^{ij} = F^{ij} < \rho^{(0)} >, \quad j = 1, 2, \cdots, N. \qquad (11.22)$$

Let

$$F^{ij} = H_1^{ij} + H_2^{ij} + H_3^{ij} \tag{11.23}$$

where the entries in the sparse matrices H_1^{ij}, H_2^{ij} and H_3^{ij} belong to region I, II and III in the figure 11-1, respectively, we have

$$f^{ij} = (H_1^{ij} + H_2^{ij} + H_3^{ij}) < \rho^{(0)} >= f_1^{ij} + f_2^{ij} + f_3^{ij}. \tag{11.24}$$

Since matrix H_3^{ij} is very sparse and its entries are usually one to two order of magnitude smaller than those of H_1^{ij} and H_2^{ij}, we neglect its contribution in the following analysis of characteristics of matrix F^{ij}. So we have

$$f^{ij} \simeq (H_1^{ij} + H_2^{ij}) < \rho^{(0)} >= f_1^{ij} + f_2^{ij}. \tag{11.25}$$

11.2.2   Characteristics of f_1^{ij} and f_2^{ij}

Both region I and region II in figure 11-1 have two sub-regions, one is vertical and the other is horizontal. These two sub-regions have different characteristics. Again, here we only focus on matrix blocks F^{ij} with $i \in far(j)$, which account for the majority case.

1. Vertical sub-region of region I: $m \in far(i)$ and $m \in far(n)$

   Similar to (11.17), we immediately obtain

   $$P_3(h_i, h_m, h_n; \bar{x}_i, \bar{x}_m, \bar{x}_n) \simeq P_2(h_i, h_n; \bar{x}_i, \bar{x}_n) P_1(h_m). \tag{11.26}$$

   Substituting (11.26) into (11.19) and in view of (11.23), we obtain

   $$\begin{aligned}
   H_1^{ij}(m,n) \simeq & \int_{-\infty}^{+\infty} dh_j P_1(h_j) \int_{-\infty}^{+\infty} dh_m P_1(h_m) \\
   & \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_n \\
   & P_2(h_i, h_n; \bar{x}_i, \bar{x}_n) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}. \tag{11.27}
   \end{aligned}$$

In view of (11.24), we have

$$
\begin{aligned}
f_1^{ij}(m) \;\simeq\; & \int_{-\infty}^{+\infty} dh_j P_1(h_j) \int_{-\infty}^{+\infty} dh_m P_1(h_m) \\
& \int_{-\infty}^{+\infty} dh_i A_{ij} V_m^i - \bar{A}_{ij}\bar{V}_m,
\end{aligned} \tag{11.28}
$$

where

$$
V_m^i = \sum_{n \in near\{i\}} \int_{-\infty}^{+\infty} dh_n P_2(h_i, h_n; \bar{x}_i, \bar{x}_n) A_{mn} < \rho_n^{(0)} > \tag{11.29}
$$

and

$$
\bar{V}_m = \sum_{n \in near\{i\}} \bar{A}_{mn} < \rho_n^{(0)} > . \tag{11.30}
$$

The asymptotic behavior of A_{mn} and $\bar{A}_{mn}$ in terms of r_{mn}, the distance between panel m and panel n, is $\frac{1}{r_{mn}}$. Since $n \in near\{i\}$ in (11.29), the distance between panel n and panel i is quite small, as shown in (11.7). For the purpose of asymptotic analysis, these two panels can be considered as being overlapped. Therefore, the asymptotic behavior of V_m^i and $\bar{V}_m$ in terms of r_{mi} is $\frac{1}{r_{mi}}$. [2] This implies that asymptotic behavior of $f_1^{ij}(m)$ in terms of r_{mi} is $\frac{1}{r_{mi}}$. It is straightforward to show that the asymptotic behavior of $f_1^{ij}(m)$ in terms of r_{ij} is the same as that of A_{ij} and $\bar{A}_{ij}$, which is $\frac{1}{r_{ij}}$. The degeneration in (11.17) and (11.26) also makes $f_1^{ij}(m)$ independent of r_{mj}, as can be easily checked from (11.28).

2. Horizontal sub-region of region I: $n \in far(i)$ and $n \in far(m)$ Similar to (11.17), we immediately obtain

$$
P_3(h_i, h_m, h_n; \bar{x}_i, \bar{x}_m, \bar{x}_n) \simeq P_2(h_i, h_m; \bar{x}_i, \bar{x}_m) P_1(h_n). \tag{11.31}
$$

---

[2] The rapid variation in $P_2(h_i, h_n; \bar{x}_i, \bar{x}_n)$ has been absorbed into the summation in (11.29) and becomes irrelevant.

Substituting (11.31) into (11.19) and in view of (11.23), we obtain

$$H_1^{ij}(m,n) \simeq \int_{-\infty}^{+\infty} dh_j P_1(h_j) \int_{-\infty}^{+\infty} dh_n P_1(h_n)$$
$$\int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_m$$
$$P_2(h_i, h_m; \bar{x}_i, \bar{x}_m) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}. \tag{11.32}$$

In view of (11.24), we have

$$f_1^{ij}(m) \simeq \int_{-\infty}^{+\infty} dh_j P_1(h_j) \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_m$$
$$P_2(h_i, h_m; \bar{x}_i, \bar{x}_m) A_{ij} V_m - \bar{A}_{ij} \bar{V}_m, \tag{11.33}$$

where

$$V_m = \sum_{n=1}^{N} \int_{-\infty}^{+\infty} dh_n P_1(h_n) A_{mn} < \rho_n^{(0)} >, \tag{11.34}$$

and $\bar{V}_m$ is defined in (11.30). Similar to (11.28), the asymptotic behavior of $f_1^{ij}(m)$ in terms of r_{ij} is also $\frac{1}{r_{ij}}$. By definition, $m \in near\{i\}$ is always true in this sub-region. So we can also treat panel m and panel i as being overlapped in the asymptotic analysis. Therefore, the asymptotic behavior of $f_1^{ij}(m)$ in terms of r_{mj} is $\frac{1}{r_{mj}}$. Due to $P_2(h_i, h_m; \bar{x}_i, \bar{x}_m)$ in (11.33) and $m \in near\{i\}$, $f_1^{ij}(m)$ changes exponentially fast with r_{mi}.

3. Vertical sub-region of region II: $m \in far(j)$ and $m \in far(n)$ Similar to (11.17), we immediately obtain

$$P_3(h_j, h_m, h_n; \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_2(h_j, h_n; \bar{x}_j, \bar{x}_n) P_1(h_m). \tag{11.35}$$

Substituting (11.35) into (11.20) and in view of (11.23), we obtain

$$H_2^{ij}(m,n) \simeq \int_{-\infty}^{+\infty} dh_i P_1(h_i) \int_{-\infty}^{+\infty} dh_m P_1(h_m)$$

126

$$\int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_n$$

$$P_2(h_j, h_n; \bar{x}_j, \bar{x}_n) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}. \tag{11.36}$$

In view of (11.24), we have

$$
f_2^{ij}(m) \simeq \int_{-\infty}^{+\infty} dh_i P_1(h_i) \int_{-\infty}^{+\infty} dh_m P_1(h_m)
$$

$$
\int_{-\infty}^{+\infty} dh_j A_{ij} V_m^j - \bar{A}_{ij} \bar{V}_m, \tag{11.37}
$$

where

$$
V_m^j = \sum_{n \in near\{j\}} \int_{-\infty}^{+\infty} dh_n P_2(h_j, h_n; \bar{x}_j, \bar{x}_n) A_{mn} < \rho_n^{(0)} > . \tag{11.38}
$$

Equations (11.37) and (11.38) are dual to (11.28) and (11.29) in the sense that we just need to switch from i to j. Hence following the same analysis for (11.28) and (11.29) we conclude that the asymptotic behavior of $f_2^{ij}(m)$ in terms of r_{ij} and r_{mj} is $\frac{1}{r_{ij}}$ and $\frac{1}{r_{mj}}$, respectively, and $f_2^{ij}(m)$ is not related to r_{mi}.

4. Horizontal sub-region of region II: $n \in far(j)$ and $n \in far(m)$

   Similar to (11.17), we immediately obtain

$$
P_3(h_j, h_m, h_n; \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_2(h_j, h_m; \bar{x}_j, \bar{x}_m) P_1(h_n). \tag{11.39}
$$

Substituting (11.39) into (11.20) and in view of (11.23), we obtain

$$
H_2^{ij}(m, n) \simeq \int_{-\infty}^{+\infty} dh_i P_1(h_i) \int_{-\infty}^{+\infty} dh_n P_1(h_n)
$$

$$
\int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m
$$

$$
P_2(h_j, h_m; \bar{x}_j, \bar{x}_m) A_{ij} A_{mn} - \bar{A}_{ij} \bar{A}_{mn}. \tag{11.40}
$$

In view of (11.24), we have

$$
\begin{aligned}
f_2^{ij}(m) \quad \simeq \quad & \int_{-\infty}^{+\infty} dh_i P_1(h_i) \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \\
& P_2(h_j, h_m; \bar{x}_j, \bar{x}_m) A_{ij} V_m - \bar{A}_{ij} \bar{V}_m.
\end{aligned} \tag{11.41}
$$

Equations (11.41) is dual to (11.33) in the sense that we just need to switch from i to j. Hence following the same analysis for (11.33) we conclude that the asymptotic behavior of $f_2^{ij}(m)$ in terms of r_{ij} and r_{mi} is respectively $\frac{1}{r_{ij}}$ and $\frac{1}{r_{mi}}$, and $f_2^{ij}(m)$ changes exponentially fast with r_{mj}.

11.2.3 $M^{(i)}$ is hierarchically low-rank

In view of (11.21) and (11.25), it is clear that asymptotic behavior of $M_{mj}^{(i)}$ in terms of r_{mj} is the same as that of $f_1^{ij}(m) + f_2^{ij}(m)$. From the analysis for (11.28),(11.33),(11.37) and (11.41), $M_{mj}^{(i)}$ should behave asymptotically as $\frac{1}{r_{mj}}$, except for the case where $m \in near\{j\}$. Hence matrix $M^{(i)}$ is hierarchically low-rank, similar to a discretized integral operator with $\frac{1}{r}$ kernel. To verify our analysis, we have done some numerical experiments on a 2D and a 3D structure.

2D examples: Circular Wire Over Ground   The singular values of the 4 sub-blocks of the matrix $M^{(i)}$ for the case of $i = 1$ and $i = 10$ are shown in figures 11-2 and 11-3. It is clear that both matrices are hierarchically low-rank.

3D examples: 1D bars   We use two 1D bar in parallel as a simple example. Each bar is divided along the length into 30 panels. The singular values of the 4 sub-blocks of the matrix $M^{(i)}$ and one of its sub-blocks, $M_{11}^{(i)}$, are shown in figures 11-4 and 11-5. Here we randomly pick the case of $i = 60$. It is clear that matrix $M^{(i)}$ is hierarchically low-rank.
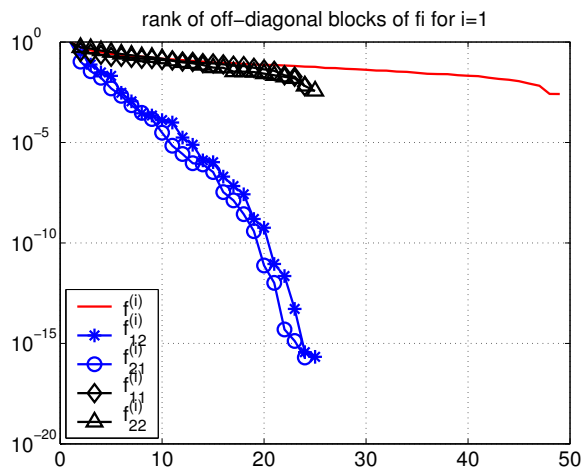
Figure 11-2: Distribution of the singular values of the four sub-blocks of the matrix $M^{(i)}$ for $i = 1$, circular wire over ground plane.
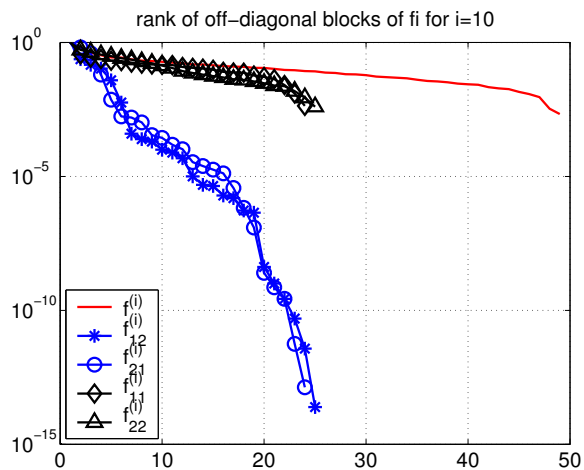
Figure 11-3: Distribution of the singular values of the four sub-blocks of the matrix $M^{(i)}$ for $i = 10$, circular wire over ground plane.
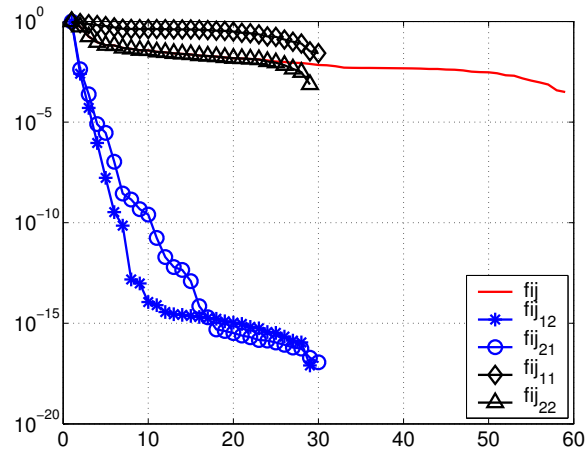
Figure 11-4: Distribution of the singular values of the four sub-blocks of the matrix $M^{(i)}$ for $i = 60$, 1D bars.
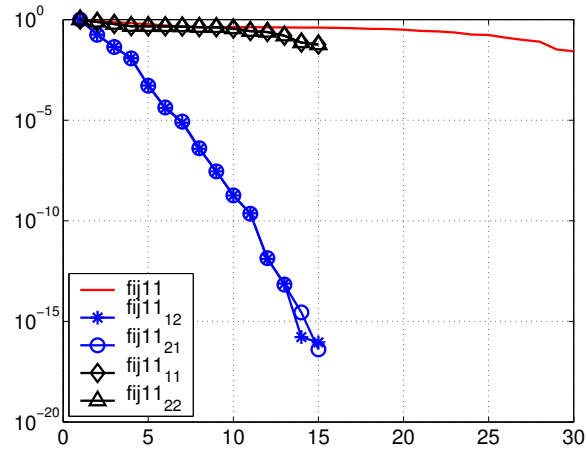
Figure 11-5: Distribution of the singular values of the four sub-blocks of the matrix $M_{11}^{(i)}$ for $i = 60$, 1D bars.
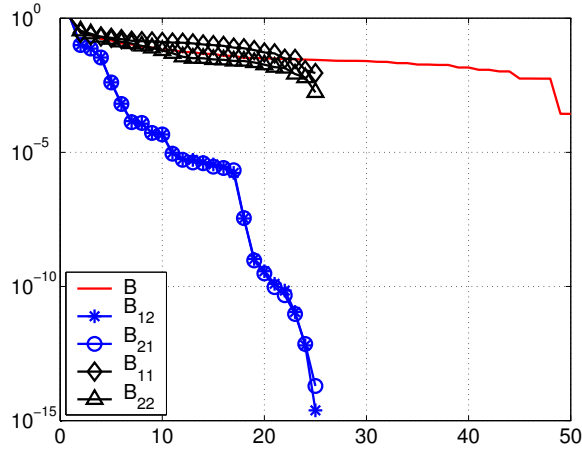
Figure 11-6: Distribution of the singular values of the four sub-blocks of the matrix B, circular wire over ground plane.

11.2.4 B is low-rank

In view of (11.21) and (11.25), it is clear that the asymptotic behavior of $b_i(m)$ in terms of r_{mi} is the same as that of $f_1^{ij}(m) + f_2^{ij}(m)$. [3] From the analysis for (11.28),(11.33),(11.37) and (11.41), $b_i(m)$ should behave asymptotically as $\frac{1}{r_{mi}}$, except for the case where $m \in near\{i\}$. Hence matrix B is hierarchically low-rank, similar to a discretized integral operator with $\frac{1}{r}$ kernel.

To verify our analysis, we have done some numerical experiments on the same set of examples as in section 11.2.3. Even with the sparsified matrix F, filling matrix B still needs $O(N^3)$ work. Hence we can not analyze large and complicated structures. But we believe the results shown here are compelling enough to motivate a sparsification algorithm based on the H-matrix method.

2D examples: Circular Wire Over Ground   The singular values of the 4 sub-blocks of the matrix B is shown in figure 11-6. Furthermore, the singular values of the 4 sub-blocks of the matrix B_{11}, a sub-block of matrix B, is shown in figure 11-7. It is clear that matrix B is hierarchically low-rank.

---

[3]Index j has been absorbed because of matrix-vector product in (11.21) and hence the asymptotic behavior of $M_{mj}^{(i)}$ in terms of r_{mj} becomes irrelevant here.
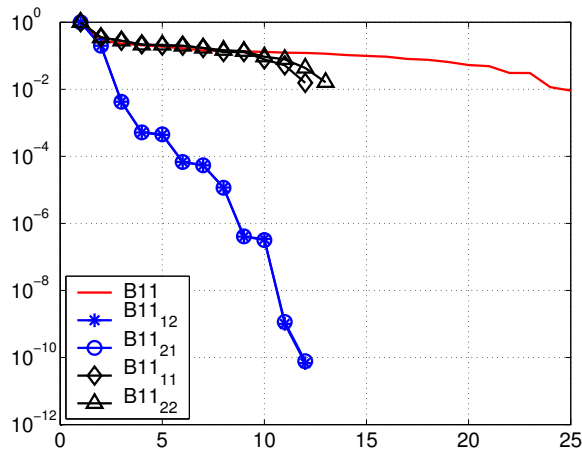
Figure 11-7: Distribution of the singular values of the four sub-blocks of the matrix B_{11}, circular wire over ground plane.
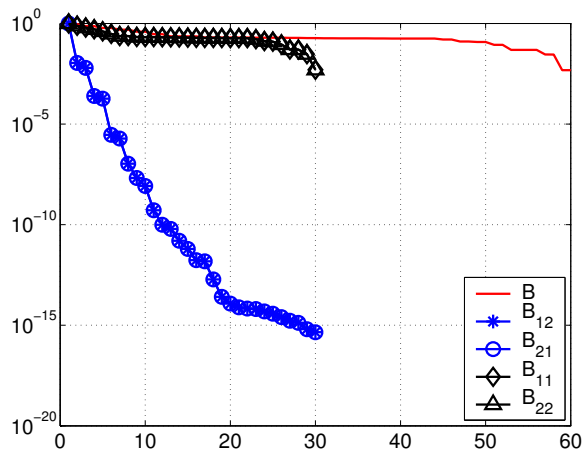
Figure 11-8: Singular value distribution of the four sub-blocks of the matrix B, 1D bars.

3D examples: 1D bars   We use two 1D bar in parallel as a simple example. Each bar is divided along the length into 30 panels. The singular values of the 4 sub-blocks of the matrix B and one of its sub-block, B_{11}, are shown in figures 11-8 and 11-9. It is clear that matrix B is hierarchically low-rank.

11.2.5   Matrix B is symmetric

It turns out that an important fact, $B = B^T$, can be used to reduce the cost of constructing the H-matrix for B substantially. So it is worthwhile to show it here. From (10.49) it is easy
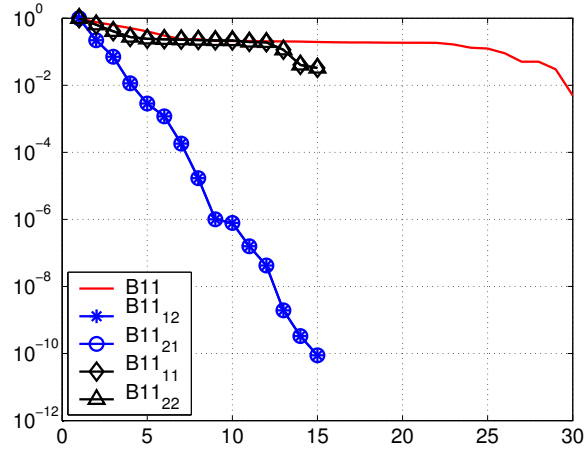
Figure 11-9: Singular value distribution of the four sub-blocks of the matrix B_{11}, 1D bars.

to check that

$$F_{mn}^{ij} = F_{ij}^{mn} \qquad (11.42)$$

In view of (11.21), we have

$$b_i(m) = b_m(i), \qquad (11.43)$$

i.e.,

$$B = B^T. \qquad (11.44)$$

The identically distributed singular values of the off-diagonal blocks in figures 11-6, 11-7, 11-8 and 11-9 correlate very well with this observation.

11.3 Construction of H-matrix for matrix B: A simple case

The fact that matrix B is hierarchically low-rank implied that it is possible to construct a data-sparse representation of matrix B using either the hierarchical matrix (H-matrix) method in [33, 34, 5] or the hierarchical SVD method in [52]. Since we do not have analytical form of each individual entry of B, we can not use the degenerated polynomial interpolation scheme in the H-matrix method [5, 35]. Instead we resort to a sampling process similar to the one in [52, 65]. Hence we need the access to each individual entry of B. As implied in (11.21), calculation of column vector b_i involves constructing the H-matrix for $M^{(i)}$ and carrying out one matrix-vector product $M^{(i)} < \rho^{(0)} >$. Each step needs

133

$O(Nlog(N))$ work. But in constructing the H-matrix for B, usually only a small fraction of b_i is needed. Unfortunately, calculation of even a small fraction of b_i still involves setup of the H-matrix for the whole matrix $M^{(i)}$, which takes $O(Nlog(N))$ time. [4] Hence the total time would be $O(N^2 log(N))$ if the H-matrix for whole matrix $M^{(i)}$ is constructed for $i = 1, 2, ..., N$.

In this paper, we propose a so-called combined sampling process to substantially reduce this cost. The key idea is to combine small segments of vector b_i for different i into a long vector and compute this long vector in one go. This means that one has to assemble a part of matrix $M^{(i)}$ for different i into a single matrix $\tilde{M}$, construct its H-matrix representation and perform matrix vector product $\tilde{M} < \rho^{(0)} >$. In order to focus our attention to the main ideas, we deliberately use a trivially simple example in this section to explain this combined sampling process. Algorithm for the more general cases is shown in the following section.

Assume that matrix B has a two-level H-matrix representation shown in figure 11-10, where $R_i(i = 1, 2, ..., 6)$ are low-rank matrices and $D_i(i = 1, 2, 3, 4)$ are full-rank matrices. Further more, assume that the size of matrix B is $N \times N = 32 \times 32$ and the rank of all R_i is the same $r = 2$. Consequently, the size of matrix D_i is 8×8. It should be pointed out that the actual values of N and r are unimportant, we use them here just for the convenience of notation.

11.3.1   Standard sampling process

A low-rank matrix Q can be written as [33, 34, 5, 52]

$$Q = W^T V, \; Q \in R^{M \times M}, \; W, V \in R^{r \times M}, \tag{11.45}$$

---

[4]Suppose we want to compute $y = Ax$, where A corresponds to a discretized integral operator with $\frac{1}{r}$ kernel. It is interesting that all well-known fast solver algorithms can not do better than $O(N)$ when only one entry of y is needed, despite the fact that all these algorithms can compute the whole vector y with N entries in $O(N)$ or $O(Nlog(N))$ time. The multipole expansion in Fast Multipole Method (FMM), sampling in hierarchical SVD, panel clustering in H-matrix method, and direct matrix setup in Pre-corrected FFT method involve all N source panels, regardless how many entries in y are to be computed. This initialization cost is amortized over all N entries of y. But if only a small number of entries in y are needed, then the overhead of initial setup is as expensive as direct calculation of those few entries in y.

Figure 11-10: Two-level H-matrix representation of matrix B. R_i is the low-rank matrix and D_i is the full-rank matrix.

where r is the rank of Q. This decomposition is not unique, one can scale W with an arbitrary factor α and scale V with $\frac{1}{\alpha}$. For simplicity, we assume here that both W and V have been normalized such that the decomposition is unique. A number of heuristics to construct this WV decomposition have been proposed, such as the rank-revealing QR decomposition in [52] and the adaptive low-rank approximation in [65]. We do not intend to repeat these two algorithms here. But it suffices to point out that both methods need access to r columns and r rows of matrix Q in (11.45) and both methods have been claimed to access only $O(Nlog(N))$ entries in a hierarchically low-rank matrix in order to construct its sparse representation.

It is easy to check that

$$\begin{cases} Q_1 = Q_2^T \\ Q_1 = W_1^T V_1 \\ Q_2 = W_2^T V_2 \end{cases} \Rightarrow \begin{cases} W_1 = V_2 \\ V_1 = W_2. \end{cases} \tag{11.46}$$

As shown in (11.44), matrix B is symmetric. Therefore, the low-rank blocks in figure 11-10 satisfy $R_1 = R_2^T$, $R_3 = R_4^T$ and $R_5 = R_6^T$. Let the WV decomposition of $R_i (i = 1, 2, ..., 6)$ be

$$R_i = W_i^T V_i, \ R_i \in R^{8 \times 8}, \ W_i, V_i \in R^{2 \times 8}, \ i = 1, 2, 3, 4. \tag{11.47}$$

$$R_i = W_i^T V_i, \ R_i \in R^{16 \times 16}, \ W_i, V_i \in R^{2 \times 16}, \ i = 5, 6. \tag{11.48}$$

In view of (11.46), we have

$$V_1 = W_2, V_2 = W_1 \tag{11.49}$$

$$V_3 = W_4, V_4 = W_3 \tag{11.50}$$

$$V_5 = W_6, V_6 = W_5. \tag{11.51}$$

Hence we only need column vectors $W_i^T (i = 1, 2, ..., 6)$ to construct the WV decomposition of $R_i(i = 1, 2, ..., 6)$. Row vectors $V_i(i = 1, 2, ..., 6)$ are redundant.

11.3.2 Combined sampling process

In order to obtain each D_i, we need to compute 8 column vectors in B. Hence to get all $D_i(i = 1, 2, 3, 4)$, we effectively have to compute the whole matrix B. Consider putting D_i into a big matrix as the following

$$T_1 = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \end{bmatrix} = \begin{bmatrix} \tilde{b}_1 & \tilde{b}_2 & \cdots & \tilde{b}_8 \\ \tilde{b}_9 & \tilde{b}_{10} & \cdots & \tilde{b}_{16} \\ \tilde{b}_{17} & \tilde{b}_{18} & \cdots & \tilde{b}_{24} \\ \tilde{b}_{25} & \tilde{b}_{26} & \cdots & \tilde{b}_{32} \end{bmatrix}, \tag{11.52}$$

where $\tilde{b}_i$ is one quarter of b_i with suitably selected entries. For example, $\tilde{b}_1 = b_1(1:8)$, $\tilde{b}_9 = b_9(9:16)$, $\tilde{b}_{17} = b_{17}(17:24)$ and $\tilde{b}_{25} = b_{25}(25:32)$. [5] Clearly, the size of matrix T_1 is 32×8. Therefore, instead of sampling the whole matrix B, we only need to compute 8 column vectors in T_1.

As explained before, to obtain the low-rank representation of R_i, we need to sample two

---

[5]Matlab matrix notation is used here.

column vectors in B. Consider putting $R_i(i = 1, 2, 3, 4)$ into a big matrix as

$$T_2 = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \end{bmatrix} = \begin{bmatrix} W_1^T \\ W_2^T \\ W_3^T \\ W_4^T \end{bmatrix} \begin{bmatrix} V_1 & V_2 & V_3 & V_4 \end{bmatrix} = W^T V. \tag{11.53}$$

To obtain W, we just need to compute

$$S_1 = \begin{bmatrix} \tilde{b}_9 & \tilde{b}_{10} \\ \tilde{b}_1 & \tilde{b}_2 \\ \tilde{b}_{25} & \tilde{b}_{26} \\ \tilde{b}_{17} & \tilde{b}_{18} \end{bmatrix}, \tag{11.54}$$

where $\tilde{b}_i$ is again one quarter of b_i with suitably selected entries. Therefore, instead of sampling a total of 8 column vectors in matrix B, we only need to compute 2 column vectors in S_1.

Finally, to obtain the low-rank representation of $R_i(i = 5, 6)$, we need to sample two column vectors in B. Consider putting $R_i(i = 5, 6)$ into a big matrix as

$$T_3 = \begin{bmatrix} R_5 \\ R_6 \end{bmatrix} = \begin{bmatrix} W_5^T \\ W_6^T \end{bmatrix} \begin{bmatrix} V_5 & V_6 \end{bmatrix} = W^T V. \tag{11.55}$$

To obtain W, we just need to compute

$$S_2 = \begin{bmatrix} \tilde{b}_{17} & \tilde{b}_{18} \\ \tilde{b}_1 & \tilde{b}_2 \end{bmatrix}, \tag{11.56}$$

where $\tilde{b}_i$ is one half of b_i with suitably selected entries. Therefore, instead of sampling a total of 4 column vectors in matrix B, we only need to compute 2 column vectors in S_2.

Now the problem boils down to computing a combined vector in the form of

$$\tilde{b} = \begin{bmatrix} \tilde{b}_\alpha \\ \tilde{b}_\beta \\ \vdots \\ \tilde{b}_\gamma \end{bmatrix} = \begin{bmatrix} \tilde{M}^{(\alpha)} \\ \tilde{M}^{(\beta)} \\ \vdots \\ \tilde{M}^{(\gamma)} \end{bmatrix} < \rho^{(0)} > = \tilde{M} < \tilde{\rho}^{(0)} >, \qquad (11.57)$$

where α, β and γ represent the column index in (11.52), (11.54) and (11.56), and $\tilde{M}^{(\alpha)}$, $\tilde{M}^{(\beta)}$ and $\tilde{M}^{(\gamma)}$ are respectively a part of $M^{(\alpha)}$, $M^{(\beta)}$ and $M^{(\gamma)}$ defined in (11.21) with suitably selected rows. If $\tilde{M}$ is hierarchically low-rank, then H-matrix method can be used to calculate $\tilde{b}$ in $O(Nlog(N))$ time. This immediately implies that the low-rank representation for R_i in (11.47) and (11.48) and the small full-rank matrices D_i in (11.52) can be computed efficiently.

11.3.3   Combined matrix is hierarchically low-rank

As shown in section 11.2.2, the asymptotic behavior of $M_{mj}^{(i)}$ in terms of r_{mj} is uniformly $\frac{1}{r_{mj}}$ for $i = 1, 2, ...N$. This asymptotic behavior should still hold for $\tilde{M}_{mj}$. Therefore, matrix $\tilde{M}$ is hierarchically low-rank.

To verify our analysis, we have done simple numerical experiments on the 3D example used in section 11.2.3. The two bars are divided into 32 panels each. So the size of matrix B and $\tilde{M}$ is 64×64. We have filled three combined matrices like $\tilde{M}$ in (11.57). They are

$$\tilde{M}_1 = \begin{bmatrix} \tilde{M}^{(1)} \\ \tilde{M}^{(17)} \\ \tilde{M}^{(33)} \\ \tilde{M}^{(49)} \end{bmatrix} \qquad (11.58)$$

$$\tilde{M}_2 = \begin{bmatrix} \tilde{M}^{(2)} \\ \tilde{M}^{(18)} \\ \tilde{M}^{(34)} \\ \tilde{M}^{(50)} \end{bmatrix} \qquad (11.59)$$
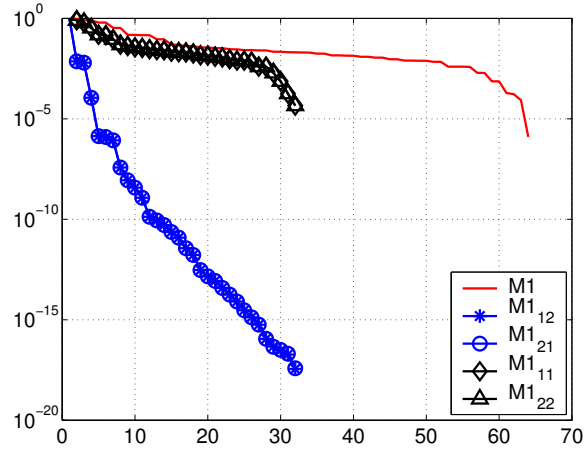
Figure 11-11: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_1$.

and

$$\tilde{M}_3 = \begin{bmatrix} \tilde{M}^{(1)} \\ \tilde{M}^{(2)} \\ \tilde{M}^{(3)} \\ \vdots \\ \tilde{M}^{(64)} \end{bmatrix}, \tag{11.60}$$

where $\tilde{M}_3$ is an extreme case in the sense that each of its rows is taken from a different $\tilde{M}^{(i)}, i = 1, 2, ..., 64$. The singular values of the 4 sub-blocks of these three matrices are shown in figures 11-11, 11-12 and 11-13. The singular values of the 4 sub-blocks of one of the diagonal blocks of these three matrices are shown in figures 11-14, 11-15 and 11-16. Clearly, matrices $\tilde{M}_1$, $\tilde{M}_2$ and $\tilde{M}_3$ are hierarchically low-rank.

11.3.4 A graphical interpretation of the combined sampling process

In summary, in order to construct the H-matrix representation of B, we need to compute T_1 in (11.52), S_1 in (11.54) and S_2 in (11.56), a total of 12 column vectors. Each one of these 12 column vectors involves the H-matrix construction of a matrix $\tilde{M}$. Since the entries in $\tilde{M}$ have the same asymptotic behavior as the entries in B, it is reasonable to assume these two matrices have the same hierarchical structure or rank map. It should be noted that matrix $\tilde{M}$ is asymmetric. Hence unlike treatment of matrix B in section 11.3.1, we have to access
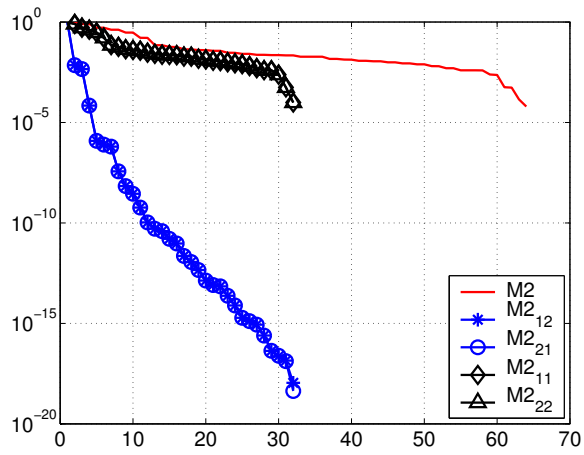
Figure 11-12: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_2$.
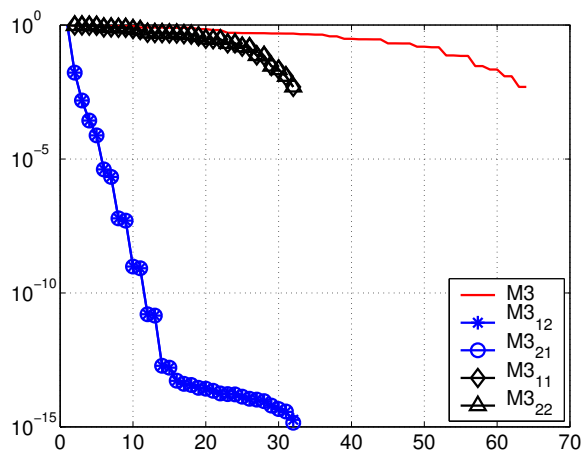
Figure 11-13: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_3$.
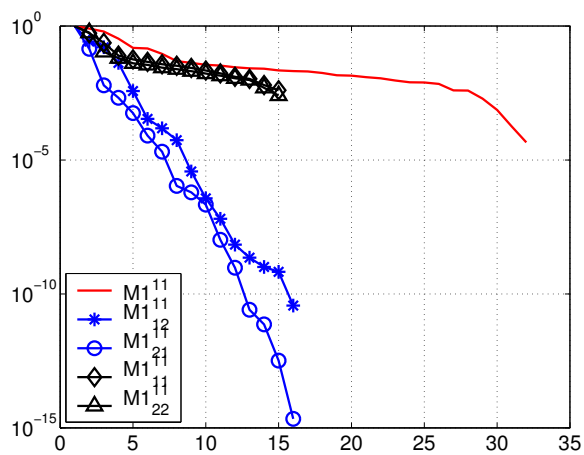
Figure 11-14: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_1^{11}$.
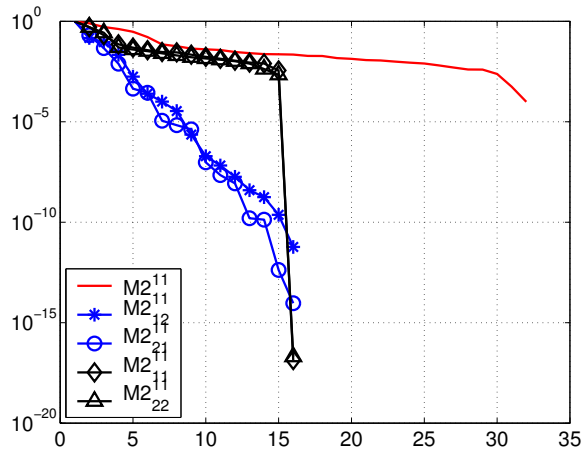
Figure 11-15: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_2^{11}$.
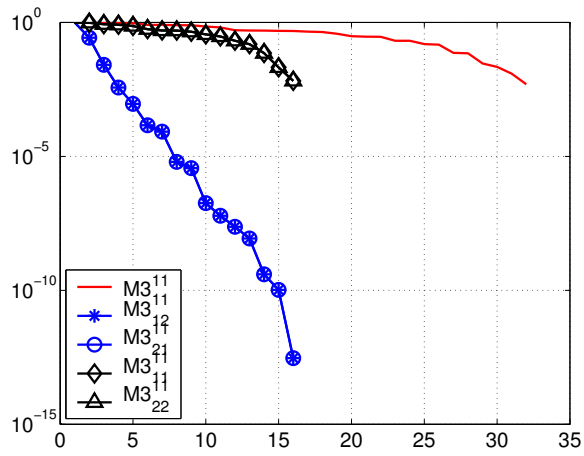
Figure 11-16: Distribution of the singular values of the four sub-blocks of the matrix $\tilde{M}_3^{11}$.
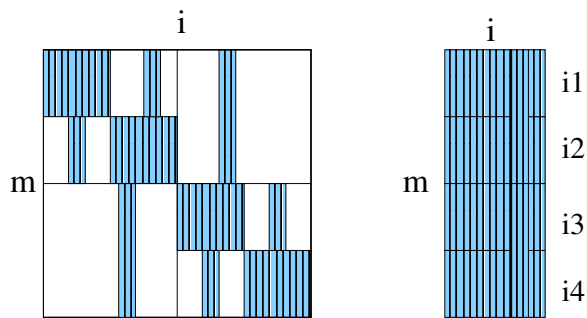
Figure 11-17: Location of sampled columns in matrix B and their compressed-row format. Due to symmetry of B, only columns need to be sampled. Notice that each column in the compressed-row format has at most 4 unique values of i.

12 columns and 12 rows in $\tilde{M}$ to construct its sparse representation. [6]

The column sampling of matrix B is shown in figure 11-17. Figure 11-18 shows the relation between column sampling for B and hierarchical structure of $\tilde{M}$. And figure 11-19 shows the sampled entries in matrix $\tilde{M}$ and their relation to the entries of matrix F^{ij}. These are three main components in the combined sampling process.

11.4   Construction of H-matrix for matrix B: General cases

In practice, the number of levels in the H-matrix representation of matrix B could be more than 10 and the rank of different matrix R_i could be different. For example, the rank map and H-matrix structure of the system matrix for a one-dimensional bar and a three-dimensional plate are shown in figures 11-20 and 11-21, respectively. Fortunately the ideas presented in section 11.3 still apply to these more general cases. In this section we show a systematic way of constructing the H-matrix for general matrix B based upon these ideas.

11.4.1   Panel ordering

It is worth mentioning that the hierarchically low-rank structures in figures 11-20 and 11-21 depend critically on panel ordering. A panel clustering algorithm described in [5] can

---

[6]Unlike constructing an H-matrix for a $\frac{1}{r}$ kernel, we have no access to individual entries of B and $\tilde{M}$. Hence we can not come up with a rank map before hand. In this paper we will use the rank map of $\bar{A}$ to guide the construction of H-matrix for both B and $\tilde{M}$. This is primarily because these three matrices share the same $\frac{1}{r}$ asymptotic behavior and the same set of panels, the ones used to discretize the nominal smooth surface.
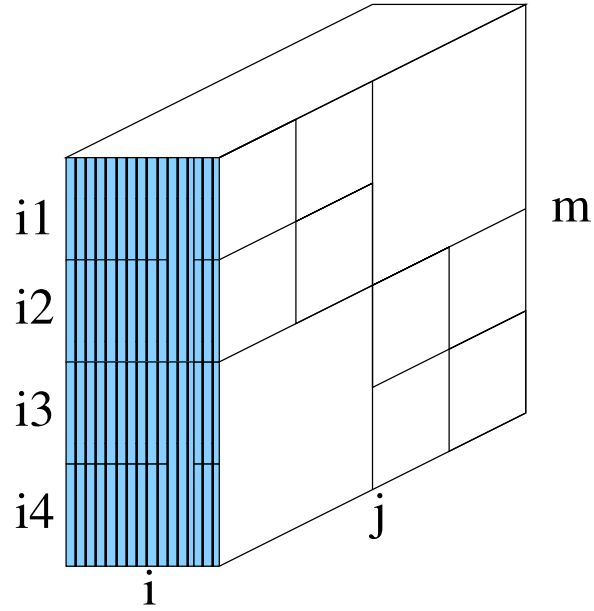
Figure 11-18: Relation between column sampling for B (on $m-i$ plane) and hierarchical structure of $\tilde{M}$ (on $m-j$ plane). Each slice on $m-j$ plane represents one matrix $\tilde{M}$ and is to be multiplied with $<\rho^{(0)}>$ to obtain the corresponding sampled column on $m-i$ plane . Number of slices is equal to number of sampled columns in figure 11-17.
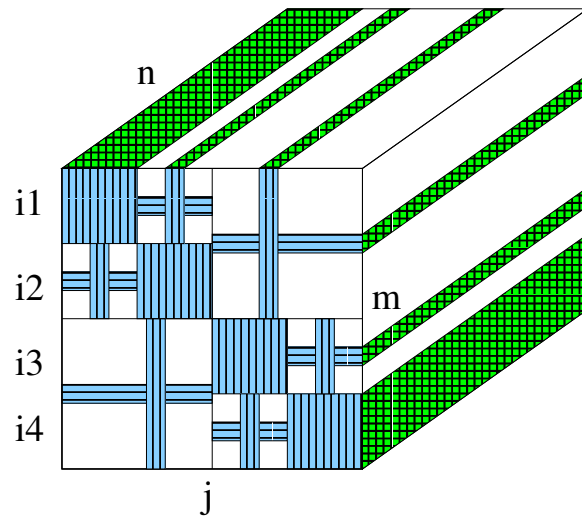
Figure 11-19: Location of sampled columns and rows in matrix $\tilde{M}$ (front $m-j$ plane) and their relation to entries in matrix F^{ij}. Each "thread" along index n direction represent one row in matrix F^{ij}. The entry $\tilde{M}_{mj}$ in the shaded region on the $m-j$ plane is the result of inner product between the "thread" and $<\rho^{(0)}>$, i.e., it is the m-th entry of the column f^{ij} defined in (11.22).
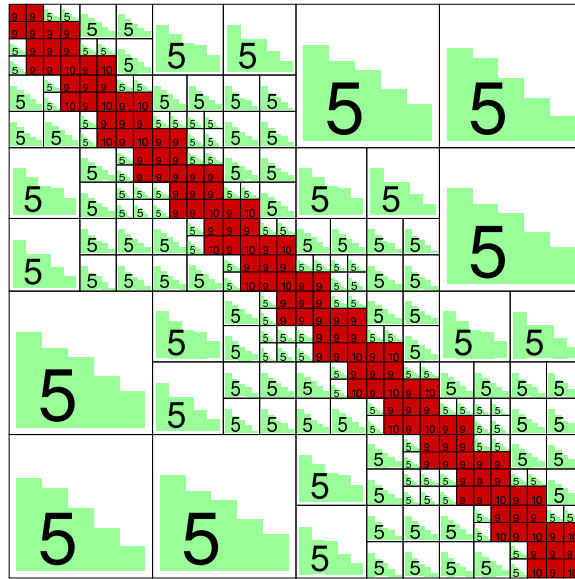
Figure 11-20: Rank map and H-matrix structure of the system matrix for a one-dimensional bar discretized with 300 segments. The shaded blocks are full-rank sub-matrices, other blocks are low-rank sub-matrices.
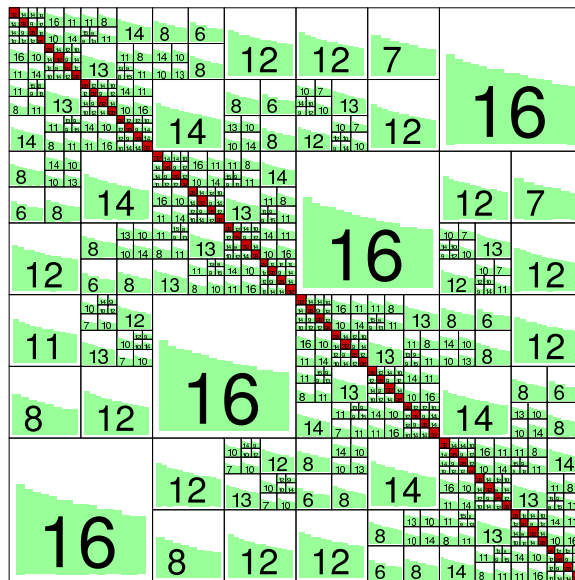
Figure 11-21: Rank map and H-matrix structure of the system matrix for a three-dimensional plate discretized with 1800 triangle panels. The shaded blocks are full-rank sub-matrices, other blocks are low-rank sub-matrices.

order the panels in such a way that the indexes of the spatially close panels are also close. Though it is in general difficult to obtain the optimal ordering for an arbitrary panel distribution, the ordering generated with panel clustering algorithm is good enough for practical purpose. Both figure 11-20 and figure 11-21 are generated using this panel clustering algorithm. Though there are some small low-rank or even full-rank matrices in the off-diagonal region in figure 11-21, these matrices in general do not change the order of computational complexity and memory usage. A heuristic algorithm similar to panel clustering is also presented in [52] and has been shown to produce satisfactory ordering for realistic 3D structures.

11.4.2 Sampling matrices

As explained in section 11.3, we have two sparse sampling matrices to fill, one for matrix B as shown in figure 11-17, and the other for matrix $\tilde{M}$ as shown in figure 11-19. The location of the nonzero entries of both matrices are derived from the same rank map of the matrix $\bar{A}$. To facilitate the following discussion, we denote the sparse matrices that contains the location of the sampling entries in figure 11-17 and 11-19 as S_B and S_M, respectively.

Though the graphical interpretation of the combined sampling process shown in figure 11-17, 11-18 and 11-19 are derived from a two-level H-matrix structure shown in figure 11-10, it turns out that realistic cases are not very far from these pictures. For example, matrices S_B and S_M derived from the H-matrix structure shown in figure 11-21 are shown in figure 11-22 and 11-24, respectively. The compressed-row format of matrix S_B is shown in figure 11-23. These pictures are very similar to figure 11-17, 11-18 and 11-19. Hence the idea of combined sampling is readily applicable to general cases.

11.4.3 Algorithm outline

Now we are ready to put together an algorithm to construct H-matrix of B. The outline of this algorithm is shown in algorithm 4.
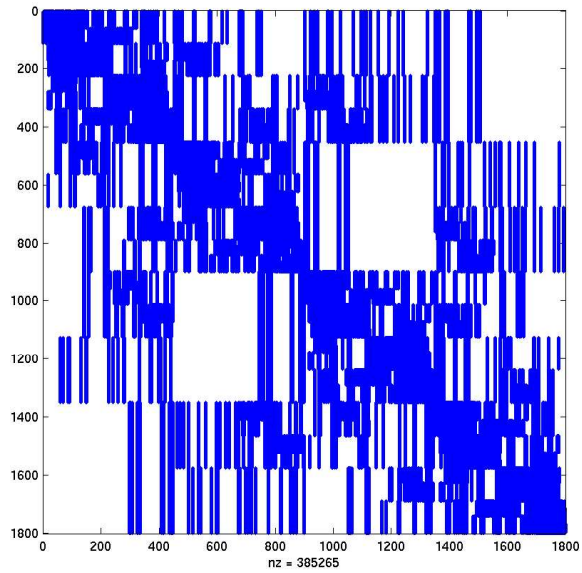
Figure 11-22: Location of the sampled columns used for constructing H-matrix of matrix
B
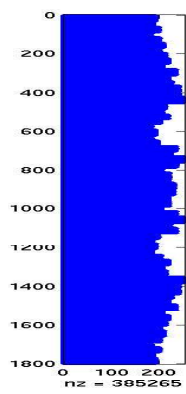
Figure 11-23: Compressed-row format of the sparse matrix in figure 11-22
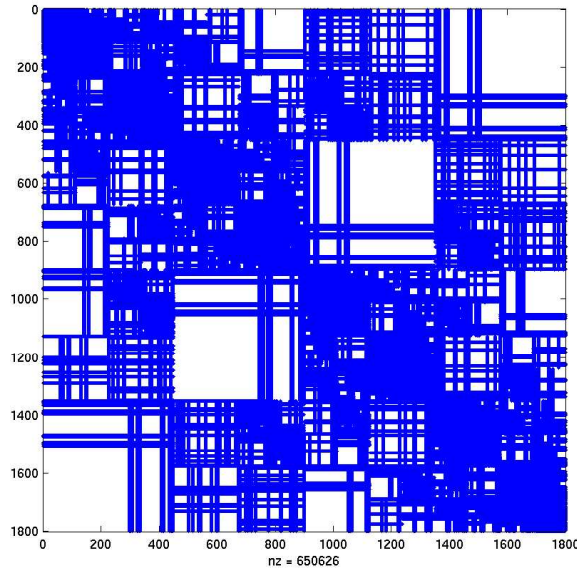
Figure 11-24: Location of the sampled entries used for constructing H-matrix of matrix $\tilde{M}$

Algorithm 4: Construct H-matrix for B.
Input: rank map of $\bar{A}$
Output: H_B
(1)   construct sparse matrix S_B and S_M from rank map of $\bar{A}$
(2)   **foreach** column k in S_B
(3)   construct H-matrix $H_{\tilde{M}} \simeq \tilde{M}$ from sampled entry $\tilde{M}_{mj} = f_m^{ij} = F^{ij}(m,:) < \rho^{(0)} >$, $i = S_B(m,k)$ and location of $\tilde{M}_{mj}$ is given by S_M.
(4)   compute $y = H_{\tilde{M}} < \tilde{\rho}^{(0)} >$
(5) $B(m,S_B(m,k)) = y_m$
(6)   construct H-matrix $H_B \simeq B$ from sampled entries $B(m,S_B(m,k))$

11.5   Construction of H-matrix for matrix B: A flaw and its fix

The simplified combined sampling process in figures 11-17, 11-18 and 11-19 as well as more general case in figures 11-22, 11-23 and 11-24 all suggest an easy estimation of the computational complexity of the algorithm. By definition, the number of non-zeros in sparse matrix S_B is equal to the number of sampling entries in the Hierarchical SVD or H-matrix. It has been shown that this number is $O(Nlog(N))$ [52, 5]. Since there is no empty row in S_B and each row should have roughly the same number of non-zeros, as shown in

147

figure 11-23, the average number of nonzeros per row or the number of columns in the compressed-row format in figure 11-17 and 11-23 is $O(log(N))$. This is the total number of matrices $\tilde{M}$ in figure 11-19. Construction of the H-matrix representation of each $\tilde{M}$ as well as carrying out matrix-vector product in (11.57) needs $O(Nlog(N))$ work. Hence the total work is $O(Nlog^2 N)$.

However, this seemingly reasonable estimation is incorrect and algorithm 4 actually needs $O(N^2)$ work. In order to appreciate this subtlety and, more importantly, to measure the effectiveness of the modified combined sampling process proposed in this section, we want to introduce a very useful concept, the **foot print** of the combined sampling process.

11.5.1   The foot print of the combined sampling process

As shown in previous sections, we have chosen to directly compute F_{mn}^{ij} in the process of constructing the H-matrix for B. As shown in (10.49), calculation of F_{mn}^{ij} involves the calculation of $A_{ij}, A_{mn}, \bar{A}_{ij}$ and $\bar{A}_{mn}$. The union of these entries is the so-called **foot print** in matrix A and $\bar{A}$. This **foot print** concept can be used as the metrics of the sparsity achieved by the proposed fast SIE method in this paper. If all entries in matrix A and $\bar{A}$ are used, we end up with a full **foot print**. If $O(N)$, $O(Nlog(N))$ or $O(Nlog^2(N))$ entries in matrix A and $\bar{A}$ are used, we end up with a sparse **foot print**. Obviously the sparser the **foot print**, the more efficient the fast SIE method. The final computational complexity is dictated by the sparsity of the **foot print** achieved.

11.5.2   The flaw

Unfortunately, direct application of the combined sampling process in algorithm 4 leads to a full foot print. As shown in figure 11-1 and explained in section 11.1, most rows in matrix F^{ij} are actually very sparse. But for a given i and j, the m-th row of F^{ij} will be full if $m \in near(i)$ or $m \in near(j)$. From figure 11-17, It is easy to check that all the diagonal entries of B are to be sampled [7] and they all satisfy $m \in near(i)$ because $m = i$. The

---

[7]This is inevitable for all sparsification techniques, including Fast Multipole, Pre-corrected FFT, Hierarchical SVD and Hierarchical Matrix. This is because these methods all compute nearby interactions directly, which means they all have to sample the diagonal blocks.

same holds true for each $\tilde{M}$ in figure 11-19. Hence we at least have N distinct full-length "threads" in figure 11-19, where N is the number of panels. Since each full-length "thread" leaves N nonzeros on the foot print of both A and $\bar{A}$, the foot print is full.

The remedy lies in the H-matrix construction of another combined matrix. The key idea is to pick out entries B_{mi} and $\tilde{M}_{mj}$ such that $m \in near(i)$ and $m \in near(j)$ and assemble them into two separate matrices, just like B and $\tilde{M}$ and re-run the combined sample process.

11.5.3 Union of full-length vectors is hierarchically low-rank

To explain the fix to the above-mentioned flaw, we need to first take another look at the characteristics of the matrix entry F_{mn}^{ij}. Since those full-length "threads" are rows in the horizontal sub-regions of either region I or region II and we intend to treat them in a unified way, we will analyze the characteristics of these two sub-regions together. Recall the indexes of the horizontal sub-regions of region I satisfy

$$m \in near(i), \ n \in far(i), \ n \in far(m), \ n \in far(j) \tag{11.61}$$

and the indexes of the horizontal sub-regions of region II satisfy

$$m \in near(j), \ n \in far(j), \ n \in far(m), \ n \in far(i). \tag{11.62}$$

Here we have relaxed the condition $i \in far(j)$ we have used for these two sub-regions in section 11.2.1 because we want to treat them as if they are one region here. Hence we will not use the simplification in (11.17) and (11.18). We have also excluded the region III from our analysis even though entries in it are part of the full-length rows. This is because this region only accounts for a very small number of entries and the values of those entries tend to be smaller than entries in other regions. Hence their contribution is negligibly small. Inaccuracy in the treatment of this region is expected to have no impact on all-over accuracy.

Similar to (10.22), $n \in far(m)$ in (11.61) and (11.62) implies that

$$P_2(h_m, h_n; x_m, x_n) \simeq P_1(h_m) P_1(h_n).$$ (11.63)

Similar to (11.17) and (11.18), (11.61) and (11.62) imply that

$$P_4(h_i, h_j, h_m, h_n; \bar{x}_i, \bar{x}_j, \bar{x}_m, \bar{x}_n) \simeq P_3(h_i, h_j, h_m; \bar{x}_i, \bar{x}_j, \bar{x}_m) P_1(h_n).$$ (11.64)

We follow the notation in (11.32) and (11.40) and call their union as H^{ij}. Substituting (11.63) and (11.64) into (11.3) and (10.49) results in

$$
\begin{aligned}
H^{ij}(m,n) \simeq\ & \int_{-\infty}^{+\infty} dh_n \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \\
& (P_3(h_i, h_j, h_m; \bar{x}_i, \bar{x}_j, \bar{x}_m) P_1(h_n) \\
& - P_2(h_i, h_j; \bar{x}_i, \bar{x}_j) P_1(h_m) P_1(h_n)) A_{ij} A_{mn} \\
=\ & \int_{-\infty}^{+\infty} dh_m P_1(h_m) \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j A_{ij} \\
& \left[\frac{P_3(h_i, h_j, h_m; \bar{x}_i, \bar{x}_j, \bar{x}_m)}{P_1(h_m)} - P_2(h_i, h_j; \bar{x}_i, \bar{x}_j) \right] \\
& \int_{-\infty}^{+\infty} dh_n P_1(h_n) A_{mn}. \\
=\ & \int_{-\infty}^{+\infty} dh_m P_1(h_m) \alpha^{ij}(h_m) \\
& \int_{-\infty}^{+\infty} dh_n P_1(h_n) A_{mn},
\end{aligned}
$$ (11.65)

where

$$
\begin{aligned}
\alpha^{ij}(h_m) =\ & \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j A_{ij} \\
& \left[\frac{P_3(h_i, h_j, h_m; \bar{x}_i, \bar{x}_j, \bar{x}_m)}{P_1(h_m)} - P_2(h_i, h_j; \bar{x}_i, \bar{x}_j) \right].
\end{aligned}
$$ (11.66)

Here we have also used the definition of the entries of $\bar{A}$ in (10.21). Since $\alpha^{ij}(h_m)$ is unrelated to n, it does not interfere with the asymptotic behavior of A_{mn}. Hence matrix H_{ij} is hierarchically low-rank, just like $\bar{A}$ or A.
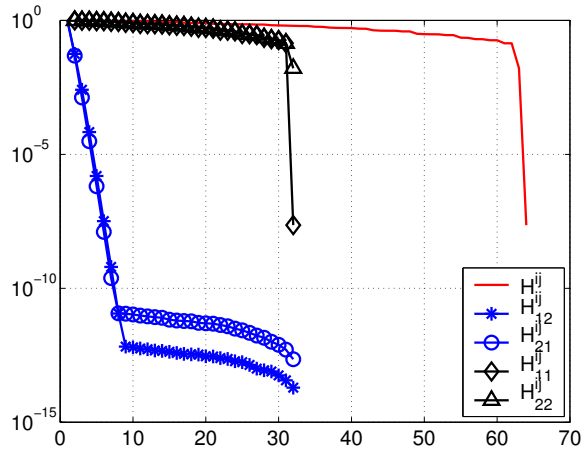
Figure 11-25: Distribution of the singular values of the four sub-blocks of the matrix H^{ij} for $m = i$ and $j = i + 1$.
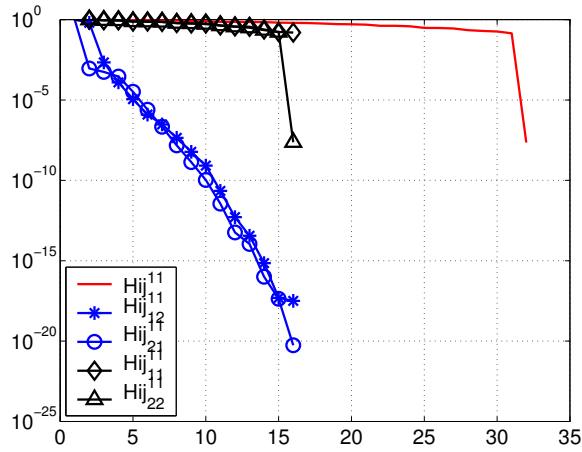
Figure 11-26: Distribution of the singular values of the four sub-blocks of the diagonal block H_{11}^{ij} in figure 11-25.

We have used two 1D bar in parallel as a simple example to verify our analysis. Each bar is divided along the length into 32 panels. The singular values of the 4 sub-blocks of the matrix H^{ij} and one of its diagonal sub-blocks, H_{11}^{ij}, are shown in figures 11-25 and 11-26. It is clear that matrix H^{ij} is hierarchically low-rank.

11.5.4   The fix to the flaw: A modified combined sampling process

The hierarchically low-rank property of H^{ij} suggests that we can again use the H-matrix technique to avoid filling the full matrix H^{ij}. In order to maximize the efficiency, we just

151

need to carefully collect the appropriate full-length vectors, assemble them into matrix H^{ij} and carry out the matrix vector product to get the corresponding entries in $\tilde{M}$.

In order to facilitate the following discussion, we decompose sparse matrix S_B in figure 11-17 into two sparse matrices S_{BF} and S_{BS}, where S_{BF} contains all the entries $S_B(m,i)$ such that $m \in near(i)$ and S_{BS} contains all the remaining entries in S_B. Figure 11-27 shows this decomposition. Further more, we decompose sparse matrix S_M in figure 11-19 into three sparse matrices S_{MF}, S_{MC} and S_{MR}, where S_{MF} contains all the entries $S_M(m,j)$ such that $m \in near(j)$, S_{MC} contains all dense blocks and all the sampling columns, excluding the entries in S_{MF}, S_{MR} contains all the sampling rows, excluding the entries in S_{MF}. Figure 11-28 shows this decomposition.

In view of the combined sampling process in figures 11-17, 11-18 and 11-19, this decomposition effectively turns the single combination of S_B and S_M into six combinations:

1. Combination A: S_{BS} and S_{MC}

2. Combination B: S_{BS} and S_{MR}

3. Combination C: S_{BS} and S_{MF}

4. Combination D: S_{BF} and S_{MC}

5. Combination E: S_{BF} and S_{MR}

6. Combination F: S_{BF} and S_{MF}

By definition, there will be no full-length "threads" in combination A and B. We can just use the same combined sampling process in figures 11-17, 11-18 and 11-19 to compute the corresponding entries of B and fill them to the locations specified by S_{BS}. Also by definition, for all the remaining four combinations, we will obtain nothing but the full-length "threads". It is easy to see from figure 11-27 and 11-28 that these full-length "threads" fit nicely into multiple matrices very similar to the matrix H^{ij} defined in (11.65). Hence we can readily use the H-matrix representation to avoid calculating every entry in these matrices. Since the asymptotic behavior of H^{ij}_{mn} is $\frac{1}{r}$, same as that of the entries in $\bar{A}$, we use rank map of $\bar{A}$ as template for the construction of the H-matrix representation for H^{ij}.
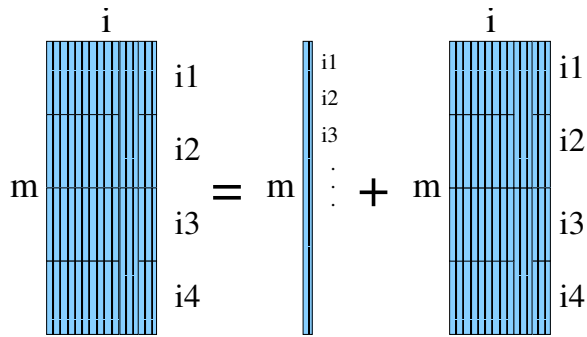
Figure 11-27: Decomposition of sparse matrix S_B in figure 11-17 (to the left of equal sign) into two sparse matrices (to the right of equal sign), S_{BF} contains all the entries $S_B(m,i)$ such that $i \in near(m)$ and S_{BS} contains all the remaining entries in S_B. Both S_{BF} and S_{BS} are in compressed-row format.
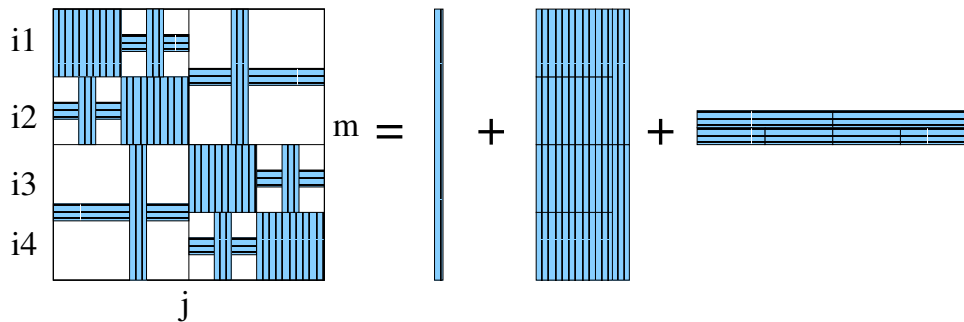
Figure 11-28: Decomposition of sparse matrix S_M in figure 11-19 (to the left of equal sign) into three sparse matrices (to the right of equal sign), S_{MF} contains all the entries $S_M(m,j)$ such that $j \in near(m)$, S_{MC} contains all dense blocks and all the sampling columns, excluding the entries in S_{MF}, S_{MR} contains all the sampling rows, excluding the entries in S_{MF}. Matrix S_{MC} and S_{MF} are in compressed-row format and S_{MR} is in compressed-column format.

In other words, the locations of the sampled entries are specified by sparse matrix S_M and the values of these sampled entries are given in (11.65).

Now we are ready to put together the algorithm for the modified combined sampling process to construct the H-matrix representation of B. The outline of this algorithm is shown in algorithm 5 (page 160).

153

11.5.5 Foot print, CPU time and memory usage of the modified combined sampling process

Since the H-matrix structure of each H^{ij} is identical, combination C, D, E and F all generate the same foot print on matrix A and $\bar{A}$ with $O(Nlog(N))$ nonzeros. Since there are at most $O(Nlog(N))$ distinct nonzeros in S_{BS} and $S_{MC} + S_{MR}$, the foot print of combination A and B is $O(Nlog^2(N))$. Therefore, the foot print of all combinations is $O(Nlog^2(N))$, clearly a sparse one.

By definition, the number of nonzeros per row in S_{BF} and S_{MF} is equal to p, the number of panels within a segment of rough surface of 3 correlation length long. Hence the number of columns in the compressed-row format of both S_{BF} and S_{MF} is p. The number of nonzeros in S_{BS}, S_{MC} and S_{MR} should still be $O(Nlog(N))$, same as that of S_B and S_M. Hence the number of columns in the compressed-row format of S_{BS} and S_{MC}, as well as the number of rows in the compressed-column format of S_{MR}, is $O(log(N))$. The breakdown of CPU time and memory usage by algorithm 5 is shown in table 11.1.

The $O(Nlog^2(N))$ foot print and table 11.1 indicate that the total work involved in the modififed sampling process is $O(Nlog^2(N))$.

11.6 Sparsification of the matrix $\bar{A}^{-1}$

Matrix $\bar{A}$ is the discretized version of the integral operator in (10.13). Many $O(N)$ or $O(Nlog(N))$ algorithms are available to sparsify it, such as Fast Multipole Method [29, 71], Pre-corrected FFT method [80], Hierarchical SVD method [52], and panel clustering method [35]. But these methods do not land themselves directly into a method to sparsify the inverse of $\bar{A}$. Thy are usually used in the inner loop of an iterative linear system solver, such as Gmres [89]. On the other hand, the hierarchical matrix (H-matrix) method developed by Hackbusch [33, 34, 5] is directly applicable to sparsify $\bar{A}^{-1}$. It is shown in [33, 34, 5] that one can construct the H-matrix of the discretized integral operator in almost linear complexity [8]. Using the matrix add, multiplication and rank-k truncation defined in

---

[8]Almost linear complexity means $O(Nlog^k(N))$, i.e., linear up to logarithmic factors

Table 11.1: Estimation of operation count and memory usage by each step in algorithm 5

step	operations	memory
3	$O(Nlog(N)) * p^2$	$O(Nlog(N))$
4	$O(Nlog(N)) * p^2$	$O(N)$
5	$O(N) * p^2$	$O(N)$
7	$O(Nlog(N)) * p * O(log(N))$	$O(Nlog(N))$
8	$O(Nlog(N)) * p * O(log(N))$	$O(N)$
9	$O(N) * p * O(log(N))$	$O(N)$
11	$O(Nlog(N)) * p * O(log(N))$	$O(Nlog(N))$
12	$O(Nlog(N)) * p * O(log(N))$	$O(N)$
13	$O(N) * p * O(log(N))$	$O(N)$
14	$O(Nlog(N)) * p$	$O(Nlog(N))$
15	$O(Nlog(N)) * p$	$O(N)$
16	$O(N) * p$	$O(N)$
20	$O(Nlog(N)) * p * O(log(N))$	$O(Nlog(N))$
21	$O(Nlog(N)) * p * O(log(N))$	$O(N)$
22	$O(N) * O(log(N))$	$O(N)$
24	$O(Nlog(N)) * O(log(N))$	$O(Nlog(N))$
26	$O(Nlog(N)) * O(log(N))$	$O(Nlog(N))$
27	$O(Nlog(N)) * O(log(N))$	$O(Nlog(N))$
28	$O(Nlog(N)) * O(log(N))$	$O(Nlog(N))$
29	$O(Nlog(N))$	$O(Nlog(N))$
31	$O(Nlog(N))$	$O(Nlog(N))$

[33, 34, 5], one can then construct the H-matrix for the approximate matrix inverse from the H-matrix of the discretized integral operator in almost linear complexity. In this paper we directly use the H-matrix method to sparsify both $\bar{A}$ and $\bar{A}^{-1}$. We shall not give the details of this method here. Please refer to [5] and the references cited therein for theoretical as well as implementational details.

11.7   Computing $trace(A^{-1}B)$

Having found the H-matrix representation for matrix $\bar{A}^{-1}$ and B, we are ready to present an efficient algorithm to compute $trace(\bar{A}^{-1}B)$. As explained in [5], the clustering tree for $\bar{A}^{-1}$ is derived from that for $\bar{A}$. Because both $\bar{A}$ and B are derived from the same set of panels on the same surfaces and they share the same asymptotic behavior, it is safe to assume that the panel clustering trees for them are the same. Due to block matrix manipulation such as add, multiplication and rank-k truncation, the clustering tree for $\bar{A}^{-1}$ can be different from that of $\bar{A}$ at leave level, though most of the two trees should be the same. Hence matrix $\bar{A}^{-1}$ and B may or may not have same leave blocks. We will cover both cases in this section.

Block partition   We will first present a generic approach. Suppose we want to compute $trace(PQ)$, where $P \in R^{N \times N}$ and $Q \in R^{N \times N}$ and their entries are arbitrary. Using a natural block partition, we have

$$
\begin{aligned}
trace(PQ) &= trace(\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}) \\
&= trace(\begin{bmatrix} P_{11}Q_{11}+P_{12}Q_{21} & * \\ & P_{21}Q_{12}+P_{22}Q_{22} \end{bmatrix}) \\
&= trace(P_{11}Q_{11})+trace(P_{12}Q_{21}) \\
&\quad +trace(P_{21}Q_{12})+trace(P_{22}Q_{22}).
\end{aligned}
\tag{11.67}
$$

If we keep on recursively partitioning in this fashion, we can compute the result in $O(N^2)$ time. This, of course, is the same as when we compute the *trace* directly. But we can do

156

much better this way if both P and Q have low-rank representation. Let P_s and Q_s be one of the sub-blocks at certain level down to the partition tree, just like P_{11} and Q_{11} in (11.67).

Low-rank matrix block times low-rank matrix block   Assume we have the low-rank representation for both P_s and Q_s

$$P_s = U_P V_P^T, \ \ P_s \in R^{M \times M}, \ \ U_P, V_P \in R^{M \times k}$$
$$Q_s = U_Q V_Q^T, \ \ Q_s \in R^{M \times M}, \ \ U_Q, V_Q \in R^{M \times k}, \quad\quad (11.68)$$

where M is the size of blocks P_s and Q_s and k is the approximate rank of P_s and Q_s, then we have

$$trace(P_s Q_s) = trace([U_P (V_P^T U_Q)] V_Q^T). \quad\quad (11.69)$$

If we compute the *trace* in the order enforced by the parenthesis in (11.69), then it only takes $2Mk^2 + Mk$ work. This is comparable to the cost of matrix-vector product, $2Mk$.

Low-rank matrix block times full-rank matrix block   In some cases, one of the two blocks P_s and Q_s does not have low-rank representation. Without loss of generality, we assume P_s is in low-rank representation as in (11.68) and Q_s has full rank . It is easy to check that computing $trace(P_s Q_s)$ takes $M^2 k + Mk$ work. This is comparable to the cost of full-rank matrix-vector product, M^2. It should be noted that for the rare cases when both P_s and Q_s do not have low-rank representation, the calculation of *trace* takes M^2 work, a lower cost than that of low-rank matrix times full-rank matrix. Hence this rare case should not complicate our analysis.

Total cost   It is shown in [33, 34, 5] that the matrix vector product takes $O(NlogN)$ work using the Hierarchical Matrix representation. As shown above, the cost of computing *trace* is different from that of matrix vector product by a factor k. Since in general $k << M$ and it is a constant independent of M, computing the *trace* also takes $O(NlogN)$ work.

11.8 Ensemble average multiple integration

In order to further reduce the CPU time of constructing the H-matrix for matrix B, we have used the following simple manipulation

$$
\begin{aligned}
< A_{ij}A_{mn} > \quad &\simeq \quad \int_{-\infty}^{+\infty} dh_i \int_{-\infty}^{+\infty} dh_j \int_{-\infty}^{+\infty} dh_m \int_{-\infty}^{+\infty} dh_n \\
&\quad P_4(h_i,h_j,h_m,h_n;x_i,x_j,x_m,x_n)A_{ij}A_{mn}, \\
&= \quad \int_{-\infty}^{+\infty} dh_i P_1(h_i) \int_{-\infty}^{+\infty} dh_j P_1(h_j) \\
&\quad \int_{-\infty}^{+\infty} dh_m P_1(h_m) \int_{-\infty}^{+\infty} dh_n P_1(h_n) \\
&\quad f(h_i,h_j,h_m,h_n), \quad\quad\quad\quad\quad\quad\quad\quad\quad (11.70)
\end{aligned}
$$

where

$$
f(h_i,h_j,h_m,h_n) = \frac{P_4(h_i,h_j,h_m,h_n;x_i,x_j,x_m,x_n)}{P_1(h_i)P_1(h_j)P_1(h_m)P_1(h_n)}A_{ij}A_{mn} \quad\quad (11.71)
$$

is the new integrand. Obviously, the approximations in (11.17),(11.18),(11.26), (11.31),(11.35) and (11.39) should be used in (11.71) whenever appropriate. Since the form of probability density function P_1 is the same as that of the weighting function for the standard Gauss-Hermite quadrature [91], we can directly use Gauss-Hermite quadrature for each fold of integration in (11.70). Due to the smoothness of the integrand in (11.71), each fold of integral in (11.70) should only need about $q = 5$ Gauss-Hermite quadrature points. Hence the total number of function evaluation in (11.71) is $q^4 = 625$.

Monte Carlo method is usually the only alternative to compute the high dimensional integral [22]. However, the square root convergence rate of Monte Carlo method may require no fewer than a few thousands function evaluations in (11.71). So Gauss-Hermite quadrature is about five to ten times faster than Monte Carlo method.

In addition, since we know the quadrature points a priori, we can pre-compute matrix entry A_{ij} for different h_i and h_j and use it to compute the integrals in (11.70). As explained in section 11.5, the foot print in A is $O(Nlog^2(N))$. Hence only $O(Nlog^2(N))$ unique entries in A need to be pre-computed, where N is number of panels. Essentially, the time-

consuming panel integration is moved out of the inner loop in computing the four-fold integral in (11.70). On the other hand, if we use Monte Carlo method in (11.70), h_i, h_j, h_m and h_n would be completely random and uncorrelated. Hence it is impossible to do any pre-computation of A_{ij} and A_{mn}. Clearly, Gauss-Hermite quadrature has an even bigger advantage in this aspect.

Algorithm 5: Construct H-matrix for B.

Input: sparse matrices S_{BF}, S_{BS}, S_{MF}, S_{MC} and S_{MR}

Output: H_B

(1)　　**foreach** column k in S_{BF}

(2)　　　**foreach** column l in S_{MF}

(3)　　　　construct H-matrix $H_{H^{ij}} \simeq H^{ij}$, $i = S_{BF}(m,k)$, $j = S_{MF}(m,l)$.

(4)　　　　compute $y = H_{H^{ij}} < \tilde{\rho}^{(0)} >$

(5)　　　　$\tilde{M}(m,j) = y_m$

(6)　　　**foreach** column l in S_{MC}

(7)　　　　construct H-matrix $H_{H^{ij}} \simeq H^{ij}$, $i = S_{BF}(m,k)$, $j = S_{MC}(m,l)$.

(8)　　　　compute $y = H_{H^{ij}} < \tilde{\rho}^{(0)} >$

(9)　　　　$\tilde{M}(m,j) = y_m$

(10)　　　**foreach** row l in S_{MR}

(11)　　　　construct H-matrix $H_{H^{ij}} \simeq H^{ij}$, $j = 1:N$, $m = S_{MR}(l,j)$,$i = S_{BF}(m,k)$.

(12)　　　　compute $y = H_{H^{ij}} < \tilde{\rho}^{(0)} >$

(13)　　　　$\tilde{M}(m,j) = y_j$

(14)　　　construct H-matrix $H_{\tilde{M}} \simeq \tilde{M}$ from sampled entry $\tilde{M}(m,j)$

(15)　　　compute $y = H_{\tilde{M}} < \tilde{\rho}^{(0)} >$

(16)　　　$B(m,S_{BF}(m,k)) = y_m$

(17)

(18)　　**foreach** column k in S_{BS}

(19)　　　**foreach** column l in S_{MF}

(20)　　　　construct H-matrix $H_{H^{ij}} \simeq H^{ij}$, $i = S_{BS}(m,k)$, $j = S_{MF}(m,l)$.

(21)　　　　compute $y = H_{H^{ij}} < \tilde{\rho}^{(0)} >$

(22)　　　　$\tilde{M}(m,j) = y_m$

(23)　　　**foreach** column l in S_{MC}

(24)　　　　compute $\tilde{M}_{mj} = f_m^{ij} = F^{ij}(m,:) < \rho^{(0)} >$, $i = S_{BS}(m,k)$, $j = S_{MC}(m,l)$

(25)　　　**foreach** row l in S_{MR}

(26)　　　　compute $\tilde{M}_{mj} = f_m^{ij} = F^{ij}(m,:) < \rho^{(0)} >$, $j = 1:N$, $m = S_{MR}(l,j)$, $i = S_{BS}(m,k)$

(27)　　　construct H-matrix $H_{\tilde{M}} \simeq \tilde{M}$ from sampled entry $\tilde{M}(m,j)$

(28)　　　compute $y = H_{\tilde{M}} < \tilde{\rho}^{(0)} >$

(29)　　　$B(m,S_{BS}(m,k)) = y_m$

(30)

(31)　　construct H-matrix $H_B \simeq B$ from sampled entries $B(m,S_{BF}(m,k))$ and $B(m,S_{BS}(m,k))$

Chapter 12

Algorithm Outline of the Stochastic Integral Equation Method

We are ready to put together the ideas discussed in previous chapters to design an algorithm for efficiently computing the mean and the variance of capacitance. The outline of this algorithm is summarized in algorithm 6. Here we assume that the piece-wise constant basis functions and the Galerkin method are used to discretize (10.13). Hence all the approximate equality in (10.46) and (10.62) are exact and $trace(A^{-T}B) = trace(A^{-1}B)$. To facilitate our discussion, we define the following notations:

C_{PI}: number of floating point operations used in calculating matrix entries A_{ij} or A_{mn} in (11.71) and (10.16). For 3D cases, the integration is over a flat triangle panel. Typical operation count is in the order of a few hundreds.

N_1: number of panels or basis functions used by Monte Carlo simulations in (10.31).

N_2: number of panels or basis functions used by SIE method in (10.15).

N_I: number of iterations used by an iterative solver for solving (10.15) or (10.31). For capacitance problem, it is relatively easy to come up with a good pre-conditioner, hence the number of iterations is typically in the order of 10.

N_s: number of Monte Carlo simulations. This number is typically in the order of a few thousands.

q: number of Gauss-Hermite quadrature points used for each fold of the ensemble average integral in (11.70) and (10.16). This number is typically anywhere between 5 and 10.

The estimated operation count and memory usage by each step of the algorithm 6 is shown in table 12.1. It is clear from table 12.1 that algorithm 6 is almost linear in both CPU time and memory usage.

A very important advantage of the SIE method is that bigger panel can be used in discretizing the nominal smooth surface. Empirical studies have shown that a size equal to one-half to one-quarter of correlation length η is appropriate. This is consistent with the findings in the stochastic finite element method [55]. On the other hand, a much finer panel size has to be used in Monte Carlo method in order to have a good resolution of rough surface profile. To see this effect, we have plotted the discretization of a rough surface profile in figure 12-1, where the standard deviation is $\sigma = 0.2$ and correlation length is $\eta = 0.2$. The plot shows that at least 8 panels per correlation length have to be used to ensure a good resolution. On the other hand, only 2 panels per correlation length are sufficient in SIE method. Hence $N_1/N_2 \geq 4$ for 2D rough surfaces, and $N_1/N_2 \geq 4^2 = 16$ for 3D rough surfaces. In addition, triangle panels have to be used to mesh rough 3D surfaces while as quadrilateral panels are sufficient for nominal smooth surfaces. This gives SIE method another factor of 2 in panel number reduction and hence $N_1/N_2 \geq 32$ for 3D rough surfaces.

The breakdown of CPU time and memory usage by Monte Carlo method is listed in table 12.2. Our partial empirical studies have shown that the constant in front of $O(Nlog(N))$ for the H-matrix method is around 40 [1]. But since this number is application-dependent, we can not make a conclusive comparison of CPU time used by SIE method and Monte Carlo method at current stage.

---

[1]This is after we have used an adaptive re-compression technique in [28]. A similar idea was also proposed in [54].
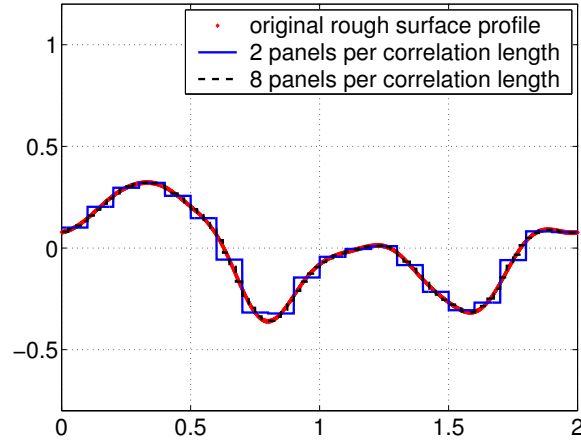
Figure 12-1: Using different mesh size to approximate the profile of a random rough surface, standard deviation is $\sigma = 0.2$, correlation length is $\eta = 0.2$

Algorithm 6: Compute $<C>$ and $var\{C\}$.
Input: nominal smooth surface discretization, rough surface statistics σ and

η

Output: $<C>$, $var\{C\}$

(1)   construct panel clustering tree for the smooth conductor surfaces

(2)   construct sparse matrices S_{BF}, S_{BS}, S_{MF}, S_{MC} and S_{MR} from rank

      map of A

(3)   pre-compute entry $A_{ij}(h_i, h_j)$ on the foot print of A

(4)   construct H-matrix representation $H_A \simeq \bar{A}$

(5)   construct H-matrix representation $H_{AI} \simeq \bar{A}^{-1}$

(6)   iteratively solve $H_A < \tilde{\rho}^{(0)} >= \tilde{L}$ using H_{AI} as preconditioner

(7)   compute $<C^{(0)}> = \tilde{L}^T < \tilde{\rho}^{(0)} >$

(8)   use algorithm 5 to construct H-matrix representation $H_B \simeq B$

(9)   compute $<C^{(2)}> = trace(\bar{A}^{-1}B) \simeq trace(H_{AI}H_B)$

(10)   compute $<C> = <C^{(0)}> + <C^{(2)}>$

(11)   compute $z = B < \tilde{\rho}^{(0)} > \simeq H_B < \tilde{\rho}^{(0)} >$

(12)   compute $var\{C\} \simeq < \tilde{\rho}^{(0)} >^T z$

Table 12.1: Estimation of operation count and memory usage by each step of the algorithm 6

step	operations	memory
1	$O(N_2 log(N_2))$	$O(N_2)$
2	$O(N_2 log(N_2))$	$O(N_2 log(N_2))$
3	$O(N_2 log^2(N_2)) * C_{PI} * q^2$	$O(N_2 log^2(N_2))$
4	$O(N_2 log(N_2))$	$O(N_2 log(N_2))$
5	$O(N_2 log^2(N_2))$	$O(N_2 log(N_2))$
6	$O(N_2 log(N_2)) * N_I$	$O(N_2)$
7	$O(N_2)$	$O(1)$
8	$O(N_2 log^2(N_2)) * q^4$	$O(N_2 log N_2)$
9	$O(N_2 log(N_2))$	$O(N_2 log N_2)$
10	$O(1)$	$O(1)$
11	$O(N_2 log(N_2))$	$O(N_2 log N_2)$
12	$O(N_2)$	$O(1)$

Table 12.2: CPU time and memory usage by Monte Carlo method

step	operations	memory
construct panel clustering tree for rough surfaces	$O(N_1 log(N_1)) * N_s$	$O(N_1 log(N_1))$
sparsify A in (10.31)	$O(N_1 log(N_1)) * C_{PI} * N_s$	$O(N_1 log(N_1))$
solve (10.31)	$O(N_1 log(N_1)) * N_I * N_s$	$O(N_1 log(N_1))$

Chapter 13

Numerical Results

In this chapter, we first use a small 2D example to verify the second-order correction scheme in section 10.6 and the method for calculating capacitance variance in section 10.7. We then use a small 3D example to demonstrate that the stochastic integral equation method can be easily extended to 3D cases. Finally we use a few large 3D examples to verify the accuracy and speed of fast stochastic integral equation solver.

13.1   A small two-dimensional example

The small two-dimensional example is a single circular conductor over ground plane, shown in figure 13-1 The mean radius of the wire is $1mm$ and the radius fluctuation is a stochastic process with respect to the angle in the polar coordinate system. The surface of the ground is assumed to be smooth. The distance between the circular wire with nominal smooth surface and the ground is $0.5mm$.

The mean and variance of the capacitances calculated using different methods are compared in table 13.1 and 13.2, respectively. Column *Smooth* is the capacitance for the conductor with nominal smooth surface. This serves as a reference. Column *Monte Carlo* is the capacitance computed using Monte Carlo simulations. The number of Monte Carlo simulations is shown in the parenthesis. Columns *SIE I* and *SIE II* are the capacitances computed using the stochastic integral equation method without and with the second-order correction term, respectively. The parameters η (the correlation length) and σ (the standard
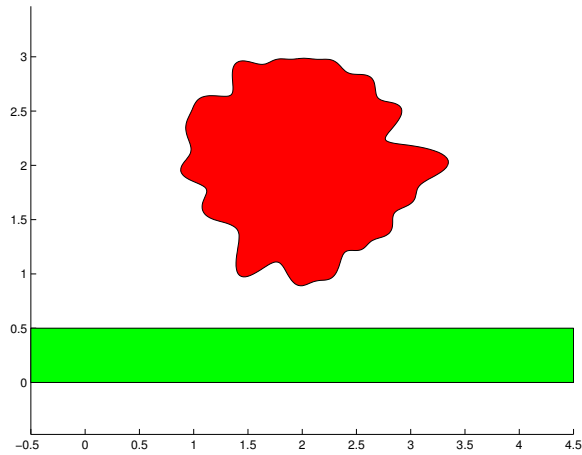
Figure 13-1: A circular wire over ground plane. The mean radius of the wire is 1*mm*. The distance between the circular wire with nominal smooth surface and the ground is 0.5*mm*.
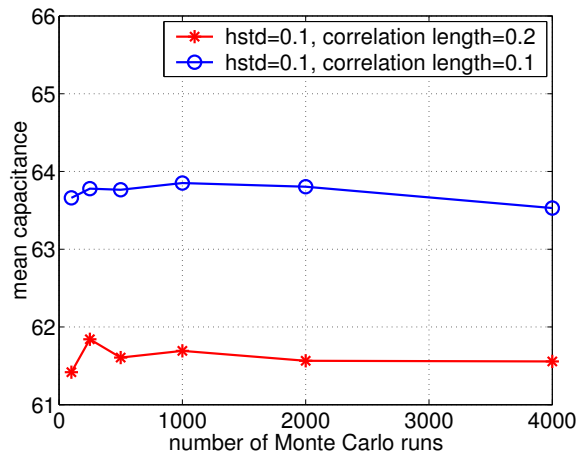
Figure 13-2: Convergence of two-dimensional mean capacitance in Monte Carlo simulations.

deviation) are two numbers we use to control the roughness of the surface. Smaller η or larger σ means a rougher surface. Figure 13-4 compares the detailed mean charge density calculated using different methods.

In order to ensure that the Monte Carlo simulation results are reasonably reliable, we have done the convergence test for both the mean and the variance of capacitance. As can be seen from figure 13-2 and 13-3, Monte Carlo simulation has converged to within 1% using 4000 runs. Hence its results can be used as benchmark to verify the accuracy of the SIE method.

As can be seen from table 13.1, table 13.2 and figure 13-4, the second-order correction
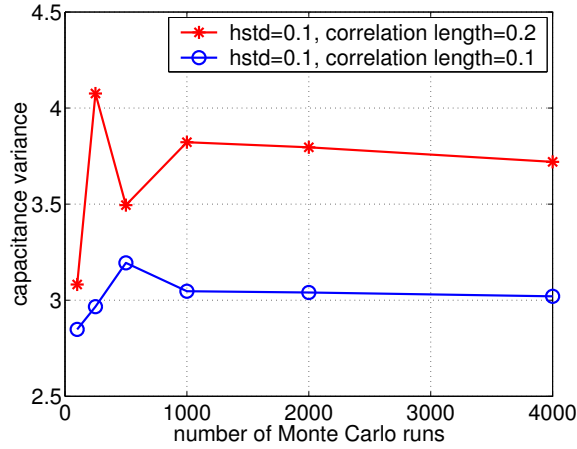
Figure 13-3: Convergence of two-dimensional capacitance variance in Monte Carlo simulations.

Table 13.1: Mean value of 2D capacitance calculated with different methods. Unit:pF. η is the correlation length and σ is the standard deviation. Both are in *mm*.

η	σ	Smooth	Monte Carlo	SIE I	SIE II
0.2	0.1	57.68	61.42(5000run)	58.69	61.19
0.1	0.1	57.68	63.53(5000run)	59.80	64.72

term significantly improves the accuracy and also gives a reasonable capacitance variance estimate. The good agreement between Monte Carlo and SIE II for various roughnesses suggests that the stochastic integral equation method with the second order correction term is a promising approach. It is worth noting that the difference between smooth surface capacitance and the mean capacitance is approximately 10%. Hence the capacitance will be under-estimated if the surface roughness is neglected.

Table 13.2: Variance of 2D capacitance by different methods. Unit:pF. η is the correlation length and σ is the standard deviation. Both are in *mm*.

η	σ	Monte Carlo	SIE II
0.2	0.1	3.72(5000run)	3.00
0.1	0.1	3.02(5000run)	2.24
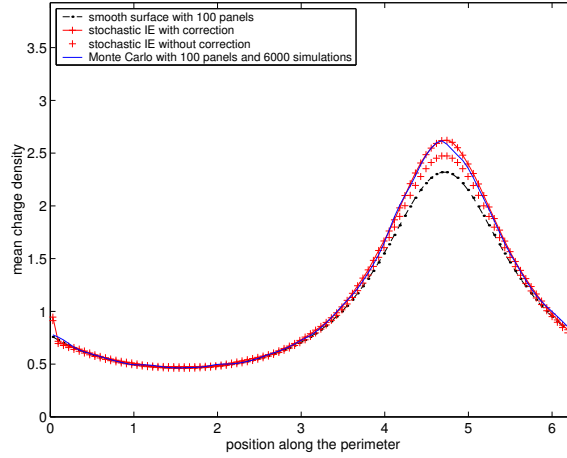
Figure 13-4: The mean charge density computed with Monte Carlo simulations and the stochastic integral equation method. The correlation length is 0.2*mm* and standard deviation is 0.1*mm*. Maximum charge density is around the surface point where the circular wire is closest to the ground.
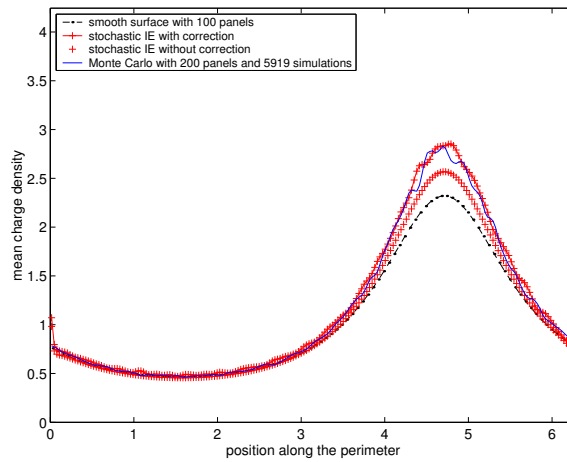
Figure 13-5: The mean charge density computed with Monte Carlo simulations and the stochastic integral equation method. The Correlation length and standard deviation are 0.1*mm*. Maximum charge density is around the surface point where the circular wire is closest to the ground.
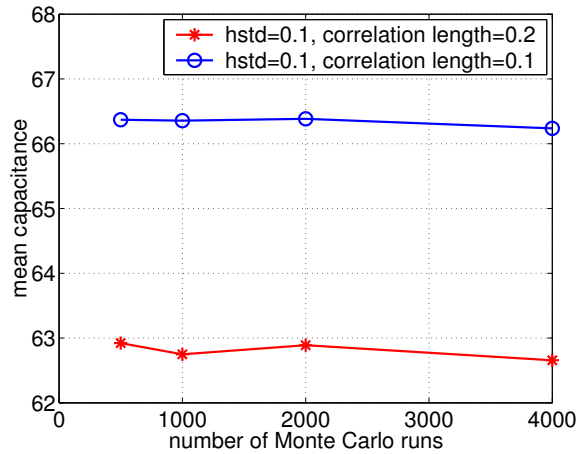
Figure 13-6: Convergence of three-dimensional mean capacitance in Monte Carlo simulations.

13.2 A small three-dimensional example

The small 3D example is a plate of zero thickness over ground plane. The ground plane is assumed to be smooth and the plate has random rough profile, as shown in figure 13-8. The mean and variance of the capacitance calculated using different methods are compared in table 13.3 and 13.4, respectively.

Again, in order to ensure that the Monte Carlo simulation results are reasonably reliable, we have done the convergence test for both the mean and the variance of capacitance. As can be seen from figure 13-6 and 13-7, Monte Carlo simulation has converged to within 1% using 4000 runs. Hence its results can be used as benchmark to verify the accuracy of the SIE method.

It is clear from table 13.3 and 13.4 that the second-order correction improves the accuracy of mean capacitance and gives accurate capacitance variance. It is worth noting that the algorithm and the implementation for 3D structures are the same as those for 2D structures.

13.3 A large three-dimensional example

The results in previous sections are generated using Matlab implementation of the ideas in chapter 10. Specifically, the dense system matrix in (10.15) is explicitly formed and
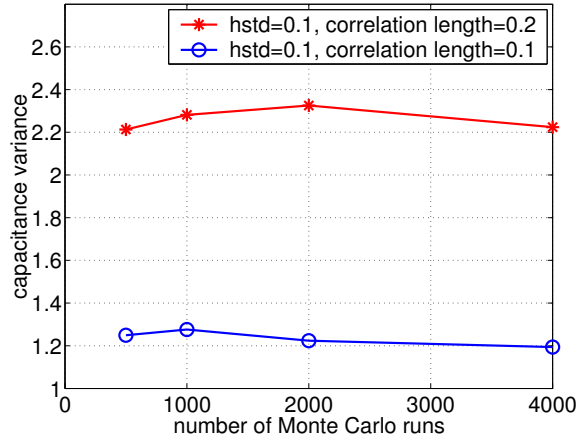
Figure 13-7: Convergence of three-dimensional capacitance variance in Monte Carlo simulations.
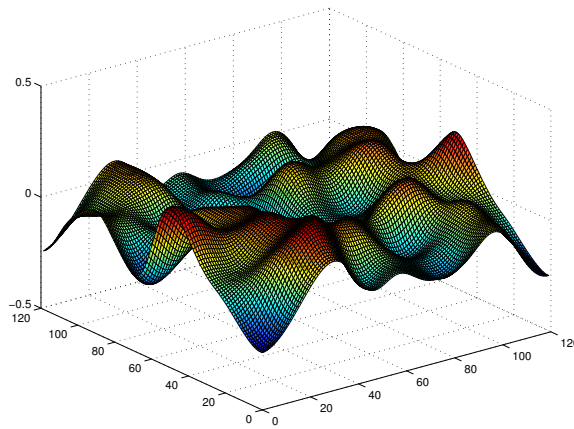
Figure 13-8: A zero-thickness plate with random profile. The correlation length is $0.2mm$ and standard deviation is $0.1mm$. The size of the nominal smooth plate is $1 \times 1mm$. The smooth ground plane is not included in this picture. The distance between nominal smooth plate and the ground plane is $0.5mm$.

Table 13.3: Mean value of 3D capacitance calculated with different methods. Unit:pF. η is the correlation length and σ is the standard deviation. Both are in *mm*.

η	σ	Smooth	Monte Carlo	SIE I	SIE II
0.2	0.1	56.599	62.656(4000run)	61.676	62.706
0.1	0.1	56.599	66.237(4000run)	63.850	65.471

Table 13.4: Variance of 3D capacitance calculated with different methods. Unit:pF. η is the correlation length and σ is the standard deviation. Both are in *mm*.

η	σ	Monte Carlo	SIE II
0.2	0.1	2.224(4000run)	2.011
0.1	0.1	1.194(4000run)	1.370

solved by LU factorization. The matrix *B* and *F* in (10.47) and (10.48) are also explicitly formed. Likewise, the matrix *A* in (10.31) is explicitly formed and solved in each Monte Carlo simulation. For notation convenience, we will call solvers based on three methods in table 13.3 *direct SIE I*, *direct SIE II* and *direct MC*, respectively.

In this section, we use the H-matrix method to sparsify matrix $\bar{A}$ in (10.15) and *A* in (10.31). We have not been able to finish implementing the ideas presented in chapter 11. We will use the name *fast SIE I* for the fast stochastic integral equation (SIE) solver without the second order correction, and the name *fast MC* for the Monte Carlo solver based on H-matrix method.

In order to check the accuracy and speed of *fast SIE I*, we again use 3D plate over ground plane example in figure 13-8. We first use the *fast SIE I* and *fast MC* to analyze the same small $1 \times 1mm$ plate and compare results to those of direct solvers in table 13.5. The good agreement in table 13.5 verifies the accuracy of *fast SIE I* and *fast MC*.

We then use the fast solver to analyze a few larger plates of size $5 \times 5mm$, $10 \times 10mm$, $15 \times 15mm$ and $20 \times 20mm$. The CPU time of *fast SIE I* is compared to $Nlog(N)$ curve in figure 13-9, and the detailed numbers are shown in table 13.6. It is clear from figure 13-9 that the CPU time of *fast SIE I* and *fast MC* (for just one solve of smooth problem) grows as $O(Nlog(N))$. This verifies the speed of *fast SIE I* and *fast MC*.

It should be noted that *fast SIE I* is about 8 to 20 times slower than one smooth problem solve, as shown in figure 13-10. Here we have used 5 Gauss-Hermite quadrature points in each fold of the two-fold ensemble average integral in (10.21). This means that calculating one entry in $\bar{A}$ takes 25 times as much CPU time as does calculating one entry in *A*. However, for larger structures, direct calculating entries of $\bar{A}$ or *A* in forming low-rank H-matrix approximation of matrix $\bar{A}$ and *A* takes progressively smaller portion of the over-

Table 13.5: Mean capacitance of the 3D plate over ground plane in figure 13-8. Unit:pF. Correlation length $\eta = 0.1$, and the standard deviation $\sigma = 0.1$. Both are in *mm*.

Smooth	fast SIE I	direct SIE I	fast MC	direct MC
56.599	63.899	63.850	67.12(4000)	66.237(4000)

Table 13.6: CPU time for a few large 3D plates over ground plane. Unit: second. $\eta = 0.1mm$ and $\sigma = 0.1mm$.

size(*mm*)	#panel	Smooth	fast SIE I
5×5	3200	43	837
10×10	12800	365	5271
15×15	28800	1268	15082
20×20	51200	3763	29638

all CPU time. Other steps, such as low-rank decomposition in section 11.3.1, forming a pre-conditioner, and solving the system with an iterative solver, are the same in *fast SIE I* and *fast MC* and take roughly same amount of CPU time for both solvers. Hence we do not see that *fast SIE I* is 25 times slower than one smooth solve by *fast MC*. On the other hand, since much more refined mesh has to be used in Monte Carlo simulations, each simulation would take longer than one solve of the smooth problem. Assuming four thousands of Monte Carlo simulations are necessary to obtain statistically accurate results, none of the five large examples in table 13.6 can be completed using *fast MC* in less than one week. It is clear that the approach based on *fast SIE I* is much more efficient than Monte Carlo method based on *fast MC*.

Though we are unable to complete thousands of Monte Carlo simulations of the large plates in table 13.6, in order to further test the accuracy of *fast SIE I*, one hundred Monte Carlo simulations of the smallest plate are carried out and the mean capacitance results are shown in table 13.7. The difference between *fast SIE I* and *fast MC* as well as the the difference between *Smooth* and *fast SIE I* are consistent with those in table 13.3. This further verifies the accuracy of *fast SIE I*.
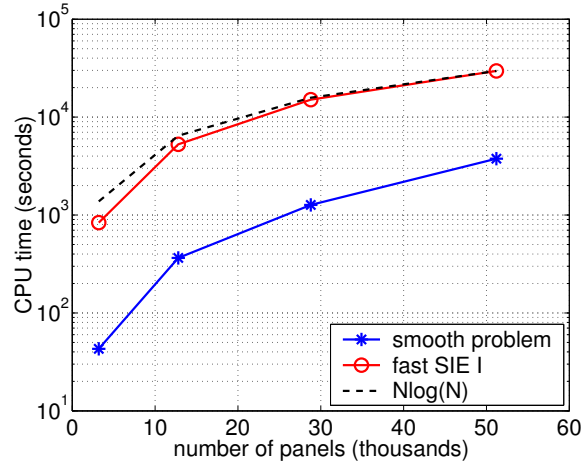
Figure 13-9: CPU time of fast SIE solver without second-order correction.
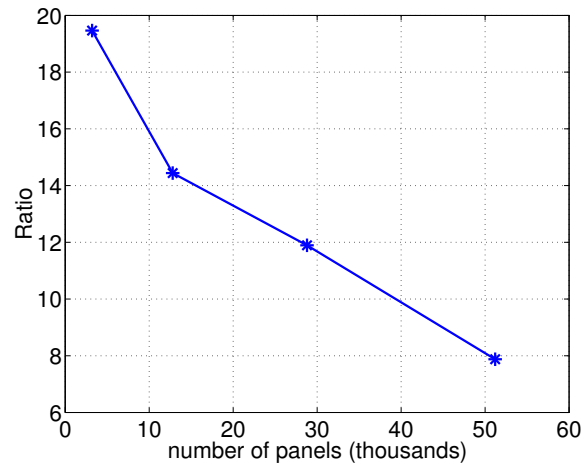
Figure 13-10: Ratio of CPU time used by fast SIE solver and regular fast integral equation solver.

Table 13.7: Mean capacitance of a few large 3D plates over ground plane. Unit:pF. $\eta = 0.1mm$ and $\sigma = 0.1mm$.

size(mm)	#panel	Smooth	fast SIE I	increase	fast MC
5×5	3200	683	741	8.47%	811(100 runs)
10×10	12800	2305	2484	7.81%	N/A
15×15	28800	4837	5199	7.48%	N/A
20×20	51200	8273	8876	7.29%	N/A

Chapter 14

Conclusions and future work

14.1 Conclusions

This dissertation has two main parts.

In part I, we have derived a recently developed surface integral formulation from a different perspective. Using a piecewise quadrature scheme to improve the accuracy of panel integration, we have fixed the low-frequency problem in the original formulation. Using a scaling technique and a local preconditioner, we have improved the accuracy and memory efficiency of the formulation. We have also generalized the pre-corrected FFT algorithm to allow the acceleration of complicated integral operators. Based on this generalization we have developed a flexible and extensible fast integral equation solver, pfft++. With 4 to 5 digit accuracy at modest computational cost and nearly $O(N)$ computational complexity, pfft++ can be easily applied to a wide range of engineering problems. Using pfft++ as the engine, we have developed a fast impedance extraction program, FastImp. Numerical examples show that FastImp can efficiently and robustly perform wideband electromagnetic analysis of general 3D structures. Both pfft++ and FastImp are now available at www.rle.mit.edu/cpg/research_codes.htm.

In part II, we have demonstrated an efficient stochastic integral equation (SIE) method for calculating the mean and the variance of capacitance of both 2D and 3D structures. This method has two advantages: 1) It avoids the time-consuming Monte Carlo simulations; 2) It avoids the discretization of rough surface, which needs much more refined mesh than

smooth surface. Based upon the hierarchical matrix method, we have proposed a combined sampling process to reduce the overall computational cost to $O(Nlog^2(N))$.

14.2   Future work

There are a number of directions we can go after FastImp and the stochastic integral equation (SIE) method.

1. 3D interconnects embedded in layered media

   The formulation used in FastImp was derived from scalar Helmholtz equations and uses $\vec{E}$ and $\frac{\partial \vec{E}}{\partial n}$ as its unknowns. This makes it very difficult to use the well-established multi-layered media Green's functions [67, 1] for the analysis of 3D structures embedded into multi-layered dielectric media. To remedy the situation, a new surface integral formulation derived from vector Helmholtz wave equations was proposed in [92]. The unknowns in this formulation are $\vec{E}$ and $\vec{H}$ on the conductor surface. This formulation is closely linked with the well-known Stratton-Chu formulation or EFIE and MFIE formulation. Hence the well-established techniques like RWG linear basis function [84], loop-star basis transformation [64, 102] and frequency normalization [113] can be used to reduce the number of unknowns and to improve the accuracy at low frequencies. Though layered Green's function, RWG basis and loop-star basis transformation are well-understood, combining them with an appropriate fast integral equation solver is not trivial at all. In addition, the layered Green's function techniques can only handle stratified media with each layer being homogeneous and interfaces between different layers being parallel to each other. These two conditions are not necessarily met by the realistic fabrication process. So either some approximation has to be made and new techniques have to be developed.

2. Model order reduction

   FastImp calculates the impedance at a given frequency point. A natural next step would be to combine it with model-order reduction techniques [73, 51, 60, 31] so that we can extract the equivalent circuit models valid across wide frequency range. The

difficulty lies in the fact that the system matrix generated from the surface integral formulation is frequency dependent. And this frequency dependency can not be written as a simple multiplicative form, which is the case in the Partial Element Equivalent Circuit (PEEC) model [40] used by FastHenry [50, 49]. Frequency-dependent model-order reduction is still an open problem [19]. It would be interesting to use the formulation in FastImp to drive the new development in this area.

3. Modeling and simulation of 3D interconnects with rough surfaces

Though only the capacitance extraction is used in part II of this dissertation to demonstrate the basic ideas of fast SIE, it is very conceivable to use the fast SIE method in FastImp. This is similar to the extension of the Pre-corrected FFT method from the capacitance extraction in [80] to the impedance extraction in FastImp. A further step is to develop a model order reduction technique compatible with the SIE method so we can simulate the impact of the interconnect surface roughness on the behavior of VLSI circuits.

Appendix A

Detailed forms of the ensemble average Green's function

The form of the ensemble average Green's function in (10.13) is dependent upon the position of source and evaluation points. We enumerate different combinations of source point position (x', y') and evaluation point position (x, y).

1. $(x, y) \in S_1$ or S_3; $(x', y') \in S_1$ or S_3

$$< \tilde{G}(x', y'; x, y) >= K_1(x', y'; x, y) = G(x', y'; x, y) \tag{A.1}$$

2. $(x, y) \in S_1$ or S_3; $(x', y') \in \tilde{S}_2$

$$
\begin{aligned}
< \tilde{G}(x', y'; x, y) >&= K_2(x', b; x, y) \\
&= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) G(x', b + h_1; x, y)
\end{aligned}
\tag{A.2}
$$

3. $(x, y) \in S_1$ or S_3; $(x', y') \in \tilde{S}_4$

$$
\begin{aligned}
< \tilde{G}(x', y'; x, y) >&= K_3(x', a; x, y) \\
&= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) G(x', a + h_1; x, y)
\end{aligned}
\tag{A.3}
$$

4. $(x,y) \in \tilde{S}_2$; $(x',y') \in S_1$ or S_3

$$< \tilde{G}(x',y';x,y) >= K_4(x',y';x,b)$$
$$= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) G(x',y';x,b+h_1) \qquad (A.4)$$

5. $(x,y) \in \tilde{S}_4$; $(x',y') \in S_1$ or S_3

$$< \tilde{G}(x',y';x,y) >= K_5(x',y';x,a)$$
$$= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) G(x',y';x,a+h_1) \qquad (A.5)$$

6. $(x,y) \in \tilde{S}_2$; $(x',y') \in \tilde{S}_4$

$$< \tilde{G}(x',y';x,y) >= K_6(x',a;x,b) \qquad (A.6)$$
$$= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) G(x',a+h_1;x,b+h_2)$$

7. $(x,y) \in \tilde{S}_4$; $(x',y') \in \tilde{S}_2$

$$< \tilde{G}(x',y';x,y) >= K_7(x',b;x,a) \qquad (A.7)$$
$$= \int_{-\infty}^{+\infty} dh_1 P_1(h_1) \int_{-\infty}^{+\infty} dh_2 P_1(h_2) G(x',b+h_1;x,a+h_2)$$

8. $(x,y) \in \tilde{S}_2$; $(x',y') \in \tilde{S}_2$

$$< \tilde{G}(x',y';x,y) >= K_8(x',b;x,b) \qquad (A.8)$$
$$= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1,h_2;x,x') G(x',b+h_1;x,b+h_2)$$

9. $(x,y) \in \tilde{S}_4$; $(x',y') \in \tilde{S}_4$

$$< \tilde{G}(x',y';x,y) >= K_9(x',a;x,a) \qquad (A.9)$$
$$= \int_{-\infty}^{+\infty} dh_1 \int_{-\infty}^{+\infty} dh_2 P_2(h_1,h_2;x,x') G(x',a+h_1;x,a+h_2)$$

180

Here a and b are defined in (10.1), and functions P_1 and P_2 are defined in (9.1) and (9.2), respectively.

Appendix B

Discretization of integral equation with rough surface

Due to the change of variable in (10.6), (10.7) and (10.8), equation (10.9) is a little un-conventional. The source points are on smooth nominal surfaces and the evaluation points are on rough surfaces. Hence the expansion and the testing basis functions are on different surfaces. The expansion can be written as

$$\tilde{\rho}(x,y) = \sum_{i=1}^{N} \tilde{\rho}_i \tilde{b}_i(x,y,), \tag{B.1}$$

where

$$\tilde{b}_i(x,y,) = \begin{cases} 1 & (x,y) \in \tilde{\Delta}_i \\ 0 & otherwise \end{cases} \tag{B.2}$$

where $\tilde{\Delta}_i$ is the i-th panel on nominal smooth surface. Using the same change of variable defined in (10.6), the testing basis can be written as

$$b_i(x,y,) = \begin{cases} \frac{1}{dl/d\tilde{l}} & (x,y) \in \Delta_i \\ 0 & otherwise \end{cases} \tag{B.3}$$

where Δ_i is the i-th panel on rough surface, and dl is the differential length on rough surface, $d\tilde{l}$ is its projection on the nominal smooth surface. Discretize (10.9) using expansion and

testing functions defined in (B.2) and (B.3), we obtain

$$[A]\tilde{\rho} = d, \tag{B.4}$$

where

$$\begin{aligned} A_{im} &= \int_{\Delta_i} dl(x,y)b_i(x,y) \int_{\tilde{\Delta}_m} d\tilde{l}(x',y')\tilde{b}_m(x',y')\hat{G}(x',y';x,y) \\ &= \int_{\tilde{\Delta}_i} d\tilde{l}(x,y) \int_{\tilde{\Delta}_m} d\tilde{l}(x',y')\tilde{G}(x',y';x,y) \end{aligned} \tag{B.5}$$

and

$$d_i = \int_{\Delta_i} dl(x,y)b_i(x,y) = \int_{\tilde{\Delta}_i} d\tilde{l}(x,y) \tag{B.6}$$

or

$$d = \tilde{L}, \tag{B.7}$$

where $\tilde{L}$ is defined in (10.17).

Appendix C

Kronecker Product

Here we summarize the definition of the Kronecker product and a few identities. For details, please refer to [27, 62].

1. Kronecker product

   If $B \in R^{m \times n}$ and $C \in R^{p \times q}$, then their Kronecker product is given by

   $$A = B \otimes C = \begin{bmatrix} b_{11}C & b_{12}C & \cdots & b_{1n}C \\ b_{21}C & b_{22}C & \cdots & b_{2n}C \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1}C & b_{m2}C & \cdots & b_{mn}C \end{bmatrix} \tag{C.1}$$

2. "vec" operation

   $$X \in R^{m \times n} \iff vec(X) = \begin{bmatrix} X(:,1) \\ \vdots \\ X(:,n) \end{bmatrix} \in R^{mn}. \tag{C.2}$$

3. Identities

   $$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{C.3}$$

   $$(A \otimes B)^T = A^T \otimes B^T \tag{C.4}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{C.5}$$

$$Y = CXB^T \quad \Leftrightarrow \quad vec(Y) = (B \otimes C)vec(X) \tag{C.6}$$

$$vec(xx^T) = x \otimes x \tag{C.7}$$

$$trace(A^T B) = (vec(B))^T vec(A) \tag{C.8}$$

Bibliography

[1] M.I. Aksun. A robust approach for the derivation of closed-form Green's unctions. *IEEE Transactions on Microwave Theory and Techniques*, 44:651–657, May 1996.

[2] M. Bachtold, M. Spasojevic, C. Lage, and P.B. Ljung. A system for full-chip and critical net parasitic extraction for ULSI interconnects using a fast 3D field solver. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(3):325–338, 2000.

[3] J. Bardhan, J.H. Lee, S. Kuo, M. Altman, B. Tidor, and J. White. Fast methods for biomolecule charge optimization. *International Conference on Modeling and Simulation of Microsystems*, April 2002.

[4] P. Beckmann and A. Spizzichino. *The scattering of electromagnetic waves from rough surfaces*. Artech House, Inc., 685 Canton Street, Norwood, MA 02062, 1987.

[5] S. Borm, L. Grasedyck, and W. Hackbusch. Hierarchical matrices. *Lecture note available at www.hmatrix.org/literature.html*, 2003.

[6] A. Brandt and A. A. Lubrecht. Multilevel matrix multiplication and fast solution of integral equations. *Journal of Computational Physics*, 90:348–370, 1990.

[7] G. Brown. A stochastic fourier transform approach to scattering from perfectly conducting randomly rough surfaces. *IEEE Transactions on Antennas and Propagation*, 30(6):1135–1143, November 1982.

[8] G. Brown. Simplifications in the stochastic fourier transform approach to random surface scattering. *IEEE Transactions on Antennas and Propagation*, 33(1):48–55, January 1985.

[9] B. Buchmann and Jacob White. Revisiting the pre-corrected FFT method for fast solution of integral equations. *draft in preparation*, 2000.

[10] Rainer Bunger and Fritz Arndt. Efficient MPIE approach for the analysis of three-dimensional microstrip structures in layered media. *IEEE Transactions on Microwave Theory and Techniques*, 45:1141–1753, August 1997.

[11] David C. Chang and Jian X. Zheng. Electromagnetic modeling of passive circuit elements in MMIC. *IEEE Transactions on Microwave Theory and Techniques*, 40:1741–1747, September 1992.

[12] K. Chen and J.X.Zhou. *Boundary Element Methods*. Academic Press, New York, 1992.

[13] W.C. Chew. *Waves and fields in inhomogeneous media*. IEEE Press, New Jersey, 1995.

[14] S.C. Chi and A.P. Agrawal. Fine line thin dielectric circuit board characterization. *Proceedings of 44th Electronic Components and Technology Conference*, pages 564–569, 1-4 May 1994.

[15] Y. Chu and W.C. Chew. A surface integral equation method for solving complicated electrically small structures. *Proceedings of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, pages 341–344, November 2003.

[16] D. Colton and R. Kress. *Integral Equation Methods in Scattering Theory*. Krieger Publishing Company, Malabar, Florida, 1992.

[17] K. M. Coperich, A. C. Cangellaris, and A. E. Ruehli. Enhanced skin effect for partial-element equivalent-circuit (PEEC) model. *IEEE Transactions on Microwave Theory and Techniques*, 48(9):1435–1442, September 2000.

[18] C.Y.Yang, G. Ouyang, and V. Jandhyala. Integral equation based time domain coupled EM-circuit simulation for packaged conductors and dielectrics. *Proceedings of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, pages 371–374, November 2003.

[19] L. Daniel and J. Phillips. Model order reduction for strictly passive and causal distributed systems. *ACM/IEEE Design Automation Conference*, 2002.

[20] L. Daniel, Alberto Sangiovanni-Vincentelli, and Jacob K. White. Using conduction modes basis functions for efficient electromagnetic analysis of on-chip and off-chip interconnect. *ACM/IEEE Design Automation Conference*, June 18-22, 2001.

[21] Luca Daniel. *Simulation and Modeling Techniques for Signal Integrity and Electromagnetic Interference on High Frequency Electronic Systems*. Ph.D. thesis, EECS Department, University of California at Berkeley, 2003.

[22] P.J. Davis and P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, Inc, Orlando, FL, second edition, 1984.

[23] C. Desoer and E. Kuh. *Basic Circuit Tehory*. McGraw Hill, New York, 1969.

[24] S. Gedney, A.M. Zhu, W.H. Tang, and P. Petre. High-order pre-corrected FFT solution for electromagnetic scattering. *IEEE Antennas and Propagation Society International Symposium*, pages 566–569, 2003.

[25] A.K. Goel. *High-Speed VLSI Interconnections*. John Wiley and Sons, Inc., New York, 1994.

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

[27] G.H. Golub and C.F. Van Loan. *Matrix Computation*. John Hopkins UniversityPress, Baltimore, 1996.

[28] L. Grasedyck. Adaptive recompression of H-matrices for bem. *to be published in Computing*, 2004.

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

[30] L. Greengard and V. Rohklin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, December 1987.

[31] Erric Grimme. *Krylov Projection Methods for Model Reduction*. Ph.D. thesis, University of Illinois at Urbana-Champaign, 1997.

[32] Frederick Warren Grover. *Inductance calculations, working formulas and tables*. Govt. Print. Off., New York, NY, 1946.

[33] W. Hackbusch. A sparse matrix arithmetic based on h-matrices. part I: Introduction to H-matrices. *Computing*, 62:89–108, 1999.

[34] W. Hackbusch. A sparse matrix arithmetic based on H-matrices. part II: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000.

[35] Wolfgang Hackbusch. *Integral Equations, Theory and Numerial Treatment*. Birkhauser Verlag, Basel, Switzerland, 1989.

[36] R. F. Harrington. *Time-harmonic electromagnetic fields*. McGraw-Hill, New York, 1961.

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

[38] R.F. Harrington. Boundary integral formulation for homogeneous material bodies. *Journal of electromagnetic waves and applications*, 3(1):1–15, 1989.

[39] H.A. Haus and J.R. Melcher. *Electromagnetic fields and energy*. Prentice-Hall, Englewood Cliffs, 1989.

[40] Hansruedi Heeb and Albert E. Ruehli. Three-dimensional interconnect analysis using partial element equivalent circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 39(11):974–982, November 1992.

190

[41] J. L. Hess and A. M. O. Smith. Calculation of potential flow about arbitrary bodies. *Progress in Aeronautical Science*, pages 1–138, 1966.

[42] R. W. Hockney and J. W. Eastwood. *Computer simulation using particles*. Adam Hilger, New York, 1988.

[43] C.L. Holloway and E.F. Kuester. Power loss associated with conducting and supperconducting rough surfaces. *IEEE Transactions on Microwave Theory and Techniques*, 48(10):1601–1610, 2000.

[44] J.F. Huang, J. Jia, and J. K. White. A note on integral equation formulations for impedance extraction in three dimensions. *Draft in preparation*, 2004.

[45] A. Ishimaru. *Wave propagation and scattering in random media*. Academic Press, New York, 1978.

[46] N. M. Josuttis. *The C++ Standard Library : A Tutorial and Reference*. Addison-Wesley, 1999.

[47] S. Kalaicheluan and J.D. Lavers. *BEM for eddy current problems*. Vol.6: Topics in BE Research, edited by C.A. Brebbia, London, 1989, pp. 79-116.

[48] M. Kamon, N.A. Marques, L.M. Silveria, and J.K. White. Automatic generation of accurate circuit models of 3D interconnect. *IEEE Transactions on Components, Packaging, and Manufacturing Technology–Part B*, 21(3):225–240, August 1998.

[49] M. Kamon, M. J. Tsuk, and J.K. White. FastHenry: A multipole-accelerated 3-D inductance extraction program. *IEEE Transactions on Microwave Theory and Techniques*, 42(9):1750–1758, September 1994.

[50] Mattan Kamon. *Fast Parasitic Extraction and Simulation of Three-dimensional Interconnect Via Quasistatic Analysis*. Ph.D. thesis MIT EECS Department, 1998.

[51] Mattan Kamon, Frank Wang, and Jacob White. Generating nearly optimally compact models from krylov-subspace based reduced-order models. *IEEE Transactions on*

Circuits and Systems II:Analog and Digital Signal Processing, 47(4):239–248, April 2000.

[52] S. Kapur and D.E. Long. IES3: A fast integral equation solver for efficient 3-dimensional extraction. *International Conference on Computer Aided-Design*, pages 448–455, 1997.

[53] S. Kapur and D.E. Long. Large scale capacitance extraction. *ACM/IEEE Design Automation Conference*, pages 744–748, 2000.

[54] S. Kapur and J. Zhao. A fast method of moments solver for efficient parameter extraction of MCMs. 34th *ACM/IEEE Design Automation Conference*, pages 141–146, 1997.

[55] M. Kleiber and T.D. Hien. *The Stochastic Finite Element Method: basic perturbation techniques and computer implementation*. John Wileys and Sons, Chichester, 1992.

[56] Andrew Koenig and Barbara E. Moo. *Accelerated C++: Practical Programming by Example*. Addison-Wesley, 2000.

[57] B.M. Kolundzija. Electromagnetic modeling of composite metallic and dielectric structures. *IEEE Transactions on Microwave Theory and Techniques*, 47(7):566–569, July 1999.

[58] A. Krawczyk. *Numerial modeling of eddy currents*. Clarendo Press, Oxford, 1993.

[59] S. Kuo, M. Altman, J. Bardhan, B. Tidor, and J. White. Fast methods for simulations of biomolecule electrostatics. *International Conference on Computer Aided-Design*, 2002.

[60] Jing-Rebecca Li, F. Wang, and J. White. Efficient model reduction of interconnect via approximate system grammians. *International Conference on Computer Aided-Design*, pages 380–383, 1999.

[61] Feng Ling, Dan Jiao, and Jian-Ming Jin. Efficient eletromagnetic modeling of microstrip structures in multilayer media. *IEEE Transactions on Microwave Theory and Techniques*, 47:1810–1818, September 1999.

[62] C.F. Van Loan. The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics*, 123:85–100, 2000.

[63] Yehia M. Massoud. *Simulation algorithms for inductive effects*. Ph.D. thesis MIT EECS Department, Cambridge, MA, 1999.

[64] J.R. Mautz and R.F. Harrington. An E-field solution for a conducting surface small or comparable to the wavelength. *IEEE Transactions on Antennas and Propagation*, 32:330–339, April 1984.

[65] M.Bebendorf and S.Rjasanow. Adaptive low-rank approximation of collocation matrices. *Computing*, 70:1–24, 2003.

[66] L.N. Medgyesi-Mitschange, J.M. Putname, and M.B. Gendre. Generalized method of moments for three-dimensional penetrable scatters. *Journal of Optical Society of America*, 11(4):1383–1398, April 1984.

[67] K.A. Michalski and Juan.R. Mosig. Multilayered media green's functions in integral equation formulations. *IEEE Transactions on Antennas and Propagation*, 45:508–519, March 1997.

[68] M.J.Tusk and J.A.Kong. A hybrid method for the calculation of the resistance and inductance of transmission lines with arbitrary cross sections. *IEEE Transactions on Microwave Theory and Techniques*, 39(8):1338–1347, August 1991.

[69] S.P. Morgan. Effect of surface roughness on eddy current losses at microwave frequencies. *Journal of Applied Physics*, 20:352–362, 1949.

[70] Juan R. mosig. Arbitrarily shaped microstrip structures and their analysis with mixed potential integral equation. *IEEE Transactions on Microwave Theory and Techniques*, 36:314–323, February 1988.

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

[72] T. Nakata, N. Takahashi, K. Fujifwara, K. Muramatsa, and Z.G. Cheng. Comparison of various methods for 3-D eddy current analysis. *IEEE Transactions on Magnetics*, 24:3159–3161, November 1988.

[73] Altan Odabasioglu, Mustafa Celik, and Lawrence Pileggi. Prima:passive reduced-order interconnect macromodeling algorithm. *International Conference on Computer Aided-Design*, pages 58–65, 1997.

[74] A. Papoulis. *Probability, random variables, and stochastic processes*. McGraw-Hill Inc., New York, 1965.

[75] R.M. Patrikar, C.Y. Dong, and W.J. Zhuang. Modelling interconnects with surface roughness. *MICROELECTRONICS JOURNAL*, 33(11):929–934, 2002.

[76] C.R. Paul. *Introduction to Electromagnetic Compatibility*. John Wiley and Sons, Inc., New York, 1992.

[77] A.F. Peterson. The interior resonance problem associated with surface integral equations of electromagnetics: Numerical consequences and a survey of remedies. *Electromagnetics*, pages 293–312, October 1990.

[78] Joel R. Phillips. *Rapid solution of potential integral equations in complicated 3-dimensional geometries*. Ph.D. thesis MIT EECS Department, 1997.

[79] Joel R. Phillips, Eli Chiprout, and David D. Ling. Efficient full-wave electromagnetic analysis via model-order reduction of fast integral transformations. *ACM/IEEE Design Automation Conference*, 1995.

[80] Joel R. Phillips and J. K. White. A precorrected-FFT method for electrostatic analysis of complicated 3D structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1059–1072, 1997.

[81] S. Ramo, J.R. Whinnery, and T.V. Duzer. *Fields and waves in communication electronics*. John Willey and sons, Inc., New York, 1994.

[82] S.M. Rao and D.R. Wilton. E-field, H-field, and combind field solution for arbitrarily shaped three dimensional dielectric bodies. *Electromagnetics*, (10):407–421, 1990.

[83] S.M. Rao, D.R. Wilton, and A.W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Transactions on Antennas and Propagation*, 30:409–418, May 1982.

[84] S.M. Rao, D.R. Wilton, and A.W. Glisson. Electromagnetic scattering by surfaces of arbitrary shapes. *IEEE Transactions on Antennas and Propagation*, 30:409–418, May 1982.

[85] A. Rong, A.C. Cangellaris, and L. Dong. Comprehensive broadband electromagnetic modeling of on chip interconnects with a surface discretization-based generalized peec model. *Proceedings of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, pages 367–370, November 2003.

[86] Wolfgang M. Rucker, Robert Hoschek, and Kurt R. Richter. Various BEM formulations for calculating eddy current in terms of field variables. *IEEE Trans. on Magnetics*, 31:1336–1341, May 1995.

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

[88] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing, Boston, MA, 1996.

[89] Youcef Saad and Martin Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, July 1986.

[90] T.K. Sarkar, E. Arvas, and S.M. Rao. Application of FFT and the conjugate gradient method for the solution of electromagnetic radiation from electrically large and small

conducting bodies. *IEEE Transactions on Antennas and Propagation*, 34:635–640, 1986.

[91] H.V. Smith. *Numerical Methods of Integration*. Chartwell-Bratt Ltd, Studentlitteratur, Lund, Sweden, 1993.

[92] Ben Song, Zhenhai Zhu, John Rockway, and J. K. White. A new surface integral formulation for wideband impedance extraction of 3-d structures. *International Conference on Computer Aided-Design*, 2003.

[93] J.M. Song, C.C. Liu, W.C. Chew, and S.W. Lee. Fast illinois solver code (FISC) solves problems of unprecedented size at center for computational electromagnetics, university of illinois. *IEEE Antennas and Propagation Magzine*, 40:27–34, June 1998.

[94] J.A. Stratton. *Electromagnetic Theory*. McGraw-Hill Book Company, New York, 1941.

[95] Bjarne Stroustrup. *The C++ Programming Language Special Edition*. Addison-Wesley, 1997.

[96] C.T. Tai. *Dyadic green's functions in electromagnetic theory*. IEEE Press, Piscataway, New Jersey, 1994.

[97] H. Tanaka and F. Okada. Precise measurements of dissipation factor in microwave printed circuit boards. *IEEE Transactions on Instruments and Measurement*, 38(2):509–514, 1989.

[98] L.N. Trefethen and D. Bau. *Numerical linear algebra*. SIAM, Philadelphia, 1997.

[99] L. Tsang, J.A. Kong, K.H. Ding, and C.O. Ao. *Scattering of electromagnetic waves: numerical simulations*. John Wiley and Sons, Inc., New York, 2001.

[100] E. Tuncer, B.T. Lee, and D.P. Neikirk. Interconnect series impedance determination using a surface ribbon method. *Proceedings of IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, pages 249–252, November 1994.

[101] B.J. Uscinski and C.J. Stanek. Acoustic scattering from a rough surface: the mean field by the integral equation method. *Waves in Random Media*, 12(2):247–263, April 2002.

[102] G. Vecchi. Loop-star decomposition of basis functions in the discretization of the efie. *IEEE Transactions on Antennas and Propagation*, 47:339–346, February 1999.

[103] J. Wang and J. K. White. A wide frequency range surface integral formualtion for 3D RLC extraction. *International Conference on Computer Aided-Design*, 1999.

[104] J.J.H. wang. *Generalized moment methods in electromagnetics*. John Willey and sons, Inc., New York, 1991.

[105] Junfeng Wang. *A new surface integral formulation of EMQS impedance extraction for 3-D structures*. Ph.D. thesis MIT EECS Department, 1999.

[106] X. Wang, P. Mucha, and J.K. White. Fast fluid analysis for multibody microma-chined devices. *Proceedings of MSM*, pages 19–22, Hilton Head island, SC, 2001.

[107] K.F. Warnick and W.C. Chew. Numerical simulation methods for rough surface scattering. *Waves in Random Media*, 11:1–30, 2001.

[108] David J. Willis. *A pFFT Accelerated High Order Panel Method*. Master thesis MIT Aeronautics and Astronautics Department, 2003.

[109] D.R. Wilton, S.M. Rao, A.W. Glisson, D.H. Schaubert, O.M. Al-bundak, and C.M.Butler. Potential integrals for uniform and linear source distributions on polyg-onal and polyhedral domains. *IEEE Transactions on Antennas and Propagation*, 32(3):276–281, March 1984.

[110] T.K. Wu and L.L. Tsai. Scattering from arbitrary-shaped lossy dielectric bodies of revolution. *Radio Science*, 12(5):709–718, 1977.

[111] Wen-Liang Wu, Allen W. Glisson, and Darko Kajfez. A study of two numerial solution precedures for the electric field integral equation at low frequency. *Applied computational electromagnetics*, 10:69–80, November 1995.

[112] L. Young. *Advances in Microwaves*. Academic Press, New York, 1971.

[113] J.S. Zhao and W.C. Chew. Integral equation solution of maxwell equations from zero frequency to microwave frequencies. *IEEE Transactions on Antennas and Propagation*, 48:1635–1645, October 2000.

[114] Zhenhai Zhu, Jingfang Huang, Ben Song, and J. K. White. Improving the robustness of a surface integral formulation for wideband impendance extraction of 3D structures. *International Conference on Computer Aided-Design*, pages 592–597, 2001.

[115] Zhenhai Zhu, Ben Song, and J. K. White. Algorithms in FastImp: A fast and wideband impedance extraction program for complicated 3D geometries. *ACM/IEEE Design Automation Conference*, 2003.