Field Solver Technologies for Variation-Aware Interconnect Parasitic Extraction

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
FEB 23 2010
LIBRARIES

by

Tarek Ali El-Moselhy

M.S., Cairo University (2005)

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

Doctor of Philosophy

ARCHIVES

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2010

© Massachusetts Institute of Technology 2010. All rights reserved.

Author ....
Department of Electrical Engineering and Computer Science
January 28, 2009

Certified by .... [illegible] ....
Luca Daniel
Associate Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Terry P. Orlando
Chairman, Department Committee on Graduate Theses

Field Solver Technologies for Variation-Aware Interconnect Parasitic Extraction

by

Tarek Ali El-Moselhy

Submitted to the Department of Electrical Engineering and Computer Science
on January 28, 2009, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Advances in integrated circuit manufacturing technologies have enabled high density on-chip integration by constantly scaling down the device and interconnect feature size. As a consequence of the ongoing technology scaling (from 45nm to 32nm, 22nm and beyond), geometrical variabilities induced by the uncertainties in the manufacturing processes are becoming more significant. Indeed, the dimensions and shapes of the manufactured devices and interconnect structures may vary by up to 40% from their design intent. The effect of such variabilities on the electrical characteristics of both devices and interconnects must be accurately evaluated and accounted for during the design phase. In the last few years, there have been several attempts to develop variation-aware extraction algorithms, i.e. algorithms that evaluate the effect of geometrical variabilities on the electrical characteristics of devices and interconnects. However, most algorithms remain computationally very expensive. In this thesis the focus is on variation-aware interconnect parasitic extraction.

In the first part of the thesis several discretization-based variation-aware solver techniques are developed. The first technique is a stochastic model reduction algorithm (SMOR). The SMOR guarantees that the statistical moments computed from the reduced model are the same as those of the full model. The SMOR works best for problems in which the desired electrical property is contained in an easily defined subspace. The second technique is the combined Neumann Hermite expansion (CNHE). The CNHE combines the advantages of both the standard Neumann expansion and the standard stochastic Galerkin method to produce a very efficient extraction algorithm. The CNHE works best in problems for which the desired electrical property (e.g. impedance) is accurately expanded in terms of a low order multivariate Hermite expansion. The third technique is the stochastic dominant singular vectors method (SDSV). The SDSV uses stochastic optimization in order to sequentially determine an optimal reduced subspace, in which the solution can be accurately represented. The SDSV works best for large dimensional problems, since its complexity is almost independent of the size of the parameter space.

In the second part of the thesis, several novel discretization-free variation aware extraction techniques for both resistance and capacitance extraction are developed. First we present a variation-aware floating random walk (FRW) to extract the capacitance/resistance in the presence of non-topological (edge-defined) variations. The complexity of such algorithm is almost independent of the number of varying parameters. Then we introduce the Hierarchical FRW to extract the capacitance/resistance of a very large number of topologically different structures, which are all constructed from the same set of building blocks. The complexity of such algorithm is almost independent of the total number of structures.

All the proposed techniques are applied to a variety of examples, showing orders of magnitude reduction in the computational time compared to the standard approaches. In addition, we solve very large dimensional examples that are intractable when using standard approaches.

Thesis Supervisor: Luca Daniel
Title: Associate Professor

Acknowledgments

I foremost want to thank my advisor, Prof. Luca Daniel, for his constant support starting from our very first meeting. His impressive attention to detail and his ability to connect seemingly unrelated concepts to form a complete homogeneous picture, enlightened every discussion I had with him. This work could have not been possible without him. I deeply appreciate the days we spent on writing, preparing presentations and just talking. Thank you Luca very much you made my PhD experience very pleasant.

I have also been very lucky to have had the chance to interact with Prof. Jacob White. I still remember back in Egypt when I was reading his papers and just hoping to meet him some day. I appreciate every discussion I had with him. Anytime I went into his office, I knew that I will be learning something new or at least understand something I know in a totally different light. Interacting with Prof. Jacob always reminds me that I still have a long road to go, but that such road will be a very exciting one. Thank you!

I want to thank Prof. Duane Boning for agreeing to be on my thesis committee and for patiently reading my thesis. I very much appreciate your valuable suggestions related to the thesis organizations. If it wasn't for Prof. Boning I would have faced a mess trying to combine my work on both extraction and cardiovascular simulations in a single thesis.

I want to thank Dr. Abe Elfadel from IBM Watson Research for inviting me as a summer intern for two consecutive summers and for mentoring me through the IBM fellowship program. I learned from Abe the industry view-point on electrical parasitic extraction. Abe also brought to my attention the floating random walk algorithm, which resulted in the second part of this thesis. Thank you Abe.

I want to thank Dr. Khaled Ismail, Prof. Yehia Massoud, Dr. Albert Ruehli, Dr. David White, and Prof. Hani Ghali for their support.

I also want to thank my lab mate Brandon Bond. I value the time we spent talking either related to research or simply about general topics (including forbidden topics such as politics and religion). I want to thank Dr. Junghoon Lee for teaching me a lot about the pre-corrected Fast Fourier Transform and for many discussions related to the immersed boundary methods. I want to thank Dr. Kin Sou for patiently explaining a lot of optimization related concepts. I want to thank Dr. Xin Hu for guiding me through the first days of my MIT experience and for sharing her code with me. I want to thank Bo Kim, Lei Zhang, Zohaib Mahmoud, Yu Chung, Dr. Amit Hochman, and Homer Reid for being such nice colleagues and for many interesting discussions. In particular, I want to thank Bo for helping me with the TQE and for all the chocolate she shared with me and my kids.

Having Egyptian friends here at MIT and in the Boston area made the experience even more exciting. A special thank you to Dr. Belal Helal, Dr. Aliaa Barakat, Sherif Akl, Dr. Ali Shoeb, Mohamed Raafat, Tamer Elkholy, Mustafa Youssef, Dr. Lamia Youseff, Dina El-Zanfaly, Ahmed Kouchouk, and Dr. Mohamed Emara.

I also want to thank my best friend since childhood Mohamed Elsayed. Not only did he take the time to visit me four times during my MIT experience (including one time to specially attend my defense) but he also was there for my parents during my time away.

I want to thank my grandmother, who has been an inspiration since my early childhood years. Her encouraging words and weekly phone calls made this experience even nicer. I also want to thank my uncle Prof. Osama Moselhi. I had so many interesting and informative discussions about life with him. Thank you very much.

Last but not least, I want to thank my family. I will start with my beloved wife Rokaya. This work could not have been possible without you. You kept up with all the days of hard work and absent mind without once complaining. Your encouraging words and positive attitude have provided me with the energy and the strength to continue that work. I must thank my wife for her unbelievable effort to serve at times as both the mom and the dad. I want to thank my three kids, Ali, Kamila and Laila for bringing joy and happiness to my life. I want to thank my brother Amr and his wife Mariam for all the support they gave my family. The whole Boston experience would not have been the same without them. I also value the endless hours I spent with my brother discussing everything from technical issues to politics. Amr also helped me with my thesis defense presentation and with the final editing of the thesis.

That said, my deepest thanks goes to my parents. I want to thank my mother and father for their unbelievable support, both emotional and financial. My father has been there for me every step of the way, with his encouraging words, smart ideas and push for success.

My mother believed in me from a very young age, encouraged me and taught me to never give up. She taught me that even in the darkest of hours, you just do your best and believe that the sun will shine tomorrow.

My acknowledgment is not complete without showing my deepest gratitude and love towards GOD. At every step of my life I was overwhelmed with his blessings and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Since the early stages of integrated circuit technologies in the 1960s, the density of devices on the chip has been following the famous Moore's law [58]. Nonetheless, for the past few years people have been questioning the ability of integrated circuit manufacturing technologies to maintain the current rate of scaling and to progress into future technology nodes [75]. Among the variety of reasons justifying such concerns, is the fact that as the technology scales down, the uncertainty in the manufactured shapes increases. Indeed, despite the use of advanced design for manufacturability (DfM) techniques [47], the resulting shapes on wafer are still different from those intended by the designer (see Figure 1-1). Such geometrical variations include phenomena such as shape variations (e.g. corner rounding) due to the lithography process, in addition to line edge roughness and width variations due to etching, as well as thickness variations due to chemical mechanical polishing (CMP). In addition to the before-mentioned on-chip variations, off-chip interconnect structures suffer from surface roughness variations resulting for instance from the standard square-pulsed electroplating processes.

For the most part, evaluating the effect of shape variations on the electrical characteristics of the manufactured structures has relied on geometrical measures. In other words, the electrical characteristics of the manufactured structures are evaluated based on the proximity of the manufactured shapes to the drawn ones. Recently, there is emerging recognition

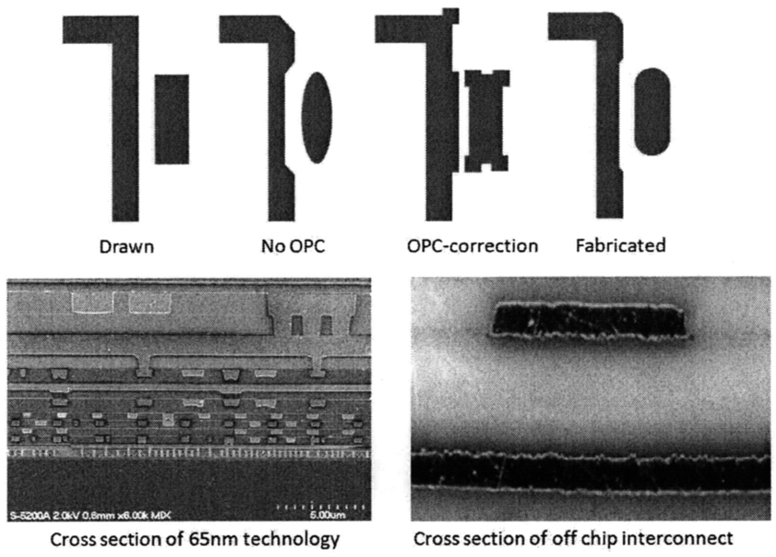

Drawn
No OPC
OPC-correction
Fabricated
S-5200A 2.0kV 0.6mm x6.00k MDX
6.00um
Cross section of 65nm technology
Cross section of off chip interconnect

Figure 1-1: Top figure: Top view illustrating that fabricated structures are different than the drawn ones despite the use of opc [69]. Lower left figure: cross section in 65nm technology demonstrating significant width and thickness variations. Lower right figure: cross section in an off-chip interconnect structure showing significant surface roughness [9]

among process engineers and designers that such geometrical measures are not sufficient. In fact, it is widely accepted [3] that manufactured shapes on wafer *will* be different from those drawn by designers in the layout (notice for instance the presence of highly irregular contours in the SRAM cell fabricated in a 45nm technology despite the use of lithographic improvement techniques, such as optical pre-correction and resolution enhancement Figure 1-2). Consequently, it is very important to evaluate the effect of such shape variations on the electrical characteristics of the manufactured devices and interconnects [5]. In this thesis the focus is primarily on developing variation-aware extraction tools, that can efficiently predict the effect of geometrical variations on the electrical properties (resistance, capacitance, inductance or full impedance) of interconnect structures (see Figure 1-3).

While variations induced by manufacturing process variabilities are very important, they are by no means the only type of variations that we care to account for. Indeed, variations can also be induced by the designer in order to optimize the design. For instance, designers typically try different layouts (or arrangements of the design components) in order to achieve certain design objectives. Such design trials are considered topologically induced variations. Similarly, designers typically try different dimensions or relative dis-

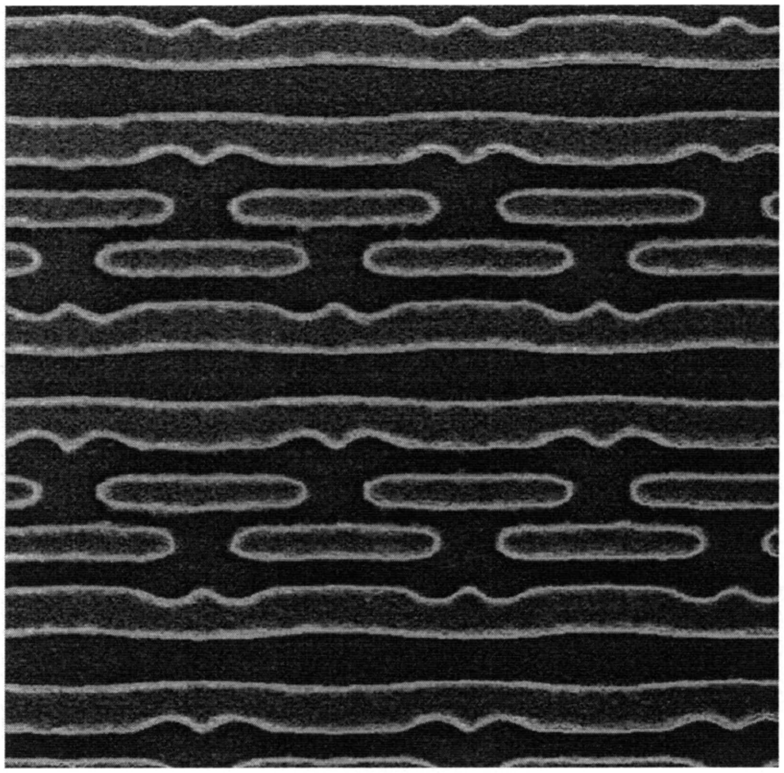


Figure 1-2: A photograph of the top view of wafer shapes of the active areas of an SRAM cell in a 45nm technology

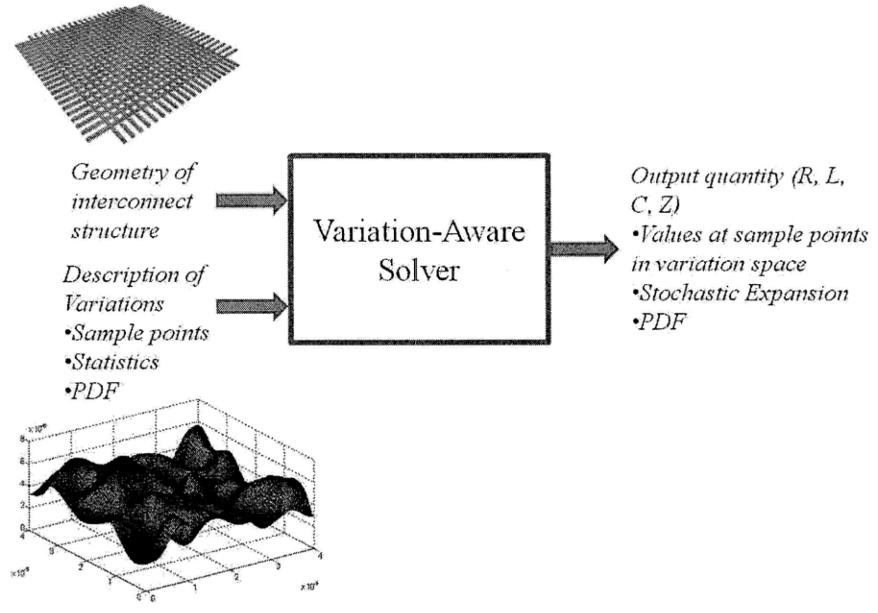

Geometry of interconnect structure
Description of Variations
•Sample points
•Statistics
•PDF
Variation-Aware Solver
Output quantity (R, L, C, Z)
•Values at sample points in variation space
•Stochastic Expansion
•PDF

Figure 1-3: General structure of a variation-aware solver

tances between components in order to choose the optimal and most robust design dimensions. Such trials are considered deterministic non-topological variations. Other applications (different than design optimization) in which designer induced variations are evident include the pre-characterization of capacitance tables [46, 22] and the generation of parameterized reduced order models [64].

1.2 Maxwell's Equations

Computing the electrical characteristics (such as resistance, capacitance or impedance) of interconnect structures relies on solving Maxwell's equations:

$$\begin{aligned}
\nabla \times \vec{E}(\mathbf{r},t) &= -\mu \frac{d\vec{H}(\mathbf{r},t)}{dt} && (1.1)\\
\nabla \times \vec{H}(\mathbf{r},t) &= \varepsilon \frac{d\vec{E}(\mathbf{r},t)}{dt} + \vec{J}(\mathbf{r},t) && (1.2)\\
\nabla \cdot \varepsilon \vec{E}(\mathbf{r},t) &= \rho_V(\mathbf{r},t) && (1.3)\\
\nabla \cdot \mu \vec{H}(\mathbf{r},t) &= 0 && (1.4)
\end{aligned}$$

where $\vec{E}(\mathbf{r},t)$, $\vec{H}(\mathbf{r},t)$ are the electric and magnetic fields respectively, μ is the permeability, ε is the permittivity, $\rho_V$is the volumetric charge density and $\vec{J}(\mathbf{r},t)$ is the current density vector.

In the rest of this thesis we assume all material properties are linear. Hence the resulting system of equations is linear and without loss of generality, it can be fully analyzed by just considering time harmonic excitations $\exp(-j\omega t)$, where ω is the angular frequency:

$$\nabla \times \vec{E}(\mathbf{r}) = j\omega\mu\vec{H}(\mathbf{r}) \tag{1.5}$$

$$\nabla \times \vec{H}(\mathbf{r}) = -j\omega\varepsilon\vec{E}(\mathbf{r}) + \vec{J}(\mathbf{r}) \tag{1.6}$$

$$\nabla \cdot \varepsilon\vec{E}(\mathbf{r}) = \rho_V(\mathbf{r}) \tag{1.7}$$

$$\nabla \cdot \mu\vec{H}(\mathbf{r}) = 0 \tag{1.8}$$

Finally, we will use the constitutive equation relating the current density to the electric field in conducting media

$$\vec{E}(\mathbf{r}) = \sigma\vec{J}(\mathbf{r}) \tag{1.9}$$

where σ is the conductivity of the media.

1.3 Standard Field Solver Classification

Field solvers are computational tools, which are used to numerically solve partial differential equations such as the Maxwell's equations (1.5-1.8). Typically the input to such solvers is the description of both the geometry and material properties. The final output is the desired electrical characteristic of the structure, such as the capacitance, resistance or even full impedance. Standard field solvers can be classified into discretization-based and discretization-free solvers. Moreover, discretization-based solvers can be classified into both differential and integral equation based solvers (Figure 1-4).

Discretization-based differential-equation methods are used to solve formulations of the Maxwell's equations, which involve the differential local operators. Such methods generally produce sparse linear systems of equations. Example of such methods, include both

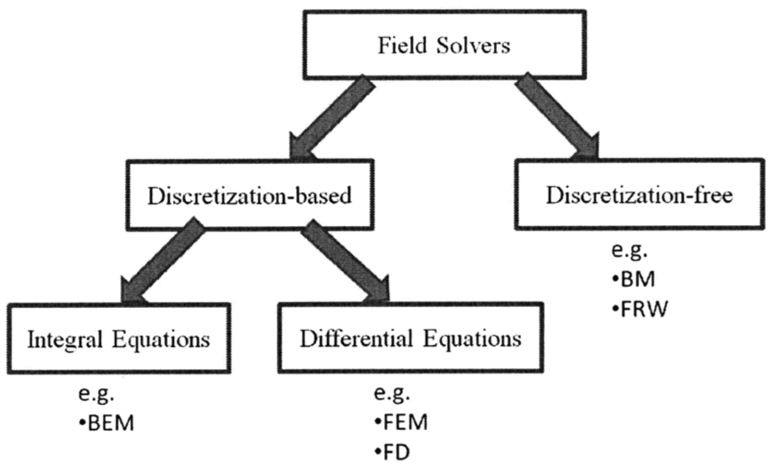


Figure 1-4: Typical classification of field solvers based on their solution technique

the finite difference (FD) and finite element methods (FEM). They are particularly suitable if the domain of computation is closed and includes complex or even inhomogeneous material configuration, e.g. capacitance table generation [22] or resistance extraction [21]. The FEM has also been applied to more complex extraction applications [84]. A variety of commercial tools have been developed based on the FEM, e.g. HFSS by Ansoft [2].

Discretization-based integral-equation methods are used to solve formulations of the Maxwell's equations, which involve the Green's function based global operators. Such methods generally produce dense linear systems of equations. Example of such methods include the boundary element method (BEM) and the partial element equivalent circuit method (PEEC). They have gained particular importance during the last two decades due to the development of the "fast" matrix vector product algorithms, such as multipole expansion [59], pre-corrected FFT [63], SVD-compression [44], H-matrix [30] and hierarchical techniques [81]. Such algorithms facilitate solving the linear systems of equations in almost linear complexity. Consequently, such methods have rendered integral equation based algorithms extremely efficient for electrical extraction applications. Examples of publicly available BEM-based tools are FastCap [59], FastHenry [43] and FastMaxwell [23]. Examples of commercially available BEM-based tools include EMX by Integrand [35].

Discretization-free solvers rely on using random walk methods in order to solve the partial differential equations. Such methods typically do not rely on assembling any linear systems of equations. There are a variety of algorithms falling under the discretization-free category such as the floating random walk (FRW) [31], walk on boundary [67] and the Brownian Motion (BM) methods [52]. In particular, the FRW has been successfully applied to both capacitance and resistance extraction [13]. The FRW is particularly useful for very large size applications, in which the main objective is to compute the self capacitance of some particular wire (net). Examples of commercially available codes based on the FRW are Magma's QuickCap [51].

1.4 Variation-Aware Field Solver Classification

State of the art variation-aware solvers (recall Figure 1-3) can also be divided into discretization based and discretization free solvers (see Figure 1-5). Variation-aware discretization-based solvers can be divided into two different categories [33], namely, non-intrusive and intrusive. "Non-intrusive" algorithms rely on computing the solution of the problem at different points in the parameter space *by using any standard deterministic field solver.* The solution anywhere in the space can then be obtained by interpolating between the computed solutions at the different sample points. Examples of non-intrusive algorithms include the well-known Monte Carlo method (MC) [79], and the stochastic collocation method (SCM) [77]. "Intrusive" solvers rely on expanding the unknown in terms of some basis polynomials. The unknown coefficient vectors of the expansion are then computed using specialized solvers. Examples of intrusive algorithms include the Neumann expansion method [85, 18], and the stochastic Galerkin method (SGM) [28]. Notice that the SGM is also known as the stochastic finite element method (SFE). The first part of this thesis is primarily concerned with both improving existing discretization-based variation-aware field solvers and developing new ones.

On the other hand, we are not aware of any variation-aware discretization-free solvers. In the second part of this thesis we will introduce very efficient discretization-based variation-aware solvers.

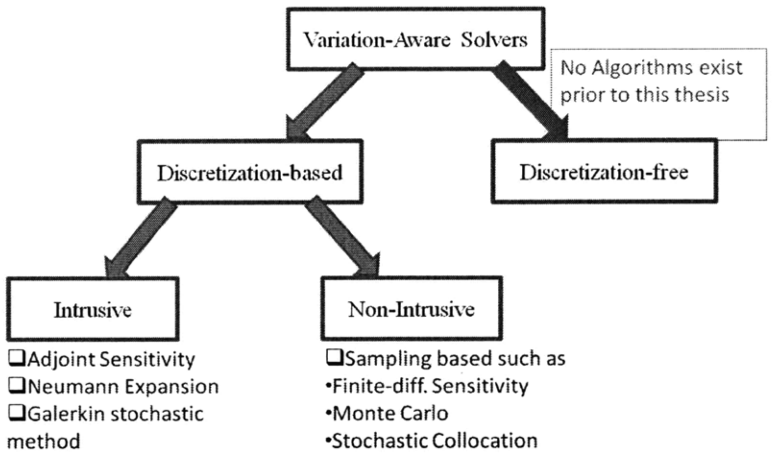


Figure 1-5: Typical classification of variation-aware field solvers

1.5 Variability Classifications

There are a variety of ways to classify the different types of variations (Figure 1-3). In this thesis we will divide the variations based on their geometrical characteristics into topological and non-topological variations (see Figure 1-6 and Figure 1-7). Non-topological variations are those that are defined by changing just the dimensions or the material properties of a given nominal structure. such structure is generally the one intended by the designer. Such variations can either be stochastic (such as manufacturing process induced variations) or deterministic (such as designer induced variations)[1]. On the other hand, topological variations are those produced by rearranging the sub-modules of a structure in order to produce a different one. Such variations are typically deterministic and are commonly induced by the designers during the early design exploration phase, in order to study the impact of topology on electrical characteristics of the structure.

1.6 Variability-Solver Pairing

In Figure 1-8 we summarize the list of solvers, which are most suitable for handling each variability class. Notice that all entries with a check-mark (in black) are currently available in the literature, whereas those without a check-mark (in red) are developed throughout this

[1]Notice that *deterministic process variations* are ofter times referred to as systematic variations [8]

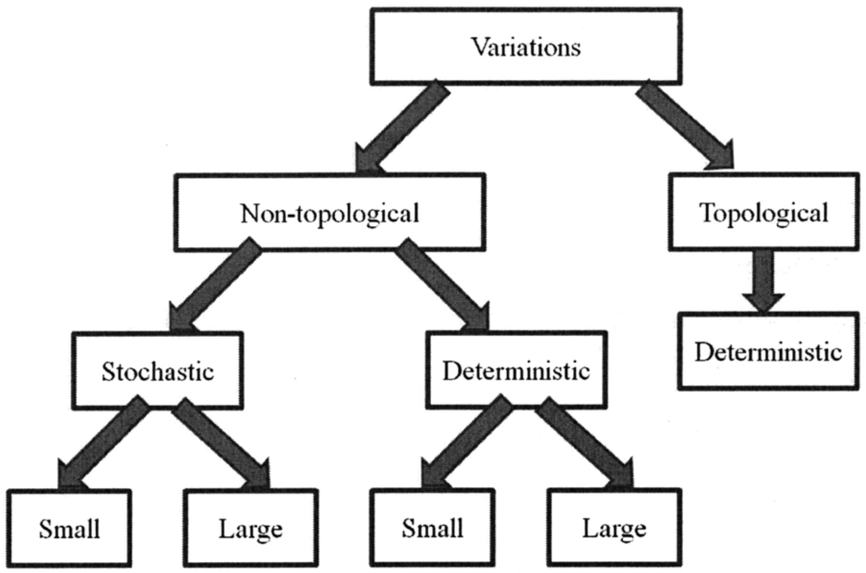


Figure 1-6: Typical classification of manufacturing and design variabilities

thesis.

With the exception of Chapter 9, in this thesis we will be primarily developing algorithms to handle non-topological variations. Furthermore, we will focus primarily on stochastic non-topological variations. That is mainly because solvers which can handle stochastic variations can equally (up to a change of norm, not change in algorithm) handle deterministic variations (see Figure 1-8).

1.7 Outline

This thesis is divided into three parts. The first part is primarily dedicated to discretization-based variation-aware solver algorithms. The general objective is to solve a linear system of equations in which the system matrix elements are nonlinear functions of a large number of random parameters. Part I starts with Chapter 2 which includes the necessary background and previous work on discretization-based variation-aware extraction algorithms. In Chapter 3 we present our contributions to the existing non-intrusive methods. In particular, we present a Krylov subspace recycling algorithm and a stochastic model order reduction algorithm (SMOR). In Chapter 4 we present our contributions to the existing intrusive methods. In particular we present a Combined Neumann Hermite Expansion method (CNHE) and a

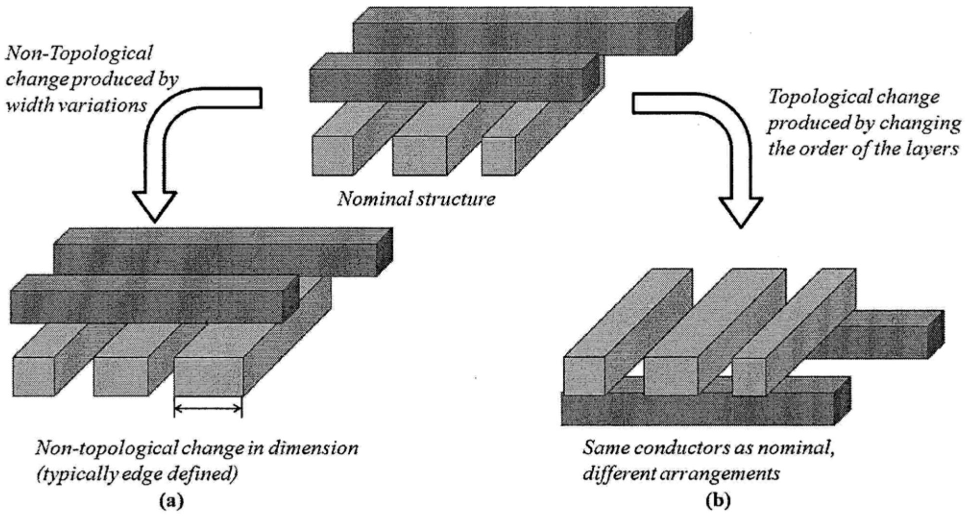

Non-Topological change produced by width variations
Nominal structure
Topological change produced by changing the order of the layers
Non-topological change in dimension (typically edge defined)
(a)
Same conductors as nominal, different arrangements
(b)

Figure 1-7: Example of non-topological and topological variations

Fast Stochastic Galerkin method (FSGM). In Chapter 5 we present a new intrusive algorithm which we called the stochastic dominant singular vectors method (SDSV).

The second part of this thesis is dedicated to discretization-free methods. Chapter 6 summarizes the standard floating random walk method. Chapter 7 summarizes our generalized floating random walk algorithm, which extends the standard FRW to efficiently treat, in a unified framework, cases which include multiple dielectric layers, floating potential metal fills and mixed Dirichlet-Neumann boundary conditions. In Chapter 8 we present our variation-aware FRW algorithm to efficiently extract the capacitance of a very large number of topologically similar structures in a complexity that is practically independent of the number of configurations. Finally in Chapter 9, we present our hierarchical FRW to extracts the capacitance of topologically different structures, which are all composed of the same set of submodules (motifs).

The third part of this thesis starts with a Chapter 10 in which we compare the performance of the main algorithms presented in this thesis on a variety of medium size and large size examples. We also make recommendations on how to choose a particular algorithm for a given application. We then follow with the conclusion and the future work in Chapters 11 and 12, respectively.

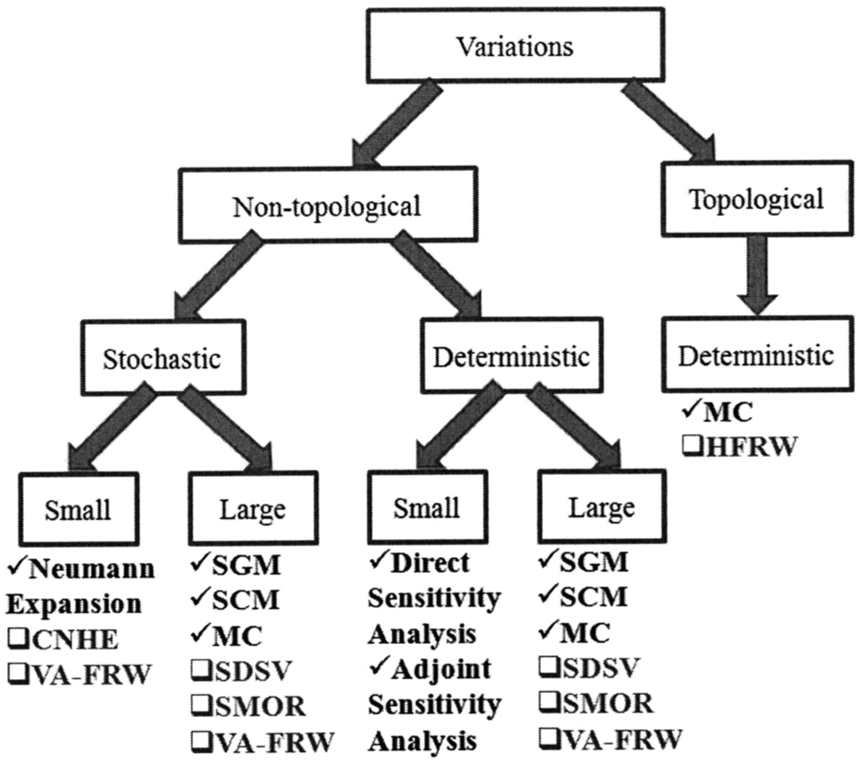


Figure 1-8: Example of appropriate solver corresponding to each variability type. Solvers with a check-mark are currently available in the literature. Solvers without the check-mark are developed throughout the rest of this thesis.

1.8 Contributions

The following is a list summarizing the contributions presented in this thesis:

1. In Chapter 3 we develop the statistical-moment-preserving stochastic model order reduction method (SMOR). The SMOR employs projection-based model reduction techniques in order to reduce the complexity of solving the stochastic linear system. In the SMOR the projection matrix is carefully chosen such that the statistical moments computed from the reduced model match exactly those computed from the full model. Our method bridges the gap between both the stochastic simulation and the parameterized model reduction communities [26].

2. In Section 4.1 we develop a modified inner product in order to efficiently compute the coefficients of the projection of the system matrix elements on the space of multivariate Hermite polynomials. The theorem exploits the sparse dependence of the

matrix elements on the parameter space to reduce the time required to compute the Hermite expansion by up to 10 orders of magnitude. This theorem facilitates applying any algorithm, which is based on the polynomial chaos expansion, to electrical extraction applications [19, 16].

3. In Section 4.2 we develop the combined Neumann Hermite expansion (CNHE) for variation-aware extraction in the presence of a large number of correlated random variables. The CNHE method combines the best features of both the standard Neumann expansion and the standard polynomial chaos expansion techniques, while avoiding their computational complexities. The CNHE also facilitates using the standard non-intrusive "fast" solvers and can therefore handle very large structures in very large dimensional parameter spaces [19, 16].

4. In Chapter 5 we develop the stochastic dominant singular vectors method (SDSV), which is a novel intrusive simulation technique based on finding the stochastic dominant singular vectors in the solution space. We also proposed a new method for compressing the coefficient matrices of the system matrix Hermite expansion. Furthermore, we presented three different relaxed variants of the algorithm, in which, instead of finding the dominant singular vectors, we find "almost" dominant singular vectors. Our algorithm has an unprecedented low complexity, which is almost equivalent to solving just the standard non-stochastic system [17].

5. In Chapter 7 we develop the generalized floating random walk (GFRW) algorithm to efficiently extract the capacitance of complex structures in the presence of multilayered dielectric media, floating metal fill and mixed Dirichlet-Neumann boundary conditions [24, 20].

6. In Chapter 8 we develop a path-recycling-based variation-aware floating random (VAFRW) algorithm. To the best of our knowledge, this is the most efficient algorithm to extract the capacitance in the presence of non-topological variations. The complexity of the VAFRW is for all practical purposes totally independent of the size of the parameter space [24, 20].

7. In Chapter 9 we develop the hierarchical floating random (FRW) algorithm, which is the first hierarchical algorithm for capacitance extraction that resolves the global interactions without the need to assemble/solve a large linear systems of equations. Instead our H-FRW resolves the global interactions using Monte Carlo simulations of Markov Chains. The hierarchical FRW is a very efficient algorithm for topological-variation-aware capacitance extraction [25].

Part I

Discretization-based Variation-Aware Solvers

Chapter 2

State of the Art in Solvers and Variation-Aware Extraction

2.1 Discretization-Based Field Solvers for Impedance Extraction

As mentioned in Chapter 1, there are a variety of field solver formulations. In this thesis we will use, as an example, the mixed potential integral equation (MPIE) formulation [65, 41, 42, 34]. The main reason for choosing the MPIE is its generality and its efficiency in extracting different electrical parameters, such as resistance, capacitance, inductance and impedance. In a general impedance extraction setting the MPIE describes the relation between the volumetric current density $\vec{J}(\mathrm{r})$ inside the conductor, the surface charge distribution $\rho(\mathrm{r})$, and the electric potential $\phi(\mathrm{r})$:

$$\begin{aligned}
\frac{\vec{J}(\mathrm{r})}{\sigma_c} + j\omega\frac{\mu}{4\pi}\int_V G(\mathrm{r},\mathrm{r}')\vec{J}(\mathrm{r}')d\mathrm{r}' &= -\nabla\phi(\mathrm{r}) && (2.1)\\
\frac{1}{4\pi\epsilon}\int_S G(\mathrm{r},\mathrm{r}')\rho(\mathrm{r}')d\mathrm{r}' &= \phi(\mathrm{r}) && (2.2)\\
\nabla\cdot\vec{J}(\mathrm{r}) &= 0 && (2.3)\\
\hat{n}\cdot\vec{J}(\mathrm{r}) &= j\omega\rho(\mathrm{r}), && (2.4)
\end{aligned}$$

where $G(\mathbf{r}, \mathbf{r}')$ is the Green's function, V, S are the conductors volume and surface area, respectively, σ_c is the conductor conductivity, ε is the complex dielectric constant including dielectric losses, μ is the magnetic permeability, ω is the angular frequency in radians.

The special case of the standard electro-quasi-static (EQS) capacitance extraction problem is described by imposing just equation (2.2). The special case of the standard magneto-quasi-static (MQS) resistance and inductance extraction problem is described by imposing just equations (2.1) and (2.3).

2.1.1 Discretization

A standard procedure for solving (2.1)-(2.2) involves discretizing the current density $\vec{J}(\mathbf{r})$ and the charge density $\rho(\mathbf{r})$ using piecewise constant basis functions (PCBF). For the charges, we use PCBF supported on triangular or quadrilateral surface panels. For the current density, we use PCBF supported on rectangular filaments. Discretization in the presence of random variations is a difficult task. Indeed, the only way to discretize a structure using a constant number of basis, while still allowing the structure to vary (see Figures 2-1 and 2-2), is to use stochastic basis function (i.e. basis that have a stochastic support):

$$\rho(\mathbf{r}) = \sum_{i=1}^{N_1} \frac{q_i}{S_i} b_i(\mathbf{r}) \tag{2.5}$$

$$\vec{J}(\mathbf{r}) = \sum_{i=1}^{N_2} \frac{I_i}{a_i} B_i(\mathbf{r}) \tag{2.6}$$

where

$$b_i(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} \text{ on surface of panel } i \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

$$B_i(\mathbf{r}) = \begin{cases} 1 & \mathbf{r} \text{ in volume of filament } i \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where N_1 and N_2 are the total number of surface panels and volume filaments, respectively; q_i is the unknown constant charge on panel i, I_i is the unknown constant current inside of filament i, S_i is the surface area of panel i, a_i is the cross-sectional area of filament i. The stochastic nature of the basis functions stems from the observation that S_i, $b_i(\mathbf{r})$, a_i, and $B_i(\mathbf{r})$ are stochastic. Another way to understand the stochastic nature of the basis function, is to realize that in order to keep the total number of surface panels and volume filament constant while still allowing the geometry to vary, is to allow the discretization elements themselves to vary. To emphasize such stochastic nature we will explicitly write the parameter dependence:

$$b_i(\mathbf{r},\mathbf{p}) = \begin{cases} 1 & \mathbf{r} \text{ on surface of panel } i \text{ parameterized by vector } \mathbf{p} \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

$$B_i(\mathbf{r},\mathbf{p}) = \begin{cases} 1 & \mathbf{r} \text{ in volume of filament } i \text{ parameterized by vector } \mathbf{p} \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where $\mathbf{p}$ is a vector of random parameters describing the variations in the geometry. Finally, notice that throughout the rest of this thesis the total number of charge and current density unknowns will be called $N = N_1 + N_2$.

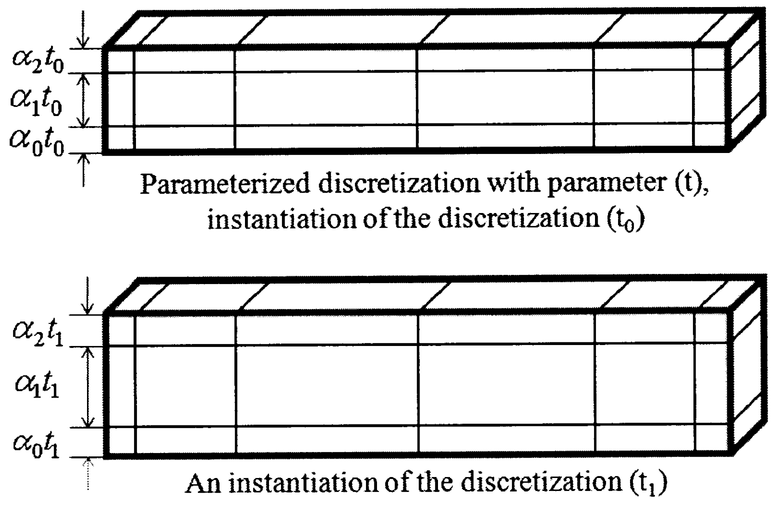

$\alpha_2 t_0$
$\alpha_1 t_0$
$\alpha_0 t_0$
Parameterized discretization with parameter (t), instantiation of the discretization (t_0)
$\alpha_2 t_1$
$\alpha_1 t_1$
$\alpha_0 t_1$
An instantiation of the discretization (t_1)

Figure 2-1: Discretized conductor surface using parameterized basis functions.

The discretized current and charge distributions (2.5-2.6) are substituted in (2.1)-(2.2). A standard Galerkin testing is then used to obtain the dense branch impedance matrix as shown for instance in [42]. Notice that due to the use of Galerkin testing, the test functions

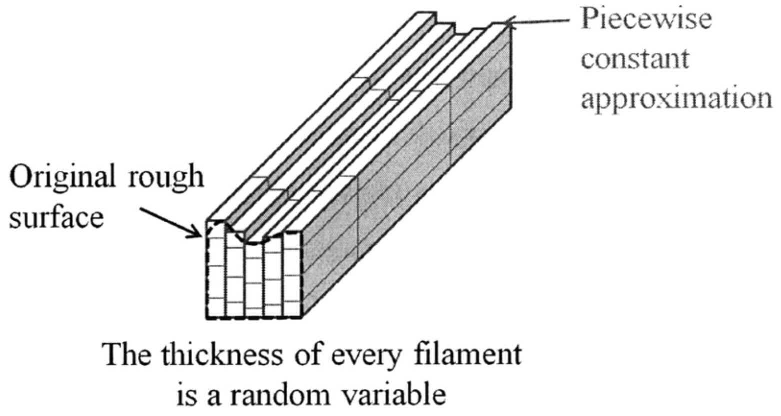

Piecewise constant approximation
Original rough surface
The thickness of every filament is a random variable

Figure 2-2: Discretized conductor volume. Conductor upper surface is rough. The thickness of every filament in a random variable.

are stochastic. Current and charge conservation constraints (2.3)-(2.4) can then be imposed using mesh analysis as described in [42] to obtain a linear system:

$$\mathbf{M}\mathbf{Z}(\mathbf{p})\mathbf{M}^T\,\mathbf{I}_m(\mathbf{p}) = \Delta\phi_m, \tag{2.11}$$

where $\mathbf{M}$ is the very sparse mesh incidence matrix, $\Delta\phi_m$ is the known RHS vector of branch voltages, $\mathbf{I}_m(\mathbf{p})$ is the vector of unknown mesh currents, and

$$\mathbf{Z}(\mathbf{p}) = \begin{bmatrix} \mathbf{R}(\mathbf{p}) + j\omega\mathbf{L}(\mathbf{p}) & 0 \\ 0 & \frac{\Gamma(\mathbf{p})}{j\omega} \end{bmatrix}. \tag{2.12}$$

The elements of the resistance, partial inductance and coefficient of potential matrices are given by:

$$\mathbf{R}_{ii}(\mathbf{p}) = \int\limits_{V_i(\mathbf{p})} \frac{\rho}{a_i(\mathbf{p})a_i(\mathbf{p})} d^3r \tag{2.13}$$

$$,\mathbf{L}_{ij}(\mathbf{p}) = \int\limits_{V_j(\mathbf{p})}\int\limits_{V_i(\mathbf{p})} \frac{G(\mathbf{r},\mathbf{r}')}{a_i(\mathbf{p})a_j(\mathbf{p})} d^3r'd^3r, \tag{2.14}$$

$$\Gamma_{ij}(\mathbf{p}) = \int\limits_{S_j(\mathbf{p})}\int\limits_{S_i(\mathbf{p})} \frac{G(\mathbf{r},\mathbf{r}')}{S_i(\mathbf{p})S_j(\mathbf{p})} d^2r'd^2r, \tag{2.15}$$

where $V_i(\mathbf{p})$, $a_i(\mathbf{p})$ are the stochastic volume and cross sectional area of filament i, respectively, and $S_i(\mathbf{p})$ is the stochastic surface area of panel i. The integrals (2.13-2.15) are stochastic functions of the random process, since the integration domain is described by random variables in $\mathbf{p}$. Throughout the rest of the thesis we assume that the length of $\mathbf{p}$ is N_P. More details related to the stochastic description of $\mathbf{p}$ follow in the next Section 2.2.

Using (2.11), the variation-aware impedance extraction problem can be abstracted as the solution of a general stochastic system

$$\mathbf{A}(\mathbf{p})\,\mathbf{x}(\mathbf{p}) = \mathbf{b} \tag{2.16}$$

where $\mathbf{A}(\mathbf{p}) = \mathbf{M}\mathbf{Z}(\mathbf{p})\mathbf{M}^T$ is a dense matrix of size $N \times N$, $\mathbf{b} = \Delta\phi_m$ is the known RHS, and the objective is to obtain the complete distribution of the $N \times 1$ mesh current vector $\mathbf{x}(\mathbf{p}) = \mathbf{I}_m(\mathbf{p})$. Notice that the special cases of MQS impedance extraction, and EQS capacitance extraction are obtained by making the following substitutions:

- for MQS impedance extraction

$$\mathbf{A}(\mathbf{p}) = \mathbf{M}\left(\mathbf{R}(\mathbf{p}) + j\omega\mathbf{L}(\mathbf{p})\right)\mathbf{M}^T;$$

- for capacitance extraction $\mathbf{A}(\mathbf{p}) = \Gamma(\mathbf{p})$, while $\mathbf{b} = \phi$ is the known potential on the panels, and the objective is to obtain the distribution of charge vector.

In electrical extraction applications, the output quantity $y(\mathbf{p})$ (e.g. capacitance or admittance) is written as a linear function of the unknown vector $\mathbf{x}(\mathbf{p})$

$$y(\mathbf{p}) = \mathbf{c}^T\mathbf{x}(\mathbf{p}) \tag{2.17}$$

where $\mathbf{c}$ is a column vector of size $N \times 1$. In certain applications, such as capacitance extraction, we might be interested in more than a single output quantity (e.g. all coupling capacitances between a target conductor and all surrounding conductors). In such cases the output is a vector rather than a scalar, and the vector $\mathbf{c}$ becomes a matrix $\mathbf{C}$.

In the rest of this part of the thesis we will present algorithms for solving the general abstracted setup (2.16) independent of the underlying application. However, we will also present specific details to resolve computational challenges arising when (2.16) is obtained by discretizing the integral equation formulation of Maxwell's equations.

2.2 Random Variability Modeling

In this thesis we will assume that the vector $\mathbf{p}$ of random process variations is described by a multivariate Gaussian probability density function. We will further assume that in general such random variables are spatially correlated. Based on the spatial correlation function, we divide the variations into strongly spatially correlated and weakly spatially correlated variations.

As an example of strongly spatially correlated variations, we consider surface roughness in "off-chip" interconnects. More precisely, the volume of the conductors V is bounded by rough surfaces S, which we assume is described by a discretized stochastic Gaussian process:

$$\mathcal{W}(\mathbf{p}) = \frac{\exp\left(-0.5\mathbf{p}^T\Sigma^{-1}\mathbf{p}\right)}{(2\pi)^{0.5n}\sqrt{|\Sigma|}}, \tag{2.18}$$

and

$$\Sigma_{ij} = \sigma^2 \exp\left(-\frac{\| \mathbf{r}_i - \mathbf{r}_j \|^2}{L_c^2}\right), \tag{2.19}$$

where L_c is the correlation length, $p_k \in \mathbf{p}$ is the surface height at location $\mathbf{r}_k$, and $\Sigma \in \mathbb{R}^{N_p \times N_p}$ is the covariance matrix. Notice that in general we will assume that the correlation function is Gaussian (2.19) in order to guarantee the smoothness of the surface. However, other correlation functions, such as exponential, have been also applied in the literature [74].

As an example of weakly spatially correlated variations, we will consider the case of "on-chip" interconnect width and height variations. In such case we will in general assume that the random parameters in $\mathbf{p}$ represent uncorrelated Gaussian variations in the geometrical dimensions (e.g. width and height) of the interconnects. Consequently, the joint probability density function of such random Gaussian variations is a product of the individual random variations.

$$\mathcal{P}(\mathbf{p}) = \prod_{i=1}^{N_p} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-0.5\frac{p_i^2}{\sigma_i^2}\right), \tag{2.20}$$

2.2.1 Karhunen-Loeve Expansion

If the vector $\mathbf{p}$ is composed of correlated random variables (for instance $\mathbf{p}$ could be the rough surface height at different locations) then a Karhunen Loeve expansion (or in other words a truncated Singular Value Decomposition SVD) is used to expand the random process as a summation of independent random variables [48],

$$\mathbf{p} \simeq \mathbb{E}[\mathbf{p}] + \mathcal{M}\sqrt{\Lambda}\ \vec{\eta}, \tag{2.21}$$

where $\vec{\eta}$ is the vector of independent Gaussian random variables with zero mean and unit variance, and $\mathcal{M}$ and Λ are the eigenvectors and eigenvalue matrices of the correlation matrix Σ

$$\Sigma\mathcal{M} = \mathcal{M}\Lambda. \tag{2.22}$$

The elements of the matrix $\mathbf{A}(\mathbf{p})$ in (2.16) are then expressed as a function of the expanded random process:

$$\mathbf{A}_{ij}(\mathbf{p}) = \mathbf{A}_{ij}\left(\mathbb{E}[\mathbf{p}] + \mathcal{M}\sqrt{\Lambda}\vec{\eta}\right). \tag{2.23}$$

2.2.2 Polynomial Chaos Expansion

To simplify computations, expression (2.23) is often times expanded in terms of orthogonal basis polynomials $\Psi_k(\vec{\eta})$ in the variable $\vec{\eta}$. Such expansion is referred to as polynomial chaos expansion [76].

$$\mathbf{A}(\mathbf{p}) = \sum_{i=0}^{K} \mathbf{A}_i \Psi_i(\vec{\eta}) \tag{2.24}$$

where K is a function of both N_O, the order of the orthogonal polynomial expansion, and N_M, the number of dominant eigenfunctions obtained from the Karhunen Loeve expansion [7]:

$$K = 1 + \sum_{n=1}^{N_O} \begin{pmatrix} n + N_M - 1 \\ n \end{pmatrix}. \tag{2.25}$$

A complete Askey scheme has been developed [78] to choose the set of orthogonal polynomials such that its weighting function is the probability density function of the set of independent random variables $\vec{\eta}$. For the particular case of Gaussian random variables, the set of probabilistic Hermite polynomials is the standard choice

$$\begin{aligned}
\mathbf{A}_i &= < \mathbf{A}(\mathbf{p}), \Psi_i(\vec{\eta}) > \qquad (2.26)\\
&= \int\limits_{\eta_1} \cdots \int\limits_{\eta_{N_M}} \mathbf{A}\left(\vec{\eta}\right) \Psi_i(\vec{\eta}) \frac{\exp(-\frac{\vec{\eta}^T\vec{\eta}}{2})}{(2\pi)^{\frac{N_M}{2}}} d\eta_1 \cdots d\eta_{N_M},\\
&= \mathbb{E}\left[\mathbf{A}(\vec{\eta})\Psi_i(\vec{\eta})\right] \qquad (2.27)
\end{aligned}$$

For more details on the probabilistic Hermite polynomials and the normalization constants see Appendix A.6.

Despite the fact that the exponential function in (2.26) is separable, the fact that $\mathbf{A}(\mathbf{p})$ in (2.23) is dependent on all the independent random variables in vector $\vec{\eta}$ used to expand the random process, results in an integral (2.26) of very large dimension N_M independent of N_O. Such integrals are computationally impractical to evaluate. Several techniques have been proposed to avoid this inherent complexity, such as Monte Carlo integration (not to be confused with Monte Carlo simulation), and sparse grid integrations [27]. Nevertheless, the problem of calculating a large number of such integrals in a computationally efficient framework remains one of the bottlenecks of any stochastic algorithm that relies on the polynomial chaos expansion. In Section 4.1 we propose a new theorem to overcome this inherent difficulty.

2.2.3 Mathematical Assumptions

Unless explicitly otherwise stated, throughout this part of the thesis we will make the following assumptions:

1. The number of unknowns N is constant over the entire parameter space. Basically we are assuming that we are starting with a fine enough discretization that is accurate even for the most extreme parameter values. Or in other words, we do not need to discretize each time the parameter values vary, but instead the support of the basis is

parameterized and changes dynamically, while still maintaining the desired level of accuracy.

2. The linear system of equations (2.16) is non-singular at any point in the parameter space. It is very obvious that such an assumption holds only if the linear system is representing an accurate discretization of a physical structure. It is furthermore obvious that this assumption cannot hold if the parameter set is modeled using a Gaussian probability density function since such a distribution implies that the parameter values have a nonzero probability of being arbitrary away from their nominal values. To avoid such complications, we will always truncate and renormalize the Gaussian probability density function to its 3σ limits. However, we will abuse the terminology and refer to the truncated Gaussian PDF as a Gaussian PDF.

2.3 Non-Intrusive Sampling Methods

In non-intrusive sampling based methods, the system (2.16) is solved N_S times, for different realizations of the random process $\mathbf{p}$, obtaining the set of solutions $\{\mathbf{x}(\mathbf{p}_1), \mathbf{x}(\mathbf{p}_2), \cdots, \mathbf{x}(\mathbf{p}_{N_S})\}$. The statistical characteristics of $\mathbf{x}(\mathbf{p})$ are then computed from the obtained solution set (samples). The computational complexity of sampling based methods is $O(N_S\rho(N))$, where $\rho(N)$ is the complexity of solving a single system. Examples of non-intrusive methods include the Monte Carlo method [57], and the stochastic collocation method (SCM) [54, 77].

In Monte Carlo methods the parameter space is sampled randomly. The statistics can then be computed from the random samples. However, building a closed form parameterized expression for the output quantity is very complicated (as complicated as interpolation in high dimensional spaces). The main advantage of Monte Carlo methods is that its complexity is theoretically independent of the dimension of the parameter space. The main disadvantage of Monte Carlo methods is the need for a very large number N_S, since the convergence is very slow $O(\frac{1}{\sqrt{N_S}})$.

In the stochastic collocation method (SCM) the problem of constructing a parameterized expression for the output function is solved by expanding the output in terms of multivariate orthogonal polynomials:

$$y(\vec{\eta}) = \sum_{i=1}^{K} y_i \Psi_i(\vec{\eta}) \tag{2.28}$$

where the coefficients are then computed using the quadrature-based samples

$$y_i = \int_{\vec{\eta}} y(\vec{\eta})\Psi_i(\vec{\eta})\mathcal{P}(\vec{\eta}) = \sum_{q=1}^{K} y(\vec{\eta}_q)\Psi_i(\vec{\eta}_q)\mathcal{P}(\vec{\eta}_q)\alpha_q, \tag{2.29}$$

where $\vec{\eta}_q$ and α_q are the quadrature points and the associated weights, respectively. Different quadrature schemes have been proposed in literature. In particular, sparse grid quadrature schemes (such as Smolyak construction [71]) have been proposed to efficiently compute integrals of moderate dimensions and Monte Carlo methods have been proposed to compute integrals of large dimensions. The main disadvantage of the SCM is that a very large number of sample points is required in order to efficiently sample large dimensional parameter spaces.

2.3.1 Discussion of Non-Intrusive Methods

The main advantage of non-intrusive methods is the ability to use the well-developed state of the art deterministic solvers in order to solve the individual linear systems in reduced complexity. In particular, by using the so-called "fast" solvers [59, 43, 1], the complexity of solving the linear system is reduced from $O(N^2)$ to $O(N \log(N))$ or even $O(N)$.

Another advantage of the non-intrusive methods, which has not yet been fully exploited, is that the linear systems produced by the sampling methods are all "similar". One way to exploit such similarity is by solving the linear systems using iterative Krylov subspace methods combined with recycling of the Krylov subspaces [11, 62, 80]. Another possible way to exploit such similarity is to adopt some of the techniques developed in the projection based model reduction community [14, 64, 10]. A third way to exploit such similarity is to combine both the model reduction techniques and the Krylov recycling techniques in a unified framework.

Finally, it should be noted that the main complexity associated with sampling based method is the challenge of accurately and sufficiently sampling a high dimensional parameter space. This challenge is faced by many different research communities. Over the years many techniques have been developed, such as quasi-Monte Carlo sampling [60], Latin Hypercube sampling [56], importance sampling [29], Monte-Carlo Markov Chain [32], optimization-based sampling [10, 72] and many more. The field of stochastic solvers has not yet exploited many of those techniques. The reason might be that more fundamental critical problems are still being addressed and have not been yet fully resolved.

2.4 Intrusive Neumann Expansion Method

The stochastic system matrix $\mathbf{A}(\mathbf{p})$ in (2.16) is first written as $\mathbf{A}(\mathbf{p}) = \mathbf{A}_0 + \Delta_{\mathbf{A}}(\mathbf{p})$, a sum of a deterministic expectation matrix $\mathbf{A}_0 = \mathbb{E}[\mathbf{A}(\mathbf{p})]$ and a stochastic matrix $\Delta_{\mathbf{A}}(\mathbf{p})$. Second, $\mathbf{A}(\mathbf{p})$ is substituted in the linear system (2.16), and the inverse is expanded using the Neumann expansion to obtain:

$$\mathbf{x}(\mathbf{p}) = \mathbf{x}_0 - \mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{x}_0 + \mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{x}_0 + \dots \tag{2.30}$$

where $\mathbf{x}_0 = \mathbf{A}_0^{-1}\mathbf{b}$. The above series converges provided that

$$\max |\lambda(\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p}))| < 1. \tag{2.31}$$

The required statistics of the unknown current vector $\mathbf{x}(\mathbf{p})$ can then be obtained from (2.30), however, such computation (even just the average) is computationally very expensive [28, 86]. As an example, let us consider computing the average of (2.30)

$$\mathbb{E}[\mathbf{x}(\mathbf{p})] = \mathbb{E}[\mathbf{x}_0] - \mathbb{E}\left[\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{x}_0\right] + \mathbb{E}\left[\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{x}_0\right] + \dots \tag{2.32}$$

Using $\mathbb{E}[\mathbf{x}_0] = \mathbf{x}_0$ and $\mathbb{E}[\Delta_{\mathbf{A}}(\mathbf{p})] = 0$, equation (2.32) is rewritten as:

$$\mathbb{E}[\mathbf{x}(\mathbf{p})] = \mathbf{x}_0 + \mathbf{A}_0^{-1}\mathbb{E}\left[\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\right]\mathbf{x}_0 + \dots \tag{2.33}$$

The main complexity is associated with the term $\mathbb{E}\left[\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\right]$, which involves computing the expectation of the product of two stochastic matrices separated by a deterministic matrix. This term can be rewritten using the Kronecker product $\otimes$ and the vector operator vec$(\cdot)$ as [1]:

$$\mathbb{E}\left[\Delta_{\mathbf{A}}(\mathbf{p})\mathbf{A}_0^{-1}\Delta_{\mathbf{A}}(\mathbf{p})\right] = \text{vec}^{-1}\left(\mathbb{E}\left[\Delta_{\mathbf{A}}(\mathbf{p})^T \otimes \Delta_{\mathbf{A}}(\mathbf{p})\right] \text{vec}(\mathbf{A}_0^{-1})\right) \tag{2.34}$$

The complexity of storing and evaluating (2.34) is $O(N^4)$, which is computationally prohibitive. The complexity of computing the statistics increases even more dramatically if the order of the term in the Neumann expansion is increased and/or if we compute high order statistical moments.

A couple of techniques have been proposed to reduced the complexity of computing (2.34) [85, 18]. In particular, the variation-aware capacitance extraction algorithm in [85] relies on using the H-matrix [30] to sparsify both $\mathbf{A}_0^{-1}$ and $\Delta_{\mathbf{A}}(\mathbf{p})^T \otimes \Delta_{\mathbf{A}}(\mathbf{p})$. The final complexity of the algorithm is $O(Nlog^2N)$. On the other hand, the variation-aware resistance/inductance extraction algorithm in [18] relies on using "stochastic" high order basis functions in order to reduce N (without reducing the complexity of the problem $O(N^4)$). Both [85, 18] are specialized for very particular applications. Moreover, they are useful only for computing the average and possibly a low order approximation of the variance. It is very hard to generalize such algorithms to general or more complex applications. Furthermore, computing high order statistics using such a method is for all practical purposes computationally impossible.

2.4.1 Variants of the Neumann Expansion

Another disadvantage of the Neumann expansion technique is that it does not reveal any information on the values of the electrical properties at particular points in the parameter space. Instead, in such technique only the statistical moments of the distribution are computed. The ability to estimate the electrical characteristic at particular points in the

[1]When the vector operator is applied to a matrix $\mathbf{A}$ of dimension $N \times N$ the result is a vector of size $N^2 \times 1$ obtained by stacking the columns of the matrix into a large vector. Using Matlab's notation vec$(\mathbf{A}) = \mathbf{A}(:)$

parameter space is generally essential for both designers and fabrication process control engineers.

To avoid such disadvantage, a couple of variants of the Neumann expansion algorithm have also been proposed in the literature. Such algorithms typically rely on combining the Neumann expansion algorithm with another expansion in order to simplify both computing the statistics and evaluating the value of the output at points in the parameter space. In particular, in [28] the Karhunen Loeve expansion has been combined with the Neumann expansion to develop the improved Neumann expansion, which is efficient for a class of problems in which the system matrix is linear in the variations. Unfortunately, such a condition is typically not satisfied for electrical extraction applications. Another technique which combines the Taylor expansion with the Neumann expansion was developed by [39]. Such a technique is only valid for small perturbation analysis and cannot be generalized to handle large variations. Indeed the technique in [39] can be considered as a high order sensitivity analysis.

2.5 Intrusive Stochastic Galerkin Method (SGM)

The stochastic Galerkin method (SGM) has been developed by Ghanem [28]. It has been traditionally called the stochastic finite element method. In SGM the system matrix $\mathbf{A}(\mathbf{p})$ is first expanded using polynomial chaos (2.26). The unknown $\mathbf{x}(\vec{\eta})$ is then written as an expansion of the same orthogonal polynomials $\mathbf{x}(\vec{\eta}) = \sum_{j=0}^{K} \mathbf{x}_j \Psi_j(\vec{\eta})$, where $\mathbf{x}_j$ are unknown vectors. Both expansions are then substituted in (2.16)

$$\sum_{i=0}^{K} \mathbf{A}_i \Psi_i(\vec{\eta}) \sum_{j=0}^{K} \mathbf{x}_j \Psi_j(\vec{\eta}) = \mathbf{b}. \tag{2.35}$$

A Galerkin testing, i.e. projection on the space of the same set of orthogonal polynomials, is applied to obtain a linear system of equations.

$$\begin{aligned} \sum_{i=0}^{K} \sum_{j=0}^{K} \left\langle \Psi_i(\vec{\eta}) \Psi_j(\vec{\eta}), \Psi_\ell(\vec{\eta}) \right\rangle \mathbf{A}_i \mathbf{x}_j &= \left\langle \mathbf{b}, \Psi_\ell(\vec{\eta}) \right\rangle \\ \forall \ell &\in \{1, \cdots, K\} \end{aligned} \tag{2.36}$$

Equation (2.36) is equivalent to a linear system of the form

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{2.37}$$

where

$$\tilde{\mathbf{A}} = \sum_{i=0}^{K} \left(\begin{pmatrix} \gamma_{i00} & \gamma_{i10} & \cdots & \gamma_{iK0} \\ \gamma_{i01} & \gamma_{i11} & \cdots & \gamma_{iK1} \\ & & \ddots & \\ \gamma_{i0K} & \gamma_{i1K} & \cdots & \gamma_{iKK} \end{pmatrix} \otimes \mathbf{A}_i \right)$$

$$\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x}_0^T & \mathbf{x}_1^T & \cdots & \mathbf{x}_K^T \end{pmatrix}^T, \quad \tilde{\mathbf{b}} = \begin{pmatrix} \mathbf{b}^T & 0 & \cdots & 0 \end{pmatrix}^T,$$

$\gamma_{ij\ell} = \langle \Psi_i(\vec{\eta})\Psi_j(\vec{\eta}), \Psi_\ell(\vec{\eta}) \rangle$ and $\otimes$ is the Kronecker product. The size of this linear system is $O(NK)$, i.e. the original system size N, times K the total number of multivariate orthogonal polynomials used to expand the random function. Notice that for a moderate size 100-dimensional parameter space, K is $5,000$. Solving the corresponding linear system would require $O(K^2N^2)$ time complexity when using an iterative solver, and $O(K^3N^3)$ time complexity when using a direct solver. Such complexity would result in an extremely inefficient electrical extraction algorithm, probably much worse that Monte Carlo simulation.

2.5.1 SGM Research Directions

Ongoing research in the SGM focuses primarily on reducing the complexity of solving (2.37). To achieve such a goal some attempts have been directed towards using the so-called doubly orthogonal polynomials in order to decouple the large system matrix [4, 12]. However, such a technique suffers from several drawbacks. In particular, it is only applicable if the system matrix is a first order expansion of the random parameters. This is a severe limitation since it is an assumption not valid in many problems. Furthermore, in such technique the multivariate basis are constructed using tensor products. Consequently, the number of independent system solves grows exponentially with N_p, which deems such a technique less efficient than standard non-intrusive methods for large dimensional parameter spaces.

Another research trend is to represent the unknown in terms of a reduced space. Examples of techniques typically used to build a reduced basis for the unknown include using an approximated Karhunen-Loeve expansion of the unknown [55], using the generalized spectral decomposition [61] and using stochastic reduced basis [68]. The first technique is computationally very expensive due to the use of the Neumann expansion. The second algorithm is also very expensive due to the need to compute all the basis of the reduced space at once. The final algorithm is exactly equivalent to applying the full orthogonalization Krylov subspace method (FOM) [66] to the full SGM (2.37), and therefore cannot really be considered efficient.

Other attempts have been made toward using approximate "fast" algorithms, such as fast multipole, in order to reduce the complexity of solving the linear system of equations generated by the stochastic Galerkin method. Unfortunately, thus far such attempts have not been successful.

Chapter 3

Contributions to Non-intrusive Methods

As mentioned in Section 2.3, non-intrusive methods rely on sampling the parameter space p (or equivalently $\vec{\eta}$) at different points in the space $\vec{\eta}_k$ and solving the linear system

$$\mathbf{A}(\vec{\eta}_k)\mathbf{x}(\vec{\eta}_k) = \mathbf{b} \tag{3.1}$$

$$y(\vec{\eta}_k) = \mathbf{c}^T\mathbf{x}(\vec{\eta}_k) \tag{3.2}$$

It is apparent that in non-intrusive methods (sampling-based methods) we need to solve a very large number of "similar" systems. This similarity is due to the fact that all systems share the same topology and just differ by the numerical values of the geometrical dimensions parameterizing the topology (see Figure 3-1). Such similarity has not been explicitly exploited by non-intrusive stochastic methods. The objective of this chapter is to exploit such similarity in order to reduce the average total computational time required for solving a large number of "similar" configurations. We do that using two different techniques:

- We develop Krylov subspace recycling strategies, which enable reusing the computations performed during the solution of one linear system in order to reduce the computational effort required for solving another linear system.

  Specifically, we first present a new deterministic *Galerkin Krylov Recycling (GKR)* algorithm, which improves standard Krylov subspace recycling for the solution of the large number of similar linear systems, produced by sampling the space of random

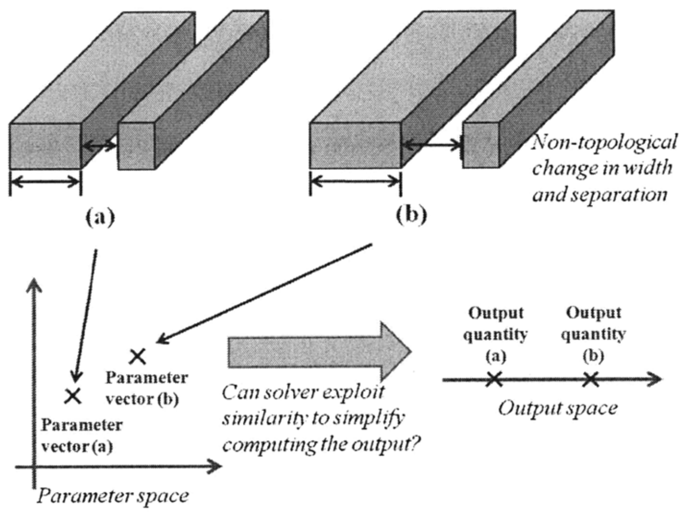


Figure 3-1: Two similar structures. Can solver exploit such similarity?

geometrical variations. In addition, we investigate different strategies to determine the sequence in which the linear systems are solved to achieve maximum recycling. Finally, we provide a proximity measure to quantify the degree of similarity between sample linear systems. Such proximity measure is used to cluster the systems into subgroups such that the explored subspace is only shared among similar systems.

- We use projection based model reduction to reduce the complexity of solving the stochastic linear system (by reducing its dimension). We present and proof a technique to build the projection matrices such that the statistical moments (and/or the coefficients of the projection of the stochastic vector on some orthogonal polynomials) computed from the reduced system are the same as those computed from the original system.

Notational Note. Throughout this chapter, we will freely interchange between the notations $\mathbf{A}_k$ and $\mathbf{A}(\vec{\eta}_k)$.

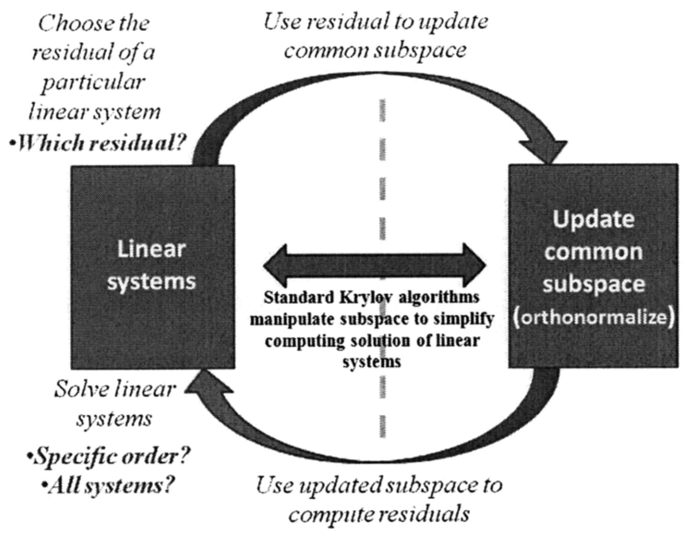

Choose the residual of a particular linear system
•Which residual?
Use residual to update common subspace
Linear systems
Update common subspace (orthonormalize)
Standard Krylov algorithms manipulate subspace to simplify computing solution of linear systems
Solve linear systems
•Specific order?
•All systems?
Use updated subspace to compute residuals

Figure 3-2: Main recycling concept. In red are the questions addressed in this subsection

3.1 Non-intrusive Sampling using Galerkin Krylov Method for Recycling (GKR)

In this section we discuss a Krylov subspace technique that is based on orthogonalizing the residual with respect to the Krylov search space via a Galerkin projection. We will refer to our algorithm as Galerkin Krylov method for Recycling (GKR). The recycling aspect of our algorithm is discussed in the next subsection. Notice that the proposed algorithm differs from the full orthogonalization method (FOM) [66], since we do not use the Hessenberg decomposition, nor do we choose the orthogonal basis vectors such that they are compatible with such decomposition.

To solve a linear system $\mathbf{A}_k\mathbf{x}_k = \mathbf{b}$ using GKR, at iteration $n+1$, the unknown vector is first expressed as $\mathbf{x}_k = \mathbf{Q}_{n+1}\mathbf{z}_k$, where $\mathbf{Q}_{n+1} \in \mathcal{K}(\mathbf{A}_k, \mathbf{b})$ is an orthogonal matrix of size $N \times (n+1)$ in the Krylov subspace of $\mathbf{A}_k$ and $\mathbf{b}$. Matrix $\mathbf{Q}_{n+1}$ is constructed by augmenting the space spanned by $\mathbf{Q}_n$ by the orthogonal projection of the residual on the compliment of $\mathbf{Q}_n$:

$$\begin{aligned} \mathbf{q}_{n+1} &= \frac{\left(\mathbf{I}_n - \mathbf{Q}_n\mathbf{Q}_n^H\right)\mathbf{r}_n}{\|\left(\mathbf{I}_n - \mathbf{Q}_n\mathbf{Q}_n^H\right)\mathbf{r}_n\|} \\ \mathbf{Q}_{n+1} &= \begin{bmatrix} \mathbf{Q}_n & \mathbf{q}_{n+1} \end{bmatrix} \end{aligned} \tag{3.3}$$

A Galerkin projection is then used to orthogonalize the residual $\mathbf{A}_k\mathbf{Q}_{n+1}\mathbf{z}_k - \mathbf{b}$ with respect to the subspace spanned by $\mathbf{Q}_{n+1}$:

$$\mathbf{Q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_{n+1}\mathbf{z}_k - \mathbf{Q}_{n+1}^H\mathbf{b} = 0.$$

Notice that matrix $\mathbf{Q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_{n+1}$ is assembled by extending the matrix $\mathbf{Q}_n^H\mathbf{A}_k\mathbf{Q}_n$ available at step n:

$$\mathbf{Q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_{n+1} = \begin{pmatrix} \mathbf{Q}_n^H\mathbf{A}_k\mathbf{Q}_n & \mathbf{Q}_n^H\mathbf{A}_k\mathbf{q}_{n+1} \\ \mathbf{q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_n & \mathbf{q}_{n+1}^H\mathbf{A}_k\mathbf{q}_{n+1} \end{pmatrix} \tag{3.4}$$

Only the following two matrix-vector products are required at each step: $\mathbf{A}_k\mathbf{q}_{n+1}$ and $\mathbf{q}_{n+1}^H\mathbf{A}_k$. Furthermore, Givens rotation-based QR decomposition [66] is used to efficiently compute $\mathbf{z}_k$. The proposed algorithm can handle non-symmetric non-positive definite matrices, which is essential since many of the system matrices generated via field solvers (e.g. coefficient of potential matrix generated by a collocation method) are not symmetric and are often numerically non-positive definite.

3.1.1 Krylov Subspace Recycling using GKR

The main idea behind Krylov subspace recycling is to *cheaply test* if the solution of system $k+1$ is contained in the subspace explored during the solution of systems 1 to k. If the solution is contained in the subspace, then it can be cheaply and easily computed. Otherwise, the current subspace is expanded until it includes the solution of system $k+1$ (to the desired accuracy). Two important steps are required in order to do such a test:

1. Map the explored subspace into the $\mathbf{A}_{k+1}$-space
2. Find the best representation of the RHS in the projected space

The fundamental premise behind recycling algorithms is that both the above steps can be implemented efficiently. The mapping of the explored subspace $\mathbf{Q}_{n+1}$ onto $\mathbf{A}_{k+1}$-space involves the product

$$\mathbf{A}_{k+1}\mathbf{Q}_{n+1}. \tag{3.5}$$

Computing the matrix-matrix product (3.5) is generally very expensive. However, for cases in which the matrices can be expressed as

$$\mathbf{A}_k = \mathbf{A}_0 + \Delta_k \tag{3.6}$$

where $\mathbf{A}_0$ is some nominal matrix and Δ_k is some sparse, low rank, or sparsifiable matrix, then the product (3.5) can be computed very efficiently. In this subsection we will consider cases in which Δ_k is sparse. In particular, we will consider similar linear systems generated by structured (e.g. sparse grid) sampling of a weakly coupled physical parameter space. Such parameter spaces are typically used to model on-chip width and thickness variations, but not the strongly coupled off-chip surface roughness variations. For more details on the sparsity patterns of the matrices Δ_k see Subsection 4.3.1.

The second step, which involves finding the optimal presentation of the RHS in the mapped space, typically requires solving a linear system. In the majority of Krylov subspace algorithms the generation of the subspace and the solution of such linear system are closely coupled (e.g. using A-orthonormalization in GCR or Hessenberg decomposition in GMRES)[1]. Such coupling (which corresponds to additional coupling between the left and right sides of Figure 3-2) typically simplifies solving the linear system, however it presents a huge overhead when using recycling to solve multiple linear systems. As explained next, our proposed algorithm gets rid of such unnecessary coupling.

In GKR, when solving the system $\mathbf{A}_{k+1}\mathbf{x}_{k+1} = \mathbf{b}$, we first use the explored subspace $\mathbf{Q}_{n+1}$ (explored during the solution of systems 1 to k) to find an approximate solution

$$\mathbf{Q}_{n+1}^H \mathbf{A}_{k+1} \mathbf{Q}_{n+1} \mathbf{z}_{k+1} = \mathbf{Q}_{n+1}^H \mathbf{b}. \tag{3.7}$$

We efficiently compute the term

$$\mathbf{Q}_{n+1}^H \mathbf{A}_{k+1} \mathbf{Q}_{n+1} = \mathbf{Q}_{n+1}^H \mathbf{A}_0 \mathbf{Q}_{n+1} + \mathbf{Q}_{n+1}^H \Delta_{k+1} \mathbf{Q}_{n+1}. \tag{3.8}$$

[1]For more details examine the standard recycling algorithms in Appendix A.3. In particular, for the A-orthonormalization in GCR examine step 4 in Algorithm 21, and for the Hessenberg decomposition in GMRES examine step 4 in Algorithm 22

by using the following remarks:

1. Both terms in (3.8) are computed using the idea proposed in (3.4).
2. The first term is computed only once and then shared among all the linear systems.
3. The second term is relatively very cheap to compute since it involves the sparse matrix Δ_{k+1}.

We then efficiently solve (3.7) using the Givens rotation-based QR decomposition [66]. If the residual $\mathbf{b}-\mathbf{A}_{k+1}\mathbf{Q}_{n+1}\mathbf{z}_{k+1}$ is small, then we proceed to the next system. Otherwise, we use (3.3) to update the explored subspace until the residual becomes sufficiently small. Due to the assumed "similarity" between the linear systems, we expect that some sufficiently large $\mathbf{Q}_{n+1}$ (but of dimension much less than N) will include the solution of all linear systems $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$. Notice that the expanded subspace can then be used to solve any subsequent system without the need for any special manipulations.

Algorithm 1 Galerkin Krylov for Recycling (GKR)

```
1:  $n \leftarrow 0, \mathbf{Q}_n \leftarrow \mathbf{b}$
2:  **for** each linear system $k$ **do**
3:      $\mathbf{q}_{n+1} \leftarrow [\ ]$
4:      **repeat**
5:          $\mathbf{Q}_{n+1} \leftarrow \left[\ \mathbf{Q}_n \quad (\mathbf{I}_{n\times n} - \mathbf{Q}_n\mathbf{Q}_n^H)\mathbf{q}_{n+1}\ \right]$
6:          compute $\mathbf{Q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_{n+1}$
7:          solve for $\mathbf{z}_k$, $\left(\mathbf{Q}_{n+1}^H\mathbf{A}_k\mathbf{Q}_{n+1}\right)\mathbf{z}_k = \mathbf{Q}_{n+1}^H\mathbf{b}$.
8:          $\mathbf{r}_k \leftarrow \mathbf{b} - \mathbf{A}_k\mathbf{Q}_{n+1}\mathbf{z}_k$
9:          $n \leftarrow n+1$
10:         $\mathbf{q}_{n+1} \leftarrow \mathbf{r}_k$
11:     **until** convergence achieved
12: **end for**
```

The main advantage of Algorithm 1 is that the construction of the matrix $\mathbf{Q}_{n+1}$ (Step 5) is independent of the matrix $\mathbf{A}_k$. This is the fundamental difference from the GMRES recycling algorithm which requires a matrix dependent Hessenberg decomposition, and from the GCR recycling algorithm which requires a matrix based orthogonalization.

3.1.2 Concurrent Krylov Subspace Recycling

In general all the linear systems $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$ are available before any linear system solves. Consequently, we are at liberty to solve such systems in any particular order. In fact, we are even at liberty to solve the linear systems concurrently. In particular want to determine the sequence in which the linear systems are solved such that the amount of recycling is maximized. Four different algorithms, providing different options for the concurrent solution of all the given linear system, are investigated in this subsection (see Algorithm 2). Namely, one can choose to process first a linear system with the minimum (m) or maximum (M) residual, and one can make such a choice after each iteration (F: fine grain), or after each complete linear system solve (C: coarse grain).

1. Option (M) guarantees to monotonically reduce the maximum residual over all systems. In addition, the search direction associated with the largest residual is more likely to reduce the residual of a larger number of linear systems, since it is a direction that has not been explored before.

2. On the other hand, the benefit of choosing the system with the minimum residual (Option (m)) is that with a large probability the number of unsolved systems will be reduced, and therefore the incremental work per step will be reduced. Option (m) in Algorithm 2 also indicates that with a high probability the linear systems will be solved sequentially, ordered based on ascending residual norm. On the other hand, a larger number of iterations will be probably needed to explore the entire space.

3. In option (C) a system is first chosen, then fully solved. The resulting subspace is then used to compute an approximate solution for all remaining systems. Then according to options (m) or (M) a next system is chosen and the process is repeated. Consequently, option (C) is a sequential algorithm, however, the sequence in which the linear systems are solved is determined dynamically (see Figure 3-3) based on the magnitude of the residual following each system solve.

4. On the other hand, in option (F) the residual of all remaining systems is updated concurrently. This option results in a strictly parallel algorithm (see Figure 3-3). This

observation clarifies most of the advantages and disadvantages of such option. In particular, the main advantage is that the amount of available up to date information is maximized at every iteration, which enables us to make the best informed choices (i.e. choose the most appropriate system to proceed with). On the other hand, the main disadvantage is that all systems have to be uploaded to the memory at once, which in many cases constitutes a memory bottleneck. Moreover, the communication of information between systems might result in a significant computational overhead.

Algorithm 2 Concurrent Krylov Subspace Recycling Framework: Fine Grain (ml, MF), Coarse Grain (mC, MC)

```
1: $h \leftarrow 1, \mathbf{r}_h \leftarrow \mathbf{b}$
2: while some systems still need to be solved do
3:     $\mathbf{Q} \leftarrow \mathbf{A}_0\mathbf{r}_h$, where 0 is the index of the nominal system
4:     for each unsolved system $k$ do
5:         $\mathbf{A}_k\mathbf{r}_h \leftarrow \mathbf{Q} + (\mathbf{A}_k - \mathbf{A}_0)\mathbf{r}_h$
6:         do algorithm iteration (GCR orthogonalization, or GMRES Hessenberg form, or
           GKR QR for system solve)
7:         get residual $\mathbf{r}_k$
8:         if coarse grain (mC or MC) AND $\mathbf{r}_k <$ threshold then
9:             $h \leftarrow \text{argmin}\{\mathbf{r}_k\}$ (or $h \leftarrow \text{argmax}\{\mathbf{r}_k\}$) over all unsolved systems.
10:        end if
11:    end for
12:    if fine grain (mF or MF) then
13:        $h \leftarrow \text{argmin}\{\mathbf{r}_k\}$ (or $h \leftarrow \text{argmax}\{\mathbf{r}_k\}$) over all unsolved systems.
14:    end if
15: end while
```

3.1.3 Proximity Measure for Subspace Recycling

Given the fact that in non-intrusive methods we typically need to solve a very large number of systems, it might be the case that not all the systems share the same small subspace, but rather clusters of linear systems share the same small subspaces. It is therefore important to group such systems based on their proximity, in order to guarantee reducing the computation per iteration, and possibly even the number of iterations. We propose to use as a cheap proximity measure between the system $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$ and another system $\mathbf{A}_k\mathbf{x}_k = \mathbf{b}$ the

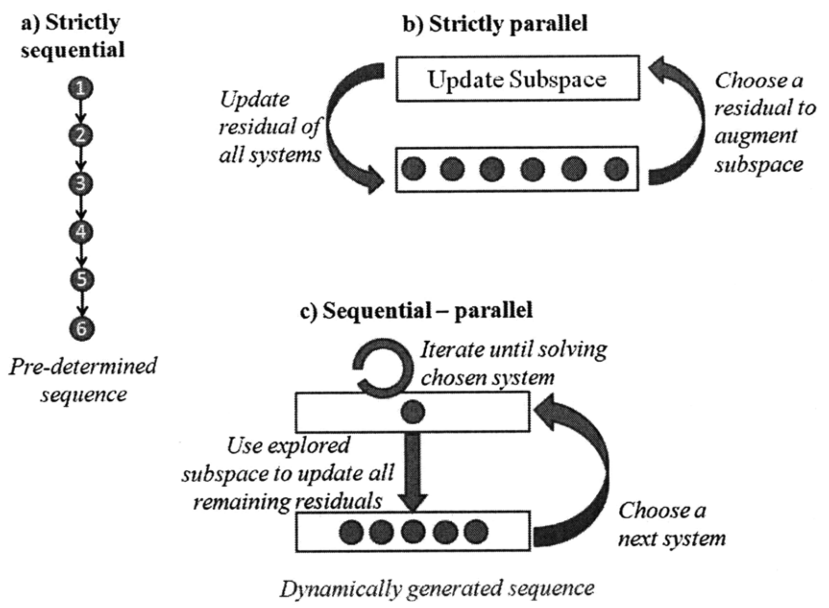

a) Strictly sequential
1
2
3
4
5
6
Pre-determined sequence
b) Strictly parallel
Update Subspace
Update residual of all systems
Choose a residual to augment subspace
c) Sequential – parallel
Iterate until solving chosen system
Use explored subspace to update all remaining residuals
Choose a next system
Dynamically generated sequence

Figure 3-3: Three different sequencing schemes to solve the set of linear systems. a) Sequential b) Fine Grain Parallel (F) c) Coarse Grain Parallel (C)

following quantity

$$d(\mathbf{A}_k - \mathbf{A}_i, \mathbf{x}_i) = \| (\mathbf{I} - \mathbf{Q}_i\mathbf{Q}_i^H)(\mathbf{A}_k - \mathbf{A}_i)\mathbf{x}_i \|_2, \tag{3.9}$$

where $\mathbf{Q}_i$ is an orthogonal matrix whose column span is the Krylov subspace $\mathcal{K}(\mathbf{A}_i, \mathbf{b})$, explored during the solution of the linear system $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$. The following proposition demonstrates that the proximity measure is small when the solution of the two systems are similar.

Proposition 3.1.1. *If* $\max \lambda(\mathbf{A}_i^{-1}(\mathbf{A}_k - \mathbf{A}_i)) < 1$ *and* $d(\mathbf{A}_k - \mathbf{A}_i, \mathbf{x}_i) = 0$*, then a second-order approximation for* $\mathbf{x}_k$ *can be obtained by recycling the subspace* $\mathbf{Q}_i$*.*

Algorithm 3 summarizes how the proposed proximity measure can be used to effectively cluster the set of linear systems such that maximum recycling is achieved. Notice that given a solution $\mathbf{x}_i$ and the explored subspace $\mathbf{Q}_i$ for the system $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$, the proximity measure $d(\mathbf{A}_k - \mathbf{A}_i, \mathbf{x}_i)$ is computed in $O(N)$ for every unsolved systems $\mathbf{A}_k\mathbf{x}_k = \mathbf{b}$. Only those unsolved systems with small $d(\mathbf{A}_k - \mathbf{A}_i, \mathbf{x}_i)$ are considered nearby systems of $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}$.

Algorithm 3 Linear System's Clustering for Efficient Recycling

```
1: $i \leftarrow 0$
2: $n \leftarrow 1$ ($n$ is the cluster counter)
3: while some systems still need to be solved do
4:     solve $\mathbf{A}_i \mathbf{x}_i = b$ for $\mathbf{x}_i$ building orthogonal $\mathbf{Q}_i$ s.t. column span of $\mathbf{Q}_i$ is $\mathcal{K}(\mathbf{A}_i, \mathbf{b})$.
5:     for each unsolved linear system $k$ do
6:         $\mathbf{D}(k,n) \leftarrow d(\mathbf{A}_k - \mathbf{A}_i, \mathbf{x}_i)$ using (3.9)
7:     end for
8:     Solve together using recycling all systems for which $\mathbf{D}(k,n)$ is smaller than threshold.
9:     $i \leftarrow \text{argmax}_k\{\min_n \mathbf{D}(k,n)\}$, over all unsolved systems $k$
10:    $n \leftarrow n+1$
11: end while
```

3.2 Statistical Moment Preserving Model Order Reduction

An apparent parallel research area is that of parameterized linear model order reduction (PROM). In particular, there is clear similarity between non-intrusive variation-aware solvers and multi-point-matching projection-based model reduction methods [64], since both of them inherently rely on solving the given linear system at different sample points in the parameter space. In this section we develop an efficient non-intrusive variation-aware extraction algorithm, by using a multi-point matching projection-based model reduction approach. In particular, we develop a method to generate the projection matrix, such that the statistical moments of the original system (or equivalently the coefficients of the projection of the statistical quantities on some space of orthogonal polynomials) are preserved when computed from the reduced system.

Important Notice: Most applications encountered in the parameterized model reduction literature are characterized by small dimensional parameter spaces (e.g. < 20). The objective in such applications is to generate small order models (e.g. $O(10)$). On the other hand, in variation-aware extraction we are instead faced with applications characterized by very large dimensional parameter spaces (> 100). The objective is to generate reduced models of $O(1000)$, since such models will be computationally significantly faster to simulate than any available non-intrusive or intrusive method. Furthermore, while for parameterized model order reduction the model generation cost is less of an issue because it can be done offline, in variation-aware extraction the model generation cost is absolutely critical since it typically represents more than 90% of the total computation time.

3.2.1 Main Theorem

Our objective is to construct the basis for the reduced space (the right projection matrix) such that the statistical moments computed from our reduced model match exactly the statistical moments computed from the original system

$$\begin{aligned}\mathbb{E}\left[y(\vec{\eta})^k\right] &= \int y(\vec{\eta})^k \mathcal{P}(\vec{\eta})d\vec{\eta} = \int \left(\mathbf{c}^T\mathbf{A}(\vec{\eta})^{-1}\mathbf{b}\right)^k \mathcal{P}(\vec{\eta})d\vec{\eta} \\ &\simeq \sum_{q=1}^{N_q} \alpha_q(\mathbf{c}^T\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b})^k\mathcal{P}(\vec{\eta}_q) \end{aligned} \qquad (3.10)$$

where $\mathcal{P}(\vec{\eta})$ is the multivariate probability density function (PDF) associated with $\vec{\eta}$, α_q are the weights associated with the quadrature points $\vec{\eta}_q$ and N_q is the total number of quadrature points required by the numerical quadrature scheme used to calculate the integral (N_q increases as k increases).

Instead of (or in addition to) preserving statistical moments, we might be also interested in matching the coefficients β_i of the projection of the solution $y(\vec{\eta}) = \sum_{i=0}^{K} \beta_i \Psi_i(\vec{\eta})$ on the space of multivariate polynomials, $\Psi_i(\vec{\eta})$, orthogonal with respect to the probability measure $\mathcal{P}(\vec{\eta})$, e.g. Hermite polynomials [78]:

$$\begin{aligned}\beta_i = \langle y(\vec{\eta}), \Psi_i(\vec{\eta})\rangle &= \int y(\vec{\eta})\Psi_i(\vec{\eta})\mathcal{P}(\vec{\eta})d\vec{\eta} \\ &\simeq \sum_{q=1}^{N_q} \alpha_q \mathbf{c}^T\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b}\Psi_i(\vec{\eta}_q)\mathcal{P}(\vec{\eta}_q) \end{aligned} \qquad (3.11)$$

Theorem 3.2.1 below provides a way to achieve both objectives, i.e. matching of the statistical moments, and matching of the coefficients for a multivariate Hermite representation of the solution.

Theorem 3.2.1. *IF* $\mathbf{U}$ *is an orthonormal projection matrix and*

$$\textit{colspan}\{\mathbf{U}\} \supset \textit{span}\{\mathbf{b}, \mathbf{A}(\vec{\eta}_1)^{-1}\mathbf{b}, \mathbf{A}(\vec{\eta}_2)^{-1}\mathbf{b}, \cdots, \mathbf{A}(\vec{\eta}_{N_q})^{-1}\mathbf{b}\}$$

where $\{\vec{\eta}_q : q = 1, \cdots, N_q\}$ *is a set of quadrature points such that (3.10) and (3.11) are exactly integrable for some choice of orthogonal polynomials* $\Psi_i(\vec{\eta})$ *THEN*

$$\mathbb{E}\left[y(\vec{\eta})^k\right] = \mathbb{E}\left[y_r(\vec{\eta})^k\right] \tag{3.12}$$

and

$$\langle y(\vec{\eta}), \Psi_i(\vec{\eta})\rangle = \langle y_r(\vec{\eta}), \Psi_i(\vec{\eta})\rangle \tag{3.13}$$

where the reduced output $y_r(\vec{\eta})$ *is given by*

$$y_r(\vec{\eta}) = \mathbf{c}^T\mathbf{U}(\mathbf{U}^H\mathbf{A}(\vec{\eta})\mathbf{U})^{-1}\mathbf{U}^H\mathbf{b} \tag{3.14}$$

Proof. Using a quadrature rule with the given points on the reduced system we obtain:

$$\begin{aligned}
\mathbb{E}\left[y_r(\vec{\eta})^k\right] &= \sum_{q=1}^{N_q} \alpha_q \left(\mathbf{c}^T\mathbf{U}(\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)\mathbf{U})^{-1}\mathbf{U}^H\mathbf{b}\right)^k \mathcal{P}(\vec{\eta}_q) \\
\langle y_r(\vec{\eta}), \Psi_i(\vec{\eta})\rangle &= \sum_{q=1}^{N_q} \alpha_q \mathbf{c}^T\mathbf{U}(\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)\mathbf{U})^{-1}\mathbf{U}^H\mathbf{b}\Psi_i(\vec{\eta}_q)\mathcal{P}(\vec{\eta}_q)
\end{aligned}$$

Using first lemma 3, and then lemma 2 in [14] the common term in both expressions becomes:

$$\mathbf{c}^T\mathbf{U}(\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)\mathbf{U})^{-1}\mathbf{U}^H\mathbf{b} = \mathbf{c}^T\mathbf{U}\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b} = \mathbf{c}^T\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b}$$

therefore

$$\begin{aligned}
\mathbb{E}\left[y_r(\vec{\eta})^k\right] &= \sum_{q=1}^{N_q} \alpha_q(\mathbf{c}^T\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b})^k\mathcal{P}(\vec{\eta}_q) = \mathbb{E}\left[y(\vec{\eta})^k\right] \\
\langle y_r(\vec{\eta}), \Psi_i(\vec{\eta})\rangle &= \sum_{q=1}^{N_q} \alpha_q\mathbf{c}^T\mathbf{A}(\vec{\eta}_q)^{-1}\mathbf{b}\Psi_i(\vec{\eta}_q)\mathcal{P}(\vec{\eta}_q) =< y(\vec{\eta}), \Psi_i(\vec{\eta}) >
\end{aligned}$$

□

It should be noted that even if the integration rule does not accurately compute the moments of the full system, the moments computed from the reduced system will match those *numerically* computed using the full system.

Theorem 3.2.1 does not rely on a particular integration rule. Consequently we are at liberty to choose any of the available numerical integrations schemes, including but not limited to Monte Carlo, Quasi Monte Carlo, importance sampling, tensor products, or sparse grid products. Furthermore, since what we really want is to accurately represent the space spanned by $\mathbf{U}$, rather than to accurately compute the integral, we are at liberty to use even those complicated optimization methods for sampling point selections [10]. For simplicity, we will construct $\mathbf{U}$ using sparse grid quadrature schemes if the parameter space dimension is less than 100 and Monte Carlo integration otherwise.

3.2.2 Proximity Measure for Computational Efficiency

Stacking the solutions at all quadrature points in one projection matrix would result in three different problems, namely, it would require a very large number of linear system solves and would result in a large-dimensional and ill-conditioned projection matrix. The standard solution for the second and third problems (which does not address the first one) is to compress the projection subspace using a singular value decomposition (SVD).

We propose here instead an alternative solution that addresses *all* three problems at the same time. Specifically, before solving a particular large system at a new quadrature point $\vec{\eta}_q$, we propose to compute instead the following residual

$$\mathbf{r}(\vec{\eta}_q) = \mathbf{b} - \mathbf{A}(\vec{\eta}_q)\mathbf{x}_r(\vec{\eta}_q) = \mathbf{b} - \mathbf{A}(\vec{\eta}_q)\mathbf{U}\left(\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)\mathbf{U}\right)^{-1}\mathbf{U}^H\mathbf{b}$$

If the norm of the residual is large, then the current subspace does not faithfully represent the solution and therefore needs to be expanded with the solution of the new system orthogonalized with respect to the current subspace. If, on the other hand, the norm is small then the solution at the new sampling point is accurately represented using the currently explored subspace and does not need to be added to the basis. The advantage of using such a proximity measure is that only *one* matrix-vector product is done in the original space $O(N^2)$ and all the rest of the computation, i.e. computing $\mathbf{x}_r(\vec{\eta})$, is done in the reduced space, and is therefore very efficient. Notice that a similar residual measure was used in [72] to determine the points at which the parameter space should be sampled.

Algorithm 4 Stochastic Model Reduction Method for Solving Linear Systems with Random Matrices.

```
1: $\mathbf{U} \leftarrow \mathbf{b}$
2: $q \leftarrow 0$
3: for each quadrature point $\vec{\eta}_q$ do
4:   $q \leftarrow q+1$
5:   generate linear system $\mathbf{A}(\vec{\eta}_q)$.
6:   compute $\mathbf{x}_r(\vec{\eta}_q) \leftarrow \mathbf{U}\left(\mathbf{U}^H\mathbf{A}(\vec{\eta}_q)\mathbf{U}\right)^{-1}\mathbf{U}^H\mathbf{b}$
7:   compute residual $\mathbf{r}(\vec{\eta}_q) \leftarrow \mathbf{b} - \mathbf{A}(\vec{\eta}_q)\mathbf{x}_r(\vec{\eta}_q)$.
8:   if $\frac{||\mathbf{r}(\vec{\eta}_q)||}{||\mathbf{b}||} >$ theshold then
9:     solve for $\mathbf{x}(\vec{\eta}_q)$, $\mathbf{A}(\vec{\eta}_q)\mathbf{x}(\vec{\eta}_q) = \mathbf{b}$
10:    extend the basis $\mathbf{U}$ with $\mathbf{x}(\vec{\eta}_q) - \mathbf{U}\mathbf{U}^H\mathbf{x}(\vec{\eta}_q)$
11:  end if
12: end for
```

Algorithm 4 summarizes our complete stochastic model reduction variation-aware extraction approach. The complexity of computing the Galerkin projection in Step 6 of Algorithm 4 is reduced by using the multivariate Hermite expansion of the system matrix (2.24). More precisely, each time the basis $\mathbf{U}_{r+1} = \begin{bmatrix} \mathbf{U}_r & \mathbf{u} \end{bmatrix}$ is expanded from $\mathbf{U}_r$ by $\mathbf{u}$, the reduced system matrix can be updated as follows

$$\mathbf{U}_{r+1}^H\mathbf{A}(\vec{\eta})\mathbf{U}_{r+1} = \sum_{i=0}^{K}\mathbf{U}_{r+1}^H\mathbf{A}_i\mathbf{U}_{r+1}\Psi_i(\vec{\eta})$$

where

$$\begin{bmatrix} \mathbf{U}_r^H \\ \mathbf{u}^H \end{bmatrix}\mathbf{A}_i\begin{bmatrix} \mathbf{U}_r & \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_r^H\mathbf{A}_i\mathbf{U}_r & \mathbf{U}_r^H\mathbf{A}_i\mathbf{u} \\ \mathbf{u}^H\mathbf{A}_i\mathbf{U}_r & \mathbf{u}^H\mathbf{A}_i\mathbf{u} \end{bmatrix} \tag{3.15}$$

Since the terms $\mathbf{U}_r^H\mathbf{A}_i$, $\mathbf{A}_i\mathbf{U}_r$ and $\mathbf{U}_r^H\mathbf{A}_i\mathbf{U}_r$ are readily available at iteration $r+1$, we only need to compute $\mathbf{A}_i\mathbf{u}$ and $\mathbf{u}^H\mathbf{A}_i$.

3.2.3 Further Model Enhancement

Following the computation of an appropriate projection matrix, the reduced model is constructed. The validity of such model is verified by computing the residual produced by the model at randomly generated points in the parameter space. If the residual is large at any of the testing points then the solution at such a point is used to expand the projection matrix. The validation of the model is terminated when the success rate, i.e. the number of systems with residual smaller than a certain threshold, is larger than some preset percentage (e.g. 99%). The reduced model can then be used to estimate the unknown vector at any sample point in the parameter space. Notice, that the addition of such basis to the projection matrix still guarantees the preservation of the statistical moments (or the projection coefficients). In fact, any addition to the projection subspace does not affect the moments, but only adds to the accuracy of the model.

3.2.4 Computational Complexity

The computational complexity of our method is $O(N_S N^2)$, where N_S is the final size of the reduced space. The main computational advantages of our method is that in large dimensional problems an accurate model is constructed using a number of solves N_S much smaller than that required by the non-intrusive stochastic collocation method (SCM) N_q. This is primarily because the unknowns at many of the points in the parameter space can be accurately computed using the reduced model, which is computationally extremely efficient.

3.2.5 Relation to non-intrusive methods and to variational TBR

The main connection between our method, the non-intrusive stochastic collocation method (SCM) [77], and the variational TBR (vTBR) [64] is the fact that the sample points are generated based on a suitable quadrature scheme. In both our method and in SCM, any number of statistical moments can be explicitly preserved. This allows for proper choice of the integration scheme based on the order of the statistical moments to be preserved. On the other hand, in vTBR first and second order statistics are coincidentally enforced, since vTBR approximates the Grammian (which is also the covariance matrix) using quadrature. Consequently, any statistical moments of more than second order cannot be preserved. Furthermore, if the objective is to preserve just first order statistics then vTBR results in extra unnecessary computations since it implicitly preserves second order statistics. Consequently, variational-TBR can be considered a special case of our more general statistical moment matching technique in which the covariance matrix is preserved $\mathbb{E}\left[\mathrm{x}\mathrm{x}^T\right]$.

On the other hand, both our method and vTBR use explicitly the projection approach. This allows for the efficient truncation of the model by finding a reduced basis to represent the column span of the solution vectors at the different sample points. This is not the case of the stochastic collocation method, in which the solution at *all* the sample points are explicitly used to construct the model. Consequently, for large dimensional parameter spaces the stochastic collocation method becomes inefficient.

3.3 Results

3.3.1 On-chip Parameter-Variation using GKR and SMOR

In this section we show a typical on-chip parasitic extraction example with 16 conductors. In a cross-section view, the conductors are arranged on a 4×4 regular grid. This geometry is often used to construct the capacitance look-up table necessary for efficient full chip extraction. The nominal dimensions of the conductors are 42nm width and 31nm height. The nominal separation between the conductors is 42nm. Each conductor is discretized in 30 segments, resulting in a total of N=481 unknowns (including one unknown potential since this is a 2D problem). The width of each conductor is assumed an independent Gaussian random variable with standard deviation 15.75nm, resulting in $N_P = 16$

We first compare our new Galerkin Krylov Recycling (GKR) from Section 3.1 with the standard GCR and GMRES Krylov subspace recycling (Appendix A.3). To allow for maximum reusage of the subspace, we sample the parameter space using a second order sparse grid Smolyak constructions, resulting in a total of 561 grid points. We compare the details of the different simulations in Table 3.1.

Table 3.1: Table Comparing the Performance of Different Recycling Algorithms

Algorithm	Property	Time [sec]	# iterations	# MV-products
GCR	no recycling	436	14102	14102
our GKR	MF	162	129	62010
our GKR	mF	198	152	67741
our GKR	MC	174	133	62742
our GKR	mC	149	160	56073
GCR recycling	MF	195	124	61059
GCR recycling	mF	204	154	62355
GCR recycling	MC	204	158	56815
GCR recycling	mC	194	130	62207
GMRES recycling	MF	800	388	197096
GMRES recycling	mF	230	203	93204
GMRES recycling	MC	253	214	93905
GMRES recycling	mC	190	207	76904

The four different combinations of residual choice and concurrent granularity are given the indices M,m,f,c, for max, min, fine grain, and coarse grain respectively (see Algorithm 2 in Section 3.1.2). In all cases we observe that the coarse grain minimum residual implementation is the most efficient algorithm within the non-intrusive class. This is due to the fact the in our implementation the delta matrices are not sparse enough and therefore algorithms that will result in fast system elimination perform better than those that do not. In addition, we observe that both our Galerkin Krylov Recycling (GKR) and the standard GCR recycling perform better than the GMRES recycling. Since we have a large number of similar linear system, the overhead encountered in GMRES recycling to minimize the residual of a particular system is unnecessary. Instead it suffices to ensure orthogonality of the residuals with respect to the explored subspace, given that the algorithm will undoubtedly explore a larger subspace.

Next we use our SMOR algorithm on the same example. We assume that a second order stochastic expansion can accurately represent the variations in both the system matrix and the vector x. We use a total number of 561 quadrature points to construct the basis. Such quadrature points are placed according to a second order sparse grid Smolyak construction. When using our proximity measure, to avoid solving systems that do not add sufficiently

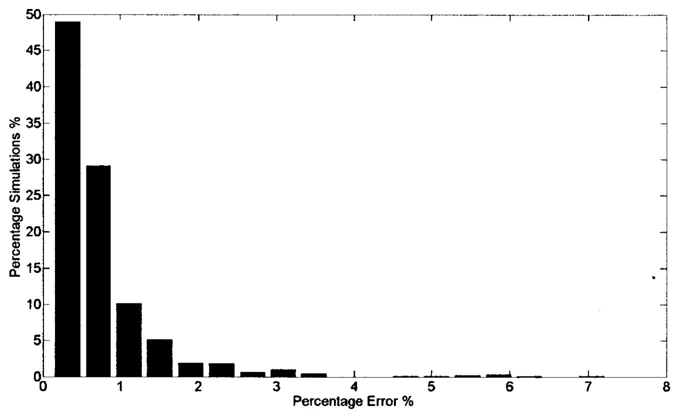


Figure 3-4: Histogram of the error of the proposed model at 1000 randomly generated points in the parameter space for the small grid example

to the column span of the projection matrix, our approach only generated a total of only 32 basis.

The total time required by our approach is 198 seconds as opposed to 436 seconds required by the standard SCM, or the 149 seconds required by the fastest recycling algorithm. Consequently, for this particularly small example our approach is 2.2$\times$ faster than the SCM and 1.3$\times$ slower than GKR (mC) algorithm. We tested the accuracy of our reduced model by computing the solutions at 1000 randomly generated points, and comparing them to the full system. Figure 3-4 shows a histogram of the resulting error. We observe that the error is less than 3% for 98% of the simulated structures. We then used our reduced model to compute the second order statistical moments and all second order Hermite coefficients of the projection of the output capacitances, and observed a 1% match compared to the SCM.

3.3.2 Large On-Chip Example using SMOR

This is an example of 100 conductors arranged on a 10$\times$10 grid. Each conductor is of size 45nm $\times$ 30nm. Each conductor is discretized using a total of 50 unknowns resulting in a total of 5000 unknowns. The width of each conductor is assumed to be an independent Gaussian random variable with variance of 20% the nominal width. The total number of random variables is 100. Using a second order sparse grid construction would require more than 20,000 simulations. Instead we choose to use a total of 5000 Monte Carlo simulations to build the basis and another 5000 simulations to test the accuracy of the

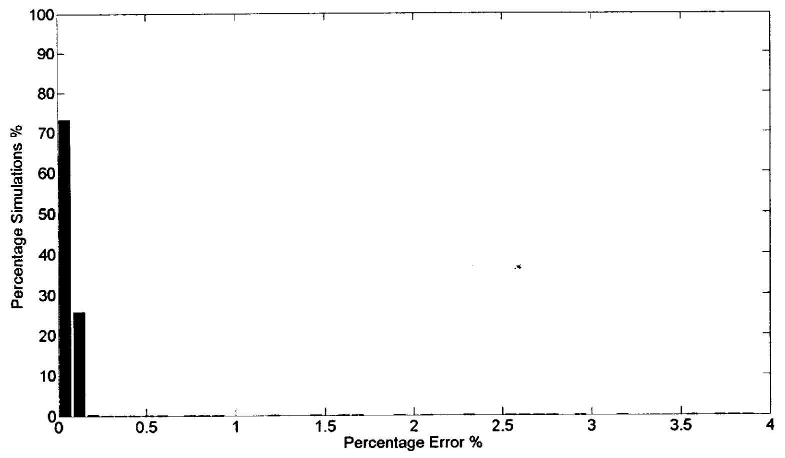


Figure 3-5: Histogram of the error of the proposed model at 5000 randomly generated points in the parameter space for the large grid example

model. We observe that only 112 basis are required to achieve an accuracy of better than 2% in the residual estimation for 99.5% of the cases under study (see Figure 3-5). This means that our algorithm is about 130 times faster than the sparse grid technique and at least 65 times faster than Monte Carlo method. The statistical moments up to first order and all coefficients of the projection of the output capacitances on the space of first order Hermite polynomials, computed using our approach and the standard collocation method match up to 1%. Accuracy of higher order statistics could not be validated, since it is not possible to run a stochastic collocation method on this problem to be used as a reference.

3.3.3 Off-Chip Surface Roughness using SMOR

This example is a parallel place capacitor, where the upper surface is characterized by a rough surface (see Figure 3-6). The rough surface is described by a Gaussian PDF of $\sigma = 1$, and correlation length $L = 1$. The plate surfaces are $20L \times 20L$. The distance between the plates is $5L$. The total number of unknowns discretizing both plates is 21,000. The total number of independent random variables describing the stochastic surface is 323. The total number of orthogonal polynomials required for a second order description of the stochastic capacitance is 52,650. The total number of sparse grid nodes, i.e. independent solves, required for a second order accurate expansion is more than 200,000. This number is even larger than $N = 21,000$. We therefore sample the parameter space at 1,500 points

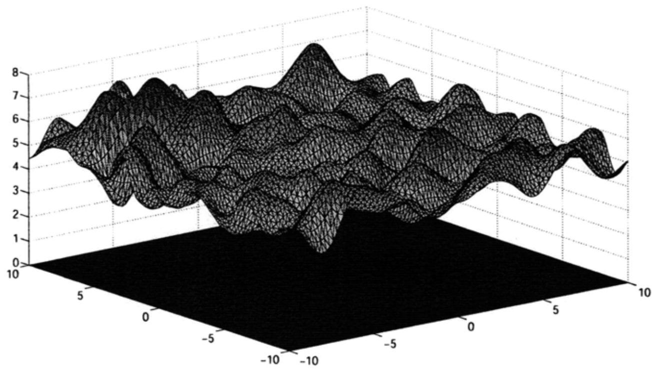


Figure 3-6: Large off-chip parallel plate capacitor with rough surface

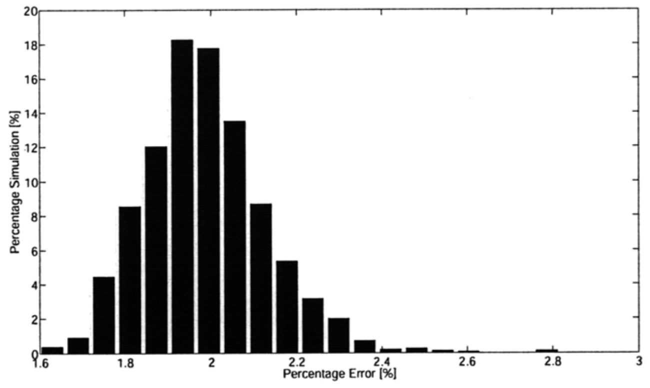


Figure 3-7: Histogram of the error of the proposed model at 1500 randomly generated points in the parameter space

using the Monte Carlo sampling method. The size of the resulting projection matrix is 997. This projection matrix is then used to compute the solution of a different set of 1,500 instantiations of the structure. Each instantiation is solved using fastcap [59]. The results computed using the reduced model and using the exact model are compared in Figure 3-7. We observe that 99.9% of the cases exhibit error less than 5%. The probability density function of the mutual capacitance between both plates is shown in Figure 3-8. The non-Gaussian nature of the curve is clearly visible and the deviation between maximum and minimum capacitances is about 10% from the mean. In terms of computational efficiency, our algorithm is two orders of magnitude more efficient than the stochastic collocation method.

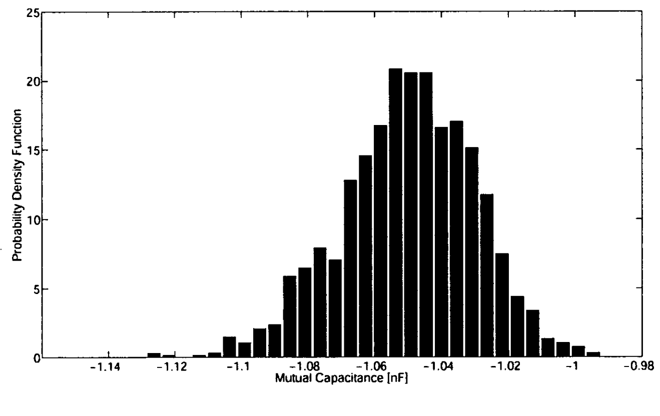


Figure 3-8: Probability density function of mutual capacitance, mean capacitance is approximately 40% larger than that of the parallel plate capacitor without rough surface

Chapter 4

Contributions to Intrusive Stochastic Methods

In this chapter we present advancements and enhancements to the available intrusive stochastic methods and associated mathematical machinery in order to make them computationally more efficient.

1. *Modified Inner Product*: We introduce a modified inner product to efficiently compute the coefficients of any polynomial chaos expansion (in particular of Hermite expansions) without computing any large dimensional integrals. Our algorithm is exact in the sense that it computes the exact coefficient as the standard Hermite expansion. The computational efficiency stems from exploiting the fact that a single matrix element depends on a very small number of the physical parameter.

2. *Combined Neumann Hermite expansion (CNHE)*: a new intrusive method, presented in Section 4.2, for strongly correlated problems in which the Hermite expansion is combined with the Neumann expansion to compute efficiently the complete statistical characteristics of the unknown vector. This algorithm is particularly suitable for impedance extraction of off-chip rough surface interconnects.

3. *Fast Stochastic Galerkin Method (FSGM)*: an efficient and accelerated approach, presented in Section 4.3, for solving the large linear system produced by the Stochastic Galerkin Method when the physical parameter space is weakly correlated. Our

accelerated algorithm uses the precorrected FFT algorithm and it exploits the fact that in weakly correlated problems, a single matrix element typically depends on a small number of the varying parameters. This algorithm is particularly suitable for impedance extraction of on-chip interconnects in the presence of width and height variations.

Notational Note: Throughout this chapter we will refer to a vector that includes only a subset of the elements of another vector using an underline symbol. For instance, $\underline{\mathbf{p}}$ a vector of length D is related to $\mathbf{p}$ a vector of length N_p in the following manner: $D \leq N_p$ and all elements in $\underline{\mathbf{p}}$ are also in $\mathbf{p}$.

4.1 Modified Hermite Chaos Expansion

Let us introduce a new inner product to overcome the main computational difficulty associated with applying a polynomial chaos expansion, namely the need to calculate integrals of very high dimensionality (2.26). Our inner product applies to *any* probability density function of the parameters (e.g. Gaussian, Uniform, Beta) and the corresponding polynomial chaos expansion (e.g. Hermite, Laguerre, Jacobi). For simplicity of exposition we will assume Gaussian distributed parameters with the associated Hermite polynomials.

Definition 4.1.1. *Let our modified inner product $\langle \cdot, \cdot \rangle_{\mathcal{W}}$ be defined by*

$$\langle f(\underline{\mathbf{p}}), \Psi_i(\underline{\vec{\eta}}) \rangle_{\mathcal{W}} = \int\limits_{\underline{\mathbf{p}}} \int\limits_{\underline{\vec{\eta}}} f(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) \mathcal{W}(\underline{\mathbf{p}}, \underline{\vec{\eta}}) d\underline{\mathbf{p}} d\underline{\vec{\eta}}, \tag{4.1}$$

where

$$\mathcal{W}(\underline{\mathbf{p}}, \underline{\vec{\eta}}) = \frac{\exp\left(-0.5 \mathbf{v}^T \mathbf{C}_V^{-1} \mathbf{v}\right)}{(2\pi)^{\frac{D+N_O}{2}} \sqrt{|\mathbf{C}_V|}},$$

and $\mathbf{C}_V$ is the correlation matrix between $\underline{\mathbf{p}}$ and $\underline{\vec{\eta}}$ and

$$\mathbf{v} = \begin{pmatrix} \underline{\mathbf{p}} \\ \underline{\vec{\eta}} \end{pmatrix}, \quad \mathbf{C}_V = \mathbb{E}[\mathbf{v}\mathbf{v}^T] = \begin{pmatrix} \mathbb{E}[\underline{\mathbf{p}}\, \underline{\mathbf{p}}^T] & \mathbb{E}[\underline{\mathbf{p}}\, \underline{\vec{\eta}}^T] \\ \mathbb{E}[\underline{\vec{\eta}}\, \underline{\mathbf{p}}^T] & \mathbb{E}[\underline{\vec{\eta}}\, \underline{\vec{\eta}}^T] \end{pmatrix}$$

where $\mathbb{E}$ is the expectation operator, $\underline{\eta}$ is an at most N_O element subvector of $\vec{\eta}$ upon which the N_O^{th} order Hermite polynomial $\Psi_{N_O}(\underline{\vec{\eta}}) = \Psi_{N_O}(\vec{\eta})$ depends, and $|\mathbf{C}_V|$ is the determinant of the correlation matrix $\mathbf{C}_V$.

Theorem 4.1.1. *The Hermite polynomials $\Psi(\vec{\eta})$ are orthogonal with respect to our modified inner product (4.1).*

Proof. The complete proof is in appendix B.2. □

Theorem 4.1.2. *The coefficients f_i of the standard Hermite expansion of the function $f(\underline{\mathbf{p}})$ can be calculated using our modified inner product:*

$$f_i = \langle f(\underline{\mathbf{p}}), \Psi_i(\underline{\vec{\eta}}) \rangle = \langle f(\underline{\mathbf{p}}), \Psi_i(\underline{\vec{\eta}}) \rangle_W \tag{4.2}$$

Proof. The complete proof is in appendix B.3. □

Theorem 4.1.3. *The maximum dimension of the integrals in (4.2) required to obtain the Hermite expansion using our modified inner product is $D + N_O$, where D is the length of vector $\underline{\mathbf{p}}$ and N_O is the order of the Hermite polynomial.*

In other words, the dimension of the integral in (4.2) is independent of the number of orthogonal random variables N_M used for expanding the random process.

Proof. The proof follows directly from the application of Theorem 4.1.2, however, it remains to be shown that the correlation matrix $\mathbb{E}[\underline{\mathbf{p}}\underline{\eta}^T]$ is known. The calculation of this correlation matrix depends on the particular technique used to generate the set of basis random variables $\vec{\eta}$. For instance, for any Karhunen Loeve based expansion (KL-expansion), we can write component vector $\underline{\mathbf{p}} = \tilde{\mathbf{I}}_1\mathbf{p}$, where $\tilde{\mathbf{I}}_1$ is a rectangular identity matrix to choose the appropriate elements of the vector $\mathbf{p}$. Using (2.21):

$$\begin{aligned}
\mathbb{E}[\underline{\mathbf{p}\vec{\eta}}] &= \tilde{\mathbf{I}}_1\mathcal{M}\sqrt{\Lambda}\mathbb{E}[\vec{\eta}\underline{\eta}^T] \\
&= \tilde{\mathbf{I}}_1\mathcal{M}\sqrt{\Lambda}\mathbb{E}[\vec{\eta}\vec{\eta}^T]\tilde{\mathbf{I}}_2^T \\
&= \tilde{\mathbf{I}}_1\mathcal{M}\sqrt{\Lambda}\tilde{\mathbf{I}}_2^T
\end{aligned}$$

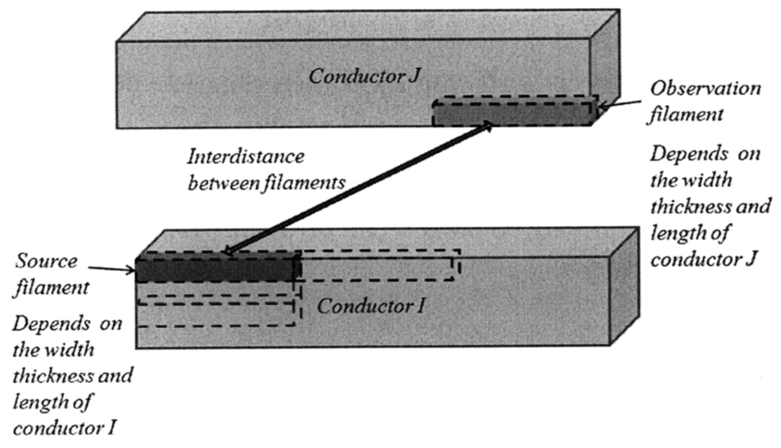

Conductor J
Observation filament
Depends on the width thickness and length of conductor J
Interdistance between filaments
Source filament
Depends on the width thickness and length of conductor I
Conductor I

Figure 4-1: A matrix element represents the mutual interaction between discretization elements. Consequently, it depends on a small number of geometrical variables.

where $\underline{\vec{\eta}} = \tilde{\mathbf{I}}_2\vec{\eta}$ and $\tilde{\mathbf{I}}_2$ is a rectangular identity matrix to choose the appropriate elements of the the vector $\vec{\eta}$. □

Using our modified inner product, we can easily calculate the expansion coefficient $\mathbf{A}_i$ in (2.24):

$$\mathbf{A}_i = \left\langle \mathbf{A}(\underline{\mathbf{p}}), \Psi_i(\underline{\eta}) \right\rangle_{\mathcal{W}} = \int\limits_{\underline{\mathbf{p}}} \int\limits_{\underline{\vec{\eta}}} \mathbf{A}(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) \mathcal{W}(\underline{\mathbf{p}}, \underline{\vec{\eta}}) d\underline{\mathbf{p}} d\underline{\vec{\eta}} \tag{4.3}$$

The main computational advantage of using our modified inner product in Definition 4.1.1 to compute the coefficients of the Hermite chaos expansion stems from the fact that N_M, the total length of the vector $\vec{\eta}$, can be easily on the order of 100, whereas D, the length of the vector $\underline{\mathbf{p}}$, is at most 7 (three components for the source panel/filament dimensions and three for the test panel/filament dimensions and one for the separation between the panels/filaments, see Figure 4-1), and N_O, the order of the expansion, is typically 2 (since a second order expansion is sufficient for most applications). Consequently, the dimension of the integral for a second order polynomial expansion is reduced from 100 to 9. In terms of required computation time, our modified inner product accelerates the computation of the coefficients as compared to the standard tensor rule of a q-point quadrature scheme by a factor of q^{91} (for q=8 this corresponds to a speedup of 82 orders of magnitude), and as compared to Monte Carlo integration or sparse grid integrations by a factor of $\frac{(N_M)^q}{q^9}$ (for N_M=100, q=8 this corresponds to a speedup of a factor 10^7).

4.2 Combined Neumann and Hermite Expansion Method

In this section we will develop a new intrusive method which exploits the best features of both the stochastic Galerkin method (SGM, Section 2.5) and the Neumann Expansion Method (Section 2.4). The approach in this section is ideal for impedance extraction in the presence of surface roughness.

We propose a combined approach where we use the Neumann expansion (2.30) to avoid the complexity of solving a large linear system, and we use the modified polynomial chaos expansion to simplify the calculation of the statistics of the RHS vector. This combined approach is implemented by first expanding $\mathbf{A}(\mathbf{p})$ in terms of Hermite polynomials using the modified inner product (4.1)

$$\mathbf{A}(\mathbf{p}) = \mathbf{A}_0 + \sum_{i=1}^{K} \mathbf{A}_i \Psi_i(\vec{\eta}) \tag{4.4}$$

and then substituting this expansion in the Neumann expansion (2.30):

$$\begin{aligned} \mathbf{x}(\vec{\eta}) &= \mathbf{A}_0^{-1}\mathbf{b} - \mathbf{A}_0^{-1}\left(\sum_{i=1}^{K} \mathbf{A}_i \Psi_i(\vec{\eta})\right)\mathbf{A}_0^{-1}\mathbf{b} + && (4.5) \\ &+ \mathbf{A}_0^{-1}\left(\sum_{i=1}^{K} \mathbf{A}_i \Psi_i(\vec{\eta})\right)\mathbf{A}_0^{-1}\left(\sum_{i=1}^{K} \mathbf{A}_i \Psi_i(\vec{\eta})\right)\mathbf{A}_0^{-1}\mathbf{b} + \dots && (4.6) \end{aligned}$$

From (4.6) we observe that the problem has been transformed from solving a huge linear system into doing instead a large number of small matrix-vector multiplications, or small linear system solves. Before proceeding with the time and memory complexity analysis it is worth mentioning that in general we are not interested in the entire unknown vector $\mathbf{x}(\vec{\eta})$, but rather just in some linear combination of its components $y(\vec{\eta}) = \mathbf{c}^T\mathbf{x}(\vec{\eta})$ related to quantities at the ports of the structure under analysis. For instance, $y(\vec{\eta})$ could represent one or more port currents for admittance extraction, or port charge for capacitance extraction. For a single port, equation (4.6) is simplified to:

$$y(\vec{\eta}) \simeq y_0 - \sum_{i=1}^{K} \mathbf{z}_0^T \mathbf{u}_i \Psi_i(\vec{\eta}) + \sum_{i=1}^{K}\sum_{j=1}^{K} \mathbf{v}_i^T \mathbf{t}_j \Psi_i(\vec{\eta})\Psi_j(\eta), \tag{4.7}$$

where $y_0 = \mathbf{c}^T\mathbf{x}_0$, $\mathbf{x}_0 = \mathbf{A}_0^{-1}\mathbf{b}$, $\mathbf{z}_0 = \mathbf{A}_0^{-T}\mathbf{c}$, $\mathbf{u}_i = \mathbf{A}_i\mathbf{x}_0$, $\mathbf{v}_i = \mathbf{A}_i^T\mathbf{z}_0$, and $\mathbf{t}_j = \mathbf{A}_0^{-1}\mathbf{u}_j$. Notice that the complexity of evaluating (4.7) is significantly less than that of evaluating (4.6) due to the use of the adjoint equations to compute $\mathbf{z}_0$ and $\mathbf{v}_i$.

Algorithm 5 Combined Neumann Hermite Expansion (CNHE)

```
1: Compute the coefficient matrices $\mathbf{A}_i$ using (4.3) as in Theorem 4.1.2.
2: Solve the nominal $\mathbf{A}_0\mathbf{x}_0 = \mathbf{b}$ and adjoint problems $\mathbf{A}_0^T\mathbf{z}_0 = \mathbf{c}$ for $\mathbf{x}_0$ and $\mathbf{z}_0$, respectively.
3: for each coefficient matrix $\mathbf{A}_i$ do
4:    $\mathbf{u}_i \leftarrow \mathbf{A}_i\mathbf{x}_0$
5:    $\mathbf{v}_i \leftarrow \mathbf{A}_i^T\mathbf{z}_0$
6:    Solve $\mathbf{A}_0\mathbf{t}_j = \mathbf{u}_j$ for $\mathbf{t}_j$
7: end for
8: Use (4.7) to assemble the second order expansion for the required output $y(\eta)$ (for instance the input current).
```

4.2.1 CNHE Complexity Analysis

Memory

Our algorithm requires computing the matrices $\mathbf{A}_0$ and $\mathbf{A}_i$, which might wrongly indicate that the total memory requirement is $O(N^2) + O(KN^2)$. However, as obvious from Algorithm 5, all the computations involve only matrices of size $N \times N$. Consequently, our algorithm has the same memory complexity $O(N^2)$ as sampling based approaches. Furthermore, our algorithm is very memory efficient if compared to the $O(N^4)$ complexity of the standard Neumann expansion, or the $O(K^2N^2)$ complexity of the standard stochastic Galerkin method.

Time

We will base our analysis on using iterative methods. We solve a total of K unique linear systems for $O(K\rho(N))$, where $\rho(N)$ is the complexity of solving one system. In addition, we compute K matrix-vector products $O(KN^2)$ and $K^{1.5}$ vector-vector products $O(NK^{1.5})$. The final complexity is therefore $O(KN^2)$. This means that our algorithm is very time efficient if compared to the $O(N^4)$ complexity of the standard Neumann expansion, the $O(N^2K^{1.5})$ complexity of the standard stochastic Galerkin simulation, or the

$O(N_S N^2)$ complexity of the Monte-Carlo like algorithms, where N_S is the number of independent solves.

Parallelizability

One of the main advantages of the proposed algorithm is its inherent parallelizability as is evident from (4.7). Since there is practically no communication required between the different processors, the computational time required to calculate the different matrix-vector and vector-vector products can be reduced by N_{Proc}, the number of available processors.

4.3 "Fast" Stochastic Galerkin Method (FSGM)

In this section we propose an efficient algorithm to solve the large Stochastic Galerkin Method linear system (see (2.37) in Section 2.5). The proposed approach is an alternative to the CNHE approach for problems in which the Hermite expansion coefficient matrices $\{\mathbf{A}_i : i > 0\}$ are sparse. Such problems include parasitic extraction of on-chip interconnect structures in the presence of width and thickness process variations.

Throughout this section we will use the same notation as that introduced in the Section 2.5. However, we will sometimes refer to the set of orthogonal polynomials $\{\Psi_k(\vec{\eta}) : k = 0, \cdots, K\}$ using a multiple indexing scheme. For instance, we refer to the second order polynomials using the double index $\Psi_{ij}(\vec{\eta})$, with the understanding that the mapping between k and the pair (i, j) is 1-to-1. Consequently,

$$\Psi_{00}(\vec{\eta}) = 1 \quad \Psi_{i0}(\vec{\eta}) = \eta_i \quad \Psi_{ii}(\vec{\eta}) = \eta_i^2 - 1 \quad \Psi_{ij}(\vec{\eta}) = \eta_i \eta_j \tag{4.8}$$

Recall that one of the main problems associated with solving the linear system (2.37) using direct methods (e.g. Gaussian elimination) is the need to store a very large matrix of size $KN \times KN$. Solving such system with direct methods is therefore almost impossible for any practical value of K and N. On the other hand, if iterative methods (such as GMRES) are used, then only the product of $\tilde{\mathbf{A}}\tilde{\mathbf{x}}$ needs to be computed. Instead of first constructing the entire $\tilde{\mathbf{A}}$ and then computing its product with $\tilde{\mathbf{x}}$, we compute the matrix-vector product using the expression

$$\mathbf{v}(\mathbf{p}) = \mathbf{A}(\mathbf{p})\mathbf{x}(\mathbf{p}) = \sum_{i=0}^{K}\sum_{j=0}^{K} \mathbf{A}_i \mathbf{x}_i \Psi_i(\vec{\eta})\Psi_j(\vec{\eta}). \tag{4.9}$$

Notice that because (2.37) is constructed using a Galerkin projection, we only care about terms in (4.9) that reside in the space of multivariate Hermite polynomials spanned by the testing polynomials. To obtain such terms the product $\Psi_i(\vec{\eta})\Psi_j(\vec{\eta})$ is expanded in terms of Hermite polynomials. Table 4.1 summarizes how the product of any two second order Hermite polynomials is expanded in terms of other Hermite polynomials. Note that in Table 4.1 all high order polynomials (those that have more than 2 indices) do not contribute to the second order expansion. Using Table 4.1, the matrix-vector product can be re-written as follows

$$\mathbf{v}(\mathbf{p}) = \sum_{m=0}^{K} \mathbf{v}_m \Psi_m(\vec{\eta}), \tag{4.10}$$

where

$$\mathbf{v}_m = \sum_{m:\langle \Psi_i\Psi_j, \Psi_m\rangle > 0} \mathbf{A}_i \mathbf{x}_j \left\langle \Psi_i(\vec{\eta})\Psi_j(\vec{\eta}), \Psi_m(\vec{\eta})\right\rangle \tag{4.11}$$

Algorithm 6 Computation of the Matrix Vector Products $\mathbf{v}_m$ in (4.11)

```
1: $\mathbf{v_m} \leftarrow 0$
2: **for** $i = 0 : K$ **do**
3:   **for** $j = 0 : K$ **do**
4:     **if** $\Psi_i\Psi_j$ has a component in the second order space (see Table 4.1) **then**
5:       $\mathbf{u} \leftarrow \mathbf{A}_i\mathbf{x}_j$
6:       **for all** $m$ such that $\langle \Psi_i\Psi_j, \Psi_m\rangle \neq 0$ **do**
7:         $\mathbf{v}_m \leftarrow \mathbf{v}_m + \mathbf{u}$
8:       **end for**
9:     **end if**
10:  **end for**
11: **end for**
```

Algorithm 6 represents an efficient implementation of (4.10). Several observations contribute to the computational efficiency of our algorithm. From a memory point of view, since we never assemble $\tilde{\mathbf{A}}$, we only need to store the K matrices $\mathbf{A}_i$. Consequently, our algorithm requires only $O(KN^2)$ storage. From a computational point of view, our algorithm exploits the fact that the matrix $\tilde{\mathbf{A}}$ is sparse, and given its Kronecker product based

Table 4.1: Expansion of product of multivariate Hermite polynomials in terms of Hermite polynomials. Ψ_{xx} stands for any second order Hermite polynomial.

Ψ_i	Ψ_j	$\Psi_i \Psi_j$	Second order Ψ_m, s. t. $\langle \Psi_i \Psi_j, \Psi_m \rangle > 0$
Ψ_{00}	Ψ_{xx}	Ψ_{xx}	Ψ_{xx}
$\Psi_{k_1 0}$	$\Psi_{k_2 0}$	$\Psi_{k_1 k_2}$	$\Psi_{k_1 k_2}$
$\Psi_{k_1 0}$	$\Psi_{k_1 0}$	$\sqrt{2}\Psi_{k_1 k_1} + \Psi_{00}$	$\Psi_{k_1 k_1}$, Ψ_{00}
$\Psi_{k_1 0}$	$\Psi_{k_2 k_3}$	$\Psi_{k_1 k_2 k_3}$	0
$\Psi_{k_1 0}$	$\Psi_{k_2 k_2}$	$\Psi_{k_1 k_2 k_2}$	0
$\Psi_{k_1 0}$	$\Psi_{k_1 k_3}$	$\sqrt{2}\Psi_{k_1 k_1 k_3} + \Psi_{k 0}$	$\Psi_{k 0}$
$\Psi_{k_1 0}$	$\Psi_{k_1 k_1}$	$\sqrt{3}\Psi_{k_1 k_1 k_1} + \sqrt{2}\Psi_{k_1 0}$	$\Psi_{k_1 0}$
$\Psi_{k_1 k_1}$	$\Psi_{k_2 k_2}$	$\Psi_{k_1 k_1 k_2 k_2}$	0
$\Psi_{k_1 k_1}$	$\Psi_{k_1 k_1}$	$\sqrt{6}\Psi_{k_1 k_1 k_1 k_1} + 2\sqrt{2}\Psi_{k_1 k_1} + \Psi_{00}$	$\Psi_{k_1 k_1}$, Ψ_{00}
$\Psi_{k_1 k_1}$	$\Psi_{k_2 k_3}$	$\Psi_{k_1 k_1 k_2 k_3}$	0
$\Psi_{k_1 k_1}$	$\Psi_{k_1 k_3}$	$\sqrt{3}\Psi_{k_1 k_1 k_1 k} + \sqrt{2}\Psi_{k_1 k_3}$	$\Psi_{k_1 k_3}$
$\Psi_{k_1 k_2}$	$\Psi_{k_1 k_2}$	$2\Psi_{k_1 k_1 k_2 k_2} + \sqrt{2}\Psi_{k_1 k_1} + \sqrt{2}\Psi_{k_2 k_2} + \Psi_{00}$	$\Psi_{k_1 k_1}$, $\Psi_{k_2 k_2}$, Ψ_{00}
$\Psi_{k_1 k_2}$	$\Psi_{k_1 k_3}$	$\sqrt{2}\Psi_{k_1 k_1 k_2 k_3} + \Psi_{k_2 k_3}$	$\Psi_{k_2 k_3}$
$\Psi_{k_1 k_2}$	$\Psi_{k_3 k_4}$	$\Psi_{k_1 k_2 k_3 k_4}$	0

construction (see (2.37)) it can be assembled using only K distinct submatrices, namely $\mathbf{A}_i$. That fact is exploited in Step 5 of Algorithm 6, where $\mathbf{A}_i \mathbf{x}_j$ is computed only *once* and then added to the relevant components.

4.3.1 Exploiting the Particular Sparse Structure of $\mathbf{A}_i$

The total number of matrix vector products, i.e. the sparsity of the two dimensional matrix in Fig 4-2, is $O(KN_P)$, or $O(K^{1.5})$. This means that the computational complexity of Algorithm 6 is reduced from $O(K^2N^2)$ to $O(K^{1.5}N^2)$. Unfortunately, this is still computationally very expensive. To further reduce the computational complexity of the algorithm we now need to exploit the sparsity of the coefficients matrices $\mathbf{A}_i$.

Recall from Section 2.2 that the random *geometrical parameters* in vector $\mathbf{p}$ (such as wire heights and widths) in the on-chip extraction problem are assumed uncorrelated. Therefore, any matrix element $\mathbf{A}_{ij}(\underline{\mathbf{p}})$ in (2.16) is determined only by the *independent* parameters describing the dimensions of the source and test conductors and the separation between them. Consequently, the number of parameters on which a single matrix ele-

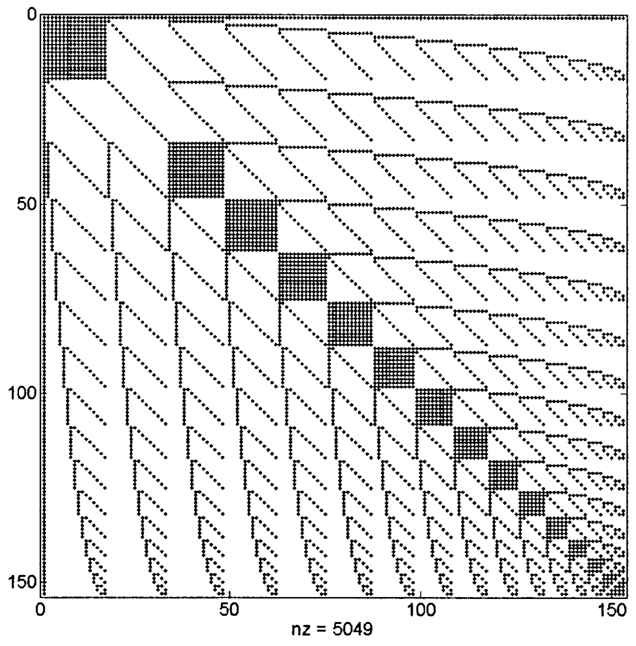


Figure 4-2: Sparsity pattern of the product table. Example of a 16 variable second order expansion, i.e. $N_p = 16$, $N_O = 2$ and $K = 153$. The number of non-zeros elements is $O(K^{1.5})$.

ment depends is very small compared to the total number of parameters. This observation means that the coefficient matrices will be relatively sparse, and therefore the matrix vector product computational effort will be significantly reduced using sparse matrix techniques. Figure 4-3 shows four of the most typical sparsity patterns for both first and second order terms. Examples of geometrical variations producing Figure 4-3 are demonstrated in Figure 4-4. To understand such figures recall that the system matrix of (2.16) is constructed using integral equation methods. Consequently, the sparsity pattern corresponding to each variation in a conductor location or dimensions is a strip of rows and columns. Furthermore, notice that the sparsity pattern of second order variations is generated by intersecting the patterns of first order variations.

Observation 4.3.1. *The sparsity pattern of the coefficient matrices is as follows:*

1. *the average matrix* $\mathbf{A}_0$ *(projection of* $\mathbf{A}(\mathbf{p})$ *on the constant term) is dense and has* $O(N^2)$ *nonzero elements.*

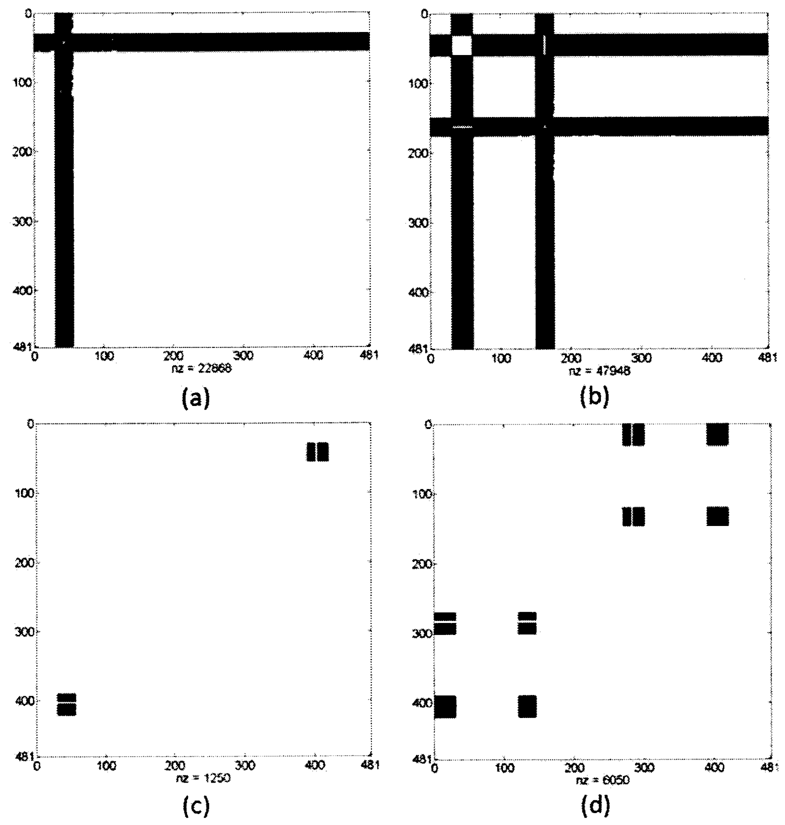


Figure 4-3: Different sparsity pattern of the coefficient matrices. a) single variable (width) associated with a single conductor b) single variable (distance) associated with two conductors c) Two variables, each associated with a single conductor (width, width) d) Two variables, each associated with a two conductors (distance, distance).

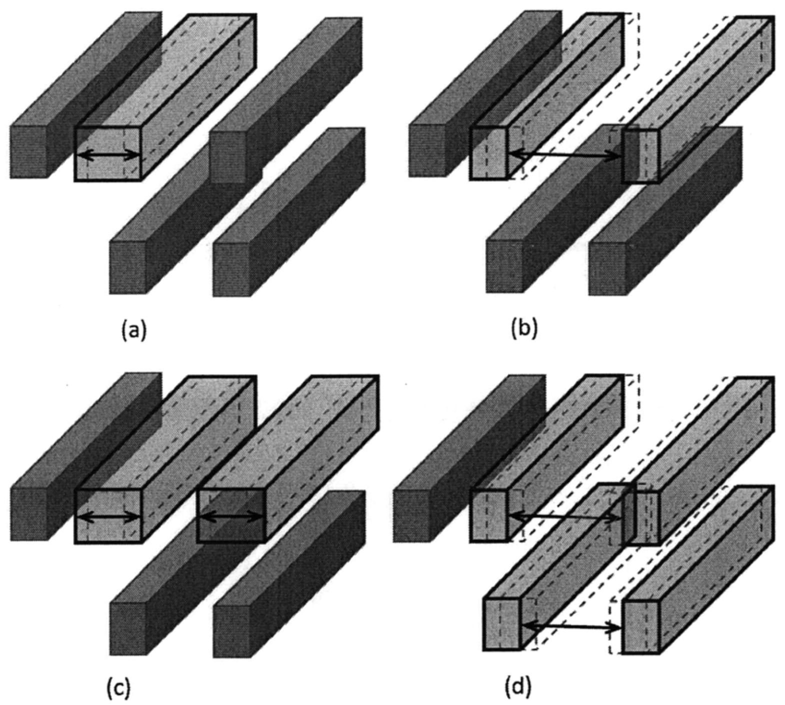


Figure 4-4: Examples of typical variations producing the sparsity patterns in Figure 4-3.

2. *the coefficient matrix resulting from projecting* $\mathbf{A}(\mathbf{p})$ *on first or second order Hermite polynomials involving only one variable has at most* $O(N)$ *nonzero elements.*

3. *the coefficient matrix resulting from projecting* $\mathbf{A}(\mathbf{p})$ *on second order Hermite polynomials involving two variables has at most* $O(1)$ *nonzero elements.*

From the previous observations, it is clear that the only expensive matrix vector product is that involving the constant projection matrix $\mathbf{A}_0$. This matrix is multiplied by all the components of $\mathbf{x}(\mathbf{p})$. For ease of notation we will assemble all the vectors $\mathbf{x}_j$ into a large matrix $\mathbf{X}$, such that $\mathbf{x}_j$ is in column j of $\mathbf{X}$. The cost of the matrix vector product $\mathbf{A}_0\mathbf{X}$ is $O(N^2K)$, the cost of the rest of the matrix vector products is at most $O(K^{1.5}N)$.

4.3.2 Handling Matrix Vector Products involving the Average Matrix

We use the precorrected fast Fourier transform (pFFT) to accelerate the matrix vector product $\mathbf{A}_0\mathbf{X}$. Since $\mathbf{A}_0$ is the mean of the stochastic system matrix, it has exactly the same structure of the deterministic system. Consequently, we can use our available pFFT implementation [23] to compute the matrix vector product in asymptotic complexity $O(KNlogN)$.

This means that the final complexity of our entire algorithm is just $O(K\,N\ \log\,N)$. Our complete FSGM is summarized in Algorithm 7.

Algorithm 7 FSGM (embedded in GMRES) to solve (2.37)

```
 1: compute $\mathbf{A}_i$ for expansion (2.24) using (4.3)
 2: $\tilde{\mathbf{q}}_0 \leftarrow \tilde{\mathbf{b}}$ (from (2.37))
 3: $n \leftarrow 0$
 4: $||\mathbf{r}|| \leftarrow ||\tilde{\mathbf{q}}_0||$
 5: **while** $||\mathbf{r}|| >$ threshold **do**
 6:     $\mathbf{Q}_n \leftarrow$ reshape $\tilde{\mathbf{q}}_n$ from $KN \times 1$ to $N \times K$
 7:     get $\mathbf{v}_\mathrm{m}$ from Alg. 6 starting from $i = 1$ instead of $i = 0$ and using $\mathbf{x}_j \leftarrow \mathbf{Q}_n(:, j)$
 8:     use pfft to compute $\mathbf{V} \leftarrow \mathbf{A}_0\mathbf{Q}_n$.
 9:     $\mathbf{v}_\mathrm{m} \leftarrow \mathbf{v}_\mathrm{m} + \mathbf{V}(:, m)$
10:     assemble $\tilde{\mathbf{v}}_{n+1}$ by stacking all $\mathbf{v}_\mathrm{m}$ in a vector
11:     use $\tilde{\mathbf{v}}_{n+1}$ to compute $\tilde{\mathbf{q}}_{n+1}$ and $||\mathbf{r}||$ using the standard GMRES step
12:     $n \leftarrow n + 1$
13: **end while**
```

4.4 Results

4.4.1 Validation of the Second Order Hermite Expansion

In this example we verify the accuracy of a second order multivariate Hermite polynomial. In Figure 4-5 we show the accuracy of expanding the matrix elements corresponding to a 2D capacitance extraction formulation in the presence of width and thickness variations. The total number of random parameters is 100. We show the values of a complete matrix row (i.e. the interaction of one element with all the other elements in the structure) computed using the analytical integrations versus its approximation using the expansion at a randomly generated sample of the 100-parameter vector. The maximum percentage error over all values is less than 0.075%.

In Figure 4-6 we show the accuracy of expanding the matrix elements corresponding to a 3D capacitance extraction formulation in the presence of width and thickness variations. The total number of random parameters is 295. We show the values of a complete matrix row (i.e. the interaction of one element with all the other elements in the structure) computed using the analytical integrations versus its approximation using the expansion

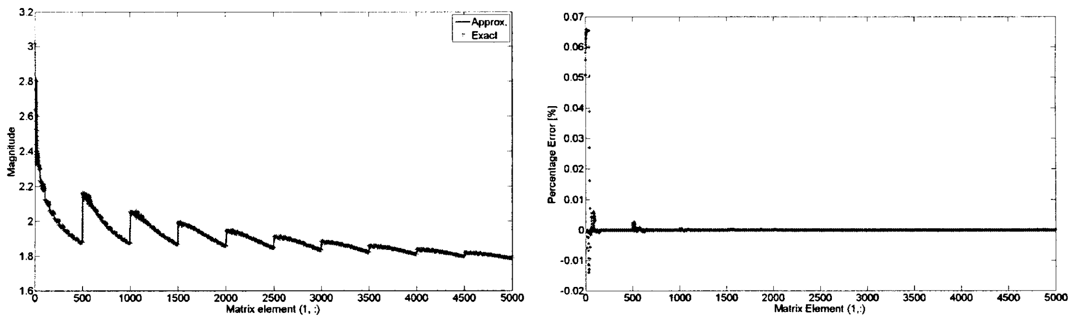


Figure 4-5: Hermite polynomials expansion of the matrix elements of a 2D problem with width/height variations. Upper left is a comparison of the values of a row of the system matrix computed using the exact integration routines versus our expansion for randomly generated values of the 100-parameters. Upper right figure is the percentage error.

at a randomly generated sample of the 295-parameter vector. In Figure 4-6 we show the histogram of the maximum percentage error (over all elements in a row of the matrix) for 1000 randomly generated points in the 295-dimensional parameter space. The maximum error is less than 0.5% for all the values.

In Figure 4-7 we show the accuracy of expanding the matrix elements corresponding to a 3D capacitance extraction formulation in the presence of surface roughness. The total number of random parameters is 201. We show the dominant (largest in magnitude) values of a matrix row (i.e. the interaction of one element with the other elements in the structure) computed using the analytical integrations versus its approximation using the expansion at randomly generated values of the parameters. The maximum error is less than 2% over all values.

4.4.2 CNHE Accuracy Validation on Small Highly Correlated Problems (Surface Roughness)

Using two relatively smaller examples, in this section we verify the accuracy of our Combined Neumann Hermite Expansion (CNHE) algorithm proposed in Section 4.2. The examples had to be chosen small enough to be accurately discretized and simulated using 10,000 very accurate Monte Carlo simulations as a golden reference.

The first example is a single 50μm wide, 0.5mm long, and 15μm thick microstrip line.

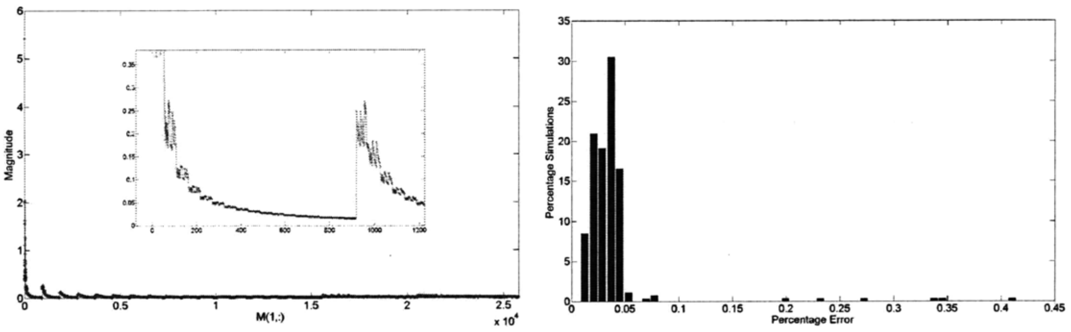


Figure 4-6: Hermite polynomials expansion of the matrix elements of a 3D problem with width/height variations. Left figure is a comparison of the values of a row of the system matrix computed using the exact integration routines versus our expansion for randomly generated values of the parameters. Right figure is a histogram of the percentage error at a 1000 randomly generated samples of the 295-dimension parameter vector.

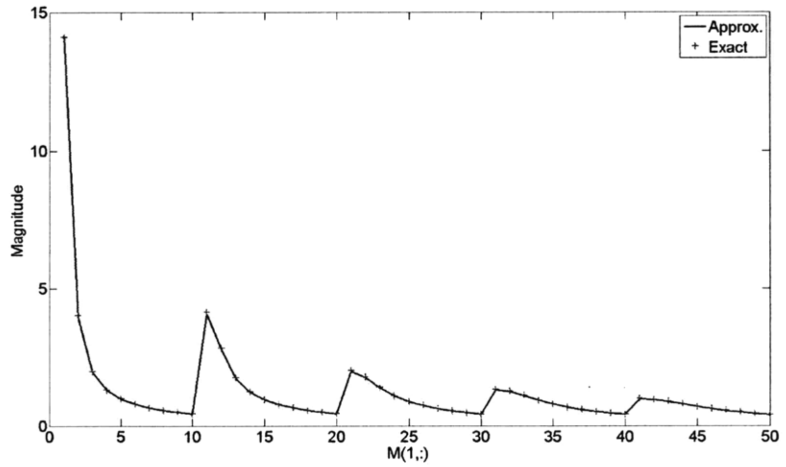


Figure 4-7: Comparison of the values of a row of the system matrix computed using the exact integration routines versus our expansion for randomly generated values of the 201-dimensional parameter space. Maximum expansion error is less than 2%.

The upper surface of the line is highly irregular and is described with a Gaussian random process of standard deviation $\sigma = 3\mu\text{m}$, and correlation length $L_c = 50\mu\text{m}$. We use a total of 200 unknowns to model the current density inside of the microstrip line, and a total of 19 independent random variables to model the rough surface. The number of orthogonal polynomials for a second order ($N_O = 2$), 19-variables ($N_M = 19$) Hermite expansion of the system matrix is computed from (2.25), and is equal to $K = 210$. A comparison of the complete statistical distributions of the DC input resistance of the microstrip line, as obtained from our CNHE algorithm, from Monte Carlo analysis, and from the standard SGM method with our modified inner product 4.1 is shown in Figure 4-8. We observe very good agreement between all techniques. The mean of all three distributions is identically equal to 0.0122. The standard deviation for both Monte Carlo and SGM is identically equal to 0.001, while our CNHE computes the standard deviation 0.00097, which corresponds to a 3% error.

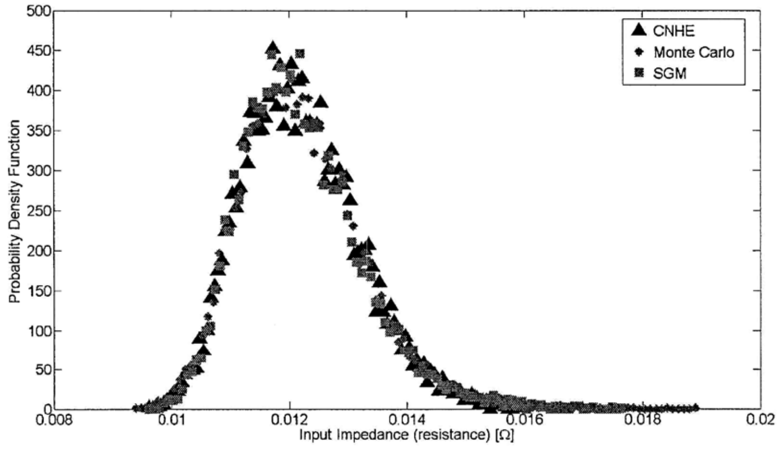


Figure 4-8: Comparison between the probability density function of the microstrip line obtained from the reference Monte Carlo, SGM improved by Theorem 4.1.2 to setup the matrix, and our complete new algorithm (CNHE).

The second example is a similar, but longer (1mm) microstrip line, with roughness described by a different correlation length $L_c = 25\mu\text{m}$. The total number of random variables required to model such a rough surface is 96. Although still very simple, this structure is already too complex for the standard SGM method which would require a matrix size $10^6 \times 10^6$, i.e. more than 10^{12}GB of memory. Note that, since in this problem the geometrical parameters (describing the surface roughness) are highly correlated, our FSGM algorithm in Section 4.3 cannot be employed to reduce the memory and speed requirements of the standard SGM. Using our new CNHE technique, we instead encountered no difficulties computing efficiently and accurately for instance the statistics of the input resistance of such an interconnect structure. The average input resistance obtained from our CNHE algorithm is 0.0243Ω as compared to 0.0241Ω obtained from reference (10,000 Monte Carlo simulations). The standard deviation is 8.64×10^{-4} as compared to 8.73×10^{-4} from the Monte Carlo reference, demonstrating less than 1% error. Such results verify the accuracy of our CNHE. The time and memory performance on both microstrip lines examples are instead included in the comparison table in Section 4.4.4.

4.4.3 CNHE on a Large Example with Surface Roughness

In this example we demonstrate how our CNHE algorithm can be used to extract surface roughness effects on the current distribution inside of a two turn inductor. The inductor has a side length of $1000\mu m$ and cross sectional dimensions of $60\mu m \times 15\mu m$. The frequency band of interest is from $1MHz$ to $1GHz$. The inductor is discretized using $2,750$ non-uniform segments such that the skin depth at the highest frequency $\sqrt{2/\omega\mu\sigma} = 2\mu m$ is accurately captured. The time and memory performance of this large example will be summarized in the following Section 4.4.4. Here we compare instead the results obtained from different Gaussian rough surface realizations, i.e. different standard deviations σ and different correlation lengths L_c. The considered surfaces are characterized by ($\sigma = 3\mu m, L_c = 50\mu m$), ($\sigma = 3\mu m, L_c = 5\mu m$) and a completely smooth and deterministic surface. We have observed that the imaginary part of the input impedance divided by $j\omega$ (inductance) typically decreases by less than 5% as a consequence of the surface roughness. On the other hand, the real part of the input impedance (resistance) increases by about 10% to 20% as a consequence of the roughness. In addition, we have observed that the standard deviation of the impedance distribution is proportional to the correlation length.

In Figure 4-9 the probability density functions of the real part of the input impedance at 1GHz are shown for both small $L_c = 5\mu m$ and large $L_c = 50\mu m$ correlation length, respectively. We observe that the standard deviation of the real part of the input impedance

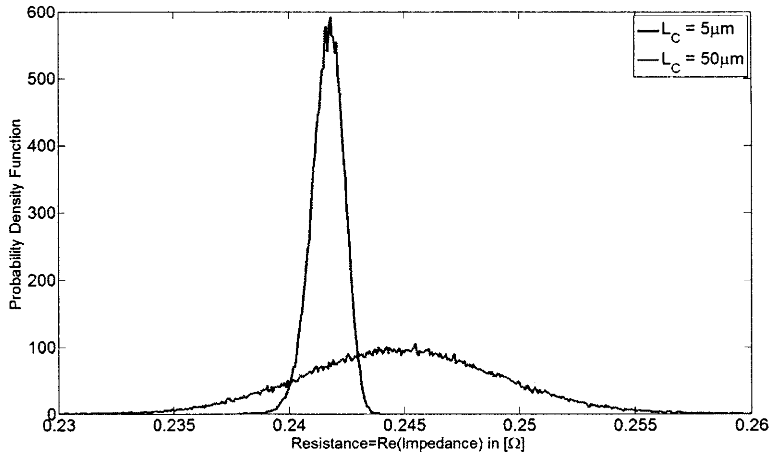


Figure 4-9: Probability density function of the real part of the input impedance at 1GHz for correlation lengths $L_c = 5\mu m$ and $L_c = 50\mu m$. The resistance of the non-rough surface structure is 0.22Ω, which is 9.8% smaller than the mean of the $L_c = 5\mu m$ distribution and 11.3% smaller than the mean of the $L_c = 50\mu m$ distribution.

is increased by a factor of 5 when the correlation length increases from $L_c = 5\mu m$ to $L_c = 50\mu m$ for the same standard deviation. The fact that the standard deviation decreases with the decrease of correlation length (increase of surface randomness) is a consequence of the cancellation effects resulting from the distributed nature of the surface.

4.4.4 CNHE Computational Complexity Comparisons

In this section we compare the computational performance of the non-intrusive Monte Carlo based algorithms, the standard Neumann expansion method, the standard Stochastic Galerkin Method (SGM) and our Combined Hermite-Neumann expansion method (CNHE) on four different interconnect structures, namely,

- the first single microstrip example (200 unknowns, $L_c = 50\mu m$) described in Section 4.4.2,
- the second single microstrip example (400 unknowns, $L_c = 25\mu m$) in Section 4.4.2,
- a two-wire transmission line (800 unknowns, $L_c = 50\mu m$),
- and the 2-turn square inductor (2750 unknowns, $L_c = 50\mu m$) described in Section 4.4.3.

Table 4.2: Time and Memory Performance Comparison of Monte Carlo, Neumann Expansion, Stochastic Galerkin Method (SGM) and the combined Neumann-Hermite expansion (CNHE)

Example	Technique & Properties for 5% accuracy	Memory	Time (MATLAB)
Short Microstrip Line DC only	Monte Carlo, 10,000	0.32 MB	24 min.
	Neumann, $2^{nd} order$	0.32 MB	1 min.
	SGM, N_M=19	58 MB	(12 days)
	SGM+ Thm. 4.1.2, N_M=19	58 MB	120 min.
	our CNHE, N_M=19	0.32 MB	1.8 min.
Long Microstrip Line	Monte Carlo, 10,000	1.2 MB	2.4 hours
	Neumann, 2^{nd} order	1.2 MB	0.25 hours
	SGM, N_M=96	(72 GB)	-
	our CNHE, N_M=96	1.2 MB	0.5 hours
Trans-mission Line 10 freq.	Monte Carlo, 10,000	10 MB	16 hours
	Neumann, 2^{nd} order	10 MB	24 hours
	SGM, N_M=105	(300 TB)	-
	our CNHE, N_M=105	10 MB	7 hours
Two-turn Inductor 10 freq.	Monte Carlo, 10,000	121 MB	(150 hours)$^+$
	Neumann, 2^{nd} order	121 MB	(828 hours)$^+$
	SGM, N_M=400	(800 PB)	-
	our CNHE, N_M=400	121 MB	8 hours $^+$

The upper surface of all the structures is assumed to be rough. Note once again that the FSGM cannot be applied because the geometrical parameters are highly correlated. Notice GKR is less efficient in this highly correlated large problem than standard Monte Carlo analysis. This is due to the fact that GKR requires the difference between the system matrices to be sparse, which can only be achieved if a structured sampling technique is used, for instance (sparse grid sampling). However, for highly correlated large problems, sparse grid sampling requires significantly more sample points than that required by the Monte Carlo method to achieve the same accuracy. This in turn explains the inefficiency of the GKR algorithm for the following large examples.

The comparison results (memory and time requirements) are summarized in Table 4.2. All the simulations have been run in MATLAB on Intel Xeon, CPU 3.4GHz, 4-processor, 4GB RAM. Parameter N_M in Table 4.2 indicates the number of independent random variables used to expand the random process, and it corresponds to the dimension of the integrals required for calculating the coefficients of the Hermite expansion when not employing our modified inner product in 4.1 and Theorem 4.1.2. However, using our modified inner product the dimension of the integral is 4 independent of the number of random variables N_M. The number beside the Monte Carlo label in Table 4.2 indicates the total number of simulation runs. The note "10 freq." in the table indicates results of a 10 point frequency sweep. The frequency band of interest is from 1MHz to 1GHz. Brackets indicate estimated values. The $^+$ superscript indicates running the simulation on the 4 cores in parallel. It can be immediately inferred from the table that our CNHE algorithm is the only practical algorithm for problems of large size.

4.4.5 On-chip Parameter-Variation using FSGM

In this section we use the same 16-conductor 2D on-chip example presented in Section 3.3.1. The objective is to use our "Fast" Stochastic Galerkin Method (FSGM) presented in Section 4.3 to obtain an analytic expansion of the capacitance matrix as a function of the width variations. The capacitance computed using the stochastic model are then compared to those computed using exact simulations. Our target is an error of at most 2% over all the sampled geometries. The total number of terms in a 16-variable second order expansion is K =153. The total time to construct the expansion is approximately 2 hours. This time is not added to the total simulation time since, it can always be done offline, parallelized on several machines and optimized such that it becomes fairly insignificant. The total number of unknowns is KN, i.e. 153×481 = 73,593. The total solution time is 52sec.

Figure 4-10 shows the probability density function of the capacitance between two nearby conductors. It is obvious that the capacitance distribution is approximately Gaussian and that the second order model captures all nonlinearities quite accurately. This is further demonstrated in Fig. 4-11, in which we show the error between the actual capacitance and that computed using the second order stochastic model (resulting from our stochastic simulation) for a total of 10,000 randomly selected sample points in the parameter space. We observe that a total of 97% of test cases exhibit error less than 2% compared to the exact simulations.

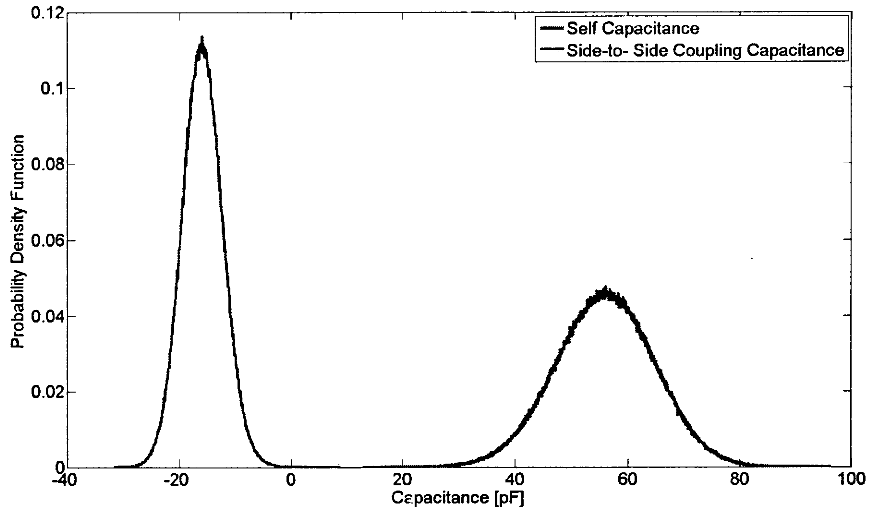


Figure 4-10: Probability density functions of the self capacitance of conductor 10 and the coupling capacitance between conductors 10 and 9.

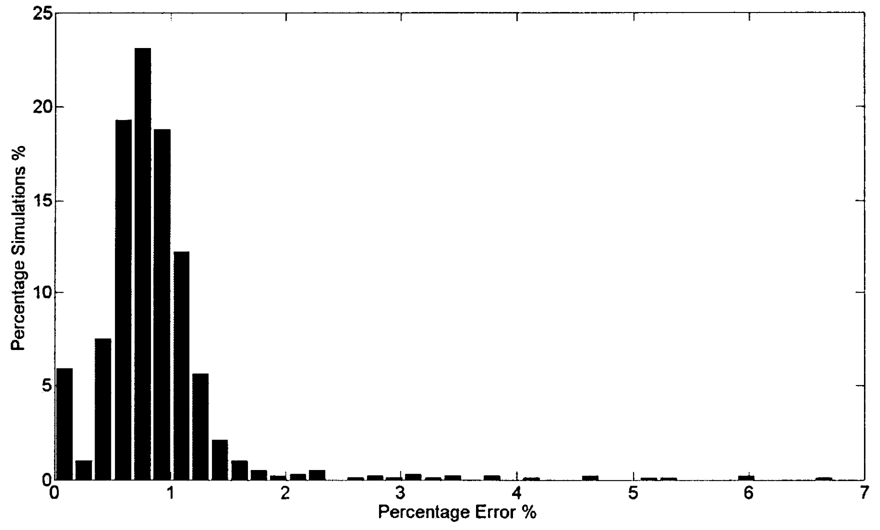


Figure 4-11: Error between model and exact simulation.

Table 4.3: Table Comparing the Performance of Different Algorithms

Algorithm	Property	Time [sec]	# iter.	# MV-prod.
SGM	-	N/A (need 40GB)	-	
our FSGM	-	52	43	
GCR	no recycling	430	14102	14102
our GKR	MF	162	129	62010
our GKR	mF	198	152	67741
our GKR	MC	174	133	62742
our GKR	mC	149	160	56073
GCR recycling	MF	195	124	61059
GCR recycling	mF	204	154	62355
GCR recycling	MC	204	158	56815
GCR recycling	mC	194	130	62207
GMRES recycling	MF	800	388	197096
GMRES recycling	mF	230	203	93204
GMRES recycling	MC	253	214	93905
GMRES recycling	mC	190	207	76904

Our FSGM is then compared to a standard non-intrusive method accelerated by standard GCR and GMRES Krylov subspace recycling (Appendix A.3), and by our new Galerkin Krylov Recycling (GKR) from Section 3.1. Looking at Table 4.3 we observe that our intrusive FSGM significantly outperforms all the non-intrusive Krylov subspace recycling algorithms in this example.

Chapter 5

Novel Intrusive Method

Let us recall that the main idea behind all stochastic solvers is to represent the solution as a product of two unknowns, one in the deterministic space and the other in the stochastic space.

$$\mathbf{x}(\vec{\eta}) = \mathbf{U}(\vec{\eta})\mathbf{v}(\vec{\eta}) \tag{5.1}$$

In most algorithms, in order to avoid having a non-linear representation, one or the other terms is assumed known and the other is then computed. For instance,

1. In SGM, the known is $\mathbf{v}(\vec{\eta}) = \mathbf{h}(\vec{\eta})$ (see (5.3)) and the unknown is $\mathbf{U}$, which is deterministic of size $N \times K$.

2. In any non-intrusive solver (e.g. MC, SCM, SMOR) the known is $\mathbf{v}(\vec{\eta}) = \delta(\vec{\eta} - \vec{\eta}_0)$ and the unknown is $\mathbf{U}$, which is deterministic of size $N \times 1$.

3. In model reduction techniques the known is $\mathbf{U}$, which is deterministic of size $N \times r$, and the unknown is $\mathbf{v}(\vec{\eta})$, which is stochastic of size $r \times 1$.

4. In stochastic projection methods the known is $\mathbf{U}(\vec{\eta})$, which is stochastic of size $N \times r$, and the unknown is $\mathbf{v}$, which is deterministic of size $r \times 1$. A method will be presented in Section 5.2.3 of this chapter.

5. In Neumann expansion methods, the known is $\mathbf{U}(\vec{\eta})$, which is stochastic and is assumed to live in the Krylov subspace of $\mathcal{K}(\mathbf{A}_0^{-1}\mathbf{A}(\vec{\eta}), \mathbf{A}_0^{-1}\mathbf{b})$. The unknown is $\mathbf{v}$,

which is deterministic and is obtained directly from the coefficients of the Neumann expansion (instead of solving any linear systems).

It has been recently recognized that model reduction based algorithms are very efficient in handling stochastic problems. This is primarily due to the fact that the stochastic linear system is assembled by introducing variations to a nominal structure. Such variations only change the shape of the structure but not its topology. Consequently, the resulting solutions tend to share similar characteristics and therefore live in a subspace which can be accurately spanned using a reduced set of basis. The main research interest is therefore to choose a reduced basis that best represents the solution space. Two interesting techniques have been proposed. In [55] the Neumann expansion is used to find an approximation of the covariance matrix of the solution. The Karhunen-Loeve expansion of the approximate covariance matrix is then used to find an optimal reduced basis. The proposed technique suffers however from all the drawbacks of the Neumann based techniques and is computationally very expensive. The second technique [61] tries to simultaneously find both $\mathbf{U}$ and $\mathbf{v}(\vec{\eta})$ by using the so-called generalized spectral method. In such method, both $\mathbf{U}$ and $\mathbf{v}(\vec{\eta})$ are computed simultaneously using a fixed point iteration, such that the residual is orthogonalized with respect to the space spanned by $\mathbf{U}$ and $\mathbf{v}(\vec{\eta})$. Due to the use of the Galerkin projection (residual orthogonalization rather than minimization) the entire matrix $\mathbf{U}$ must be computed simultaneously, which is computationally very expensive. Furthermore, the notion of optimality is defined with respect to the Galerkin projection, rather than the Petrov Galerkin projection, which results in suboptimal basis.

In this chapter we present a novel "intrusive" algorithm, which relies on identifying a small number of dominant directions that best span the overall deterministic-stochastic solution space. We present a variety of techniques to help efficiently identify the dominant subspace. Our algorithm, which we call the stochastic dominant singular values (SDSV) method, has an unprecedented low complexity $O(N^2 + N_P^4)$. Since even in worst case application N_P (the dimension of the parameter space, i.e. the number of uncorrelated parameters) is typically several orders of magnitude smaller than N (the size of the linear system, i.e. the number of discretization elements), from a practical point of view the complexity of our algorithm is basically just $O(N^2)$, making it independent of the dimension of the parameter space.

5.1 Stochastic Dominant Singular Vectors (SDSV) Method

5.1.1 Main Algorithm

To avoid the complexities associated with the standard stochastic solvers, we suggest the use of a *nonlinear* representation of $\mathbf{x}(\vec{\eta})$, where both the deterministic component as well as the stochastic component are assumed unknown. To better understand the idea behind our work let us consider expressing $\mathbf{x}(\vec{\eta})$ in terms of its dominant basis:

$$\mathbf{x}(\vec{\eta}) = \sum_{i=0}^{K} \mathbf{x}_i \Psi_i(\vec{\eta}) = \mathbf{X}\mathbf{h}(\vec{\eta}) = \sum_{i=0}^{r} \sigma_i \mathbf{u}_i \tilde{\mathbf{v}}_i^T \mathbf{h}(\vec{\eta}) = \sum_{i=0}^{r} \mathbf{u}_i \mathbf{v}_i^T \mathbf{h}(\vec{\eta}). \tag{5.2}$$

where $\sum\limits_{i=1}^{r} \sigma_i \mathbf{u}_i \tilde{\mathbf{v}}_i^T$ is the SVD of $\mathbf{X}$, $\sigma_i \tilde{\mathbf{v}}_i = \mathbf{v}_i$ and r is the total number of dominant basis and $\mathbf{h}(\vec{\eta})$ is a vector of the Hermite orthogonal polynomials:

$$\mathbf{h}(\vec{\eta}) = \begin{pmatrix} \Psi_0(\vec{\eta}) & \Psi_1(\vec{\eta}) & \cdots & \Psi_K(\vec{\eta}) \end{pmatrix}^T. \tag{5.3}$$

Note that $\mathbf{v}_i^T \mathbf{h}(\vec{\eta})$ is a scalar polynomial and that (5.2) can therefore be expressed as

$$\mathbf{x}(\vec{\eta}) = \sum_{i=0}^{r} \mathbf{v}_i^T \mathbf{h}(\vec{\eta}) \mathbf{u}_i \tag{5.4}$$

where $\mathbf{u}_i$ is an unknown vector of length N and $\mathbf{v}_i$ is an unknown vector of length K representing a direction in the stochastic space.

The Key Idea. *We propose to decompose the solution vector* $\mathbf{x}(\vec{\eta})$ *in the form of the summation (5.4) and find its components sequentially, i.e. at every iteration* n *of our algorithm we solve only for* $\mathbf{u}_n$ *and* $\mathbf{v}_n$*. These vectors are computed such that they minimize the norm of the residual at iteration* n*.*

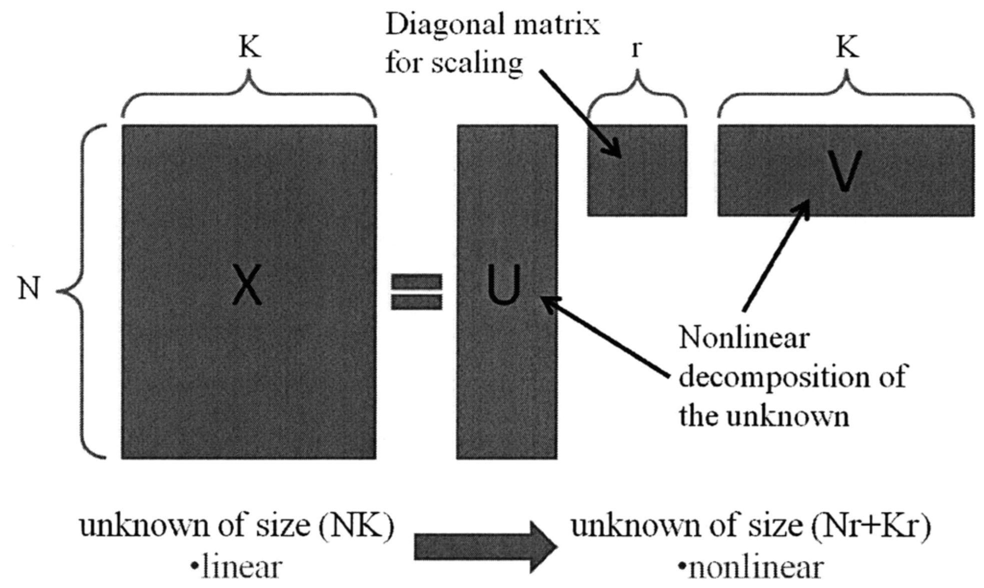

K
Diagonal matrix for scaling
r
K
N
X
=
U
V
Nonlinear decomposition of the unknown
unknown of size (NK) •linear
unknown of size (Nr+Kr) •nonlinear

Figure 5-1: The transformation of the unknown domain from linear of size NK to nonlinear of size Nr+Kr.

To achieve our objective we first substitute (5.4) in (2.16) to obtain

$$\mathbf{A}(\vec{\eta})\sum_{i=0}^{r}\mathbf{v}_i^T\mathbf{h}(\vec{\eta})\mathbf{u}_i = \mathbf{b}(\vec{\eta}) \tag{5.5}$$

Assume that at step n of our algorithm we know all vectors $\mathbf{u}_i, \mathbf{v}_i : i = 0, ..., n-1$ and define

$$\mathbf{x}_n(\vec{\eta}) = \sum_{i=0}^{n}\mathbf{v}_i^T\mathbf{h}(\vec{\eta})\mathbf{u}_i = \mathbf{x}_{n-1}(\vec{\eta}) + \mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{u}_n \tag{5.6}$$

and

$$\mathbf{r}_n(\vec{\eta}) = \mathbf{b}(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{x}_n(\vec{\eta}) \tag{5.7}$$

Equation (5.7) can be put in a recursive form by using (5.6)

$$\begin{aligned}\mathbf{r}_n(\vec{\eta}) &= \mathbf{b}(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{x}_{n-1}(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{u}_n \\ &= \mathbf{r}_{n-1}(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{u}_n \end{aligned}\tag{5.8}$$

where $\mathbf{x}_0 = \mathbf{0}$ and $\mathbf{r}_0 = \mathbf{b}$.

As mentioned above, at step n of our algorithm we find $\mathbf{u}_n$ and $\mathbf{v}_n$ such that the stochastic norm of the residual $\mathbf{r}_n(\vec{\eta})$ is minimized

$$\min_{\mathbf{u}_n,\mathbf{v}_n} ||\mathbf{r}_n(\vec{\eta})||_S^2 \tag{5.9}$$

where the objective function

$$\begin{aligned} ||\mathbf{r}_n(\vec{\eta})||_S^2 &= ||\mathbf{r}_{n-1}(\vec{\eta})||_S^2 - 2\mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{r}_{n-1}(\vec{\eta})\right] \\ &\quad + \mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\right] \end{aligned} \tag{5.10}$$

One approach to minimizing (5.9) is to set the gradient $\mathbf{f}'$ of the objective function $f = ||\mathbf{r}_n(\vec{\eta})||_S^2$ to zero, which can be achieved using Newton's method, i.e. solving at each iteration the linear system

$$\mathbf{f}'' \begin{pmatrix} \Delta\mathbf{u}_n \\ \Delta\mathbf{v}_n \end{pmatrix} = -\mathbf{f}' \tag{5.11}$$

where $\mathbf{f}''$ is the Hessian of the objective function.

$$\mathbf{f}' = \begin{pmatrix} \frac{df}{d\mathbf{u}_n} \\ \frac{df}{d\mathbf{v}_n} \end{pmatrix} \tag{5.12}$$

$$\mathbf{f}'' = \begin{pmatrix} \frac{d^2f}{d\mathbf{u}_n^2} & \frac{d^2f}{d\mathbf{u}_n d\mathbf{v}_n} \\ \frac{d^2f}{d\mathbf{v}_n d\mathbf{u}_n} & \frac{d^2f}{d\mathbf{v}_n^2} \end{pmatrix} \tag{5.13}$$

where

$$\frac{df}{d\mathbf{u}_n} = -2\mathbb{E}\left[\mathbf{A}(\vec{\eta})^T\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{r}_{n-1}(\vec{\eta})\right] + 2\mathbb{E}\left[\mathbf{A}(\vec{\eta})^T\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\right] \tag{5.14}$$

$$\frac{df}{d\mathbf{v}_n} = -2\mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{r}_{n-1}(\vec{\eta})\mathbf{h}(\vec{\eta})\right] + 2\mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{h}(\vec{\eta})\right] \tag{5.15}$$

$$\frac{d^2f}{d\mathbf{u}_n^2} = 2\mathbb{E}\left[\mathbf{A}(\vec{\eta})^T\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{A}(\vec{\eta})\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\right] \tag{5.16}$$

$$\frac{d^2f}{d\mathbf{v}_n^2} = 2\mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{h}(\vec{\eta})\mathbf{h}(\vec{\eta})^T\right] \tag{5.17}$$

$$\frac{d^2 f}{d\mathbf{v}_n d\mathbf{u}_n} = -2\mathbb{E}\left[\mathbf{h}(\vec{\eta})\mathbf{r}_{n-1}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\right] + 4\mathbb{E}\left[\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{h}(\vec{\eta})\mathbf{u}_n^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\right] \tag{5.18}$$

For efficient implementation, the expressions for $\mathbf{f}'$ and $\mathbf{f}''$ are given in terms of the PCE (Hermite) expansion of both the system matrix $\mathbf{A}(\vec{\eta}) = \sum_{i=0}^{K} \mathbf{A}_i\Psi_i(\vec{\eta})$ and the residual vector $\mathbf{r}_n(\vec{\eta}) = \mathbf{R}_n\mathbf{h}(\vec{\eta})$. The implementation details are omitted, however, the resulting expressions are summarized in Appendix B.4.

The number of the free optimization parameters (length of both $\mathbf{u}_n$ and $\mathbf{v}_n$) is $O(N+K)$, which means that at every iteration we only need to solve a linear system of the same size for a complexity $O(N+K)^2$. Such minimization is performed only r times, where r is the number of dominant basis of $\mathbf{x}(\vec{\eta})$. Note, typically $r \ll K \simeq N$. Consequently, the complexity of our algorithm scales with just $O(N^2)$, practically independent of the number of parameters. A more detailed complexity analysis will follow in Section 5.1.4. Note that we use the norm of the residual as an indicator of the accuracy of the solution. In other words, we keep looking sequentially for dominant basis until the norm of the residual becomes smaller than a given threshold.

Algorithm 8 presents a complete summary of our proposed algorithm. Note that both unknowns $\mathbf{u}_n$ and $\mathbf{v}_n$ are combined into a single vector $\mathbf{z}$.

Algorithm 8 The Stochastic Dominant Singular Vectors Method (SDSV)

```
 1: $\mathbf{x}(\vec{\eta}) \leftarrow 0$, $\mathbf{r}(\vec{\eta}) \leftarrow \mathbf{b}(\vec{\eta})$
 2: $\mathbf{u}_n \leftarrow \mathbf{x}_0$, the solution of the nominal problem
 3: $\mathbf{v}_n \leftarrow \mathbf{e}_1$
 4: $\mathbf{z} \leftarrow \begin{pmatrix}\mathbf{u}_n^T & \mathbf{v}_n^T\end{pmatrix}^T$
 5: while $||\mathbf{r}(\vec{\eta})||_S^2 >$ Threshold do
 6:   repeat
 7:     form first derivative $\mathbf{f}'$ as in (5.12)
 8:     form Hessian $\mathbf{f}''$ as in (5.13)
 9:     solve linear system $\mathbf{f}''\Delta\mathbf{z} = -\mathbf{f}'$
10:     $\mathbf{z} \leftarrow \mathbf{z} + \Delta\mathbf{z}$
11:     $\mathbf{u}_n \leftarrow \mathbf{z}(1:N,1)$, $\mathbf{v}_n \leftarrow \mathbf{z}(N+1:N+K,1)$
12:   until $||\mathbf{f}'|| <$ Threshold
13:   $\mathbf{x}(\vec{\eta}) \leftarrow \mathbf{x}(\vec{\eta}) + \mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$.
14:   $\mathbf{r}(\vec{\eta}) \leftarrow \mathbf{r}(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$.
15:   $\mathbf{u}_n \leftarrow \mathbf{x}_0$
16:   $\mathbf{v}_n \leftarrow \mathbf{e}_1$
17:   $\mathbf{z} \leftarrow \begin{pmatrix}\mathbf{u}_n^T & \mathbf{v}_n^T\end{pmatrix}^T$
18: end while
```

Important Note. The solution of the linear system (5.11) can also be implemented using a matrix-free Newton's method or a standard fixed-point iteration.

5.1.2 Convergence Analysis

The following theorem summarizes the convergence properties of our algorithm.

Theorem 5.1.1. *For any nonsingular matrix* $\mathbf{A}(\vec{\eta})$ *the sequence of residuals* $||\mathbf{r}_n(\vec{\eta})||_S^2$ *is strictly decreasing for any non-zero update of the solution* $\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$ *(5.6).*

Proof. The optimal $\mathbf{u}_n$ and $\mathbf{v}_n$ occur when the gradient of the nonlinear objective function (5.10) is zero. Taking the derivative of (5.10) with respect to $\mathbf{u}_n$ we obtain:

$$\mathbf{0} = \mathbb{E}\left[-\mathbf{A}^T(\vec{\eta})\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{r}_{n-1}(\vec{\eta}) + \mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\mathbf{v}_n^T\mathbf{h}(\vec{\eta})\right] \tag{5.19}$$

Multiplying (5.19) by $\mathbf{u}_n^T$ and substituting the resulting expression in (5.10)

$$||\mathbf{r}_n(\vec{\eta})||_S^2 = ||\mathbf{r}_{n-1}(\vec{\eta})||_S^2 - ||\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})||_S^2 \tag{5.20}$$

Since $\mathbf{A}(\vec{\eta})$ is nonsingular, the vector $\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$ is non-zero for any non-zero update to the solution $\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$. Consequently $||\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})||_S^2 > 0$ and $||\mathbf{r}_n(\vec{\eta})||_S^2 < ||\mathbf{r}_{n-1}(\vec{\eta})||_S^2$.
□

5.1.3 Fast Stochastic Matrix Vector Product

The computational complexity of our algorithm is mainly determined by a matrix vector product of the form $\mathbf{A}(\vec{\eta})\mathbf{u}_n$ (see (5.10)). If the system matrix is expanded in terms of the multivariate Hermite polynomials, then such a matrix vector product is computed in $O(N^2K)$

$$\mathbf{A}(\vec{\eta})\mathbf{u}_n = \sum_{i=0}^{K}\mathbf{A}_i\Psi_i(\vec{\eta})\mathbf{u}_n = \sum_{i=0}^{K}\mathbf{A}_i\mathbf{u}_n\Psi_i(\vec{\eta}) = \mathbf{T}\mathbf{h}(\vec{\eta}) \tag{5.21}$$

where we have called the result of this product $\mathbf{t}(\vec{\eta}) = \mathbf{T}\mathbf{h}(\vec{\eta})$. The standard method for computing $\mathbf{T}$ is to form the products $\mathbf{A}_i\mathbf{u}_n$ successively, i.e. successively computing the columns of $\mathbf{T}$. Instead, we will consider computing the rows of $\mathbf{T}$ successively (see Figure 5-2). Note that the k^{th} row of $\mathbf{T}$ is formed by taking the product of the k^{th} row of each $\mathbf{A}_i$ with $\mathbf{u}_n$. Collect all such rows in a single matrix $\mathcal{A}_k \in \mathbb{R}^{K\times N}$ in which the i^{th} row is the k^{th} row of matrix $\mathbf{A}_i$, i.e. $\mathcal{A}_k(i,:) = \mathbf{A}_i(k,:)$. Use the SVD of $\mathcal{A}_k = \mathbf{U}_{K\times q}\mathbf{S}_{q\times q}\mathbf{V}^H_{N\times q}$ to compute the product of $\mathcal{A}_k$ (after decomposition) with $\mathbf{u}_n$ in $O(Nq + Kq)$, where q is the total number of dominant basis of $\mathbf{A}_k$. The transpose of the resulting vector is the k^{th} row of matrix $\mathbf{T}$. Repeating such a process for every row, the total complexity of forming $\mathbf{A}(\vec{\eta})\mathbf{u}_n$ is $O(N^2q + KNq) \simeq O(N^2)$. For the more general case of computing $\mathbf{A}(\vec{\eta})\mathbf{w}(\vec{\eta})$, i.e. the product of a parameter dependent matrix and a parameter dependent vector, we use the SVD to represent the vector $\mathbf{w}(\vec{\eta})$ in addition to the previous algorithm. Similar arguments reveal that the complexity is $O(q\tilde{q}N^2 + KNq\tilde{q} + K^{3/2}N\tilde{q}) \simeq O(N^2)$, where $\tilde{q}$ is the number of dominant basis of $\mathbf{w}(\vec{\eta})$.

Important Observation. *The q dominant basis of the matrix $\mathcal{A}_k$ need to be computed only once as a pre-processing step, while computing the Hermite expansion of $\mathbf{A}(\vec{\eta})$, and stored for later usage. We use the standard power iteration method to compute the dominant basis. We assume that $q \ll N, K$.*

The existence of a low-rank approximation for every $\mathcal{A}_k$ is supported by the structure of the matrix $\mathbf{A}(\vec{\eta})$, in particular the fact that every matrix entry in the k^{th} row of $\mathbf{A}(\vec{\eta})$ depends on the discretization element k and depends primarily on the small subset of parameters affecting elements in close proximity to element k. We verify such observation in our experimental examples.

5.1.4 Complexity Analysis

All subsequent analysis is based on solving the linear system (5.11) using Krylov subspace iterative methods such as GMRES.

Memory: The proposed algorithm requires storing the system matrix of (5.11) which is of memory complexity $O((N+K)^2)$. For efficient implementation one might want to load to the memory all the coefficient matrices $\mathbf{A}_i$ of the Hermite expansion of $\mathbf{A}(\vec{\eta})$, which requires a memory of $O(N^2q_{max} + KNq_{max})$, where q_{max} is the maximum dominant basis

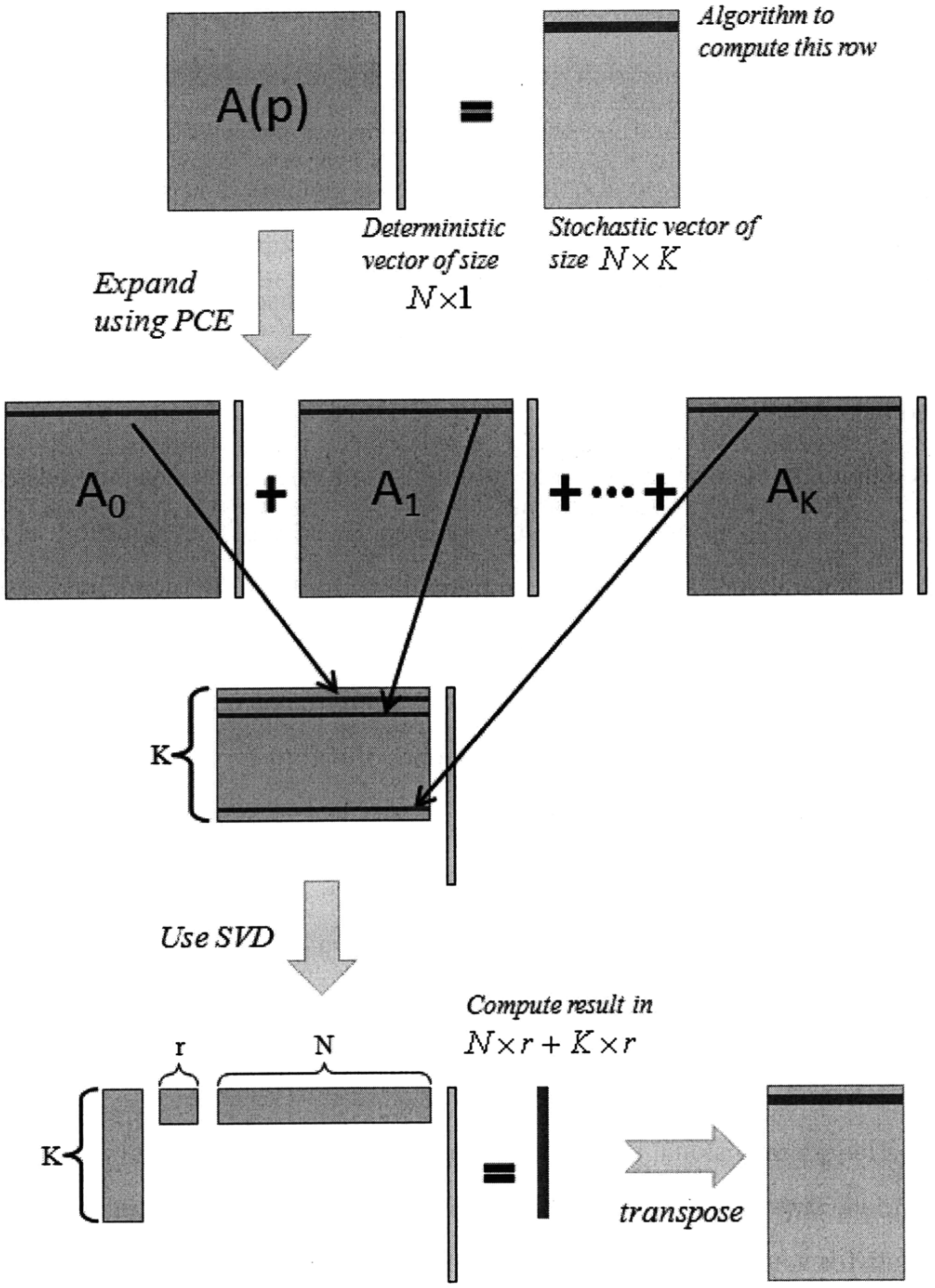

Algorithm to compute this row
A(p)
=
Deterministic vector of size $N\times 1$
Stochastic vector of size $N\times K$
Expand using PCE
A_0
+
A_1
+…+
A_K
K
Use SVD
Compute result in $N\times r + K\times r$
r
N
K
=
transpose

Figure 5-2: Illustration of the fast matrix vector product algorithm to compute a single row. Must repeat algorithm for each row. System matrix compression is implemented only once at the system setup phase. Total memory requirements to store $\mathbf{A}(\vec{\eta})$ is $Nr_{\max}$

for any row following the fast stochastic matrix vector product algorithm in Section 5.1.3.

Time: Most of the computational time is spent on both the matrix filling and the system solve. The most expensive part of the matrix filling is the assembly of $\frac{d^2 f}{d\mathbf{u}_n^2}$, which has to be formed at every iteration due to the change of $\mathbf{v}_n$. Note that since we are using a Krylov subspace method for solving the linear system, we do not need to compute explicitly the matrix-matrix product in $\frac{d^2 f}{d\mathbf{u}_n^2}$. Instead, given $\hat{\mathbf{u}}_n$ and $\hat{\mathbf{v}}_n$ (the guesses to the unknown vectors obtained from the Krylov iteration) we propose to first compute the products $\mathbf{w}(\vec{\eta}) = \mathbf{A}(\vec{\eta})\hat{\mathbf{u}}_n$ for a complexity $O(N^2 + KN)$ as discussed in Section 5.1.3. Then the product $\mathbf{A}(\vec{\eta})^T\mathbf{w}(\vec{\eta})$ is computed for an additional complexity of $O(N^2 + K^{3/2}N)$ as discussed in Section 5.1.3.

The complexity of solving the linear system using Krylov subspace methods is $O(N + K)^2 = O(N^2 + N_P^4)$. Consequently, the total complexity of our algorithm is $O(N^2 + NK^{3/2})$. In general $K^{3/2} < N$ and the total complexity is $O(N^2)$, which corresponds to the *asymptotic complexity of solving just a single nominal system.* This is superior to any other algorithm available in literature since it basically means that the complexity of our algorithm is practically independent of the number of parameters.

5.2 Variants of the SDSV Method

5.2.1 Relaxing the Optimization Problem

Consider a large dimensional problem, e.g. one in which the original system size is $N = 20,000$ and the size of uncorrelated parameters $N_P = 300$ (the total number of orthogonal polynomials for a second order Hermite expansion $K \simeq 50,000$). The size of the linear system to be solved is $O(N + K) \simeq O(70,000)$, which might become prohibitive. To manage such large size problems we propose the following relaxation. We first determine the component of the residual with the largest norm

$$k_{max} = \arg\max_k ||\mathbf{R}_{n-1}(:,k)||$$

Table 5.1: Components of both the system matrix and $\mathbf{v}_n$ contributing to a particular component of the residual.

residual direction	Matrix or $\mathbf{v}_n$	$\mathbf{v}_n$ or Matrix
η_k	η_k	1
	η_k	$\eta_k^2 - 1$
	η_j	$\eta_j \eta_k,\ j \neq k$
$\eta_k^2 - 1$	$\eta_k^2 - 1$	1
	$\eta_k^2 - 1$	$\eta_k^2 - 1$
	η_k	η_k
	$\eta_k \eta_j$	$\eta_k \eta_j,\ j \neq k$
$\eta_{k_1} \eta_{k_2}$	$\eta_{k_1} \eta_{k_2}$	1
	η_{k_1}	η_{k_2}
	$\eta_{k_1}^2 - 1$	$\eta_{k_1} \eta_{k_2}$
	$\eta_{k_2}^2 - 1$	$\eta_{k_1} \eta_{k_2}$
	$\eta_{k_1} \eta_j$	$\eta_j \eta_{k_2},\ j \neq k_1, k_2$

where as mentioned previously $\mathbf{r}_{n-1}(\vec{\eta}) = \mathbf{R}_{n-1}\mathbf{h}(\vec{\eta})$. Then determine the components of $\mathbf{v}_n$ that have a contribution to the direction k_{max} of the maximum residual. In other words, find all the components j of $\mathbf{v}_n$ that, when multiplied with the system matrix, will result in some component in the direction k_{max}.

$$j : \langle \mathbf{A}(\vec{\eta})\Psi_j, \Psi_{k_{max}} \rangle \neq 0$$

The number of such components is $O(N_P)$ as can be inferred from Table 5.1. We then solve a relaxed version of the optimization problem in which only the N_P identified components of $\mathbf{v}_n$ are allowed to vary, while all the other components are fixed to 0. Consequently, the reduced problem is of size $O(N + N_P) \simeq O(20, 300)$.

5.2.2 Stochastic Iterative Method

In this algorithm we will present another alternative that again does not find the optimal solution, but just a "good" one. Using (5.19) and (5.20), we can write the following relation:

$$||\mathbf{r}_{n+1}(\vec{\eta})||_S^2 = ||\mathbf{r}_n(\vec{\eta})||_S^2 - \left\langle \mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta}), \mathbf{r}_n(\vec{\eta}) \right\rangle_S. \tag{5.22}$$

It is obvious from (5.22) that the value of the objective function is minimized when the vector $\mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$ is aligned with $\mathbf{r}_n(\vec{\eta})$.

Main Idea: Instead of optimal alignment we will settle down for good alignment. In other words, we will enforce

$$\mathbf{A}_0\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta}) = \mathbf{r}_1 d_1(\vec{\eta}) \tag{5.23}$$

where $\mathbf{A}_0$ is the average of $\mathbb{E}[\mathbf{A}(\vec{\eta})]$; $\mathbf{r}_1$ and $d_1(\vec{\eta})$ are the dominant deterministic and stochastic directions of the residual $\mathbf{r}_n(\vec{\eta})$. Such vectors can be computed using the standard power iteration. Equation (5.23) can be easily solved to obtain

$$\mathbf{A}_0\mathbf{u}_n = \mathbf{r}_1$$
$$\mathbf{v}_n^T\mathbf{h}(\vec{\eta}) = d_1(\vec{\eta})$$

Algorithm 9 summarizes our complete approach.

Algorithm 9 The stochastic iterative method: An approximate SDSV Method

```
1: $\mathbf{x}_0(\vec{\eta}) \leftarrow 0, \mathbf{r}_0(\vec{\eta}) \leftarrow \mathbf{b}(\vec{\eta}), n \leftarrow 0$
2: **while** $||\mathbf{r}_n(\vec{\eta})||_S^2 >$ Threshold **do**
3:     use power iteration to compute $\mathbf{r}_1$ and $d_1(\vec{\eta})$, the dominant deterministic and stochastic directions of $\mathbf{r}_n(\vec{\eta})$
4:     solve for $\mathbf{u}_n$, $\mathbf{A}_0\mathbf{u}_n = \mathbf{r}_1$
5:     $\mathbf{v}_n^T\mathbf{h}(\vec{\eta}) \leftarrow d_n(\vec{\eta})$
6:     $\mathbf{r}_{n+1}(\vec{\eta}) \leftarrow \mathbf{r}_n(\vec{\eta}) - \mathbf{A}(\vec{\eta})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$
7:     $\mathbf{x}_{n+1}(\vec{\eta}) \leftarrow \mathbf{x}_n(\vec{\eta}) - \mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\vec{\eta})$
8:     $n \leftarrow n + 1$
9: **end while**
```

Computational Complexity: The complexity of the power iteration is that of a small number of matrix vector products, i.e. $O(NK)$. The complexity of the system solve is $O(NlogN)$, since such a system can be easily accelerated using "fast" matrix vector product algorithms (e.g. precorrected FFT or fast multipole). The worst case complexity of the residual update is $O(KN^2)$. However, in practical implementations the matrix $\mathbf{A}(\vec{\eta})$ is either composed of one dense matrix and $K - 1$ sparse matrices or represented using a law rank approximation. In the former case the complexity of the residual update is

$O(N^2 + KN)$, while in the latter it is $O(N^2 r)$, where r is the maximum order of the low rank representation. Consequently, the complexity of our algorithm scales as $O(N^2)$, independent of the size of the parameter space.

5.2.3 A Stochastic Krylov Method

Instead of actually solving the optimization problem we can build basis for $\mathbf{x}(\vec{\eta})$, using a stochastic Krylov algorithm. Recall that in standard deterministic algorithms (e.g. GMRES and GCR) the Krylov subspace $\mathcal{K}(\mathbf{A}, \mathbf{b})$ is the same space as $\mathcal{K}(\mathbf{A}, \mathbf{r})$, and that the latter is usually the one used to build a basis for the solution.

Our idea is to develop a *stochastic* Krylov algorithm. However, we want to use basis for our space that have the form of $\mathbf{u}\mathbf{v}^T\mathbf{h}(\vec{\eta})$, which immediately eliminates using the complete residual in order to build the search subspace. Instead, *we will only use the dominant basis of the residual to build the search subspace.*

$$\mathbf{r}(\vec{\eta}) = \mathbf{R}(\vec{\eta})\mathbf{h}(\vec{\eta}) = \sum_{i=1}^{K} \mathbf{r}_i d_i(\vec{\eta}) \tag{5.24}$$

At every iteration we first get the dominant direction of $\mathbf{r}(\vec{\eta})$ and use it to extend the basis $\mathbf{U}(\vec{\eta})$.

$$\mathbf{U}_{n+1}(\vec{\eta}) = \begin{bmatrix} \mathbf{U}_n(\vec{\eta}) & \mathbf{r}_1 d_1(\vec{\eta}) \end{bmatrix} \tag{5.25}$$

Notice that such an algorithm can be interpreted as a model reduction algorithm, in which the left and right projection matrices are *stochastic.* The solution is then computed as

$$\mathbf{x}(\vec{\eta}) = \mathbf{U}(\vec{\eta})\mathbf{z}$$

where

$$\mathbf{z} = \mathbb{E}\left[\mathbf{U}(\vec{\eta})^H \mathbf{A}(\vec{\eta})^H \mathbf{A}(\vec{\eta}) \mathbf{U}(\vec{\eta})\right]^{-1} \mathbb{E}\left[\mathbf{U}(\vec{\eta})^H \mathbf{A}(\vec{\eta})^H \mathbf{b}\right] \tag{5.26}$$

Important remarks

1. The idea of orthogonalizing the basis vectors in $\mathbf{U}(\vec{\eta})$ is not directly applicable in our algorithm. That is because the difference of two vectors of the form $\mathbf{u}\left(\mathbf{v}^T\mathbf{h}(\vec{\eta})\right)$

is not directly expressible using a single vector of the same form. Consequently, we opt to avoid the orthogonalization step.

2. $\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})$ is a stochastic matrix, which is computed recursively by computing only the product of the system matrix and the last vector added to $\mathbf{U}(\vec{\eta})$. Consequently, the matrix $\mathbb{E}\left[\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\right]$ can be incrementally computed using a single matrix vector product at every iteration. However, the price we pay is that such matrix has to be stored following each iteration. We store it in the form of a three dimensional array of size $\mathbb{R}^{N\times K\times n}$ (see Algorithms 10 and 11 for more information about the implementation details).

3. The main computational complexity in our algorithm is the requirement to store the matrix-matrix product $\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})$. To avoid performance deterioration due to large memory requirements, we propose to use a restart method [66]. In such a method, the stored search direction are destroyed after a fixed number of iterations (determined primarily by the size of memory). A new matrix is rebuilt from scratch. The monotonic decrease of the error norm is guaranteed, however, the speed of convergence is reduced since the previous subspace is neglected.

Algorithm 10 Stochastic Krylov Method

```
1: $\mathbf{r} \leftarrow \mathbf{b}$
2: $\mathbf{AU} \leftarrow []$
3: **repeat**
4:     $\mathbf{u} \leftarrow$ randn(N)
5:     **while** power iteration did not converge **do**
6:         $\mathbf{u} \leftarrow \mathbf{R}\mathbf{R}^T\mathbf{x}$
7:         normalize $\mathbf{u}$
8:     **end while**
9:     $\mathbf{v} \leftarrow \mathbf{R}^T\mathbf{u}$
10:    $\mathbf{AU} \leftarrow \begin{bmatrix} \mathbf{AU} & \mathbf{Auv}^T\mathbf{h} \end{bmatrix}$
11:    solve $\mathbb{E}\left[\mathbf{U}^T\mathbf{A}^T\mathbf{AU}\right]\mathbf{z} = \mathbb{E}\left[\mathbf{U}^T\mathbf{A}^T\mathbf{b}\right]$
12:    $\mathbf{r} = \mathbf{b} - \mathbf{AUz}$
13: **until** $||\mathbf{r}||_S < threshold$
```

Theorem 5.2.1. *The solution* $\mathbf{z}$ *minimizes the norm of the residual in the subspace* $\mathbf{U}(\vec{\eta})$

Algorithm 11 Computing the unknown $\mathbf{z}$ at iteration $n+1$

```
1: $\mathbf{M}_{n+1 \times n+1} \leftarrow \begin{bmatrix} \mathbf{M}_{n\times n} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$
2: for i = 1 : n+1 do
3:     $\mathbf{M}(n+1, i) \leftarrow 0$
4:     for k = 1 : K do
5:         $\mathbf{M}(n+1, i) \leftarrow \mathbf{M}(n+1, i) + \mathbf{AU}[:, k, n+1]^T \mathbf{AU}[:, k, i]$
6:     end for
7:     $\mathbf{M}(i, n+1) \leftarrow \mathbf{M}(n+1, i)$
8: end for
9: $\mathbf{b}(n+1) \leftarrow 0$
10: for k = 1 : K do
11:     $\mathbf{b}(n+1) \leftarrow \mathbf{b}(n+1) + \mathbf{AU}[:, k, n+1]^T \mathbf{B}[:, k]$
12: end for
13: solve for z, $\mathbf{Mz} = \mathbf{b}$
```

Proof.

$$
\begin{aligned}
\mathbf{r}(\vec{\eta}) &= \mathbf{b} - \mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\mathbf{z} \\
\mathbb{E}\left[\mathbf{r}(\vec{\eta})^T\mathbf{r}(\vec{\eta})\right] &= \mathbb{E}\left[\mathbf{b}^T\mathbf{b} - 2\mathbf{b}^T\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\mathbf{z} + \mathbf{z}^T\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\mathbf{z}\right]
\end{aligned}
$$

minimizing the norm of the residual with respect to $\mathbf{z}$ we obtain

$$
\begin{aligned}
\frac{d\mathbb{E}\left[\mathbf{r}(\vec{\eta})^T\mathbf{r}(\vec{\eta})\right]}{d\mathbf{z}} &= \mathbb{E}\left[-2\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{b} + 2\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\mathbf{z}\right] = 0 \\
\mathbb{E}\left[\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{A}(\vec{\eta})\mathbf{U}(\vec{\eta})\right]\mathbf{z} &= \mathbb{E}\left[\mathbf{U}(\vec{\eta})^T\mathbf{A}(\vec{\eta})^T\mathbf{b}\right]
\end{aligned}
$$

□

5.3 Results

5.3.1 Validation of Low Rank Matrix Approximation

In this subsection we demonstrate the efficiency of the matrix compression algorithm presented in Section 5.1.3. Our example system matrix is generated from the 3D capacitance extraction formulation in the presence of surface roughness. The total size of the matrix is $N = 21,000$ and the total number of independent random variables is $N_P = 201$

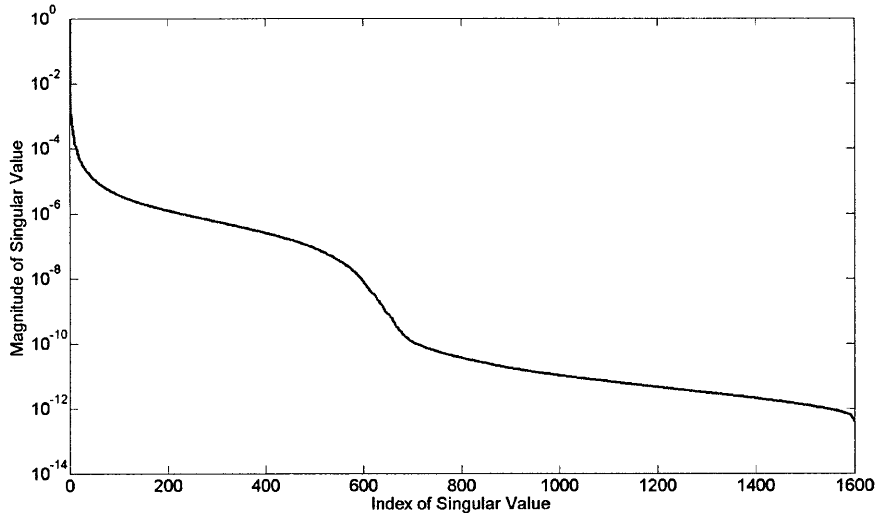


Figure 5-3: Part of the SVD spectrum of the matrix $\mathcal{A}_1$. Only 4 dominant modes are required for 99.9% accuracy.

($K = 20503$). Figure 5-3 shows the some of the singular values of the matrix $\mathcal{A}_1$. It can be inferred from the figure that with only 4 dominant right and left singular vectors the matrix can be approximated to 99.9% accuracy.

It is important to understand that such a preprocessing step, means that K is reduced from 20,503 to 4, corresponding to a reduction ratio of more than 5000.

5.3.2 Small On-Chip 2-D Capacitance Extraction

The first example is the 16-conductor on-chip 2-D structure discussed in detail is Section 3.3.1. Recall that the number of discretization elements is $N = 481$ and the total number of parameters $N_P = 16$. The total number of terms required for a second order Hermite expansion is $K = 153$. We set our algorithm to terminate when the norm of the residual is less than 10^{-4}. We observe that a total of 64 directions are sufficient to represent the solution to the desired accuracy (see Figure 5-4). In a Matlab implementation, the solution is obtained in 25sec, which is a factor of $32\times$ faster than the standard SGM, a factor of $2\times$ faster than the accelerated SGM and a factor of $4\times$ faster than the SCM.

Using the relaxed formulation in Section 5.2.1, the same accuracy is obtained in just 12.5sec using a total of 78 search directions. In other words, the total simulation time is reduced by a factor of $2\times$, while the number of iterations (dominant search directions) is

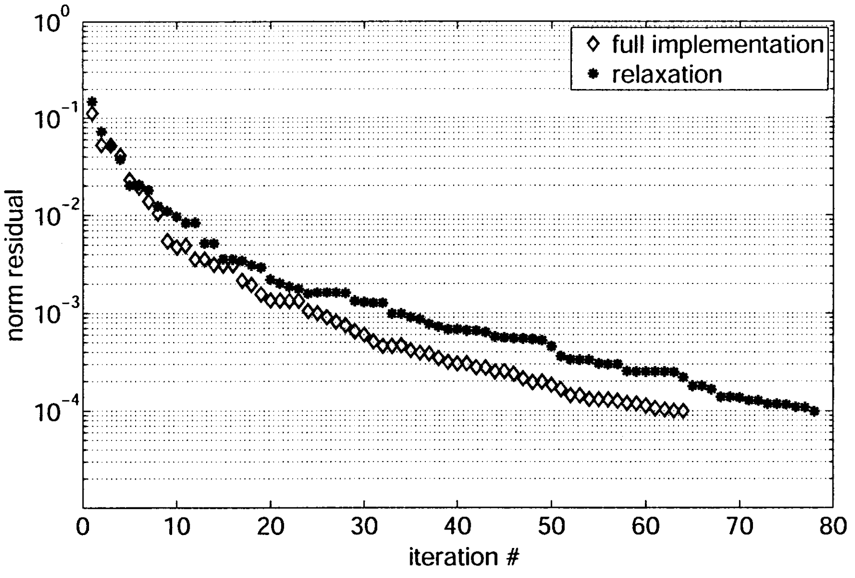


Figure 5-4: Residual norm versus iteration for the full optimization problem and the relaxed optimization problem

increased by around 20% (see Figure 5-4).

5.3.3 Large On-Chip 2-D Capacitance Extraction

The second example is a large 2-D structure consisting of 100 conductors, arranged on a 10×10 grid. The details of the example have been discussed in Section 3.3.2. Recall that $N = 5000$, $N_P = 100$ and $K = 5151$. A total of 142 dominant bases are required to reach a residual of norm 10^{-4}. Using the computed $\mathrm{x}(\vec{\eta})$, we are able to estimate the capacitance of 99% of a 1000 randomly generated structures with accuracy better than 1%. The complete problem (142 search directions) is solved in the time required to solve just 2000 deterministic systems. Our algorithm is $10\times$ faster than the second order sparse grid SCM, which requires more than $20,000$ deterministic solves for the same accuracy. The problem cannot be solved using the standard SGM due to memory constrains (system matrix requires about 200TB).

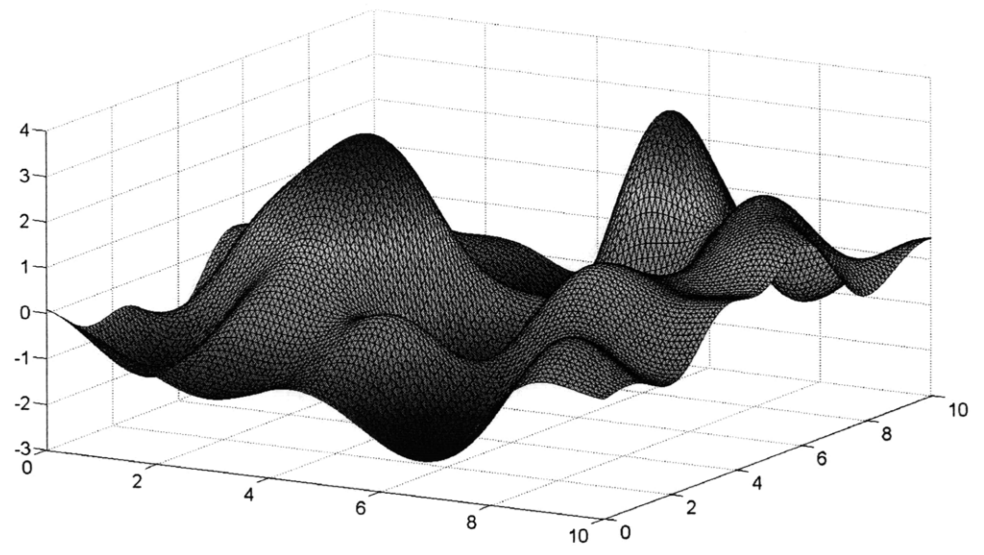


Figure 5-5: One instantiation of a on-package I/O plate pad with very rough surface.

5.3.4 Large On-Package I/O Pad 3-D Capacitance Extraction

In this example we compute the self capacitance of a large plate (I/O package pad) placed in the 3-dimensional space at $z = 0$. The surface width and length are $100\mu\text{m} \times 100\mu\text{m}$ (Figure 5-5). The surface of the plate is very rough. The roughness is described by a Gaussian multivariate PDF and a Gaussian correlation function. The standard deviation is 10 μm and the correlation length is 10 μm. The plate is discretized using $N = 20,000$ triangular panels. The rough surface is described using a total of 91 random variables. The total number of orthogonal polynomials is $K = 4278$. Using the suggested fast matrix vector product algorithm (see Section 5.1.3), the maximum number of modes required to store any matrix row is $q_{max} = 60$. Consequently, the cost of computing the matrix vector product is just $60N^2$. Using just 122 dominant search directions we reduce the norm of the residual to less than 10^{-3}. The total time (not including the system matrix expansion time) required to solve the entire problem is 47minutes. This is at least a factor of $100\times$ faster than the combined Neumann Hermite expansion and a factor of $120\times$ faster than the best sampling-based method.

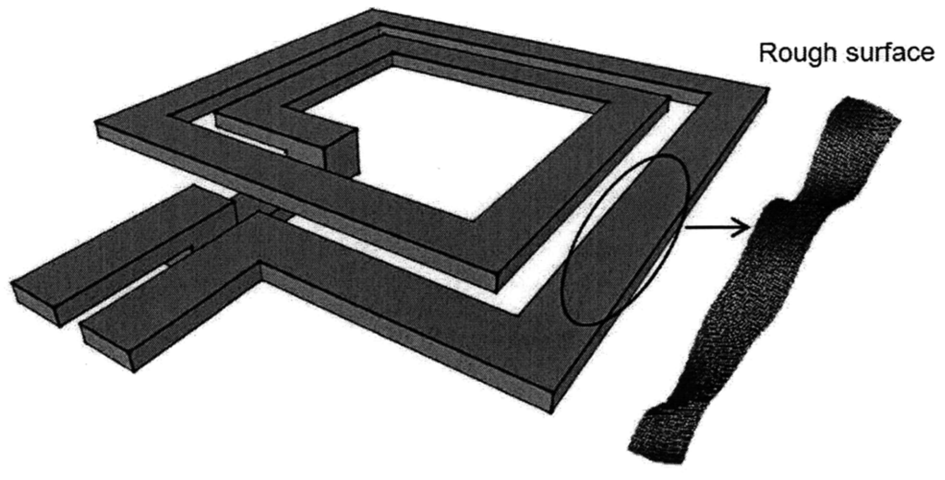


Figure 5-6: 2-turn Inductor with upper rough surface. Part of the surface magnified.

5.3.5 Large On-Package 3-D Impedance Extraction

The final example is a 2 turn inductor similar to the structure in Section 4.4.3 (see Figure 5-6). The side length of the inductor is $1mm$ and the turns are $60\mu m$ in width and $15\mu m$ in thickness. The surface of the inductor is rough with a correlation length of $10\mu m$ and a variance of $3\mu m$. The inductor is discretized using $N = 2750$ volume filaments, and the surface is described using a total of $N_P = 400$ independent random variables. Using a Matlab implementation of our algorithm, the structure is simulated in 4hours. as compared to 32hours using the CNHE, an estimated 150hours using the SCM, and an estimated 828hours using the standard Neumann expansion. Notice that the SGM fails in this example due to excessive memory requirements (800,000 TB). In conclusion, our algorithm is a factor of $8\times$ faster than the CNHE, a factor of $37\times$ faster than the best sampling-based approach and a factor of $200\times$ faster than the standard Neumann expansion.

Part II

Discretization-free Variation-Aware Solvers

Chapter 6

Background on Floating Random Walk (FRW)

6.1 The Standard FRW

The FRW algorithm [13] is a *stochastic* algorithm for solving a *deterministic* Laplace equation problem subject, in its standard version, to Dirichlet boundary conditions. In this subsection we summarize how the FRW computes the potential $\phi(\mathbf{r})$ at a point $\mathbf{r} \in \Omega$, where Ω is the domain external to a group of conductors (e.g. the region of open space delimited by the surfaces of conductors i,j, and k in Figure 6-1). Each conductor is assigned a known constant potential, and the boundary at infinity is assumed at zero potential, i.e. the boundary of Ω is described by Dirichlet boundary conditions. The core idea of the FRW consists of using recursively the Green's Theorem to express the potential at any point inside Ω as linear combination (i.e. infinite dimensional integral) of the known potentials at the boundaries of Ω. This is achieved by starting from the observation that the potential $\phi(\mathbf{r})$ at any point $\mathbf{r} \in \Omega$ can be expressed in terms of the potential $\phi(\mathbf{r}^{(1)})$ at the boundary S_1 of a surrounding *homogeneous* sub-domain:

$$\phi(\mathbf{r}) = \oint_{S_1} P_1(\mathbf{r}, \mathbf{r}^{(1)})\phi(\mathbf{r}^{(1)})d\mathbf{r}^{(1)}, \tag{6.1}$$

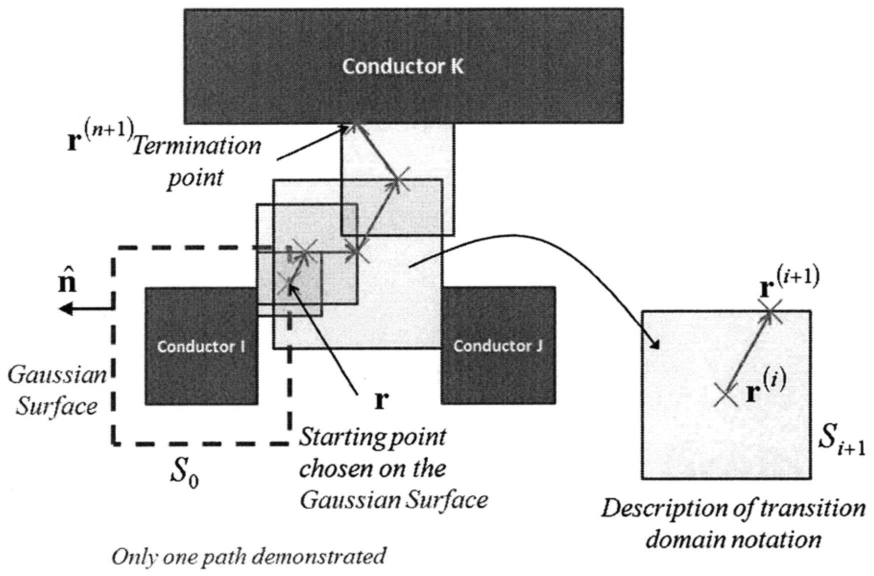

Conductor K
$\mathbf{r}^{(n+1)}$ Termination point
$\hat{\mathbf{n}}$
Gaussian Surface
Conductor I
Conductor J
$\mathbf{r}$
Starting point chosen on the Gaussian Surface
S_0
Only one path demonstrated
$\mathbf{r}^{(i+1)}$
$\mathbf{r}^{(i)}$
S_{i+1}
Description of transition domain notation

Figure 6-1: Typical random walk path from conductor i to conductor j.

where $P_1(\mathbf{r}, \mathbf{r}^{(1)})$ is a Green's function derived from the solution of the Laplace equation with Dirichlet boundary conditions (an example is given in Appendix B.5). Recursion is then used to express the potential $\phi(\mathbf{r}^{(1)})$ in terms of the potential of the boundary S_2 of another homogeneous domain enclosing the point $\mathbf{r}^{(1)}$.

$$\phi(\mathbf{r}^{(1)}) = \oint_{S_2} P_2(\mathbf{r}^{(1)}, \mathbf{r}^{(2)})\phi(\mathbf{r}^{(2)})d\mathbf{r}^{(2)}, \tag{6.2}$$

Substitute (6.2) in (6.1) to obtain

$$\phi(\mathbf{r}) = \oint_{S_1} d\mathbf{r}^{(1)} P_1(\mathbf{r}, \mathbf{r}^{(1)}) \oint_{S_2} d\mathbf{r}^{(2)} P_2(\mathbf{r}^{(1)}, \mathbf{r}^{(2)})\phi(\mathbf{r}^{(2)}), \tag{6.3}$$

The process can then be repeated indefinitely to obtain:

$$\begin{aligned} \phi(\mathbf{r}) &= \lim_{n\to\infty} \oint_{S_1} d\mathbf{r}^{(1)} P_1(\mathbf{r}, \mathbf{r}^{(1)}) \oint_{S_2} d\mathbf{r}^{(2)} P_2(\mathbf{r}^{(1)}, \mathbf{r}^{(2)}) \times \cdots \times \\ &\quad \times \cdots \times \oint_{S_n} d\mathbf{r}^{(n)} P_n(\mathbf{r}^{(n-1)}, \mathbf{r}^{(n)})\phi(\mathbf{r}^{(n)}). \end{aligned} \tag{6.4}$$

The following observations enable efficient computation of the multi-dimensional integral (6.4):

Observation 6.1.1. *The Green's function $P_i(\mathbf{r}^{(i-1)}, \mathbf{r}^{(i)})$ can be interpreted as a probability density function, since it is always positive, and since its integral over S_i is one. Such probability density function can be interpreted as the probability of picking a point $\mathbf{r}^{(i)}$ on the boundary S_i of the homogeneous domain surrounding the point $\mathbf{r}^{(i-1)}$. As we will see in the following few paragraphs, this probability density function will assume the meaning of the probability of moving from the point $\mathbf{r}^{(i-1)}$ to the point $\mathbf{r}^{(i)}$.*

Observation 6.1.2. *In the floating random walk algorithm the domain boundary S_i is the boundary of the* **largest** *homogeneous square/circle (in 2D) or cube/sphere (in 3D) having the point $\mathbf{r}^{(i-1)}$ in its center, and not including any conductor. By construction, S_i will overlap in part at least with one conductor surface, where the potential is known and the recursion can terminate. The region of space delimited by the boundary S_i will be denoted as a "transition domain".*

Following these two observations, the multidimensional integral (6.4) is computed by the FRW using Monte Carlo integration, where only one quadrature point is selected for each integral over a transition domain boundary. The sequence of quadrature points on the boundary of different transition domains can be interpreted as a "random walk (or path)" from a transition domain to another, whose stopping criterion is achieved when a step of the walk falls within a small distance ϵ from a surface with **known** potential (e.g. conductor surface). The expected value of the potential is then given by

$$\mathbb{E}[\phi(\mathbf{r})] = \frac{1}{M} \sum_{m=1}^{M} \Phi_m, \tag{6.5}$$

where $\mathbb{E}[\cdot]$ is the expectation operator and Φ_m is the potential obtained at the termination of the path m (starting from location $\mathbf{r}$ and terminating at some point with known potential Φ_m), and M is the total number of paths. It can shown that the average potential (6.5) is an unbiased estimator of the potential at point $\mathbf{r}$. Algorithm 12 summarizes the procedure.

The above algorithm for potential computation can be easily used for capacitance ex-

Algorithm 12 Standard FRW for Homogeneous Media

```
1: $\phi(\mathbf{r}) \leftarrow 0, M \leftarrow 0$
2: Pick a point r at which we desire to compute the potential.
3: repeat
4:   $M \leftarrow M + 1$
5:   $i \leftarrow 1, \mathbf{r}^{(i)} \leftarrow \mathbf{r}$, PathIsIncomplete ← true
6:   while PathIsIncomplete do
7:     Construct a transition domain as in observation 6.1.2
8:     Pick a point $\mathbf{r}^{(i+1)}$ on the boundary of domain according to the transition probability (observation 6.1.1)
9:     if $\mathbf{r}^{(i+1)}$ has a prescribed potential $\Phi$ (i.e. lies on a conductor boundary) then
10:      $\phi(\mathbf{r}) \leftarrow \phi(\mathbf{r}) + \Phi$
11:      PathIsIncomplete ← false
12:    else
13:      $i \leftarrow i + 1$
14:    end if
15:  end while
16: until convergence or maximum iteration count achieved
17: $\phi(\mathbf{r}) \leftarrow \frac{\phi(\mathbf{r})}{M}$
```

traction. To compute for instance the capacitance C_{ij} between conductor i and conductor j, we can set conductor j at unit potential, while setting all the other conductors at zero potential.

$$C_{ij} = q_i|_{\phi_j=1} = \int_{S_0} \varepsilon \vec{E}(\mathbf{r}) \cdot \hat{n}\, d\mathbf{r} = \int_{S_0} \varepsilon\left(-\nabla\phi(\mathbf{r})\right) \cdot \hat{n}\, d\mathbf{r}, \tag{6.6}$$

where S_0 is a closed surface, called Gaussian surface, surrounding conductor i, $\hat{n}$ is the corresponding normal $\hat{n}$, ε is the permittivity, and $\vec{E}(\mathbf{r})$ is the electric field vector. The potential $\phi(\mathbf{r})$ can be expressed using (6.4) since it is the solution of the Laplace equation with Dirichlet boundary conditions (all conductors except for j are at zeros potential). The final capacitance formula is given by:

$$C_{ij} = -\int_{S_0} d\mathbf{r}\, p_0 \int_{S_1} d\mathbf{r}^{(1)} \frac{\varepsilon \nabla_n P_1(\mathbf{r}, \mathbf{r}^{(1)})}{p_0 P_1(\mathbf{r}, \mathbf{r}^{(1)})} P_1(\mathbf{r}, \mathbf{r}^{(1)}) \times$$
$$\times \int_{S_2} d\mathbf{r}^{(2)} P_2(\mathbf{r}^{(1)}, \mathbf{r}^{(2)}) \times \cdots \times \int_{S_n} d\mathbf{r}^{(n)} P_n(\mathbf{r}^{(n-1)}, \mathbf{r}^{(n)}) \phi(\mathbf{r}^{(n)}) \tag{6.7}$$

where $\nabla_n = \hat{n}\cdot\nabla$ and $\nabla_n P_1(\mathbf{r}, \mathbf{r}^{(1)})$ is computed analytically from the analytical expression for $P_1(\mathbf{r}, \mathbf{r}^{(1)})$ (see Appendix B.5). Similar to the potential computation, the multidimensional integral (6.7) is computed using standard Monte Carlo integration by choosing one quadrature point for each integral. However, the first point of each path is randomly selected using a random variable p_0 uniformly distributed over the close surface S_0 surrounding conductor i. The capacitance C_{ij} is then computed by averaging the contributions of all the M paths from

$$C_{ij} = \frac{1}{M} \sum_{m=1}^{M} \omega_m \Phi_m, \tag{6.8}$$

where for the m^{th} path,

$$\omega_m = \frac{\varepsilon \nabla_n P_1(\mathbf{r}, \mathbf{r}^{(1)})}{p_0 P_1(\mathbf{r}, \mathbf{r}^{(1)})}, \tag{6.9}$$

and the potential Φ_m at the end of the path, given the boundary conditions, is

$$\Phi_m = \begin{cases} 1 & \text{if path } m \text{ terminates on conductor } j \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

Simplifying

$$C_{ij} = \frac{1}{M} \sum_{\substack{\text{path } m \text{ terminates} \\ \text{on conductor } j}} \omega_m. \tag{6.11}$$

One can further observe that the FRW paths used to calculate C_{ij} are not affected by the numerical value of the conductor potentials. Therefore, one single run of the FRW can provide all the entries for column i of the capacitance matrix by simply re-using (6.10) and (6.11) for each entry (i, j).

6.1.1 Discussion on the FRW

The FRW algorithm has a variety of computational advantages. Foremost, the algorithm does not require any memory utilization, except for a single output capacitance matrix. This is very attractive since one of the main bottlenecks of deterministic algorithms is the large memory requirements which in many cases renders efficient accurate computations impossible. In addition, the random walk algorithm is inherently parallel. This feature should be appreciated in light of the recent advances and trends in multithreaded, multicore computer architectures. The total time complexity required of the FRW is $T = t_{walk}\frac{M}{N_p}$, where t_{walk} is the time required for a single random walk and N_{proc} is the number of processors. The single walk time t_{walk} is composed of

1. The time to compute the random variable ω_m (only once).
2. The time to compute the transition domain corresponding to a given transition point (Figure 6-1) (repeated at every step). This time dominates the walk time, since the necessary computations involve geometrical manipulations.
3. The time to compute the next point in the path (repeated at every step).
4. The time to evaluate the stopping decision (repeated at every step).

The above discussion reveals that t_{walk} is linearly proportional to the number of steps of a single random walk (path length). This in turn explains why structures characterized by a dense conductor configurations (which typically exhibit small average path lengths) perform best with FRW. Fortunately, such structures are very common in integrated circuit capacitance extraction applications. Notice also that such structures are the hardest to simulate using the standard discretization based methods, which makes FRW methods very appealing for large scale capacitance extraction applications.

6.2 Standard FRW in Multi-layered Media

The standard floating random walk algorithm can handle arbitrary layered media. This is achieved by treating the interfaces between the dielectric layers as constraints on the transition domain size and consequently as intermediate stopping points [13] (see Figure 6-2). The difference between a conductor surface and a dielectric interface is that the potential of the former is known, while that of the latter is unknown. Consequently, after stopping at a dielectric interface, the random walk is resumed from there. The continuations are repeated until the walk terminates at a conductor surface. If we consider current technologies with multiple thin layered dielectric configurations, such an algorithm becomes very time consuming, since the average transition domain size can become very small. A more efficient

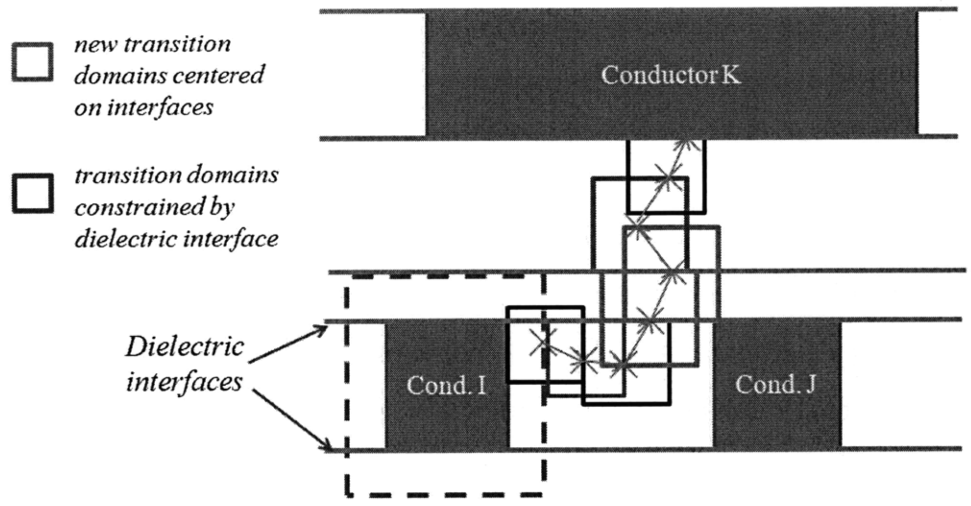

new transition domains centered on interfaces
transition domains constrained by dielectric interface
Dielectric interfaces
Conductor K
Cond. I
Cond. J

Figure 6-2: Standard floating random walk algorithm in multilayered media. Transition domains constrained by dielectric interfaces.

approach was derived for simple dielectric configurations [36]. It relies on pre-computing (using a stochastic algorithm) and tabulating the transition probabilities offline. These values are then recalled during the actual walk. Unfortunately, this approach is limited to a small number of dielectrics and is hard to generalize since it requires the precomputation and tabulation of the transition probabilities associated with all possible transition domains necessary to complete the random walk. Furthermore, it does not seem to exploit the possibility of computing the layered Green's function using a deterministic algorithm, nor does it benefit from the distinct advantages of computing the layered Green's function online rather than offline. In Section 7.3 we present a novel algorithm that solves the above two problems.

6.3 Standard FRW for Floating-Potential Metal Fill

"Charge neutral floating potential metal fill" are small pieces of metal inserted in empty areas of the layout to facilitate planarization. They are commonly referred to as just "fill". There are basically two different techniques that are currently used to extract the capacitance of multi-conductor structures in the presence of fill. The first is to extract the complete capacitance matrix for all the conductors and the fill, and then use the standard capacitance reduction algorithm to compute the desired conductor capacitance matrix C_c.

$$C_c = \left(C_{cc} - C_{ff}^{-1} C_{fc} C_{cf}\right) \tag{6.12}$$

where C_{cc}, $C_{cf} = C_{fc}^T$, and C_{ff} are respectively the conductor self capacitance matrix, the conductor-to-fill mutual capacitance matrix, and the fill self capacitance matrix. Note that when doing a full extraction, the fill is not considered at a floating potential, and is assigned either zero or unit potential. Therefore, the problem of floating potentials is basically bypassed. A clear disadvantage of such technique is that it requires the computation of a much larger capacitance matrix, which is always very expensive and in many cases undesirable.

A second technique, presented in [6], consists of imposing explicitly a charge neutrality condition for the fills, and using a finite difference scheme to directly approximate the derivative term in the equation

$$\oint_{\Gamma_F} \nabla_n \phi(\mathbf{r}) d\mathbf{r} = 0, \tag{6.13}$$

where Γ_F is the surface of the fill. The derivative approximation relates the potential of the fill to the potential of a nearby surface, and thereby acts as a walk from the fill to such nearby surface. The problem with this technique is that the distance between the fill and the next step is very small, and the walk tends to stagnate at the fill surface. In Section 7.1, we demonstrate that our generalized algorithm is an alternative technique, which solves the fill extraction problem efficiently and accurately.

6.4 Standard FRW for Homogeneous Neumann Boundary Conditions

The path reflection algorithm [50] is generally used to mimic the Neumann boundary conditions within the standard floating random walk (see Figure 6-3). Despite its efficiency to treat straight boundaries, this reflection algorithm is not suitable for more general boundary configurations (e.g. at corners or irregular boundaries). In Section 7.1 we demonstrate how

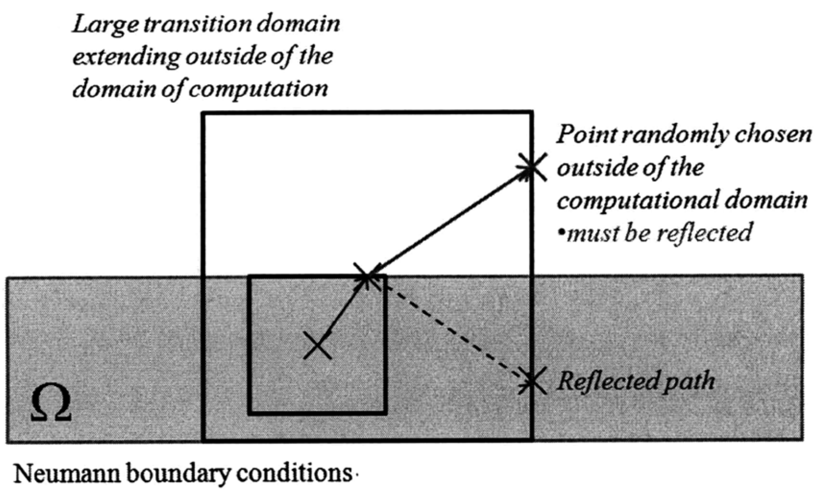

Large transition domain extending outside of the domain of computation
Point randomly chosen outside of the computational domain
•must be reflected
Reflected path
Ω
Neumann boundary conditions

Figure 6-3: Standard reflection algorithm used to handle the Neumann boundary conditions.

our generalized floating random walk can handle Neumann boundary conditions without the need for path reflection.

Chapter 7

Generalized Floating Random Walk Algorithm

Unfortunately, the FRW, in its state of the art, is a very limited algorithm that works predominately for the capacitance extraction of simple homogeneous Dirichlet problems (see for instance the specialized and quite inefficient algorithms required in order to handle multilayered configurations [36] or floating-potential metals [6]). In this chapter we present a new generalized framework for the FRW, which includes handling multilayered or inhomogeneous dielectric regions, floating-potential metal fill and resistance extraction, including both Neumann and mixed Dirichlet-Neumann boundary conditions.

7.1 Generalization of the Floating Random Walk

One of the main advantages of the floating random walk algorithm is that it relies only on the *existence* of the transition probability, rather than on the particular form of the transition probability density function. This fact inspires generalizing the FRW from cases where one knows a closed form transition probability (e.g. centered homogeneous square domain) to more complex cases where one can compute a transition probability, for instance with any available numerical technique. Among complex problems are three dimensional non-Manhattan type geometries with inhomogeneous media, and problems described by mixed Dirichlet and Neumann boundary conditions.

In the generalized FRW (GFRW) setup we might want to make sure that the walk does not reach certain parts of the transition domain boundaries. Consider for example a region of the transition domain boundary that is touching a conductor with unspecified potential (Figure 7-1). If we were to select a point on such regions, the walk will not be able to proceed.

Definition 7.1.1. *The parts of the boundary of a given transition domain on which we do not want the walk to proceed are called "forbidden". The remaining parts of the boundary on which we allow the walk to proceed are called "allowed".*

Definition 7.1.2. *The transition probability from any point $\mathbf{r}_I$ inside of a given transition domain Ω to a point $\mathbf{r}'$ on the allowed boundary Γ_A of the transition domain is defined as a function $P(\mathbf{r}_I, \mathbf{r}')$ that satisfies the following three conditions*

$$\begin{aligned} \phi(\mathbf{r}_I) &= \int\limits_{\Gamma_A} P(\mathbf{r}_I, \mathbf{r}')\phi(\mathbf{r}')d\mathbf{r}' && (7.1) \\ 1 &= \int\limits_{\Gamma_A} P(\mathbf{r}_I, \mathbf{r}')d\mathbf{r}' && (7.2) \\ P(\mathbf{r}_I, \mathbf{r}') &> 0 \quad \forall\, \mathbf{r}' \in \Gamma_A, \mathbf{r}_I \in \Omega. && (7.3) \end{aligned}$$

The following theorem summarizes cases in which such a transition probability can be properly defined:

Theorem 7.1.1. *Consider a Laplace equation*

$$\nabla \kappa(\mathbf{r}) \nabla \phi(\mathbf{r}) = 0 \tag{7.4}$$

defined on a non-simply connected multi-medium domain, $\Omega = \bigcup_{i=1}^{N_D} \Omega_i$ (see Figure 7-1 and Figure 7-4), where each subdomain medium is defined by $\kappa(\mathbf{r}) = \kappa_i$ if $\mathbf{r} \in \Omega_i$

IF at least part of the domain boundary is assigned Dirichlet conditions (allowed part)

$$\phi(\mathbf{r}) = \phi_0(\mathbf{r}), \mathbf{r} \in \Gamma_D \tag{7.5}$$

and the remaining part (forbidden part) is fully covered either by homogeneous Neumann *conditions*

$$\nabla_n \phi(\mathbf{r}) = 0, \mathbf{r} \in \Gamma_N \tag{7.6}$$

or by flux neutrality,

$$\int_{\Gamma_F} \kappa(\mathbf{r}) \nabla_n \phi(\mathbf{r}) d\mathbf{r} = 0 \tag{7.7}$$

THEN there exists a transition probability from any internal point of the domain to the allowed part of the boundary (i.e. with the Dirichlet conditions).

Proof. The complete proof is given in Appendix B.7. □

As we will show in details in Sections 7.3, 7.4 and 7.5 conditions (7.5), (7.6) and (7.7) can cover the vast majority of practical problems of interest in VLSI electrical extraction.

Algorithm 13 summarizes the main steps of our generalized floating random walk algorithm. These steps apply to any Laplace equation problem regardless of it's complex-

Algorithm 13 Template of Generalized Floating Random Walk (GFRW)

```
1: Pick a point.
2: repeat
3:    Generate a transition domain surrounding the point and satisfying the conditions of
      Theorem. 7.1.1
4:    Compute the transition probability using any available technique. (e.g. finite differ-
      ence in Section 7.2)
5:    Pick a point on the boundary according to the computed transition probability.
6: until stopping criterion is met.
```

ity. Notice that generating random variables following any arbitrary transition probability, as required in step 5 of Algorithm 13, is implemented using the inverse transform sampling [15] coupled with a standard uniform number generator (e.g. Matlab's rand()).

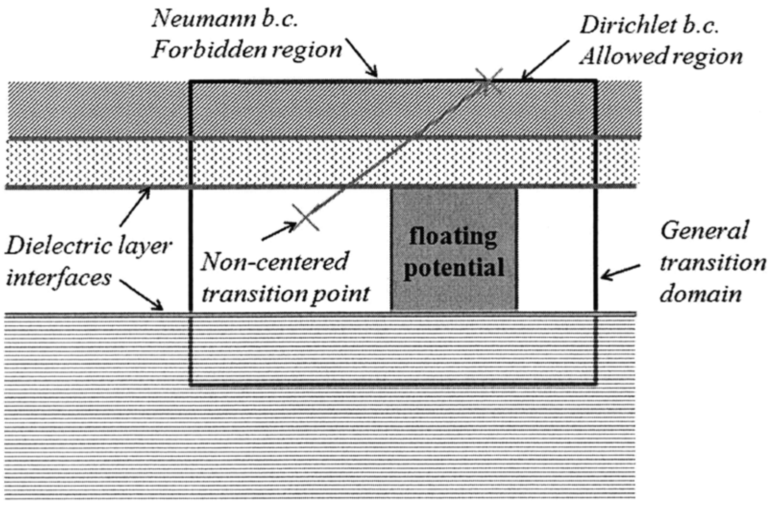

Neumann b.c.
Forbidden region
Dirichlet b.c.
Allowed region
Dielectric layer interfaces
Non-centered transition point
floating potential
General transition domain

Figure 7-1: A generalized transition domain including charge neutral metal, multilayered media and described by a mixture of Neumann and Dirichlet boundary conditions. Both allowed and forbidden boundaries clearly identified.

7.2 A General Technique to Compute the Probability Density Function

One simple way to enforce (7.4), (7.5), (7.6) and (7.7) is to use the finite difference method, obtaining the linear system

$$\begin{pmatrix} \mathbf{E}_{11} & \mathbf{E}_{12} & \mathbf{E}_{13} \\ 0 & \mathbf{I} & 0 \\ \mathbf{E}_{31} & 0 & \mathbf{E}_{33} \end{pmatrix} \begin{pmatrix} \phi_I \\ \phi_A \\ \phi_F \end{pmatrix} = \begin{pmatrix} 0 \\ \Phi_0 \\ 0 \end{pmatrix} \tag{7.8}$$

where $\mathbf{E}_{11}$, $\mathbf{E}_{12}$ and $\mathbf{E}_{13}$ are the standard extremely sparse finite difference matrices discretizing the Laplace operator in (7.4). The second row of the matrix corresponds to the discretization of the Dirichlet condition (7.5) and Φ_0 is a vector containing the known numerical values of the potential on the allowed part of the boundary. $\mathbf{E}_{31}$ and $\mathbf{E}_{33}$ in the third row discretize conditions (7.6) and (7.7), and ϕ_F is a vector of potentials at the grid points on the forbidden boundary of the transition domain. This system of equations can be reduced by eliminating the intermediate variable ϕ_F to obtain

$$\phi_I = \mathcal{P}\ \phi_A, \tag{7.9}$$

where matrix $\mathcal{P} = -\left(\mathbf{E}_{11} - \mathbf{E}_{13}\mathbf{E}_{33}^{-1}\mathbf{E}_{31}\right)^{-1}\mathbf{E}_{12}$. We can observe that (7.9) is a discretized version of (7.1), and since it was obtained by enforcing the conditions (7.4), (7.5), (7.6) and (7.7) in Theorem 7.1.1, then $\mathcal{P}$ automatically also satisfies (7.2) and (7.3) in discretized form

$$\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}^T = \mathcal{P}\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}^T$$

$$\mathcal{P}_{ij} > 0 \quad \forall\, i,j$$

and represents therefore the desired transition probability density function in discretized form. Specifically, $\mathcal{P}_{ij}$ is the probability of moving from internal point i on the finite difference discretization grid, to a point j on the allowed boundary of the transition domain. Notice that in order to compute the random variable ω_m (see for instance (6.11)) associated with every path, we need to compute the gradient of the probability density function in the direction of the normal to the Gaussian surface S_0. Since we compute the entire matrix $\mathcal{P}$, we can compute numerically the required gradient using a finite difference approximation to the operator ∇.

7.3 Applications of the GFRW to Non-Homogeneous Dielectric Medium

In this section we show in details how to use Theorem 7.1.1 to extract capacitance in non-homogeneous dielectric media. Let κ_i be the dielectric constant ε_i of the different dielectric media. As mentioned before in Algorithm 13, the transition domains are constrained by surrounding conductors (Figure 7-3), and *not* by dielectric interfaces. In this specific ca-

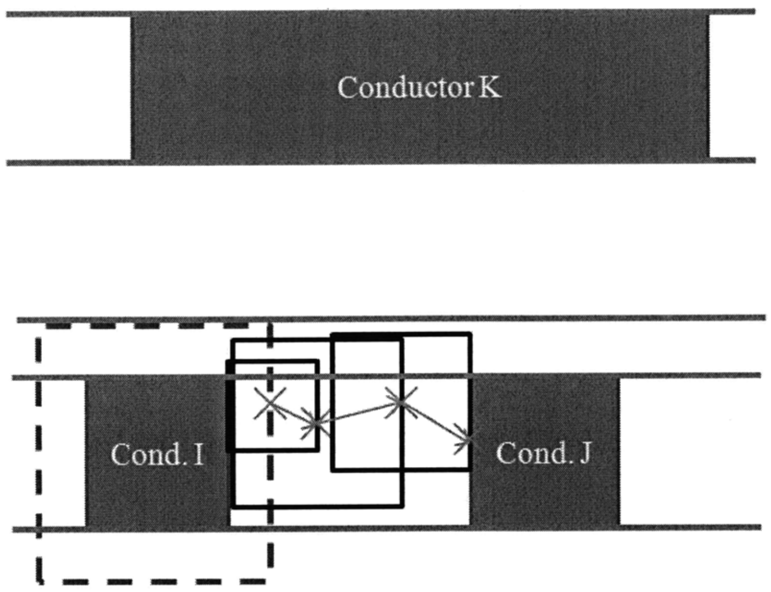


Figure 7-2: A generalized floating random walk in multilayered media. Walk is *not* constrained by the dielectric media.

pacitance extraction setup there are no forbidden boundaries (Neumann or flux neutrality), therefore (7.9) is reduced to

$$\mathcal{P} = -\mathbf{E}_{11}^{-1}\mathbf{E}_{12}, \tag{7.10}$$

In order to further optimize our approach we note that each transition point does not necessarily need to be in the center of the transition domain, as prescribed by the standard FRW algorithms. In fact, one can simply use the appropriate row of matrix $\mathcal{P}$, representing the desired transition probability from *any* given point inside of the non-homogeneous dielectric transition domain to its boundary.

Since the transition domains are determined only by the geometry, which is fixed at every step for every walk, and since the computational domain can be fully covered by a small number of unique transition domains, the transition probabilities (7.10) need to be calculated only for a small number of such domains. It should be further emphasized that transition domains and probabilities calculated in the first random walk can be re-used in all the subsequent walks. Moreover, the required memory to store the transition probabilities is insignificant (typically one can store up to 150 different 3D transition probability matrices in less that 2GB).

A single step of our generalized FRW for non-homogeneous dielectric media is finally summarized in Algorithm 14, where the transition probabilities are computed incrementally only for the needed transition domains and then stored for efficient re-use.

Algorithm 14 A Single Step of the Generalized FRW for Non-Homogeneous Dielectric Media

```
1: for a given transition point do
2:    Search for a precomputed transition domain fully enclosing the given point
3:    if precomputed domain found (e.g. point 1 in Figure 7-3) then
4:       Interpolate from precomputed transition probability and proceed to next transition
         point
5:    else {precomputed domain not found (point 2 Figure 7-3)}
6:       Generate new transition domain such that it extends to all neighboring conductors,
         fully enclosing the new point. (Note: the new point will not be generally in center
         of the new domain)
7:       Call finite difference routine and compute (7.10), the full transition probability
         matrix for a grid of points within the interior of the domain
8:       Update database with new transition domain and corresponding transition proba-
         bilities
9:    end if
10: end for
```

Note that in general VLSI applications the dielectric medium is stratified, such that the value of the dielectric constant varies only in the z-direction. In such cases one does not need to use 3D finite difference and can instead efficiently compute the transition probability using analytical spectral analysis in the x-y domain and 1D finite difference for the z-direction.

7.4 Application of the GFRW to Floating Potential Metal Fill

As mentioned in Section 6.3, a "fill" is a charge neutral metal with floating potential. We notice that (7.7) in Theorem 7.1.1 can be exploited to impose charge neutrality, therefore our GFRW can be used to handle fill extremely efficiently. Specifically, Theorem 7.1.1 suggests constructing the transition domain such that it fully encloses one or more fill, and it extends as much as possible without including any other conductors. Finally, when using

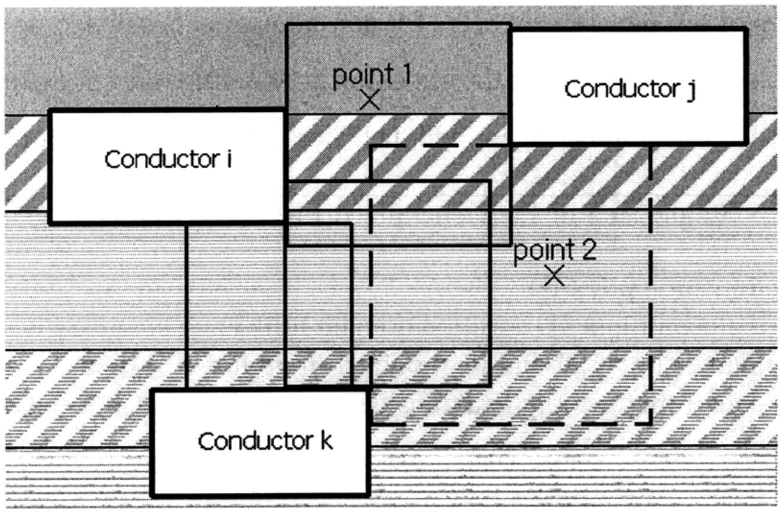


Figure 7-3: Typical non-homogeneous dielectric media. Transition point 1 lies within one of the transition domains (solid boundaries) for which the transition probabilities have been precomputed. Transition point 2 will require the generation of a new transition domain (dashed boundary) and the computation of all its transition probabilities.

for instance finite difference (as in Section 7.2) to compute numerically the PDF, one can simply notice that the charge neutrality is captured by the third row of (7.8).

7.5 Application of the GFRW to Resistance Extraction

When performing resistance extraction, one needs to solve the Laplace equation in a closed domain with Dirichlet conditions at the ports, homogeneous Neumann boundary conditions at the external boundaries of the domain, and current conservation condition at ports (perfect conductors) not connected to a voltage source, i.e. floating ports (see Figure 7-4). Since the current conservation condition can be captured by the flux neutrality condition (7.7) of Theorem 7.1.1, we can use our generalized FRW to handle resistance extraction. Specifically, Theorem 7.1.1, with κ_i being the conductivity σ_i, suggests constraining the transition domains only with the boundaries of the solution domain, while allowing such transition domains to fully enclose one or more floating ports (see Figure 7-4). The homogeneous Neumann and the current conservation conditions define the forbidden boundaries of any transition domain. All other boundaries are allowed.

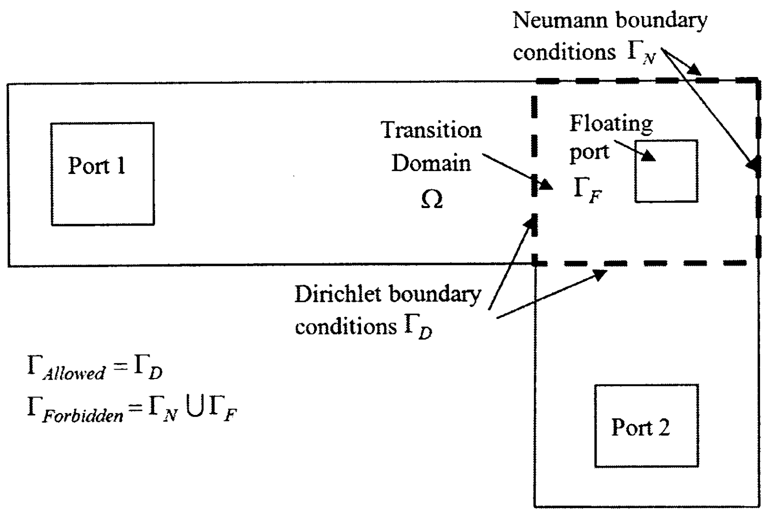

Neumann boundary conditions Γ_N
Port 1
Transition Domain Ω
Floating port Γ_F
Dirichlet boundary conditions Γ_D
$\Gamma_{Allowed} = \Gamma_D$
$\Gamma_{Forbidden} = \Gamma_N \cup \Gamma_F$
Port 2

Figure 7-4: A top view of a typical 2D IC resistor. For resistance extraction ports 1 and 2 are assigned unit and zero potential, respectively. The net current flowing through the floating potential port is assumed zero.

7.6 Results

All results in this section are obtained from a Matlab implementation of our algorithms running on an Intel Duo CPU at 2.40 GHz with 2GB of memory. For fair comparison, also the standard FRW and BEM algorithms have been implemented in Matlab on the same platform. Unless explicitly stated, all FRW results have been obtained from a serial implementation. It should be noted that the FRW (both the standard and our algorithms) are embarrassingly parallel, therefore the computational time is expected to be divided by the total number of processors.

Notice that, to avoid complications associated with paths straying away, in our FRW implementations each structure in the following examples is embedded in a bounding box of dimensions $20\times$ larger than the maximum dimension of the smallest hypothetical box which encloses the entire structure. The bounding box is centered around the center point of the structure. The contribution of any path reaching such bounding box is added to the self capacitance of the conductor from which the path originated.

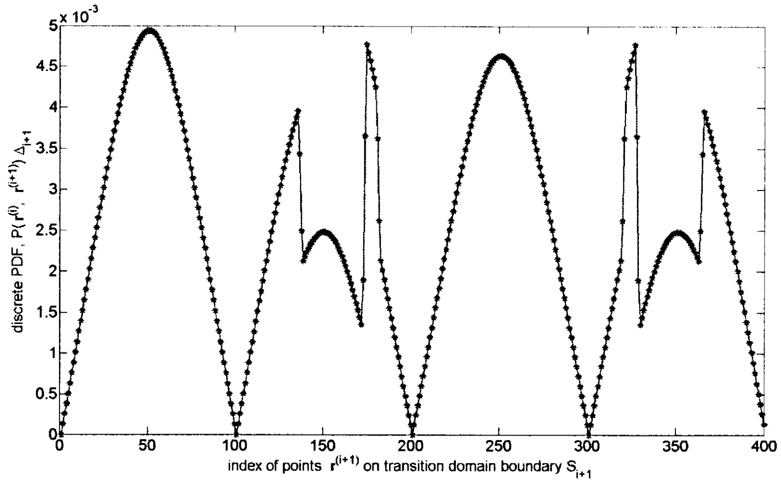


Figure 7-5: Example of a numerically computed PDF $P(\mathbf{r}^{(i)}, \mathbf{r}^{(i+1)})$, where $\mathbf{r}^{(i+1)}$ is the variable parameterizing the contour of a transition square. The discontinuities in the curve correspond to the changes in the dielectric constants.

7.6.1 Generalized FRW for Dielectrics

In this example we show implementation results of the numerically obtained PDF, and the effect of using a non-centered transition domain on the speed of convergence of the GFRW (Algorithm 14 in Section7.3). The geometry is composed of 20 conductors (Figure 8-5) embedded in a multilayered dielectric stack composed of a substrate with dielectric constant $\varepsilon = 11.9$, and 10 successive layers of dielectric constants ranging from 2.2 to 4.4, and finally a half space of free space. The capacitance matrix is computed using both the standard FRW and the GFRW. We have observed that using our new PDF the average path length is reduced from 19 to 6 steps, consequently, the solution time is reduced by a factor of 3. We further observed that the number of unique PDF computations is on the order of 1000, which is very small compared to the total number of random walks ($\simeq 10^5$). This in turn explains why the average walk cost remains approximately the same. Finally, we show in Figure 7-5 a sample PDF computed for a portion of the dielectric stack.

7.6.2 Resistance Extraction

In this example we extract the resistance of the 2-D, 3-port, T-shaped interconnect structure. The normalized lengths of the horizontal and vertical segments of the structure are 15 and

7 units, respectively. The normalized widths are 3 and 2 units respectively. The total number of FRW paths is chosen such that the standard deviation of the computed resistance is less that 2%. We have observed that due to the small size of the geometry, only 134 different PDF computations are required to complete a total of more than 1 million walks. Consequently, the time required to compute the PDFs is negligible compared to the total simulation time. We have further observed that the PDF, associated with transition domains partly subject to Neumann boundary conditions, tend to bias the direction of the path toward the boundary with the Neumann conditions. Constructing the transition domains such that they are centered around the transition points might cause the walk to stagnate because the average size of the domain would approach zero. However, in our approach we avoid stagnation by combining our GFRW with large non-centered transition domains. In this example the simulation time of our GFRW is 10% less than that obtained using just the standard reflection algorithms.

7.6.3 A Large 3D Example with Fill and Multilayers

In this subsection we present a three dimensional example composed of three metal layers (Figure 7-6). The first (lower most) layer has a dielectric constant of 5, and includes three parallel conductors 60nm wide, 60nm thick and 1400nm long. The upper most layer includes 3 parallel conductors (90nm wide, 90nm thick and 1400nm long) orthogonal to those in the lower most layer. This layer has a dielectric constant of 4. The second layer includes fill (metal structures of floating potential) arranged on a regular grid of size 10×10. Each fill is a cube of dimension 70nm. The separation between the fill is 70nm. The total number of conductors including the fill is therefore 106. Our GFRW can easily handle this structure by surrounding the entire layer of fill with a single transition domain, as described in Section 7.4. The probability density matrix associated with the fill transition domain is computed in 5minutes using finite difference discretization.

Three elements contribute to a significant performance improvement of our GFRW over the standard FRW. First, the standard FRW needs to compute the complete 106×106 capacitance matrix (including the fills), while our GFRW can compute directly the smaller

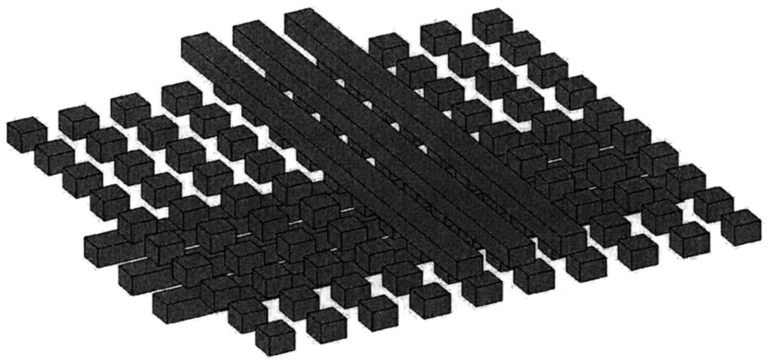


Figure 7-6: A 3D bus structure including fill and multilayers.

6×6 matrix of interest between the wires. Second, the standard FRW would be forced to compute the same complete 106×106 capacitance matrix even in the those cases when only a few or even one single coupling capacitance between two wires is desired, while our GFRW can compute individually any required wire capacitance. For instance, when computing the capacitance between the central wire on the top layer and the central wire in the bottom layer, for the same final accuracy of 10%, our GFRW is around $8000\times$ faster than the standard FRW. Third and most importantly, for any given accuracy, the standard FRW needs orders of magnitude more paths to compute the wire capacitances shielded by intermediate fill, while our GFRW simply bypasses the fill altogether. For instance, when choosing a final target accuracy of 2% our GFRW had no problem achieving the target in 24 minutes (as compared to an estimated 4 months required by the standard FRW).

In order to compare our GFRW to BEM we have implemented a dedicated BEM using the free-space Green's function, discretizing the interface between layers, and handling metal fills by enforcing the charge neutrality condition as described in Section 6.3 or [83]. The performance of the Matlab implementation of our GFRW (24min) is just minimally better than the performance of our Matlab implementation of the dedicated BEM (25min). The computed results match up to 2%.

Chapter 8

Variation-Aware Floating Random Walk

In this chapter we argue that the floating random walk (FRW) algorithm [13] can be modified in order to efficiently extract the capacitance/resistance of a very large number of similar structures (configurations) in a time almost completely independent of the number of configurations. With similar structures we primarily mean structures constructed by introducing non-topological variations (variations defined by dimensional changes) to a nominal structure, or structures constructed by applying small topological variations to a nominal structures (such as adding a small number of conductors to an existing structure).

We present a new efficient variation-aware extraction framework based on recycling the paths of the FRW. Two important and general applications are provided to prove the validity of such algorithm, namely sensitivity and structured incremental analysis. Before proceeding with our variation-aware algorithm we will give several definitions.

Definition 8.0.1. *A "valid transition domain" is a domain for which the relation between the potential of any internal point and the potential of the boundary points can be described by a proper transition probability function. A generalized form of such domains was introduced by Theorem. 7.1.1 in Section 7.1.*

Definition 8.0.2. *A "valid path" is a path in which every transition point is bounded by a valid transition domain. In addition, the path must terminate at a conductor surface. If it does not terminate at a conductor surface, then it is called "incomplete".*

Definition 8.0.3. *The "path dependence list" is the set of conductors that constrain the* *transition domains of the path or that can potentially overlap with one or more transition domains of the path, once altered by some allowed parameter perturbation.*

Definition 8.0.4. *A set of "similar geometries" refers to a parameterized template of a geometry. When the parameter space is sampled, and a particular geometry is instantiated, such resulting geometry is referred to as a "configuration".*

The objective of the next subsection is to develop a general framework for using FRW in variation-aware extraction. More precisely, the goal is to efficiently compute the capacitance matrices of all configurations of a set of similar geometries.

8.1 Main Algorithm for Path Recycling

One of the main advantages of the floating random walk algorithm is the fact that the algorithm is inherently local. This locality is twofold. First, the FRW computes the unknown at a very specific location (as opposed to discretization based methods in which the unknown everywhere is computed). Second, each floating random walk path typically explores only a small part of the geometry. In this chapter we are primarily interested in the second type of locality, which implies that each single path has generally a very sparse path dependence list, i.e. the number of conductors in the path dependence list is very small compared to the total number of conductors as will be shown in an example in Section 8.4.2 (Figure 8-8). Such a property is even more emphasized in typical VLSI structures with a large density of conductors, where the paths are typically very short and confined. More specifically, assume that one were to compute the capacitance of a given nominal configuration. Now modify the size of some of the conductors by perturbing some of their parameters. One can observe that many of the paths used in the nominal configuration do not depend on the perturbed conductors. Consequently, such paths can be mapped to the new configuration and be completely reused to compute the capacitance of the new configuration.

Even the paths that depend on a perturbed conductor can be partially reused in the new configuration. Specifically one can truncate them and complete them by resuming the walk from the first non-reusable transition domain affected by the change in configuration. Since the number of paths that require updating is very small compared to the total number of paths, our path recycling FRW algorithm will obtain almost instantly the solution for the new configuration. The complete procedure is formalized in Algorithm 15.

Notice that all the paths associated with the capacitance extraction of a given configuration need to be statistically independent. However, different configurations do not require to have statistically independent paths. Moreover, in some special cases, such as sensitivity analysis, where the output depends on the difference between the capacitances of two different configurations, sharing the same paths is not only an advantage but even a necessity to ensure numerical accuracy (see Theorem 8.2.1).

Algorithm 15 Path Recycling

```
1: while not all configurations tagged do
2:    Tag one of the untagged configurations
3:    repeat
4:       Generate a path for the last tagged configuration
5:       for each untagged configuration $k$ do
6:          Map current path to configuration $k$
7:          if not valid in configuration $k$ then
8:             truncate and complete path
9:          end if
10:         Use path to update capacitance matrix of configuration $k$
11:      end for
12:   until convergence of the last tagged configuration
13: end while
```

In our experiments we have observed that in many cases the number of paths that need to be truncated and completed is actually very small. Furthermore, in many cases there are efficient ways to determine the sequence in which the configurations are tagged based on some measure of similarity, such that the majority of paths are reused between similar configurations.

8.1.1 Memory Management

One can observe that Algorithm 15 avoids having to calculate and store *all* the paths for a configuration and then map them and check them for every new configuration. Algorithm 15 adopts instead a much more efficient "path-by-path" idea. That means that a path is first generated, then mapped to all configurations, then checked for validity and finally updated if necessary. Once recycled in all configurations, the path is discarded and one moves on to another path. The main advantage of solving for all configurations simultaneously is that the algorithm does not require additional memory utilization to store the details of every simulated or re-simulated path (e.g. transition points, transition domains, and the conductors constraining the transition domains of every path). Consequently, the variation-aware FRW preserves one of the main advantages of the original FRW algorithm, namely, the fact that it requires minimal memory usage.

8.2 Sensitivity Analysis

In this section we demonstrate that the path recycling FRW algorithm can be used to enable efficient finite-difference-based capacitance sensitivity analysis of the capacitance matrix with respect to a large number of independent variations. Recall that to compute the sensitivity of a particular quantity Z (here Z refers to either the capacitance or the resistance), with respect to a parameter P, using the finite difference (FD) sensitivity analysis, one needs first to solve the nominal configuration to compute $Z(P_0)$, then perturb slightly by ΔP the parameter of interest and compute $Z(P_0+\Delta P)$, and finally compute the sensitivity as:

$$\frac{\partial Z}{\partial P} = \frac{Z(P_0 + \Delta P) - Z(P_0)}{\Delta P}. \tag{8.1}$$

As long as the perturbation is small, the finite difference sensitivity for a positive parameter perturbation is the same as for a negative perturbation. Consequently, we are free to choose the most convenient sign. One of the key ideas in this section is to always choose geometrical perturbations that "reduce" the size of the conductors as opposed to "increasing" them (as demonstrated by the ΔP shift in the lower conductor k in Figure 8-1). In this case the computational domain occupied by the transition domains during the computation of the nominal capacitance is simply extended, and therefore every single one of the transition domains and random walk sequences can be reused in the perturbed configuration without any truncation. However, paths that were previously terminating at the perturbed surface of

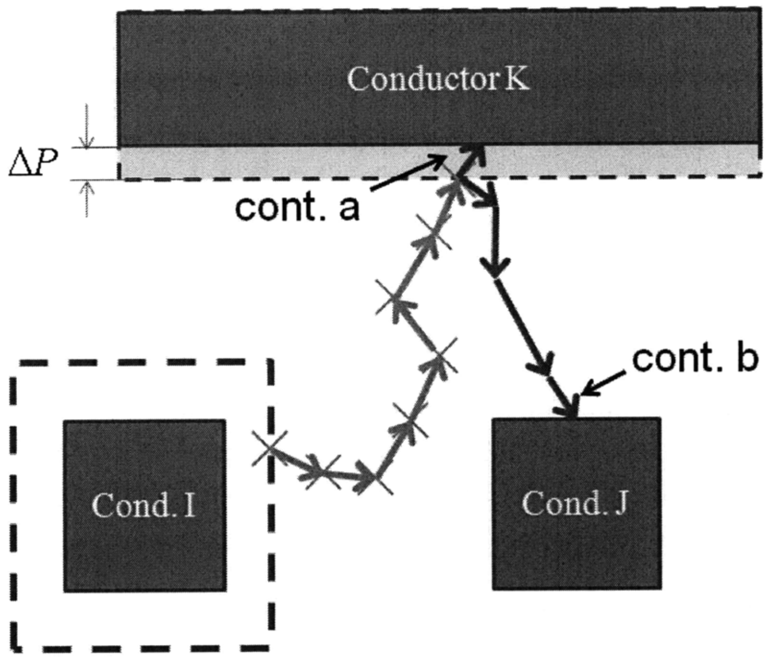


Figure 8-1: Path continuation following small perturbation in conductor K. Most path proceed as path (a). A small number of paths (proportional to change in capacitance) proceeds as path (b)

a conductor are now obviously not touching it in the perturbed configuration, and therefore must be continued until they stop at a conductor in the perturbed configuration. This indicates that a capacitance difference between the nominal and the perturbed configurations can be observed by the FRW only if at least one path in the nominal configuration terminates at one of the conductor surfaces belonging to the subset of surfaces that will undergo perturbation.

Algorithm 16 summarizes our proposed procedure for computing the sensitivity matrix element $\frac{\partial C_{ij}}{\partial P_k}$ of the capacitance vector representing the capacitances between a specific conductor i (target) and all other N conductors in the system $j = 1, 2, \ldots, N$, with respect to the set of parameters $\{P_k : k = 1, 2, \ldots, K\}$. In such algorithm we compute incrementally and simultaneously all the capacitances C_{ij}^k, where in configuration k we perturb parameter k and $k = 0$ corresponds to the nominal unperturbed configuration.

Algorithm 16 Combined Nominal and Sensitivity FRW Solver

```
1: Generate $K$ configurations by changing parameter $P_k$ by $\Delta P_k$ (conductor dimensions are "reduced" as opposed to "increased")
2: Initialize $C_{ij}^k \leftarrow 0$ for all conductor pairs $(i,j)$ and configurations $k$.
3: Initialize path counter $m \leftarrow 0$.
4: repeat
5:    $m \leftarrow m + 1$
6:    Compute one path of FRW for the nominal configuration $k = 0$ starting from conductor $i$
7:    Let $j$ be the index of the conductor at which the path terminated.
8:    Add value of path to the nominal capacitance $C_{ij}^0$
9:    for each perturbed configuration $k$ = 1 : K do
10:       if path terminates at an unperturbed surface of $j$ then
11:          Add value of path to capacitance $C_{ij}^k$
12:       else
13:          Continue path (e.g. path 2 in Figure 8-1) until it terminates on some conductor $l$
14:          Add value of continued path to $C_{il}^k$
15:       end if
16:    end for
17: until desired accuracy achieved
18: for each perturbed parameter $k$=1:K do
19:    for each conductor $j$=1:N do
20:       $\frac{\partial C_{ij}^k}{\partial P_k} = \frac{C_{ij}^k - C_{ij}^0}{m \Delta P_k}$
21:    end for
22: end for
```

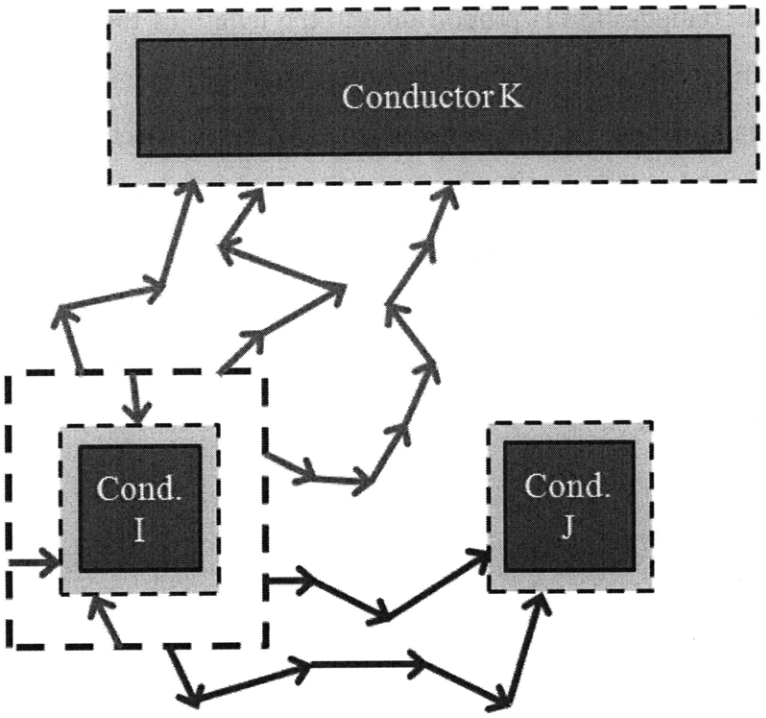


Figure 8-2: Worst case scenario. All the edges are perturbed, consequently all paths have to be continued

8.2.1 Complexity Estimation

In this section we show that for all practical purposes (not from a formal complexity analysis) the complexity of Algorithm 16 is independent of the number of parameters K. Let us assume the worst case scenario where all conductors in the configuration are perturbed as described in the beginning of this Subsection 8.2 (see Figure 8-2). This means that all the random walk paths will undergo step 13 in Algorithm 16, which can be interpreted as a random walk starting at a distance ΔP_k away from the surface of conductor j, and continuing until it terminates on any conductor and in particular, most likely on conductor j itself (path a in Figure 8-1). The change in the capacitance value is intimately related to the number of paths that terminate at a conductor different from j (path b in Figure 8-1). Since the difference between the capacitance of the nominal and perturbed systems is typically very small, it is very reasonable to expect (and we have actually observed experimentally) that the majority of such paths terminate back at conductor j in one iteration (path a in Figure 8-1), or few more iterations in rare cases. Since the incremental cost of computing the capacitances of the perturbed configuration is proportional to the length of the continuation paths, the total cost of computing the finite-difference-based sensitivity matrix is not more than $2\times$ the cost of solving just the nominal capacitance. This very conservative estimate indicates that our method, despite being finite difference based, has a computational complexity that is independent of both the number of parameters (unlike standard finite-difference method) and the number of output capacitances (unlike standard adjoint method). Consequently, our proposed algorithm is computationally superior to both the adjoint method and the standard finite-difference method.

8.2.2 Error Analysis

In this subsection we derive error bounds for the sensitivity analysis as suggested in our algorithm. Define the random variables $C_{ij}^0 = \frac{1}{M}\sum\limits_{m=1}^{M} \omega_m^{(0)}$ and $C_{ij}^k = \frac{1}{M}\sum\limits_{m=1}^{M} \omega_m^{(k)}$, where M is the total number of random walks, $\{\omega_m^{(0)}\}$ and $\{\omega_m^{(k)}\}$ are sequences of independent identically distributed (i.i.d.) random variables associated with the nominal and perturbed random walks, respectively. The mean of both random variables will be denoted as μ_0 and μ_k, respectively. Due to the path continuation process (which results in path sharing), the pairs $(\omega_m^{(0)}, \omega_m^{(k)})$ are correlated. The capacitance sensitivity $\frac{\Delta C_{ij}}{\Delta P_k}$ with respect to a deterministic parameter perturbation ΔP_k as obtained from the FRW is defined as:

$$\frac{\partial C_{ij}}{\partial P_k} = \frac{C_{ij}^k - C_{ij}^0}{\Delta P_k}. \tag{8.2}$$

In the following theorem we summarize some of the properties of our proposed capacitance sensitivity estimator.

Theorem 8.2.1. *The sensitivity, as obtained from (8.2), is an unbiased estimator of the average finite difference sensitivity. Furthermore, the variance of this estimator is given by:*

$$var\left(\frac{\partial C_{ij}}{\partial P_k}\right) = \frac{1}{M}\frac{var(\omega^{(0)}) + var(\omega^{(k)}) - 2cov(\omega^{(0)}, \omega^{(k)})}{\Delta P_k^2}. \tag{8.3}$$

Proof. Complete proof in Appendix B.6. □

We observe that the standard deviation of the estimator decreases asymptotically as $O(\sqrt{M})$. We further note that the correlation between the different paths enhances the accuracy of the sensitivity estimator. Note that the sample covariance is computed from the Pearson product-moment correlation coefficient:

$$cov(\omega^{(0)}, \omega^{(k)}) = \frac{1}{M} \sum_{m=1}^{M} \omega_m^{(0)} \omega_m^{(k)} - \mu_0 \mu_k, \tag{8.4}$$

where μ_0 and μ_k are the average of the nominal and perturbed random walks respectively. Notice, that computing the sample covariance is computationally very cheap, $O(M)$.

8.3 Large Variational Analysis using Quadrature Sampling

Another example where our path recycling algorithm proves to be really efficient is the solution of a large set of configurations constructed by sampling the parameter space using very specific techniques (e.g. tensor or sparse grids as in Appendix A.5), as opposed to general random sampling. This case is more complicated than the sensitivity analysis since the involved variations are large and cannot be assumed infinitesimal. Furthermore, in this case conductors must both be allowed to shrink and to expand. Obviously when conductors are decreased in size, the same exact method used for the sensitivity would work just fine. However, when instead a given conductor is increased in size, any path that includes such a conductor in its path dependence list will become at least partially useless for the perturbed configuration.

In this section we propose a way to select an optimal sequence of configurations so that path recycling is maximized when such configurations are constructed using sparse grid sampling. Assume that the total parameter set describing the geometrical variations is composed of K parameters. Assume that a set of configurations is constructed based on sampling the parameter space using Smolyak algorithm. Assume B_0^K is the nominal configuration. This nominal is constructed by taking the union of all conductors in all configuration (see Figure 8-3). The nominal configuration is also referred to as the master configuration. Any configuration can be constructed from the master configuration by

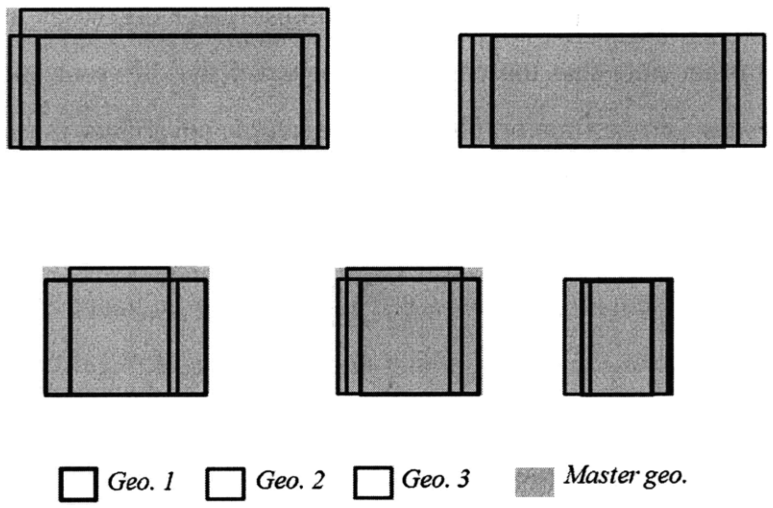


Figure 8-3: Nominal configuration, also called master, constructed by taking the union of all conductors in all configuration.

shrinking the conductors.

Assume $B_{\mathbf{j}}^{K}$ is the subset of configurations in which the parameters indexed by the j-tuple $\mathbf{j} = (i_1, i_2, \ldots, i_j) \in \{1, \cdots, K\}^j$ are different from their nominal values. In other words, $B_{\mathbf{j}}^{K}$ would contain configurations of order j, i.e. only j parameters are allowed to change at the same time. In Algorithm 17 we propose to simulate the configurations in a top-down (breadth-first) fashion, i.e. starting from the nominal configuration $B_{\mathbf{0}}^{K}$ and completing the configurations according to their order $B_{\mathbf{1}}^{K}$, $B_{\mathbf{2}}^{K}$, $\cdots$, $B_{\mathbf{K}}^{K}$ (see Figure 8-4). The partitioning operation in steps 2 and 8 in Algorithm 17 consists of dividing the paths into groups $\wp_{i_l}$, such that each group includes all the paths that need to be at least partially recomputed when parameter i_l is perturbed. Note that these groups are not mutually exclusive. Steps 6 and 10 in Algorithm 17 implicitly assume that the paths in $\bigcup\limits_{k=1}^{K} \wp_{i_k} \setminus \bigcap\limits_{l=1}^{j} \wp_{i_l}$ can be reused to compute the capacitance of the current configuration $B_{\mathbf{j}}^{K}$, as proven by the following proposition

Proposition 8.3.1. *All paths in the set* $\bigcup\limits_{k=1}^{K} \wp_{i_k} \setminus \bigcap\limits_{l=1}^{j} \wp_{i_l}$ *can be reused when simulating configuration* $B_{\mathbf{j}}^{K}$ *within Algorithm 17.*

Proof. If a path $p \in \bigcup\limits_{k=1}^{K} \wp_{i_k} \setminus \bigcap\limits_{l=1}^{j} \wp_{i_l}$, then it belongs to at most m of the groups $\wp_{i_l}$ defined by the partition operation in step 2, where $m < j$. Therefore based on the breadth-first ordering structure of Algorithm 17 the path p was already simulated in a configuration of order m and can be reused to populate the random walks of B_j^K. □

Algorithm 17 Optimal FRW for computing configurations based on grid sampling

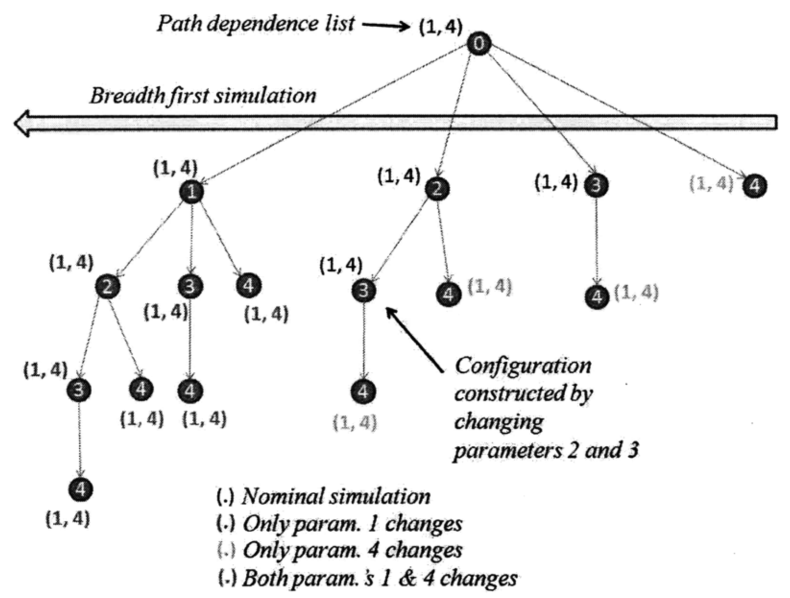
```
1: Simulate the nominal configuration $B_0^K$.
2: Partition paths of nominal configuration $B_0^K$ into groups $\wp_{i_l}$.
3: for configuration order $j = 1 : L$ do
4:   for each configuration in $B_j^K$ do
5:     for each path $p \in \bigcup_{k=1}^{K} \wp_{i_k}$ do
6:       if $p \in \bigcap_{l=1}^{j} \wp_{i_l}$ then
7:         Recompute path $p$.
8:         Add recomputed path to appropriate groups.
9:       end if
10:      Add value of path $p$ to the appropriate capacitance of configuration $B_j^K$
11:    end for
12:  end for
13: end for
```

From Proposition 8.3.1 we observe that the number of resimulated paths (i.e. the cardinality of the set $\bigcap_{l=1}^{j} \wp_{i_l}$) is less than $\min_{l=1,\dots,j} N_{i_l}$, where N_{i_l} is the cardinality of $\wp_{i_l}$. Consequently, the number of re-simulated paths *decreases* as the number of varying parameters increases.

To the best of the authors' knowledge, the presented algorithm is the only variation-aware extraction algorithm, for which the incremental computational effort required to solve high order perturbations is strictly non-increasing as a function of the perturbation order.

8.4 Results

All results in this section are obtained from a Matlab implementation of our algorithms running on an Intel Duo CPU at 2.40 GHz with 2GB of memory. For fair comparison, also the standard FRW and BEM algorithms have been implemented in Matlab on the same platform. Unless explicitly stated, all FRW results have been obtained from a serial implementation. It should be noted that the FRW (both the standard and our algorithms)



Path dependence list
(1, 4)
Breadth first simulation
Configuration constructed by changing parameters 2 and 3
(.) Nominal simulation
(.) Only param. 1 changes
(.) Only param. 4 changes
(.) Both param.'s 1 & 4 changes

Figure 8-4: Configuration arranged in a tree structure. Simulation sequence is breath first.

are embarrassingly parallel, therefore the computational time is expected to be divided by the total number of processors.

Notice that, to avoid complications associated with paths straying away, in our FRW implementations each structure in the following examples is embedded in a bounding box of dimensions $20\times$ larger than the maximum dimension of the smallest hypothetical box which encloses the entire structure. The bounding box is centered around the center point of the structure. The contribution of any path reaching such bounding box is added to the self capacitance of the conductor from which the path originated.

8.4.1 Capacitance Sensitivity Analysis (small variations)

The effectiveness of our combined nominal and sensitivity FRW solver in Algorithm 16 is demonstrated by computing the sensitivities of a 20 conductors structure (Figure 8-5) to variations in the conductor geometries. Conductor 14 in Figure 8-5 is the target conductor for which we extract the capacitance vector (of length 20). Configuration k is constructed by reducing simultaneously the width and thickness of conductor k by 2%, while keeping all other conductors in their nominal size. The total number of configurations including the nominal is 21. Configuration 21 is the nominal. In Figure 8-6 we compare the capacitances

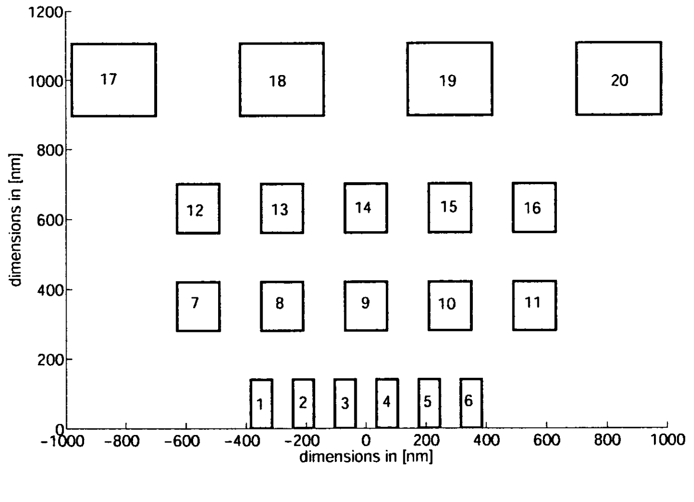


Figure 8-5: Two-dimensional cross-sectional-view of 20 conductors geometry.

$C_{14,19}$ obtained from our FRW algorithm with those obtained from a standard boundary element method (BEM) for all different configurations. The total number of FRW paths is chosen such that the standard deviation of the computed capacitances is less that 1%. From Figure 8-6 we infer that the absolute variation $C^{(i)}_{14,19} - C^{(21)}_{14,19}$ computed using the FRW is within 1% of that computed using the BEM. This is due to the error cancellation resulting from the correlation (path sharing) between the perturbed and nominal configurations. The sample correlation coefficient is approximately 0.8. In Figure 8-7 we plot the percentage relative variation in the capacitance $\frac{C^{(i)}_{14,19} - C^{(21)}_{14,19}}{C^{(21)}_{14,19}} \times 100$ for the different configurations i. The total time required to complete the sensitivity analysis, i.e. compute the sensitivity matrix representing the sensitivities of the 20 output capacitance with respect to the 20 varying parameters, using our variation-aware FRW is only a factor of 1.39 larger than (i.e. 1.39$\times$) the nominal simulation time. Notice that the standard finite difference sensitivity analysis would require 20$\times$ the nominal simulation time since the time grows linearly with the number of parameters. Similarly, the standard adjoint sensitivity analysis would require 20$\times$ the nominal simulation time since the time grows linearly with the number of output capacitances. Furthermore, our FRW implementation is about 2 times slower than our BEM for the nominal structure. Consequently, our FRW sensitivity analysis is about 10$\times$ faster than the BEM finite difference sensitivity analysis and about 10$\times$ faster than BEM adjoint sensitivity analysis [73].

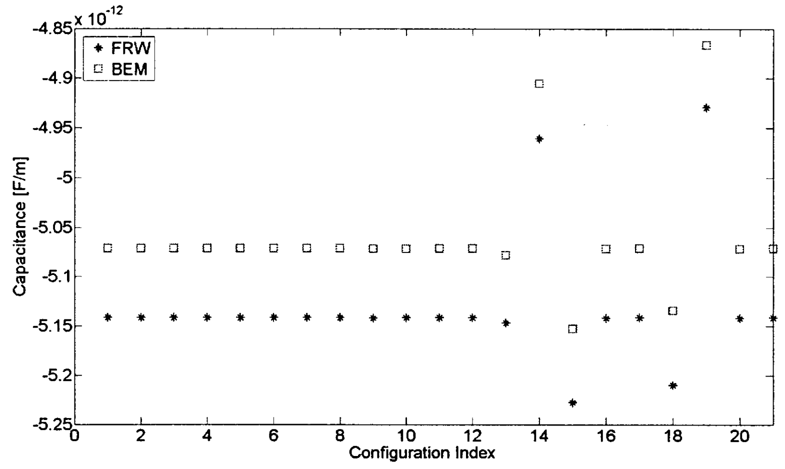


Figure 8-6: Capacitance $C^{(i)}_{14,19}$ obtained from both FRW and BEM algorithms.

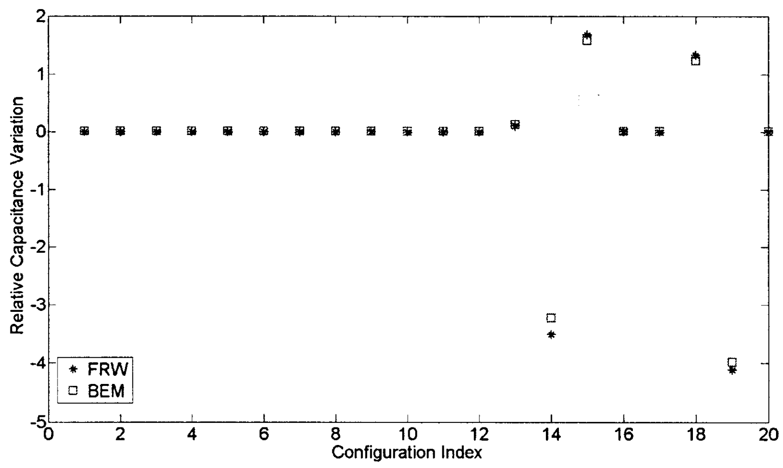


Figure 8-7: Relative capacitance variation $\frac{C^{(i)}_{14,19}-C^{(21)}_{14,19}}{C^{(21)}_{14,19}} \times 100$ demonstrating error cancellation due to random walk correlation.

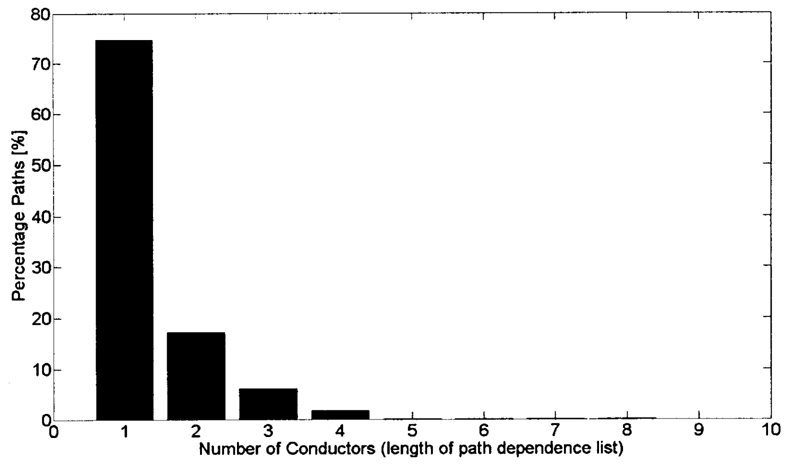


Figure 8-8: Percentage of paths dependent on a particular number of conductors.

8.4.2 Capacitance Variational Analysis (large variations)

In this subsection our general path recycling FRW Algorithm 15 in Section 8.3 is used to efficiently compute capacitances of configurations produced by large geometrical perturbations of the conductors in Figure 8-5.

First, in Figure 8-8 we demonstrate the sparse dependence of the nominal random walk paths on the overall set of conductors. We observe that more than 73% of all paths are responsible for the self-capacitance and therefore end at the target conductor without touching any other conductor. Another 18% of the paths depend only on 2 conductors. We further observe that almost all the rest of the paths depend on no more than 3 or 4 conductors. Consequently, any perturbation affecting more than 5 conductors can be simulated with almost no additional effort. Such sparse dependence constitutes the fundamental strength of the FRW.

We first generate 9 different configurations by expanding conductors 13 and 15 (i.e. the right and left conductors surrounding conductor 14) by factors of $(0, 0)$, $(12.5\%, 0)$, $(25\%, 0)$, $(0, 12.5\%)$, $(0, 25\%)$, $(12.5\%, 12.5\%)$, $(12.5\%, 25\%)$, $(25\%, 12.5\%)$ and $(25\%, 25\%)$, respectively. The accuracy of the variational analysis presented in Section 8.3 is demonstrated in Figure 8-9 by comparing the capacitances $C_{18,14}$ and $C_{13,14}$ obtained from our algorithm with those obtained from the standard boundary element method. The accuracy is better than 5% for all configurations. Furthermore, in Figure 8-10 we show the simula-

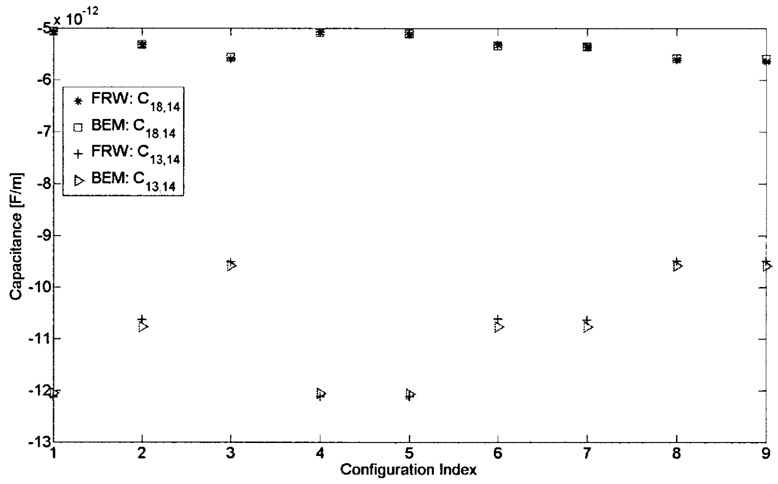


Figure 8-9: Validation of the accuracy of the random walk algorithm variational analysis compared to a robust BEM code.

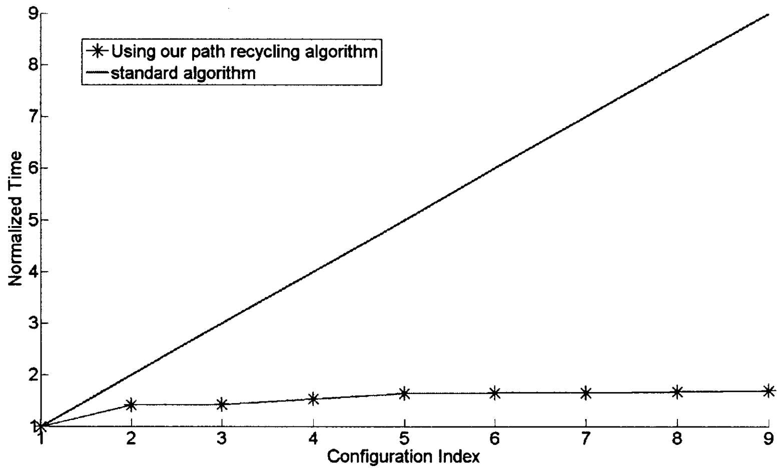


Figure 8-10: Cumulative simulation time for handling additional configurations using FRW as compared to the estimated standard time.

tion time required to compute the capacitance of 9 different configurations using the FRW algorithm as compared to the linear increase in time typical of the standard method without path recycling. The sublinear complexity of our algorithm is clearly demonstrated.

Finally, we validate our variation aware Algorithm 17 in Section 8.3 by computing the capacitance matrix of all configurations produced by sampling the 20-dimensional space using sparse grid constructions with Q=2,3,4, and 5 (Q defined in Appendix A.5). The total number of configurations is 41, 861, 12341, and 135751, respectively. The relation between the number of configurations and the average simulation time per configuration

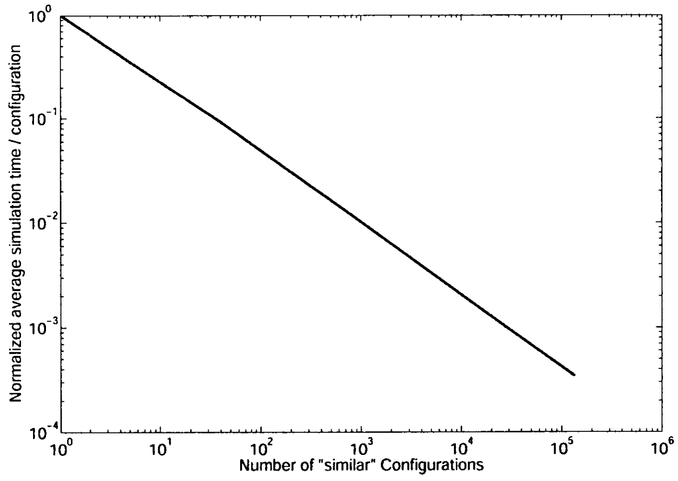


Figure 8-11: Log-log plot demonstrating the reduction in the average simulation time with the increase number of configurations. Configurations constructed based on the 5^{th} order sparse grid.

is shown in Figure 8-11. We observe that the average simulation time per configuration is reduced when the number of similar configurations are increased. Practically speaking, the time required to solve a total of 30,000 configurations is the same time required for solving less than 50 independent configurations, or equivalently the average simulation time per one solve is reduced by $600\times$. The time required to compute all 30,000 configurations using our Matlab implementation of our variation-aware FRW algorithm is about 50min, compared to about 4 hours required by the Krylov-recycling-based BEM. This corresponds to $5\times$ speedup.

Chapter 9

Hierarchical Floating Random Walk Algorithm

With the adoption of restricted design rules (RDR) [47] and ultra regular fabric paradigms [37] for controlling design printability at the 22nm node and beyond, there is an emerging need for a layout-driven, pattern-based parasitic extraction of alternative fabric layouts. Such paradigms are imposing two requirements of micro and macro regularity on the layouts. "Micro" regularity is achieved by restricting shape edges to lie on a restricted design grid that also imposes stringent directionality on shape orientation. "Macro" regularity is achieved by using a very restricted set of litho-friendly logic cells.

Such regularities have motivated using a relatively small number of optimized (litho-friendly and robust) layouts as building blocks to construct any arbitrary structure [37]. These building blocks will be referred to subsequently as "motifs". A designer typically examines different arrangements of the "motifs" in order to choose the "optimal" (in terms of printability, area, and electrical performance) design. Consequently, from an extraction view point, there is a need to develop tools that are aware to *topological* variations. Consider for instance Figures 9-1 and 9-2. Figure 9-1 represents a layout that has been decomposed into 6 motifs. The arrangement of these motifs constitutes one "configuration". Figure 9-2 is a different arrangement of the *same* motifs present in Figure 9-1. This is a second configuration. The number of configurations resulting from such steps is potentially very large, i.e. $O(N_M!)$, where N_M is the number of motifs.

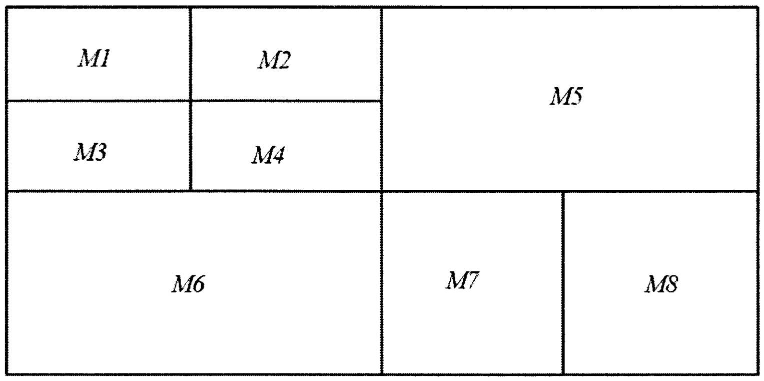


Figure 9-1: Domain decomposition of a hypothetical structure. Structure is composed of 8 different subdomain. Each subdomain with its enclosed shapes constitute a layout motif. The arrangement of these motifs constitutes one configuration.

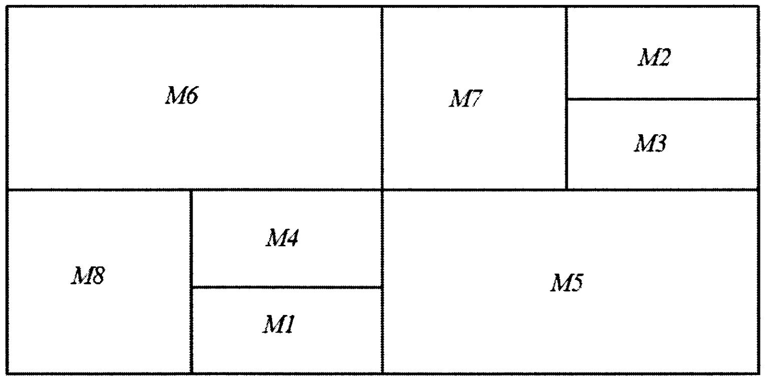


Figure 9-2: An alternative layout configuration made of the same set of motifs defined in Figure 9-1.

In this chapter we present the hierarchical floating random walk algorithm, which we will show can compute the capacitances of *all* $O(N_M!)$ configurations in a highly efficient manner. The efficiency of the algorithm is near-optimal in the sense that it is practically independent of the number of configurations.

9.1 Hierarchical Floating Random Walk (HFRW)

Any hierarchical algorithm is composed of three fundamental steps, namely

1. Domain decomposition: the computational domain is partitioned into smaller subdomains (building blocks)

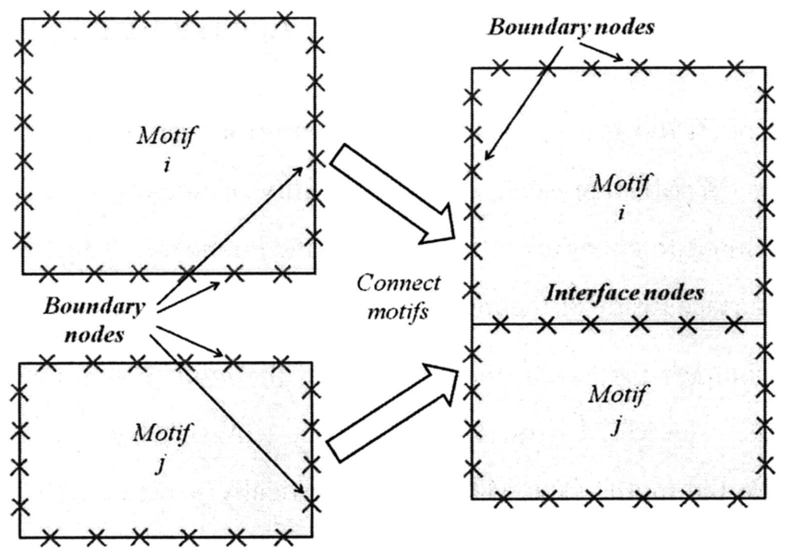


Figure 9-3: Process of connecting two different motifs. Some boundary nodes are transformed into interface nodes. In resulting combination, each node has a unique global index and possibly multiple local indices (one for each motif it belongs to).

2. Resolving the local interactions: each domain is computed independently

3. Resolving the global interactions: domains are connected together

Different algorithms differ by the details of the last two steps. In the following method we will resolve the local interactions by computing a Markov Transition Matrix (MTM) for each domain. Furthermore, we will resolve the global interactions by constructing a large Markov Chain from the computed MTMs and simulating such a chain using random walk methods.

9.1.1 Domain Decomposition

The first step of our HFRW is a domain decomposition, in which we partition the domain into motifs (see Figure 9-4). For the sake of presenting our idea, motifs are assumed to be relatively independent structures, that contain dense metal configurations, and that have regular interfaces that can be easily recomposed together to construct different configurations. In practical extraction flows, such motifs will be determined by the technology and the associated standard library (especially with the employment of fabric-aware design methodologies, ultra regular fabric paradigms and restricted design rules (RDR)).

9.1.2 Resolving Local Interaction: Markov Transition Matrix

The second step of HFRW is to generate *independently* for every motif a complete Markov transition matrix (MTM), representing the probability of moving from any point on the boundary of the motif to either any other point on the boundary of the motif, or any point on a conductor surface inside of the motif.

In order to compute the MTM, the boundary of the motif is first discretized into N_B smaller segments. The MTM is of size $N_B \times (N_B + N_C)$, where N_C is the number of conductors inside the motif. The MTM is stochastically generated by initiating a large number of paths from each of the center points of the N_B segments. Such paths are allowed to proceed within the motif and only stop when they reach one of the motif boundary points, or a conductor surface inside the motif. Every time a path starting from point i on the motif boundary reaches another point j on the boundary or on a conductor surface (all the points on a single conductor are given the same index), the matrix entry (i, j) is incremented by 1. The final step in the computation of MTM is to normalize the rows of the matrix by dividing each row by its total sum.

The generation of the MTM is very simple since the typical motif is small and dense, and therefore the paths are typically short. Furthermore, since the number of stopping points is large, the average path length is small. Moreover, since every motif can be handled independently, and even every point on the boundary of any motif can be handled independently, the generation of the MTMs is "embarrassingly" parallelizable. Algorithm 18 summarizes the computation of the MTM for a given set of N_M motifs.

Clearly, the proposed algorithm requires storing the MTM of each motif. A large amount of memory would be required if the number of motifs is large and if each MTM is fully dense. None of the two conditions are likely to occur in practical layouts. In particular, if the motif boundary points are numbered judiciously, the Markov transition matrix will be sparse and structured (see for instance Figure 9-8) and often banded with small bandwidth. Such banded structure is expected because distant points are likely to have very small transition probabilities. The MTM storage requirement will therefore be minimal. In fact, even if the MTM is dense, it can easily fit in the local storage of a GPU for a

GPU-based parallel implementation of HFRW.

Algorithm 18 Generation of MTM $\mathcal{T}_k$ for motif $\mathcal{M}_k$ within HFRW

```
1: for each motif $\mathcal{M}_k$ do
2:   $\mathcal{T}_k \leftarrow 0$
3:   for each boundary point $i$ of motif $\mathcal{M}_k$ do
4:     repeat
5:       generate a FRW path starting from point $i$ and directed inside of motif $\mathcal{M}_k$
6:       if path terminates at point $j$ on the boundary then
7:         $\mathcal{T}_k(i,j) \leftarrow \mathcal{T}_k(i,j) + 1$
8:         break
9:       else if path terminates at a conductor $l$ inside motif $\mathcal{M}_k$ then
10:        $\mathcal{T}_k(i, N_B + l) \leftarrow \mathcal{T}_k(i, N_B + l) + 1$
11:        break
12:      end if
13:    until convergence is achieved
14:    $S = sum(\mathcal{T}_k(i,:))$
15:    $\mathcal{T}_k(i,:) \leftarrow \frac{\mathcal{T}_k(i,:)}{S}$
16:  end for
17: end for
```

9.1.3 Resolving the Global Interactions

The third step is the recomposition step, in which the different motifs are combined together to construct different configurations of interest (see Figure 9-2). Notice that when motifs are combined together some of the boundaries of the motifs become interfaces in the constructed configuration. To keep track of the arrangement of the motifs each boundary and interface in the configuration is divided into segments, and each center point of every segment is given a global index (see Figure 9-3). We use simple maps to relate the global index of every interface/boundary point to its local index within every motif it belongs to. Note that boundary points belong to a single motif, while interface points belong to at least two motifs.

Algorithm 19 describes how the capacitance vector of a particular conductor I is extracted using the HFRW. According to this algorithm, the complete HFRW path consists of a standard FRW path inside the first motif and a sequence of Markov transitions between points on the motif interfaces (see Figure 9-4). The Markov transition part of the HFRW

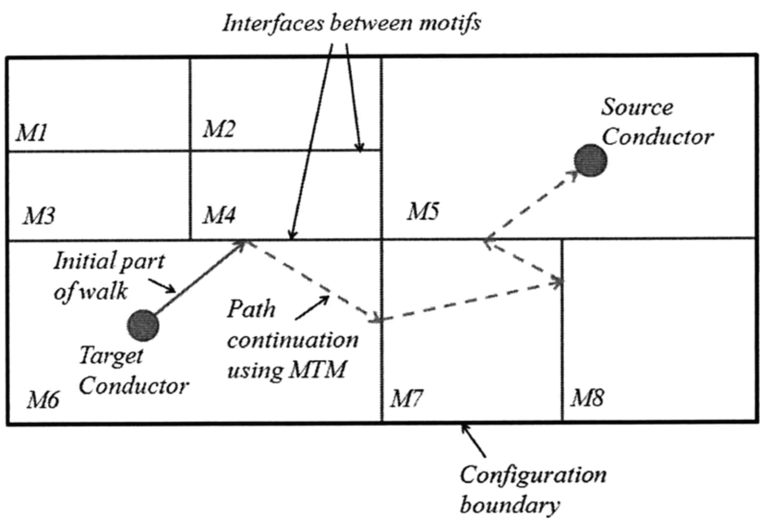


Figure 9-4: Details/Terminology of the HFRW algorithm.

(walk on interfaces) does not require any geometric manipulations, such as transition domain determination, and is therefore extremely efficient. The HFRW path terminates when it reaches either a conductor or a configuration boundary. In the former the HFRW path value is added to the value of the capacitance between conductor I and the conductor at which the path terminates. In the latter the value of the HFRW path is added to the self capacitance of conductor I.

Algorithm 19 HFRW for a given configuration

```
1: repeat
2:    generate a FRW path from conductor $I$ fully contained in its motif $\mathcal{M}$ reaching either
      a conductor inside $\mathcal{M}$ or a point on the boundary of $\mathcal{M}$
3:    if path reached a point on the interface between motifs then
4:       repeat
5:          choose one of the motifs to which the point belongs
6:          Use MTM of new motif to make a transition
7:       until transition reaches a conductor or a configuration boundary
8:    end if
9:    if transition terminated on a conductor then
10:      add value of path (6.9) to capacitance $C(I, L)$, where $L$ is the index of the termi-
         nating conductor
11:   else {transition terminated on configuration boundary}
12:      add value of path (6.9) to self capacitance $C(I, 0)$
13:   end if
14: until convergence achieved
```

9.1.4 Comparison between HFRW and standard hierarchical domain decomposition

The fundamental difference between our hierarchical floating random walk algorithm and the standard hierarchical FEM/BEM algorithms is in the way we resolve the global interactions. The vast majority of other hierarchical algorithms [82, 49, 38, 40] rely on assembling a large linear system of equations in order to resolve the global interactions. Instead, in our algorithm we resolve the global interaction by using the random walk method in order to simulate the large Markov Chain constructed when connecting the motifs together. Consequently, our method completely avoids any large linear systems and scales very well with the size of the problem.

Several advantages follow from such main difference. In particular, our method is embarrassingly parallel, which enables it to take advantages from multithreaded and multicore computer architectures. More importantly, the resolution of the global interaction step using our method becomes computationally extremely cheap. Consequently, our method is ideal for fabric-aware extraction, i.e. extraction of a large set of configurations all constructed from the same set of motifs. This is the subject of the next section.

9.2 Fabric-Aware 3D HFRW Algorithm

In this section we present an algorithm for 3D capacitance extraction of a large number of configurations constructed by different recompositions of a set of motifs. Recall that if we have N_M motifs then there are $O(N_M!)$ possible different configurations. Algorithm 20 summarizes the steps of our proposed approach.

Since all configurations are constructed from the same set of motifs, the MTMs for each motif can be precomputed separately as shown in Step 2. The complexity of this part of the algorithm depends linearly on the number of motifs $O(N_M)$, and does not depend on the total number of configurations.

Step 6 (standard FRW) in Algorithm 20 is independent of the configuration structure, and therefore it is implemented once per motif and reused for all configurations. This step

Algorithm 20 HFRW for all configurations

```
1: for each motif $\mathcal{M}_k$ do
2:    use Algorithm 18 to compute MTM $\mathcal{T}_k$
3: end for
4: for each desired capacitance matrix column $C(I,:)$ do
5:    repeat
6:       generate a FRW path from conductor $I$ reaching either a conductor inside the
         same motif or a point on the boundary of the motif
7:       for each configuration do
8:          use MTMs to walk on interfaces and terminate on a conductor or configuration
            boundary
9:          add value of HFRW path to appropriate capacitance
10:      end for
11:   until convergence achieved
12: end for
```

is very efficient since the standard FRW requires short paths when the boundary of the domain is close to the conductor as in the case of a motif.

The remaining part of each HFRW path depends on the particular recomposition of the motifs, therefore it must be implemented separately for each configuration. Since this part of the algorithm does not involve any geometrical manipulations it is extremely cheap. Consequently, the bottleneck of the algorithm are Steps 2 and 6, and the complexity of our algorithm is almost completely independent of the total number of configurations.

9.3 Theoretical Analysis of HFRW

Many standard "Absorbing Markov Chain" theorems and well known results can be exploited to certify properties of our HFRW algorithm once we show how it is possible to appropriately construct a large Markov Transition Matrix $\mathcal{T}$ for the entire configuration:

$$\mathcal{T} = \begin{bmatrix} \mathbf{Q} & \mathbf{R} \\ 0 & \mathbf{I} \end{bmatrix} \tag{9.1}$$

where $\mathbf{Q}$ is the transition probability matrix between "transient states" (in our case any point on the interface between motifs as shown in Figure 9-5); $\mathbf{R}$ is the transition probability matrix from the transient states to the "absorbing states" (in our case all the conductors

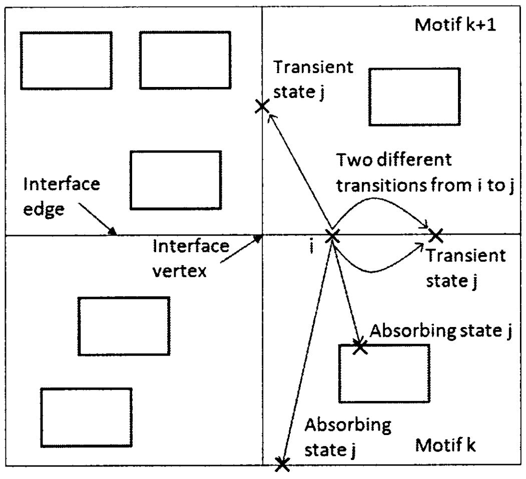


Figure 9-5: Transition to interfaces (transient states) contribute to the **Q** matrix and transition to conductors and configuration boundaries (absorbing states) contribute to **R**.

and any point on the external configuration boundary). Matrices **0** and **I** simply define the behavior of the absorbing states: once an absorbing state is reached the probability to remain in it is one, and consequently the probability to transition to any other state is zero. The upper part [**Q** **R**] of the Markov Transition Matrix $\mathcal{T}$ can be related to the individual Markov Transition Matrices of each motif using the Law of Total Probability

$$\begin{bmatrix} \mathbf{Q} & \mathbf{R} \end{bmatrix}(i,j) = \sum_{\text{each motif } k} P[i \in \mathcal{M}_k] \quad P[i \to j \mid i \in \mathcal{M}_k]$$

or in other words, the sum of the probabilities of choosing motif $\mathcal{M}_k$ in step 5 of Algorithm 19 multiplied by the conditional probabilities of transitioning from point i to point j, given the choice of motif $\mathcal{M}_k$. Notice that

$$P[i \to j \mid i \in \mathcal{M}_k] = \begin{cases} \mathcal{T}_k(i,j) & j \in \mathcal{M}_k \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{T}_k(i,j)$ is the MTM of motif $\mathcal{M}_k$ constructed by Algorithm 18. Also notice that in the simple 2D uniform media case:

$$P[i \in \mathcal{M}_k] = \begin{cases} 1/2 & i \text{ on interface edge of 2D } \mathcal{M}_k \\ 1/4 & i \text{ on interface vertex of 2D } \mathcal{M}_k \\ 0 & \text{otherwise} \end{cases}$$

and in the simple 3D uniform media case:

$$P[i \in \mathcal{M}_k] = \begin{cases} 1/2 & i \text{ on interface surface of 3D } \mathcal{M}_k \\ 1/4 & i \text{ on interface edge of 3D } \mathcal{M}_k \\ 1/8 & i \text{ on interface vertex of 3D } \mathcal{M}_k \\ 0 & \text{otherwise} \end{cases}$$

Having cast our HFRW algorithm as an Absorbing Markov Chain problem it is easy to rigorously answer many legitimate questions using the literature available on that topic [45]. For instance, the following theorem can be used to prove the termination of each HFRW "path" in a finite number of transitions, and to even provide a precise estimate on the average number of transitions before termination.

Theorem 9.3.1. *Assume that HFRW starts at a point i on an interface between motifs (i.e. a transient state), then the average length of the walk on interfaces, or expected number of transitions before reaching a conductor or the configuration boundary (i.e. an absorbing state) is finite and is given by the row sum of the i-th row in the "fundamental matrix"* $\mathbf{N} = (\mathbf{I} - \mathbf{Q})^{-1}$.

Proof. The proof follows directly from Theorem 3.3.5 in [45]. □

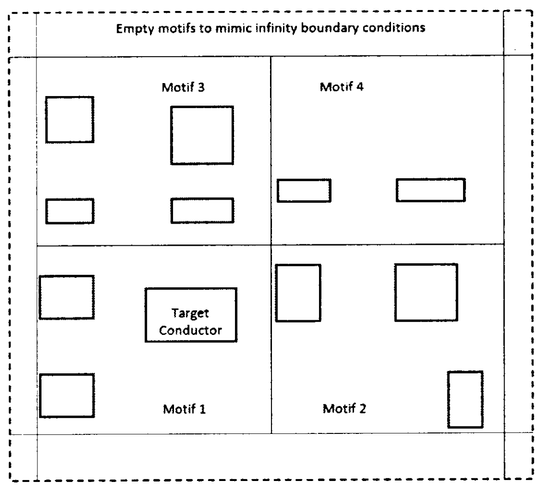


Figure 9-6: Geometry partitioned into different motifs. Empty motifs at the boundary (shown only partially) are used to mimic the infinity boundary condition.

9.4 Results

9.4.1 Accuracy Validation

The first example validates the accuracy of the proposed algorithm. The 2D geometry for this example is shown in Figure 9-6, and is composed of 12 conductors. To implement our HFRW, the geometry is divided into 4 different motifs. In addition, empty motifs are used at the boundary (shown only partially in Figure 9-6) to mimic the infinity boundary condition.

The time required to compute the MTM for all motifs is half the time required to simulate all 12 conductors using the standard FRW. We extracted the capacitance between a target conductor in motif 1 (see Figure 9-6) and all the other 11 conductors using both our HFRW and the standard FRW. The time required for our HFRW is approximately half that required for the standard FRW for the same accuracy. Therefore the time required for our complete algorithm is about the same time $(1.01\times)$ required for the standard FRW. When compared to a standard FRW with 2×10^6 different random paths to guarantee convergence, our approach obtained for all extracted capacitances a 1% accuracy.

9.4.2 A 2D Fabric-Aware Extraction Example

In this example we use the same motifs used in Figure 9-6 (from the previous example 9.4.1) to construct a total of $4! = 24$ different configurations, corresponding to all possible different recompositions of the four internal motifs. The capacitance matrices of each of the configurations are computed using both our HFRW and the standard FRW. All the values of the computed capacitances for each configuration are within 2% of the values computed using the standard FRW. The total time to compute *all* 24 capacitance matrices using our HFRW is about equal ($1.1\times$) to the time required to compute the capacitance matrix of just *one* configuration using the standard FRW. This corresponds to a $21\times$ speedup.

9.4.3 A Large 3D Example

In this subsection we demonstrate that the HFRW can also be used to treat 3D structures very efficiently. The example under consideration is a 5 layer structure (Figure 9-7). Each layer has its own (different) dielectric constant. Two of such layers each contain a total of 100 cubic shaped conductors arranged on a 10×10 grid. The size of each conductor is 100nm. These small conductors represent for instance "metal fill", i.e. small floating conductors inserted in empty regions of the layout to facilitate the planarization. The other three layers each contain 3 parallel long wires of dimensions 100nm$\times$1400nm$\times$100nm. The wires are separated by 100nm. Each of the five layers is 300nm thick. Each layer is treated as a motif. We recompose such motifs to construct a total of 120 different configurations. Note that each configuration will include a total of 209 total conductors. For each configuration we extract four different capacitances. The largest of the 5 MTMs, each associated with one of the 5 layers, has size 1536×1636, and is 95% sparse (Figure 9-8). The time required to compute all 5 MTMs is approximately 15 minutes in a code implemented in Matlab and running on a Intel Duo CPU at 2.4GHz with 2GB of memory. Such time can be significantly reduced by using C/C++ code and by using a parallel implementation of the algorithm. After the MTMs are computed, the subsequent computational time required to solve all possible 120 configurations is 15 minutes in Matlab on the same machine (5min in Matlab on three parallel processes) as opposed to an estimated time of about 1800min

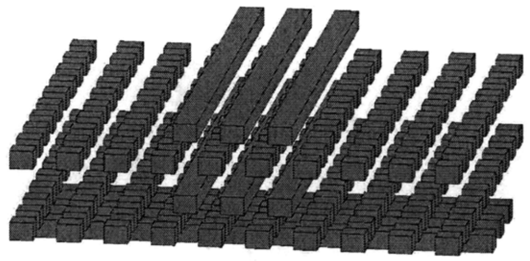


Figure 9-7: A five layers, 209 conductor structure. Each layer is treated as a motif.

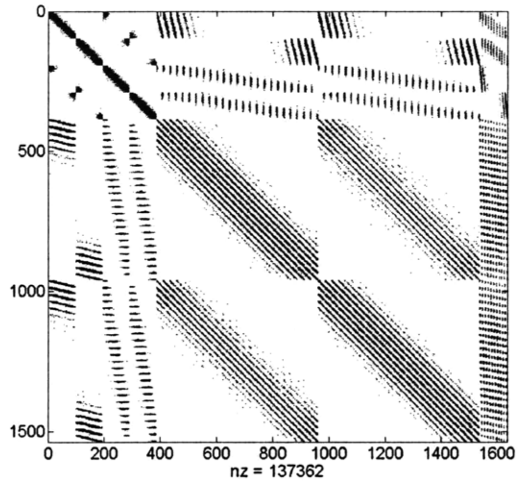


Figure 9-8: A typical sparsity pattern of the Markov Transition Matrix (MTM) for a motif including 100 cubic conductors.

required to solve all configurations using the standard FRW. This corresponds to a $60\times$ speedup.

9.4.4 Verification of Theoretical Analysis

In this subsection we use the 2D example in Subsection 9.4.1 to verify Theorem 9.3.1. We first assemble the global Markov Transition Matrix $\mathcal{T}$ as described in Section 9.3. Then using Theorem 9.3.1 we compute the average path length of any path starting from the boundary of motif 1. The average path length is shown in Figure 9-9 using the red squares. Notice that the x-axis is just the index of the starting point on the boundary of motif 1. We then initiate a total of 5,000,000 FRW paths from points on the boundary of the first motif (at least 1000 paths are initiated from any point). Such paths are allowed to progress until they either reach a conductor surface or a point on the "infinity" boundary. We then

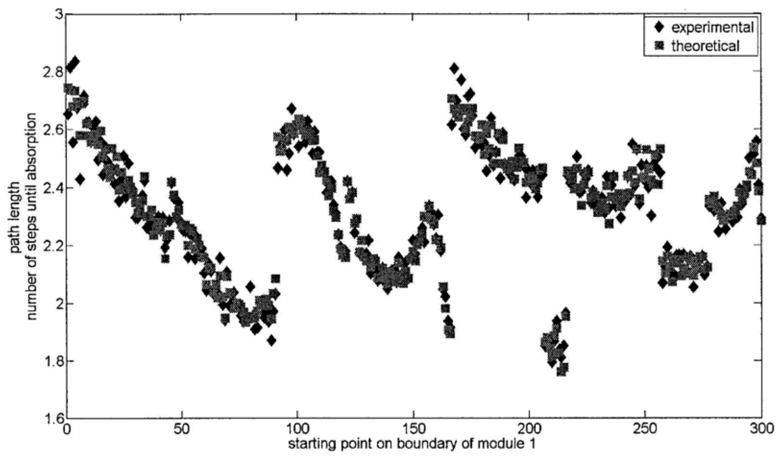


Figure 9-9: Theoretical path length average versus experimental path length average.

compute, for every boundary point, the average length of the FRW paths until termination. This length is plotted using the blue diamonds in 9-9. Extremely good matching is observed between the experimental results and the theoretical estimates. Finally, we observe that the average path length starting from *any* point on the boundary is less that 3, which in turn explains further why the path continuation part is extremely efficient.

Part III

Conclusions and Future Work

Chapter 10

Summary and Comparison of Various Methods

10.1 Summary of Various Methods

In this thesis we have presented a variety of techniques for variation aware interconnect parasitic extraction. Following is a summary of those techniques:

1. The stochastic model order reduction algorithm (SMOR), in Section 3.2, is applicable to the extraction of any electrical property (e.g. resistance, capacitance, inductance and full impedance) in the presence of any type of non-topological variations (width/thickness or surface roughness). It's accuracy is constrained by the initial discretization, i.e. the method does not support adopting the number of the discretization elements based on the values of the varying parameters. The SMOR works best for low/medium dimensional parameter spaces or parameter spaces with very well defined structures. The variations in the computed output quantities (e.g. R, L, C or Z) can be very large, provided they live on some easily defined low dimensional manifold. The SMOR fully exploits "fast" field solver techniques. The SMOR relies for efficiency on the Hermite expansion of the system matrix. The SMOR fully exploits parallelization.

2. The Combined Neumann Hermite Expansion Method (CNHE), in Section 4.2, is

applicable to the extraction of any electrical property in the presence of any non-topological variation. It's applicability is constrained by the convergence of the Neumann expansion, which is only guaranteed if the input variability results in a small change in the system matrix. The main computational advantage of the CNHE is obtained from exploiting the adjoint method, which is only efficient for small dimensional output spaces. Furthermore, the variability of the output quantity (R, L, C, Z) should be accurately described using a low order multivariate expansion in the parameter space, since the complexity of evaluating high order terms is very large. The CNHE exploits fast field solver techniques. The CNHE fundamentally relies on the Hermite expansion of the system matrix. The CNHE can exploit parallelization.

3. The stochastic dominant singular vectors (SDSV), in Chapter 5, is applicable to the extraction of any electrical property in the presence of any non-topological variation. It's accuracy is constrained by the initial discretization, i.e. the method does not support adopting the number of the discretization elements based on the values of the varying parameters. The SDSV is very suitable for large dimensional parameter spaces. The output space can be very complex, provided the output lives on a low dimensional manifold. One important difference between the SDSV and the SMOR is that the manifold of the computed electrical property (R, L, C, Z) is automatically computed by the SDSV while it is manually provided to the SMOR. Variants of the SDSV can fully exploit fast solver techniques. The SDSV relies on computing the Hermite expansion of the system matrix. Moreover, it also requires additional manipulation to identify the dominant direction of the system matrix.

4. The variation-aware FRW (VAFRW), in Chapter 8, is suitable only for capacitance and resistance extraction. The VAFRW is particularly suitable for extracting the large coupling capacitances (including self capacitance) of a given conductor, in the presence of large-dimensional structured edge-based variations, e.g. variations produced by sampling the parameter space using a sparse grid quadrature scheme. VAFRW cannot efficiently handle material property changes. The main advantage of the VAFRW is that its complexity is almost independent of both the structure size

Table 10.1: Summary of the important properties of all developed algorithms

Alg.	Application	Problem Size	Parameter Space Size	Input Variability	Output Space Size	Output Variability
SMOR Chp. 3.2	R, L, C, Z	Large	Medium	cont. by init. disc.	Large	Low dim. manifold
CNHE Chp. 4.2	R, L, C, Z	Large	Large	Small conv. of N.E	Small	Low order
SDSV Chp. 5	R, L, C, Z	Large	Large	cont. by init. disc	Large	Low dim. manifold
VAFRW Chp. 8	C, R self	Very large	Very large	structured weakly coupled	Small	Large
H-FRW Chp. 9	C, R self	Very large	Very large	topological	Medium	Large

and the parameter space size. The VAFRW is embarrassingly parallelizable.

5. The hierarchical FRW (HFRW), in Chapter 9, is suitable for both capacitance and resistance extraction. The HFRW is most suitable for computing the strong coupling capacitances in the presence of topological variations. The main advantage of the HFRW is that its complexity is almost independent of the number of different configurations. The HFRW is embarrassingly parallelizable.

In Table 10.1 we summarize the general properties of the presented algorithms. Notice the use of the following abbreviations: "cont. by init. disc." for "controlled by initial discretization", "Low dim. manifold" for "low dimensional manifold", and "conv. of N.E." for "convergence of Neumann expansion". One can use such a table to choose the most appropriate technique for the application at hand.

10.2 Comparison Results of Various Methods

In this section we compare the performance of the techniques we have presented, namely, the stochastic model order reduction method (SMOR), the combined Neumann Hermite expansion (CNHE), the stochastic dominant singular vector (SDSV), and VA/H-FRW. Notice that we have combined both the VA-FRW and the H-FRW to produce a single efficient algorithm called the VA/H-FRW. All algorithms are tested on the following four examples:

1. Large 3D On-Chip capacitance extraction example in the presence of width and thickness variations. This example is composed of 6 conductors arranged on top of 289 small metal fill as shown in Figure 10-1. The red conductor (in the middle of the second layer) is the target conductor for which we want to compute the capacitance vector. The total number of discretization elements is N=25,788 and the total number of parameters is N_P=295.

2. Large 3D parallel-plate capacitance extraction example in the presence of surface roughness. This example is composed of 2 square plate conductors of size 20 separated by a distance of 1 Figure 10-3. The upper plate is very rough ($\sigma = 0.2$ and $L_c = 1$). The total number of discretization elements is N=21,000 and the total number of parameters is N_P=323.

3. Large 3D impedance extraction in the presence of random geometrical variations. This example (shown in Figure 10-4) is composed of four different inductors. The dimensions of every wire (total of 48) is an independent random variable. For the discretization based methods, the total number of discretization elements is N=10358 and the total number of parameters is $N_P = 48$.

4. Large 3D impedance extraction in the presence of random surface roughness. This example (shown in Figure 10-4) is composed of four different inductors. The upper surface of each conductor is assumed rough. For the discretization based methods, the total number of discretization elements is N=10358 and the total number of parameters is $N_P = 400$.

All algorithms are optimized for computational efficiency. Whenever applicable, we exploit parallelization, computational reuse (e.g. table look-ups), numerical approximations, data compression and any other available implementation tricks (provided the accuracy threshold is met). Table 10.2 summarizes the comparison results. Notice that we are reporting

Table 10.2: Comparison of the performance of the best previous method, SMOR, CNHE, SDSV, VA/H-FRW for four different examples. N for SMOR, CNHE and SDSV corresponds to the total number of discretization elements. P/K for SMOR, CNHE and SDSV corresponds respectively to the total number of parameters and the total number of orthogonal polynomials (dimension of the basis) used to describe the variability. The percentage beside the Time (e.g. 3% and 8%) refers to the accuracy threshold. Reported times are all in hours.

		Previous	CNHE	SDSV	SMOR	H/VA-FRW
Ex.1: 3D-cap.	Time 3%	(600)	(1200)	8	(24)	1
param. var.	Time 8%	60	12	6	12	0.25
N=25K, P/K=295/44K	Memory	20GB	20GB	20GB	20GB	5MB
Ex.2: 3D-cap.	Time 3%	(650)	(260)	12	(70)	6
rough surface	Time 8%	65	13	7	14	2
N=21K, P/K=323/53K	Memory	20GB	20GB	20GB	20GB	20MB
Ex.3: 3D-imp.	Time 3%	10	6	8	7	–
param. var.	Time 8%	10	6	8	7	–
N=10K, P/K=48/1K	Memory	10GB	10GB	10GB	10GB	–
Ex.4: 3D-imp.	Time 3%	(1400)	(720)	12	(56)	–
rough surface	Time 8%	(140)	36	10	28	–
N=10K, P/K=400/81K	Memory	16GB	16GB	16GB	16GB	–

two different times corresponding to 3% and 8% accuracy thresholds. Notice further that the entries in the table are rounded to simplify comparing the results.

From the table and our previous results and discussions, we can draw several conclusions related to the relative performance of the various methods:

1. The VA/H-FRW is the best method to extract the resistance or capacitance in the presence of non-topological variations in the metal (not in the material properties). Such methods scale extremely well with the size of the problem and with the dimension of the parameter space.

2. The H-FRW is the best method to extract the resistance or capacitance in the presence of topological variations. Such method scales extremely well with the number of independent configurations.

3. The SDSV is the best method to extract the impedance in the presence of large di-

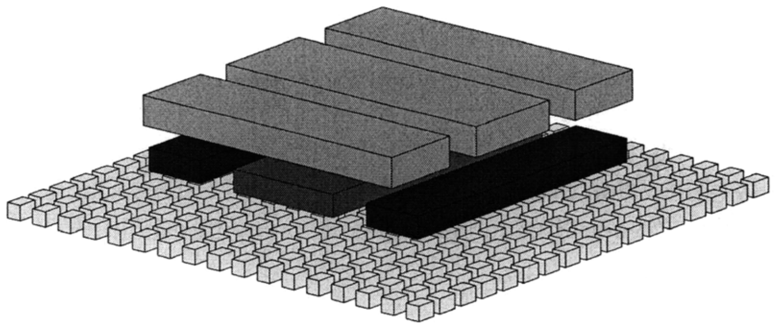


Figure 10-1: Large Capacitance Extraction Structure. Discretized using a total of 25788 different panels.

mensional non-topological variations. Despite the need for a large setup time, such method scales extremely well with the dimension of the parameter space.

4. The CNHE is the best method to extract the impedance in the presence of small dimensional non-topological variations, provided that the output is accurately described using a low order multivariate expansion in the parameter space. Such method requires solving linear systems which only involve the average matrix. Furthermore, such methods utilize the adjoint concepts to significantly reduce the computational effort.

5. The SMOR is the best method to extract the impedance in the presence of small dimensional non-topological variations, provided that the output is a high order multivariate expansion in the parameter space. Such method fully exploits fast field solver techniques and therefore significantly reduces the solution time of the independent linear system solves.

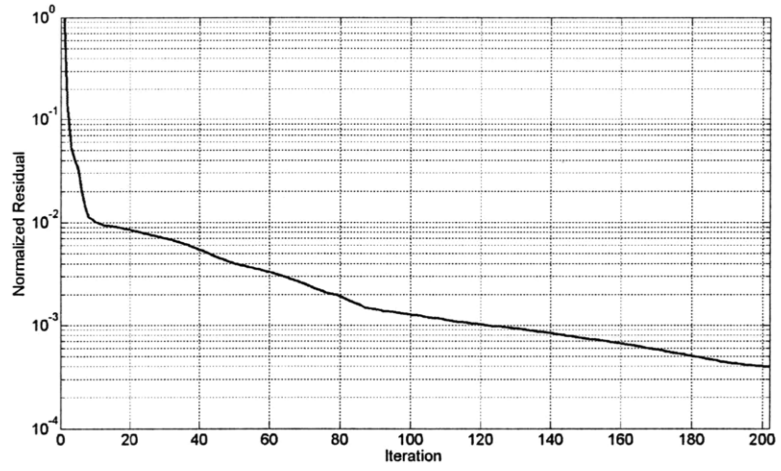


Figure 10-2: Residual versus iteration count as computed by the SDSV for the 295-conductor (parameter-variation) capacitance extraction example.

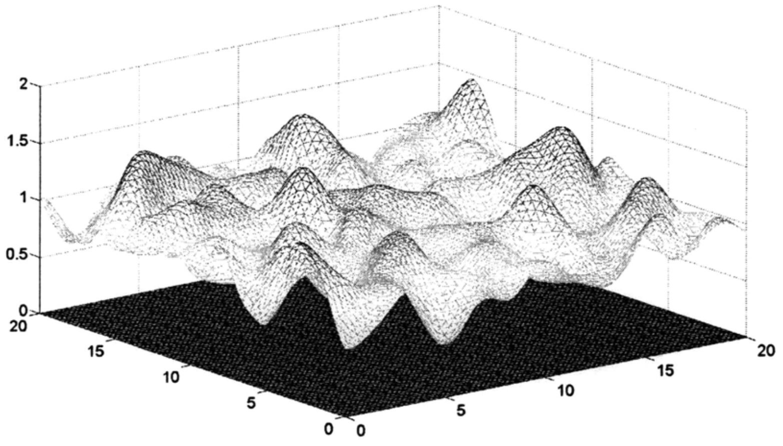


Figure 10-3: Large Parallel Plate Capacitance Extraction Structure. Discretized using a total of 21,000 different panels.

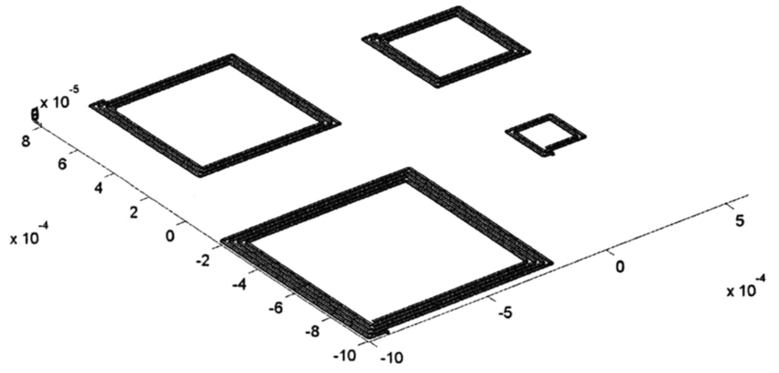


Figure 10-4: Array of 4 Inductors.

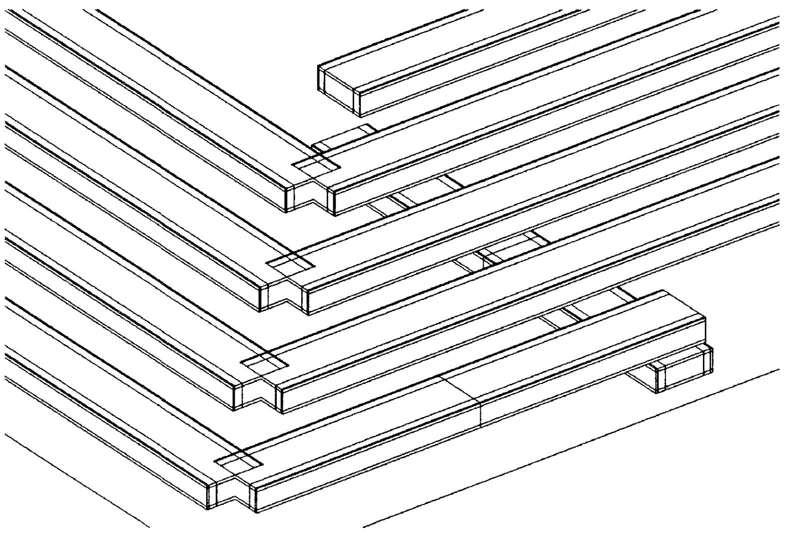


Figure 10-5: Details of the fine conductor volume discretization (4-inductor array).

Chapter 11

Conclusions

In this thesis we have presented a variety of algorithms and techniques both to enhance the efficiency of existing variation-aware extraction algorithms and to develop new variation-aware extraction algorithms. In particular, we have proposed two important mathematical machineries to facilitate applying any Hermite polynomial based algorithms in electrical extraction applications:

1. In Section 4.1 we have proposed a new inner product to compute the coefficients of the projection of the multivariate system matrix elements on the space of multivariate Hermite polynomials in a complexity that is *independent of the size of the parameter space*. Our algorithm reduces the complexity of computing the multivariate Hermite expansion of the system matrix by more than 5 orders of magnitude.

2. In Section 5.1.3 we have presented a new algorithm to compress the stochastic system matrix $\mathbf{A}(\vec{\eta})$. Our algorithm relies on the physical interpretation of the rows of the system matrix. The proposed algorithm is used only once as a preprocessing step to compress the entire system matrix. Using our new algorithm the number of basis functions used to represent a matrix row is typically reduced by more that 3 orders of magnitude.

We have also presented two important enhancements to existing discretization-based variation-aware extraction algorithms, in particular

1. In Chapter 3 we have increased the efficiency of non-intrusive stochastic solvers by exploiting the similarity among the different linear systems of equations generated by sampling the parameter space. In particular, in Section 3.1 we have proposed the Galerkin Krylov for Recycling (GKR) method, which reduces the complexity of solving a large number of similar linear systems by efficiently recycling the Krylov subspace. Our GKR provides up to a factor of 5 reduction in the computational time. In Section 3.2 we have also proposed a projection based model reduction approach (SMOR), which guarantees that the statistical moments computed from the reduced model are exactly equal to those computed from the full original model. We have formalized a theorem which details the sufficient conditions which must be imposed on the projection matrix, such that the statistical moments are equal whether computed from the original model or the reduced one. The proposed SMOR reduces the computational time of the best non-intrusive method by about 1 order of magnitude for problems with large dimensional parameter spaces.

2. In Section 4.2 we have proposed a Combined Neumann Hermite Expansion (CNHE) method. This new method combines the advantages of both the Neumann expansion (size of resulting system is the same as nominal system) and the stochastic Galerkin method (ease of computing statistics). Furthermore, the method can utilize fast solvers. The method uses an adjoint-like approach to reduce the complexity of solving complex variation-aware extraction problems by several orders of magnitude. The CNHE is particularly suitable for problems characterized by densely coupled parameter spaces (e.g. extraction in the presence of surface roughness). Unlike the standard Neumann Expansion method, the CNHE facilitates computing a Hermite expansion for the output quantity. Furthermore, for large dimensional problems the CNHE is typically more that three orders of magnitude faster than the standard Neumann Expansion method. Moreover, the CNHE can compute the solutions of densely coupled problems that are not solvable using the standard stochastic Galerkin method.

Finally, we have proposed three new variation-aware extraction algorithms:

1. In Chapter 5 we have presented the stochastic dominant singular vectors method (SDSV). The method relies on finding an optimal reduced basis in order to represent the unknown solution. We have also presented a couple of (computationally very efficient) variants of the main algorithm, which rely on finding suboptimal (rather than an optimal) basis. The SDSV and all its variants have optimal scaling with the dimension of the parameter space, in the sense that the complexity is almost independent of the size of the parameter space. The main disadvantage of the method is that it requires a relatively large setup time. However, it remains (to the best of our knowledge) the *only* discretization-based method, which can handle very large size problems, e.g. problems in which the total number of discretization elements $N > 20,000$ and the total number of independent parameters $N_P > 200$.

2. In Chapter 7 we have generalized the standard FRW such that it efficiently handles multilayered media, floating potential metal fills, and mixed Dirichlet-Neumann boundary conditions. Furthermore, in Chapter 8 we have developed the VAFRW, which is a method that relies on path recycling in order to efficiently extract the capacitance of a very large number of "similar" configurations. More particularly, the VAFRW is best suited for extracting the self and large coupling capacitances of the same target conductor in all similar configurations. With similar configurations we mean, configurations that are constructed by applying edge perturbations (width, height, and relative distance variations) to some nominal configuration. The VAFRW is most efficient if the configurations are constructed by structured sampling of the parameter space, e.g. sampling the parameter space using some quadrature scheme. The VAFRW is optimal in the sense that its complexity is almost independent of the dimension of the parameter space. When computing the variation-aware capacitance of a structure in the presence of a large number of varying parameters, the VAFRW provides more than 4 orders of magnitude reduction in computational time as compared to the standard FRW algorithm. The VAFRW *cannot* be used for impedance extraction.

3. In Chapter 9 we have developed the Hierarchical-FRW algorithm for variation aware

resistance/capacitance extraction in the presence of topological variations. The algorithm relies on hierarchical methodologies to subdivide the computational effort into two parts, namely, resolving the local interactions and resolving the global interactions. The HFRW resolves the local interactions by solving some appropriate PDE inside of a closed domain to compute a Markov Transition Matrix. On the other hand, the global interactions are resolved by simulating an appropriate Markov Chain using a Monte Carlo method (i.e. without the need for any linear system solves). In the HFRW, resolving the local interactions is more computationally expensive than resolving the global interactions. Consequently, the complexity of solving a large number of topologically different configurations which are constructed from the same set of building blocks is optimal, in the sense that it is almost independent of the number of total configurations. When extracting the capacitance of a large number of configurations all constructed from the same set of building blocks, the algorithm provides more than 2 orders of magnitude reduction in computational time compared to the fastest available discretization-based and discretization-free solvers.

Chapter 12

Future Work

In this section we will suggest some possible research directions that emerged from this thesis. In some of these possible research directions, we will demonstrate preliminary results.

12.1 Fast Stochastic Matrix Vector Product using Fast-Multipole

The science (or art) of fast matrix vector product algorithms has benefited the extraction community for the last decades. Developing a fast matrix vector product to handle parameterized/stochastic matrix-vector products of the form

$$\mathbf{A}(\mathbf{p})\mathbf{x}(\mathbf{p}) \tag{12.1}$$

is therefore an essential step towards moving stochastic algorithms to the mainstream extraction flow. We have presented a few ideas throughout this thesis primarily directed towards the compression of the system matrix. However, we believe that maintaining a hierarchical structure of the parameter space, such that far interactions are associated with a low dimensional parameter space and nearby interaction are associated with the fine description of the parameter spaces can result in a very efficient fast algorithm.

12.2 Computing Optimal Basis for the Stochastic System Matrix

The expansion of the system matrix in terms of orthogonal polynomials typically results in a very large number of terms, which in general constitutes the computational bottleneck of stochastic solvers. However, since the stochastic system matrix is known, we should consider finding a reduced set of basis to represent such matrix

$$\mathbf{A}(\vec{\eta}) = \sum_{i=0}^{K} \mathbf{A}_i d_i(\vec{\eta}) \tag{12.2}$$

We want to find $\mathbf{A}_i$ and $d_i(\vec{\eta})$ such that the norm of the residual is minimized

$$\begin{aligned} \mathbb{E}\left[\mathbf{a}(\vec{\eta})d_i(\vec{\eta})\right] &= \mathbb{E}\left[\mathbf{a}_i d_i^2(\vec{\eta})\right] \\ \mathbb{E}\left[\mathbf{a}(\vec{\eta})\mathbf{a}_i^T\mathbf{h}(\vec{\eta})\right] &= \mathbb{E}\left[\mathbf{a}_i^T\mathbf{a}_i d_i(\vec{\eta})\mathbf{h}(\vec{\eta})\right] \end{aligned}$$

which leads to

$$\begin{aligned} \frac{\mathbb{E}\left[\mathbf{a}(\vec{\eta})d_i(\vec{\eta})\right]}{\mathbb{E}\left[d_i^2(\vec{\eta})\right]} &= \mathbf{a}_i \\ \mathbb{E}\left[\mathbf{h}(\vec{\eta})\mathbf{a}(\vec{\eta})^T\right]\mathbf{a}_i &= \mathbf{a}_i^T\mathbf{a}_i\mathbb{E}\left[d_i(\vec{\eta})\mathbf{h}(\vec{\eta})\right] \end{aligned} \tag{12.3}$$

Using $d_i(\vec{\eta}) = \vec{\beta}_i^T\mathbf{h}(\vec{\eta})$ we obtain

$$\begin{aligned} \frac{\begin{bmatrix} \mathbf{a}_0 & \cdots & \mathbf{a}_K \end{bmatrix}\vec{\beta}_i}{||\vec{\beta}_i||^2} &= \mathbf{a}_i \\ \begin{bmatrix} \mathbf{a}_0^T \\ \vdots \\ \mathbf{a}_K^T \end{bmatrix}\mathbf{a}_i &= \mathbf{a}_i^T\mathbf{a}_i\vec{\beta}_i \end{aligned}$$

Combining both equations we obtain the following eigenvalue problems

$$\begin{bmatrix} \mathbf{a}_0^T \\ \vdots \\ \mathbf{a}_K^T \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 & \cdots & \mathbf{a}_K \end{bmatrix} \vec{\beta}_i = ||\vec{\beta}_i||^2 ||\mathbf{a}_i||^2 \vec{\beta}_i$$

$$\begin{bmatrix} \mathbf{a}_0 & \cdots & \mathbf{a}_K \end{bmatrix} \begin{bmatrix} \mathbf{a}_0^T \\ \vdots \\ \mathbf{a}_K^T \end{bmatrix} \mathbf{a}_i = ||\vec{\beta}_i||^2 ||\mathbf{a}_i||^2 \mathbf{a}_i$$

The above eigenvalue problems are very expensive, however, it is interesting to see if one can cheaply find approximations to the dominant directions $\vec{\beta}_i$. The standard Hermite expansion can then be projected on such directions such that the total number of terms in the expansion is significantly reduced.

12.3 Application of Developed Algorithms to Different Research Fields

Several other research fields have been facing the same or at least similar challenges as the ones we have been solving in this thesis. We believe that many of the algorithms presented in this thesis can be put to use in such applications. One example is given in [53], in which the forward problem involves solving the 1-D diffusion equation:

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial}{\partial x}\left(\nu(x)\frac{\partial u}{\partial x}\right) + S(x,t) \\ \frac{\partial u}{\partial x}(x=0) &= \frac{\partial u}{\partial x}(x=1) = 0 \\ u(x,t=0) &= 0. \end{aligned} \tag{12.4}$$

where $S(x,t)$ is the given source term and $\nu(x)$ is the diffusion coefficient. The diffusion coefficient is assumed to be random and to follow:

$$M(x) = \log(v(x) - v_0) \tag{12.5}$$

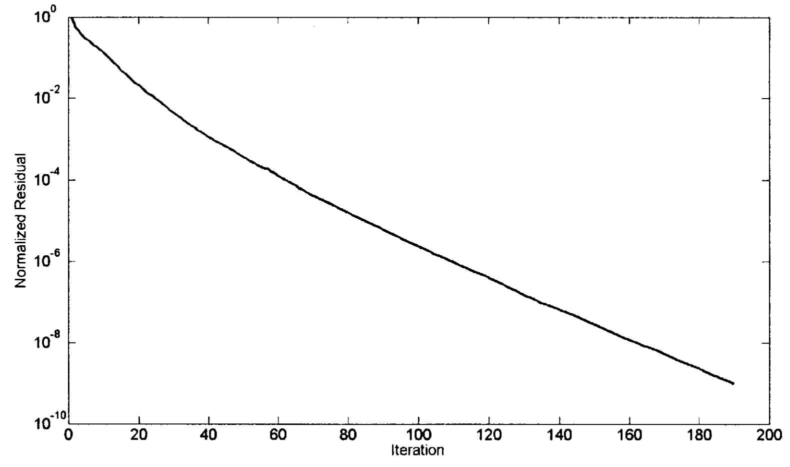


Figure 12-1: Residual versus iteration count.

where $M(x)$ is a Gaussian process. More details related to the description of the problem are in [53].

Using our SDSV method on this simple example we were able in 2 seconds to reduce the residual by 10 orders of magnitude (Figure 12-1). More work is needed to solve the same problem in multi-dimensions and to use more complex descriptions of the randomness encountered in the problem.

12.4 Applying the FRW to Full-Impedance Extraction

Extending the FRW algorithm to handle full impedance extraction, would enable using path recycling ideas to develop very efficient variation-aware impedance extraction algorithms. The main challenge is that the partial differential equations governing the impedance extraction formulations are not solely dependent on the Laplace operator. In fact, such equations involve both complex operators (e.g. Helmholtz) and relatively complex conservation laws. That said, in [52] a Brownian Motion based algorithm has been presented to efficiently solve the Helmholtz equation. We want to combine such idea, with novel methodologies to enforce the conservation laws, in order to solve the impedance extraction problem using random walk methods.

Appendix A

Mathematical Machinery

A.1 Capacitance Extraction in 2D

In the two-dimensional space (extending to infinity), the potential $\phi(r)$ is given by [70]

$$\phi(r) = \phi_0 + \int_{\Gamma} G(r, r')\rho(r')dr' \tag{A.1}$$

$$\int_{\Gamma} \rho(r)dr = 0 \tag{A.2}$$

where ϕ_0 is a constant unknown potential and Γ is the boundary of the conductor surfaces. Notice that unlike the 3D case, the potential is not zero at infinity. Instead the charge is forced to vanish at infinity.

A.2 Analytical Integration Formulas

For 2D, for a rectangular surface panel, the collocation integral is given by

$$\int_{\ell} \log(x^2+y^2)dx = -x\log(x^2+y^2) - 2\left(x - y\tan^{-1}\frac{x}{y}\right)\Bigg|_{\ell}$$

For 3D, for a rectangular surface panel, the collocation integral is given by

$$\int_S \frac{1}{\sqrt{x^2+y^2+z^2}}dxdy =$$

$$x\sinh^{-1}\left(\frac{y}{\sqrt{x^2+z^2}}\right) + y\sinh^{-1}\left(\frac{x}{\sqrt{y^2+z^2}}\right) - z\tan^{-1}\left(\frac{xy}{z\sqrt{x^2+y^2+z^2}}\right)\Bigg|_S$$

For 3D, for a rectangular volume filament, the collocation integral is given by

$$\int_V \frac{1}{\sqrt{x^2+y^2+z^2}}dxdydz =$$

$$xy\sinh^{-1}\left(\frac{z}{\sqrt{x^2+y^2}}\right) + xz\sinh^{-1}\left(\frac{y}{\sqrt{x^2+z^2}}\right) + yz\sinh^{-1}\left(\frac{x}{\sqrt{y^2+z^2}}\right)$$

$$-\frac{x^2}{2}\tan^{-1}\left(\frac{yz}{x\sqrt{x^2+y^2+z^2}}\right) - \frac{y^2}{2}\tan^{-1}\left(\frac{xz}{y\sqrt{x^2+y^2+z^2}}\right) - \frac{z^2}{2}\tan^{-1}\left(\frac{xy}{z\sqrt{x^2+y^2+z^2}}\right)\Bigg|_V$$

A.3 Standard Krylov Subspace Recycling

One fundamental characteristic of the linear systems produced by the sampling methods is that they are all "similar". Such similarities can be exploited using iterative methods combined with recycling of the Krylov subspaces shown for instance in Algorithm 21 and in Algorithm 22, where to simply notation we denoted $\mathbf{A}(\mathbf{p}_k)$ as $\mathbf{A}_k$.

In Algorithm 21 and 22, the columns of $\mathbf{Q}$ span the recycled subspace constructed by adding the subspaces (columns of $\mathbf{Q}_k$) of each system k. Although all $\mathbf{A}_k$ are dense matrices, the matrices $\mathbf{A}_k - \mathbf{A}_{k-1}$ are in some cases very sparse. In addition, matrix $\mathbf{T} = \mathbf{A}_{k-1}\mathbf{Q}$ in both algorithms does not need to be computed because it is already available when solving system k. Consequently, computing the residual of system k on the explored subspace $\mathbf{Q}_k$ requires only $O(N)$ operations. Furthermore, the fundamental relations in the standard Krylov subspace algorithms are slightly modified. As an example consider the Hessenberg decomposition in the GMRES Algorithm 22 which would normally read $\mathbf{AQ} = \mathbf{QH}$, while in subspace recycling reads $\mathbf{A}_k\mathbf{Q} = \mathbf{Q}_k\mathbf{H}_k$. We further notice that it is important to keep the size of the common subspace $\mathbf{Q}$ under control, in order to prevent the cost of even one projection to approach $O(N^2)$.

Algorithm 21 Standard GCR Krylov Subspace Recycling

```
1: $Q \leftarrow [\ ]$
2: for each system $k$ do
3:   if Q is not empty then
4:     Compute matrix $Q_k$ such that $A_kQ_k = T + (A_k - A_{k-1})Q$, where $T = A_{k-1}Q$ is an orthogonal matrix.
5:     Solve for $z_k$ such that $r_k = A_kQ_kz_k - b$ and $r_k$ is orthogonal to $A_kQ_k$
6:   else
7:     $Q_k \leftarrow b$
8:   end if
9:   while $\| r_k \| >$ threshold do
10:    Extend $Q_k \leftarrow \begin{bmatrix} Q_k & A_kr_k \end{bmatrix}$ such that $T = A_kQ_k$ is an orthonormal matrix
11:    Solve for $z_k$ such that $r_k = A_kQ_kz_k - b$ and $r_k$ is orthogonal to $A_kQ_k$
12:  end while
13:  $Q \leftarrow Q_k$
14:  $x_k \leftarrow Q_kz_k$
15: end for
```

Algorithm 22 Standard GMRES Krylov Subspace Recycling

```
1: $Q \leftarrow [\ ]$
2: for each system $k$ do
3:   if Q is not empty then
4:     Compute orthonormal matrix $Q_k$ such that $T + (A_k - A_{k-1})Q = Q_kH_k$ , where $T = A_{k-1}Q$ and $H_k$ is Hessenberg.
5:     Solve for $z_k$ such that the norm of $r_k = A_kQz_k - b$ is minimized
6:   else
7:     $Q_k \leftarrow \frac{b}{\|b\|}$
8:   end if
9:   while $\| r_k \| >$ threshold do
10:    Compute $q$ such that $T = A_kQ_k = \begin{bmatrix} Q_k & q \end{bmatrix} H_k$, and $\begin{bmatrix} Q_k & q \end{bmatrix}$ is an orthonormal matrix, and $H_k$ is Hessenberg
11:    Solve for $z_k$ such that the norm of $r_k = A_kQ_kz_k - b$ is minimized
12:    $Q_k \leftarrow \begin{bmatrix} Q_k & q \end{bmatrix}$
13:  end while
14:  $Q \leftarrow Q_k$ without the last column
15:  $x_k \leftarrow Qz_k$
16: end for
```

The final and most important observation is that the asymptotic complexity of solving N_S linear systems with recycling is still $O(N_S N^2)$. In other words, recycling can only produce constant speedup factors, and cannot change the order of complexity of the problem.

A.4 Projection-based Model Reduction

In projection methods the unknown is assumed to live in a subset of the N-dimensional space spanned by a small number of basis vectors, i.e. $\mathbf{x} = \mathbf{U}\mathbf{z}$, where $\mathbf{U} \in \mathbb{R}^{N\times r}$ is a collection of r basis vectors. Using the reduced representation of $\mathbf{x}$ and using some left projection matrix $\mathbf{V} \in \mathbb{R}^{r\times N}$ the reduced system and output are given by:

$$\begin{aligned} \mathbf{VAUz} &= \mathbf{Vb} \\ y &= \mathbf{c}^T\mathbf{U}\left(\mathbf{VAU}\right)^{-1}\mathbf{Vb} \end{aligned}$$

One standard way to construct the projection matrix $\mathbf{U}$ is to use multi-point matching techniques in which the columns of $\mathbf{U}$ are the solution of the linear system at different values of the parameter vector. Note that the similarity between projection based model reduction methods and non-intrusive stochastic simulation methods stems from the fact that both methods rely on solving the linear system at multiple points in the parameter space.

One standard way to construct the left projection matrix is to let $\mathbf{V} = \mathbf{U}^H$, i.e. use a Galerkin projection

$$y = \mathbf{c}^T\mathbf{U}\left(\mathbf{U}^H\mathbf{AU}\right)^{-1}\mathbf{U}^H\mathbf{b}$$

A.5 High Dimensional Parameter Space Sampling using Sparse Grids

One of the main objectives when dealing with variation-aware extraction is to derive parameterized models for the desired electrical property e.g. capacitance or resistance. The majority of techniques used to generate such models include the fundamental step of sampling the parameter space. A standard sampling scheme for one-dimensional parameter spaces is to choose sample points coinciding with the nodes of a Q-point quadrature scheme. This is especially useful when the parameter is associated with a probability density function (PDF). In such case, one chooses the quadrature scheme, such that its weighting function is the PDF (i.e. Hermite-Gaussian quadrature for Gaussian PDF, or Laguerre-Gaussian quadrature for uniform PDF). Let us call the set of 1D sample points $S^{(1)} = \{P_i^{(1)} : 1 \leq i \leq Q\}$. A simple way to generalize such sampling technique to D-dimensional spaces is through tensor products $S^{(D)} = \{P_{i_1}^{(1)} \otimes P_{i_2}^{(1)} \otimes \cdots \otimes P_{i_D}^{(1)} : 1 \leq i_1, i_2, \cdots, i_D \leq Q\}$. Unfortunately, the tensor product construction suffers from the disadvantage that the total number of sampling points is Q^D, which grows exponentially with the dimension. This phenomena is also known as "curse of dimensionality".

One advantageous approach to dealing with this challenge is to sample the parameter space using the Smolyak sparse grid construction [27]. In such construction the total number of sample points N_g depends polynomially on the number of parameters. The grid points in such construction are given by $S^{(D)} = \{P_{i_1}^{(1)} \otimes P_{i_2}^{(1)} \otimes \cdots \otimes P_{i_D}^{(1)} : \mathbf{i} \in \mathbb{N}_Q^D\}$, where $\mathbf{i} = [i_1, \cdots, i_D]$, $\mathbb{N}_Q^D = \{\mathbf{i} \in \mathbb{N}^D : \sum_{d=1}^{D} i_d = D + Q\}$, and $\mathbb{N}_Q^D = \emptyset$ for $Q < 0$. Similar to tensor grid constructions, Smolyak constructions provide bounds on grid interpolation errors as function of the order of the construction.

A.6 Hermite Polynomials

In this thesis we use the "probabilistic" orthonormal Hermite polynomials. In 1D such polynomials are given by

$$\Psi_k(\eta_k) = \frac{(-1)^k}{k!} \exp(\frac{\eta_1^2}{2}) \frac{d^k}{d\eta_1^k} \exp(-\frac{\eta_1^2}{2}) \tag{A.3}$$

where $\eta_1 \in (-\infty, \infty)$. The associated weighting function is the standard Gaussian PDF $\mathcal{P}(\eta_1)$.

The first four polynomials are

$$\begin{aligned}
\Psi_0(\eta_1) &= 1 \\
\Psi_1(\eta_1) &= \eta_1 \\
\Psi_2(\eta_1) &= \frac{1}{\sqrt{2}}\left(\eta_1^2 - 1\right) \\
\Psi_3(\eta_1) &= \frac{1}{\sqrt{6}}\left(\eta_1^3 - 3\eta_1\right) \\
\Psi_4(\eta_1) &= \frac{1}{\sqrt{24}}\left(\eta_1^4 - 6\eta_1^2 + 3\right)
\end{aligned}$$

The multivariate extension is obtained by taking the product of first order terms. The degree of a multivariate term is determined by the sum of the degrees of the univariate terms.

A.7 Continuous Karhunen Loeve Expansion

Starting from the Mercer theorem, we know that:

$$C(\mathbf{r},\mathbf{r}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{r})\phi_i(\mathbf{r}') \tag{A.4}$$

where $\phi_i(\mathbf{r})$ is an orthonormal basis forming the eigenfunctions of $C(\mathbf{r},\mathbf{r}')$, and λ_i are the corresponding eigenvalues. Using the orthogonality of the the eigenfunctions we obtain

$$\int C(\mathbf{r},\mathbf{r}')\phi_i(\mathbf{r}')d\mathbf{r}' = \lambda_i\phi_i(\mathbf{r}) \tag{A.5}$$

Finally, noting that

$$C(\mathbf{r},\mathbf{r}') = \mathbb{E}\left[\mathbf{p}(\mathbf{r})\mathbf{p}(\mathbf{r}')\right] \tag{A.6}$$

we can easily deduce that

$$\mathbf{p}(\mathbf{r}) = \sum_{i=1}^{\infty} \sqrt{\lambda_i}\phi_i(\mathbf{r})\eta_i \tag{A.7}$$

where η_i is a set of zero mean, unit variance random uncorrelated random variables (for our purpose we will use the standard Gaussian random variables).

A.8 Stochastic Inner Product and Stochastic Norm

We define the stochastic inner product between the vectors $\mathbf{u}(\vec{\eta})$ and $\mathbf{v}(\vec{\eta})$ to be

$$\begin{aligned}\langle \mathbf{u}(\vec{\eta}), \mathbf{v}(\vec{\eta})\rangle &= \mathbb{E}\left[\mathbf{u}(\vec{\eta})^T \mathbf{v}(\vec{\eta})\right] \\ &= \int_{\vec{\eta}} \mathbf{u}(\vec{\eta})^T \mathbf{v}(\vec{\eta}) \frac{\exp\left(-0.5\vec{\eta}^T\vec{\eta}\right)}{(2\pi)^{0.5N_p}} d\vec{\eta}\end{aligned}$$

where $\mathbb{E}\left[\cdot\right]$ is the expectation operator.

Consequently, the stochastic norm of a vector $\mathbf{u}(\vec{\eta})$ is $||\mathbf{u}(\vec{\eta})||_S^2 = \langle \mathbf{u}(\vec{\eta}), \mathbf{u}(\vec{\eta})\rangle = \mathbb{E}\left[||\mathbf{u}(\vec{\eta})||^2\right]$.

Proposition A.8.1. *The stochastic inner product $\langle \mathbf{v}(\vec{\eta}), \mathbf{u}(\vec{\eta})\rangle$ between two stochastic processes is equal to the standard deterministic Euclidean inner product $\langle \mathbf{V}(:), \mathbf{U}(:)\rangle = \mathbf{V}(:)^H\mathbf{U}(:)$ between the two vectors containing the polynomial chaos expansion coefficients of the two stochastic processes $\mathbf{v}(\vec{\eta})$, $\mathbf{u}(\vec{\eta})$*

Proof.

$$\begin{aligned}\langle \mathbf{v}(\vec{\eta}), \mathbf{u}(\vec{\eta})\rangle &= \mathbb{E}\left[\mathbf{v}(\vec{\eta})^H \mathbf{u}(\vec{\eta})\right] \\ &= \mathbb{E}\left[\sum_{i=0}^{K} \mathbf{v}_i^H \Psi_i(\vec{\eta}) \sum_{j=0}^{K} \mathbf{u}_j \Psi_j(\vec{\eta})\right] \\ &= \sum_{i=0}^{K}\sum_{j=0}^{K} \mathbf{v}_i^H \mathbf{u}_j \mathbb{E}\left[\Psi_i(\vec{\eta})\Psi_j(\vec{\eta})\right] \\ &= \sum_{k=0}^{K} \mathbf{v}_k^H \mathbf{u}_k = \mathbf{V}(:)^H \mathbf{U}(:) = \langle \mathbf{V}(:), \mathbf{U}(:)\rangle, \qquad \text{(A.8)}\end{aligned}$$

where $\mathbf{U}$ and $\mathbf{V} \in \mathbb{R}^{N\times K}$ are the matrices including respectively all the components $\mathbf{u}_i$ and $\mathbf{v}_i \in \mathbb{R}^{N\times 1} : i = 1, \cdots, K$ as defined by the unknown vector in (2.37). □

Appendix B

Proofs

B.1 Proof of Proposition 3.1.1

Proof. Let $\mathbf{x}_k = \mathbf{x}_i + \Delta_{\mathbf{x}}$ and $\Delta_{\mathbf{A}} = \mathbf{A}_k - \mathbf{A}_i$. If $\max \lambda(\mathbf{A}_i^{-1}\Delta_{\mathbf{A}}) < 1$, then using the Neumann expansion $\Delta_{\mathbf{x}}$ is expressed as

$$\Delta_{\mathbf{x}} = \sum_{k=1}^{\infty} \left(-\mathbf{A}_i^{-1}\Delta_{\mathbf{A}}\right)^k \mathbf{x}_i$$

If $d(\Delta_{\mathbf{A}}, \mathbf{x}_i) = 0$ then $\Delta_{\mathbf{A}}\mathbf{x}_i$ is contained in the subspace spanned by $\mathbf{Q}_i$ and $\Delta_{\mathbf{A}}\mathbf{x}_i = \mathbf{Q}_i\mathbf{z}$ for some vector $\mathbf{z}$. The leading term in the Neumann expansion is

$$\begin{aligned} \Delta_{\mathbf{x}} &= -\mathbf{A}_i^{-1}\Delta_{\mathbf{A}}\mathbf{x}_i \\ \mathbf{A}_i\Delta_{\mathbf{x}} &= -\Delta_{\mathbf{A}}\mathbf{x}_i \\ \mathbf{A}_i\Delta_{\mathbf{x}} &= -\mathbf{Q}_i\mathbf{z}, \end{aligned} \tag{B.1}$$

Notice that the solution of (B.1) is a second order accurate approximation for $\Delta_{\mathbf{x}}$. Since $\mathbf{Q}_i$ spans $\mathcal{K}(\mathbf{A}_i, \mathbf{b})$ and the right hand side of (B.1) is in $\mathbf{Q}_i$, then $\Delta_{\mathbf{x}}$ is in the space explored during the solution of $\mathbf{x}_i$. □

B.2 Proof of Theorem 4.1.1

Proof.

$$\begin{aligned}
< \Psi_i(\vec{\eta}), \Psi_j(\vec{\eta}) >_{\mathcal{W}} &= \int_{\vec{\eta}} \int_{\underline{\mathbf{p}}} \mathcal{W}(\underline{\mathbf{p}}, \vec{\eta}) \Psi_i(\vec{\eta}) \Psi_j(\vec{\eta}) d\underline{\mathbf{p}} d\vec{\eta} \\
&= \int_{\vec{\eta}} \Psi_i(\vec{\eta}) \Psi_j(\vec{\eta}) \left(\int_{\underline{\mathbf{p}}} \mathcal{W}(\underline{\mathbf{p}}, \vec{\eta}) d\underline{\mathbf{p}} \right) d\vec{\eta} \\
&= \int_{\vec{\eta}} \Psi_i(\vec{\eta}) \Psi_j(\vec{\eta}) \mathcal{P}(\vec{\eta}) d\vec{\eta} \\
&= \begin{cases} |\Psi_i(\underline{\vec{\eta}})|^2 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{B.2}$$

where $\mathcal{P}(\vec{\eta})$ is the Gaussian probability density function associated with the standard Hermite expansion. The relation $\int_{\underline{\mathbf{p}}} \mathcal{W}(\underline{\mathbf{p}}, \vec{\eta}) d\underline{\mathbf{p}} = \mathcal{P}(\vec{\eta})$ is a result of the interpretation of the weighting function as a probability density function (PDF), and on the fact that integrating the multivariate probability density function with respect to a subset of its random variables is still a PDF for the remaining subset of random variables. □

B.3 Proof of Theorem 4.1.2

Proof.

$$\begin{aligned}
f_i = \langle f(\underline{\mathbf{p}}), \Psi_i(\vec{\eta}) \rangle_{\mathcal{W}} &= \int_{\underline{\vec{\eta}}} \int_{\underline{\mathbf{p}}} f(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) \mathcal{W}(\underline{\mathbf{p}}, \underline{\vec{\eta}}) d\underline{\mathbf{p}} d\underline{\vec{\eta}} \\
&= \int_{\underline{\vec{\eta}}} \int_{\underline{\mathbf{p}}} f(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) \int_{\mathbf{p}_1} \mathcal{W}(\Delta\mathbf{p}, \underline{\vec{\eta}}) d\mathbf{p}_1 d\underline{\mathbf{p}} d\underline{\vec{\eta}} \\
&= \int_{\underline{\vec{\eta}}} \int_{\Delta\mathbf{p}} \mathcal{W}(\Delta\mathbf{p}, \underline{\vec{\eta}}) f(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) d\Delta\mathbf{p} d\underline{\vec{\eta}}
\end{aligned}$$

where $\Delta\mathbf{p}$ is a vector containing the elements of vector $\underline{\mathbf{p}}$ in addition to any $N_M - N_O - D$ other elements of vector $\mathbf{p}$. The choice of those extra elements is arbitrary for the sake of proof provided those elements are distinct from the elements of $\underline{\mathbf{p}}$. The vector containing those other elements is denoted by $\mathbf{p}_1$. The second identity is the result of integrating a multivariate PDF over a subset of its random variables. Next we use a variable transformation to express $\Delta\mathbf{p}$ and $\underline{\vec{\eta}}$ in terms of $\vec{\eta}$. To do that recall from the definition of the modified inner product that

$$\begin{aligned} \mathcal{W}(\Delta\mathbf{p}, \underline{\vec{\eta}}) &= \frac{\exp\left(-0.5\mathbf{v}^T\mathbf{C}_V^{-1}\mathbf{v}\right)}{(2\pi)^{\frac{N_M}{2}}\sqrt{|\mathbf{C}_V|}} \\ \mathbf{v} &= \begin{pmatrix} \Delta\mathbf{p} \\ \underline{\vec{\eta}} \end{pmatrix} \end{aligned} \tag{B.3}$$

Recall further that the relation between the parameter vector $\mathbf{p}$ and the uncorrelated random variables $\vec{\eta}$ is obtained from the Karhunen Loeve expansion (2.21). This means that we can express

$$\mathbf{v} = \begin{pmatrix} \Delta\mathbf{p} \\ \underline{\vec{\eta}} \end{pmatrix} = \tilde{\mathcal{M}}\vec{\eta} = \begin{pmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \Delta\vec{\eta} \\ \underline{\vec{\eta}} \end{pmatrix} \tag{B.4}$$

This relation facilitates computing the submatrices of the correlation matrix $\mathbf{C}_V$

$$\begin{aligned} \mathbb{E}[\Delta\mathbf{p}\ \Delta\mathbf{p}^T] &= \mathbf{M}_{11}\mathbf{M}_{11}^T + \mathbf{M}_{12}\mathbf{M}_{12}^T \\ \mathbb{E}[\Delta\mathbf{p}\ \underline{\vec{\eta}}^T] &= \mathbf{M}_{12} \\ \mathbb{E}[\underline{\vec{\eta}}\ \Delta\mathbf{p}^T] &= \mathbf{M}_{12}^T \end{aligned}$$

which when substituting in (B.3) leads to

$$\mathcal{W}(\Delta\mathbf{p}, \underline{\vec{\eta}})d\mathbf{v} = \frac{\exp\left(-0.5\vec{\eta}^T\tilde{\mathcal{M}}^T\mathbf{C}_V^{-1}\tilde{\mathcal{M}}\vec{\eta}\right)}{(2\pi)^{\frac{N_M}{2}}\sqrt{|\mathbf{C}_V|}}|\tilde{\mathcal{M}}|d\vec{\eta}$$

where

$$\tilde{\mathcal{M}}^T\mathbf{C}_V^{-1}\tilde{\mathcal{M}} = \begin{pmatrix} \mathbf{M}_{11}^T & \mathbf{0} \\ \mathbf{M}_{12}^T & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{M}_{11}\mathbf{M}_{11}^T + \mathbf{M}_{12}\mathbf{M}_{12}^T & \mathbf{M}_{12} \\ \mathbf{M}_{12}^T & \mathbf{I} \end{pmatrix}^{-1} \tilde{\mathcal{M}}$$

Using the block matrix inverse formula we obtain

$$\mathbf{C}_V^{-1} = \begin{pmatrix} \left(\mathbf{M}_{11}\mathbf{M}_{11}^T\right)^{-1} & -\left(\mathbf{M}_{11}\mathbf{M}_{11}^T\right)^{-1}\mathbf{M}_{12} \\ -\mathbf{M}_{12}^T\left(\mathbf{M}_{11}\mathbf{M}_{11}^T\right)^{-1} & \mathbf{I} + \mathbf{M}_{12}^T\left(\mathbf{M}_{11}\mathbf{M}_{11}^T\right)^{-1}\mathbf{M}_{12} \end{pmatrix} \tag{B.5}$$

Using the explicit inverse formula (B.5) to compute the new correlation matrix we obtain

$$\tilde{\mathcal{M}}\mathbf{C}_V^{-1}\tilde{\mathcal{M}} = \mathbf{I}$$

The Jacobian of the transformation can be related to the determinant of $\mathbf{C}_V$ by

$$\sqrt{|\mathbf{C}_V|} = \sqrt{|\mathbf{M}_{11}\mathbf{M}_{11}^T|} = |\mathbf{M}_{11}| = |\tilde{\mathcal{M}}|$$

Substituting all relations in (B.3) we obtain

$$\begin{aligned} \mathcal{W}(\Delta\mathbf{p}, \underline{\vec{\eta}}) &= \frac{\exp\left(-0.5\vec{\eta}^T\vec{\eta}\right)}{(2\pi)^{\frac{N_M}{2}}} \\ &= \mathcal{P}(\vec{\eta}) \end{aligned}$$

$$\begin{aligned} f_i &= \int_{\vec{\eta}} \mathcal{P}(\vec{\eta}) f(\underline{\mathbf{p}}) \Psi_i(\underline{\vec{\eta}}) d\vec{\eta} \\ &= \left\langle f(\underline{\mathbf{p}}), \Psi_i(\underline{\vec{\eta}}) \right\rangle \end{aligned} \tag{B.6}$$

□

B.4 Expressions for (5.11)

$$f = ||\mathbf{r}_n(\mathbf{p})||_S^2$$

$$= \sum_{k=1}^{K} \left(\sum_{ij,k} \mathbf{u}_n^T \mathbf{A}_i^T \mathbf{v}_n(j) \sum_{ij,k} \mathbf{A}_i \mathbf{v}_n(j) \mathbf{u}_n - 2 \sum_{ij,k} \mathbf{u}_n^T \mathbf{A}_i^T \mathbf{v}_n(j) \mathbf{R}_{n-1}(:,k) + ||\mathbf{R}_{n-1}(:,k)||^2 \right) \tag{B.7}$$

A summary of the expressions required to assemble $\mathbf{f}'$:

$$\frac{df}{d\mathbf{u}_n} = \sum_{k=1}^{K}\left(2\sum_{ij,k}\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij,k}\mathbf{A}_i\mathbf{v}_n(j)\mathbf{u}_n - 2\sum_{ij,k}\mathbf{A}_i^T\mathbf{v}_n(j)\mathbf{R}_{n-1}(:,k)\right)$$

$$\frac{df}{d\mathbf{v}_n(l)} = \sum_{k=1}^{K}\left(2\sum_{ij,k}\mathbf{u}_n^T\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij=l,k}\mathbf{A}_i\mathbf{u}_n - 2\sum_{ij=l,k}\mathbf{u}_n^T\mathbf{A}_i^T\mathbf{R}_{n-1}(:,k)\right)$$

A summary of the expressions required to assemble $\mathbf{f}''$:

$$\begin{aligned}
\frac{d^2f}{d\mathbf{u}_n^2} &= \sum_{k=1}^{K}2\sum_{ij,k}\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij,k}\mathbf{A}_i\mathbf{v}_n(j)\\
\frac{d^2f}{d\mathbf{v}_n(l)\mathbf{v}_n(m)} &= \sum_{k=1}^{K}2\sum_{ij=m,k}\mathbf{u}_n^T\mathbf{A}_i^T\sum_{ij=l,k}\mathbf{A}_i\mathbf{u}_n\\
\frac{d^2f}{d\mathbf{v}_n(l)d\mathbf{u}_n} &= \sum_{k=1}^{K}\left(2\left(\sum_{ij,k}\mathbf{u}_n^T\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij=l,k}\mathbf{A}_i+\right.\right.\\
&\qquad\left.\left.\sum_{ij=l,k}\mathbf{u}_n^T\mathbf{A}_i^T\sum_{ij,k}\mathbf{A}_i\mathbf{v}_n(j)\right) - 2\sum_{ij=l,k}\mathbf{R}_{n-1}(:,k)^T\mathbf{A}_i\right)
\end{aligned}$$

As an example, the last expression can be efficiently implemented using the Hermite expansion as

$$\begin{aligned}
\frac{d^2f}{d\mathbf{v}_n(l)d\mathbf{u}_n} &= \sum_{k=1}^{K}\left(2\left(\sum_{ij,k}\mathbf{u}_n^T\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij=l,k}\mathbf{A}_i+\right.\right.\\
&\qquad\left.\left.\sum_{ij=l,k}\mathbf{u}_n^T\mathbf{A}_i^T\sum_{ij,k}\mathbf{A}_i\mathbf{v}_n(j)\right) - 2\sum_{ij=l,k}\mathbf{R}_{n-1}(:,k)^T\mathbf{A}_i\right)
\end{aligned}$$

where we use the notation $\sum_{ij,k}$ to indicate a summation over all $i,j : \langle H_iH_j, H_k\rangle \neq 0$, the notation $\sum_{ij=l,k}$ to indicate a summation over all $i : \langle H_iH_l, H_k\rangle \neq 0$, the Matlab notation $\mathbf{R}_{n-1}(:,k)$ to indicate the k^{th} column of the matrix $\mathbf{R}_{n-1}$ and the notation $\mathbf{v}_n(j)$ to indicate the j^{th} element of vector $\mathbf{v}_n$.

B.5 A simple Transition Probability

The transition probability (Green's function) from a point $\mathbf{r} = (x, y)$ inside of the domain of a 2D rectangular homogeneous domain in the x-y plane, oriented such that the length in the x-direction is a and that in the y-direction is b, is given by:

$$P(\mathbf{r}, \mathbf{r}') = \begin{cases} \sum\limits_{n=0}^{\infty} \frac{2\sin(\frac{n\pi}{a}x')}{a\sinh(\frac{n\pi}{a}b)} \sin(\frac{n\pi}{a}x)\sinh(\frac{n\pi}{a}y) & \mathbf{r}' = (x', b) \\ \sum\limits_{n=0}^{\infty} \frac{2\sin(\frac{n\pi}{a}x')}{a\sinh(\frac{n\pi}{a}b)} \sin(\frac{n\pi}{a}x)\sinh(\frac{n\pi}{a}(b-y)) & \mathbf{r}' = (x', 0) \\ \sum\limits_{n=0}^{\infty} \frac{2\sin(\frac{n\pi}{b}y')}{b\sinh(\frac{n\pi}{b}a)} \sin(\frac{n\pi}{b}y)\sinh(\frac{n\pi}{b}x) & \mathbf{r}' = (a, y') \\ \sum\limits_{n=0}^{\infty} \frac{2\sin(\frac{n\pi}{b}y')}{b\sinh(\frac{n\pi}{b}a)} \sin(\frac{n\pi}{b}y)\sinh(\frac{n\pi}{b}(a-x)) & \mathbf{r}' = (0, y') \end{cases}$$

B.6 Proof of (8.3)

$$\begin{aligned} var\,(\Delta C) &= E\left[\left(\frac{1}{M}\sum_{m=1}^{M}(\omega_m^{(0)} - \mu_0) - (\omega_m^{(k)} - \mu_k)\right)^2\right] \\ &= \frac{1}{M^2}E\left(\sum_{m=1}^{M}(\omega_m^{(0)} - \mu_0)^2 + \sum_{m=1}^{M}(\omega_m^{(k)} - \mu_k)^2 -\right. \\ &- \left. 2\sum_{m=1}^{M}(\omega_m^{(0)} - \mu_0)\sum_{m=1}^{M}(\omega_m^{(k)} - \mu_k)\right). \end{aligned}$$

We use the independence of the random variables to eliminate all the cross terms resulting from both the squaring and cross-multiplication of summations to obtain:

$$\begin{aligned} var\,(\Delta C) &= \frac{1}{M^2}\left(\sum_{m=1}^{M}E(\omega_m^{(0)} - \mu_0)^2 + \sum_{m=1}^{M}E(\omega_m^{(k)} - \mu_k)^2 -\right. \\ &- \left. 2\sum_{m=1}^{M}E(\omega_m^{(0)} - \mu_0)(\omega_m^{(k)} - \mu_k)\right) \\ &= \frac{1}{M}\left(var(\omega^{(0)}) + var(\omega^{(k)}) - 2cov(\omega^{(0)}, \omega^{(k)})\right). \end{aligned}$$

B.7 Proof of Theorem 7.1.1

Using the standard Green's function technique to solve the Laplace equation with the corresponding boundary conditions we define the following partial differential equation

$$\begin{aligned}
&\varepsilon_i \nabla^2 G(\mathbf{r},\mathbf{r}') = \delta(\mathbf{r}-\mathbf{r}') \\
&G(\mathbf{r},\mathbf{r}') = 0 \quad \mathbf{r}' \in \Gamma_D, \quad \nabla_n G(\mathbf{r},\mathbf{r}') = 0 \quad \mathbf{r}' \in \Gamma_N \\
&G(\mathbf{r},\mathbf{r}') = G_0, \quad \int_{\Gamma_F} \nabla_n G(\mathbf{r},\mathbf{r}') d\Gamma' = 0 \quad \mathbf{r}' \in \Gamma_F \ ,
\end{aligned}$$

where Γ_D and Γ_N are the boundaries with the Dirichlet and Neumann boundary conditions, respectively. Γ_F is the boundary of the charge neutral fill. G_0 is an unspecified constant representing the potential of the fill. The existence of such Green's function is guaranteed from the fact that the Poisson equation has a unique solution for the mixed boundary condition case.

Using the integral equation formulation for each domain, summing the obtained relations over all domains and realizing that the sum of the integrals vanish at the dielectric boundaries, we obtain

$$\phi(\mathbf{r}) = \int\limits_{\Gamma_D \bigcup \Gamma_N \bigcup \Gamma_F} \phi(\mathbf{r}')\nabla_n G(\mathbf{r},\mathbf{r}') - G(\mathbf{r},\mathbf{r}')\nabla_n \phi(\mathbf{r}') d\mathbf{r}' \tag{B.8}$$

Using the prescribed boundary conditions of both the potential and the Green's function:

$$\phi(\mathbf{r}) = \int\limits_{\Gamma_D} \phi(\mathbf{r}')\nabla_n G(\mathbf{r},\mathbf{r}') d\mathbf{r}' \tag{B.9}$$

The transition probability density function from any internal point to a point of the allowed part of the boundary is given by $P(\mathbf{r},\mathbf{r}') = \nabla_n G(\mathbf{r},\mathbf{r}')$. To prove that such a probability function sums to one we use the divergence theorem and the partial differential equation defining the Green's function:

$$1 = \int \nabla_n G(\mathbf{r},\mathbf{r}') d\mathbf{r}' = \int\limits_{\Gamma_D} \nabla_n G(\mathbf{r},\mathbf{r}') d\mathbf{r}' \tag{B.10}$$

Finally, we need to prove the positivity of $\nabla_n G(\mathbf{r}, \mathbf{r}')$. The main idea is to prescribe a potential of $\delta(\mathbf{r}' - \mathbf{r}_B)$ to the Dirichlet (allowed) boundary and use (B.9) to obtain:

$$\phi(\mathbf{r}) = \nabla_n G(\mathbf{r}, \mathbf{r_B}). \tag{B.11}$$

Consequently, to prove positivity it is sufficient to demonstrate that $\phi(\mathbf{r})$ is between 0 and 1, i.e. the potential at any point within the domain must be smaller than the maximum potential at the Dirichlet boundary and larger than the minimum potential of the Dirichlet boundary. This is achieved by noting that the solution of the Laplace equation is a harmonic function for each dielectric domain. Therefore such solutions satisfy the maximum principle, i.e. the maximum potential is either at Γ_D,Γ_N, Γ_F or on the interfaces between the dielectric layers. We will demonstrate that this maximum (minimum) cannot be at Γ_N, Γ_F or the dielectric interfaces and that therefore the maximum and minimum potentials are on Γ_D.

Assume such maximum is at a dielectric interface, then at the interface $\nabla_n \phi(\mathbf{r})$ has opposite directions, which would imply using Gauss' law that there is charge accumulation at the interface. Since there are no charges accumulated at dielectric interfaces, by contradiction the maximum is not at an interface.

Assume such a maximum is on Γ_F, then the gradient vector of the potential at Γ_F is directed outwards, which as a consequence of Gauss' law means there is a charge accumulation at the fill. Since the fill is charge neutral we reach a contradiction.

Assume that the maximum potential is at Γ_N, then at a point infinitesimally away from the boundary and inside the domain the value of the potential is equal to that at the boundary, which by the maximum principle implies that the solution is constant. Since the Dirichlet boundary is excited using a non-constant potential we reach a contradiction.

B.8 Efficient Computation of the PDF of Stratified dielectric Media

The transition probability from the center of a cube with z-direction stratification to the cube boundaries is the superposition of the six potentials at the center when each side of the domain is excited with unit potential, while all other sides are held at zero potential. For the two cases in which both the x- or y-directed sides are at zero boundary conditions such computation is straight forward. Consider the case when all sides of the domain are at zero potential except for the upper z-plate:

$$\begin{aligned}
\phi(x,y,z) &= \sum_{n,m} A(m,n)\sin(\frac{n\pi}{L_x}x)\sin(\frac{m\pi}{L_y}y)\phi_z(k_z(n,m)z) \\
k_z^2 &= -(\frac{n\pi}{L_x})^2 - (\frac{m\pi}{L_y})^2 \\
A(m,n) &= \frac{4 < \phi(x,y,L_z), \sin(\frac{n\pi}{L_x}x)\sin(\frac{m\pi}{L_y}y) >}{L_x L_y \phi_z(k_z(m,n)L_z)},
\end{aligned}$$

where

$$\begin{aligned}
\phi_z(k_z,z) &= \begin{pmatrix} \cosh(k_z z) & \sinh(k_z z) \end{pmatrix} \prod_k T_{k,k+1} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\
T_{i,i+1} &= \begin{pmatrix} \cosh(k_z z_{i+1}) & \sinh(k_z z_{i+1}) \\ \frac{\varepsilon_i}{\varepsilon_{i+1}}\sinh(k_z z_{i+1}) & \frac{\varepsilon_i}{\varepsilon_{i+1}}\cosh(k_z z_{i+1}) \end{pmatrix}.
\end{aligned}$$

Note the function ϕ_z is computed using the transmission matrix theory such that the boundary conditions at the z-directed layers and interfaces are satisfied. The transition probability is such case is therefore

$$PDF = \sum_{n,m} \frac{4\sin(\frac{n\pi}{L_x}x')\sin(\frac{m\pi}{L_y}y')}{L_x L_y \phi_z(k_z(m,n)L_z)} \sin(\frac{n\pi}{L_x}x)\sin(\frac{m\pi}{L_y}y)\phi_z(k_z(n,m)z)$$

The case in which the z direction is held between zero potential is more involved. Consider for example the case in which all surfaces are at zero potential except that at $y = L_y$. To obtain k_z we need to numerically solve the following second order ordinary differential equation

$$\frac{d^2\phi_z}{dz^2} = -k_z^2\phi_z \tag{B.12}$$

$$\varepsilon_i \tfrac{d\phi_z}{dz}(z_i^-) = \varepsilon_{i+1}\tfrac{d\phi_z}{dz}(z_i^+), \quad \phi_z(0) = 0, \quad \phi_z(L_z) = 0 \ ,$$

where z_i are the coordinates of the boundaries separating the layers and L_z is the total length of the transition domain in the z-direction.

The numerical solution of (B.12) is much simpler than that of the original partial differential equation, since (B.12) is one-dimensional rather than three-dimensional. Following the computation of the modes in the z direction, the potential at the center is expressed as:

$$\phi(x, y, z) = \sum_{k_z, n} A(k_z, n)\sin(\frac{n\pi}{L_x}x)\sin(k_y y)\phi_z(k_z, z)$$
$$< \phi(x, L_y, z), \sin(\frac{n_0\pi}{L_x}x) >_x = \frac{L_x}{2}\sin(k_y L_y)\sum_{k_z} A(k_z, n_0)\phi_z(k_z, z)$$

Since the functions $\phi_z(k_z, z)$ are not orthogonal, i.e. $< \phi_z(k_z^{(i)}, z), \phi_z(k_z^{(j\neq i)}, z) \neq 0 >$, the transition probability is given by

$$\begin{aligned}
PDF &= \sum_{\ell} U^T M^{-1} V \\
M(i,j) &= < \phi_z(k_z^{(i)}, z), \phi_z(k_z^{(j)}, z) >_z \\
V(i) &= \frac{\sin(\frac{n_\ell \pi}{L_x}x')\phi_z(k_z^{(i)}, z')}{0.5L_x \sin(k_y L_y)} \\
U(i) &= \sin(\frac{n_\ell \pi}{L_x}x)\sin(k_y y)\phi_z(k_z^{(i)}, z)
\end{aligned}$$

Bibliography

[1] Fastmaxwell. *http://www.rle.mit.edu/cpg/fastmaxwell.*

[2] Ansoft. Hfss.

[3] Semiconductor Industry Association. International technology roadmap for semiconductors. 2003.

[4] Ivo Babuska, Raúl Tempone, and Georgios E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[5] S. Banerjee, P. Elakkumanan, L. W. Liebmann, J. A. Culp, and M. Orshansky. Electrically driven optical proximity correction. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 6925 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, April 2008.

[6] Shabbir Batterywala, Rohit Ananthakrishna, Yansheng Luo, and Alex Gyure. A statistical method for fast and accurate capacitance extraction in the presence of floating dummy fills. *VLSI Design, International Conference on*, 0:129–134, 2006.

[7] D. W. Bolton. The multinomial theorem. *The Mathematical Gazette*, 52(382):336–342, 1968.

[8] Duane S. Boning and Sani Nassif. Models of process variations in device and interconnect. In *Design of High Performance Microprocessor Circuits, chapter 6.* IEEE Press, 1999.

[9] H. Braunisch, Xiaoxiong Gu, A. Camacho-Bragado, and Leung Tsang. Off-chip rough-metal-surface propagation loss modeling and correlation with measurements. In *Electronic Components and Technology Conference, 2007. ECTC '07. Proceedings. 57th*, pages 785–791, 29 2007-June 1 2007.

[10] T. Bui-Thanh, K. Willcox, and O. Ghattas. Model reduction for large-scale systems with high-dimensional parametric input space. *SIAM J. Sci. Comput.*, 30(6):3270–3288, 2008.

[11] Tony F. Chan and Michael K. Ng. Galerkin projection methods for solving multiple linear systems. *SIAM J. Sci. Comput.*, 21(3):836–850, 1999.

[12] Paul G. Constantine, David F. Gleich, and Gianluca Iaccarino. A spectral galerkin method for random matrix equations. Technical Report UQ-08-01, Stanford University, 2008.

[13] Y.L. Le Coz and R.B. Iverson. A stochastic algorithm for high speed capacitance extraction in integrated circuits. *Solid-State Electronics*, 35(7):1005 - 1012, 1992.

[14] L. Daniel, C. S. Ong, S. C. Low, K. H. Lee, and J. K. White. A multiparameter moment matching model reduction approach for generating geometrically parameterized interconnect performance models. *IEEE Trans. Computer-Aided Design*, 23(5):678–93, May 2004.

[15] Luc Devroye. Non-uniform random variate generation, 1986.

[16] T. El-Moselhy and L. Daniel. Solutions of stochastic linear systems of equations for variation aware impedance extraction. *submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

[17] T. El-Moselhy and L. Daniel. Stochastic dominant singular vectors method for variation-aware extraction. *submitted to Design Automation Conference, 2010.*

[18] T. El-Moselhy and L. Daniel. Stochastic high order basis functions for volume integral equation with surface roughness. In *Electrical Performance of Electronic Packaging, 2007 IEEE*, pages 73–76, Oct. 2007.

[19] T. El-Moselhy and L. Daniel. Stochastic integral equation solver for efficient variation-aware interconnect extraction. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 415–420, June 2008.

[20] T. El-Moselhy, I. Elfadel, and L. Daniel. A generalized floating random walk algorithm for variation-aware parasitic extraction. *submitted to IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.*

[21] T. El-Moselhy, I.M. Elfadel, and B. Dewey. An efficient resistance sensitivity extraction algorithm for conductors of arbitrary shapes. In *Design Automation Conference, 2009. DAC 2009. 46th ACM/IEEE*, pages 770–775, June 2009.

[22] T. El-Moselhy, I.M. Elfadel, and D. Widiger. Efficient algorithm for the computation of on-chip capacitance sensitivities with respect to a large set of parameters. In *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pages 906–911, June 2008.

[23] T. El-Moselhy, Xin Hu, and L. Daniel. pfft in fastmaxwell: A fast impedance extraction solver for 3d conductor structures over substrate. In *Design, Automation and Test in Europe Conference and Exhibition, 2007. DATE '07*, pages 1–6, April 2007.

[24] T.A. El-Moselhy, I.M. Elfadel, and L. Daniel. A capacitance solver for incremental variation-aware extraction. In *Computer-Aided Design, 2008. ICCAD 2008. IEEE/ACM International Conference on*, pages 662–669, Nov. 2008.

[25] T.A. El-Moselhy, I.M. Elfadel, and L. Daniel. A hierarchical floating random walk algorithm for fabric-aware 3d capacitance extraction. In *Computer-Aided Design, 2009. ICCAD 2009. IEEE/ACM International Conference on*, Nov. 2009.

[26] Tarek El-Moselhy and Luca Daniel. Variation-aware interconnect extraction using statistical moment preserving model order reduction. In *DATE '10: Proceedings of the conference on Design, automation and test in Europe*, San Jose, CA, USA, 2010. EDA Consortium.

[27] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *NUMER. ALGORITHMS*, 18:209–232, 1998.

[28] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Dover Publications, 2003.

[29] Peter W. Glynn and Donald L. Iglehart. Importance sampling for stochastic simulations. *Management Science*, 35(11):1367–1392, 1989.

[30] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[31] A. Haji-Sheikh and E. M. Sparrow. The floating random walk and its application to monte carlo solutions of heat equations. *SIAM Journal on Applied Mathematics*, 14(2):370–389, 1966.

[32] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[33] M. Herzog, A. Gilg, M. Paffrath, P. Rentrop, and U. Wever. *From Nano to Space*, chapter Intrusive versus Non-Intrusive Methods for Stochastic Finite Elements, pages 161–174. Springer Berlin Heidelberg, 2008.

[34] X. Hu. Full-wave analysis of large conductor systems over substrate. P.hD. Thesis, Massachusetts Institute of Technology, January 2006.

[35] Integrand. Emx.

[36] J.N. Jere and Y.L. Le Coz. An improved floating-random-walk algorithm for solving the multi-dielectric dirichlet problem. *Microwave Theory and Techniques, IEEE Transactions on*, 41(2):325–329, Feb 1993.

[37] Tejas Jhaveri, Vyacheslav Rovner, Lawrence Pileggi, Andrzej J. Strojwas, Dipti Motiani, Veerbhan Kheterpal, Kim Yaw Tong, Thiago Hersan, and Davide Pandini. Maximization of layout printability/manufacturability by extreme layout regularity. *Journal of Micro/Nanolithography, MEMS and MOEMS*, 6(3):031011, 2007.

[38] L. J. Jiang, B. J. Rubin, J. D. Morsey, H. T. Hu, and A. Elfadel. Novel capacitance extraction method using direct boundary integral equation method and hierarchical approach. In *Electrical Performance of Electronic Packaging, 2006 IEEE*, pages 331–334, Oct. 2006.

[39] Rong Jiang, Wenyin Fu, J. M. Wang, V. Lin, and C. C. P. Chen. Efficient statistical capacitance variability modeling with orthogonal principle factor analysis. In *ICCAD '05: Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, pages 683–690, Washington, DC, USA, 2005. IEEE Computer Society.

[40] Dan Jiao, Jianfang Zhu, and S. Chakravarty. A fast frequency-domain eigenvalue-based approach to full-wave modeling of large-scale three-dimensional on-chip interconnect structures. *Advanced Packaging, IEEE Transactions on*, 31(4):890–899, Nov. 2008.

[41] M. Kamon. Fast parasitic extraction and simulation of three-dimensional interconnect via quasistatic analysis. P.hD. Thesis, Massachusetts Institute of Technology, June 1998.

[42] M. Kamon, N. Marques, and J. White. Fastpep: a fast parasitic extraction program for complex three-dimensional geometries. In *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, pages 456–460, Nov 1997.

[43] M. Kamon, M.J. Tsuk, and J.K. White. Fasthenry: a multipole-accelerated 3-d inductance extraction program. *Microwave Theory and Techniques, IEEE Transactions on*, 42(9):1750–1758, Sept. 1994.

[44] Sharad Kapur and David E. Long. Ies3: Efficient electrostatic and electromagnetic simulation. *Computing in Science and Engineering*, 5(4):60–67, 1998.

[45] John George Kemeny and James Laurie Snell. *Finite Markov chains.* University series in undergraduate mathematics. VanNostrand, New York, repr edition, 1969.

[46] A. Labun. Rapid method to account for process variation in full-chip capacitance extraction. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 23(6):941–951, June 2004.

[47] Lars Liebmann. Dfm, the teenage years. *Design for Manufacturability through Design-Process Integration II*, 6925:692502, 2008.

[48] M. Loeve. *Probability theory II.* Springer, 1994.

[49] Taotao Lu, Zeyi Wang, and Wenjian Yu. Hierarchical block boundary-element method (hbbem): a fast field solver for 3-d capacitance extraction. *Microwave Theory and Techniques, IEEE Transactions on*, 52(1):10–19, Jan. 2004.

[50] P. Maffezzoni. Analysis of substrate coupling by means of a stochastic method. *Electron Device Letters, IEEE*, 23(6):351–353, Jun 2002.

[51] Magma. Quickcap.

[52] Salil S. Kulkarni Mandar K. Chati, Mircea D. Grigoriu and Subrata Mukherjee. Random walk method for the two- and three-dimensional laplace, poisson and helmholtz's equations. *International Journal for Numerical Methods in Engineering*, 51(10):1133–1156, 2001.

[53] Youssef M. Marzouk and Habib N. Najm. Dimensionality reduction and polynomial chaos acceleration of bayesian inference in inverse problems. *Journal of Computational Physics*, 228(6):1862 – 1902, 2009.

[54] Lionel Mathelin and M. Yousuff Hussaini. A stochastic collocation algorithm. *NASA*, pages 2003–212153, 2003.

[55] Hermann G. Matthies and Andreas Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1295 – 1331, 2005. Special Issue on Computational Methods in Stochastic Mechanics and Reliability Analysis.

[56] M. D. Mckay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[57] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[58] Gordon Moore. Moore's law.

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

[60] Harald Niederreiter. Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, 84(6):957–1041, 1978.

[61] Anthony Nouy. A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 196(45-48):4521 – 4537, 2007.

[62] Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti. Recycling krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[63] Joel R. Phillips, Jacob K. White, and Associate Member. A precorrected-fft method for electrostatic analysis of complicated 3-d structures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16:1059–1072, 1997.

[64] J.R. Phillips. Variational interconnect analysis via PMTBR. In *Proc. of IEEE/ACM International Conference on Computer Aided-Design*, pages 872–9, November 2004.

[65] A.E. Ruehli. Equivalent circuit models for three-dimensional multiconductor systems. *Microwave Theory and Techniques, IEEE Transactions on*, 22(3):216–221, Mar 1974.

[66] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2003.

[67] K.K. Sabelfeld and N.A. Simonov. *Random Walks On Boundary For Solving Pdes.* VSP, 1994.

[68] Sachin K. Sachdeva, Prasanth B. Nair, and Andy J. Keane. Hybridization of stochastic reduced basis methods with polynomial chaos expansions. *Probabilistic Engineering Mechanics*, 21(2):182 - 192, 2006.

[69] Lou Scheffer. An overview of on-chip interconnect variation. In *SLIP '06: Proceedings of the 2006 international workshop on System-level interconnect prediction*, pages 27–28, New York, NY, USA, 2006. ACM.

[70] Fan Shi, Palghat Ramesh, and Subrata Mukherjee. On the application of 2d potential theory to electrostatic simulation. *Communications in Numerical Methods in Engineering*, 11(8):691–701, 1995.

[71] S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Mathematics Doklady*, (4):240–243, 1963.

[72] Jorge Fernández Villena and L. Miguel Silveira. Arms - automatic residue-minimization based sampling for multi-point modeling techniques. In *DAC '09: Proceedings of the 46th Annual Design Automation Conference*, pages 951–956, New York, NY, USA, 2009. ACM.

[73] J. Wang and J. White. Fast algorithms for computing electrostatic geometric sensitivities. In *Simulation of Semiconductor Processes and Devices, 1997. SISPAD '97., 1997 International Conference on*, pages 121–123, Sep 1997.

[74] Karl F. Warnick and Weng Cho Chew. Numerical simulation methods for rough surface scattering. *Waves in Random Media*, 11, 2001.

[75] JR. White, K.P. and W.J. Trybula. Dfm for the next generation [semiconductor manufacturing]. In *Electronics Manufacturing Technology Symposium, 1996., Nineteenth IEEE/CPMT*, pages 109–116, Oct 1996.

[76] Norbert Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[77] Dongbin Xiu and Jan S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM J. Sci. Comput.*, 27(3):1118–1139, 2005.

[78] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.*, 24(2):619–644, 2002.

[79] Dongbin Xiu and Daniel M. Tartakovsky. Numerical methods for differential equations in random domains. *SIAM J. Sci. Comput.*, 28(3):1167–1185, 2006.

[80] Zuochang Ye, Zhenhai Zhu, and J.R. Phillips. Incremental large-scale electrostatic analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 28(11):1641–1653, Nov. 2009.

[81] Wenjian Yu and Zeyi Wang. Enhanced qmm-bem solver for three-dimensional multiple-dielectric capacitance extraction within the finite domain. *Microwave Theory and Techniques, IEEE Transactions on*, 52(2):560–566, Feb. 2004.

[82] Wenjian Yu, Zeyi Wang, and Jiangchun Gu. Fast capacitance extraction of actual 3-d vlsi interconnects using quasi-multiple medium accelerated bem. *Microwave Theory and Techniques, IEEE Transactions on*, 51(1):109–119, Jan 2003.

[83] Wenjian Yu, Mengsheng Zhang, and Zeyi Wang. Efficient 3-d extraction of interconnect capacitance considering floating metal fills with boundary element method. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(1):12–18, Jan. 2006.

[84] Jianfang Zhu and Dan Jiao. A unified finite-element solution from zero frequency to microwave frequencies for full-wave modeling of large-scale three-dimensional on-chip interconnect structures. *Advanced Packaging, IEEE Transactions on*, 31(4):873–881, Nov. 2008.

[85] Zhenhai Zhu and J. White. Fastsies: a fast stochastic integral equation solver for modeling the rough surface effect. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 675–682, Nov. 2005.

[86] Zhenhai Zhu, J. White, and A. Demir. A stochastic integral equation method for modeling the rough surface effect on interconnect capacitance. In *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 887–891, Washington, DC, USA, 2004. IEEE Computer Society.