Stochastic Dominant Singular Vectors Method for Variation-Aware Extraction

Tarek El-Moselhy
Massachusetts Institute of Technology
tmoselhy@mit.edu

Luca Daniel
Massachusetts Institute of Technology
luca@mit.edu

ABSTRACT

In this paper we present an efficient algorithm for variation-aware interconnect extraction. The problem we are addressing can be formulated mathematically as the solution of linear systems with matrix coefficients that are dependent on a set of random variables. Our algorithm is based on representing the solution vector as a summation of terms. Each term is a product of an unknown vector in the deterministic space and an unknown direction in the stochastic space. We then formulate a simple nonlinear optimization problem which uncovers sequentially the most relevant directions in the combined deterministic-stochastic space. The complexity of our algorithm scales with the sum (rather than the product) of the sizes of the deterministic and stochastic spaces, hence it is orders of magnitude more efficient than many of the available state of the art techniques. Finally, we validate our algorithm on a variety of on-chip and off-chip capacitance and inductance extraction problems, ranging from moderate to very large size, not feasible using any of the available state of the art techniques.

Categories and Subject Descriptors: G.1.3 [Numerical Analysis]: Numerical Linear Algebra – *Linear systems, Singular value decomposition*; G.3 [Probability and Statistics]; J.6 [Computer-Aided Engineering]
General Terms: Algorithms, Design, Theory
Keywords: Stochastic Simulation, variation-aware extraction, Stochastic PDEs, Stochastic dominant singular vectors, intrusive algorithms, integral equations, surface roughness, parasitic extraction

1. INTRODUCTION

Advanced integrated circuit manufacturing steps such as etching, chemical mechanical polishing (CMP), electro-deposition, and photolithography produce large variabilities in the geometries of the manufactured interconnect structures. Such geometrical variabilities are not deterministic and are beginning to play a major role in determining the electrical characteristics of the interconnect structures. The development of efficient variation aware extraction solvers is a critical step to promptly and accurately account for the effects of such variabilities. Recently, a few attempts have been made towards developing such solvers [1, 2, 3].

Variation-aware solvers can be divided in general into two different categories, namely, those based on sampling algorithms (sometimes identified as non-intrusive) and those based on a stochastic formulation (sometimes identified as intrusive). All such algorithms are better understood if we abstract the problem of extracting the input impedance of interconnect structures with a large number of random variations to one of solving a large linear system of the form $\mathbf{A}(\mathbf{p})\mathbf{x}(\mathbf{p}) = \mathbf{b}(\mathbf{p})$, where $\mathbf{p}$ is a large vector of random variables. The "non-intrusive" algorithms are basically those algorithms that rely on sampling the parameter space vector $\mathbf{p}$, and then solving a deterministic system $\mathbf{A}(\mathbf{p}_k)\mathbf{x}(\mathbf{p}_k) = \mathbf{b}(\mathbf{p}_k)$ for every sample $\mathbf{p}_k$ of the parameter space. The term "non-intrusive" refers to the fact that any standard deterministic solver can be used to compute the unknown vector $\mathbf{x}(\mathbf{p}_k)$ in each solve. Examples of non-intrusive algorithms include the well-known Monte Carlo method, and the stochastic collocation method [4]. The computational complexity of standard sampling based methods is directly proportional to N_S, the number of sample points in the parameter space $O(N_S N^2)$. The main disadvantage of such methods is that generally a very large number of sample points N_S is required in order to accurately sample a high dimensional parameter space. In order to address this issue, the approach in [5] computes a small number of dominant basis vectors which span the solution space. The dominant basis vectors represent a reduced system and are constructed sequentially. At each step the reduced system is solved at a new point from a given set of quadrature points for the statistical moments of the output. The resulting residual is then used to determine if the current basis needs to be extended. The complexity of such algorithm is $O(N_b N^2)$, where N_b is the total number of dominant basis vectors, hence $N_b \ll N_S$.

On the other hand, "intrusive" algorithms solve stochastic linear systems using specialized techniques. Examples of intrusive algorithms are the Neumann expansion method [1, 6], the stochastic finite element method (SFE) [7], and the combined Neumann-Hermite expansion method (CNHE) [3]. Unfortunately, as for non-intrusive methods, also intrusive methods can be quite expensive. For instance, the complexity of the Neumann expansion method is $O(N^4)$, the complexity of the stochastic Galerkin method is $O(N_P^3 N^2)$ and the complexity of the combined Neumann-Hermite expansion is $O(N_P^2 N^2)$.

In this paper we present a novel "intrusive" algorithm to obtain sequentially a small number of dominant basis vectors, spanning the space of the solution, as in the "non-intrusive" algorithm [5]. In addition to being "intrusive", there are several fundamental differences distinguishing the algorithm in this paper from the one in [5]. Instead of constructing the dominant basis vectors from a given set of quadrature points, in this paper the basis vectors are constructed using a global optimization based approach, resulting in an optimal selection and in a global error bound. While in [5] the dominant basis vectors span only a deterministic space, in this paper we represent the solution using two sets of basis vectors, one spanning the deterministic space and the other spanning the stochastic space. Such representation allows computing directly an efficient reduced expansion for the unknown solution, rather than just statistical moments. The intrusive algorithm in this paper has an unprecedented low complexity $O(N^2 + N_p^4)$. Since even in worst case applications N_P is typically several orders of magnitude smaller that N, from a practical point of view the complexity is basically just $O(N^2)$, making our algorithm independent of the dimension of the parameter space. The details of the basic algorithm are given in section 3. A relaxation is then given in 4. Section 5 studies convergence properties. Section 6 presents an accelerated technique for computing the product of a stochastic matrix and a stochastic vector. Section 7 shows a detailed analysis of the computational complexity of the overall algorithm. Finally, in Section 8 we show the accuracy and speed of our algorithm on a variety of examples.

2. BACKGROUND

2.1 Impedance Extraction

In a general impedance extraction setting, we can use the standard mixed potential integral equation [8] to describe the relation between the volumetric current density $\mathbf{j}(\mathbf{r})$ inside of the conductors, the surface charge distribution $\rho(\mathbf{r})$ on the surface of the conductors, and the electric potential $\phi(\mathbf{r})$:

$$\frac{\mathbf{j}(\mathbf{r})}{\sigma_c} + j\omega\frac{\mu}{4\pi}\int_V G(\mathbf{r},\mathbf{r}')\mathbf{j}(\mathbf{r}')d\mathbf{r}' = -\nabla\phi(\mathbf{r}) \quad (1)$$

$$\frac{1}{4\pi\varepsilon}\int_S G(\mathbf{r},\mathbf{r}')\rho(\mathbf{r}')d\mathbf{r}' = \phi(\mathbf{r}) \quad (2)$$

$$\nabla\cdot\mathbf{j}(\mathbf{r}) = 0 \quad (3)$$

$$\hat{n}\cdot\mathbf{j}(\mathbf{r}) = j\omega\rho(\mathbf{r}), \quad (4)$$

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, and 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). The special case of the standard magneto-quasi-static (MQS) resistance and inductance extraction problem is described by imposing just equations (1) and (3).

A standard procedure for solving (1)-(4), or the corresponding EQS and MQS subsets, involves discretizing the current density $\mathbf{j}(\mathbf{r})$ and the charge density $\rho(\mathbf{r})$ with piecewise constant basis functions and using Galerkin testing to obtain a linear system of equations of the form:

$$\tilde{\mathbf{A}}(\tilde{\mathbf{p}})\,\tilde{\mathbf{x}}(\tilde{\mathbf{p}}) = \tilde{\mathbf{b}}(\tilde{\mathbf{p}}) \quad (5)$$

where $\tilde{\mathbf{A}}(\tilde{\mathbf{p}})$, $\tilde{\mathbf{x}}(\tilde{\mathbf{p}})$ and $\tilde{\mathbf{b}}(\tilde{\mathbf{p}})$ are described in detail in [9] for EQS, in [10] for MQS, and in [8] for EMQS and fullwave; $\tilde{\mathbf{p}}$ is a vector of random geometrical variations described by a multivariate Gaussian probability density function with a Gaussian or exponential correlation function. Examples of random geometrical variations are width and thickness variations in on-chip interconnects or rough surface in on-package or on-board interconnects.

Following the general practice, the Karhunen-Loeve expansion (or in other words a truncated Singular Value Decomposition) [11] is used to transform the correlated physical parameters $\tilde{\mathbf{p}}$ into *independent* Gaussian random variables $\mathbf{p}$, such that $\tilde{\mathbf{p}} = \sqrt{\mathbf{C}}\mathbf{p}$, where $\mathbf{C}$ is the correlation matrix associated with the multivariate PDF. Substituting $\tilde{\mathbf{p}} = \sqrt{\mathbf{C}}\mathbf{p}$ into (5) one can obtain the general statement of the problem

$$\mathbf{A}(\mathbf{p})\mathbf{x}(\mathbf{p}) = \mathbf{b}(\mathbf{p}) \quad (6)$$

where $\mathbf{A}(\mathbf{p}) = \tilde{\mathbf{A}}(\sqrt{\mathbf{C}}\mathbf{p})$, $\mathbf{x}(\mathbf{p}) = \tilde{\mathbf{x}}(\sqrt{\mathbf{C}}\mathbf{p})$, $\mathbf{b}(\mathbf{p}) = \tilde{\mathbf{b}}(\sqrt{\mathbf{C}}\mathbf{p})$, and as mentioned previously $\mathbf{p}$ is a vector of N_P *independent* Gaussian random variables.

2.2 Hermite Expansion

The Hermite expansion of a given scalar function $f(\mathbf{p})$ is

$$f(\mathbf{p}) = \sum_{i=1}^{K} f_i\, H_i(\mathbf{p}) \quad (7)$$

where $H_i(\mathbf{p})$ is a set of multivariate Hermite orthogonal polynomials [7] and K is a function of both N_O, the order of the orthogonal polynomial expansion, and N_P, the dimension of the parameter vector $\mathbf{p}$ [12]:

$$K = 1 + \sum_{n=1}^{N_O}\begin{pmatrix} n+N_P-1 \\ n \end{pmatrix}. \quad (8)$$

Consequently, for a second order expansion $N_O = 2$, the dependence of K on N_P is quadratic $K = O(N_p^2)$. The coefficients of the expansion f_i can be efficiently computed as suggested in [3].

If the Hermite expansion is applied to a vector or a matrix then, it is understood that, it is applied element-wise.

2.3 Stochastic Inner Product and Norm

We define the stochastic inner product between two vectors $\mathbf{u}(\mathbf{p})$ and $\mathbf{v}(\mathbf{p})$ to be

$$\langle\mathbf{u}(\mathbf{p}),\mathbf{v}(\mathbf{p})\rangle = \mathbb{E}\left[\mathbf{u}(\mathbf{p})^T\mathbf{v}(\mathbf{p})\right] = \int_{\mathbf{p}} \mathbf{u}(\mathbf{p})^T\mathbf{v}(\mathbf{p})\frac{\exp(-0.5\mathbf{p}^T\mathbf{p})}{(2\pi)^{0.5N_p}}d\mathbf{p}$$

where $\mathbb{E}[\cdot]$ is the expectation operator. Consequently, the stochastic norm of a vector $\mathbf{u}(\mathbf{p})$ is $||\mathbf{u}(\mathbf{p})||_S = \sqrt{\langle\mathbf{u}(\mathbf{p}),\mathbf{u}(\mathbf{p})\rangle} = \sqrt{\mathbb{E}\left[||\mathbf{u}(\mathbf{p})||^2\right]}$.

2.4 Representation of Parameter Dependence

The unknown $\mathbf{x}(\mathbf{p})$ lives in the N dimensional stochastic space. One way to describe the dependence of the vector $\mathbf{x}(\mathbf{p})$ on the parameters is to expand $\mathbf{x}(\mathbf{p})$ in terms of multivariate orthogonal polynomials spanning the parameter space:

$$\mathbf{x}(\mathbf{p}) = \sum_{i=1}^{K}\mathbf{x}_i H_i(\mathbf{p})$$

This last expansion can be expressed in the more compact form

$$\mathbf{x}(\mathbf{p}) = \mathbf{X}\mathbf{h}(\mathbf{p}) \quad (9)$$

where $\mathbf{X}$ is a *deterministic* matrix of size $N\times K$ with columns $\mathbf{x}_i$ and $\mathbf{h}(\mathbf{p}) = (H_1(\mathbf{p}),\cdots,H_K(\mathbf{p}))^T$ is a vector of K orthogonal polynomials of the *stochastic* variable $\mathbf{p}$. This decomposition represents the unknown vector as a product of two different spaces, namely, the deterministic space, and the stochastic space. The idea behind both the standard stochastic collocation and Galerkin methods is to fix the stochastic component $\mathbf{h}(\mathbf{p})$, and solve only for the deterministic component $\mathbf{X}$.

In particular in the non-intrusive stochastic collocation method (SCM) [4], the parameter space is sampled at a point using the delta function and the solution is found at that particular point. The challenge with the SCM is that a very large number of sample points is needed in order to accurately sample the parameter space. Consequently, solving the linear system at every sample point in the parameter space is computationally very expensive $O(N_S N^2)$.

On the other hand, in the intrusive stochastic Galerkin Method (SGM), also known as stochastic finite element (SFE) [7], the solution matrix $\mathbf{X}$ is found such that the residual is orthogonal to the stochastic space spanned by $\mathbf{h}(\mathbf{p})$. The SGM is computationally very expensive since solving simultaneously for the entire unknown matrix $\mathbf{X}$ is $O(N^2 N_p^3)$, i.e. even more expensive than the SCM.

3. STOCHASTIC DOMINANT SINGULAR VECTORS METHOD

To avoid the complexities associated with both the SCM and the SGM we suggest the use of a *nonlinear* representation of $\mathbf{x}(\mathbf{p})$, where both the deterministic and stochastic components are assumed unknown. To better understand the idea behind our work let us consider expressing $\mathbf{x}(\mathbf{p})$ in terms of its dominant basis by expressing $\mathbf{X}$ in terms of its singular value decomposition (SVD) $\mathbf{X} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \tilde{\mathbf{v}}_i^T$

$$\mathbf{x}(\mathbf{p}) = \mathbf{X}\mathbf{h}(\mathbf{p}) = \sum_{i=1}^{r} \mathbf{u}_i \mathbf{v}_i^T \mathbf{h}(\mathbf{p}) \tag{10}$$

where $\sigma_i \tilde{\mathbf{v}}_i = \mathbf{v}_i$ and r is the total number of dominant basis. Note that $\mathbf{v}_i^T \mathbf{h}(\mathbf{p})$ is a scalar polynomial and that (10) can therefore be expressed as

$$\mathbf{x}(\mathbf{p}) = \sum_{i=1}^{r} \mathbf{v}_i^T \mathbf{h}(\mathbf{p}) \mathbf{u}_i \tag{11}$$

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. It should be noted that the representation of the unknown in (11) has been previously and independently suggested in [13]. Our algorithm (presented in the rest of the paper) is however significantly different than that in [13].

The Key Idea. *We propose to decompose the solution vector* $\mathbf{x}(\mathbf{p})$ *in the form of the summation (11) 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.*

To achieve our objective we first substitute (11) in (6) to obtain

$$\mathbf{A}(\mathbf{p}) \sum_{i=1}^{r} \mathbf{v}_i^T \mathbf{h}(\mathbf{p}) \mathbf{u}_i = \mathbf{b}(\mathbf{p}) \tag{12}$$

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

$$\mathbf{x}_n(\mathbf{p}) = \sum_{i=1}^{n} \mathbf{v}_i^T \mathbf{h}(\mathbf{p}) \mathbf{u}_i = \mathbf{x}_{n-1}(\mathbf{p}) + \mathbf{v}_n^T \mathbf{h}(\mathbf{p}) \mathbf{u}_n \tag{13}$$

and

$$\mathbf{r}_n(\mathbf{p}) = \mathbf{b}(\mathbf{p}) - \mathbf{A}(\mathbf{p}) \mathbf{x}_n(\mathbf{p}) \tag{14}$$

Equation (14) can be put in a recursive form by using (13)

$$\begin{aligned} \mathbf{r}_n(\mathbf{p}) &= \mathbf{b}(\mathbf{p}) - \mathbf{A}(\mathbf{p})\mathbf{x}_{n-1}(\mathbf{p}) - \mathbf{A}(\mathbf{p})\mathbf{v}_n^T \mathbf{h}(\mathbf{p})\mathbf{u}_n \\ &= \mathbf{r}_{n-1}(\mathbf{p}) - \mathbf{A}(\mathbf{p})\mathbf{v}_n^T \mathbf{h}(\mathbf{p})\mathbf{u}_n \end{aligned} \tag{15}$$

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

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

$$\min_{\mathbf{u}_n, \mathbf{v}_n} ||\mathbf{r}_n(\mathbf{p})||_S^2 \tag{16}$$

where the objective function is

$$\begin{aligned} f = ||\mathbf{r}_n(\mathbf{p})||_S^2 &= ||\mathbf{r}_{n-1}(\mathbf{p})||_S^2 - 2\mathbb{E}\left[\mathbf{u}_n^T \mathbf{A}(\mathbf{p})^T \mathbf{v}_n^T \mathbf{h}(\mathbf{p}) \mathbf{r}_{n-1}(\mathbf{p})\right] + \\ &\quad + \mathbb{E}\left[\mathbf{u}_n^T \mathbf{A}(\mathbf{p})^T \mathbf{v}_n^T \mathbf{h}(\mathbf{p}) \mathbf{A}(\mathbf{p}) \mathbf{u}_n \mathbf{v}_n^T \mathbf{h}(\mathbf{p})\right] \end{aligned} \tag{17}$$

One approach to minimizing (16) is to set the gradient $\mathbf{f}'$ of the objective function f to zero, which can be achieved using Newton's method, solving at each iteration the linear system

$$\mathbf{f}' = -\mathbf{f}''\mathbf{dz} \tag{18}$$

$\mathbf{f}''$ is the Hessian of the objective function. Detailed expressions for $\mathbf{f}'$ and $\mathbf{f}''$ are given in appendix A. Notice that (16) can also be efficiently solved using a fixed point iteration.

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 size $O(N+K)$ for a complexity $O(N+K)^2$. Such minimization is performed only r times, where r is the number of dominant basis of $\mathbf{x}(\mathbf{p})$. 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 7.

The norm of the residual is used 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 1 presents a complete summary of our proposed algorithm. Notice that both unknowns $\mathbf{u}_n$ and $\mathbf{v}_n$ are combined into a single vector $\mathbf{z}$.

Algorithm 1 Stochastic Dominant Singular Vectors Method (SDSV)

```
1: $\mathbf{x}(\mathbf{p}) \leftarrow 0$, $\mathbf{r}(\mathbf{p}) \leftarrow \mathbf{b}(\mathbf{p})$
2: $\mathbf{u}_n \leftarrow \mathbf{x}_0$, the solution of the nominal problem
3: $\mathbf{v}_n \leftarrow \mathbf{e}_1$
4: $\mathbf{z} \leftarrow \left(\mathbf{u}_n^T\ \mathbf{v}_n^T\right)^T$
5: while $||\mathbf{r}(\mathbf{p})||_S^2 >$ Threshold do
6:   repeat
7:     form first derivative $\mathbf{f}'$ as in (22)
8:     form Hessian $\mathbf{f}''$ as in (23)
9:     solve linear system $\mathbf{f}''\mathbf{dz} = -\mathbf{f}'$
10:    $\mathbf{z} \leftarrow \mathbf{z} + \mathbf{dz}$
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:  $\mathbf{x}(\mathbf{p}) \leftarrow \mathbf{x}(\mathbf{p}) + \mathbf{u}_n \mathbf{v}_n^T \mathbf{h}(\mathbf{p})$.
14:  $\mathbf{r}(\mathbf{p}) \leftarrow \mathbf{r}(\mathbf{p}) - \mathbf{A}(\mathbf{p}) \mathbf{u}_n \mathbf{v}_n^T \mathbf{h}(\mathbf{p})$.
15:  $\mathbf{u}_n \leftarrow \mathbf{x}_0$
16:  $\mathbf{v}_n \leftarrow \mathbf{e}_1$
17:  $\mathbf{z} \leftarrow \left(\mathbf{u}_n^T\ \mathbf{v}_n^T\right)^T$
18: end while
```

For efficient implementation we use the Hermite expansion to express both the system matrix $\mathbf{A}(\mathbf{p}) = \sum_{i=1}^{K} \mathbf{A}_i H_i(\mathbf{p})$ and the residual vector $\mathbf{r}_n(\mathbf{p}) = \mathbf{R}_n \mathbf{h}(\mathbf{p})$. The implementation details are omitted, however, one example of the resulting expressions can be found in Appendix A.

4. RELAXING THE OPTIMIZATION

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. First, the index k_{max} corresponding to the vector coefficient of the residual Hermite expansion with the largest norm is determined

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

where $\mathbf{r}_{n-1}(\mathbf{p}) = \mathbf{R}_{n-1}\mathbf{h}(\mathbf{p})$. Then the indices of the elements of $\mathbf{v}_n$ that have a contribution in the k_{max}-direction is determined. In other words, we find all the indices j of the elements of the vector $\mathbf{v}_n$, such that $\mathbf{v}_n(j)$ multiplied with the system matrix will result in some component in the k_{max}-direction:

$$j : \langle \mathbf{A}(\mathbf{p})H_j(\mathbf{p}), H_{k_{max}}(\mathbf{p})\rangle \neq 0.$$

The number of such indices is $O(N_P)$. 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. CONVERGENCE ANALYSIS

The following theorem summarizes the convergence properties of our algorithm.

THEOREM 5.1. *For any nonsingular matrix* $\mathbf{A}(\mathbf{p})$ *the sequence of the norms of the residuals* $\{||\mathbf{r}_n(\mathbf{p})||_S\}$ *is strictly decreasing for any non-zero update of the solution* $\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\mathbf{p})$ *(13).*

PROOF. The optimal $\mathbf{u}_n$ and $\mathbf{v}_n$ occur when the derivative of the nonlinear objective function in (17) is zero

$$\mathbf{0} = -\mathbb{E}\left[\mathbf{A}^T(\mathbf{p})\mathbf{v}_n^T\mathbf{h}(\mathbf{p})\mathbf{r}_{n-1}(\mathbf{p}) + \mathbf{A}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\mathbf{p})\mathbf{v}_n^T\mathbf{h}(\mathbf{p})\right] \tag{19}$$

Multiplying (19) by $\mathbf{u}_n$ and substituting the resulting expression in (17)

$$||\mathbf{r}_n(\mathbf{p})||_S^2 = ||\mathbf{r}_{n-1}(\mathbf{p})||_S^2 - ||\mathbf{A}(\mathbf{p})\mathbf{u}_n\mathbf{v}_n^T\mathbf{h}(\mathbf{p})||_S^2 \tag{20}$$

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

6. FAST STOCHASTIC MATRIX VECTOR PRODUCT

The computational complexity of our algorithm is mainly determined by a matrix vector product of the form $\mathbf{A}(\mathbf{p})\mathbf{u}_n$ (see (17)). The complexity of computing such a matrix vector product using the traditional Hermite expansion of the system matrix

$$\mathbf{A}(\mathbf{p})\mathbf{u}_n = \sum_{i=1}^{K}\mathbf{A}_iH_i(\mathbf{p})\mathbf{u}_n = \sum_{i=1}^{K}\mathbf{A}_i\mathbf{u}_nH_i(\mathbf{p}) = \mathbf{T}\mathbf{h}(\mathbf{p}), \tag{21}$$

would be $O(N^2K)$, since the columns of $\mathbf{T}$ are computed successively by computing the products $\mathbf{A}_i\mathbf{u}_n$. Instead, we compute the rows of $\mathbf{T}$ successively, leading to a final complexity of $O(N^2)$. In the following, we summarize the key steps allowing such reduced complexity. We 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 $\mathbf{u}_n$. All such rows are collected 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. $\mathcal{A}_k(i,:) = \mathbf{A}_i(k,:)$. The SVD of $\mathcal{A}_k = \mathbf{U}_{K\times q}\mathbf{S}_{q\times q}\mathbf{V}'_{N\times q}$ is then used 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 $\mathbf{A}(\mathbf{p})\mathbf{u}_n$ is $O(N^2q+KNq) \simeq O(N^2)$. For the more general case of computing $\mathbf{A}(\mathbf{p})\mathbf{w}(\mathbf{p})$, i.e. the product of a parameter dependent matrix and a parameter dependent vector, we use the SVD to represent the vector $\mathbf{w}(\mathbf{p})$ 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 $\mathbf{w}(\mathbf{p})$.

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* $\mathbf{A}(\mathbf{p})$*, 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 $\mathbf{A}(\mathbf{p})$, in particular the fact that every matrix entry in the k^{th} row of $\mathbf{A}(\mathbf{p})$ 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.

7. COMPLEXITY ANALYSIS

The following analysis is based on using Krylov subspace iterative methods (e.g. GMRES) to solve the linear system (18).

7.1 Memory

The proposed algorithm requires storing the system matrix of (18), i.e. it requires memory complexity $O((N+K)^2)$. For efficient implementation one might want to load in memory all the coefficient matrices $\mathbf{A}_i$ of the Hermite expansion of $\mathbf{A}(\mathbf{p})$, which requires a memory of $O(N^2q_{max}+KNq_{max})$, where q_{max} is the maximum dominant basis for any row following the fast stochastic matrix vector product algorithm in Section 6.

7.2 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 $\frac{d^2f}{d\mathbf{u}_n^2}$ (see Appendix A), which has to be formed at every iteration due to the change of $\mathbf{v}_n$. Since a Krylov subspace method is used for solving the linear system, the matrix-matrix product in $\frac{d^2f}{d\mathbf{u}_n^2}$ is not computed explicitly. Instead, given $\hat{\mathbf{u}}_n$ and $\hat{\mathbf{v}}_n$ (the guesses to the unknown vectors obtained from the Krylov iteration), the product $\mathbf{w}(\mathbf{p}) = \mathbf{A}(\mathbf{p})\hat{\mathbf{u}}_n$ is first computed for a complexity $O(N^2+KN)$ as discussed in Section 6. Then, the product $\mathbf{A}(\mathbf{p})^T\mathbf{w}(\mathbf{p})$ is computed for an additional complexity of $O(N^2+K^{3/2}N)$ as discussed in Section 6. The complexity of solving (18) is $O(N+K)^2 = O(N^2+N_P^4)$. Consequently, the total complexity is $O(N^2+NK^{3/2})$. In general $K^{3/2} < N$, therefore the final 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 such complexity is independent of the number of parameters.

8. RESULTS

All results in this section are obtained from a Matlab implementation of our algorithms running on an Intel Xeon CPU at 2.66 GHz

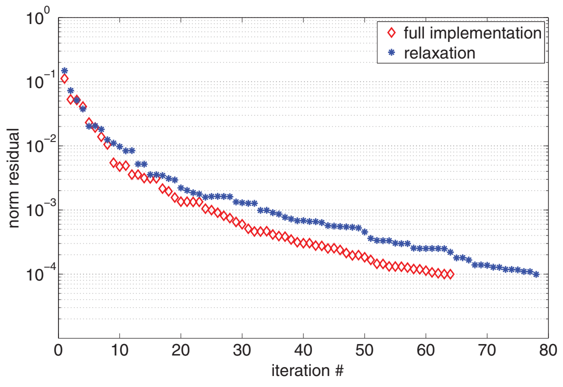

Figure 1: Residual norm versus iteration for both the full and the relaxed optimization

with 4GB of memory.

8.1 Small On-Chip 2-D Capacitance Extraction Example

The first example is a 16-conductor on-chip 2-D structure. The conductors are arranged in a 4×4 grid typically used as a template for building capacitance tables. Each conductor is discretized using 30 panels, resulting in a total of 481 unknowns. We assume that the width of each conductor is an independent Gaussian random variable, resulting in $N_P = 16$. 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 became less than 10^{-4}. We observed that a total of 64 directions are sufficient to represent the solution to the desired accuracy. Solving this with a Matlab implementation of our algorithm required 25sec which is a factor of $8\times$ faster than the SMOR [5], a factor of $18\times$ faster than the SCM [4], and a factor of $32\times$ faster than the SGM [7].

Using the relaxed formulation in Section 4 a total of 78 directions are needed to obtained the same accuracy. In other words, the number of iterations (dominant search directions) increases by about 20% compared to the original formulation (see Figure 1), however, the total simulation time is reduced by a factor of $2\times$ or just 12.5sec.

8.2 Large On-Chip 2-D Capacitance Extraction Example

The second example is a larger 2-D structure consisting of 100 conductors, arranged on a 10×10 grid. In this new setup $N = 5000$, $N_P = 100$ and $K = 5151$. A total of 142 dominant bases are required to reach a residual of norm 10^{-4}. Using the computed $\mathbf{x}(\mathbf{p})$ 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 a total of just 2000 solves. Our algorithm has about the same time complexity as the SMOR [5], which (unlike our algorithm) can exploit the sparsity of the coefficient matrices of the system matrix Hermite expansion. Our algorithm is $10\times$ faster than the second order sparse grid SCM [4] which requires more than $20,000$ solves for the same accuracy. The problem cannot be solved using the SGM [7] due to memory constrains (system matrix requires about 200TB).

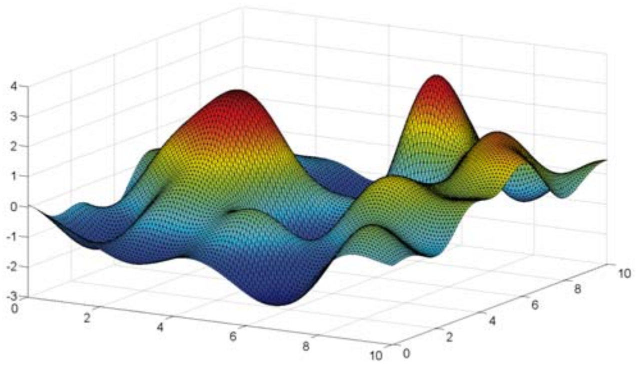

Figure 2: One instantiation of a plate with very rough surface.

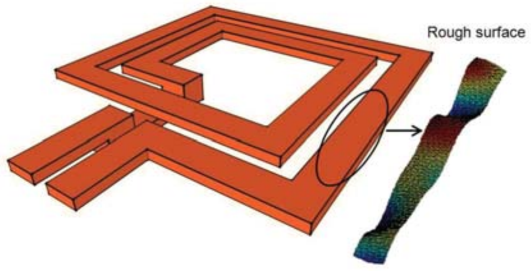

Figure 3: 2-turn Inductor with upper rough surface. Part of the surface magnified.

8.3 Large Off-Chip I/O Pad 3-D Capacitance Extraction Example

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 normalized surface width and length directions are 10×10 unit lengths (Figure 2). The surface of the place is very rough. The roughness is described by a Gaussian multivariate PDF and a Gaussian correlation function. The standard deviation is 1 unit length and the correlation length is 1 unit length. The plate is discretized using $N = 20,000$ triangular panels. The rough surface is described using a total of 91 uncorrelated (independent) random variables. The total number of orthogonal polynomials is $K = 4278$. Using the suggested fast matrix vector product algorithm (see Section 6) 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 required to solve the entire problem is 47 minutes. This is a factor of $10\times$ faster than the SMOR [5], a factor of $120\times$ faster than the best sampling-based approach and at least a factor of $100\times$ faster than the combined Neumann Hermite expansion [3].

8.4 Large On-Package 3-D Inductance and Resistance Extraction Example

The final example is a 2 turn inductor similar to the structure in [3]. The side length of the inductor is $1000\mu m$ 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 we simulated the structure in 4 hours, while at least 8 hours were required for the SMOR, 32 hours were required for the combined Neumann Hermite expansion (CNHE), a total of 150 hours were required for the SCM, and finally an estimated total of 828 hours would be required for the standard Neumann expansion [7]. The SGM fails in this example due to excessive memory requirements (800,000 TB). In conclusion, our algorithm is a factor of $2\times$ faster than the SMOR, a factor of $8\times$ faster than the CNHE, a factor of $100\times$ faster than the standard Neumann expansion and a factor of $37\times$ faster than the best sampling-based approach.

9. CONCLUSION

In this paper we have presented a new algorithm for solving stochastic linear systems of equations, i.e. linear systems in which the system matrix and right hand side are functions of a multivariate random vector. In our approach we express the solution in terms of its dominant bases in the combined deterministic-stochastic space. Such bases are computed sequentially, such that at every step the stochastic norm of the residual is minimized. The iteration terminates when the norm of the residual becomes smaller than a preset threshold, thereby providing control on the accuracy vs. computational time trade-off. Our algorithm has an unprecedented low complexity $O(N^2+N_p^4)$. Since even in worst case applications N_P is typically several orders of magnitude smaller that N, from a practical point of view the complexity is basically just $O(N^2)$, making the complexity independent of the number of parameters. Furthermore, we have presented a relaxation of the problem which significantly simplifies the complexity to $O(N^2+N_P^2)$, while typically only increasing by less than 20% the number of dominant bases, required for the same accuracy. The proposed algorithm has been validated on a variety of on-chip and off-chip capacitance and inductance extraction problems (N up to 20,000 and N_p up to 400 uncorrelated parameters). Such examples are significantly larger than any other examples available in the literature. We were able to solve such large examples in few hours on a 4GB machine. In all examples we have observed factors of $8\times$ to $100\times$ reduction in the computational time (for the same accuracy) compared to the best available techniques in literature, such as the stochastic collocation method, the stochastic Galerkin method, the Neumann expansion or the combined Neumann Hermite expansion.

Acknowledgment

This work has been partially supported by Mentor Graphics, AMD, IBM, the Semiconductor Corporations and the Interconnect Focus Center, one of five research centers funded under the Focus Center Research Program, a DARPA and Semiconductor Research Corporation program.

APPENDIX

A. EXPRESSIONS FOR EQUATION (18)

$$\mathbf{f}' = \begin{pmatrix} \frac{df}{d\mathbf{u}_n} \\ \frac{df}{d\mathbf{v}_n} \end{pmatrix} \tag{22}$$

$$\mathbf{f}'' = \begin{pmatrix} \frac{d^2f}{d\mathbf{u}_n^2} & \frac{d^2f}{d\mathbf{u}_n d\mathbf{v}_n} \\ \frac{d^2f}{d\mathbf{v}_n d\mathbf{u}_n} & \frac{d^2f}{d\mathbf{v}_n^2} \end{pmatrix} \tag{23}$$

where we define $d(\mathbf{p}) = \mathbf{v}_n^T\mathbf{h}(\mathbf{p})$ and

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

$$\begin{aligned}
\frac{df}{d\mathbf{v}_n} &= -2\mathbb{E}\left[\mathbf{h}(\mathbf{p})\mathbf{u}_n^T\mathbf{A}(\mathbf{p})^T\mathbf{r}_{n-1}(\mathbf{p})\right] \\
&\quad +2\mathbb{E}\left[d(\mathbf{p})\mathbf{h}(\mathbf{p})\mathbf{u}_n^T\mathbf{A}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\mathbf{u}_n\right] \\
\frac{d^2f}{d\mathbf{u}_n^2} &= 2\mathbb{E}\left[d(\mathbf{p})^2\mathbf{A}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\right] \\
\frac{d^2f}{d\mathbf{v}_n^2} &= 2\mathbb{E}\left[\mathbf{u}_n^T\mathbf{A}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\mathbf{u}_n\mathbf{h}(\mathbf{p})\mathbf{h}(\mathbf{p})^T\right] \\
\frac{d^2f}{d\mathbf{v}_n d\mathbf{u}_n} &= -2\mathbb{E}\left[\mathbf{h}(\mathbf{p})\mathbf{r}_{n-1}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\right] \\
&\quad +4\mathbb{E}\left[d(\mathbf{p})\mathbf{h}(\mathbf{p})\mathbf{u}_n^T\mathbf{A}(\mathbf{p})^T\mathbf{A}(\mathbf{p})\right]
\end{aligned}$$

The above expressions can be efficiently expressed using the Hermite expansions of the system matrix $\mathbf{A}(\mathbf{p})$ and the residual $\mathbf{r}(\mathbf{p})$. As an example, the l^{th} row of the second order derivative matrix $\frac{d^2f}{d\mathbf{u}_n d\mathbf{v}_n}$ can be expressed as

$$\frac{d^2f}{d\mathbf{v}_n(l)d\mathbf{u}_n} = \sum_{k=1}^{K}\left(-2\sum_{ij=l,k}\mathbf{R}_{n-1}(:,k)^T\mathbf{A}_i\right.$$
$$\left.+2\left(\sum_{ij,k}\mathbf{u}_n^T\mathbf{A}_i^T\mathbf{v}_n(j)\sum_{ij=l,k}\mathbf{A}_i + \sum_{ij=l,k}\mathbf{u}_n^T\mathbf{A}_i^T\sum_{ij,k}\mathbf{A}_i\mathbf{v}_n(j)\right)\right)$$

where we use the notation $\sum_{ij,k}$ to indicate a summation over all the indices $i,j : \langle H_iH_j, H_k\rangle \neq 0$, 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. REFERENCES

[1] Z. Zhu and J. White, "FastSies: A Fast Stochastic Integral Equation Solver for Modeling the Rough Surface Effect" *International Conference on Computer Aided Design 2005*.

[2] H. Zhu, X. Zeng, W. Cai, J. Xue and D. Zhou, "A sparse grid based spectral stochastic collocation method for variations-aware capacitance extraction of interconnect under nanometer process technology", *Design Automation and Test in Europe, 2007*.

[3] T. El-Moselhy and L. Daniel "Stochastic Integral Equation Solver for Efficient Variation-Aware Interconnect Extraction" *ACM/IEEE Design Automation Conference*, 2008.

[4] D. Xiu and J. Hesthaven, "High-Order Collocation Method for Differential Equations with Random Inputs" *SIAM J. Sci. Comput.*, 2005.

[5] T. El-Moselhy and L. Daniel "Variation-Aware Interconnect Extraction using Statistical Moment Preserving Model Order Reduction" *Design Automation and Test in Europe, 2010*.

[6] T. Moselhy and L. Daniel, "Stochastic High Order Basis Functions for Volume Integral Equation with Surface Roughness," *EPEP 2007*.

[7] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Spring-Verlag, 1991.

[8] M. Kamon, N. Marques and J. White "FastPep: a fast parasitic extraction program for complex three-dimensional geometries" *ACM/IEEE International Conference on Computer Aided Design*, 1997.

[9] K. Nabors and J. White, "FASTCAP A multipole-accelerated 3-D capacitance extraction program," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, pp. 1447-1459, November 1991.

[10] M. Kamon, M. J. Tsuk, and J.K. White, "FastHenry: A multipole-accelerated 3-D inductance extraction program," *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 9, pp. 1750-1758, September 1994.

[11] M.Loeve, *Probability Theory*, Spring-Verlag, 1977.

[12] D.W. Bolton, "The Multinomial Theorem" *The Mathematical Gazette*, December 1968.

[13] A. Nouy, "A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations" *Computer Methods in Applied Mechanics and Engineering*, vol. 196, pp. 4521 - 4537, 2007.