Random Sampling of Moment Graph: A Stochastic Krylov-Reduction Algorithm

Zhenhai Zhu and Joel Phillips
Cadence Berkeley Labs, Berkeley, CA 94704
{zhzhu, jrp}@cadence.com

ABSTRACT

In this paper we introduce a new algorithm for model order reduction in the presence of parameter or process variation. Our analysis is performed using a graph interpretation of the multi-parameter moment matching approach, leading to a computational technique based on Random Sampling of Moment Graph (RSMG). Using this technique, we have developed a new algorithm that combines the best aspects of recently proposed parameterized moment-matching and approximate TBR procedures. RSMG attempts to avoid both exponential growth of computational complexity and multiple matrix factorizations, the primary drawbacks of existing methods, and illustrates good ability to tailor algorithms to apply computational effort where needed. Industry examples are used to verify our new algorithms.

1. INTRODUCTION

A central topic in the recent research literature of electronic design automation is the analysis of circuit performance under parameter variability. It is believed that proper treatment of performance-related yield questions, induced by increased amounts of process and environmental variability, will be essential to economically produce integrated circuits at future technology nodes. It is hoped that a move to analysis tools that explicitly incorporate variability will enable more robust circuit designs.

Our concern in this paper is with model order reduction algorithms for interconnect analysis. For deterministic, non-parameter varying networks, model order reduction is a mature area with several well-established algorithms, the most familiar of which are the Krylov-subspace algorithms PVL [1] and PRIMA [2]. More recently, several approaches have been proposed for reduction that considers process variability effects [3, 4, 5, 6, 7, 8] and they explicitly generate parametric models. Algorithms in [3, 4, 5, 7] are moment matching type algorithms, or related to moment matching type algorithms. Usually the moment-matching methods match multi-dimensional moments involving the parameters and frequency. The algorithm in [6] appears to be quite general and able to model a variety of nonlinear process descriptions. However, for purposes of this paper we will consider it only for purposes of parametric model generation.

The intent of this paper is to investigate some computational quandaries that have arisen in this recent literature. Roughly stated, the moment matching schemes suffer from the problem that the computational effort to generate the model grows exponentially with the number of parameters, if all parameters are treated equally. Several workers have remarked that it is in general *not* a good idea to treat all the moments equally. Generally it is better to match many moments in the frequency variable and only a few in the parameter variables. This is particularly true in the case of process variation where the range of parameter variation is restricted. Unfortunately this observation does not improve matters greatly for the software implementer since it isn't obvious *which* moments are the important ones. Since in a multi-metal-layer interconnect process, the number of parameters can stretch well into double digits, even keeping the multi-dimensional moments associated with frequency and *one* additional order per parameter can result in unacceptably large computation times. To make matters worse in some cases such low-order approximations are not sufficiently accurate.

From the results that have appeared in the literature, the PMTBR method in [6] appears to sidestep this problem. In many cases the number of projection vectors that must be generated in constructing the reduced order model appears to be only weakly related to the number of parameters. In fact often the number of projection vectors that must be generated for engineering accuracy is only 3-5 per input, considerably *less* than the number of parameters. This type of performance cannot be equaled by any of the moment matching schemes that have appeared to date even for affine model dependencies, and certainly not for, e.g., quadratic models where there are 20+ parameters.

On the other hand, the method of [6] suffers from a large constant-factor computational penalty, because a combination of the capacitance and conductance matrices must be factored, rather than just the conductance matrix as in the moment-matching approaches (when expansions around zero frequency are considered). This leads to a large potential slowdown as the capacitance matrix can be much denser than the conductance matrix if coupling effects are considered. What is worse, multiple factorizations must be performed, one for each sampling point. In summary, in building the projection matrices, in the moment-matching approaches, each vector[1] is cheap to compute, but a potentially large number of vectors is required. The TBR-like approach requires few vectors, but each vector costs somewhat more to generate. Table 1 compares the moment matching approaches in [4, 7] and the PMTBR method in [6]. We can see that moment matching approaches and PMTBR method are complementary to each other. This raises the question if there is a new method that combines the strength of the two classes and avoids their weaknesses.

The empirical results from the variational version of PMTBR would seem to predict that the actual dimension of the projection space needed for engineering accuracies is much smaller than would be expected based on moment-counting arguments. This is in line with previous empirical observations about being able to drop various moments in multi-dimensional problems. More importantly, the success of PMTBR suggests a possible strategy to obtain the "important" moments in a reliable and computationally effective way. PMTBR is based on the idea of taking random samples of the range space of Grammian operator, followed by a singular value decomposition to perform rank detection and redundant component removal. Is there a similar approach for moment-matching? Such

[1]We will count in size-N vectors, to make appropriate comparisons with approaches such as in [7] and [5] that work in higher-dimensional spaces.

978-3-9810801-2-4/DATE07 © 2007 EDAA

Table 1: Comparison of PMTBR and Moment Matching Methods. N is the size of original system, p is number of parameters m_q is the order of matched derivative for each λ_i, and N_s is the number of samples used. $O(N^\alpha)$ is the estimated CPU for LU back solve, where $\alpha \in [1,2]$

Methods	model size	construction	projection size
[4]	$O(p^{m_q})$	$O(N^\alpha)$	$O(p^{m_q})$
[7]	$O(m_q)$	$O(N^\alpha)$	$O(p^{m_q})$
PMTBR	N_s	$N_s O(N^\alpha)$	N_s

an approach is not immediately obvious – we cannot, for example, take random combinations of moments, because this would require constructing too many moments, and the numerical errors intrinsic to power methods would obscure any redundant information between the various subspaces.

In this paper we describe a new algorithm, Random Sampling of Moment Graph (RSMG), that uses machinery similar to moment matching. We present a formal way of describing the "moment space" that we term the moment graph. We describe a methodical way of sampling it, and a mechanism for error control and stopping criteria. The accuracy of RSMG, and the number of sample vectors needed for model construction, are competitive with PMTBR. However, due to the recursive formulation, RSMG inherits the potential computational advantages. We do not need to factor each sampling system matrix. Only one LU factorization and a small number of sparse matrix-vector products are involved in each sampling. Hence, on a per-sample basis, the efficiency is improved.

The outline of the paper is as follows. The variation-aware model order reduction is defined in the next section. The moment graph concept is introduced in Section 3, and the recursive sampling procedure is detailed in Section 4. In Section 5, we present empirical results from industrial problems to indicate that the RSMG algorithm is both fast and accurate. Conclusions are given in section 6.

2. PROBLEM DEFINITION

Consider a linear system in standard state-space form

$$E(\bar{\lambda})\frac{dx}{dt} = A(\bar{\lambda})x(t) + Bu(t), \quad y(t) = C^T x(t), \tag{1}$$

where the state vector x has a very large order N and vector $\bar{\lambda}$ contains a list of parameters. These parameters could be process variables such as width and spacing for interconnect analysis problems. They could also be voltage threshold and channel length for analog circuit analysis problems. It is common to use simple functions such as polynomials to approximate the nonlinear functions $E(\bar{\lambda})$ and $A(\bar{\lambda})$ as

$$\begin{aligned} E(\bar{\lambda}) &= E_0 + \sum_{i=1}^{p} E_i\lambda_i + \sum_{j,k=1}^{p} E_{jk}\lambda_j\lambda_k + \ldots \\ A(\bar{\lambda}) &= A_0 + \sum_{i=1}^{p} A_i\lambda_i + \sum_{j,k=1}^{p} A_{jk}\lambda_j\lambda_k + \ldots \end{aligned} \tag{2}$$

where p is the number of parameters, A_i and E_i are the sensitivity of matrix $A(\bar{\lambda})$ and $E(\bar{\lambda})$ with respect to λ_i. Matrices A_{jk} and E_{jk} are second-order derivative with respect to λ_j and λ_k, respectively.

For the sake of clarity, in this paper we will restrict our notation in this paper to the case where both matrices $E(\bar{\lambda})$ and $A(\bar{\lambda})$ are described by an affine model:

$$\begin{aligned} E(\bar{\lambda}) &= E_0 + \sum_{i=1}^{p} E_i\lambda_i \\ A(\bar{\lambda}) &= A_0 + \sum_{i=1}^{p} A_i\lambda_i. \end{aligned} \tag{3}$$

Being based on the general moment-matching ideas, the proposed approaches in this paper encompass higher order models such as (2) and the necessary extensions to the notation and algorithms is fairly straightforward, at least in concept.

The commonly used projection-based approach is to find an orthogonal projection matrix V of size $N \times q$ and transform (1) to

$$\hat{E}(\bar{\lambda})\frac{d\hat{x}}{dt} = \hat{A}(\bar{\lambda})\hat{x}(t) + \hat{B}u(t), \quad y(t) = \hat{C}^T\hat{x}(t), \tag{4}$$

where

$$\hat{E}(\bar{\lambda}) = V^T E_0 V + \sum_{i=1}^{p} V^T E_i V\lambda_i = \hat{E}_0 + \sum_{i=1}^{p} \hat{E}_i\lambda_i \tag{5}$$

$$\hat{A}(\bar{\lambda}) = V^T A_0 V + \sum_{i=1}^{p} V^T A_i V\lambda_i = \hat{A}_0 + \sum_{i=1}^{p} \hat{A}_i\lambda_i \tag{6}$$

$$\hat{B} = V^T B \tag{7}$$

$$\hat{C} = V^T C. \tag{8}$$

Note that the reduced matrices in (5) and (6) are parametrized in the same way as are the original matrices in (3). Algorithms in [4, 7, 6] are different mainly by the ways the projection matrix V is constructed.

3. MOMENT GRAPH

Re-writing the system in (1) in Laplace domain and in view of (3), we have

$$[s(E_0 + \sum_{i=1}^{p} E_i\lambda_i) - (A_0 + \sum_{i=1}^{p} A_i\lambda_i)]x = Bu \tag{9}$$

Multiplying both sides with $-A_0^{-1}$, we obtain

$$[I - \sum_{i=1}^{p} \tilde{A}_i\lambda_i - s\tilde{E}_0 - s\sum_{i=1}^{p} \tilde{E}_i\lambda_i]x = \tilde{B}u, \tag{10}$$

where

$$\tilde{A}_i = -A_0^{-1}A_i \tag{11}$$

$$\tilde{E}_i = A_0^{-1}E_i \tag{12}$$

$$\tilde{B} = -A_0^{-1}B \tag{13}$$

We can formally express the inverse of the system pencil as a multivariable Taylor series

$$\begin{aligned} x(s,\bar{\lambda}) &= [I - \sum_{i=1}^{p} \tilde{A}_i\lambda_i - s\tilde{E}_0 - s\sum_{i=1}^{p} \tilde{E}_i\lambda_i]^{-1}\tilde{B}u \\ &= \sum_{k=0}^{\infty}[\sum_{i=1}^{p} \tilde{A}_i\lambda_i + s\tilde{E}_0 + s\sum_{i=1}^{p} \tilde{E}_i\lambda_i]^k\tilde{B}u \end{aligned} \tag{14}$$

For the sake of clarity, we will use a two-parameter case in this section to explain the concept of moment graph. Extension to multiple parameter cases is straight forward. The simplified equation (14)

for the case of two parameters is

$$\begin{aligned} x(s,\bar{\lambda}) &= \sum_{k=0}^{\infty}[\tilde{A}_1\lambda_1+\tilde{A}_2\lambda_2+s\tilde{E}_0+s\tilde{E}_1\lambda_1+s\tilde{E}_2\lambda_2]^k\tilde{B}u \\ &= [I+\tilde{A}_1\lambda_1+\tilde{A}_2\lambda_2+s\tilde{E}_0+\tilde{A}_1^2\lambda_1^2+\tilde{A}_2^2\lambda_2^2+\tilde{E}_0^2s^2 \\ &\quad +(\tilde{A}_1\tilde{E}_0+\tilde{E}_0\tilde{A}_1+\tilde{E}_1)s\lambda_1+(\tilde{A}_2\tilde{E}_0+\tilde{E}_0\tilde{A}_2+\tilde{E}_2)s\lambda_2 \\ &\quad +(\tilde{A}_1\tilde{A}_2+\tilde{A}_1\tilde{A}_2)\lambda_1\lambda_2+\cdots]\tilde{B}u. \end{aligned} \quad (15)$$

It is shown in [4] that in order to match a certain moment in the transfer function, the corresponding terms in (15) has to be included in the Krylov subspace for the projection matrix. For example, if we want to match the moment of $s\lambda_1$, then $(\tilde{A}_1\tilde{E}_0+\tilde{E}_0\tilde{A}_1+\tilde{E}_1)\tilde{B}$ has to be included in the column span of V. This is a direct extension of the single-parameter Krylov subspace commonly used for the nominal model order reduction [9, 2]. The column span of projection matrix V is

$$\begin{aligned} &span\left\{\tilde{B},\tilde{A}_1\tilde{B},\tilde{A}_2\tilde{B},\tilde{A}_1^2\tilde{B},(\tilde{A}_1\tilde{A}_2+\tilde{A}_2\tilde{A}_1)\tilde{B},\tilde{A}_2^2\tilde{B},\cdots\right\}\bigcup \\ &span\left\{\tilde{E}_0\tilde{B},(\tilde{A}_1\tilde{E}_0+\tilde{E}_0\tilde{A}_1+\tilde{E}_1)\tilde{B},(\tilde{A}_2\tilde{E}_0+\tilde{E}_0\tilde{A}_2+\tilde{E}_2)\tilde{B},\cdots\right\} \\ &\bigcup span\left\{\tilde{E}_0^2\tilde{B},\cdots\right\}\bigcup\cdots \end{aligned} \quad (16)$$

Here we deliberately arrange the span in (16) such that the first line corresponds to the moments with s^0 and the second line corresponds to the moments with s^1. It can be shown that the method in [7] generates exactly the same terms as in (16).

For multi-parameter cases, this process can be very tedious and error-prone. A much more intuitive graphical interpretation of the power series in (14) is proposed in [10]. A simple tree structure graph for the two-parameter case in (15) is shown in Figure 1, where the maximum order of the moments is 3. There are two components in this graph: nodes and directed edges.

- Each node represents one term in (15). A node has three properties: index, value and level. The 3-tuple index of each node, as shown in Figure 1, is the power for s, λ_1 and λ_2, respectively. For example, node index $(1,0,0)$ means $s^1\lambda_1^0\lambda_2^0$ and node index $(1,1,0)$ means $s^1\lambda_1^1\lambda_2^0$. The value at root node $(0,0,0)$ is $\tilde{B}$. The value at other nodes is the sum of all the incoming edge values. For example, the node $(1,0,0)$ has only one incoming edge and its value is $\tilde{E}_0\tilde{B}$. The node $(1,1,0)$ has three incoming edges and its value is $\tilde{A}_1\tilde{E}_0\tilde{B}+\tilde{E}_0\tilde{A}_1\tilde{B}+\tilde{E}_1\tilde{B}$. The level of each node is the summation of its index. For example, the level of node $(1,1,0)$ is 2.

- Each edge represents one of the terms in the parenthesis after the first equal sign in (15). An edge has four properties: multiplier, value, index and level. A multiplier is the matrix A_i and E_i in (15). The multiplier is next to its corresponding edge in Figure 1. The value of each directed edge is the product of the multiplier and the value of the starting node of the edge. The index of an edge is the difference between the indexes of its starting and terminating nodes. For example, the index of the edge between node $(1,0,0)$ and $(2,0,0)$ is $(1,0,0)$. The level of an edge is the summation of its index. Clearly, a level-1 edge can only connect a level-k and a level-$(k+1)$ node.

The depth of a graph is the maximum level of the nodes in the graph or the maximum moment order in (15).

Comparing Figure 1 and equation (15), it is clear that one can directly obtain the coefficients in (15) by reading the value and the index of each node from the graph. This means that one can directly obtain the column span of the projection matrix from the graph. The value of each node in Figure 1 becomes a term in (16).

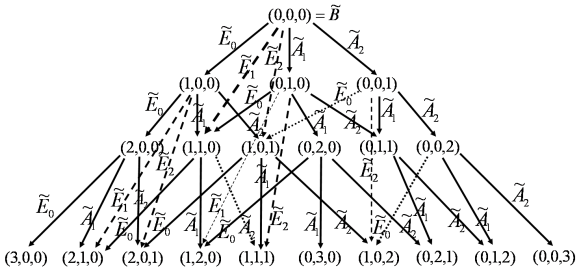

Figure 1: Moment tree graph for two-parameter case in (15). The depth is 3.

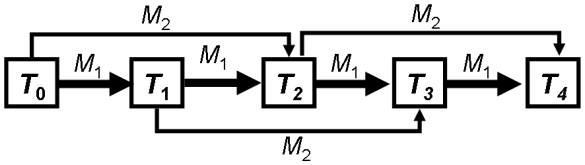

Figure 2: Moment line graph for the recursion (28) where the sampling index k is dropped for the sake of clarity

It is shown in [4] that using the projection matrix V with q columns in (16) can match up the first q moments (corresponding to the first m_q derivatives in each parameter) of the original system in (10). For p parameters and m_q derivatives matched for each parameter, the number of columns in (16) or the order of reduced system is $q=O(p^{m_q})$. This exponential growth is clearly illustrated by the tree graph in Figure 1 and is the fundamental difficulty of the moment matching methods in [4, 7].

4. RANDOM SAMPLING OF MOMENT GRAPH

As pointed out in [4], the matrix containing the columns in (16) is usually rank-deficient, particularly for large number of parameters. This suggests that there should be a more efficient way to generate the projection matrix V.

One of the main ideas in PMTBR algorithm [6] is that a small set of randomly generated sampling state vectors are sufficient to obtain good approximation to projection space. In this section, we show that a similar random sampling idea can be used to efficiently construct a basis for the projection space induced by the moment graph. We first show that one can "compress" the tree structure in Figure 1 into a line structure by summing up weighted nodes and edges of same level. We then use this moment line as the building block to efficiently reconstruct the projection matrix V that has the same subspace span as the original tree structure moment graph up to user-specified accuracy.

4.1 From moment tree to moment line

Evaluating the state vector $x(s,\lambda)$ in (14) at a random set $(s_k,\bar{\lambda}_k)$, we obtain

$$\begin{aligned} x(s_k,\bar{\lambda}_k) &= [I-\sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i}-s_k\tilde{E}_0-s_k\sum_{i=1}^{p}\tilde{E}_i\lambda_{k,i}]^{-1}\tilde{B} \\ &= \tilde{B}+(\sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i}+s_k\tilde{E}_0)\tilde{B}+(s_k\sum_{i=1}^{p}\tilde{E}_i\lambda_{k,i})\tilde{B} \\ &\quad +(\sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i}+s_k\tilde{E}_0+s_k\sum_{i=1}^{p}\tilde{E}_i\lambda_{k,i})^2\tilde{B}+\cdots \end{aligned} \quad (17)$$

To facilitate the discussion in the rest of this paper, we first define a few notations.

The terms in (17) all follow a similar pattern. For example, we can write the first three terms as

$$\tilde{B} = \tilde{B}s_k^0\lambda_{k,1}^0\lambda_{k,2}^0\cdots\lambda_{k,p}^0 \quad (18)$$

$$\tilde{A}_1\tilde{B}\lambda_{k,1} = \tilde{A}_1\tilde{B}s_k^0\lambda_{k,1}^1\lambda_{k,2}^0\cdots\lambda_{k,p}^0 \quad (19)$$

$$\tilde{E}_0\tilde{B}s_k = \tilde{E}_0\tilde{B}s_k^1\lambda_{k,1}^0\lambda_{k,2}^0\cdots\lambda_{k,p}^0. \quad (20)$$

So they are all in the form of $\mathcal{T}s_k^{i_0}\lambda_{k,1}^{i_1}\lambda_{k,2}^{i_2}\cdots\lambda_{k,p}^{i_p}$, where $\mathcal{T}$ is $\tilde{B}$, $\tilde{A}_1\tilde{B}$ and $\tilde{E}_0\tilde{B}$, respectively

DEFINITION 1 (COMPOSITE POWER). *For a term in the form of $\mathcal{T}s_k^{i_0}\lambda_{k,1}^{i_1}\lambda_{k,2}^{i_2}\cdots\lambda_{k,p}^{i_p}$ in (17), its composite power is*

$$L = \sum_{\alpha=0}^{p} i_\alpha \quad (21)$$

REMARK 1. *L is simply the level of the nodes in the tree graph shown in Figure 1.*

DEFINITION 2 (MOMENT RECURSION TERMS). *The m-th moment recursion term $T_m^{(k)}$ is the sum of all terms in (17) whose composite power is m. For example,*

$$T_0 = \tilde{B} \quad (22)$$

$$T_1^{(k)} = (\sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i} + s_k\tilde{E}_0)\tilde{B} \quad (23)$$

$$T_2^{(k)} = (\sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i} + s_k\tilde{E}_0)^2\tilde{B} + (s_k\sum_{i=1}^{p}\tilde{E}_i\lambda_{k,i})\tilde{B} \quad (24)$$

REMARK 2. *T_0, $T_1^{(k)}$ and $T_2^{(k)}$ are the sum of node values scaled by $s_k^{i_0}\lambda_{k,1}^{i_1}\lambda_{k,2}^{i_2}$ at level 0,1 and 2, respectively. Here (i_0,i_1,i_2) is the node index shown in Figure 1.*

DEFINITION 3 (MULTIPLIER). *The level-m multiplier is the sum of all the terms in the parenthesis after the first equal sign in (17) whose composite power is m. For example, the level-1 and level-2 multipliers are respectively*

$$M_1^{(k)} = \sum_{i=1}^{p}\tilde{A}_i\lambda_{k,i} + s_k\tilde{E}_0 \quad (25)$$

$$M_2^{(k)} = s_k\sum_{i=1}^{p}\tilde{E}_i\lambda_{k,i}. \quad (26)$$

REMARK 3. *$M_1^{(k)}$ ($M_2^{(k)}$) is the weighted sum of the multipliers for all edges going from level l to $l+1$ (l to $l+2$). The weighting coefficient is $s_k^{i_0}\lambda_{k,1}^{i_1}\lambda_{k,2}^{i_2}$, where (i_0,i_1,i_2) are the indexes of the edges involved. For example, for the edge between node $(1,1,0)$ and $(0,1,0)$, the weighted edge value is $s_k^1\lambda_{k,1}^0\lambda_{k,2}^0\tilde{E}_0$.*

In view of (22)-(24) and (25)-(26), it is clear that

$$\begin{aligned} T_1^{(k)} &= M_1^{(k)}T_0 \\ T_2^{(k)} &= M_1^{(k)}T_1^{(k)} + M_2^{(k)}T_0. \end{aligned} \quad (27)$$

Hence we speculate the following:

PROPOSITION 1 (MOMENT RECURSION). *Given $M_i^{(k)}, i=1,2$ in (25) and (26), the following recursion holds for $l \geq 2$*

$$T_l^{(k)} = M_1^{(k)}T_{l-1}^{(k)} + M_2^{(k)}T_{l-2}^{(k)}. \quad (28)$$

PROOF. By definition 2, all terms with composite power $l-1$ are in $T_{l-1}^{(k)}$. By definition 3, all the terms in $M_1^{(k)}T_{l-1}^{(k)}$ have composite power l. The same holds true for $M_2^{(k)}T_{l-2}^{(k)}$. Since every term in $T_l^{(k)}$ is the product of some terms in $M_j^{(k)}$ and some terms in $T_{l-j}^{(k)}$, and the union of $M_i^{(k)}T_{l-i}^{(k)}, i=1,2$ cover all the possible combinations, $M_1^{(k)}T_{l-1}^{(k)} + M_2^{(k)}T_{l-2}^{(k)}$ has all the terms with composite power l. Hence (28) directly follows. □

The recursion in (28) can be represented by Figure 2, a much more simplified graph than that in Figure 1. In view of (11)-(12) and (25)-(26), we have

$$M_1^{(k)}R = A_0^{-1}(-\sum_{i=1}^{p}\lambda_{k,i}A_iR + s_kE_0R) \quad (29)$$

$$M_2^{(k)}R = A_0^{-1}(s_k\sum_{i=1}^{p}\lambda_{k,i}E_iR), \quad (30)$$

where R can be a matrix or a vector with conforming size. Clearly, application of each multiplier only involves a few sparse matrix-vector product and one LU back solve. These can be done very efficiently. In addition, we only need to do one LU factorization of sparse matrix A_0 in (29) and (30). So the cost of recursion in (28) is comparable to that of the moment matching method.

4.2 Random Sampling

Figure 2 shows how to generate progressively higher order moments. However, only one node is generated for each moment order. There are two basic ways to generate more nodes for each moment order: the depth-first scheme and the breadth-first scheme.

The depth-first scheme. This scheme repeatedly generate the moment line shown in Figure 2. With a starting seed node and a set of randomly generated $(s_k,\tilde{\lambda}_k)$, one node of progressively higher level is generated from (28), up to certain pre-defined order. The starting seed node and another set of randomly generated $(s_k,\tilde{\lambda}_k)$ are then used to generate another moment line in the same manner. This process is repeated until the newly generated nodes do not add rank to the projection matrix V any more.

The drawback of a depth-first scheme is obvious: it generates the same number of samples for each moment order. In other words, it treats each moment order equally. This is in general not a good idea because usually lower-order moments are more important. So we will focus on the breadth-first scheme in the remaining of this paper.

The breadth-first sampling scheme. With a starting seed node, an order-one node is generated for a set of randomly generated $(s_k,\tilde{\lambda}_k)$. This process is repeated until the newly generated order-one node does not add rank to the projection matrix V. Then an order-two node is generated for a set of randomly generated $(s_k,\tilde{\lambda}_k)$. And this process is repeated until the newly generated order-two node does not add rank to the projection matrix V. We then move to order-three nodes and so forth. The precess stops when the number of samples reach a pre-determined value or the added nodes do not add rank any more.

The key idea in the breadth-first sampling is that the order of the sampled node is not increased until the newly generated node stops adding rank to the projection matrix V. The detailed steps are shown as Algorithm 1.

It should be noted that the numerical stability in Algorithm 1 is ensured because each newly generated column is back orthogonalized against all previous columns. We will not run into numerical stability problem suffered by AWE [11, 12]. But if we directly generate columns in (16) and then do SVD to throw away redundant

columns, we are essentially computing the power series directly. This would be no different than AWE.

Algorithm 1: Random Sampling of Moment Graph
Input: $A_0, A_1, \cdots, A_p; E_0, E_1, \cdots, E_p; B; C$; joint PDF for parameters $\bar{\lambda}$ and frequency s; N_s: number of samplings; tol: truncation tolerance for rank revealing QR
Output: $\hat{A}_0, \hat{A}_1, \cdots, \hat{A}_p; \hat{E}_0, \hat{E}_1, \cdots, \hat{E}_p; \hat{B}; \hat{C}$

(1) LU factor A_0
(2) Compute $\tilde{B}$ in (13) and set $T_0 = \tilde{B}$
(3) $S_1 = T_0$
(4) $P = [S_1]$
(5) $l = 1$
(6) **foreach** $k = 1 : N_s$
(7) Sample s_k $\bar{\lambda}_k$ using given PDFs
(8) **if** $l = 1$
(9) Compute $T_l^{(k)} = M_1^{(k)} S_1$ using (29)
(10) **else**
(11) Compute $T_l^{(k)} = M_1^{(k)} S_1 + M_2^{(k)} S_0$ using (29) and (30)
(12) $P = [P \;\; T_l^{(k)}]$
(13) run rank revealing QR to obtain rank r and the full-rank columns $P_1, P_2, ..., P_r$
(14) **if** $r < k$, i.e., P is rank deficient
(15) $S_0 = S_1, \quad S_1 = P_r$
(16) $l = l + 1$
(17) Use P as projector and compute $\hat{A}_i, \hat{E}_i, \hat{B}, \hat{C}$ as shown in (5)-(8)

LEMMA 1. *Let N_c denote the number of columns in (16) and N_s the number of samples in Algorithm 1. Let tol be the input tolerance to Algorithm 1. If $N_s \geq N_c$ and $tol = 0$, the models generated by Algorithm 1 match the same moments as in (16) with probability one.*

PROOF. By construction, each level-l sampling node generated by Algorithm 1 is a linear combination of all level-l nodes in Figure 1. [2] If $tol = 0$, then this sampling node will always be kept until the number of level-l sampling nodes is equal to the number of level-l nodes in Figure 1. The case that one level-l node in Figure 1 is missed will happen if and only if $s_k = 0$ and $\lambda_{k,i} = 0, i = 1, 2, ..., p$ for all the samples drawn from the joint PDF $P(\lambda, s)$. The set of samples for which this is true is a set of measure zero. Therefore, excepting this measure zero set, after N_s samplings, Algorithm 1 will construct the whole column span with N_c columns in (16). This means, with probability one, that the algorithm recovers exactly the results obtained by moment matching methods. Hence the result directly follows. □

THEOREM 1. *For a given moment matching requirement, the computational complexity of random sampling of moment graph is upper bounded by that of the moment matching methods.*

PROOF. Follows directly from the lemma above. □

REMARK 4. *The $N_s \geq N_c$ samples only happen in the worst case scenario, with very stringent tol. Typically, one needs many fewer samples then N_c.*

[2] This is true even if different sets of $(s_k, \bar{\lambda}_k)$ are used for $T_l^{(k)}$, $T_{l-1}^{(k)}$ and $T_{l-2}^{(k)}$ in (28).

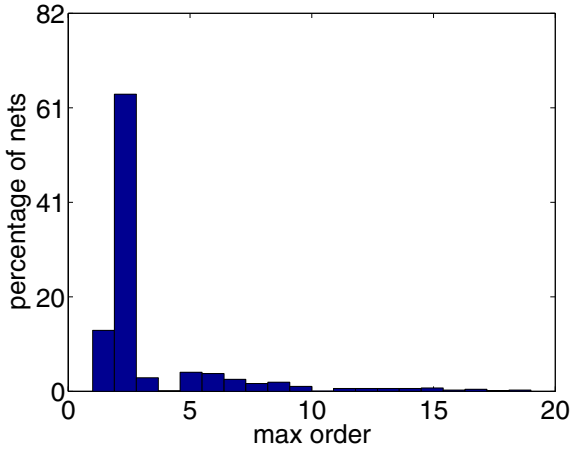

Figure 3: Number of projection vectors computed to reach 1% error at maximum frequency.

5. NUMERICAL RESULTS

We have implemented the RSMG Algorithm 1, PMTBR [6] and PRIMA [2] in C++ and performed various numerical experiments on a desktop PC with Pentium III microprocessor. In PRIMA implementation, we used the projection matrix generated from circuit with nominal process parameters to reduce the parameterized model as shown in (5-8). Though this is not the original PRIMA, we still call it 'PRIMA' due to the same projection.

There are two types of experiments. First we run RSMG Algorithm 1 on a set of several thousand signal nets extracted from some industrial test cases. This is to show RSMG Algorithm is practically useful. Then, we pick a representative case to perform the in-depth analysis and compare RSMG to PMTBR and PRIMA.

The process variables considered here are inter-level spacing, dielectric constant, interconnect width, and interconnect thickness/spacing for each layer. This results in more than thirty process variables in total. Affine models were built for each extracted capacitance and resistance.

5.1 Histogram of projector size

Figure 3 shows the results of applying the RSMG Algorithm 1 to the previously mentioned large set of signal nets. We tabulate the order needed to reach engineering accuracy of 1% in the transfer function at the highest frequency (10GHz) considered, over a large set of randomly sampled process conditions. As can be seen, usually a model can be constructed with 2 or 3 samples, and the vast majority of cases can be handled with fewer than five samples.

5.2 Accuracy

Now we focus on a representative case picked from the many signal nets used in last section. This example has 550 nodes and 31 process parameters. In Figure 4, we compare the maximum relative error in transfer function at the highest frequency (10GHz) considered, over a large set of randomly sampled process conditions. Note that we have extended the accuracy range to ten digits. While not relevant in ultimate application, investigation of such a range of precision is important in assessing algorithm stagnation, which can indicate a flaw or hidden inefficiency in the approach.

The tolerance for incremental QR factorization in Algorithm 1 decides the number of samples at each level. As indicated by lemma 1, at strict tolerances, Algorithm 1 reduces to the full moment matching, which is already known to be quite inefficient when many parameters are involved. This is clearly shown in Figure 4. We use three difference tolerance values in Figure 4. For this particular

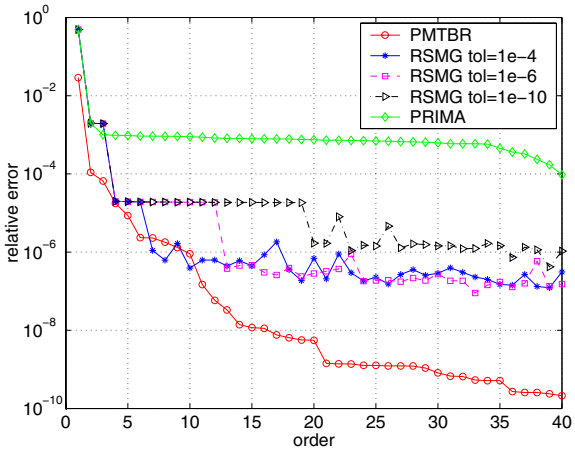

Figure 4: Max relative error in transfer function for a large set of randomly selected points in the 31-dimension parameter space.

example, it appears that $tol = 1e-4$ strikes a good balance between the breadth at each level and depth of the sampling. From our experiments, a good tolerance value depends on the range of parameters and is critical to the efficiency of the Algorithm 1.

Figure 4 shows that the accuracy of PMTBR and RSMG is comparable up to order 10, at which point the relative error has already dropped to $1e-6$. So for all engineering intents and purposes, the accuracy of RSMG is very competitive comparing to that of PMTBR. The stagnation of RSMG at higher order seems to imply two things: 1) More sophisticated sampling schemes are to be used for high accuracy applications; 2) The projection space generated by moment matching method, as illustrated by the moment graph in Figure 1, is genuinely different from that approximated by the PMTBR method. This issue will be a good direction for future investigation.

5.3 Speed

There are three main steps in RSMG, PMTBR and PRIMA: A) generating new columns in Krylov sub-space; B) back orthogonalization using incremental QR factorization; C) reducing the system matrices *ABCE* using (5-8). The difference is mainly in step A. Depending on the over all percentage of step A in the whole process, the speed up due to RSMG may vary.

We use the same example in Figure 4 to compare CPU time by different algorithms. The order or the projection size is fixed to be 4 which gives us about 4-digits accuracy (PRIMA gives about 3-digit accuracy). Table 2 shows the ratio of CPU time used by RSMG and PMTBR to that of PRIMA. It is clear from Table 2 that while the overall speed up due to RSMG is about 20%, the speedup at step A is 3.5 times. RSMG only needs to peform one LU factorization while PMTBR needs to do 4. But the operations involving the sensitivity matrices A_i and E_i in (29-30) somewhat offset RSMG's advantage. This is also why the step A of the RSMG is more expensive than that of PRIMA. In addition, the example we used here only involves diagonal matrices C_0 and C_i, i.e., there is no coupling capacitance. This means that factorizing A_0 and $sE(\tilde{\lambda}) + A(\tilde{\lambda})$ takes the same CPU time (sample frequency s is usually a real number in PMTBR). This also takes away an important advantage of using RSMG.

It should be noted that in full chip simulation, one typically has hundreds thousands of nets like the example shown here. So a 20% reduction in CPU time can be significant. In addition, we have not exploit the high sparsity of matrices A_i and E_i in our implementation. There is considerable room for efficiency improvement in performing (29-30).

Table 2: The ratio of CPU time used by PMTBR and RSMG to that used by PRIMA with fixed order=4

Methods	total	step A
PMTBR	1.33	7
RSMG	1.05	2

6. CONCLUSIONS

In this paper we present a new algorithm based on random sampling of moment graph (RSMG). Empirical studies indicate that the RSMG algorithm circumvents the exponential complexity of full moment matching approaches (e.g. [4, 7]) in the average case. We also show that RSMG is competitive in accuracy comparing to the approximate TBR method (PMTBR [6]), but has about 20% advantage in CPU time.

REFERENCES

[1] P. Feldmann and R. W. Freund, "Efficient linear circuit analysis by Padé approximation via the Lanczos process," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 5, pp. 639–649, May 1995.

[2] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 645–654, August 1998.

[3] Y. Liu, L. T. Pileggi, and A. J. Strojwas, "Model order reduction of RC(L) interconnect including variational analysis," in 36^{th} *ACM/IEEE Design Automation Conference*, June 1999, pp. 201–206.

[4] L. Daniel, O. Siong, L. Chay, K. Lee, and J. White, "A multi-parameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 5, pp. 678–693, May 2004.

[5] J. Wang, P. Ghanta, and S. Vrudhula, "Stochastic analysis of interconnect performance in the presence of process variations," in *International Conference on Computer Aided-Design*, San Jose, CA, November 2004, pp. 880–886.

[6] Joel R. Phillips, "Variational interconnect analysis via PMTBR," in *International Conference on Computer Aided-Design*, San Jose, CA, November 2004, pp. 872–879.

[7] X. Li, P. Li, and L. Pileggi, "Parameterized interconnect order reduction with Explicit-and-Implicit multi-Parameter moment matching for Inter/Intra-Die variations," in *International Conference on Computer Aided-Design*, San Jose, CA, November 2005, pp. 806–812.

[8] James D. Ma and Rob A. Rutenbar, "Interval-valued reduced order statistical interconnect modeling," in *International Conference on Computer Aided-Design*, 2004.

[9] Eric Grimme, *Krylov Projection Methods for Model Reduction*, Ph.D. thesis, Coordinated-Science Laboratory, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 1997.

[10] Jung Hoon Lee, Luca Daniel, and Jacob White, "Formalization of the moment matching graph idea," *Technical report of Research Labs of Electronics, MIT*, 2004.

[11] Lawrence T. Pillage and Ronald A. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 352–366, April 1990.

[12] Peter Feldmann and Roland W. Freund, "Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm," in 32^{nd} *ACM/IEEE Design Automation Conference*, San Francisco, CA, 1995, pp. 474–479.