Big-Data Tensor Recovery for High-Dimensional Uncertainty Quantification of Process Variations

Zheng Zhang, *Member, IEEE*, Tsui-Wei Weng, *Student Member, IEEE*, and Luca Daniel, *Member, IEEE*
(Invited Paper)

***Abstract*— Fabrication process variations are a major source of yield degradation in the nanoscale design of integrated circuits (ICs), microelectromechanical systems (MEMSs), and photonic circuits. Stochastic spectral methods are a promising technique to quantify the uncertainties caused by process variations. Despite their superior efficiency over Monte Carlo for many design cases, stochastic spectral methods suffer from the curse of dimensionality, i.e., their computational cost grows very fast as the number of random parameters increases. In order to solve this challenging problem, this paper presents a high-dimensional uncertainty quantification algorithm from a big data perspective. Specifically, we show that the huge number of (e.g., 1.5×10^{27}) simulation samples in standard stochastic collocation can be reduced to a very small one (e.g., 500) by exploiting some hidden structures of a high-dimensional data array. This idea is formulated as a tensor recovery problem with sparse and low-rank constraints, and it is solved with an alternating minimization approach. The numerical results show that our approach can efficiently simulate some IC, MEMS, and photonic problems with over 50 independent random parameters, whereas the traditional algorithm can only deal with a small number of random parameters.**

***Index Terms*— High dimensionality, integrated circuits (ICs), integrated photonics, microelectromechanical system (MEMS), polynomial chaos, process variation, stochastic simulation, tensor, uncertainty quantification.**

I. Introduction

FABRICATION process variations (surface roughness of interconnects and nano-photonic devices and random doping effects of transistors) have become a critical issue in nanoscale design, because they can significantly influence chip performance and decrease product yield [2]. Efficient stochastic modeling and simulation algorithms should be developed and implemented in electronic design automation (EDA) software in order to estimate and control the uncertainties in a nanoscale chip design. For several decades, Monte Carlo techniques [3], [4] have been the mainstream stochastic simulators in commercial tools due to their ease of implementation. Nevertheless, they have a slow convergence rate, and thus generally require a large number of repeated simulations. In recent years, the emerging stochastic spectral methods [5], [6] have been investigated, and they prove efficient for many design cases with a small number of parameters, including integrated circuits (ICs) [7]–[17], microelectromechanical systems (MEMSs) [18], [19], and photonic circuits [20], [21].

The key idea of stochastic spectral methods is to approximate a stochastic solution (e.g., the uncertain voltage or power dissipation of a circuit) as the linear combination of some specialized basis functions such as generalized polynomial chaos [22]. Two main classes of simulators have been implemented to obtain the coefficients of each basis functions. In an intrusive (i.e., nonsampling) simulator such as stochastic Galerkin [5] and stochastic testing [9], a deterministic equation is constructed such that the unknown coefficients can be directly computed by a single simulation. Generally, stochastic testing [9] is more efficient than stochastic Galerkin for many applications, since the resulting Jacobian matrix can be decoupled and the step sizes in transient analysis can be selected adaptively. In a sampling-based simulator such as stochastic collocation [6], a few solution samples are first computed by repeated simulations. Then, some postprocessing techniques are used to reconstruct the unknown coefficients. The methods in [15] and [16] reduce the complexity by selecting critical samples or critical basis functions. When the number of random parameters is small, these solvers can provide highly accurate solutions with significantly (e.g., $100\times$ to $1000\times$) higher efficiency than Monte Carlo. Unfortunately, stochastic spectral methods suffer from the curse of dimensionality, i.e., their computational cost grows very fast as the number of random parameters increases.

A. Related Work

Several advanced uncertainty quantification algorithms that solve high-dimensional problems have been reported in the literature. The following are some representative high-dimensional solvers for IC and MEMS applications.

1) *Sparse techniques:* In a high-dimensional polynomial chaos expansion, very often the coefficients of most basis functions are close to zero. In [18], this property was exploited for analog IC applications using adaptive analysis of variance (ANOVA) [23]–[25]. In [26], compressed sensing [27] was employed to minimize the ℓ_1-norm of the coefficient vector.
2) *Matrix low-lank approach:* The intrusive solver reported in [28] recognizes that the polynomial chaos

Manuscript received August 29, 2016; revised November 4, 2016; accepted November 7, 2016. This work was supported in part by the NSF NEEDS Program and in part by the AIM Photonics Program. Recommended for publication by Associate Editor S. Grivet-Talocia upon evaluation of reviewers' comments.

The authors are with the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: z zhang@mit.edu; twweng@mit.edu; luca@mit.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCPMT.2016.2628703

expansion can be expressed by an order of magnitude smaller specialized polynomial basis and computes its coefficients and polynomials iteratively by a nonlinear optimization starting from the most dominant terms.

3) *Model order reduction:* In [29], an efficient reduced model was used to obtain most solution samples within a sampling-based solver. The reduced model is constructed by refinements. When a parameter value is detected for which the reduced model is inaccurate, the original large-scale equation is solved to update the model on the fly.
4) *Hierarchical Approach:* Using generalized polynomial chaos expansions to describe devices and subsystems, the tensor-train hierarchical uncertainty quantification framework in [19] was able to handle complex systems with many uncertainties. The basic idea is to treat the stochastic output of each device/subsystem as a new random input. As a result, the system-level uncertainty quantification has only a small number of random parameters when new basis functions are used.

B. Contributions

This paper presents a sampling-based high-dimensional stochastic solver from a big data perspective. The standard stochastic collocation approach is well known to be affected by the curse of dimensionality, and it was applicable only to problems with a few random parameters. In this paper, we represent the huge number of required solution samples as a tensor, which is a high-dimensional generalization of a matrix or a vector [30].[1] In order to overcome the curse of dimensionality in stochastic collocation, we suggest a tensor recovery approach: We use a small number of simulation samples to estimate the whole tensor. This idea is implemented by exploiting the hidden low-rank property of a tensor and the sparsity of a generalized polynomial chaos expansion. Numerical methods are developed to solve the proposed tensor recovery problem. We also apply this framework to simulate some IC, MEMS, and photonic design cases with lots of process variations and compare it to standard-sampling-based stochastic spectral methods.

C. Paper Organization

This paper is organized as follows. Section II briefly reviews stochastic collocation and tensor computation. Section III describes our tensor recovery model to reduce the computational cost of high-dimensional stochastic collocation. Numerical techniques are described in Section IV to solve the resulting optimization problem. Section V explains how to obtain a generalized polynomial chaos expansion from the obtained tensor factors. The simulation results of some high-dimensional IC, MEMS, and photonic circuit examples are reported in Section VI. Finally, Section VII concludes this paper and points out future work.

[1]Tensor is an efficient tool to reduce the computational and memory cost of many problems (e.g., deep learning and data mining) in big data analysis.

II. Preliminaries

A. Uncertainty Quantification Using Stochastic Collocation

Let the vector $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_d] \in \mathbb{R}^d$ denote a set of mutually independent random parameters that describe process variations (e.g., deviation of transistor threshold voltage and thickness of a dielectric layer in MEMS fabrication). We intend to estimate the uncertainty of an output of interest $y(\boldsymbol{\xi})$. This parameter-dependent output of interest can describe, for instance, the power consumption of an analog circuit or the frequency of a MEMS resonator.

1) Generalized Polynomial Chaos: Assuming that y smoothly depends on $\boldsymbol{\xi}$ and that y has a bounded variance,[2] we apply a truncated generalized polynomial chaos expansion [22] to approximate the stochastic solution

$$y(\boldsymbol{\xi}) \approx \sum_{|\boldsymbol{\alpha}|=0}^{p} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \text{ with } \mathbb{E}[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \Psi_{\boldsymbol{\beta}}(\boldsymbol{\xi})] = \delta_{\boldsymbol{\alpha},\boldsymbol{\beta}}. \quad (1)$$

Here, the operator $\mathbb{E}$ denotes the expectation, δ denotes a Delta function, and the basis functions $\{\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}$ are orthonormal polynomials, where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_d] \in \mathbb{N}^d$ is a vector indicating the highest polynomial order of each parameter in the corresponding basis. The total polynomial order $|\boldsymbol{\alpha}| = |\alpha_1| + \ldots + |\alpha_d|$ is bounded by p, and thus the total number of basis functions is $K = (p+d)!/(p!d!)$. Since $\boldsymbol{\xi}$ are mutually independent, for each parameter ξ_k, one can first construct a set of univariate orthonormal polynomials $\phi_{k,\alpha_k}(\xi_k)$ with $\alpha_k = 0, \ldots, p$. Then, the multivariate polynomial basis function with index $\boldsymbol{\alpha}$ is

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \phi_{k,\alpha_k}(\xi_k). \quad (2)$$

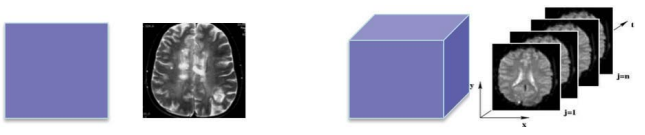
The univariate polynomial functions can be obtained by the three-term recurrence relation in [31], and the main steps are summarized in Appendix A.

2) Stochastic Collocation: Since all basis functions in (1) are orthonormal to each other, the coefficient $c_{\boldsymbol{\alpha}}$ can be obtained by a projection framework

$$c_{\boldsymbol{\alpha}} = \int_{\mathbb{R}^d} y(\boldsymbol{\xi}) \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) \rho(\boldsymbol{\xi}) d\boldsymbol{\xi}, \text{ with } \rho(\boldsymbol{\xi}) = \prod_{k=1}^{d} \rho_k(\xi_k). \quad (3)$$

Note that $\rho(\boldsymbol{\xi})$ is the joint probability density function of vector $\boldsymbol{\xi}$ and $\rho_k(\xi_k)$ is the marginal density of ξ_k. The above integral needs to be evaluated with a proper numerical technique. Popular integration techniques include randomized approaches such as Monte Carlo [3] and deterministic approaches like tensor product and sparse grid [32]. Monte Carlo is feasible for extremely high-dimensional problems, but its numerical accuracy is low. Deterministic approaches can generate very accurate results using a low-order quadrature rule, but they are feasible only for problems with a small or medium number of random parameters due to the curse of dimensionality. This paper considers the tensor product implementation, which was regarded as much less efficient than sparse grid techniques in almost all previous publications.

[2]In this paper, we assume that y is a scalar.



(a) matrix (b) 3-D tensor

Fig. 1. (a) 2-D data array (e.g., a medical image) is a matrix. (b) 3-D data array (e.g., multiple slices of images) is a tensor.

We briefly introduce the idea of tensor product numerical integration. Let $\{(\xi_k^{i_k}, w_k^{i_k})\}_{i_k=1}^{n}$ be n pairs of 1-D quadrature points (or samples) and weights for parameter ξ_k. Such quadrature points and weights can be obtained by various numerical techniques, which can be found in [33]. In this paper, we use the Gauss quadrature rule [34] to generate such 1-D samples and weights, as summarized in Appendix B. A Gauss quadrature rule with n samples can generate exact results when the univariate integrand is a polynomial function of ξ_k and when the highest polynomial degree is not higher than $2n-1$. By tensorizing all 1-D quadrature points/weights, the d-dimensional integral in (3) can be evaluated as

$$c_{\boldsymbol{\alpha}} = \sum_{1\le i_1,\ldots,i_d\le n} y(\boldsymbol{\xi}_{i_1\ldots i_d})\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{i_1\ldots i_d})w_{i_1\ldots i_d}. \tag{4}$$

Here, $\boldsymbol{\xi}_{i_1\ldots i_d} = [\xi_1^{i_1}, \ldots, \xi_d^{i_d}]$ and $w_{i_1\ldots i_d} = w_1^{i_1}\ldots w_d^{i_d}$ are the resulting multidimensional quadrature samples and weights, respectively. Obtaining each solution sample $y(\boldsymbol{\xi}_{i_1\ldots i_d})$ may require a time-consuming numerical simulation. For instance, a periodic steady-state solver may be called to compute the frequency of an oscillator. In device modeling, a large-scale solver must be called to solve a complex partial differential equation or integral equation for each quadrature sample. The numerical implementation (4) requires n^d times of such expensive device or circuit simulations.

B. Tensor and Tensor Decomposition

1) Tensor: Tensor is a high-dimensional generalization of matrix. A matrix $\mathbf{X} \in \mathbb{R}^{n_1\times n_2}$ is a second-order tensor, and its element indexed by $\mathbf{i} = (i_1, i_2)$ can be denoted by $x_{i_1 i_2}$ or $\mathbf{X}(\mathbf{i})$. For a general dth-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1\times\ldots n_d}$, its element indexed by $\mathbf{i} = (i_1, \ldots, i_d)$ can be denoted by $x_{i_1\ldots i_d}$ or $\boldsymbol{\mathcal{X}}(\mathbf{i})$. Here, the integer $k \in [1, d]$ is the index for a mode of $\boldsymbol{\mathcal{X}}$. Fig. 1 shows a matrix and a third-order tensor.

Given any two tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ of the same size, their inner product is defined as

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}} \rangle := \sum_{i_1\ldots i_d} x_{i_1\ldots i_d} y_{i_1\ldots i_d}.$$

The Frobenius norm of tensor $\boldsymbol{\mathcal{X}}$ is further defined as $||\boldsymbol{\mathcal{X}}||_F := (\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}} \rangle)^{1/2}$.

2) Tensor Decomposition: A tensor $\boldsymbol{\mathcal{X}}$ is rank-1 if it can be written as the outer product of some vectors

$$\boldsymbol{\mathcal{X}} = \mathbf{u}_1 \circ \cdots \circ \mathbf{u}_d \;\Leftrightarrow\; x_{i_1\ldots i_d} = \mathbf{u}_1(i_1)\cdots\mathbf{u}_d(i_d) \tag{5}$$

where $\mathbf{u}_k(i_k)$ denotes the i_kth element of vector $\mathbf{u}_k \in \mathbb{R}^{n_k}$. Similar to matrices, a low-rank tensor can be written as a canonical decomposition [35], which expresses $\boldsymbol{\mathcal{X}}$ as the sum of some rank-1 tensors

$$\boldsymbol{\mathcal{X}} = \mathbb{T}_{\text{cp}}(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}) := \sum_{j=1}^{r} \mathbf{u}_1^j \circ \cdots \circ \mathbf{u}_d^j. \tag{6}$$

Here, $\mathbf{U}^{(k)} = [\mathbf{u}_k^1, \ldots, \mathbf{u}_k^r] \in \mathbb{R}^{n_k\times r}$ is a matrix including all factors corresponding to mode k, operator $\mathbb{T}_{\text{cp}}$ converts all matrix factors to a tensor represented by canonical decomposition, and the minimum integer r that ensures (6) to hold is called *tensor rank*. As a demonstration, Fig. 2 shows the low-rank factorizations of a matrix and third-order tensor, respectively. Tensor decomposition (6) can significantly reduce the cost of storing high-dimensional data arrays. For simplicity, let us assume $n_k = n$. Directly representing tensor $\boldsymbol{\mathcal{X}}$ requires storing n^d scalars, whereas only ndr scalars need to be stored if the above low-rank factorization exists.

Note that there are other kinds of tensor factorizations such as Tucker decomposition [36] and tensor-train decomposition [37]. We introduce only canonical decomposition in this paper because we will use it to solve high-dimensional uncertainty quantification problems in the subsequent sections. Interested readers are referred to [30] for a detailed survey of tensor decompositions, as well as [38] for a tutorial with applications in EDA.

III. Tensor Recovery Approach

Formulation (4) was applicable to only problems with five or six random parameters due to the n^d simulation samples. This section describes our tensor recovery approach that can significantly reduce the computational cost of tensor product stochastic collocation. With this framework, (4) can be more efficient than sparse grid approaches and Monte Carlo simulation for many high-dimensional design cases.

A. Reformulating Stochastic Collocation With Tensors

We first define the following two tensors:

1) tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1\times\ldots\times n_d}$, with $n_k = n$ and each element being $y_{i_1\ldots i_d} = y(\boldsymbol{\xi}_{i_1\ldots i_d})$;
2) tensor $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \in \mathbb{R}^{n_1\times\ldots\times n_d}$, with $n_k = n$ and its element indexed by $(i_1, \ldots i_d)$ being $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_{i_1\ldots i_d})w_{i_1\ldots i_d}$.

Tensor $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ depends only on the basis function in (1) and the multidimensional quadrature weights in (4). Furthermore, according to (2), it is straightforward to see that $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ is a rank-1 tensor with the following canonical decomposition:

$$\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} = \mathbf{v}_1^{\alpha_1} \circ \cdots \circ \mathbf{v}_d^{\alpha_d}$$

with

$$\mathbf{v}_k^{\alpha_k} = \left[\phi_{k,\alpha_k}(\xi_k^1)w_k^1, \ldots, \phi_{k,\alpha_k}(\xi_k^n)w_k^n\right]^T \in \mathbb{R}^{n\times 1}. \tag{7}$$

Note that $\xi_k^{i_k}$ and $w_k^{i_k}$ are the i_kth 1-D quadrature point and weight for parameter ξ_k, respectively, as described in Section II-A.

With the above two tensors, (4) can be written in the following compact form:

$$c_{\boldsymbol{\alpha}} = \langle \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle. \tag{8}$$

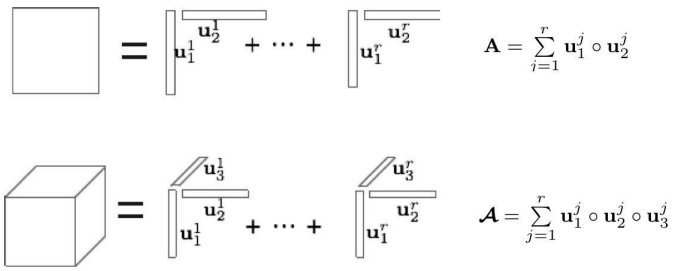


Fig. 2. Top: low-rank factorization of a matrix. Bottom: canonical decomposition of a third-order tensor.

Since $\mathcal{W}_{\boldsymbol{\alpha}}$ is straightforward to obtain, the main computational cost is to compute $\mathcal{Y}$. Once $\mathcal{Y}$ is computed, the $c_{\boldsymbol{\alpha}}$ values and thus the generalized polynomial chaos expansion (1) can be obtained easily. Unfortunately, directly computing $\mathcal{Y}$ is impossible for high-dimensional cases since it requires simulating a specific design case n^d times.

B. Tensor Recovery Problem (Ill-Posed)

We define two index sets.

1) Let $\mathcal{I}$ include all indices $(i_1, \ldots, i_d)$ for the elements of $\mathcal{Y}$. The number of elements in $\mathcal{I}$, denoted by $|\mathcal{I}|$, is n^d.
2) Let Ω be a small subset of $\mathcal{I}$, with $|\Omega| \ll |\mathcal{I}|$. For each index $(i_1, \ldots, i_d) \in \Omega$, the corresponding solution sample $y_{i_1 \ldots i_d}$ is already obtained by a numerical simulator.

In order to reduce the computational cost, we aim at estimating the whole tensor $\mathcal{Y}$ using the small number of available simulation data specified by Ω. With the sampling set Ω, a projection operator $\mathbb{P}$ is defined for $\mathcal{Y}$

$$\mathcal{B} = \mathbb{P}_\Omega(\mathcal{Y}) \;\Leftrightarrow\; b_{i_1 \ldots i_d} = \begin{cases} y_{i_1 \ldots i_d}, & \text{if } (i_1, \ldots, i_d) \in \Omega \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

We want to find a tensor $\mathcal{X}$ such that it matches $\mathcal{Y}$ for the elements specified by Ω

$$\|\mathbb{P}_\Omega(\mathcal{X} - \mathcal{Y})\|_F^2 = 0. \tag{11}$$

However, this problem is *ill-posed* because any value can be assigned to $x_{i_1 \ldots i_d}$ if $(i_1, \ldots, i_d) \notin \Omega$.

C. Regularized Tensor Recovery Model

In order to make the tensor recovery problem well posed, we add the following constraints based on heuristic observations and practical implementations.

1) *Low-Rank Constraint:* Very often we observe that the high-dimensional solution data array $\mathcal{Y}$ has a low tensor rank. Therefore, we expect that its approximation $\mathcal{X}$ has a low-rank decomposition described in (6).
2) *Sparse Constraint:* As shown in the previous work of ANOVA decomposition [18] and compressed sensing [26], most of the coefficients in a high-dimensional generalized polynomial chaos expansion have a very small magnitude. This implies that the ℓ_1-norm of a vector collecting all coefficients $c_{\boldsymbol{\alpha}}$, which is computed as

$$\sum_{|\boldsymbol{\alpha}|=0}^{p} |c_{\boldsymbol{\alpha}}| \approx \sum_{|\boldsymbol{\alpha}|=0}^{p} |\langle \mathcal{X}, \mathcal{W}_{\boldsymbol{\alpha}} \rangle| \tag{12}$$

should be very small.

Finalized Tensor Recovery Model: Combining the above low-rank and sparse constraints together, we suggest the finalized tensor recovery model (9), as shown at the bottom of this page, to compute $\mathcal{X}$ as an estimation of $\mathcal{Y}$. In this formulation, $\mathcal{X}$ is assumed to have a rank-r decomposition and we compute its matrix factors $\mathbf{U}^{(k)}$ instead of the whole tensor $\mathcal{X}$. This treatment has a significant advantage: The number of unknown variables is reduced from n^d to dnr, which is now a linear function of parameter dimensionality d.

D. Cross Validation

An interesting question is how accurate is $\mathcal{X}$ compared to the exact tensor $\mathcal{Y}$. Our tensor recovery formulation enforces consistency between $\mathcal{X}$ and $\mathcal{Y}$ at the indices specified by Ω. It is desired that $\mathcal{X}$ also has a good predictive behavior—$x_{i_1 \ldots i_d}$ is also close to $x_{i_1 \ldots i_d}$ for $(i_1, \ldots, i_d) \notin \Omega$. In order to measure the predictive property of our results, we define a heuristic prediction error

$$\epsilon_{\text{pr}} = \sqrt{\frac{\sum_{(i_1,\ldots,i_d)\in\Omega'} (x_{i_1 \ldots i_d} - y_{i_1 \ldots i_d})^2 \; w_{i_1 \ldots i_d}}{\sum_{(i_1,\ldots,i_d)\in\Omega'} (y_{i_1 \ldots i_d})^2 \; w_{i_1 \ldots i_d}}}.$$

Here, $\Omega' \subset \mathcal{I}$ is a small-size index set such that $\Omega' \cap \Omega = \emptyset$. The solution $\mathcal{X}$ is regarded as a good approximate to $\mathcal{Y}$ if ϵ_{pr} is small; then, (1) can be obtained accurately using (8), and the statistical behavior (e.g., probability density function) of $y(\boldsymbol{\xi})$ can be well predicted. Estimating ϵ_{pr} requires simulating the design problem at some extra quadrature samples. However, a small-size Ω' can provide a good heuristic estimation.

$$\min_{\mathbf{U}^{(1)},\ldots,\mathbf{U}^{(d)}\in\mathbb{R}^{n\times r}} f(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}) = \frac{1}{2}\|\mathbb{P}_\Omega(\mathbb{T}_{\text{cp}}(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}) - \mathcal{Y})\|_F^2 + \lambda \sum_{|\boldsymbol{\alpha}|=0}^{p} |\langle \mathbb{T}_{\text{cp}}(\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(d)}), \mathcal{W}_{\boldsymbol{\alpha}} \rangle| \tag{9}$$

Algorithm 1 Alternating Minimization for Solving (9)

1: Initialize: $\mathbf{U}^{(k),0} \in \mathbb{R}^{n\times r}$ for $k = 1, \ldots d$;
2: **for** $l = 0, 1, \ldots$
3: **for** $k = 1, \ldots, d$ **do**
4: solve (13) by Alg. 2 to obtain $\mathbf{U}^{(k),l+1}$;
5: **end for**
6: **break** if converged;
7: **end for**
8: **return** $\mathbf{U}^{(k)} = \mathbf{U}^{(k),l+1}$ for $k = 1, \ldots, d$.

At present, we do not have a rigorous approach to find the optimal values of λ and r. In practice, their values are chosen heuristically. Specifically, we increment λ and r until ϵ_{pr} becomes small enough. Occasionally, optimization problem (9) may be solved several times for different values of λ and r. However, like other sampling-based stochastic solvers, the computational cost of postprocessing [i.e., solving (9)] is generally negligible compared to the cost of simulating solution samples indexed by $\boldsymbol{\Omega}$.

IV. Solve Problem (9)

The optimization problem (9) is solved iteratively in our implementation. Specifically, starting from a provided initial guess of the low-rank factors $\{\mathbf{U}^{(k)}\}_{k=1}^d$, alternating minimization is performed recursively using the result of the previous iteration as a new initial guess. Each iteration of alternating minimization consists of d steps. At the kth step, the kth-mode factor matrix $\mathbf{U}^{(k)}$ corresponding to parameter ξ_k is updated by keeping all other factors fixed and by solving (9) as a convex optimization problem.

A. Outer Loop: Alternating Minimization

1) Algorithm Flow: We use an iterative algorithm to solve (9). Let $\mathbf{U}^{(k),l}$ be the mode$-k$ factors of $\boldsymbol{\mathcal{X}}$ after l iterations. Starting from an initial guess $\{\mathbf{U}^{(k),0}\}_{k=1}^d$, we perform the following iterations.

i) At iteration $l+1$, we use $\{\mathbf{U}^{(k),l}\}_{k=1}^d$ as an initial guess and obtain updated tensor factors $\{\mathbf{U}^{(k),l+1}\}_{k=1}^d$ by alternating minimization.
ii) Each iteration consists of d steps; at the kth step, $\mathbf{U}^{(k),l+1}$ is obtained by solving

$$\mathbf{U}^{(k),l+1} = \arg\min_{\mathbf{X}} f(\ldots, \mathbf{U}^{(k-1),l+1}, \mathbf{X}, \mathbf{U}^{(k+1),l}, \ldots). \tag{13}$$

Since all factors except that of mode k are fixed, (13) becomes a convex optimization problem and its global minimum can be computed by the solver in Section IV-B. The alternating minimization method ensures that the cost function decreases monotonically to a local minimal. The pseudocodes are summarized in Algorithm 1.

2) Convergence Criteria: With matrices $\{\mathbf{U}^{(k),l}\}_{k=1}^d$ obtained after l iterations of the outer loops of Algorithm 1, we define

$$f_l := f(\mathbf{U}^{(1),l}, \ldots, \mathbf{U}^{(d),l})$$
$$\boldsymbol{\mathcal{X}}_l := \mathbb{T}_{\text{cp}}(\mathbf{U}^{(1),l}, \ldots, \mathbf{U}^{(d),l})$$
$$c_{\boldsymbol{\alpha}}^l := \langle \boldsymbol{\mathcal{X}}_l, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle.$$

The first term is the updated cost function of (9), the second term is the updated tensor solution, and the last term is the updated coefficient corresponding to basis function $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ in (1). Let $\mathbf{c}^l = [\ldots, c_{\boldsymbol{\alpha}}^l, \ldots] \in \mathbb{R}^K$ collect all coefficients in (1), then we define the following quantities for error control:

i) Relative update of the tensor factors:

$$\epsilon_{l,\text{tensor}} = \sqrt{\frac{\sum_{k=1}^d \|\mathbf{U}^{(k),l} - \mathbf{U}^{(k),l-1}\|_F^2}{\sum_{k=1}^d \|\mathbf{U}^{(k),l-1}\|_F^2}}.$$

ii) Relative update of $\mathbf{c} = [\ldots, c_{\boldsymbol{\alpha}}, \ldots]$

$$\epsilon_{l,\text{gPC}} = \|\mathbf{c}^l - \mathbf{c}^{l-1}\| / \|\mathbf{c}^{l-1}\|.$$

iii) Relative update of the cost function:

$$\epsilon_{l,\text{cost}} = |f_l - f_{l-1}| / |f_{l-1}|.$$

The computed factors $\mathbf{U}^{(1),l}, \ldots, \mathbf{U}^{(d),l}$ are regarded as a local minimal, and thus Algorithm 1 terminates if $\epsilon_{l,\text{tensor}}$, $\epsilon_{l,\text{gPC}}$, and $\epsilon_{l,\text{cost}}$ are all small enough.

B. Inner Loop: Subroutine for Solving (13)

Following the procedures in Appendix C, we rewrite Problem (13) as a generalized LASSO problem:

$$\mathbf{vec}(\mathbf{U}^{(k),l+1}) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda|\mathbf{F}\mathbf{x}| \tag{14}$$

where $\mathbf{A} \in \mathbb{R}^{|\Omega|\times nr}$, $\mathbf{F} \in \mathbb{R}^{K\times nr}$, and $\mathbf{b} \in \mathbb{R}^{|\Omega|\times 1}$ and $\mathbf{x} = \mathbf{vec}(\mathbf{X}) \in \mathbb{R}^{nr\times 1}$ is the vectorization of $\mathbf{X}$ [i.e., the $(jn-n+i)$th element of $\mathbf{x}$ is $\mathbf{X}(i, j)$ for any integer $1 \le i \le n$ and $1 \le j \le r$]. Note that $|\Omega|$ is the number of available simulations samples in tensor recovery and $K = (p+d)!/(p!d!)$ is the total number of basis functions in (1).

We solve (14) by the alternating direction method of multipliers (ADMM) [39]. Problem (14) can be rewritten as

$$\min_{\mathbf{x},\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda|\mathbf{z}| \quad \text{s.t. } \mathbf{F}\mathbf{x} - \mathbf{z} = \mathbf{0}.$$

By introducing an auxiliary variable $\mathbf{u}$ and starting with initial guesses $\mathbf{x}^0$, $\mathbf{u}^0 = \mathbf{z}^0 = \mathbf{F}\mathbf{x}^0$, the following iterations are performed to update $\mathbf{x}$ and $\mathbf{z}$:

$$\begin{aligned}
\mathbf{x}^{j+1} &= (\mathbf{A}^T\mathbf{A} + s\mathbf{F}^T\mathbf{F})^{-1}(\mathbf{A}^T\mathbf{b} + s\mathbf{F}^T(\mathbf{z}^j - \mathbf{u}^j)) \\
\mathbf{z}^{j+1} &= \text{shrink}_{\lambda/s}(\mathbf{F}\mathbf{x}^{j+1} + \mathbf{z}^j + \mathbf{u}^j) \\
\mathbf{u}^{j+1} &= \mathbf{u}^j + \mathbf{F}\mathbf{x}^{j+1} - \mathbf{z}^{j+1}.
\end{aligned} \tag{15}$$

Here, $s > 0$ is an augmented Lagrangian parameter, and the soft thresholding operator is defined as

$$\text{shrink}_{\lambda/s}(a) = \begin{cases} a - \lambda/s, & \text{if } a > \lambda/s \\ 0, & \text{if } |a| < \lambda/s \\ a + \lambda/s, & \text{if } a < -\lambda/s. \end{cases}$$

The pseudocodes for solving (13) are given in Algorithm 2.

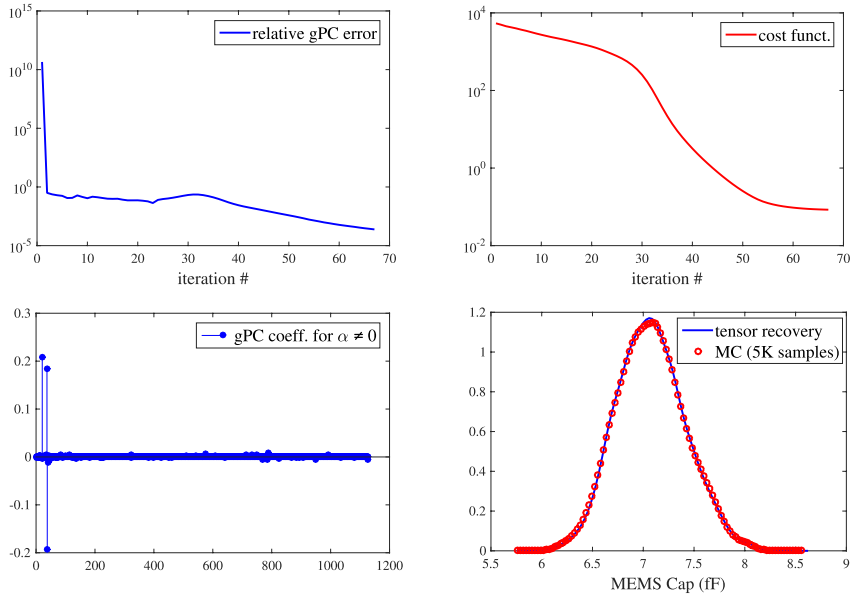


Fig. 3. Numerical results of the MEMS capacitor, with $r = 3$ and $\lambda = 0.01$. Top-left: relative error of the generalized polynomial chaos coefficients in iterations. Top-right: decrease of the cost function in (9). Bottom-left: sparsity of the obtained generalized polynomial chaos expansion. Bottom-right: obtained probability density function compared to that from Monte Carlo.

Algorithm 2 ADMM for Solving (13)

1: Initialize: form $\mathbf{A}, \mathbf{F}$ and $\mathbf{b}$ according to Appendix C, specify initial guess $\mathbf{x}^0$, $\mathbf{u}^0$ and $\mathbf{z}^0$;
2: **for** $j = 0, 1, \ldots$ **do**
3: compute $\mathbf{x}^{j+1}$, $\mathbf{z}^{j+1}$ and $\mathbf{u}^{j+1}$ according to (15);
4: **break** if $\|\mathbf{F}\mathbf{x}^{j+1} - \mathbf{z}^{j+1}\| < \epsilon_1$ & $\|\mathbf{F}^T(\mathbf{z}^{j+1} - \mathbf{z}^j)\| < \epsilon_2$;
5: **end for**
6: **return** $\mathbf{U}^{(k),l+1} = \text{reshape}(\mathbf{x}^{j+1}, [n, r])$.

C. Remarks

The cost function of (9) is nonconvex, and therefore, it is nontrivial to obtain a globally optimal solution with theoretical guarantees. Theoretically speaking, the numerical solution of a nonconvex optimization problem depends on the given initial guess. Although researchers and engineers are very often satisfied with a local minimal, the obtained result may not be good enough for certain cases. In our examples, we find that using random initial guess works well for most cases. However, novel numerical solvers are still highly desired to compute the globally optimal solution of (9) with theoretical guarantees.

V. Generating Stochastic Model (1)

Assuming that the low-rank factors $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(\mathbf{d})}$ of $\boldsymbol{\mathcal{X}}$ have been computed, we are ready to compute the coefficient $c_{\boldsymbol{\alpha}}$ for each basis function in (1). Specifically, replacing $\boldsymbol{\mathcal{Y}}$ with $\boldsymbol{\mathcal{X}}$ in (8) and exploiting the rank-1 property of $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ in (7), we can easily compute $c_{\boldsymbol{\alpha}}$ by

$$c_{\boldsymbol{\alpha}} \approx \langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle = \sum_{j=1}^{r} \left(\prod_{k=1}^{d} \langle \mathbf{u}_k^j, \mathbf{v}_k^{\alpha_k} \rangle \right)$$

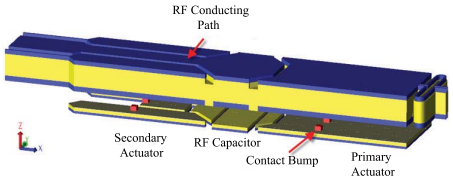


Fig. 4. Schematic of an RF MEMS capacitor [41].

where $\mathbf{v}_k^{\alpha_k}$ is a low-rank factor of $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ in (7). The above expression can be computed by efficient vector inner products.

Once the generalized polynomial chaos expansion (1) is obtained, various statistical information of the performance metric $y(\boldsymbol{\xi})$ can be obtained. For instance, the expectation and standard deviation of $y(\boldsymbol{\xi})$ can be obtained analytically; the density function of $y(\boldsymbol{\xi})$ can be obtained by sampling (1) or using the maximum entropy algorithm [40].

VI. Numerical Results

In order to verify the effectiveness of our tensor recovery uncertainty quantification framework, we show the simulation results of three examples ranging from ICs, MEMSs, and photonic circuits. Since our focus is to solve high-dimensional problems, we simply assume that all process variations are mutually independent to each other, although they are likely to be correlated in practice. All codes are implemented in MATLAB and run on a Macbook with a 2.5-GHz CPU and a 16-GB memory.

A. MEMS Example (With 46 Random Parameters)

We first consider the MEMS device in Fig. 4, which was described in detail in [41]. This example has 46 random parameters describing the material and geometric uncertainties

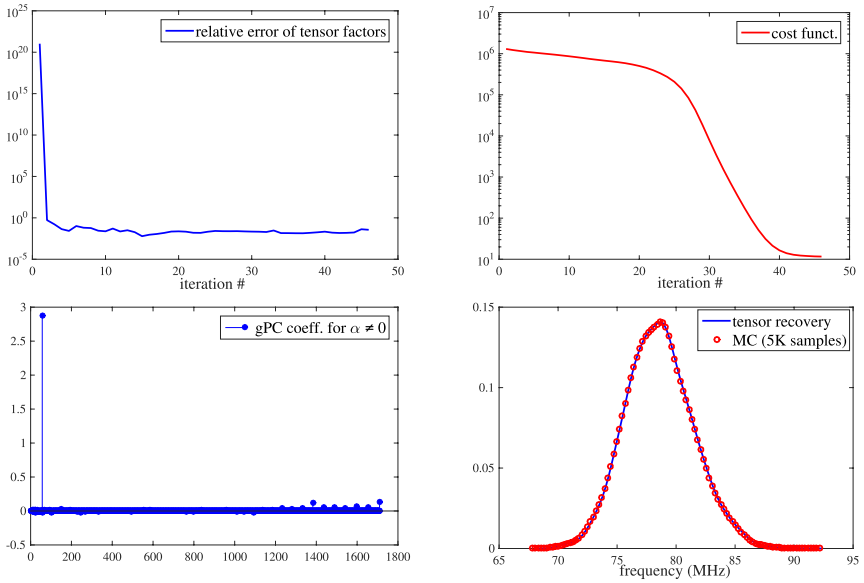


Fig. 5. Numerical results of the ring oscillator, with $r = 3$ and $\lambda = 0.1$. Top-left: relative error of the tensor factors for each iteration. Top-right: decrease in the cost function in (9). Bottom-left: sparsity of the obtained generalized polynomial chaos expansion. Bottom-right: obtained density function compared to that from Monte Carlo using 5000 samples.

TABLE I
COMPARISON OF SIMULATION COST FOR THE MEMS CAPACITOR

method	tensor product	sparse grid	proposed
simulation samples	8.9×10^{21}	4512	300

in CMOS fabrication. The capacitance of this device depends on both bias voltage and process parameters. We assume that a fixed dc voltage is applied to this device, such that we can approximate the capacitance as a second-order generalized polynomial chaos expansion of 46 random parameters. Assume that we use three Gauss quadrature points for each parameter. Consequently, as shown in Table I, a tensor product integration requires $3^{46} \approx 8.9 \times 10^{21}$ simulation samples and the Smolyak sparse grid technique still requires 4512 simulation samples.

We simulate this device using only 300 quadrature samples randomly selected from the tensor product integration rules, and then our tensor recovery method estimates the whole tensor $\mathcal{Y}$ [which contains all 3^{46} samples for the output $y(\boldsymbol{\xi})$]. The relative approximation error for the whole tensor is about 0.1% (measured by cross validation). As shown in Fig. 3, our optimization algorithm converges with less than 70 iterations and the generalized polynomial chaos coefficients are obtained with a small relative error (below 10^{-4}), and the obtained model is very sparse and the obtained density function of the MEMS capacitor is almost identical with that from Monte Carlo. Note that the number of repeated simulations in our algorithm is only about quarter of the total number of basis functions.

B. Multistage CMOS Ring Oscillator (With 57 Parameters)

We continue to consider the CMOS ring oscillator in Fig. 6. This circuit has seven stages of CMOS inverters;

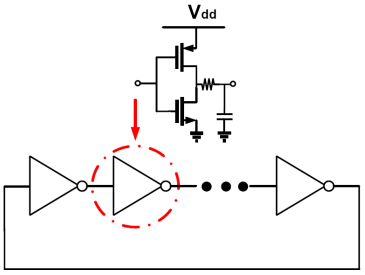


Fig. 6. Schematic of a CMOS ring oscillator.

TABLE II
COMPARISON OF SIMULATION COST FOR THE RING OSCILLATOR

method	tensor product	sparse grid	proposed
simulation samples	1.6×10^{27}	6844	500

57 random parameters are used to describe the variations of threshold voltages, gate-oxide thickness, and effective gate length/width. We intend to obtain a second-order polynomial chaos expansion for its oscillation frequency by calling a periodic steady-state simulator repeatedly. The required number of simulations for different algorithms is listed in Table II, which clearly shows the superior efficiency of our approach for this example.

We simulate this circuit using only 500 samples randomly selected from the $3^{57} \approx 1.6 \times 10^{27}$ tensor product integration samples, and then our algorithm estimates the whole tensor $\mathcal{Y}$ with a 1% relative error. As shown in Fig. 5, our optimization algorithm converges after 46 iterations and the tensor factors are obtained with less than 1% relative errors, and the obtained model is very sparse and the obtained density function of the oscillator frequency is almost identical with that from Monte Carlo. Note that the number of our

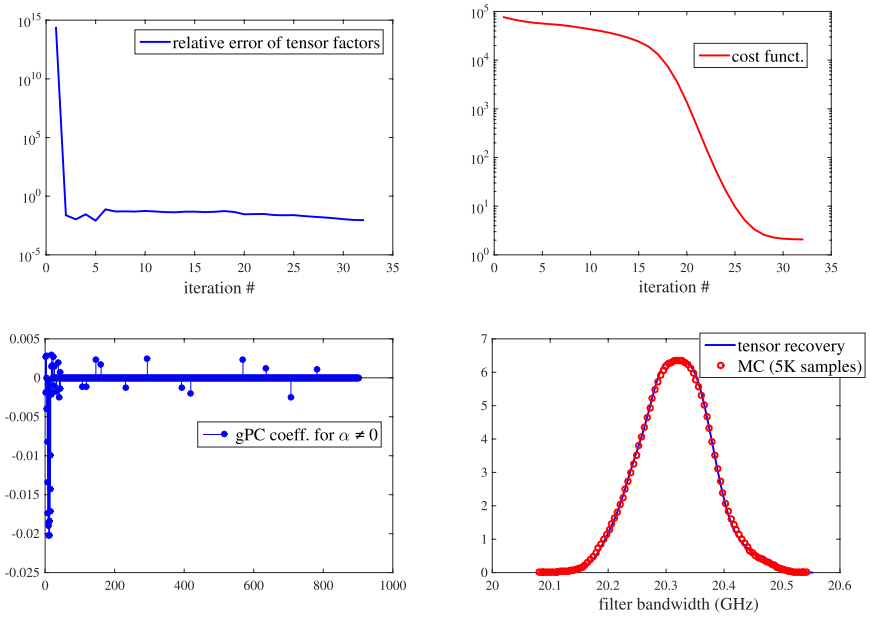


Fig. 7. Numerical results of the photonic bandpass filter, with $r = 3$ and $\lambda = 0.1$. Top-left: relative error of the tensor factors for each iteration. Top-right: decrease in the cost function in (9). Bottom-left: sparsity of the obtained generalized polynomial chaos expansion. Bottom-right: obtained density function of the filter bandwidth compared to that from Monte Carlo using 5000 samples.

simulations (i.e., 500) is much smaller than the total number of basis functions (i.e., 1711) in the generalized polynomial chaos expansion.

C. Photonic Bandpass Filter (With 41 Parameters)

Finally, we consider the photonic bandpass filter in Fig 8. This Chebyshev-type filter has 20 ring resonators and was originally designed to have a 3-dB bandwidth of 20 GHz, a 26-dB minimum return loss, a 400-GHz free spectral range, and a 1.55-μm operation wavelength. A total of 41 random parameters are used to describe the variations of the effective phase index (n_{eff}) of each ring as well as the gap (g) between adjoint rings and between the first/last ring and the bus waveguides. These parameters are assumed to be independent Gaussian variables, with $n_{\text{eff},i} = 2.2315585 + \mathcal{N}(0, 5 \times 10^{-6})$ and $g_i = 0.3 + \mathcal{N}(0, 10^{-3})\mu$m. We intend to obtain a second-order polynomial chaos expansion for the 3-dB bandwidth at the DROP port of this filter. The required number of simulations for different algorithms is listed in Table III. Similar to the results of the previous two examples, our tensor recovery approach is significantly more efficient than the standard tensor product stochastic collocation and the sparse grid implementation.

We simulate this photonic circuit using only 500 samples randomly selected from the $3^{41} \approx 3.6 \times 10^{19}$ tensor product integration samples, and then our algorithm estimates the whole tensor $\mathcal{Y}$ with a 0.1% relative error. As shown in Fig. 7, our optimization algorithm converges after 32 iterations and the tensor factors are obtained with less than 1% relative errors, and the obtained model is also sparse and the obtained density function of the bandwidth is almost identical with that from Monte Carlo. Note that the number of our simulations (i.e., 500) is much smaller than the total number of basis functions (i.e., 903) in the generalized polynomial chaos expansion.

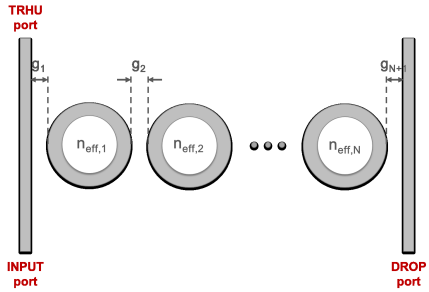


Fig. 8. Schematic of a photonic bandpass filter, with $N = 20$.

TABLE III
COMPARISON OF SIMULATION COST FOR THE PHOTONIC CIRCUIT

method	tensor product	sparse grid	proposed
simulation samples	3.6×10^{19}	3445	500

VII. CONCLUSION

This paper has presented a big data approach for solving the challenging high-dimensional uncertainty quantification problem. Our key idea is to estimate the high-dimensional simulation data array from an extremely small subset of its samples. This idea has been described as a tensor recovery model with low-rank and sparse constraints. Detailed numerical methods have been described to solve the resulting optimization problem. The simulation results on a CMOS ring oscillator, a MEMS RF capacitor, and an integrated photonic circuit show that our algorithm can be easily applied to problems with about 40 to 60 random parameters. Instead of using a huge number of (e.g., about 10^{27}) quadrature samples, our algorithm requires only several hundreds, which is even much smaller than the number of basis functions. The proposed algorithm is much more efficient than sparse grid and Monte Carlo for

our tested cases, whereas Monte Carlo used to be the only feasible approach to handle the underlying high-dimensional numerical integration.

There exist some open theoretical questions that are worth further investigations.

1) First, it is desirable to develop a rigorous framework such that the tensor rank r and the regularization parameter λ can be determined in an optimal manner.
2) Second, the resulting tensor recovery model is nonconvex. A framework that can obtain its global optimal or relax the model to a convex one will be valuable.
3) Third, it is worth improving our method such that it can efficiently handle a vector output $\mathbf{y}(\boldsymbol{\xi})$.
4) Finally, our framework generates the subset Ω in a random way. How to generate Ω optimally is still unclear.

It is also possible to extend our framework to other engineering applications, such as power systems and robotics.

APPENDIX A
CONSTRUCTING ORTHONORMAL POLYNOMIALS [31]

Consider a single random parameter $\xi_k \in \mathbb{R}$ with a probability density function $\rho_k(\xi_k)$, one can construct a set of polynomial functions subject to the orthonormal condition

$$\int_{\mathbb{R}} \phi_{k,\alpha}(\xi_k)\phi_{k,\beta}(\xi_k)\rho_k(\xi_k)d\xi_k = \delta_{\alpha,\beta}$$

where $\delta_{\alpha,\beta}$ is a Delta function and integer α is the highest degree of $\phi_{k,\alpha}(\xi_k)$. Such polynomials can be constructed as follows [31]. First, one constructs orthogonal polynomials $\{\pi_{k,\alpha}(\xi_k)\}_{\alpha=0}^{p}$ with an leading coefficient 1 recursively

$$\pi_{k,\alpha+1}(\xi_k) = (\xi_k - \gamma_\alpha)\pi_{k,\alpha}(\xi_k) - \kappa_\alpha \pi_{k,\alpha-1}(\xi_k)$$

for $\alpha = 0, 1, \ldots p-1$, with initial conditions $\pi_{k,-1}(\xi_k) = 0$, $\pi_{k,0}(\xi_k) = 1$, and $\kappa_0 = 1$. For $\alpha \geq 0$, the recurrence parameters are defined as

$$\gamma_\alpha = \frac{\mathbb{E}\big(\xi_k \pi_{k,\alpha}^2(\xi_k)\big)}{\mathbb{E}\big(\pi_{k,\alpha}^2(\xi_k)\big)}, \quad \kappa_{\alpha+1} = \frac{\mathbb{E}\big(\xi_k \pi_{k,\alpha+1}^2(\xi_k)\big)}{\mathbb{E}\big(\xi_k \pi_{k,\alpha}^2(\xi_k)\big)}. \quad (16)$$

Here, $\mathbb{E}$ denotes the operator that calculates expectation. Second, one can obtain $\{\phi_{k,\alpha}(\xi_k)\}_{\alpha=0}^{p}$ by normalization

$$\phi_{k,\alpha}(\xi_k) = \frac{\pi_{k,\alpha}(\xi_k)}{\sqrt{\kappa_0\kappa_1\ldots\kappa_\alpha}}, \quad \text{for } \alpha = 0, 1, \ldots, p.$$

APPENDIX B
GAUSS QUADRATURE RULE [34]

Given $\xi_k \in \mathbb{R}$ with a density function $\rho_k(\xi_k)$ and a smooth function $q(\xi_k)$, the Gauss quadrature evaluates the integral

$$\int_{\mathbb{R}} q(\xi_k)\rho_k(\xi_k)d\xi_k \approx \sum_{i_k=1}^{n} q\big(\xi_k^{i_k}\big) w_k^{i_k}$$

with an error decreasing exponentially as n increases. An exact result is obtained if $q(\xi_k)$ is a polynomial function of degree $\leq 2n-1$. One can obtain $\{(\xi_k^{i_k}, w_k^{i_k})\}_{i_k=1}^{n}$ by reusing the recurrence parameters in (16) to form a symmetric tridiagonal matrix $\mathbf{J} \in \mathbb{R}^{n\times n}$

$$\mathbf{J}(i,j) = \begin{cases} \gamma_{i-1}, & \text{if } i = j \\ \sqrt{\kappa_i}, & \text{if } i = j+1 \\ \sqrt{\kappa_j}, & \text{if } i = j-1 \\ 0, & \text{otherwise} \end{cases} \quad \text{for } 1 \leq i,\ j \leq n.$$

Let $\mathbf{J} = \mathbf{Q}\Sigma\mathbf{Q}^T$ be an eigenvalue decomposition and $\mathbf{Q}$ a unitary matrix, then $\xi_k^{i_k} = \Sigma(i_k, i_k)$ and $w_k^{i_k} = (\mathbf{Q}(1, i_k))^2$.

APPENDIX C
ASSEMBLING THE MATRICES AND VECTOR IN (14)

Consider the tensor factors $\mathbf{U}^{(1),l+1}$, …, $\mathbf{U}^{(k-1),l+1}$, $\mathbf{X}$, and $\mathbf{U}^{(k+1),l}$, …, $\mathbf{U}^{(d),l}$ in (13). We denote the (i,j) element of $\mathbf{U}^{(k'),l}$ (or $\mathbf{X}$) by scalar $u_{i,j}^{(k'),l}$ (or $x_{i,j}$), and its jth column by vector $\mathbf{u}_j^{(k'),l}$ (or $\underline{\mathbf{x}}_j$) $\in \mathbb{R}^{n\times 1}$. Then, the cost function in (13) is

$$f(\ldots, \mathbf{U}^{(k-1),l+1}, \mathbf{X}, \mathbf{U}^{(k+1),l}, \ldots) = \frac{1}{2}\sum_{\mathbf{i}\in\Omega}\left(\sum_{j=1}^{r} x_{i_k,j}\mu_{\mathbf{i},j} - \mathcal{Y}(\mathbf{i})\right)^2 + \lambda \sum_{|\boldsymbol{\alpha}|\leq p}\left|\sum_{j=1}^{r} \nu_{\boldsymbol{\alpha},j}\big\langle \underline{\mathbf{x}}_j, \mathbf{w}_{\alpha_k}^{(k)}\big\rangle\right|$$

where the scalars $\mu_{\mathbf{i},j}$ and $\nu_{\boldsymbol{\alpha},j}$ are computed as follows:

$$\mu_{\mathbf{i},j} = \prod_{k'=1}^{k-1} u_{i_{k'},j}^{(k'),l+1} \prod_{k'=k+1}^{d} u_{i_{k'},j}^{(k'),l}$$

$$\nu_{\boldsymbol{\alpha},j} = \prod_{k'=1}^{k-1} \big\langle \mathbf{u}_j^{(k'),l+1}, \mathbf{w}_{\alpha_{k'}}^{(k')}\big\rangle \prod_{k'=k+1}^{d} \big\langle \mathbf{u}_j^{(k'),l}, \mathbf{w}_{\alpha_{k'}}^{(k')}\big\rangle.$$

Since each row (or element) of $\mathbf{A}$ (or $\mathbf{b}$) corresponds to an index $\mathbf{i} = (i_1, \ldots, i_d) \in \Omega$ and each row of $\mathbf{F}$ corresponds to a basis function $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$, in this Appendix, we use $\mathbf{i}$ as the row index (or element index) of $\mathbf{A}$ (or $\mathbf{b}$) and $\boldsymbol{\alpha}$ as the row index of $\mathbf{F}$.[3] Now we specify the elements of $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{F}$ of (14).

1) For every $\mathbf{i} \in \Omega$, $\mathbf{b}(\mathbf{i}) = \mathcal{Y}(\mathbf{i})$.
2) Since $x_{i_k,j}$ is the $((j-1)n + i_k)$th element of $\mathbf{x} = \mathbf{vec}(\mathbf{X}) \in \mathbb{R}^{nr\times 1}$, for every $\mathbf{i} \in \Omega$, we have
$$\mathbf{A}(\mathbf{i}, (j-1)n + i_k) = \begin{cases} \mu_{\mathbf{i},j}, & \text{for } j = 1, \ldots, r \\ 0, & \text{otherwise.} \end{cases}$$
3) Since $\underline{\mathbf{x}}_j$ includes the elements of $\mathbf{x} \in \mathbb{R}^{nr\times 1}$ ranging from index $(j-1)n+1$ to jn, given an index vector $\boldsymbol{\alpha}$, the corresponding row of $\mathbf{F}$ can be specified as
$$\mathbf{F}(\boldsymbol{\alpha}, jn - n + i_k) = \nu_{\boldsymbol{\alpha},j}\mathbf{v}_k^{\alpha_k}(i_k) = \nu_{\boldsymbol{\alpha},j}\phi_{k,\alpha_k}\big(\xi_k^{i_k}\big) w_k^{i_k}$$
for all integers $j \in [1, r]$ and $i_k \in [1, n]$.

REFERENCES

[1] Z. Zhang, T.-W. Weng, and L. Daniel, "A big-data approach to handle process variations: Uncertainty quantification by tensor recovery," in *Proc. IEEE Workshop Signal Power Integr.*, May 2016, pp. 1–4.
[2] D. S. Boning, "Variation," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 63–71, Feb. 2008.
[3] S. Weinzierl, "Introduction to Monte Carlo methods," NIKHEF, Amsterdam, The Netherlands, Tech. Rep. NIKHEF-00-012, 2000.

[3] We can order all elements of Ω in a specific way. If $\mathbf{i}$ is the kth element in Ω, then $\mathbf{A}(\mathbf{i}, j)$ and $\mathbf{b}(\mathbf{i})$ denote $\mathbf{A}(k, j)$ and $\mathbf{b}(k)$, respectively.

[4] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.

[5] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*. New York, NY, USA: Springer-Verlag, 1991.

[6] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM J. Sci. Comput.*, vol. 27, no. 3, pp. 1118–1139, Mar. 2005.

[7] P. Manfredi, D. Vande Ginste, D. De Zutter, and F. G. Canavero, "Stochastic modeling of nonlinear circuits via SPICE-compatible spectral equivalents," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 7, pp. 2057–2065, Jul. 2014.

[8] I. S. Stievano, P. Manfredi, and F. G. Canavero, "Parameters variability effects on multiconductor interconnects via Hermite polynomial chaos," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 1, no. 8, pp. 1234–1239, Aug. 2011.

[9] Z. Zhang, T. A. El-Moselhy, I. A. M. Elfadel, and L. Daniel, "Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 32, no. 10, pp. 1533–1545, Oct. 2013.

[10] K. Strunz and Q. Su, "Stochastic formulation of SPICE-type electronic circuit simulation with polynomial chaos," *ACM Trans. Model. Comput. Simul.*, vol. 18, no. 4, pp. 15:1–15:23, Sep. 2008.

[11] Z. Zhang, T. A. El-Moselhy, P. Maffezzoni, I. A. M. Elfadel, and L. Daniel, "Efficient uncertainty quantification for the periodic steady state of forced and autonomous circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 10, pp. 687–691, Oct. 2013.

[12] R. Pulch, "Modelling and simulation of autonomous oscillators with random parameters," *Math. Comput. Simul.*, vol. 81, no. 6, pp. 1128–1143, Feb. 2011.

[13] M. R. Rufuie, E. Gad, M. Nakhla, R. Achar, and M. Farhan, "Fast variability analysis of general nonlinear circuits using decoupled polynomial chaos," in *Proc. 18th IEEE Workshop Signal Power Integr. (SPI)*, May 2014, pp. 1–4.

[14] A. Yücel, H. Bağci, and E. Michielssen, "An ME-PC enhanced HDMR method for efficient statistical analysis of multiconductor transmission line networks," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 5, no. 5, pp. 685–696, May 2015.

[15] M. Ahadi and S. Roy, "Sparse linear regression (SPLINER) approach for efficient multidimensional uncertainty quantification of high-speed circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 35, no. 10, pp. 1640–1652, Oct. 2015.

[16] M. Ahadi, A. K. Prasad, and S. Roy, "Hyperbolic polynomial chaos expansion (HPCE) and its application to statistical analysis of nonlinear circuits," in *Proc. IEEE Workshop Signal Power Integr.*, May 2016, pp. 1–4.

[17] P. Manfredi, D. V. Ginste, D. De Zutter, and F. G. Canavero, "Generalized decoupled polynomial chaos for nonlinear circuits with many random parameters," *IEEE Microw. Wireless Compon. Lett.*, vol. 25, no. 8, pp. 505–507, Aug. 2015.

[18] Z. Zhang *et al.*, "Stochastic testing simulator for integrated circuits and MEMS: Hierarchical and sparse techniques," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Sep. 2014, pp. 1–8.

[19] Z. Zhang, X. Yang, I. V. Oseledets, G. E. Karniadakis, and L. Daniel, "Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 1, pp. 63–76, Jan. 2015.

[20] T.-W. Weng, Z. Zhang, Z. Su, Y. Marzouk, A. Melloni, and L. Daniel, "Uncertainty quantification of silicon photonic devices with correlated and non-Gaussian random parameters," *Opt. Express*, vol. 23, no. 4, pp. 4242–4254, Feb. 2015.

[21] Z. Zubac, J. Fostier, D. De Zutter, and D. V. Ginste, "Efficient uncertainty quantification of large two-dimensional optical systems with a parallelized stochastic Galerkin method," *Opt. Express*, vol. 23, no. 24, pp. 30833–30850, 2015.

[22] D. Xiu and G. E. Karniadakis, "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM J. Sci. Comput.*, vol. 24, no. 2, pp. 619–644, Feb. 2002.

[23] X. Yang, M. Choi, G. Lin, and G. E. Karniadakis, "Adaptive ANOVA decomposition of stochastic incompressible and compressible flows," *J. Comput. Phys.*, vol. 231, no. 4, pp. 1587–1614, Feb. 2012.

[24] X. Ma and N. Zabaras, "An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations," *J. Comput. Phys.*, vol. 229, no. 10, pp. 3884–3915, May 2010.

[25] H. Rabitz and O. Aliş, "General foundations of high-dimensional model representations," *J. Math. Chem.*, vol. 25, nos. 2–3, pp. 197–233, 1999.

[26] X. Li, "Finding deterministic solution from underdetermined equation: Large-scale performance variability modeling of analog/RF circuits," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1661–1668, Nov. 2011.

[27] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[28] T. El Moselhy and L. Daniel, "Stochastic dominant singular vectors method for variation-aware extraction," in *Proc. 47th Design Autom. Conf.*, Jun. 2010, pp. 667–672.

[29] T. El-Moselhy and L. Daniel, "Variation-aware interconnect extraction using statistical moment preserving model order reduction," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, Mar. 2010, pp. 453–458.

[30] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.

[31] W. Gautschi, "On generating orthogonal polynomials," *SIAM J. Sci. Statist. Comput.*, vol. 3, no. 3, pp. 289–317, Sep. 1982.

[32] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numer. Algorithms*, vol. 18, pp. 209–232, Mar. 1998.

[33] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*. North Chelmsford, MA, USA: Courier Corporation, 2007.

[34] G. H. Golub and J. H. Welsch, "Calculation of gauss quadrature rules," *Math. Comput.*, vol. 23, pp. 221–230, 1969.

[35] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of 'Eckart-Young' decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[36] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

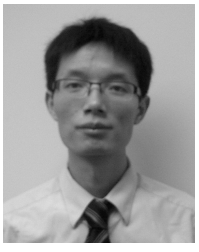
[37] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[38] Z. Zhang, L. Daniel, K. Batselier, H. Liu, and N. Wong, "Tensor computation: A new framework for high-dimensional problems in EDA," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, to be published, doi: 10.1109/TCAD.2016.2618879.

[39] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[40] Z. Zhang, N. Farnoosh, T. Klemas, and L. Daniel, "Maximum-entropy density estimation for MRI stochastic surrogate models," *IEEE Antennas Wireless Propag. Lett.*, vol. 13, pp. 1656–1659, 2014.

[41] Z. Zhang, M. Kamon, and L. Daniel, "Continuation-based pull-in and lift-off simulation algorithms for microelectromechanical devices," *J. Microelectromech. Syst.*, vol. 23, no. 5, pp. 1084–1093, Oct. 2014.



Zheng Zhang (M'15) received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2015.

His industrial experiences include Coventor Inc. Cambridge, and Maxim-IC, Colorado Springs, CO, USA, academic visiting experiences include the University of California at San Diego, La Jolla, CA, USA, Brown University, Providence, RI, USA, and Politechnico di Milano, Milan, Italy; government laboratory experiences include the Argonne National Laboratory, Lemont, IL, USA. Currently, he is a Post-Doctoral Associate with the Research Laboratory of Electronics, MIT. His current research interests include uncertainty quantification, tensor and model order reduction, with application to nanoelectronics, energy, and biomedical problems.

Dr. Zhang was a recipient of the 2016 ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation, the 2015 Doctoral Dissertation Seminar Award (i.e., Best Thesis Award) from the Microsystems Technology Laboratory of MIT, the 2014 Best Paper Award from the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the 2014 Chinese Government Award for Outstanding Students Abroad, and the 2011 Li Ka-Shing Prize from the University of Hong Kong.

Tsui-Wei Weng (S'12) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2011 and 2013, respectively. She is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Her current research interests include mixed integer programming and nonconvex optimization problems in machine learning, as well as uncertainty quantification in emerging technology such as artificial intelligence and nanophotonics.

Luca Daniel (S'98–M'03) received the Ph.D. degree in electrical engineering from the University of California at Berkeley (UC Berkeley), Berkeley, CA, USA, in 2003.

His industry experiences include the HP Research Laboratory, Palo Alto, Santa Clara, CA, USA, in 1998, and the Cadence Berkeley Laboratory, Berkeley, in 2001. He is currently a Full Professor with the Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA, USA. His current research interests include integral equation solvers, uncertainty quantification and parameterized model order reduction, applied to RF circuits, silicon photonics, MEMSs, magnetic resonance imaging scanners, and the human cardiovascular system.

Prof. Daniel was a recipient of the 1999 IEEE TRANSACTIONS ON POWER ELECTRONICS Best Paper Award, the 2003 Best Ph.D. Thesis Awards from the Electrical Engineering and the Applied Math departments at UC Berkeley, the 2003 ACM Outstanding Ph.D. Dissertation Award in Electronic Design Automation, the 2009 IBM Corporation Faculty Award, the 2010 IEEE Early Career Award in Electronic Design Automation, the 2014 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award, and seven best paper awards in conferences.