High-Dimensional Uncertainty Quantification via Rank- and Sample-Adaptive Tensor Regression

Zichang He and Zheng Zhang, Member, IEEE

(Invited Paper)

Abstract—Fabrication process variations can significantly influence the performance and yield of nano-scale electronic and photonic circuits. Stochastic spectral methods have achieved great success in quantifying the impact of process variations, but they suffer from the curse of dimensionality. Recently, low-rank tensor methods have been developed to mitigate this issue, but two fundamental challenges remain open: how to automatically determine the tensor rank and how to adaptively pick the informative simulation samples. This paper proposes a novel tensor regression method to address these two challenges. We use a \( \ell_q/\ell_2 \) group-sparsity regularization to determine the tensor rank. The resulting optimization problem can be efficiently solved via an alternating minimization solver. We also propose a two-stage adaptive sampling method to reduce the simulation cost. Our method considers both exploration and exploitation via the estimated Voronoi cell volume and nonlinearity measurement respectively. The proposed model is verified with synthetic and some realistic circuit benchmarks, on which our method can well capture the uncertainty caused by 19 to 100 random variables with only 100 to 600 simulation samples.

Index Terms—Tensor regression, high dimensionality, uncertainty quantification, polynomial chaos, process variation, rank determination, adaptive sampling.

I. INTRODUCTION

Fabrication process variations (e.g., surface roughness of interconnects and photonic waveguide, and random doping effects of transistors) have been a major concern in nano-scale chip design. They can significantly influence chip performance and decrease product yield [2]. Monte Carlo (MC) is one of the most popular methods to quantify the chip performance under uncertainty, but it requires a huge amount of computational cost [3]. Instead, stochastic spectral methods based on generalized polynomial chaos (gPC) [4] offer efficient solutions for fast uncertainty quantification by approximating a real uncertain circuit variable as a linear combination of some stochastic basis functions [5–7]. These techniques have been increasingly used in design automation [8–15]. A main challenge of stochastic spectral method is the curse of dimensionality: the computational cost grows very fast as the number of random parameters increases. In order to address this fundamental challenge, many high-dimensional solvers have been developed. The representative techniques include (but are not limited to) compressive sensing [16, 17], hyperbolic regression [18], analysis of variance (ANOVA) [19, 20], model order reduction [21], and hierarchical modeling [22, 23], and tensor methods [23, 24].

Low-rank tensor approximation has shown promising performance in solving high-dimensional uncertainty quantification problems [24–29]. By low-rank tensor decomposition, one may reduce the number of unknown variables in uncertainty quantification to a linear function of the parameter dimensionality. However, there is a fundamental question: how can we determine the tensor rank and the associated model complexity? Because it is hard to exactly determine a tensor rank a-priori [30], existing methods often use a tensor rank pre-specified by the user or use a greedy method to update the tensor rank until convergence [24, 31, 32]. These methods often offers inaccurate rank estimation and are complicated in computation. Besides rank determination, another important question is: how can we adaptively add a few simulation samples to update the model with a low computation budget? This is very important in electronic and photonic design automation, because obtaining each piece of data sample requires time-consuming device-level or circuit-level numerical simulations.

Paper contributions. We propose a novel tensor regression method for high-dimensional uncertainty quantification. Tensor regression has been studied in machine learning and image data analysis [33–35]. Despite some existing work of automatic rank determination [36, 37] and adaptive sampling [38, 39] for tensor decomposition and completion, there is no work about tensor regression, its automatic rank determination and adaptive sampling for uncertainty quantification. The main contributions of this paper include:

- We formulate high-dimensional uncertainty quantification as a tensor regression problem. We further propose a \( \ell_q/\ell_2 \) group-sparsity regularization method to determine rank automatically. Based on variation equality, the tensor-structured regression problem can be efficiently solved via a block coordinate descent algorithm with an analytical solution in each subproblem.
- We propose a two-stage adaptive sampling method to reduce the simulation cost. This method balances the exploration and exploitation via combining the estimation of Voronoi cell volumes and the nonlinearity of an output function.
- We verify the proposed uncertainty quantification model on a 100-dim synthetic function, a 19-dim photonic band-pass filter, and a 57-dim CMOS ring oscillator. Our model can well capture the high-dimensional stochastic output with only 100-600 samples.

Compared with our conference paper [1], this manuscript
presents the following additional results:
• The detailed implementations of the proposed method, including both the compact tensor regression solver and the adaptive sampling procedure (Section III and IV)
• The post-processing step of extracting statistical information from the obtained tensor regression model (Section V).
• The enriched experiments (Section VI), including a demonstrative synthetic example and detailed comparisons with other methods.

II. NOTATION AND PRELIMINARIES
Throughout this paper, a scalar is represented by a lowercase letter, e.g., \( x \in \mathbb{R} \); a vector or matrix is represented by a boldface lowercase or capital letter respectively, e.g., \( x \in \mathbb{R}^n \) and \( X \in \mathbb{R}^{m \times n} \). A tensor, which describes a multidimensional data array, is represented by a bold calligraphic letter, e.g., \( \mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d} \). The \((i_1, i_2, \ldots, i_d)\)-th data element of a tensor \( \mathcal{X} \) is denoted as \( x_{i_1,i_2,\ldots,i_d} \). Obviously \( \mathcal{X} \) reduces to a matrix \( X \) when \( d = 2 \), and its data element is \( x_{i_1,i_2} \). In this section, we will briefly introduce the background of generalized polynomial chaos (gPC) and tensor computation.

A. Generalized Polynomial Chaos Expansion
Let \( \xi = [\xi_1, \ldots, \xi_d] \in \mathbb{R}^d \) be a random vector describing fabrication process variations with mutually independent components. We aim to estimate the interested performance metric \( y(\xi) \) (e.g., chip frequency or power) under such uncertainty. We assume that \( y(\xi) \) has a finite variance under the process variations. A truncated gPC expansion approximates \( y(\xi) \) as the summation of a series of orthonormal basis functions [4]:

\[
y(\xi) \approx \hat{y}(\xi) = \sum_{\alpha \in \Theta} c_{\alpha} \Psi_{\alpha}(\xi),
\]

where \( \alpha \in \mathbb{N}^d \) is an index vector in the index set \( \Theta \), \( c_{\alpha} \) is the coefficient, and \( \Psi_{\alpha} \) is a polynomial basis function of degree \( |\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_d \). One of the most commonly used index set is the total degree one, which selects multivariate polynomials up to a total degree \( p \), i.e.,

\[
\Theta = \{ \alpha | \alpha_k \in \mathbb{N}, 0 \leq \alpha_k \leq p \},
\]

leading to a total of \( \frac{(d+p)!}{dp!} \) terms of expansion. Let \( \phi_{\alpha_k}^{(k)}(\xi_k) \) denote the order-\( \alpha_k \) univariate basis of the \( k \)-th random parameter \( \xi_k \), the multivariate basis is constructed via taking the product of univariate orthonormal polynomial basis:

\[
\Psi_{\alpha}(\xi) = \prod_{k=1}^{d} \phi_{\alpha_k}^{(k)}(\xi_k).
\]

Therefore, given the joint probability density function \( \rho(\xi) \), the multivariate basis satisfies the orthonormal condition:

\[
\langle \Psi_{\alpha}(\xi), \Psi_{\beta}(\xi) \rangle = \int_{\mathbb{R}^d} \Psi_{\alpha}(\xi) \Psi_{\beta}(\xi) \rho(\xi) \, d\xi = \delta_{\alpha,\beta}.
\]

We show the detailed construction of univariate basis functions in Appendix A.

In order to estimate the unknown coefficients \( c_{\alpha} \)'s, several popular methods can be used, including intrusive (i.e., non-sampling) methods (e.g., stochastic Galerkin [40] and stochastic testing [6]) and non-intrusive (i.e., sampling) methods (e.g., stochastic collocation based on pseudo-projection or regression [41]). It is well known that gPC expansion suffers the curse of dimensionality. The computational cost grows exponentially as the dimension of \( \xi \) increases.

B. Tensor and Tensor Decomposition
Given two tensors \( \mathcal{X} \) and \( \mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \cdots \times n_d} \), their inner product is defined as:

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

A tensor \( \mathcal{X} \) can be unfolded into a matrix along the \( k \)-th mode/dimension, denoted as \( \text{Unfold}_k(\mathcal{X}) := X(k) \in \mathbb{R}^{n_k \times \prod_{i \neq k} n_i} \). Conversely, folding the \( k \)-mode matricization back to the original tensor is denoted as \( \text{Fold}_k(X(k)) := \mathcal{X} \).

Given a \( d \)-dim tensor, it can be factorized as a summation some rank-1 vectors, which is called CANDECOMP/PARAFAC (CP) decomposition [42]:

\[
\mathcal{X} = \sum_{r=1}^{R} a_r^{(1)} \circ a_r^{(2)} \cdots \circ a_r^{(d)} = [A^{(1)}, A^{(2)}, \ldots, A^{(d)}],
\]

where \( \circ \) denotes the outer product. The last term is the Krusal form, where factor matrix \( A^{(k)} = [a^{(k)}_1, \ldots, a^{(k)}_R] \in \mathbb{R}^{n_k \times R} \) includes all vectors associated with the \( k \)-th dimension. The smallest number of \( R \) that ensures the above equality is called a CP rank. The \( k \)-th mode unfolding matrix \( X(k) \) can be written with CP factors as

\[
X(k) = A^{(k)} \Psi_{(k)}^T
\]

with

\[
A^{(k)} = A^{(d)} \circ \cdots \circ A^{(k-1)} \circ A^{(k+1)} \cdots \circ A^{(1)},
\]

where \( \circ \) denotes the Khatri-Rao product, which performs column-wise Kronecker products [42]. More details of tensor operations can be found in [42].

III. PROPOSED TENSOR REGRESSION METHOD

A. Low-Rank Tensor Regression Formulation
To approximate \( y(\xi) \) as a tensor regression model, we choose a full tensor-product index set for the gPC expansion:

\[
\Theta = \{ \alpha = [\alpha_1, \alpha_2, \cdots, \alpha_d] \mid 0 \leq \alpha_k \leq p, \forall k \in [1, d] \}.
\]

This specifies a gPC expansion with \( (p+1)^d \) basis functions. Let \( i_k = \alpha_k + 1 \), then we can define two \( d \)-dimensional tensors \( \mathcal{X} \) and \( B(\xi) \) with their \((i_1, i_2, \ldots, i_d)\)-th elements as

\[
x_{i_1i_2\cdots i_d} = c_{\alpha} \text{ and } b_{i_1i_2\cdots i_d}(\xi) = \Psi_{\alpha}(\xi).
\]

Combining Eqs (1), (8) and (9), the truncated gPC expansion can be written as a tensor inner product

\[
y(\xi) \approx \hat{y}(\xi) = \langle \mathcal{X}, B(\xi) \rangle.
\]
The tensor $\mathcal{B}(\xi) \in \mathbb{R}^{(p+1)\times\cdots\times(p+1)}$ is a rank-1 tensor that can be exactly represented as:

$$\mathcal{B}(\xi) = \phi^{(1)}(\xi_1) \circ \phi^{(2)}(\xi_2) \circ \cdots \circ \phi^{(d)}(\xi_d),$$  \hspace{1cm} (11)

where $\phi^{(k)}(\xi_k) = [\phi^{(k)}_1(\xi_k), \ldots, \phi^{(k)}_n(\xi_k)]^T \in \mathbb{R}^{p+1}$ collects all univariate basis functions of random parameter $\xi_k$ up to order-$p$.

The unknown coefficient tensor $\mathcal{X}$ has $(p + 1)^d$ variables in total, but we can describe it via a rank-$R$ CP approximation:

$$\mathcal{X} \approx \sum_{r=1}^{R} u_r^{(1)} \circ u_r^{(2)} \circ \cdots \circ u_r^{(d)} = [U^{(1)}, U^{(2)}, \ldots, U^{(d)}].$$  \hspace{1cm} (12)

It decreases the number of unknown variables to $(p + 1)dR$, which only linearly depends on $d$ and thus effectively overcomes the curse of dimensionality.

Our goal is to compute coefficient tensor $\mathcal{X}$ given a set of data samples $\{\xi_n, y(\xi_n)\}_{n=1}^N$ via solving the following optimization problem

$$\min_{\{U^{(k)}\}_{k=1}^R} h(\mathcal{X}) = \frac{1}{2} \sum_{n=1}^{N} \Big( y_n - \langle [U^{(1)}, U^{(2)}, \ldots, U^{(d)}], B^n \rangle \Big)^2,$$  \hspace{1cm} (13)

where $y_n = y(\xi_n)$, $B^n = \mathcal{B}(\xi_n)$, and $\xi_n$ denotes the $n$-th sample.

\section{Automatic Rank Determination}

The low-rank approximation (12) assumes that $\mathcal{X}$ can be well approximated by $R$ rank-1 terms. In practice, it is hard to determine $R$ in advance. In this work, we leverage a group-sparsity regularization function to shrink the tensor rank from an initial estimation. Specifically, define the following vector:

$$\nu := [v_1, v_2, \ldots, v_R] \text{ with } v_r = \left( \sum_{k=1}^{d} \|u_r^{(k)}\|_2^2 \right)^{\frac{1}{2}} \forall r \in [1, R].$$  \hspace{1cm} (14)

We further use its $\ell_q$ norm with $q \in (0, 1]$ to measure the sparsity of $\nu$:

$$g(\mathcal{X}) = \|\nu\|_q, \quad q \in (0, 1].$$  \hspace{1cm} (15)

This function groups all factors of a rank-1 term together and enforces the sparsity among $R$ groups. The rank is reduced when the $r$-th columns of all factor matrices are enforced to zero. When $q = 1$, this method degenerates to a group lasso, and a smaller $q$ leads to a stronger shrinkage force.

Based on this rank-shrinkage function, we compute the tensor-structured gPC coefficients by solving a regularized tensor regression problem:

$$\min_{\{U^{(k)}\}_{k=1}^R} f(\mathcal{X}) = h(\mathcal{X}) + \lambda g(\mathcal{X}),$$  \hspace{1cm} (16)

where $\lambda > 0$ is a regularization parameter. As shown in Fig. 1, after solving this optimization problem, some columns with the same column indices among all matrices $U^{(k)}$'s are close to zero. These columns can be deleted and the actual rank of our obtained tensor becomes $R \leq \hat{R}$, where $\hat{R}$ is the number of remaining columns in each factor matrix.

\section{A More Tractable Regularization}

It is non-trivial to minimize $f(\mathcal{X})$ since $g(\mathcal{X})$ is non-differentiable and non-convex with respect to $U^{(k)}$'s. Therefore, we replace the regularization function with a more tractable one based on the following variational equality.

\textbf{Lemma 1} (Variational equality \cite{43}). Let $\alpha \in (0, 2)$, and $\beta = \frac{\alpha}{2 - \alpha}$. For any vector $y \in \mathbb{R}^p$, we have the following equality

$$\|y\|_\alpha = \min_{z \in \mathbb{R}^p} \frac{1}{2} \sum_{j=1}^{p} \frac{y_j^2}{z_j} + \frac{1}{2} \|z\|_\beta,$$  \hspace{1cm} (17)

where the minimum is uniquely attained for $z_j = |y_j|^{\frac{2}{\alpha}} |\alpha|^{\alpha - 1}, j = 1, 2, \ldots, p$.

\textbf{Proof.} See Appendix B. \hfill $\square$

If we take $p = R$, $\alpha = q$, $z = \eta$, and $y_j = \sqrt{\frac{d}{\sum_{k=1}^{d} \|u_r^{(k)}\|_2^2}}$ on the right-hand side of Eq. (17), then we have

$$\hat{g}(\mathcal{X}, \eta) = \frac{1}{2} \sum_{r=1}^{R} \frac{\|u_r^{(k)}\|_2^2}{\eta_r} + \frac{1}{2} \|\eta\|_{\frac{2}{2-\frac{2}{\alpha}}}.$$  \hspace{1cm} (18)

The original rank-shrinking function is

$$g(\mathcal{X}) = \min_{\eta \in \mathbb{R}^R} \hat{g}(\mathcal{X}, \eta).$$  \hspace{1cm} (19)

The minimal value in (19) is attained by setting $\eta$ as

$$\eta_r = (z_r)^{2-q} \|\eta\|_{\frac{q}{q-1}}, \text{ with } z_r = \left( \sum_{k=1}^{d} \|u_r^{(k)}\|_2^2 \right)^{\frac{1}{2}} \forall r \in [1, R].$$  \hspace{1cm} (20)

As a result, we solve the following optimization problem as an alternative to (16):

$$\min_{\{U^{(k)}\}_{k=1}^R, \eta} \hat{f}(\mathcal{X}) = h(\mathcal{X}) + \lambda \hat{g}(\mathcal{X}, \eta).$$  \hspace{1cm} (21)
D. A Block Coordinate Descent Solver for Problem (21)

Now we present an alternating minimization solver for Problem (21). Specifically, we decompose Problem (21) into \((d + 1)\) sub-problems with respect to \(\{U(k)\}_{k=1}^{d}\) and \(\eta\), and we can obtain the analytical solution to each sub-problem.

- **U\(^{(k)}\)-subproblem:** By fixing \(\eta\) and all tensor factors except \(U(k)\), we can see that the variational equality induces a convex subproblem. Based on the first-order optimality condition, \(U(k)\) can be updated analytically:

\[
\text{vec}(U(k)) = (\Phi^T \Phi + \lambda \tilde{\Lambda})^{-1} \Phi^T y. \tag{22}
\]

where \(\Phi = [\Phi_1, \Phi_2, \ldots, \Phi_N]^T\), \(\Phi_n = \text{vec}(B_n^{(k)} U_n^{(k)})^T\) for any \(n \in [1, N]\), and \(\tilde{\Lambda} = \text{diag}(\frac{1}{\eta_1}, \ldots, \frac{1}{\eta_R}) \otimes I \in \mathbb{R}^R(p+1) \times (p+1)\). Here \(\otimes\) denotes a Kronecker product, \(U_n^{(k)}\) is a series of Khatri-Rao product defined as Eq. (7), and \(B_n^{(k)}\) is the \(k\)-th mode matricization of the tensor \(B_n = B(\xi_n)\). For simplicity, we leave the derivation of Eq. (22) to Appendix C.

- **\(\eta\)-subproblem:** Suppose that \(\{U(k)\}_{k=1}^{d}\) are fixed, the formulation and solution of the \(\eta\)-subproblem are shown in (19) and (20), respectively. Suppose that the tensor rank is reduced in the optimization, i.e., \(u_r^{(k)} = 0, \forall k \in [1, d]\), then we can see that \(\eta_r\) will become 0 from Eq. (20). To avoid the numerical issue, let \(\epsilon > 0\) be a small scalar, we update \(\eta\) as

\[
\eta_r = (z_r)^{2-q}\|z\|_q^{q-1} + \epsilon,
\]

with \(z_r = \left(\sum_{k=1}^{d} \|u_r^{(k)}\|_2\right)^{\frac{1}{2}} \forall r \in [1, R]\). \tag{23}

E. Discussions

We would like to highlight a few key points in practical implementations.

- The solution depends on the initialization process. In the first iteration of updating the \(k\)-th factor matrices, we suggest the following initialization

\[
\Phi = [\Phi_1, \Phi_2, \ldots, \Phi_N]^T \quad \text{with} \quad \Phi_n = \text{vec}(O_{n,k}^T), \forall n \in [1, N], \tag{24}
\]

\[
O_{n,k} = \left[\phi^{(k)}(\xi_n^k), \ldots, \phi^{(k)}(\xi_n^p)\right] \in \mathbb{R}^{(p+1) \times R},
\]

where \(\xi_n^k\) is the \(k\)-th variable of sample \(\xi_n\), \(\phi^{(k)}(\xi_n^k) \in \mathbb{R}^{p+1}\) collects all univariate basis functions of \(\xi_n^k\) up to degree \(p\), and \(O_{n,k}\) stores \(R\) copies of \(\phi^{(k)}(\xi_n^k)\). In an adaptive sampling setting (see Section IV), we need to solve (21) after adding new samples. In this case, we use a warm-up initialization by setting the initial guess of \(\{U(k)\}_{k=1}^{d}\) as the solution obtained based on the last-round sampling.

- The regularization parameter \(\lambda\) is highly related to the force of rank shrinkage. To adaptively balance the empirical loss and the rank shrinkage term, we suggest an iterative update of the parameter

\[
\lambda = \lambda_0 \max(\eta), \tag{25}
\]

Algorithm 1: Overall Adaptive Tensor Regression

**Input:** Initial sample pairs \(\{\xi_n, y(\xi_n)\}_{n=1}^{N}\), unitary polynomial order \(p\), initial tensor rank \(R\)

**Output:** Constructed surrogate model [Eq. (26)]

**while Adaptive sampling does not stop**

- Construct the basis tensor \(B(\xi)\)

  **if No additional samples then**

  - Initialize with Eq. (24)

  **else**

  - Initialize \(\{U(k)\}_{k=1}^{d}\) with the last solution

  **while Tensor regression does not stop**

  - for \(k = 1, 2, \ldots, d\) do

    - update \(U(k)\) via Eq. (22)

    - Update \(\eta\) via Eq. (23)

  - Update regularization parameter \(\lambda\) via Eq. (25)

  Shrink the tensor rank to \(R\) if possible

  Select new sample pairs based on Alg. 2

where \(\lambda_0\) is chosen via a cross validation.

The overall algorithm, including an adaptive sampling which will be introduced in Section IV, is summarized in Alg. 1. After solving the factor matrices \(\{U(k)\}_{k=1}^{d}\), i.e. the coefficient tensor \(\lambda\), the surrogate on a sample \(\xi\) can be efficiently calculated as

\[
\hat{y}(\xi) = \langle \lambda, B(\xi) \rangle = \sum_{r=1}^{R} \prod_{k=1}^{d} \left[\phi^{(k)}(\xi_n^k)\right]^T u_r^{(k)}. \tag{26}
\]

In this work tensor \(\lambda\) is approximated by a low-rank CP decomposition. It is also possible to use other kinds of tensor decompositions. In those cases, although the tensor rank are be defined in a different way, the idea of enforcing group-sparsity over tensor factors still works. It is also worth noting that (21) can be seen as a generalization of weighted group lasso. To further exploit the sparsity structure of the gPC coefficients, many variants can be developed from the statistic regression perspective, including the sparse group lasso, tensor-structured Elastic-Net regression and so forth [44].

IV. ADAPTIVE SAMPLING APPROACH

Another fundamental question in uncertainty quantification is how to select the parameter samples \(\xi\) for simulation. We aim to reduce the simulation cost by selecting only a few informative samples for detailed device- or circuit-level simulations.

Given a set of initial samples \(\Theta\), we design a two-stage method to balance the exploration and exploitation in our active sampling process. In the first stage we estimate the volume of some Voronoi cells via a Monte Carlo method to measure the sampling density in each region. In the second stage, we roughly measure the nonlinearity of \(y(\xi)\) at some candidate samples via a Taylor expansion. We choose new samples that are located in a low-density region and make \(y(\xi)\) highly nonlinear. In our implementation, the initial samples \(\Theta = \{\xi_n, y(\xi_n)\}_{n=1}^{N}\) are generated by the Latin Hybercube
Specifically, each sample density in some standard LH samples {\(LH\) sampling method [45]. Specifically, we first generate given a sample \(\xi\), we measure the non-linearity of \(y(\xi)\) via the difference of \(y(\xi)\) and its first-order Taylor expansion around the closest Voronoi cell center \(a \in \Omega\) [48]. We do not know exactly the expression of \(y(\xi)\), but we have already built a surrogate model \(\hat{y}(\xi)\) based on previous simulation samples. Therefore, the nonlinearity of \(y(\xi)\) can be roughly estimate as

\[
\gamma(\xi) = |\hat{y}(\xi) - \hat{y}(a) - \nabla\hat{y}(a)^T(\xi - a)|. 
\] (29)

In the second stage, we will choose the sample \(\xi^*\) that has the largest \(\gamma(\xi)\) from the candidate set of \(\Gamma_\Omega\):

\[
\xi^* = \arg\max_{\xi \in \Gamma_\Omega} (\gamma(\xi)). \tag{30}
\]

To summarize, we select the most nonlinear sample from the least-sampled cell space, which is a good trade-off between exploration and exploitation. Based on the above, we summarize the adaptive sampling procedure in Alg. 2.

C. Discussion

The proposed adaptive sampling method can be easily extended to a batch version by searching for the top-\(K\) least-sampled regions in the first stage. We can stop sampling when we exceed a sampling budget or when the constructed surrogate model achieves a desired accuracy.

The sampling criteria do not rely on the structure of the targeted surrogate model. Therefore, the proposed sampling method can be easily extended to a batch version by searching for the top-\(K\) least-sampled regions in the first stage. We can stop sampling when we exceed a sampling budget or when the constructed surrogate model achieves a desired accuracy.
Algorithm 2: Adaptive sampling procedure

Input: Initial samples pairs \( \Theta = \{ \xi_n, y(\xi_n) \} \)
Output: Sample pairs \( \Theta^* \) with the additional sample
Uniformly generate \( M = 100N \) Monte Carlo samples
\( \{ \psi_m \}_{m=1}^{M} \in [0,1]^d \)
for \( m = 1, 2, \ldots, M \) do
  Find the closest cell \( C_n \) center to \( \psi_m \)
  \[ \text{vol}(C_n) \leftarrow \text{vol}(C_n) + 1 \]
  Find the cell with the biggest estimated volume \( \text{vol} \)
  and the sample set \( \Gamma \) assigned to this cell
  \( \Gamma_{\Theta} \leftarrow \text{Inverse_transform_sampling}(\Gamma) \)
  Calculate the nonlinearity measure \( \gamma(\Gamma_{\Theta}) \) via Eq. (29)
  Select \( \xi^* \) according to Eq. (30)
  \( \Theta^* \leftarrow \Theta \cup \{ \xi^*, y(\xi^*) \} \)

where \( \mathbb{E}[y(\xi)|\xi_j] \) denotes the conditional expectation of \( y(\xi) \) over all random variables except \( \xi_j \). The variance of this conditional expectation can be estimated as

\[
\text{Var} \left[ \mathbb{E}[y(\xi)|\xi_j] \right] = \sum_{i_j=2}^{p+1} x_{1\ldots i_j-1\ldots1\ldots}^2 - x_{1\ldots}^2 \quad (34)
\]

The total sensitivity index \( T_j \) measures the contribution to the variance of \( y(\xi) \) by variable \( \xi_j \) and its interactions with all other variables:

\[
T_j = 1 - \frac{\text{Var} \left[ \mathbb{E}[y(\xi)|\xi_j] \right]}{\sigma^2} \quad (35)
\]

Here \( \xi_{ij} \) includes all elements of \( \xi \) except \( \xi_j \). The involved variance of a conditional expectation is estimated as

\[
\text{Var} \left[ \mathbb{E}[y(\xi)|\xi_{ij}] \right] = \sum_{(i_1, i_2, \ldots, i_d), i_j=1}^{R} x_{i_1\ldots i_d\ldots}^2 - x_{i_1\ldots}^2 \quad (36)
\]

Similarly, we can also express any higher-order index representing the effect from the interaction between a set of variables with an analytical form.

VI. NUMERICAL RESULTS

In this section, we will verify the proposed tensor-regression uncertainty quantification method in one synthetic function and two photonic/electronic IC benchmarks.

A. Baseline Methods for Comparison

We compare our proposed method with the following approaches.

- Tensor regression with adaptive sampling based on space exploration only introduced in Section IV-A (denoted as Space).
- Tensor regression with adaptive sampling based on exploiting nonlinearity only introduced in model with only Section IV-B (denoted as Nonlinear).
- Tensor regression model with random sampling (denoted as Rand). In each iteration of adding samples, new samples are simply randomly selected.
- Fixed-rank tensor regression (denoted as Fixed rank). This method uses a tensor ridge regularization in the regression objective function [35]:

\[
\min_{\{u^{(k)}\}_{k=1}^{d}} f(\mathcal{X}) = h(\mathcal{X}) + \lambda \sum_{k=1}^{d} ||u^{(k)}||_2^2 \quad (37)
\]

The standard ridge regression does not induce a sparse structure. We will keep the tensor rank fixed in solving Eq. (37).
Fig. 3. Results of approximating the synthetic function. (a) Testing error on $10^5$ MC samples. (b) The estimated rank. (c) Probability density functions of the function value.

Fig. 4. Sensitivity analysis of the synthetic function in (39). The proposed method fits the results from Monte Carlo [49] with $10^7$ simulations very well.

B. Synthetic Function (100-dim)

We first consider the following high-dimensional analytical function [52]:

$$f(\xi) = 3 - \frac{5}{d} \sum_{k=1}^{d} k\xi_k + \frac{1}{d} \sum_{k=1}^{d} k\xi_k^3 + \xi_1\xi_2^2 + \xi_2\xi_4 - \xi_3\xi_5 + \xi_{51} + \xi_{50}\xi_{54} + \ln \left( \frac{1}{3d} \sum_{k=1}^{d} k(\xi_k^2 + \xi_k^4) \right)$$

(39)

where dimension $d = 100$, $\xi_{20} \sim \mathcal{U}([1, 3])$, and $\xi_k \sim \mathcal{U}([1, 2])$, $k \neq 20$. We aim to approximate $f(\xi)$ by a tensor-regression gPC model and perform sensitivity analysis.

Assume that we use 2nd-order univariate basis functions for each random variable, then we will need $3^{100}$ multivariate basis functions in total. To approximate the coefficient tensor, we initialize it with a rank-5 CP decomposition and use $q = 0.5$ in regularization. We initialize the training with 200 Latin-Hypercube samples, and adaptively select 9 batches of additional samples with each batch having 20 new samples. We test the accuracy of different models on additional $10^5$ samples. Fig. 3 (a) shows the relative $\ell_2$ testing errors of different methods. The proposed method outperforms all other methods. Fig. 3 (b) shows the estimated tensor rank as the number of training samples increases. The proposed method shrinks the tensor rank differently with other methods while achieving the best performance. It shows that a correct determination of the tensor rank helps the function approximation. Fig. 3 (c) plots the predicted probability density function of our obtained model and it matches the Monte-Carlo simulation result of the original function very well.

We compare the complexity and accuracy of different methods in Table I. We treat the result from $10^5$ Monte Carlo simulations as the ground truth. For the other models the mean and standard deviation are both extracted from the polynomial coefficients. Given the same amount of (limited) training samples, the proposed method achieves the highest approximation accuracy.

Now we perform sensitivity analysis to identify the random variables that are most influential to the output. Fig. 4 plots the main and total sensitivity metrics from the proposed method and from a Monte Carlo estimation [49] with $10^7$ simulations.

### TABLE I

**MODEL COMPARISONS ON THE SYNTHETIC FUNCTION.**

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Variable #</th>
<th>Mean</th>
<th>Stdvar</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>$10^5$</td>
<td>N/A</td>
<td>-162.95</td>
<td>4.80</td>
</tr>
<tr>
<td>Sparse gPC</td>
<td>380</td>
<td>5151</td>
<td>-163.04</td>
<td>2.27</td>
</tr>
<tr>
<td>Fixed rank</td>
<td>380</td>
<td>15x100</td>
<td>-163.24</td>
<td>5.02</td>
</tr>
<tr>
<td><strong>Proposed</strong></td>
<td>380</td>
<td>3x100</td>
<td>-162.93</td>
<td>4.86</td>
</tr>
</tbody>
</table>

*In the alternating solver, there are 100 subproblems with 3 unknown variables in each one (the rank has been shrunk).
We aim to approximate the 3-dB bandwidth rings and between the first/last ring and the bus waveguides. A total of 19 independent Gaussian 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 adjacent rings and between the first/last ring and the bus waveguides. We aim to approximate the 3-dB bandwidth \(f_{3\text{dB}}\) at the DROP port as a tensor-regression gPC model.

We use 2nd order univariate polynomial basis functions for each random parameter, and have 3\(^{19}\) multivariate basis functions in total in the tensor regression gPC model. We initialize the gPC coefficients as a rank-4 CP tensor decomposition, and set \(q = 0.5\) in our regularization. We initialize the training with 60 Latin-Hypercube samples, and adaptively select 300 additional samples in total by 6 batches. We test the obtained model with \(3 \times 10^5\) samples. Fig. 6 (a) shows the relative \(\ell_2\) testing errors of all methods. The proposed method outperforms other methods significantly when the number of samples is small. Fig. 6 (b) shows that the tensor rank reduces to 2 in all methods. Fig. 6 (c) plots the predicted probability density function of obtained tensor regression model, which is indistinguishable from the result of Monte Carlo simulation.

We do an one-shot approximation with different initial tensor ranks \(R\) and different regularization parameters \(\lambda\) as illustrated in Fig. 9. For a specific benchmark, the best estimated tensor rank highly depends on the number of training samples. Given the limited number of simulation samples, the rank-1 initialization works the best in this example. It coincides with the results shown in Fig. 6, where the predicted rank is 1. We also compare the complexity and accuracy of all methods in Table II. The proposed method achieves the best accuracy with limited simulation samples.

### TABLE II

**MODEL COMPARISONS ON THE PHOTONIC BAND-PASS FILTER.**

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Variable #</th>
<th>Mean</th>
<th>Stdvar</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>(10^5)</td>
<td>N/A</td>
<td>21.6511</td>
<td>0.0938</td>
</tr>
<tr>
<td>Sparse gPC</td>
<td>100</td>
<td>210</td>
<td>21.6537</td>
<td>0.0735</td>
</tr>
<tr>
<td>Fixed rank</td>
<td>100</td>
<td>12x19</td>
<td>21.6677</td>
<td>0.1906</td>
</tr>
<tr>
<td>Proposed</td>
<td>100</td>
<td>3x19</td>
<td>21.6567</td>
<td>0.0955</td>
</tr>
</tbody>
</table>

With much fewer function evaluations, our proposed method can precisely identify the indices of some most dominant random variables that contribute to the variance of output.

### TABLE III

**MODEL COMPARISONS ON THE CMOS RING OSCILLATOR.**

<table>
<thead>
<tr>
<th>Sample #</th>
<th>Variable #</th>
<th>Mean</th>
<th>Stdvar</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monte Carlo</td>
<td>(3 \times 10^4)</td>
<td>N/A</td>
<td>12.7920</td>
<td>0.3829</td>
</tr>
<tr>
<td>Sparse gPC</td>
<td>600</td>
<td>1711</td>
<td>12.7931</td>
<td>0.3777</td>
</tr>
<tr>
<td>Fixed rank</td>
<td>600</td>
<td>12x57</td>
<td>12.7929</td>
<td>0.3822</td>
</tr>
<tr>
<td>Proposed</td>
<td>600</td>
<td>6x57</td>
<td>12.7918</td>
<td>0.3830</td>
</tr>
</tbody>
</table>

In order to see the influence of the tensor rank initialization, we do the one-shot approximation with different initial tensor ranks \(R\) and different regularization parameters \(\lambda\) as illustrated in Fig. 7. For a specific benchmark, the best estimated tensor rank highly depends on the number of training samples. Given the limited number of simulation samples, the rank-1 initialization works the best in this example. It coincides with the results shown in Fig. 6, where the predicted rank is 1. We also compare the complexity and accuracy of all methods in Table II. The proposed method achieves the best accuracy with limited simulation samples.

### D. CMOS Ring Oscillator (57-dim)

We continue to consider the 7-stage CMOS ring oscillator in Fig. 8. This circuit has 57 random variation parameters, including Gaussian parameters describing the temperature, variations of threshold voltages and gate-oxide thickness, and uniform-distribution parameters describing the effective gate length/width. We aim to approximate the oscillator frequency with tensor-regression gPC under the process variations.

We use 2nd-order univariate basis functions for each random parameter, leading to \(3^{57}\) multivariate basis functions in total. We initialize the gPC coefficients as a rank-4 tensor and set \(q = 0.5\) in the regularization term. We initialize the training with 500 Latin-Hypercube samples, and adaptively select 300 additional samples in total by 6 batches. We test the obtained model with \(3 \times 10^4\) additional samples. Fig. 10 (a) shows the relative \(\ell_2\) testing errors of all methods. The proposed method outperforms other methods significantly when the number of samples is small. Fig. 10 (b) shows that the estimated tensor rank reduces to 2 in all methods. Fig. 10 (c) plots the predicted probability density function of obtained tensor regression model, which is indistinguishable from the result of Monte Carlo simulation.

We do an one-shot approximation with different initial tensor ranks \(R\) and different regularization parameters \(\lambda\) as illustrated in Fig. 9. Conforming with the results shown in Fig. 10, a rank-2 model is more suitable in this example. We compare the proposed method with the fixed rank model and the 2nd-order sparse gPC in Table III, where the proposed compact tensor model is shown to have the best approximation accuracy.

### VII. Conclusion

This paper has proposed a tensor regression framework for quantifying the impact of high-dimensional process variations. By low-rank tensor representation, this formulation can reduce the number of unknown variables from an exponential

---

**Fig. 5.** Schematic of a band-pass filter with 9 micro-ring resonators.
Fig. 6. Result of the photonic filter. (a) Testing error on $10^5$ MC samples. (b) The estimated rank. (c) Probability density functions of the 3-dB bandwidth $f_{3dB}$ at the DROP port.

Fig. 7. One-shot approximations for the photonic band-pass filter with 800 training samples under different ranks and $\lambda$. The rank-1 initialization works the best in this example.

Fig. 8. Schematic of a CMOS ring oscillator.

Fig. 9. One-shot approximations for the CMOS ring oscillator with 150 training samples under different ranks and $\lambda$. In this example, the rank-2 model works the best in most cases.

function of parameter dimensionality to only a linear one, therefore it works well with a limited simulation budget. We have addressed two fundamental challenges: automatic tensor rank determination and adaptive sampling. The tensor rank is estimated via a $\ell_q/\ell_2$-norm regularization. The simulation samples are chosen based on a two-stage adaptive sampling method, which utilizes the Voronoi cell volume estimation and the nonlinearity measure of the quantity of interest. Our model has been verified by both synthetic and realistic examples with 19 to 100 random parameters. The numerical experiments have shown that our method can well capture the high-dimensional stochastic performance with much less simulation data.

**APPENDIX A**

**BASIS FUNCTION CONSTRUCTION**

The classical families of univariate orthogonal polynomials (including both continuous and discrete ones) for some distributions are listed in Table IV [4]. For other kinds of distributions, the orthonormal polynomial basis can be constructed via a three-term recurrence relation [53].

**APPENDIX B**

**PROOF OF LEMMA 1**

We consider two cases $\alpha \in (0, 2)$ [43] and $\alpha = 2$ [35].
When $\alpha \in (0,2)$, $\kappa(z) := \frac{1}{2} \sum_{j=1}^{p} \frac{y_j^2}{z_j} + \frac{1}{2} ||z||_\beta$ is a continuously differentiable function for any $z_i \in (0, \infty)$. When $y_j \neq 0$, $\lim_{z_j \to \infty} \kappa(z) = \infty$ and $\lim_{z_j \to 0} \kappa(z) = \infty$. Therefore, the infimum of $\kappa(z)$ exists and it is attained. According to the first-order optimality and enforcing the derivative w.r.t. $z_j \ (z_j > 0)$ to be zero, we can obtain

$$z_j = |y_j|^{2-\alpha} ||z||_{\frac{\alpha}{\alpha-1}}. \quad (40)$$

With $y_j = \|z\|_{\frac{1-\alpha}{\alpha}} \|z\|_{\frac{\alpha}{\alpha-1}} z_j \|z\|_{\frac{\alpha-1}{\alpha}}$, we have $\|z\|_{\frac{\alpha}{\alpha-1}} = \|z\|_{\frac{1-\alpha}{\alpha}} (\sum_{j=1}^{p} z_j^{\frac{\alpha}{\alpha-1}})^{-\frac{\alpha}{\alpha-1}} = \|y\|_{\alpha}$, therefore we obtain the optimal solution $z_j = |y_j|^{2-\alpha} ||y||_{\alpha}^{-1}$ in Lemma 1. If $y_j = 0$, the solution to $\min_{z_j \geq 0} \kappa(z)$ is $z_j = 0$, which is also consistent with Lemma 1.

When $\alpha = 2$, $\|z\|_1$ is non-differentiable. Given a scalar $y_j$, we have $y_j = \frac{y_j}{2z_j} + \frac{1}{2} z_j$ only when $z_j = y_j$ (we let $\frac{y_j^2}{2z_j} = 0$ when $y_j = \frac{y_j}{2z_j} = 0$). Similarly, given a vector $y \in \mathbb{R}^p$, we have $\|y\|_1 = \frac{1}{2} \sum_{j=1}^{p} y_j^2 + \frac{1}{2} ||z||_1$ only when $z = |y|$, which is also consistent with Lemma 1.

APPENDIX C
DETAILED DERIVATION OF EQ. (22)

Let $\Lambda = \text{diag}(\frac{1}{n}, \ldots, \frac{1}{n})$ and $*$ denote a Hadamard product, we can rewrite the objective function of an $U^{(k)}$ subproblem as

$$f_{k}(U^{(k)})$$

$$= \frac{1}{2} \sum_{n=1}^{R} \left[ y_n - \langle U^{(k)}(\hat{U}^{(k)}_n)^T, B_n^{(k)} \rangle \right]^2 + \frac{1}{2} \sum_{r=1}^{R} \frac{||u_r^{(k)}||_2^2}{\eta_r}$$

$$= \frac{1}{2} \sum_{n=1}^{R} \left[ y_n - \text{Tr} \left( U^{(k)} \left( B_n^{(k)} \right)^T \right) \right]^2 + \frac{1}{2} \text{Tr} \left( U^{(k)} \Lambda U^{(k)}^T \right)$$

with $B_n^{(k)} = B_n^{(k)} U^{(k)}$. Enforcing the following 1st-order optimality condition

$$\frac{\partial f_{k}(U^{(k)})}{\partial U^{(k)}} = 0$$

we can obtain the analytical solution in Eq. (22).

APPENDIX D
AN EXAMPLE TO SHOW OBSERVATION 1

Suppose we already have two samples $[0.2, 0.6]$, and we consider a candidate sample 0.4 in the interval $[0,1]$ equipped with a uniform distribution. Based on Box–Muller transform, their corresponding Gaussian-distributed samples are $[-0.8416, 0.2533]$ and $-0.2533$, respectively. It is easy to find that the PDF value of 0.2533 is larger than that of $-0.8416$ in a standard Gaussian distribution. Apparently, the candidate sample is equally close to the two examples in a uniformly-sampled space, but it is closer to the one with a higher probability density in the Gaussian-sampled space.

REFERENCES


