Tensor Computation: A New Framework for High-Dimensional Problems in EDA

Zheng Zhang, Kim Batselier, Haotian Liu, Luca Daniel and Ngai Wong

(Invited Keynote Paper)

***Abstract*—Many critical EDA problems suffer from the curse of dimensionality, i.e. the very fast-scaling computational burden produced by large number of parameters and/or unknown variables. This phenomenon may be caused by multiple spatial or temporal factors (e.g. 3-D field solvers discretizations and multi-rate circuit simulation), nonlinearity of devices and circuits, large number of design or optimization parameters (e.g. full-chip routing/placement and circuit sizing), or extensive process variations (e.g. variability/reliability analysis and design for manufacturability). The computational challenges generated by such high dimensional problems are generally hard to handle efficiently with traditional EDA core algorithms that are based on matrix and vector computation. This paper presents "tensor computation" as an alternative general framework for the development of efficient EDA algorithms and tools. A tensor is a high-dimensional generalization of a matrix and a vector, and is a natural choice for both storing and solving efficiently high-dimensional EDA problems. This paper gives a basic tutorial on tensors, demonstrates some recent examples of EDA applications (e.g., nonlinear circuit modeling and high-dimensional uncertainty quantification), and suggests further open EDA problems where the use of tensor computation could be of advantage.**

I. Introduction

A. Success of Matrix & Vector Computation in EDA Hystory

The advancement of fabrication technology and the development of Electronic Design Automation (EDA) are two engines that have been driving the progress of semiconductor industries. The first integrated circuit (IC) was invented in 1959 by Jack Kilby. However, until the early 1970s designers could only handle a small number of transistors manually. The idea of EDA, namely designing electronic circuits and systems automatically using computers, was proposed in the 1960s. Nonetheless, this idea was regarded as science fiction until SPICE [1] was released by UC Berkeley. Due to the success of SPICE, numerous EDA algorithms and tools were further developed to accelerate various design tasks, and designers could design large-scale complex chips without spending months or years on labor-intensive work.

The EDA area indeed encompasses a very large variety of diverse topics, e.g., hardware description languages, logic synthesis, formal verification. This paper mainly concerns computational problems in EDA. Specifically, we focus on modeling, simulation and optimization problems, whose performance heavily relies on effective numerical implementation. Very often, numerical modeling or simulation core tools are called repeatedly by many higher-level EDA tools such as design optimization and system-level verification. Many efficient matrix-based and vector-based algorithms have been developed to address the computational challenges in EDA. Here we briefly summarize a small number of examples among the numerous research results.

In the context of circuit simulation, modified nodal analysis [2] was proposed to describe the dynamic network of a general electronic circuit. Standard numerical integration and linear/nonlinear equation solvers (e.g., Gaussian elimination, LU factorization, Newton's iteration) were implemented in the early version of SPICE [1]. Driven by communication IC design, specialized RF simulators were developed for periodic steady-state [3]–[7] and noise [8] simulation. Iterative solvers and their parallel variants were further implemented to speed up large-scale linear [9]–[11] and nonlinear circuit simulation [12], [13]. In order to handle process variations, both Monte Carlo [14], [15] and stochastic spectral methods [16]–[26]) were investigated to accelerate stochastic circuit simulation.

Efficient models were developed at almost every design level of hierarchy. At the process level, many statistical and learning algorithms were proposed to characterize manufacturing process variations [27]–[29]. At the device level, a huge number of physics-based (e.g., BSIM [30] for MOSFET and RLC interconnect models) and math-based modeling frameworks were reported and implemented. Math-based approaches are also applicable to circuit and system-level problems due to their generic formulation. They start from a detailed mathematical description [e.g., a partial differential equation (PDE) or integral equation describing device physics [31]–[33] or a dynamic system describing electronic circuits] or some measurement data, then generate compact models by model order reduction [34]–[42] or system identification [43]–[46]. These techniques were further extended to problems with design parameters or process uncertainties [47]–[55].

Thanks to the progress of numerical optimization [56]–[58], many algorithmic solutions were developed to solve EDA problems such as VLSI placement [59], routing [60], logic synthesis [61] and analog/RF circuit optimization [62], [63]. Based on design heuristics or numerical approximation, the performance of many EDA optimization engines were accelerated. For instance, in analog/RF circuit optimization, posynomial or polynomial performance models were extracted

Z. Zhang and L. Daniel are with Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA. E-mails: {z_zhang, luca}@mit.edu

K. Batselier and N. Wong are with Department of Electrical and Electronic Engineering, the University of Hong Kong. E-mails: {kimb, nwong}@eee.hku.hk

H. Liu is with Cadence Design Systems, Inc. San Jose, CA. E-mail: haotian@cadence.com

to significantly reduce the number of circuit simulations [64]–[66].

B. Algorithmic Challenges and Motivation Examples

Despite the success in many EDA applications, conventional matrix-based and vector-based algorithms have certain intrinsic limitations when applied to problems with high dimensionality. These problems generally involve an extremely large number of unknown variables or require many simulation/measurement samples to characterize a quantity of interest. Below we summarize some representative motivation examples among numerous EDA problems:

- **Parameterized 3-D Field Solvers.** Many devices are described by PDEs or integral equations [31]–[33] with d spatial dimensions. With n discretization elements along each spatial dimension (i.e., x-, y- or z-direction), the number of unknown elements is approximately N=n^d in a finite-difference or finite-element scheme. When n is large (e.g. more than thousands and often millions), even a fast iterative matrix solver with $O(N)$ complexity cannot handle a 3-D device simulation. If design parameters (e.g. material properties) are considered and the PDE is further discretized in the parameter space, the computational cost quickly extends beyond the capability of existing matrix- or vector-based algorithms.
- **Multi-Rate Circuit Simulation.** Widely separated time scales appear in many electronic circuits (e.g. switched capacitor filters and mixers), and they are difficult to simulate using standard transient simulators. Multi-time PDE solvers [67] reduce the computational cost by discretizing the differential equation along d temporal axes describing different time scales. Similar to a 3-D device simulator, this treatment also be affected by the curse of dimensionality. Frequency-domain approaches such as multi-tone harmonic balance [68], [69] may be more efficient for some RF circuits with d sinusoidal inputs, but their complexity also becomes prohibitively high as d increases.
- **Probabilistic Noise Simulation.** When simulating a circuit influenced by noise, some probabilistic approaches (such as those based on Fokker-Planck equations [70]) compute the joint density function of its d state variables along the time axis. In practice, the d-variable joint density function must be finely discretized in the d-dimensional space, leading to a huge computational cost.
- **Nonlinear or Parameterized Model Order Reduction.** The curse of dimensionality is a long-standing challenge in model order reduction. In multi-parameter model order reduction [47], [48], [54], [55], a huge number of moments must be matched, leading to a huge-size reduced-order model. In nonlinear model order reduction based on Taylor expansions or Volterra series [36]–[38], the complexity is an exponential function of the highest degree of Taylor or Volterra series. Therefore, existing matrix-based algorithms can only capture low-order nonlinearity.
- **Design Space Exploration.** Consider a classical design space exploration problem: optimize the circuit performance (e.g., small-signal gain of an operational amplifier) by choosing the best values of d design parameters (e.g. the sizes of all transistors). When the performance metric is a strongly nonlinear and discontinuous function of design parameters, sweeping the whole parameter space is possibly the only feasible solution. Even if a small number of samples are used for each parameter, a huge number of simulations are required to explore the whole parameter space.
- **Variability-Aware Design Automation.** Process variation is a critical issue in nano-scale chip design. Capturing the complex stochastic behavior caused by process uncertainties can be a data-intensive task. For instance, a huge number of measurement data points are required to characterize accurately the variability of device parameters [27]–[29]. In circuit modeling and simulation, the classical stochastic collocation algorithm [22]–[24] requires many simulation samples in order to construct a surrogate model. Although some algorithms such as compressed sensing [29], [71] can reduce measurement or computational cost, lots of hidden data information cannot be fully exploited by matrix-based algorithms.

C. Toward Tensor Computations?

In this paper we argue that one effective way to address the above challenges is to utilize tensor computation. Tensors are high-dimensional generalizations of vectors and matrices. Tensors were developed well over a century ago, but have been mainly applied in physics, chemometrics and psychometrics [72]. Due to their high efficiency and convenience in representing and handling huge data arrays, tensors are only recently beginning to be successfully applied in many engineering fields, including (but not limited to) signal processing [73], big data [74], machine learning and scientific computing. Nonetheless, tensors still seem a relatively unexplored and unexploited concept in the EDA field.

The goals and organization of this paper include:

- Providing a hands-on "primer" introduction to tensors and their basic computation techniques (Section II and appendices), as well as the most practically useful techinques such as tensor decomposition (Section III) and tensor completion (Section IV);
- Summarizing, as guiding examples, a few recent tensor-based EDA algorithms, including progress in high-dimensional uncertainty quantification (Section V) and nonlinear circuit modeling and simulation (Section VI);
- Suggesting some theoretical and application open challenges in tensor-based EDA (Sections VII and VIII) in order to stimulate further research contributions.

II. Tensor Basics

This section reviews some basic tensor notions and operations necessary for understanding the key ideas in the paper. Different fields have been using different conventions for tensors. Our exposition will try to use one of the most popular and consistent notation.

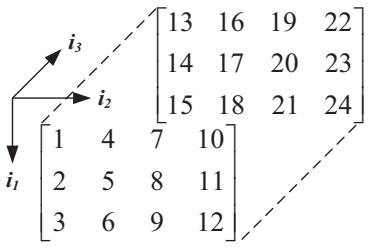


Fig. 1. An example tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{3\times4\times2}$.

A. Notations and Preliminaries

We use boldface capital calligraphic letters (e.g. $\boldsymbol{\mathcal{A}}$) to denote tensors, boldface capital letters (e.g. $\boldsymbol{A}$) to denote matrices, boldface letters (e.g. $\boldsymbol{a}$) to denote vectors, and roman (e.g. a) or Greek (e.g. α) letters to denote scalars.

Tensor. A tensor is a high-dimensional generalization of a matrix or vector. A vector $\boldsymbol{a} \in \mathbb{R}^n$ is a 1-way data array, and its ith element a_i is specified by the index i. A matrix $\boldsymbol{A} \in \mathbb{R}^{n_1\times n_2}$ is a 2-way data array, and each element $a_{i_1 i_2}$ is specified by a row index i_1 and a column index i_2. By extending this idea to the high-dimensional case $d \geq 3$, a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ represents a d-way data array, and its element $a_{i_1 i_2\cdots i_d}$ is specified by d indices. Here, the positive integer d is also called the order of a tensor. Fig. 1 illustrates an example $3 \times 4 \times 2$ tensor.

B. Basic Tensor Arithmetic

Definition 1: **Tensor inner product.** The inner product between two tensors $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times\cdots\times n_d}$ is defined as

$$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1, i_2, \ldots, i_d} a_{i_1\cdots i_d} b_{i_1\cdots i_d}.$$

As norm of a tensor $\boldsymbol{\mathcal{A}}$, it is typically convenient to use the Frobenius norm $||\boldsymbol{\mathcal{A}}||_F := \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}$.

Definition 2: **Tensor k-mode product.** The k-mode product $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_k \boldsymbol{U}$ of a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1\times\cdots\times n_k\times\cdots\times n_d}$ with a matrix $\boldsymbol{U} \in \mathbb{R}^{p_k\times n_k}$ is defined by

$$b_{i_1\cdots i_{k-1} j i_{k+1}\cdots i_d} = \sum_{i_k=1}^{n_k} u_{j i_k} a_{i_1\cdots i_k\cdots i_d}, \tag{1}$$

and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1\times\cdots\times n_{k-1}\times p_k\times n_{k+1}\times\cdots\times n_d}$.

Definition 3: **k-mode product shorthand notation.** The multiplication of a d-way tensor $\boldsymbol{\mathcal{A}}$ with the matrices $\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}$ along each of its d modes respectively is

$$[\![\boldsymbol{\mathcal{A}}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!] \triangleq \boldsymbol{\mathcal{A}} \times_1 \boldsymbol{U}^{(1)} \times_2 \cdots \times_d \boldsymbol{U}^{(d)}.$$

When $\boldsymbol{\mathcal{A}}$ is diagonal with all 1's on its diagonal and 0's elsewhere, then $\boldsymbol{\mathcal{A}}$ is omitted from the notation, e.g. $[\![\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]$.

Definition 4: **Rank-1 tensor.** A rank-1 d-way tensor can be written as the outer product of d vectors

$$\boldsymbol{\mathcal{A}} = \boldsymbol{u}^{(1)} \circ \boldsymbol{u}^{(2)} \circ \cdots \circ \boldsymbol{u}^{(d)} = [\![\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(d)}]\!], \tag{2}$$

where $\boldsymbol{u}^{(1)} \in \mathbb{R}^{n_1}, \ldots, \boldsymbol{u}^{(d)} \in \mathbb{R}^{n_d}$. The entries of $\boldsymbol{\mathcal{A}}$ are completely determined by $a_{i_1 i_2\cdots i_d} = u^{(1)}_{i_1} u^{(2)}_{i_2} \cdots u^{(d)}_{i_d}$.

TABLE I
STORAGE COSTS OF MAINSTREAM TENSOR DECOMPOSITION APPROACHES.

Decomposition	Elements to store	Comments
Canonical Polyadic [81], [82]	ndr	see Fig. 2
Tucker [83]	$r^d + ndr$	see Fig. 3
Tensor Train [84]	$n(d-2)r^2 + 2nr$	see Fig. 4

Some additional notations and operations are introduced in Appendix A. The applications in Sections V and VI will make it clear that the main problems in tensor-based EDA applications are either computing a tensor decomposition or solving a tensor completion problem. Both of them will now be discussed in order.

III. Tensor Decomposition

A. Computational Advantage of Tensor Decompositions.

The number of elements in a d-way tensor is $n_1 n_2 \cdots n_d$, which grows very fast as d increases. Tensor decompositions compress and represent a high-dimensional tensor by a smaller number of factors. As a result, it is possible to solve high-dimensional problems (c.f. Sections V to VII) with a lower storage and computational cost. Table I summarizes the storage cost of three mainstream tensor decompositions. State-of-the-art implementations of these methods can be found in [75]–[77]. As specific examples, for instance:

- While the weight layers of a neural network could consume almost all of the memory in server, using instead a canonical or tensor-train decomposition would result in an extraordinary compression([78], [79]) by up to a factor of $200,000$.
- High-order models describing nonlinear dynamic systems can also be significantly compressed using tensor decomposition as will be shown in details in Section VI.
- High-dimensional integration and convolution are long-standing challenges in many engineering fields (e.g. computational finance and image processing). These two problems can be written as the inner product of two tensors, and while a direct computation would have a complexity of $O(n^d)$, using a low-rank canonical or tensor-train decomposition, results in an extraordinarely lower $O(nd)$ complexity [80].

In this section we will briefly discuss the most popular and useful tensor decompositions, highlighting advantages of each.

B. Canonical Polyadic Decomposition

Polyadic Decomposition. A polyadic decomposition expresses a d-way tensor as the sum of r rank-1 terms:

$$\boldsymbol{\mathcal{A}} = \sum_{i=1}^{r} \sigma_i \, \boldsymbol{u}^{(1)}_i \circ \cdots \circ \boldsymbol{u}^{(d)}_i = [\![\boldsymbol{\mathcal{D}}; \boldsymbol{U}^{(1)}, \cdots, \boldsymbol{U}^{(d)}]\!]. \tag{3}$$

The subscript i of the unit-norm $\boldsymbol{u}^{(1)}_i$ vectors indicates a summation index and not the vector entries. The $\boldsymbol{u}^{(k)}_i$ vectors

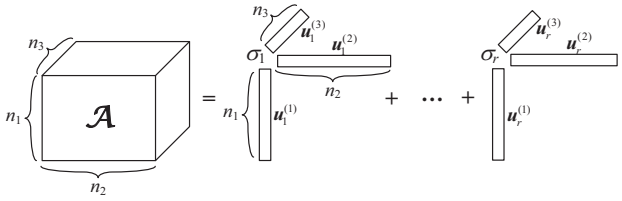


Fig. 2. Decomposing $\boldsymbol{\mathcal{A}}$ into the sum of r rank-1 outer products.

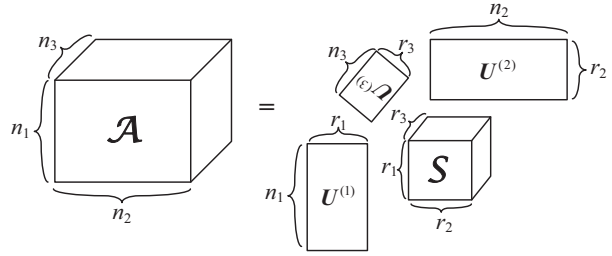


Fig. 3. The Tucker decomposition decomposes a 3-way tensor $\boldsymbol{\mathcal{A}}$ into a core tensor $\boldsymbol{\mathcal{S}}$ and factor matrices $\boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \boldsymbol{U}^{(3)}$.

are called the mode-k vectors. Collecting all vectors of the same mode k in matrix $\boldsymbol{U}^{(k)} \in \mathbb{R}^{n_k \times r}$, this decomposition is rewritten as the k-mode products of matrices $\{\boldsymbol{U}^{(k)}\}_{k=1}^{d}$ with a cubical diagonal tensor $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{r \times \cdots \times r}$ containing all the σ_i values. Note that we can always absorb each of the scalars σ_i into one of the mode vectors, then write $\boldsymbol{\mathcal{A}} = [\![\boldsymbol{U}^{(1)}, \cdots, \boldsymbol{U}^{(d)}]\!]$.

Example 1: The polyadic decomposition of a 3-way tensor is shown in Fig. 2.

Tensor Rank. The minimum $r := R$ for the equality (3) to hold is called the *tensor rank* which, unlike the matrix case, is in general NP-hard to compute [85].

Canonical Polyadic Decomposition (CPD). The corresponding decomposition with the minimal R is called the *canonical polyadic decomposition* (CPD). It is also called *Canonical Decomposition* (CANDECOMP) [81] or *Parallel Factor* (PARAFAC) [82] in the literature. A CPD is unique, up to scaling and permutation of the mode vectors, under mild conditions. A classical uniqueness result for 3-way tensors is described by Kruskal [86]. These uniqueness conditions do not apply to the matrix case[1].

The computation of a polyadic decomposition, together with two variants are discussed in Appendix B.

C. Tucker Decomposition

Tucker Decomposition. Removing the constraint that $\boldsymbol{\mathcal{D}}$ is cubical and diagonal in (3) results in

$$\begin{aligned} \boldsymbol{\mathcal{A}} &= \boldsymbol{\mathcal{S}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \cdots \times_d \boldsymbol{U}^{(d)} \\ &= [\![\boldsymbol{\mathcal{S}}; \boldsymbol{U}^{(1)}, \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(d)}]\!] \end{aligned} \tag{4}$$

[1]Indeed, for a given matrix decomposition $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}$ and any nonsingular matrix $\boldsymbol{T}$ we have that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{T}\boldsymbol{T}^{-1}\boldsymbol{V}$. Only by adding sufficient conditions (e.g. orthogonal or triangular factors) the matrix decomposition can be made unique. Remarkably, the CPD for higher order tensors does not need any such conditions to ensure its uniqueness.

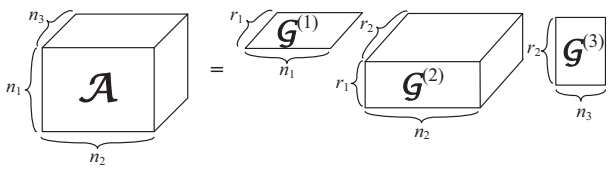


Fig. 4. The Tensor Train decomposition decomposes a 3-way tensor $\boldsymbol{\mathcal{A}}$ into two matrices $\boldsymbol{\mathcal{G}}^{(1)}, \boldsymbol{\mathcal{G}}^{(3)}$ and a 3-way tensor $\boldsymbol{\mathcal{G}}^{(2)}$.

with the factor matrices $\boldsymbol{U}^{(k)} \in \mathbb{R}^{n_k \times r_k}$ and a core tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$. The Tucker decomposition can significantly reduce the storage cost when r_k is (much) smaller than n_k. This decomposition is illustrated in Fig. 3.

Multilinear Rank. The minimal size $(r_1, r_2, \ldots, r_d)$ of the core tensor $\boldsymbol{\mathcal{S}}$ for (4) to hold is called the multilinear rank of $\boldsymbol{\mathcal{A}}$, and it can be computed as $r_1 = \text{rank}(\boldsymbol{\mathcal{A}}_{(1)}), \ldots, r_d = \text{rank}(\boldsymbol{\mathcal{A}}_{(d)})$. Note that $\boldsymbol{\mathcal{A}}_{(k)}$ is a matrix obtained by reshaping (see Appendix A) $\boldsymbol{\mathcal{A}}$ along its kth mode. For the matrix case we have that $r_1 = r_2$, i.e., the row rank equals the column rank. This is not true anymore when $d \geq 3$.

Tucker vs. CPD. The Tucker decomposition can be considered as an expansion in rank-1 terms that is not necessarily canonical, while the CPD does not necessarily have a minimal core. This indicates the different usages of these two decompositions: the CPD is typically used to decompose data into interpretable mode vectors while the Tucker decomposition is most often used to compress data into a tensor of smaller size. Unlike the CPD, the Tucker decomposition is in general not unique[2].

A variant of the Tucker decomposition, called high-order singular value decomposition (SVD) or HOSVD, is summarized in Appendix C.

D. Tensor Train Decomposition

Tensor Train (TT) Decomposition. A tensor train decomposition [84] represents a d-way tensor $\boldsymbol{\mathcal{A}}$ by two 2-way tensors and $(d-2)$ 3-way tensors. Specifically, each entry of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is expressed as

$$a_{i_1 i_2 \cdots i_d} = \boldsymbol{\mathcal{G}}^{(1)}_{i_1} \boldsymbol{\mathcal{G}}^{(2)}_{i_2} \cdots \boldsymbol{\mathcal{G}}^{(d)}_{i_d}, \tag{5}$$

where $\boldsymbol{\mathcal{G}}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ is the k-th core tensor, $r_0 = r_d = 1$, and thus $\boldsymbol{\mathcal{G}}^{(1)}$ and $\boldsymbol{\mathcal{G}}^{(d)}$ are matrices. The vector $(r_0, r_1, \cdots, r_d)$ is called the *tensor train rank*. Each element of the core $\boldsymbol{\mathcal{G}}^{(k)}$, denoted as $g^{(k)}_{\alpha_{k-1} i_k \alpha_{k+1}}$ has three indices. By fixing the 2nd index i_k, we obtain a matrix $\boldsymbol{\mathcal{G}}^{(k)}_{i_k}$ (or vector for $k = 1$ or $k = d$).

Computing Tensor Train Decompositions. Computing a tensor train decomposition consists of doing $d-1$ consecutive reshapings and low-rank matrix decompositions. An advantage of tensor train decomposition is that a quasi-optimal approximation can be obtained with a given error bound and with an automatic rank determination [84].

[2]One can always right-multiply the factor matrices $\boldsymbol{U}^{(k)}$ with any nonsingular matrix $\boldsymbol{T}^{(k)}$ and multiply the core tensor $\boldsymbol{\mathcal{S}}$ with their inverses $\boldsymbol{T}^{(k)^{-1}}$. This means that the subspaces that are defined by the factor matrices $\boldsymbol{U}^{(i)}$ are invariant while the bases in these subspaces can be chosen arbitrarily.

E. Choice of Tensor Decomposition Methods

Canonical and tensor train decompositions are preferred for high-order tensors since the their resulting tensor factors have a low storage cost linearly dependent on n and d. For some cases (e.g., functional approximation), a tensor train decomposition is preferred due to a unique feature, i.e., it can be implemented with cross approximation [87] and *without* knowing the whole tensor. This is very attractive, because in many cases obtaining a tensor element can be expensive. Tucker decompositions are mostly applied to lower-order tensors due to their storage cost of $O(r^d)$, and are very useful for finding the dominant subspace of some modes such as in data mining applications.

IV. Tensor Completion (or Recovery)

Tensor decomposition is a powerful tool to reduce storage and computational cost, however most approaches need a whole tensor *a-priori*. In practice, obtaining each element of a tensor may require an expensive computer simulation or non-trivial hardware measurement. Therefore, it is necessary to estimate a whole tensor based on only a small number of available elements. This can be done by tensor completion or tensor recovery. This idea finds applications in many fields. For instance in biomedical imaging, one wants to reconstruct the whole magnetic resonance imaging data set based on a few measurements. In design space exploration, one may only have a small number of tensor elements obtained from circuit simulations, while all other sweeping samples in the parameter space must be estimated.

A. Ill-Posed Tensor Completion/Recovery

Let $\mathcal{I}$ include all indices for the elements of $\boldsymbol{\mathcal{A}}$, and its subset Ω holds the indices of some available tensor elements. A projection operator $\mathbb{P}_\Omega$ is defined for $\boldsymbol{\mathcal{A}}$:

$$\boldsymbol{\mathcal{B}} = \mathbb{P}_\Omega(\boldsymbol{\mathcal{A}}) \;\Leftrightarrow\; b_{i_1\cdots i_d} = \begin{cases} a_{i_1\cdots i_d}, \text{ if } i_1\cdots i_d \in \Omega \\ 0, \text{ otherwise.} \end{cases}$$

In tensor completion, one wants to find a tensor $\boldsymbol{\mathcal{X}}$ such that it matches $\boldsymbol{\mathcal{A}}$ for the elements specified by Ω:

$$\|\mathbb{P}_\Omega(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{A}})\|_F^2 = 0. \tag{6}$$

This problem is **ill-posed**, because any value can be assigned to $x_{i_1\cdots i_d}$ if $i_1\cdots i_d \notin \Omega$.

B. Regularized Tensor Completion

Regularization makes the tensor completion problem well-posed by adding constraints to (6). Several existing ideas are summarized below.

- **Nuclear-Norm Minimization.** This idea searches for the minimal-rank tensor by solving the problem:

$$\min_{\boldsymbol{\mathcal{X}}} \|\boldsymbol{\mathcal{X}}\|_* \quad \text{s.t. } \mathbb{P}_\Omega(\boldsymbol{\mathcal{X}}) = \mathbb{P}_\Omega(\boldsymbol{\mathcal{A}}). \tag{7}$$

  The nuclear norm of a matrix is the sum of all singular values, but the nuclear norm of a tensor does not have a rigorous or unified definition. In [88], [89], the tensor nuclear norm $\|\boldsymbol{\mathcal{X}}\|_*$ is heuristically approximated using the weighted sum of matrix nuclear norms of $\boldsymbol{\mathcal{X}}_{(k)}$'s for all modes. This heuristic makes (7) convex, and its optimal solution can be computed by available algorithms [90], [91]. Note that in (7) one has to compute a full tensor $\boldsymbol{\mathcal{X}}$, leading to an exponential complexity with respect to the order d.
- **Approximation with Fixed Ranks.** Some techniques compute a tensor $\boldsymbol{\mathcal{X}}$ by fixing its tensor rank. For instance, one can solve the following problem

$$\begin{aligned} \min_{\boldsymbol{\mathcal{X}}} \; & \|\mathbb{P}_\Omega(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{A}})\|_F^2 \\ \text{s. t.} \; & \text{multilinear rank}(\boldsymbol{\mathcal{X}}) = (r_1, \ldots, r_d) \end{aligned} \tag{8}$$

  with $\boldsymbol{\mathcal{X}}$ parameterized by a proper low-multilinear rank factorization. Kresner et al. [92] computes the higher-order SVD representation using Riemannian optimization [93]. In [94], the unknown $\boldsymbol{\mathcal{X}}$ is parameterized by a tensor train decomposition. The low-rank factorization significantly reduces the number of unknown variables. However, how to choose an optimal tensor rank still remains an open question.
- **Probabilistic Tensor Completion.** In order to automatically determine the tensor rank, some probabilistic approaches based on Bayesian statistics have been developed. Specifically, one may treat the tensor factors as unknown random variables assigned with proper prior probability density functions to enforce low-rank properties. This idea has been applied successfully to obtain polyadic decomposition [95], [96] and Tucker decomposition [97] from incomplete data with automatic rank determination.
- **Low-Rank and Sparse Constraints.** In some cases, a low-rank tensor $\boldsymbol{\mathcal{A}}$ may have a sparse property after a linear transformation. Let $\boldsymbol{z} = [z_1, \ldots, z_m]$ with $z_k = \langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{W}}_k \rangle$, one may find that many elements of $\boldsymbol{z}$ are close to zero. To exploit the low-rank and sparse properties simultaneously, the following optimization problem [98], [99] may be solved:

$$\begin{aligned} \min_{\boldsymbol{\mathcal{X}}} \; & \frac{1}{2}\|\mathbb{P}_\Omega(\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{A}})\|_F^2 + \lambda \sum_{k=1}^{m} |\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{W}}_k \rangle| \\ \text{s. t.} \; & \text{multilinear rank}(\boldsymbol{\mathcal{X}}) = (r_1, \ldots, r_d). \end{aligned} \tag{9}$$

  In signal processing, $\boldsymbol{z}$ may represent the coefficients of multidimensional Fourier or wavelet transforms. In uncertainty quantification, $\boldsymbol{z}$ collects the coefficients of a generalized polynomial-chaos expansion. The formulation (9) is generally non-convex, and locating its global minimum is non-trivial.

C. Choice of Tensor Recovery Methods

Low-rank constraints have proven to be a good choice for instance in signal and image processing (e.g., MRI reconstruction) [88], [89], [92], [96]. Both low-rank and sparse properties may be considered for high-dimensional functional approximation (e.g., polynomial-chaos expansions) [98]. Nuclear-norm minimization and probabilistic tensor completion are very attractive in the sense that tensor ranks can be automatically determined, however they are not so efficient or reliable

for high-order tensor problems. It is expensive to evaluate the nuclear norm of a high-order tensor. Regarding probabilistic tensor completion, implementation experience shows that many samples may be required to obtain an accurate result.

V. Applications in Uncertainty Quantification

Tensor techniques can advance the research of many EDA topics due to the ubiquitous existence of high-dimensional problems in the EDA community, especially when considering process variations. This section summarizes some recent progress on tensor-based research in solving high-dimensional uncertainty quantification problems, and could be used as guiding reference for the effective employment of tensors in other EDA problems.

A. Uncertainty Quantification (UQ)

Process variation is one of the main sources causing yield degradation and chip failures. In order to improve chip yield, efficient stochastic algorithms are desired in order to simulate nano-scale designs. The design problems are generally described by complex differential equations, and they have to be solved repeatedly in traditional Monte-Carlo simulators.

Stochastic spectral methods have emerged as a promising candidate due to their high efficiency in EDA applications [16]–[26]. Let the random vector $\boldsymbol{\xi} \in \mathbb{R}^d$ describe process variation. Under some assumptions, an output of interest (e.g., chip frequency) $y(\boldsymbol{\xi})$ can be approximated by a truncated generalized polynomial-chaos expansion [100]:

$$y(\boldsymbol{\xi}) \approx \sum_{|\boldsymbol{\alpha}|=0}^{p} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}). \tag{10}$$

Here $\{\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}$ are orthonormal polynomial basis functions; the index vector $\boldsymbol{\alpha} \in \mathbb{N}^d$ indicates the polynomial order, and its element-wise sum $|\boldsymbol{\alpha}|$ is bounded by p. The coefficient $c_{\boldsymbol{\alpha}}$ can be computed by

$$c_{\boldsymbol{\alpha}} = \mathbb{E}\left(\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) y(\boldsymbol{\xi})\right) \tag{11}$$

where $\mathbb{E}$ denotes expectation.

Main Challenge. Stochastic spectral methods become inefficient when there are many random parameters, because evaluating $c_{\boldsymbol{\alpha}}$ involves a challenging d-dimensional numerical integration. In high-dimensional cases, Monte Carlo was regarded more efficient than stochastic spectral methods. However, we will show that with tensor computation, stochastic spectral methods can outperform Monte Carlo for some challenging UQ problems.

B. High-D Stochastic Collocation by Tensor Recovery

Problem Description. In stochastic collocation [101]–[103], (11) is evaluated by a quadrature rule. For instance, with n_j integration samples and weights [104] properly chosen for each element of $\boldsymbol{\xi}$, $c_{\boldsymbol{\alpha}}$ can be evaluated by

$$c_{\boldsymbol{\alpha}} = \langle \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle. \tag{12}$$

Here both $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ are tensors of size $n_1 \times \cdots \times n_d$. The rank-1 tensor $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ only depends on $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ and quadrature weights, and thus is easy to compute. Obtaining $\boldsymbol{\mathcal{Y}}$ exactly is almost impossible because it has the values of y at all integration samples. Instead of computing all elements of $\boldsymbol{\mathcal{Y}}$ by $n_1 n_2 \cdots n_d$ numerical simulations, we estimate $\boldsymbol{\mathcal{Y}}$ using only a small number of (say, several hundreds) simulations. As shown in compressive sensing [71], the approximation (10) usually has sparse structures, and thus the low-rank and sparse tensor completion model (9) can be used. Using tensor recovery, stochastic collocation may require only a few hundred simulations, thus can be very efficient for some high-dimensional problems.

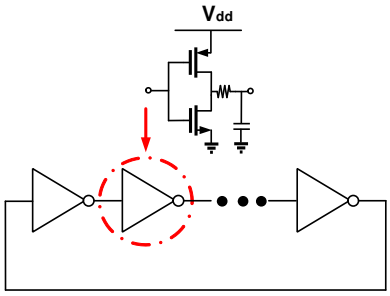


Fig. 5. Schematic of a multistage CMOS ring oscillator (with 7 inverters).

TABLE II
Comparison of simulation cost for the ring oscillator, using different kinds of stochastic collocation.

method	tensor product	sparse grid	tensor completion
total samples	1.6×10^{27}	6844	500

Example [98], [99]. The CMOS ring oscillator in Fig. 5 has 57 random parameters describing threshold voltages, gate-oxide thickness, and effective gate length/width. Since our focus is to handle high dimensionality, all parameters are assumed mutually independent. We aim to obtain a 2nd-order polynomial-chaos expansion for its frequency by repeated periodic steady-state simulations. Three integration points are chosen for each parameter, leading to $3^{57} \approx 1.6 \times 10^{27}$ samples to simulate in standard stochastic collocation. Advanced integration rules such as sparse grid [105] still needs over 6000 simulations. As shown in Table II, with tensor completion (9), the tensor representing 3^{57} solution samples can be well approximated by using only 500 samples. As shown in Fig. 6, the optimization solver converges after 46 iterations, and the tensor factors are computed with less than 1% relative errors; the obtained model is very sparse, and the obtained density function of the oscillator frequency is very accurate.

Why Not Use Tensor Decomposition? Since $\boldsymbol{\mathcal{Y}}$ is not given *a priori*, neither CPD nor Tucker decomposition is feasible here. For the above example our experiments show that tensor train decomposition requires about 10^5 simulations to obtain the low-rank factors with acceptable accuracy, and its cost is even higher than Monte Carlo.

C. High-D Hierarchical UQ with Tensor Train

Hierarchical UQ. In a hierarchical UQ framework, one estimates the high-level uncertainty of a large system that consists of several components or subsystems by applying

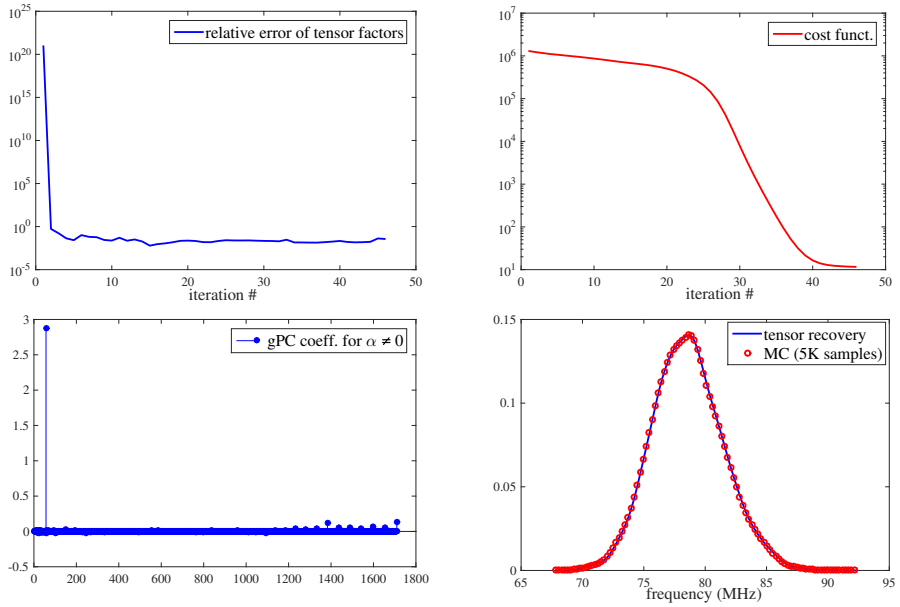


Fig. 6. Tensor-recovery results of the ring oscillator. Top left: relative error of the tensor factors; top right: decrease of the cost function in (9); bottom left: sparsity of the obtained polynomial-chaos expansion; bottom right: obtained density function v.s. Monte Carlo using 5000 samples.

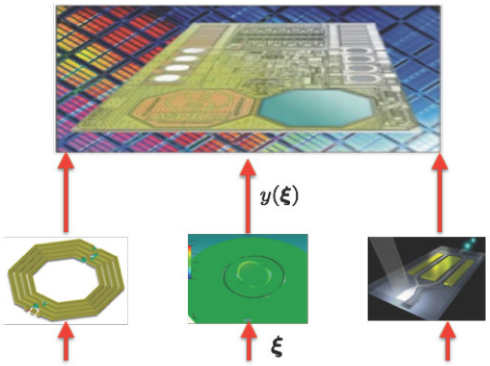


Fig. 7. Hierarchical uncertainty quantification. The stochastic outputs of bottom-level components/devices are used as new random inputs for upper-level uncertainty analysis.

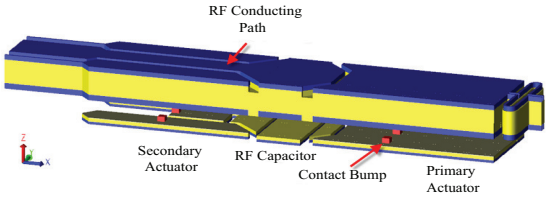
stochastic spectral methods at different levels of the design hierarchy. Assume that several polynomial-chaos expansions are given in the form (10), and each $y(\boldsymbol{\xi})$ describes the output of a component or subsystem. In Fig. 7, $y(\boldsymbol{\xi})$ is used as a new random input such that the system-level simulation can be accelerated by ignoring the bottom-level variations $\boldsymbol{\xi}$. However, the quadrature samples and basis functions of y are unknown, and one must compute such information using a 3-term recurrence relation [104]. This requires evaluating the following numerical integration with high accuracy:

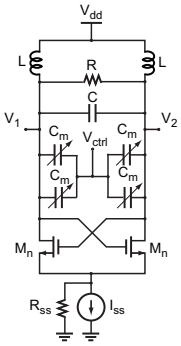
$$\mathbb{E}\left(g\left(y\left(\boldsymbol{\xi}\right)\right)\right) = \langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{W}} \rangle, \tag{13}$$

where the elements of tensors $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{\hat{n}_1 \times \cdots \times \hat{n}_d}$ are $g(y(\boldsymbol{\xi}_{i_1 \cdots i_d}))$ and $w_1^{i_1} \cdots w_d^{i_d}$, respectively. Note that $\boldsymbol{\xi}_{i_1 \cdots i_d}$ and $w_1^{i_1} \cdots w_d^{i_d}$ are the d-dimensional numerical quadrature samples and weights, respectively.

Choice of Tensor Decompositions. We aim to obtain a low-rank representation of $\boldsymbol{\mathcal{Y}}$, such that $\boldsymbol{\mathcal{G}}$ and $\mathbb{E}\left(g\left(y\left(\boldsymbol{\xi}\right)\right)\right)$ can be computed easily. Due to the extremely high accuracy requirement in the 3-term recurrence relation [104], tensor completion methods are not feasible. Neither canonical tensor decomposition nor Tucker decomposition is applicable here, as they need the whole high-way tensor $\boldsymbol{\mathcal{Y}}$ before factorization. Tensor-train decomposition is a good choice, since it can compute a high-accuracy low-rank representation without knowing the whole tensor $\boldsymbol{\mathcal{Y}}$; therefore, it was used in [18] to accelerate the 3-term recurrence relation and the subsequent hierarchical UQ flow.

Example. The tensor-train-based flow has been applied to the oscillator with four MEMS capacitors and 184 random parameters shown in Fig. 8, which previously could only be solved using random sampling approaches. In [18], a sparse generalized polynomial-chaos expansion was first computed as a stochastic model for the MEMS capacitor $y(\boldsymbol{\xi})$. The discretization of $y(\boldsymbol{\xi})$ on a 46-dimensional integration grid was represented by a tensor $\boldsymbol{\mathcal{Y}}$ (with 9 integration points along each dimension), then $\boldsymbol{\mathcal{Y}}$ was approximated by a tensor train decomposition. After this approximation, (13) was easily computed to obtain the new orthonormal polynomials and quadrature points for y. Finally, a stochastic oscillator simulator [21] was called at the system level using the newly obtained basis functions and quadrature points. As shown in Table III, this circuit was simulated by the tensor-train-based hierarchical approach in only 10 min in MATLAB, whereas Monte Carlo with 5000 samples required more than 15 hours [18]. The variations of the steady-state waveforms from both methods are almost the same, cf. Fig. 9.





Parameter	Value
V_{dd}	2.5V
V_{ctrl}	0-2.5V
C	10.83fF
R	110Ω
L	8.8nH
$(W/L)_n$	4/0.25
I_{ss}	0.5mA
R_{ss}	10^6Ω

Fig. 8. Left: the schematic of a MEMS switch acting as capacitor C_{m}, which has 46 process variations; right: an oscillator using 4 MEMS switches as capacitors (with 184 random parameters in total).

TABLE III
SIMULATION TIME OF THE MEMS-IC CO-DESIGN IN FIG. 8

method	Monte Carlo	proposed [18]
total samples	15.4 hours	10 minutes

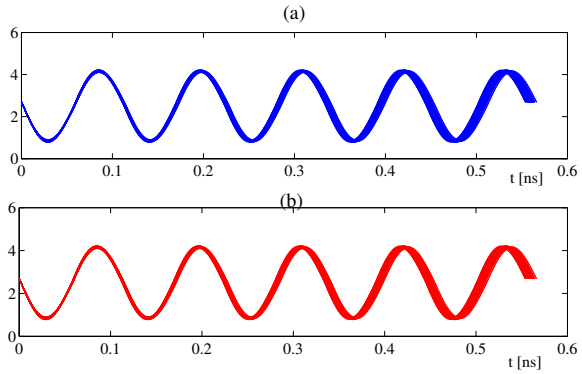


Fig. 9. Realization of the steady-state waveforms for the oscillator in Fig. 8. Top: tensor-based hierarchical approach; bottom: Monte Carlo.

VI. Applications in Nonlinear Circuit Modeling

Nonlinear devices or circuits must be well modeled in order to enable efficient system-level simulation and optimization. Capturing the (possibly high) nonlinearity can result in high-dimensional problems. Fortunately, the multiway nature of a tensor allows the easy capturing of high-order nonlinearities of analog, mixed-signal circuits and in MEMS design.

A. Nonlinear Modeling and Model Order Reduction

Similar to the Taylor expansion, it is shown in [37], [38], [106]–[108] that many nonlinear dynamical systems can be approximated by expanding the nonlinear terms around an equilibrium point, leading to the following ordinary differential equation

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{x}^{(2)} + \boldsymbol{C}\boldsymbol{x}^{(3)} + \boldsymbol{D}(\boldsymbol{u}\otimes\boldsymbol{x}) + \boldsymbol{E}\boldsymbol{u}, \qquad (14)$$

where the state vector $\boldsymbol{x}(t) \in \mathbb{R}^n$ contains the voltages and/or currents inside a circuit network, and the vector $\boldsymbol{u}(t) \in \mathbb{R}^m$ denotes time-varying input signals. The $\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}$ notation refers to repeated Kronecker products (cf. Appendix A). The matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ describes linear behavior, while the matrices $\boldsymbol{B} \in \mathbb{R}^{n\times n^2}$ and $\boldsymbol{C} \in \mathbb{R}^{n\times n^3}$ describe 2nd- and 3rd-order polynomial approximations of some nonlinear behavior. The matrix $\boldsymbol{D} \in \mathbb{R}^{n\times nm}$ captures the coupling between the state variables and input signals and $\boldsymbol{E} \in \mathbb{R}^{n\times m}$ describes how the input signals are injected into the circuit. This differential equation will serve as the basis in the following model order reduction applications.

Matrix-based Nonlinear Model Order Reduction. The idea of nonlinear model order reduction is to extract a compact reduced-order model that accurately approximates the input-output relationship of the original large nonlinear system. Simulation of the reduced-order model is usually much faster, so that efficient and reliable system-level verification is obtained. For instance, projection-based nonlinear model order reduction methods reduce the original system in (14) to a compact reduced model with size $q \ll n$

$$\dot{\hat{\boldsymbol{x}}} = \hat{\boldsymbol{A}}\hat{\boldsymbol{x}} + \hat{\boldsymbol{B}}\hat{\boldsymbol{x}}^{(2)} + \hat{\boldsymbol{C}}\hat{\boldsymbol{x}}^{(3)} + \hat{\boldsymbol{D}}(\boldsymbol{u}\otimes\hat{\boldsymbol{x}}) + \hat{\boldsymbol{E}}\boldsymbol{u}, \qquad (15)$$

where $\hat{x} \in \mathbb{R}^q$, $\hat{\boldsymbol{A}} \in \mathbb{R}^{q\times q}$, $\hat{\boldsymbol{B}} \in \mathbb{R}^{q\times q^2}$, $\hat{\boldsymbol{C}} \in \mathbb{R}^{q\times q^3}$, $\hat{\boldsymbol{D}} \in \mathbb{R}^{q\times qm}$ and $\hat{\boldsymbol{E}} \in \mathbb{R}^{q\times m}$. The reduction is achieved through applying an orthogonal projection matrix $\boldsymbol{V} \in \mathbb{R}^{n\times q}$ on the system matrices in (14). Fig. 10 illustrates how projection-based methods reduce $\boldsymbol{B}$ to a dense system matrix $\hat{\boldsymbol{B}}$ with a smaller size.

Most traditional matrix-based weakly nonlinear model order reduction methods [36]–[40] suffer from the exponential growth of the size of the reduced system matrices $\hat{\boldsymbol{B}}, \hat{\boldsymbol{C}}, \hat{\boldsymbol{D}}$. As a result, simulating high-order strongly nonlinear reduced models is sometimes even slower than simulating the original system.

Tensor-based Nonlinear Model Order Reduction. A tensor-based reduction scheme was proposed in [109]. The coefficient matrices $\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$ of the polynomial system (14) were reshaped into the respective tensors $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n\times n\times n}$, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{n\times n\times n\times n}$ and $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{n\times n\times m}$, as demonstrated in Fig. 11(a). These tensors were then decomposed via e.g. CPD, Tucker or Tensor Train rank-1 SVD, resulting in a tensor approximation of (14) as

$$\begin{aligned}\dot{\boldsymbol{x}} =& \boldsymbol{A}\boldsymbol{x} + [\![\boldsymbol{B}^{(1)}, \boldsymbol{x}^T\boldsymbol{B}^{(2)}, \boldsymbol{x}^T\boldsymbol{B}^{(3)}]\!] \\ &+ [\![\boldsymbol{C}^{(1)}, \boldsymbol{x}^T\boldsymbol{C}^{(2)}, \boldsymbol{x}^T\boldsymbol{C}^{(3)}, \boldsymbol{x}^T\boldsymbol{C}^{(4)}]\!] \\ &+ [\![\boldsymbol{D}^{(1)}, \boldsymbol{x}^T\boldsymbol{D}^{(2)}, \boldsymbol{u}^T\boldsymbol{D}^{(3)}]\!] + \boldsymbol{E}\boldsymbol{u}, \qquad (16)\end{aligned}$$

TABLE IV
COMPUTATION AND STORAGE COMPLEXITIES OF DIFFERENT NONLINEAR MODEL ORDER REDUCTION APPROACHES ON A q-STATE REDUCED SYSTEM WITH dTH-ORDER NONLINEARITY.

Reduction methods	Function evaluation cost	Jacobian matrix evaluation cost	Storage cost
Traditional matrix-based method [36]–[40]	$O(q^{d+1})$	$O(q^{d+2})$	$O(q^{d+1})$
Tensor-based method [109]	$O(qdr)$	$O(q^2dr)$	$O(qdr)$
Symmetric tensor-based method [110]	$O(qr)$	$O(q^2r)$	$O(qr)$

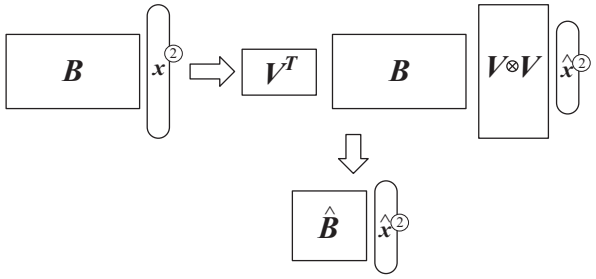


Fig. 10. Traditional projection-based nonlinear model order reduction methods reduce a large system matrix $\boldsymbol{B}$ to a small but dense matrix $\hat{\boldsymbol{B}}$ through an orthogonal projection matrix $\boldsymbol{V}$.

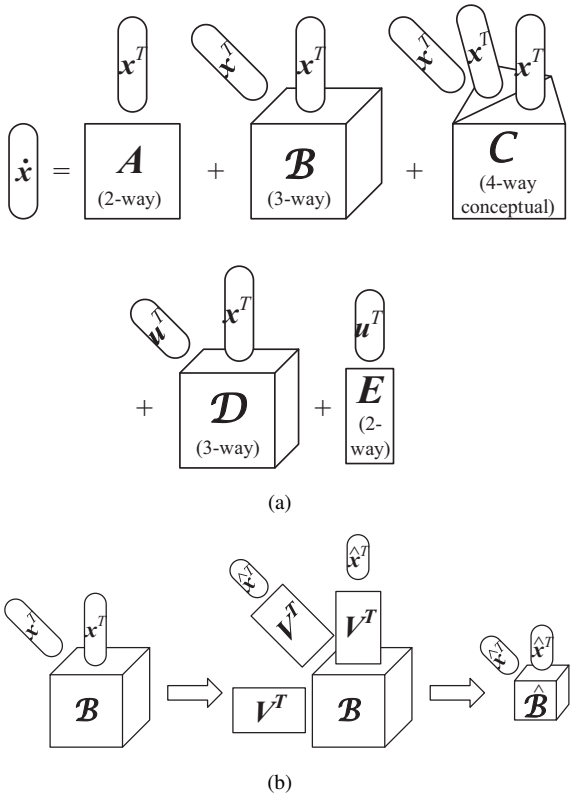


Fig. 11. Tensor structures used in [109]. (a) Tensor representation of the original nonlinear system in (14); (b) tensor $\boldsymbol{\mathcal{B}}$ is reduced to a compact tensor $\hat{\boldsymbol{\mathcal{B}}}$ with a projection matrix $\boldsymbol{V}$ in [109].

where $\boldsymbol{B}^{(k)}, \boldsymbol{C}^{(k)}, \boldsymbol{D}^{(k)}$, denote the kth-mode factor matrix from the polyadic decomposition of the tensors $\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{D}}$ respectively. Consequently, the reduced-order model inherits the same tensor structure as (16) (with smaller sizes of the mode factors). If we take tensor $\boldsymbol{\mathcal{B}}$ as an example, its reduction process in [109] is shown in Fig. 11(b).

Computational and Storage Benefits. Unlike previous matrix-based approaches, simulation of the tensor-structure reduced model completely avoids the overhead of solving high-order dense system matrices, since the dense Kronecker products in (14) are resolved by matrix-vector multiplications between the mode factor matrices and the state vectors. Therefore, substantial improvement on efficiency can be achieved. Meanwhile, these mode factor matrices can significantly reduce the memory requirement since they replace all dense tensors and can be reduced and stored beforehand. Table IV shows the computational complexities of function and Jacobian matrix evaluations when simulating a reduced model with dth-order nonlinearity, where r denotes the tensor rank used in the polyadic decompositions in [109]. The storage costs of those methods are also listed in the last column of Table IV.

Symmetric Tensor-based Nonlinear Model Order Reduction. A symmetric tensor-based order reduction method in [110] further utilizes the all-but-first-mode partial symmetry of the system tensor $\boldsymbol{\mathcal{B}}$ ($\boldsymbol{\mathcal{C}}$), i.e., the mode factors of $\boldsymbol{\mathcal{B}}$ ($\boldsymbol{\mathcal{C}}$) are exactly the same, except for the first mode only. This partial symmetry property is also kept by its reduced-order model. The symmetric tensor-based reduction method in [110] provides further improvements of computation performance and storage requirement over [109], as shown in the last row of Table IV.

B. Volterra-Based Simulation and Identification for ICs

Volterra theory has long been used in analyzing communication systems and in nonlinear control [111], [112]. It can be regarded as a kind of Taylor series with memory effects since its evaluation at a particular time point requires input information from the past. Given a certain input and a black-box model of a nonlinear system with time/frequency-domain Volterra kernels, the output response can be computed by the summation of a series of multidimensional convolutions. For instance, a 3rd-order response can be written in a discretized form as

$$y_3[k] = \sum_{m_1=1}^{M} \sum_{m_2=1}^{M} \sum_{m_3=1}^{M} h_3[m_1, m_2, m_3] \prod_{i=1}^{3} u[k - m_i], \tag{17}$$

where h_3 denotes the 3rd-order Volterra kernel, u is the discretized input and M is the memory. Such a multidimensional convolution is usually done by multidimensional fast Fourier transforms (FFT) and inverse fast Fourier transforms (IFFT). Although the formulation does not preclude itself from modeling strong nonlinearities, the exponential complexity growth in multidimensional FFT/IFFT computations results in the curse of dimensionality that forbids its practical implementation.

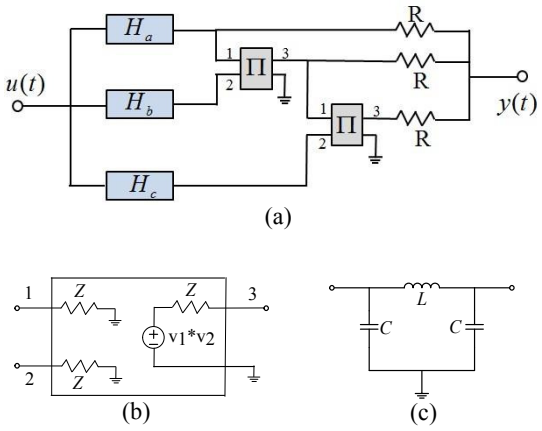


Fig. 12. (a) System diagram of a 3rd-order mixer circuit. The symbol Π denotes a mixer; (b) the equivalent circuit of the mixer. $Z = R = 50\,\Omega$; (c) the circuit schematic diagram of the low-pass filters H_a, H_b and H_c, with $L = 42.52\,\text{nH}$ and $C = 8.5\,\text{pF}$.

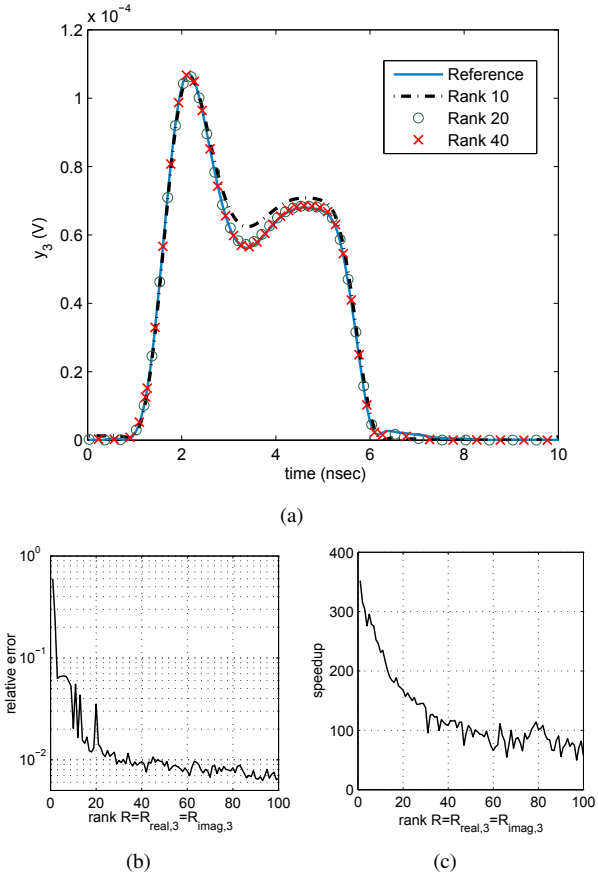


Fig. 13. Numerical results of the mixer. (a) Time-domain results of y_3 computed by the method in [113] with different rank approximations; (b) relative errors of [113] with different ranks; (c) speedups brought by [113] with different ranks.

Tensor-Volterra Model-based Simulation. Obviously, the 3rd-order Volterra kernel h_3 itself can be viewed as a 3-way tensor. By compressing the Volterra kernel into a polyadic decomposition, it is proven in [113] that the computationally expensive multidimensional FFT/IFFT can be replaced by a number of cheap one-dimensional FFT/IFFTs without compromising much accuracy.

Computational and Storage Benefits. The chosen rank for the polyadic decomposition has a significant impact on both the accuracy and efficiency of the simulation algorithm. In [113], the ranks were chosen a priori and it was demonstrated that the computational complexity for the tensor-Volterra based method to calculate an dth-order response is in $O((R_{\text{real}} + R_{\text{imag}})dm \log m)$, where m is the number of steps in the time/frequency axis, and R_{real} and R_{imag} denote the prescribed ranks of the polyadic decomposition used for the real and imaginary parts of the Volterra kernel, respectively. In contrast, the complexity for the traditional multidimensional FFT/IFFT approach is in $O(dm^d \log m)$. In addition, the tensor-Volterra model requires the storage of only the factor matrices in memory, with space complexity $O((R_{\text{real}} + R_{\text{imag}})dm)$, while $O(m^d)$ memory is required for the conventional approach.

In [113], the method was applied to compute the time-domain response of a 3rd-order mixer system shown in Fig. 12. The 3rd-order response y_3 is simulated to a square pulse input with $m = 201$ time steps. As shown in Fig. 13(a), a rank-20 (or above) polyadic decomposition for both the real and imaginary parts of the kernel tensor matched the reference result from multidimensional FFT/IFFT fairly well. Figs. 13(b) and (c) demonstrate a certain trade-off between the accuracy and efficiency when using different ranks for the polyadic decomposition. Nonetheless, a 60x speedup is still achievable for ranks around 100 with a 0.6% error.

System Identification. In [114]–[116], similar tensor-Volterra models were used to identify the black-box Volterra kernels h_i. It was reported in [114]–[116] that given certain input and output data, identification of the kernels in the polyadic decomposition form could significantly reduce the parametric complexity with good accuracy.

VII. Future Topics: EDA Applications

This section describes some EDA problems that could be potentially solved with, or that could benefit significantly from employing tensors. Since many EDA problems are characterized by high dimensionality, the potential application of tensors in EDA can be vast and is definitely not limited to the topics summarized below.

A. EDA Optimization with Tensors

Many EDA problems require solving a large-scale optimization problem in the following form:

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}), \quad \text{s. t.} \quad \boldsymbol{x} \in \mathcal{C} \tag{18}$$

where $\boldsymbol{x} = [x_1, \cdots, x_n]$ denotes n design or decision variables, $f(\boldsymbol{x})$ is a cost function (e.g., power consumption of a chip, layout area, signal delay), and $\mathcal{C}$ is a feasible set specifying some design constraints. This formulation can describe problems such as circuit optimization [64]–[66], placement [59], routing [60], and power management [117]. The optimization problem (18) is computationally expensive if $\boldsymbol{x}$ has many elements.

It is possible to accelerate the above large-scale optimization problems by exploiting tensors. By adding some extra variables $\hat{\boldsymbol{x}}$ with $\hat{n}$ elements, one could form a longer vector $\bar{\boldsymbol{x}} = [\boldsymbol{x}, \hat{\boldsymbol{x}}]$ such that $\bar{\boldsymbol{x}}$ has $n_1 \times \cdots \times n_d$ variables in total. Let $\bar{\boldsymbol{\mathcal{X}}}$ be a tensor such that $\bar{\boldsymbol{x}} = \text{vec}(\bar{\boldsymbol{\mathcal{X}}})$, let $\boldsymbol{x} = \boldsymbol{Q}\bar{\boldsymbol{x}}$ with $\boldsymbol{Q}$ being the first n rows of an identity matrix, then (18) can be written in the following tensor format:

$$\min_{\bar{\boldsymbol{\mathcal{X}}}} \ \bar{f}(\bar{\boldsymbol{\mathcal{X}}}), \quad \text{s. t.} \quad \bar{\boldsymbol{\mathcal{X}}} \in \bar{\mathcal{C}} \tag{19}$$

with $\bar{f}(\bar{\boldsymbol{\mathcal{X}}}) = f(\boldsymbol{Q}\text{vec}(\bar{\boldsymbol{\mathcal{X}}}))$ and $\bar{\mathcal{C}} = \{\bar{\boldsymbol{\mathcal{X}}} | \boldsymbol{Q}\text{vec}(\bar{\boldsymbol{\mathcal{X}}}) \in \mathcal{C}\}$.

Although problem (19) has more unknown variables than (18), the low-rank representation of tensor $\bar{\boldsymbol{\mathcal{X}}}$ may have much fewer unknown elements. Therefore, it is highly possible that solving (19) will require much lower computational cost for lots of applications.

B. High-Dimensional Modeling and Simulation

Consider a general algebraic equation resulting from a high-dimensional modeling or simulation problem

$$\boldsymbol{g}(\boldsymbol{x}) = 0, \quad \text{with } \boldsymbol{x} \in \mathbb{R}^N \text{ and } N = n^d \tag{20}$$

which can be solved by Newton's iteration. When an iterative linear equation solver is applied inside a Newton's iteration, it is possible to solve this problem at the complexity of $O(N) = O(n^d)$. However, since N is an exponential function of n, the iterative matrix solver quickly becomes inefficient as d increases. Instead, we rewrite (20) as the following equivalent optimization problem:

$$\min_{\boldsymbol{x}} \ f(\boldsymbol{x}) = \|\boldsymbol{g}(\boldsymbol{x})\|_2^2, \quad \text{s. t.} \quad \boldsymbol{x} \in \mathbb{R}^N.$$

This least-square optimization is a special case of (18), and thus the tensor-based optimization idea may be exploited to solve the above problem at the cost of $O(n)$.

A potential application lies in the PDE or integral equation solvers for device simulation. Examples include the Maxwell equations for parasitic extraction [31]–[33], the Navier-Stokes equation describing bio-MEMS [118], and the Poisson equation describing heating effects [119]. These problems can be described as (20) after numerical discretization. The tensor representation of $\boldsymbol{x}$ can be easily obtained based on the numerical discretization scheme. For instance, on a regular 3-D cubic structure, a finite-difference or finite-element discretization may use n_x, n_y and n_z discretization elements in the x, y and z directions respectively. Consequently, $\boldsymbol{x}$ could be compactly represented as a 3-way tensor with size $n_x \times n_y \times n_z$ to exploit its low-rank property in the spatial domain.

This idea can also be exploited to simulate multi-rate circuits or multi-tone RF circuits. In both cases, the tensor representation of $\boldsymbol{x}$ can be naturally obtained based on the time-domain discretization or multi-dimensional Fourier transform. In multi-tone harmonic balance [68], [69], the dimensionality d is the total number of RF inputs. In the multi-time PDE solver [67], d is the number of time axes describing different time scales.

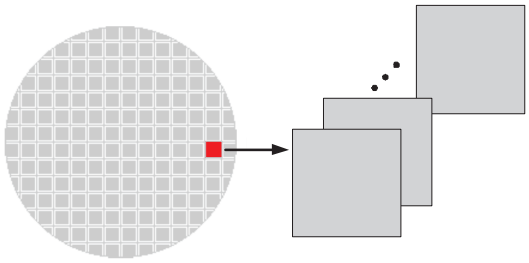


Fig. 14. Represent multiple testing chips on a wafer as a single tensor. Each slice of the tensor captures the spatial variations on a single die.

C. Process Variation Modeling

In order to characterize the inter-die and intra-die process variations across a silicon wafer with k dice, one may need to measure each die with an $m \times n$ array of devices or circuits [27]–[29]. The variations of a certain parameter (e.g., transistor threshold voltage) on the ith die can be described by matrix $\boldsymbol{A}_i \in \mathbb{R}^{m \times n}$, and thus one could describe the whole-wafer variation by stacking all matrices into a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{k \times m \times n}$, with $\boldsymbol{A}_i$ being the ith slice. This representation is graphically shown in Fig. 14.

Instead of measuring each device on each die (which requires kmn measurements in total), one could measure only a few devices on each wafer, then estimate the full-wafer variations using tensor completion. One may employ convex optimization to locate the globally optimal solution of this 3-way tensor completion problem.

VIII. Future Topics: Theoretical Challenges

Tensor theory is by itself an active research topic. This section summarizes some theoretical open problems.

A. Challenges in Tensor Decomposition

Polyadic and tensor train decompositions are preferred for high-order tensors due to their better scalability. In spite of their better computational scalability, the following challenges still exist:

- **Rank Determination in CPD.** The tensor ranks are usually determined by two methods. First, one may fix the rank and search for the tensor factors. Second, one may increase the rank incrementally to achieve an acceptable accuracy. Neither methods are optimal in the theoretical sense.
- **Optimization in Polyadic Decomposition.** Most rank-r polyadic decomposition algorithms employ alternating least-squares (ALS) to solve non-convex optimization problems. Such schemes do not guarantee the global optimum, and thus it is highly desirable to develop global optimization algorithms for the CPD.
- **Faster Tensor Train Decomposition.** Computing the tensor train decomposition requires the computation of many low-rank decompositions. The state-of-the-art implementation employs "cross approximation" to perform low-rank approximations [87], but it still needs too many iterations to find a "good" representation.

- **Preserving Tensor Structures and/or Properties.** In some cases, the given tensor may have some special properties such as symmetry or non-negativeness. These properties need to be preserved in their decomposed forms for specific applications.

B. Challenges in Tensor Completion

Major challenges of tensor completion include:

- **Automatic Rank Determination.** In high-dimensional tensor completion, it is important to determine the tensor rank automatically. Although some probabilistic approaches such as variational Bayesian methods [96], [97] have been reported, they are generally not robust for very high-order tensors.
- **Convex Tensor Completion.** Most tensor completion problems are formulated as non-convex optimization problems. Nuclear-norm minimization is convex, but it is only applicable to low-order tensors. Developing a scalable convex formulation for the minimal-rank completion still remains an open problem for high-order cases.
- **Robust Tensor Completion.** In practical tensor completion, the available tensor elements from measurement or simulations can be noisy or even wrong. For these problems, the developed tensor completion algorithms should be robust against noisy input.
- **Optimal Selection of Samples.** Two critical fundamental questions should be addressed. First, how many samples are required to (faithfully) recover a tensor? Second, how can we select the samples optimally?

IX. Conclusion

By exploiting low-rank and other properties of tensors (e.g., sparsity, symmetry), the storage and computational cost of many challenging EDA problems can be significantly reduced. For instance, in the high-dimensional stochastic collocation modeling of a CMOS ring oscillator, exploiting tensor completion required only a few hundred circuit/device simulation samples vs. the huge number of simulations (e.g., 10^{27}) required by standard approaches to build a stochastic model of similar accuracy. When applied to hierarchical uncertainty quantification, a tensor-train approach allowed the easy handling of an extremely challenging MEMS/IC co-design problem with over 180 uncorrelated random parameters describing process variations. In nonlinear model order reduction, the high-order nonlinear terms were easily approximated by a tensor-based projection framework. Finally, a $60\times$ speedup was observed when using tensor computation in a 3rd-order Volterra-series nonlinear modeling example, while maintaining a 0.6% relative error compared with the conventional FFT/IFFT approach. These are just few initial representative examples for the huge potential that a tensor computation framework can offered to EDA algorithms. We believe that the space of EDA applications that could benefit from the use of tensors is vast and remains mainly unexplored, ranging from EDA optimization problems, to device field solvers, and to process variation modeling.

Appendix A
Additional Notations and Definitions

Diagonal, Cubic and Symmetric Tensors. The diagonal entries of a tensor $\boldsymbol{\mathcal{A}}$ are the entries $a_{i_1 i_2 \cdots i_d}$ for which $i_1 = i_2 = \cdots = i_d$. A tensor $\boldsymbol{\mathcal{S}}$ is diagonal if all of its non-diagonal entries are zero. A cubical tensor is a tensor for which $n_1 = n_2 = \cdots = n_d$. A cubical tensor $\boldsymbol{\mathcal{A}}$ is symmetric if $a_{i_1 \cdots i_d} = a_{\pi(i_1,\ldots,i_d)}$ where $\pi(i_1, \ldots, i_d)$ is any permutation of the indices.

The **Kronecker product** [120] is denoted by $\otimes$. We use the notation $x^{\circledast} = x \otimes x \otimes \cdots \otimes x$ for the d-times repeated Kronecker product.

Definition 5: **Reshaping.** Reshaping, also called **unfolding**, is another often used tensor operation. The most common reshaping is the matricization, which reorders the entries of $\boldsymbol{\mathcal{A}}$ into a matrix. The mode-n matricization of a tensor $\boldsymbol{\mathcal{A}}$, denoted $\boldsymbol{\mathcal{A}}_{(n)}$, rearranges the entries of $\boldsymbol{\mathcal{A}}$ such that the rows of the resulting matrix are indexed by the nth tensor index i_n. The remaining indices are grouped in ascending order.

Example 2: The 3-way tensor of Fig. 1 can be reshaped as a 2×12 matrix or a 3×8 matrix, and so forth. The mode-1 and mode-3 unfoldings are

$$\boldsymbol{\mathcal{A}}_{(1)} = \begin{pmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{pmatrix},$$

$$\boldsymbol{\mathcal{A}}_{(3)} = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & \cdots & 21 & 22 & 23 & 24 \end{pmatrix}.$$

The column indices of $\boldsymbol{\mathcal{A}}_{(1)}, \boldsymbol{\mathcal{A}}_{(3)}$ are $[i_2 i_3]$ and $[i_1 i_2]$, respectively.

Definition 6: **Vectorization.** Another important reshaping is the vectorization. The vectorization of a tensor $\boldsymbol{\mathcal{A}}$, denoted $\text{vec}(\boldsymbol{\mathcal{A}})$, rearranges its entries in one vector.

Example 3: For the tensor in Fig. 1, we have

$$\text{vec}(\boldsymbol{\mathcal{A}}) = \begin{pmatrix} 1 & 2 & \cdots & 24 \end{pmatrix}^T.$$

Appendix B
Computation and Variants of the Polyadic Decomposition

Computing Polyadic Decompositions. Since the tensor rank is not known a priori, in practice, one usually computes a low-rank $r < R$ approximation of a given tensor $\boldsymbol{\mathcal{A}}$ by minimizing the Frobenius norm of the difference between $\boldsymbol{\mathcal{A}}$ and its approximation. Specifically, the user specifies r and then solves the minimization problem

$$\underset{\boldsymbol{\mathcal{D}}, \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}}{\text{argmin}} ||\boldsymbol{\mathcal{A}} - [\![\boldsymbol{\mathcal{D}}; \boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)}]\!]||_F$$

where $\boldsymbol{\mathcal{D}} \in \mathbb{R}^{r \times r \times \cdots \times r}, \boldsymbol{U}^{(i)} \in \mathbb{R}^{n_i \times r} (i = \{1, \ldots, d\})$. One can then increment r and compute new approximations until a "good enough" fit is obtained. A common method for solving this optimization problem is the Alternating Least Squares (ALS) method [82]. Other popular optimization algorithms are nonlinear conjugate gradient methods, quasi-Newton or nonlinear least squares (e.g. Levenberg-Marquardt) [121]. The computational complexity per iteration of the ALS, Levenberg-Marquardt (LM) and Enhanced Line Search (ELS) methods to

compute a polyadic decomposition of a 3-way tensor, where $n = \min(n_1, n_2, n_3)$, are given in Table V.

TABLE V
COMPUTATIONAL COSTS OF 3 TENSOR DECOMPOSITION METHODS FOR A 3-WAY TENSOR [122].

Methods	Cost per iteration
ALS	$(n_2n_3 + n_1n_3 + n_1n_2)(7n^2 + n) + 3nn_1n_2n_3$
LM	$n_1n_2n_3(n_1 + n_2 + n_3)^2n^2$
ELS	$(8n + 9)n_1n_2n_3$

Two variants of the polyadic decomposition are summarized below.

1) PARATREE or tensor-train rank-1 SVD (TTr1SVD): This polyadic decomposition [123], [124] consists of orthogonal rank-1 terms and is computed by consecutive reshapings and SVDs. This computation implies that the obtained decomposition does not need an initial guess and will be unique for a fixed order of indices. Similar to SVD in the matrix case, this decomposition has an approximation error easily expressed in terms of the σ_i's [123].

2) CPD for Symmetric Tensors: The CPD of a symmetric tensor does not in general result in a summation of symmetric rank-1 terms. In some applications, it is more meaningful to enforce the symmetric constraints explicitly, and write $\boldsymbol{\mathcal{A}} = \sum_{i=1}^{R} \lambda_i \boldsymbol{v}_i^d$, where $\lambda_i \in \mathbb{R}, \boldsymbol{\mathcal{A}}$ is a d-way symmetric tensor. Here $\boldsymbol{v}_i^d$ is a shorthand for the d-way outer product of a vector $\boldsymbol{v}_i$ with itself, i.e., $\boldsymbol{v}_i^d = \boldsymbol{v}_i \circ \boldsymbol{v}_i \circ \cdots \circ \boldsymbol{v}_i$.

APPENDIX C
HIGHER-ORDER SVD

The Higher-Order SVD (HOSVD) [125] is obtained from the Tucker decomposition when the factor matrices $\boldsymbol{U}^{(i)}$ are orthogonal, when any two slices of the core tensor $\boldsymbol{\mathcal{S}}$ in the same mode are orthogonal, $\langle \boldsymbol{\mathcal{S}}_{i_k=p}, \boldsymbol{\mathcal{S}}_{i_k=q} \rangle = 0$ if $p \neq q$ for any $k = 1, \ldots, d$, and when the slices of the core tensor $\boldsymbol{\mathcal{S}}$ in the same mode are ordered according to their Frobenius norm, $||\boldsymbol{\mathcal{S}}_{i_k=1}|| \geq ||\boldsymbol{\mathcal{S}}_{i_k=2}|| \geq \cdots \geq ||\boldsymbol{\mathcal{S}}_{i_k=n_k}||$ for $k = \{1, \ldots, d\}$. Its computation consists of d SVDs to compute the factor matrices and a contraction of their inverses with the original tensor to compute the HOSVD core tensor. For a 3-way tensor this entails a computational cost of $2n_1n_2n_3(n_1+n_2+n_3)+5(n_1^2n_2n_3+n_1n_2^2n_3+n_1n_2n_3^2)2(n_1^3+n_2^3+n_3^3)/3(n_1^3+n_2^3+n_3^3)/3$ [122].

REFERENCES

[1] L. Nagel and D. O. Pederson, "SPICE (Simulation Program with Integrated Circuit Emphasis)," University of California, Berkeley, Tech. Rep., April 1973.

[2] C.-W. Ho, R. Ruehli, and P. Brennan, "The modified nodal approach to network analysis," *IEEE Trans. Circuits Syst.*, vol. 22, no. 6, pp. 504–509, June 1975.

[3] K. Kundert, J. K. White, and A. Sangiovanni-Vincenteli, *Steady-state methods for simulation analog and microwave circuits*. Kluwer Academic Publishers, Boston, 1990.

[4] T. Aprille and T. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *IEEE Proc.*, vol. 60, no. 1, pp. 108–114, Jan. 1972.

[5] ——, "A computer algorithm to determine the steady-state response of nonlinear oscillators," *IEEE Trans. Circuit Theory*, vol. CT-19, no. 4, pp. 354–360, July 1972.

[6] K. Kundert, J. White, and A. Sangiovanni-Vincentelli, "An envelope-following method for the efficient transient simulation of switching power and filter circuits," in *Proc. Int. Conf. Computer-Aided Design*, 1988 Nov.

[7] L. Petzold, "An efficient numerical method for highly oscillatory ordinary differential equations," *SIAM J. Numer. Anal.*, vol. 18, no. 3, pp. 455–479, June 1981.

[8] A. Demir, A. Mehrotra, and J. Roychowdhury, "Phase noise in oscillators: A unifying theory and numerical methods for characterization," *IEEE Trans. Circuits Syst. I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 655–674, 2000.

[9] J. N. Kozhaya, S. R. Nassif, and F. N. Najm, "A multigrid-like technique for power grid analysis," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 21, no. 10, pp. 1148–1160, 2002.

[10] T.-H. Chen and C. C.-P. Chen, "Efficient large-scale power grid analysis based on preconditioned Krylov-subspace iterative methods," in *Proc. Design Automation Conf.*, 2001, pp. 559–562.

[11] Z. Feng and P. Li, "Multigrid on GPU: tackling power grid analysis on parallel SIMT platforms," in *Proc. Intl. Conf. Computer-Aided Design*, 2008, pp. 647–654.

[12] R. Telichevesky and J. K. White, "Efficient steady-state analysis based on matrix-free Krylov-subpsace methods," in *Proc. Design Automation Conf.*, June 1995, pp. 480–484.

[13] X. Liu, H. Yu, and S. Tan, "A GPU-accelerated parallel shooting algorithm for analysis of radio frequency and microwave integrated circuits," *IEEE Trans. VLSI*, vol. 23, no. 3, pp. 480–492, 2015.

[14] S. Weinzierl, "Introduction to Monte Carlo methods," theory Group, The Netherlands, Tech. Rep. NIKHEF-00-012, 2000.

[15] A. Singhee and R. A. Rutenbar, "Why Quasi-Monte Carlo is better than Monte Carlo or Latin hypercube sampling for statistical circuit analysis," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 29, no. 11, pp. 1763–1776, 2010.

[16] Z. Zhang, X. Yang, G. Marucci, P. Maffezzoni, I. M. Elfadel, G. Karniadakis, and L. Daniel, "Stochastic testing simulator for integrated circuits and MEMS: Hierarchical and sparse techniques," in *Proc. Custom Integr. Circuits Conf.* San Jose, CA, Sept. 2014, pp. 1–8.

[17] Z. Zhang, I. A. M. Elfadel, and L. Daniel, "Uncertainty quantification for integrated circuits: Stochastic spectral methods," in *Proc. Int. Cont. Computer-Aided Design*. San Jose, CA, Nov 2013, pp. 803–810.

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

[19] 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," *Optics Express*, vol. 23, no. 4, pp. 4242 – 4254, Feb 2015.

[20] 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. CAD Integr. Circuits Syst.*, vol. 32, no. 10, Oct. 2013.

[21] 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. II: Exp. Briefs*, vol. 60, no. 10, Oct. 2013.

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

[23] J. Wang, P. Ghanta, and S. Vrudhula, "Stochastic analysis of interconnect performance in the presence of process variations," in *Proc. Design Auto Conf.*, 2004, pp. 880–886.

[24] S. Vrudhula, J. M. Wang, and P. Ghanta, "Hermite polynomial based interconnect analysis in the presence of process variations," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2001–2011, Oct. 2006.

[25] M. Rufuie, E. Gad, M. Nakhla, R. Achar, and M. Farhan, "Fast variability analysis of general nonlinear circuits using decoupled polynomial chaos," in *Workshop Signal and Power Integrity*, May 2014, pp. 1–4.

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

[27] D. S. Boning, K. Balakrishnan, H. Cai, N. Drego, A. Farahanchi, K. M. Gettings, D. Lim, A. Somani, H. Taylor, D. Truque, and X. Xie, "Variation," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 63–71, Feb. 2008.

[28] L. Yu, S. Saxena, C. Hess, A. Elfadel, D. Antoniadis, and D. Boning, "Remembrance of transistors past: Compact model parameter extraction using Bayesian inference and incomplete new measurements," in *Proc. Design Automation Conf*, 2014, pp. 1–6.

[29] W. Zhang, X. Li, F. Liu, E. Acar, R. A. Rutenbar, and R. D. Blanton, "Virtual probe: A statistical framework for low-cost silicon characterization of nanoscale integrated circuits," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1814–1827, 2011.

[30] Y. S. Chauhan, S. Venugopalan, M. A. Karim, S. Khandelwal, N. Paydavosi, P. Thakur, A. M. Niknejad, and C. C. Hu, "BSIM–industry standard compact MOSFET models," in *Proc. ESSCIRC*, 2012, pp. 30–33.

[31] K. Nabors and J. White, "FastCap: a multipole accelerated 3-D capacitance extraction program," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 10, no. 1, pp. 1447–1459, Nov 1991.

[32] M. Kamon, M. J. Tsuk, and J. K. White, "FASTHENRY: a multipole-accelerated 3-D inductance extraction program," *IEEE Trans. Microw. Theory Tech.*, vol. 42, no. 9, pp. 1750–1758, Sept. 1994.

[33] J. Phillips and J. K. White, "A precorrected-FFT method for electrostatic analysis of complicated 3-D structures," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 16, no. 10, pp. 1059–1072, Oct 1997.

[34] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 17, no. 8, pp. 645–654, Aug. 1998.

[35] J. R. Phillips, L. Daniel, and L. M. Silveira, "Guaranteed passive balancing transformations for model order reduction," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 22, no. 8, pp. 1027–1041, Aug. 2003.

[36] J. Roychowdhury, "Reduced-order modeling of time-varying systems," *IEEE Trans. Circuits and Syst. II: Analog and Digital Signal Process.*, vol. 46, no. 10, pp. 1273–1288, Oct 1999.

[37] P. Li and L. Pileggi, "Compact reduced-order modeling of weakly nonlinear analog and RF circuits," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 24, no. 2, pp. 184–203, Feb. 2005.

[38] J. R. Phillips, "Projection-based approaches for model reduction of weakly nonlinear time-varying systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 171–187, Feb. 2003.

[39] C. Gu, "QLMOR: a projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1307–1320, Sep. 2011.

[40] Y. Zhang, H. Liu, Q. Wang, N. Fong, and N. Wong, "Fast nonlinear model order reduction via associated transforms of high-order Volterra transfer functions," in *Proc. Design Autom. Conf.*, Jun. 2012, pp. 289–294.

[41] B. N. Bond and L. Daniel, "Stable reduced models for nonlinear descriptor systems through piecewise-linear approximation and projection," *IEEE Trans. CAD of Integr. Circuits and Syst.*, vol. 28, no. 10, pp. 1467–1480, 2009.

[42] M. Rewienski and J. White, "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 2, pp. 155–170, Feb. 2003.

[43] B. Gustavsen and S. Semlyen, "Rational approximation of frequency domain responses by vector fitting," *IEEE Trans. Power Delivery*, vol. 14, no. 3, p. 10521061, Aug.

[44] S. Grivet-Talocia, "Passivity enforcement via perturbation of Hamiltonian matrices," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 1755–1769, Sept.

[45] C. P. Coelho, J. Phillips, and L. M. Silveira, "A convex programming approach for generating guaranteed passive approximations to tabulated frequency-data," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 23, no. 2, pp. 293–301, Feb. 2004.

[46] B. N. Bond, Z. Mahmood, Y. Li, R. Sredojevic, A. Megretski, V. Stojanovic, Y. Avniel, and L. Daniel, "Compact modeling of nonlinear analog circuits using system identification via semidefinite programing and incremental stability certification," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 29, no. 8, p. 11491162, Aug.

[47] L. Daniel, C. S. Ong, S. C. Low, K. H. Lee, and J. White, "A multiparameter moment-matching model-reduction approach for generating geometrically parameterized interconnect performance models," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 23, no. 5, pp. 678–693, May 2004.

[48] ——, "Geometrically parameterized interconnect performance models for interconnect synthesis," in *Proc. IEEE/ACM Intl. Symp. Physical Design*, May 2002, pp. 202–207.

[49] K. C. Sou, A. Megretski, and L. Daniel, "A quasi-convex optimization approach to parameterized model order reduction," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 27, no. 3, pp. 456–469, March 2008.

[50] B. N. Bond and L. Daniel, "Parameterized model order reduction of nonlinear dynamical systems," in *Proc. Intl. Conf. Computer Aided Design*, Nov. 2005, pp. 487–494.

[51] ——, "A piecewise-linear moment-matching approach to parameterized model-order reduction for highly nonlinear systems," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 26, no. 12, pp. 2116–2129, 2007.

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

[53] F. Ferranti, L. Knockaert, and T. Dhaene, "Guaranteed passive parameterized admittance-based macromodeling," *IEEE Trans. Advanced Packag.*, vol. 33, no. 3, pp. 623–629, 2010.

[54] J. F. Villena and L. M. Silveira, "SPARE–a scalable algorithm for passive, structure preserving, parameter-aware model order reduction," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 29, no. 6, pp. 925–938, 2010.

[55] L. M. Silveira and J. R. Phillips, "Resampling plans for sample point selection in multipoint model-order reduction," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2775–2783, 2006.

[56] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[57] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.

[58] D. P. Bertsekas, *Nonlinear programming*. Athena Scientific, 1999.

[59] K. Shahookar and P. Mazumder, "VLSI cell placement techniques," *ACM Comput. Surveys*, vol. 23, no. 2, pp. 143–220, 1991.

[60] J. Cong, L. He, C.-K. Koh, and P. H. Madden, "Performance optimization of VLSI interconnect layout," *Integration, the VLSI Journal*, vol. 21, no. 1, pp. 1–94, 1996.

[61] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[62] G. Gielen, H. Walscharts, and W. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE J. Solid-State Circuits*, vol. 25, no. 3, pp. 707–713, 1990.

[63] W. Cai, X. Zhou, and X. Cui, "Optimization of a GPU implementation of multi-dimensional RF pulse design algorithm," in *Bioinformatics and Biomedical Engineering, IEEE Intl. Conf. on*, 2011, pp. 1–4.

[64] M. Hershenson, S. P. Boyd, and T. H. Lee, "Optimal design of a CMOS op-amp via geometric programming," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 20, no. 1, pp. 1–21, 2001.

[65] X. Li, P. Gopalakrishnan, Y. Xu, and T. Pileggi, "Robust analog/RF circuit design with projection-based posynomial modeling," in *Proc. Intl. Conf. Computer-aided design*, 2004, pp. 855–862.

[66] Y. Xu, K.-L. Hsiung, X. Li, I. Nausieda, S. Boyd, and L. Pileggi, "OPERA: optimization with ellipsoidal uncertainty for robust analog IC design," in *Proc. Design Autom. Conf.*, 2005, pp. 632–637.

[67] J. Roychowdhury, "Analyzing circuits with widely separated time scales using numerical PDE methods," *IEEE Trans. Circuits Syst.: Fundamental Theory and Applications*, vol. 48, no. 5, pp. 578–594, May 2001.

[68] R. C. Melville, P. Feldmann, and J. Roychowdhury, "Efficient multi-tone distortion analysis of analog integrated circuits," in *Proc. Custom Integr. Circuits Conf.*, May 1995, pp. 241–244.

[69] N. B. De Carvalho and J. C. Pedro, "Multitone frequency-domain simulation of nonlinear circuits in large- and small-signal regimes," *IEEE Trans. Microwave Theory and Techniques*, vol. 46, no. 12, pp. 2016–2024, Dec 1998.

[70] M. Bonnin and F. Corinto, "Phase noise and noise induced frequency shift in stochastic nonlinear oscillators," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 60, no. 8, pp. 2104–2115, 2013.

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

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

[73] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, March 2015.

[74] N. Vervliet, O. Debals, L. Sorber, and L. D. Lathauwer, "Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 71–79, Sep. 2014.

[75] B. W. Bader, T. G. Kolda *et al.*, "MATLAB Tensor Toolbox Version 2.6," February 2015. [Online]. Available: http://www.sandia.gov/~tgkolda/TensorToolbox/

[76] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer. (2016, Mar.) Tensorlab 3.0. [Online]. Available: http://www.tensorlab.net

[77] I. Oseledets, S. Dolgov, V. Kazeev, O. Lebedeva, and T. Mach. (2012) TT-Toolbox 2.2. [Online]. Available: http://spring.inm.ras.ru/osel/download/tt22.zip

[78] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015.

[79] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[80] M. Rakhuba and I. V. Oseledets, "Fast multidimensional convolution in low-rank tensor formats via cross approximation," *SIAM Journal on Scientific Computing*, vol. 37, no. 2, pp. A565–A582, 2015.

[81] 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.

[82] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, no. 1, p. 84, 1970.

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

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

[85] J. Håstad, "Tensor rank is NP-complete," *J. Algorithms*, vol. 11, no. 4, pp. 644–654, 1990.

[86] J. B. Kruskal, "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[87] I. Oseledets and E. Tyrtyshnikov, "TT-cross approximation for multidimensional arrays," *Linear Algebra and its Applications*, vol. 422, no. 1, pp. 70–88, 2010.

[88] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, p. 119, Jan. 2011.

[89] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Machine Intelligence*, vol. 35, no. 1, pp. 208–220, Jan. 2013.

[90] J. Douglas and H. Rachford, "On the numerical solution of heat conduction problems in two and three space variables," *Trans. American Math. Society*, vol. 82, no. 2, pp. 421–439, Jul. 1956.

[91] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite-element approximations," *Comp. Math. Appl.*, vol. 2, no. 1, pp. 17–40, Jan. 1976.

[92] D. Kressner, M. Steinlechner, and B. Vandereycken, "Low-rank tensor completion by Riemannian optimization," *BIT Numer. Math.*, vol. 54, no. 2, pp. 447–468, Jun. 2014.

[93] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

[94] S. Holtz, T. Rohwedder, and R. Schneider, "The alternating linear scheme for tensor optimization in the tensor train format," *SIAM J. Sci. Comput.*, 2012.

[95] P. Rai, Y. Wang, S. Guo, G. Chen, D. Dunson, and L. Carin, "Scalable Bayesian low-rank decomposition of incomplete multiway tensors," in *Proc. Int. Conf. Machine Learning*, 2014, pp. 1800–1809.

[96] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Trans. Pattern Anal. and Machine Intelligence*, vol. 37, no. 9, pp. 1751–1763, 2015.

[97] ——, "Bayesian sparse Tucker models for dimension reduction and tensor completion," *arXiv:1505.02343*, May 2015.

[98] Z. Zhang, T.-W. Weng, and L. Daniel, "A big-data approach to handle process variations: Uncertainty quantification by tensor recovery," in *Proc. Int. Workshop Signal and Power Integrity*, May 2016.

[99] ——, "A big-data approach to handle many process variations: tensor recovery and applications," *IEEE Trans. Comp., Packag. Manuf. Techn.*, submitted in 2016.

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

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

[102] I. Babuška, F. Nobile, and R. Tempone, "A stochastic collocation method for elliptic partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 45, no. 3, pp. 1005–1034, Mar 2007.

[103] F. Nobile, R. Tempone, and C. G. Webster, "A sparse grid stochastic collocation method for partial differential equations with random input data," *SIAM J. Numer. Anal.*, vol. 46, no. 5, pp. 2309–2345, May 2008.

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

[105] H.-J. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, vol. 13, pp. 147–269, 2004.

[106] T. Wang, H. Liu, Y. Wang, and N. Wong, "Weakly nonlinear circuit analysis based on fast multidimensional inverse Laplace transform," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2012, pp. 547–552.

[107] H. Liu and N. Wong, "Autonomous Volterra algorithm for steady-state analysis of nonlinear circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 6, pp. 858–868, Jun. 2013.

[108] Y. Zhang and N. Wong, "Compact model order reduction of weakly nonlinear systems by associated transform," *Intl. J. Circuit Theory and Applications*, 2015.

[109] H. Liu, L. Daniel, and N. Wong, "Model reduction and simulation of nonlinear circuits via tensor decomposition," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1059–1069, Jul. 2015.

[110] J. Deng, H. Liu, K. Batselier, Y. K. Kwok, and N. Wong, "STORM: a nonlinear model order reduction method via symmetric tensor decomposition," in *Proc. Asia and South Pacific Design Autom. Conf.*, Jan. 2016, pp. 557–562.

[111] E. Bedrosian and S. O. Rice, "The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs," *Proc. IEEE*, vol. 59, no. 12, pp. 1688–1707, Dec. 1971.

[112] W. Rugh, *Nonlinear System Theory – The Volterra-Wiener Approach*. Baltimore, MD: Johns Hopkins Univ. Press, 1981.

[113] H. Liu, X. Xiong, K. Batselier, L. Jiang, L. Daniel, and N. Wong, "STAVES: Speedy tensor-aided volterra-based electronic simulator," in *Proc. Int. Computer-Aided Design*, Nov 2015, pp. 583–588.

[114] G. Favier, A. Y. Kibangou, and T. Bouilloc, "Nonlinear system modeling and identification using Volterra-PARAFAC models," *Int. J. Adapt. Control Signal Process*, vol. 26, no. 1, pp. 30–53, Jan. 2012.

[115] A. Khouaja and G. Favier, "Identification of PARAFAC-Volterra cubic models using an alternating recursive least squares algorithm," in *Proc. Europ. Signal Process. Conf.*, 2004, pp. 1903–1906.

[116] K. Batselier, Z. Chen, H. Liu, and N. Wong, "A tensor-based Volterra series black-box nonlinear system identification and simulation framework," in *Proc. Intl. Conf. Computer Aided Design*, 2016.

[117] L. Benini, A. Bogliolo, G. A. Paleologo, and G. De Micheli, "Policy optimization for dynamic power management," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 18, no. 6, pp. 813–833, 1999.

[118] D. Vasilyev, M. Rewienski, and J. White, "Macromodel generation for BioMEMS components using a stabilized balanced truncation plus trajectory piecewise-linear approach," *IEEE Trans. CAD of Integr. Circuits and Syst.*, vol. 25, no. 2, pp. 285–293, 2006.

[119] W. Yu, T. Zhang, X. Yuan, and H. Qian, "Fast 3-D thermal simulation for integrated circuits with domain decomposition method," *IEEE Trans. CAD of Integr. Circuits Syst.*, vol. 32, no. 12, pp. 2014–2018, 2013.

[120] C. F. V. Loan, "The ubiquitous Kronecker product," *J. Comp. Appl. Math.*, vol. 123, no. 1-2, pp. 85–100, Nov. 2000.

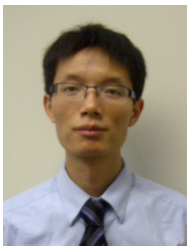
[121] L. Sorber, M. V. Barel, and L. D. Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(l_r, l_r, 1)$ terms, and a new generalization," *SIAM J. Optim.*, vol. 23, no. 2, pp. 695–720, 2013.

[122] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *J. Chemometrics*, vol. 23, no. 7-8, pp. 393–405, JUL-AUG 2009.

[123] K. Batselier, H. Liu, and N. Wong, "A constructive algorithm for decomposing a tensor into a finite sum of orthonormal rank-1 terms," *SIAM J. Matrix Anal. Appl.*, vol. 26, no. 3, pp. 1315–1337, Sep. 2015.

[124] J. Salmi, A. Richter, and V. Koivunen, "Sequential unfolding SVD for tensors with applications in array signal processing," *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4719–4733, Dec. 2009.

[125] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2000.



Zheng Zhang (M'15) received the Ph.D degree (2015) in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT), Cambridge, MA. Currently he is a Postdoc Associate with the Research Laboratory of Electronics at MIT. His research interests include uncertainty quantification, tensor and model order reduction, with diverse engineering applications including nanoelectronics, energy systems and biomedical imaging. His industrial experiences include Coventor Inc. and Maxim-IC; academic visiting experiences include UC San Diego, Brown University and Politechnico di Milano; government lab experiences include the Argonne National Laboratory.

Dr. Zhang received 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 Donald O. Pederson Best Paper Award from 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.

Luca Daniel (S'98-M'03) is a Full Professor in the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (MIT). He received the Ph.D. degree in Electrical Engineering from the University of California, Berkeley, in 2003. Industry experiences include HP Research Labs, Palo Alto (1998) and Cadence Berkeley Labs (2001).

Dr. Daniel current research interests include integral equation solvers, uncertainty quantification and parameterized model order reduction, applied to RF circuits, silicon photonics, MEMs, Magnetic Resonance Imaging scanners, and the human cardiovascular system.

Prof. Daniel was the recipient of the 1999 IEEE Trans. on Power Electronics best paper award; the 2003 best PhD 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 Trans. On Computer Aided Design best paper award; and seven best paper awards in conferences.

Kim Batselier (M'13) received the M.S. degree in Electro-Mechanical Engineering and the Ph.D. Degree in Engineering Science from the KULeuven, Belgium, in 2005 and 2013 respectively. He worked as a research engineer at BioRICS on automated performance monitoring until 2009. He is currently a Post-Doctoral Research Fellow at The University of Hong Kong since 2013. His current research interests include linear and nonlinear system theory/identification, algebraic geometry, tensors, and numerical algorithms.

Haotian Liu (S'11) received the B.S. degree in Microelectronic Engineering from Tsinghua University, Beijing, China, in 2010, and the Ph.D. degree in Electronic Engineering from the University of Hong Kong, Hong Kong, in 2014. He is currently a software engineer with Cadence Design Systems, Inc. San Jose, CA.

In 2014, Dr. Liu was a visiting scholar with the Massachusetts Institute of Technology (MIT), Cambridge, MA. His research interests include numerical simulation methods for very-large-scale integration (VLSI) circuits, model order reduction, parallel computation and static timing analysis.

Ngai Wong (S'98-M'02) received his B.Eng. and Ph.D. degrees in Electrical and Electronic Engineering from The University of Hong Kong, Hong Kong, in 1999 and 2003, respectively.

Dr. Wong was a visiting scholar with Purdue University, West Lafayette, IN, in 2003. He is currently an Associate Professor with the Department of Electrical and Electronic Engineering, The University of Hong Kong. His current research interests include linear and nonlinear circuit modeling and simulation, model order reduction, passivity test and enforcement, and tensor-based numerical algorithms in electronic design automation (EDA).