STORM: A Nonlinear Model Order Reduction Method via Symmetric Tensor Decomposition

Jian Deng, Haotian Liu, Kim Batselier, Yu-Kwong Kwok and Ngai Wong
Department of Electrical and Electronic Engineering
The University of Hong Kong
{dengjian,htliu,kimb,ykwok,nwong}@eee.hku.hk

Abstract—Nonlinear model order reduction has always been a challenging but important task in various science and engineering fields. In this paper, a novel symmetric tensor-based order-reduction method (STORM) is presented for simulating large-scale nonlinear systems. The multidimensional data structure of symmetric tensors, as the higher order generalization of symmetric matrices, is utilized for the effective capture of high-order nonlinearities and efficient generation of compact models. Compared to the recent tensor-based nonlinear model order reduction (TNMOR) algorithm [1], STORM shows advantages in two aspects. First, STORM avoids the assumption of the existence of a low-rank tensor approximation. Second, with the use of the symmetric tensor decomposition, STORM allows significantly faster computation and less storage complexity than TNMOR. Numerical experiments demonstrate the superior computational efficiency and accuracy of STORM against existing nonlinear model order reduction methods.

I. INTRODUCTION

To facilitate efficient modeling and simulation of large-scale dynamical systems, model order reduction has been intensively studied in the past decades [2]. The principle of model order reduction is to extract a compact reduced-order model (ROM) which approximates the input-output relationship of the original system accurately, so that efficient simulation and reliable system-level verification can be achieved. The effectiveness and robustness of model order reduction have been proved in various domains, e.g., circuit design [3]–[5], mechanical systems [6] and chemical engineering [7].

Since model order reduction for linear time-invariant (LTI) systems is by now mature, large amounts of research effort are being devoted to nonlinear cases, which are also more practical for real-world problems [8]. For instance, simulations of radio-frequency integrated circuits (RFIC) are highly time-consuming due to their intrinsic nonlinearities and large problem sizes [9]. However, nonlinear model order reduction is much more challenging: the behaviors of nonlinear dynamical systems are much more complex than linear systems, and there is a lack of a universal and efficient way to extract the nonlinear ROM [10].

Several projection-based nonlinear model order reduction methods [9], [11], [12] have been derived from linear model order reduction approaches. In [9], a method called NORM is proposed to match the moments of high-order transfer functions explicitly. NORM shows a great advantage in comparison with the methods in [11], [12], since the lower order approximation is skipped. However, NORM builds the ROM whose matrix form tends to be very dense and whose size grows exponentially as the order increases. This fact limits the practicality of NORM for large sparse systems as well as strongly nonlinear systems.

In fact, existing high-order nonlinear model order reduction problems are always doomed by an explosion of matrix dimensions. To overcome such curse of dimensionality, the data structure of tensors [13], as higher order generalizations of matrices, is introduced to effectively capture the high-order nonlinearities and generate highly compact models. The recent work [1] proposed a tensor-based nonlinear model order reduction method named TNMOR. It relies on the CANDECOMP/PARAFAC (CP) decomposition [14], [15] to find a low-rank rank approximation of the original system. Based on the assumption of the existence of a low-rank approximation, TNMOR allows faster simulation and smaller memory requirement than previous nonlinear model order reduction methods.

In this paper, a symmetric tensor-based order-reduction method called STORM is presented. Comparing to TNMOR, STORM avoids the premise that a low-rank approximation of the original system exists, which is a limitation of TNMOR. Meanwhile, via the symmetric tensor structure and the proposed decomposition algorithm, STORM provides significant improvements of computational performance and storage requirement, for both the order-reduction stage and the ROM simulation stage.

In the rest of this paper, Section II reviews the theoretical backgrounds of STORM, including Volterra series, symmetric tensor decomposition, as well as the existing projection-based nonlinear model order reduction methods. The details of STORM are discussed in Section III. In Section IV, numerical examples are given to verify the validity and advantages of STORM. Finally, Section V concludes this paper.

II. BACKGROUND

The theory on Volterra systems, symmetric tensors and their symmetric rank-1 decomposition are first briefly reviewed. Then, we show the principles and limitations of the existing projection-based nonlinear model order reduction methods such as NORM and TNMOR.
A. Volterra System

Volterra series is an important mathematical tool for nonlinear analysis [16]. It decomposes the original nonlinear system into a series of homogeneous nonlinear systems. In this work, we consider a nonlinear multi-input multi-output (MIMO) time-invariant system modeled by the differential-algebraic equation (DAE)

$$\frac{d}{dt}[g(x(t))] + f(x(t)) = Bu(t), \quad y(t) = DT x(t),$$  

(1)

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^l$ are the state and input vectors, respectively; $B$ and $D$ are the input and output matrices, respectively; $g(\cdot)$ and $f(\cdot)$ are smooth nonlinear vector functions of the state vector $x$.

According to the Volterra theorem and variational analysis [17], the system response $x$ can be written as a summation of the responses $x_i$ of the $i$th-order homogeneous nonlinear system, namely $x(t) = \sum_{i=1}^{\infty} x_i(t)$. Therefore, the response of (1) to the inputs $\alpha u(t)$ becomes

$$\frac{d}{dt}[g(x(t))] + f(x(t)) = B\alpha u(t), \quad x(t) = \sum_{i=1}^{\infty} \alpha^i x_i(t).$$  

(2)

The Taylor series expansion of the nonlinear system (1) can be obtained around its equilibrium point $x_0$, that is

$$\frac{d}{dt}[C_1 x + C_2 (x \otimes x) + C_3 (x \otimes x \otimes x) + \cdots] + G_1 x + G_2 (x \otimes x) + G_3 (x \otimes x \otimes x) + \cdots = Bu,$$  

(3)

where $G_k = \frac{1}{k!} \frac{\partial^k f}{\partial x^k} \bigg|_{x=x_0}$, $C_k = \frac{1}{k!} \frac{\partial^k g}{\partial x^k} \bigg|_{x=x_0} \in \mathbb{R}^{n \times n^k}$, and $\otimes$ denotes the Kronecker product, here we use the shorthand $x^\otimes = x \otimes x \otimes \cdots \otimes x$ for $d$-time repeated Kronecker product $\otimes$.

Merging (3) and (2), we get

$$\frac{d}{dt}[\alpha C_1 x_1 + \alpha^2 (C_1 x_2 + C_2 x_2^\otimes) + \cdots] + [\alpha G_1 x_1 + \alpha^2 (G_1 x_2 + G_2 x_2^\otimes) + \cdots] = \alpha Bu.$$  

(4)

By matching the coefficients of the monomials $\alpha^i$, we have the following Volterra subsystems

$$\frac{d}{dt}[C_1 x_1 + G_1 x_1] = Bu,$$  

(5a)

$$\frac{d}{dt}[C_1 x_2 + G_2 x_2] = -\frac{d}{dt} \left[ C_2 x_1^\otimes \right] - G_2 x_2^\otimes,$$  

(5b)

$$\frac{d}{dt}[C_1 x_3 + G_3 x_3] = -\frac{d}{dt} \left[ C_3 x_1^\otimes + C_2 (x_1 \otimes x_1) \right] - G_3 x_3 \otimes$$

$$- G_2 (x_1 \otimes x_2) x_3,$$  

(5c)

and so on, where $(x_1 \otimes x_2) x_3 = x_1 \otimes x_2 \otimes x_3$, or generally $(x_{i_1} \otimes \cdots \otimes x_{i_n}) x_{i_{n+1}} = \sum_{i_{n+1} + \cdots + i_n = k} x_{i_1} \otimes \cdots \otimes x_{i_n}, i_1, \ldots, i_n \in \mathbb{Z}^+.$

B. Symmetric tensor decomposition

Tensors are high-order generalizations of matrices. Following the conventional terminology used by the tensor community, a $d$th-order tensor is a $d$-way array defined as

$$\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d},$$  

(6)

Tensors are denoted by calligraphic letters, e.g., $\mathcal{A}$ and $\mathcal{G}$.

Fig. 1. (a) A tensor $\mathcal{A} \in \mathbb{R}^{3 \times 4 \times 2}$, (b) A symmetric tensor $\mathcal{A} \in \mathbb{R}^{2 \times 2 \times 2}$.

For example, Fig. 1(a) shows a 3rd-order $3 \times 4 \times 2$ tensor. In particular, scalars, vectors and matrices can be regarded as 0th-order, 1st-order and 2nd-order tensors, respectively. A high-order tensor can be unfolded into a 2nd-order matrix by the “matricization” process. The $k$-mode matricization is aligning each $k$th-direction “vector fiber” to be the columns of the matrix.

1) Symmetric tensor: A tensor $\mathcal{X}$ is called cubical if every mode has the same size, for example, $\mathcal{X} \in \mathbb{R}^{I \times I \times \cdots \times I}$. If $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ is symmetric if $\mathcal{A}_{i_1 \cdots i_d} = \mathcal{A}_{\pi(i_1 \cdots i_d)}$ where $\pi(i_1 \cdots i_d)$ is any permutations of the indices $i_1 \cdots i_d$ [18]. Therefore, a symmetric tensor is also a cubical tensor, which means $n_1 = \cdots = n_d = I$. An example of a 3rd-order symmetric tensor $\mathcal{A}$ is shown in Fig. 1(b). In addition, tensors can be partially symmetric in two or more modes. For example, a 3rd-order tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times K}$ is partially symmetric in the first and second mode if its frontal slices are all symmetric, namely $X_k = X_1^T (k = 1, \ldots, K)$.

2) Symmetric tensor decomposition: A rank-1 tensor of order $d$ can be written as the outer product of $d$ vectors

$$\mathcal{A} = a^{(1)} \circ a^{(2)} \circ \cdots \circ a^{(d)}, \quad a^{(k)} \in \mathbb{R}^{n_k \times 1},$$  

(7)

where $\circ$ denotes the outer product. Its element $\mathcal{A}_{i_1 \cdots i_d} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(d)}_{i_d}$, where $a^{(k)}_{i_k}$ is the $i_k$th entry of vector $a^{(k)}$. When $a^{(1)} = a^{(2)} = \cdots = a^{(d)}$, $\mathcal{A}$ is a symmetric rank-1 tensor. A symmetric rank-1 tensor decomposition expresses a symmetric tensor $\mathcal{A}$ as a linear combination of symmetric rank-1 tensors

$$\mathcal{A} = \sum_{r=1}^{R} \lambda_r w_r \circ w_r \circ \cdots \circ w_r, \quad w_r \in \mathbb{R}^{n \times 1},$$  

(8)

where $\lambda_r \in \mathbb{R}$ and $R \in \mathbb{Z}^+$. The CP decomposition drops the symmetry requirement of the rank-1 terms. Here we introduce vec$(\mathcal{A}) \in \mathbb{R}^{n^d \times 1}$ to denote the vectorization of a $d$-way symmetric tensor $\mathcal{A}$ by aligning all entries of $\mathcal{A}$ into a single vector. Using vec$(\mathcal{A})$, (8) can be rewritten as

$$\text{vec}(\mathcal{A}) = WL,$$  

(9)

where $W = [w_1^\otimes, \ldots, w_R^\otimes]$ and $L = [\lambda_1, \ldots, \lambda_R]^T \in \mathbb{R}^{R \times 1}$. The inverse vectorization operation is defined as $\mathcal{A} = \text{unvec}(a)$, which reshapes a vector $a \in \mathbb{R}^{n^d}$ into a tensor $\mathcal{A} \in \mathbb{R}^{n \times \cdots \times n}$. 

The rank of a symmetric tensor $\mathcal{A}$, $\text{rank}(\mathcal{A})$, is the minimum value of $R$ needed for the approximation (8). A symmetric tensor decomposition, or a rank-$R$ approximation of a 3rd-order symmetric tensor is shown in Fig. 2. From now on, all tensors will be assumed symmetric unless otherwise stated.

C. Existing projection-based nonlinear model order reduction methods

In this section we briefly introduce both the NORM and TNMOR methods. NORM [9] first derives frequency-domain high-order nonlinear Volterra transfer functions $H_2(s_1, s_2)$, $H_3(s_1, s_2, s_3)$, etc. associated with the subsystems in (5). Then, these high-order transfer functions are expanded into multivariate polynomials of $s_1, s_2$, etc. via Taylor expansion, such that the coefficients (moments) can be explicitly matched. In NORM, the size of an $N$th-order ROM is $O(N^{N+1})$ supposing up to 4th-order moments of each Volterra transfer function are matched.

The final step in NORM is the replacement of the original nonlinear system (3) by a smaller system (the ROM) via the transformations

$$\tilde{x} = V^T x, \quad \tilde{B} = V^T B, \quad \tilde{D} = V^T D,$$

$$\tilde{G}_i = V^T G_i(V^\top), \quad \tilde{C}_i = V^T C_i(V^\top),$$

where $i = 1, \ldots, N$ and $V$ is the orthogonal projection matrix of the ROM. Suppose $q$ is the size of the reduced state, $\tilde{G}_i$ and $\tilde{C}_i$ will be dense matrices with $O(q^{i+1})$ entries, despite the sparsity of $G_i$ and $C_i$. To store these dense matrices, the memory space required grows exponentially.

TNMOR [1] gives the equivalent tensor models $\tilde{G}_i$ and $\tilde{C}_i$ for $G_i$ and $C_i$ respectively, and applies a CP decomposition [15] to find low-rank approximations

$$\tilde{G}_i \approx \{G_i^{(1)}, \ldots, G_i^{(i+1)}\} = \sum_{r=1}^{R_{g,i}} g_{1,r} \circ \cdots \circ g_{i+1,r},$$

$$\tilde{C}_i \approx \{C_i^{(1)}, \ldots, C_i^{(i+1)}\} = \sum_{r=1}^{R_{c,i}} c_{1,r} \circ \cdots \circ c_{i+1,r},$$

where $i = 1, 2, \ldots, N$, $R_{g,i}$ and $R_{c,i}$ are the tensor ranks of $G_i$ and $C_i$, respectively, $g_{s,i}, c_{s,i} \in \mathbb{R}^n$, $G_i^{(k)} = [g_{s_1,i}^{(k)}, \ldots, g_{s_{r_s},i}^{(k)}] \in \mathbb{R}^{n \times r_s}$, and $C_i^{(k)} = [c_{s_1,i}^{(k)}, \ldots, c_{s_{r_s},i}^{(k)}] \in \mathbb{R}^{n \times r_c}$, for $k = 1, \ldots, i+1$. Then it follows the same routine in [11], [12] to derive the projection matrix $V$. Finally, the ROM of TNMOR is given by

$$\tilde{x} = V^T x, \quad \tilde{B} = V^T B, \quad \tilde{D} = V^T D,$$

$$\tilde{G}_i = V^T G_i V, \quad \tilde{C}_i = V^T C_i V,$$

$$\tilde{G}_i = [G_i^{(1)}, \ldots, G_i^{(i+1)}] = [V^T G_i^{(1)}, \ldots, V^T G_i^{(i+1)}],$$

where $\tilde{G}_i = [\tilde{G}_i^{(1)}, \ldots, \tilde{G}_i^{(i+1)}] = [V^T \tilde{G}_i^{(1)}, \ldots, V^T \tilde{G}_i^{(i+1)}]$, $i = 2, \ldots, N$.

The size of an $N$th-order ROM of TNMOR is in $O(k_1 l + k_2 (r_g + c_2) + k_3 (r_g + c_3) + \cdots + k_N (r_g + r_c)) = O(Nkr)$, where $k = \max\{k_1, \ldots, k_N\}$ and $r = \max\{l, r_g + r_c, \ldots, r_g + r_c\}$. Therefore, the advantage of TNMOR depends explicitly on the existence of low-rank approximations of the tensor-based models.

III. STORM

In this section, the STORM algorithm is developed. We first explain how we use symmetric tensors to represent multivariate polynomial systems. Then, we briefly explain the STEROID algorithm [19] to compute a symmetric rank-1 decomposition of a symmetric tensor as given in (8). Finally, we describe the projection-based steps in STORM for extracting the ROM from the original system.

A. Symmetric tensor model for multivariate polynomial systems

Given the definitions of $G_i$ and $C_i$ in (3), these coefficient matrices can be treated respectively as the 1st-mode matricizations of $(i+1)$th-order tensors $G_i$ and $C_i$,

$$G_i, C_i \in \mathbb{R}^{(1) \times \cdots \times (i+1)},$$

where the elements $(G_i)_{s_1 \cdots s_{i+1}}$ and $(C_i)_{s_1 \cdots s_{i+1}}$ are coefficients of the $\Pi_{k=1}^{i+1} x_{s_k}$ term in $G_i$ and $C_i$, respectively. For instance, $G_2$ is an $n \times n^2$ matrix while $G_2$ is a 3rd-order $n \times n \times n$ tensor, i.e., $G_2$ is the 1st-mode matricization of $G_2$.

Each row vector $G_i(j,:)$ in $\mathbb{R}^{1 \times n^2}$ of $G_i$ can take as the transpose of the vectorization of the $i$th-order symmetric tensor $G_{i,j}$, namely $G_i(j,:)$ $\equiv$ vec($G_{i,j}$). It also works for $C_i$, while $C_i(j,:)$ $\equiv$ vec($C_{i,j}$). Based on (9), for $i > 1$, $G_i$ and $C_i$ can be transformed into

$$G_i = \text{blkdiag}(L_{G_{i,1}}^T, \ldots, L_{G_{i,n}}^T)[W_{G_{i,n}}^T, \ldots, W_{G_{i,1}}^T]^T,$$

$$C_i = \text{blkdiag}(L_{C_{i,1}}^T, \ldots, L_{C_{i,n}}^T)[W_{C_{i,1}}^T, \ldots, W_{C_{i,n}}^T]^T,$$

where $\text{blkdiag}(\cdot)$ outputs a block diagonal matrix.

Let $X_i = x_{(i)}$. Then the $i$th degree terms of the $j$th multivariate polynomial is vec($g_{i,j}$)$^T X_i = L_{G_{i,j}}^T W_{G_{i,j}} X_i$, where from (9) we have that $W_{G_{i,j}} = [W_{G_{i,j,1}}, \ldots, W_{G_{i,j,n}}]$. We can prove by induction, that $W_{G_{i,j}} X_i = [(w_{G_{i,j,1}}(x))^T, \ldots, (w_{G_{i,j,n}}(x))^T]^T$. In order to reduce the computational complexity of computing $W_{G_{i,j}} X_i$, we introduce $U_{G_{i,j}} \triangleq [w_{G_{i,j,1}}, \ldots, w_{G_{i,j,n}}, R_{G_{i,j}}]$, then $W_{G_{i,j}}^T X_i = \odot^i(U_{G_{i,j}}^T x)$. The operator $\odot^i$ stands for the $i$ times’ repeated
Hadamard product $\odot$. Let $L_{G_i} \triangleq \text{blkdiag}(L_{G_i,1}^T, \ldots, L_{G_i,n}^T)$, $M_{C_i}(x) \triangleq [(\odot'(U_{C_i}^T x)^T), \ldots, (\odot'((U_{C_i}^T x)^T))^T]^T$. The same goes for $C_i$ when $i > 1$. Then the system (3) can be transformed into

\[
\frac{d}{dt} [C_1 x + L_{C_1} M_{C_2}(x) + L_{C_3} M_{C_3}(x) + \cdots] + G_1 x + L_{C_2} M_{G_2}(x) + L_{C_3} M_{G_3}(x) + \cdots = Bu.
\]  

(15)

The computational complexity and storage cost of (15) are much smaller than (3), because we can utilize the form of $M_{C_i}(x)$ and $M_{C_i}(x)$ in (15) to accurately compute the high-order nonlinearities. It is worth mentioning that $G_{i,j} = \text{unvec}(G_i(j,:) )$ and $C_{i,j} = \text{unvec}(C_i(j,:))$ may not necessarily be symmetric. However, we can always symmetrize $G_{i,j}$ and $C_{i,j}$ before we apply the symmetric tensor decomposition, which does not change the inner product of $\text{vec}(G_{i,j})^T X_i$.

**B. Symmetric rank-1 tensor decomposition algorithm**

We demonstrate a symmetric tensor eigen-rank-one iterative decomposition (STEROID) algorithm [19] to obtain (9). Although the method is called iterative, it is not in the sense of other tensor decomposition algorithms. No initial guess is required and there is no minimization of an error function, rather the iterative element in STEROID lies in a repeated application of the eigendecomposition algorithm. Hence a comparison with traditional CP methods does not really apply.

In each iteration, the first step is a reshaping of a symmetric tensor into a square symmetric matrix. Given a symmetric $d$-order symmetric tensor $\mathcal{A}$, we can reshape it into a symmetric matrix $A \in \mathbb{R}^{n^d/2 \times n^d/2}$ when $d$ is an even number. Otherwise we need to embed $A$ into a $(d+1)$th-order symmetric tensor $B$ before the reshaping. The details of the embedding step is given in [19]. Since each matrix in the algorithm is symmetric only $n^d (n^d+1)/2$ entries need to be kept in memory. The most expensive computational step is the tridiagonalization of the symmetric matrix in order to compute its eigendecomposition. This requires $8n^d \frac{d^3}{3}$ flops. In practice however, the tensors will be sparse and therefore contain many zero entries. This can be exploited by using a column compressed data structure to store the matrix and use an Implicitly Restarted Lanczos Method to compute the dominant eigenvectors. This usually results in a significant saving in memory storage and runtime. After the reshaping, the eigenvalue decomposition of $A$ can be computed as $A = \sum_{i=1}^{n^d/2} \lambda_i w_i w_i^T$, $w_i \in \mathbb{R}^{n^d/2 \times 1}$. It is shown in [19] that each eigenvectors $w_i$ can be reshaped into another $d/2$-way symmetric tensor $V_i$. Therefore the above iteration will continue until the dimension of $w_i$ becomes $\mathbb{R}^{n \times 1}$.

The last step of STEROID is to collect all $w_i$ into the matrix $W$, and get the vector $L$ in (9) by solving the following least-square problem

\[
\text{argmin}_L \| \text{vec}(A) - W L \|_F.
\]  

(16)

**C. ROM construction**

According to [16], we can obtain the matrix transfer function for (3). For example, the first three transfer functions are

\[
H_1(s) = (sC_1 + G_1)^{-1} B,
\]

(17a)

\[
H_2(s_1, s_2) = -\frac{1}{2} (sC_1 + G_1)^{-1} (sC_2 + G_2) (H_1(s_1) \otimes H_1(s_2)),
\]

(17b)

\[
H_3(s_1, s_2, s_3) = -\frac{1}{6} (sC_1 + G_1)^{-1} ([sC_3 + G_3] \otimes H_1(s_1) \otimes H_1(s_2)) + (sC_2 + G_2) (H_1(s_1) \otimes H_2(s_2, s_3)),
\]

(17c)

where

\[
\bar{s} = s_1 + s_2, \tilde{s} = s_1 + s_2 + s_3,
\]

\[
H_1(s_1) \otimes H_1(s_2) = (H_1(s_1) \otimes H_1(s_2) + H_1(s_2) \otimes H_1(s_1)),
\]

\[
H_1(s_1) \otimes H_1(s_2) \otimes H_1(s_3) = \sum_{(i_1, i_2, i_3) = ((1, 2, 3), (1, 2, 1), (1, 2, 1), (1, 2, 1))} (H_1(s_{i_1}) \otimes H_2(s_{i_2}, s_{i_3}) + H_2(s_{i_2}, s_{i_3}) \otimes H_1(s_{i_4})).
\]

(18)

If we define

\[
A = -G_1^{-1} C_1, R_1 = G_1^{-1} B, R_2 = R_1 \otimes R_1, R_3 = R_1^R, A_l \otimes m \oplus \otimes n = A^l \otimes A^m \oplus \cdots \oplus A^n, \]

\[
\overline{A^{\otimes m}} = \frac{1}{2} (A^{\otimes m} + A^{m \otimes l}),
\]

(19)

then based on (17), the Taylor expansion of these transfer functions can be derived as

\[
H_1(s) = \sum_{k=0}^{\infty} M_{1,k} s^k = \sum_{k=0}^{\infty} A^k R_1 s^k,
\]

(20a)

\[
H_2(s_1, s_2) = \sum_{k=0}^{\infty} \sum_{l=0}^{k} s_1^l s_2^{k-l} M_{2,k,l},
\]

(20b)

\[
H_3(s_1, s_2, s_3) = \sum_{k=0}^{\infty} \sum_{l=0}^{k} \sum_{m=0}^{l} s_{1}^{l} s_{2}^{m} s_{3}^{k-l-m} M_{3,k,l,m}.
\]

(20c)

From (20a), it is seen that the $k$th-order moment $M_{1,k}$ of the first-order transfer function $H_1(s)$ is equal to $A^k R_1$. Therefore, we can compute a Krylov subspace to match up to $k$th-order moments of $H_1$, which is

\[
V_1 = [v_1, v_1, \ldots, v_1, v_1] = \text{orth}(K_{k+1}(A, R_1)).
\]

(21)

where $K_{k+1}(A, R_1)$ returns the first $k + 1$ vectors of the Krylov subspace spanned by $A$.

By substituting (20a) into (17b), the $k$th-order moment $M_{2,k,l}$ of the second-order transfer function $H_2(s)$ can be expressed as

\[
\sum_{p=0}^{k} \sum_{q=0}^{p} \sum_{s=0}^{p} \left( \frac{p}{q} \right) G_{s}^{-1} C_{2} \cdot A^{(l-q) \otimes (k-l-p+q)} R_{2}
\]

\[
- \sum_{p=0}^{k} \sum_{q=0}^{p} \sum_{s=0}^{p} \left( \frac{p}{q} \right) G_{s}^{-1} C_{2} \cdot A^{(l-q) \otimes (k-l-p+q)} R_{2}
\]

(22)

Here we avoid the explicit computation of $A^i, i \in \mathbb{Z}^+$ when deriving the starting vectors for each Krylov subspace, which
can lead to numerical stability problems [8]. Instead, we replace \( A^{m\otimes n}R_2 \) by \((v_{1,m} \otimes v_{1,n})\), where \(v_{1,m}\) and \(v_{1,n}\) are column vectors from \(V_1\) in (21). And it can be proved by induction that this replacement will not change the subspace. If we define \( N_{C_i} \triangleq \left[ U_{C_i,1}, \ldots, U_{C_i,n} \right]^T \), \( N_{C_i} \triangleq \left[ U_{C_i,1}, \ldots, U_{C_i,n} \right]^T \) then we utilize the symmetric tensor model of \( C_2 \) and \( G_2 \), namely \( C_2(v_{1,m} \otimes v_{1,n}) = L_{C_2}((N_{C_2}v_{1,m}) \otimes (N_{C_2}v_{1,n})) \) and \( G_2(v_{1,m} \otimes v_{1,n}) = L_{G_2}((N_{G_2}v_{1,m}) \otimes (N_{G_2}v_{1,n})) \), to avoid the complexity to compute the Kronecker products. As a result, in the computation of the starting vectors of the Krylov subspaces, the computation of both \( A^1 \) and the Kronecker products \( A^{m\otimes n}R_2 \) are avoided. To match up to \( k_2 \)th-order moments of \( H_2 \), we get the projection matrix \( V_2 \) as

\[
V_2 = \text{orth}((\{K_{k_2-m-n}(A, G_1^{-1}L_{C_2}((N_{C_2}v_{1,m}) \otimes (N_{C_2}v_{1,n}))), \\
k_{k_2-m-n+1}(A, G_1^{-1}L_{G_2}((N_{G_2}v_{1,m}) \otimes (N_{G_2}v_{1,n}))\}), \\
m \geq 0, n \geq 0, m \leq n, m + n \leq k_2)).
\]

Similar procedure can be applied to generate \( V_j \) when \( j > 2 \), which matches up to the \( k \)th-order moments of the \( j \)th-order transfer function \( H_j \). In the end, the projection matrix for reducing the nonlinear system is denoted by \( V_p = \text{orth}\{V_1, V_2, \ldots\} \). A general procedure to obtain the projection matrix \( V_p \) for an expansion around the DC operating points is given in Algorithm 1. The proof of the validity of the \( V_p \) can be found in [9].

The size of an \( N \)th-order ROM is \( O(k_1l + k_2l + k_3l + \cdots + k_Nl) = O(Nkl) \), while \( l \) is the size of the input signal. Comparing with \( O(k^{N+1}N) \) in NORM and \( O(Nkl) \) in TNMOR, a slimmer ROM can be achieved by STORM. Moreover, it avoids the limitation of the existence of a low-rank tensor decomposition in TNMOR. Finally, the symmetric tensor-based ROM is given by the following projection

\[
\begin{align*}
\dot{x} &= V_p^T x, \quad \dot{B} = V_p^TB, \quad \dot{D} = V_p^TD, \\
\hat{L}_{C_i} &= V_p^T L_{C_i} V_p, \quad \hat{L}_{G_i} = V_p^T L_{G_i}, \\
\hat{N}_{C_i} &= N_{C_i} \ast V_p, \quad \hat{N}_{G_i} = N_{G_i} \ast V_p, \\
\tilde{C}_i &= \hat{L}_{C_i}((s^i(\hat{N}_{C_i}^T))^T), \quad \tilde{G}_i = \hat{L}_{G_i}((s^i(\hat{N}_{G_i}^T))^T),
\end{align*}
\]

where \( s^i \) denotes the \( i \) times repeated Khatri-Rao product\(^2\). By utilizing this structure, only the \( L_{C_i}, L_{G_i}, \hat{N}_{C_i} \) and \( \hat{N}_{G_i} \) matrices need to be stored. Thus in the simulation stage, significant speedup is achieved as shown in the next section.

**IV. NUMERICAL EXAMPLES**

In this section, the proposed STORM method is demonstrated and compared with two existing projection-based nonlinear model order reduction methods, NORM and TNMOR. All three approaches are implemented in Matlab [20], and the time-domain transient analysis is solved by the built-in ordinary differential equation solver in Matlab. Two numerical experiments are performed on an Intel Core i7 desktop PC with 2.6GHz CPU and 8GB RAM.

The first example is a double-balanced mixer circuit in Fig. 3(a), where \( V_{rf}(= V_{rf+} - V_{rf-}) \) and \( V_{lo}(= V_{lo+} - V_{lo-}) \) are

\(^{2}\text{Khatri-Rao product is a column-wise Kronecker Product of two matrices.}\)

![Algorithm 1 Generation of projection matrix for STORM](image)

**Algorithm 1 Generation of projection matrix for STORM**

**Input:** \( N, G_i, C_i, B, k_i, k_N \leq k_N-1 \leq \cdots \leq k_1 \)

**Output:** the projection matrix \( V_p \)

1. \( A = -G_1^{-1}C_1, R_1 = G_1^{-1}B \)
2. Generate matrix \( V_1 = [v_{1,0}, v_{1,1}, \ldots, v_{1,k_1}] \) for \( K_{k_1+1}(A, R_1) \)
3. for \( i = 2 \to N \) do
   4. \( G_i = \text{blkdiag}(L_{G_i,1}^T, \ldots, L_{G_i,n}^T)(W_{G_i,1}, \ldots, W_{G_i,n})^T \)
   5. \( C_i = \text{blkdiag}(L_{C_i,1}^T, \ldots, L_{C_i,n}^T)(W_{C_i,1}, \ldots, W_{C_i,n})^T \)
   6. Build the starting vector set \( v_{k_i} \) based on the moment expression and the symmetric tensor model
   7. Get the projection matrix \( V_i = \text{orth}(K_{k_i+1}(A, v_{k_i})) \) which matches up to \( k_i \)th-order moments of \( H_i \)
8. end for
9. \( V_p = \text{orth}\{V_1, V_2, \ldots, V_N\} \)

Fig. 3. (a) A double-balanced mixer. (b) A nonlinear transmission line.

The RF and local oscillator (LO) inputs, respectively. Three transient simulations of the ROMs generated by different nonlinear model order reduction methods are performed from \( T = 0\)ns to \( T = 40\)ns with the time step \( \Delta T = 1\)ps. The reduction time and generated ROM size are recorded in Table I. The transient simulation time and relative error of different methods are listed in Table II. The error is calculated by the output difference between ROM and full model, \( ||V_{\text{ROM}} - V_{\text{full}}||_2/||V_{\text{full}}||_2 \). According to the result, we find that STORM shows a better computational efficiency of transient analysis compared to NORM and TNMOR. Moreover, the reduction scheme of STORM can match higher order moments while the other two methods cannot.

The second example is a nonlinear transmission line circuit shown in Fig. 3(b). It is a common benchmark for testing
nonlinear model order reduction methods. The transient simulation is executed to $T = 400\,\text{ns}$ with a step size $\Delta T = 1\,\text{ps}$. TNMOR is not suitable for this case because the low-rank approximation does not exist. However, STORM still works and outperforms NORM in terms of the simulation time. Moreover, the NORM fails when the size of the original system exceeds 1000 because of shortage of memory, while STORM still works well. The details of the comparisons are recorded in Table III and Table IV.

V. Conclusion

This paper has presented a symmetric tensor-based order-reduction method called STORM. The key feature of STORM is to build symmetric tensor models for high-order polynomial nonlinear systems. By utilizing the symmetric tensor decomposition and the new projection-based reduction scheme under the symmetric tensor framework, faster ways of model order reduction and transient analysis are demonstrated. The advantage of STORM has been proved by comparison with existing nonlinear model order reduction methods on some benchmark circuits. STORM has shown the potential of the symmetric tensor structure for efficient simulation of high-order nonlinear large-scale systems.

TABLE III

<table>
<thead>
<tr>
<th>Method</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>CPU time (s)</th>
<th>size of ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM</td>
<td>2</td>
<td>4</td>
<td>0.5s</td>
<td>15</td>
</tr>
<tr>
<td>STORM</td>
<td>2</td>
<td>4</td>
<td>0.4s</td>
<td>15</td>
</tr>
</tbody>
</table>

TABLE IV

<table>
<thead>
<tr>
<th>Transient</th>
<th>full model</th>
<th>NORM</th>
<th>STORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>80</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>33.61</td>
<td>16.35</td>
<td>7.18</td>
</tr>
<tr>
<td>speedup</td>
<td>—</td>
<td>2x</td>
<td>4x</td>
</tr>
<tr>
<td>error</td>
<td>—</td>
<td>0.03%</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

TABLE I

<table>
<thead>
<tr>
<th>Method</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>CPU time (s)</th>
<th>size of ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORM</td>
<td>2</td>
<td>2</td>
<td>—</td>
<td>35.35</td>
<td>48</td>
</tr>
<tr>
<td>TNMOR</td>
<td>2</td>
<td>2</td>
<td>—</td>
<td>9.53</td>
<td>36</td>
</tr>
<tr>
<td>STORM</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>16.31</td>
<td>49</td>
</tr>
</tbody>
</table>

TABLE II

<table>
<thead>
<tr>
<th>Transient</th>
<th>full model</th>
<th>NORM</th>
<th>TNMOR</th>
<th>STORM</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>93</td>
<td>48</td>
<td>36</td>
<td>49</td>
</tr>
<tr>
<td>CPU time (s)</td>
<td>991.51</td>
<td>172.73</td>
<td>16.86</td>
<td>3.23</td>
</tr>
<tr>
<td>speedup</td>
<td>—</td>
<td>6x</td>
<td>60x</td>
<td>300x</td>
</tr>
<tr>
<td>error</td>
<td>—</td>
<td>5.05%</td>
<td>3.24%</td>
<td>4.08%</td>
</tr>
</tbody>
</table>

ACKNOWLEDGEMENTS

This work was supported in part by the Hong Kong Research Grants Council under Project HKU 172085/14 and the University Research Committee of The University of Hong Kong.

REFERENCES