An Adaptive Dynamical Low-Rank Tensor Approximation Scheme for Fast Circuit Simulation

Kim Batselier, Quan Chen, Ngai Wong
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: kimb,quanchen,nwong@eee.hku.hk

***Abstract*—Tensors, as higher order generalization of matrices, have received growing attention due to their readiness in representing multidimensional data intrinsic to numerous engineering problems. This paper develops an efficient and accurate dynamical update algorithm for the low-rank mode factors. By means of tangent space projection onto the low-rank tensor manifold, the repeated computation of a full tensor Tucker decomposition is replaced with a much simpler solution of nonlinear differential equations governing the tensor mode factors. A worked-out numerical example demonstrates the excellent efficiency and scalability of the proposed dynamical approximation scheme.**

I. Introduction

Tensors, as higher order generalization of matrices, have found great success and applications in various fields such as mobile communications, biomedical engineering, signal processing and big data [1], [2], [3], [4], [5]. An extensive survey on tensor decompositions and applications can be found in [6]. Whenever the efficient decomposition of a tensor into low-rank factors exists, it often provides an effective means to overcome the curse of dimensionality arising from the explosion of data in modern modeling problems. However, utilization of tensors in design automation (EDA) remains relatively obscure despite their perfect fit to many electronic design problems. In particular, this paper introduces a dynamical low-rank approximation to accelerate circuit simulation based on a tensor formulation and operator splitting. Specifically, suppose we have the following tensor differential equation

$$\dot{\mathcal{A}}(t) = F(\mathcal{A}(t))$$

where $\mathcal{A}$ is a time-varying tensor and $F(\cdot)$ is an arbitrary nonlinear tensor function. We then compute a low-rank approximate $\mathcal{Y}(t)$ of $\mathcal{A}(t)$ such that $||\dot{\mathcal{Y}} - F(\mathcal{Y})||_F$ is minimal.

The idea of dynamical low-rank approximation is not new [7], [8], [9], [10] and dates back to as early as 1930 [11]. The central idea is to project the dynamics onto the tangent space of a low-rank matrix/tensor manifold whereby nonlinear differential equations are derived for the low-rank matrix/tensor factors, e.g., [8], [9], [10]. Consequently, instead of evaluating the full dense time-varying matrix/tensor, only the low-rank factors are updated over time, resulting in significant savings in computation while maintaining accuracy [12]. Nonetheless, numerical integration in the aforesaid differential equations often involves inverting a core matrix that is necessarily near-singular by construction. Regularization by adding small numbers to this matrix is conventionally done but apparently this is highly ad-hoc and the error behaviour is hard to predict. Recently, a standard operator splitting method has been applied to the dynamical low-rank approximation of a time-varying matrix [12]. By doing so, the ill-conditioned matrix inverse is completely avoided.

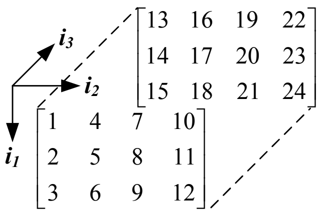

Fig. 1. An example tensor $\mathcal{A} \in \mathbb{R}^{3\times4\times2}$.

This paper contributes by extending this operator splitting strategy to a general time-varying tensor. In the following, Section II reviews the necessary basics of tensors and tangent spaces in differential geometry. Section III presents the proposed dynamical tensor low-rank update algorithm based on orthogonal projections and operator splitting methods. The effectiveness of our developed algorithm is verified by means of a numerical example in Section IV. Finally, Section V draws the conclusions.

II. Background

A. Tensor Basics

A dth-order (or d-way) tensor, assumed real in this work, is a multi-way array $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ generalizing the matrix format to its dth-order counterpart [6], wherein the n_i's are called the dimensions. We use calligraphic font, e.g., $\mathcal{A}$, to denote a tensor. An example of a 3rd-order real tensor is shown in Fig. 1. Obviously, a tensor reduces to a matrix and a vector when $d = 2$ and $d = 1$, respectively. The inner product between two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ is defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i_1,i_2,\cdots,i_d} \mathcal{A}_{i_1i_2\cdots i_d}\mathcal{B}_{i_1i_2\cdots i_d} = \text{vec}(\mathcal{A})^T \text{vec}(\mathcal{B}) \quad (1)$$

where vec$(\cdot)$ is the vectorization operator that stacks the tensor entries into a tall column vector. In the index tuple $[i_1i_2\cdots i_d]$, i_1 is conventionally assumed to be the fastest changing index while i_d the slowest, so vec$(\mathcal{A})$ will arrange the entries $\mathcal{A}_{11\cdots1}$, $\mathcal{A}_{21\cdots1}$, $\cdots$, $\mathcal{A}_{12\cdots1}$, $\cdots$, $\mathcal{A}_{n_1n_2\cdots n_d}$ from top to bottom. The norm of a tensor is defined to be the Frobenius norm $||\mathcal{A}|| = ||\mathcal{A}||_F = \sqrt{<\mathcal{A},\mathcal{A}>}$.

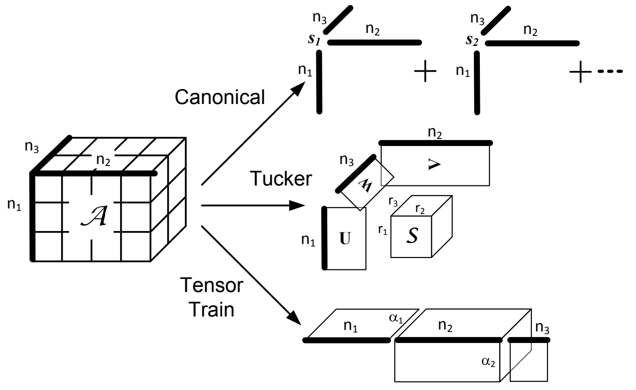

Fig. 2. Possible tensor decompositions for $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

The k-mode product of a tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times \cdots \times n_k \times \cdots \times n_d}$ with a matrix $U \in \mathbb{R}^{p_k \times n_k}$ is defined by [6]

$$(\mathcal{A}_{\times_k} U)_{i_1 \cdots i_{k-1} j_k i_{k+1} \cdots i_d} = \sum_{i_k=1}^{n_k} U_{j_k i_k} \mathcal{A}_{i_1 \cdots i_k \cdots i_d}, \quad (2)$$

and $\mathcal{A}_{\times_k} U \in \mathbb{R}^{n_1 \times \cdots \times n_{k-1} \times p_k \times n_{k+1} \times \cdots \times n_d}$.

A rank-1 tensor is defined by the outer product of vectors. As an example, with a scalar s and vectors $a \in \mathbb{R}^{n_1}$, $b \in \mathbb{R}^{n_2}$ and $c \in \mathbb{R}^{n_3}$, a 3rd-order outer product $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is

$$\mathcal{X} = s(a \circ b \circ c) = s_{\times_1} a_{\times_2} b_{\times_3} c, \quad (3)$$

wherein $\mathcal{X}_{i_1 i_2 i_3} = s a_{i_1} b_{i_2} c_{i_3}$. We remark that the first s in (3) is in $\mathbb{R}$ and the second is in $\mathbb{R}^{1 \times 1 \times 1}$ representing a "tensor scalar". These dimensions are not explicitly stated whenever they are obvious from context. A useful result of the vectorization of an outer product is that $\text{vec}(a \circ b \circ c) = \text{vec}(1_{\times_1} a_{\times_2} b_{\times_3} c) = c \otimes b \otimes a$, where $\otimes$ denotes the matrix Kronecker product [6]. Similar to the matrix SVD, an arbitrary tensor can always be decomposed into a sum of outer products generally called the canonical decomposition, depicted in the top branch of Fig. 2. Other popular tensor decompositions include the Tucker decomposition [1] where a tensor is cast as a usually smaller dense "core tensor" multiplied with orthogonal matrices on various modes, or the tensor train (TT) [3] as a product cascade of multiple 3rd-order tensors in between a head and a tail matrix.

B. Tangent Space on a Manifold

First we consider the manifold of orthogonal rank-r matrices $V \in \mathbb{R}^{m \times r}$ $(m > r)$:

$$\mathcal{M}_V = \left\{ V : V \in \mathbb{R}^{m \times r}, V^T V = I \right\}. \quad (4)$$

Another name for $\mathcal{M}_V$ in (4) is the Stiefel manifold, e.g., [7]. Each column in V is of unit norm and orthogonal to all its columns on the left, so $\dim(\mathcal{M}_V) = mr - (1 + 2 + \cdots r) = mr - r(r+1)/2$. Treating V as a function of time and differentiating $V^T V = I$ with respect to time we obtain $\dot{V}^T V + V^T \dot{V} = 0$. The tangent space at V on the Stiefel manifold $\mathcal{M}_V$ is therefore

$$\mathcal{T}_V \mathcal{M}_V = \left\{ \delta V : \delta V^T V + V^T \delta V = 0 \right\}. \quad (5)$$

It is readily checked that the skew symmetry requires $(1 + 2 + \cdots + r) = r(r+1)/2$ constraints so the number of degrees of freedom in δV, and therefore $\dim(\mathcal{T}_V \mathcal{M}_V)$, is $mr - r(r+1)/2$. We now consider a rank-r matrix $Y \in \mathbb{R}^{m \times n}$ expressed in its SVD form $Y = U \Sigma V^T$ where $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$ are orthogonal and $\Sigma = diag(\sigma_1, \cdots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$. The matrix Y is a point on the manifold of rank-r matrices

$$\mathcal{M}_Y = \left\{ Y : Y = U \Sigma V^T \in \mathbb{R}^{m \times n} \text{ has rank } r \right\}. \quad (6)$$

Since U and V are living on the Stiefel manifolds their dimensions are $mr - r(r+1)/2$ and $nr - r(r+1)/2$, respectively, and added with the r positive singular values in Σ this amounts to $\dim(\mathcal{M}_Y) = (m+n)r - r^2$. Observing that $\dot{Y} = \dot{U} \Sigma V^T + U \dot{\Sigma} V^T + U \Sigma \dot{V}^T$, the elements of the tangent space $\delta Y = (\delta U) \Sigma V^T + U (\delta \Sigma) V^T + U \Sigma (\delta V)^T$ in this case are constrained by δU, δV being on the Stiefel manifolds and $Y(t)$ always being of rank r. Compared to the previous orthogonal matrix case (5), it is not so obvious to identify the tangent space $\mathcal{T}_Y \mathcal{M}_Y$ for (6). However, using $\perp$ to denote the orthogonal complement as before, a smart observation is that $U_\perp^T (\delta Y) V_\perp \in \mathbb{R}^{(m-r) \times (n-r)}$ is always zero which is fulfilled by

$$\mathcal{T}_Y \mathcal{M}_Y = \left\{ \delta Y = \begin{bmatrix} U & U_\perp \end{bmatrix} \begin{bmatrix} Z_{11} & Z_{12} \\ Z_{21} & 0 \end{bmatrix} \begin{bmatrix} V^T \\ V_\perp^T \end{bmatrix} : \right.$$
$$\left. Z_{11} \in \mathbb{R}^{r \times r}, Z_{12} \in \mathbb{R}^{r \times (n-r)}, Z_{21} \in \mathbb{R}^{(m-r) \times r} \right\}, \quad (7)$$

and this $\mathcal{T}_Y \mathcal{M}_Y$ must be correct as it represents a linear space whose dimension is equal to the number of parameters in Z_{ij}'s, which is $\dim(\mathcal{T}_V \mathcal{M}_Y) = r^2 + (m-r)r + (n-r)r = (m+n)r - r^2 = \dim(\mathcal{M}_Y)$.

III. Low-Rank Tensor Update

A. Projection and operator splitting

We first illustrate the idea of low-rank dynamical approximation of a time-varying matrix, whereas the tensor generalization is derived later. From Section II-B, a dynamical rank-r $m \times n$ matrix $Y(t)$ will have its time differential lying in its tangent space. Therefore, one way to prescribe a rank-r approximation to a given time-varying matrix $A(t) \in \mathbb{R}^{m \times n}$ is by solving the optimization problem $\min ||\dot{Y}(t) - \dot{A}(t)||$ [7], [8], [9] which results in nonlinear differential equations along every matrix mode. However, these works all involve the cumbersome inverse of a core matrix which is by construction nearly singular when the approximating rank r is higher than the actual rank of A, i.e., over-approximation. A major drawback in these schemes is that the effective rank of a dynamical system is generally not known beforehand, and over-approximation is required in practice. Regularization by adding small finite numbers to the core matrix makes it nonsingular but necessarily introduces errors.

A robust low-rank matrix updating-approach by the technique of operator splitting is demonstrated in [12], which completely skips the matrix inversion. First, given a time-varying matrix A, the constraint $\min ||\dot{Y} - \dot{A}||$ is solved via projecting $\dot{A}$ onto the tangent space of $Y = USV^T$ where U, V are orthogonal and S generalizes Σ to any rank-r nonsingular matrix. It is readily checked that the same tangent space as in (7) is obtained. Then, projection of $\dot{A}$ onto $\mathcal{T}_Y\mathcal{M}_Y$ is done through the projector $P_{\mathcal{T}_Y}(\dot{A})$ [12]

$$\begin{aligned}\dot{Y} = P_{\mathcal{T}_Y}(\dot{A}) &= \dot{A} - U_\perp U_\perp^T \dot{A} V_\perp V_\perp^T \\ &= \dot{A} - (I - UU^T)\dot{A}(I - VV^T) \\ &= \dot{A}VV^T - UU^T\dot{A}VV^T + UU^T\dot{A}. \qquad (8)\end{aligned}$$

Contrasting with (7), the first line in (8) corresponds to subtracting the lower right (zero) block in the core matrix of δY that does not get projected on the tangent space, whereas the last line corresponds to computing and summing the components [nonzero blocks in (7)] that lie on the tangent space. This allows us to compute the projection onto the tangent space in one way or another, viz. by addition or elimination. Starting with some initial U_0, Y_0, S_0, first-order operator splitting then permits one to approximate Y via solving three differential equations, namely, $\dot{Y} = \dot{A}VV^T$, $\dot{Y} = -UU^T\dot{A}VV^T$ and $\dot{Y} = UU^T\dot{A}$ in this order along a fixed time step. The terminal condition from solving one differential equation is used as the initial condition for the next equation. More implementation details can be found in [12].

Now we consider the manifold $\mathcal{M}_\mathcal{Y}$ of 3rd-order tensors $\mathcal{Y} = \mathcal{S}_{\times_1}U_{\times_2}V_{\times_3}W \in \mathbb{R}^{n_1\times n_2\times n_3}$ expressed in the Tucker format with $\mathcal{S} \in \mathbb{R}^{r_1\times r_2\times r_3}$ and U, V, W orthogonal matrices (cf. Fig. 2). Obviously,

$$\begin{aligned}\dot{\mathcal{Y}} = &\dot{\mathcal{S}}_{\times_1}U_{\times_2}V_{\times_3}W + \mathcal{S}_{\times_1}\dot{U}_{\times_2}V_{\times_3}W \\ &+ \mathcal{S}_{\times_1}U_{\times_2}\dot{V}_{\times_3}W + \mathcal{S}_{\times_1}U_{\times_2}V_{\times_3}\dot{W}. \qquad (9)\end{aligned}$$

Similar to the matrix case, $\dot{\mathcal{Y}}$ is found as the orthogonal projection of $\dot{\mathcal{A}}$ onto the tangent space $\mathcal{T}_\mathcal{Y}\mathcal{M}_\mathcal{Y}$ at $\mathcal{Y}$ by

$$\begin{aligned}\dot{\mathcal{Y}} =& P_{\mathcal{T}_\mathcal{Y}}(\dot{\mathcal{A}}) = \dot{\mathcal{A}} - \dot{\mathcal{A}}_{\times_1}U_\perp U_\perp^T{}_{\times_2}V_\perp V_\perp^T{}_{\times_3}W_\perp W_\perp^T \\ =& \dot{\mathcal{A}}_{\times_1}UU^T + \dot{\mathcal{A}}_{\times_2}VV^T + \dot{\mathcal{A}}_{\times_3}WW^T \\ &- \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T - \dot{\mathcal{A}}_{\times_2}VV^T{}_{\times_3}WW^T \\ &- \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_3}WW^T + \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T{}_{\times_3}WW^T.\end{aligned}$$

By applying the same reasoning as in (8) in subtracting zero blocks in the core tensor that do not fall onto the tangent space, the following simpler expression can be written down

$$\begin{aligned}\dot{\mathcal{Y}} =& \dot{\mathcal{A}}_{\times_2}VV^T{}_{\times_3}WW^T - \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T{}_{\times_3}WW^T \\ &+ \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_3}WW^T - \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T{}_{\times_3}WW^T \\ &+ \dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T. \qquad (10)\end{aligned}$$

This brings us to the major contribution in this work, which is the application of first-order Lie-Trotter splitting in solving the tensor equation (10) from t_0 to $t_1 = t_0 + h$. This involves solving differential equations of the form $\dot{\mathcal{Y}}_k = P_k(\dot{\mathcal{A}})$, where $P_k(\cdot)$ is the k-th term in the right-hand side of (10) and $\dot{\mathcal{Y}}_k$ is the k-th intermediate result. Due to space constraints it is not possible to describe the whole solution strategy for the Lie-Trotter splitting. We will therefore illustrate how to solve the first 2 terms. The same reasoning can be applied to solve the remaining terms. The first set of differential equations that need to be solved is $\dot{\mathcal{Y}}_1 = P_1(\dot{\mathcal{A}}) = \dot{\mathcal{A}}_{\times_2}VV^T{}_{\times_3}WW^T$. Assuming that an initial Tucker decomposition $\mathcal{Y}(t_0) = S(t_0)_{\times_1}U(t_0)\times_2 V(t_0)_{\times_3}W(t_0)$ is available and that $V(t), W(t)$ remain constant over the interval $[t_0, t_1]$, then integration of $\dot{\mathcal{Y}}_1 = P_1(\dot{\mathcal{A}})$ over the time interval $[t_0, t_1]$ results in

$$\begin{aligned}\mathcal{Y}_1(t_1) - \mathcal{Y}(t_0) = &\mathcal{A}(t_1)_{\times_2}V(t_0)V(t_0)^T{}_{\times_3}W(t_0)W(t_0)^T \\ &- \mathcal{A}(t_0)_{\times_2}V(t_0)V(t_0)^T{}_{\times_3}W(t_0)W(t_0)^T. \qquad (11)\end{aligned}$$

Introducing the notation $\Delta\mathcal{A} \triangleq \mathcal{A}(t_1) - \mathcal{A}(t_0)$ and replacing $\mathcal{Y}(t)$ by its Tucker decomposition we can rewrite (11) as

$$\begin{aligned}S_1(t_1)_{\times_1}U_1(t_1)_{\times_2}V(t_0)_{\times_3}W(t_0) = &S(t_0)_{\times_1}U(t_0)_{\times_2}V(t_0) \\ &{}_{\times_3}W(t_0) + \Delta\mathcal{A}_{\times_2}V(t_0)V(t_0)^T{}_{\times_3}W(t_0)W(t_0)^T.\end{aligned}$$

Observe that we have given the core tensor $S_1(t_1)$ and matrix factor $U_1(t_1)$ on the left hand side of (11) a subscript index 1 to indicate that these will be the first intermediate results that are computed. Multiplying (11) along the second and third modes with $V(t_0)^T$ and $W(t_0)^T$ respectively then results in

$$\begin{aligned}S_1(t_1)_{\times_1}U_1(t_1) = &S(t_0)_{\times_1}U(t_0) \\ &+ \Delta\mathcal{A}_{\times_2}V(t_0)^T{}_{\times_3}W(t_0)^T, \qquad (12)\end{aligned}$$

which allows us to compute $S_1(t_1)$ and an orthogonal $U_1(t_1)$ from the Tucker decomposition of the right hand side of (12). These computed quantities together with $V(t_0), W(t_0)$ serve then as inputs for solving the second set of differential equations $\dot{\mathcal{Y}}_2 = P_2(\dot{\mathcal{A}}) = -\dot{\mathcal{A}}_{\times_1}UU^T{}_{\times_2}VV^T{}_{\times_3}WW^T$. This means that we now set $S(t_0) \triangleq S_1(t_1), U(t_0) \triangleq U_1(t_1)$ in the Tucker decomposition of $\mathcal{Y}_2$ and assume that $U(t), V(t), W(t)$ remain constant over the time interval $[t_0, t_1]$. Similar reasoning as in solving the first set of differential equations and multiplication along modes $1, 2, 3$ with $U(t_0)^T, V(t_0)^T$ and $W(t_0)^T$ respectively then results in

$$S_2(t_1) = S(t_0) - \Delta\mathcal{A}_{\times_1}U(t_0)^T{}_{\times_2}V(t_0)^T{}_{\times_3}W(t_0)^T,$$

from which the second intermediate result $S_2(t_1)$ is computed. The remaining steps of the Lie-Trotter operator splitting are solved in a similar fashion. The whole procedure of applying first-order Lie-Trotter operator splitting to solve $\dot{\mathcal{Y}} = P_{\mathcal{T}_\mathcal{Y}}(\dot{\mathcal{A}})$ is summarized in Algorithm 3.1, where Tucker($\cdot$) computes the Tucker decomposition with orthogonal factor matrices U, V, W.

```
Algorithm 3.1: First-order splitting method
Input: $S(t_0), U(t_0), V(t_0), W(t_0) \leftarrow$ Tucker($\mathcal{Y}(t_0)$), $\Delta\mathcal{A}$
Output: $\mathcal{Y}(t_1)$
    $\mathcal{T}_1 \leftarrow S(t_0)_{\times_1}U(t_0) + \Delta\mathcal{A}_{\times_2}V(t_0)^T{}_{\times_3}W(t_0)^T$
    $S_1(t_1), U_1(t_1) \leftarrow$ Tucker($\mathcal{T}_1$)
    $S(t_0) \leftarrow S_1(t_1), U(t_0) \leftarrow U_1(t_1)$
    $\mathcal{T}_2 \leftarrow S(t_0) - \Delta\mathcal{A}_{\times_1}U(t_0)^T{}_{\times_2}V(t_0)^T{}_{\times_3}W(t_0)^T$
    $S_2(t_1) \leftarrow$ Tucker($\mathcal{T}_2$)
    $S(t_0) \leftarrow S_2(t_1)$
    $\mathcal{T}_3 \leftarrow S(t_0)_{\times_2}V(t_0) + \Delta\mathcal{A}_{\times_1}U(t_0)^T{}_{\times_3}W(t_0)^T$
    $S_3(t_1), V_3(t_1) \leftarrow$ Tucker($\mathcal{T}_3$)
    $S(t_0) \leftarrow S_3(t_1), V(t_0) \leftarrow V_3(t_1)$
    $\mathcal{T}_4 \leftarrow S(t_0) - \Delta\mathcal{A}_{\times_1}U(t_0)^T{}_{\times_2}V(t_0)^T{}_{\times_3}W(t_0)^T$
    $S_4(t_1) \leftarrow$ Tucker($\mathcal{T}_4$)
    $S(t_0) \leftarrow S_2(t_4)$
    $\mathcal{T}_5 \leftarrow S(t_0)_{\times_3}W(t_0) + \Delta\mathcal{A}_{\times_1}U(t_0)^T{}_{\times_2}V(t_0)^T$
    $S_5(t_1), W_5(t_1) \leftarrow$ Tucker($\mathcal{T}_5$)
    $S(t_0) \leftarrow S_5(t_1), W(t_0) \leftarrow W_5(t_1)$
    $\mathcal{Y}(t_1) = S(t_0) \times_1 U(t_0) \times_2 V(t_0) \times_3 W(t_0)$
```

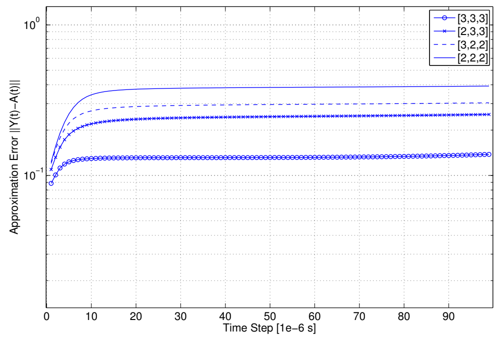

Fig. 3. Approximation errors $||\mathcal{A}(t) - \mathcal{Y}(t)||$ for different core sizes.

IV. Numerical Examples

All experiments were run in Matlab on a Intel quad-core running at 3GHz with 7 GB of RAM. The Matlab implementation of Algorithm 3.1 is freely available on request.

A. Numerical Experiment

One good avenue for applying this low-rank update technique is in the simulation of reaction-diffusion models commonly employed in transistor and MOSFET simulations, e.g, (eqns. (2) & (3) in [13]) where 3rd-order time-varying tensors naturally arise. Due to space constraints we choose to verify and apply Algorithm 3.1 to a repeatable and similar setup [8]. This involves the following reaction-diffusion problem

$$\frac{\partial u}{\partial t} = \Delta u + u^3, \quad (x, y, z) \in \Omega = [0, 1]^3, t > 0$$

with Dirichlet boundary conditions and initial data

$$u(x, y, z, 0) = 30 \exp(-100(x - \frac{1}{2})^2 - 100(y - \frac{1}{2})^2 \\ - 100(z - \frac{1}{2})^2) + \chi(x, y, z),$$

where $\chi(x, y, z) \in (0, 10^{-3})$ is a random number. The variable $u(x, y, z, t) \in \Omega = [0, 1]^3$ was discretized into a $100\times100\times10$ 3rd-order tensor for each time t. The nonlinear PDE was then solved using a backward Euler method and Newton's method over a time span of 10^{-4} seconds with a time step size of 10^{-6} seconds, generating a time varying 3rd-order tensor $\mathcal{A}(t) \triangleq u(x, y, z, t)$. This took about 435.44 seconds. Figure 3 shows the approximation error $||\mathcal{A}(t) - \mathcal{Y}(t)||$ over each time step for different runs of Algorithm 3.1 where the size of the core tensor varies between $[2, 2, 2]$, $[3, 2, 2]$, $[2, 3, 3]$ and $[3, 3, 3]$. Solving the nonlinear PDE using the low-rank tensor updating algorithm took 8.5 seconds for each of the different core sizes. As expected, the approximation error decreases as the size of the core tensor increases. Increasing the core tensor size beyond $[3, 3, 3]$ did not decrease the approximation error any further.

V. Conclusion

This paper presented an effective and novel low-rank tensor dynamical approximation scheme useful for electronic simulation. It starts with the Tucker decomposition of a tensor, which then facilitates multi-way operator splitting to construct nonlinear differential equations amenable to efficient solution. New operator splitting results in low-rank tensor manifold and tangent space are derived. This splitting of the projector operator for a 3-rd order tensor and the inverse free updating of the Tucker core and factors have to our knowledge not been described anywhere in the literature. Furthermore, this low-rank updating technique is very general and can be applied to all sorts of circuit simulations where the solution is of low rank nature. Numerical experiment has verified the excellent speed and accuracy of the proposed tensor-based simulation scheme.

Acknowledgment

This work was supported in part by the Hong Kong Research Grants Council under General Research Fund (GRF) Project 17208514, and the University Research Committee of The University of Hong Kong.

References

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

[2] 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.

[3] I. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[4] A. Ferreol, L. Albera, and P. Chevalier, "Fourth-order blind identification of underdetermined mixtures of sources (FOBIUM)," *IEEE Trans. Signal Process.*, vol. 53, no. 5, pp. 1640–1653, May 2005.

[5] 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.

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

[7] O. Koch and C. Lubich, "Dynamical low-rank approximation," *SIAM J. Matrix Anal. & Appl.*, vol. 29, no. 2, pp. 434–454, 2007.

[8] A. Nonnenmacher and C. Lubich, "Dynamical low-rank approximation: applications and numerical experiments," *Math. Comput. Simulation*, vol. 79, no. 4, pp. 1346–1357, Dec. 2008.

[9] O. Koch and C. Lubich, "Dynamical tensor approximation," *SIAM J. Matrix Anal. & Appl.*, vol. 31, no. 5, pp. 2360–2375, 2010.

[10] U. Shalit, D. Weinshall, and G. Chechik, "Online learning in the embedded manifold of low-rank matrices," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 429–458, January 2012.

[11] P. A. M. Dirac, "Note on exchange phenomena in the Thomas atom," *Math. Proc. Cambridge Philosophical Society*, vol. 26, no. 3, pp. 376–385, Jul. 1930.

[12] C. Lubich and I. V. Oseledets, "A projector-splitting integrator for dynamical low-rank approximation," *BIT Numerical Mathematics*, vol. 54, no. 1, pp. 171–188, Mar. 2014.

[13] Q. Chen, W. Schoenmaker, S.-H. Weng, C.-K. Cheng, G.-H. Chen, L. Jiang, and N. Wong, "A fast time-domain EM-TCAD coupled simulation framework via matrix exponential," in *ICCAD'12*, 2012, pp. 422–428.