FAST ADAPTIVE PARAFAC DECOMPOSITION ALGORITHM WITH LINEAR COMPLEXITY

Viet-Dung Nguyen[1] *Karim Abed-Meraim*[1] *Nguyen Linh-Trung*[2]

[1] PRISME Laboratory, University of Orléans, France, [2] VNU University of Engineering and Technology, Vietnam

ABSTRACT

We present a fast adaptive PARAFAC decomposition algorithm with low computational complexity. The proposed algorithm generalizes the Orthonormal Projection Approximation Subspace Tracking (OPAST) approach for tracking a class of third-order tensors which have one dimension growing with time. It has linear complexity, good convergence rate and good estimation accuracy. To deal with large-scale problems, a parallel implementation can be applied to reduce both computational complexity and storage. We illustrate the effectiveness of our algorithm in comparison with the state-of-the-art algorithms through simulation experiments.

Index Terms— Tensor decomposition, 3-way tensors, adaptive algorithm.

1. INTRODUCTION

Processing streaming massive volume data with time constraints is crucial in many applications such as communication [1], biomedical imaging [2], and statistical signal analysis [3]. The data might be in a multidimensional form (typically, one of the dimensions is time) and can be naturally presented by multiway arrays (tensors). Tensor decomposition thereby provides an important tool to analyze, understand or eventually compress the data.

One of the most widely-used tensor decomposition is the Parallel Factor Analysis (PARAFAC) because it can be considered as a generalization of the Singular Value Decomposition (SVD) for multiway arrays. Moreover, PARAFAC is unique under mild conditions (i.e., up to scale and permutation) without imposing constraints such as orthogonality, nonnegativity, or sparseness.

However, algorithms for PARAFAC are usually high computationally demanding. In an adaptive signal processing context, it leads to a need for fast tracking techniques. Surprisingly, there exist only a few adaptive PARAFAC algorithms in the literature. Adaptive algorithms for third-order tensors which have one dimension growing in time were first proposed by Nion and Sidiropoulos [1]. Their algorithms are adaptive techniques which are developed based on simultaneous diagonalization [4] (PARAFAC-SDT), and on minimization of weighted least squares criterion [5] (PARAFAC-RLST). Another recent work was also proposed by Mardani, Mateos and Giannakis [2], but for incomplete streaming instead of full data.

It is observed that the above-mentioned algorithms have quadratic computational complexity per time instant; that is, given a streaming tensor $\mathcal{X}(t) \in \mathbb{R}^{I\times J(t)\times K}$, where dimensions I and K are fixed and dimension J grows with time, their complexity is $O(IKR^2)$, where R is the tensor rank. This cost, however, might be a serious handicap in real-time applications.

In this paper, we generalize the orthonormal projection approximation subspace tracking (OPAST) approach [6] for tracking the loading factors (components) of third-order tensors which have one dimension growing with time. In particular, we efficiently exploit the slowly time-varying assumption and the Khatri-Rao product structure of the adaptive PARAFAC model to gain advantages in terms of estimation performance and computational complexity. As a result, the proposed algorithm, namely three-dimensional orthonormal projection approximation subspace tracking for third-order tensors (3D-OPAST), yields linear computational complexity in a standard setup while having comparable or even superior performance to the state-of-the-art algorithms.

2. BATCH AND ADAPTIVE PARAFAC

2.1. Batch PARAFAC

Consider a tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, its PARAFAC decomposition can be presented as

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \tag{1}$$

which is a sum of R rank-one tensors and R is called to be tensor rank. Here, the symbol $\circ$ defines the outer product. The set of vectors $\{\mathbf{a}_r\}$, $\{\mathbf{b}_r\}$ and $\{\mathbf{c}_r\}$ can be grouped into the so-called loading matrices $\mathbf{A} = [\mathbf{a}_1 \cdots \mathbf{a}_R] \in \mathbb{R}^{I\times R}$, $\mathbf{B} = [\mathbf{b}_1 \cdots \mathbf{b}_R] \in \mathbb{R}^{J\times R}$, and $\mathbf{C} = [\mathbf{c}_1 \cdots \mathbf{c}_R] \in \mathbb{R}^{K\times R}$. In practice, (1) is only an approximate tensor, i.e.,

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r + \mathcal{N}, \tag{2}$$

where $\mathcal{N}$ is a noise tensor. Thus, given a data tensor $\mathcal{X}$, the PARAFAC decomposition tries to achieve an R-rank best approximation. Equation (1) can also be rewritten in matrix form as

$$\mathbf{X}^{(1)} = (\mathbf{A} \odot \mathbf{C})\mathbf{B}^T, \tag{3}$$

where $\odot$ refers to the Khatri-Rao product (or the columnwise Kronecker product) and $\mathbf{X}^{(1)} \in \mathbb{R}^{IK\times J}$ whose entry $\mathbf{X}^{(1)}_{(i-1)K+k,j} = x_{ijk}$. We can write analogous expressions for $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(3)}$ [1]. The PARAFAC decomposition is generically unique if it satisfies the following condition [7], [1]:

$$2R(R-1) \le I(I-1)K(K-1), \qquad R \le J. \tag{4}$$

2.2. Adaptive PARAFAC

For adaptive PARAFAC decomposition of third-order tensors, the dimensions of $\mathcal{X}$ are growing with time; i.e., $\mathcal{X}(t) \in \mathbb{R}^{I(t)\times J(t)\times K(t)}$ in the general case. However, in this work, we consider the case where only one dimension changes in time, for example $J(t)$, while the other two dimensions are fixed.

To directly compare our proposed algorithm with those in [1], we follow their basic model and assumptions. Particularly, the matrix representation of the adaptive PARAFAC model is presented as

$$\mathbf{X}^{(1)}(t) \simeq \mathbf{H}(t)\mathbf{B}^T(t) \tag{5}$$

where $\mathbf{H}(t) = \mathbf{A}(t) \odot \mathbf{C}(t) \in \mathbb{R}^{IK\times R}$, $\mathbf{B}(t) \in \mathbb{R}^{J(t)\times R}$. Considering the model at two successive times, $t-1$ and t, we have

$$\mathbf{X}^{(1)}(t) = [\mathbf{X}^{(1)}(t-1), \mathbf{x}(t)], \tag{6}$$

where $\mathbf{x}(t) \in \mathbb{R}^{IK}$ is a vectorized representation of new data concatenated to $\mathbf{X}^{(1)}(t-1)$. If we assume that $\mathbf{A}$ and $\mathbf{C}$ follow an unknown slowly time-varying model (i.e., $\mathbf{A}(t) \simeq \mathbf{A}(t-1)$ and $\mathbf{C}(t) \simeq \mathbf{C}(t-1)$), then $\mathbf{H}(t) \simeq \mathbf{H}(t-1)$. Therefore,

$$\mathbf{B}^T(t) \simeq [\mathbf{B}^T(t-1), \mathbf{b}(t)^T]. \tag{7}$$

It means that, at each time, we only need to estimate the *row vector* $\mathbf{b}(t)$ and append it to $\mathbf{B}(t-1)$ instead of updating the whole $\mathbf{B}(t)$. We also assume that the tensor rank R is known, and at each time when a new data is added to the old tensor, the uniqueness of the new tensor fulfills (4). Now we have enough materials to present our algorithm.

3. ALGORITHM PRINCIPLE

Recall the matrix representation of the PARAFAC model in (3) without the superscript (1) for simplicity

$$\mathbf{X} = (\mathbf{A} \odot \mathbf{C})\mathbf{B}^T. \tag{8}$$

By using matrix factorization [4], (8) can be rewritten as

$$\mathbf{X} = \mathbf{W}\mathbf{E},$$

where

$$\mathbf{W} = (\mathbf{A} \odot \mathbf{C})\mathbf{Q}^{-1}, \quad \mathbf{E} = \mathbf{Q}\mathbf{B}^T,$$

for some nonsingular $\mathbf{Q}$. The objective is now to find the ambiguity matrix $\mathbf{Q}$ or $\mathbf{Q}^{-1}$ to recover the Khatri-Rao product structure $\mathbf{H} = \mathbf{A}\odot\mathbf{C}$. Matrix $\mathbf{Q}$ can be estimated by solving a simultaneous diagonalization [4], [8] or non-symmetric joint diagonalization [9]. Hence, $\mathbf{A}$ and $\mathbf{C}$ can be found, based on the following observation of $\mathbf{H}$:

$$\begin{aligned}\mathbf{H} = \mathbf{A} \odot \mathbf{C} &= [\mathbf{a}_1 \otimes \mathbf{c}_1 \cdots \mathbf{a}_R \otimes \mathbf{c}_R] \\ &= [\mathrm{vec}\{\mathbf{c}_1\mathbf{a}_1^T\} \cdots \mathrm{vec}\{\mathbf{c}_R\mathbf{a}_R^T\}].\end{aligned}$$

Since each column of $\mathbf{H}$ has the form of a vectorized rank-1 matrix, we can achieve $\mathbf{c}_i$ and $\mathbf{a}_i$ as left and right singular vectors of matrix $\mathbf{H}_i = \mathrm{unvec}(\mathbf{a}_i \otimes \mathbf{c}_i)$. It is straightforward to obtain $\mathbf{B}$ when either $\mathbf{Q}$ or $\mathbf{Q}^{-1}$ and $\mathbf{E}$ are available.

Table 1: OPAST algorithm

$$\begin{aligned}
\mathbf{y}(t) &= \mathbf{W}^H(t-1)\mathbf{x}(t) \\
\mathbf{q}(t) &= \frac{1}{\beta}\mathbf{Z}(t-1)\mathbf{y}(t) \\
\gamma &= 1/(1+\mathbf{y}^H(t)\mathbf{q}(t)) \\
\tau &= \frac{1}{\|\mathbf{q}(t)\|^2}\left(\frac{1}{\sqrt{1+\|\mathbf{q}(t)\|^2(\|\mathbf{x}(t)\|^2-\|\mathbf{y}(t)\|^2)}} - 1\right) \\
\mathbf{p}(t) &= \mathbf{W}(t-1)(\tau\mathbf{q}(t) - \gamma(1+\tau \parallel \mathbf{q}(t) \parallel^2)\mathbf{y}(t)) + (1+\tau \parallel \mathbf{q}(t) \parallel^2)\gamma\mathbf{x}(t) \\
\mathbf{Z}(t) &= \frac{1}{\beta}\mathbf{Z}(t-1) - \gamma\mathbf{q}(t)\mathbf{q}^H(t) \\
\mathbf{W}(t) &= \mathbf{W}(t-1) + \mathbf{p}(t)\mathbf{q}^H(t)
\end{aligned}$$

Because of its high complexity, applying the above procedure is not suitable for the adaptive model. In [1], one of the proposed methods requires sliding-window SVD tracking to estimate both the left and right orthogonal subspaces, $\mathbf{W}(t)$ and $\mathbf{E}(t)$, and then exploits the block common between $\mathbf{B}(t-1)$ and $\mathbf{B}(t)$ to construct the recursive update of $\mathbf{Q}(t)$ and $\mathbf{Q}^{-1}(t)$. We refer the reader to [1] for more details.

In this work, we only track the left orthogonal subspace $\mathbf{W}(t)$ using the OPAST method in [6], and then use orthonormal projection approximation of the two successive subspaces $\mathbf{W}(t-1)$ and $\mathbf{W}(t)$ to find out the recursive update of $\mathbf{H}(t)$ and $\mathbf{Q}(t)$. At each step, we preserve the Khatri-Rao product structure of $\mathbf{H}(t)$ by minimizing a cost function that measures the collinear deviation of sub-vectors inside each column of $\mathbf{H}(t)$. Then, $\mathbf{b}(t)$ is estimated by calculating the pseudo-inverse of $\mathbf{H}(t)$ exploiting its reduced rank update structure. As a consequence, the proposed algorithm has linear computational complexity of order $O(IKR)$ while sustaining good performance.

4. ALGORITHM DERIVATION

4.1. Overview of OPAST Principle

Consider the cost function

$$\begin{aligned}f(\mathbf{W}) &= E\{ \parallel \mathbf{x}(t) - \mathbf{W}\mathbf{W}\mathbf{x}(t) \parallel^2 \} \\ &= \mathrm{tr}\left\{\mathbf{R}_{xx} - 2\mathbf{W}^H\mathbf{R}_{xx}\mathbf{W} + \mathbf{W}^H\mathbf{R}_{xx}\mathbf{W}\mathbf{W}^H\mathbf{W}\right\},\end{aligned} \tag{9}$$

where $\mathbf{W}$ is an unitary matrix spanning the principal subspace of $\mathbf{R}_{xx} = E\{\mathbf{x}(t)\mathbf{x}^H(t)\}$. Minimizing (9) yields the abstract form of the OPAST [6] as follows (using informal notation)

$$\begin{aligned}\mathbf{W}(t) &= \mathbf{R}_{xx}\mathbf{W}(t-1)[\mathbf{W}^H(t-1)\mathbf{R}_{xx}\mathbf{W}(t-1)]^{-1} \\ \mathbf{W}(t) &= \mathbf{W}(t)[\mathbf{W}^H(t)\mathbf{W}(t)]^{-1/2}\end{aligned} \tag{10}$$

Equation (10) is an orthogonalization of $\mathbf{W}(t)$. By using the projection approximation, i.e., $\mathbf{W}(t) \simeq \mathbf{W}(t-1)$, a fast implementation of the algorithm has been derived in [6] and summarized in Table 1.

4.2. 3D-OPAST

In this section, we assume that $\mathbf{A}(t-1)$, $\mathbf{B}(t-1)$ and $\mathbf{C}(t-1)$ are available, as well as their related matrices $\mathbf{H}^{\#}(t-1)$ and

$\mathbf{H}(t-1)$. We then build on the projection approximation approach to construct the recursive update expressions for $\mathbf{A}(t)$, $\mathbf{B}(t)$ and $\mathbf{C}(t)$. Our algorithm can be summarized in four steps as follows: (i) given $\mathbf{x}(t)$, estimate $\mathbf{W}(t)$ using OPAST; (ii) estimate $\mathbf{H}(t)$ from $\mathbf{W}(t)$ and $\mathbf{Q}(t-1)$ by iteratively minimizing a criterion which measures its deviation from a Khatri-Rao structure; (iii) extract $\mathbf{A}(t)$ and $\mathbf{C}(t)$ from $\mathbf{H}(t)$; and (iv) update the estimate of $\mathbf{H}(t)$ and its pseudo-inverse and calculate $\mathbf{b}^T(t)$. We now consider these steps in detail.

Step 1: Estimate $\mathbf{W}(t)$

To estimate $\mathbf{W}(t)$, at time instant t, we run the OPAST algorithm as described in Table 1.

Step 2: Estimate $\mathbf{H}(t)$ from $\mathbf{W}(t)$ and $\mathbf{Q}(t-1)$

At this step, ideally, we want to recover $\mathbf{H}(t)$ which "preserves" the Khatri-Rao product structure. To this end, we first consider the following expression:

$$\begin{aligned}\mathbf{H}(t) &= \mathbf{W}(t)\mathbf{Q}(t) \\ &\simeq [\mathbf{W}(t-1) + \mathbf{p}(t)\mathbf{q}^T(t)][\mathbf{Q}(t-1)(\mathbf{I}+\boldsymbol{\varepsilon}(t)), \quad (11)\end{aligned}$$

where we use last equation of OPAST and the updating rule

$$\mathbf{Q}(t) \simeq \mathbf{Q}(t-1)(\mathbf{I}+\boldsymbol{\varepsilon}(t)), \quad (12)$$

where $\boldsymbol{\varepsilon}(t)$ is an $R \times R$ unknown matrix. The objective is to find $\boldsymbol{\varepsilon}(t)$ which imposes the Khatri-Rao product structure on $\mathbf{H}(t)$. We note that finding entries of $\boldsymbol{\varepsilon}(t)$ all at once leads to a computational complexity of order $O(IKR^2)$. Instead, to preserve the linear complexity, we choose $\boldsymbol{\varepsilon}(t)$ to be zero except for its j-th column vector ($j = t \mod (R)$), i.e.,

$$\boldsymbol{\varepsilon}(t) = [\mathbf{0} \quad \varepsilon_j \quad \mathbf{0}] = \varepsilon_j \mathbf{e}_j^T, \quad (13)$$

where ε_j is j-th column of $\boldsymbol{\varepsilon}(t)$ and $\mathbf{e}_j$ is the unit vector whose j-th entry is one. Substitute (13) into (11) yields

$$\mathbf{H}(t) \simeq \mathbf{M}(t) + (\mathbf{M}(t)\boldsymbol{\varepsilon}_j)\mathbf{e}_j^T, \quad (14)$$

where

$$\begin{aligned}\mathbf{M}(t) &= [\mathbf{W}(t-1) + \mathbf{p}(t)\mathbf{q}^T(t)][\mathbf{Q}(t-1) \\ &= \mathbf{H}(t-1) + \mathbf{p}(t)\tilde{\mathbf{q}}^T(t), \quad (15)\end{aligned}$$

with $\tilde{\mathbf{q}}(t) = \mathbf{Q}^T(t-1)\mathbf{q}(t)$. Therefore, only the j-th column of $\mathbf{H}(t)$ is affected by ε_j according to

$$\mathbf{h}_j(t) \simeq \mathbf{m}_j(t) + \mathbf{M}(t)\varepsilon_j \quad (16)$$

$$\begin{pmatrix} \mathbf{h}_j^1(t) \\ \cdots \\ \mathbf{h}_j^i(t) \\ \cdots \\ \mathbf{h}_j^I(t) \end{pmatrix} \simeq \begin{pmatrix} \mathbf{m}_j^1(t) + \mathbf{M}_j^1(t)\varepsilon_j \\ \cdots \\ \mathbf{m}_j^i(t) + \mathbf{M}_j^i(t)\varepsilon_j \\ \cdots \\ \mathbf{m}_j^I(t) + \mathbf{M}_j^I(t)\varepsilon_j \end{pmatrix} \quad (17)$$

where $\mathbf{h}_j^i(t)$ and $\mathbf{m}_j^i(t)$ ($i = 1, \ldots, I$) are $K \times 1$ sub-vectors of $\mathbf{h}_j(t)$ and $\mathbf{m}_j(t)$, and $\mathbf{M}_j^i(t)$ are sub-matrices of $\mathbf{M}(t)$.

Observed that two successive sub-vectors $\mathbf{h}_j^i(t)$ and $\mathbf{h}_j^{i+1}(t)$ are collinear by following the definition of $\mathbf{H}(t) = \mathbf{A}(t) \odot \mathbf{C}(t)$. Thus, preserving the collinear subvectors inside each column of $\mathbf{H}(t)$ corresponds to imposing the Khatri-Rao structure on $\mathbf{H}(t)$.

Before proceeding further, we define the following cost function which measures the deviation from the collinear condition of two vectors $\mathbf{u} = [u_1, \ldots, u_I]^T$ and $\mathbf{v} = [v_1, \ldots, v_I]^T$:

$$f(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{I-1} (u_i v_{i+1} - v_i u_{i+1})^2 + (u_I v_1 - v_I u_1)^2. \quad (18)$$

Let $\tilde{\mathbf{u}} = \mathbf{D}_I \mathbf{u} = [a_I, a_1, \ldots, a_{I-1}]^T$, where $\mathbf{D}_I$ is a downshift permutation matrix [10]. The cost function (18) can be written as $f(\mathbf{u}, \mathbf{v}) = \| \tilde{\mathbf{u}} * \mathbf{v} - \tilde{\mathbf{v}} * \mathbf{u} \|^2$, where $*$ refers to the Hadamard product (elementwise product). Thus, we minimize the following cost function:

$$\min_{\varepsilon_j} \sum_{i=1}^{I-1} \| \tilde{\mathbf{h}}_j^i * \mathbf{h}_j^{i+1} - \tilde{\mathbf{h}}_j^{i+1} * \mathbf{h}_j^i \|^2 + \| \tilde{\mathbf{h}}_j^I * \mathbf{h}_j^1 - \tilde{\mathbf{h}}_j^1 * \mathbf{h}_j^I \|^2. \quad (19)$$

To solve (19), we used a first order linearization (expansion) in term of ε_j. By this way, ε_j is computed by solving an $R \times R$ linear system[1]. Once we have ε_j, the j-th columns of $\mathbf{H}(t)$ and $\mathbf{Q}(t)$ are updated using (12) and (16) respectively.

Step 3: Extract $\mathbf{A}(t)$ and $\mathbf{C}(t)$ from $\mathbf{H}(t)$

At this step, we use the same method as [1] (i.e., a single Bi-SVD iteration [11]) to extract $\mathbf{A}(t)$ and $\mathbf{C}(t)$ from $\mathbf{H}(t)$

$$\mathbf{a}_i(t) = \mathbf{H}_i^T(t)\mathbf{c}_i(t-1), \quad \mathbf{c}_i(t) = \frac{\mathbf{H}_i(t)\mathbf{a}_i(t)}{\| \mathbf{H}_i(t)\mathbf{a}_i(t) \|}, \quad (20)$$

for $i = 1, \ldots, R$.

Step 4: Estimate $\mathbf{H}^{\#}(t)$ and $\mathbf{b}^T(t)$

By combining (13), (14) and (15), $\mathbf{H}(t)$ reveals a rank-2 update structure

$$\mathbf{H}(t) = \mathbf{H}(t-1) + \mathbf{p}(t)\tilde{\mathbf{q}}^T(t) + \mathbf{z}(t)\mathbf{e}_j^T,$$

where $\mathbf{z}(t) = \mathbf{M}(t)\varepsilon_j$. Thus, given the knowlege of $\mathbf{H}^{\#}(t-1)$, we can use the matrix inversion lemma to calculate $\mathbf{H}^{\#}(t)$ with a linear complexity.

Initialization

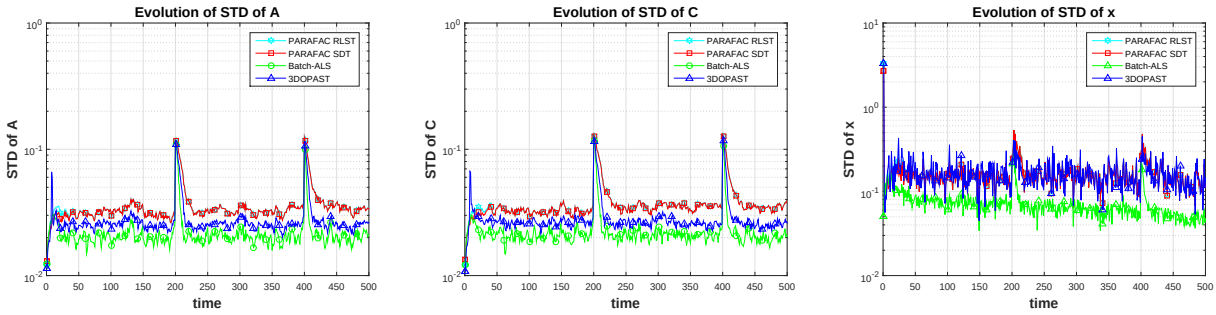
In our simulation, we choose to capture J_0 slices, then run the batch PARAFAC algorithm to obtain initial estimation. J_0 can be chosen to be the smallest number satisfying the uniqueness condition in (4).

Remark 1: A better estimate of $\mathbf{b}(t)$ and $\mathbf{H}^{\#}(t)$ can be obtained by using the Khatri-Rao product structure leading to

$$\mathbf{H}^{\#}(t) = [\mathbf{A}^T(t)\mathbf{A}(t) * \mathbf{C}^T(t)\mathbf{C}(t)]^{-1}[\mathbf{A}(t) \odot \mathbf{C}(t)]^T. \quad (21)$$

However, such an implementation costs $O(IKR^2)$ and is, thus, disregarded in this paper.

[1] Due to space limitation, the details of this linearization are omited.



(a) Loading matrix $\mathbf{A}(t)$ (b) Loading matrix $\mathbf{C}(t)$ (c) Observation vector $\mathbf{x}(t)$

Fig. 1: Performance comparison of 3D-OPAST with PARAFAC-RLST, PARAFAC-SDT and batch ALS.

Table 2: Experimental parameters

I	J_0	K	R	T	$\varepsilon_A, \varepsilon_C$	λ	σ^2
20	50	20	8	500	10^{-3}	0.8	10^{-3}

Remark 2: If R digital signal processor units are available, the matrix-vector product can be replaced by vector-vector products leading the cost of $O(IK)$ instead of $O(IKR)$. If we consider, for example, the implementation given by the first equation of Table 1, we can rewrite it as

$$y_i(t) = \mathbf{w}_i^H(t-1)\mathbf{x}(t), \qquad i = 1, \ldots, R. \tag{22}$$

The same procedure can be applied to other equations.

5. SIMULATION

In this section, we compare the performance of the proposed algorithm to that of the state-of-the-art algorithms in [1]. To have a fair comparison, we use the framework presented in [1][2]. In particular, a time-varying PARAFAC model is generated at each time instant as follows.

$$\mathbf{A}(t) = (1-\varepsilon_A)\mathbf{A}(t-1) + \varepsilon_A \mathbf{N}_A(t), \tag{23}$$
$$\mathbf{C}(t) = (1-\varepsilon_C)\mathbf{C}(t-1) + \varepsilon_C \mathbf{N}_C(t), \tag{24}$$

where $\mathbf{N}_A(t)$ and $\mathbf{N}_C(t)$ are random matrices which have the same size as $\mathbf{A}(t)$ and $\mathbf{C}(t)$ correspondingly, ε_A and ε_C control the speed of variation for $\mathbf{A}(t)$ and $\mathbf{C}(t)$ between two successive observations. A vector $\mathbf{b}(t)$ is generated with i.i.d. Gaussian entries. Then, the input data $\mathbf{x}(t)$ is given by

$$\mathbf{x}(t) = [\mathbf{A}(t) \odot \mathbf{C}(t)]\mathbf{b}^T(t).$$

Thus, this observation vector follows the model and the assumptions in Section 2.2. Then, the noisy observation is $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) + \sigma^2\mathbf{n}(t)$, where $\mathbf{n}(t)$ is the noise vector and parameter σ controls the noise level. When comparing performance of the proposed algorithms with the algorithms in [1], we keep all their default parameters as offered by the authors. A summary of the parameters used in the experiments is given in Table 2.

Performance criterion for the estimation of $\mathbf{A}(t)$ and $\mathbf{C}(t)$, is measured by the standard deviation (STD) between the true loading matrix and its estimation up to scale and permutation at each time

$$\text{STD}_{\mathbf{A}}(t) = ||\mathbf{A}(t) - \mathbf{A}_{es}(t)||_F, \tag{25}$$
$$\text{STD}_{\mathbf{C}}(t) = ||\mathbf{C}(t) - \mathbf{C}_{es}(t)||_F. \tag{26}$$

For $\mathbf{B}(t)$, because of the time-shift structure as presented before, we verify its performance through $\mathbf{x}(t)$, as

$$\text{STD}_{\mathbf{B}(t)} = ||\mathbf{x}(t) - \mathbf{x}_{\text{es}}(t)||_2. \tag{27}$$

To highlight the convergence rate of all compared algorithms, we design an experiment as follows: the speed of variation of $\mathbf{A}(t)$ and $\mathbf{C}(t)$ (i.e., the value of ε_A and ε_C) abruptly changes at specific time instants while being remained constant in other time instants. This experiment is similar to the jump scenario to assess the performance of subspace tracking [12], [13]. We compare four algorithms, PARAFAC-RLST, PARAFAC-SDT, 3D-OPAST (exponential window is used for all algorithms) and batch Alternating Least Square (ALS). The batch ALS is applied repeatedly to streaming tensors and only serves as "lower bound" for comparison. Among four algorithms, 3D-OPAST outperforms PARAFAC-RLST and PARAFAC-SDT and have the performance close to that of the batch ALS as indicated in Figure 1. Again, we note that the computational complexity of 3D-OPAST is $O(IKR)$ as compared to $O(IKR^2)$ of PARAFAC-RLST and PARAFAC-SDT. Moreover, 3D-OPAST also has a better convergence rate than PARAFAC-RLST and PARAFAC-SDT, as shown by plots at the abrupt change instant.

6. CONCLUSION

In this paper, we have presented a new adaptive PARAFAC decomposition algorithm which has a linear computational complexity. Surprisingly, while enjoying its lower complexity, the proposed algorithm also shows a good performance in terms of estimation accuracy and tracking ability.

ACKNOWLEDGMENT

This work was supported by the National Foundation for Science and Technology Development (NAFOSTED) of Vietnam under Grant No. 102.02-2015.32.

[2]Matlab program from http://dimitri.nion.free.fr

7. REFERENCES

[1] D.Nion and N.D.Sidiropoulos, "Adaptive algorithms to track the parafac decomposition of a third-order tensor," *Signal Processing, IEEE Transactions on*, vol. 57, no. 6, pp. 2299–2310, 2009.

[2] M. Mardani, G. Mateos, and G. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *Signal Processing, IEEE Transactions on*, vol. 63, no. 10, pp. 2663–2677, 2015.

[3] K. Slavakis, G. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge," *Signal Processing Magazine, IEEE*, vol. 31, no. 5, pp. 18–31, 2014.

[4] L. De Lathauwer, "A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization," *SIAM Journal on Matrix Analysis and Applications*, vol. 28, no. 3, pp. 642–666, 2006.

[5] S. Haykin, *Adaptive filter theory*, Pearson Education India, 2008.

[6] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *Signal Processing Letters, IEEE*, vol. 7, no. 3, pp. 60–62, 2000.

[7] 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.

[8] F. Roemer and M. Haardt, "A closed-form solution for parallel factor (PARAFAC) analysis," in *Proc. IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP 2008)*, Las Vegas, NV, Apr. 2008, pp. 2365 – 2368.

[9] VD. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Parallelizable PARAFAC decomposition of 3-way tensors," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, 2015.

[10] G. H. Golub and C. F. Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.

[11] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *Signal Processing, IEEE Transactions on*, vol. 45, no. 5, pp. 1222–1240, 1997.

[12] P. Strobach, "Fast recursive subspace adaptive ESPRIT algorithms," *Signal Processing, IEEE Transactions on*, vol. 46, no. 9, pp. 2413–2430, 1998.

[13] R. Badeau, G. Richard, and B. David, "Fast and stable YAST algorithm for principal and minor subspace tracking," *Signal Processing, IEEE Transactions on*, vol. 56, no. 8, pp. 3437–3446, 2008.