Tensor Networks for Latent Variable Analysis: Novel Algorithms for Tensor Train Approximation

Anh-Huy Phan, *Member, IEEE*, Andrzej Cichocki, *Fellow, IEEE*, André Uschmajew, Petr Tichavský, *Senior Member, IEEE*, George Luta, and Danilo P. Mandic, *Fellow, IEEE*

***Abstract*— Decompositions of tensors into factor matrices, which interact through a core tensor, have found numerous applications in signal processing and machine learning. A more general tensor model that represents data as an ordered network of subtensors of order-2 or order-3 has, so far, not been widely considered in these fields, although this so-called tensor network (TN) decomposition has been long studied in quantum physics and scientific computing. In this article, we present novel algorithms and applications of TN decompositions, with a particular focus on the tensor train (TT) decomposition and its variants. The novel algorithms developed for the TT decomposition update, in an alternating way, one or several core tensors at each iteration and exhibit enhanced mathematical tractability and scalability for large-scale data tensors. For rigor, the cases of the given ranks, given approximation error, and the given error bound are all considered. The proposed algorithms provide well-balanced TT-decompositions and are tested in the classic paradigms of blind source separation from a single mixture, denoising, and feature extraction, achieving superior performance over the widely used truncated algorithms for TT decomposition.**

***Index Terms*— Blind source separation, image denoising, nested Tucker, tensor network (TN), tensor train (TT) decomposition, tensorization, Tucker-2 (TK2) decomposition, truncated singular value decomposition (SVD).**

Manuscript received October 4, 2018; revised July 19, 2019; accepted November 13, 2019. Date of publication February 5, 2020; date of current version October 29, 2020. The work of A.-H. Phan and A. Cichocki was supported by the Ministry of Education and Science of the Russian Federation under Grant 14.756.31.0001. The work of P. Tichavský was supported by the Czech Science Foundation under Project 17-00902S. *(Corresponding author: Anh-Huy Phan.)*

A.-H. Phan is with the Skolkovo Institute of Science and Technology (Skoltech), 143026 Moscow, Russia, and also with the Tokyo University of Agriculture and Technology, Tokyo 183-8538, Japan (e-mail: a.phan@skoltech.ru).

A. Cichocki is with the Skolkovo Institute of Science and Technology (Skoltech), 143026 Moscow, Russia, also with the College of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, China, also with the Tokyo University of Agriculture and Technology, Tokyo, Japan, and also with the Systems Research Institute, 01-447 Warsav, Poland (e-mail: a.cichocki@skoltech.ru).

A. Uschmajew is with the Max Planck Institute for Mathematics in the Sciences, 04103 Leipzig, Germany (e-mail: uschmajew@mis.mpg.de).

P. Tichavský is with the Czech Academy of Sciences, Institute of Information Theory and Automation, 182 00 Prague, Czech Republic (e-mail: tichavsk@utia.cas.cz).

G. Luta is with the Lombardi Comprehensive Cancer Center, Georgetown University, Washington, DC 20057 USA (e-mail: george.luta@georgetown.edu).

D. P. Mandic is with the Imperial College, London SW7 2AZ, U.K. (e-mail: d.mandic@imperial.ac.uk).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2956926

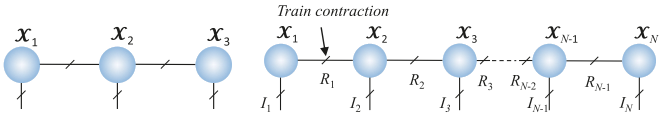

Fig. 1. Graphical illustration of a TK2 tensor (left) and a TT-tensor (right) $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$. A node represents a third-order core tensor $\mathcal{X}_n$.

I. Introduction

TENSOR decompositions (TDs) are rapidly finding application in signal processing paradigms, including the identification of independent components in multivariate data through higher order cumulant tensors, signal retrieval in code-division multiple access (CDMA) telecommunications, extraction of hidden components from neural data, training of dictionaries in supervised learning systems, image completion, and various tracking scenarios. Most current applications are based on the CANDECOMP/PARAFAC (CPD) [1]–[5] and Tucker decompositions [6], while their variants, such as the parallel profiles with linear dependences (PARALIND), the combination between PARAFAC and Tucker decompositions (PARATUCK) [7], [8] or Block Term decompositions [9], and the tensor deflation or tensor rank splitting [10], [11], were developed for a specific task (for a review, see [12]–[14] and the references therein).

Within the TDs, the data tensor is factorized into a set of factor matrices and a core tensor, the entries of which model interaction between the factor matrices. Such TDs are the natural extensions of matrix factorizations and allow for most two-way factor analysis methods to be generalized to their multiway analysis counterparts. However, despite mathematical elegance, the aforementioned TDs easily become computationally intractable or even ill-conditioned when applied to higher order tensors.

To help resolve these issues, which are a critical obstacle in a more widespread use of TDs in practical applications, we here consider another kind of tensor approximation, whereby multiple small core tensors are interconnected to construct an ordered network. More specifically, we focus on the tensor train (TT) decomposition, in which core tensors connect to only one or two other cores (see the illustration in Fig. 1), so that the tensor network (TN) appears as a "train" of tensors [15]. The TT-decomposition has been brought

into the TD community through the work of Oseledets and Tyrtyshnikov [15], although the model itself was developed earlier in quantum computation and chemistry under the name of the matrix product states (MPSs) [16], [17]. Compared with the rank issues in standard TDs, the ranks in the TT-decomposition can be determined in a stable way, e.g., through a rank reduction using truncated singular value decomposition (SVD). Moreover, by casting the data into the TT-format, the paradigms such as solving a huge system of linear equations or eigenvalue decomposition (EVD) of large-scale data can be reduced to solving the smaller scale subproblems of the same kind [18], [19]. Owing to the enhanced tractability in computation, the Hierarchical Tucker format and the TTs have also been successfully used for tensor completion in, e.g., seismic data analysis and parametric PDEs [20]–[22]. Despite success, TT-decomposition as well as other TNs are yet to gain the same popularity in signal processing and machine learning as the standard CPD and Tucker decompositions. To this end, this article aims to address this void in the literature and present applications of TNs, a framework that can serve for the separation of signals even from a single data channel.

As with many other TDs, the basic problem in the TT-decomposition is to find an optimal representation for the cases

- when the TT-ranks are given or
- when the approximation error is prescribed.

Existing algorithms for TT-decomposition are based on truncated SVD and sequential projection (SeqProj) (TT-SVD) [15], [17], [23], whereby the core tensors are composed from the leading singular vectors of the projected data onto the subspace of the other core tensors. This method is simple and works efficiently when data are amenable to the so-imposed strict models, as is the case in quantum physics. However, for general data, the ranks are not known beforehand so that the truncation algorithm is less efficient, and consequently, the TT solutions do not achieve the optimal approximation error. More importantly, TT-SVD often yields models with badly unbalanced ranks and is not guaranteed to yield a tensor with a minimal total TT-rank or a minimal number of parameters.

In this article, we introduce novel algorithms to approximate a large-scale tensor by TT-tensors, with a particular emphasis on the stability and minimum number of parameters. This is achieved based on an alternating update scheme, which sequentially updates one, two, or three core tensors at a time. The main technique also rests upon the Tucker-2 (TK2) decomposition, a simple case of the TT-decomposition for the order-3 tensors. Our proposed algorithms are particularly suited for low-rank approximations with an exact error bound.

This article is organized as follows. The TT-tensors and tensor operators are introduced in Section II. The TT-SVD algorithm is elaborated in Section III. Algorithms for TK2 are presented in Section IV and are used as a basic tool for the TT-decomposition of higher order tensors. Section V presents algorithms for the cases when the TT-rank is specified or when the noise level is given. Section VI shows that the decompositions can perform even faster when a data tensor is replaced by its crude TT-approximation. The proposed suite of algorithms for TT-decomposition is verified by simulations on signal and image denoising and latent variable analysis.

II. Preliminaries

We shall next present the definitions of tensor contraction, TT decomposition, and orthogonalization for a TT. The following three-tensor contractions are defined for an order-N tensor, $\mathcal{A}$, of size $I_1 \times I_2 \times \cdots \times I_N$ and an order-K tensor, $\mathcal{B}$, of size $J_1 \times J_2 \times \cdots \times J_K$.

Definition 1 (TT Contraction): The train contraction performs a tensor contraction between the last mode of tensor $\mathcal{A}$ and the first mode of tensor $\mathcal{B}$, where $I_N = J_1$, to yield a tensor $\mathcal{C} = \mathcal{A} \bullet \mathcal{B}$ of size $I_1 \times \cdots \times I_{N-1} \times J_2 \times \cdots \times J_K$, the elements of which are given by $c_{i_1,\ldots,i_{N-1},j_2,\ldots,j_K} = \sum_{i_N=1}^{I_N} a_{i_1,\ldots,i_{N-1},i_N} b_{i_N,j_2,\ldots,j_K}$.
Fig. 1 illustrates the principle of the train contraction.

Definition 2 (Left and Right Contractions): The n-mode left contraction between two tensors $\mathcal{A}$ and $\mathcal{B}$ is denoted by $\mathcal{C}_L = \mathcal{A} \ltimes_n \mathcal{B}$ and computes the contraction product between their first n modes to yield a tensor $\mathcal{C}_L$ of size $I_{n+1} \times \cdots \times I_N \times J_{n+1} \times \cdots \times J_K$. The right contraction denoted by $\mathcal{C}_R = \mathcal{A} \rtimes_n \mathcal{B}$ computes the contraction product between their last n modes and yields a tensor $\mathcal{C}_R$ of size $I_1 \times \cdots \times I_{N-n} \times J_1 \times \cdots \times J_{K-n}$.

Definition 3 (TT Decomposition or TT-Format [15], [17]): The TT decomposition of a tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$, with a TT-rank $(R_1, R_2, \ldots, R_{N-1})$, has the form

$$\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_{N-1} \bullet \mathcal{X}_N$$

where $\mathcal{X}_n$ are the core tensors of size $R_{n-1} \times I_n \times R_n$ and $R_0 = R_N = 1$.

A tensor $\mathcal{X}$ in the TT-format is called a TT-tensor and can be expressed equivalently through a product of its sub TT-tensors $\mathcal{X} = \mathcal{X}_{<n} \bullet \mathcal{X}_{n:m} \bullet \mathcal{X}_{>m}$, where $\mathcal{X}_{<n}$ and $\mathcal{X}_{>m}$ are, respectively, the TT-tensors composed by all core tensors to the left of $\mathcal{X}_n$ and to the right of $\mathcal{X}_m$, $m \geq n$, that is

$$\begin{aligned} \mathcal{X}_{<n} &= \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_{n-1} \\ \mathcal{X}_{>m} &= \mathcal{X}_{m+1} \bullet \mathcal{X}_{m+2} \bullet \cdots \bullet \mathcal{X}_N \\ \mathcal{X}_{n:m} &= \mathcal{X}_n \bullet \mathcal{X}_{n+1} \bullet \cdots \bullet \mathcal{X}_m. \end{aligned}$$

Note that a TT-tensor can always be compressed, e.g., using the TT-SVD algorithm (see Algorithm 1) with a perfect accuracy, $\epsilon = 0$, such that the ranks satisfy

$$R_n \leq \min(R_{n-1} I_n, I_{n+1} R_{n+1}) \quad \text{or} \quad R_n \leq \min\left(\prod_{k=1}^{n} I_k, \prod_{l=n+1}^{N} I_l\right)$$

for $n = 1, 2, \ldots, N-1$. The first inequalities above imply the second ones.

Definition 4 (Tensor Unfolding): The mode-$(\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_J)$ unfolding converts a tensor $\mathcal{X}$ into a tensor $\mathcal{Y} = [\mathcal{X}]_{(\boldsymbol{n}_1,\boldsymbol{n}_2,\ldots,\boldsymbol{n}_J)}$, given by $\mathcal{X}(i_1, i_2, \ldots, i_N) = \mathcal{Y}(i_{\boldsymbol{n}_1}, i_{\boldsymbol{n}_2}, \ldots, i_{\boldsymbol{n}_J})$, where $i_{\boldsymbol{n}_j}$ is a linear index of $(i_{n_j(1)}, i_{n_j(2)}, \ldots, i_{n_j(K_j)})$ [1], $K_j = card(n_j)$. The mode-$\boldsymbol{n}$ matricization is $\mathbf{X}_{(\boldsymbol{n})} = [\mathcal{X}]_{(\boldsymbol{n},\{1,\ldots,N\}\setminus\boldsymbol{n})}$.

Definition 5 (Left- and Right-Orthogonality Conditions [18], [24]): A core tensor $\mathcal{X}_n$ of a TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ is said to satisfy the left-orthogonality condition

if $\mathcal{X}_n \ltimes_2 \mathcal{X}_n = \mathbf{I}_{R_n}$ and the right-orthogonality condition if $\mathcal{X}_n \rtimes_2 \mathcal{X}_n = \mathbf{I}_{R_{n-1}}$.

The mode-n left-orthogonalization is achieved using the orthogonal Tucker-1 decomposition of $\mathcal{X}_n$ in the form $\mathcal{X}_n = \tilde{\mathcal{X}}_n \bullet \mathbf{L}$ or from the QR decomposition of the mode-(1,2) matricization $[\mathcal{X}_n]_{(1,2)} = \mathbf{Q}\,\mathbf{R}$, where $[\tilde{\mathcal{X}}_n]_{(1,2)} = \mathbf{Q}$ is an orthogonal matrix, and $\mathbf{L} = \mathbf{R}$. The TT-tensor $\mathcal{X}$ now becomes

$$\mathcal{X} = \mathcal{X}_1 \bullet \cdots \bullet \mathcal{X}_{n-1} \bullet \tilde{\mathcal{X}}_n \bullet (\mathbf{L} \bullet \mathcal{X}_{n+1}) \bullet \cdots \bullet \mathcal{X}_N.$$

Similarly, the mode-n right-orthogonalization performs the orthogonal Tucker-1 decomposition $\mathcal{X}_n = \mathbf{R} \bullet \tilde{\mathcal{X}}_n$ and results in

$$\mathcal{X} = \mathcal{X}_1 \bullet \cdots \bullet (\mathcal{X}_{n-1} \bullet \mathbf{R}) \bullet \tilde{\mathcal{X}}_n \bullet \mathcal{X}_{n+1} \bullet \cdots \bullet \mathcal{X}_N\,.$$

Left-orthogonalization up to mode-n performs $(n-1)$ left-orthogonalizations of the core tensors to the left of n such that $\mathcal{X}_k \ltimes_2 \mathcal{X}_k = \mathbf{I}_{R_k}$ for $k = 1, 2, \ldots, n-1$.

Right-orthogonalization up to mode-n performs $(N-n)$ right-orthogonalizations of the core tensors to the right of n such that $\mathcal{X}_k \rtimes_2 \mathcal{X}_k = \mathbf{I}_{R_{k-1}}$ for $k = n+1, n+2, \ldots, N$.

In this article, we consider the following two approximations of a tensor $\mathcal{Y}$ by a tensor $\mathcal{X}$ in the TT-format.

1) **The TT-approximation with a given TT-rank** based on a minimization in the form

$$\min \quad D = \|\mathcal{Y} - \mathcal{X}\|_F^2. \tag{1}$$

2) **The TT-approximation with a given approximation accuracy (denoising problem)** based on the rank minimization problem with the error-bound constraint

$$\min \quad \sum_{n=1}^{N-1} R_n \quad \text{s.t.} \quad \|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon^2 \tag{2}$$

such that the estimated TT-tensor $\mathcal{X}$ should have minimum total TT-rank or minimum number of parameters

$$\min \quad \sum_{n=1}^{N} I_n R_{n-1} R_n \quad \text{s.t.} \quad \|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon^2 \tag{3}$$

where ε^2 represents the approximation accuracy. We also address the case with an exact error bound, that is, (3) with an equality constraint.

III. TT-SVD, TT-Truncation and Density Matrix Renormalization Group (DMRG) Algorithms

In many practical settings, the TT-decomposition can be performed efficiently using an SeqProj and truncation algorithm, known as the TT-SVD [15], [17], [23]. More specifically, the first core $\mathcal{X}_1$ is obtained from the R_1 leading singular vectors of the reshaping matrix $\mathbf{Y}_{(1)}$, subject to the error norm being less than ϵ times the data norm, that is

$$\|\mathbf{Y}_{(1)} - \mathbf{U}\ \mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{V}^T\|_F^2 \le \epsilon^2\,\|\mathbf{Y}_{(1)}\|_F^2 \tag{4}$$

or $\|\boldsymbol{\sigma}\|_2^2 \ge (1-\epsilon^2)\|\mathbf{Y}_{(1)}\|_F^2$. The projected data $\mathrm{diag}(\boldsymbol{\sigma})\mathbf{V}^T$ are then reshaped into a matrix $\mathbf{Y}_2$ of size $(R_1\ I_2) \times (I_3 I_4 \cdots I_N)$, and the second core tensor $\mathcal{X}_2$ is estimated from the leading left singular vectors of this matrix, whereas the rank R_2 is chosen such that the norm of the residual is less than $(1-\epsilon^2)^{1/2}\|\mathbf{Y}_2\|_F$.

Algorithm 1 TT-SVD [15], [17]

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, TT-rank $(R_1, R_2, \ldots, R_{N-1})$ or approximation accuracy ϵ
Output: A TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ such that $\min \|\mathcal{Y} - \mathcal{X}\|_F^2$ or $\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \epsilon^2 \|\mathcal{Y}\|_F^2$

begin
  for $n = 1, \ldots, N-1$ **do**
1 $\mathbf{Y} = \mathtt{reshape}(\mathcal{Y}, (I_n\, R_{n-1}) \times \prod_{k=n+1}^{N} I_k)$
2 Truncated SVD $\mathbf{Y} \approx \mathbf{U}\ \mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{V}^T$ with given rank R_n or such that $\|\boldsymbol{\sigma}\|_2^2 \ge (1-\epsilon^2)\|\mathbf{Y}\|_F^2$
3 $\mathcal{X}_n = \mathtt{reshape}(\mathbf{U}, R_{n-1} \times I_n \times R_n)$
4 $\mathcal{Y} \leftarrow \mathrm{diag}(\boldsymbol{\sigma})\,\mathbf{V}^T$
5 $\mathcal{X}_N = \mathcal{Y}$

The SeqProj and truncation procedure is repeated in order to find the remaining core tensors. The algorithm, summarized in Algorithm 1, executes only $(N-1)$ sequential data projections and $(N-1)$ truncated-SVDs in order to estimate N core tensors. For the exact error bound case, we can use the result in Supplementary material.

Rounding Operation: The TT-SVD algorithm can be modified for the decomposition with TT-ranks specified and can be efficiently implemented if the input is already provided in the TT-format and is used for further truncation. The rounding operation is illustrated in Example III. In terms of the approximation accuracy, it can be shown that [15]

$$\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \sum_{k=1}^{N-1} \epsilon_k^2$$

where ϵ_k is the truncation error at the kth step.

When the data are subject to small noise and admit the TT-format, the TT-SVD works well; however, the algorithm is less efficient when data are heavily corrupted by noise or when the approximation is a relatively small rank model.

Remark 1: For the approximation with a given TT-rank in (1), TT-SVD is not guaranteed to achieve the minimum approximation error, as illustrated in Examples II, III, and VI.

For the denoising problem in (2), the resulting TT-tensor from TT-SVD satisfies the error bound, but often exhibits badly unbalanced TT-ranks.

Remark 2: In the presence of noise, an increase in the TT-rank of $\mathcal{X}$ will make it easier to explain the data, so that the approximation error will eventually become smaller than the tolerance error ε^2. However, when the TT-ranks are too high for the problem at hand, adding more terms into $\mathcal{X}$ implies adding noise into the approximation, thus reducing the reconstruction error. In other words, TT-SVD tends to select higher TT-ranks than that needed for the decomposition problem with the bounded error. This is illustrated in Example IV.

DMRG algorithm is another algorithm for TT-decomposition [18], [25], [26], which works as an alternating least squares algorithm. Each time it solves a minimization problem over two consecutive core tensors, $\mathcal{X}_n$ and $\mathcal{X}_{n+1}$, by means of SVD, then updates $\mathcal{X}_{n+1}$ and $\mathcal{X}_{n+2}$ and so on. Like the TT-SVD, DMRG can determine the TT-ranks based on singular values. However, both algorithms aim at minimizing the approximation error, but are not formulated for the decomposition with a given error bound. These algorithms are best suited to the decomposition with given ranks or when the error bound is negligible.

Following Remarks 1 and 2, Sections IV and V present more efficient algorithms for the two approximation problems in (1) and (2).

IV. TK2: TT-Decomposition for Order-3 Tensors

Before presenting algorithms for the TT-decomposition, we shall start with the decomposition for order-3 tensors and illuminate its relation to the TK2. The algorithm developed in this section will serve as a basis for updating core tensors in the TT-decompositions of higher order tensors.

Definition 6: **TK2 decomposition [27]** of an order-3 tensor, $\mathcal{Y}$, of size $I_1 \times I_2 \times I_3$ is given by

$$\mathcal{Y} = \mathbf{X}_1 \bullet \mathcal{X}_2 \bullet \mathbf{X}_3 \tag{5}$$

where $\mathcal{X}_2$ is the core tensor of size $R_1 \times I_2 \times R_2$, $\mathbf{X}_1$ and $\mathbf{X}_3$ are the two factor matrices of sizes $I_1 \times R_1$ and $R_2 \times I_3$, respectively, while the TT-rank of the decomposition is (R_1, R_2).

By definition, the TK2 decomposition is a TT-decomposition of an order-3 tensor (see also Fig. 1). Because of rotational ambiguity, without loss in generality, the matrices $\mathbf{X}_1$ and $\mathbf{X}_3$ can be assumed to have orthonormal columns (for $\mathbf{X}_1$) and rows (for $\mathbf{X}_3$), that is, $\mathbf{X}_1^T\mathbf{X}_1 = \mathbf{I}_{R_1}$ and $\mathbf{X}_3\mathbf{X}_3^T = \mathbf{I}_{R_2}$. We will show that for both problems (1) and (3), the two core tensors $\mathbf{X}_1$ and $\mathbf{X}_3$ are sequentially estimated by means of EVD, while $\mathcal{X}_2$ need not be estimated explicitly.

A. TT-Decomposition With Given Rank

The optimal core tensor $\mathcal{X}_2$ is $\mathcal{X}_2^\star = \mathbf{X}_1^T \bullet \mathcal{Y} \bullet \mathbf{X}_3^T$, and the Frobenius norm of the approximation error is given by

$$\begin{aligned} D &= \left\|\mathbf{Y} - \mathbf{X}_1 \bullet \mathcal{X}_2^\star \bullet \mathbf{X}_3\right\|_F^2 = \|\mathcal{Y}\|_F^2 - \left\|\mathcal{X}_2^\star\right\|_F^2 \\ &= \|\mathcal{Y}\|_F^2 - \operatorname{tr}\left(\left(\mathbf{X}_3^T \otimes \mathbf{X}_1\right)^T \mathbf{Q}\left(\mathbf{X}_3^T \otimes \mathbf{X}_1\right)\right) \quad (6) \\ &= \|\mathcal{Y}\|_F^2 - \operatorname{tr}\left(\mathbf{X}_1^T\,\mathbf{Q}_1\,\mathbf{X}_1\right) = \|\mathcal{Y}\|_F^2 - \operatorname{tr}\left(\mathbf{X}_3^T\,\mathbf{Q}_3\,\mathbf{X}_3\right) \quad (7) \end{aligned}$$

where the matrix $\mathbf{Q} = \mathbf{Y}_{(2)}^T\mathbf{Y}_{(2)} = [\Omega]_{(1,2)}$ is the mode-(1,2) unfolding of a tensor Ω of size $I_1 \times I_3 \times I_1 \times I_3$, and $\mathbf{Q}_1$ and $\mathbf{Q}_3$ are, respectively, the matrices of size $I_1 \times I_1$ and $I_3 \times I_3$

$$\begin{aligned} \mathbf{Q}_1(i,j) &= \sum_{r=1}^{R_2} \mathbf{X}_3(r,:)\,\Omega(i,:,j,:)\,\mathbf{X}_3^T(r,:) \\ \mathbf{Q}_3(i,j) &= \sum_{r=1}^{R_1} \mathbf{X}_1(:,r)^T\,\Omega(:,i,:,j)\,\mathbf{X}_1(:,r). \end{aligned}$$

The new estimate $\mathbf{X}_1^*$ in the problem (7) comprises R_1 principal eigenvectors of $\mathbf{Q}_1$, while $\mathbf{X}_3^*$ comprises the R_2 principal eigenvectors of the matrix $\mathbf{Q}_3$. When $I_2 \gg I_1\ I_3$, we need not process the original tensor $\mathcal{Y}$ but only the matrix $\mathbf{Q}$ of size $I_1\ I_3 \times I_1\ I_3$. The algorithm works like the higher order orthogonal iteration algorithm [6], while its global convergence is assured under the assumption that the R_nth and (R_n+1)th largest eigenvalues of $\mathbf{Q}_1$ and $\mathbf{Q}_3$ are nonidentical; otherwise, the dominant invariant subspace is not unique [28].

B. TT-Decomposition With Error Bound Constraint

We seek a model with a minimum number of parameters

$$\begin{aligned} &\min\ \ I_1R_1 + I_3R_2 + I_2R_1R_2 \\ &\text{s.t.}\ \ \|\mathcal{Y} - \mathbf{X}_1 \bullet \mathcal{X}_2 \bullet \mathbf{X}_3\|_F^2 \le \varepsilon^2, \quad \mathbf{X}_1^T\mathbf{X}_1 = \mathbf{I}_{R_1},\ \mathbf{X}_3\mathbf{X}_3^T = \mathbf{I}_{R_2}. \end{aligned} \tag{8}$$

We will show that the factor matrices $\mathbf{X}_1$ and $\mathbf{X}_3$ can be updated in a similar fashion to the previous case.

Lemma 1: In the optimization problem (8), the core tensor $\mathcal{X}_2$ can be eliminated using $\mathcal{X}_2^\star = \mathbf{X}_1^T \bullet \mathcal{Y} \bullet \mathbf{X}_3^T$ without changing the minimal achievable number of parameters. Moreover, for the fixed rank parameters, this choice provides a minimal error bound. Proof is provided in Supplementary Material [39].

For the TK2 with an exact error bound, the optimal $\mathbf{X}_1^*$ is found from the EVD of $\mathbf{Q}_1$, as stated in Supplementary Material. In practice, we can solve the decomposition with the inequality constraint and then enforce the equality constraint in the last iteration.

Similar to the update of $\mathbf{X}_1$, the matrix $\mathbf{X}_3$ comprises R_2 principal eigenvectors of the matrix $\mathbf{Q}_3$, where R_2 is either given or determined based on the bound $\|\mathcal{Y}\|_F^2 - \varepsilon^2$. The algorithm for TK2 sequentially updates $\mathbf{X}_1$ and $\mathbf{X}_3$.

C. Initialization and Refining Procedure

For the TK-2 decomposition with a given rank, the most efficient initialization method is based on the singular vectors of tensor unfoldings or the SeqProj like in TT-SVD.

For the decomposition with an error bound constraint, we adopt the TT-SVD SeqProj method, e.g., initialize $\mathbf{X}_3$ as an identity matrix and then estimate $\mathbf{X}_1$. The core $\mathbf{X}_3$ will be updated in the next update circle. The algorithm can converge quickly but often false to local minima, because the rank of $\mathbf{X}_1$ is often overestimated and relatively small in the first update, and then, $\mathbf{X}_3$ is estimated with a higher rank than the optimal one.

Truncated HOSVD: More efficiently, we apply the higher order SVD [6] to find a truncated model with the smallest number of parameters, which holds

$$\sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3} g_{r_1,r_2,r_3}^2 \le \|\mathcal{Y}\|_F^2 - \varepsilon^2$$

where $\mathcal{G} = \mathbf{U}_1^T \bullet \mathcal{Y} \bullet \mathbf{U}_3^T$, $\mathbf{U}_1$ and $\mathbf{U}_3$ comprise the singular vectors of mode-1 and mode-3 unfoldings, $\mathbf{Y}_{(1)}$ and $\mathbf{Y}_{(3)}$, respectively. The factor matrices are then selected as $\mathbf{X}_1 = \mathbf{U}_1(:, 1:R_1)$ and $\mathbf{X}_3 = \mathbf{U}_3(:, 1:R_3)$. This initialization method usually provides better models with a lower approximation error and a smaller number of parameters than the SeqProj-based method.

Example 1: The red curves illustrated in Fig. 2(left) show the number of TK2-parameters over iterations for the decomposition of random tensors of size $100 \times 10 \times 100$ with a relative error bound of 0.8. The SeqProj-like initialization yields a rank-(13, 92) model, which contains 22 460 parameters within two iterations and remains badly balanced after 200 iterations. Using the truncated HOSVD, we obtain an initial rank-(35, 27) model having only 15 650 parameters and a lower approximation error. The model is then updated and has 12 650, 12 300,

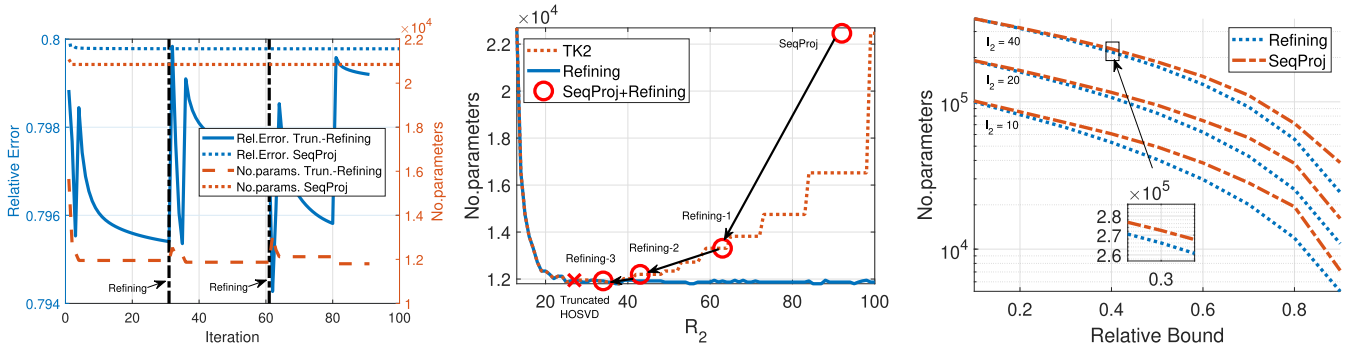

Fig. 2. Left: Model refining method for TK2 decomposition with a bound constraint for the tensors of size $100 \times 10 \times 100$. Center: Performance of single run of TK2 decomposition with a bound constraint and multistage refining for various ranks R_2. Right: TK2 with various bound constraint values.

and 11 950 parameters in the first three iterations, and remains unchanged and well balanced, while the approximation error converges after 30 iterations.

Model Refining: Truncated HOSVD initialization demonstrates to be efficient to TK2 with a bound constraint. In practice, the estimated model can often be squeezed to a smaller one. For a feasible solution obtained by the update in Section IV-B, our approach is to generate a truncated HOSVD model with the same rank and then use it to initialize another TK2 decomposition. We continue the model refining to the estimated model, if it is more compact. Otherwise, we terminate the procedure.

Illustration of the **model refining** in the above example is given in Fig. 2(left). The obtained models after two refinings have 11 870 and 11 800 parameters, respectively. In addition, in Fig. 2(center), we compare the number of model parameters between the single run of the TK2 algorithm and multistage refining for various values of the initial rank-R_2. When the rank is relatively small, $R_2 < 15$, there is no feasible solution. When $R_2 \geq 37$, refining always yields an estimated model close to the optimal. For the model having 22 460 parameters initialized by SeqProj, we execute the refining three times and obtain a final model with 11 920 parameters.

More demonstrations of the truncated HOSVD and refining is provided in Fig. 2(right) for the approximation of tensors of size $100 \times I_2 \times 100$. In all the test cases, truncated HOSVD+Refining yields smaller models than the SeqProj.

In summary, we present algorithms for TK2 with/without an error bound constraint and an efficient initialization and model-refining procedure. When solving TK2 as the subproblems within the TT-decomposition, the algorithms often converge very quickly in 10–20 iterations.

D. TT—A Nested Network of TK2

We will show that a nested network of TK2 decompositions forms a TT network. First, TK2 decomposition of $\mathcal{Y}$ gives two core tensors, $\mathcal{X}_1$ and $\mathcal{X}_N$, for the first and last modes

$$\mathcal{Y} \approx \mathcal{X}_1 \bullet \mathcal{Z}_2 \bullet \mathcal{X}_N$$

where $\mathcal{Z}_2 = \mathcal{X}_1^T \bullet \mathcal{Y} \bullet \mathcal{X}_N^T$ is of size $R_1 \times I_2 \times I_3 \times \cdots \times I_{N-1} \times R_{N-1}$. The matrices are estimated from a matrix of size $I_1 I_N \times I_1 I_N$ as in (6).

Next, we estimate two core tensors $\mathcal{X}_2$ of size $R_1 \times I_2 \times R_2$ and $\mathcal{X}_{N-1}$ of size $R_{N-2} \times I_{N-1} \times R_{N-1}$ within TK2 of $\mathcal{Z}_2$

$$\mathcal{Z}_2 \approx \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-2}$$

where $\mathcal{Z}_3$ is a tensor of size $R_2 \times I_3 \times I_4 \times \cdots \times I_{N-2} \times R_{N-2}$.

It can be verified that the estimation of the three core tensors $\mathcal{X}_2, \mathcal{Z}_3, \mathcal{X}_{N-2}$ within the TT-tensor $\mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1} \bullet \mathcal{X}_N$, while fixing the two orthogonal matrices $\mathcal{X}_1$ and $\mathcal{X}_N$, becomes the estimation of a TK2 decomposition of $\mathcal{Z}_2$

$$\begin{aligned}\|\mathcal{Y} - \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1} \bullet \mathcal{X}_N\|_F^2 \\ = \|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2 + \|\mathcal{Z}_2 - \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1}\|_F^2.\end{aligned}$$

Similarly, we perform the TK2 decomposition of $\mathcal{Z}_3$ to get the core tensors $\mathcal{X}_3$ and $\mathcal{X}_{N-3}$. Fig. 3 illustrates the recursive procedure of the TK2 decompositions, which finally forms a TT-decomposition of $\mathcal{Y}$.

Remark 3: Different from TT-SVD [15], [17], [23], which estimates the core tensors from left to right or right to left, the nested TK2 network estimates a pair of core tensors from both sides simultaneously, from the outer to inner of the network.

TK2 with a bound constraint can be used to construct a TT-model with a bounded approximation error. In the first layer, $\mathcal{X}_1$ and $\mathcal{X}_N$ are estimated within a smallest TK2 model such that

$$\|\mathcal{Y} - \mathcal{X}_1 \bullet \mathcal{Z}_2 \bullet \mathcal{X}_N\|_F^2 = \|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2 \leq \varepsilon^2.$$

This is achieved when $\|\mathcal{Y}\|_F^2 - \|\mathcal{Z}_2\|_F^2$ is close to or attains the bound $\varepsilon_1^2 = \varepsilon^2$ so that $\mathcal{X}_1$ and $\mathcal{X}_N^T$ have small ranks. In the second layer, we solve a TK2 with a much smaller bound

$$\|\mathcal{Z}_2 - \mathcal{X}_2 \bullet \mathcal{Z}_3 \bullet \mathcal{X}_{N-1}\|_F^2 \leq \varepsilon_2^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{Z}_2\|_F^2 \ll \varepsilon_1^2.$$

A similar procedure is applied to the core tensors $\mathcal{Z}_3, \mathcal{Z}_4, \ldots$, but the approximation errors are decreasing significantly. If the bound is attained in the first layer, i.e., $\varepsilon_2^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{Z}_2\|_F^2 = 0$, then higher layers solve exact TK2 models. Implying that the factor matrices within TK2 will have full rank or very high rank, i.e., $R_3 \approx R_2 I_2$, $R_{N-1} \approx I_{N-1} R_N$, $R_4 \approx R_3 I_3 \approx R_2 I_2 I_3$. In this case, the dimensions of the core tensors, especially the central cores, grow dramatically, and as a result, the final TT-model is not very compact. This behavior is also observed in the TT-SVD.

Remark 4: In order to deal with the large rank issue, we suggest scaling the error bounds in some first layers to smaller than the required bounds, e.g., by a factor of $\exp(-1 + n/\lfloor N/2 \rfloor)$, where $n = 1, 2, \ldots, \lfloor N/2 \rfloor$ is the layer index and $\lfloor N/2 \rfloor$ is the greatest integer less than or equal to

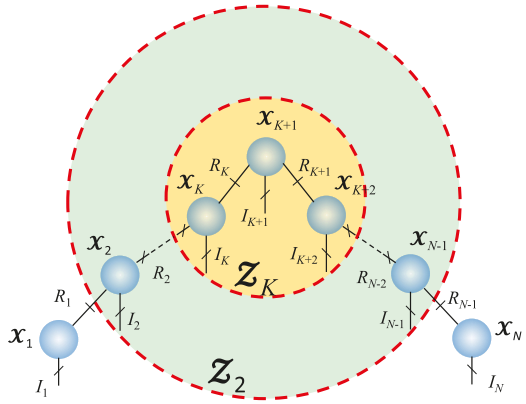

Fig. 3. Nested network of TK2 decompositions forms a TT network.

$N/2$. Example II demonstrates that NestedTK2 provides better models than the TT-SVD.

V. Alternating Multicore Update (AMCU) Algorithms

This section presents novel algorithms for TT-decomposition. We first present a simple form of the Frobenius norm of a TT-tensor, followed by a formulation of optimization problems to update single or a few core tensors at a time.

Lemma 2 (Frobenius Norm of a TT-Tensor [29]): Under the left-orthogonalization up to $\mathcal{X}_n$ and the right-orthogonalization up to $\mathcal{X}_m$, where $n \leq m$, the Frobenius norm of a TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ is equivalent to the Frobenius norm of $\mathcal{X}_{n:m}$, that is, $\|\mathcal{X}\|_F^2 = \|\mathcal{X}_{n:m}\|_F^2$.

Proof: With the left- and right-orthogonalizations, the two matricizations $[\mathcal{X}_{<n}]_{(n)}^T$ and $[\mathcal{X}_{>m}]_{(1)}^T$ are the orthogonal matrices. Hence, $\|\mathcal{X}\|_F^2 = \|[\mathcal{X}_{<n}]_{(n)}^T \bullet \mathcal{X}_{n:m} \bullet [\mathcal{X}_{>m}]_{(1)}\|_F^2 = \|\mathcal{X}_{n:m}\|_F^2$. □

A. Objective Function and Generalized Framework for the AMCU Algorithm

We now proceed to simplify the two optimization problems for the sub-TT-tensors that comprise a single core or a few consecutive core tensors. For this purpose, we assume that the TT-tensor $\mathcal{X}$ is left-orthogonalized up to $\mathcal{X}_n$ and right-orthogonalized up to $\mathcal{X}_m$, where in our methods, m can take one of the values $n, n+1$, or $n+2$.

Let $\mathcal{X}_{n:m} = \mathcal{X}_n \bullet \mathcal{X}_{n+1} \bullet \cdots \bullet \mathcal{X}_m$, then following Lemma 2, the error function in (1) and in (2) can be written as:

$$\begin{aligned} D &= \|\mathcal{Y}\|_F^2 + \|\mathcal{X}\|_F^2 - 2\langle \mathcal{Y}, \mathcal{X}\rangle \\ &= \|\mathcal{Y}\|_F^2 + \|\mathcal{X}_{n:m}\|_F^2 - 2\langle \mathcal{T}_{n:m}, \mathcal{X}_{n:m}\rangle \\ &= \|\mathcal{Y}\|_F^2 - \|\mathcal{T}_{n:m}\|_F^2 + \|\mathcal{T}_{n:m} - \mathcal{X}_{n:m}\|_F^2 \end{aligned} \tag{9}$$

where $\mathcal{T}_{n:m}$ is of size $R_{n-1} \times I_n \times \cdots \times I_m \times R_m$ and represents a tensor contraction between $\mathcal{Y}$ and $\mathcal{X}$ along all modes but the mode-$(n, n+1, \ldots, m)$ expressed as

$$\mathcal{T}_{n:m} = (\mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}) \rtimes_{N-m} \mathcal{X}_{>m} \quad \text{for } n = 1, 2, \ldots. \tag{10}$$

The objective function in (9) indicates that the sub-TT-tensor $\mathcal{X}_{n:m}$ is the best approximation to $\mathcal{T}_{n:m}$ in both problems (1) and (3). Following this, we can update $(m - n + 1)$ core tensors $\mathcal{X}_n, \ldots, \mathcal{X}_m$, while fixing the other cores $\mathcal{X}_j$, for $j < n$ or $j > m$. Since the cost function in (9) is formulated with the orthogonality conditions on $\mathcal{X}_j$, the new estimates $\mathcal{X}_n, \ldots, \mathcal{X}_m$ need to be orthogonalized in order to proceed to the next update. The algorithm should update the core tensors following the left-to-right order, i.e., increasing n, then switch to the right-to-left update procedure, i.e., decreasing n.

More specifically, in a single-core update, for which $m = n$, the algorithm sequentially updates first the core tensors $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_{N-1}$, and then $\mathcal{X}_N, \mathcal{X}_{N-1}, \ldots, \mathcal{X}_2$.

When $m = n + 1$, the update can be performed with overlapping core indices, e.g., $(\mathcal{X}_1, \mathcal{X}_2)$, $(\mathcal{X}_2, \mathcal{X}_3)$, …, as in the DMRG optimization scheme [25]. This method sequentially optimizes (reduces) ranks on the two sides of the core tensors. When the tensor is of a relatively high order, say 20, the ranks of the first core quickly tend to become small in the first few iterations, while the ranks of some last core tensors remain relatively high. For such cases, updating ranks on only one side of the core tensors is recommended. For example, the update $(\mathcal{X}_1, \mathcal{X}_2)$, $(\mathcal{X}_3, \mathcal{X}_4)$ adjusts the ranks on the left side of $\mathcal{X}_2$ and $\mathcal{X}_4$. Ranks on the right side of $\mathcal{X}_2$ and $\mathcal{X}_4$, i.e., R_2 and R_4, will be optimized when the algorithm runs the right-to-left update procedure, e.g., $(\mathcal{X}_4, \mathcal{X}_5)$, $(\mathcal{X}_2, \mathcal{X}_3)$. In the left-to-right update, the ranks R_2 and R_4 are not optimized, but can be adjusted after the left-orthogonalization of $\mathcal{X}_2$ and $\mathcal{X}_4$. Example IV compares the performance of the proposed algorithm over different numbers of overlapping core indices.

We also show that this update process is important in order to reduce the computational costs in a progressive computation of the contracted tensors $\mathcal{T}_{n:m}$, while for the particular cases of $m = n, n+1$, and $n+2$, we can derive efficient update rules for the core tensors $\mathcal{X}_n, \ldots, \mathcal{X}_m$.

B. Progressive Computation of Contracted Tensors $\mathcal{T}_{n:m}$

The computation of the contracted tensors $\mathcal{T}_{n:m}$ in (10), for $n = 1, 2, \ldots,$ is the most computationally expensive step in Algorithm 2, which requires $\mathcal{O}(\sum_{k=1}^{n-1} R_{k-1} R_k \prod_{j=k}^{N} I_j)$ operations for the left contraction $\mathcal{L}_n = \mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}$, and $\mathcal{O}(R_n \sum_{k=m+1}^{N} R_{k-1} R_k \prod_{j=n}^{k} I_j)$ operations for the right contraction $\mathcal{T}_{n:m} = \mathcal{L}_n \rtimes_{N-m} \mathcal{X}_{>m}$. For a particular case of $I_n = I$ and $R_n = R$ for all n, the computational cost to compute $\mathcal{T}_n$ is of order $\mathcal{O}(RI^N + R^2 I^{N-1})$.

Since the left contraction $\mathcal{L}_n$ can be expressed from $\mathcal{L}_{n-1}$ as

$$\mathcal{L}_n = \mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y} = \mathcal{X}_{n-1} \ltimes_2 \mathcal{L}_{n-1}$$

where $\mathcal{L}_1 = \mathcal{Y}$, the contracted tensors $\mathcal{T}_{n:m}$ can be computed efficiently through a progressive computation of $\mathcal{L}_n$. Similarly, $\mathcal{T}_{n:m}$ can also be computed through the right-contracted tensors as $\mathcal{T}_{n:m} = \mathcal{X}_{<n} \ltimes_{n-1} \mathcal{R}_m$, where $\mathcal{R}_m = \mathcal{Y} \rtimes_{N-m} \mathcal{X}_{>m} = \mathcal{R}_{m+1} \rtimes_2 \mathcal{X}_{m+1}$. In the left-to-right update procedure, the contracted tensors $\mathcal{T}_{n:m}$ are computed from the left-side-contracted tensors $\mathcal{L}_n$. The tensors $\mathcal{L}_{n+1}, \ldots, \mathcal{L}_{n+s-1}$ for the next update are then computed sequentially from $\mathcal{L}_n$ as in Step 5 in Algorithm 2. Here, $1 \leq s \leq k$, while $(k - s)$ represents the number of overlapping core indices. When the algorithm is in the right-to-left update

Algorithm 2 AMCU

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, and rank-$(R_1, R_2, \ldots, R_{N-1})$ or approximation accuracy ε^2, k: the number of core tensors to be updated per iteration $1 \le s \le k$ where $(k - s)$ indicates the number of overlapping core indices, and $\widetilde{N}$ is the index of the first core to be updated in the right-to-left update
Output: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ of rank-$(R_1, R_2, \ldots, R_{N-1})$ such that $\min \|\mathcal{Y} - \mathcal{X}\|_F^2$ (or $\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon^2$)
begin
1 Initialize $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$, e.g., by rounding $\mathcal{Y}$
repeat
% Left-to-Right update----------
for $n = 1, s+1, 2s+1, \ldots$ **do**
% Tensor contraction in (10)----------
2 $\mathcal{T}_{n:m} = \mathcal{L}_n \ltimes_{N-m} \mathcal{X}_{>m}$ /* $m = n + k - 1$, $\mathcal{L}_1 = \mathcal{Y}$ */
% Best TT-approximation to $\mathcal{T}_{n:m}$---------
3 $[\mathcal{X}_n, \ldots, \mathcal{X}_m] =$ bestTT_approx$(\mathcal{T}_{n:m})$
for $i = n, n+1, \ldots, n+s-1$ **do**
4 $\mathcal{X} =$ Left_Orthogonalize$(\mathcal{X}, i)$
% Update left-side-contracted tensor-
5 $\mathcal{L}_{i+1} = \mathcal{X}_i \ltimes_2 \mathcal{L}_i$
% Right-to-Left update----------
for $n = \widetilde{N}, \widetilde{N} - s, \widetilde{N} - 2s, \ldots$ **do**
6 $\mathcal{T}_{n:m} = \mathcal{L}_n \ltimes_{N-m} \mathcal{X}_{>m}$
7 $[\mathcal{X}_n, \ldots, \mathcal{X}_m] =$ bestTT_approx$(\mathcal{T}_{n:m})$
for $i = m, m-1, \ldots, m-s+1$ **do**
8 $\mathcal{X} =$ Right_Orthogonalize$(\mathcal{X}, i)$
until *a stopping criterion is met*

procedure, the left-side-contracted tensors $\mathcal{L}_n$ are available and do not need to be computed.

A similar procedure can be implemented to exploit the right-contracted tensors $\mathcal{R}_m$ by first executing the right-to-left update procedure and then switching to the left-to-right update order.

This computation method is adapted from the alternating linear scheme [18], [30] or the two-site DMRG algorithm [19], [25] for solving linear systems or EVDs in which all variables are in the TT-format. The AMCU is briefly described in Algorithm 2. The routine bestTT_approx within the AMCU in Step 3 computes the best TT-approximation to $\mathcal{T}_{n:m}$, which can be a low-rank matrix approximation or the low-multilinear rank TK2 decomposition, depending on whether $m = n+1$ or $m = n+2$. In general, the choice is free, but when $m = n$ (single-core updates), the challenge becomes to find a rank-adaptive procedure for the denoising problem in (2), as discussed in the next section. The alternating double- and triple-core update algorithms are presented in Supplementary Material.

C. Alternating Single-Core Update (ASCU)

We consider a simple case of the AMCU algorithm when $m = n$. The contracted tensor $\mathcal{T}_n$ is then of size $R_{n-1} \times I_n \times R_n$, and the error function in (9) becomes

$$D = \|\mathcal{Y}\|_F^2 - \|\mathcal{T}_n\|_F^2 + \|\mathcal{T}_n - \mathcal{X}_n\|_F^2 \quad \text{for } n = 1, 2, \ldots, N. \tag{11}$$

As mentioned earlier, we can perform the TT-decomposition in two different ways.

Algorithm 3 ASCU Algorithm (two-side rank adjustment)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$ and an error bound ε
Output: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ of minimum total TT-rank such that $\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon^2$
begin
1 Initialize $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$
repeat
% Left-to-Right update----------
for $n = 1, 2, \ldots, N-1$ **do**
2 $\mathcal{T}_n = \mathcal{L}_n \times_{N-n} \mathcal{X}_{>n}$
% Solve TK2 decomposition----------
3 $\|\mathcal{T}_n - \mathbf{A}_n \bullet \mathcal{X}_n \bullet \mathbf{B}_n\|_F^2 \le \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{T}_n\|_F^2$
% Adjust adjacent cores----------
4 $\mathcal{X}_{n-1} \leftarrow \mathcal{X}_{n-1} \bullet \mathbf{A}_n$, $\mathcal{X}_{n+1} \leftarrow \mathbf{B}_n \bullet \mathcal{X}_{n+1}$
5 $\mathcal{X} =$ Left_Orthogonalize$(\mathcal{X}, n)$
% Update left-side-contracted tensors-
6 $\mathcal{L}_n \leftarrow \mathbf{A}_n^T \bullet \mathcal{L}_n$, $\mathcal{L}_{n+1} \leftarrow \mathcal{X}_n \ltimes_2 \mathcal{L}_n$
% Right-to-Left update----------
for $n = N, N-1, \ldots, 2$ **do**
7 $\mathcal{T}_n = \mathcal{L}_n \times_{N-n} \mathcal{X}_{>n}$
8 $\|\mathcal{T}_n - \mathbf{A}_n \bullet \mathcal{X}_n \bullet \mathbf{B}_n\|_F^2 \le \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{T}_n\|_F^2$
9 $\mathcal{X}_{n-1} \leftarrow \mathcal{X}_{n-1} \bullet \mathbf{A}_n$, $\mathcal{X}_{n+1} \leftarrow \mathbf{B}_n \bullet \mathcal{X}_{n+1}$
10 $\mathcal{X} =$ Right_Orthogonalize$(\mathcal{X}, n)$
until *a stopping criterion is met*

1) For the **TT-approximation with a specified rank**, we obtain a solution $\mathcal{X}_n = \mathcal{T}_n$.
2) For the **TT-decomposition with a given accuracy**, $\mathcal{X}_n$ should have a minimum number of parameters, such that

$$\|\mathcal{T}_n - \mathcal{X}_n\|_F^2 \le \varepsilon_n^2 \tag{12}$$

where $\varepsilon_n^2 = \varepsilon^2 - \|\mathcal{Y}\|_F^2 + \|\mathcal{T}_n\|_F^2$ is assumed to be nonnegative. Note that adjusting the ranks R_{n-1} and R_n also requires manipulating $\mathcal{X}_{n-1}$ and $\mathcal{X}_{n+1}$ accordingly, and $\mathcal{T}_n$ implicitly depends on these manipulations.

Remark 5: A negative accuracy ε_n^2 indicates that either the rank R_{n-1} or R_n is quite small, and needs to be increased, that is, the core $\mathcal{X}_{n-1}$ or $\mathcal{X}_{n+1}$ should be adjusted to have higher ranks. Often, the TT-ranks R_n are set to sufficiently high values and then gradually decrease or at least behave in a nonincreasing manner during the update of the core tensors.

It is not straightforward to update $\mathcal{X}_n$ in the above problem; however, by expressing $\mathcal{X}_n = \mathbf{A}_n \bullet \tilde{\mathcal{X}}_n \bullet \mathbf{B}_n$ as a TK2 (5), the problem in (12) reduces to finding a TK2 $\mathcal{T}_n \approx \mathbf{A}_n \bullet \tilde{\mathcal{X}}_n \bullet \mathbf{B}_n$ with a minimum number of parameters such that

$$\|\mathcal{T}_n - \mathbf{A}_n \bullet \tilde{\mathcal{X}}_n \bullet \mathbf{B}_n\|_F^2 \le \varepsilon_n^2$$

where $\mathbf{A}_n$ and $\mathbf{B}_n$ are the matrices of size $R_{n-1} \times \tilde{R}_{n-1}$ and $\tilde{R}_n \times R_n$.

Remark 6: The TK2 $\mathbf{A}_n \bullet \tilde{\mathcal{X}}_n \bullet \mathbf{B}_n$ can be obtained using algorithms in Section IV. The new estimate of $\mathcal{X}$ is still of order-N, because $\mathbf{A}_n$ and $\mathbf{B}_n$ can be merged into $\mathcal{X}_{n-1}$ and $\mathcal{X}_{n+1}$ as $\mathcal{X} = \mathcal{X}_1 \bullet \cdots \bullet (\mathcal{X}_{n-1} \bullet \mathbf{A}_n) \bullet \mathcal{X}_n \bullet (\mathbf{B}_n \bullet \mathbf{X}_{n+1}) \bullet \cdots \bullet \mathbf{X}_N$.

In this way, the three cores $\mathcal{X}_{n-1}$, $\mathcal{X}_n$, and $\mathcal{X}_{n+1}$ are updated. Because $\mathbf{A}_n$ and $\mathbf{B}_n^T$ are the orthogonal matrices, the newly adjusted cores $\mathcal{X}_{n-1} \bullet \mathbf{A}_n$ and $\mathbf{B}_n \bullet \mathbf{X}_{n+1}$ obey the left- and right-orthogonality conditions. Algorithm 3 outlines the single-core update algorithm based on the TK2 decomposition. A graphical illustration of the update scheme is given in Fig. 4.

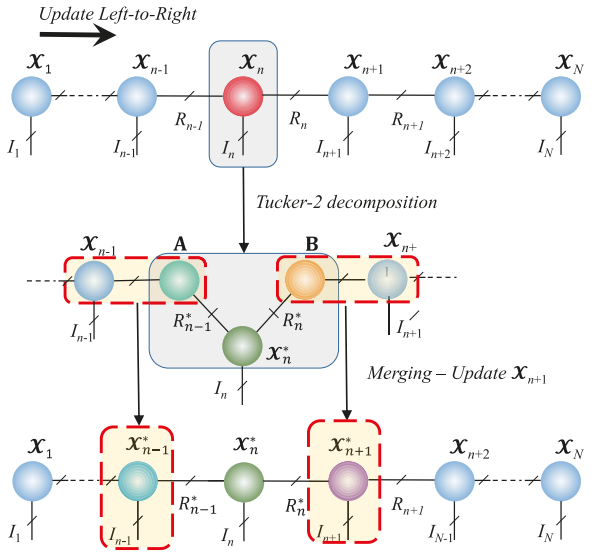

Fig. 4. Update scheme of the ASCU algorithm for the case of a three-core update. The core tensor $\mathcal{X}_n$ is approximated by TK2 decomposition, $\mathbf{A} \bullet \mathcal{X}_n^* \bullet \mathbf{B}$ with minimal ranks, R_{n-1}^* and R_n^*. The two core tensors, $\mathcal{X}_{n-1}$ and $\mathcal{X}_{n+1}$, are then updated by $\mathbf{A}$ and $\mathbf{B}$, respectively.

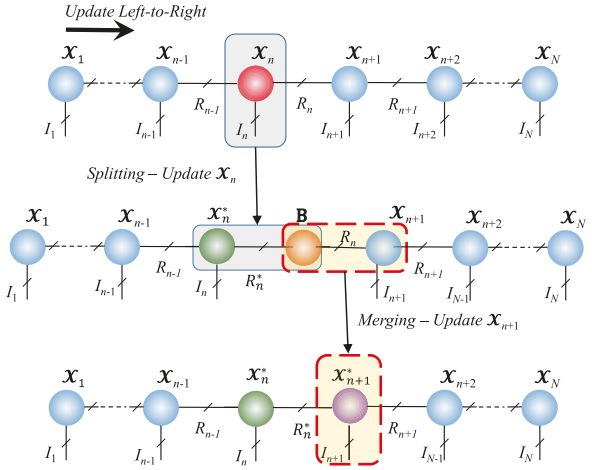

Fig. 5. Update scheme of the ASCU algorithm for the case of two-core update. The core tensor $\mathcal{X}_n$ is split into two core tensors, $\mathcal{X}_n^*$ and $\mathbf{B}$, with a minimal rank R_n^*. The core tensor $\mathcal{X}_{n+1}$ is then updated by $\mathbf{B}$.

Alternatively, instead of adjusting the two ranks, R_{n-1} and R_n, of $\mathcal{X}_n$, we can update only one rank, either R_{n-1} or R_n, corresponding to the right-to-left or left-to-right update order procedure. Assuming that the core tensors are updated in the left-to-right order, we need to find $\mathcal{X}_n$, which has minimum rank-R_n and satisfies $\|\mathcal{T}_n - \mathcal{X}_n \bullet \mathbf{B}_n\|_F^2 \leq \varepsilon_n^2$. This problem reduces to the truncated SVD of the mode-(1,2) matricization of $\mathcal{T}_n$ with an accuracy ε_n^2, that is, $[\mathcal{T}_n]_{(1,2)} \approx \mathbf{U}_n \, \mathbf{\Sigma} \, \mathbf{V}_n^T$, where $\mathbf{\Sigma} = \text{diag}(\sigma_{n,1}, \ldots, \sigma_{n,R_n^\star})$. Here, for the new optimized rank $R_n^\star$, the following holds:

$$\sum_{r=1}^{R_n^\star} \sigma_{n,r}^2 \geq \|\mathcal{Y}\|_F^2 - \varepsilon^2 > \sum_{r=1}^{R_n^\star - 1} \sigma_{n,r}^2. \tag{13}$$

For the approximation with an exact error bound, the first singular value is adjusted by the difference in the approximation error $(\sum_{r=1}^{R_n^*} \sigma_{n,r}^2 + \varepsilon^2 - \|\mathcal{Y}\|_F^2)^{1/2}$.

The core tensor $\mathcal{X}_n$ is then updated by reshaping $\mathbf{U}_n$ to an order-3 tensor of size $R_{n-1} \times I_n \times R_n^\star$, while the core $\mathcal{X}_{n+1}$ needs to be adjusted accordingly as

$$\mathcal{X}_{n+1}^\star = \mathbf{\Sigma} \, \mathbf{V}_n^T \bullet \mathcal{X}_{n+1}. \tag{14}$$

When the algorithm updates the core tensors in the right-to-left order, we update $\mathcal{X}_n$ by using the $R_{n-1}^\star$, leading to the right singular vectors of the mode-1 matricization of $\mathcal{T}_n$, and adjust the core $\mathcal{X}_{n-1}$ accordingly, that is

$$\begin{aligned} [\mathcal{T}_n]_{(1)} &\approx \mathbf{U}_n \, \mathbf{\Sigma} \, \mathbf{V}_n^T \\ \mathcal{X}_n^\star &= \text{reshape}\big(\mathbf{V}_n^T, \big[R_{n-1}^\star, I_n, R_n\big]\big) \\ \mathcal{X}_{n-1}^\star &= \mathcal{X}_{n-1} \bullet \mathbf{U}_n \mathbf{\Sigma}. \end{aligned} \tag{15}$$

To summarize, the proposed method updates one core and adjusts (or rotates) another core. Hence, it updates two cores at a time. The new estimate $\mathcal{X}_n^\star$ satisfies the left- or right-orthogonality conditions, and does not need to be orthogonalized again. The algorithm is listed in Algorithm 4, and its update scheme is illustrated in Fig. 5. Another observation is that the core tensor $\mathcal{X}_{n+1}$ or $\mathcal{X}_{n-1}$ will be updated in the next iteration after updating $\mathcal{X}_n$. Hence, the update of $\mathcal{X}_{n+1}$ in (14), i.e., in Step 5, and the update of $\mathcal{X}_{n-1}$ in (15), i.e., in Step 9, can be skipped, except for the last update.

Algorithm 4 ASCU Algorithm (one side rank adjustment)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$ and accuracy ε
Output: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ of minimum total TT-rank such that $\|\mathcal{Y} - \mathcal{X}\|_F^2 \leq \varepsilon^2$

begin
1 $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ by rounding $\mathcal{Y}$
repeat
% Left-to-Right update----------
for $n = 1, 2, \ldots, N-1$ **do**
2 $\mathcal{T}_n = \mathcal{L}_n \times_{N-n} \mathcal{X}_{>n}, \quad [\mathcal{T}_n]_{(1,2)} \approx \mathbf{U} \, \mathbf{\Sigma} \, \mathbf{V}^T$
3 $\mathcal{X}_n = \text{reshape}(\mathbf{U}, R_{n-1} \times I_n \times R_n)$
% Adjust adjacent cores----------
4 $\mathcal{X}_{n+1} \leftarrow (\mathbf{\Sigma} \, \mathbf{V}^T) \bullet \mathcal{X}_{n+1}$
% Update left-side-contracted tensor-
5 $\mathcal{L}_{n+1} \leftarrow \mathcal{X}_n \ltimes_2 \mathcal{L}_n$
% Right-to-Left update----------
for $n = N, N-1, \ldots, 2$ **do**
6 $\mathcal{T}_n = \mathcal{L}_n \times_{N-n} \mathcal{X}_{>n}, \quad [\mathcal{T}_n]_{(1)} \approx \mathbf{U} \, \mathbf{\Sigma} \, \mathbf{V}^T$
7 $\mathcal{X}_n = \text{reshape}(\mathbf{V}^T, R_{n-1} \times I_n \times R_n)$
8 $\mathcal{X}_{n-1} \leftarrow \mathcal{X}_{n-1} \bullet (\mathbf{U} \, \mathbf{\Sigma})$
until *a stopping criterion is met*

D. TT-SVD as a Variant of ASCU With One Update Round

Consider the approximation of a tensor $\mathcal{Y}$ of size $I_1 \times I_2 \times \cdots \times I_N$ using the ASCU algorithm with one-side rank adjustment at a given accuracy ε^2. The horizontal slices of the core tensors $\mathcal{X}_n$ are initialized by the unit vectors $\boldsymbol{e}_r$ as $\text{vec}(\mathbf{X}_n(r, :, :)) = \boldsymbol{e}_r$, for $r = 1, 2, \ldots, R_{n-1}$, where $R_n = \prod_{k=n+1}^{N} I_k$, i.e., the mode-1 matricizations of the core tensors are identity matrices, $[\mathcal{X}_n]_{(1)} = \mathbf{I}_{R_{n-1}}$. Therefore, the contracted tensor $\mathcal{T}_1$ is the data $\mathcal{Y}$, and the mode-1 approximation error is simply the global approximation error $\varepsilon_1^2 = \varepsilon^2$. For this reason, the ASCU estimates the first core tensor $\mathcal{X}_1$ as in TT-SVD.

Since the core tensors $\mathcal{X}_3, \ldots, \mathcal{X}_N$ are not updated, the contracted tensor $\mathcal{T}_2$ represents the projection of $\mathcal{Y}$ onto the

TABLE I

COMPARISON OF SUBOPTIMIZATION PROBLEMS PER ITERATION BETWEEN THE AMCU ALGORITHMS. THE ADCU_k OR ATCU_k DENOTES THE ADCU OR ATCU ALGORITHM WITH k OVERLAPPING CORE INDICES, WHEREAS ASCU_k DENOTES THE ASCU ALGORITHM WITH k-SIDE RANK ADJUSTMENT

AMCU	Sub opt.-problems	Update order of core tensors
ASCU_1	Low-rank matrix approximation to $\mathcal{T}_n$	$\mathcal{X}_n$, $\mathcal{X}_{n+1}$ (work as ADCU_1)
ASCU_2	Low multilinear-rank TK2 approximation to $\mathcal{T}_n$	$\mathcal{X}_{n-1}$ and $\mathcal{X}_n$ (work as ATCU_2)
ADCU	Low-rank matrix approximation to $\mathcal{T}_{n,n+1}$	$\mathcal{X}_n$ and $\mathcal{X}_{n+1}$
ATCU	Low multilinear-rank TK2 approximation to $\mathcal{T}_{n,n+1,n+2}$	$\mathcal{X}_n$, $\mathcal{X}_{n+1}$ and $\mathcal{X}_{n+2}$

subspace spanned by $\mathcal{X}_1$, implying that ASCU estimates $\mathcal{X}_2$ in a similar way as TT-SVD. The difference here is that the mode-2 approximation accuracy ε_2^2 in ASCU is affected by the term $\|\mathcal{Y}\|_F^2 - \|\mathcal{T}_2\|_F^2$ [see (12)], which is only zero or negligible for the exact or high-accuracy approximation $\varepsilon \approx 0$.

The remaining core tensors $\mathcal{X}_3, \ldots, \mathcal{X}_N$ are updated similarly, but with different approximation accuracies. Another difference is that TT-SVD estimates the core tensors once, while ASCU runs the right-to-left update after it completes the first round left-to-right update, and so on.

To summarize, TT-SVD acts as ASCU with one update cycle, but with a different error tolerance. ASCU attains an approximation error closer to the predefined accuracy.

E. Comparison of the AMCU Algorithms

Table I summarizes the suboptimization problems of the ASCU, alternating double-core update (ADCU), and triple-core update (ATCU) algorithms. In general, the ASCU with one-side rank adjustment (ASCU_1) works as the ADCU with one overlapping core index (ADCU_1), while the ASCU with two-side rank adjustment (ASCU_2) updates the cores similar to the updates of the ATCU with two overlapping core indices (ATCU_2). When the TT-rank is fixed, the ADCU with nonoverlapping core indices (ADCU_0) is two times faster than the (ASCU_1), while ATCU_0 is faster than ADCU_0. However, the difference is significant only when the number of cores is large. More comparisons are provided in Section VII.

VI. AMCU ALGORITHM FOR INPUT TENSOR IN TT-FORMAT

Consider a data tensor $\mathcal{Y}$ given in the TT-tensor format, which can be obtained, for example, by the prior compression of data using TT-SVD or Nested-TK2. Our alternating algorithms can be implemented with a much lower computational cost due to the efficient tensor contractions between two tensors, $\mathcal{Y}$ and $\mathcal{X}$. In other words, we assume that $\mathcal{Y} = \mathcal{Y}_1 \bullet \mathcal{Y}_2 \bullet \cdots \bullet \mathcal{Y}_N$, where $\mathcal{Y}_n$ are of size $S_{n-1} \times I_n \times S_n$. We next introduce the principles of fast contractions between two TT-tensors, followed by a formulation of update rules for the AMCU algorithm.

A. Contraction Between TT-Tensors

As previously stated, the most computationally expensive step in the AMCU algorithms is to compute the contraction

Algorithm 5 AMCU Algorithm for TT-Tensor

Input: TT-tensor $\mathcal{Y} = \mathcal{Y}_1 \bullet \mathcal{Y}_2 \bullet \cdots \bullet \mathcal{Y}_N$, and approximation accuracy ε, $\tilde{N}$ is the index of the first core to be updated in the right-to-left update, k is the number of core tensors to be updated per iteration, and $1 \le s \le k$ where $(k - s)$ indicates the number of overlapping indices,
Output: TT-tensor $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ such that $\|\mathcal{Y} - \mathcal{X}\|_F^2 \le \varepsilon$ with lower total TT-ranks

begin
1 Initialize $\mathcal{X} = \mathcal{X}_1 \bullet \mathcal{X}_2 \bullet \cdots \bullet \mathcal{X}_N$ by rounding $\mathcal{Y}$
% Precompute the right-contracted matrices $\mathbf{\Psi}_n$----------
for $n = N-1, \ldots, 1$ **do**
2 $\mathbf{\Psi}_n = (\mathcal{Y}_{n+1} \bullet \mathbf{\Psi}_{n+1}) \times_2 \mathcal{X}_{n+1}$ /* $\mathbf{\Psi}_N = 1$ */
repeat
% Left-to-Right update----------
for $n = 1, s+1, 2s+1, \ldots$ **do**
% Contracted tensor $\mathcal{T}_{n:m}$, $m = n+k-1$
3 $\mathcal{T}_{n:m} = \mathbf{\Phi}_n \bullet \mathcal{Y}_n \bullet \mathcal{Y}_{n+1} \bullet \cdots \bullet \mathcal{Y}_m \bullet \mathbf{\Psi}_m$
% Best TT-approximation to $\mathcal{T}_{n:m}$-----
4 $[\mathcal{X}_n, \ldots, \mathcal{X}_m] = \text{bestTT\_approx}(\mathcal{T}_{n:m}, \varepsilon)$
for $i = n, n+1, \ldots, n+s-1$ **do**
5 $\mathcal{X} = \text{Left\_Orthogonalize}(\mathcal{X}, i)$
6 $\mathbf{\Phi}_{i+1} \leftarrow \mathcal{X}_i \ltimes_2 (\mathbf{\Phi}_i \bullet \mathcal{Y}_i)$
% Right-to-Left update----------
for $n = \tilde{N}, \tilde{N}-s, \tilde{N}-2s, \ldots$ **do**
7 $\mathcal{T}_{n:m} = \mathbf{\Phi}_n \bullet \mathcal{Y}_n \bullet \mathcal{Y}_{n+1} \bullet \cdots \bullet \mathcal{Y}_m \bullet \mathbf{\Psi}_m$
8 $[\mathcal{X}_n, \ldots, \mathcal{X}_m] = \text{bestTT\_approx}(\mathcal{T}_{n:m}, \varepsilon)$
for $i = m, m-1, \ldots, m-s+1$ **do**
9 $\mathcal{X} = \text{Right\_Orthogonalize}(\mathcal{X}, i)$
10 $\mathbf{\Psi}_{i-1} \leftarrow (\mathcal{Y}_i \bullet \mathbf{\Psi}_i) \times_2 \mathcal{X}_i$
until *a stopping criterion is met*

Fig. 6. Benchmark images are used in Examples 2 and 5.

tensors $\mathcal{T}_{n:m}$. For the two TT-tensors $\mathcal{Y}$ and $\mathcal{X}$, we then have

$$\begin{aligned}\mathcal{T}_{n:m} &= (\mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}) \rtimes_{N-m} \mathcal{X}_{>m} \\ &= (\mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}_{<n}) \bullet \mathcal{Y}_{n:m} \bullet (\mathcal{Y}_{>m} \rtimes_{N-m} \mathcal{X}_{>m}) \\ &= \mathbf{\Phi}_n \bullet \mathcal{Y}_{n:m} \bullet \mathbf{\Psi}_m\end{aligned}$$

where the matrices $\mathbf{\Phi}_n = \mathcal{X}_{<n} \ltimes_{n-1} \mathcal{Y}_{<n}$ are of size $R_{n-1} \times S_{n-1}$ and represent a left contraction between $\mathcal{X}_{<n}$ and $\mathcal{Y}_{<n}$ along the first $(n-1)$ modes, and the matrices $\mathbf{\Psi}_n = \mathcal{Y}_{>n} \rtimes_{N-n} \mathcal{X}_{>n}$ are of size $S_n \times R_n$ and represent a right contraction between $\mathcal{Y}_{>n}$ and $\mathcal{X}_{>n}$ along all but mode-1. The contraction matrices $\mathbf{\Phi}_n$ and $\mathbf{\Psi}_n$ can be efficiently computed as

$$\mathbf{\Phi}_{n+1} = (\mathcal{X}_{<n} \bullet \mathcal{X}_n) \ltimes_{n-1} (\mathcal{Y}_{<n} \bullet \mathcal{Y}_n) = \mathcal{X}_n \ltimes_2 (\mathbf{\Phi}_n \bullet \mathcal{Y}_n)$$
$$\mathbf{\Psi}_{n-1} = (\mathcal{Y}_n \bullet \mathcal{Y}_{>n}) \rtimes_{N-n} (\mathcal{X}_n \bullet \mathcal{X}_{>n}) = (\mathcal{Y}_n \bullet \mathbf{\Psi}_n) \rtimes_2 \mathcal{X}_n$$

with the respective complexities of $\mathcal{O}(I_n R_{n-1} S_n (R_n + S_{n-1}))$ and $\mathcal{O}(I_n R_n S_{n-1} (S_n + R_{n-1}))$.

B. Generalized Framework for AMCU

The algorithm for the TT-tensor is summarized in Algorithm 5. It is important to emphasize that the right- and left-contraction matrices, $\mathbf{\Phi}_n$ and $\mathbf{\Psi}_n$, are not computed when updating the core tensors, but instead, we update either $\mathbf{\Phi}_{n+1}$

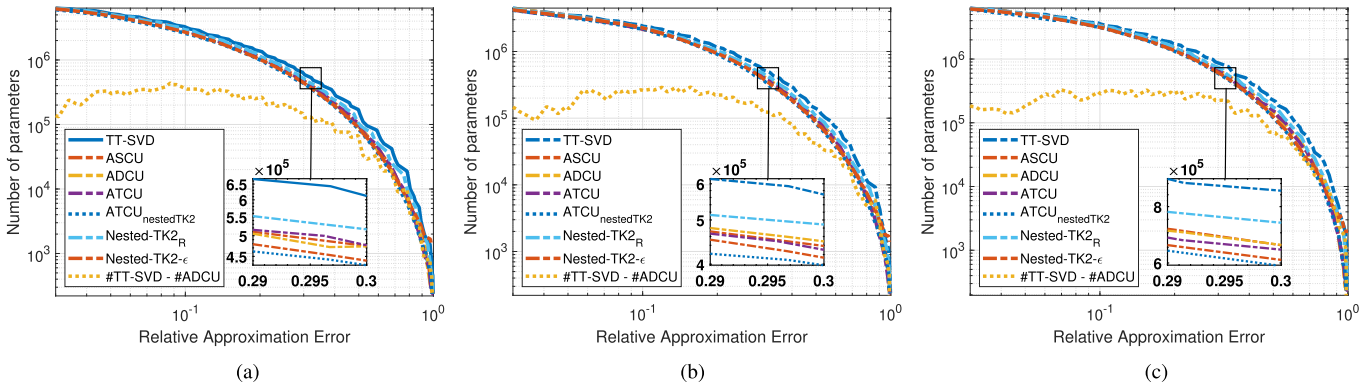

Fig. 7. Number of parameters within the models estimated by TT-SVD, nested-TK2, and AMCU algorithms for different error bounds. Decomposition of (a) Lena image, (b) Barbara image, and (c) House image.

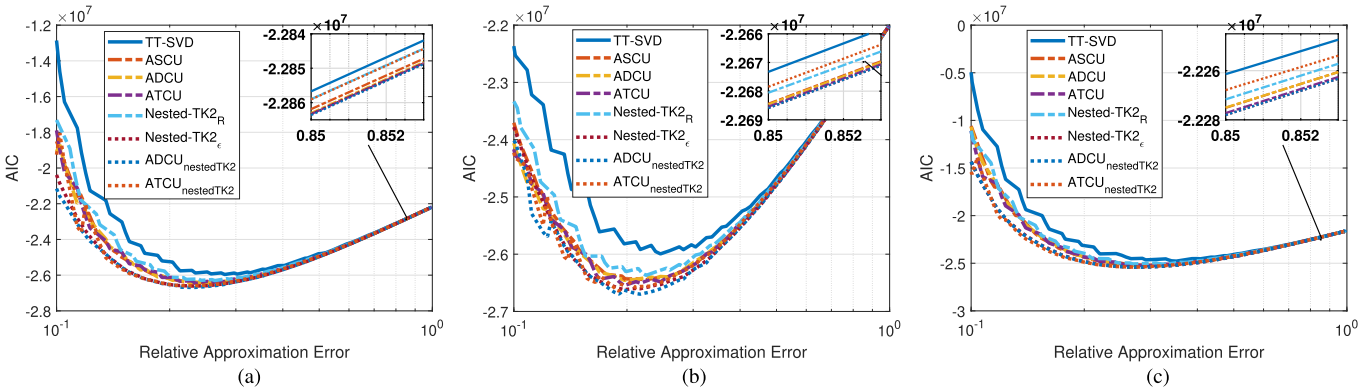

Fig. 8. AIC of the models estimated by TT-SVD, nested-TK2, and AMCU algorithms for different error bounds. Decomposition of (a) Lena image, (b) Barbara image, and (c) House image.

or $\mathbf{\Psi}_{n-1}$. In order to achieve this, we first compute the right-contraction matrices, $\mathbf{\Psi}_n$, before entering the main loop. Here, we denote $\mathbf{\Psi}_N = \mathbf{\Phi}_1 = 1$. At the first iteration, the algorithm executes the left-to-right update procedure, estimates $\mathcal{X}_{1:k}$ as the best TT-approximation to the tensor $\mathcal{Y}_{1:k} \bullet \mathbf{\Psi}_k$, orthogonalizes them, and then updates the left-contraction matrices $\mathbf{\Phi}_2, \mathbf{\Phi}_3, \ldots, \mathbf{\Phi}_s$ accordingly, where $1 \leq s \leq k$ and $(k-s)$ denotes the number of core tensors to be updated in two consecutive iterations. The algorithm next updates the core tensors $\mathcal{X}_{s+1}, \mathcal{X}_{s+2}, \ldots, \mathcal{X}_{s+k}$.

Similarly, the algorithm computes the new core tensors, $\mathcal{X}_n, \mathcal{X}_{n+1}, \ldots, \mathcal{X}_{n+k-1}$, left- orthogonalizes them, and then updates the left-contraction matrices, $\mathbf{\Phi}_n$, without computing the right-contraction matrices, $\mathbf{\Psi}_n$. While running the right-to-left update, the algorithm need not compute the left-contraction matrices but updates the right-contraction $\mathbf{\Psi}_{n-1}, \ldots, \mathbf{\Psi}_{n+k-2}$.

For $k = 1$, the ASCU algorithm updates $\mathcal{X}_n$ as in Section V-C. For $k = 2$, the ADCU algorithm computes a low-rank approximation to the mode-(1,2) unfolding of $\mathcal{T}_{n,n+1} = \mathbf{\Phi}_n \bullet \mathcal{Y}_n \bullet \mathcal{Y}_{n+1} \bullet \mathbf{\Psi}_{n+1}$ or a truncated SVD of the following matrix:

$$[\mathcal{T}_{n,n+1}]_{(1,2)} = [\mathbf{\Phi}_n \bullet \mathcal{Y}_n]_{(1,2)} [\mathcal{Y}_{n+1} \bullet \mathbf{\Psi}_{n+1}]_{(1)}$$

where $[\mathbf{\Phi}_n \bullet \mathcal{Y}_n]_{(1,2)}$ and $[\mathcal{Y}_{n+1} \bullet \mathbf{\Psi}_{n+1}]_{(1)}$ are, respectively, of sizes $R_{n-1} I_n \times S_n$ and $S_n \times I_{n+1} R_{n+1}$. When $S_n < R_{n-1} I_n$ and $S_n < I_{n+1} R_{n+1}$, the SVD is computed for a reduced size matrix $\mathbf{U}_n \mathbf{V}_n^T$, where $\mathbf{U}_n$ and $\mathbf{V}_n$ are the upper triangular matrices in the QR decompositions of $[\mathbf{\Phi}_n \bullet \mathcal{Y}_n]_{(1,2)}$ and $[\mathcal{Y}_{n+1} \bullet \mathbf{\Psi}_{n+1}]_{(1)}^T$.

For the ATCU algorithm, tensor contractions are computed for three indices $[n, n+1, n+2]$ as

$$\mathcal{T}_{\boldsymbol{n}} = \mathbf{\Phi}_n \bullet \mathcal{Y}_n \bullet \mathcal{Y}_{n+1} \bullet \mathcal{Y}_{n+2} \bullet \mathbf{\Psi}_{n+2}.$$

The algorithm solves the TK2 decomposition of the mode-((1,2),3,(4,5)) unfolding of $\mathcal{T}_{\boldsymbol{n}}$ as [see also (5)]

$$\min_{\mathbf{U}_n, \mathbf{V}_n} \|\mathcal{Z}_n - \mathbf{U}_n \bullet \mathcal{X}_{n+1} \bullet \mathbf{V}_n^T\|_F^2$$

where $\mathbf{U}_n = [\mathcal{X}_n]_{(1,2)}$, $\mathbf{V}_n = [\mathcal{X}_{n+2}]_{(1)}^T$, $\mathcal{Z}_n = [\mathcal{T}_{\boldsymbol{n}}]_{(1,2),3,(4,5)} = \mathbf{A}_n \bullet \mathcal{Y}_{n+1} \bullet \mathbf{B}_n^T$, and $\mathbf{A}_n = [\mathbf{\Phi}_n \bullet \mathcal{Y}_n]_{(1,2)}$ are of size $R_{n-1} I_n \times S_n$, while $\mathbf{B}_n = [\mathcal{Y}_{n+2} \bullet \mathbf{\Psi}_{n+2}]_{(2,3)}$ are of size $I_{n+2} R_{n+2} \times S_{n+1}$. The two factor matrices, $\mathbf{U}_n$ and $\mathbf{V}_n$, are sequentially estimated as the principal components of the matrices $(\mathcal{Z} \bullet \mathbf{V}_n) \times_2 (\mathcal{Z} \bullet \mathbf{V}_n)$ and the matrices $(\mathbf{U}_n^T \bullet \mathcal{Z}) \times_2 (\mathbf{U}_n^T \bullet \mathcal{Z})$.

VII. Simulations

We first validated the proposed algorithms for fitting higher order tensors constructed from color images at a predefined error bound, followed by two examples on the denoising of exponentially decaying signals, which admit the TT-representation. Second, the proposed class of TT-decomposition algorithms was tested on the denoising of benchmark color images. The final example considers blind source separation from a single-channel mixture. The TK2 algorithms in subproblems for the ASCU and the ATCU stop when the difference in the relative errors is smaller than

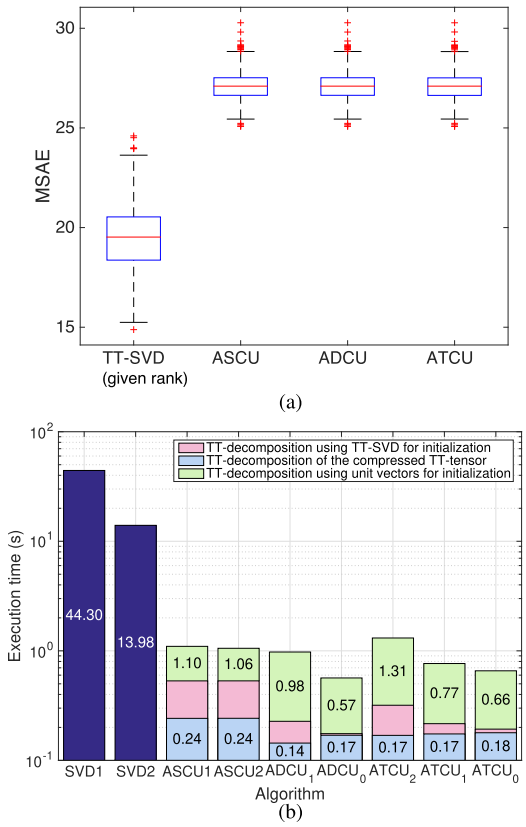

Fig. 9. Denoising of signals at poor SNR. (a) Mean squared angular errors when $K = 2^{22}$. (b) Execution time in seconds when the signal length $K = 2^{24}$.

10^{-4} in 100 iterations. Usually, the algorithms converge in less than ten iterations.

Example 2 (Fitting Image by TT-Decomposition): This example demonstrates the effectiveness of the proposed algorithms over the standard TT-SVD algorithm in a decomposition of benchmark color images of size $256 \times 256 \times 3$ shown in Fig. 6. Due to the space limit, we illustrate the performances for the three images "Lena," "Barbara," and "House." The decomposition aims to express the images by small patches of size $8 \times 8 \times 3$. In order to achieve this, we applied the horizontal and vertical shifts within a window of $[-2, 2]$ to generate 24 copies, which together with the original images created a tensor of size $25 \times 256 \times 256 \times 3$. The data were then folded by Kronecker folding [31] to yield order-7 tensors, $\mathcal{Y}$, of size $25 \times 4 \times 4 \times 4 \times 4 \times 4 \times 192$.

In Fig. 7, we compared the number of parameters within the estimated models when the relative approximation errors, ϵ, were bounded, that is, $\|\mathcal{Y} - \mathcal{X}\|_F \leq \epsilon \|\mathcal{Y}\|_F$. We did not consider the case when the number of model parameters exceeded the number of data entries 4 915 200, e.g., when the approximation error bound $\epsilon < 0.03$. In all the tests, the models based on TT-SVD comprised a higher number of parameters than those obtained by the AMCU algorithms. For example, in order to explain the tensor constructed from the "Lena" image at a relative approximation error of 0.0870, TT-SVD yielded a model consisting of 3 734 573 parameters, while the models estimated by ASCU, ADCU, and ATCU needed 308 964, 439 208, and 439 272 fewer parameters to achieve the relative errors of 0.0851, 0.0870, and 0.0869, respectively. The difference in the number of parameters between TT-SVD and ADCU can be observed as dotted lines in Fig. 7.

TABLE II

TT-RANKS OF SIGNALS x_r OF LENGTH $K = 2^{22}$ AND OF THEIR ESTIMATES $\hat{x}_r$ USING THE TT-SVD AND THE AMCU ALGORITHMS IN EXAMPLE IV. THE SQUARED ANGULAR ERROR IS GIVEN ON THE LOGARITHMIC SCALE, AND THE EXECUTION TIME IS IN SECONDS

Signal	TT-ranks	SAE (dB)	Time (s)
x_1	2-2-3-3-3-3-3-4-4-5-6-7-8-10-13-19-26-32-16-8-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-967-789-443-228-115-58-30-16-8-4-2	4.18	9.69
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-2-2-3-3-6-11-20-34-32-16-8-4-2	27.49	3.25
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-2-2-3-5-8-14-28-49-45-24-16-8-4-2	26.66	2.01
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-2-4-5-10-13-26-20-40-24-16-8-4-2	27.89	2.54
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-1-2-2-3-5-8-14-28-48-22-24-16-8-4-2	27.61	2.81
$\hat{x}_{ATCU_1}$	1-1-1-1-1-1-1-2-2-3-5-8-14-26-37-22-24-16-8-4-2	27.64	2.41
$\hat{x}_{ATCU_0}$	1-1-1-1-1-1-1-2-4-3-5-10-14-22-33-22-24-16-8-4-2	28.18	2.62
x_2	2-4-8-16-32-56-47-38-32-26-22-18-15-13-12-10-8-7-6-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-675-959-782-440-226-114-57-28-15-8-4-2	6.18	9.63
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-2-4-8-13-21-35-65-92-54-27-15-8-4-2	22.73	2.08
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-2-3-5-8-12-20-37-71-94-52-26-13-7-4-2	23.10	1.52
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-2-4-4-8-11-22-36-72-85-54-27-15-8-4-2	23.34	1.45
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-2-3-5-8-12-20-37-70-104-56-28-13-7-4-2	23.11	1.83
$\hat{x}_{ATCU_0}$	1-1-1-1-1-1-1-2-4-7-11-22-37-66-105-54-27-15-8-4-2	23.07	1.67
x_3	2-2-2-2-2-3-3-3-3-4-4-4-5-6-7-9-12-16-8-4-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-966-789-443-228-115-58-29-15-8-4-2	4.41	9.67
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-1-1-2-2-2-3-4-7-11-13-8-4-2	31.48	3.10
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-1-1-2-2-3-6-10-18-32-16-8-8-4-2	32.47	2.15
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-1-2-2-4-2-4-6-12-16-16-8-4-2	34.58	2.52
$\hat{x}_{ATCU_2}$	1-1-1-1-1-1-1-1-1-2-2-3-5-8-14-23-8-8-8-4-2	33.52	2.77
$\hat{x}_{ATCU_0}$	1-1-2-1-1-2-1-1-2-1-2-3-2-3-6-6-9-16-8-4-2	31.49	2.58
x_4	2-2-2-2-2-3-3-3-3-3-3-3-3-3-3-3-2-1-1-1		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-111-207-378-653-920-762-433-223-112-56-28-14-7-4-2	4.36	9.69
$\hat{x}_{ASCU}$	1-1-1-1-1-1-1-1-1-1-1-2-2-3-3-3-3-2-1-1-1	35.88	2.91
$\hat{x}_{ADCU_1}$	1-1-1-1-1-1-1-1-1-1-1-2-2-2-3-4-8-15-11-7-4-2	35.89	2.17
$\hat{x}_{ADCU_0}$	1-2-1-2-1-2-1-2-1-2-1-2-4-8-13-26-24-16-8-4-2	39.35	2.50
$\hat{x}_{ATCU_0}$	1-1-2-1-1-2-1-1-2-1-1-2-2-3-3-3-3-2-1-1-1	36.12	2.61
x_5	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2		
$\hat{x}_{TT-SVD}$	2-4-8-16-31-59-112-210-387-677-966-788-443-228-115-58-29-15-8-4-2	5.17	9.69
$\hat{x}_{ASCU}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	3.21
$\hat{x}_{ADCU_1}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	3.17
$\hat{x}_{ATCU_0}$	2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2-2	46.04	2.76

For the decomposition with low accuracies (high error bounds), the TT-models in all algorithms became relatively low rank or rank-1; hence, there was no or not much difference in terms of the number of parameters. However, with the same parameters, our algorithms yielded lower approximation errors. The remarkably high difference exceeding 100 000 parameters was observed when the relative approximation errors were lower than 0.4. In summary, AMCU needed fewer parameters than TT-SVD for the same approximation error.

We also check the nested-TK2 method with the same TT-ranks as in TT-SVD (Nested-TK2$_R$) or with the same error bound (Nested-TK2$_\varepsilon$). Fig. 7 indicates that Nested-TK2$_R$ achieved lower approximation errors than the TT-SVD, while Nested-TK2$_\varepsilon$ yielded much smaller models. The results of Nested-TK2$_\varepsilon$ provided good initialization.

In Fig. 8, we provide a more intuitive way of ranking the estimated models for the three images based on the Akaike information criterion (AIC) in the case of least squares estimation. For each relative approximation error, the best

TABLE III
PERFORMANCE COMPARISON OF ALGORITHMS CONSIDERED IN EXAMPLE V IN TERMS OF MSE (dB), PSNR (dB), AND SSIM FOR IMAGE DENOISING WHEN SNR = 10 dB

Algorithms	MSE	PSNR	SSIM	MSE	PSNR	SSIM
	Lena			Pepper		
TT-SVD	35.11	32.68	0.892	40.40	32.07	0.861
TT-ASCU	**27.37**	**33.76**	**0.927**	**31.47**	**33.15**	**0.924**
TT-ADCU	28.04	33.65	0.926	32.09	33.07	0.923
Tucker	34.59	32.74	0.919	38.96	32.23	0.917
BRTF	40.30	32.07	0.840	46.85	31.42	0.825
K-SVD	34.76	32.72	0.908	35.74	32.60	0.918
	Pens			Barbara		
TT-SVD	44.92	31.61	0.884	32.30	33.04	0.901
TT-ASCU	**36.61**	**32.50**	**0.908**	**24.92**	**34.16**	**0.934**
Tucker	48.56	31.27	0.884	33.20	32.92	0.919
BRTF	42.80	31.82	0.877	31.87	33.10	0.899
K-SVD	50.04	31.14	0.862	35.41	32.64	0.908
	House			House2		
TT-SVD	23.70	34.38	0.877	41.07	32.00	0.905
TT-ASCU	**19.30**	**35.28**	**0.899**	**38.53**	**32.27**	**0.926**
Tucker	23.64	34.40	0.885	48.11	31.31	0.909
BRTF	27.93	33.67	0.823	42.99	31.80	0.867
K-SVD	22.18	34.67	0.881	46.44	31.46	0.907

model is the one with the lowest AIC score. The results again confirm that our algorithms provided better models than the TT-SVD. Upon closer inspection, we can see that the models determined by ATCU were slightly better than those by ASCU and ADCU for relatively low-accuracy approximation. The comparison also indicates that the nested-TK2 yielded better models than TT-SVD. The models obtained by TT-SVD were indeed not optimal in terms of the total number of model parameters.

Example 3 (Reconstruction of Known Target Ranks): Harmonic retrieval is at the very core of signal-processing applications. To illustrate the potential of TT-decomposition in this context, we considered the reconstruction of an exponentially decaying signal $x(t)$ from a noisy observation $y(t) = x(t) + e(t)$ of $K = 2^d$ samples, where

$$x(t) = \exp\left(\frac{-5t}{K}\right) \sin\left(\frac{2\pi f}{f_s}t + \frac{\pi}{3}\right) \tag{16}$$

$d = \{22, 24 \text{ or } 26\}$ with $f = 10$ Hz, $f_s = 100$ Hz, while $e(t)$ represents the additive Gaussian noise, which was randomly generated such that the signal-to-noise ratio SNR $= -20$ dB.

In order to retrieve the original signal $x(t)$ from the noisy signal $y(t)$, we adopt the tensor-based method in [32], which first constructs a higher order tensor from the signal and then approximates this tensor by a low-rank model. For example, De Lathauwer [32] suggested to use the canonical polyadic TD and the block term decomposition. In this example, the observed signal was reshaped to an order-$(d-2)$ quantics tensor $\mathcal{Y}$ of size $4 \times 2 \times \cdots \times 2 \times 4$ [33], [34]. With this tensorization, the sinusoid yields a TT-tensor of rank-$(2, 2, \ldots, 2)$, whereas the signal $\exp(t)$ yields a rank-1 tensor. Hence, its Hadamard product, i.e., $x(t)$, admits a TT model of rank-$(2, 2, \ldots, 2)$ [23] and gives the TT-model, $\mathcal{Y} = \mathcal{X} + \mathcal{E}$, where $\mathcal{X}$ is the TT-tensor of the signal $x(t)$ and $\mathcal{E}$ is reshaped from the noise. In other words, we attempted to approximate the tensor $\mathcal{Y}$ by a TT-tensor with a prior known TT-rank.

In order to compare TT-SVD with the AMCU algorithms, the tensor $\mathcal{Y}$ was first approximated using TT-SVD such that $\|\mathcal{Y} - \hat{\mathcal{X}}\|_F^2 \leq \varepsilon^2$, where ε is a measure of the added noise.

Fig. 10. Lena image corrupted by noise at 10-dB SNR and the patches reconstructed by different methods in Example 5. (a) Noisy image. (b) TT-ASCU, mse = 27:37 dB. (c) TT-SVD, mse = 35.11 dB. (d) K-SVD, mse = 34.76 dB. (e) From left to right, TT-ASCU, TT-SVD, and K-SVD.

The estimated TT-tensors had quite high ranks, which exceeded the TT-rank of $\mathcal{X}$, and were then "truncated" to the TT-rank of $\mathcal{X}$ [15] using the TT-tensor toolbox [29].

Alternatively, to obtain a TT-tensor having the same ranks as $\mathcal{X}$, the TT-SVD algorithm computed only $R_n = 2$ leading singular vectors from the projected data. The outcome TT-tensor was used to initialize the AMCU algorithms.

We ran 500 independent trials and assessed performance through the relative error $\delta(\boldsymbol{y}, \hat{\boldsymbol{x}}) = (\|\boldsymbol{y} - \hat{\boldsymbol{x}}\|_2^2)/\|\boldsymbol{y}\|_2^2$ and the squared angular error $SAE(\boldsymbol{x}, \hat{\boldsymbol{x}}) = -20\log_{10}\arccos(\boldsymbol{x}^T\hat{\boldsymbol{x}})/(\|\boldsymbol{x}\|_2\|\hat{\boldsymbol{x}}\|_2)$ (dB).

Fig. 9(a) illustrates a performance comparison in terms of SAEs for the case $K = 2^{22}$, demonstrating that, on average, the signals reconstructed by our proposed algorithms exhibit an 8-dB higher SAE than when using the TT-SVD with the rank specified. For $K = 2^{24}$ and $K = 2^{26}$, the average SAEs of the TT-SVD were improved to 25.70 and 29.07 dB, but were still lower than the respective mean SAEs of 32.56 and 38.17 dB achieved using our algorithms.

For completeness, Fig. 9(b) compares the execution times of the considered algorithms, where ASCU_1 and ASCU_2 denote the ASCU algorithms with one- and two-side rank adjustments, respectively, while ADCU_k and ATCU_k indicate the ADCU and ATCU algorithms with k overlapping

core indices, where $k = 0, 1, 2$. When the signal length $K = 2^{24}$, the TT-SVD with rounding took an average execution time of 44.30 s on a computer based on Intel Xeon E5-1650, clocked at 3.50 GHz and with 64 GB of main memory. For a given TT-rank, this algorithm worked faster and completed the approximation in 13.98 s. Since TT-SVD provided good initial values, ASCU converged quickly in 0.53 s, while the ADCU and ATCU algorithms were approximately two times faster than the ASCU. Even when the core tensors were initialized by unit vectors $\boldsymbol{e}_r$, the proposed algorithms converged in less than 2 s. For this kind of initialization, the ATCU was on average the fastest and ASCU the slowest algorithm.

Finally, we illustrate the performance of the AMCU algorithms in Algorithm 5 for the task of fitting the TT-tensors $\mathcal{Y}_{\tilde{\varepsilon}}$, which were approximations to $\mathcal{Y}$ with an accuracy of $\tilde{\varepsilon} = 0.3$, using the TT-SVD. The algorithms achieved an average SAE of 26.95 dB when the signal length $K = 2^{22}$ and an SAE of 32.52 dB when $K = 2^{24}$. There was no significant loss in accuracy compared with the AMCU fit to the tensor $\mathcal{Y}$. Moreover, the AMCU algorithm required shorter running times, e.g., 0.24 s for ASCU, and 0.17 s for ADCU and ATCU. The running time of TT-SVD for initialization is not counted in the total running times of AMCU algorithms.

Example 4 (Denoising With Unknown Target Ranks): To illustrate the utility of the TT-decomposition as a tool for denoising, we considered the noisy signals, $y(t) = x(t) + e(t)$, and the degraded versions of a signal, $x(t)$, through contamination with additive Gaussian noise, $e(t)$, where $x(t)$ can take one of the following forms:

$$\begin{aligned} x_1(t) &= \frac{\sin(2000\ t^{2/3})}{4t^{1/4}} \\ x_2(t) &= \sin(t^{-1}) \\ x_3(t) &= \sin\left(\frac{5(t+1)}{2}\right)\cos(100(t+1)^2) \\ x_4(t) &= \text{sign}(\sin(8\pi t))(1+\sin(80\pi t)) \end{aligned}$$

or the damped signal used in Example III. The signals $y(t)$ in our example had a length of $K = 2^{22}$ and were tensorized (reshaped) to tensors $\mathcal{Y}$ of order-22 and size $2 \times 2 \times \cdots \times 2$. With this tensorization, the five signals $x_r(t)$ can be well approximated by the tensors in the TT-format, with their TT-ranks given in Table II, where $x_5(t)$ is the signal in (16).

We applied the alternating single- and multicore update algorithms to approximate the noisy tensor with SNR = 0 dB. The approximation $\|\mathcal{Y} - \hat{\mathcal{X}}\|_F^2 \leq \varepsilon^2$ was first performed using TT-SVD with the accuracy level of $\varepsilon^2 = \sigma^2 K$ and σ the standard deviation of the Gaussian noise. The reconstructed signals achieved SAEs of 4.18, 6.18, 4.43, 4.36, and 5.17 dB for the five signals $x_r(t)$, respectively. When using the ASCU, ADCU, and ATCU algorithms, much better performances were obtained with average respective SAEs = 33.11, 33.49, and 33.23 dB. The performance comparison is presented in Table II, where ADCU_1 and ADCU_0 denote the performances of the ADCU algorithms with one overlapping index and nonoverlapping indices, respectively. For example, for the reconstruction of the signal $x_1(t)$, ADCU_1 enforced the first eight-core tensors to be quite small with a rank of 1 and could not suppress the TT-ranks of the last core tensors. Consequently, the TT-ranks R_{15}, R_{16}, R_{17}, and R_{18} exceeded those of ADCU_0, and the TT-tensor estimated by ADCU_1 had 11 578 entries, which was more than the 6798 entries estimated by ADCU_0. Another important observation is that the signal reconstructed by ADCU_1 was worse than the reconstruction by ADCU_0, by about 1-dB SAE.

In addition to higher angular errors, the TT-SVD yielded approximations with TT-ranks significantly higher than those of the sources. This detrimental effect did not occur for the ASCU algorithm. For this example, the TT-SVD took on average 9.67 s to estimate all the core tensors of the five tensors, $\hat{\mathcal{X}}_r$, while the ADCU and ATCU algorithms needed 2.24 and 2.37 s, respectively, and were slightly faster than the ASCU.

Example 5 (Image Denoising): Next, we tested the proposed algorithms in a novel application of the TT-decomposition for image denoising. Given that the intensities of pixels in a small window are highly correlated, our method was able to learn hidden structures, which represent relations between small patches of pixels. These structures were then used to reconstruct the image as a whole.

For a color image $\mathbf{Y}$ of size $I \times J \times 3$, degraded by additive Gaussian noise, the basic idea behind the proposed method is that for each block of pixels of size $h \times w \times 3$, given by $\mathbf{Y}_{r,c} = \mathbf{Y}(r : r+h-1, c : c+w-1, :)$, a small tensor $\mathcal{Y}_{r,c}$ of size $h \times w \times 3 \times (2d+1) \times (2d+1)$, comprising $(2d+1)^2$ blocks around $\mathbf{Y}_{r,c}$ is constructed, in the form $\mathcal{Y}_{r,c}(:, :, :, d+1+i, d+1+j) = \mathbf{Y}_{r+i,c+j}$, where $i, j = -d, \ldots, 0, \ldots, d$, and d represents the neighborhood width. Every (r, c)-block $\mathbf{Y}_{r,c}$ is then approximated through the TT-decomposition $\|\mathcal{Y}_{r,c} - \mathcal{X}_{r,c}\|_F^2 \leq \varepsilon^2$, where ε^2 is the noise level. A pixel is then reconstructed as an average of all its approximations by TT-tensors, which cover that pixel.

We used six benchmark color images of size $256 \times 256 \times 3$ (illustrated in Fig. 6) and corrupted them by white Gaussian noise at SNR = 10 dB. Latent structures were learned for patches of sizes $8 \times 8 \times 3$, i.e., $h = w = 8$, in the search area of width $d = 3$. To the noisy images, we applied DCT spatial filtering before the block reconstruction and applied several TDs, including the TT-SVD, the Tucker approximation (TKA) with a predefined approximation error, the Bayesian robust tensor factorization (BRTF) for low-rank CP decomposition [35], and the ASCU. The TKA operates in a similar way to the TK2 algorithm in Section IV, but estimates five factor matrices for order-5 tensors. In addition, we recovered the image with sparsity constraints using a dictionary of 256 atoms learned by K-SVD [36]. For this method, three layers of color images were flattened to the size 256×768. The dictionary was learned for the patches of size 8×8.

The quality of the images reconstructed by the five different methods was assessed using three indices: mean-squared error (mse), peak signal-to-noise ratio (PSNR), and the structural similarity index (SSIM). The results are shown in Table III and illustrated in Fig. 10. By learning the similarities between the patches, our proposed method was able to recover the image and achieved better performance than the well-known denoising method based on dictionary learning. Moreover,

the results confirm the superiority of ASCU over the TT-SVD and other TDs. Using the ADCU algorithm, we obtained comparable performances with those of ASCU.

An expanded version of this example is provided in Supplementary Material.

Example 6 (Blind Source Separation of Exponentially Decaying Signals From a Single-Channel Mixture): This example is provided in Supplementary Material [39].

VIII. Conclusion and Further Extensions

We have presented novel algorithms for TT-decomposition, which are capable of simultaneously adjusting the ranks of two- or three-core tensors while keeping the other cores fixed. Compared with the TT-SVD, the proposed algorithms have achieved lower approximation errors for the decomposition with a given TT-rank and yielded tensors with lower TT-ranks for constrained approximations with a prescribed error tolerance. By employing progressive computation of contracted tensors and prior compression, the proposed algorithms have been shown to exhibit low computational complexity. The proposed algorithms can be naturally extended to the TT-decomposition with the nonnegativity constraints or decompositions of incomplete data. The AMCU methods can also be applied to the tensor chain decomposition [37]. In the sequel of this article, we aim to illuminate the use of the proposed algorithms in blind source separation and for a conversion of a TT-tensor into a low-rank tensor in the canonical polyadic decomposition. The proposed algorithms have been implemented in the MATLAB package TENSORBOX [38].

Acknowledgment

Some of the simulations for this project was performed on the Pardus and Zhores CDISE HPC clusters at Skoltech.

References

[1] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Found. Trends Mach. Learn.*, vol. 9, nos. 4–5, pp. 249–429, 2016.

[2] Y. Wu, H. Tan, Y. Li, J. Zhang, and X. Chen, "A fused CP factorization method for incomplete tensors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 751–764, Mar. 2019.

[3] X. Chen *et al.*, "A generalized model for robust tensor factorization with noise modeling by mixture of Gaussians," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5380–5393, Nov. 2018.

[4] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S.-I. Amari, "Bayesian robust tensor factorization for incomplete multiway data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 736–748, Apr. 2016.

[5] F. Ju, Y. Sun, J. Gao, Y. Hu, and B. Yin, "Vectorial dimension reduction for tensors based on Bayesian inference," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4579–4592, Oct. 2018.

[6] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(R_1, R_2,...,R_N) approximation of higher-order tensor," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, Mar. 2000.

[7] R. Bro, R. A. Harshman, N. D. Sidiropoulos, and M. E. Lundy, "Modeling multi-way data with linearly dependent loadings," *J. Chemometrics*, vol. 23, nos. 7–8, pp. 324–340, 2009.

[8] G. Favier and A. de Almeida, "Overview of constrained PARAFAC models," *EURASIP J. Adv. Signal Process.*, vol. 2014, no. 1, pp. 1–25, 2014.

[9] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms—Part I: Lemmas for partitioned matrices," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1022–1032, 2008.

[10] A.-H. Phan, P. Tichavský, and A. Cichocki, "Tensor deflation for CANDECOMP/PARAFAC—Part I: Alternating subspace update algorithm," *IEEE Trans. Signal Process.*, vol. 63, no. 12, pp. 5924–5938, Nov. 2015.

[11] A.-H. Phan, P. Tichavský, and A. Cichocki, "Tensor deflation for CANDECOMP/PARAFAC. Part 3: Rank splitting," *ArXiv e-prints*, 2015.

[12] A. Cichocki, R. Zdunek, A.-H. Phan, and S. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Hoboken, NJ, USA: Wiley, 2009.

[13] A. Cichocki *et al.*, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015.

[14] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jan. 2017.

[15] I. V. Oseledets and E. E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions," *SIAM J. Sci. Comput.*, vol. 31, no. 5, pp. 3744–3759, 2009.

[16] A. Klumper, A. Schadschneider, and J. Zittartz, "Equivalence and solution of anisotropic spin-1 models and generalized t-j Fermion models in one dimension," *J. Phys. A, Math. Gen.*, vol. 24, no. 16, p. L955, Aug. 1991.

[17] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, vol. 91, Oct. 2003, Art. no. 147902.

[18] S. Holtz, T. Rohwedder, and R. Schneider, "The alternating linear scheme for tensor optimization in the tensor train format," *SIAM J. Sci. Comput.*, vol. 34, no. 2, pp. 683–713, 2012.

[19] D. Kressner, M. Steinlechner, and A. Uschmajew, "Low-rank tensor methods with subspace correction for symmetric eigenvalue problems," *SIAM J. Sci. Comput.*, vol. 36, no. 5, pp. A2346–A2368, 2014.

[20] 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.

[21] L. Grasedyck, M. Kluge, and S. Krämer, "Variants of alternating least squares tensor completion in the tensor train format," *SIAM J. Sci. Comput.*, vol. 37, no. 5, pp. 2424–2450, 2015.

[22] C. Da Silva and F. J. Herrmann, "Optimization on the hierarchical tucker manifold—Applications to tensor completion," *Linear Algebra its Appl.*, vol. 481, pp. 131–173, Sep. 2015.

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

[24] D. Kressner and F. Macedo, "Low-rank tensor methods for communicating Markov processes," in *Quantitative Evaluation of Systems*, G. Norman and W. Sanders, Eds. Springer, 2009, pp. 25–40.

[25] S. R. White, "Density-matrix algorithms for quantum renormalization groups," *Phys. Rev. B, Condens. Matter*, vol. 48, no. 14, Oct. 1993, Art. no. 10345.

[26] I. V. Oseledets and B. N. Khoromskij, "DMRG+QTT approach to computation of the ground state for the molecular Schrödinger operator," MPI MiS Preprint 69/2010, Max Planck Inst. Math. Sci., Leipzig, Germany, 2010.

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

[28] Y. Xu, "On the convergence of higher-order orthogonal iteration," *Linear Multilinear Algebra*, vol. 66, no. 11, pp. 2247–2265, 2018.

[29] I. V. Oseledets. (2014). *TT-Toolbox*. [Online]. Available: https://github.com/oseledets/TT-Toolbox

[30] S. V. Dolgov and D. V. Savostyanov, "Alternating minimal energy methods for linear systems in higher dimensions," *SIAM J. Sci. Comput.*, vol. 36, no. 5, pp. A2248–A2271, 2014.

[31] A.-H. Phan, A. Cichocki, P. Tichavský, D. P. Mandic, and K. Matsuoka, "On revealing replicating structures in multiway data: A novel tensor decomposition approach," in *Latent Variable Analysis and Signal Separation* (Lecture Notes in Computer Science), vol. 7191. Berlin, Germany: Springer, 2012, pp. 297–305.

[32] L. De Lathauwer, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-(L_r,L_r,1) terms," *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 4, pp. 1451–1474, 2011.

[33] I. V. Oseledets, "Approximation of $2^d \times 2^d$ matrices using tensor decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 2130–2145, 2010.

[34] B. N. Khoromskij, "$O(d \log N)$-quantics approximation of $N-d$ tensors in high-dimensional numerical modeling," *Constructive Approximation*, vol. 34, no. 2, pp. 257–280, 2011.

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

[36] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, Nov. 2006.
[37] B. N. Khoromskij, "Tensors-structured numerical methods in scientific computing: Survey on recent advances," *Chemometrics Intell. Lab. Syst.*, vol. 110, no. 1, pp. 1–19, 2012.
[38] A.-H. Phan and P. Tichavský, and A. Cichocki. (2012). *MATLAB TENSORBOX package*. [Online]. Available: https://github.com/phananhhuy/TensorBox
[39] A.-H. Phan, A. Cichocki, A. Uschmajew, P. Tichavský, G. Luta, and D. Mandic, "Tensor networks for latent variable analysis. Part I: Algorithms for tensor train decomposition (supplementary)," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

Anh-Huy Phan (Member, IEEE) received the master's degree from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2005, and the Ph.D. degree from the Kyushu Institute of Technology, Kitakyushu, Japan, in 2011.

From October 2011 to March 2018, he was a Research Scientist with the Laboratory for Advanced Brain Signal Processing, Brain Science Institute (BSI), RIKEN, Wako, Japan, and from April 2012 to April 2015, a Visiting Research Scientist with the TOYOTA Collaboration Center, BSI-RIKEN. Since May 2018, he has been an Assistant Professor with the Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. He is currently a Visiting Associate Professor with the Tokyo University of Agriculture and Technology (TUAT), Fuchu, Japan. He is the author of three monographs. His research interests include multilinear algebra, tensor computation, tensor networks, nonlinear system, blind source separation, and brain–computer interface.

Prof. Phan received the Best Paper Awards for articles in the IEEE *Signal Processing Magazine* (SPM) in 2018 and the International Conference on Neural Information Processing (ICONIP) in 2016 and the Outstanding Reviewer Award for maintaining the prestige of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2019.

Andrzej Cichocki (Fellow, IEEE) received the M.Sc. (Hons.), Ph.D., and Dr.Sc. (Habilitation) degrees in electrical engineering from the Warsaw University of Technology, Warsaw, Poland.

He was an Alexander-von-Humboldt Research Fellow and a Guest Professor with University Erlangen-Nurnberg, Erlangen, Germany. From 1995 to 2018, he was the Team Leader and the Head of the Laboratory for Advanced Brain Signal Processing, RIKEN Brain Science Institute, Wako, Japan. He is currently a Professor with the Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. He is also a Visiting/Adjunct Professor with the Tokyo University of Agriculture and Technology (TUAT), Fuchu, Japan, Hangzhou Dianzi University (HDU) Hangzhou, China, Nicolaus Copernicus University (UMK), Toruń, Poland, and the Institute of Systems Research (IBS), Polish Academy of Science, Warsaw. He is the author of more than 500 peer-review articles and six monographs in English (two of them translated to Chinese). His research focus on deep learning, tensor decompositions, tensor networks for big data analytics, multiway blind source separation, and brain–computer interface and their biomedical applications. His publications currently report over 37 000 citations according to Google Scholar, with an H-index of 84.

Prof. Cichocki is currently among the three most cited Polish Computer Scientists. He has served as the Founding Editor-in-Chief for the *Journal Computational Intelligence and Neuroscience*. He has also served as an Associated Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, and the *Journal of Neuroscience Methods*.

André Uschmajew received the Ph.D. degree from Technical University Berlin, Berlin, Germany, in 2013. He is currently a Research Group Leader with the Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany, in 2013. His research interests include theory and optimization methods for low-rank approximation of tensors and high-dimensional equations.

Petr Tichavský (Senior Member, IEEE) received the Ph.D. degree in theoretical cybernetics and the Research Professor degree from the Czechoslovak Academy of Sciences, Prague, Czech Republic, in 1992 and 2017, respectively.

He is currently with the Institute of Information Theory and Automation, Czech Academy of Sciences, Prague. He is the author or coauthor of research articles in the area of sinusoidal frequency/frequency-rate estimation, adaptive filtering and tracking of time-varying signal parameters, algorithm-independent bounds on achievable performance, sensor array processing, independent component analysis, blind source separation, and tensor decompositions.

Dr. Tichavský has been a member of the IEEE SPS Committee Signal Processing Theory and Methods from 2008 to 2011 and since 2016. He was also the General Co-Chair of the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP 2011 in Prague. He served as an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS from 2002 to 2004 and an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2005 to 2009 and from 2011 to 2016.

George Luta received the Licentiate Diploma degree in mathematics from the University of Timişoara, Timişoara, Romania, in 1987, and the M.S. and Ph.D. degrees in biostatistics from the University of North Carolina at Chapel Hill, NC, USA, in 1996 and 2006, respectively.

After working at the University of North Carolina at Chapel Hill and the National Institute of Statistical Sciences Research Triangle Park, NC, USA, he joined the Department of Biostatistics, Bioinformatics and Biomathematics, Georgetown University, Washington, DC, USA, in 2007, where he is currently a Professor of biostatistics. His current honorary appointments include being a Visiting Professor at Aarhus University, Aarhus, Denmark, and a Visiting Professor at Parker Institute, Frederiksberg, Denmark. He has published more than 100 peer-reviewed articles that have received more than 6000 citations. His current methodological interests include statistical methods based on nonnegative matrix and tensor factorizations, and statistical methods for measures of health disparity.

Prof. Luta is an Elected Member of the International Statistical Institute.

Danilo P. Mandic (Fellow, IEEE) is currently a Professor of signal processing with the Imperial College London, U.K., where he has been working in the area of nonlinear and multidimensional adaptive signal processing and time-frequency analysis. He has been a Guest Professor with Katholieke Universiteit Leuven (KU Leuven), Leuven, Belgium, and a Frontier Researcher with the RIKEN Center for Brain Science, Tokyo, Japan. His publication record includes two research monographs, recurrent neural networks for prediction and complex valued nonlinear adaptive filters, noncircularity, widely linear and neural models, an edited book, signal processing for information fusion, and more than 200 publications on signal and image processing.