Partitioned Alternating Least Squares Technique for Canonical Polyadic Tensor Decomposition

Petr Tichavský, Anh-Huy Phan, and Andrzej Cichocki

***Abstract*—Canonical polyadic decomposition (CPD), also known as parallel factor analysis, is a representation of a given tensor as a sum of rank-one components. Traditional method for accomplishing CPD is the alternating least squares (ALS) algorithm. Convergence of ALS is known to be slow, especially when some factor matrices of the tensor contain nearly collinear columns. We propose a novel variant of this technique, in which the factor matrices are partitioned into blocks, and each iteration jointly updates blocks of different factor matrices. Each partial optimization is quadratic and can be done in closed form. The algorithm alternates between different random partitionings of the matrices. As a result, a faster convergence is achieved. Another improvement can be obtained when the method is combined with the enhanced line search of Rajih *et al.* Complexity per iteration is between those of the ALS and the Levenberg–Marquardt (damped Gauss–Newton) method. It is important, however, that the idea of alternating quadratic optimization with partitioned factor matrices is general and can be applied to other variants of the tensor decomposition problems, e.g., when non-Gaussian additive noise is considered.**

I. Introduction

CANONICAL polyadic decomposition (CPD) of a tensor, also known as parallel factor analysis (PARAFAC), is an extension of a low-rank decomposition to higher-way arrays, usually called tensors. It has many applications, e.g., in chemometrics, psychometrics, telecommunication, sensor array processing, data mining, neuroscience, blind source separation, arithmetic complexity and others, see, e.g., [1]–[3], and the references therein. Most of the existing methods of PARAFAC are based on the alternating least squares (ALS) algorithm that proceeds iteratively, and minimizes a criterion (that is usually quadratic) of the fit with respect to individual factors one by one.

Convergence of the ALS algorithm was studied in [4], [5]. It was shown to be linear, sublinear, or superlinear even in the case of rank-one fitting. In the case of a general rank, the global convergence is not guaranteed, and side local minima may occur. The convergence of ALS is known often to be slow if some of the modes contain nearly collinear vectors, where the iteration ends in a "convergence bottleneck," or in "swamp" situations or nearly "degenerate" cases, when the factors are highly collinear in all modes [6]. In order to improve convergence of the ALS in such cases, many alternative algorithms were developed, namely, the enhanced line search (ELS) [7], semialgebraic method [8], Levenberg–Marquardt (LM) algorithm [9], and its fast implementations [10], [11], nonlinear search algorithm using dog-leg trust-region method [12]. A comprehensive comparison of these techniques exceeds the scope of this letter.

Although these algorithms have proved to be quite good in many applications, ALS remains to be the most frequently used algorithm for CPD because of its simplicity. We propose a novel variant of this technique, which is less simple, but still transforms the problem to a series of quadratic optimizations so that monotonic convergence is guaranteed just as in ALS. In the method, the factor matrices are partitioned into blocks, and in each step blocks of different matrices are jointly updated. Each optimization subproblem is quadratic and can be solved in closed form. The technique is referred to as partitioned ALS (PALS). The partitioning can be taken at random or to follow some criteria to reduce the complexity. As a default, we consider random partitioning of the factor matrices.

The algorithm has appeared to converge much more regularly than the ordinary ALS and requires much fewer iterations in difficult scenarios, where the factor matrices contain collinear or nearly collinear columns. Indeed, the convergence of ALS can be made faster through the ELS technique, but both ALS and ALS+ELS have the same points of slow or no convergence, and in these points PALS may further reduce the fitting error. Note that PALS can also be combined with the ELS technique.

The letter is organized as follows. Section II presents the PALS algorithm, and ELS for completeness. Section III summarizes some implementation details. Section IV contains simulations and Section V concludes the letter.

II. Partitioned ALS

For simplicity, we restrict our presentation to third-order tensors. An extension of the proposed algorithm to higher-order tensors is straightforward. Alternatively, for decomposition of higher-order tensor one may use a technique of folding its dimension described in [13], which converts the CPD of the original tensor to CPDs of order-3 tensors.

Assume that a third-order tensor $\mathcal{T}$ of the dimension $I \times J \times K$ has elements

$$T_{ijk} = \sum_{f=1}^{R} A_{if} B_{jf} C_{kf} \tag{1}$$

where A_{if}, B_{jf}, and C_{kf}, are elements of factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, respectively, that have dimensions $I \times R$, $J \times R$, and $K \times R$. Here, R is the number of the factors. Symbolically, we

Manuscript received April 14, 2016; revised June 01, 2016; accepted June 03, 2016. Date of publication June 06, 2016; date of current version June 22, 2016. The work of P. Tichavský was supported by the Czech Science Foundation through project No. 14-13713S. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Bruno Demissie.

P. Tichavský is with the Institute of Information Theory and Automation, Prague 182-08, Czech Republic (e-mail: tichavsk@utia.cas.cz).

A. H. Phan is with the Laboratory for Advanced Brain Signal Processing, Brain Science Institute, RIKEN, Wakoshi 351-0198, Japan (e-mail: phan@brain.riken.jp).

A. Cichocki is with the Laboratory for Advanced Brain Signal Processing, Brain Science Institute, RIKEN, Wakoshi 351-0198, Japan, and also with System Research Institute, Warsaw 01-447, Poland (e-mail: cia@brain.riken.jp).

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

Digital Object Identifier 10.1109/LSP.2016.2577383

TABLE I
OUTLINE OF THE PALS ALGORITHM

Input: tensor $\mathcal{Y}$, initial factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of R columns.
Repeat (until convergence)
1) Select a random partitioning (i_1, i_2, i_3) of $\{1, 2, \ldots, R\}$.
2) Update $(\mathbf{A}_1, \mathbf{B}_2, \mathbf{C}_3)$ by minimizing the cost function, considering the rest of $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ as fixed.
3) Update $(\mathbf{A}_2, \mathbf{B}_3, \mathbf{C}_1)$ by minimizing the cost function, considering the rest of $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ as fixed.
4) Update $(\mathbf{A}_3, \mathbf{B}_1, \mathbf{C}_2)$ by minimizing the cost function, considering the rest of $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ as fixed.

End

write [1]

$$\mathcal{T} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]. \tag{2}$$

The CPD model (2) of a tensor is multilinear. It means that the model is linear with respect to each of the factor matrices separately, but not jointly. Joint estimation of these factor matrices is effectively nonlinear. On the other hand, optimization of the model fit with respect to each of the factor matrices can proceed separately in closed form, and it is a global minimum of the cost function. The ALS consists in a cyclic optimization with respect to $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$. The cost function

$$\varphi(\mathcal{T}, \mathbf{A}, \mathbf{B}, \mathbf{C}) \stackrel{\triangle}{=} \|\mathrm{vec}(\mathcal{T} - [[\mathbf{A}, \mathbf{B}, \mathbf{C}]])\|^2 \tag{3}$$

can be minimized in closed form, and convergence is monotonic.

Let i_1, i_2, i_3 be disjoint subsets of the set $\{1, 2, \ldots, R\}$ such that their union is the whole set. Note that one or two sets of i_1, i_2, i_3 can be empty. Let $\mathbf{A}_1$ be composed of those columns of $\mathbf{A}$ indexed by i_1: in MATLAB notation it would be $\mathbf{A}_1 = \mathbf{A}(:, i_1)$. Similarly, $\mathbf{A}_2$, $\mathbf{A}_3$, $\mathbf{B}_1, \ldots, \mathbf{C}_3$ are defined.

The key observation of the PALS technique is that the tensor is a linear function of $(\mathbf{A}_1, \mathbf{B}_2, \mathbf{C}_3)$ provided that the other part of the factor matrices are fixed, because

$$\mathcal{T} = [[\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1]] + [[\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2]] + [[\mathbf{A}_3, \mathbf{B}_3, \mathbf{C}_3]]. \tag{4}$$

The algorithm may proceed by cyclic update with respect to $(\mathbf{A}_1, \mathbf{B}_2, \mathbf{C}_3)$, $(\mathbf{A}_2, \mathbf{B}_3, \mathbf{C}_1)$, and $(\mathbf{A}_3, \mathbf{B}_1, \mathbf{C}_2)$. As a default, we propose to take the partitioning (i_1, i_2, i_3) at random after each cycle above. The resultant algorithm is summarized in Table I.

III. Implementation Details

A. Partitioned ALS

Computation of the update of $(\mathbf{A}_1, \mathbf{B}_2, \mathbf{C}_3)$ in the PALS can be performed as follows. The relation between the vectorized tensor and the parameter

$$\boldsymbol{\theta} = [\mathrm{vec}(\mathbf{A}_1)^T, \mathrm{vec}(\mathbf{B}_2)^T, \mathrm{vec}(\mathbf{C}_3)^T]^T \tag{5}$$

can be written as

$$\mathrm{vec}(\mathcal{T}) = \mathbf{J}\,\boldsymbol{\theta} \tag{6}$$

where

$$\mathbf{J} = \frac{\partial \mathrm{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}}. \tag{7}$$

Let tensor $\mathcal{Y}$ be a noisy observation of $\mathcal{T}$. Then, the least squares solution for $\boldsymbol{\theta}$ can be written as

$$\hat{\boldsymbol{\theta}} = \mathbf{J}^{\dagger}\mathrm{vec}(\mathcal{Y}) = (\mathbf{J}^H\mathbf{J})^{-1}\mathbf{J}^H\mathrm{vec}(\mathcal{Y}) = \mathbf{H}^{-1}\mathbf{g}. \tag{8}$$

with $\mathbf{H} = \mathbf{J}^H\mathbf{J}$ and $\mathbf{g} = \mathbf{J}^H\mathrm{vec}(\mathcal{Y})$; the symbol † stands for the Moore–Penrose pseudoinverse, and ^{H} is the Hermitian transposition [1], [10].

Now, the matrix $\mathbf{J}$ can be partitioned in three blocks as

$$\mathbf{J} = [\mathbf{J}_A, \mathbf{J}_B, \mathbf{J}_C] \tag{9}$$

where

$$\mathbf{J}_A = (\mathbf{C}_1 \odot \mathbf{B}_1) \otimes \mathbf{I}_I \tag{10}$$

$\odot$ and $\otimes$ stand for the Khatri–Rao and the Kronecker product, respectively, $\mathbf{I}_I$ is for the identity matrix of the size $I \times I$, $\mathbf{J}_B$ is composed of $|i_2|$ blocks of the form

$$\mathbf{J}_B = \{\mathbf{C}_{:,i} \otimes (\mathbf{I}_J \otimes \mathbf{A}_{:,i})\}_{i \in i_2} \tag{11}$$

and $\mathbf{J}_C$ is composed of $|i_3|$ blocks of the form

$$\mathbf{J}_C = \{\mathbf{I}_K \otimes (\mathbf{B}_{:,i} \otimes \mathbf{A}_{:,i})\}_{i \in i_3} \tag{12}$$

and $|i_2|$ and $|i_3|$ denote numbers of elements in i_2 and i_3, respectively. Note that computation of the error gradient $\mathbf{g} = \mathbf{J}^H\mathrm{vec}(\mathcal{Y})$ has the same complexity as computation of $\mathbf{G}$ in the ordinary ALS, i.e., $O(R^4)$ if $I = J = K = R$. The matrix $\mathbf{H} = \mathbf{J}^H\mathbf{J}$ can be written as

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{AA} & \mathbf{H}_{AB} & \mathbf{H}_{AC} \\ \mathbf{H}_{BA} & \mathbf{H}_{BB} & \mathbf{H}_{BC} \\ \mathbf{H}_{CA} & \mathbf{H}_{CB} & \mathbf{H}_{CC} \end{bmatrix} \tag{13}$$

where

$$\begin{aligned}
\mathbf{H}_{AA} &= \mathbf{J}_A^H\mathbf{J}_A = [(\mathbf{B}_1^H\mathbf{B}_1) \star (\mathbf{C}_1^H\mathbf{C}_1)] \otimes \mathbf{I}_I \\
\mathbf{H}_{BB} &= \mathbf{J}_B^H\mathbf{J}_B = [(\mathbf{A}_2^H\mathbf{A}_2) \star (\mathbf{C}_2^H\mathbf{C}_2)] \otimes \mathbf{I}_J \\
\mathbf{H}_{CC} &= \mathbf{J}_C^H\mathbf{J}_C = [(\mathbf{A}_3^H\mathbf{A}_3) \star (\mathbf{B}_3^H\mathbf{B}_3)] \otimes \mathbf{I}_K \\
\mathbf{H}_{AB} &= \mathbf{J}_A^H\mathbf{J}_B = \{(\mathbf{C}_{:,i}^H\mathbf{C}_{:,j})(\mathbf{A}_{:,j}\mathbf{B}_{:,i}^H)\}_{i \in i_1, j \in i_2} \\
\mathbf{H}_{AC} &= \mathbf{J}_A^H\mathbf{J}_C = \{(\mathbf{B}_{:,i}^H\mathbf{B}_{:,j})(\mathbf{A}_{:,j}\mathbf{C}_{:,i}^H)\}_{i \in i_1, j \in i_3} \\
\mathbf{H}_{BC} &= \mathbf{J}_B^H\mathbf{J}_C = \{(\mathbf{A}_{:,i}^H\mathbf{A}_{:,j})(\mathbf{B}_{:,j}\mathbf{C}_{:,i}^H)\}_{i \in i_2, j \in i_3} \\
\mathbf{H}_{BA} &= \mathbf{H}_{AB}^H, \quad \mathbf{H}_{CA} = \mathbf{H}_{AC}^H, \quad \mathbf{H}_{CB} = \mathbf{H}_{BC}^H.
\end{aligned}$$

Here, $\star$ denotes the Hadamard (elementwise) product. The matrix $\mathbf{H}$ has the size $R^2 \times R^2$ so its inversion takes $O(R^6)$ operations in general. However, it can be easily seen that if the sizes of i_1, i_2, i_3 are not balanced, e.g., they obey $|i_1| = O(R)$, $|i_2| = O(1)$ and $|i_3| = O(1)$, then the inversion of $\mathbf{H}$ can be found in $O(R^3)$ operations. This is the number of iterations needed to invert the dominant diagonal block $\mathbf{H}_{AA}$ because a lemma on inversion of partitioned matrices can be used to compute the rest. Thus, a computationally cheaper variant of the algorithm would use unbalanced partitions (i_1, i_2, i_3).

Note that the ordinary ALS iteration is the special case of unbalanced PALS iteration with $i_1 = \{1, \ldots, R\}$ and $i_2 = i_3 = \{\}$. It has the complexity of $O(R^4)$ mainly due to the computation of the error gradient $\mathbf{g}$.

If any of the tensor dimensions I, J, K is greater than the tensor rank, the size of $\mathbf{H}$ can be larger, indeed. In the Appendix, we show how the matrix can be inverted through the inversion

of a matrix of the size $L \times L$, where $L = R^2 - |i_1|^2 - |i_2|^2 - |i_3|^2$. The complexity is $O(R^6)$ for balanced partitionings $|i_1| \approx |i_2| \approx |i_3| \approx R/3$ and $O(R^4)$ for unbalanced partitionings, e.g., $|i_1| = R - 2$, $|i_2| = |i_3| = 1$.

B. ALS/PALS for Incomplete Data

Finally, consider CPD of incomplete tensors. It means that some elements of the tensor $\mathcal{Y}$ are not available. This task is a special case of weighted CPD, where each tensor element has its own weight in the quadratic criterion. In the case of missing entries, the corresponding weights are set to zero and all other tensor elements can have the weight one. In general, the weights can be arbitrary nonnegative real numbers. Assume that the nonnegative weights are stored in the tensor $\mathcal{W}$ of the same shape as $\mathcal{Y}$, and the target criterion is

$$\|\text{vec}(\sqrt{\mathcal{W}}) \star [\text{vec}(\mathcal{Y}) - \text{vec}(\mathbf{A}\,(\mathbf{C} \odot \mathbf{B})^T)]\|^2.$$

The ALS (traditional or partitioned) can proceed as $\boldsymbol{\theta} = \mathbf{H}^{-1}\mathbf{g}$, cf., (8), with the difference that

$$\mathbf{H} = \mathbf{J}^H \,\text{diag}(\text{vec}(\mathcal{W}))\mathbf{J} \tag{14}$$

and

$$\mathbf{g} = \mathbf{J}^H \,\text{diag}(\text{vec}(\mathcal{W}))\text{vec}(\mathcal{Y}) = \mathbf{J}^H \,\text{vec}(\mathcal{W} \star \mathcal{Y})\,. \tag{15}$$

We note that there is no significant difference between complexity of one iteration of ALS and one iteration of PALS in this case, because there is not fast inversion algorithm for the matrix $\mathbf{H}$ in (14) with general $\mathcal{W}$.

C. Enhanced Line Search

In this section, we briefly describe the ELS technique [7]. Assume one ALS/PALS iteration to be represented by the update $\mathbf{A}' \leftarrow \mathbf{A}$, $\mathbf{B}' \leftarrow \mathbf{B}$, and $\mathbf{C}' \leftarrow \mathbf{C}$. The ELS technique replaces this step by outcome of the minimization of the cost function (3) on the parametric set $\mathbf{A}'_t = \mathbf{A} + t(\mathbf{A}' - \mathbf{A})$, $\mathbf{B}'_t = \mathbf{B} + t(\mathbf{B}' - \mathbf{B})$, and $\mathbf{C}'_t = \mathbf{C} + t(\mathbf{C}' - \mathbf{C})$. The cost function takes the form of a polynomial in variable t of degree six, and its minimum can be obtained as a root of the first-order derivative of the polynomial. Finding the optimum step size t was shown to improve the convergence, if one or two factor matrices contain nearly collinear columns. The method was less successful if all three factor matrices contained nearly collinear columns.

IV. Simulations

We have tested five CPD algorithms: ALS, ALS+ELS, PALS, PALS+ELS, and LM algorithms on the case of a cubic tensor of the size $20 \times 20 \times 20$ of rank 20 with collinearity in two and three modes, respectively. We have generated three orthogonal matrices of the size 20×20, denoted $\mathbf{A}_0$, $\mathbf{B}_0$, and $\mathbf{C}_0$ at the beginning. We divide each of them to four blocks of size 20×5, i.e., $\mathbf{A}_0 = [\mathbf{A}_{01}, \mathbf{A}_{02}, \mathbf{A}_{03}, \mathbf{A}_{04}]$. Then, the factor matrix $\mathbf{A}$ was generated as $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4]$, where $\mathbf{A}_k = c\,\mathbf{A}_{0k}(:,1)\mathbf{1}_{1\times 5} + \sqrt{1-c^2}\mathbf{A}_{0k}$ for $k = 1, 2, 3, 4$, c is a free parameter, and $\mathbf{1}_{1\times 5}$ is a row vector of 1's of the size 1×5. We have set $c = 0.95$. Thanks to this definition, each of the blocks $\mathbf{A}_k$ contains five nearly collinear columns.

Similarly, $\mathbf{B}$ and $\mathbf{C}$ were composed of four blocks of nearly collinear columns obtained using corresponding blocks of $\mathbf{B}_0$ and $\mathbf{C}_0$. The decomposition was performed 100 times, with the same random initial factor matrices for all methods, but new in each trial. In Fig. 1, we show the median of the mse versus the iteration number over all 100 trials, and the minimum mse obtained by the algorithms under the test. We observed that although there was no noise added in the tensor $\mathcal{T} = [[\mathbf{A}, \mathbf{B}, \mathbf{C}]]$, we can see that none of the algorithms was able to recover the exact-fit decomposition in the 100 random runs. It is because the algorithms were stuck in false local minima. We observe, however, that the achieved mse of PALS, PALS+ELS were lower than the mse of the other methods.

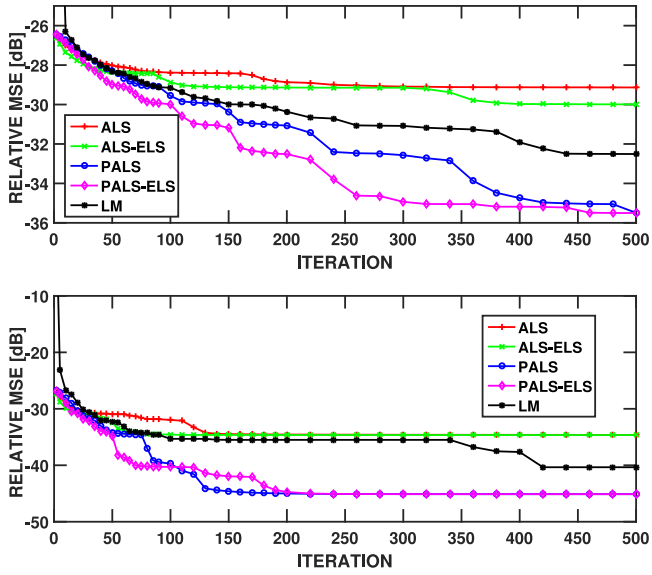

Fig. 1. Convergence in the triple bottleneck case. Median (upper diagram) and minimum (lower diagram) relative mse (fitting error) of 100 independent runs of the algorithms, initialized by random initial factor matrices.

In the second example, referred to as a double bottleneck, see Fig. 2, we keep the same $\mathbf{A}$ and $\mathbf{B}$ as in the first example, and $\mathbf{C} = \mathbf{C}_0$. The median mse is the best for PALS and PALS+ELS, but it is not close to zero again. However, the minimum mse converges to machine zero $\approx 10^{-30}$ for ALS, ALS+ELS, PALS, PALS+ELS. We have counted that this case happened in 1 out of 100 trials for ALS and ALS-ELS, but in 10 out of 100 trials of PALS and PALS-ELS. We conclude that probability of the global convergence of PALS and PALS+ELS is higher than probability of the convergence of the other techniques including the LM algorithm, which is usually considered to be the best.

In the case that the tensor rank does not exceed the tensor dimension, it is possible to improve the global convergence of the algorithms by considering a sophisticated initialization, namely the direct trilinear decomposition (DTLD) [15]. Therefore, we tested the algorithms on decomposing reduced-size tensors. The tensors were built as in the former two examples, but then they are truncated to the size $18 \times 18 \times 18$. The rank of the tensor remains to be the same, but DTLD is no longer applicable. In order not to make the decomposition that hard, we set the parameter c to a lower value, $c = 0.8$. The results are shown in Fig. 3. Now, we can see that PALS and PALS-ELS are good again, and LM started to work.

For completeness, we report computational time of the five algorithms needed to perform the 500 iterations of the algorithms, obtained on a standard PC with four kernels containing

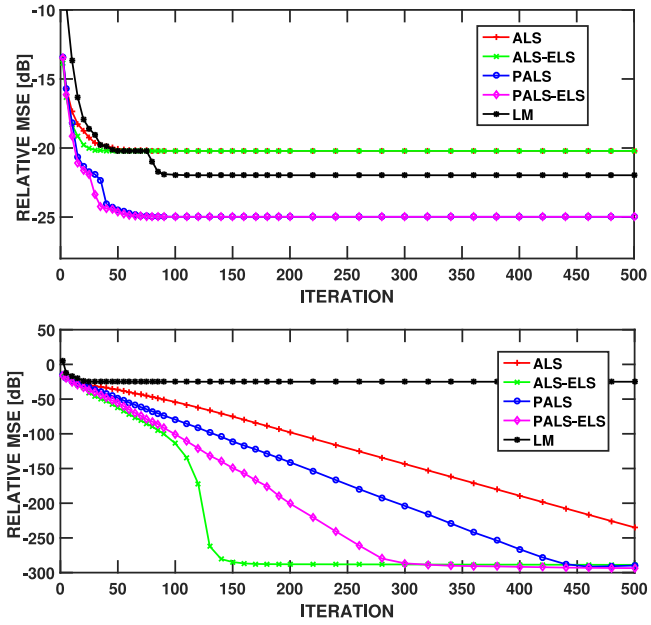

Fig. 2. Convergence in the double bottleneck case. Median (upper diagram) and minimum (lower diagram) relative mse (fitting error) of 100 independent runs of the algorithms, initialized by random initial factor matrices.

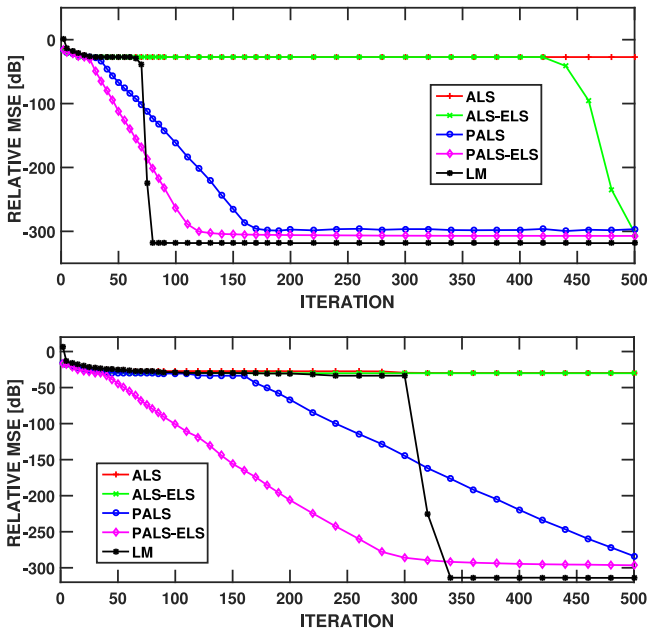

Fig. 3. Convergence in the double bottleneck and triple bottleneck case with $c = 0.8$ and reduced size of the tensor: median of 100 independent runs of the algorithms, initialized by random initial factor matrices.

3.1 GHz processors. The times were 0.51, 1.82, 9.32, 11.68, and 125.02 s, respectively.

V. Conclusion

The PALS and PALS+ELS algorithms have proved higher probability of convergence than ALS and ALS+ELS, respectively. While learning curves of the traditional methods may contain long periods of apparently none or very little decrease of the cost function followed by sudden jumps, convergence of the proposed methods is smoother. In general, the proposed methods achieve lower cost function value than the other two algorithms. The complexity of PALS and PALS+ELS algorithms is, however, higher and is between the complexity of ALS and ALS+ELS and that of the LM algorithm, unless the tensor has missing entries. In the latter case, the complexity of ALS, ALS+ELS, PALS, and PALS+ELS is about the same.

The idea of PALS can also be used in other tensor decomposition models: fully or partially nonnegative tensor factorization, or robust tensor factorizations [14].

MATLAB code of our implementation of the algorithms has been posted on the web page of the first author.

Appendix
Fast Inversion of Matrix $\mathbf{H}$

In this section, we consider only real-valued tensors, for simplicity. Following the expression of the full Hessian stated in [10, Th. 4.2], the Hessian matrix $\mathbf{H}$ in (13) can be expressed in the low-rank adjustment as

$$\mathbf{H} = \mathbf{G} + \mathbf{Z}\,\mathbf{P}_R\,\mathbf{P}\,\mathtt{diag}(\mathbf{v})\mathbf{P}^T\mathbf{Z}^T \tag{16}$$

where

$$\mathbf{G} = \mathtt{bdiag}(\mathbf{\Gamma}_1 \otimes \mathbf{I}_I, \mathbf{\Gamma}_2 \otimes \mathbf{I}_J, \mathbf{\Gamma}_3 \otimes \mathbf{I}_K) \tag{17}$$

$$\mathbf{Z} = \mathtt{bdiag}(\mathbf{I}_{d_1} \otimes \mathbf{A}, \mathbf{I}_{d_2} \otimes \mathbf{B}, \mathbf{I}_{d_3} \otimes \mathbf{C}) \tag{18}$$

$$\mathbf{v} = [\mathtt{vec}(\mathbf{Q}_{1,2}^{(3)})^T, \mathtt{vec}(\mathbf{Q}_{1,3}^{(2)})^T, \mathtt{vec}(\mathbf{Q}_{2,3}^{(1)})^T]^T \tag{19}$$

$$\mathbf{\Gamma}_1 = \mathbf{Q}_{1,1}^{(2)} \star \mathbf{Q}_{1,1}^{(3)},\ \mathbf{\Gamma}_2 = \mathbf{Q}_{2,2}^{(1)} \star \mathbf{Q}_{2,2}^{(3)}$$

$$\mathbf{\Gamma}_3 = \mathbf{Q}_{3,3}^{(1)} \star \mathbf{Q}_{3,3}^{(2)} \tag{20}$$

$d_n = |i_n|$, $\mathbf{Q}^{(1)} = \mathbf{A}^T\mathbf{A}$, $\mathbf{Q}^{(2)} = \mathbf{B}^T\mathbf{B}$, $\mathbf{Q}^{(3)} = \mathbf{C}^T\mathbf{C}$, and $\mathbf{Q}_{k,l}^{(n)} = \mathbf{Q}^{(n)}(i_k, i_l)$ is a matrix of size $d_k \times d_l$ taken from $\mathbf{Q}^{(n)}$, $\mathbf{P}_R$ is permutation matrix of size $R^2 \times R^2$ which maps $\mathtt{vec}(\mathbf{X})$ to $\mathtt{vec}(\mathbf{X}^T) = \mathbf{P}_R\,\mathtt{vec}(\mathbf{X})$ for an $R \times R$ matrix $\mathbf{X}$, and $\mathbf{P}$ is an expansion matrix of 0 and 1 which maps the vector $\mathbf{v}$ of length $L = R^2 - d_1^2 - d_2^2 - d_3^2$ to vectorization $\mathtt{vec}(\mathbf{\Xi}) = \mathbf{P}\mathbf{v}$ of matrix $\mathbf{\Xi}$ of size $R \times R$

$$\mathbf{\Xi} = \begin{bmatrix} \mathbf{0} & \mathbf{Q}_{1,2}^{(3)} & \mathbf{Q}_{1,3}^{(2)} \\ \mathbf{Q}_{1,2}^{(3)T} & \mathbf{0} & \mathbf{Q}_{2,3}^{(1)} \\ \mathbf{Q}_{1,3}^{(2)T} & \mathbf{Q}_{2,3}^{(1)T} & \mathbf{0} \end{bmatrix}. \tag{21}$$

Elements of $\mathbf{v}$ are assumed to be nonzero, otherwise we can delete zero entries, shorten this vector, and modify the expansion matrix $\mathbf{P}$ accordingly. Following the matrix inversion lemma (Woodbury identity), the inversion of $\mathbf{H}$ in (16) can be performed as

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\mathbf{P}_R\mathbf{P}\,\mathbf{\Phi}^{-1}\,\mathbf{P}^T\mathbf{Z}^T\mathbf{G}^{-1} \tag{22}$$

where $\mathbf{\Phi}$ is a matrix of size $L \times L$

$$\mathbf{\Phi} = \mathbf{P}^T\mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z}\mathbf{P}_R\mathbf{P} + \mathtt{diag}(1 \oslash \mathbf{v}) \tag{23}$$

and $\oslash$ denotes the elementwise division. Thus, the inversion of $\mathbf{H}$ can be implemented through inversions of $\mathbf{\Gamma}_1, \mathbf{\Gamma}_2, \mathbf{\Gamma}_3$, and $\mathbf{\Phi}$. The largest one is $\mathbf{\Phi}$ having the size $L \times L$.

REFERENCES

[1] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Sep. 2009.

[2] 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*. Chichester, U.K.: Wiley, 2009.

[3] P. Comon, "Tensors: A brief introduction," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 44–53, May 2014.

[4] M. J. Mohlenkamp, "Musings on multilinear fitting," *Linear Algebra Appl.*, vol. 438, pp. 834–852, 2013.

[5] M. Espig, W. Hackbusch, and A. Khachatryan, "On the convergence of alternating least squares optimisation in tensor format representations," arxiv:1506.00062v1.

[6] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *Chemometrics*, vol. 23, pp. 393–405, 2009.

[7] M. Rajih, P. Comon, and R. A. Harshman, "Enhanced line search: A novel method to accelerate PARAFAC," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1148–1171, Sep. 2008.

[8] F. Roemer and M. Haardt, "A semi-algebraic framework for approximate CP decompositions via simultaneous matrix diagonalizations (SECSI)," *Signal Process.*, vol. 93, pp. 2722–2738, Sep. 2013.

[9] P. Paatero, "A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis," *Chemometrics Intell. Lab. Syst.*, vol. 38, pp. 223–242, 1997.

[10] A. H. Phan, P. Tichavský, and A. Cichocki, "Low complexity damped Gauss–Newton algorithms for parallel factor analysis," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 126–147, Jan. 2013.

[11] P. Tichavský, A. H. Phan, and A. Cichocki, "A further improvement of a fast damped Gauss–Newton algorithm for CANDECOMP-PARAFAC tensor decomposition," presented at the *Int. Conf. Acoustics, Speech, and Signal Processing*, Vancouver, BC, Canada, May 2013, pp. 5964–5968.

[12] L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion," ESAT-SISTA, K.U. Leuven, Leuven, Belgium, *Internal Rep.* pp. 13–177, 2013.

[13] A. H. Phan, P. Tichavský, and A. Cichocki, "CANDECOMP/PARAFAC decomposition of high-order tensors through tensor reshaping," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4847–4860, Oct. 2013.

[14] R. C. Farias and P. Comon, "Canonical polyadic tensor decomposition in the presence of non Gaussian noise," presented at the *European Signal Processing Conf.*, Nice, France, 2015, pp. 1331–1335.

[15] E. Sanchez and B. R. Kowalski, "Tensorial resolution: A direct trilinear decomposition," *J. Chemometrics*, vol. 4, no. 1, pp. 29–45, 1990.