Non-Orthogonal Tensor Diagonalization

Petr Tichavský[1], Anh-Huy Phan[2] and Andrzej Cichocki[2,3]

[1] *Institute of Information Theory and Automation of the CAS, Prague, Czechia*
[2] *Riken BSI, 351-0198 Saitama, Japan*
[3] *Skolkovo Institute of Science and Technology (Skoltech), Moscow 143026, Russia*

Abstract

Tensor diagonalization means transforming a given tensor to an exactly or nearly diagonal form through multiplying the tensor by non-orthogonal invertible matrices along selected dimensions of the tensor. It has a link to an approximate joint diagonalization (AJD) of a set of matrices. In this paper, we derive (1) a new algorithm for a symmetric AJD, which is called two-sided symmetric diagonalization of an order-three tensor, (2) a similar algorithm for a non-symmetric AJD, also called a two-sided diagonalization of an order-three tensor, and (3) an algorithm for three-sided diagonalization of order-three or order-four tensors. The latter two algorithms may serve for canonical polyadic (CP) tensor decomposition, and in certain scenarios they can outperform traditional CP decomposition methods. Finally, we propose (4) similar algorithms for tensor block diagonalization, which is related to tensor block-term decomposition. The proposed algorithm can either outperform the existing block-term decomposition algorithms, or produce good initial points for their application.

Keywords: Multilinear models, canonical polyadic decomposition, parallel factor analysis, block-term decomposition, joint matrix diagonalization

1. Introduction

The approximate joint diagonalization (AJD) of a set of matrices has been recently recognized to be instrumental in signal processing, mainly because

[0]This work was supported by the Czech Science Foundation through Projects No. 14-13713S and 17-00902S.

of its importance in practical signal processing problems, such as source separation, blind beamforming, image denoising, blind channel identification for multiple-input, multiple-output (MIMO) telecommunication system, Doppler-shifted echo extraction in radar, and ICA [10].

Perhaps one of the first such algorithms is the joint approximate diagonalization of eigenmatrices (JADE) algorithm proposed in [8]. In this algorithm, the matrices under consideration are Hermitian and the considered joint diagonalizer is a unitary matrix. More recently, generalizations and/or new decompositions have been of considerable interest, see [10] and the references therein. They concern new sets of matrices, a nonunitary joint diagonalizer, and new decompositions.

The set of given matrices to be diagonalized can be viewed as a tensor. Hence, the AJD problem can, in its turn, be viewed as a special case of the tensor diagonalization, as we show later in this paper.

The concept of tensor diagonalization was first introduced by P. Comon and his co-workers [4, 5]. It works for order-three tensors of a cubic shape. The tensor diagonalization in those papers was orthogonal: it sought orthogonal matrices that would transform the given tensor into a diagonal one. The method was based on Jacobi rotations.

The tensor diagonalization studied in this paper is non-orthogonal. We consider two-sided tensor diagonalization of order-three tensors, which can be symmetric or nonsymmetric, and three-sided diagonalization of order-three or order-four tensors. All algorithms in this paper are based on the same principle. The main idea is similar to that of an AJD algorithm UWEDGE [20], it can be described in words as "diagonalize until further diagonalization is impossible", but the implementation and performance are different.

In the case of symmetric diagonalization of order-three tensors, we obtain a novel method of AJD. Utilizing nonsymmetric two-sided diagonalization of order-three tensors, on the other hand, we obtain a novel method of canonical polyadic (CP) tensor decomposition, which follows the idea of SECSI framework for CP decomposition [25].

The cases of three-sided diagonalization of order-three and order-four tensors represent another method of CP decomposition of order-three tensors, and joint approximate diagonalization of several order-three tensors, respectively. A generalization to four-sided and more-sided diagonalization of higher-order tensors is straightforward.

The tensor diagonalization methods considered in this paper can be easily modified for block diagonalization. In many applications, ordinary diago-

nalization is not appropriate, and like in independent subspace analysis [33], one rather seeks subspaces of columns that represent multidimensional signal components that should be separated or eliminated. The joint block diagonalization of the set of these matrices was studied e.g., in [11]-[16]. In the area of tensor decompositions, we speak about block-term decomposition, promoted by de Lathauwer and his co-workers [7, 8]. This decomposition means that a given tensor is rewritten as a sum of several tensors of the same size but a lower multilinear rank. The block term decomposition was used to propose a blind DS-CDMA receiver in [9].

In practice, initializing a BTD without getting captured in false local minima of the criterion function is a very challenging problem. Another difficulty is that the appropriate block sizes might not be known in advance. In some cases we have empirically found that the tensor diagonalization can be used to carry out a suitable block-term decomposition, i.e., to find appropriate block sizes, provided there is no or little noise. This has already been observed in [15].

There are a few related conference publications on the topic. The original version of this paper considered tensor diagonalization through generalized Jacobi (Givens) rotations [22]. An algorithm for two-sided diagonalization of an order-three tensor was proposed in a conference paper [30]. An algorithm for three-sided diagonalization of an order-three tensor was proposed in [31]. This paper presents a different approach to the same problem.

The paper is organized as follows: Section 2 presents the basic principles of tensor diagonalization and shows its connection to CP decomposition. In Section 3, iterative algorithms are proposed to perform the three-sided and two-sided symmetric and nonsymmetric diagonalizations, either in the real or complex domain. In Section 4, new algorithms for joint block diagonalization and block-term decomposition are developed. Section 5 presents some numerical examples, and Section 6 concludes the paper.

2. Tensor diagonalization principle

The main idea of tensor diagonalization is to find the so-called de-mixing matrices that transform a given tensor into another one that is diagonally dominant. In the AJD, we are given a set of matrices $\mathbf{R}_m$, $m = 1, \ldots, M$, and we seek the so-called de-mixing matrix $\mathbf{A}$ such that $\mathbf{A}\mathbf{R}_m\mathbf{A}^H$, $m = 1, \ldots, M$ are all diagonally dominant. It means that the off-diagonal elements

are significantly smaller in magnitude than the diagonal elements. Here, ^{H} denotes the Hermitian transpose.

There are several measures of success and several algorithms that accomplish the diagonalization, see [10] for a review. Some of them can be modified to provide approximate joint block diagonalization.

One possible modification of the problem is the nonsymmetric AJD. Here we assume again that the given matrices $\mathbf{R}_m$, $m = 1, \ldots, M$ are square, and we seek invertible matrices $\mathbf{A}$, $\mathbf{B}$ such that $\mathbf{A}\mathbf{R}_m\mathbf{B}^T$, $m = 1, \ldots, M$ are all diagonally dominant. The symbol ^{T} stands for the matrix transpose. A tensor formulation of the same problem can be following:

Let $\mathcal{T}$ be a tensor of size $n \times n \times m$ composed of the slices $\{\mathbf{R}_m\}$, $m = 1, \ldots, M$. The outcome of the diagonalization is the tensor

$$\mathcal{E} = \mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B}$$

where $\times_i$ denotes the tensor-matrix multiplication along the dimension i, $i = 1, 2$. A successful diagonalization means that $\|\mathrm{off}_2(\mathcal{E})\|_F$ is small compared to diagonal elements of $\mathcal{E}$, where $\|\cdot\|_F$ is the Frobenius norm, and $\mathrm{off}_2(\mathcal{E})$ is the operator that nullifies all diagonal elements of all frontal slices of the tensor. In other words, if $\mathcal{E}$ has elements $\mathcal{E}_{ijk}$, then $\mathrm{off}_2(\mathcal{E})$ has elements $(1 - \delta_{ij})\mathcal{E}_{ijk}$, where δ_{ij} is the Kronecker delta.

Similarly, three-sided diagonalization of an order-four tensor $\mathcal{T}$ of the size $N \times N \times N \times M$ with elements t_{ijkm}, $i, j, k = 1, \ldots, N$, $m = 1, \ldots, M$ consists of finding three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ of size $N \times N$ such that

$$\mathcal{E} = \mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \tag{1}$$

is *spatially* nearly diagonal in the sense

$$\|\mathrm{off}_3(\mathcal{E})\|_F^2 \ll \|\mathcal{E}\|_F^2$$

where the operator off_3 nullifies all elements of $\mathcal{E}$ that lie on the spatial diagonals of the tensor. To be exact, the operator off_3 acts on a tensor $\mathcal{E}$ with elements $\mathcal{E}_{ijkm}$ so that $\mathrm{off}_3(\mathcal{E})$ has elements $(1 - \delta_{ij}\delta_{jk})\mathcal{E}_{ijkm}$. In the special case $M = 1$, $\mathcal{T}$ and $\mathcal{E}$ are order-three tensors because of having only three variable indices. This diagonalization is illustrated in Fig. 1 for the case $M = 1$. The multiplication in (1) can be written as

$$e_{ijkm} = \sum_{\alpha,\beta,\gamma=1}^{N} t_{\alpha\beta\gamma m} a_{i\alpha} b_{j\beta} c_{k\gamma} \tag{2}$$

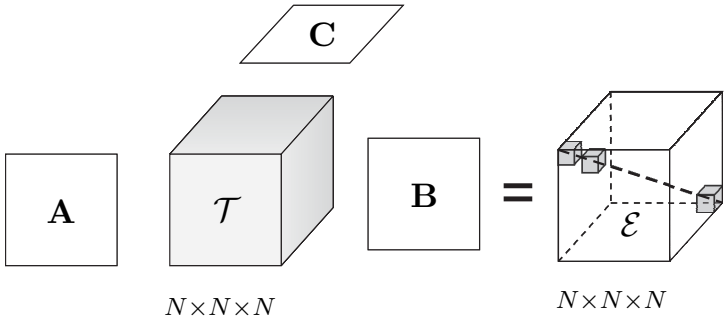

Figure 1: Three-sided tensor diagonalization transforms a tensor $\mathcal{T}$ to a diagonally dominant tensor $\mathcal{E}$ using demixing matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$: $\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \mathcal{E}$.

where e_{ijkm}, $t_{\alpha\beta\gamma m}$, $a_{i\alpha}$, $b_{j\beta}$ and $c_{k\gamma}$ are elements of tensors $\mathcal{E}$, $\mathcal{T}$ and matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, respectively.

The success of the diagonalization can be defined in different ways. The algorithms proposed in this paper are based on the principle "diagonalize until further diagonalization is impossible". Let us explain the principle on the three-sided diagonalization.

The condition that the resulting tensor $\mathcal{E}$ "cannot be diagonalized any more" means that

$$\|\mathrm{off}_3(\mathcal{E}')\|_F \geq \|\mathrm{off}_3(\mathcal{E})\|_F \tag{3}$$

for all

$$\mathcal{E}' = \mathcal{E} \times_1 \tilde{\mathbf{A}} \times_2 \tilde{\mathbf{B}} \times_3 \tilde{\mathbf{C}} \tag{4}$$

where diagonals of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$ are filled with 1's, symbolically

$$\mathrm{diag}(\tilde{\mathbf{A}}) = \mathrm{diag}(\tilde{\mathbf{B}}) = \mathrm{diag}(\tilde{\mathbf{C}}) = \mathrm{diag}(\mathbf{I}) = (1, \ldots, 1)^T \ . \tag{5}$$

The objective function to be minimized is the norm of the gradient of $\|\mathrm{off}(\mathcal{E}')\|_F$ with respect to the vector of off-diagonal elements of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{C}}$ at the point $\tilde{\mathbf{A}} = \tilde{\mathbf{B}} = \tilde{\mathbf{C}} = \mathbf{I}$. Ideally, the norm of the gradient should be zero, and the corresponding Hessian matrix should be positive definite.

The tensor diagonalization is not unique. Like in the CP decomposition, there is a permutation ambiguity, meaning that the order of rows in $\mathbf{A}$ and accordingly in $\mathbf{B}$ and $\mathbf{C}$ can be arbitrary. Moreover, there is also a scale ambiguity if $\mathrm{off}_3(\mathcal{E}) = \mathbf{0}$, i.e., when the tensor admits an exact CP decomposition, and all factor matrices are invertible. In that case, we will show that the tensor diagonalization is essentially unique, and its outcome is equivalent

to that of the CP decomposition. In other cases, the tensor diagonalization is not unique. The diagonalization might have several (perhaps infinitely many) possible outcomes, but any of them characterizes the tensor in a specific sense, and may reveal a hidden block structure in the tensor. Usually we observe that the output core tensor $\mathcal{E}$ contains many nulls (entries with negligible magnitudes) and is sparse in this sense.

3. Algorithm TEDIA

Tensor diagonalization can be achieved by cyclic application of elementary rotations for all pairs of distinct indices $i, j = 1, \ldots, N$, as it was shown in earlier versions of this paper [22]. Here, however, we present an easier way to achieve this goal by using a gradient method with an exact line search, similar to [28]. We explain it in the case of three-sided diagonalization first.

3.1. Three-Sided Diagonalization

Assume that $\mathcal{E}$ is a partially diagonalized tensor obtained during the optimization process. Let $\mathbf{G}_A$ be the gradient of the function $\|\text{off}(\mathcal{E} \times_1 \tilde{\mathbf{A}})\|_F^2$ with respect to $\tilde{\mathbf{A}}$ at $\tilde{\mathbf{A}} = \mathbf{I}$. Similarly, let $\mathbf{G}_B$ and $\mathbf{G}_C$ be gradients o $\|\text{off}(\mathcal{E} \times_2 \tilde{\mathbf{B}})\|_F^2$ with respect to $\tilde{\mathbf{B}}$ at $\tilde{\mathbf{B}} = \mathbf{I}$, and of $\|\text{off}(\mathcal{E} \times_3 \tilde{\mathbf{C}})\|_F^2$ with respect to $\tilde{\mathbf{C}}$ at $\tilde{\mathbf{C}} = \mathbf{I}$, respectively. The diagonal elements of $\mathbf{G}_A$, $\mathbf{G}_B$, and $\mathbf{G}_C$ are set to zero, because the diagonals of $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{C}}$ are fixed.

It can be shown (see Appendix A) that

$$\begin{aligned} \mathbf{G}_A &= \text{off}[(\text{off}_3(\mathcal{E}))_{(1)}\mathcal{E}_{(1)}^T] \\ \mathbf{G}_B &= \text{off}[(\text{off}_3(\mathcal{E}))_{(2)}\mathcal{E}_{(2)}^T] \\ \mathbf{G}_C &= \text{off}[(\text{off}_3(\mathcal{E}))_{(3)}\mathcal{E}_{(3)}^T] \end{aligned} \tag{6}$$

where $\mathcal{E}_{(i)}$ and $(\text{off}_3(\mathcal{E}))_{(i)}$ are the mode-i matricizations of $\mathcal{E}$ and $\text{off}_3(\mathcal{E})$, respectively.

Once $\mathbf{G}_A$, $\mathbf{G}_B$ and $\mathbf{G}_C$ are computed, we seek a scalar step of size t which minimizes the following polynomial of degree 6,

$$\varphi(t) = \|\text{off}_3(\mathcal{E} \times_1 (\mathbf{I} + t\mathbf{G}_A) \times_2 (\mathbf{I} + t\mathbf{G}_B) \times_3 (\mathbf{I} + t\mathbf{G}_C)\|_F^2 \ . \tag{7}$$

Let t_m be the minimizer of $\varphi(t)$. It can be found among the roots of a polynomial of degree 5, $\varphi'(t)$, as the one which minimizes the cost function, $\varphi(t)$. Then, an update of $\mathcal{E}$ is obtained as

$$\mathcal{E} \leftarrow \mathcal{E} \times_1 (\mathbf{I} + t_m\mathbf{G}_A) \times_2 (\mathbf{I} + t_m\mathbf{G}_B) \times_3 (\mathbf{I} + t_m\mathbf{G}_C) \ . \tag{8}$$

Table 1: Algorithm TEDIA for three-sided diagonalization

Input: Tensor $\mathcal{T}$ of size $N \times N \times N \times M$, stopping constant ε, initial demixing matrices, if they exist (identity matrices by default).

Output: Mixing matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and a core tensor $\mathcal{E} := \mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$

Repeat:

1) Compute the gradients $\mathbf{G}_A, \mathbf{G}_B, \mathbf{G}_C$ in (6)

2) Compute coefficients of the polynomial (7)

$$\begin{aligned}
\varphi(t) &= c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5 + c_6 t^6 \\
\mathcal{E}_1 &= \mathrm{off}_3(\mathcal{E} \times_1 \mathbf{G}_A + \mathcal{E} \times_2 \mathbf{G}_B + \mathcal{E} \times_3 \mathbf{G}_C) \\
\mathcal{E}_2 &= \mathrm{off}_3(\mathcal{E} \times_1 \mathbf{G}_A \times_2 \mathbf{G}_B + \mathcal{E} \times_2 \mathbf{G}_B \times_3 \mathbf{G}_C + \mathcal{E} \times_1 \mathbf{G}_A \times_3 \mathbf{G}_C) \\
\mathcal{E}_3 &= \mathrm{off}_3(\mathcal{E} \times_1 \mathbf{G}_A \times_2 \mathbf{G}_B \times_3 \mathbf{G}_C) \\
c_0 &= < \mathrm{off}_3(\mathcal{E}), \mathcal{E} >= \|\mathrm{off}_3(\mathcal{E})\|_F^2 \\
c_1 &= 2 < \mathcal{E}_1, \mathcal{E} > \\
c_2 &= \|\mathrm{off}_3(\mathcal{E}_1)\|_F^2 + 2 < \mathcal{E}_2, \mathcal{E} > \\
c_3 &= 2 < \mathcal{E}_1, \mathcal{E}_2 > + 2 < \mathcal{E}_3, \mathcal{E} > \\
c_4 &= \|\mathrm{off}_3(\mathcal{E}_2)\|_F^2 + 2 < \mathcal{E}_1, \mathcal{E}_3 > \\
c_5 &= 2 < \mathcal{E}_2, \mathcal{E}_3 > \\
c_6 &= \|\mathrm{off}_3(\mathcal{E}_3)\|_F^2
\end{aligned}$$

3) Find the root t_m of $\varphi'(t) = c_1 + 2c_2 t + 3c_3 t^2 + 4c_4 t^3 + 5c_5 t^4 + 6c_6 t^5$ minimizing $\varphi(t)$

4) Update $\mathcal{E}$, $\mathbf{A}, \mathbf{B}, \mathbf{C}$ as in (8) and (9) .

Until $\|\mathbf{G}_A\|_F + \|\mathbf{G}_B\|_F + \|\mathbf{G}_C\|_F < \varepsilon$

The estimated demixing matrices are updated as

$$\begin{aligned}
\mathbf{A} &\leftarrow (\mathbf{I} + t_m \mathbf{G}_A)\mathbf{A} \\
\mathbf{B} &\leftarrow (\mathbf{I} + t_m \mathbf{G}_B)\mathbf{B} \\
\mathbf{C} &\leftarrow (\mathbf{I} + t_m \mathbf{G}_C)\mathbf{C}
\end{aligned} \tag{9}$$

The algorithm is summarized in Table 1. In this Table, the notation $< \mathcal{E}_i, \mathcal{E}_j >$ denotes a scalar product of the tensors $\mathcal{E}_i$, $\mathcal{E}_j$, i.e., a sum of the entries of the elementwise product of these tensors.

The computational complexity of the TEDIA algorithm is $O(N^4 M)$ operations per iteration. In the case of $M = 1$, the complexity per iteration is

roughly the same as the complexity of one iteration in the Alternating Least Squares (ALS) algorithm with the Enhanced Line Search (ELS). The convergence of TEDIA appears to be smoother than that of ALS or ALS/ELS, namely in difficult scenarios, as we show in the simulation section.

3.2. Two-Sided Diagonalization

A modification of TEDIA to two-sided diagonalization is straightforward. The difference is that

$$\begin{aligned} \mathbf{G}_A &= \text{off}[(\text{off}_2(\mathcal{E}))_{(1)}\mathcal{E}_{(1)}^T] \\ \mathbf{G}_B &= \text{off}[(\text{off}_2(\mathcal{E}))_{(2)}\mathcal{E}_{(2)}^T] \end{aligned} \tag{10}$$

and the polynomial $\varphi(t)$ is of degree 4,

$$\varphi(t) = \|\text{off}_2(\mathcal{E} \times_1 (\mathbf{I} + t\mathbf{G}_A) \times_2 (\mathbf{I} + t\mathbf{G}_B)\|_F^2 \ . \tag{11}$$

In order to minimize this polynomial, we need to solve a polynomial equation $\varphi'(t) = 0$, which is of order three. This can be done algebraically and the method becomes numerically simpler than within the three-sided diagonalization. For large tensor sizes, however, the complexity of the polynomial rooting is negligible, of order $O(1)$, compared to complexity of computing the coefficients of the polynomials, which is $O(N^4)$.

In the case of symmetric two-sided diagonalization, there is only one demixing matrix $\mathbf{A}$, the corresponding gradient matrix is

$$\mathbf{G}_A = \text{off}[(\text{off}_2(\mathcal{E}))_{(1)}\mathcal{E}_{(1)}^T],$$

and the polynomial $\varphi(t)$ would be of degree 4 again,

$$\varphi(t) = \|\text{off}_2(\mathcal{E} \times_1 (\mathbf{I} + t\mathbf{G}_A) \times_2 (\mathbf{I} + t\mathbf{G}_A)\|_F^2 \ . \tag{12}$$

4. Application in CP tensor decompositions

A natural utilization of TEDIA is in the CP tensor decomposition. It was shown in [25] and [26] (so-called SECSI framework) that two-sided tensor diagonalization can be applied in CP tensor decomposition. Similarly, the three-sided diagonalization can also be used for this purpose. Let us discuss this issue in more detail.

First of all, the tensor might have a shape different from $N \times N \times N$ or $N \times N \times N \times M$. A direct tensor diagonalization makes no sense then

because the diagonalizing matrices must be square and invertible. The Tucker compression is advised in this case. The Tucker compression is based on finding orthogonal matrices $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3$ such that

$$\mathcal{T} \approx \mathcal{T}_C \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \times_3 \mathbf{Q}_3$$

where $\mathcal{T}_C$ is the compressed tensor of the required shape, and $\times_i$ denotes a mode-i tensor-matrix multiplication. The Tucker compression can be achieved by the HOOI algorithm [17], see [18] for more literature on this topic. Note that TEDIA can serve as a tool for the Tucker compression as well; it would only need to be a block version of it (see the next section). However, its performance in the compression appears to not be as good as that of the conventional Tucker compression.

Note that higher-order tensors can also be decomposed through the CP decomposition of order-three tensors through tensor re-shaping [32]. Therefore we shall focus on CP decomposition of a cube-shaped tensor.

Another remark is on computational accuracy. Tensor diagonalization is not a statistically efficient procedure of CP decomposition, because it is not equivalent to the maximum likelihood estimate. The performance of a TEDIA-based procedure in terms of the CP decomposition cost function can be improved by a suitable post-processing in which the adequate maximum likelihood cost function is optimized.

The theoretical CP decomposition might involve rank-deficient factor matrices. In that case, the optimum de-mixing matrices would not be invertible. This is in conflict with the tensor diagonalization, which is always assumed to produce invertible demixing matrices. TEDIA therefore might not be useful in such cases, and either block TEDIA or block term decomposition would be more appropriate [6].

Assume that a tensor $\mathcal{T}$ is diagonalized by three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ such that the product $\mathcal{E} = \mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$ cannot be diagonalized any more in the sense of Section 2. Further, assume that the tensor is of size $N \times N \times N$ and its rank is $R \leq N$. It may occur that the core tensor $\mathcal{E}$ has only at most R significant nonzero elements, while the magnitudes of the other elements are negligible. Zeroing other than the R significant elements, we get a rank-R approximation of the core tensor, which implies a rank-R approximation of the original tensor. If the significant elements lie on the main spatial diagonal of the tensor, we have obtained ordinary diagonalization and CP decomposition with regular factor matrices. However, if the

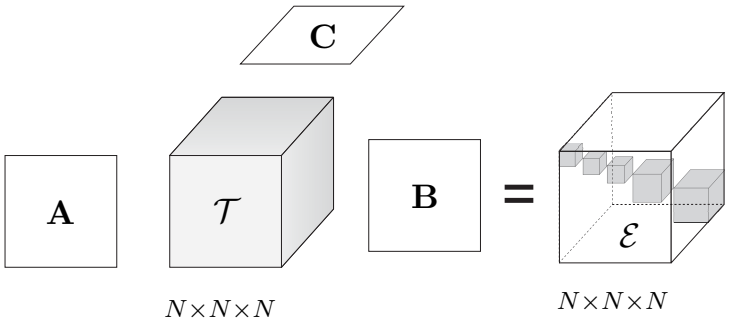

Figure 2: Tensor block diagonalization transforms a tensor $\mathcal{T}$ to a block diagonal tensor $\mathcal{E}$ by factor matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$: $\mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \mathcal{E}$.

multilinear rank of the tensor is not (N, N, N), such outcome is impossible, and not all significantly large elements lie on the diagonal. In that case, some factor matrices in the CP approximation of the tensor would be rank deficient.

5. Block diagonalization

The tensor block diagonalization is a natural generalization of the diagonalization considered in the previous sections. It can have the form of symmetric or nonsymmetric two-sided block diagonalization, or three-sided diagonalization, as illustrated in Figure 2. In this section, we assume that the block structure is known. The block structure of $\mathcal{E}$ can be represented by an indicator tensor $\mathcal{M}$ of the same size as $\mathcal{E}$ which contains zeros in the place of the desired blocks, and ones elsewhere. In the special case when there is only a single block, we receive a novel method of the Tucker compression.

We can consider the operator $\text{boff}_{\mathcal{M}}$ which nullifies all block elements of the input tensor,

$$\text{boff}_{\mathcal{M}}(\mathcal{E}) = \mathcal{E} \star \mathcal{M} \tag{13}$$

where $\star$ stands for the elementwise product. Now, we apply the principle "diagonalize until further diagonalization is impossible" to get a block diagonalization procedure similar to Section 4. In the case of the three-sided block diagonalization, we define $\mathbf{G}_A$, $\mathbf{G}_B$ and $\mathbf{G}_C$ as the gradient of the function $\|\text{boff}_{\mathcal{M}}(\mathcal{E} \times_1 \mathbf{A})\|_F^2$ with respect to $\mathbf{A}$ at $\mathbf{A} = \mathbf{I}$, gradient of $\|\text{boff}_{\mathcal{M}}(\mathcal{E} \times_2 \mathbf{B})\|_F^2$ with respect to $\mathbf{B}$ at $\mathbf{B} = \mathbf{I}$, and gradient of $\|\text{boff}_{\mathcal{M}}(\mathcal{E} \times_3 \mathbf{C})\|_F^2$ with respect to $\mathbf{C}$ at $\mathbf{C} = \mathbf{I}$, respectively. The diagonal elements of $\mathbf{G}_A$, $\mathbf{G}_B$, and $\mathbf{G}_C$ are

set to zero, because the diagonals of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are fixed. The result is

$$\begin{aligned} \mathbf{G}_A &= \text{off}[(\mathcal{M} \star \mathcal{E})_{(1)} \mathcal{E}_{(1)}^T] \\ \mathbf{G}_B &= \text{off}[(\mathcal{M} \star \mathcal{E})_{(2)} \mathcal{E}_{(2)}^T] \\ \mathbf{G}_C &= \text{off}[(\mathcal{M} \star \mathcal{E})_{(3)} \mathcal{E}_{(3)}^T] \end{aligned} \tag{14}$$

Finally, we find the optimum step-size t_m by minimizing the function

$$\varphi(t) = \|\text{boff}_{\mathcal{M}}(\mathcal{E} \times_1 (\mathbf{I} + t\mathbf{G}_A) \times_2 (\mathbf{I} + t\mathbf{G}_B) \times_3 (\mathbf{I} + t\mathbf{G}_C))\|_F^2 \ . \tag{15}$$

Then, the core tensor $\mathcal{E}$ and the de-mixing matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are updated as in (8) and (9), respectively.

6. Blind Block Diagonalization

Blind block diagonalization is understood to be block diagonalization undertaken without knowing the block structure in advance. In [15] it was shown that an ordinary approximate joint diagonalization algorithm for matrices can be used to obtain a joint block diagonalization of these matrices. It appears that a similar link exists between the tensor diagonalization and tensor block-term decomposition. A given tensor may not admit full diagonalization, but may still admit block-diagonalization, as is shown schematically in Fig. 2. We can assume that the diagonal blocks cannot be diagonalized any further, because their tensor ranks exceed their dimensions, and each of the blocks separately obeys the zero gradient condition $\mathbf{G}_A = \mathbf{G}_B = \mathbf{G}_C = \mathbf{0}$. It is straightforward to prove that a compound block diagonal tensor obeys this condition as well.

Note that the diagonalization is invariant under row permutations of matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. In other words, if $\pi(\cdot)$ is an arbitrary permutation of $(1, \ldots, N)$, then

$$\mathcal{E}' = \mathcal{T} \times_1 \mathbf{A}' \times_2 \mathbf{B}' \times_3 \mathbf{C}' \tag{16}$$

is an equivalent diagonalization where $\mathcal{E}'$, $\mathbf{A}'$, $\mathbf{B}'$ and $\mathbf{C}'$ have elements $e'_{ijkm} = e_{\pi(i),\pi(j),\pi(k),m}$, $a'_{i,\alpha} = a_{\pi(i),\alpha}$, $b'_{j,\beta} = b_{\pi(j),\beta}$, and $c'_{k\gamma} = c_{\pi(k),\gamma}$, respectively, for $i, j, k, \alpha, \beta, \gamma = 1, \ldots, N$.

It follows that the block structure may not be revealed after a mixing and demixing (diagonalization) unless a suitable permutation π has been found. The permutation should be the same in all modes in order to guarantee that

all diagonal elements of the original tensor will appear on the diagonal of the permuted tensor.

The degree of diagonality or block diagonality of a tensor $\mathcal{E}$ can be judged via the matrix $\mathbf{F}$ of size $N \times N$, having elements

$$f_{ij} = \sum_{m=1}^{M} \sum_{k=1}^{N} |e_{kijm}| + |e_{ikjm}| + |e_{ijkm}| \tag{17}$$

The tensor $\mathcal{E}$ is said to be diagonal (block diagonal) if and only if $\mathbf{F}$ is diagonal (block diagonal).

The main idea is to cluster the indices $\{1, \ldots, N\}$ according to their similarities: the similarity of two indices i, j, defined as $f_{ij} + f_{ji}$, measures how likely it is that i and j belong to the same block (both in $\mathbf{F}$ and in $\mathcal{E}$). If $f_{ij} + f_{ji}$ exceeds a certain threshold, it is inferred that i and j belong to the same block. In the opposite case, when $f_{ij} \approx f_{ji} \approx 0$, it is still possible that i and j belong to the same block, because they may be connected ("communicate") through a sequence of some other indices. Only if no such sequence exists, then i and j belong to different blocks.

Assume that there are K clusters, $1 \leq K \leq N$, each of cardinality κ_i, $i = 1, \ldots, K$. We seek a permutation π such that $\pi(1), \ldots, \pi(\kappa_1)$ belong to the first cluster, $\pi(\kappa_1+1), \ldots, \pi(\kappa_1+\kappa_2)$ belong to the second cluster, and so on. The order of the clusters is arbitrary; one may prefer to have the clusters sorted according to their cardinalities.

Such permutation can be found, e.g., using the well-known reverse Cuthill-McKee algorithm (RCM)[21], implemented in Matlab$^{\text{TM}}$ as function *symrcm*. The RCM algorithm, applied to the matrix $\mathbf{F}$, reveals an ordering of the columns and rows such that the reordered matrix is block diagonal, if such ordering exists.

In the noisy case, when the blocks of the core tensors are fuzzy, we have better experience with standard clustering methods, such as *hierarchical clustering with the average-linking policy* [19], which take $\mathbf{F}$ for a similarity matrix. In short, the algorithm begins with a trivial clustering which consists of N singletons, and in each subsequent step it merges those two clusters that have the maximum average similarity between their members. The algorithm is summarized in Table 2.

Note that even if the desired block structure of the core tensor is known in advance, it might be useful to apply the blind diagonalization and clustering as a pre-processing step for the ordinary (non-blind) block diagonalization,

Table 2: Clustering of components in MATLAB notation

Input: Similarity matrix $\mathbf{F}$ of size $N \times N$ (destroyed in the procedure)
Output: Permutation J of indices $1, \ldots, N$ such that $\mathbf{F}(J, J)$ is approximately block diagonal.

$\mathbf{F}(1 : N + 1 : N^2) = 0$ % nullify diagonal of $\mathbf{F}$, i.e., $\mathbf{F}(i, i) = 0$ for $i = 1, \ldots, N$
$S = repmat((1 : N)', 1, N)$; % auxiliary array of size $N \times N$, i.e., $S(i, j) = i$
$L = ones(N, 1)$; % auxiliary array of the clusters lengths
For $i = N : -1 : 2$ % in the $i - th$ step two of i clusters are merged.
    $[m1, \sim] = \max(\mathbf{F}(1 : i, 1 : i))$;
    $[m2, j] = \max(m1)$; % (j, k) be the clusters with the highest similarity
    $[m3, k] = \max(\mathbf{F}(:, j))$;
    if $j > k$, $aux = j$; $j = k$; $k = aux$; end % to make sure that $k > j$
    $Lnew = L(j) + L(k)$; % length of the new cluster, union of j, k
    $Snew = [S(j, 1 : L(j)), S(k, 1 : L(k)), zeros(1, N - Lnew)]$;
        % ... indices belonging to the new cluster
    $ind = [1 : j - 1, j + 1 : k - 1, k + 1 : i]$; % indices of the other clusters
    $Fnew = (L(j) * \mathbf{F}(j, ind) + L(k) * \mathbf{F}(k, ind))/Lnew$;
        % similarities between the new cluster and the other clusters
    $\mathbf{F} = [0, Fnew; Fnew', \mathbf{F}(ind, ind)]$; % update of the similarity matrix
    $S = [Snew; S(ind, :)]$; % update indices in the clusters
    $L = [Lnew; L(ind)]$; % update the cluster lengths
End
$J = S(1, :)$;
End

because it may reduce the number of iterations of the latter algorithm needed to achieve convergence.

7. Simulations

7.1. Example 1: CP decomposition

In this example, we apply the three-sided tensor diagonalization and other methods to decompose a cubic tensor of size $20 \times 20 \times 20$ of rank 20 with collinearity in three modes. First, we generated three orthogonal matrices of the size 20×20, denoted $\mathbf{A}_0$, $\mathbf{B}_0$ and $\mathbf{C}_0$. We divided each of them to four blocks of size 20×5, i.e., $\mathbf{A}_0 = [\mathbf{A}_{01}, \mathbf{A}_{02}, \mathbf{A}_{03}, \mathbf{A}_{04}]$. The factor matrix

$\mathbf{A}$ was then built of four blocks $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3, \mathbf{A}_4]$, 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, $\mathbf{A}_{0k}(:,1)$ is the first column of $\mathbf{A}_{0k}$, and $\mathbf{1}_{1\times 5}$ is a row vector of 1's of the size 1×5. We set $c = 0.99$. In this setting, the angle between columns in each block is 8.1^o. The condition number of $\mathbf{A}$ is 34.99 . Similarly, $\mathbf{B}$ and $\mathbf{C}$ were constructed using corresponding blocks of $\mathbf{B}_0$, and $\mathbf{C}_0$. Finally, we added an i.i.d. Gaussian noise component to each tensor element so that a chosen signal-to-noise ratio (SNR) is attained [24]. The tensor has the property that even with zero noise, traditional iterative CP decomposition methods have serious difficulties in finding the right global minimum of the cost function due to the presence of many secondary local minima, unless the iterations begin in close vicinity of the right solution. Thus, the decomposition is hard for all existing methods.

If there is no noise, then the true CP decomposition can be found through the Direct Tri-Linear Decomposition [29], which is based on generalized eigendecomposition of a matrix pair. However, DTLD is quite sensitive to additive noise. Even if the variance of the additive noise is so low that the signal-to-noise ratio is at 90 dB, the estimates provided by the method are no longer good enough. If the outcome of DTLD is used to initialize ALS or ALS-ELS, the latter algorithms frequently remain trapped in secondary local minima. On the other hand, the TEDIA algorithm, namely its two-sided version, appears to be robust against the wrong initialization.

In Figure 3 we compare median performance of DTLD and median learning curves for the four methods: ALS, ELS, TEDIA-2 and TEDIA-3 taken from 200 independent runs of the algorithms. In each run we generate new factor matrices to build the tensor, and new additive noise. In the left-hand-side diagram, the factor/demixing matrices are initialized by identity matrices (this approach is equivalent to random initialization). In the latter diagram, the algorithms are initialized by the outcome of the DTLD. We can see that TEDIA-2 achieves the best median relative fitting error in the former case, while in the latter case its performance was close to that of ALS-ELS. Figure 4 presents probability of achieving a given fitting error in the experiment. It confirms the previous observation that TEDIA-2 is the best performing method in the sense of global convergence in the given scenario.

To be exact, the DTLD algorithm that we used is based on the generalized eigendecomposition (GE) of the first two frontal slices of the tensor. In [29], it was proposed to use the GE of the two frontal slices of the tensor compressed to the size of $N \times N \times 2$. However, in our scenario, the former variant was

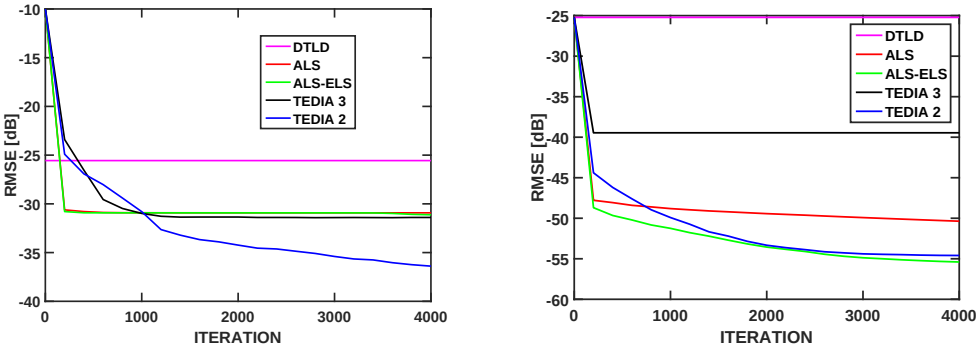

Figure 3: Median relative mean-squared fitting error of DTLD, ALS, ALS+ELS, TEDIA 3 and TEDIA 2, (1) initialized by identity matrices (left diagram), and (2) initialized by the outcome of DTLD (right diagram) for $c = 0.99$ and SNR=90 dB.

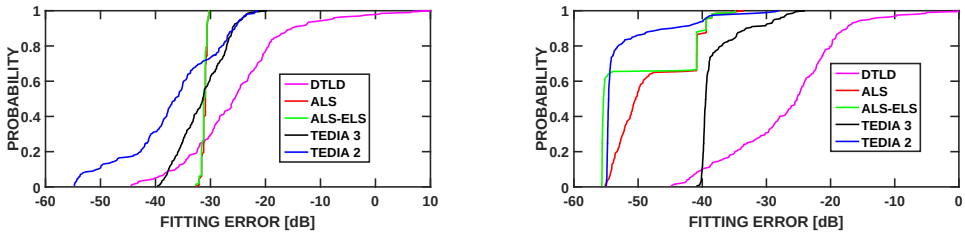

Figure 4: Probability of attaining a given fitting error for DTLD, ALS, ALS+ELS, TEDIA 3 and TEDIA 2, (1) initialized by identity matrices (left diagram), and (2) initialized by the outcome of DTLD (right diagram).

more accurate.

If the input SNR is higher, above 100 dB, DTLD works perfectly and no margin for improving the results remains. For SNR lower than 80 dB, performance of TEDIA does not look good in terms of the fitting criterion, because it optimizes a different criterion.

7.2. Example 2: Approximate Joint Block AJD

We have compared performance of seven approximate joint block (AJD) algorithms: (1) U-WEDGE completed by clustering of rows of a demixing matrix: this algorithm is blind to the assumed block structure. This algorithm is used to initialize all subsequent ones; (2) algorithm JBD NCG [6], (3) the LLAJD algorithm [13], and finally (4) the block two-sided TEDIA algorithm proposed in this paper.

We consider ten target matrices, each having four diagonal blocks of the size 10×10. The blocks were taken at random, different at each simulation trial: each block is taken as the product $\mathbf{X}_{km}\mathbf{X}_{km}^T$, where $\mathbf{X}_{km}$ is Gaussian-distributed with zero mean and variance one, mutually independent entries and independent in different slices. Here, k is the block index, $k = 1, 2, 3, 4$ and m is the slice index, $m = 1, \ldots, 10$. Thus the resultant core tensor $\mathcal{S}$ has dimensions of $40 \times 40 \times 10$ and is composed of four blocks of the size $10 \times 10 \times 10$. The noisy tensors in the simulations are not simply generated by adding a random noise but they are built of sample covariance matrices of T random vectors having the required theoretical covariance values. To be specific, let $\mathbf{R}_m$ be the $m-$th frontal slice of the original tensor, and $\mathbf{R}_m = \mathbf{X}_m\mathbf{X}_m^T$, $\mathbf{X}_m$ be block diagonal with blocks $\mathbf{X}_{km}$, then the corresponding noisy tensor slice is $\widehat{\mathbf{R}}_m = \frac{1}{T}\mathbf{X}_m\mathbf{Y}_T\mathbf{Y}_T^T\mathbf{X}_m^T$, where $\mathbf{Y}_T$ is a random matrix of the size $N \times T$ with Gaussian i.i.d. entries of zero mean and unit variance. The resultant tensor is block dominant, but not exactly block diagonal.

The mixing matrix $\mathbf{A}$ was taken at random, independently in each simulation trial. We computed it from a random unitary matrix $\mathbf{A}_0$ as $\mathbf{A} = c\mathbf{A}_0(:, 1)\mathbf{1}_{1\times 40} + \sqrt{1-c^2}\mathbf{A}_0$, like in Section 7.1, to obtain mixing matrix with collinear columns. We set $c = 0.8$. The mixture is the tensor $\mathcal{T} = \mathcal{S}\times_1\mathbf{A}\times_2\mathbf{A}$. The block structure of the core tensor $\mathcal{S}$ implies the tensor $\mathcal{T}$ decomposition as a sum

$$\mathcal{T} = \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3 + \mathcal{T}_4 \ . \tag{18}$$

Each of the tensors $\mathcal{T}_i$, $i = 1, 2, 3, 4$, has the size $40\times 40\times 10$ and a multilinear rank of $(10, 10, 10)$. The block-term decomposition has several indeterminacies, e.g., the bases of the independent subspaces can be quite arbitrary, but the decomposition (18) is unique up to the order of the terms in the sum. Therefore, we shall measure success of the approximate joint block diagonalization by mean-squared errors of appropriately sorted estimates of $\mathcal{T}_i$, $i = 1, 2, 3, 4$.

We studied performance of four JBD algorithms: UWEDGE followed by collecting the columns so that the block structure is revealed, JBD of [13] initialized by the outcome of UWEDGE, JBD of Lahat et.al [13] with default (random) initialization, NGG algorithm of [12], and Block TEDIA initialized by the outcome of UWEDGE. Results are presented in Figure 5. First, we observe that the best performance is obtained by the algorithm of Lahat [13] that has been initialized by the outcome of UWEDGE. It is because the data generation model is in accord with this algorithm. The second best algorithm

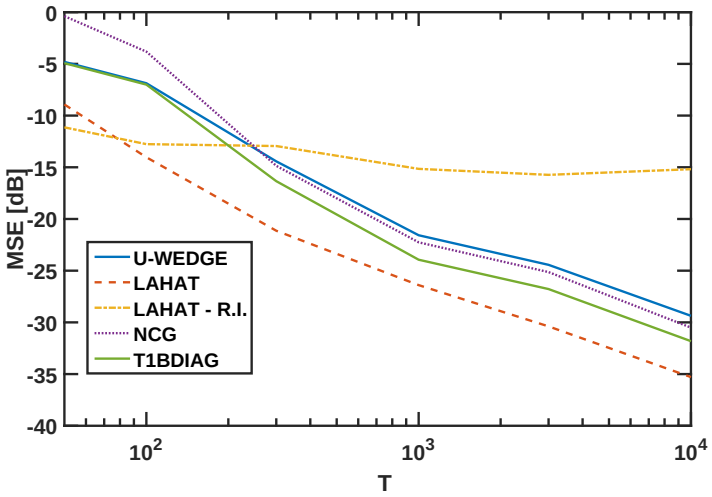

Figure 5: Mean-squared fitting error in dB of separated tensors of multilinear rank (10,10,10) for UWEDGE, JBD [13], JBD [13] with random initialization, NCG algorithm [12] and symmetric block TEDIA versus the sample size T.

is the block TEDIA. The running times were 0.32 s, 0.99 s, 4.1 s and 0.98 s for UWEDGE, JBD [13], NCG [12] and TEDIA, respectively.

7.3. Example 3: Three-sided block diagonalization

The initial tensor of the size $20 \times 20 \times 20$ was block diagonal, with four random blocks along its main diagonal, each of the size $5 \times 5 \times 5$. These blocks were computed as a diagonal tensor having 1's on its main diagonal plus Gaussian random noise with zero mean and unit variance.

The initial tensor was the desired core tensor $\mathcal{E}$. The factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ were taken at random, as in the previous example. We compute them from random unitary matrices $\mathbf{A}_0$, $\mathbf{B}_0$, $\mathbf{C}_0$ as $\mathbf{A} = c\mathbf{A}_0(:,1)\mathbf{1}_{1\times 20} + \sqrt{1-c^2}\mathbf{A}_0$, e.t.c, like in Section 7.1, We set $c = 0.5$ and $c = 0.9$, respectively. The mixture is the tensor $\mathcal{T} = \mathcal{E} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$. The block structure of the core tensor $\mathcal{S}$ implies the tensor $\mathcal{T}$ decomposition as in (18).

Each of the tensor $\mathcal{T}_i$, $i = 1, 2, 3, 4$, has a multilinear rank of $(5, 5, 5)$. The block-term decomposition has several indeterminacies, e.g., the bases of the independent subspaces can be quite arbitrary, but the decomposition (18) is unique up to the order of the terms in the sum. Therefore, we shall measure success of the approximate joint block diagonalization by mean-

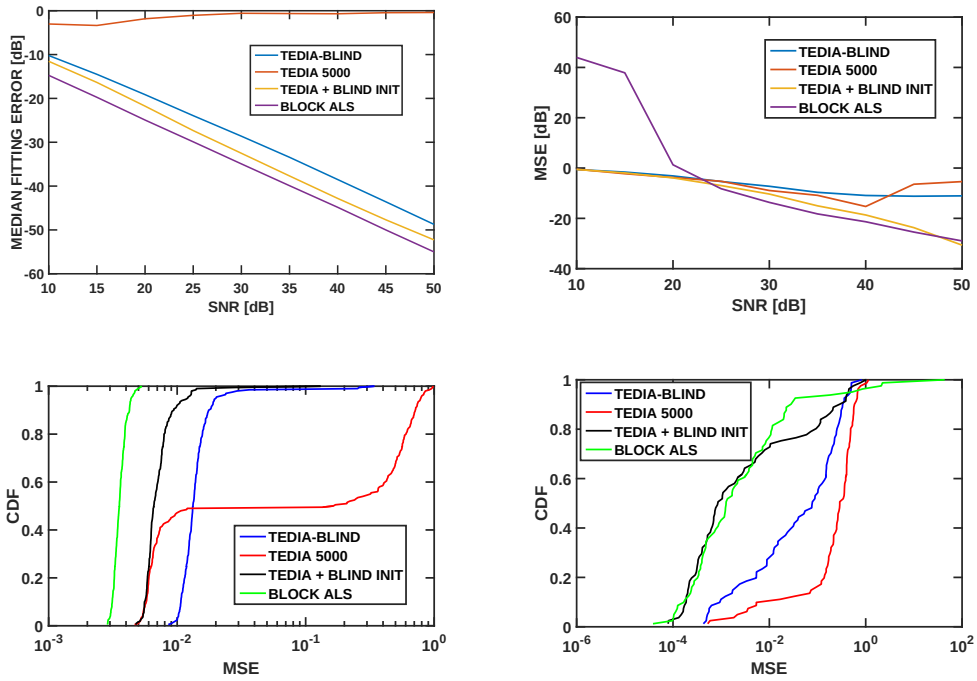

Figure 6: Median error of separated tensors of a multilinear rank of (5,5,5) and cumulative distribution function of the error for (1) blind TEDIA with 1000 iterations, (2) for block TEDIA with 5000 iterations, (3) block TEDIA with 1000 iterations but initialized by outcome of the blind TEDIA, and (4) block ALS with 100 iterations initialized by the blind TEDIA. Left diagram: $c = 0.5$, SNR=20 dB; right diagram: $c = 0.9$, SNR=30 dB.

squared errors of appropriately sorted estimates of $\mathcal{T}_i$, $i = 1, 2, 3, 4$. Gaussian noise is added into the tensor $\mathcal{T}$ according to pre-specified SNR values.

We have tested four BTD algorithms: (1) Blind TEDIA with 1000 iterations, i.e., three-sided diagonalization followed by permuting the columns so that the block structure is revealed, (2) Fixed block-size TEDIA with 5000 iterations and random initialization (3) Fixed block-size TEDIA with 1000 iterations after being initialized by outcome of the blind TEDIA (4) Block Alternating Least Squares with 100 iterations after it is initialized by the blind TEDIA. Results of 100 independent trials are presented in Figure 6.

We can see that convergence of block TEDIA is relatively slow, 5000 iterations are not enough unless the algorithm is initialized properly, e.g., by the outcome of the blind TEDIA. All three of these algorithms work relatively well even at low SNR values. We note, however, that the block

TEDIA without proper initialization has a certain probability of failure, and this probability increases with growing SNR. On the other hand, block ALS fails in the difficult scenario with $c = 0.9$ and low SNR, despite having been initialized by the blind TEDIA. If the algorithm is initialized randomly, it usually fails.

Note that one run of the blind TEDIA (1000 iterations) takes 1.36 seconds, the additional 1000 iterations of the block TEDIA requires an additional 1.44 seconds. The 5000 iterations of the block TEDIA takes 9.35 seconds, and one run of the block ALS with 100 iterations requires 71.9 seconds: it is very slow compared to TEDIA.

8. Conclusions

TEDIA is a technique of non-orthogonal tensor diagonalization and block diagonalization. In difficult scenarios with many local minima and a little additive noise it can outperform traditional methods of CP tensor decomposition such as the alternating least squares (ALS), and ALS with the exact line search. This happens because TEDIA optimizes a different cost function than the other methods. In particular, the two-sided version of TEDIA appears to be less sensitive to the presence of local minima of the cost function than the traditional methods. Complexity of TEDIA is similar to that of ALS, $O(N^4)$ operations per iteration.

In the area of the block term decomposition, the situation is similar. We have shown that TEDIA allows fitting the assumed block structure of the tensor directly, but usually it is useful to begin the separation with the blind TEDIA first.

Potential applications can be found in DS-CDMA systems or in tensor deconvolution, in particular in feature extraction and other areas.

Matlab code of the proposed technique is posted on the website of the first author.

Appendix A

In this Appendix we derive the expression (6) for $\mathbf{G}_A$. The other gradients, $\mathbf{G}_B$ and $\mathbf{G}_C$ follow from the symmetry of the problem.

Let

$$\mathcal{E}' = \mathcal{E} \times_1 \mathbf{A} \ .$$

The (k, ℓ, m)-th element of the tensor is

$$e'_{k\ell m} = \sum_{\alpha=1}^{N} e_{\alpha\ell m} a_{\alpha k} \ .$$

Then

$$\frac{\partial e'_{k\ell m}}{\partial a_{ij}} \ = \ \sum_{\alpha} e_{\alpha\ell m}\delta_{\alpha i}\delta_{kj} = e_{i\ell m}\delta_{kj}$$

and

$$\begin{aligned}
\|\mathrm{off}_3(\mathbf{E}')\|^2 &= \sum_{(k,\ell,m)\neq(k,k,k)} (e'_{k\ell m})^2 \\
\frac{\partial\|\mathrm{off}_3(\mathbf{E}')\|^2}{\partial a_{ij}} &= 2 \sum_{(k,\ell,m)\neq(k,k,k)} e_{k\ell m}\frac{\partial e'_{k\ell m}}{\partial a_{ij}} \\
&= 2 \sum_{(k,\ell,m)\neq(k,k,k)} e_{k\ell m}e_{i\ell m}\delta_{kj} \\
&= 2 \sum_{(\ell,m)\neq(j,j)} e_{j\ell m}e_{i\ell m} \\
&= 2 \sum_{\ell,m=1}^{N} [\mathrm{off}_3(\mathcal{E})]_{j\ell m}e_{i\ell m} \ .
\end{aligned}$$

The multiplication factor two is immaterial, because we take the optimum step size. A similar computation holds true for the gradient $\mathbf{G}_A$ (14) in the block TEDIA.

References

[1] P. Comon, X. Luciani, A.L.F. de Almeida, *Tensor decompositions: alternating least squares and other tales*, Journal of Chemometrics, 23 (2009), pp. 393-405.

[2] T.G. Kolda and B.W. Bader, "Tensor decompositions and applications, *SIAM Review*, 51 (2009), pp. 455–500.

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

[4] P. Comon, *Tensor Diagonalization, A useful Tool in Signal Processing,* in IFAC-SYSID, 10th IFAC Symposium on System Identification, M. Blanke and T. Soderstrom, ed., Copenhagen, Denmark, 1994, pp. 77–82.

[5] P. Comon, M. Sorensen, *Tensor Diagonalization by Orthogonal Transforms*, Rapport de recherche ISRN I3S/RR-2007-06-FR, Universite Nice, Sophia Antipolis. The paper is available on the Internet as http://www.i3s.unice.fr/ mh/RR/2007/RR-07.06-P.COMON.pdf.

[6] A. Stegeman, *Candecomp/Parafac: from diverging components to a decomposition in block terms,* SIAM J. Matrix Anal. Appl., 33 (2012), pp. 291–316.

[7] L. De Lathauwer, *Decompositions of a higher-order tensor in block terms - Part II: Definitions and uniqueness*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1033-1066.

[8] L. De Lathauwer and D. Nion, *Decompositions of a higher-order tensor in block terms - Part III: Alternating Least Squares Algorithms*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1067-1083.

[9] D. Nion and L. De Lathauwer, *A Block Component Model based blind DS-CDMA receiver,* IEEE Tran. Signal Proc., 56 (2008), pp. 5567-5579.

[10] G. Chabriel, M. Kleinsteuber , E. Moreau, H. Shen, P. Tichavský, and A. Yeredor, *Joint Matrices Decompositions and Blind Source Separation*, IEEE Signal Proc. Magazine, 31 (2014), pp. 34-43.

[11] H. Ghennioui, F. El Mostafa, N. Thirion-Moreau, A. Adib, and E. Moreau, *A Nonunitary Joint Block Diagonalization Algorithm for Blind Separation of Convolutive Mixtures of Sources*, IEEE Signal Proc. Letters, 14 (2007), pp. 860–863.

[12] D. Nion, *A Tensor Framework for Nonunitary Joint Block Diagonalization*, IEEE Trans. Signal Proc., 59 (2011), pp. 4585–4594.

[13] D. Lahat, J. Cardoso and H. Messer, *Second-Order Multidimensional ICA: Performance Analysis*, IEEE Trans. Signal Proc., 60 (2012), pp. 4598 - 4610.

[14] P. Tichavský, Z. Koldovský, *Algorithms for nonorthogonal approximate joint block diagonalization*, in Proc. EUSIPCO 2012, Bucharest, Romania, 2012, pp. 2094–2098.

[15] P. Tichavský, A. Yeredor, and Z. Koldovský, *On Computation of Approximate Joint Block-Diagonalization using Ordinary AJD*, in Proc. Latent Variable Analysis and Independent Component Analysis, Lecture Notes in Computer Science 7191, F. Theis et al., eds., 2012, pp. 163-171.

[16] Y. Cai, D. Shi, and S. Xu, *A Matrix Polynomial Spectral Approach for General Joint Block Diagonalization*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 839-863.

[17] L. De Lathauwer, B. De Moor, and J. Vandewalle, *On the best rank-1 and rank-(R1,R2,. . . ,RN) approximation of higher-order tensors,* SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324-1342.

[18] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53 78.

[19] G. Gan, C. Ma, and J. Wu, *Data Clustering Theory, Algorithms, and Applications*, SIAM, 2007.

[20] P. Tichavský and A. Yeredor, *Fast Approximate Joint Diagonalization Incorporating Weight Matrices*, IEEE Trans. on Signal Proc., 57 (2009), pp. 878-891.

[21] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proc. 24th Nat. Conf. ACM, 1969, pp. 157172.

[22] P. Tichavský, A.H. Phan, A. Cichocki, *Tensor diagonalization - a new tool for PARAFAC and block-term decomposition*, http://arxiv.org/abs/1402.1673

[23] A.H. Phan, P. Tichavský, and A. Cichocki, "Low Rank Tensor Deconvolution", *Proc. ICASSP 2015*, Brisbane, pp. 2169–2173.

[24] P. Tichavský, A.H. Phan, A. Cichocki, "Partitioned Alternating Least Squares Technique for Canonical Polyadic Tensor Decomposition", *IEEE Signal Processing Letters*, vol. 23, no.7, pp. 993–997, July 2016.

[25] F. Roemer and M. Haardt, *A semi-algebraic framework for approximate CP decompositions via Simultaneous Matrix Diagonalizations (SECSI),* Signal Proc., 93 (2013), pp. 2722-2738.

[26] K. Naskovska, M. Haardt, P. Tichavský, G. Chabriel, J. Barrere, *Extension of the semi-algebraic framework for approximate CP decompositions via non-symmetric simultaneous matrix diagonalization*, in Proc. ICASSP 2016, Shanghai, China, 2016, pp. 2971-2975.

[27] P. Tichavský, A.H. Phan, A. Cichocki, *Two-Sided Diagonalization of Order-Three Tensors*, in Proc. EUSIPCO 2015, Nice, pp. 998–1002.

[28] M. Rajih, P. Comon, and R. A. Harshman, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. Appl., 30 (2008), pp.1148–1171.

[29] E. Sanchez and B.R. Kowalski, *Tensorial resolution: a direct trilinear decomposition*, J. Chemometrics, 4 (1990), pp. 2945.

[30] H. Shen and M. Kleinsteuber, *A Block-Jacobi Algorithm for Non-Symmetric Joint Diagonalization of Matrices*, in Proc. Latent Variable Analysis and Independent Component Analysis, Lecture Notes in Computer Science 9237, E. Vincent et al., eds., 2015, pp. 320327.

[31] V. Maurandi, E. Moreau, *Fast Jacobi algorithm for non-orthogonal joint diagonalization of non-symmetric third-order tensors*, in Proc. EUSIPCO 2015, Nice, pp. 1311–1315.

[32] A.H. Phan, P. Tichavský and A. Cichocki, *CANDECOMP/PARAFAC decomposition of high-order tensors through tensor reshaping*, IEEE Tran. Signal Proc. 61 (2013), pp. 4847–4860.

[33] F. J. Theis, *Towards a general independent subspace analysis*, Advances in Neural Information Processing Systems, 19 (2007), pp. 1361-1368.