TWO-SIDED DIAGONALIZATION OF ORDER-THREE TENSORS

Petr Tichavský[1], *Anh Huy Phan*[2] *and Andrzej S Cichocki*[2]

[1]Institute of Information Theory and Automation,
P.O.Box 18, 182 08 Prague 8, Czech Republic
[2] Brain Science Institute, RIKEN, Wakoshi, Japan.

ABSTRACT

This paper presents algorithms for two-sided diagonalization of order-three tensors. It is another expression for joint non-symmetric approximate diagonalization of a set of square matrices, say $\mathbf{T}_1, \ldots, \mathbf{T}_M$: We seek two non-orthogonal matrices $\mathbf{A}$ and $\mathbf{B}$ such that the products $\mathbf{A}\mathbf{T}_m\mathbf{B}^T$ are close to diagonal in a sense. The algorithms can be used for a block tensor decomposition and applied e.g. for tensor deconvolution and feature extraction using the convolutive model.

Index Terms— Multilinear models; canonical polyadic decomposition; parallel factor analysis; block-term decomposition; joint matrix diagonalization

1. INTRODUCTION

The two-sided diagonalization of an order-three tensor $\mathcal{T}$ can be viewed as a non-symmetric joint diagonalization of a set of matrices. A given tensor of the size $N \times N \times M$ is represented by M matrices of the size $N \times N$, say $\mathbf{T}_1, \ldots, \mathbf{T}_M$. The nonsymmetric matrix diagonalization means finding $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$ such that the matrices

$$\mathbf{E}_m = \mathbf{A}\mathbf{T}_m\mathbf{B}^T \tag{1}$$

are diagonally dominant in a sense. Here, ^{T} stands for the matrix transposition. The *non-symmetric* Approximate Joint Diagonalization (AJD) is an extension of a *symmetric* AJD, see [1], which has found many applications in blind audio source separation or in operational modal analysis [2]. Conceptually, the diagonalization is similar to three-sided diagonalization of order-3 tensor or three-sided diagonalization of order-4 tensor [3]. In the latter case one seeks three matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ such that their product with the given tensor of the size $N \times N \times N \times M$ along its first three dimensions is close to a spatially diagonal tensor. With a slight abuse of notation we keep the same name for the two-sided diagonalization, TEDIA (TEnsor DIAgonalization).

There are many existing algorithms for a symmetric AJD, where $\mathbf{A} = \mathbf{B}$ or $\mathbf{A} = \mathbf{B}^*$, "*" denotes complex conjugation, but not that many algorithms for the nonsymmetric AJD [4]. The algorithm presented in this paper produces, as a solution, matrices $\mathbf{A}$ and $\mathbf{B}$ of determinant 1 such that the demixed tensor $\boldsymbol{\mathcal{E}}$ composed of the slices $\mathbf{E}_m$ in (1) obeys certain stopping condition described later, called a *block revealing condition*. Basically, the condition means that all further optimum elementary rotations are trivial. Thus the solution is not unique in the strict sense, but the subspaces spanned by columns of $\mathbf{A}$, $\mathbf{B}$, are unique.

The main purpose of the diagonalization is the block-term decomposition (BTD) [5, 6]. In the case of the two-sided diagonalization, it is a non-symmetric approximate joint block-diagonalization of a set of matrices. Note that the *symmetric* joint block-diagonalization of a set of matrices is subject of the papers [7]–[10].

For the BTD, the main existing algorithms are alternating least squares (ALS) [6] and non-linear least squares (NLS) [11]. These algorithms have a drawback of a poor global convergence. If they are not initialized properly, they are frequently trapped in false local minima. Multiple random initialization of these algorithms may help only a little. The tensor diagonalization serves as a tool finding a suitable approximate block-term decomposition, which can be improved by the dedicated methods.

Consider the off-diagonality operator $\mathrm{off}_2(\cdot)$ which transforms the core tensor $\boldsymbol{\mathcal{E}}$ into a tensor of the same size, having the elements $(1-\delta_{ij})e_{ijk}$ for $i, j, \ldots, N$ and $k = 1, \ldots, M$, where δ_{ij} is the Kronecker delta. In other words, the operator nullifies the elements e_{iik}, $i = 1, \ldots, N$ and $k = 1, \ldots, M$, and keeps the remaining tensor elements unchanged. Then, diagonality of the core tensor can be measured in terms of the Frobenius norm of $\mathrm{off}_2(\boldsymbol{\mathcal{E}})$, denoted $\|\mathrm{off}_2(\boldsymbol{\mathcal{E}})\|_F$.

As in the accompanying paper [3], TEDIA does not minimize the above mentioned criterion directly, but aims at achieving the block-revealing condition.

The rest of the paper is organized as follows: in Section 2 an iterative algorithm is proposed to perform the diagonalization either in the real or complex domains. Section 3 describes usage of the procedure for block diagonalization. Section 4 presents applications, and Section 5 concludes the paper.

[0]This work was supported by the Czech Science Foundation through Project No. 14-13713S.

2. ALGORITHM TEDIA

In this section we present two versions of the algorithm: sequential and parallel.

Assume that $\boldsymbol{\mathcal{E}}$ is a partially diagonalized tensor obtained during the optimization process. The sequential TEDIA proceeds by a cyclic application of elementary rotations for all pairs of distinct indices $i, j = 1, \ldots, N$. The elementary rotations are represented by matrices $\mathbf{A}_{ij}(\boldsymbol{\theta})$, $\mathbf{B}_{ij}(\boldsymbol{\theta})$ and which have the same 0-1 elements as the $N \times N$ identity matrix except for the 2×2 submatrices with the row and column indices (i, j), $i \neq j$, of the form

$$[\mathbf{A}_{ij}(\boldsymbol{\theta})]_{[ij],[ij]} = \begin{bmatrix} \sqrt{1+\theta_1\theta_2} & \theta_2 \\ \theta_1 & \sqrt{1+\theta_1\theta_2} \end{bmatrix} \quad (2)$$

$$[\mathbf{B}_{ij}(\boldsymbol{\theta})]_{[ij],[ij]} = \begin{bmatrix} \sqrt{1+\theta_3\theta_4} & \theta_4 \\ \theta_3 & \sqrt{1+\theta_3\theta_4} \end{bmatrix} \quad (3)$$

so that the Frobenius norm of off$_2(\boldsymbol{\mathcal{E}}')$ is minimized, where $\boldsymbol{\mathcal{E}}'$ is the transformed tensor

$$\boldsymbol{\mathcal{E}}' = \boldsymbol{\mathcal{E}}'(\boldsymbol{\theta}) = \boldsymbol{\mathcal{E}} \times_1 \mathbf{A}_{ij}(\boldsymbol{\theta}) \times_2 \mathbf{B}_{ij}(\boldsymbol{\theta}) \quad (4)$$

and $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4)^T$. Note that the elementary rotations are constructed so that their determinants are one. The matrix $\mathbf{A}_{ij}(\boldsymbol{\theta})$ depends only on the first two elements of the vector $\boldsymbol{\theta}$, and similarly $\mathbf{B}_{ij}(\boldsymbol{\theta})$ depends only on the second pair of elements of $\boldsymbol{\theta}$. The core tensor and the factor matrices $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{B}}$ are updated as

$$\boldsymbol{\mathcal{E}} \longleftarrow \boldsymbol{\mathcal{E}}'(\boldsymbol{\theta}), \quad \widetilde{\mathbf{A}} \longleftarrow \widetilde{\mathbf{A}}\,\mathbf{A}_{ij}^{-1}(\boldsymbol{\theta}), \quad \widetilde{\mathbf{B}} \longleftarrow \widetilde{\mathbf{B}}\,\mathbf{B}_{ij}^{-1}(\boldsymbol{\theta}) \quad (5)$$

where $\boldsymbol{\theta}$ is an approximate minimizer of the cost function $\Xi(\boldsymbol{\theta}) = \frac{1}{2}\|\text{off}_2(\boldsymbol{\mathcal{E}}'(\boldsymbol{\theta}))\|_F^2$. The parameter value $\boldsymbol{\theta}$ is found as

$$\boldsymbol{\theta} = -\mathbf{H}^{-1}\mathbf{g} \quad (6)$$

where $\mathbf{g}$ and $\mathbf{H}$ are the gradient and approximate Hessian of $\Xi(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ at the point $\boldsymbol{\theta} = \mathbf{0}$. The optimization can proceed in the same way in the real-domain and in the complex domain, if $\mathbf{g}$ and $\mathbf{H}$ are computed as

$$\mathbf{g} = \mathbf{J}^H \text{vec}(\text{off}_2(\boldsymbol{\mathcal{E}})), \qquad \mathbf{H} = \mathbf{J}^H\mathbf{J} \quad (7)$$

where the superscript ^{H} denotes Hermitian transpose, and

$$\mathbf{J} = \mathbf{J}(\boldsymbol{\theta}) = \left.\frac{\partial \text{vec}(\text{off}_2(\boldsymbol{\mathcal{E}}'(\boldsymbol{\theta})))}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\mathbf{0}} \quad (8)$$

is the Jacobi matrix of the size $N^3 \times 4$ evaluated at $\boldsymbol{\theta} = \mathbf{0}$.

A straightforward computation leads to the result

$$\mathbf{g}^{(i,j)} = \begin{bmatrix} \lambda_{ij} - \mathbf{u}_{ji}^H\mathbf{u}_{ii} \\ \lambda_{ij}^* - \mathbf{u}_{ij}^H\mathbf{u}_{jj} \\ \mu_{ij} - \mathbf{u}_{ij}^H\mathbf{u}_{ii} \\ \mu_{ij}^* - \mathbf{u}_{ji}^H\mathbf{u}_{jj} \end{bmatrix} \quad (9)$$

and $\mathbf{H}^{(i,j)} =$

$$\begin{bmatrix} \lambda_{jj} - \|\mathbf{u}_{ji}\|^2 & 0 & 0 & \mathbf{u}_{jj}^H\mathbf{u}_{ii} \\ 0 & \lambda_{ii} - \|\mathbf{u}_{ij}\|^2 & \mathbf{u}_{ii}^H\mathbf{u}_{jj} & 0 \\ 0 & \mathbf{u}_{jj}^H\mathbf{u}_{ii} & \mu_{jj} - \|\mathbf{u}_{ij}\|^2 & 0 \\ \mathbf{u}_{ii}^H\mathbf{u}_{jj} & 0 & 0 & \mu_{ii} - \|\mathbf{u}_{ji}\|^2 \end{bmatrix} \quad (10)$$

where $\mathbf{u}_{ij}$ is a column vector with elements e_{ijm}, $m = 1, \ldots, M$, $\lambda_{ij} = \sum_{k,\ell=1}^N e_{jk\ell}^* e_{ik\ell}$, $\mu_{ij} = \sum_{k,\ell=1}^N e_{kj\ell}^* e_{ki\ell}$. Note that the Hessian in (10) is, after permuting its rows and columns, block diagonal; its inversion can be found by inverting two matrices of the size 2×2.

We propose to do a single step of the Gauss-Newton method for each pair (i, j) only in each sweep, provided that the cost function decreases.

In some cases it may happen, however, that the cost function increases, or there is some other problem, namely in optimization in the real domain: some of the conditions $1 + \theta_1\theta_2 \geq 0$, $1+\theta_3\theta_4 \geq 0$, $1+\theta_5\theta_6 \geq 0$ may be violated. In the latter cases we suggest not to accept such step, but to switch to the damped Gauss-Newton method, and apply an update

$$\boldsymbol{\theta} = -(\mathbf{H} + \mu\mathbf{I}_4)^{-1}\mathbf{g} \quad (11)$$

where μ is a suitable positive constant such that the regularity conditions are fulfilled and the criterion decreases. Optionally it is possible to add a few more steps of the damped Gauss-Newton method for the same (i, j) until the cost function stops reducing.

The computational complexity of building up one Hessian matrix and the gradient is $O(N^2M)$, the solution of the 4×4 system has a complexity of $O(1)$. Since there are $O(N^2)$ pairs (i, j), each sweep needs $O(N^4M)$ operations.

The sweeps are iterated until the Euclidean norm of optimum parameter θ in every elementary rotations becomes smaller than a user chosen constant, say $\varepsilon = 10^{-6}$. The number of the sweeps needed to achieve a convergence depends, indeed, on the dimension of the problem, presence of additive noise, and other factors. The algorithm can start, for example, with $\mathbf{A} = \mathbf{B} = \mathbf{I}$. When the algorithm terminates, all gradients $\mathbf{g}^{(i,j)}$ are (approximately) zero. The condition $\mathbf{g}^{(i,j)} = \mathbf{0}$ for all pairs (i, j), $i \neq j$, is the block-revealing condition.

Finally, we describe a parallel version of the TEDIA algorithm. Here, a_{ij}, a_{ji}, b_{ij}, b_{ji} (elements of $\boldsymbol{\theta}$) are computed as solutions of the linear systems (6) keeping the core tensor the same for all pairs (i, j). Since the core tensor is kept constant, the computation can proceeds for all pairs (i, j) in parallel. In Matlab^{TM} it is realized through elementwise vector and matrix operations, so that tedious "for" loops are avoided.

These elements are stored in auxiliary matrices $\mathbf{A}_{aux}$, $\mathbf{B}_{aux}$. The diagonal of these matrices is computed separately to achieve the condition $\det \mathbf{A}_{aux} = \det \mathbf{B}_{aux} = 1$. The initial diagonals of $\mathbf{A}_{aux}$, $\mathbf{B}_{aux}$ are set to ones. Then, the diagonal of these matrices is adjusted by appropriate constants c_A, c_B such that

$$\det(\mathbf{A}_{aux} - c_A\mathbf{I}) = \det(\mathbf{B}_{aux} - c_B\mathbf{I}) = 1.$$

We note that $\det(\mathbf{A}_{aux} - c\mathbf{I})$ is a characteristic polynomial of $\mathbf{A}_{aux}$. Then c_A is selected as the root of the polynomial minus 1 that is closest to zero. Similarly c_B is computed.

Once $\mathbf{A}_{aux}$, $\mathbf{B}_{aux}$ of determinant 1 are found, the core tensor is updated as

$$\begin{array}{lcl} \boldsymbol{\mathcal{E}} & \longleftarrow & \boldsymbol{\mathcal{E}} \times_1 \mathbf{A}_{aux} \times_2 \mathbf{B}_{aux} \\ \widetilde{\mathbf{A}} & \longleftarrow & \widetilde{\mathbf{A}}\,\mathbf{A}_{aux}^{-1}, \qquad \widetilde{\mathbf{B}} \longleftarrow \widetilde{\mathbf{B}}\,\mathbf{B}_{aux}^{-1}\,. \end{array} \tag{12}$$

If $\mathrm{off}_2(\boldsymbol{\mathcal{E}})$ is not reduced, then the step is not accepted and the algorithm computes $\mathbf{A}_{aux}$, $\mathbf{B}_{aux}$ using $\boldsymbol{\theta}$'s obtained through the damped Gauss-Newton formula (11). Advantage of the parallel implementation is an increased speed compared to the sequential version.

3. BLOCK DIAGONALIZATION

As in [10, 3], it holds that the block-diagonalization can be obtained by applying TEDIA and sorting rows in matrices $\mathbf{A}$ and $\mathbf{B}$ so that the blocks in the core tensors are revealed.

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

$$f_{ij} = \sum_{m=1}^{M} |e_{ijm}| \tag{13}$$

The tensor $\boldsymbol{\mathcal{E}}$ is said diagonal (block-diagonal) if and only if $\mathbf{F}$ is diagonal (block-diagonal). Then, $\mathbf{F}$ or its symmetrized version $\mathbf{F} + \mathbf{F}^T$ may be considered as a measure of similarity between columns in the mixing matrices, or an indicator of "probability" that they belong to the same block.

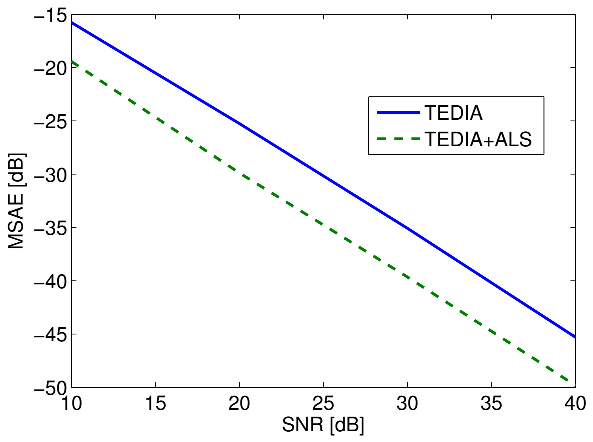
Such permutation can be found e.g. using the well known reverse Cuthill-McKee algorithm (RCM)[12], implemented in Matlab$^{\mathrm{TM}}$ as function *symrcm*, applied to the matrix $\mathbf{F}$. In the noisy case, when the blocks of the core tensors are fuzzy, we have better experiences with standard clustering methods, such as *hierarchical clustering with the average-linking policy* [13], taking $\mathbf{F}$ as similarity matrix [3].

4. EXAMPLES

4.1. Block term decomposition

First, we tested performance of TEDIA on simulated data. The initial (core) tensor was computed as block diagonal, with three blocks of the size $5 \times 5 \times 15$, generated randomly, normally distributed with zero mean and variance 1. The tensor has the size $15 \times 15 \times 15$, i.e. $N = M = 15$.

The mixing matrices $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ are obtained as products $\mathbf{Q}_A\mathbf{A}_c$ and $\mathbf{Q}_B\mathbf{A}_c$, respectively, where $\mathbf{Q}_A, \mathbf{Q}_B$ are random orthogonal matrices of the size 15×15, and $\mathbf{A}_c = \varepsilon \mathbf{1}_{15\times 15} + \gamma \mathbf{I}_{15}$, where $\varepsilon = (\sqrt{1 - c + Nc} - \sqrt{1-c})/N$,
$\gamma = \sqrt{1-(N-1)\varepsilon^2} - \varepsilon$, and c is a parameter to control colinearity of columns of the demixing matrices.



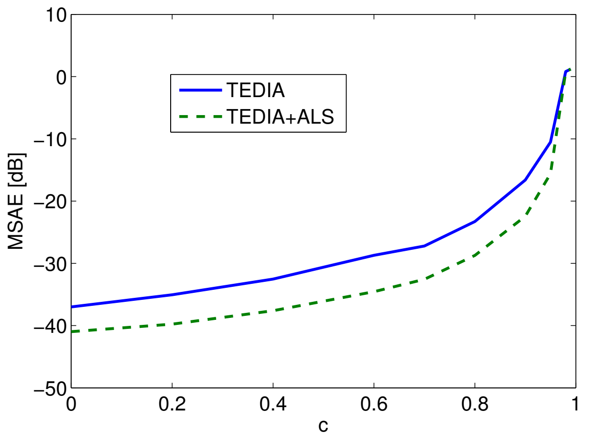


Fig. 1. Mean square angular error in dB of subspaces in partitioned factor matrices of the tensor obtained using TEDIA and of ALS initialized by the outcome of TEDIA, for $c = 0.2$ as a function of the input SNR (upper diagram) and for SNR=30dB and varying c (lower diagram).

Accuracy of the estimation of the blocks of the mixing matrices $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ will be determined in terms of mean square angles between the linear space spanned by the quadruples $(\mathbf{a}_1, \ldots, \mathbf{a}_5)$, $(\mathbf{a}_6, \ldots, \mathbf{a}_{10})$, $(\mathbf{a}_{11}, \ldots, \mathbf{a}_{15})$, $(\mathbf{b}_1, \ldots, \mathbf{b}_5)$, $(\mathbf{b}_6, \ldots, \mathbf{b}_{10})$, $(\mathbf{b}_{11}, \ldots, \mathbf{b}_{15})$, and their estimates (MSAE). The angles between the subspaces can be computed in Matlab$^{\mathrm{TM}}$ by function "subspace".

Performance of TEDIA will be compared with performance the ALS algorithm [6], which is initialized by the outcome of TEDIA. The result – an average MSAE of 100 independent trials for each pair (c,SNR) as a function of the input SNR for $c = 0.2$ and as a function of c for SNR=30 dB – are plotted in Fig. 1. We can see that the ALS algorithm exceeds performance of the plain TEDIA by 3-4 dB. In the noiseless case, TEDIA separates the blocks perfectly. The main difference is in the computation time. The TEDIA proceeds in

0.45s on average. Execution of the ALS takes 6.6s on average, and the initialization is crucial. Without the initialization using TEDIA, the ALS is not reliable.

We conclude that TEDIA itself is not statistically optimum estimator of the subspaces, it is inferior to ALS. Its advantage is twofold: (1) it does not need to know sizes of the blocks in advance, and (2) it finds a good solution without the unreliable multiple random initializations, and if used prior ALS or NLS, it significantly reduces the number of their iterations.

4.2. Diagonalization for DS-CDMA receiver

This example illustrates efficacy of TEDIA for block term decomposition used in DS-CDMA system [14]. Consider a system of R users transmitting at the same time within the same bandwidth, frames of J symbols spread by DS-CDMA codes of length I towards an array of K antennas over a specular multipath channel with P discrete paths. The tensor $\boldsymbol{\mathcal{Y}}$ of size $I \times J \times K$ comprising of samples $y_{i,j,k}$ of received signals can be expressed through a convolutive model given in a block component model

$$\boldsymbol{\mathcal{Y}} = \sum_{r=1}^{R} \boldsymbol{\mathcal{H}}_r \times_2 \mathbf{S}_r \times_3 \mathbf{A}_r \tag{14}$$

where $\mathbf{S}_r$ are $(J \times L)$ Toeplitz matrices containing interfering symbols of the r-th user, tensors $\boldsymbol{\mathcal{H}}_r$ are of size $I \times L \times P$, vectors $\text{vec}(\boldsymbol{\mathcal{H}}_r(:,:,p))$ represent spreading waveforms of the r-th user over the p-th path, and the $(K \times P)$ matrices $\mathbf{A}_r$ holds responses of the k-th antenna to the signal of the r-user coming from the p-th path. For detailed derivation of the model, we refer to [14].

The expression (14) can be written as

$$\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{H}} \times_2 [\mathbf{S}_1, \ldots, \mathbf{S}_R] \times_3 [\mathbf{A}_1, \ldots, \mathbf{A}_R] \tag{15}$$

where $\boldsymbol{\mathcal{H}}(i,:,:) = \text{blkdiag}(\boldsymbol{\mathcal{H}}_1(i,:,:,), \ldots, \boldsymbol{\mathcal{H}}_R(i,:,:,))$. Assume that $LR \leq J$ and $PR \leq I$. Then, the blocks can be estimated by applying two-side tensor diagonalization to the tensor $\boldsymbol{\mathcal{Y}}$ along modes 2 and 3.

Sub matrices $\mathbf{S}_{0,r}$ of size $(J \times L)$ are then factorized to find the Toeplitz matrices $\widetilde{\mathbf{S}}_{0,r}$ as $\mathbf{S}_{0,r} = \widetilde{\mathbf{S}}_{0,r}\mathbf{Q}_r$ [18][1]. The matrices $\mathbf{A}_{0r}$ and $\widetilde{\mathbf{S}}_{0,r}$ are then used as a good initial value for the block decomposition in (14).

For our computer simulation, we set the spreading codes of length $I = 20$ or 30, short frames of $J = I$ BPSK-symbols, $K = I$ antennas, $L = 5$ interfering symbols, $P = 5$ major paths per user and $R = 2$ users. Parameters were randomly drawn from the Gaussian distribution, and the tensor data was degraded by additive Gaussian noise with zero mean at signal to noise ratio SNR = 10, 20, 30 and 40 dB. As reported in [14], the ALS algorithm for this tensor decomposition needs a large number of iterations when the condition number of the matrix $\mathbf{A} = [\mathbf{A}_1, \mathbf{A}_2]$ is high, say 100, 200. In our simulations, the condition number was 100. The ALS algorithm [14, 19] will stop when difference of consecutive approximation error $|\varepsilon - \varepsilon_{old}| < 10^{-5}\varepsilon_{old}$ where $\varepsilon = \|\boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}}\|_F$, or when the number of iterations exceeds 3000.

[1]An implementation of the Toeplitz factorisation is provided at http://perso-etis.ensea.fr/ nion

Table 1. Performance comparison for the ALS algorithm using random initialization, or TEDIA s outputs in Example 4.2 at different SNR = 10, 20, 30, 40 dB. Execution time is given in second, while error means the approximation error $\|\boldsymbol{\mathcal{Y}} - \hat{\boldsymbol{\mathcal{Y}}}\|_F$.

Algorithm		SNR (dB)			
		10	20	30	40
$I = J = K = 20$					
TEDIA	BER	0.1896	0.1128	0.0177	0.0012
	Time	0.8	0.7	0.7	0.7
	Error	0.898	0.656	0.268	0.263
ALS+ Random	BER	0.0175	0.0026	0.0176	0.0003
	Time	18.4	18.4	16.4	13.5
	Error	0.432	0.137	0.048	0.014
ALS+ TEDIA	BER	0.0017	0.0004	0	0
	Time	10.6	6.8	4.2	3.9
	Error	0.432	0.137	0.043	0.014
$I = J = K = 30$					
TEDIA	BER	0.0783	0.0343	0.0017	0.0006
	Time	0.8	0.7	0.8	0.7
	Error	0.813	0.656	0.268	0.263
ALS+ Random	BER	0	0	0.0012	0
	Time	19.7	18.3	23.1	24.5
	Error	0.417	0.138	0.065	0.021
ALS+ TEDIA	BER	0	0	0	0
	Time	13.9	10.6	8.6	10.1
	Error	0.416	0.132	0.041	0.013

Table 1 reports the performance of TEDIA and the ALS algorithm initialized using random value or TEDIA's output. When the noise was low, e.g., SNR $\geq$ 30 dB, the sequences estimated from output of TEDIA without any further decomposition achieved relatively low bit error rate (BER). TEDIA executed within less than 1 second on a computer with 3.33 GHz core i7 CPU. When initializing by random values, the ALS algorithm took 14-19 seconds for the test case $I = J = K = 20$, but did not achieve good BERs as this algorithm initialized using TEDIA. When $I = J = K = 30$, execution time of the ALS algorithm was approximately 20-25 seconds. When initializing using TEDIA, the ALS algorithm converged earlier and achieved better BERs when $I = J = K = 20$ and perfect estimation for the case of $I = J = K = 30$.

5. CONCLUSION

We have presented the technique of non-orthogonal two-side diagonalization of order-three tensors. This technique can be used for CP and block tensor decomposition in the real or complex domains as a good initial solution for other dedicated methods of the decomposition. Applications can be found in tensor deconvolution, in particular for CDMA receivers, but also in other areas, e.g. in the tensor deconvolution [20].

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

6. REFERENCES

[1] G. Chabriel, M. Kleinsteuber , E. Moreau, H. Shen, P. Tichavský, and A. Yeredor, "Joint Matrices Decompositions and Blind Source Separation", *IEEE Signal Processing Magazine*, Vol.31, No. 3, pp. 34-43, March 2014.

[2] J. Antoni and S. Chauhan, "A study and extension of second-order blind source separation to operational modal analysis," *J. Sound Vib.*, vol. 332, no. 4, pp. 1079-1106, Feb. 2013.

[3] P. Tichavský, A.H. Phan, A. Cichocki, "Non-Orthogonal Tensor Diagonalization, a Tool for Block Tensor Decompositions", http://arxiv.org/abs/1402.1673

[4] M. Sorensen, P. Comon, S. Icart, and L. Deneire, "Approximate tensor diagonalization by invertible transforms," In *Proc. EUSIPCO'09*, Glasgow, Scotland, August 24–28, 2009, pp. 500–504.

[5] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms - Part II: Definitions and uniqueness", *SIAM J. Matrix Anal. Appl.*, Vol. 30, No. 3, pp. 1033-1066, 2008.

[6] 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.*, Vol. 30, No. 3, pp. 1067-1083, 2008.

[7] D. Nion, "A Tensor Framework for Nonunitary Joint Block Diagonalization", *IEEE Trans. Signal Processing*, Vol. 59, No. 10, pp. 4585–4594, 2011.

[8] D. Lahat, J. Cardoso and H. Messer, "Second-Order Multidimensional ICA: Performance Analysis", *IEEE Trans. Signal Processing*, Vol. 60, No. 9, pp. 4598 - 4610, 2012.

[9] P. Tichavský, Z. Koldovský, "Algorithms for nonorthogonal approximate joint block-diagonalization", *Proc. EUSIPCO 2012*, Bucharest, Romania, August 27 - 31, 2012, pp. 2094–2098.

[10] P. Tichavský, A. Yeredor, and Z. Koldovský, "On Computation of Approximate Joint Block-Diagonalization using Ordinary AJD", F. Theis et al. (Eds.): *LVA/ICA 2012*, LNCS 7191, pp. 163-171, 2012.

[11] L. Sorber, *Data Fusion, Tensor Factorizations by Complex Optimization*, PhD Dissertation, KU Leuven, Belgium, May 2014.

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

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

[14] D. Nion and L. De Lathauwer,"A Block Component Model based blind DS-CDMA receiver," *IEEE Transactions on Signal Processing*, vol. 56, no. 11, pp. 5567-5579, 2008.

[15] 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 Journal of Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324-1342, 2000.

[16] A.H. Phan, A. Cichocki, P. Tichavský, "On Fast Algorithms for Orthogonal Tucker Decomposition", *Proc. ICASSP 2014*, Florence, Italy, 4.-9.5. 2014, pp. 6816–6820.

[17] A.H. Phan, P. Tichavský, A. Cichocki, and Z.Koldovský "Low rank blind nonnegative matrix deconvolution", *Proc. ICASSP 2012*, Kyoto, Japan, pp. 1893–1896.

[18] E. Moulines, P. Duhamel, J. Cardoso, and S. Mayrargue, "Subspace methods for the blind identification of multichannel fir filters," *IEEE Transactions on Signal Processing*, vol. 43, no. 2, pp. 516-525, Feb 1995.

[19] Y. Chen, D. Han, and L. Qi,"New ALS methods with extrapolating search directions and optimal step size for complex–valued tensor decompositions," *IEEE Transactions on Signal Processing*, vol. 59, no. 12, pp. 5888-5898, 2011.

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