A fast algorithm for the CP decomposition of large tensors

R. Andre, Xavier Luciani, Eric Moreau

A fast algorithm for the CP decomposition of large tensors

Un algoritmo veloce per la decomposizione di grandi tensori

R. André, X. Luciani and E. Moreau

Abstract The canonical polyadic decomposition is one of the most used tensor decomposition. However classical decomposition algorithms such as alternating least squares suffer from convergence problems and thus the decomposition of large tensors can be very time consuming. Recently it has been shown that the decomposition can be rewritten as a joint eigenvalue decomposition problem. In this paper we propose a fast joint eigenvalue decomposition algorithm then we show how it can benefit the canonical polyadic decomposition of large tensors.

Abstract *La decomposizione canonica di tensori è usata in diversi campi tra cui quello dei data sciences. Tuttavia, in classici algoritmi di decomposizione, come l'alternating least squares, si possono riscontrare problemi di convergenza e proprio per questo motivo che la decomposizione di grandi tensori puo essere molto dispendiosa in termini di tempo di calcolo. Recentemente, sono stati sviluppati algoritmi di decomposizione canonica veloci, basati sulla diagonalizzazione di un insieme di matrici su una base comune di autovettori. In questo articolo proponiamo un algoritmo originale per risolvere quest'ultimo problema. In seguito mettiamo in evidenza l'aspetto più interessante di questo approccio al fine di effettuare la decomposizione canonica di grandi tensori.*

Key words: Tensor, Canonical polyadic decomposition, PARAFAC, algorithms, Joint eigenvalues decomposition

Rémi André, Xavier Luciani and Eric Moreau
Aix Marseille Université, Université de Toulon, CNRS, ENSAM, LSIS, Marseille, France, e-mail: luciani@univ-tln.fr

1 Introduction

In many data sciences applications, the collected data have a multidimensional structure and can thus be stored in multiway arrays (tensors). In this context, multiway analysis provides efficient tools to analyze such data sets. In particular, the Canonical Polyadic Decomposition (CPD) also known as PARAllel FACtor analysis (PARAFAC) has been successfully applied in various domains such as chemometrics, telecommunications, psychometrics and data mining, just to mention a few [7].

The CPD models the data thanks to multilinear combinations as described bellow. Let us consider a data tensor $\mathscr{T}$ of order Q (*i.e.* a Q-dimensions array) and size $I_1 \times \cdots \times I_Q$. Its CPD of rank N is then defined by:

$$\mathscr{T}_{i_1,\cdots,i_Q} = \sum_{n=1}^{N} F^{(1)}_{i_1,n} \cdots F^{(Q)}_{i_Q,n} + \mathscr{E}_{i_1,\cdots,i_Q} \tag{1}$$

where $\mathbf{F}^{(q)}$ is the q-th factor matrix of size $I_q \times N$ and $\mathscr{E}$ is a tensor modeling the noise or the model error. One crucial point here is that this decomposition has usually an unique solution up to trivial scaling and permutation indeterminacy. The idea is then that the meaningful information lies in the factor matrices. Thus we want to estimate these matrices from the data. Several algorithms were proposed in this purpose. The most popular is the Alternating Least Squares algorithm (ALS) [6]. This iterative algorithm is very simple to implement and usually provides accurate results. However it suffers from well known convergence problems. In particular the convergence is very sensitive to the initialization and the algorithm can be easily stuck in a local minimum of the cost function. A smart initialization is always possible but in practice one had usually better to perform several runs of the algorithm with random initialization. A second consequence is that it is difficult to set efficiently the threshold of the stopping criterion. Indeed it frequently occurs that the algorithm escapes from a local minimum after a very large number of iterations and during these iterations the variations of the cost function can be very small. Thereby the decomposition performed with ALS can result in a high effective computational cost and thus can be time consuming when dealing with high rank CPD of large tensors. Several other iterative algorithms were proposed to solve those convergence problems but in practice the computational cost of these solutions remains high. More details about ALS convergence problems and other iterative CPD algorithms can be found in [7], [3] and [1].

Recently several authors showed how to rewrite the CPD as a Joint EigenValues Decomponsition (JEVD) of a matrix set [5, 8, 10]. The JEVD consists in finding the eigenvector matrix $\mathbf{A}$ that jointly diagonalizes a given set of K non-defective matrices $\mathbf{M}^{(k)}$ in the following way:

$$\mathbf{M}^{(k)} = \mathbf{A}\mathbf{D}^{(k)}\mathbf{A}^{-1}, \quad \forall k = 1,\ldots,K. \tag{2}$$

This approach allows to reduce the computational cost of the CPD. Indeed JEVD algorithms converge in few iterations with an excellent convergence rate. Several JEVD algorithms have been proposed in the last decade [4,8,9]. In a recent paper we have introduced an algorithm called JDTE [2]. This algorithm offers a good trade-

off between speed and precision but its performances decrease with the matrix size. In the CPD context, it means that this algorithm is not suitable for high rank CPD. As a consequence, we propose in this paper an improved version of this algorithm.

The paper is organized as follow. In the next section we recall a simple and economic way to rewrite the CPD as a JEVD problem. Then in section 3 we describe the proposed JEVD algorithm. Finally, in section 4 we evaluate our approach for the decomposition of large tensors by means of numerical simulations.

In the following, the operator $\mathsf{Diag}\{\cdot\}$ represents the diagonal matrix built from the diagonal of the matrix argument, the operator $\mathsf{ZDiag}\{\cdot\}$ sets to zero the diagonal of the matrix argument and $\|.\|$ is the Frobenius norm of the argument matrix or tensor.

2 From CPD to JEVD

There are several ways to rewrite the CPD as a JEVD problem. Here we use the method described in [8] because the associated algorithm, called DIAG, has the lowest numerical complexity.

We consider the tensor $\mathscr{T}$ and its CPD of rank N defined in introduction. The first step consists in rearranging entries of $\mathscr{T}$ into an unfolding matrix $\mathbf{T}$ of size $\prod_{p=1}^{P} I_p \times \prod_{q=P+1}^{Q} I_q$ by merging the first P modes on the rows of $\mathbf{T}$ and the $Q-P$ other modes on its columns. Defining:

$$\mathbf{Y}^{(P,1)} = \mathbf{F}^{(P)} \odot \mathbf{F}^{(P-1)} \odot \cdots \odot \mathbf{F}^{(1)} \text{ and } \mathbf{Y}^{(Q,P+1)} = \mathbf{F}^{(Q)} \odot \mathbf{F}^{(Q-1)} \odot \cdots \odot \mathbf{F}^{(P+1)}, \tag{3}$$

where $\odot$ is the Khatri-Rao product, we can thus rewrite (1) in a matrix form:

$$\mathbf{T} = \mathbf{Y}^{(P,1)} \left(\mathbf{Y}^{(Q,P+1)}\right)^{\mathsf{T}}. \tag{4}$$

Of course, other merging of the tensor modes could have been chosen, leading to other unfolding matrices. The choice of the unfolding matrix can have a huge impact on the numerical complexity of the DIAG algorithm [8]. As a rule of thumb, when all tensor dimensions are large we recommend to chose $P = Q-2$ and to place the smaller dimension at the end ($I_Q \leq I_q, \forall q$).

The second step is the singular value decomposition (SVD) of $\mathbf{T}$, truncated at order N. We denote $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}^{\mathsf{T}}$ the matrices of this truncated SVD.

At this stage, their exists a non singular square matrix $\mathbf{A}$ of size $N \times N$ such that:

$$\mathbf{Y}^{(P,1)} = \mathbf{U}\mathbf{A} \text{ and } \left(\mathbf{Y}^{(Q,P+1)}\right)^{\mathsf{T}} = \mathbf{A}^{-1}\mathbf{S}\mathbf{V}^{\mathsf{T}}. \tag{5}$$

$\left(\mathbf{Y}^{(Q,P+1)}\right)^{\mathsf{T}}$ can be seen as an horizontal block matrix:

$$\left(\mathbf{Y}^{(Q,P+1)}\right)^{\mathsf{T}} = \left[\boldsymbol{\phi}^{(1)}\left(\mathbf{Y}^{(Q-1,P+1)}\right)^{\mathsf{T}}, \cdots, \boldsymbol{\phi}^{(I_Q)}\left(\mathbf{Y}^{(Q-1,P+1)}\right)^{\mathsf{T}}\right], \tag{6}$$

where $\boldsymbol{\phi}^{(1)}, \cdots, \boldsymbol{\phi}^{(I_Q)}$ are the I_Q diagonal matrices built from the I_Q rows of matrix $\mathbf{F}^{(Q)}$. Then, (5) and (6) yield:

$$\mathbf{S}\mathbf{V}^{\mathsf{H}} = \left[\Gamma^{(1)\mathsf{T}}, \cdots, \Gamma^{(I_Q)\mathsf{T}}\right], \tag{7}$$

where $\Gamma^{(k_1)} = \mathbf{Y}^{(Q-1,P+1)}\phi^{(k_1)}\mathbf{A}^{\intercal}$ for $k_1 = 1,\cdots,I_Q$. Assuming that matrices $\Gamma^{(k_1)}$ and matrix $\mathbf{Y}^{(Q-1,P+1)}$ are full column rank, then they all admit a Moore-Penrose matrix inverse denoted by $\sharp$. Thereby, we can define for any couple (k_1,k_2) with $k_1 = 1,\cdots,I_Q-1$ and $k_2 = k_1+1,\cdots,I_Q$:

$$\mathbf{M}^{(k_1,k_2)} \stackrel{\text{def}}{=} \left(\Gamma^{(k_1)\sharp}\Gamma^{(k_2)}\right)^{\intercal}, \tag{8}$$

$$= \mathbf{A}\mathbf{D}^{(k_1,k_2)}\mathbf{A}^{-1}, \tag{9}$$

where $\mathbf{D}^{(k_1,k_2)} = \phi^{(k_2)}\phi^{(k_1)\sharp}$ are diagonal matrices. As a result, $\mathbf{A}$ performs the JEVD of the set of matrices $\mathbf{M}^{(k_1,k_2)}$ and can be estimated using a JEVD algorithm. An important observation must be made here. In the previous step we have built $I_Q(I_Q-1)/2$ matrices $\mathbf{M}^{(k_1,k_2)}$. When dealing with large tensors this value can be very high with respect to the matrix size. In practice this does not help to improve the estimation of $\mathbf{A}$ significantly and dramatically increases the numerical complexity of the JEVD step. Thereby, we propose as an alternative to build only a subset of I_Q-1 matrices, for instance by taking $k_2 = k_1+1$ in (8). This can be seen as an economic version of the DIAG algorithm.

After the JEVD, matrices $\mathbf{Y}^{(P,1)}$ and $\mathbf{Y}^{(Q,P+1)}$ are immediately deduced from $\mathbf{A}$ using (5). Finally, we can easily deduce $\mathbf{F}^{(a)},\cdots,\mathbf{F}^{(b)}$ from $\mathbf{Y}^{(b,a)}$ as explained in [8].

In the next section, we propose an algorithm to solve the JEVD step. In order to simplified the notation, subscript k_1 and k_2 are replaced by unique subscript k so that equation (9) becomes:

$$\mathbf{M}^{(k)} = \mathbf{A}\mathbf{D}^{(k)}\mathbf{A}^{-1}, \quad \forall k = 1,\ldots,K. \tag{10}$$

where $K = I_Q(I_Q-1)/2$ or $K = I_Q-1$ depending on whether we choose the original DIAG algorithm or the economic version.

3 A fast JEVD algorithm

We propose here a fast algorithm to compute an estimate of $\mathbf{A}$, denoted $\mathbf{B}$, up to a permutation and scaling indeterminacy of the columns. This indeterminacy is inherent to the JEVD problem.

We want that $\mathbf{B}$ jointly diagonalizes the set of matrices $\mathbf{M}^{(k)}$. It means that matrices $\widehat{\mathbf{D}}^{(k)}$ defined by:

$$\widehat{\mathbf{D}}^{(k)} = \mathbf{B}^{-1}\mathbf{M}^{(k)}\mathbf{B}, \quad \forall k = 1,\ldots,K \tag{11}$$

must be as diagonal as possible. $\mathbf{B}$ is called the diagonalizing matrix. This kind of problem can be efficiently solved by an iterative procedure based on multiplicative updates. Before the first iteration, we set $\widehat{\mathbf{D}}^{(k)} = \mathbf{M}^{(k)}$, then at each iteration, matrices $\mathbf{B}$ and $\widehat{\mathbf{D}}^{(k)}$ are updated by a new matrix $\mathbf{X}$ as follow:

$$\mathbf{B} \leftarrow \mathbf{B}\mathbf{X} \text{ and } \widehat{\mathbf{D}}^{(k)} \leftarrow \mathbf{X}^{-1}\widehat{\mathbf{D}}^{(k)}\mathbf{X}, \quad \forall k = 1,\ldots,K. \tag{12}$$

The strategy that we now propose to compute the updating matrix $\mathbf{X}$ can be seen as a modified version of the one we proposed in [2]. The main difference is that here

we resorts to a sweeping procedure. It means that $\mathbf{X}$ is built from a set of $N(N-1)/2$ matrices, denoted $\mathbf{X}^{(i,j)}$ ($i=1,\ldots,N-1$ and $j=i+1,\ldots,N$) as follow:

$$\mathbf{X} = \prod_{i=1}^{N-1} \prod_{j=i+1}^{N} \mathbf{X}^{(i,j)}. \tag{13}$$

As a consequence, at each iteration, the updates in (12) consist now in $N(N-1)/2$ successive (i,j)-updates of $\mathbf{B}$ and $\widehat{\mathbf{D}}^{(k)}$, defined as:

$$\mathbf{B} \leftarrow \mathbf{B}\mathbf{X}^{(i,j)} \text{ and } \widehat{\mathbf{D}}^{(k)} \leftarrow \left(\mathbf{X}^{(i,j)}\right)^{-1} \widehat{\mathbf{D}}^{(k)} \mathbf{X}^{(i,j)}, \quad \forall k = 1,...,K. \tag{14}$$

Furthermore, because of the scaling indeterminacy of the JEVD problem we can impose the following structure to matrices $\mathbf{X}^{(i,j)}$: $\mathbf{X}^{(i,j)}$ is equal to the identity matrix at the exception of entries $X_{i,j}^{(i,j)}$ and $X_{j,i}^{(i,j)}$ that are equal to two unknown parameters:

$$X_{i,j}^{(i,j)} = x_1^{(i,j)}, \tag{15}$$

$$X_{j,i}^{(i,j)} = x_2^{(i,j)}. \tag{16}$$

We now explained how these parameters are computed for a given couple (i,j). First of all, let us define the function C as

$$C(\mathbf{X}^{(i,j)}) = \sum_{k=1}^{K} \|\mathsf{ZDiag}\{\left(\mathbf{X}^{(i,j)}\right)^{-1} \widehat{\mathbf{D}}_i^{(k))} \mathbf{X}^{(i,j)}\}\|^2. \tag{17}$$

C is a classical diagonalization criterion that is equal to zero if the K updated matrices are diagonal. Therefore we look for $\mathbf{X}^{(i,j)}$ that minimize C.

Matrix $\mathbf{X}^{(i,j)}$ can be decomposed as $\mathbf{X}^{(i,j)} = (\mathbf{I} + \mathbf{Z}^{(i,j)})$, where $\mathbf{Z}^{(i,j)} = \mathsf{ZDiag}\{\mathbf{X}^{(i,j)}\}$. The criterion can then be written as:

$$C(\mathbf{X}^{(i,j)}) = \tilde{C}(\mathbf{Z}^{(i,j)}) = \sum_{k=1}^{K} \|\mathsf{ZDiag}\{(\mathbf{I}+\mathbf{Z}^{(i,j)})^{-1} \widehat{\mathbf{D}}^{(k)} (\mathbf{I}+\mathbf{Z}^{(i,j)})\}\|^2. \tag{18}$$

We consider in fact an approximation of $C(\mathbf{X}^{(i,j)})$ assuming that we are close to the diagonalizing solution *i.e.* $\mathbf{X}^{(i,j)}$ is close to the identity matrix. This implies that $\|\mathbf{Z}^{(i,j)}\| \ll 1$ and thus the first order Taylor expansion of $(\mathbf{I}+\mathbf{Z}^{(i,j)})^{-1}$ yields:

$$(\mathbf{I}+\mathbf{Z}^{(i,j)})^{-1} \widehat{\mathbf{D}}^{(k)} (\mathbf{I}+\mathbf{Z}^{(i,j)}) \simeq (\mathbf{I}-\mathbf{Z}^{(i,j)}) \widehat{\mathbf{D}}^{(k)} (\mathbf{I}+\mathbf{Z}^{(i,j)}) \tag{19}$$

$$\simeq \widehat{\mathbf{D}}^{(k)} - \mathbf{Z}^{(i,j)} \widehat{\mathbf{D}}^{(k)} + \widehat{\mathbf{D}}^{(k)} \mathbf{Z}^{(i,j)} - \mathbf{Z}^{(i,j)} \widehat{\mathbf{D}}^{(k)} \mathbf{Z}^{(i,j)} \tag{20}$$

$$\simeq \widehat{\mathbf{D}}^{(k)} - \mathbf{Z}^{(i,j)} \widehat{\mathbf{D}}^{(k)} + \widehat{\mathbf{D}}^{(k)} \mathbf{Z}^{(i,j)}. \tag{21}$$

In the same way, matrices $\widehat{\mathbf{D}}^{(k)}$ can be decomposed as $\widehat{\mathbf{D}}^{(k)} = \Lambda^{(k)} + \mathbf{O}^{(k)}$, where $\Lambda^{(k)} = \mathsf{Diag}\{\widehat{\mathbf{D}}^{(k)}\}$ and $\mathbf{O}^{(k)} = \mathsf{ZDiag}\{\widehat{\mathbf{D}}^{(k)}\}$. Here our assumption means that matrices $\widehat{\mathbf{D}}^{(k)}$ are almost diagonal and thus $\|\mathbf{O}^{(k)}\| \ll 1$. It yields:

$$\widehat{\mathbf{D}}^{(k)} - \mathbf{Z}^{(i,j)} \widehat{\mathbf{D}}^{(k)} + \widehat{\mathbf{D}}^{(k)} \mathbf{Z}^{(i,j)} \simeq \Lambda^{(k)} + \mathbf{O}^{(k)} - \mathbf{Z}\Lambda^{(k)} + \Lambda^{(k)} \mathbf{Z} \tag{22}$$

and finally we can approximate $\tilde{C}(\mathbf{Z}^{(i,j)})$ by $C_a(\mathbf{Z}^{(i,j)})$:

$$C_a(\mathbf{Z}^{(i,j)}) = \sum_{k=1}^{K} \|\mathsf{ZDiag}\{\mathbf{O}^{(k)} - \mathbf{Z}^{(i,j)}\Lambda^{(k)} + \Lambda^{(k)}\mathbf{Z}^{(i,j)}\}\|^2 \tag{23}$$

$$= \sum_{k=1}^{K} \sum_{\substack{m,n=1 \\ m\neq n}}^{N} (O_{mn}^{(k)} + Z_{mn}\Lambda_{mm}^{(k)} - Z_{mn}\Lambda_{nn}^{(k)})^2 \tag{24}$$

$$= \sum_{k=1}^{K} \Big(\big(O_{ij}^{(k)} - x_1^{(i,j)}\big(\Lambda_{jj}^{(k)} - \Lambda_{ii}^{(k)}\big)\big)^2 + \big(O_{ji}^{(k)} - x_2^{(i,j)}\big(\Lambda_{ii}^{(k)} - \Lambda_{jj}^{(k)}\big)\big)^2 + \sum_{\substack{m,n=1 \\ m\neq n}}^{N} (O_{mn}^{(k)})^2 \Big). \tag{25}$$

We can then easily show that C_a is minimum for :

$$(x_1^{(i,j)}, x_2^{(i,j)}) = \left(\frac{\sum_{k=1}^{K} O_{ij}^{(k)}(\Lambda_{jj}^{(k)} - \Lambda_{ii}^{(k)})}{\sum_{k=1}^{K}(\Lambda_{ii}^{(k)} - \Lambda_{jj}^{(k)})^2}, \frac{\sum_{k=1}^{K} O_{ji}^{(k)}(\Lambda_{ii}^{(k)} - \Lambda_{jj}^{(k)})}{\sum_{k=1}^{K}(\Lambda_{jj}^{(k)} - \Lambda_{ii}^{(k)})^2} \right). \tag{26}$$

We call this algorithm SJDTE for Sweeping Joint eigenvalue Decomposition based on a Taylor Expansion.

4 Numerical Simulations

We have included SJDTE in the DIAG procedure described in section 2 for the JEVD step. Two versions of this CPD algorithm were implemented corresponding to original and economic versions of DIAG. In the following, these are referred as DIAG-SJDTE and DIAG-SJDTE-eco respectively and are compared with the ALS for the CPD of large tensors. Three comparison criteria are used: the reconstruction error, r, defined by: $r = \frac{\|\mathscr{T} - \widehat{\mathscr{T}}\|^2}{\|\mathscr{T}\|^2}$, the number of computed iteration, n_{it}, and the cputime of matlab (elapsed time during the algorithm run), t_{cpu}. Of course the cputime strongly depends on the implementation of the algorithms and for this reason the computational cost might be preferred. However in the present case, numerical complexities of compared algorithms involve subroutines such as truncated SVDs whose numerical complexity is hard to evaluate with precision. Furthermore, all the algorithms compared in this section were carefully implemented in house in Matlab language and optimized for it.

As explained in the introduction a classical problem with ALS is to choose an appropriate threshold for the stopping criterion. As a consequence, we implemented two versions. In the first version (ALS-1), the ALS procedure is stopped when the relative difference between two successive values of the reconstruction error term is lower than 10^{-3} or when the number of iterations reach 50. In the second version (ALS-2), we set these two values to 10^{-8} and 200. SJDTE is stopped when the relative difference between two successive values of C is lower than 10^{-3} or when the number of iterations reach 10. Comparisons are made according to three scenarios by means of Monte-Carlo (MC) simulations. For each MC run three new matrix factors of size $200 \times N$ are randomly drawn from a normal distribution and a new

Table 1 Average values of the reconstruction error, number of iterations and cputime.

Algorithm	Scenario 1 : $N=5$			Scenario 2 : $N=15$			Scenario 3 : $N=30$		
	r	n_{it}	t_{cpu}	r	n_{it}	$t_{cpu}(s)$	r	n_{it}	t_{cpu}
ALS-1	0.0786	7.16	4.87	0.0565	10.44	10.9	0.0428	13.39	28.55
ALS-2	0.0655	9.97	6.47	0.0470	16.20	16.59	0.0358	21.1	44.28
DIAG-SJDTE	0.0001	4.26	8.96	0.0001	4.76	56.77	0.0003	5.12	407.53
DIAG-SJDTE-eco	0.0004	4.3	6.78	0.0002	4.87	12.55	0.0003	5.24	24.57

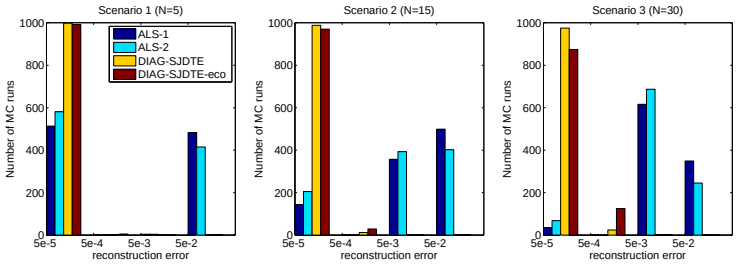

Fig. 1 Distributions of the reconstruction error for the four algorithms according to the value of N.

tensor is built from the CPD model. A white Gaussian noise is then added to its entries in order to obtain a signal to noise ratio of 40 dB. Then the five algorithms are run to compute the CPD of rank N of the noisy tensor. We distinguish three scenarios according to the chosen value of N: $N=5$ (scenario 1), $N=15$ (scenario 2) and $N=30$ (scenario 3). Average values of r, n_{it} and t_{cpu} computed from 1000 MC runs are reported in table 1 for each algorithm and scenario. In order to have a more precise idea of the convergence rate of the algorithms, we show in figure 1 the distribution of r in the ranges $[5.10^{-5};5.10^{-4}[$, $[5.10^{-4};5.10^{-3}[$, $[5.10^{-3};5.10^{-2}[$ and $[5.10^{-2};+\infty[$.

Convergence problems of the ALS clearly appear from these results. Whatever the considered scenario, the average value of r remains high for both ALS-1 and ALS-2. Figure 1 shows that for $N=5$ less than 60% of the values of r fall in the range $[5.10^{-5};5.10^{-4}[$. Moreover, this percentage dramatically decreases for increasing values of N besides a significant rate of the values of r fall in the range $[5.10^{-2};+\infty[$. In these conditions, convergence times of ALS-1 are very low and compete with those of DIAG but considering the convergence rates of ALS-1 these cputime values are misleading. Indeed, we could say that in average ALS-1 quickly converge to a local minimum and in practice it should be run from different starting point in order to obtain satisfying convergence hence increasing the total cputime. Furthermore, comparing ALS-1 and ALS-2 results, it seems that decreasing the threshold of the stopping criterion has little impact on the convergence. Conversely, DIAG-SJDTE and DIAG-SJDTE-eco offer very good results in term of reconstruction errors : from 90% to 100% of he values of r fall in the range $[5.10^{-5};5.10^{-4}[$. Moreover, these performances are very stable with respect to the value of N, with a little advantage for DIAG-SJDTE. Now, regarding the average cputime, DIAG-SJDTE-eco is very

less time consuming than DIAG-SJDTE for $N = 15$ ($13s$ against $57s$) and $N = 30$ ($25s$ against $408s$) whereas the average iteration numbers of both algorithms is the same. Considering the small difference between both algorithms regarding the reconstruction error, we can clearly recommend the use of DIAG-SJDTE-eco when N is large.

5 Conclusion

We have proposed an original JEVD algorithm and showed how it can help for computing the canonical polyadic decomposition of large tensors. Preliminary results showed in this work point out that this approach provides very good convergence rates comparing to a reference CPD algorithm. Moreover it converges in very few iterations and the computing times are very low, including for high rank CPD. Further studies will be conducted to refine this conclusion. In particular, we want now to evaluate the impact of the choice of the subset of matrices $\mathbf{M}^{(k)}$ and of the JEVD algorithm inside the DIAG procedure.

References

1. Acar, E., Dunlavy, D.M., Kolda, T.G.: A scalable optimization approach for fitting canonical tensor decompositions. Journal of Chemometrics **25**(2), 67–86 (2011)
2. Andre, R., Trainini, T., Luciani, X., Moreau, E.: A fast algorithm for joint eigenvalue decomposition of real matrices. In: European Signal Processing Conference (EUSIPCO 2015)
3. Comon, P., Luciani, X., de Almeida, A.L.F.: Tensor decompositions, alternating least squares and other thales. Journal of Chemometrics **23** (2009)
4. Fu, T., Gao, X.: Simultaneous diagonalization with similarity transformation for non-defective matrices. In: IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP 2006), vol. 4, pp. 1137–1140
5. Hajipour, S., Albera, L., Shamsollahi, M., Merlet, I.: Canonical polyadic decomposition of complex-valued multi-way arrays based on simultaneous schur decomposition. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013), pp. 4178–4182
6. Harshman, R.A.: Foundation of PARAFAC procedure: Models and conditions for an 'explanatory' multi-mode factor analysis. UCLA working papers in Phonetics (16), 1–84 (1970)
7. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
8. Luciani, X., Albera, L.: Canonical polyadic decomposition based on joint eigenvalue decomposition. Chemometrics and Intelligent Laboratory Systems **132**(0), 152 – 167 (2014)
9. Luciani, X., Albera, L.: Joint eigenvalue decomposition of non-defective matrices based on the lu factorization with application to ica. IEEE Transactions on Signal Processing **63**(17), 4594–4608 (2015)
10. Roemer, F., Haardt, M.: A closed-form solution for multilinear parafac decompositions. In: Fifth IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2008), pp. 487–491