Parallelization Scheme for Canonical Polyadic Decomposition of Large-Scale High-Order Tensors

Abdelhak Boudehane, Laurent Albera, Arthur Tenenhaus, Laurent Le Brusquet, Remy Boyer

Parallelization Scheme for Canonical Polyadic Decomposition of Large-Scale High-Order Tensors

Abdelhak Boudehane[a], Laurent Albera[b], Arthur Tenenhaus[a], Laurent Le Brusquet[a], Rémy Boyer[c]

[a] *Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France.*
[b] *Université Rennes 1, Inserm, LTSI - UMR 1099, F-35000 Rennes, France.*
[c] *Centre de Recherche en Informatique, Signal et Automatique de Lille, Univ. Lille 1, Villeneuve-d'Ascq, France.*

Abstract

Modeling multidimensional data using tensor models, in particular through the Canonical Polyadic (CP) model, can be found in large numbers of timely and important signal based applications. However, the computational complexity in the case of high-order and large-scale tensors remains a challenge that prevents the implementation of the CP model in practice. While some algorithms, in the literature, deal with large-scale problems, others target high-order tensors. Nevertheless, these algorithms encounter major issues when both problems are present. In this paper, we propose a parallelizable strategy, based on the tensor network theory, to deal simultaneously with both high-order and large-scale problems. We show the usefulness of the proposed strategy in reducing the computational time on a realistic electroencephalography data set.

Keywords: High-order, large-scales, tensor decomposition, EEG, parallel computing.

Email addresses: abdelhak.boudehane@gmail.com (Abdelhak Boudehane), laurent.albera@univ-rennes1.fr (Laurent Albera), arthur.tenenhaus@centralesupelec.fr (Arthur Tenenhaus), Laurent.Lebrusquet@centralesupelec.fr (Laurent Le Brusquet), remy.boyer@univ-lille.fr (Rémy Boyer)

1. Introduction

Data analysis using tensor models [1] is taking wider place in various applications such as signal processing [2] and neuroscience [3]. Tensors preserve the multidimensional structure of the data and provide separate factor analysis [4]. These properties facilitate the interpretation of the decomposition. One of the most attractive features of the Canonical Polyadic (CP) model is its uniqueness, *i.e.*, up to mild conditions, the factors of the tensor are unique [5]. To some extent, this property limits spurious physical interpretation. Tensor approaches have been exploited for modeling ElectroEncephaloGraphy (EEG) signals in the context of extended epileptic source localization [6]. In this context, tensor modeling offers better separation of sources when compared to conventional methods [6, 7]. However, computational complexity requirements of tensor decomposition algorithms are prohibitive for high-order/large-scale tensors. This appears clearly when dealing with high-order tensors (> 3) where the number of entries increase exponentially with respect to the order [8]. Furthermore, the computational complexity of the factors estimation algorithm increases when the dimensions of the tensor are large and appears to be an end-point for analyzing large-scale tensors [8, 9]. This is the case for EEG signals analysis that are observed for a large number of electrodes, frequency bins, temporal samples and repetitions. Whereas, the relevant information is usually concentrated in few factor matrices, such as in EEG source localization where the aim is to estimate the spatial information represented by one factor matrix (first mode) [10]. Nevertheless, the estimation of the corresponding factor using classical tensor algorithms, such as Alternating Least Squares (ALS) algorithm [11] and gradient-based approaches [12], requires full tensor decomposition, *i.e.*, the estimation of all the factors. This makes the expensive process of full tensor decomposition inevitable. Tensor-Train [13] (TT) based scheme called Joint dImensionality Reduction And Factors rEtrieval (JIRAFE) has been proposed [14, 15], to deal with high-order tensors [15]. JIRAFE helps reducing the computational complexity by breaking the high-order tensor into a

set of coupled 3-rd order tensors, called the TT-cores. A sequential estimation scheme has been proposed for factor retrieval. The dominant complexity of this scheme relies on the TT-SVD algorithm [13], which is the step used to break the high-order CP tensor into TT-cores. This step becomes intractable when dealing with large-scale tensors since it consists of applying a set of SVDs on full unfolded tensor. In addition, the dimensions of the TT-cores remain important for large-scale tensors. Grid-PARAFAC [16] is a parallelizable algorithm that targets large-scale tensors. Yet, Grid-PARAFAC consists of ALS and iterative steps whose complexity appears intractable when the order increases. In this paper, we propose a TT-based scheme, targeting high-order and large-scale problems simultaneously. First, we reduce the computational time of the dimensionality reduction step by using parallel TT-SVD structure through a split-and-merge strategy [17]. Roughly speaking, the computational time is divided by the number of sub-matrices considered in the split-and-merge algorithm. Moreover, we reduce the computational complexity of each of the parallel steps using the randomized SVD (RSVD) [18]. Once the dimensionality reduction is done, the TT-model allows the estimation of each factor separately by decomposing the corresponding TT-core, without the need to estimate the other factors. Furthermore, we propose a gridding-based scheme to break the large-scale TT-cores into smaller sub-tensors sharing common change-of-basis matrices. Thus, we exploit this coupling to reduce the time elapsed to estimate the factor from the TT-core. The usefulness of the proposed approach is demonstrated on a realistic EEG dataset described in [6]. In this application, the objective is to localize the epileptic sources with sufficient accuracy and reduced computational time. We compare our approach to an ALS-based approach used in [6] applied to 3-rd and 4-th order tensors and to the native JIRAFE. *Notations:* Vectors, matrices and tensors are represented by $\boldsymbol{x}$, $\boldsymbol{X}$ and $\boldsymbol{\mathcal{X}}$, respectively. The symbols $(\cdot)^T$, $(\cdot)^{-T}$ and $(\cdot)^\dagger$ denote, respectively, the transpose, the inverse of transpose and the Moore–Penrose inverse matrix. The Frobenius norm is defined by $||\cdot||_F$. The mode-q product, denoted by $\times_q$, is defined between tensor $\boldsymbol{\mathcal{X}}$ of size $N_1 \times \cdots \times N_Q$

and matrix $\boldsymbol{A}$ of size $M \times N_q$ as $[\boldsymbol{\mathcal{X}} \times_q \boldsymbol{A}] = \sum_{n_q=1}^{N_q} [\boldsymbol{\mathcal{X}}]_{n_1 n_2 \cdots n_q \cdots n_Q} [\boldsymbol{A}]_{m n_q}$. The mode-$\{p, q\}$ product, denoted by $\times_q^p$, is defined between tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ as $[\boldsymbol{\mathcal{A}} \times_q^p \boldsymbol{\mathcal{B}}] = \sum_{k=1}^{N_q} [\boldsymbol{\mathcal{A}}]_{n_1,\ldots,n_{q-1},k,n_{q+1},\ldots,n_Q} [\boldsymbol{\mathcal{B}}]_{m_1,\ldots,m_{p-1},k,m_{p+1},\ldots,m_P}$. The operator $diag(\boldsymbol{V})$ converts vector $\boldsymbol{V}$ in to a diagonal matrix. The Khatri-Rao product is denoted by $\odot$. The tensor $\boldsymbol{\mathcal{I}}_{Q,R}$, is a Q-th order tensor (hypercube) with ones on the diagonal and zero otherwise, denoting the identity tensor of size $R \times R \times \ldots \times R$. The matrix $\boldsymbol{X}^{(k)}$ of size $N_k \times N_1 \cdots N_{k-1} N_{k+1} \cdots N_Q$ refers to the k-mode unfolding of $\boldsymbol{\mathcal{X}}$ of size $N_1 \times \cdots \times N_Q$.

2. High-order large-scale tensor decomposition

2.1. Canonical Polyadic Decomposition

Assuming a high-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{N_1 \times N_2 \times \ldots \times N_Q}$ of order Q (> 3) and canonical rank R. Tensor $\boldsymbol{\mathcal{X}}$ follows a CP decomposition [19], *i.e.*, it can be written as the n-mode product of Q factor matrices as $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{I}}_{Q,R} \times_1 \boldsymbol{P}_1 \times_2 \ldots \times_Q \boldsymbol{P}_Q$, where $\boldsymbol{P}_q \in \mathbb{R}^{N_q \times R}$, for $1 \leq q \leq Q$, are the factor matrices. The widely used algorithms, to estimate these factor matrices, are usually based on either ALS or gradient based approaches [11]. However, these algorithms face problems since the complexity increases with the order and with the dimensions. The problem of high-order tensors can be addressed using the JIRAFE algorithm explained in the next section.

2.2. JIRAFE scheme

From the equivalence between the CP and the TT models, the high-order tensor $\boldsymbol{\mathcal{X}}$ can be written as a set of 3-rd order tensors called TT-cores, *i.e.*, $\boldsymbol{\mathcal{X}} = \boldsymbol{G}_1 \times_2^1 \boldsymbol{\mathcal{G}}_2 \times_3^1 \ldots \times_{Q-1}^1 \boldsymbol{\mathcal{G}}_{Q-1} \times_Q^1 \boldsymbol{G}_Q$ (TT-core factorization), with $\boldsymbol{\mathcal{G}}_q = \boldsymbol{\mathcal{I}}_{3,R} \times_1 \boldsymbol{M}_{q-1} \times_2 \boldsymbol{P}_q \times_3 \boldsymbol{M}_q^{-T}$ (3-rd order CP), $\boldsymbol{G}_1 = \boldsymbol{P}_1 \boldsymbol{M}_1^{-1}$, $\boldsymbol{G}_Q = \boldsymbol{M}_{Q-1} \boldsymbol{P}_Q^T$, where $\boldsymbol{\mathcal{G}}_q$, for $1 < q < Q$, is the q-th TT-core tensor of size $R \times N_q \times R$, and the TT-ranks are $(R, R, \ldots, R)$. The matrices $\boldsymbol{M}_q$, of size $R \times R$, are the change-of-basis matrices for $1 \leq q \leq Q - 1$. The important result is that a rank-R Q-th order CP model is equivalent to a structured TT-model where the TT-cores follow a

rank-R 3-rd order CP model. The TT-cores are obtained using the TT-SVD algorithm, which consists of applying SVD on the tensor unfoldings. Once the TT-cores estimated, one TT-core is used in Tri-ALS process, *i.e.*, 3-rd order ALS, in order to estimate the corresponding factor. Taking advantage of the coupling between the TT-cores, the rest of factors are estimated through using Bi-ALS processes, *i.e.*, 3-rd order ALS with a pre-estimated matrix.

2.3. Limitations

The equivalence between TT and the CP models helps mitigating the impact of the order Q by using the JIRAFE method where the factor estimation problem is solved as coupled least squares problems. However, sometimes the dimensions of the tensor $(N_1, \ldots, N_Q)$ are large. This issue affects the JIRAFE scheme: the TT-cores are obtained by TT-SVD that consists of applying SVDs on the sequential tensor reshapings. The computational cost of this step depends on the size of these reshapings and becomes prohibitive for large-scale tensor. Furthermore, in the second step, Tri-ALS/Bi-ALS algorithms are used on the TT-cores of size $R \times N_q \times R$. Knowing that the complexity of the Tri-ALS, given by $O(3R^3N_q)$ per iteration, is dimension-dependent, the computational time of this second step increases in a quadratic way with N_q, which is large for large-scale tensors. In the following, we explain the solutions that we propose in order to deal with these limitations.

3. Split and merge strategy

3.1. TT-SVD algorithm

The TT-SVD algorithm, as illustrated in Fig. 1, estimates the TT-cores by extracting the dominant singular subspace using SVDs. At the first step, an SVD is applied on the first unfolding $\boldsymbol{X}^{(1)}$ $(N_2N_3 \ldots N_Q \times N_1)$ of the tensor $\boldsymbol{\mathcal{X}}$ in order to extract the first dominant subspace, such that $\boldsymbol{X}^{(1)} = \boldsymbol{U}^{(1)}\boldsymbol{V}^{(1)}$, where $\boldsymbol{U}^{(1)}$ is the left singular matrix and $\boldsymbol{V}^{(1)}$ is the singular values diagonal matrix multiplied by the right singular matrix. The first TT-core is given by the

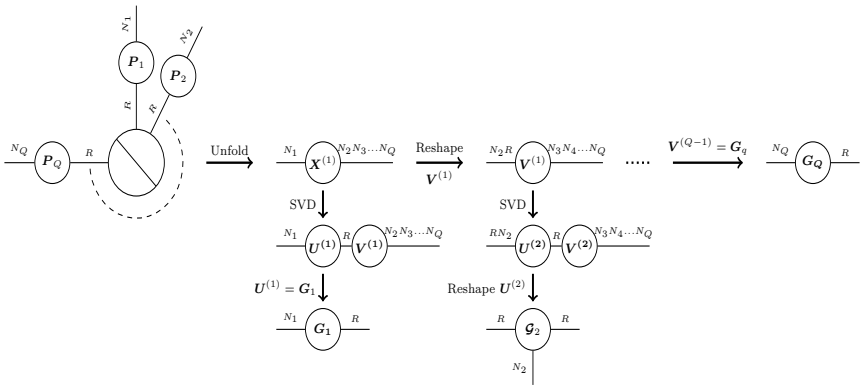


Figure 1: Factor graph modelization of the TT-SVD algorithm

left singular matrix, *i.e.*, $\boldsymbol{G}_1 = \boldsymbol{U}^{(1)}$. Then, the second TT-core $\boldsymbol{\mathcal{G}}_2$ is obtained by applying the SVD on the remaining matrix $\boldsymbol{V}^{(1)}$ of size $(N_3 N_4 \dots N_Q \times RN_2)$ to extract the new left singular matrix $\boldsymbol{U}^{(2)}$ reshaped into a $R \times N_2 \times R$ tensor $\boldsymbol{\mathcal{G}}_2$. In the same way, we continue the process until all the TT-cores are computed. Since the TT-SVD algorithm is composed of SVD steps, whose complexity is given in section 6.1, the complexity of the TT-SVD depends on the tensor dimensions N_q. This means that the complexity of the TT-SVD increases significantly in the case of large-scale tensors. In the following, we propose a solution to reduce the computational time using "split and merge" [20, 21].

3.2. Split and Merge Scheme for the TT-SVD

The split and merge strategy allows the parallelization of SVD steps in order to reduce the computational time [22]. We describe how we implement this strategy in order to parallelize the SVDs in Algo. 1. In the "split" step, we divide the tensor's first unfolding $\boldsymbol{X}^{(1)}$ $(N_2 N_3 \dots N_Q \times N_1)$ into a set of smaller matrices $\boldsymbol{X}_J^{(1)T}$ $(\frac{N_2 N_3 \dots N_Q}{J} \times N_1)$ following the larger dimension, as shown in Fig. 2, *i.e.*, $\boldsymbol{X}^{(1)} = [\boldsymbol{X}_1^{(1)T} \boldsymbol{X}_2^{(1)T} \dots \boldsymbol{X}_J^{(1)T}]^T$, where J is the number of sub-matrices that we chose depending on the number of calculators that we can launch in parallel. Then, we apply the SVD on each of the sub-matrices in a

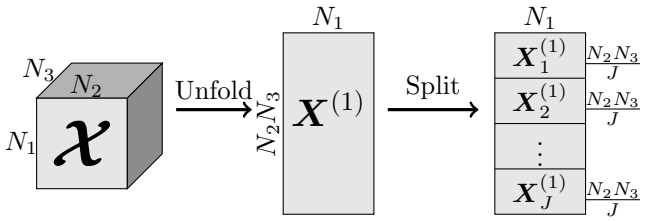


Figure 2: Splitting the first unfolding of a 3-rd order tensor into J sub-matrices

parallel way, *i.e.*, $\boldsymbol{X}_j^{(1)} = \tilde{\boldsymbol{U}}_j \boldsymbol{S}_j \boldsymbol{V}_j^T$ for $1 \leq j \leq J$, where the size of $\tilde{\boldsymbol{U}}_j$, $\boldsymbol{S}_j$ and $\boldsymbol{V}_j$ are $(\frac{N_2 N_3 \dots N_Q}{J} \times R)$, $(R \times R)$ and $(N_1 \times R)$, respectively. In step 3, the matrices $\boldsymbol{S}_j \boldsymbol{V}_j^T$ are concatenated to form the matrix $\boldsymbol{Y}$ of size $(JR \times N_1)$, while the matrix $\tilde{\boldsymbol{U}}$ is block diagonal of size $(N_2 N_3 \dots N_Q \times JR)$. In step 4, we apply SVD on the matrix $\boldsymbol{Y}$, *i.e.*, $\boldsymbol{Y} = \boldsymbol{U}_y \boldsymbol{S}_y \boldsymbol{V}_y^T$. Finally, the SVD decomposition of the tensor's unfolding is given by $\boldsymbol{U}_x = \tilde{\boldsymbol{U}} \boldsymbol{U}_y$, $\boldsymbol{S}_x = \boldsymbol{S}_y$, $\boldsymbol{V}_x = \boldsymbol{V}_y$. Once done, we reshape the product $\boldsymbol{S}_x \boldsymbol{V}_x$, and use it in the same scheme described in Algo. 1, in order to estimate the next TT-core, *i.e.*, and we repeat this process for all SVD steps described in Fig. 1 until all the TT-cores are computed. Notice that this algorithm is parallelized, *i.e.*, the SVDs (step 2) in Algo 1 are done in parallel in order to reduce the computational time. Hence, instead of being interested in the sum of computational time of all the SVDs applied on the tensor unfolding, we are rather interested in the maximal computation time of the sub-matrices obtained by splitting the tensor's unfolding. Roughly speaking, this divides the computation time by the number of sub-matrices. Note that, in Algo 1, we assume that the rank is known. However, in practical situations, the rank can be estimated using physical constraints [23, 24].

4. Gridding strategy on the TT-cores

Once the TT-core $\boldsymbol{\mathcal{G}}_q$, of dimensions $R \times N_q \times R$, is computed, a Tri-ALS is used in order to estimate the factor matrix $\boldsymbol{P}_q$. Nevertheless, when the tensor dimensions N_q become large, the computation time of the ALS increases significantly since it has a dimension-size dependent complexity. Moreover, this

Algorithm 1 Split and merge for SVD

Input: Tensor's unfolding $\boldsymbol{X}^{(1)}$ and rank R

Output: The matrices $\boldsymbol{U}_x$, $\boldsymbol{S}_x$ and $\boldsymbol{V}_x$

1: Splitting the tensor's unfolding: $\boldsymbol{X}^{(1)} = [\boldsymbol{X}_1^{(1)T} \boldsymbol{X}_2^{(1)T} \dots \boldsymbol{X}_J^{(1)T}]^T$

2: **parfor** $j \in \{1, \dots, J\}$

Apply SVD on each $\boldsymbol{X}_j^{(1)}$ in parallel: $\boldsymbol{X}_j^{(1)} = \tilde{\boldsymbol{U}}_j \boldsymbol{S}_j \boldsymbol{V}_j^T$

end parfor

3: Forming the matrices: $\boldsymbol{U} = diag(\tilde{\boldsymbol{U}}_1, \cdots, \tilde{\boldsymbol{U}}_J)$ and $\boldsymbol{Y} = \left[\boldsymbol{S}_1 \boldsymbol{V}_1^T \quad \cdots \quad \boldsymbol{S}_J \boldsymbol{V}_J^T\right]^T$

4: Applying SVD on $\boldsymbol{Y}$: $\boldsymbol{Y} = \boldsymbol{U}_y \boldsymbol{S}_y \boldsymbol{V}_y^T$

5: The SVD of $\boldsymbol{X}^{(1)}$ is given by: $\boldsymbol{U}_x = \tilde{\boldsymbol{U}} \boldsymbol{U}_y$, $\boldsymbol{S}_x = \boldsymbol{S}_y$, $\boldsymbol{V}_x = \boldsymbol{V}_y$.

high complexity is multiplied by the number of iterations. In order to reduce the computation time, we use a gridding scheme [16], that we adapt to our context. In fact, we divide the TT-core $\boldsymbol{\mathcal{G}}_q$, following the larger dimension N_q into smaller sub-tensors $\boldsymbol{\mathcal{G}}_{q_l}$, as shown in Fig. 3. This means that the factor

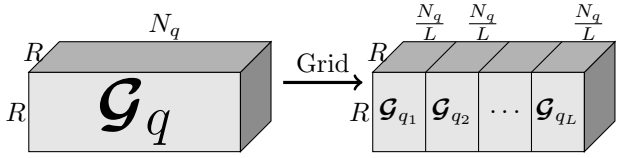


Figure 3: Dividing a TT-core into L sub-tensors

$\boldsymbol{P}_q$ is split as well into L sub-matrices as $\boldsymbol{P}_q = \left[\boldsymbol{P}_{q_1}^T \boldsymbol{P}_{q_2}^T \dots \boldsymbol{P}_{q_L}^T\right]^T$. The second unfolding of $\boldsymbol{\mathcal{G}}_q$ is written in function of the sub-tensors $\boldsymbol{\mathcal{G}}_{q_l}$ as

$$\boldsymbol{G}_q^{(2)} = \left[\boldsymbol{G}_{q_1}^{(2)T} \boldsymbol{G}_{q_2}^{(2)T} \dots \boldsymbol{G}_{q_L}^{(2)T}\right]^T = \left[\boldsymbol{P}_{q_1}^T \boldsymbol{P}_{q_2}^T \dots \boldsymbol{P}_{q_L}^T\right]^T [(\boldsymbol{M}_q^{-T} \odot \boldsymbol{M}_{q-1})^T]^\dagger \quad (1)$$

. From eq. (1), each of the sub-tensors $\boldsymbol{\mathcal{G}}_{q_l}$ can be factorized separately to obtain the sub-factor $\boldsymbol{P}_{q_l}$, while the matrices $\boldsymbol{M}_{q-1}$ and $\boldsymbol{M}_q$ are common for all the sub-tensors (see Fig. 4). In other words, we obtain a set of sub-tensors $\boldsymbol{\mathcal{G}}_{q_l}$ coupled on the first and the third modes [16, eqs. (17) and (18)]. Hence, each sub-tensor $\boldsymbol{\mathcal{G}}_{q_l}$ is written as a function of the matrices $\boldsymbol{M}_{q-1}$ and $\boldsymbol{M}_q$ and the factor l-th sub-matrix $\boldsymbol{P}_{q_l}$ as $\boldsymbol{\mathcal{G}}_{q_l} = [\![\boldsymbol{M}_{q-1}, \boldsymbol{P}_{q_l}, \boldsymbol{M}_q^{-T}]\!]$. Therefore, $\boldsymbol{M}_{q-1}$ and $\boldsymbol{M}_q$ can be estimated by applying Tri-ALS on one of the sub-tensors $\boldsymbol{\mathcal{G}}_{q_l'}$, instead of using the whole TT-core. Note that the choice of $\boldsymbol{\mathcal{G}}_{q_l'}$ depends on

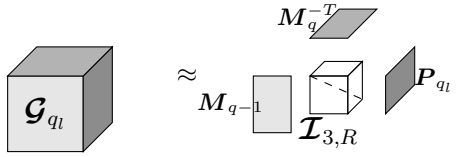


Figure 4: l-th sub-tensor of the q-th TT-core

the application, since in some contexts, such as sparse or incomplete data, the choice of the right sub-tensor leads to better factor estimation. Once the change of basis matrices estimated using a $\boldsymbol{\mathcal{G}}_{q'_l}$, we use them, in closed-form relations, to compute the remaining factor sub-matrices $\boldsymbol{P}_{q_l}$, for $l \neq l'$ as follow

$$\boldsymbol{P}_{q_l} = \boldsymbol{G}_{q_l}^{(2)}[(\boldsymbol{M}_q^{-T} \odot \boldsymbol{M}_{q-1})^T]^\dagger. \tag{2}$$

This strategy allows us to reduce the computation time by using only a small part of the TT-core in order to estimate $\boldsymbol{M}_{q-1}$ and $\boldsymbol{M}_q$ by means of an ALS algorithm, while the rest of the TT-core is used in a closed-form expressions in order to estimate the rest of the factor sub-matrices. We recall the expressions of the first and the last factor matrices in function of the first change of basis matrices and the TT-cores

$$\boldsymbol{P}_1 = \boldsymbol{G}_1\boldsymbol{M}_1,\ \boldsymbol{P}_Q = \boldsymbol{G}_Q^T\boldsymbol{M}_{Q-1}^{-T}. \tag{3}$$

We summarize the steps of factors estimation in Algo. 2 using the Parallelized TT-SVD (P-TT-SVD) where the SVD steps are performed as described in Algo. 1. It is to be mentioned that we take advantage of the coupling between the TT-cores, *i.e.*, knowing that the TT-cores $\boldsymbol{\mathcal{G}}_q$ and $\boldsymbol{\mathcal{G}}_{q+1}$ share a common change-of-basis matrix, namely $\boldsymbol{M}_q$, while $\boldsymbol{\mathcal{G}}_q$ and $\boldsymbol{\mathcal{G}}_{q-1}$ have $\boldsymbol{M}_{q-1}$ as a common change-of-basis matrix. Thus, once $\boldsymbol{\mathcal{G}}_q$ is decomposed, we use $\boldsymbol{M}_q$ to estimate $\boldsymbol{\mathcal{G}}_{q+1}$ in a Bi-ALS step, *i.e.*, a 3-rd order ALS with a pre-estimated matrix. On the other side, we use $\boldsymbol{M}_{q-1}$ to estimate $\boldsymbol{\mathcal{G}}_{q-1}$. In the same way, we continue taking advantage of the pre-estimated change of base matrices on the left and the right side in parallel [25].

Algorithm 2 R-P-JIRAFE

Input: Tensor $\boldsymbol{\mathcal{X}}$ and rank R

Output: Factor matrices $\boldsymbol{P}_q$

1: Estimation of TT-cores: $[\boldsymbol{G}_1, \boldsymbol{\mathcal{G}}_2, \ldots, \boldsymbol{\mathcal{G}}_{Q-1}, \boldsymbol{G}_Q] = \text{P-TT-SVD}(\boldsymbol{\mathcal{X}}, R)$

2: **for** $q \in \{2, \ldots, Q-1\}$ Choose $\boldsymbol{\mathcal{G}}_{q_{l'}}$ and estimate change-of-basis matrices

$$[\boldsymbol{M}_{q-1}, \boldsymbol{P}_{q_{l'}}, \boldsymbol{M}_q] = \text{Tri-ALS}(\boldsymbol{\mathcal{G}}_{q_{l'}})$$

3: **for** $l \neq l'$ Retrieve the remaining sub-matrices of the factor

$$\boldsymbol{P}_{q_l} = \boldsymbol{G}_{q_l}^{(2)}[(\boldsymbol{M}_q^{-T} \odot \boldsymbol{M}_{q-1})^T]^\dagger$$

end for

end for

4: Determine $\boldsymbol{P}_1$ and $\boldsymbol{P}_Q$: $\boldsymbol{P}_1 = \boldsymbol{G}_1 \boldsymbol{M}_1$, $\boldsymbol{P}_Q = \boldsymbol{G}_Q^T \boldsymbol{M}_{Q-1}^{-T}$

5. Randomized singular value decomposition

The dominant computational complexity for both the first and the second steps of JIRAFE scheme is the cost of the SVD. The parallelization of the TT-SVD reduces the computation time by splitting the tensor's reshapings into smaller matrices. However, the computational cost of each parallel step in the split and merge process can be reduced significantly by replacing the ordinary SVDs by randomized SVD (RSVD) [18]. In fact, instead of computing the SVD of an R-rank matrix $\boldsymbol{X}$ of dimensions $N \times M$, the RSVD consists of computing the SVD of a smaller matrix $\boldsymbol{Q}^T\boldsymbol{X}$, where $\boldsymbol{Q}$ of dimensions $N \times F$ is a random matrix with orthonormal columns, so that $\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{X} \approx \boldsymbol{X}$. The number of columns F is called the oversampling parameter. Finding $\boldsymbol{Q}$ is done by generating a random matrix $\boldsymbol{W}$ of size $M \times F$, which is projected on the matrix $\boldsymbol{X}$. Then, We extract the orthonormal basis of the projected matrix to obtain $\boldsymbol{Q}$, *i.e.*, $\boldsymbol{Q} = \text{orth}(\boldsymbol{X}\boldsymbol{W})$, where $\text{orth}(\boldsymbol{A})$ is the operator that extract the orthonormal basis of the range of $\boldsymbol{A}$. Once $\boldsymbol{Q}$ calculated, we can compute the SVD decomposition of $\boldsymbol{Q}^T\boldsymbol{X}$ of dimensions $F \times M$, *i.e.*, $\boldsymbol{Q}^T\boldsymbol{X} = \tilde{\boldsymbol{U}}\boldsymbol{S}\boldsymbol{V}^T$. Thus, the SVD decomposition of $\boldsymbol{X}$ can be obtained as follow $\boldsymbol{X} \approx \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{X} = \boldsymbol{Q}\tilde{\boldsymbol{U}}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T$, with $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$. However, the error of the RSVD remains significant compared to the classical SVD, due to the slow decay of the singular values. In order to fix this problem, the random matrix $\boldsymbol{W}$ is multiplied by

$(\boldsymbol{X}\boldsymbol{X}^T)^p\boldsymbol{X}$ instead of $\boldsymbol{X}$ in the projection step. The power p allows a faster decay to reduce the effect of the smaller singular values [18]. It is to be mentioned that the larger the value of p we choose, the faster the decay of the singular values. Moreover, increasing F, the number of columns of $\boldsymbol{Q}$, results in better performances and lower error [18]. Nevertheless, since $\boldsymbol{Q}$ is column-orthonormal, the upper limit of F is fixed by $F \leq N$. We summarize the steps of RSVD as :

- Generate random matrix $\boldsymbol{W}$ $(M \times F)$.
- Extract the orthonormal basis $\boldsymbol{Q} = \text{orth}((\boldsymbol{X}\boldsymbol{X}^T)^p\boldsymbol{X}\boldsymbol{W})$.
- Compute the SVD of the product $[\tilde{\boldsymbol{U}}, \boldsymbol{S}, \boldsymbol{V}^T] = \text{SVD}(\boldsymbol{Q}^T\boldsymbol{X})$.
- calculate the left singular matrix as $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$

Note that, for an RSVD applied on matrix $\boldsymbol{X}$ described above, authors in [18] approximate the error by

$$\mathbb{E}\left\|\boldsymbol{X} - \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{X}\right\| \leq \left[1 + \frac{4\sqrt{R+F}}{F-1} \cdot \sqrt{\min\{N, M\}}\right] \sigma_{R+1}(\boldsymbol{X}),$$

where $\mathbb{E}$ is the expectation with respect to $\boldsymbol{W}$ and $\sigma_{R+1}(\boldsymbol{X})$ is the $R+1$-th singular value of $\boldsymbol{X}$. Since the error approximation depends on $\sigma_{R+1}(\boldsymbol{X})$, it is recommended to have a sufficient power iterations number p to guarantee a fast decay of the singular values and a small value of $\sigma_{R+1}(\boldsymbol{X})$ to reduce the approximation error significantly. Nevertheless, in our case, the RSVD is applied on sub matrices $\boldsymbol{X}_j$ (see Algo. 1). Hence, the approximation error on matrix $\boldsymbol{X}$ becomes [22, eq. (9)]

$$\mathbb{E}\left\|\boldsymbol{X} - \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{X}\right\| \leq \sum_{j=1}^{J}\left[1 + \frac{4\sqrt{R+F}}{F-1} \cdot \sqrt{\min\{\frac{N}{J}, M\}}\right] \sigma_{R+1}(\boldsymbol{X}_j),$$

where $\sigma_{R+1}(\boldsymbol{X}_j)$ is the is the $R+1$-th singular value of $\boldsymbol{X}_j$.

6. Complexity analysis

A complexity analysis comparing JIRAFE and ordinary ALS/NLS algorithms have been already done in [14], with interesting results in the favor of JIRAFE. Hence, in this section, we will analyse the computational complexity of

R-P-JIRAFE introduced above, taking the complexity of JIRAFE as a reference. For the sake of simplicity, we will consider a tensor with size $N \times N \times \ldots \times N$.

6.1. Split and merge complexity

In this section, we evaluate the complexity of the TT-SVD and the P-TT-SVD for high-order large-scale tensors following the approximations used in [26]. As mentioned previously, the TT-SVD algorithm consists in applying SVDs on the R-rank Q-th order tensor unfoldings. This means that, in the first step, we apply an SVD on the first tensor unfolding of dimensions $N \times N^{Q-1}$. The complexity of truncated implementation economy SVD of a matrix $\boldsymbol{Z}$, of size $m \times n$, where only the first r singular values/vectors are calculated, is given by $O(r^2(m+n) + rmn)$. Hence, the complexity of the first step of the TT-SVD is given by $O(R^2(N + N^{Q-1}) + RN^Q)$. For large dimensions, we approximate the complexity of this step to the second term, *i.e.*, $O(RN^Q)$. Once the first dominant subspace extracted, the SVD is then applied on the remaining matrix, tensorized and then unfolded following the second dimension. This matrix is of a size $RN \times N^{Q-2}$. Hence, the computational complexity of this step is given by $O(R^2N^{Q-1})$. Thus, the complexity of the TT-SVD is given by

$$k(\text{TT-SVD}) = O(RN^Q) + O(R^2N^{Q-1}) + \ldots + O(R^2N^2). \tag{4}$$

However, using the split-and-merge algorithm, the tensor's unfoldings are divided into J sub-matrices, giving a complexity equals to $O(\frac{RN^Q}{J})$ for each sub-matrix from the first unfolding and $O(\frac{R^2N^{Q-1}}{J})$ for the second. Since these SVDs are applied on the sub-matrices on different calculators in parallel, the computational time of the P-TT-SVD will depend on the time elapsed by one calculator. Hence, instead of being interested in the global computational complexity of all the SVDs launched in parallel, we are interested only in the complexity of one SVD applied on a sub-matrix. Thus, thanks to the parallelization, the theoretical complexity of the split-and-merge algorithm for the larger dimension is divided by the number of sub-matrices. We write the

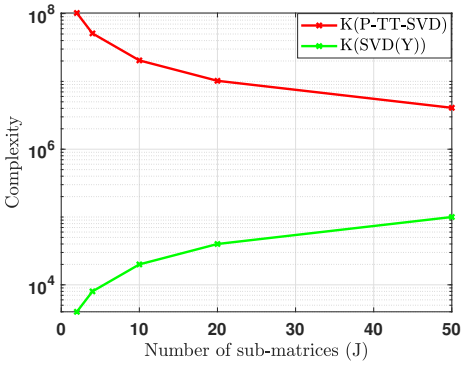


Figure 5: Computational complexity of P-TT-SVD and the SVDs applied on $\boldsymbol{Y}$ in function of J for $Q = 4$, $R = 2$ and $N = 100$

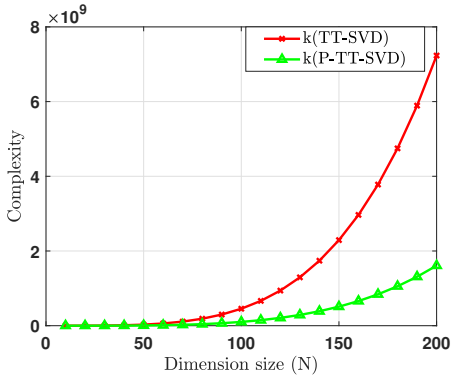


Figure 6: Computational complexity in function of the size N for $Q = 4$, $R = 2$ and $J = 4$

new complexity of the P-TT-SVD

$$k(\text{P-TT-SVD}) = O\left(\frac{RN^Q}{J}\right) + O\left(\frac{R^2N^{Q-1}}{J}\right) + \ldots + O\left(\frac{R^2N^2}{J}\right). \quad (5)$$

From eq. (4) and eq. (5), we can see that the overall complexity of the P-TT-SVD is divided by the number of sub-matrices considered in the split-and-merge algorithm. The complexity of the SVD applied on $\boldsymbol{Y}$ of size $(JR \times N)$, in step 4 of Algo. 1, is given by $k(\text{SVD}(Y)) = O(R^2(JR + N) + R^2NJ)$. For a low-rank large-scale tensor, this complexity is approximated by $O(R^2NJ)$. However, this complexity is computed for the first SVD step of the P-TT-SVD only. For the remaining $Q - 2$ P-TT-SVD steps, the size of $\boldsymbol{Y}$ becomes $(JR^2 \times N)$, which means that its complexity becomes $O(R^3NJ)$. Thus, the global complexity of SVDs applied in step 4 for all split and merge steps of P-TT-SVD as

$$k(\text{SVD}(Y)) = O(R^2NJ) + (Q - 2) \times O(R^3NJ) \quad (6)$$

As we compare $k(\text{SVD}(Y))$ in eq.(6) to k(P-TT-SVD) given in eq.(5) (see Fig. 5), we notice that k(SVD(Y)) is less significant and can be neglected for large tensors. Hence, the complexity of P-TT-SVD algorithm is approximated by k(P-TT-SVD).

In Fig. 6, we plot the complexity of the TT-SVD algorithm and the complexity of the P-TT-SVD in function of the dimension size N, for a fixed

tensor order $Q = 4$ and $R = 2$, while the number of the considered sub-matrices is fixed at $J = 4$ to be launched on 4 calculators working in parallel. We noticed, on Fig. 6, a fast exponential increase of the complexity of the TT-SVD algorithm, in function of the dimension size, compared to the slower increase of the complexity of the P-TT-SVD. Consequently, as the tensor dimension size increases, the computational time of the TT-SVD is expected to grow faster than the computional time of the P-TT-SVD.

6.2. Randomized SVD complexity

The randomized SVD reduces the computation of the parallel SVDs that we apply on the splitted tensor reshapings. However, for the first tensor unfolding in the P-TT-SVD process, instead of calculating the SVD for an $N \times (\frac{N}{J})^{Q-1}$ matrix, for which the complexity is evaluated in the previous section as $O(R^2(\frac{N}{J})^{Q-1})$, we use the RSVD to reduce the unfolding into a $F \times M$ matrix, for which the complexity is reduced to $O(R^2 M)$.

6.3. Gridding complexity

The second step of JIRAFE algorithm consists of factors retrieval by applying Tri-ALS algorithm on the TT-cores of size $R \times N \times R$. The computational complexity of a tri-ALS is given by $O(3R^3N)$ for one iteration. this complexity is then multiplied by the number of iterations in order to find the global complexity of the Tri-ALS, *i.e.*,

$$k(\text{Tri-ALS}) = O(3R^3N) \times \text{number of iterations.} \tag{7}$$

The gridding strategy allows us to divide the TT-cores following the largest dimension of size N by the number of sub-tensors L we wish to consider. Hence, the complexity of the Tri-ALS applied on the first sub-tensor becomes $O(3R^3\frac{N}{L})$ multiplied by the number of iterations. We add the complexity of Eq. 2 that we apply $(L-1)$ times closed-form in order to estimate the remaining sub-matrices of the factor. Since the dominant complexity of Eq. 2 is the complexity of the pseudo-inverse operation on the matrix $[(\boldsymbol{M}_q^{-T} \odot \boldsymbol{M}_{q-1})^T]$ of size $R^2 \times R$, that is

approximated to an SVD operation, we evaluate the complexity of this equation as $O(R^4)$. We obtain the following complexity for the gridding scheme.

$$k(Tri-ALS) = O(3R^3\frac{N}{L}) \times \text{number of iterations} + (L-1)O(R^4). \quad (8)$$

From eq. (7) and eq. (8), when the tensor size N increases, the complexity of Tri-ALS increases drastically, since it is multiplied by the number of iterations, while we can increase the number of sub-tensors L, in the case of grid scheme, in order to reduce the complexity multiplied by the number of iterations.

7. Simulation and discussion

In this section we evaluate the performances of our method using numerical simulations. First, we study the impact of the different parameters on the estimation error and the running time using synthetic data. Afterwards, we test our approach on a realistic EEG source localization context. The simulations are performed using MatLab software on a computer having a processor *Intel(R) Core(TM) i7-8650U CPU @* 1.90GHz *(8 CPUs),* 2.1GHz.

7.1. Synthetic simulation

In this part, we use a rank-2 4-th order hyper-cubic tensor $\boldsymbol{\mathcal{X}}$, following a CP model, of size $N \times N \times N \times N$, with $N = 100$. The tensor is constructed from randomly generated factors following a standard normal distribution. A Gaussian white noise tensor $\boldsymbol{\mathcal{E}}$, of the same size and order as $\boldsymbol{\mathcal{X}}$, is then added to the initial tensor, to reach a desired level of SNR. The reconstruction normalised mean squared error (NMSE) and the computational time are computed as the average over 100 noise realizations.

7.1.1. Impact of the number of sub-matrices on the TT-SVD algorithm

First, we study the impact of the number of sub-matrices on the reconstruction normalised squared error (NMSE) and the running time. For this, we apply a randomized P-TT-SVD (RP-TT-SVD, *i.e.*, a P-TT-SVD with randomised SVDs), on the tensor for $J \in \{2, 4, 10, 20\}$ and for different SNR

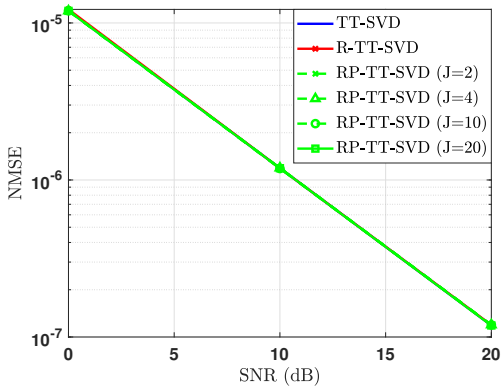


Figure 7: Normalised reconstruction mean square error (NMSE) in function of SNR (dB).

values. We compare RP-TT-SVD to the classical TT-SVD and the randomised TT-SVD (R-TT-SVD, *i.e.*, the TT-SVD with RSVDs). It is to be mentioned that we use the RSVD algorithm described in [18, algorithm 4.4]. Authors in [18] suggest that two power iterations ($p = 2$) would be sufficient to quicken the decay of the singular values and obtain a good approximation of the matrix. Likewise, they claim that, for rank-R $N \times M$ matrix $\boldsymbol{X}$, it would be enough to take a column-orthonormal matrix $\boldsymbol{Q}$ of size $N \times 2R$. Thus, we fix the number of power iterations at $p = 2$ and the second dimension size of $\boldsymbol{Q}$ at $2R$. It is to be mentioned that, for $J > 25$, the column-orthonormality condition on $\boldsymbol{Q}$, mentioned in section 5, can no longer be satisfied. In fact, in the last step of the RP-TT-SVD, splitting the remaining tensor unfolding into $J > 25$ sub-matrices means that matrix $\boldsymbol{Q}$ will have $\frac{N}{J} < 4$ columns and $2R = 4$ rows, which violates the column-orthogonality condition. Hence, we stop at $J = 20$. The NMSE results in Fig. 7 show identical reconstruction error for all the compared algorithms. Hence, changing the number of sub-matrices J has no effect on the reconstruction error. On the other hand, table 1 shows no significant changes in computational time as we increase SNR. This is thanks to the fact that all the compared methods are algebraic. RP-TT-SVD shows the lowest computational time, followed by R-TT-SVD, while the classical TT-SVD has the highest time.

	Running time (s)			
SNR (dB)	0	10	20	Gain
TT-SVD	5.79	5.28	5.82	1
R-TT-SVD	0.49	0.51	0.51	11
RP-TT-SVD ($J = 2$)	0.23	0.24	0.25	23
RP-TT-SVD ($J = 4$)	0.12	0.12	0.12	46
RP-TT-SVD ($J = 10$)	0.071	0.069	0.070	80
RP-TT-SVD ($J = 20$)	0.072	0.071	0.071	79

Table 1: Running time (s) in function of SNR (dB). The last column represents the gain of each algorithm compared to the TT-SVD. The gain of each algorithm computed as the average computational time of the TT-SVD over the SNR values divided by the average computational time of each algorithm.

Moreover, the gain of the RP-TT-SVD increases as we increase the number of sub-matrices, as observed for the values $J = 2$, $J = 4$ and $J = 10$. However, for $J = 20$, the gain decreases slightly despite the fact that the dimensions of sub-matrices are smaller. This is due to the increase of the number of sub-matrices to be computed. Hence, J should be chosen in order to make a compromise between the number of the sub-matrices to be computed and their dimensions. In our case, $J = 10$ shows the most interesting results in this case.

7.1.2. Impact of the number of sub-tensors on the factor estimation

In this part, we study the impact of the number of sub-tensors L on both the tensor reconstruction error and the running time. For this, we use the classical TT-SVD to estimate the TT-cores. Once the TT-cores estimated, we estimate the factors twice : once using Tri-ALS and once using Grid-ALS, in order to compare the reconstruction error and running time of these algorithms. Note that we take into account only the elapsed time to estimate the factors from the TT-cores. For the Grid-ALS, we consider L sub-tensors, with $L \in \{2, 4, 10\}$. Fig. 8 shows that the NMSE decreases as we increase the SNR values since the ALS compared algorithms are iterative (noise-sensitive). Moreover, it shows similar reconstruction error between Tris-ALS and Grid-ALS for all the considered L values. On the other hand, Table 2 shows a decrease in terms of

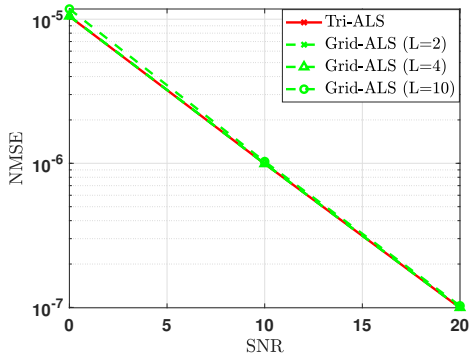


Figure 8: Normalised reconstruction mean square error (NMSE) in function of SNR (dB).

	Running time (s)			
SNR (dB)	0	10	20	Gain
Tri-ALS	0.123	0.080	0.038	1
Grid ($L = 2$)	0.109	0.057	0.029	1.2
Grid ($L = 4$)	0.095	0.050	0.027	1.4
Grid ($L = 10$)	0.090	0.048	0.024	1.5

Table 2: Running time (s) in function of SNR (dB). The last column represents the gain of each algorithm compared to the Tri-ALS algorithm.

computational time as we increase the number of sub-tensors L. Overall, the dimensions of the TT-cores in this case are not too large ($2 \times 100 \times 2$), which explains the small computational time elapsed to perform the decomposition. However, the gain is expected to increase for larger dimensions as the running time increases comparing to the time elapsed for elementary operations.

7.2. Realistic data

In this section, the efficiency of our strategy is evaluated on realistic EEG signals. The aim is to localize brain epileptic sources and to estimate their spatial extent, using the scalp EEG data and the lead field matrix, which models the transfer between brain and scalp electrical activity. This ill-posed linear inverse problem is well-known as the brain source imaging problem. The lead field matrix is computed using a real head geometry and the Boundary

Element Method (BEM) to solve Poisson equation in the head medium [27]. EEG spatio-temporal matrices $\boldsymbol{Z}$ are generated as $\boldsymbol{Z} = \boldsymbol{G}\boldsymbol{S}$, where $\boldsymbol{S}$ represents the source signals and $\boldsymbol{G}$ represents the known lead field matrix standing for the attenuation that the source signals are subject to [10]. It is to be mentioned that the source signals are taken from real interictal epileptic data with spikes. We apply a wavelet transform on the matrix $\boldsymbol{Z}$, using the Morlet wavelet, in order to obtain a space-time-scales 3-rd order tensor that we call $\boldsymbol{\mathcal{X}}$ $(N_1 \times N_2 \times N_3)$. Finally, we simulate the repetition of spikes which is observed in practice. In order to do that, we introduce a weight factor matrix as a 4-th dimension of a new 4-th order tensor that we call $\boldsymbol{\mathcal{Y}}$. The goal is to estimate the spatial factor matrix, that we call $\boldsymbol{A}_1$, of size $N_1 \times R$, where R is the tensors canonical rank, corresponding to the first mode. The second factor matrix is $\boldsymbol{A}_2$, of size $(N_2 \times R)$, representing the temporal dynamics, while the third factor matrix $(N3 \times R)$ stands for the scales obtained by the wavelet transform. The fourth factor $(N_4 \times R)$ is the spike repetition factor matrix. We apply our approach on the tensor $\boldsymbol{\mathcal{Y}}$ and compare it to JIRAFE and the classical 4-th order Nonlinear Least Squares (NLS) algorithm provided by tensorlab toolbox [28], and to the 3-rd order NLS that we apply on the tensor $\boldsymbol{\mathcal{X}}$. Once $\boldsymbol{A}_1$ estimated, under the assumption of a known lead field matrix, and since only a small number of dipoles contribute to each patch of interest, a sparsity penalty is imposed on the matrix $\boldsymbol{A}_1$ in order to identify the position/extent of each patch [10]. This step is done by the algorithm called Source Imaging based on Structured SparsitY (SISSY) [10]. For the sake of approaching realistic context, we consider 3 scenarios according to the position of the epileptic patches, the number of patches and the level of correlation between the signal dynamics :

- First scenario: two distant patches on the brain, with less correlated dynamics, *i.e.*, the correlation between the dynamics between the two patches is lower than a threshold fixed to 0.7.
- Second scenario: two close patches are present, with higher correlation than the threshold.

- Third scenario: three patches are present, from which two are spatially close and correlated, while the third patch is distant with dynamics less correlated to those of the first and second patches.

We add white Gaussian noise, standing for the noise generated by instruments. In addition, we add modeled noise standing for the non epileptic brain activity called "the background activity" [10]. The dimensions we chose are $N_1 = 91$ electrodes, $N_2 = 200$ time samples with a sampling frequency of $f_s = 256$Hz, while the scale number is $N_3 = 60$ obtained by the wavelet transform. The repetition number is $N_4 = 50$. The simulations are done for 100 runs for each scenario, for three values of SNR, *i.e.*, SNR = 0 dB, SNR = 10 dB and SNR = 20 dB. We use as performance criterion the distance of localization error (DLE) [29], defined as the distance between the original sources and the estimated sources. Likewise, we compare the time of execution of the four approaches. In this application, the main goal is to estimate the spatial dimension, *i.e.*, the first factor matrix only. This means that there is no need to estimate the remaining factors. Therefore, we adapt our algorithm to this context and we stop once the first factor is estimated. Hence, the computational time is reduced in a drastic way. Note that eq. (3) means that the estimation of the first factor matrix requires only the first TT-core $\boldsymbol{G}_1$ obtained from the first TT-SVD step, and the first change-of-base matrix $\boldsymbol{M}_1$ obtained from a single sub-tensor $\boldsymbol{\mathcal{G}}_{2_{l'}}$ in the grid of the second TT-core $\boldsymbol{\mathcal{G}}_2$ (see Fig. 3). Since $\boldsymbol{G}_1$ and $\boldsymbol{\mathcal{G}}_2$ are obtained by the first two steps of the TT-SVD, it becomes unnecessary to estimate the rest of the TT-cores in the case where only the first factor is needed. However,

Algorithm 3 R-P-JIRAFE

Input: Tensor $\boldsymbol{\mathcal{Y}}$ and rank R

Output: The first factor matrix $\boldsymbol{P}_1$

1: Estimation of TT-cores: $[\boldsymbol{G}_1, \boldsymbol{\mathcal{G}}_2] = \text{P-TT-SVD}(\boldsymbol{\mathcal{Y}}, R)$

2: Choose $\boldsymbol{\mathcal{G}}_{2_{l'}}$ and estimate $\boldsymbol{M}_1$: $l' = \text{argmax}_l ||\boldsymbol{\mathcal{G}}_{2_l}||_{Fro}$

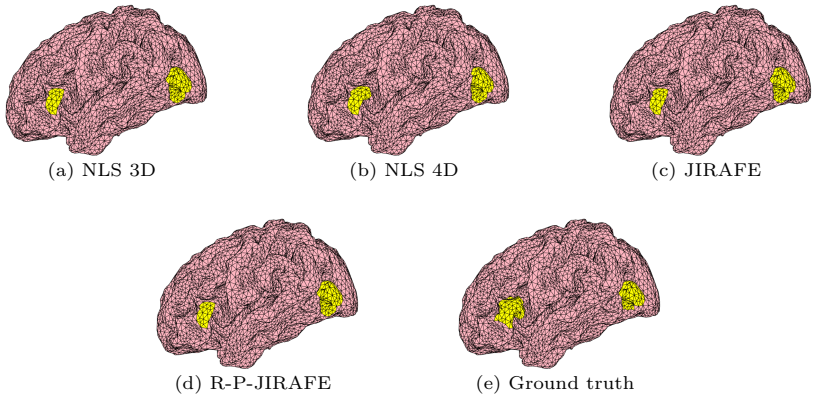
$[\boldsymbol{M}_1, \boldsymbol{P}_{2_{l'}}, \boldsymbol{M}_2] = \text{Tri-ALS}(\boldsymbol{\mathcal{G}}_{2_{l'}})$

3: Determine the first factor: $\boldsymbol{P}_1$ $\boldsymbol{P}_1 = \boldsymbol{G}_1 \boldsymbol{M}_1$

we mentioned previously that the choice of the l'-th sub-tensor depends on

the application. We remind that, in our case, the second factor represents the temporal dimension. Hence, knowing that the source signals are impulsive, we chose the sub-tensor $\boldsymbol{\mathcal{G}}_{2_{l'}}$ that has the greatest Frobenius norm value. This means that we chose the part of the signal that corresponds to the peak of the impulse, where the source signal is at its greatest value compared to background activity and noise. In other words, we chose $\boldsymbol{\mathcal{G}}_{2_{l'}}$ so that $l' = \text{argmax}_l ||\boldsymbol{\mathcal{G}}_{2_l}||_{Fro}$. We describe this process in Algo. 3, where we use only two steps of P-TT-SVD in order to estimate the first and the second TT-cores in step 1. Then, in step 2, we choose the sub-tensor $\boldsymbol{\mathcal{G}}_{2_{l'}}$ that we integrate into a Tri-ALS process to estimate the change-of-basis matrix $\boldsymbol{M}_1$. Finally, we use $\boldsymbol{M}_1$ in step 3 to determine the first factor $\boldsymbol{P}_1$.

7.2.1. First scenario



(a) NLS 3D (b) NLS 4D (c) JIRAFE

(d) R-P-JIRAFE (e) Ground truth

Figure 9: Example of epileptic source localization for two distant patches (SNR = 10 dB)

In this scenario we consider two distant patches, to simulate the case where the interictal activity is generated by two distant epileptic generators in the brain. Since the two sources are not close, it is reasonable to consider a low correlation between the corresponding activities. As shown in Fig. 9, the source localization allows for the reconstruction of the activity of the two patches on the surface of the brain. We can see that the 4 methods have similar results, with slight differences, in terms of the shape of the patches, compared to the ground

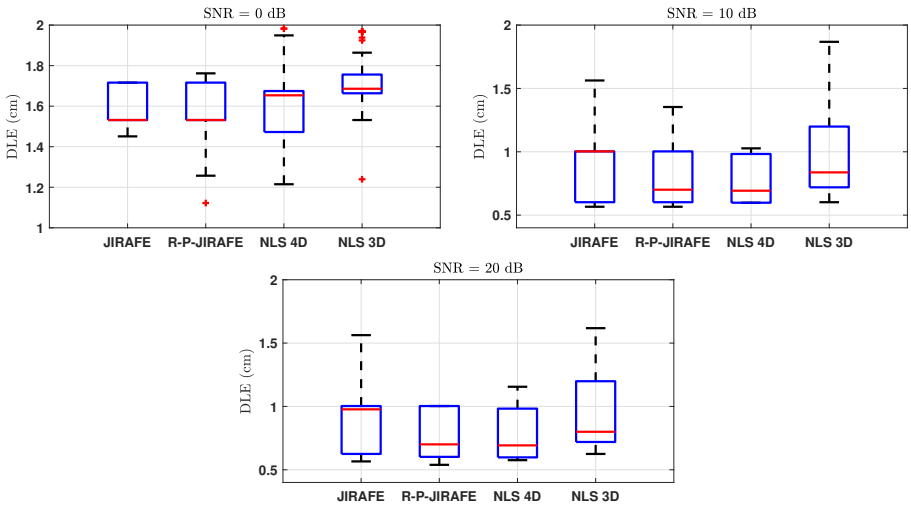


Figure 10: Boxplots of DLE (cm) for two distant patches over 100 runs

truth taken as a reference. Fig. 10 represents the DLE of the 4 approaches, for 3 values of SNR, obtained for the two distant patches. At SNR=0 dB, the upper quartile of the 4-th order NLS is slightly lower than the median of JIRAFE, R-P-JIRAFE and 3-rd order NLS. As we increase the SNR, we notice that the DLE of the 4 methods is becoming comparable with slight differences in terms of the median in favor of R-P-JIRAFE and 4-th order NLS. However, comparing the upper/lower quartiles, the minimum and the maximum, we can see that the methods using 4-th order tensor, *i.e.*, JIRAFE, R-P-JIRAFE and 3-rd order NLS, are slightly better than 3-rd order NLS. On the other hand, Table

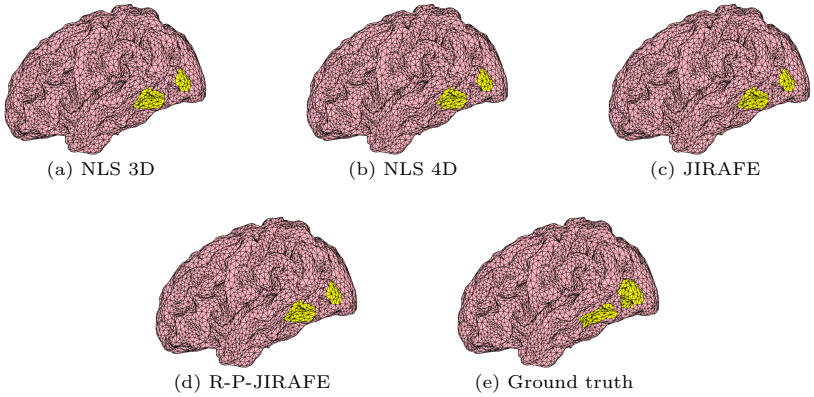
	Execution time (s)			
SNR (dB)	JIRAFE	R-P-JIRAFE	NLS-CPD (4D)	NLS-CPD (3D)
0	1.80	0.16	25.50	0.73
10	1.77	0.14	25.23	0.63
20	1.72	0.13	25.09	0.60

Table 3: Execution time for two distant patches over 100 runs, where the shortest execution time is marked in red

3 shows that R-P-JIRAFE has the shortest execution time, with a constant gain

regarding the other methods as we increase the SNR. In addition, the execution time of R-P-JIRAFE is even lower than the execution time of 3-rd order NLS with a constant gain (around 4.5).

7.2.2. Second scenario



(a) NLS 3D (b) NLS 4D (c) JIRAFE

(d) R-P-JIRAFE (e) Ground truth

Figure 11: Example of epileptic source localization for two close patches (SNR = 10 dB)

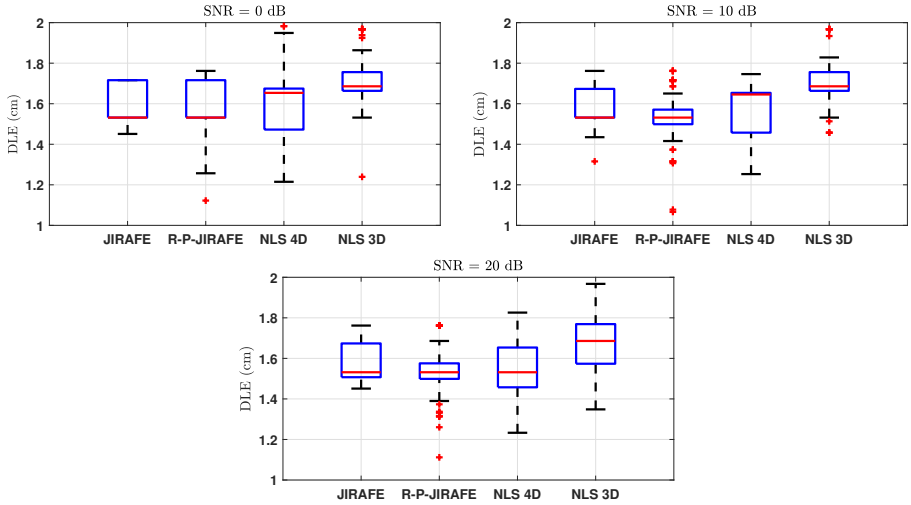


Figure 12: Boxplots of DLE (cm) for two close patches over 100 runs

We consider two close patches, with a high correlation between their activities. Since the electrical activity propagates from one patch to the other,

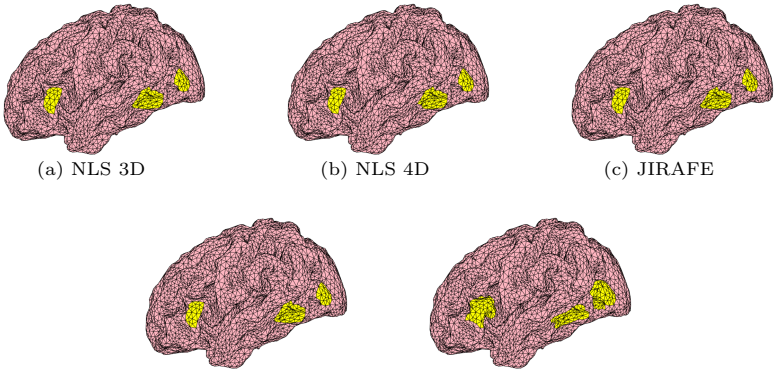
the proximity of both patches justifies their high correlation. We can see in Fig. 11 that the reconstruction of the patch activity using the 4 methods gives identical results with a slight difference in the shape of patches comparing to the ground truth. In order to visualize the estimation error for each method, we plot the DLE of the 4 methods in Fig. 12. At SNR = 0 dB, the boxplots show a slightly lower DLE error in favor of the approaches using 4-th order tensor compared to 3-rd order NLS. At SNR = 10 and 20 dB, the difference in favor of R-P-JIRAFE, JIRAFE and 4-th order NLS becomes clearer as the upper quartiles of these methods become lower than the median of 3-rd order NLS. However, the results of R-P-JIRAFE, JIRAFE and 4-th order NLS are comparable in general. In the same way as in the first scenario, Table 4, shows that R-P-JIRAFE has the shortest execution time. In fact, R-P-JIRAFE is about 5 times faster than the 3-rd order NLS and around 150 times faster than 4-th order NLS for all SNR values.

	Execution time (s)			
SNR (dB)	JIRAFE	R-P-JIRAFE	NLS-CPD (4D)	NLS-CPD (3D)
0	1.43	0.14	19.12	0.68
10	1.30	0.12	18.26	0.54
20	1.30	0.11	17.69	0.51

Table 4: Execution time for two close patches over 100 runs, where the shortest execution time is marked in red

7.2.3. Third scenario

We consider three patches, two of them are close and correlated, while the third one is distant and has a low correlation with the two other patches. As in the first two scenarios, we can see the identical reconstruction of the three patches on the surface of the brain, using the 4 methods, in Fig. 13, with a slight difference in the shape of the patches comparing to the ground truth. DLE boxplots in Fig. 14, show that the median of all methods is comparable, with slight differences, through the different values of SNR. However, comparing the minimum, maximum, upper and lower quartiles, we can see that 4-th order



(a) NLS 3D (b) NLS 4D (c) JIRAFE

(d) R-P-JIRAFE (e) Ground truth

Figure 13: Example of epileptic source localization for one distant patch and two close patches (SNR = 10 dB)

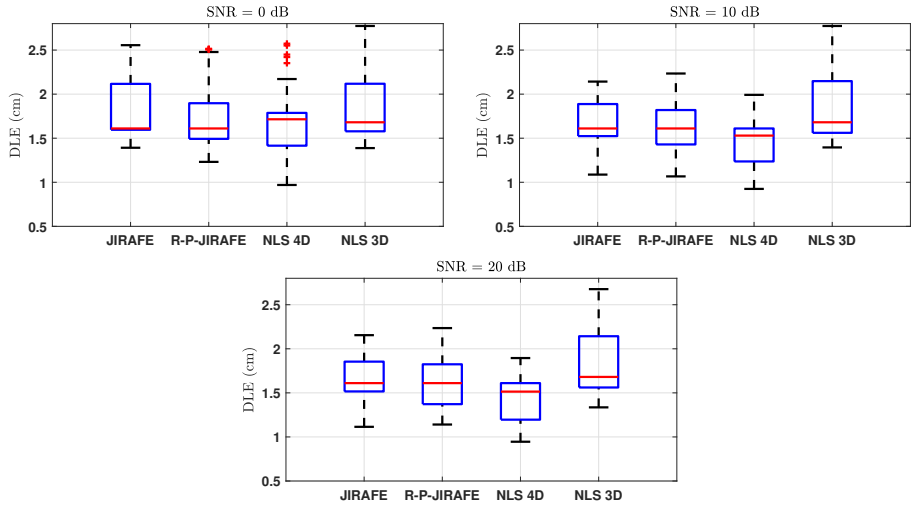


Figure 14: Boxplots of DLE (cm) for one distant patch and two close patches over 100 runs

NLS has the lowest values followed by R-P-JIRAFE, JIRAFE and finally 3-rd order NLS. On the other hand, Table 5, shows the same temporal gain in favor of R-P-JIRAFE, over 3-rd order NLS. However, 4-th order NLS becomes slower as the canonical rank of the tensor becomes 3, this makes this approach 200 times slower than R-P-JIRAFE.

	Execution time (s)			
SNR (dB)	JIRAFE	R-P-JIRAFE	NLS-CPD (4D)	NLS-CPD (3D)
0	2.13	0.22	44.20	1.13
10	1.94	0.21	44.12	1.05
20	1.88	0.19	44.69	1.02

Table 5: Execution time for the three patches scenario over 100 runs, where the shortest execution time is marked in red

7.3. Discussion

The results presented previously show no significant difference in the reconstruction of the patch activity between the compared methods. Regarding the DLE, no significant difference has been noticed between the four methods in the first scenario, except for a slight gap that appears as we increase the SNR values, in the favor of the three methods using the 4-th order tensor, *i.e.*, JIRAFE, R-P-JIRAFE and 4-th order NLS. As we move to the second and the third scenarios, with the presence of two close patches with a high correlation of their activities, the latter methods (using the 4-th order tensor model) gain clearly the upper hand over 3-rd order NLS. In fact, 4-th order NLS, JIRAFE and R-P-JIRAFE have comparable DLEs, while 3-rd order NLS has higher DLE values. From this we note that, in the presence of correlated and close patches (scenarios 2 and 3), the methods using 4-th order tensor have lower DLE values than 3-rd order NLS, which means that they provide a better localization. This observation is valid for both the second and the third scenarios, thanks to the better separability properties provided by the additional repetition dimension added to the 3-rd order tensor to obtain the 4-th order tensor model. Thus, using the 4-th order tensor results in lower DLEs, *i.e.*, a better the source localization. It is to be noted that R-P-JIRAFE shows better DLE results than JIRAFE for low SNR values, with less dispersed results in the second scenario, by means of choosing the sub-tensor with the greatest Frobenius norm in step 2 of Algo. 3, which is the less affected by noise. This makes the estimation of the factors more accurate. Likewise, throughout the 3 scenarios, the DLE of R-P-

JIRAFE is comparable to that of 4-th order NLS, with insignificant differences. On the other hand, when comparing the computation time, R-P-JIRAFE shows the best performance for all the three scenarios with significant gains (around 5 times faster than 3-rd order NLS, and 200 times faster than 4-th order NLS). Therefore, it is fair to say that R-P-JIRAFE offers the best compromise between localization accuracy and computational time among the 4 tested methods.

8. Conclusion

Tensor modeling has brought many advantages to data analysis, helping to take into account the multidimensionality of the data. However, this advantages have a computational cost that appears when dealing with high-order and large-scale tensors, which is often the case in EEG context. As the tensors are becoming, at the same time, of high-order and large-scale, the computational complexity increases drastically. In this paper, we proposed a methodological scheme based on JIRAFE, algorithm to deal simultaneously with the problems of high-order and large scales, with the aim of parallelizing and optimizing the computation steps. In practise, this approach has shown efficiency in reducing the computational time into reasonable values compared to the traditional tensor decomposition algorithms. Moreover, this comparison allowed us to confirm practically the advantages of using high-order tensors. In fact, the additional information contained in the added dimensions allows better separation between sources in the case of multiple brain sources. As a result, high-orders (> 3) may be considered with the aim of improving the separability and, thus, the localization. This has been shown by adding a dimension of repetitions, which results in a more precised source localization compared to the usual 3-rd order tensor model (space, time, scale or frequency).

References

[1] N. Lee, A. Cichocki, Fundamental tensor operations for large-scale data analysis using tensor network formats, Multidimensional Syst. Signal Process. 29 (3) (2018) 921–960. doi:10.1007/s11045-017-0481-0.

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, H. A. PHAN, Tensor decompositions for signal processing applications: From two-way to multiway component analysis, IEEE Signal Processing Magazine 32 (2) (2015) 145–163. doi:10.1109/MSP.2013.2297439.

[3] F. Cong, Q. H. Lin, L. D. Kuang, X. F. Gong, P. Astikainen, T. Ristaniemi, Tensor decomposition of EEG signals: A brief review, Journal of Neuroscience Methods 248 (2015) 59 – 69. doi:https://doi.org/10.1016/j.jneumeth.2015.03.018.

[4] L. D. Lathauwer, A short introduction to tensor-based methods for factor analysis and blind source separation, in: 2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA), 2011, pp. 558–563.

[5] A. Stegeman, N. D. Sidiropoulos, On Kruskal's uniqueness condition for the CANDECOMP/PARAFAC decomposition, Linear Algebra and its Applications 420 (2) (2007) 540–552. doi:https://doi.org/10.1016/j.laa.2006.08.010.

[6] H. Becker, L. Albera, P. Comon, M. Haardt, G. Birot, F. Wendling, M. Gavaret, C. Bénar, I. Merlet, EEG extended source localization: Tensor-based vs. conventional methods, NeuroImage 96 (2014) 143 – 157. doi:https://doi.org/10.1016/j.neuroimage.2014.03.043.

[7] H. Becker, L. Albera, P. Comon, R. Gribonval, F. Wendling, I. Merlet, Brain-source imaging: From sparse to tensor models, IEEE Signal Processing Magazine 32 (6) (2015) 100–112. doi:10.1109/MSP.2015.2413711.

[8] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends® in Machine Learning 9 (4-5) (2016) 249–429. doi:10.1561/2200000059.

[9] C. I. Kanatsoulis, N. D. Sidiropoulos, Large-scale Canonical Polyadic decomposition via regular tensor sampling, in: 2019 27th European Signal Processing Conference (EUSIPCO), 2019, pp. 1–5. doi:10.23919/EUSIPCO.2019.8902959.

[10] H. Becker, L. Albera, P. Comon, J.-C. Nunes, R. Gribonval, J. Fleureau, P. Guillotel, I. Merlet, SISSY: An efficient and automatic algorithm for the

analysis of EEG sources based on structured sparsity, NeuroImage 157 (2017) 157 – 172. doi:https://doi.org/10.1016/j.neuroimage.2017.05.046.

[11] P. Comon, X. Luciani, A. L. F. De Almeida, Tensor decompositions, alternating least squares and other tales, Journal of Chemometrics: A Journal of the Chemometrics Society 23 (7-8) (2009) 393–405. doi:https://doi.org/10.1002/cem.1236.

[12] E. Acar, T. G. Kolda, D. M. Dunlavy, All-at-once Optimization for Coupled Matrix and Tensor Factorizations, in 9th Workshop on Mining and Learning with Graphs, San Diego, CA (2011).

[13] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317. doi:10.1137/090752286.

[14] Y. Zniyed, R. Boyer, A. L. F. de Almeida, G. Favier, High-order CPD estimation with dimensionality reduction using a tensor train model, 2018 26th European Signal Processing Conference (EUSIPCO) (2018) 2613–2617doi:10.23919/EUSIPCO.2018.8553466.

[15] Y. Zniyed, R. Boyer, A. L. de Almeida, G. Favier, Multidimensional harmonic retrieval based on Vandermonde tensor train, Signal Processing 163 (2019) 75 – 86. doi:https://doi.org/10.1016/j.sigpro.2019.05.007.

[16] A. H. Phan, A. Cichocki, PARAFAC algorithms for large-scale problems, Neurocomputing 74 (11) (2011) 1970 – 1984. doi:https://doi.org/10.1016/j.neucom.2010.06.030.

[17] P. Hall, D. Marshall, R. Martin, Merging and splitting eigenspace models, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (9) (2000) 1042–1049. doi:10.1109/34.877525.

[18] N. Halko, P. Martinsson, J. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2011) 217–288. doi:10.1137/090771806.

[19] R. A. Harshman, Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis, UCLA Working Papers in Phonetics 16 (1970) 1–84.

[20] M. W. Berry, D. Mezher, B. Philippe, A. Sameh, Parallel algorithms for the singular value decomposition, in: Handbook of parallel computing and statistics, Chapman and Hall/CRC, 2005, pp. 133–180.

[21] J. Tzeng, Split-and-combine singular value decomposition for large-scale matrix, Journal of Applied Mathematics 2013 (03 2013). doi:10.1155/2013/683053.

[22] F. Liang, R. Shi, Q. Mo, A split-and-merge approach for singular value decomposition of large-scale matrices, Statistics and its interface 9 (4) (2016) 453—459. doi:10.4310/sii.2016.v9.n4.a5.

[23] Q. Shi, H. Lu, Y.-m. Cheung, Tensor rank estimation and completion via cp-based nuclear norm, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 949–958.

[24] N. Taheri, A. Kachenoura, A. Karfoul, X. Han, K. Ansari-Asl, I. M. L. Senhadji, L. Senhadji, L. Albera, Rank estimation and tensor decomposition using physics-driven constraints for brain source localization, in: 2019 27th European Signal Processing Conference (EUSIPCO), 2019, pp. 1–4. doi:10.23919/EUSIPCO.2019.8902585.

[25] A. Boudehane, Y. Zniyed, A. Tenenhaus, L. L. Brusquet, R. Boyer, Breaking the curse of dimensionality for coupled matrix-tensor factorization, 2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP) (2019) 689–693doi:10.1109/CAMSAP45676.2019.9022462.

[26] Y. Zniyed, R. Boyer, A. L. F. de Almeida, G. Favier, A TT-based hierarchical framework for decomposing high-order tensors, SIAM Journal on Scientific Computing 42 (2) (2020) A822–A848. doi:10.1137/18M1229973.

[27] D. Cosandier-Rimélé, J.-M. Badier, P. Chauvel, F. Wendling, A physiologically plausible spatio-temporal model for EEG signals recorded with intracerebral electrodes in human partial epilepsy, IEEE transactions on bio-medical engineering 54 (2007) 380–8. doi:10.1109/TBME.2006.890489.

[28] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, L. De Lathauwer, Tensorlab 3.0

(2016).
URL https://www.tensorlab.net

[29] J. Cho, S. Hong, Y. Jung, H. Kang, H. Kim, M. Suh, K. Jung, C. Im, Evaluation of algorithms for intracranial EEG (iEEG) source imaging of extended sources: Feasibility of using iEEG source imaging for localizing epileptogenic zones in secondary generalized epilepsy, Brain Topography 24 (2) (2011) 91–104. doi:10.1007/s10548-011-0173-2.