Canonical polyadic decomposition of 3rd order semi-nonnegative semi-symmetric tensors using LU and QR matrix factorizations

Lu Wang, Laurent Albera, Amar Kachenoura, Huazhong Shu, Lotfi Senhadji

RESEARCH

Canonical polyadic decomposition of 3rd order semi-nonnegative semi-symmetric tensors using LU and QR matrix factorizations

Lu Wang[1,2,4], Laurent Albera[1,2,4,5*], Amar Kachenoura[1,2,4], Huazhong Shu[3,4] and Lotfi Senhadji[1,2,4]

*Correspondence: laurent.albera@univ-rennes1.fr
[1]INSERM, UMR 1099, F-35000 Rennes, France
Full list of author information is available at the end of the article

Abstract

Semi-symmetric three-way arrays are essential tools in Blind Source Separation (BSS) particularly in Independent Component Analysis (ICA). These arrays can be built by resorting to higher order statistics of the data. The Canonical Polyadic (CP) decomposition of such semi-symmetric three-way arrays allows us to identify the so-called mixing matrix, which contains the information about the intensities of some latent source signals present in the observation channels. In addition, in many applications, such as the Magnetic Resonance Spectroscopy (MRS), the columns of the mixing matrix are viewed as relative concentrations of the spectra of the chemical components. Therefore, the two loading matrices of the three-way array, which are equal to the mixing matrix, are nonnegative. Most existing CP algorithms handle the symmetry and the nonnegativity separately. Up to now, very few of them consider both the semi-nonnegativity and the semi-symmetry structure of the three-way array. Nevertheless, like all the methods based on line search, trust region strategies and alternating optimization, they appear to be dependent on initialization, requiring in practice a multi-initialization procedure. In order to overcome this drawback, we propose two new methods, called $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$, to solve the problem of CP decomposition of semi-nonnegative semi-symmetric three-way arrays. Firstly, we rewrite the constrained optimization problem as an unconstrained one. In fact, the nonnegativity constraint of the two symmetric modes is ensured by means of a square change of variable. Secondly, a Jacobi-like optimization procedure is adopted because of its good convergence property. More precisely, the two new methods use LU and QR matrix factorizations, respectively, which consist in formulating high-dimensional optimization problems into several sequential polynomial and rational subproblems. By using both LU and QR matrix factorizations we aim at studying the influence of the used matrix factorization. Numerical experiments on simulated arrays emphasize the advantages of the proposed methods especially the one based on LU factorization, in the presence of high-variance model error and of degeneracies such as bottlenecks. A BSS application on MRS data confirms the validity and improvement of the proposed methods.

Keywords: Canonical polyadic decomposition; semi-nonnegative semi-symmetric tensor; joint diagonalization by congruence; individuals differences in scaling analysis; blind source separation; independent component analysis; magnetic resonance spectroscopy

Introduction

Higher Order (HO) arrays, commonly called tensors, play an important role in numerous applications, such as chemometrics [1], telecommunications [2], and biomed-

ical signal processing [3]. They can be seen as HO extensions of vectors (1-way arrays) and matrices (2-way arrays). In many practical situations the available data measurements cannot be arranged into a tensor form directly, that is to say, the observation diversity is insufficient either in time or frequency. However, if the latent data satisfies the statistical independence assumption, which is reasonable in many applications, meaningful HO arrays can be built by resorting to HO Statistics (HOS) of the data [4]. In this instance, the HO arrays are partially symmetric or Hermitian due to the special algebraic structure of the basic HOS, such as moments and cumulants. In Independent Component Analysis (ICA), the latent physical phenomena which are assumed to be statistically independent can be revealed by decomposing the HO array into factors. There exists several ways to decompose a given HO array, such as the Tucker model [5,6]. Among the existing reliable HO array decomposition models, the Canonical Polyadic (CP) decomposition model have attracted much attention. Indeed its uniqueness properties can be ensured under the sufficient conditions established by Kruskal [7]. In addition unlike the HO Singular Value Decomposition (HOSVD) [6], CP model does not impose any orthogonality constraint on its factors.

Theoretically, a polyadic decomposition exactly fits an array by a sum of rank-one terms [8]. A CP decomposition is defined as a polyadic decomposition with a minimal number of rank-one terms which are needed to exactly fit a given HO array. Currently the CP decomposition is gaining importance in several applications, for example, in exploratory data analysis [9], sensor array processing [10], telecommunications [11, 12], ICA [13] and in multiple-input multiple-output radar systems [14]. A multitude of methods were developed to compute the CP decomposition. They include the iterative Alternating Least Squares (ALS) procedure [15], which gains popularity due to its simplicity of implementation and low numerical complexity. Uschmajew proved the local convergence property of ALS under some conditions [16]. However, this convergence can be slow. Therefore an Enhanced Line Search (ELS) procedure was proposed by Rajih et al. [17] to cope with the slow convergence problem of ALS. Other approaches were also proposed, such as the conjugate gradient algorithm [18] and joint eigenvalue decomposition based algorithms [19, 20], to cite a few. Some HO arrays enjoy certain properties, such as ***i***) *symmetry* and ***ii***) *nonnegativity*, which can not be simply handled by the aforementioned general CP decomposition methods. Therefore, special CP models become more and more important.

The first special form of the CP model for 3-way arrays that are symmetric in two modes, brings the concept of INdividuals Differences in SCALing (INDSCAL) analysis [21]. On one hand, INDSCAL analysis has been studied as a way of multiple factor analysis [22] with applications to chemometrics, psychology and marketing research. On the other hand, in the domain of signal processing, and more particularly in Blind Source Separation (BSS), the INDSCAL analysis is widely known as the Joint Diagonalization of a set of matrices by Congruence (JDC). During the past two decades, many successful JDC methods have been proposed, such as Yeredor's Alternating Columns and Diagonal Center (ACDC) algorithm [23], the Joint Approximate Diagonalization (JAD) algorithm proposed by Cardoso and Souloumiac [24], the Fast Frobenius DIAGonalization (FFDIAG) algorithm proposed by Ziehe et al. [25], Afsari's LUJ1D algorithm [26], and many others [27–33].

A recent survey of JDC can be found in [34]. The second special form of CP model is defined when all the factors in the CP decomposition are constrained to be nonnegative, commonly known as Nonnegative Tensor Factorization (NTF). NTF can be regarded as the extension of Nonnegative Matrix Factorization (NMF) [35] to higher orders. In many applications, the physical properties are inherently nonnegative, such as chemistry [1] and fluorescence spectroscopy [36, 37]. In those applications the results are only meaningful if the nonnegativity constraint is satisfied. Various methods for computing NTF and also NMF can be found in [38, 39].

So far, the CP model with both the symmetry and nonnegativity constraints has not received much attention. Coloigner et al. proposed a family of algorithms based on line search and trust region strategies [40]. Wang et al. developed an alternating minimization scheme [41]. Those methods appear to depend on initialization, and therefore in practice require a multi-initialization procedure, leading to an increase of numerical complexity. In this paper, we propose to fit the CP model of a 3-way array by imposing both the semi-nonnegativity and the semi-symmetry constraints. More precisely, we impose a nonnegativity constraint on the two symmetric modes of the INDSCAL model, which leads to the semi-nonnegative INDSCAL model or equivalently the CP decomposition of semi-nonnegative semi-symmetric 3-way arrays. Such a model is often encountered in ICA problems where a nonnegative mixing matrix is frequently considered. For example, in Magnetic Resonance Spectroscopy (MRS), the columns of the mixing matrix represent the positive concentrations of the source metabolites. Then the 3-way array built by stacking the matrix slices of a cumulant array is both nonnegative and symmetric in two modes. In such a case, the semi-nonnegative INDSCAL problem is equivalent to the JDC problem subject to a nonnegativity constraint on the joint transformation matrix. We propose two new algorithms to solve the semi-nonnegative INDSCAL problem, called $\mathrm{JD}^{+}_{\mathrm{LU}}$ and $\mathrm{JD}^{+}_{\mathrm{QR}}$. Firstly, we rewrite the constrained optimization problem as an unconstrained one. Actually the nonnegativity constraint is ensured by means of a square change of variable. Secondly, we propose two Jacobi-like approaches using LU and QR matrix factorizations, respectively, which consist in formulating high-dimensional optimization problems into several sequential polynomial and rational subproblems. By using both LU and QR matrix factorizations we aim at studying the influence of the used matrix factorization. Numerical experiments highlight the advantages of the proposed methods especially $\mathrm{JD}^{+}_{\mathrm{LU}}$, in the case of dealing with high-variance model error and with degeneracies such as bottlenecks. A BSS application on MRS signals confirms the validity and improvement of the proposed methods. A part of this work has been recently presented at the 8th IEEE Sensor Array and Multichannel signal processing workshop [42].

The rest of the paper is organized as follows. After the presentation of some notations, the second section introduces some basic definitions of the multilinear algebra then gives the semi-nonnegative INDSCAL problem formulation. In section three, we describe the proposed algorithms in details and we also provide an analysis of the numerical complexities. Section four shows the computer simulation results. Finally we conclude the paper.

Multilinear algebra prerequisites and problem statement

Notations

The following notations are used throughout this paper. $\mathbb{R}^{N_1 \times N_2 \times \cdots \times N_i}$ and $\mathbb{R}_+^{N_1 \times N_2 \times \cdots \times N_i}$ denote the set of real-valued $(N_1 \times N_2 \times \cdots \times N_i)$ arrays and the set of nonnegative real-valued $(N_1 \times N_2 \times \cdots \times N_i)$ arrays, respectively. Vectors, matrices and HO arrays are denoted by bold lower case letters ($\boldsymbol{a}$, $\boldsymbol{b}$, $\cdots$), bold upper case letters ($\boldsymbol{A}$, $\boldsymbol{B}$, $\cdots$) and bold calligraphic letters ($\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{B}}$, $\cdots$), respectively. The (i,j)-th entry of a matrix $\boldsymbol{A}$ is symbolized by $A_{i,j}$. Sometimes the MATLAB® column/row notation is adopted to indicate submatrices of a given matrix or subarrays of a HO array. Also $\boldsymbol{a}_i$ denotes the i-th column vector of matrix $\boldsymbol{A}$. $\boxdot$ denotes the Hadamard product (element-wise product), and $\boldsymbol{A}^{\boxdot 2} = \boldsymbol{A} \boxdot \boldsymbol{A}$. $\odot$ denotes the Khatri-Rao product. $\boldsymbol{A}^{\sharp}$ denotes the pseudo inverse of $\boldsymbol{A}$. The superscripts, $^{-1}$, $^{\mathsf{T}}$ and $^{-\mathsf{T}}$ stand for the inverse, the transpose and the inverse after transpose operators, respectively. The $(N \times N)$ identity matrix is denoted by $\boldsymbol{I}_N$. $\boldsymbol{0}_N$ stands for N-dimensional vectors of zeros. $|a|$ denotes the absolute value of a. $\|\boldsymbol{A}\|_F$ and $\det(\boldsymbol{A})$ stand for the Frobenius norm and determinant of matrix $\boldsymbol{A}$, respectively. $\mathrm{diag}(\boldsymbol{A})$ returns a matrix comprising only the diagonal elements of $\boldsymbol{A}$. $\mathrm{Diag}(\boldsymbol{b})$ is the diagonal matrix whose diagonal elements are given by the vector $\boldsymbol{b}$. $\mathrm{off}(\boldsymbol{A})$ vanishes the diagonal components of the input matrix $\boldsymbol{A}$. $\mathrm{vec}(\boldsymbol{A})$ reshapes a matrix $\boldsymbol{A}$ into a column vector by stacking its columns vertically.

Definitions and problem formulation

Now we introduce some basic definitions in multilinear algebra which are necessary for the problem formulation.

Definition 1 *The outer product* $\boldsymbol{\mathcal{C}} = \boldsymbol{u}^{(1)} \circ \boldsymbol{u}^{(2)} \circ \boldsymbol{u}^{(3)}$ *of three vectors* $\boldsymbol{u}^{(i)} \in \mathbb{R}^{N_i}$ $(1 \leq i \leq 3)$ *is a* 3*-way array of* $\mathbb{R}^{N_1 \times N_2 \times N_3}$ *whose elements are defined by* $\mathcal{C}_{i_1,i_2,i_3} = u^{(1)}_{i_1} u^{(2)}_{i_2} u^{(3)}_{i_3}$.

Definition 2 *Each* 3*-way array* $\boldsymbol{\mathcal{C}}$ *expressed as the outer product of three vectors is a rank-*1 3*-way array.*

More generally, the rank of a 3-way array is defined as follows:

Definition 3 *The rank of an array* $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$*, denoted by* $\mathrm{rk}(\boldsymbol{\mathcal{C}})$*, is the minimal number of rank-*1 *arrays belonging to* $\mathbb{R}^{N_1 \times N_2 \times N_3}$ *that yield* $\boldsymbol{\mathcal{C}}$ *in a linear combination.*

Despite the similarity between the definition of the tensor rank and its matrix counterpart, the rank of a three-way array may exceed its dimensions [4].

Definition 4 *A* 3*-way array slice is a* 2*-dimensional section (fragment) of a* 3*-way array, obtained by fixing one of the three indices [38].*

For example, the k-th frontal slice of a 3-way array $\boldsymbol{\mathcal{C}}$ can be denoted by $\boldsymbol{\mathcal{C}}_{:,:,k}$ using MATLAB notation and sometimes it is also denoted by $\boldsymbol{C}^{(k)}$.

The low-rank INDSCAL model of a 3-way array is defined as follows:

Definition 5 *For a given P, corresponding to the number of rank-1 terms, the INDSCAL model of a 3-way array $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{N\times N\times K}$ can be expressed as:*

$$\boldsymbol{\mathcal{C}} = \sum_{p=1}^{P} \boldsymbol{a}_p \circ \boldsymbol{a}_p \circ \boldsymbol{d}_p + \boldsymbol{\mathcal{V}} \tag{1}$$

where the 3-way array $\boldsymbol{\mathcal{V}}$ represents the model residual.

The notation $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!] + \boldsymbol{\mathcal{V}}$ refers to the INDSCAL decomposition (1) of $\boldsymbol{\mathcal{C}}$ with the associated loading matrices $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_P] \in \mathbb{R}^{N\times P}$ and $\boldsymbol{D} = [\boldsymbol{d}_1, \cdots, \boldsymbol{d}_P] \in \mathbb{R}^{K\times P}$. If and only if the residual $\boldsymbol{\mathcal{V}}$ is a null tensor, we have an exact INDSCAL decomposition.

An exact INDSCAL decomposition is considered to be essentially unique when it is only subject to scale and permutation indeterminacies. It means that an INDSCAL decomposition is insensitive to a scaling of the three vectors $\boldsymbol{a}_p$, $\boldsymbol{a}_p$ and $\boldsymbol{d}_p$ provided that the product of the 3 scale numbers is equal to 1, and an arbitrary permutation of the rank-1 terms. A necessary and sufficient uniqueness condition for the INDSCAL model was established by Afsari [43].

The INDSCAL model can also be described by using the frontal slices of $\boldsymbol{\mathcal{C}}$:

$$\forall\, k \in \{1, 2, \cdots, K\},\ \boldsymbol{C}^{(k)} = \boldsymbol{\mathcal{C}}_{:,:,k} = \boldsymbol{A}\boldsymbol{D}^{(k)}\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{V}^{(k)} \tag{2}$$

where $\boldsymbol{D}^{(k)}$ is a diagonal matrix whose diagonal contains the elements of the k-th row of $\boldsymbol{D}$, and $\boldsymbol{V}^{(k)} = \boldsymbol{\mathcal{V}}_{:,:,k}$.

In this paper we propose to fit the INDSCAL model of 3-way arrays, while imposing nonnegativity constraints on both equal loading matrices $\boldsymbol{A}$. It will be referred to as the semi-nonnegative INDSCAL model, as follows:

Problem 1 *Given $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{N\times N\times K}$ and an integer P, find a semi-nonnegative INDSCAL model of $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!]$, subject to the $(N \times P)$ matrix $\boldsymbol{A}$ having nonnegative components.*

The semi-nonnegative INDSCAL problem is equivalent to the JDC problem subject to the nonnegativity constraint on the joint transformation matrix. In this paper, we mainly focus on the case of square nonnegative joint transformation matrix, for which $N = P$. The case of $N > P$ will be discussed briefly in the next section. Therefore the problem that we tackle in this paper is defined as follows:

Problem 2 *Given a 3-way array $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{N\times N\times K}$ with K symmetric frontal slices $\boldsymbol{C}^{(k)} \in \mathbb{R}^{N\times N}$, find a $(N \times N)$ joint transformation matrix $\boldsymbol{A}$ and K diagonal matrices $\boldsymbol{D}^{(k)}$ of dimension $(N \times N)$ such that:*

$$\forall\, k \in \{1, 2, \cdots, K\},\ \boldsymbol{C}^{(k)} = \boldsymbol{A}\boldsymbol{D}^{(k)}\boldsymbol{A}^{\mathsf{T}} + \boldsymbol{V}^{(k)} \tag{3}$$

by minimizing the residual term $\boldsymbol{V}^{(k)}$ in a least-squares sense, subject to $\boldsymbol{A}$ having nonnegative components.

JDC cost functions

If the residual array $\boldsymbol{\mathcal{V}}$ is a realization of a Gaussian random array, it is logical to fit the INDSCAL model by the following Direct Least Square (DLS) criterion [23,44]:

$$J_{\mathrm{DLS}}(\boldsymbol{A},\boldsymbol{D}) = \sum_{k=1}^{K} \left\| \boldsymbol{C}^{(k)} - \boldsymbol{A}\boldsymbol{D}^{(k)}\boldsymbol{A}^{\mathsf{T}} \right\|_F^2 \tag{4}$$

and to minimize (4) with respect to $\boldsymbol{A}$ and $\boldsymbol{D}$. Note that, in the field of ICA, only the loading matrix $\boldsymbol{A}$ is of interest since it corresponds to the mixing matrix of several latent source signals. The minimization of (4) with respect to $\boldsymbol{D}$, when $\boldsymbol{A}$ is fixed, was given by Yeredor in [23]:

$$\boldsymbol{D}^{(k)} = \mathrm{Diag}\{[(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A}) \boxdot (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})]^{-1}(\boldsymbol{A} \odot \boldsymbol{A})^{\mathsf{T}} \,\mathrm{vec}\,(\boldsymbol{C}^{(k)})\} \tag{5}$$

When $\boldsymbol{A}$ is orthogonal, we can replace $\boldsymbol{D}^{(k)}$ by $\mathrm{Diag}\{(\boldsymbol{A} \odot \boldsymbol{A})^{\mathsf{T}} \,\mathrm{vec}\,(\boldsymbol{C}^{(k)})\}$ in (4). Then the extra parameter $\boldsymbol{D}$ can be eliminated and the minimization of (4) is equivalent to minimizing the following InDirect Least Square (IDLS) criterion [45, 46]:

$$J_{\text{IDLS-O}}(\boldsymbol{A}) = \sum_{k=1}^{K} \left\| \mathrm{off}(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{C}^{(k)}\boldsymbol{A}) \right\|_F^2 \tag{6}$$

In some cases such as in ICA, the orthogonality assumption of $\boldsymbol{A}$ can be satisfied by using a spatial whitening procedure [47]. However, it is known that the whitening procedure may introduce additional errors. Therefore many algorithms propose to relax the orthogonality constraint by introducing the following cost function [25,31]:

$$J_{\mathrm{IDLS}}(\boldsymbol{A}) = \sum_{k=1}^{K} \left\| \mathrm{off}(\boldsymbol{A}^{-1}\boldsymbol{C}^{(k)}\boldsymbol{A}^{-\mathsf{T}}) \right\|_F^2 \tag{7}$$

Frequently the minimization of criterion (7) is performed on a matrix $\boldsymbol{Z} \overset{\text{def}}{=} \boldsymbol{A}^{-1}$ instead of $\boldsymbol{A}$ for simplicity and $\boldsymbol{Z}$ is called the joint diagonalizer. To use this criterion, the matrix $\boldsymbol{A}$ (or $\boldsymbol{Z}$) should be properly constrained in order to avoid the trivial zero solution and/or degenerate solutions [34].

Besides the criterions (4) and (7), Afsari [26] presented a new cost function, which is invariant to column scaling of $\boldsymbol{A}$. Pham proposed an information theoretic criterion [48], which requires each matrix $\boldsymbol{C}^{(k)}$ to be positive definite. Tichavský and Yeredor gave a special weighted least square criterion [49].

Methods

Problem reformulation

Existing semi-nonnegative INDSCAL algorithms are based on the minimization of the cost function (4) [40, 41]. They are able to achieve a better estimation of $\boldsymbol{A}$ than ACDC when the data satisfies the semi-nonnegative INDSCAL model at the cost of a higher computational complexity. We propose to use criterion (7) based on elementary factorizations of $\boldsymbol{A}$ due to the fast convergence property of this kind of

procedures. Generally, it is quite difficult to impose the nonnegativity constraint on $\boldsymbol{A}$, while computing its inverse $\boldsymbol{A}^{-1}$ by minimizing (7). Let us consider the structure of $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!]$ with the following assumptions:

① $\boldsymbol{A} \in \mathbb{R}_{+}^{N \times N}$ is nonsingular;

② $\boldsymbol{D} \in \mathbb{R}^{K \times N}$ does not contain zero entries.

Then each frontal slice of $\boldsymbol{\mathcal{C}}$ is nonsingular and its inverse can be expressed as follows:

$$\left(\boldsymbol{C}^{(k)}\right)^{-1} = \boldsymbol{A}^{-\mathsf{T}} \left(\boldsymbol{D}^{(k)}\right)^{-1} \boldsymbol{A}^{-1} \tag{8}$$

We use $\boldsymbol{C}^{(k,-1)}$ to denote $(\boldsymbol{C}^{(k)})^{-1}$ for simplicity. Equation (8) shows that $\boldsymbol{C}^{(k,-1)}$ also preserves the jointly diagonalizable structure. Furthermore, instead of $\boldsymbol{A}^{-1}$, $\boldsymbol{A}$ serves as the joint diagonalizer. Then $\boldsymbol{A}$ can be estimated by minimizing the following modified criterion based on (7):

$$J(\boldsymbol{A}) = \sum_{k=1}^{K} \left\| \operatorname{off}(\boldsymbol{A}^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} \boldsymbol{A}) \right\|_F^2 \tag{9}$$

By such a manipulation, most algorithms based on criterion (7) can now estimate $\boldsymbol{A}$ directly. However, none of them can guarantee the nonnegativity of $\boldsymbol{A}$. In order to impose the nonnegativity constraint on $\boldsymbol{A}$, we resort to use a square change of variable which was introduced by Chu et al. [50] for NMF, next adopted by Royer et al. for NTF [37] and by Coloigner et al. for semi-nonnegative INDSCAL [40]:

$$\boldsymbol{A} = \boldsymbol{B} \boxdot \boldsymbol{B} = \boldsymbol{B}^{\boxdot 2} \tag{10}$$

where $\boldsymbol{B} \in \mathbb{R}^{N \times N}$. Then problem 2 can be reformulated as follows:

Problem 3 *Given $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!] \in \mathbb{R}^{N \times N \times K}$, find the square nonnegative loading matrix $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$ such that $\boldsymbol{B}$ minimizes the following cost function:*

$$J(\boldsymbol{B}) = \sum_{k=1}^{K} \left\| \operatorname{off}\left((\boldsymbol{B}^{\boxdot 2})^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} \boldsymbol{B}^{\boxdot 2} \right) \right\|_F^2 \tag{11}$$

LU and QR parameterizations of B

In order to minimize (11), one may consider a gradient-like approach. However, the performance of this kind of method is sensitive to the initial guess and to the search step size. In addition, the calculation of gradient of (11) with respect to $\boldsymbol{B}$ is computationally expensive due to the existence of Hadamard product. Other algorithms, using Jacobi-like procedures [25, 26, 31], parameterize $\boldsymbol{A}$ as a product of several special elementary matrices and estimate each elementary matrix successively. We propose to follow such a minimization scheme.

Now let us recall the following definitions and lemmas:

Definition 6 *A unit upper (or lower) triangular matrix is an upper (or lower, respectively) triangular matrix whose main diagonal elements are equal to* 1.

Definition 7 *An elementary upper (or lower) triangular matrix with parameters* $\{i, j, u_{i,j}\}$ *and* $i < j$ *is a unit upper (or lower, respectively) triangular matrix whose non-diagonal elements are zeros except the* (i, j)*-th entry, which is equal to* $u_{i,j}$*.*

$\boldsymbol{U}^{(i,j)}(u_{i,j})$ with $1 \leq i < j \leq N$ denotes an elementary upper triangular matrix:

$$\boldsymbol{U}^{(i,j)}(u_{i,j}) = \begin{pmatrix} \boldsymbol{I}_{i-1} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \dots & 1 & \dots & u_{i,j} & \dots \\ \mathbf{0} & \vdots & \boldsymbol{I}_{j-i-1} & \vdots & \mathbf{0} \\ \dots & 0 & \dots & 1 & \dots \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{I}_{N-j} \end{pmatrix} \tag{12}$$

Similarly, $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$ with $1 \leq j < i \leq N$ corresponds to an elementary lower triangular matrix.

Definition 8 *A Givens rotation matrix with parameters* $\{i, j, \theta_{i,j}\}$ *and* $i < j$ *is equal to an identity matrix except for the* (i, i)*-th,* (j, j)*-th,* (i, j)*-th and* (j, i)*-th entries, which are equal to* $\cos(\theta_{i,j})$*,* $\cos(\theta_{i,j})$*,* $-\sin(\theta_{i,j})$ *and* $\sin(\theta_{i,j})$*, respectively.*

$\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$ with $1 \leq i < j \leq N$ indicates the corresponding Givens rotation matrix:

$$\boldsymbol{Q}^{(i,j)}(\theta_{i,j}) = \begin{pmatrix} \boldsymbol{I}_{i-1} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \dots & \cos(\theta_{i,j}) & \dots & -\sin(\theta_{i,j}) & \dots \\ \mathbf{0} & \vdots & \boldsymbol{I}_{j-i-1} & \vdots & \mathbf{0} \\ \dots & \sin(\theta_{i,j}) & \dots & \cos(\theta_{i,j}) & \dots \\ \mathbf{0} & \vdots & \mathbf{0} & \vdots & \boldsymbol{I}_{N-j} \end{pmatrix} \tag{13}$$

Lemma 1 *Any* $(N \times N)$ *unit lower triangular matrix* $\boldsymbol{L}$ *whose* (i, j)*-th component is* $\ell_{i,j}$ $(i > j)$ *can be factorized as the following product of* $N(N-1)/2$ *elementary lower triangular matrices [51, Chapter 3]:*

$$\boldsymbol{L} = \prod_{j \in \mathscr{J}_1} \prod_{i \in \mathscr{I}_1(j)} \boldsymbol{L}^{(i,j)}(\ell_{i,j}) \tag{14}$$

where the two sets of indices $\mathscr{J}_1$ *and* $\mathscr{I}_1(j)$ *are defined by* $\mathscr{J}_1 = \{1, 2, \dots, N\}$ *and* $\mathscr{I}_1(j) = \{j+1, j+2, \dots, N\}$ *for the sake of convenience. Similarly, any* $(N \times N)$ *unit upper triangular matrix* $\boldsymbol{U}$ *whose* (i, j)*-th component is equal to* $u_{i,j}$ $(i < j)$ *can be factorized as a product of elementary upper triangular matrices as follows:*

$$\boldsymbol{U} = \prod_{i \in \mathscr{I}_2} \prod_{j \in \mathscr{J}_2(i)} \boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{15}$$

where $\mathscr{I}_2$ *and* $\mathscr{J}_2(i)$ *are two sets of indices, defined by* $\mathscr{I}_2 = \{N-1, N-2, \dots, 1\}$ *and* $\mathscr{J}_2(i) = \{N, N-1, \dots, i+1\}$*.*

Lemma 2 *Any $(N \times N)$ orthonormal matrix $\boldsymbol{Q}$ can be factorized as the following product of $N(N-1)/2$ Givens rotation matrices [52, Chapter 14]:*

$$\boldsymbol{Q} = \prod_{i \in \mathscr{I}_2} \prod_{j \in \mathscr{J}_2(i)} \boldsymbol{Q}^{(i,j)}(\theta_{i,j}) \tag{16}$$

where $\mathscr{I}_2$ and $\mathscr{J}_2(i)$ are defined in lemma 1.

For any nonsingular matrix $\boldsymbol{B} \in \mathbb{R}^{N \times N}$, the LU matrix factorization decomposes it as $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{\Pi}$, where $\boldsymbol{L} \in \mathbb{R}^{N \times N}$ is a unit lower triangular matrix, $\boldsymbol{U} \in \mathbb{R}^{N \times N}$ is a unit upper triangular matrix, $\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$ is a diagonal matrix and $\boldsymbol{\Pi} \in \mathbb{R}^{N \times N}$ is a permutation matrix. $\boldsymbol{B}$ also admits the QR matrix factorization as $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{\Lambda}$, where $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ is an orthonormal matrix, $\boldsymbol{R} \in \mathbb{R}^{N \times N}$ is a unit upper triangular matrix, and $\boldsymbol{\Lambda} \in \mathbb{R}^{N \times N}$ is a diagonal matrix. Due to the indeterminacies of the JDC problem, the global minimum of (11), say $\boldsymbol{B}$, can be expressed as $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{U}$ and $\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{R}$ without loss of generality. Moreover, by incorporating lemma 1 and lemma 2, we obtain the two following elementary factorizations of $\boldsymbol{B}$:

$$\boldsymbol{B} = \prod_{j \in \mathscr{J}_1} \prod_{i \in \mathscr{I}_1(j)} \boldsymbol{L}^{(i,j)}(\ell_{i,j}) \prod_{i \in \mathscr{I}_2} \prod_{j \in \mathscr{J}_2(i)} \boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{17}$$

$$\boldsymbol{B} = \prod_{i \in \mathscr{I}_2} \prod_{j \in \mathscr{J}_2(i)} \boldsymbol{Q}^{(i,j)}(\theta_{i,j}) \prod_{i \in \mathscr{I}_2} \prod_{j \in \mathscr{J}_2(i)} \boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{18}$$

As a consequence, the minimization of (11) with respect to $\boldsymbol{B}$ is converted to the estimate of $N(N-1)$ parameters: $\ell_{i,j}$ and $u_{i,j}$ for the LU decomposition (17), or $\theta_{i,j}$ and $u_{i,j}$ for the QR decomposition (18). Instead of simultaneously computing the $N(N-1)$ parameters, we propose two Jacobi-like procedures which perform $N(N-1)$ sequential optimizations. This yields two new algorithms: ***i***) the first algorithm based on (17), named $\text{JD}^{+}_{\text{LU}}$, estimates each $\ell_{i,j}$ and $u_{i,j}$ successively, and ***ii***) the second one based on (18), called $\text{JD}^{+}_{\text{QR}}$, estimates each $\theta_{i,j}$ and $u_{i,j}$ sequentially.

Now, the difficulty is how to estimate four kinds of parameter, namely $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$ and $\boldsymbol{U}^{(i,j)}(u_{i,j})$ for $\text{JD}^{+}_{\text{LU}}$, and $\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$ and $\boldsymbol{U}^{(i,j)}(u_{i,j})$ for $\text{JD}^{+}_{\text{QR}}$. Two points should be noted here: ***i***) $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$ and $\boldsymbol{U}^{(i,j)}(u_{i,j})$ belong to the same category of matrices, therefore they can be estimated by the same algorithmic procedure just with an emphasis on the relation between the i and j indices ($i < j$ for $\boldsymbol{U}^{(i,j)}(u_{i,j})$ and $j < i$ for $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$); ***ii***) for both $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms, the procedure of estimating $\boldsymbol{U}^{(i,j)}(u_{i,j})$ is identical. Consequently, the principal problem is reduced to estimating two kinds of parameters, namely $\boldsymbol{U}^{(i,j)}(u_{i,j})$ and $\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$.

Minimization with respect to the elementary upper triangular matrix $\boldsymbol{U}^{(i,j)}(u_{i,j})$

In this section, we minimize (11) with respect to $\boldsymbol{U}^{(i,j)}(u_{i,j})$ with $1 \leq i < j \leq N$. Let $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ denote the current estimate of $\boldsymbol{A}$ and $\boldsymbol{B}$ before estimating the parameter $u_{i,j}$, respectively. Let $\tilde{\boldsymbol{A}}^{(\text{new})}$ and $\tilde{\boldsymbol{B}}^{(\text{new})}$ stand for $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ updated by $\boldsymbol{U}^{(i,j)}(u_{i,j})$, respectively. Furthermore, the update of $\tilde{\boldsymbol{B}}$ is defined as follows:

$$\tilde{\boldsymbol{B}}^{(\text{new})} = \tilde{\boldsymbol{B}}\boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{19}$$

In order to compute the parameter $u_{i,j}$, a typical way is to minimize the criterion (11) with respect to $u_{i,j}$ by replacing matrix $\tilde{\boldsymbol{B}}$ by $\tilde{\boldsymbol{B}}^{(\text{new})}$. For the sake of convenience, we denote $J(u_{i,j})$ instead of $J(\tilde{\boldsymbol{B}}^{(\text{new})})$. Then $J(u_{i,j})$ can be expressed as follows:

$$J(u_{i,j}) = \sum_{k=1}^{K} \left\| \text{off}\left\{ [(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}]^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} [(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}] \right\} \right\|_F^2 \tag{20}$$

The expression of the Hadamard square of the update $\tilde{\boldsymbol{B}}^{(\text{new})}$ is shown in the following proposition:

Proposition 1 *$\tilde{\boldsymbol{A}}^{(new)} = (\tilde{\boldsymbol{B}}^{(new)})^{\boxdot 2} = (\tilde{\boldsymbol{B}}\boldsymbol{U}^{(i,j)}(u_{i,j}))^{\boxdot 2}$ can be expressed as a function of $u_{i,j}$ as follows:*

$$\tilde{\boldsymbol{A}}^{(new)} = (\tilde{\boldsymbol{B}}^{(new)})^{\boxdot 2} = \tilde{\boldsymbol{B}}^{\boxdot 2}\, \boldsymbol{U}^{(i,j)}(u_{i,j}^2) + 2\, u_{i,j}(\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)\boldsymbol{e}_j^{\mathsf{T}} \tag{21}$$

where $\tilde{\boldsymbol{b}}_i$ and $\tilde{\boldsymbol{b}}_j$ denote the i-th and j-th columns of $\tilde{\boldsymbol{B}}$, respectively, and $\boldsymbol{e}_j$ is the j-th column of the identity matrix $\boldsymbol{I}_N$.

Inserting (21) into the cost function (20), we have:

$$\begin{aligned} J(u_{i,j}) = \sum_{k=1}^{K} \left\| \text{off}\left(\tilde{\boldsymbol{C}}^{(k,\text{new})}\right) \right\|_F^2 = \sum_{k=1}^{K} \Bigg\| \text{off}\Bigg(& \underbrace{\boldsymbol{U}^{(i,j)}(u_{i,j}^2)^{\mathsf{T}} \tilde{\boldsymbol{C}}^{(k)} \boldsymbol{U}^{(i,j)}(u_{i,j}^2)}_{①} \\ & + \underbrace{u_{i,j}\boldsymbol{U}^{(i,j)}(u_{i,j}^2)^{\mathsf{T}} \tilde{\boldsymbol{c}}^{(k,1)} \boldsymbol{e}_j^{\mathsf{T}}}_{②} + \underbrace{u_{i,j}\boldsymbol{e}_j \tilde{\boldsymbol{c}}^{(k,2)} \boldsymbol{U}^{(i,j)}(u_{i,j}^2)}_{③} + \underbrace{u_{i,j}^2 \tilde{c}^{(k,3)} \boldsymbol{e}_j \boldsymbol{e}_j^{\mathsf{T}}}_{④} \Bigg) \Bigg\|_F^2 \end{aligned} \tag{22}$$

where $\tilde{\boldsymbol{C}}^{(k)} = \tilde{\boldsymbol{A}}^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} \tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{c}}^{(k,1)} = 2\, \tilde{\boldsymbol{A}}^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)$, $\tilde{\boldsymbol{c}}^{(k,2)} = 2\, (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)^{\mathsf{T}} \times \boldsymbol{C}^{(k,-1)} \tilde{\boldsymbol{A}}$ and $\tilde{c}^{(k,3)} = 4\, (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)$ are a $(N \times N)$ constant matrix, a $(N \times 1)$ constant column vector, a $(1 \times N)$ constant row vector and a constant scalar, respectively. The term ① in (22) transforms the j-th column and the j-th row of $\tilde{\boldsymbol{C}}^{(k)}$. The term ② in (22) is a zero matrix except its j-th column containing non-zero elements, while the term ③ contains non-zero entries only on its j-th row. And the term ④ is a zero matrix except its (j,j)-th component being non-zero. In addition, $\tilde{\boldsymbol{C}}^{(k,\text{new})} = ① + ② + ③ + ④$ is a $(N \times N)$ symmetric matrix. Hence (22) shows that only the j-th column and j-th row of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ involve the parameter $u_{i,j}$, while the other elements remain constant. Therefore, the minimization of the cost function (20) is equivalent to minimizing the sum of the squares of the j-th columns of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ except their (j,j)-th elements with $k \in \{1, \cdots, K\}$. The required elements of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ can be expressed by the following proposition.

Proposition 2 *The elements of the j-th column except the (j,j)-th entry of $\tilde{\boldsymbol{C}}^{(k,new)}$ is a second degree polynomial function in $u_{i,j}$ as follows, for every value n*

different of j:

$$\tilde{C}_{n,j}^{(k,new)} = \tilde{C}_{n,i}^{(k)} u_{i,j}^2 + \tilde{c}_n^{(k,1)} u_{i,j} + \tilde{C}_{n,j}^{(k)} \tag{23}$$

where $\tilde{C}_{n,i}^{(k)}$ and $\tilde{C}_{n,j}^{(k)}$ are the (n,i)-th and (n,j)-th components of matrix $\tilde{\boldsymbol{C}}^{(k)}$, respectively, and $\tilde{c}_n^{(k,1)}$ is the n-th element of vector $\tilde{\boldsymbol{c}}^{(k,1)}$.

The proof of this proposition is straightforward. Indeed, we can show that the elements of the j-th column except the (j,j)-th entry of the term ① in (22) can be expressed by $\tilde{C}_{n,i}^{(k)} u_{i,j}^2 + \tilde{C}_{n,j}^{(k)}$ with $1 \leq n \leq N$ and $n \neq j$, and those elements of the term ② in (22) are equal to $\tilde{c}_n^{(k,1)} u_{i,j}$ with $1 \leq n \leq N$ and $n \neq j$. The sum of these elements directly leads to (23). The terms ③ and ④ do not need to be considered, since they do not affect the off-diagonal elements in the j-th column. Proposition 2 shows that the minimization of the cost function (20) can be expressed in the following compact matrix form:

$$J(u_{i,j}) = \sum_{k=1}^{K} \left\| \boldsymbol{E}^{(k)} \boldsymbol{u}_{i,j} \right\|_F^2 = \boldsymbol{u}_{i,j}^{\mathsf{T}} \boldsymbol{Q}_E \boldsymbol{u}_{i,j} \tag{24}$$

where $\boldsymbol{Q}_E = \sum_{k=1}^{K} (\boldsymbol{E}^{(k)})^{\mathsf{T}} \boldsymbol{E}^{(k)}$ is a (3×3) symmetric coefficient matrix. $\boldsymbol{E}^{(k)}$ is a $((N-1) \times 3)$ matrix defined as follows: the first column contains the i-th column of $\tilde{\boldsymbol{C}}^{(k)}$ without the j-th element, the second column contains vector $\tilde{\boldsymbol{c}}^{(k,1)}$ without the j-th entry and the third column contains the j-th column of $\tilde{\boldsymbol{C}}^{(k)}$ without the j-th component. $\boldsymbol{u}_{i,j} = [u_{i,j}^2, u_{i,j}, 1]^{\mathsf{T}}$ is a 3-dimensional parameter vector.

Equation (24) shows that $J(u_{i,j})$ is a fourth degree polynomial function. The global minimum $u_{i,j}$ can be obtained by computing the roots of its derivative and selecting the one yielding the smallest value of (24). Once the optimal $u_{i,j}$ is computed, $\tilde{\boldsymbol{B}}^{(\text{new})}$ is updated by (19) and the joint diagonalizer $\tilde{\boldsymbol{A}}^{(\text{new})}$ is updated by computing $(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}$. Then the same procedure is repeated to compute the next $u_{i,j}$ with another (i,j) index.

The minimization of (11) with respect to the elementary lower triangular matrix $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$ with $1 \leq j < i \leq N$ can be computed in the same way. Proposition 2 is also valid for the parameter $\ell_{i,j}$ when $1 \leq j < i \leq N$. The detailed derivation is omitted here. The processing of all the $N(N-1)$ parameters $u_{i,j}$ and $\ell_{i,j}$ is called a LU sweep. In addition, for estimating $\boldsymbol{L}^{(i,j)}(\ell_{i,j})$, the (i,j) index obeys the following order:

$$\begin{aligned} &(2,1),(3,1),\ldots,(N,1),(3,2),(4,2),\ldots,(N,2),\ldots, \\ &\qquad\qquad (N-1,N-2),(N,N-2),(N,N-1) \end{aligned} \tag{25}$$

Regarding $\boldsymbol{U}^{(i,j)}(u_{i,j})$, the (i,j) index varies according to the following sequence:

$$\begin{aligned} &(N-1,N),(N-2,N),(N-2,N-1),\ldots, \\ &\qquad (2,N),(2,N-1),\ldots,(2,3),(1,N),(1,N-1),\ldots,(1,2) \end{aligned} \tag{26}$$

The proposed $\text{JD}_{\text{LU}}^{+}$ algorithm is comprised of several LU sweeps.

Minimization with respect to the Given rotation matrix $\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$

Now we minimize (11) with respect to $\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$ with $1 \leq i < j \leq N$. By abuse of notation, in this section we continue to use $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ to denote the current estimate of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively, before estimating the parameter $\theta_{i,j}$. Also let $\tilde{\boldsymbol{A}}^{(\text{new})}$ and $\tilde{\boldsymbol{B}}^{(\text{new})}$ stand for $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ updated by $\boldsymbol{Q}^{(i,j)}(\theta_{i,j})$, respectively. The update of $\tilde{\boldsymbol{B}}$ is defined as follows:

$$\tilde{\boldsymbol{B}}^{(\text{new})} = \tilde{\boldsymbol{B}}\boldsymbol{Q}^{(i,j)}(\theta_{i,j}) \tag{27}$$

Similarly, for computing the parameter $\theta_{i,j}$, we can minimize the criterion (11) with respect to $\theta_{i,j}$ by replacing matrix $\tilde{\boldsymbol{B}}$ by $\tilde{\boldsymbol{B}}^{(\text{new})}$. We denote $J(\theta_{i,j})$ instead of $J(\tilde{\boldsymbol{B}}^{(\text{new})})$ for convenience purpose. Then $J(\theta_{i,j})$ can be expressed as follows:

$$J(\theta_{i,j}) = \sum_{k=1}^{K} \left\| \text{off}\left\{ [(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}]^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} [(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}] \right\} \right\|_F^2 \tag{28}$$

The Hadamard square of the update $\tilde{\boldsymbol{B}}^{(\text{new})}$ now can be rewritten as shown in the following proposition.

Proposition 3 *$\tilde{\boldsymbol{A}}^{(new)} = (\tilde{\boldsymbol{B}}^{(new)})^{\boxdot 2} = (\tilde{\boldsymbol{B}}\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2}$ can be written as a function of $\theta_{i,j}$ as follows:*

$$\tilde{\boldsymbol{A}}^{(new)} = (\tilde{\boldsymbol{B}}^{(new)})^{\boxdot 2} = \tilde{\boldsymbol{B}}^{\boxdot 2} (\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2} + \sin(2\theta_{i,j})(\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)(\boldsymbol{e}_i^{\mathsf{T}} - \boldsymbol{e}_j^{\mathsf{T}}) \tag{29}$$

where $\tilde{\boldsymbol{b}}_i$ and $\tilde{\boldsymbol{b}}_j$ denote the i-th and j-th columns of $\tilde{\boldsymbol{B}}$, respectively, and $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ are the i-th and j-th columns of the identity matrix $\boldsymbol{I}_N$, respectively.

Inserting (29) into the cost function (28), we obtain:

$$\begin{aligned} J(\theta_{i,j}) = \sum_{k=1}^{K} \left\| \text{off}\left(\tilde{\boldsymbol{C}}^{(k,\text{new})}\right) \right\|_F^2 = \sum_{k=1}^{K} \Bigg\| \text{off} \Bigg(& \underbrace{[(\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2}]^{\mathsf{T}} \tilde{\boldsymbol{C}}^{(k)} (\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2}}_{①} \\ & + \underbrace{\sin(2\theta_{i,j})[(\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2}]^{\mathsf{T}} \tilde{\boldsymbol{c}}^{(k,1)} (\boldsymbol{e}_i^{\mathsf{T}} - \boldsymbol{e}_j^{\mathsf{T}})}_{②} \\ & + \underbrace{\sin(2\theta_{i,j})(\boldsymbol{e}_i - \boldsymbol{e}_j)\tilde{\boldsymbol{c}}^{(k,2)} (\boldsymbol{Q}^{(i,j)}(\theta_{i,j}))^{\boxdot 2}}_{③} + \underbrace{\sin^2(2\theta_{i,j})\tilde{c}^{(k,3)} (\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i^{\mathsf{T}} - \boldsymbol{e}_j^{\mathsf{T}})}_{④} \Bigg) \Bigg\|_F^2 \end{aligned} \tag{30}$$

where $\tilde{\boldsymbol{C}}^{(k)} = \tilde{\boldsymbol{A}}^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} \tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{c}}^{(k,1)} = \tilde{\boldsymbol{A}}^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)$, $\tilde{\boldsymbol{c}}^{(k,2)} = (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} \tilde{\boldsymbol{A}}$ and $\tilde{c}^{(k,3)} = (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)^{\mathsf{T}} \boldsymbol{C}^{(k,-1)} (\tilde{\boldsymbol{b}}_i \boxdot \tilde{\boldsymbol{b}}_j)$ are a $(N \times N)$ constant matrix, a $(N \times 1)$ constant column vector, a $(1 \times N)$ constant row vector and a constant scalar, respectively. The term ① in (30) transforms the i-th and j-th columns and the i-th and j-th rows of $\tilde{\boldsymbol{C}}^{(k)}$. The term ② in (30) is a zero matrix except its i-th and

j-th columns containing non-zero elements, while the term ③ contains non-zero entries only on its i-th and j-th rows. And the term ④ is a zero matrix except its (i,i)-th, (j,j)-th, (i,j)-th and (j,i)-th components being non-zero. $\tilde{\boldsymbol{C}}^{(k,\text{new})} = ① + ② + ③ + ④$ is a $(N \times N)$ symmetric matrix. Hence (30) shows that only the i-th and j-th columns and the i-th and j-th rows of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ involve the parameter $\theta_{i,j}$, while the other components remain constant. It is noteworthy that the (i,j)-th and (j,i)-th components are twice affected by the transformation. Considering the symmetry of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$, we propose to minimize the sum of the squares of the (i,j)-th entries of the K matrices $\tilde{\boldsymbol{C}}^{(k,\text{new})}$, instead of minimizing all the off-diagonal entries. Although minimizing this quantity is not equivalent to minimizing the global cost function (28), such a simplified minimization scheme is commonly adopted in many algorithms, such as [20,31]. We denote this local minimization by $\tilde{J}(\theta_{i,j})$. The (i,j)-th component of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ is expressed in the following proposition.

Proposition 4 *The (i,j)-th entry of $\tilde{\boldsymbol{C}}^{(k,new)}$ can be expressed as a function of $\theta_{i,j}$ as follows:*

$$\begin{aligned}
\tilde{C}_{i,j}^{(k,new)} &= -\sin^2(2\theta_{i,j})\tilde{c}^{(k,3)} \\
&+ \sin^2(\theta_{i,j})(\tilde{C}_{i,i}^{(k)}\cos^2(\theta_{i,j}) + \tilde{C}_{j,i}^{(k)}\sin^2(\theta_{i,j})) \\
&+ \cos^2(\theta_{i,j})(\tilde{C}_{i,j}^{(k)}\cos^2(\theta_{i,j}) + \tilde{C}_{j,j}^{(k)}\sin^2(\theta_{i,j})) \\
&+ \sin(2\theta_{i,j})(\tilde{c}_i^{(k,1)}\cos^2(\theta_{i,j}) + \tilde{c}_j^{(k,1)}\sin^2(\theta_{i,j})) \\
&- \sin(2\theta_{i,j})(\tilde{c}_j^{(k,2)}\cos^2(\theta_{i,j}) + \tilde{c}_i^{(k,2)}\sin^2(\theta_{i,j}))
\end{aligned} \tag{31}$$

where $\tilde{C}_{i,i}^{(k)}$, $\tilde{C}_{j,j}^{(k)}$, $\tilde{C}_{i,j}^{(k)}$ and $\tilde{C}_{j,i}^{(k)}$ are the (i,i)-th, (j,j)-th, (i,j)-th and (j,i)-th components of matrix $\tilde{\boldsymbol{C}}^{(k)}$, respectively. $\tilde{c}_i^{(k,q)}$ and $\tilde{c}_j^{(k,q)}$ are the i-th and j-th elements of vector $\tilde{\boldsymbol{c}}^{(k,q)}$ with $q \in \{1,2\}$, respectively.

It is straightforward to show that the (i,j)-th entry of the term ① in (30) can be expressed by $\sin^2(\theta_{i,j})\cos^2(\theta_{i,j})(\tilde{C}_{i,i}^{(k)} + \tilde{C}_{j,j}^{(k)}) + \sin^4(\theta_{i,j})C_{j,i}^{(k)} + \cos^4(\theta_{i,j})\tilde{C}_{i,j}^{(k)}$, the (i,j)-th element of the term ② is $\sin(2\theta_{i,j})(\cos^2(\theta_{i,j})\tilde{c}_i^{(k,1)} + \sin^2(\theta_{i,j})\tilde{c}_j^{(k,1)})$, the (i,j)-th component of the term ③ is equal to $-\sin(2\theta_{i,j})(\sin^2(\theta_{i,j})\tilde{c}_i^{(k,2)} + \cos^2(\theta_{i,j})\tilde{c}_j^{(k,2)})$, and that of the term ④ is $-\sin^2(2\theta_{i,j})\tilde{c}^{(k,3)}$. Then proposition 4 can be proved.

In order to simplify the notation of (31), we resort to the Weierstrass change of variable: $t_{i,j} = \tan(\theta_{i,j})$. Then we obtain:

$$\sin(2\theta_{i,j}) = \frac{2t_{i,j}}{1+t_{i,j}^2},\ \cos(2\theta_{i,j}) = \frac{1-t_{i,j}^2}{1+t_{i,j}^2},\ \sin^2(\theta_{i,j}) = \frac{t_{i,j}^2}{1+t_{i,j}^2},\ \cos^2(\theta_{i,j}) = \frac{1}{1+t_{i,j}^2} \tag{32}$$

By substituting (32) into (31), we obtain an alternative expression of the (i,j)-th entry of $\tilde{\boldsymbol{C}}^{(k,\text{new})}$ which is described in the following proposition. Then the minimization of $\tilde{J}(\theta_{i,j})$ transforms to $\tilde{J}(t_{i,j})$.

Proposition 5 *the (i,j)-th entry of $\tilde{\boldsymbol{C}}^{(k,new)}$ can be expressed by a rational function of $t_{i,j}$ as follows:*

$$\tilde{C}_{i,j}^{(k,new)} = \frac{f_4^{(k)} t_{i,j}^4 + f_3^{(k)} t_{i,j}^3 + f_2^{(k)} t_{i,j}^2 + f_1^{(k)} t_{i,j} + f_0^{(k)}}{(1+t_{i,j}^2)^2} \tag{33}$$

where $f_4^{(k)} = \tilde{C}_{j,i}^{(k)}$, $f_3^{(k)} = -2\tilde{c}_i^{(k,1)}$, $f_2^{(k)} = \tilde{C}_{i,i}^{(k)} + \tilde{C}_{j,j}^{(k)} + 2\tilde{c}_j^{(k,2)} - 4\tilde{c}^{(k,3)}$, $f_1^{(k)} = 2\tilde{c}_i^{(k,2)} - \tilde{c}_j^{(k,1)}$ and $f_0^{(k)} = \tilde{C}_{j,j}^{(k)}$.

Equation (33) easily shows that the sum of the squares of the (i,j)-th entries of the K matrices $\tilde{\boldsymbol{C}}^{(k,\text{new})}$, is a rational function in $t_{i,j}$, namely $\tilde{J}(t_{i,j})$, where the degrees of the numerator and the denominator are 8 and 8, respectively. $\tilde{J}(t_{i,j})$ can be expressed in the following compact matrix form:

$$\tilde{J}(t_{i,j}) = \sum_{k=1}^{K} \left\| (\boldsymbol{f}^{(k)})^{\mathsf{T}} \boldsymbol{\tau}_{i,j} \right\|_F^2 = \boldsymbol{\tau}_{i,j}^{\mathsf{T}} \boldsymbol{Q}_F \boldsymbol{\tau}_{i,j} \tag{34}$$

where $\boldsymbol{Q}_F = \sum_{k=1}^{K} \boldsymbol{f}^{(k)} (\boldsymbol{f}^{(k)})^{\mathsf{T}}$ is a (5×5) symmetric coefficient matrix, $\boldsymbol{f}^{(k)} = [f_4^{(k)}, f_3^{(k)}, f_2^{(k)}, f_1^{(k)}, f_0^{(k)}]^{\mathsf{T}}$ is a 5-dimensional vector, and $\boldsymbol{\tau}_{i,j}$ is a 5-dimensional parameter vector defined as follows,

$$\boldsymbol{\tau}_{i,j} = \frac{1}{(1+t_{i,j}^2)^2} \left[t_{i,j}^4,\, t_{i,j}^3,\, t_{i,j}^2,\, t_{i,j},\, 1 \right]^{\mathsf{T}} \tag{35}$$

The global minimum $t_{i,j}$ can be obtained by computing the roots of its derivative and selecting the one yielding the smallest value of $\tilde{J}(t_{i,j})$. Once $t_{i,j}$ is obtained, $\theta_{i,j}$ can be computed from the inverse tangent function $\theta_{i,j} = \arctan(t_{i,j})$. It is noteworthy that the found $\theta_{i,j}$ cannot guarantee to decrease the actual cost function (28). If $\theta_{i,j}$ leads to an increase of (28), we reset $\theta_{i,j} = 0$. Otherwise, $\tilde{\boldsymbol{B}}^{(\text{new})}$ is updated as described in (27) and the joint diagonalizer $\tilde{\boldsymbol{A}}^{(\text{new})}$ is updated by computing $(\tilde{\boldsymbol{B}}^{(\text{new})})^{\boxdot 2}$. The same procedure will be repeated to compute $\theta_{i,j}$ with the next (i,j) index. The order of the (i,j) indices is defined in equation (26). The processing of all the $N(N-1)/2$ parameters $\theta_{i,j}$ and also the other $N(N-1)/2$ parameters $u_{i,j}$ is called a QR sweep. Several QR sweeps yield the proposed $\text{JD}_{\text{QR}}^{+}$ algorithm.

Both of the $\text{JD}_{\text{LU}}^{+}$ and $\text{JD}_{\text{QR}}^{+}$ algorithms can be stopped when the value of cost function (11) or its relative change between two successive sweeps fall below a fixed small positive threshold. Such a stopping criterion is guaranteed to be met since the cost function is non-increasing in each Jacobi-like sweep.

Practical issues

In practice, we observe that if each frontal slice of the 3-way array $\boldsymbol{\mathcal{C}}$ is almost exactly jointly diagonalizable due to a high Signal to Noise Ratio (SNR), the classical non-constrained JDC methods can also give a nonnegative $\boldsymbol{A}$ with high probability. In this situation the explicit nonnegativity constraint could be unnecessary and

could increase the computational burden. Therefore, we propose to relax the nonnegativity constraint by directly decomposing $\boldsymbol{A}$ into elementary LU and QR forms, respectively, instead of using the decompositions of $\boldsymbol{B}$ as follows:

$$\boldsymbol{A} = \prod_{j\in\mathscr{J}_1}\prod_{i\in\mathscr{I}_1(j)} \boldsymbol{L}^{(i,j)}(\ell_{i,j}) \prod_{i\in\mathscr{I}_2}\prod_{j\in\mathscr{J}_2(i)} \boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{36}$$

$$\boldsymbol{A} = \prod_{i\in\mathscr{I}_2}\prod_{j\in\mathscr{J}_2(i)} \boldsymbol{Q}^{(i,j)}(\theta_{i,j}) \prod_{i\in\mathscr{I}_2}\prod_{j\in\mathscr{J}_2(i)} \boldsymbol{U}^{(i,j)}(u_{i,j}) \tag{37}$$

where the index sets $\mathscr{I}_1(j)$, $\mathscr{J}_1$, $\mathscr{I}_2$ and $\mathscr{J}_2(i)$ are defined in lemma 1. By inserting (36) and (37) into the cost function (9), the ways of estimating the two sets of parameters $\{\ell_{i,j}, u_{i,j}\}$, and $\{\theta_{i,j}, u_{i,j}\}$ are identical to those of Afsari's LUJ1D and QRJ1D methods [26], respectively. Therefore, in practice, in order to give an automatically SNR-adaptive method, for $\mathrm{JD}^{+}_{\mathrm{LU}}$, in each Jacobi-like iteration, we suggest to compute $u_{i,j}$ by LUJ1D first. If all the elements in the j-th column of $\tilde{\boldsymbol{A}}\boldsymbol{U}^{(i,j)}(u_{i,j})$ have the same sign ε, the update $\tilde{\boldsymbol{A}}^{(\mathrm{new})} = \varepsilon\tilde{\boldsymbol{A}}\boldsymbol{U}^{(i,j)}(u_{i,j})$ is adopted. Otherwise, $u_{i,j}$ is computed by minimizing (20) and $\tilde{\boldsymbol{A}}^{(\mathrm{new})}$ is updated by computing (21). Each $\ell_{i,j}$ is computed similarly. Furthermore, the proposed $\mathrm{JD}^{+}_{\mathrm{QR}}$ and QRJ1D are combined in the same manner.

Afsari reported in [26] that if the rows of matrices $\tilde{\boldsymbol{C}}^{(k)}$ $(k \in \{1, \cdots, K\})$ are not balanced in their norms, the computation of the parameter could be inaccurate. In order to cope with this effect, we apply Afsari's row balancing scheme every few sweeps. Such a scheme updates each $\tilde{\boldsymbol{C}}^{(k,\mathrm{new}))}$ by $\tilde{\boldsymbol{C}}^{(k,\mathrm{new})} = \boldsymbol{\Lambda}\tilde{\boldsymbol{C}}^{(k)}\boldsymbol{\Lambda}$ and $\tilde{\boldsymbol{A}}^{(\mathrm{new})}$ by $\tilde{\boldsymbol{A}}^{(\mathrm{new})} = \tilde{\boldsymbol{A}}\boldsymbol{\Lambda}$ using a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}_{+}^{N\times N}$, whose diagonal elements are defined as follows:

$$\Lambda_{n,n} = \frac{1}{\sqrt{\sum_{k=1}^{K} \|\tilde{\boldsymbol{C}}^{(k)}_{n,:}\|^2}}, \quad n \in \{1, 2, \cdots, N\} \tag{38}$$

where $\tilde{\boldsymbol{C}}^{(k)}_{n,:}$ denotes the n-th row of $\tilde{\boldsymbol{C}}^{(k)}$.

In ICA, when a non-square matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{N\times P}$ with $N > P$ is encountered, the invertibility assumption of the frontal slices $\boldsymbol{C}^{(k)}$ does not hold. In this situation, we can compress $\boldsymbol{A}$ by means of a nonnegative matrix $\boldsymbol{W}_{+} \in \mathbb{R}_{+}^{P\times N}$ such that the resulting matrix $\bar{\boldsymbol{A}} = \boldsymbol{W}_{+}\boldsymbol{A}$ is a nonnegative square matrix. Then the $\mathrm{JD}^{+}_{\mathrm{LU}}$ and $\mathrm{JD}^{+}_{\mathrm{QR}}$ algorithms can be used to compute the compressed loading matrix $\bar{\boldsymbol{A}}$. $\boldsymbol{W}_{+}$ can be computed by using the NonNegative COMPression algorithm (NN-COMP) that we proposed in [53]. More precisely, given a realization of an observation vector, we obtain the square root of the covariance matrix, denoted by $\boldsymbol{\Upsilon} \in \mathbb{R}^{N\times P}$. The classical prewhitening matrix is computed by $\boldsymbol{W} = \boldsymbol{\Upsilon}^{\sharp} \in \mathbb{R}^{P\times N}$ where $\sharp$ denotes the pseudo inverse operator [47]. Then the NN-COMP algorithm computes a linear transformation matrix $\boldsymbol{\Psi} \in \mathbb{R}^{P\times P}$ such that $\boldsymbol{W}_{+} = \boldsymbol{\Psi}\boldsymbol{W}$ has nonnegative components. Once $\bar{\boldsymbol{A}}$ is estimated, the original matrix $\boldsymbol{A}$ is obtained as follows:

$$\boldsymbol{A} = \boldsymbol{W}^{\sharp}\boldsymbol{\Psi}^{-1}\bar{\boldsymbol{A}} = \boldsymbol{\Upsilon}\boldsymbol{\Psi}^{-1}\bar{\boldsymbol{A}} \tag{39}$$

It should be noted that generally $\boldsymbol{A}$ does not need to be computed in such an ICA problem, since the sources can be estimated directly by means of $\bar{\boldsymbol{A}}$.

Numerical complexity

The numerical complexities of $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ are analyzed in terms of the number of floating point operations (flops). A flop is defined as a multiplication followed by an addition. In practice, only the number of multiplications, required to identify the loading matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{N\times N}$ from a 3-way array $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{N\times N\times K}$, is considered; which does not affect the order of magnitude of the numerical complexity.

For both algorithms, the inverses $\boldsymbol{C}^{(k,-1)}$ ($k \in \{1,\cdots,K\}$) of the frontal slices of $\boldsymbol{\mathcal{C}}$ cost N^3K flops, the initialization of $\tilde{\boldsymbol{C}}^{(k)}_{\text{ini}} = \tilde{\boldsymbol{A}}^{\mathsf{T}}_{\text{ini}}\boldsymbol{C}^{(k,-1)}\tilde{\boldsymbol{A}}_{\text{ini}}$ requires $2N^3K$ flops, and at each sweep, the calculation of parameters $u_{i,j}$ needs $N(N-1)(5N^2+12N-8)K/2$ flops. In addition, in the case of the $\text{JD}^{+}_{\text{LU}}$ algorithm, the calculation cost of $\tilde{\boldsymbol{A}}^{(\text{new})}$, $\tilde{\boldsymbol{B}}^{(\text{new})}$ and $\tilde{\boldsymbol{C}}^{(k,\text{new})}$, with $k \in \{1,\cdots,K\}$, is $N(N-1)(4N+(4N+1)K)$ flops, and the numerical complexity of computing the parameters $\ell_{i,j}$ is equal to that of $u_{i,j}$. Regarding the $\text{JD}^{+}_{\text{QR}}$ algorithm, for each sweep, the complexity of calculating the parameters $\theta_{i,j}$ is equal to $N(N-1)(5N^2+3N+29)K/2$ flops, and the estimation of $\tilde{\boldsymbol{A}}^{(\text{new})}$, $\tilde{\boldsymbol{B}}^{(\text{new})}$ and $\tilde{\boldsymbol{C}}^{(k,\text{new})}$, with $k \in \{1,\cdots,K\}$, costs $N(N-1)(5N+(12N+20)K/2)$ flops. In practice, the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ techniques are combined with LUJ1D and QRJ1D [26], respectively, leading to the magnitude of global numerical complexities of $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ being between $\mathcal{O}(N^3K)$ and $\mathcal{O}(N^4K)$. A recent nonnegative JDC method called $\text{ACDC}^{+}_{\text{LU}}$ [41] is also based on a square change of variable and LU matrix factorization. It minimizes the cost function (4) with respect to $\boldsymbol{A}$ and $\boldsymbol{D}$ alternately, leading to a higher numerical complexity. By means of the reformulation of the cost function, the proposed methods avoid the estimation of $\boldsymbol{D}$, therefore achieve a lower complexity compared to $\text{ACDC}^{+}_{\text{LU}}$. The explicit expressions of the overall complexity of $\text{JD}^{+}_{\text{LU}}$, $\text{JD}^{+}_{\text{QR}}$ and $\text{ACDC}^{+}_{\text{LU}}$ [41], as well as those of four classical JDC algorithms, namely ACDC [23], FFDIAG [25], LUJ1D [26] and QRJ1D [26], are listed in table 1. One can notice that numerical complexities of the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ methods are at most one order of magnitude higher than those of the four JDC algorithms, and still lower than that of $\text{ACDC}^{+}_{\text{LU}}$. Moreover, $\text{JD}^{+}_{\text{LU}}$ is less computationally expensive than $\text{JD}^{+}_{\text{QR}}$.

Simulation results

This section is twofold. In the first part, the performance of the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms is evaluated with simulated semi-nonnegative semi-symmetric 3-way arrays $\boldsymbol{\mathcal{C}}$. Several experiments are designed to study the convergence property, the influence of SNR, the impact of the third dimension K of $\boldsymbol{\mathcal{C}}$, the effect of the coherence of the loading matrix $\boldsymbol{D}$ and the influence of the condition number of the diagonal matrices $\boldsymbol{D}^{(k)}$. We also evaluate the proposed methods for estimating a non-square matrix $\boldsymbol{A}$. The proposed algorithms are compared with four classical nonorthogonal JDC methods, namely ACDC [23], FFDIAG [25], LUJ1D [26], QRJ1D [26], and the nonnegative JDC method $\text{ACDC}^{+}_{\text{LU}}$ [41]. In the second part, the source separation ability of the proposed algorithms is studied through a BSS application. In this context, the $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ are used to jointly diagonalize several matrix slices of the fourth order cumulant array [40] of the observations and compared with several classical ICA [47, 54, 55] and NMF [56] methods.

Simulated semi-nonnegative INDSCAL model

The synthetic semi-nonnegative semi-symmetric 3-way array $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!] \in \mathbb{R}^{N\times N\times K}$ of rank N is generated randomly according to the semi-nonnegative INDSCAL model (3). When used without further specification, all the algorithms are manipulated under the following conditions:

i) Model generation: the loading matrix $\boldsymbol{A} \in \mathbb{R}_+^{N\times N}$ is randomly drawn from a uniform distribution on the interval $[0, 1]$. The loading matrix $\boldsymbol{D} \in \mathbb{R}^{K\times N}$ is drawn from a Gaussian distribution with a mean of 1 and a deviation of 0.5. The pure array $\boldsymbol{\mathcal{C}}$ is perturbed by a residual INDSCAL noise array $\boldsymbol{\mathcal{V}}$. The loading matrices of $\boldsymbol{\mathcal{V}}$ are drawn from a zero-mean unit-variance Gaussian distribution. The resulting noisy 3-way array can be written as follows:

$$\boldsymbol{\mathcal{C}}_N = \frac{\boldsymbol{\mathcal{C}}}{\|\boldsymbol{\mathcal{C}}\|_F} + \sigma_N \frac{\boldsymbol{\mathcal{V}}}{\|\boldsymbol{\mathcal{V}}\|_F} \tag{40}$$

where σ_N is a scalar controlling the noise level. Then the SNR is defined by $\mathrm{SNR} = -20\log_{10}(\sigma_N)$.

ii) Initialization: in each Monte Carlo trial, all the algorithms are initialized by a same random matrix whose components obey the uniform distribution over $[0, 1]$.

iii) Afsari's row balancing scheme: the LUJ1D, QRJ1D, $\mathrm{JD}_{\mathrm{LU}}^{+}$ and $\mathrm{JD}_{\mathrm{QR}}^{+}$ algorithms perform the row balancing scheme once per each run of five sweeps.

iv) Stopping criterion: all the algorithms stop either when the relative error of the corresponding criterion between two successive sweeps is less than 10^{-5} or when the number of sweeps exceeds 200. A sweep of ACDC includes a full AC phase and a DC phase.

v) Performance measurement: the performance is measured by means of the error between the true loading matrix $\boldsymbol{A}$ and the estimate $\tilde{\boldsymbol{A}}$, the numerical complexity, and the CPU time. We define the following scale-invariant and permutation-invariant distance [40]:

$$\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}}) = \frac{1}{N}\sum_{n=1}^{N} \min_{(n,n')\in I_n^2} d(\boldsymbol{a}_n, \tilde{\boldsymbol{a}}_{n'}) \tag{41}$$

where $\boldsymbol{a}_n$ and $\tilde{\boldsymbol{a}}_{n'}$ are the n-th column of $\boldsymbol{A}$ and the n'-th column of $\tilde{\boldsymbol{A}}$, respectively. I_n^2 is defined recursively by $I_1^2 = \{1, \cdots, N\} \times \{1, \cdots, N\}$, and $I_{n+1}^2 = I_n^2 - J_n^2$ where $J_n^2 = \mathrm{argmin}_{(n,n')\in I_n^2} d(\boldsymbol{a}_n, \tilde{\boldsymbol{a}}_{n'})$. In addition, $d(\boldsymbol{a}_n, \tilde{\boldsymbol{a}}_{n'})$ is defined as the pseudo-distance between two vectors [13]:

$$d(\boldsymbol{a}_n, \tilde{\boldsymbol{a}}_{n'}) = 1 - \frac{\|\boldsymbol{a}_n^{\mathsf{T}}\tilde{\boldsymbol{a}}_{n'}\|^2}{\|\boldsymbol{a}_n\|^2\|\tilde{\boldsymbol{a}}_{n'}\|^2} \tag{42}$$

The criterion (41) is an upper bound of the optimal permutation-invariant criterion. It avoids the burdensome computation of all the permutations. A small value of (41) means a good performance in the sense that $\tilde{\boldsymbol{A}}$ is close to $\boldsymbol{A}$.

vi) Test environment: the simulations are carried out in Matlab v7.14 on Mac OS X and run on Intel Quad-Core CPU 2.8 GHz with 32 GB memory. Moreover, we repeat all the experiments with 500 Monte Carlo trials.

Convergence

In this experiment, the convergences of the $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms are compared to those of ACDC, FFDIAG, LUJ1D, QRJ1D and $\text{ACDC}^{+}_{\text{LU}}$. The dimensions of the 3-way array $\boldsymbol{\mathcal{C}}_N \in \mathbb{R}^{N\times N\times K}$ are set to $N = 5$ and $K = 15$. The performance is assessed under three SNR conditions: SNR $= -5$, 10 and 25 dB, respectively. Figure 1 shows the convergence curves measured in terms of the cost function as a function of sweeps. It shows that FFDIAG, LUJ1D and QRJ1D exhibit fast convergence behavior. They converge in less than 20 sweeps. $\text{ACDC}^{+}_{\text{LU}}$ decreases the cost function (4) quasi-linearly. ACDC and $\text{ACDC}^{+}_{\text{LU}}$ do not converge in a maximum of 200 sweeps. The proposed $\text{JD}^{+}_{\text{LU}}$ algorithm converges in about 100 sweeps when SNR = 25 dB and SNR = 10 dB, and it converges in about 40 sweeps when SNR $= -5$ dB. Regarding $\text{JD}^{+}_{\text{QR}}$, it reduces the cost function (11) to the values relatively higher than those achieved by $\text{JD}^{+}_{\text{LU}}$, and converges in about 50 sweeps whatever the SNR is. It seems that FFDIAG, LUJ1D and QRJ1D achieve the fastest convergence rate. It should be noted that while an algorithm may converge to a point in which the value of the cost function is close to zero, such a point could be a local minimum far from the desire matrix $\boldsymbol{A}$ as shown in figure 2. The top picture in figure 2 shows the convergence curves measured in terms of the estimating error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ as a function of sweeps when SNR = 25 dB. It shows that the solutions of FFDIAG, LUJ1D and QRJ1D are still far from optimum. ACDC and $\text{ACDC}^{+}_{\text{LU}}$ give better estimations of $\boldsymbol{A}$ than the previous three methods. The best results are achieved by the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ methods. The middle picture in figure 2 displays the convergence curves when SNR = 10 dB. It can be observed that ACDC converges to a local minimum which is not the global one, and that the performance of the proposed methods are still better than those of the five other algorithms. For a low SNR $= -5$ dB, as shown in the bottom picture in figure 2, both the methods based on alternating optimization, namely ACDC and $\text{ACDC}^{+}_{\text{LU}}$, converge to local minima which are less desirable. The proposed algorithms are always able to converge to better results than the classical methods. The average numerical complexities and CPU time of all the algorithms over Monte Carlo trials are shown in table 2. It is observed that FFDIAG, LUJ1D and QRJ1D require a small amount of calculations, whereas $\text{ACDC}^{+}_{\text{LU}}$ requires a large amount of calculations. The proposed $\text{JD}^{+}_{\text{LU}}$ just costs a bit more flops and CPU time than ACDC, but it is still much more efficient. Concerning the $\text{JD}^{+}_{\text{QR}}$ algorithm, it is more costly than $\text{JD}^{+}_{\text{LU}}$, with a comparable performance. We can then conclude that $\text{JD}^{+}_{\text{LU}}$ offers the best performance/complexity compromise in these experiments.

Effect of SNR

In this section, we study the behaviors of the seven algorithms as a function of SNR. The dimensions of the 3-way array $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$ and $K = 15$. We repeat the experiments with SNR ranging from -30 dB to 50 dB with a step of 2 dB. The top picture in figure 3 depicts the average curves of $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ of the seven algorithms as a

function of SNR. The obtained results show that the performance of all the methods increases as SNR grows. For the unconstrained methods, generally ACDC performs better than FFDIAG, LUJ1D and QRJ1D. The nonnegativity constraint obviously helps $\text{ACDC}^{+}_{\text{LU}}$, $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ to improve the results for lower SNR values. The performance of ACDC and $\text{ACDC}^{+}_{\text{LU}}$ remains stable for higher SNR values due to the small number of available sweeps and the lack of good initializations. Generally the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms outperform the others when SNR is between -20 dB and 30 dB and perform similar to FFDIAG, LUJ1D and QRJ1D when SNR is above 45 dB. The average numerical complexity and CPU time at each SNR level of all the methods in this experiment are shown in the bottom of figure 3. It shows that the proposed methods achieve better estimations of $\boldsymbol{A}$ and cost less flops and CPU time than $\text{ACDC}^{+}_{\text{LU}}$. The $\text{JD}^{+}_{\text{LU}}$ gives the best performance/complexity trade-off for all the considered SNR values.

Effect of dimension K

In ICA, the third dimension K of the 3-way array $\boldsymbol{\mathcal{C}}_N \in \mathbb{R}^{N\times N\times K}$ corresponds to the number of covariance matrices at different lags, or the number of matrix slices derived from a cumulant array. In this section, we study the influence of K on the performance of the seven algorithms. The first and second dimensions of $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$. The SNR value is fixed to 10 dB. We repeat the experiment with K ranging from 3 to 55. The top picture in figure 4 shows the average curves of $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ of all the algorithms as a function of K. For the five existing methods, ACDC, $\text{ACDC}^{+}_{\text{LU}}$, FFDIAG, LUJ1D and QRJ1D, their performance is quite stable with respect to K. The performance of the proposed methods progresses as K increases, and then practically stabilizes for high values of K. It indicates that after some point (e.g. $K \geq 20$), the additional information brought by an increase of K does not further improve the results. The proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms maintain competitive advantages through all the K values. The two images in the bottom of figure 4 present the average numerical complexity and CPU time of all the algorithms in this experiment, respectively. It shows that the numerical complexity of $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ is between that of ACDC and $\text{ACDC}^{+}_{\text{LU}}$. The $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ methods seem to be the most effective algorithms compared to the other methods.

Effect of coherence of $\boldsymbol{D}$

In this experiment, the effect of the coherence of the third loading matrix $\boldsymbol{D}$ of the 3-way array $\boldsymbol{\mathcal{C}} = [\![\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{D}]\!]$ is evaluated. Let $\boldsymbol{d}_n$ and $\boldsymbol{d}_m$ denote the n-th and m-th columns of $\boldsymbol{D}$, respectively. The angle $\psi_{n,m}$ between $\boldsymbol{d}_n$ and $\boldsymbol{d}_m$ can be derived by using the following Euclidean dot product formula $\boldsymbol{d}_n^{\mathsf{T}}\boldsymbol{d}_m = \|\boldsymbol{d}_n\|\|\boldsymbol{d}_m\|\cos(\psi_{n,m})$. Then the coherence ρ of $\boldsymbol{D}$ is defined as the maximum absolute cosine of angle $\psi_{n,m}$ between the columns of $\boldsymbol{D}$ as follows:

$$\rho = \max_{\substack{n,m \\ n\neq m}} |\cos(\psi_{n,m})| \text{ with } \cos(\psi_{n,m}) = \frac{\boldsymbol{d}_n^{\mathsf{T}}\boldsymbol{d}_m}{\|\boldsymbol{d}_n\|\|\boldsymbol{d}_m\|} \tag{43}$$

The quantity ρ is also known as the modulus of uniqueness of JDC [43]. By its definition (43), ρ falls in the range of $[0, 1]$. The JDC problem is considered to be

ill-conditioned when ρ is close to 1. Such an ill-conditioned problem can be met in ICA when $\boldsymbol{A}$ has nearly collinear column vectors. For example, in order to perform ICA, provided that all the sources are non Gaussian, which is often the case in practice, we can build a 3-way array $\boldsymbol{\mathcal{C}}$ by stacking the matrix slices of the fourth order cumulant array of the observation data. Then the loading matrix $\boldsymbol{D}$ can be expressed as follows:

$$\boldsymbol{D} = (\boldsymbol{A} \odot \boldsymbol{A})\boldsymbol{C}_{4,\{\boldsymbol{s}\}} \tag{44}$$

where $\boldsymbol{C}_{4,\{\boldsymbol{s}\}} = \text{diag}\,[\mathcal{C}_{1,1,1,1,\{\boldsymbol{s}\}}, \cdots, \mathcal{C}_{N,N,N,N,\{\boldsymbol{s}\}}]$ is a $(N \times N)$ diagonal matrix with $\mathcal{C}_{n,n,n,n,\{\boldsymbol{s}\}}$ being the fourth order cumulant of the n-th source, $n \in \{1, \cdots, N\}$, and where $\odot$ denotes the Khatri-Rao product. It can be observed that the coherence of the columns of $\boldsymbol{A}$ will influence the coherence of the matrix $\boldsymbol{D}$. In the following test, the dimensions of the 3-way array $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$ and $K = 15$. The SNR value is fixed to 10 dB. In order to control ρ, firstly we randomly generate an orthogonal matrix $\boldsymbol{D} \in \mathbb{R}^{15\times 5}$ so that $\rho = 0$ by orthogonalizing a (15×5) random matrix. Secondly we rotate its 5 columns such that all the internal angles between any columns are equal to a predefined value ψ. Therefore ρ is only controlled by the angle ψ and equals to $|\cos(\psi)|$. We repeat the experiment with the angle ψ ranging from 0 to $\pi/2$ with a step of $\pi/60$. A small ψ value means a large ρ value. The top picture in figure 5 displays the average curves of $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ of all the algorithms as a function of ψ. It shows that the nonnegativity constrained methods $\text{ACDC}^{+}_{\text{LU}}$, $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$, outperform the unconstrained ones ACDC, FFDIAG, LUJ1D and QRJ1D. The proposed algorithms are more efficient, particularly when the coherence level is high. The average numerical complexity and CPU time displayed in the bottom of figure 5 indicate that the $\text{JD}^{+}_{\text{LU}}$ algorithm provides the best performance/complexity compromise, while the $\text{JD}^{+}_{\text{QR}}$ algorithm is also competitive with regard to $\text{ACDC}^{+}_{\text{LU}}$.

Effect of condition number of $\boldsymbol{D}^{(k)}$

When the JDC problem is considered, a diagonal matrix $\boldsymbol{D}^{(k)}$ could contain some diagonal elements which, despite being non-zero, are many orders of magnitude lower than some other elements, leading to an ill-conditioned matrix $\boldsymbol{C}^{(k)}$. For the proposed methods, the inverse of such a matrix $\boldsymbol{C}^{(k)}$ would contain numerical errors. In this experiment, we study the performance of the seven algorithms as a function of the condition number of one of the diagonal matrices $\boldsymbol{D}^{(k)}$. The dimensions of the 3-way array $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$ and $K = 15$. The SNR value is set to 10 dB. We vary the condition number of the first diagonal matrix $\boldsymbol{D}^{(1)}$ from 1 to 1000 by fixing the ratio of its largest diagonal element to its smallest diagonal element. The top picture in figure 6 displays the average curves of the estimating error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ of the seven algorithms as a function of the condition number of $\boldsymbol{D}^{(1)}$. The results reveal that a highly ill-conditioned diagonal matrix $\boldsymbol{D}^{(1)}$ has a clear negative effect on the estimation accuracy of all the algorithms. The nonnegativity constrained methods $\text{ACDC}^{+}_{\text{LU}}$, $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ outperform the classical algorithms ACDC, FFDIAG, LUJ1D and QRJ1D whatever the condition number is. The proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms maintain advantages when the condition number is less than 100.

Regarding the cases of larger condition numbers, $\text{ACDC}^{+}_{\text{LU}}$ is more superior since it does not need to invert the highly ill-conditioned matrix. It is worthy pointing out that in practice, we can choose these sufficiently well-conditioned matrices $\boldsymbol{C}^{(k)}$ for the proposed methods, whose condition numbers are below a predefined threshold. In addition, a weighted cost function for which weights would depend on the condition number of each matrix can be considered. On the other hand, the performance of the classical methods can also be improved by choosing a particular subset of available matrices [57] and by properly weighting the cost functions [49]. In order to give a fair comparison, all the algorithms operate on the same set of matrices in all the experiments of this paper. In addition, the average numerical complexity and CPU time at each condition number of all the methods in this experiment are shown in the bottom of figure 6. It shows that the proposed methods give the best performance/complexity trade-off compared to $\text{ACDC}^{+}_{\text{LU}}$ whatever the condition number is.

Test with a non-square matrix $\boldsymbol{A}$

As described in the section of practical issues, when a non-square matrix $\boldsymbol{A} \in \mathbb{R}_{+}^{N\times P}$ with $N > P$ is met in ICA, we propose to compress it by a nonnegative compression matrix $\boldsymbol{W}_{+} \in \mathbb{R}_{+}^{P\times N}$ [53], such that the resulting matrix $\bar{\boldsymbol{A}} = \boldsymbol{W}_{+}\boldsymbol{A}$ is a $(P \times P)$ nonnegative square matrix. Then the proposed methods can be applied to estimate $\bar{\boldsymbol{A}}$. Similarly to the classical prewhitening, the nonnegative compression step could introduce numerical errors. In this experiment, we compare our methods to ACDC and $\text{ACDC}^{+}_{\text{LU}}$ through a simulated ICA model. The latter algorithms can directly estimate a non-square matrix $\boldsymbol{A}$ from the fourth order cumulant matrix slices. The ICA model is established as follows:

$$\boldsymbol{x}[f] = \boldsymbol{A}\boldsymbol{s}[f] + \boldsymbol{\nu}[f] \tag{45}$$

where $\boldsymbol{x}[f] = [x_1[f], \cdots, x_N[f]]^{\mathsf{T}}$ is the $(N \times 1)$ observation vector, $\boldsymbol{s}[f] = [s_1[f], s_2[f], s_3[f]]^{\mathsf{T}}$ is the (3×1) zero-mean unit-variance source vector whose elements are independently drawn from a uniform distribution over $[-\sqrt{3}, \sqrt{3}]$, $\boldsymbol{\nu} = [\nu_1[f], \cdots, \nu_N[f]]^{\mathsf{T}}$ is the $(N \times 1)$ zero-mean unit-variance Gaussian noise vector, and $\boldsymbol{A}$ is the $(N \times 3)$ nonnegative mixing matrix whose components are independently drawn from a uniform distribution over $[0, 1]$. In this context the SNR is defined by:

$$\text{SNR} = 20\log_{10}(\|\{\boldsymbol{A}\boldsymbol{s}[f]\}\|_F / \|\{\boldsymbol{\nu}[f]\}\|_F) \tag{46}$$

For the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms, the given realization of $\{\boldsymbol{x}[f]\}$ is compressed by means of a matrix $\boldsymbol{W}_{+} \in \mathbb{R}_{+}^{3\times N}$ computed using the method proposed in [53], leading to a 3-dimensional vector $\{\bar{\boldsymbol{x}}[f]\}$. We compute the fourth order cumulant array of $\{\bar{\boldsymbol{x}}[f]\}$ and choose the first 3 matrix slices in order to build a 3-way array. Hence $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ decompose a $(3 \times 3 \times 3)$ array. Once the compressed mixing matrix $\bar{\boldsymbol{A}}$ is estimated, the original mixing matrix is obtained by equation (39). Regarding ACDC and $\text{ACDC}^{+}_{\text{LU}}$, the fourth order cumulant array of $\{\boldsymbol{x}[f]\}$ is directly computed without compression. We apply ACDC and $\text{ACDC}^{+}_{\text{LU}}$ on two 3-way arrays with different third dimensions. The first array of dimension $(N \times N \times 3)$

is built by choosing the first 3 matrix slices from the fourth order cumulant array, while the second array of dimension $(N \times N \times N)$ is built using the first N matrix slices. We study the impact of the number of observations N on the performance of the JDC algorithms, by varying N from 4 to 24. The SNR value is fixed to 5 dB. The number of samples used to estimate the cumulants is set to 10^3. Figure 7 shows the average curves of the estimating error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ of all the algorithms as a function of N. As it can be seen, when $N \leq 15$, the larger the value of N, the more accurate estimation of $\boldsymbol{A}$ is achieved. When $N > 15$, the further increase of N does not bring significant improvement in terms of the estimation accuracy. ACDC and $\text{ACDC}^{+}_{\text{LU}}$ give better results when the array with a larger third dimension is considered. Their results on $(N \times N \times N)$ arrays outperform the proposed methods when $N = 4$. $\text{ACDC}^{+}_{\text{LU}}$ also gives the best estimation on $(N \times N \times N)$ arrays with $N = 5$. It suggests that the numerical errors introduced by the compression step limit the performance of the proposed methods when only a small number of observation is available. Such a negative effect can be partially compensated by using a large number of observations, since the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ methods maintain the highest performance in terms of estimation accuracy when $N \geq 6$. The performance ACDC and $\text{ACDC}^{+}_{\text{LU}}$ can be further improved by using a $(N \times N \times N^2)$ array, which contains all the N^2 fourth order cumulant matrix slices. However, it leads to a higher numerical complexity especially for a large value of N. Regarding the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ methods, their performance can also be improved by using all the 9 matrix slices of the fourth order cumulant array of the compressed observation vector. Nevertheless, the experimental result has already shown that by using only a small number of matrix slices, $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ can maintain lower numerical complexities than ACDC and $\text{ACDC}^{+}_{\text{LU}}$, while achieving better estimating results, when a large value of N is considered. Therefore, despite the negative influence of the nonnegative compression, the proposed methods still offer a good performance/complexity compromise to estimate a non-square matrix $\boldsymbol{A}$.

BSS application on MRS data

In this section, we aim to illustrate the potential capability of the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms for solving a real-life BSS problem by an application carried on simulated MRS data.

MRS is a powerful non-invasive analytical technique for analyzing the chemical content of MR-visible nuclei and therefore enjoys particular advantages for assessing metabolism. The chemical property of each nucleus determines the frequency at which it appears in the MR spectrum, giving rise to peaks corresponding to specific metabolites [58]. Therefore, the MRS observation spectra can be modeled as the mixture of the spectrum of each constitutional source metabolite. More specifically, it can be written as the noisy linear instantaneous mixing model described in equation (45), where $\boldsymbol{x}[f] \in \mathbb{R}^N$ is the MRS observation vector, $\boldsymbol{s}[f] \in \mathbb{R}^P$ is the source vector representing the statistically quasi-independent source metabolites, $\boldsymbol{\nu} \in \mathbb{R}^N$ is the instrumental noise vector, and $\boldsymbol{A} \in \mathbb{R}_{+}^{N \times P}$ is the nonnegative mixing matrix containing the positive concentrations of the source metabolites. SNR is defined as in equation (46). In this experiment, two simulated MRS source metabolites $\{s_1[f]\}$ and $\{s_2[f]\}$, namely the Choline (Cho) and Myo-inositol (Ins) (see figure 8 (b)),

are generated by Lorentzian and Gaussian functions [59]. Each of the sources contains 10^3 samples. The observation vector $\boldsymbol{x}[f]$ is generated according to (45). The components of the $(N \times 2)$ mixing matrix $\boldsymbol{A}$ are randomly drawn from a uniform distribution. The additive noise $\boldsymbol{\nu}[f]$ is modeled as a zero-mean unit-variance Gaussian vector. The ICA methods based on the proposed JD^+_{LU} and JD^+_{QR} algorithms, namely JD^+_{LU}-ICA and JD^+_{QR}-ICA, consist of four steps: i) compressing $\{\boldsymbol{x}[f]\}$ by means of a matrix $\boldsymbol{W}_+ \in \mathbb{R}^{2\times N}_+$ [53], ii) estimating the fourth order cumulant array of the compressed observations and stacking all the cumulant matrix slices in a 3-way array, iii) decomposing the resulting 3-way array by means of JD^+_{LU} and JD^+_{QR}, respectively, and iv) reconstructing the sources. The JD^+_{LU}-ICA and JD^+_{QR}-ICA are compared to four state-of-the-art BSS algorithms, namely two efficient ICA methods CoM_2 [54] and SOBI [47], the NonNegative ICA (NNICA) method with a line search along the geodesic [55] and the NMF method [56] based on alternating nonnegativity least squares. The performance is assessed by means of the error $\alpha(\{\boldsymbol{s}[f]\}^{\mathsf{T}}, \{\tilde{\boldsymbol{s}}[f]\}^{\mathsf{T}})$ between the true source $\boldsymbol{s}[f]$ and its estimate $\tilde{\boldsymbol{s}}[f]$, the numerical complexity, and the CPU time. To find out the detailed analysis of the numerical complexity of the classical ICA algorithms, the reader can refer to the book chapter [60]. Figure 8 shows an example of the separation results of all the methods with $N = 32$ observations and a SNR of 10 dB. Regarding CoM_2, SOBI, NNICA and NMF, there are some obvious disturbances presented in the estimated metabolites. As far as JD^+_{LU}-ICA and JD^+_{QR}-ICA are concerned, the estimated source metabolites are quasi-perfect. Furthermore, the comprehensive performance of all the methods will be studied by the following experiments with 200 independent Monte Carlo trials.

In the first experiment, the effect of the number of observations N is evaluated. The SNR is fixed to 10 dB. The six methods are compared with N ranging from 4 to 116 with a step of 4. The average curves of error $\alpha(\{\boldsymbol{s}[f]\}^{\mathsf{T}}, \{\tilde{\boldsymbol{s}}[f]\}^{\mathsf{T}})$ as a function of N are shown in the left image of figure 9. It can be seen that the estimating errors of all the methods improve as N increases. It suggests that in noisy BSS contexts, using more sensors often yields better results. The proposed JD^+_{LU}-ICA and JD^+_{QR}-ICA methods maintain the competitive advantages. The average curves of the numerical complexities of this experiment are shown in the bottom left picture of figure 9. We can notice that the numerical complexities of all the methods increase with N. The complexities of JD^+_{LU}-ICA and JD^+_{QR}-ICA seem identical in the logarithmic scaled plot, that is because theoretically their complexities are mainly dominated by the computation of the nonnegative compression step and of the cumulants. Indeed, JD^+_{LU}-ICA is more computationally efficient than JD^+_{QR}-ICA in the step of CP decomposition of the cumulant array. This can be verified by the average CPU time of those methods, shown in the bottom right image of figure 9. We can observe that JD^+_{LU}-ICA is slower than CoM_2, but it is faster than NNICA, SOBI and NMF.

In the second experiment, we study the influence of SNR on the performance of the six methods. The number of observations N is set to 32. SNR is varied from 0 dB to 50 dB with a step of 2 dB. The average curves of the estimating error $\alpha(\{\boldsymbol{s}[f]\}^{\mathsf{T}}, \{\tilde{\boldsymbol{s}}[f]\}^{\mathsf{T}})$, as well as those of the numerical complexities and CPU time as a function of SNR of all the six methods are shown in figure 10. The proposed JD^+_{LU}-ICA and JD^+_{QR}-ICA methods provide the best estimating results

with moderate computational complexities and CPU time. Generally speaking, the $\text{JD}^{+}_{\text{LU}}$-ICA algorithm offers the best performance/complexity trade-off in this BSS experimental context.

Conclusion

We have proposed two methods, called $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$, in order to achieve the CP decomposition of semi-nonnegative semi-symmetric 3-way arrays. The nonnegativity constraint is imposed on the two symmetric modes of 3-way arrays by means of a square change of variable, giving rise to an unconstrained joint diagonalization by congruence problem. Therefore, the nonnegative loading matrix can be estimated by computing the joint diagonalizer. We consider the elementary LU and QR parameterizations of the Hadamard square root of the nonnegative joint diagonalizer, leading to two Jacobi-like optimization procedures. In each Jacobi-like iteration, the optimization is formulated into a minimization of a polynomial or rational function with respect to only one parameter. In addition, the numerical complexity for each algorithm has been analyzed.

The performance of the proposed $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$ algorithms is evaluated with simulated semi-nonnegative semi-symmetric 3-way arrays. Four classical nonorthogonal JDC methods without nonnegativity constraints, including ACDC [23], FF-DIAG [25], LUJ1D [26], QRJ1D [26], and one nonnegative JDC method $\text{ACDC}^{+}_{\text{LU}}$ [41], are tested as reference methods. The performance is assessed in terms of the matrix estimation accuracy, the numerical complexity, and the CPU time. The convergence property, the influence of SNR, the impact of dimension, the effect of coherence, and the influence of condition number, are extensively studied by Monte Carlo experiments. The obtained results show that the proposed algorithms offer better estimation accuracy by means of exploiting the nonnegativity *a priori*. The $\text{JD}^{+}_{\text{LU}}$ algorithm provides the best performance/complexity compromise.

The proposed algorithms are suitable tools for solving the ICA problems where a nonnegative mixing matrix is considered, such as in MRS. In this case, the 3-way array built by stacking the matrix slices of a HO cumulant array fulfills the semi-nonnegative semi-symmetric structure. We proposed two ICA methods, namely $\text{JD}^{+}_{\text{LU}}$-ICA and $\text{JD}^{+}_{\text{QR}}$-ICA, based on CP decomposition of the fourth order cumulant array using $\text{JD}^{+}_{\text{LU}}$ and $\text{JD}^{+}_{\text{QR}}$, respectively. The source separation ability of the proposed algorithms is verified through a BSS application carried out on simulated MRS data. The $\text{JD}^{+}_{\text{LU}}$-ICA and $\text{JD}^{+}_{\text{QR}}$-ICA are compared to one NMF method [56], one nonnegative ICA method [55], and two classical ICA methods, namely CoM_2 [54] and SOBI [47]. The performance is comprehensively studied as a function of the number of observations and of SNR. The experimental results demonstrate the improvement of the proposed methods in terms of the source estimation accuracy, and also show that exploiting the two *a priori* of the data, namely the nonnegativity of the mixing matrix and the statistical independence of the sources, allows us to achieve better estimation results. The $\text{JD}^{+}_{\text{LU}}$-ICA algorithm provides the best performance/complexity trade-off.

Competing interests
The authors declare that they have no competing interests.

Author details

[1]INSERM, UMR 1099, F-35000 Rennes, France. [2]LTSI, Université de Rennes 1, Bât. 22, Campus de Beaulieu, F-35000 Rennes, France. [3]Laboratory of Image Science and Technology, School of Computer Science and Engineering, Southeast University, Qun Xian Building, No. 2 Sipailou, 210096 Nanjing, China. [4]Centre de Recherche en Information Biomédicale Sino-Français (CRIBs), F-35000 Rennes, France. [5]Centre INRIA Rennes-Bretagne Atlantique, Bât. 12G, Campus de Beaulieu, F-35042 Rennes, France.

References

1. Smilde, A., Bro, R., Geladi, P.: Multi-way Analysis: Applications in the Chemical Sciences. Wiley, West Sussex, England (2004)
2. de Almeida, A.L.F., Favier, G., Ximenes, L.R.: Space-time-frequency (STF) MIMO communication systems with blind receiver based on a generalized PARATUCK2 model. IEEE Trans. Signal Process. **61**(8), 1895–1909 (2013)
3. De Vos, M., Vergult, A., De Lathauwer, L., De Clercq, W., Van Huffel, S., Dupont, P., Palmini, A., Van Paesschen, W.: Canonical decomposition of ictal scalp EEG reliably detects the seizure onset zone. Neuroimage. **37**(3), 844–854 (2007)
4. Comon, P., Luciani, X., de Almeida, A.L.F.: Tensor decompositions, alternating least squares and other tales. J. Chemometr. **23**, 393–405 (2009)
5. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
6. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
7. Kruskal, J.B.: Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. Linear Algebra and its Applications **18**(2), 98–138 (1977)
8. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys. **6**(1), 164–189 (1927)
9. Kroonenberg, P.M.: Applied Multiway Data Analysis. Wiely, Hoboken, New Jersey (2008)
10. Sidiropoulos, N.D., Bro, R., Giannakis, G.B.: Parallel factor analysis in sensor array processing. IEEE Trans. Signal Process. **48**(8), 2377–2388 (2000)
11. de Almeida, A.L.F., Favier, G., Motab, J.C.M.: PARAFAC-based unified tensor modeling for wireless communication systems with application to blind multiuser equalization. Signal Process. **87**(2), 337–351 (2007)
12. de Almeida, A.L.F., Favier, G.: Double khatri-rao space-time-frequency coding using semi-blind PARAFAC based receiver. IEEE Signal Process. Lett. **20**(5), 471–474 (2013)
13. Albera, L., Ferréol, A., Comon, P., Chevalier, P.: Blind identification of overcomplete mixtures of sources (BIOME). Linear Algebra and its Applications **391**, 3–30 (2004)
14. Röemer, F., Haardt, M.: Tensor-based channel estimation and iterative refinements for two-way relaying with multiple antennas and spatial reuse. IEEE Trans. Signal Process. **58**(11), 5720–5735 (2010)
15. Harshman, R.A., Lundy, M.E.: PARAFAC: Parallel factor analysis. Computational Statistics & Data Analysis **18**(1), 39–72 (1994)
16. Uschmajew, A.: Local convergence of the alternating least squares algorithm for canonical tensor approximation. SIAM. J. Matrix Anal. Appl. **33**(2), 639–652 (2012)
17. Rajih, M., Comon, P., Harshman, R.A.: Enhanced line search: A novel method to accelerate PARAFAC. SIAM J. Matrix Anal. Appl. **30**(3), 1128–1147 (2008)
18. Acar, E., Dunlavy, D.M., Kolda, T.G.: A scalable optimization approach for fitting canonical tensor decompositions. J. Chemometr. **25**(2), 67–86 (2011)
19. Röemer, F., Haardt, M.: A semi-algebraic framework for approximate CP decompositions via simultaneous matrix diagonalizations (SECSI). Signal Processing **93**(9), 2722–2738 (2013)
20. Luciani, X., Albera, L.: Canonical polyadic decomposition based on joint eigenvalue decomposition. Chemometr. Intell. Lab. **132**, 152–167 (2014)
21. Carroll, J.D., Chang, J.-J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. Psychometrika **35**(3), 283–319 (1970)
22. Husson, F., Pagés, J.: Indscal model: geometrical interpretation and methodology. Computational Statistics & Data Analysis **50**(2), 358–378 (2006)
23. Yeredor, A.: Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. IEEE Trans. Signal Process. **50**(7), 1545–1553 (2002)
24. Cardoso, J.F., Souloumiac, A.: Jacobi angles for simultaneous diagonalization. SIAM J. Matrix Anal. Appl. **17**, 161–164 (1996)
25. Ziehe, A., Laskov, P., Nolte, G., Muller, K.-R.: A fast algorithm for joint diagonalization with non-orthogonal transformations and its application to blind source separation. J. Mach. Learning Res. **5**, 777–800 (2004)
26. Afsari, B.: Simple LU and QR based non-orthogonal matrix joint diagonalization. In: ICA 2006, Springer LNCS 3889 (2006)
27. Van der Veen, A.J.: Joint diagonalization via subspace fitting techniques. In: Proc. ICASSP '01, vol. 5, pp. 2773–2776 (2001)
28. Yeredor, A.: On using exact joint diagonalization for noniterative approximate joint diagonalization. IEEE Signal Process. Lett. **12**(9), 645–648 (2005)
29. Vollgraf, R., Obermayer, K.: Quadratic optimization for simultaneous matrix diagonalization. IEEE Trans. Signal Process. **54**(9), 3270–3278 (2006)
30. Li, X.L., Zhang, X.D.: Nonorthogonal joint diagonalization free of degenerate solution. IEEE Trans. Signal Process. **55**(5), 1803–1814 (2007)

31. Souloumiac, A.: Nonorthogonal joint diagonalization by combining Givens and hyperbolic rotations. IEEE Trans. Signal Process. **57**(6), 2222–2231 (2009)
32. Xu, X.F., Feng, D.Z., Zheng, W.X.: A fast algorithm for nonunitary joint diagonalization and its application to blind source separation. IEEE Trans. Signal Process. **59**(7), 3457–3463 (2011)
33. Chabriel, G., Barrère, J.: A direct algorithm for nonorthogonal approximate joint diagonalization. IEEE Trans. Signal Process. **60**(1), 39–47 (2012)
34. Chabriel, G., Kleinsteuber, M., Moreau, E., Shen, H., Tichavský, P., Yeredor, A.: Joint matrices decompositions and blind source separation: A survey of methods, identification, and applications. IEEE Signal Processing Magazine **31**(3), 34–43 (2014)
35. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
36. Zhang, Q., Wang, H., Plemmons, R.J., Pauca, V.P.: Tensor methods for hyperspectral data analysis: a space object material identification study. J. Opt. Soc. Am. A. Opt. Image Sci. Vis. **25**(12), 3001–3012 (2008)
37. Royer, J.-P., Thirion-Moreau, N., Comon, P.: Computing the polyadic decomposition of nonnegative third order tensors. Signal Processing **91**(9), 2159–2171 (2011)
38. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. WILEY, West Sussex, United Kingdom (2009)
39. Zhou, G.X., Cichocki, A., Zhao, Q., Xie, S.L.: Nonnegative matrix and tensor factorizations : An algorithmic perspective. IEEE Signal Processing Magazine **31**(3), 54–65 (2014)
40. Coloigner, J., Karfoul, A., Albera, L., Comon, P.: Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors. Linear Algebra and its Applications **450**(1), 334–374 (2014)
41. Wang, L., Albera, L., Kachenoura, A., Shu, H.Z., Senhadji, L.: Nonnegative joint diagonalization by congruence based on LU matrix factorization. IEEE Signal Process. Lett. **20**(8), 807–810 (2013)
42. Wang, L., Albera, L., Kachenoura, A., Shu, H.Z., Senhadji, L.: CP decomposition of semi-nonnegative semi-symmetric tensors based on QR matrix factorization. In: SAM'14, Proceedings of the Eighth IEEE Sensor Array and Multichannel Signal Processing Workshop, A Coruna, Spain, pp. 449–452 (2014)
43. Afsari, B.: Sensitivity analysis for the problem of matrix joint diagonalization. SIAM J. Matrix Anal. Appl. **30**(3), 1148–1171 (2008)
44. Wax, M., Sheinvald, J.: A least-squares approach to joint diagonalization. IEEE Signal Process. Lett. **4**(2), 52–53 (1997)
45. Dégerine, S., Kane, E.: A comparative study of approximate joint diagonalization algorithms for blind source separation in presence of additive noise. IEEE Trans. Signal Process. **55**(6), 3022–3031 (2007)
46. Fadaili, E.M., Thirion-Moreau, N., Moreau, E.: Nonorthogonal joint diagonalization/zero diagonalization for source separation based on time-frequency distributions. IEEE Trans. Signal Process. **55**(5), 1673–1687 (2007)
47. Belouchrani, A., Abed-Meraim, K., Cardoso, J.F., Moulines, E.: A blind source separation technique using second-order statistics. IEEE Trans. Signal Process. **45**(2), 434–444 (1997)
48. Pham, D.T.: Joint approximate diagonalization of positive definite Hermitian matrices. SIAM J. Matrix Anal. Appl. **22**, 1837–1848 (2001)
49. Tichavský, P., Yeredor, A.: Fast approximate joint diagonalization incorporating weight matrices. IEEE Trans. Signal Process. **57**(3), 878–891 (2009)
50. Chu, M., Diele, F., Plemmons, R., Ragni, S.: Optimality computation and interpretation of nonnegative matrix factorizations. Technical report, Wake Forest University (2004)
51. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia, PA, USA (2000)
52. Vaidyanathan, P.P.: Multirate Systems and Filter Banks. PTR Prentice Hall, United States (1993)
53. Wang, L., Kachenoura, A., Albera, L., Karfoul, A., Shu, H.Z., Senhadji, L.: Nonnegative compression for semi-nonnegative independent component analysis. In: SAM'14, Proceedings of the Eighth IEEE Sensor Array and Multichannel Signal Processing Workshop, A Coruna, Spain, pp. 81–84 (2014)
54. Comon, P.: Independent component analysis, a new concept? Signal Process. **36**(3), 287–314 (1994)
55. Plumbley, M.D.: Algorithms for nonnegative independent component analysis. IEEE Trans. Neural Netw. **14**(3), 534–543 (2003)
56. Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. **30**(2), 713–730 (2008)
57. De Lathauwer, L.: Algebraic methods after prewhitening. In: Comon, P., Jutten, C. (eds.) Handbook of Blind Source Separation, pp. 155–177. Elsevier, Oxford, UK (2010). Chap. 5
58. Befroy, D.E., Shulman, G.I.: Magnetic resonance spectroscopy studies of human metabolism. Diabetes **60**(5), 1361–1369 (2011)
59. Moussaoui, S.: Séparation de sources non-négatives: application au traitement des signaux de spectroscopie. PhD thesis, Université Henri Poincaré (2005)
60. Albera, L., Comon, P., Parra, L.C., Karfoul, A., Kachenoura, A., Senhadji, L.: Biomedical applications. In: Comon, P., Jutten, C. (eds.) Handbook of Blind Source Separation, pp. 737–777. Elsevier, Oxford, UK (2010). Chap. 18

Tables
Figures

Table 1 Numerical complexities of seven JDC algorithms in terms of flops

	Numerical complexity
	(N, N, K): the dimensions of the 3-way array $\boldsymbol{\mathcal{C}}$; For ACDC, FFDIAG, LUJ1D and QRJ1D, N_s is the number of total sweeps; For ACDC$^+_{\mathsf{LU}}$, JD$^+_{\mathsf{LU}}$, and JD$^+_{\mathsf{QR}}$, N_s^1 is the number of sweeps without nonnegativity constraint; N_s^2 is the number of sweeps with explicit nonnegativity constraint.
ACDC	$(13/3N^3K + 3N^4 + 2N^2K + N^3 + N^2)N_s$
FFDIAG	$(2N^3K + N^3 + 2N^2K + 4N(N-1))N_s$
LUJ1D	$(4NK + N - 2K)N(N-1)N_s$
QRJ1D	$(6NK + 2.5N + 1.5K)N(N-1)N_s$
ACDC$^+_{\mathsf{LU}}$	$((15N^2 + 4N)KN(N-1) + 4/3N^2K + N^3 + N^2)N_s^1$ $+((33N^2 + 7N)KN(N-1) + 4/3N^2K + N^3 + N^2)N_s^2$
JD$^+_{\mathsf{LU}}$	$3N^3K + (4NK + N - 2K)N(N-1)N_s^1$ $+((5N^2 + 16N - 7)K + 4N)N(N-1)N_s^2$
JD$^+_{\mathsf{QR}}$	$3N^3K + (6NK + 2.5N + 1.5K)N(N-1)N_s^1$ $+((5N^2 + 15.5N + 21)K + 7N)N(N-1)N_s^2$

Table 2 Average numerical complexities (in flops) and computation time (in seconds) of the convergence experiment

	SNR $= 25$ dB		SNR $= 10$ dB		SNR $= -5$ dB	
	Complexity	Time	Complexity	Time	Complexity	Time
ACDC	2.1708×10^6	1.1357	2.0338×10^6	1.0535	1.6800×10^6	0.8761
FFDIAG	1.1001×10^5	0.0331	1.0878×10^5	0.0327	9.4380×10^4	0.0287
LUJ1D	2.8903×10^5	0.0660	2.3126×10^5	0.0526	1.4199×10^5	0.0325
QRJ1D	2.2158×10^5	0.0383	2.4445×10^5	0.0421	2.6989×10^5	0.0468
ACDC$^+_{\mathsf{LU}}$	2.4462×10^7	2.6735	2.7498×10^7	2.8034	2.9646×10^7	2.9119
JD$^+_{\mathsf{LU}}$	2.8487×10^6	0.8107	4.9684×10^6	1.1098	7.1434×10^6	1.2938
JD$^+_{\mathsf{QR}}$	3.0766×10^6	1.0455	5.0554×10^6	1.1932	8.2185×10^6	1.3026

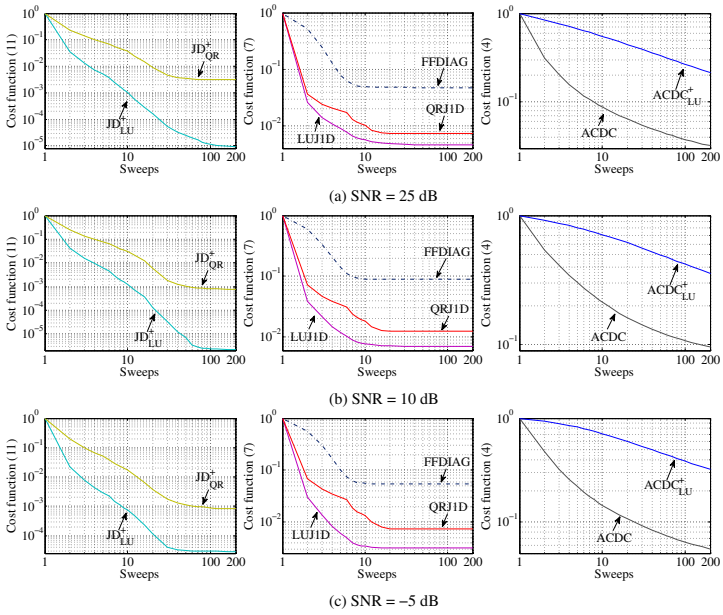


Figure 1 JDC performance versus sweeps. The average value of the cost function evolution of all the algorithms as a function of the number of sweeps with various SNR levels. The dimensions of $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$ and $K = 15$. The SNR values are set to 25 dB (top row), 10 dB (middle row) and -5 dB (bottom row), respectively.

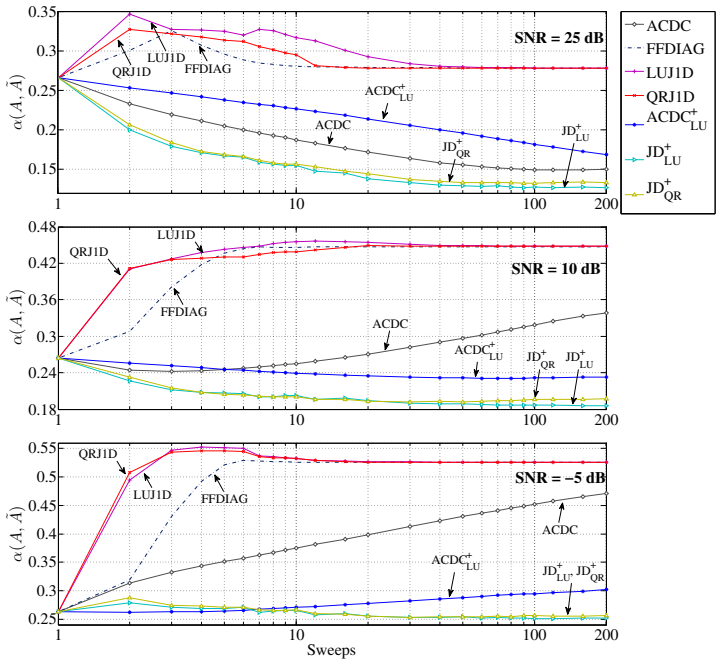


Figure 2 JDC performance versus sweeps. The average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of the number of sweeps with various SNR levels. The dimensions of $\boldsymbol{\mathcal{C}}_N$ are set to $N = 5$ and $K = 15$. The SNR values are set to 25 dB (top), 10 dB (middle) and -5 dB (bottom), respectively.

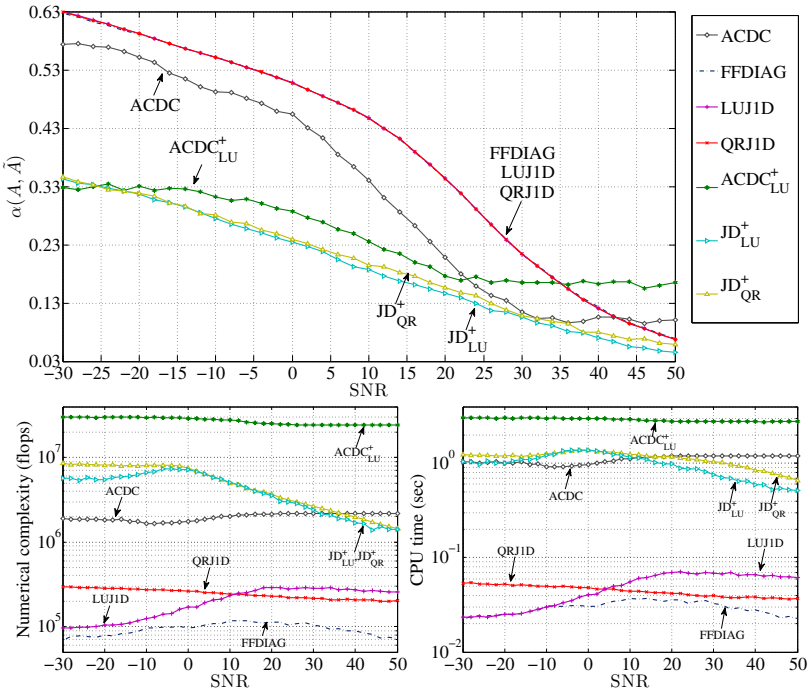


Figure 3 JDC performance versus SNR. The dimensions of $\mathcal{C}_N$ are set to $N = 5$ and $K = 15$. Top: the average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of SNR. Bottom: the average numerical complexities (left) and the CPU time (right) of all the algorithms, respectively.

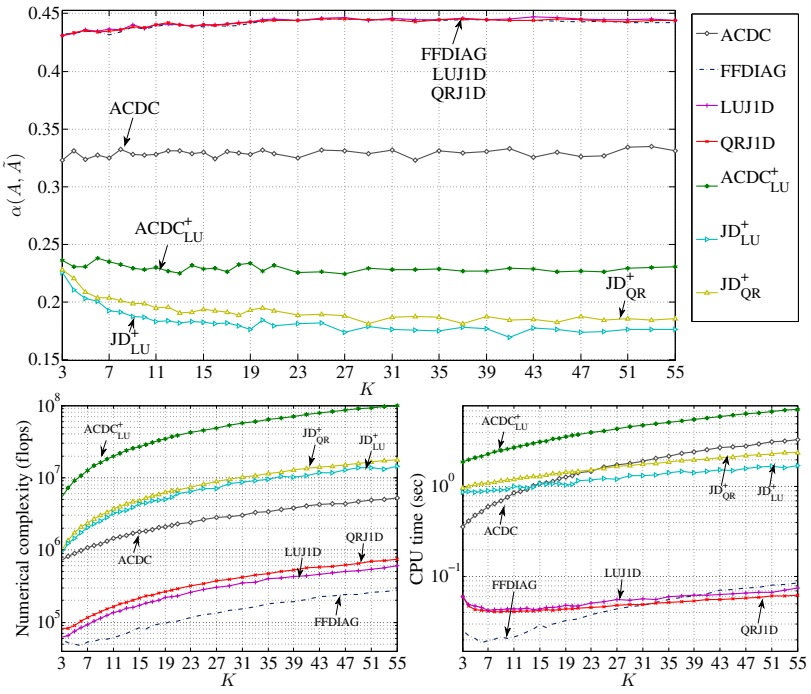


Figure 4 JDC performance versus dimension K. The first and second dimensions of $\mathcal{C}_N$ and the SNR value are set to $N = 5$ and SNR $= 10$ dB, respectively. Top: the average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of dimension K. Bottom: the average numerical complexities (left) and the CPU time (right) of all the algorithms, respectively.

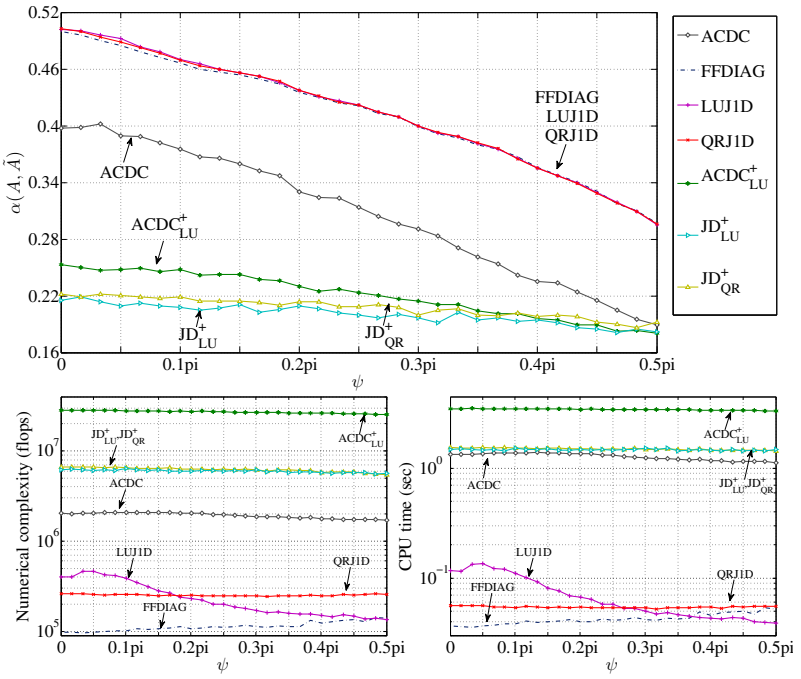


Figure 5 JDC performance versus coherence. The dimensions of $\mathcal{C}_N$ and the SNR value are set to $N = 5$, $K = 15$, and SNR $= 10$ dB, respectively. Top: the average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of internal angle ψ between any two columns of $\boldsymbol{D}$. Bottom: the average numerical complexities (left) and the CPU time (right) of all the algorithms, respectively.

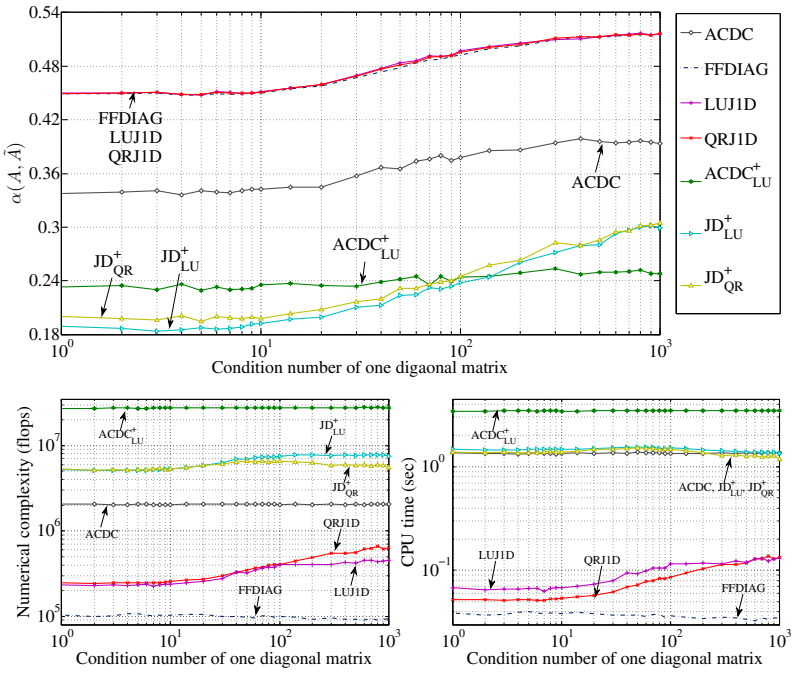


Figure 6 JDC performance versus condition number. The dimensions of $\mathcal{C}_N$ and the SNR value are set to $N = 5$, $K = 15$, and SNR $= 10$ dB, respectively. Top: the average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of the condition number of one of the diagonal matrices $\boldsymbol{D}^{(k)}$. Bottom: the average numerical complexities (left) and the CPU time (right) of all the algorithms, respectively.

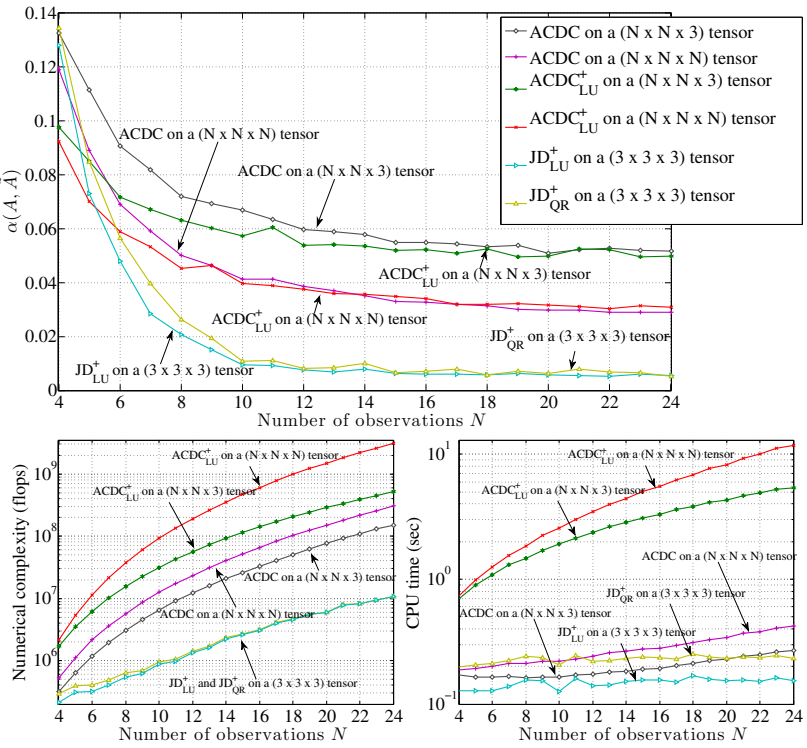
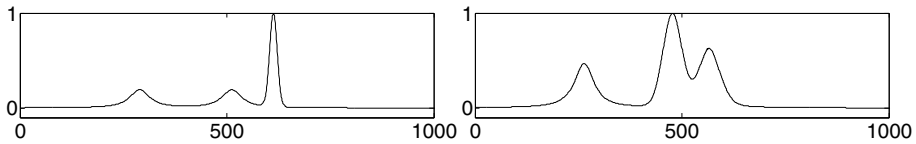
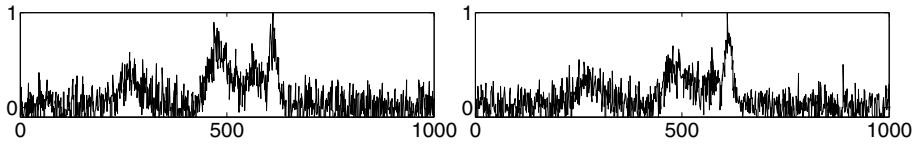


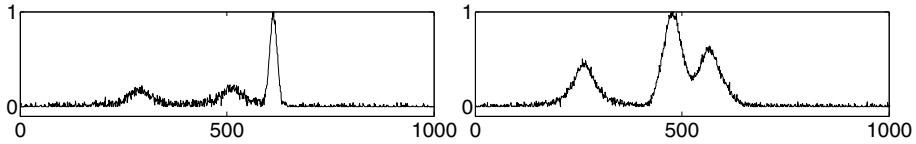
Figure 7 JDC performance on an ICA model versus number of observations. The number of sources P and the SNR value are set to $P = 3$ and SNR $= 5$ dB, respectively. Top: the average error $\alpha(\boldsymbol{A}, \tilde{\boldsymbol{A}})$ evolution of all the algorithms as a function of the number of observations N. Bottom: the average numerical complexities (left) and the CPU time (right) of all the algorithms, respectively.



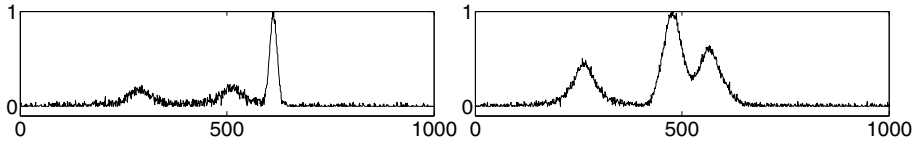
(a) Cho and Ins source metabolites



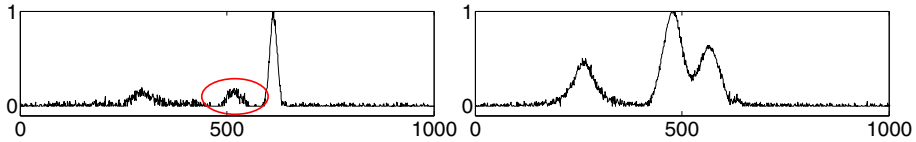
(b) Two of the MRS observations



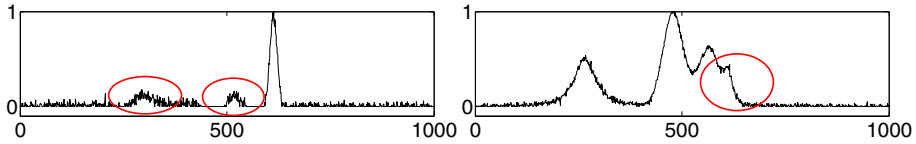
(c) Separated metabolites by $\text{JD}^{+}_{\text{LU}}$-ICA



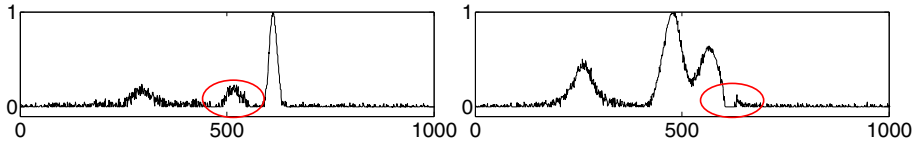
(d) Separated metabolites by $\text{JD}^{+}_{\text{QR}}$-ICA



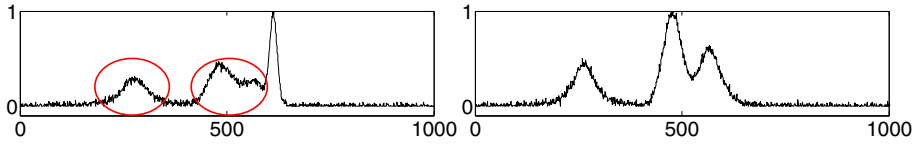
(e) Separated metabolites by CoM_2



(f) Separated metabolites by SOBI



(g) Separated metabolites by NNICA



(h) Separated metabolites by NMF

Figure 8 BSS results on MRS data. An example of the results of blind separation of 2 simulated MRS metabolites. The number of observations N is set to 32 and the SNR value is 10 dB. (a) Cho and Ins source metabolites. (b) Two of the observations. (c)-(h) separated metabolites by $\text{JD}^{+}_{\text{LU}}$-ICA, $\text{JD}^{+}_{\text{QR}}$-ICA, CoM_2, SOBI, NNICA and NMF, respectively.

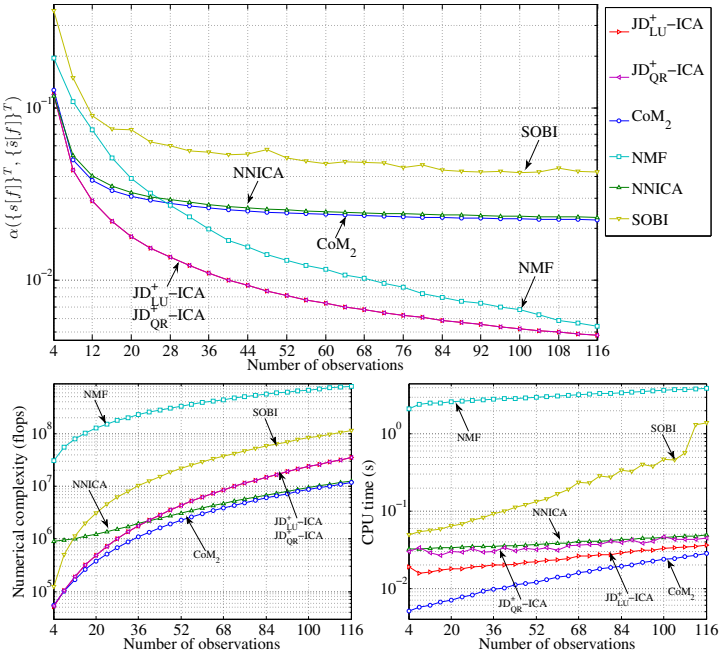


Figure 9 BSS performance on MRS data versus the number of observations. Average results of blind separation of 2 simulated MRS metabolites. The SNR value is set to 10 dB. Left: the average error $\alpha(\{\boldsymbol{s}[f]\}^{\mathsf{T}}, \{\tilde{\boldsymbol{s}}[f]\}^{\mathsf{T}})$ evolution of all the algorithms as a function of the number of observations. Right: the average numerical complexities (top) and the CPU time (bottom) of all the algorithms, respectively.

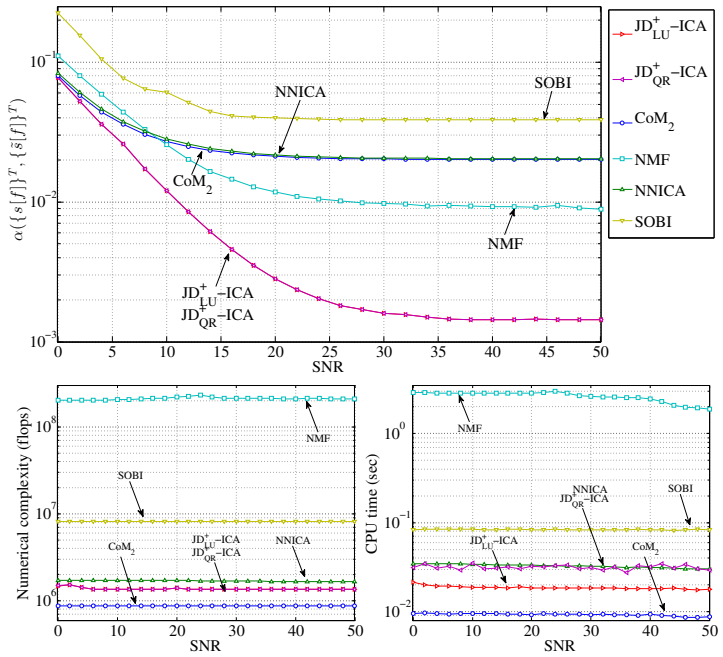


Figure 10 BSS performance on MRS data versus SNR. Average results of blind separation of 2 simulated MRS metabolites. The number of observations is set to $N = 32$. Left: the average error $\alpha(\{\boldsymbol{s}[f]\}^{\mathsf{T}}, \{\tilde{\boldsymbol{s}}[f]\}^{\mathsf{T}})$ evolution of all the algorithms as a function of SNR. Right: the average numerical complexities (top) and the CPU time (bottom) of all the algorithms, respectively.