HAL
archives-ouvertes.fr

Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors

Julie Coloigner, Ahmad Karfoul, Laurent Albera, Pierre Comon

▶ **To cite this version:**

Julie Coloigner, Ahmad Karfoul, Laurent Albera, Pierre Comon. Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors. Linear Algebra and Applications, Elsevier - Academic Press, 2014, 450, pp.334-374. <hal-00945606>

HAL Id: hal-00945606
https://hal.archives-ouvertes.fr/hal-00945606

Submitted on 12 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Line search and trust region strategies for canonical decomposition of semi-nonnegative semi-symmetric 3rd order tensors

Julie Coloigner[a,b], Ahmad Karfoul[d], Laurent Albera[a,b,c], Pierre Comon[e]

[a] *Inserm, UMR 1099, Rennes, F-35000, France*
[b] *LTSI Laboratory, University of Rennes* 1*, Rennes, F-35000, France*
[c] *INRIA, Centre Inria Rennes - Bretagne Atlantique, 35042 Rennes Cedex, France*
[d] *AL-Baath University, Faculty of Mech. and Elec. Eng., PB. 2244, Homs, Syria*
[e] *Lab. GIPSA, CNRS, UMR 5216, BP.46, 38402 Saint Martin d'Heres cedex, France*

Abstract

Numerical solutions are proposed to fit the CanDecomp/ParaFac (CP) model of real three-way arrays, when the latter are both nonnegative and symmetric in two modes. In other words, a semi-nonnegative INDSCAL analysis is performed. The nonnegativity constraint is circumvented by means of changes of variable into squares, leading to an unconstrained problem. In addition, two globalization strategies are studied, namely line search and trust region. Regarding the former, a global plane search scheme is considered. It consists in computing, for a given direction, one or two optimal stepsizes, depending on whether the same stepsize is used in various updating rules. Moreover, we provide a compact matrix form for the derivatives of the objective function. This allows for a direct implementation of several iterative algorithms such as conjugate gradient, Levenberg-Marquardt and Newton-like methods, in matrix programming environments like MATLAB. Our numerical results show the advantage of our optimization strategies when combined with a priori information such as partial symmetry.

Keywords: Canonical polyadic model, nonnegative matrix factorization, nonnegative tensor factorization, semi-nonnegative INDSCAL analysis, CanDecomp, ParaFac, nonlinear optimization, matrix derivation.

1. Introduction

INdividual Differences in multidimensional SCALing (INDSCAL) analysis consists in decomposing a three-way array into a minimal sum of vector outer products, which are symmetric in two modes [1]. It is a special case of CanDecomp/ParaFac (CP) model of three-way arrays [3, 4, 5]. INDSCAL analysis has been most often applied in psychometric literature [6], social sciences [7], marketing research [8], signal processing [9], and more generally in data analysis, such as in the context of multiple factor analysis [10]. On the other hand, it appears in signal processing, and more particularly in Independent Component Analysis (ICA) [9]. Various algorithms have been proposed imposing the semi-symmetry constraint of the INDSCAL model,

Email addresses: julie.coloigner@gmail.com (Julie Coloigner), AKarfoul@gmail.com (Ahmad Karfoul), laurent.albera@univ-rennes1.fr (Laurent Albera), pierre.comon@gipsa-lab.grenoble-inp.fr (Pierre Comon)

such as those described in [9, 11] to cite a few. Other constraints can additionally be taken into account, according to the considered application. Indeed, as described in section 2, some ICA applications involve nonnegative mixing matrices, *i.e.* mixing matrices with nonnegative components, encountered in image processing for instance [12]. Thus, imposing a semi-nonnegative constraint to the INDSCAL model, which leads to the semi-nonnegative INDSCAL model, may improve its performance. To date, no method has been yet proposed to combine both constraints.

In this paper, the goal is precisely to compute the CP decomposition, under the constraints of symmetry and nonnegativity in two modes. The nonnegativity constraint is imposed by means of changes of variable into squares, leading to an unconstrained problem. Several numerical solutions are proposed, which belong to two fundamental strategies of optimization: line search and trust region. Regarding the former, we present two approaches, which are based on the use of first and second order derivatives: Conjugate Gradient (CG) and Newton-like methods, combined with a global plane search. As far as the latter strategy is concerned, a Levenberg-Marquardt (LM) method is proposed, using an approximation of the Hessian inside a trust region. Both optimization strategies are detailed in section 5 whereas the optimization procedure is described in section 7. Moreover, we provide, in section 6, a compact matrix expression of the derivatives of the objective function, which allows for a direct implementation of our iterative algorithms in matrix programming environments, like MATLAB. Next, a detailed numerical complexity study of our algorithms is provided in section 8. In section 9, numerical results show the benefit that can be expected from our algorithms on synthetic data. This benefit is also illustrated in section 9.2 through a practical context of image processing.

2. Motivation

Originally, the INDSCAL model was proposed by Carroll and Chang [1] as a multidimensional scaling method for studying individual differences in perceptions of various stimulus domains. In this context of data analysis, in order to extract similarities between stimuli or objects, the INDSCAL model is applied to a third order tensor in which the frontal slices are distance matrices computed from the measured data [10, 1]. But the most widespread use of the INDSCAL model is in ICA, where the problem can be reduced to a joint diagonalization (by congruence) [9, 18, 19, 22, 23]. In practice, the matrices to be jointly diagonalized are different time-delayed covariance matrices like in SOBI (Second Order Blind Identification) [70], time-frequency covariance matrices [20], or slices of higher order cumulant tensor(s) like in JADE (Joint Approximate Diagonalization of Eigenmatrices) where different slices of fourth order cumulant are jointly diagonalized [21].

ICA plays an important role in many areas including biomedical engineering [14, 15, 16, 17, 24], speech and audio [25, 26, 27, 28, 29, 30, 31, 12, 30], radiocommunications [32] and document restoration [33] to cite a few. For instance in [33], the authors use ICA to restore digital document images in order to improve the text legibility. Indeed, under the statistical independence assumption, authors succeed in separating foreground text and bleed-through/show-through in palimpsest images. Furthermore, authors in [12] use ICA to solve the ambiguity in X-ray images due to multi-object overlappings. They presented a novel object decomposition technique based on multi-energy plane radiographs. This technique selectively enhances an object that is characterized by a specific chemical composition ratio of basis materials while suppressing the other overlapping objects. Besides, in the context of classification of tissues and more particularly of brain tumors [35], ICA is very effective. In fact, it allows for feature extraction from Magnetic Resonance Spectroscopy (MRS) signals, representing them as a linear

combination of tissue spectra, which are as independent as possible [36]. Moreover, using the JADE algorithm [21] applied to a mixture of sound waves computed by means of the constant-Q transform (Fourier transform with log-frequency) of a temporal waveform broken up into a set of time segments, the authors of [28] describe trills as a set of note pairs described by their spectra and corresponding time envelopes. In this case, pitch and timing of each note present in the trill can be easily deduced. All the aforementioned applications show the high efficiency of the ICA and its robustness to the presence of noise. Despite this high efficiency in resolving the proposed applicative problems, authors did not fully exploit properties enjoyed by the mixing matrix such as its nonnegativity. For instance in [12], the thickness of each organ, which stands for the mixing coefficient, is real positive. Furthermore, reflectance indices in [33] for the background, the overwriting and the underwriting, which correspond to the mixing coefficients, are also nonnegative. Regarding tissue classification from MRS data, each observation is a linear combination of independent spectra with positive weights representing concentrations [38]; the mixing matrix is again nonnegative.

By imposing the nonnegativity of the mixing matrix within the ICA process, the extraction quality can be improved as will be shown in section 9.2 through computer results. Exploiting the nonnegativity property of the mixing matrix during the ICA process gives rise to what we call semi-nonnegative ICA. More particularly, the latter can be performed by computing a constrained joint CP decomposition of cumulant arrays of different orders [39] having the nonnegative mixing matrix as loading matrices. After merging the entries of the cumulant arrays in the same third order array (see section 9.2 for more details), the reformulated problem follows the semi-nonnegative INDSCAL model. Hence there is a real interest in proposing efficient numerical methods to perform the semi-nonnegative INDSCAL analysis.

3. Notations

Throughout this paper, vectors, matrices and three-way arrays are denoted with bold lowercase ($\boldsymbol{a}, \boldsymbol{b}, \cdots$), bold uppercase ($\boldsymbol{A}, \boldsymbol{B}, \cdots$) and bold calligraphic letters ($\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \cdots$), respectively. In addition, entries of a given structure are scalars and follow the same notation as their structures. For instance, the entry in row i and column j of a matrix $\boldsymbol{A}$ is denoted by $A_{i,j}$. Similarly, the (i, j, k)-th component of a third order array $\boldsymbol{\mathcal{B}}$ is denoted by $\mathcal{B}_{i,j,k}$. Upper bounds (i.e., $n = 1, 2, \cdots, N$) are denoted by italic capital letters. The matrix $\widehat{\boldsymbol{A}}$ denotes the estimate of $\boldsymbol{A}$. The symbol $\otimes$ denotes the Kronecker product while $\odot$ and $\boxdot$ stand for the Khatri-Rao product (column-wise Kronecker product) and the Hadamard product (element-wise product), respectively. The operator vec transforms matrices into vectors, that is: $\text{vec}(\boldsymbol{T})_{i+(j-1)I} = T_{i,j}$, with $\boldsymbol{T}$ and $\text{vec}(\boldsymbol{T})$ of size $(I \times J)$ and IJ, respectively. Next, unvec is the inverse operator so that $\text{unvec}(\text{vec}(\boldsymbol{T})) = \boldsymbol{T}$. The operator Mat rearranges a block matrix (or vector) into another; such as $\text{Mat}^{(N\times P,M)}([\boldsymbol{A}_1, \boldsymbol{A}_2, ..., \boldsymbol{A}_M]^\intercal) = [\boldsymbol{A}_1{}^\intercal, \boldsymbol{A}_2{}^\intercal, ..., \boldsymbol{A}_M{}^\intercal]$, where the M blocks, $\boldsymbol{A}_i$ with $1 \leq i \leq M$, are all of size $(N \times P)$. The notation $\text{diag}(\boldsymbol{Z})$ stands for the N-dimensional vector $\boldsymbol{z}$ built from the elements of the diagonal of the $(N \times N)$ matrix $\boldsymbol{Z}$. Furthermore, the superscripts $^\sharp$ and $^\intercal$ stand for the Moore-Penrose pseudoinverse and the transpose operators, respectively. The $(N \times N)$ identity matrix is denoted by $\mathbf{I}_N$. The permutation matrix, $\boldsymbol{U}_{PN}$ of size $(PN \times PN)$, is defined by [40]: $\boldsymbol{U}_{PN} = \sum_{p=1}^{P} \sum_{n=1}^{N} \boldsymbol{E}_{pn}^{(P\times N)} \otimes \boldsymbol{E}_{np}^{(N\times P)}$, where $\boldsymbol{E}_{pn}^{(P\times N)}$ is a $(P \times N)$ elementary matrix of zeros except the (p, n)-th component which is set to one. The trace of the square matrix $\boldsymbol{Z}$ is denoted by $\text{Tr}(\boldsymbol{Z})$. The N-dimensional vector of ones and the $(N \times P)$-dimensional matrix of ones are denoted by $\mathbf{1}_N$ and $\mathbf{1}_{N\times P}$, respectively. We denote: $\boldsymbol{A}^{\odot 2} = \boldsymbol{A} \odot \boldsymbol{A}$ and $\boldsymbol{B}^{\boxdot 2} = \boldsymbol{B} \boxdot \boldsymbol{B}$. Finally $|.|$, $\|.\|$ and $\|.\|_F$ stand for the absolute value, the Euclidean norm and the Frobenius norm, respectively.

4. Preliminaries and problem formulation

This section is devoted to some basic definitions in tri-linear algebra which are required for the problem formulation. Further related definitions can be found in [41, 42, 43, 44, 45, 4].

Definition 1. *The outer product $\boldsymbol{\mathcal{T}} = \boldsymbol{u}^{(1)} \circ \boldsymbol{u}^{(2)} \circ \boldsymbol{u}^{(3)}$ of three vectors $\boldsymbol{u}^{(1)} \in \mathbb{R}^I$, $\boldsymbol{u}^{(2)} \in \mathbb{R}^J$ and $\boldsymbol{u}^{(3)} \in \mathbb{R}^K$ is a third order array of $\mathbb{R}^{I\times J\times K}$ whose elements are defined by $\mathcal{T}_{i,j,k} = u_i^{(1)} u_j^{(2)} u_k^{(3)}$. This array represents a* decomposable *tensor.*

Despite the similarity between matrices and three-way arrays, they differ in a number of their properties [45]. For instance, contrary to the matrix case, the rank of three-way arrays can exceed its smallest dimension [46]. The rank of a three-way array always exists and is defined by:

Definition 2. *The rank of a third order array $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I\times J\times K}$, denoted by* rk($\boldsymbol{\mathcal{T}}$)*, is the minimal number R of decomposable third order arrays that exactly yield $\boldsymbol{\mathcal{T}}$ in a linear combination:*

$$\boldsymbol{\mathcal{T}} = \sum_{p=1}^{R} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p \tag{1}$$

with $\boldsymbol{a}_p \in \mathbb{R}^I$, $\boldsymbol{b}_p \in \mathbb{R}^J$ and $\boldsymbol{c}_p \in \mathbb{R}^K$.

Now, let us introduce the low-rank CP model [41, 44, 47, 4] of a third order array from definitions 1 and 2, as well as the INDSCAL model [48].

Definition 3. *For a given P, corresponding to the number of components, the CP model of a third order array $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I\times J\times K}$ can be expressed as:*

$$\boldsymbol{\mathcal{T}} = \sum_{p=1}^{P} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p + \boldsymbol{\mathcal{R}} \tag{2}$$

where each $\boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p$ is a rank-1 third order array and where the third order array $\boldsymbol{\mathcal{R}}$ represents the model residual. Sometimes we use the notation $\boldsymbol{\mathcal{T}}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ to refer to the third order array $\boldsymbol{\mathcal{T}}$ with its loading matrices $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_P]$, $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_P]$ and $\boldsymbol{C} = [\boldsymbol{c}_1, \cdots, \boldsymbol{c}_P]$.

In other words, the CP model defines a rank-P approximation of tensor $\boldsymbol{\mathcal{T}}$. If $P = rk(\boldsymbol{\mathcal{T}})$ then the model residual is null and the CP decomposition is exact.

Definition 4. *The INDSCAL model of a third order array $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I\times I\times K}$, symmetric in the two first modes, is a CP model of $\boldsymbol{\mathcal{T}}$ in which the two first loading matrices are equal.*

The problem we tackle in this paper is to fit the INDSCAL model of three-way arrays subject to the nonnegativity constraint on its two identical loading matrices. The latter is referred to as the semi-nonnegative INDSCAL model:

Problem 1. *Given a real tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I\times I\times K}$, symmetric in the two first modes, and a positive integer P, the semi-nonnegative INDSCAL model is defined as follows:*

$$\boldsymbol{\mathcal{T}} = \sum_{p=1}^{P} \boldsymbol{a}_p \circ \boldsymbol{a}_p \circ \boldsymbol{c}_p + \boldsymbol{\mathcal{R}} \tag{3}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_P]$ has nonnegative components.

Since matrix $\boldsymbol{C}$ is not necessarily nonnegative, the tensor $\boldsymbol{\mathcal{T}}$ can have both negative and positive components. As a way of treating the nonnegativity constraint in the model above, one can resort to the change of variable $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$ with $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_P] \in \mathbb{R}^{I \times P}$ as proposed in [49, 50] to solve the nonnegative matrix factorization problem. As a result, Problem 1 can be reformulated as an unconstrained problem:

Problem 2. *Given a real tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I \times I \times K}$ and an integer P, the semi-nonnegative INDSCAL model can be expressed as:*

$$\boldsymbol{\mathcal{T}} = \sum_{p=1}^{P} \boldsymbol{b}_p^{\boxdot 2} \circ \boldsymbol{b}_p^{\boxdot 2} \circ \boldsymbol{c}_p + \boldsymbol{\mathcal{R}} \tag{4}$$

It is often convenient to represent three-way arrays as matrices. Among possible matrix representations, the one associated with mode-3 is defined such as the (i, j, k)-th component of $\boldsymbol{\mathcal{T}}$ corresponds to the $(k, i + (j-1)I)$-th element in $\boldsymbol{T}^{(3)}$. Then Problem 2 can be reformulated in a matrix form:

Definition 5. *Let $\boldsymbol{\mathcal{T}}(\boldsymbol{A}, \boldsymbol{A}, \boldsymbol{C})$ be a 3-rd order array of size $(I \times I \times K)$. Then a $(K \times I^2)$ matricization of $\boldsymbol{\mathcal{T}}$ denoted by the array, $\boldsymbol{T}^{(3)} = \boldsymbol{C}\,(\boldsymbol{A}^{\odot 2})^{\intercal} + \boldsymbol{R}^{(3)}$, associated with the 3-rd mode can be achieved by side-by-side stacking of the lateral slices of $\boldsymbol{\mathcal{T}}$ with $\boldsymbol{A} = \boldsymbol{B} \boxdot \boldsymbol{B}$, $\boldsymbol{B} \in \mathbb{R}^{I \times P}$ and $\boldsymbol{C} \in \mathbb{R}^{K \times P}$; $\boldsymbol{R}^{(3)}$ is the mode-3 matricization of the residual.*

The semi-nonnegative INDSCAL model (4) is typically fitted according to a least squares criterion, although others have been proposed [51]. The unconstrained optimization problem consists in the minimization of the following objective function:

$$\Psi(\boldsymbol{B}, \boldsymbol{C}) = \left\| \boldsymbol{T}^{(3)} - \boldsymbol{C} \left(\left(\boldsymbol{B}^{\boxdot 2} \right)^{\odot 2} \right)^{\intercal} \right\|_F^2 \tag{5}$$

Several solutions are proposed, which belong to two fundamental strategies of optimization: line search and trust region. Regarding the former, we present two approaches based on the use of the first and the second order derivatives of Ψ: CG and Newton-like methods are combined with a search for optimal stepsizes. For the second strategy, an LM method is proposed. In the next section, both strategies are detailed and each descent optimization procedure is described in section 7.

5. Two optimization strategies

5.1. Line search strategy

The line search strategy combines the computation of a descent direction given by an optimization procedure with the search for stepsizes minimizing the objective function along that direction. The global plane search selects two directions, and actually permits to sometimes avoid local minima. At iteration *it*, the update of matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ in (4) is defined by the following rule:

$$\begin{bmatrix} \mathrm{vec}(\boldsymbol{B}^{it+1}) \\ \mathrm{vec}(\boldsymbol{C}^{it+1}) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(\boldsymbol{B}^{it}) \\ \mathrm{vec}(\boldsymbol{C}^{it}) \end{bmatrix} + \begin{bmatrix} \mu_{\boldsymbol{B}}^{it}\, \mathbf{I}_{IP} & \mathbf{0} \\ \mathbf{0} & \mu_{\boldsymbol{C}}^{it}\, \mathbf{I}_{KP} \end{bmatrix} \begin{bmatrix} \mathrm{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it}) \\ \mathrm{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it}) \end{bmatrix}$$

where $\boldsymbol{G_B}$ and $\boldsymbol{G_C}$ are two search directions. Both stepsizes $\mu_{\boldsymbol{B}}^{it}$ and $\mu_{\boldsymbol{C}}^{it}$ can be computed by minimizing the following function:

$$\varphi_{\text{2steps}}(\mu_{\boldsymbol{B}}, \mu_{\boldsymbol{C}}) = \left\| \boldsymbol{T}^{(3)} - (\boldsymbol{C}^{it} + \mu_{\boldsymbol{C}} \boldsymbol{G}_{\boldsymbol{C}}^{it})(((\boldsymbol{B}^{it} + \mu_{\boldsymbol{B}} \boldsymbol{G}_{\boldsymbol{B}}^{it})^{\boxdot 2})^{\odot 2})^{\intercal} \right\|_F^2 \tag{6}$$

w.r.t. $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$. This can be done approximately or exactly [47, 52, 53, 54, 55]. Because of its higher numerical complexity, the authors proposed in [47] to consider the same stepsize μ for all loading matrices. Due to the satisfactory results they provided, we shall not only address the case of two independent stepsizes, but also consider in the sequel a simpler plane search where $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}} = \mu$. For the sake of clarity superscript *it* will be omitted in the rest of this section.

5.1.1. Global plane search with two stepsizes

The problem of global plane search with two stepsizes is defined as follows:

Problem 3. *Given two search directions $\boldsymbol{G_B}$ and $\boldsymbol{G_C}$, find the optimal stepsizes $(\mu_{\boldsymbol{B}}, \mu_{\boldsymbol{C}})$ by minimizing criterion (6).*

One can show that criterion (6) can be rewritten as follows:

$$\begin{aligned} \varphi_{\text{2steps}}(\mu_{\boldsymbol{B}}, \mu_{\boldsymbol{C}}) = \Big\| & \boldsymbol{F}_0 + \boldsymbol{F}_1 \mu_{\boldsymbol{B}} + \boldsymbol{F}_2 \mu_{\boldsymbol{B}}^2 + \boldsymbol{F}_3 \mu_{\boldsymbol{B}}^3 + \boldsymbol{F}_4 \mu_{\boldsymbol{B}}^4 + \boldsymbol{F}_5 \mu_{\boldsymbol{C}} + \boldsymbol{F}_6 \mu_{\boldsymbol{C}} \mu_{\boldsymbol{B}} + \boldsymbol{F}_7 \mu_{\boldsymbol{C}} \mu_{\boldsymbol{B}}^2 + \\ & \boldsymbol{F}_8 \mu_{\boldsymbol{C}} \mu_{\boldsymbol{B}}^3 + \boldsymbol{F}_9 \mu_{\boldsymbol{C}} \mu_{\boldsymbol{B}}^4 \Big\|_F^2 \end{aligned} \tag{7}$$

where:

$$\begin{array}{llll} \boldsymbol{F}_0 = \boldsymbol{T}^{(3)} - \boldsymbol{C}\boldsymbol{E}_0 & \boldsymbol{F}_1 = -\boldsymbol{C}\boldsymbol{E}_1 & \boldsymbol{F}_2 = -\boldsymbol{C}\boldsymbol{E}_2 & \boldsymbol{F}_3 = -\boldsymbol{C}\boldsymbol{E}_3, \\ \boldsymbol{F}_4 = -\boldsymbol{C}\boldsymbol{E}_4 & \boldsymbol{F}_5 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_0 & \boldsymbol{F}_6 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_1 & \boldsymbol{F}_7 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_2 \\ \boldsymbol{F}_8 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_3 & \boldsymbol{F}_9 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_4 & & \end{array} \tag{8}$$

with:

$$\begin{aligned} \boldsymbol{E}_0 &= (\boldsymbol{K}_0^{\odot 2})^{\intercal} \quad \boldsymbol{E}_1 = (\boldsymbol{K}_0 \odot \boldsymbol{K}_1 + \boldsymbol{K}_1 \odot \boldsymbol{K}_0)^{\intercal} \quad \boldsymbol{E}_2 = (\boldsymbol{K}_0 \odot \boldsymbol{K}_2 + \boldsymbol{K}_1^{\odot 2} + \boldsymbol{K}_2 \odot \boldsymbol{K}_0)^{\intercal} \\ \boldsymbol{E}_3 &= (\boldsymbol{K}_1 \odot \boldsymbol{K}_2 + \boldsymbol{K}_2 \odot \boldsymbol{K}_1)^{\intercal} \quad \boldsymbol{E}_4 = (\boldsymbol{K}_2^{\odot 2})^{\intercal} \end{aligned} \tag{9}$$

and:

$$\boldsymbol{K}_0 = \boldsymbol{A} = \boldsymbol{B}^{\boxdot 2} \qquad \boldsymbol{K}_1 = \boldsymbol{B} \boxdot \boldsymbol{G}_{\boldsymbol{B}} + \boldsymbol{G}_{\boldsymbol{B}} \boxdot \boldsymbol{B} \qquad \boldsymbol{K}_2 = \boldsymbol{G}_{\boldsymbol{B}}^{\boxdot 2} \tag{10}$$

Note that equation (7) can be reduced to a compact form as follows:

$$\varphi_{\text{2steps}}(\mu_{\boldsymbol{B}}, \mu_{\boldsymbol{C}}) = \|\boldsymbol{F}\boldsymbol{u}\|_F^2 = \boldsymbol{u}^{\intercal} \boldsymbol{F}^{\intercal} \boldsymbol{F} \boldsymbol{u} = \boldsymbol{u}^{\intercal} \boldsymbol{Q} \boldsymbol{u} \tag{11}$$

where $\boldsymbol{F} = [\text{vec}(\boldsymbol{F}_9), \text{vec}(\boldsymbol{F}_8), \text{vec}(\boldsymbol{F}_7), \text{vec}(\boldsymbol{F}_6), \text{vec}(\boldsymbol{F}_5), \text{vec}(\boldsymbol{F}_4), \text{vec}(\boldsymbol{F}_3), \text{vec}(\boldsymbol{F}_2), \text{vec}(\boldsymbol{F}_1), \text{vec}(\boldsymbol{F}_0)]$ is a $(I^2K \times 10)$ matrix and $\boldsymbol{u} = [\mu_{\boldsymbol{C}}\mu_{\boldsymbol{B}}^4, \mu_{\boldsymbol{C}}\mu_{\boldsymbol{B}}^3, \mu_{\boldsymbol{C}}\mu_{\boldsymbol{B}}^2, \mu_{\boldsymbol{C}}\mu_{\boldsymbol{B}}, \mu_{\boldsymbol{C}}, \mu_{\boldsymbol{B}}^4, \mu_{\boldsymbol{B}}^3, \mu_{\boldsymbol{B}}^2, \mu_{\boldsymbol{B}}, 1]^{\intercal}$ is a 10-dimensional vector and $\boldsymbol{Q}$ is a symmetric matrix. The latter can be computed from $\boldsymbol{F}$ using the expression $\boldsymbol{Q} = \boldsymbol{F}^{\intercal}\boldsymbol{F}$. However, in terms of numerical complexity, it is better to build $\boldsymbol{Q}$ differently. Indeed, $\boldsymbol{Q}$ is a symmetric matrix only of size (10×10) unlike $\boldsymbol{F}$, which requires a large storage. Second, using the following Khatri-Rao property $(\boldsymbol{M} \odot \boldsymbol{N})^{\intercal}(\boldsymbol{M} \odot \boldsymbol{N}) = \boldsymbol{M}^{\intercal}\boldsymbol{M} \boxdot \boldsymbol{N}^{\intercal}\boldsymbol{N}$ and (A.3) [40], the expression of components of $\boldsymbol{Q}$, except those belonging to the last row and column depending on $\boldsymbol{T}^{(3)}$, can be simplified. Initially, we have $Q_{ij} = \text{vec}(\boldsymbol{F}_{10-i})^{\intercal}\text{vec}\Big(\boldsymbol{F}_{10-j}\Big)$, for $0 \leq i, j \leq 9$. First, using the previous property (A.3), both vectors are replaced by a Khatri-Rao

product of matrices. And second, Khatri-Rao products are replaced by a Hadamard product. For instance, we get:

$$\begin{aligned} Q_{1,5} = Q_{5,1} &= \mathrm{vec}(\boldsymbol{G}_c(\boldsymbol{K}_0 \odot \boldsymbol{K}_0)^\intercal)^\intercal \mathrm{vec}(\boldsymbol{C}(\boldsymbol{K}_2 \odot \boldsymbol{K}_2)^\intercal) \quad (12) \\ &= \mathbf{1}_p{}^\intercal((\boldsymbol{K}_0 \odot \boldsymbol{K}_0) \odot \boldsymbol{G}_c))^\intercal((\boldsymbol{K}_2 \odot \boldsymbol{K}_2) \odot \boldsymbol{C})\mathbf{1}_p \\ &= \mathbf{1}_p{}^\intercal((\boldsymbol{K}_0 \odot \boldsymbol{K}_0)^\intercal(\boldsymbol{K}_2 \odot \boldsymbol{K}_2) \boxdot \boldsymbol{G}_c{}^\intercal\boldsymbol{C})\mathbf{1}_p \\ &= \mathbf{1}_p{}^\intercal(\boldsymbol{K}_2{}^\intercal\boldsymbol{K}_0 \boxdot \boldsymbol{K}_2{}^\intercal\boldsymbol{K}_0 \boxdot \boldsymbol{G}_c{}^\intercal\boldsymbol{C})\mathbf{1}_p \quad (13) \end{aligned}$$

Note that this way of simplifying the matrix trace was originally proposed by Tomasi and Bro [56]. The same technique is used for the other components except those depending on $\boldsymbol{T}^{(3)}$ and by which the numerical complexity is governed.

As it was expected after a simple glance at (6), equation (11) shows that the objective function $\varphi_{2\text{steps}}$ is a second degree polynomial in $\mu_{\boldsymbol{C}}$. Thus, the optimal stepsize $\mu_{\boldsymbol{C}}$ is a rational function in $\mu_{\boldsymbol{B}}$. Now, once $\mu_{\boldsymbol{C}}$ is computed, its expression is injected in the equation $\partial\varphi_{2\text{steps}}/\partial\mu_{\boldsymbol{B}} = 0$, which is a 24-th degree polynomial in $\mu_{\boldsymbol{B}}$. Consequently, the global minimum $\mu_{\boldsymbol{B}}$ can be obtained by finding the roots of the numerator and selecting the root yielding the smallest value of the objective function $\varphi_{2\text{steps}}$. It is noteworthy that the computation of the coefficients of the 24-th degree polynomial and more particularly the elements of matrix $\boldsymbol{Q}$ is dominant in terms of numerical complexity, compared to the computation of the roots of this polynomial.

5.1.2. Global search with one stepsize

To reduce the numerical complexity, a unique stepsize $\mu = \mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ for $\boldsymbol{B}$ and $\boldsymbol{C}$ may be calculated by minimizing the following function derived from (6):

$$\varphi_{1\text{step}}(\mu) = \left\|\boldsymbol{T}^{(3)} - (\boldsymbol{C} + \mu\boldsymbol{G}_{\boldsymbol{C}})(((\boldsymbol{B} + \mu\boldsymbol{G}_{\boldsymbol{B}})^{\boxdot 2})^{\odot 2})^\intercal\right\|_F^2 \quad (14)$$

w.r.t. μ. Then the problem of global search with a unique stepsize is defined as follows:

Problem 4. *Given two search directions $\boldsymbol{G}_{\boldsymbol{B}}$ and $\boldsymbol{G}_{\boldsymbol{C}}$, find the optimal stepsize $\mu = \mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$, by minimizing function (14).*

$\varphi_{1\text{step}}$ can be rewritten as follows:

$$\varphi_{1step} = \left\|\boldsymbol{F}_0 + \mu\boldsymbol{F}_1 + \mu^2\boldsymbol{F}_2 + \mu^3\boldsymbol{F}_3 + \mu^4\boldsymbol{F}_4 + \mu^5\boldsymbol{F}_5\right\|_F^2 \quad (15)$$

where:

$$\begin{array}{lll} \boldsymbol{F}_0 = \boldsymbol{T}^{(3)} - \boldsymbol{C}\boldsymbol{E}_0 & \boldsymbol{F}_1 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_0 - \boldsymbol{C}\boldsymbol{E}_1 & \boldsymbol{F}_2 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_1 - \boldsymbol{C}\boldsymbol{E}_2 \\ \boldsymbol{F}_3 = -\boldsymbol{C}\boldsymbol{E}_3 - \boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_2 & \boldsymbol{F}_4 = -\boldsymbol{C}\boldsymbol{E}_4 - \boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_3 & \boldsymbol{F}_5 = -\boldsymbol{G}_{\boldsymbol{C}}\boldsymbol{E}_4 \end{array} \quad (16)$$

with the five matrices $\boldsymbol{E}_0$, ..., $\boldsymbol{E}_4$ given by equations (9) and (10). Note that equation (15) can be reduced to a compact form as follows:

$$\varphi_{1step}(\mu) = \|\boldsymbol{F}\boldsymbol{u}\|_F^2 = \boldsymbol{u}^\intercal\boldsymbol{F}^\intercal\boldsymbol{F}\boldsymbol{u} = \boldsymbol{u}^\intercal\boldsymbol{Q}\boldsymbol{u} \quad (17)$$

where $\boldsymbol{F} = [\mathrm{vec}(\boldsymbol{F}_5), \mathrm{vec}(\boldsymbol{F}_4), \mathrm{vec}(\boldsymbol{F}_3), \mathrm{vec}(\boldsymbol{F}_2), \mathrm{vec}(\boldsymbol{F}_1), \mathrm{vec}(\boldsymbol{F}_0)]$ is a $(I^2K \times 6)$ matrix, $\boldsymbol{u} = [\mu^5, \mu^4, \mu^3, \mu^2, \mu, 1]^\intercal$ is a 6-dimensional vector and $\boldsymbol{Q}$ is a (6×6) symmetric matrix. $\boldsymbol{Q}$ can be computed following the low cost strategy proposed in section 5.1.1. Then we calculate the globally optimal stepsize μ of the function $\varphi_{1\text{step}}$ by finding the roots of its derivative. This alternative solution is generally less efficient compared to the procedure proposed in section 5.1.1, as shown in section 9 through computer results. However, it permits to reduce the objective function to a 10-th degree polynomial in μ, and consequently to reduce the numerical complexity per iteration.

5.2. Trust region strategy

The second strategy we consider is known as the trust region method. The search direction and the step length are computed simultaneously. The method is based on an approximation of the cost function (5), which is considered to be accurate enough inside a ball with sufficiently small radius. We shall resort to the Levenberg-Marquardt method, which uses an approximation of the Hessian, combined with an estimation of a damping parameter proposed by Madsen et al. [57].

6. Matrix derivatives

This section is devoted to the computation of the gradient, the Jacobian and the Hessian of the objective function (5) in a compact matrix form [40, 58, 59]. In order to preserve the matrix form during the operations of derivation, matrix calculus is adopted. All useful formulas are given in Appendix A.

6.1. Gradient computation

The differential of Ψ, seen as a scalar function of two real-valued matrices $\boldsymbol{B}$ and $\boldsymbol{C}$, is given by:

$$\mathrm{d}\Psi(\boldsymbol{B},\boldsymbol{C}) = D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{B}) + D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{C}) \tag{18}$$

where $D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C}) = \partial\Psi/\partial\mathrm{vec}(\boldsymbol{B})^\intercal$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C}) = \partial\Psi/\partial\mathrm{vec}(\boldsymbol{C})^\intercal$ represent the partial derivatives of Ψ w.r.t. $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively, and $dvec(\boldsymbol{B})$ and $dvec(\boldsymbol{C})$ denote the differential of $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively, stored in column vector format. Magnus and Neudecker [60] argued in favour of vector representations of matrix derivatives, which are used in this paper. Then, according to expression (18), the first order derivative of Ψ w.r.t. $\boldsymbol{B}$ and $\boldsymbol{C}$, called $D\Psi(\boldsymbol{B},\boldsymbol{C})$, can be partitioned as follows [60]:

$$D\Psi(\boldsymbol{B},\boldsymbol{C}) = [D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C}), D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})] \tag{19}$$

A compact matrix computation of both partial gradients $D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ is given hereafter. From equation (5), we have:

$$\Psi(\boldsymbol{B},\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{T}^{(3)}) - 2f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + g(\boldsymbol{B},\boldsymbol{C}) \tag{20}$$

where $f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\boldsymbol{Z}^\intercal)$ and $g(\boldsymbol{B},\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{Z}\xi\boldsymbol{Z}^\intercal)$ with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$ and $\xi = \boldsymbol{C}^\intercal\boldsymbol{C}$. Using equation (A.3) and the property of the column-wise Khatri-Rao product (A.14), the expression of $g(\boldsymbol{B},\boldsymbol{C})$ is reformulated, providing us with a simpler expression of derivatives: $g(\boldsymbol{B},\boldsymbol{C}) = \mathbf{1}_p{}^\intercal((\boldsymbol{A}^\intercal\boldsymbol{A})^{\boxdot 2}\boxdot\boldsymbol{C}^\intercal\boldsymbol{C})\mathbf{1}_p$. Then we get, using chain rules [58] and differential properties:

$$\begin{aligned}
\mathrm{d}\Psi(\boldsymbol{B},\boldsymbol{C}) &= -2\mathrm{d}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + \mathrm{d}g(\boldsymbol{B},\boldsymbol{C}) \\
&= (-2D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{Z}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{Z}) + \\
&\qquad (-2D_{\boldsymbol{C}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{C}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{C}) \\
&= (-2D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{Z}}g(\boldsymbol{B},\boldsymbol{C}))\,D_{\boldsymbol{A}}\boldsymbol{Z}\,\mathrm{dvec}(\boldsymbol{A}) + \\
&\qquad (-2D_{\boldsymbol{C}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{C}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{C}) \\
&= (-2D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{Z}}g(\boldsymbol{B},\boldsymbol{C}))\,D_{\boldsymbol{A}}\boldsymbol{Z}\,D_{\boldsymbol{B}}\boldsymbol{A}\,\mathrm{dvec}(\boldsymbol{B}) + \\
&\qquad (-2D_{\boldsymbol{C}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{C}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{C}) \\
&= (-2D_{\boldsymbol{B}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{B}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{B}) + \\
&\qquad (-2D_{\boldsymbol{C}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{C}}g(\boldsymbol{B},\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{C})
\end{aligned} \tag{21}$$

with:

$$\mathrm{dvec}(\boldsymbol{Z}) = D_{\boldsymbol{A}}\boldsymbol{Z}\,\mathrm{dvec}(\boldsymbol{A}) \qquad \mathrm{dvec}(\boldsymbol{A}) = D_{\boldsymbol{B}}\boldsymbol{A}\,\mathrm{dvec}(\boldsymbol{B})$$
$$D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = D_{\boldsymbol{Z}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\, D_{\boldsymbol{A}}\boldsymbol{Z}\, D_{\boldsymbol{B}}\boldsymbol{A}$$
$$D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) = D_{\boldsymbol{Z}} g(\boldsymbol{B},\boldsymbol{C})\, D_{\boldsymbol{A}}\boldsymbol{Z}\, D_{\boldsymbol{B}}\boldsymbol{A}$$

The expression of $D_{\boldsymbol{A}}\boldsymbol{Z}$ and $D_{\boldsymbol{B}}\boldsymbol{A}$ required during the differentiation process are given in the two following lemmas:

Lemma 1. *The gradient of $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$ w.r.t. $\boldsymbol{A}$ is given by:*

$$\begin{aligned} D_{\boldsymbol{A}}\boldsymbol{Z} &= \partial\mathrm{vec}(\boldsymbol{Z})\,/\partial\mathrm{vec}(\boldsymbol{A})^{\intercal} \\ &= \mathrm{diag}(\mathrm{vec}(\mathbf{1}_I \otimes \boldsymbol{A}))(\mathbf{I}_{IP} \otimes \mathbf{1}_I) + \mathrm{diag}(\mathrm{vec}(\boldsymbol{A}) \otimes \mathbf{1}_I)(\boldsymbol{U}_{IP} \otimes \mathbf{I}_I)(\mathbf{1}_I \otimes \mathbf{I}_{IP}) \end{aligned} \tag{22}$$

See Appendix B for a proof.

Lemma 2. *The gradient of $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$ w.r.t. $\boldsymbol{B}$ is given by:*

$$D_{\boldsymbol{B}}\boldsymbol{A} = \partial\mathrm{vec}(\boldsymbol{A})\,/\partial\mathrm{vec}(\boldsymbol{B})^{\intercal} = 2\mathrm{diag}(\mathrm{vec}(\boldsymbol{B})) \tag{23}$$

The proof is given in Appendix C. According to equalities (19) and (21), gradient $D\Psi(\boldsymbol{B},\boldsymbol{C})$ may be obtained as stated in the following lemma.

Lemma 3. *The gradient $D\Psi(\boldsymbol{B},\boldsymbol{C}) = [D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C}), D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})]$ is defined as:*

$$D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C}) = -2D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) \tag{24}$$
$$D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C}) = -2D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}) \tag{25}$$

where:

$$D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = 2\left(\mathrm{vec}(\boldsymbol{B}) \boxdot (((\boldsymbol{A} \otimes \mathbf{1}_I{}^{\intercal}) \boxdot (\boldsymbol{M} + \boldsymbol{N}))^{\intercal}\mathbf{1}_I)\right)^{\intercal} \tag{26}$$
$$D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) = 4\mathrm{vec}(\boldsymbol{A}^{\intercal}\boldsymbol{A} \boxdot \boldsymbol{C}^{\intercal}\boldsymbol{C})^{\intercal}((\mathbf{I}_P \otimes \boldsymbol{A}^{\intercal}) + (\boldsymbol{A}^{\intercal} \otimes \mathbf{I}_P)\boldsymbol{U}_{IP})\mathrm{diag}(\mathrm{vec}(\boldsymbol{B})) \tag{27}$$

and:

$$D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \mathrm{vec}\left(\boldsymbol{T}^{(3)}\boldsymbol{Z}\right)^{\intercal} \tag{28}$$
$$D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}) = 2\mathrm{vec}(\boldsymbol{C})^{\intercal}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z} \otimes \mathbf{I}_K) \tag{29}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$, $\boldsymbol{M} = \mathrm{Mat}^{(I\times 1, IP)}(\mathrm{vec}\left(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\right))$ and $\boldsymbol{N} = \mathrm{Mat}^{(I\times I, P)}(\boldsymbol{M}^{\intercal})$.

The proof is given in Appendix D.

6.2. Jacobian Computation

To compute the Jacobian matrix, a vector function $\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C}) = \mathrm{vec}(\boldsymbol{T}^{(3)} - \boldsymbol{C}((\boldsymbol{B}^{\boxdot 2})^{\odot 2})^{\intercal})$ is introduced. The Jacobian matrix for $\boldsymbol{h}$ w.r.t. $\boldsymbol{B}$ and $\boldsymbol{C}$ is obtained by computing the differential of $\boldsymbol{h}$.

$$\begin{aligned} \mathrm{d}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C}) &= \mathrm{d}(\mathrm{vec}(\boldsymbol{T}^{(3)} - \boldsymbol{C}((\boldsymbol{B}^{\boxdot 2})^{\odot 2})^{\intercal})) \\ &= D_{\boldsymbol{B}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})\mathrm{d}(\mathrm{vec}(\boldsymbol{B})) + D_{\boldsymbol{C}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})\mathrm{d}(\mathrm{vec}(\boldsymbol{C})) \end{aligned} \tag{30}$$

where $D_{\boldsymbol{B}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C}) = \partial\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})/\partial\mathrm{vec}(\boldsymbol{B})^{\intercal}$ and $D_{\boldsymbol{C}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C}) = \partial\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})/\partial\mathrm{vec}(\boldsymbol{C})^{\intercal}$. Then we can write the Jacobian $\boldsymbol{J}$ as $\boldsymbol{J} = [\boldsymbol{J}_{\boldsymbol{B}}, \boldsymbol{J}_{\boldsymbol{C}}]$, say the row concatenation of both partial Jacobians defined by $\boldsymbol{J}_{\boldsymbol{B}} = D_{\boldsymbol{B}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})$ and $\boldsymbol{J}_{\boldsymbol{C}} = D_{\boldsymbol{C}}\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C})$.

Lemma 4. *The Jacobian* $\boldsymbol{J} = [\boldsymbol{J_B}, \boldsymbol{J_C}]$ *of the vector function* $\boldsymbol{h}$ *is given by:*

$$\begin{aligned} J_{\boldsymbol{B}} &= -2(\mathbf{I}_{I^2} \otimes \boldsymbol{C})\boldsymbol{U}_{I^2P}[\text{diag}(\text{vec}(\boldsymbol{A} \otimes \mathbf{I}_I))(\mathbf{I}_{IP} \otimes \mathbf{1}_I) + \text{diag}(\text{vec}(\boldsymbol{A}) \otimes \mathbf{1}_I) \\ &\quad (\boldsymbol{U}_{PI} \otimes \mathbf{I}_N)(\mathbf{1}_I \otimes \mathbf{I}_{IP})]\text{diag}(\text{vec}(\boldsymbol{B})) \qquad (31) \\ J_{\boldsymbol{C}} &= -\boldsymbol{Z} \otimes \mathbf{I}_K \qquad (32) \end{aligned}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$ *and* $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$.

A proof is given in Appendix E. Because of the large size of the Jacobian $(I^2K, (I + K)P)$, the normal equations are used to compute the update of the Levenberg-Marquardt method. The much smaller $\boldsymbol{J}^\intercal\boldsymbol{J}$ matrix is calculated directly by rearranging its blocks to obtain simplified expressions involving fewer terms of smaller dimensions than if $\boldsymbol{J}$ were used explicitly.

Lemma 5. *The symmetric matrix* $\boldsymbol{J}^\intercal\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_B{}^\intercal\boldsymbol{J}_B & \boldsymbol{J}_B{}^\intercal\boldsymbol{J}_C \\ \boldsymbol{J}_C{}^\intercal\boldsymbol{J}_B & \boldsymbol{J}_C{}^\intercal\boldsymbol{J}_C \end{pmatrix}$ *of the vector function* $\boldsymbol{h}$ *is given by:*

$$\begin{aligned} \boldsymbol{J_B}{}^\intercal\boldsymbol{J_B} &= 4[(\text{vec}(\boldsymbol{B}) \otimes \mathbf{1}_{IP}{}^\intercal) \boxdot (\mathbf{1}_{IP} \otimes \text{vec}(\boldsymbol{B})^\intercal)] \boxdot (\boldsymbol{C}^\intercal\boldsymbol{C} \otimes \mathbf{1}_{I\times I}) \boxdot \qquad (33) \\ &\quad [2\boldsymbol{A}^\intercal\boldsymbol{A} \otimes \mathbf{I}_I + (2\mathbf{1}_P{}^\intercal \otimes (\boldsymbol{A}^\intercal \odot \mathbf{1}_{I\times I})) \boxdot (\mathbf{1}_{P\times IP} \odot (\boldsymbol{A} \otimes \mathbf{1}_I{}^\intercal))] \\ \boldsymbol{J_C}{}^\intercal\boldsymbol{J_C} &= ((\boldsymbol{A}^\intercal\boldsymbol{A})^{\boxdot 2} \otimes \mathbf{I}_K) \qquad (34) \\ \boldsymbol{J_B}{}^\intercal\boldsymbol{J_C} &= 4(\text{vec}(\boldsymbol{B}) \otimes \mathbf{1}_{KP}{}^\intercal))^\intercal \boxdot (\mathbf{1}_{P\times IP} \odot (\boldsymbol{C} \otimes \mathbf{1}_I{}^\intercal)) \boxdot ((\boldsymbol{Z}^\intercal(\mathbf{I}_I \otimes \boldsymbol{A})\boldsymbol{U}_{IP})^\intercal \otimes \mathbf{1}_K{}^\intercal)^\intercal \qquad (35) \\ \boldsymbol{J_C}{}^\intercal\boldsymbol{J_B} &= \boldsymbol{J_B}{}^\intercal\boldsymbol{J_C} \qquad (36) \end{aligned}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$

A proof is given in Appendix F.

6.3. Hessian computation

The computation of the Hessian w.r.t. to variables $\boldsymbol{B}$ and $\boldsymbol{C}$ is also given in a compact matrix form following the differentiation technique used in sections 6.1 and 6.2. The Hessian $D^2\Psi(\boldsymbol{B}, \boldsymbol{C})$ can be partitioned in 4 blocks:

$$D^2\Psi(\boldsymbol{B}, \boldsymbol{C}) = \begin{pmatrix} D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}) & D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C}) \\ D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}) & D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C}) \end{pmatrix} \qquad (37)$$

where $D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}) = D_{\boldsymbol{B}}(D_{\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}))^\intercal$ and $D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C}) = D_{\boldsymbol{C}}(D_{\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C}))^\intercal$ denote the second order partial derivatives w.r.t. $\boldsymbol{B}$ and $\boldsymbol{C}$, respectively; and where $D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C})$ and $D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C})$ represent the crossed partial derivatives such that $(D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C}))^\intercal = D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C})$.

The expression of each block is calculated separately and given in the sequel. Firstly, we compute $D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C})$ from equations (24), (26) and (27):

Lemma 6. *The partial Hessian* $D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C})$ *is given by:*

$$D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}) = -2D^2_{\boldsymbol{B},\boldsymbol{B}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B}, \boldsymbol{C}) + D^2_{\boldsymbol{B},\boldsymbol{B}}g(\boldsymbol{B}, \boldsymbol{C}) \qquad (38)$$

where:

$$\begin{aligned} D^2_{\boldsymbol{B},\boldsymbol{B}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B}, \boldsymbol{C}) &= 2([(\boldsymbol{A}^\intercal \otimes \mathbf{1}_I) \boxdot (\boldsymbol{M} + \boldsymbol{N})^\intercal]^{1/2}[(\boldsymbol{A} \otimes \mathbf{1}_I{}^\intercal) \boxdot (\boldsymbol{M} + \boldsymbol{N})]^{1/2}) \boxdot \mathbf{I}_{IP} \\ &\quad +4((\mathbf{1}_P{}^\intercal \otimes (\boldsymbol{M} + \boldsymbol{N})^\intercal) \boxdot (\text{diag}(\text{vec}(B))(\mathbf{I}_P \otimes \mathbf{1}_{I\times I})))\text{diag}(\text{vec}(B)) \qquad (39) \end{aligned}$$

and:

$$\begin{aligned} D^2_{\boldsymbol{B},\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) \quad = \quad & 4\text{diag}(((\mathbf{I}_P \otimes \boldsymbol{A}) + \boldsymbol{U}_{PI}(\boldsymbol{A} \otimes \mathbf{I}_P))\text{vec}(\boldsymbol{A}^\intercal \boldsymbol{A} \boxdot \boldsymbol{C}^\intercal \boldsymbol{C})) + 8\text{diag}(\text{vec}(\boldsymbol{B})) \times \\ & ((\mathbf{I}_P \otimes \boldsymbol{A}) + \boldsymbol{U}_{PI}(\boldsymbol{A} \otimes \mathbf{I}_P))\text{diag}(\text{vec}(\boldsymbol{C}^\intercal \boldsymbol{C}))((\mathbf{I}_P \otimes \boldsymbol{A}^\intercal) + (\boldsymbol{A}^\intercal \otimes \mathbf{I}_P)\boldsymbol{U}_{IP})\text{diag}(\text{vec}(\boldsymbol{B})) + \\ & +16\text{diag}(\text{vec}(\boldsymbol{B}))((\boldsymbol{A}^\intercal \boldsymbol{A} \boxdot \boldsymbol{C}^\intercal \boldsymbol{C}) \otimes \mathbf{I}_I)\text{diag}(\text{vec}(\boldsymbol{B})) \end{aligned} \tag{40}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$, $\boldsymbol{M} = \text{Mat}^{(I\times 1,NP)}(\text{vec}\left(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\right))$ *and* $\boldsymbol{N} = \text{Mat}^{(I\times I,P)}(\boldsymbol{M}^\intercal)$.

The proof is given in Appendix G. Secondly, we derive the second diagonal block of the Hessian, $D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$, from equations (25), (28) and (29):

Lemma 7. *The partial Hessian of* Ψ (5) *w.r.t.* $\boldsymbol{C}$ *is given by:*

$$D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C}) = 2((\boldsymbol{A}^\intercal \boldsymbol{A})^{\boxdot 2} \otimes \mathbf{I}_K) \tag{41}$$

with $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$ *and* $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$.

Proof is given in Appendix H. Thirdly, the crossed partial derivatives $D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ and $D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$ can be derived from equations (25), (28) and (29).

Lemma 8. *The partial Hessians* $D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ *and* $D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$ *are given by:*

$$D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C}) = D^2_{\boldsymbol{C},\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D^2_{\boldsymbol{C},\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) = (D^2_{\boldsymbol{B},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C}))^\intercal \tag{42}$$

where:

$$\begin{aligned} D^2_{\boldsymbol{C},\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) \quad = \quad & -4\text{diag}(\text{vec}(\boldsymbol{B}))(((\mathbf{I}_P \otimes \mathbf{1}_K{}^\intercal) \otimes \mathbf{1}_K) \boxdot ((((\mathbf{I}_I \otimes \boldsymbol{A})\boldsymbol{U}_{IP}) + \\ & (\boldsymbol{A} \otimes \mathbf{I}_I)^\intercal)(\mathbf{1}_P{}^\intercal \otimes \boldsymbol{T}^{(3)\intercal}))) \end{aligned} \tag{43}$$

and:

$$\begin{aligned} D^2_{\boldsymbol{C},\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}) \quad = \quad & 4\text{diag}(\text{vec}(\boldsymbol{B}))((\mathbf{I}_P \otimes \boldsymbol{A}) + \boldsymbol{U}_{PI}(\boldsymbol{A} \otimes \mathbf{I}_P))\text{diag}(\text{vec}(\boldsymbol{A}^\intercal \boldsymbol{A})) \\ & ((\mathbf{I}_P \otimes \boldsymbol{C}^\intercal) + (\boldsymbol{C}^\intercal \otimes \mathbf{I}_P)\boldsymbol{U}_{KP}) \end{aligned} \tag{44}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$.

The proof may be found in Appendix I.

7. The algorithms

This section is devoted to a brief description of some derivative-based optimization algorithms, used to solve problem 2.

7.1. Conjugate gradient method

The CG method is well-known and widely used in practice, particularly to solve large-scale nonlinear optimization problems. The main advantage of this algorithm resides in both its relatively small memory allocation and numerical complexity. Therefore it is considered to be faster than a simple gradient descent. A procedure of global plane search, described in section 5.1, is associated with this method, giving birth to the $\text{CG}_{1\text{step}}$ and $\text{CG}_{2\text{steps}}$ algorithms, depending on whether one or two different optimal stepsizes (as described in section 5.1) are used. Their pseudo-codes are described in Algorithm 1. Each direction is chosen to be a linear combination of the gradient and the previous direction. The update rules of $\boldsymbol{B}$ and $\boldsymbol{C}$ of the $\text{CG}_{1\text{step}}$ and $\text{CG}_{2\text{steps}}$ algorithms are given by:

$$\begin{bmatrix} \text{vec}(\boldsymbol{B}^{it+1}) \\ \text{vec}(\boldsymbol{C}^{it+1}) \end{bmatrix} = \begin{bmatrix} \text{vec}(\boldsymbol{B}^{it}) \\ \text{vec}(\boldsymbol{C}^{it}) \end{bmatrix} - \begin{bmatrix} \mu_{\boldsymbol{B}}^{it}\mathbf{I}_{IP} & \mathbf{0} \\ \mathbf{0} & \mu_{\boldsymbol{C}}^{it}\mathbf{I}_{KP} \end{bmatrix} \begin{bmatrix} \text{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it}) \\ \text{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it}) \end{bmatrix} \tag{45}$$

$$\begin{bmatrix} \text{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it}) \\ \text{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it}) \end{bmatrix} = -\begin{bmatrix} D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^\intercal \\ D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^\intercal \end{bmatrix} + \beta^{it-1} \begin{bmatrix} \text{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it-1}) \\ \text{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it-1}) \end{bmatrix} \tag{46}$$

where $\mu_{\boldsymbol{B}}^{it} = \mu_{\boldsymbol{C}}^{it}$ for $\text{CG}_{1\text{step}}$, and where β^{it-1} is computed using the Polak-Ribière formula:

$$\beta^{it-1} = \frac{(\boldsymbol{D}\Psi(\boldsymbol{B}^{it-1}, \boldsymbol{C}^{it-1}) - \boldsymbol{D}\Psi(\boldsymbol{B}^{it-2}, \boldsymbol{C}^{it-2}))\boldsymbol{D}\Psi(\boldsymbol{B}^{it-1}, \boldsymbol{C}^{it-1})^\intercal}{\boldsymbol{D}\Psi(\boldsymbol{B}^{it-2}, \boldsymbol{C}^{it-2})\boldsymbol{D}\Psi(\boldsymbol{B}^{it-2}, \boldsymbol{C}^{it-2})^\intercal} \tag{47}$$

with $\boldsymbol{D}\Psi(\boldsymbol{B}, \boldsymbol{C}) = [D_{\boldsymbol{B}}\Psi(\boldsymbol{B}, \boldsymbol{C}), D_{\boldsymbol{C}}\Psi(\boldsymbol{B}, \boldsymbol{C})]$. Several parameters β^{it-1} were proposed such as Fletcher-Reeves or Hestenes-Stiefel, to cite a few, but the Polak-Ribière parameter tends to be more robust and efficient [61]. The descent is initialized by a gradient direction. Another modification that is often used in nonlinear conjugate gradient procedures is to restart the iteration at every n steps by taking a steepest descent step. Restarting serves to periodically refresh the algorithm, erasing old information that may not be beneficial.

Algorithm 1 Pseudo-code for the $\text{CG}_{1\text{step}}$ and $\text{CG}_{2\text{steps}}$ algorithms

Initialize both matrices $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$.
Set $it = 1$, $n = I^2K$, the relative error err $= 1$ and set the maximal number *itmax* of allowed iterations and the stopping criterion threshold ϵ to a predefined value, respectively.
Calculate $D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^0, \boldsymbol{C}^0)$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^0, \boldsymbol{C}^0)$, and compute $\mu_{\boldsymbol{B}}^0$ and $\mu_{\boldsymbol{C}}^0$ if $\text{CG}_{2\text{steps}}$, according to section 5.1.1, else calculate $\mu_{\boldsymbol{B}}^0 = \mu_{\boldsymbol{C}}^0 = \mu^0$ according to section 5.1.2.
Compute $\boldsymbol{B}^1$ and $\boldsymbol{C}^1$ with a steepest descent step.
while err $> \epsilon$ or $it <$ *itmax* **do**
  Compute the gradients $D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ using lemma 3.
  If it is proportional to n, $\text{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it}) = D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^\intercal$ and $\text{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it}) = D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^\intercal$, else compute β^{it-1}, defined in (47), and update the direction $[\text{vec}(\boldsymbol{G}_{\boldsymbol{B}}^{it})^\intercal, \text{vec}(\boldsymbol{G}_{\boldsymbol{C}}^{it})^\intercal]^\intercal$ using (46).
  If $\text{CG}_{2\text{steps}}$, compute the two stepsizes $\mu_{\boldsymbol{B}}^{it}$ and $\mu_{\boldsymbol{C}}^{it}$ according to section 5.1.1, else compute $\mu_{\boldsymbol{B}}^{it} = \mu_{\boldsymbol{C}}^{it} = \mu^{it}$ according to section 5.1.2.
  Compute the matrices $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{it+1}$ using the update rule (45).
  Compute err $= |\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1}) - \Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})|/\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$.
  Set $it = it + 1$.
end while

7.2. Levenberg-Marquardt method

The LM procedure may be seen to belong to the family of trust region methods, and is based on an approximation of the Hessian by a function of the Jacobian of $\boldsymbol{h}$ (see section 6.2). It is based on solving the modified normal equations, where the so-called damping parameter λ permits to adjust the conditioning of the approximated Hessian in a trust region [57, 61]. We propose two algorithms: the first one called $\mathrm{LM}^{+}_{\mathrm{sym}}$, based on a simultaneous update of the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$ and the second one, named A-$\mathrm{LM}^{+}_{\mathrm{sym}}$, alternating between these two updates. The latter method permits to reduce the numerical complexity in comparison to the former, while keeping a good performance as described in sections 8 and 9. Then the vector update rule of the $\mathrm{LM}^{+}_{\mathrm{sym}}$ method is given by:

$$\begin{bmatrix} \mathrm{vec}\left(\boldsymbol{B}^{it+1}\right) \\ \mathrm{vec}\left(\boldsymbol{C}^{it+1}\right) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}\left(\boldsymbol{B}^{it}\right) \\ \mathrm{vec}\left(\boldsymbol{C}^{it}\right) \end{bmatrix} - \frac{1}{2}(\boldsymbol{J}^{\top}\boldsymbol{J} + \lambda^{it}\mathbf{I}_{P(I+K)})^{-1} \begin{bmatrix} D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\top} \\ D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\top} \end{bmatrix} \tag{48}$$

where:

$$(\boldsymbol{J}^{\top}\boldsymbol{J} + \lambda\mathbf{I}_{P(I+K)})^{-1} = \begin{pmatrix} \boldsymbol{J}_B{}^{\top}\boldsymbol{J}_B + \lambda\mathbf{I}_{IP} & \boldsymbol{J}_B{}^{\top}\boldsymbol{J}_C \\ \boldsymbol{J}_C{}^{\top}\boldsymbol{J}_B & \boldsymbol{J}_C{}^{\top}\boldsymbol{J}_C + \lambda\mathbf{I}_{KP} \end{pmatrix}^{-1} \tag{49}$$

where λ is the damping parameter adjusting the conditioning of the approximated Hessian, $\boldsymbol{J}^{\top}\boldsymbol{J}$, in the trust region [57].

Algorithm 2 Pseudo-code for the $\mathrm{LM}^{+}_{\mathrm{sym}}$ algorithm

Initialize matrices $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$.
Compute $D\Psi(\boldsymbol{B}^0, \boldsymbol{C}^0)$ (see lemma 3) and calculate $\boldsymbol{J}^{\top}\boldsymbol{J}$ (see lemma 5) at $(\boldsymbol{B}^0, \boldsymbol{C}^0)$.
Set $it = 0$, the relative error $\mathrm{err} = 1$, $\mu = 2$ and set the maximal number $itmax$ of allowed iterations and the stop criterion threshold ϵ to a predefined value, respectively.
while $\mathrm{err} > \epsilon$ or $it < itmax$ **do**
  Compute the matrices $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{it+1}$, based on a Cholesky decomposition, using the vector update rule (48).
  Compute ρ, corresponding to the gain ratio, which controls the update rule (48) [57, p.25].
  while $\rho < 0$ **do**
    $\lambda^{it} = \lambda^{it} * \mu$ and $\mu = 2 * \mu$
    Compute the matrices again $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{it+1}$, based on a Cholesky decomposition, using the vector update rule (48).
    Compute ρ.
  end while
  $\lambda^{it+1} = \lambda^{it} * \max(1/3, 1 - (2\rho - 1)^3)$ and $\mu = 2$
  Compute the gradient $D\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1})^{\top}$ according to lemma 3.
  Compute $\boldsymbol{J}^{\top}\boldsymbol{J}$, according to lemma 5.
  Compute $\mathrm{err} = |\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1}) - \Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})|/\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$.
  Set $it = it + 1$.
end while

The A-LM$^{+}_{\text{sym}}$ update rules are given by:

$$\text{vec}\Big(\boldsymbol{B}^{it+1}\Big) = \text{vec}\Big(\boldsymbol{B}^{it}\Big) - \frac{1}{2}(\boldsymbol{J}_B{}^{\top}\boldsymbol{J}_B + \lambda^{it}_{\boldsymbol{B}}\mathbf{I}_{IP})^{-1} D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\top} \tag{50}$$

$$\text{vec}\Big(\boldsymbol{C}^{it+1}\Big) = \text{vec}\Big(\boldsymbol{C}^{it}\Big) - \frac{1}{2}(\boldsymbol{J}_C{}^{\top}\boldsymbol{J}_C + \lambda^{it}_{\boldsymbol{C}}\mathbf{I}_{KP})^{-1} D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it})^{\top} \tag{51}$$

where $\lambda_{\boldsymbol{B}}$ and $\lambda_{\boldsymbol{C}}$ are the damping parameters, and where $\boldsymbol{J}_B{}^{\top}\boldsymbol{J}_B$ and $\boldsymbol{J}_C{}^{\top}\boldsymbol{J}_C$ are the diagonal blocks of $\boldsymbol{J}^{\top}\boldsymbol{J}$. Solving the normal equations leads to compute directly the blocks of $\boldsymbol{J}^{\top}\boldsymbol{J}$ without suffering from a Jacobian of potentially large size in several problems, yielding an enhancement in terms of numerical complexity and memory allocation. The update of both loading matrices is calculated by means of Cholesky decomposition of the well-conditioned approximated Hessian [62]. Then, $\boldsymbol{B}$ and $\boldsymbol{C}$ are each obtained by solving two triangular systems. The damping parameters λ are computed at each iteration using the technique of Madsen et al. [57].

Algorithm 3 Pseudo-code for the A-LM$^{+}_{\text{sym}}$ algorithm

Initialize matrices $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$.
Compute $D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^0, \boldsymbol{C}^0)$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^0, \boldsymbol{C}^0)$ (see lemma 3).
Calculate $\boldsymbol{J}_B{}^{\top}\boldsymbol{J}_B$, $\boldsymbol{J}_C{}^{\top}\boldsymbol{J}_C$ (see lemma 5) at $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$, respectively.
Set $it = 0$, the relative error err $= 1$, $\mu_B = 2$ and $\mu_C = 2$
Set the maximal number *itmax* of allowed iterations and the stop criterion threshold ϵ to a predefined value, respectively.
while err $> \epsilon$ or $it <$ *itmax* **do**
  Compute the matrices $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{i+1}$, based on a Cholesky decomposition, using the two update rules (50) and (51).
  Compute ρ_B and ρ_C, corresponding to the gain ratio, which control the update rules (50) and (51) [57, p.25].
  while $\rho_B < 0$ **do**
    $\lambda^{it}_B = \lambda^{it}_B * \mu_B$ and $\mu_B = 2 * \mu_B$
    Compute again the matrix $\boldsymbol{B}^{it+1}$ using the update rule (50).
    Compute ρ_B.
  end while
  $\lambda^{it+1}_B = \lambda^{it}_B * \max(1/3, 1 - (2\rho_B - 1)^3)$ and $\mu_B = 2$
  Compute the gradient $D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it})$, according to lemma 3.
  Compute $\boldsymbol{J}_C{}^{\top}\boldsymbol{J}_C$, according to lemma 5.
  while $\rho_C < 0$ **do**
    $\lambda^{it}_C = \lambda^{it}_C * \mu_C$ and $\mu_C = 2 * \mu_C$
    Compute again the matrix $\boldsymbol{C}^{it}$ using the equations (51).
    Compute ρ_C.
  end while
  $\lambda^{it+1}_C = \lambda^{it}_C * \max(1/3, 1 - (2\rho_C - 1)^3)$ and $\mu_C = 2$
  Compute the gradient $D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1})$, according to lemma 3.
  Compute $\boldsymbol{J}_B{}^{\top}\boldsymbol{J}_B$, according to lemma 5.
  Compute err $= |\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1}) - \Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})|/\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$.
  Set $it = it + 1$.
end while

Both methods have two main benefits. First, regularizing the approximated Hessian by adding a matrix proportional to the identity permits to ensure its positive definiteness, and hence the descent property. Indeed the positive definiteness of the approximated Hessian is compromised due to the rank deficiency of the Jacobian. In fact, a fundamental result in parameter identification is that the rank of the Jacobian cannot exceed the number of its free parameters. Yet in the case of INDSCAL decomposition, there are only $(I + K - 1)P$ free parameters while $\boldsymbol{J}$ has $(I+K)P$ columns. Consequently, the local identifiability of the INDSCAL model is ensured if $\boldsymbol{J}$ has exactly P null singular values, which is expected almost surely when $I^2K > (I + K)P$ [46]. Second, a full computation of the Hessian is avoided. Furthermore, the classical LM method is known for its robustness and stability against collinearity and overfactoring in case of small ranks and array dimensions [62]. Third, note that the A-LM$^{+}_{sym}$ method is equivalent to the ALS algorithm with regularization, in which the normal equations are solved using the Cholesky decomposition. This kind of approach allows us to avoid swamps [63]. A pseudo-code of these two methods are described separately in Algorithms 2 and 3.

7.3. Newton method

The Newton method exploits further information about the surface of the objective function such as the second order derivative (i.e. the Hessian) which is a good way to accelerate the local convergence [57]. In our framework, this descent direction is also combined with a global plane search procedure with two stepsizes as described in section 5.1.1. Note that our preliminary computer results showed us that the use of two stepsizes makes the Newton method converge faster in difficult contexts. Two algorithms have been implemented: the first, called mNewton, updates simultaneously matrices $\boldsymbol{B}$ and $\boldsymbol{C}$, and the second, named A-mNewton, updates them alternately. The vector update rule of the mNewton method is then defined by:

$$\begin{bmatrix} \mathrm{vec}\left(\boldsymbol{B}^{it+1}\right) \\ \mathrm{vec}\left(\boldsymbol{C}^{it+1}\right) \end{bmatrix} = \begin{bmatrix} \mathrm{vec}\left(\boldsymbol{B}^{it}\right) \\ \mathrm{vec}\left(\boldsymbol{C}^{it}\right) \end{bmatrix} - \begin{bmatrix} \mu_{\boldsymbol{B}}^{it}\,\mathbf{I}_{IP} & \mathbf{0} \\ \mathbf{0} & \mu_{\boldsymbol{C}}^{it}\,\mathbf{I}_{KP} \end{bmatrix} D^2\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{-1} \begin{bmatrix} D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\intercal} \\ D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\intercal} \end{bmatrix} \tag{52}$$

The expression above is more general than the standard Newton iteration, since we allow to

Algorithm 4 Pseudo-code for the mNewton algorithm

Initialize matrices $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$.
Set $it = 0$, the relative error err $= 1$ and set the maximal number *itmax* of allowed iterations and the stop criterion threshold ϵ to a predefined value, respectively.
while err $> \epsilon$ or $it <$ *itmax* **do**
  Compute the gradient $D\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ and the Hessian $D^2\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ using lemmas 3, 6, 7 and 8.
  Compute the modified Hessian matrix $\boldsymbol{H}(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ as described in equation (53).
  Compute the stepsizes $\mu_{\boldsymbol{B}}^{it}$ and $\mu_{\boldsymbol{C}}^{it}$, used in the equation (52), with the global plane search scheme described in section 5.1.1.
  Update the matrices $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{it+1}$ using the equation (52).
  Compute err $= |\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it+1}) - \Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})|/\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$.
  Set $it = it + 1$.
end while

modify the stepsizes via a global plane search (i.e. in two complementary subspaces). To ensure

local convergence of this algorithm, the Hessian must be positive definite. Therefore, a regularization step of this Hessian is mandatory. This regularization consists simply in adding the identity matrix multiplied by a positive value τ evaluated in an iterative procedure initialized by the Frobenius norm of the Hessian [61]. Indeed, this positive definiteness property of the Hessian is evaluated through the possibility or not to achieve a Cholesky decomposition. If this decomposition is possible, the scalar τ is kept while it is increased if this decomposition is not so. Then the modified Hessian, $\boldsymbol{H}$, can be written as:

$$\boldsymbol{H}(\boldsymbol{B}^{it}, \boldsymbol{C}^{it}) = D^2\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it}) + \tau \mathbf{I}_{P(I+K)} \tag{53}$$

The pseudo-code given in algorithm 4 describes the main steps of the mNewton method, which uses two globally optimal stepsizes. Besides the mNewton algorithm, an alternative version, named A-mNewton is proposed hereafter. The main advantage of this alternative version resides in its attractive computational cost due the approximation of the Hessian by its two diagonal blocks. Then, the update rule related to the A-mNewton algorithm is given in the following form:

$$\text{vec}\Big(\boldsymbol{B}^{it+1}\Big) = \text{vec}\Big(\boldsymbol{B}^{it}\Big) - \mu_{\boldsymbol{B}} \boldsymbol{H}_{\boldsymbol{B},\boldsymbol{B}}^{-1} D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\intercal} \tag{54}$$

$$\text{vec}\Big(\boldsymbol{C}^{it+1}\Big) = \text{vec}\Big(\boldsymbol{C}^{it}\Big) - \mu_{\boldsymbol{C}} \boldsymbol{H}_{\boldsymbol{C},\boldsymbol{C}}^{-1} \Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it})^{-1} D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it+1}, \boldsymbol{C}^{it})^{\intercal} \tag{55}$$

where $\boldsymbol{H}_{\boldsymbol{B},\boldsymbol{B}}$ and $\boldsymbol{H}_{\boldsymbol{C},\boldsymbol{C}}$ are the modified Hessians of Ψ w.r.t. $\boldsymbol{B}$ and $\boldsymbol{C}$. $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$ are the stepsizes computed using the global plane search procedure. Note that $D^2_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ is positive definite, according to equation (41), except if the rank of $\boldsymbol{Z}$ is not equal to P. In this case, $D^2_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ is only semi-definite positive and the zero eigenvalue should be removed. The pseudo-code given in algorithm 5 describes the main steps of the A-mNewton method.

Algorithm 5 Pseudo-code for the A-mNewton algorithm

Initialize matrices $\boldsymbol{B}^0$ and $\boldsymbol{C}^0$.
Set $it = 0$, the relative error err $= 1$ and set the maximal number $itmax$ of allowed iterations and the stop criterion threshold ϵ to a predefined value, respectively.
while err $> \epsilon$ or $it < itmax$ **do**
  Compute the gradients $D_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\intercal}$ and $D_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})^{\intercal}$; and the Hessians $D^2_{\boldsymbol{B}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ and $D^2_{\boldsymbol{C}}\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ using lemmas 3, 6 and 7 and 8.
  Compute the modified Hessian matrix $\boldsymbol{H}_{\boldsymbol{B},\boldsymbol{B}}(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$ and $\boldsymbol{H}_{\boldsymbol{C},\boldsymbol{C}}(\boldsymbol{B}^{it}, \boldsymbol{C}^{it})$, as described in equation (53).
  Compute the stepsizes $\mu^{it}_{\boldsymbol{B}}$ and $\mu^{it}_{\boldsymbol{C}}$ used in the equations (54) and (55), with the global plane search scheme described in section 5.1.1.
  Update the matrices $\boldsymbol{B}^{it+1}$ and $\boldsymbol{C}^{it+1}$ using the equations (54) and (55)
  Compute err $= |\Psi(\boldsymbol{B}^{it}, \boldsymbol{C}^{it}) - \Psi(\boldsymbol{B}^{it-1}, \boldsymbol{C}^{it-1})|/\Psi(\boldsymbol{B}^{it-1}, \boldsymbol{C}^{it-1})$.
  Set $it = it + 1$.
end while

8. Numerical complexity

The computational complexity of the proposed methods is given in table 1 and explained with more details below. Note that all expressions have been calculated analytically.

Let us begin by tackling the numerical complexity of the step common to all methods proposed in this paper, namely the computation of the gradient. The compact form of $D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$ given by (26) costs approximately $I^2P(2P^3+IP+K)$. As far as the term $D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ is concerned, it requires exactly $KP(2KP + I^2)$ multiplications. When all the dimensions of the three-way array are considered of the same order $O(I)$, the calculation of the gradient costs approximately $O(I^2P^5+I^3P)$.

Next, let us consider the algorithms based on the line search strategy, namely CG_{1step}, $\text{CG}_{\text{2steps}}$, A-mNewton and mNewton. The computation of the global line search, as described in sections 5.1.2 and 5.1.1, requires an order of $\theta I^2KP+P^2(I+K)$ multiplications where $\theta = 5$ for the computation of one optimal stepsize and $\theta = 9$ for two optimal stepsizes. It is noteworthy that the first term θI^2KP is associated to the computation of the last row and column of matrix $\boldsymbol{Q}$ defined in section 5.1. The second term $P^2(I+K)$ corresponds to the computation of the other components of $\boldsymbol{Q}$. Under the assumption that the three dimensions of the tensor are of the same order $O(I)$, the computation of the global line search requires an order of $O(I^3P+IP^2)$ multiplications. As far as the Newton-like methods are concerned, they are based on the computation of the Hessian of Ψ. The computation of the three blocks $D^2_{\boldsymbol{B},\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$, $D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ and $D^2_{\boldsymbol{C},\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$ requires an order of $4I^2P^4 + 5I^3P^3 + IP^5 + I^2KP$, K^2P^2 and $IKP^4 + K^2P^4 + I^2P^4 + KP^5$ multiplications, respectively. If $I = K$, the calculation of the Hessian (the two diagonal blocks for the A-mNewton algorithm and the whole Hessian for the mNewton algorithm) requires an order of $O(I^2P^4+I^3P^3 + IP^5)$ multiplications. In addition, the iterative procedure based on the Cholesky decomposition [61] costs $j(I + K)^3P^3$ multiplications for mNewton and $j(I^3P^3 + K^3P^3)$ multiplications for A-mNewton where j denotes the number of iterations. Moreover, $(I + K)^2P^2$ and $(I^2 + K^2)P^2$ multiplications are required by mNewton and A-mNewton, respectively, to solve the system of two triangular equations avoiding a direct inversion of the modified Hessian. So, the step of solving the update rules requires an order of $O(I^3P^3)$ multiplications, under the assumption of $I = K$.

Eventually, let us study the methods based on the trust region strategy, namely the LM^+_{sym} and $\text{A-LM}^+_{\text{sym}}$ methods. The computation of the three blocks $\boldsymbol{J}_B{}^\top\boldsymbol{J}_B$, $\boldsymbol{J}_C{}^\top\boldsymbol{J}_C$ and $\boldsymbol{J}_B{}^\top\boldsymbol{J}_C$ costs approximately I^2P^2, K^2P^2 and $I^2P^2(I^2 + KP)$, respectively. For $I = K$, $\text{A-LM}^+_{\text{sym}}$ requires the computation of the two diagonal blocks, requiring an order of $O(I^2P^2)$ multiplications, while LM^+_{sym} needs in addition the calculation of $\boldsymbol{J}_B{}^\top\boldsymbol{J}_C$, leading to $O(I^4P^2 + I^3P^3)$ multiplications. Besides, as for the Newton-like methods, the computational complexity of one Cholesky decomposition [56] and the solving of one system of two triangular equations used in LM^+_{sym} and $\text{A-LM}^+_{\text{sym}}$ requires $O(I^3P^3)$ multiplications. These steps are repeated as long as the damping parameter is rejected.

9. Computer results

This section is two-fold. First, the performance of the methods proposed in this paper is investigated. Second, the interest in exploiting prior information when fitting the CP model is analyzed, such as semi-nonnegativity and semi-symmetry. This analysis is performed in terms of convergence speed, accuracy of factor estimation and CPU time.

The experiments are first performed with synthetic datasets, in order to evaluate the impact of the Signal to Noise Ratio (SNR), the collinearity, the rank and the size on the behavior of the considered methods. Our methods are compared with ELS-ALS [47] and LM [64, 41] algorithms, where neither symmetry nor nonnegativity constraints are imposed. The goal of the second part

	$CG_{1,2steps}$	LM^{+}_{sym}	A-LM^{+}_{sym}	mNewton	A-mNewton
Line search $O(I^3P+IP^2)$	✓			✓	✓
Gradient $O(I^2P^5+I^3P)$	✓	✓	✓	✓	✓
$\boldsymbol{J}^\intercal\boldsymbol{J}$ $O(I^4P^2+I^3P^3)$ $O(I^2P^2)$		✓	✓		
Hessian $O(I^2P^4+I^3P^3+IP^5)$				✓	✓
Cholesky decomposition $O(I^3P^3)$		✓	✓	✓	✓

Table 1: Numerical complexity for $I = K$ of the main steps of the proposed algorithms. The symbol ✓ indicates that the method uses the related step. Furthermore, for the last three steps, the position of ✓ can indicate whether the first or the second expression of the related step is used.

is to show how the semi-nonnegative INDSCAL problem can occur in ICA applications and to assess the behavior of the proposed methods w.r.t. classical ICA [21, 70] methods (see section 9.2). To do so, some experiments are carried out on image data.

9.1. Test on synthetic data

Synthetic semi-nonnegative and semi-symmetric array sets are generated from random loading matrices, according to equation (4), with different ranks, levels of collinearity between factors and different sizes. In particular, the zero-mean unit-variance Gaussian distribution is used to simulate the matrices $\boldsymbol{B}$ and $\boldsymbol{C}$. Moreover the resulting three-way arrays are contaminated by a semi-nonnegative and semi-symmetric third-order array of noise, named $\boldsymbol{\mathcal{N}}$, also generated with the help of two Gaussian random matrices. The corresponding noisy third-order array $\boldsymbol{\mathcal{Y}}$ is written as:

$$\boldsymbol{\mathcal{Y}} = \frac{\boldsymbol{\mathcal{T}}}{\|\boldsymbol{\mathcal{T}}\|_F} + \sigma_{\mathcal{N}} \frac{\boldsymbol{\mathcal{N}}}{\|\boldsymbol{\mathcal{N}}\|_F} \tag{56}$$

where $\sigma_{\mathcal{N}}$ is a scalar controlling the noise level. The Signal to Noise Ratio (SNR) is then defined by SNR$= -20\log_{10}(\sigma_{\mathcal{N}})$.

As a performance criterion, we use a global measure allowing to quantify the estimation error of both loading matrices $\boldsymbol{A}$ and $\boldsymbol{C}$. Such a measure is scale-invariant and blind to permutation indeterminacies inherent in problem 1. It is given by:

$$\alpha = \frac{1}{2P}\sum_{k=1}^{P} \Delta_k^A + \Delta_k^C \tag{57}$$

where:

$$\Delta_k^G = \min_{(p,p')\in I^2} d(\boldsymbol{g}_p, \widehat{\boldsymbol{g}_{p'}}) \tag{58}$$

with $\boldsymbol{g}_p$ and $\widehat{\boldsymbol{g}_p}$ the p-th column of $\boldsymbol{G}$ and $\widehat{\boldsymbol{G}}$, respectively and $I^2 = \{1, ..., P\} \times \{1, ..., P\}$. Besides, d is the pseudo-distance between vectors defined by [67]:

$$d(\boldsymbol{u}, \boldsymbol{v}) = 1 - \frac{\|\boldsymbol{u}^\intercal \boldsymbol{v}\|^2}{\|\boldsymbol{u}\|^2 \|\boldsymbol{v}\|^2} \tag{59}$$

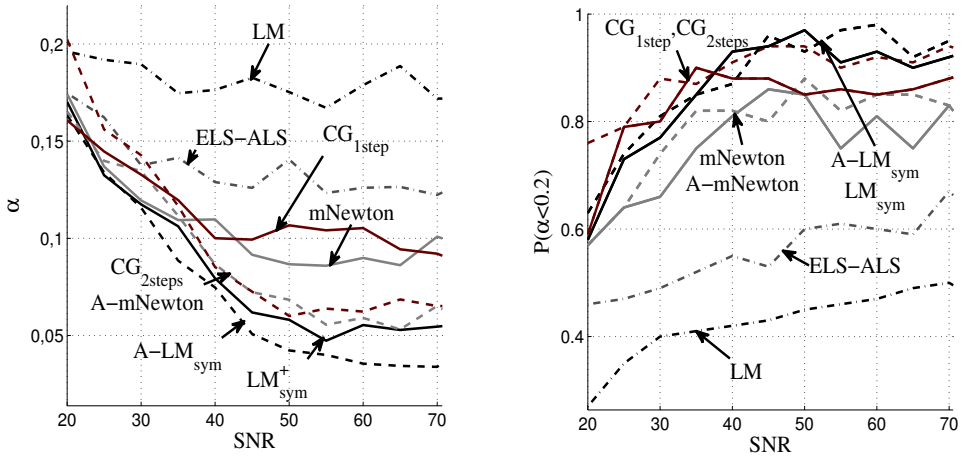
In the first step of α, the smallest distance is selected, associating a column of a loading matrix with one of its estimate. Then, this pair of indices is removed from I^2, and the step is repeated until the P columns of $\widehat{\boldsymbol{G}}$ are used and the set I^2 is empty. Such a criterion is an upper bound of the optimal criterion and enjoys several properties: i) it is usable for high rank arrays, ii) it avoids the computation of all permutations and iii) it prevents two factors from being estimated by the same computed factor. Note that, for the scenario with low rank studied in this section, we checked the good quality of the measure α in comparison with the corresponding criterion calculating all permutations. All algorithms considered in this study stop either when the relative error of the cost function between two successive iterations, as defined in the above pseudo-codes, exhibits a value below a predefined threshold of 10^{-12}, or when the number of iterations exceeds 2000. For each method, the probability of nonaberrant measures, called $P(\alpha < 0.2)$, is defined by the ratio between the number of realizations for which the measure α is less than 0.2 and the number of realizations [68]. From the nonaberrant realizations, we calculate the averaged measure α, as defined previously.

Furthermore, all results reported in this section are averaged over 200 Monte Carlo realizations, where new loading matrices and noise are generated at each realization. In addition, all methods are initialized by the same matrices $\boldsymbol{C}$ and $\boldsymbol{A}$ such as $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, with $\boldsymbol{B}$ and $\boldsymbol{C}$ drawn from a zero-mean unit-variance normal distribution.

9.1.1. Influence of SNR

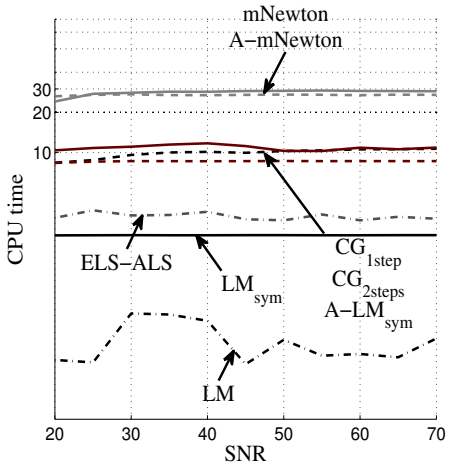
In this experiment, the impact of SNR is evaluated. The dimensions and the rank of $\boldsymbol{\mathcal{T}}$ are fixed to $(6 \times 6 \times 6)$ and 4, respectively. The nonnegative matrix $\boldsymbol{A}$ is generated with two bottlenecks parametrized as $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_1 + \beta_{col}\,\boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_3 + \beta_{col}\,\boldsymbol{a}_4]$, where each vector $\boldsymbol{a}_i$ is drawn from a standard uniform distribution and where β_{col} is equal to 0.4. Figure 1(a) shows the mean of measure α at the output of the eight methods as a function of SNR, ranging from 20 dB to 70 dB. In parallel, the probability of nonnaberrant measures is reported in figure 1(b) as a function of SNR. As expected, the decomposition is more effective in the context of high SNR values with a probability of nonnaberrant measures close to one. The benefit of exploiting semi-nonnegativity and semi-symmetry can be observed in this context. This can straightforwardly be seen by comparing the performance of the conventional LM algorithm with its constrained versions using semi-symmetry and semi-nonnegativity, i.e. $\text{LM}^{+}_{\text{sym}}$ and A-$\text{LM}^{+}_{\text{sym}}$ algorithms. More generally, we observe a significant gap between classical LM and ELS-ALS algorithms and our methods, in terms of performance and probability of nonnaberrant measures. In fact, for every SNR value, our methods outperform both unconstrained CP algorithms and give a very good estimation of the loading matrices with probabilities of nonnaberrant measures close to one. Note that classical LM and ELS-ALS algorithms have a poor probability of nonnaberrant measures, as demonstrated in Figure 1(b). Figure 1(c) displays the mean of CPU time for each method as a function of SNR. In this context, the cheapest method is the classical LM, where neither symmetry nor nonnegativity constraints are imposed. But its estimation accuracy is bad. Among the remaining methods, $\text{LM}^{+}_{\text{sym}}$ is the cheapest. We also notice that Newton-like methods are very computationally demanding. The best accuracy/complexity trade-off seems to be achieved by the $\text{LM}^{+}_{\text{sym}}$ approach.

In the second experiment, the impact of SNR is evaluated on synthetic data with higher dimensions. The dimensions and the rank of $\boldsymbol{\mathcal{T}}$ are fixed to $(100 \times 100 \times 100)$ and 25, respectively. To reduce the number of mathematical operations, memory allocation and estimated parameters, a compression step is subsequently applied on the arrays leading to a dimension reduction, i.e. a reduction in the number of parameters to be estimated. We use the same technique as HOSVD



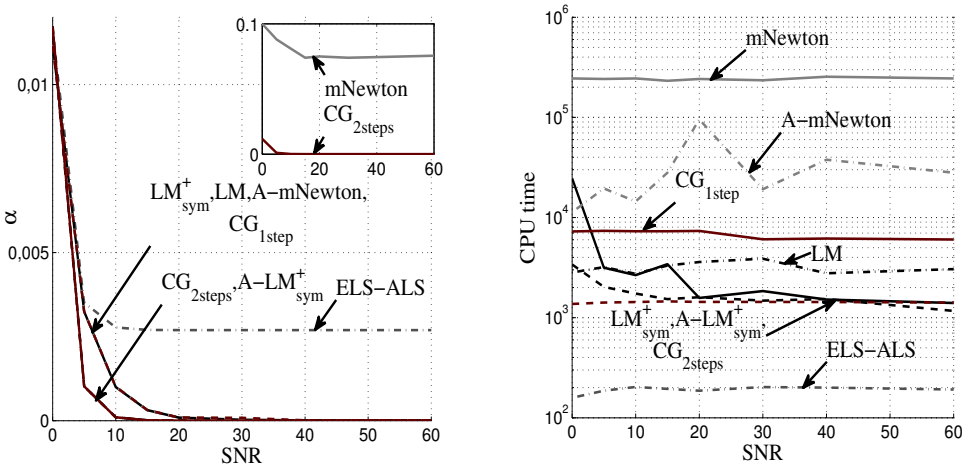
(a) Performance measure α

(b) $P(\alpha < 0.2)$



(c) CPU time

Figure 1: Influence of SNR for a $(6 \times 6 \times 6)$ array, $P = 4$ and $\beta_{col} = 0.4$ at the output of $\text{CG}_{2\text{steps}}$ (the CG algorithm with two optimal step lengths), $\text{CG}_{1\text{step}}$ (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, $\text{A-LM}^{+}_{\text{sym}}$ and two unconstrained CP methods, namely ELS-ALS and LM.



(a) Performance measure α

(b) CPU time

Figure 2: Influence of SNR for a $(100\times100\times100)$ array and $P = 25$ at the output of $\text{CG}_{2\text{steps}}$ (the CG algorithm with two optimal step lengths), $\text{CG}_{1\text{step}}$ (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, A-$\text{LM}^{+}_{\text{sym}}$ and two unconstrained CP methods, namely ELS-ALS and LM.

[41], but the compression is executed only for the third mode, for which no nonnegativity constraint is required. Indeed, a compression via truncated HOSVD transforms the tensor $\boldsymbol{\mathcal{T}}$ into another tensor with lower mode-rank (or identical if this compression is lossless). In our computer experiments, the compression is lossy because of additive noise, so that the problem we actually solve differs from (5). Figure 2(a) shows the α measure at the output of both unconstrained CP algorithms compared to our proposed ones as a function of SNR, which ranges from 0 to 60 dB. In practice, the gain in numerical complexity also makes this lossy compression mandatory for tensors of large size. Indeed, this compression step results in third order arrays of size $(100\times100\times25)$. As expected, the decomposition becomes more effective in terms of estimation accuracy when the SNR increases. Furthermore, our proposed semi-nonnegative INDSCAL algorithms outperform the ELS-ALS method, except mNewton. Contrary to the previous simulation, the LM algorithm exhibits a good performance, in the context of larger dimensions, regardless of the SNR value. Note also that for low SNR values, both the $\text{CG}_{2\text{steps}}$ and the A-$\text{LM}^{+}_{\text{sym}}$ approaches outperform the other algorithms considered in this scenario. Regarding the probabilities of nonabberant measures of all algorithms, they are equal to one for all SNR values, except mNewton. We note that mNewton is less efficient in terms of estimation accuracy in this study. Figure 2(b) displays the mean of CPU time for each method as a function of the SNR values. As depicted in figure 2(b), ELS-ALS is the cheapest method for SNR values ranging from 0 dB to 60 dB. It is worth mentioning that the measured numerical complexity is constant regardless of the SNR value. This is due to the fact that the number of iterations required for the convergence of each realization is always equal to 2000. Figure 2(b) shows also that $\text{CG}_{2\text{steps}}$ and our two LM-like methods are less costly in terms of CPU time than the classical LM method. In the context of low SNR values, $\text{CG}_{2\text{steps}}$ is even cheaper.

9.1.2. Influence of collinearity

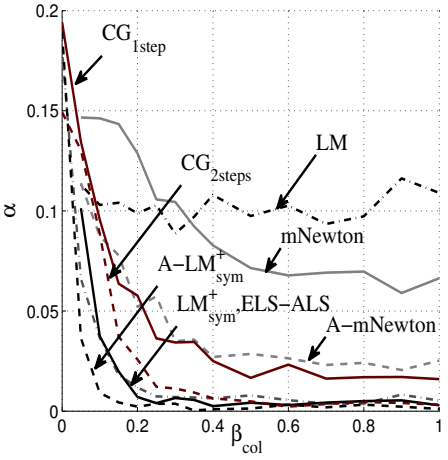
In this simulation, a difficult context is addressed and the impact of collinearity between columns of a matrix factor is evaluated. As in previous experiments, a set of (10×4) nonnegative matrices $\boldsymbol{A}$ is generated with two bottlenecks and the parameter of collinearity, called β_{col}, is ranging from 0 to 1. $\boldsymbol{C}$ is generated with a standard normal distribution. Figure 3(a) shows the mean of measure α at the output of all methods as a function of β_{col}, for an SNR value equal to 60dB. Figure 3(b) displays the probability of nonabberrant measures, $P(\alpha < 0.2)$, for each value of β_{col}. As expected, the decomposition is less effective in terms of estimation accuracy when vectors belonging to a bottleneck are close to collinear ($\beta_{col} \approx 0$). Moreover, when vectors are totally collinear ($\beta_{col} = 0$), the probability of having α lower than 0.2 is equal to zero. As shown in figure 3(b), this probability increases to one at various speeds. Besides, we note the big gap in the probability measure between the unconstrained LM algorithm and our methods with semi-nonnegativity and semi-symmetry constraints, principally when the collinearity coefficient is beyond 0.1. In fact, the probability $P(\alpha < 0.2)$ of our two LM-like methods is close to one as soon as $\beta_{col} = 0.1$, followed by both CG algorithms. Concerning ELS-ALS, its probability is lower than that of our LM-like and CG-like approaches. Among all the compared methods, for β_{col} beyond 0.1, A-LM$^{+}_{\text{sym}}$ and LM$^{+}_{\text{sym}}$ are the most effective, both in terms of estimation accuracy and probability of nonaberrant results. Figure 3(c) displays the averaged CPU time as a function of β_{col}. Both unconstrained CP methods have the lowest cost. Regarding our algorithms, for columns close to collinear, LM$^{+}_{\text{sym}}$ is the cheapest. However, when the collinearity is relaxed, the cost of A-LM$^{+}_{\text{sym}}$ is reduced and becomes comparable to the one of LM$^{+}_{\text{sym}}$. To sum up, this study brings out the good behavior of LM$^{+}_{\text{sym}}$ in terms of performance and CPU time.

9.1.3. Influence of the rank

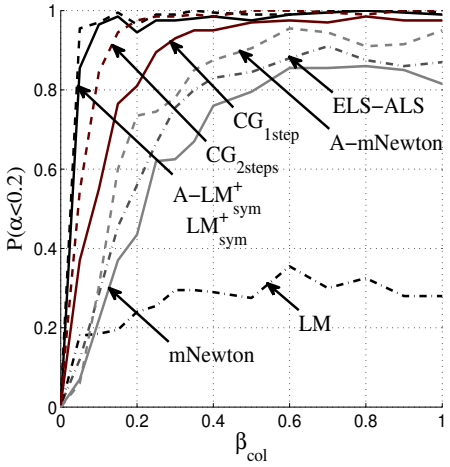
In this section, both matrices $\boldsymbol{C}$ and $\boldsymbol{B}$ are of size $(50 \times P)$ with $P \in \{20, 50\}$ and for SNR=10dB. Table 2 shows the mean of measure α at the output of all methods for $P \in \{20, 50\}$. As expected, the decomposition is more effective in the context of low rank. We note that, for low ranks, the performance accuracy of ELS-ALS and most of the semi-nonnegative INDSCAL methods are similar. Next, LM and mNewton are less efficient than the other ones, for $P = 20$. Besides, our methods outperform the unconstrained CP methods for $P = 50$, except for mNewton. Indeed, this experiment shows how the full exploitation of the nonnegativity and symmetry properties, if it exists, in the INDSCAL decomposition of high-rank arrays, allows for a good estimation accuracy. Regarding the probabilities of having nonaberrant measures, this experiment shows that these quantities are equal to one for all the considered algorithms. Note also that, for both P values, the A-LM$^{+}_{sym}$ method gives the best estimation accuracy. Table 2 shows the mean of the CPU time at the output of all methods for $P \in \{20, 50\}$. As expected, the CPU time increases with the rank. Among all semi-nonnegative INDSCAL methods, the one with the lowest cost is A-LM$^{+}_{sym}$ for low and high ranks. The best accuracy/complexity trade off seems to be achieved by the A-LM$^{+}_{sym}$ approach.

9.1.4. Influence of dimensions

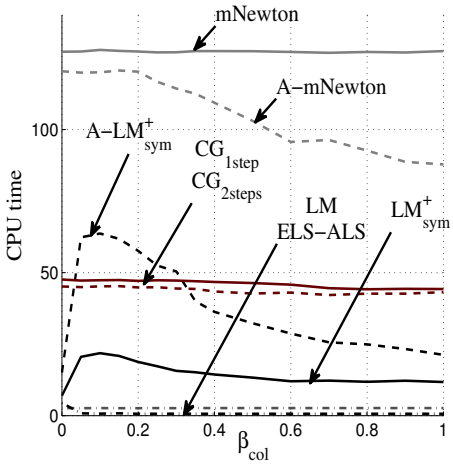
In this experiment, for an SNR value of 60dB, a set of $(I\times I\times I)$ three-way arrays are generated with I varying from 10 to 100. To address a more difficult context, a collinearity coefficient equal to 0.8 between the columns of the loading matrix $\boldsymbol{A}$ is assumed. A compression step is applied to the arrays, as in section 9.1.1. Table 3 shows the mean of the α measure at the output of all methods as a function of I ranging from 10 to 100. As shown in the experiment evaluating the influence of the collinearity (section 9.1.2), some semi-nonnegative INDSCAL methods such as



(a) Performance measure α



(b) $P(\alpha < 0.2)$



(c) CPU time

Figure 3: Influence of collinearity for a $(10 \times 10 \times 10)$ array, $P = 4$ and SNR=60dB, over the numerical complexity and the performance at the output of $\text{CG}_{\text{2steps}}$ (the CG algorithm with two optimal step lengths), CG_{1step} (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, $\text{A-LM}^{+}_{\text{sym}}$ and two unconstrained CP methods, namely ELS-ALS and LM.

Algorithms \ P	$P = 20$		$P = 50$	
	α	CPU time	α	CPU time
ELS-ALS	1.7×10^{-3}	199	3.1×10^{-2}	972.3
LM	2.3×10^{-3}	181.8	2.5×10^{-2}	3.45×10^{3}
CG_{1step}	1.7×10^{-3}	871.6	2.3×10^{-2}	1.4×10^{4}
$\text{CG}_{\text{2steps}}$	1.7×10^{-3}	803.5	2.3×10^{-2}	1.1×10^{4}
LM^{+}_{sym}	1.7×10^{-3}	1.2×10^{3}	2.7×10^{-2}	1.4×10^{4}
A-LM^{+}_{sym}	1.7×10^{-3}	353.2	2.3×10^{-2}	1.1×10^{4}
mNewton	4.1×10^{-3}	1.12×10^{4}	4.8×10^{-2}	2.3×10^{5}
A-mNewton	1.7×10^{-3}	3.5×10^{3}	2.5×10^{-2}	1×10^{5}

Table 2: Influence of the rank for a ($50 \times 50 \times 50$) array and SNR=10dB over the measure α and CPU time at the output of $\text{CG}_{\text{2steps}}$ (the CG algorithm with two optimal step lengths), CG_{1step} (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, $\text{A-LM}^{+}_{\text{sym}}$ and two unconstrained CP methods, namely ELS-ALS and LM.

$\text{CG}_{\text{2steps}}$, LM^{+}_{sym} and A-LM^{+}_{sym}, outperform the unconstrained CP algorithms, namely ELS-ALS and LM, for $I = 10$. We note that the difference in the accuracy between unconstrained CP algorithms and those with semi-nonnegativity and semi-symmetry constraints decreases when the size of the three-way arrays increases. Besides, in this easy context of large arrays and low rank, our computer results show that the probability $P(\alpha < 0.2)$ is close to one for all array sizes and all algorithms. In this context, the methods based on the alternating update rules, namely A-LM^{+}_{sym} and A-mNewton, give better results than those based on simultaneous updates. This remark is also true for both conjugate gradient-like methods, CG_{1step} and $\text{CG}_{\text{2steps}}$, for which the performance criterion does not decrease significantly when I increases. Furthermore A-LM^{+}_{sym} converges very well and gives the best results for $I \in \{50, 100\}$. Table 4 shows the mean of the CPU time at the output of all methods as a function of I ranging from 10 to 100. All methods are expensive in this context, due to the fact that, contrary to the unconstrained case, semi-nonnegative INDSCAL three-way arrays are only compressed in the third mode, for which no nonnegativity constraint is imposed. The unconstrained LM algorithm is by far the cheapest method in this context of high dimensions and low rank. The cost of both LM-like methods is close to that of ELS-ALS. In the context of high dimensions and high collinearity, the best accuracy/complexity trade off seems to be achieved by the A-LM^{+}_{sym} approach.

9.2. Application to semi-nonnegative ICA

In this section, a problem of semi-nonnegative ICA is addressed, where algorithms exploit some interesting properties enjoyed by cumulants. The semi-nonnegative ICA problem can be then transformed into a semi-nonnegative INDSCAL model fitting, and the methods described previously are used to solve it.

9.2.1. Semi-nonnegative ICA

As mentioned in section 2, the ICA concept has been widely used to solve real-world problems. However, they did not exploit further prior information such as nonnegativity. Herein, the nonnegativity constraint of the mixing coefficients is taken into account during the ICA step giving rise to the semi-nonnegative ICA. This is illustrated through the problem of blind separation of face images. To this end let's start by defining the semi-nonnegative ICA concept.

Algorithms \ I	10	20	30	50	100
ELS-ALS	0.91×10^{-3}	0.33×10^{-4}	0.55×10^{-5}	0.15×10^{-6}	0.1×10^{-6}
LM	0.10	0.41×10^{-1}	0.28×10^{-1}	0.24×10^{-1}	0.20×10^{-1}
CG_{1step}	0.2×10^{-1}	0.78×10^{-2}	0.23×10^{-2}	0.5×10^{-3}	0.5×10^{-3}
$\text{CG}_{\text{2steps}}$	0.5×10^{-3}	0.24×10^{-3}	0.19×10^{-3}	0.29×10^{-3}	0.61×10^{-3}
LM^{+}_{sym}	0.6×10^{-3}	0.2×10^{-3}	0.2×10^{-3}	0.2×10^{-3}	0.2×10^{-3}
A-LM^{+}_{sym}	0.18×10^{-4}	0.93×10^{-6}	0.36×10^{-6}	0.13×10^{-6}	0.6×10^{-7}
mNewton	0.64×10^{-1}	0.54×10^{-1}	0.54×10^{-1}	0.54×10^{-1}	0.54×10^{-1}
A-mNewton	0.21×10^{-1}	0.62×10^{-2}	0.51×10^{-2}	0.43×10^{-2}	0.40×10^{-2}

Table 3: Influence of the dimensions for a $(I \times I \times I)$ array, $P = 4$ and SNR=60dB over the measure α at the output of $\text{CG}_{\text{2steps}}$ (the CG algorithm with two optimal step lengths), CG_{1step} (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, $\text{A-LM}^{+}_{\text{sym}}$ and two classical CP methods, namely ELS-ALS and LM.

Algorithms \ I	10	20	30	50	100
ELS-ALS	3.59	27.49	36.49	51.22	120.06
LM	0.11	1.17	1.77	4.44	23.59
CG_{1step}	7.40	73.88	98.83	183.05	493.92
$\text{CG}_{\text{2steps}}$	10.04	76.63	94.13	159.42	432.03
LM^{+}_{sym}	1.42	19.46	21.96	46.11	147.48
A-LM^{+}_{sym}	4.10	58.26	57.55	90.62	238.28
mNewton	$0.12 \times 10^{+3}$	$0.21 \times 10^{+3}$	$0.31 \times 10^{+3}$	$0.76 \times 10^{+3}$	$4.10 \times 10^{+3}$
A-mNewton	$0.11 \times 10^{+3}$	$0.16 \times 10^{+3}$	$0.24 \times 10^{+3}$	$0.40 \times 10^{+3}$	$1.22 \times 10^{+3}$

Table 4: Influence of the dimensions for a $(I \times I \times I)$ array, $P = 4$ and SNR=60dB over the CPU time at the output of $\text{CG}_{\text{2steps}}$ (the CG algorithm with two optimal step lengths), CG_{1step} (with $\mu_{\boldsymbol{B}} = \mu_{\boldsymbol{C}}$ optimally computed), mNewton (the mNewton algorithm with two optimal learning steps $\mu_{\boldsymbol{B}}$ and $\mu_{\boldsymbol{C}}$), A-mNewton, $\text{LM}^{+}_{\text{sym}}$, $\text{A-LM}^{+}_{\text{sym}}$ and two classical CP methods, namely ELS-ALS and LM.

Definition 6. *Given realizations* $\{\boldsymbol{x}[t]\}$ *of a real random vector* $\boldsymbol{x}$*, find an* $(I\times P)$ *mixing matrix* $\boldsymbol{A}$ *and realizations* $\{\boldsymbol{s}[t]\}$ *of a* P*-dimensional source random vector* $\boldsymbol{s}$ *such that* $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$ *where* $\boldsymbol{A}$ *has nonnegative components and* $\boldsymbol{s}$ *has statistically independent components.*

To solve this problem, we decide to jointly use Second Order (SO), Third Order (TO) and Fourth Order (FO) cumulant arrays. Indeed, resorting to the use of different cumulant orders allows us to cover the variety of source spectra that may be involved in the observed mixture. Now, the purpose is to show how we can combine cumulants into one third order array, following the semi-nonnegative INDSCAL model.

Let $C_{n_1,n_2,\boldsymbol{x}}$, $C_{n_1,n_2,n_3,\boldsymbol{x}}$ and $C_{n_1,n_2,n_3,n_4,\boldsymbol{x}}$ be the entries of the SO, TO and the FO cumulant arrays, $\boldsymbol{C}_{2,\boldsymbol{x}}, \boldsymbol{C}_{3,\boldsymbol{x}}, \boldsymbol{C}_{4,\boldsymbol{x}}$, respectively of a zero-mean I-dimensional random vector $\boldsymbol{x}$ computed as shown in [39]. We propose now to merge the entries of the three cumulant arrays in the same matrix, named $\boldsymbol{\mathcal{T}}_{\boldsymbol{x}}^{(2,3,4)}$ of size $(M\times I^2)$ with $M = 1 + I + I^2$. More precisely, the (m_1, m_2)-th entry, $T_{m_1,m_2,\boldsymbol{x}}^{(2,3,4)}$ of $\boldsymbol{T}_{\boldsymbol{x}}^{(2,3,4)}$ is defined as follows:

$$T_{m_1,m_2,\boldsymbol{x}}^{(2,3,4)} = \begin{cases} C_{n_1,n_2,\boldsymbol{x}} \ with \quad m_1 = 1 \ and \quad m_2 = n_2 + I(n_1 - 1) \\ C_{n_1,n_2,m_1,\boldsymbol{x}} \ \forall m_1 \in \{2,\cdots,I+1\} \ with \quad m_2 = n_2 + I(n_1 - 1) \\ C_{n_1,n_2,n_3,n_4,\boldsymbol{x}} \ \forall m_1 \in \{I+2,\cdots,I+I^2+1\} \\ with \quad m_1 = n_4 + Nn_3 \ and \ m_2 = n_2 + I(n_1 - 1) \end{cases} \tag{60}$$

According to the multi-linearity property enjoyed by cumulants [39], the SO, TO and FO cumulant tensors of the observations (6) follow the relations below:

$$\begin{aligned} C_{n_1,n_2,\boldsymbol{x}} &= \sum_{p=1}^{P} A_{n_1,p}A_{n_2,p}C_{p,p,s} \\ C_{n_1,n_2,n_3,\boldsymbol{x}} &= \sum_{p=1}^{P} A_{n_1,p}A_{n_2,p}A_{n_3,p}C_{p,p,p,s} \\ C_{n_1,n_2,n_3,n_4,\boldsymbol{x}} &= \sum_{p=1}^{P} A_{n_1,p}A_{n_2,p}A_{n_3,p}A_{n_4,p}C_{p,p,p,p,s} \end{aligned} \tag{61}$$

where $C_{p,p,s}$, $C_{p,p,p,s}$ and $C_{p,p,p,p,s}$ denote the SO, TO and the FO marginal cumulants of the p-th source, respectively. By inserting (60) in (61), we obtain the following algebraic structure:

$$\boldsymbol{T}_{\boldsymbol{x}}^{(2,3,4)} = \boldsymbol{C}(\boldsymbol{A} \oslash \boldsymbol{A})^{\mathsf{T}} \tag{62}$$

with $\boldsymbol{C} = [\boldsymbol{C}_{2,\boldsymbol{x}}, \boldsymbol{C}_{3,\boldsymbol{x}}\boldsymbol{A}^{\mathsf{T}}, \boldsymbol{C}_{4,\boldsymbol{x}}(\boldsymbol{A} \odot \boldsymbol{A})^{\mathsf{T}}]^{\mathsf{T}}$ where the matrices $\boldsymbol{C}_{3,s} = \mathrm{diag}[C_{1,1,1,s},\cdots,C_{P,P,P,s}]$ and $\boldsymbol{C}_{4,s} = \mathrm{diag}[C_{1,1,1,1,s},\cdots,C_{P,P,P,P,s}]$ of size $(P \times P)$ are diagonal and the vector $\boldsymbol{C}_{2,s} = [C_{1,1,s},\cdots,C_{P,P,s}]$ is of size P. Now, according to Definition 5, equation (62) is nothing else but the matrix representation of the semi-nonnegative INDSCAL model defined in Problem 1. Note that the semi-symmetry property is satisfied since the resulted TO tensor described through its matrix representation in (62) has two equal loading matrices.

9.3. Tests on image data

In this section, the semi-nonnegative ICA is applied to images. We compare our methods to two classical ICA algorithms using the INDSCAL model, namely JADE [21] and SOBI [70]. Following the results obtained in the previous sections, and since CG_{2steps}, LM_{sym}^{+} and A-LM_{sym}^{+}

Algorithms	measure α	CPU time
JADE	0.110	0.149
SOBI	0.286	0.937
CG_{2steps}	0.036	9.473
LM^{+}_{sym}	0.036	5.025
A-LM^{+}_{sym}	0.036	6.747

Table 5: Average values of measure α and CPU times at the output of two classical ICA methods (JADE and SOBI) , CG_{2steps} (the CG algorithm with two optimal step lengths), LM^{+}_{sym} and A-LM^{+}_{sym}.

show the best efficiency/CPU timing trade-off, they are considered in this practical context. We consider a database[1] of 40 face images for this application, which are assumed to be statistically independent. For each dataset of two face images (among 780), five observations are generated as a mixture of two image sources. The nonnegative mixing matrix $\boldsymbol{A}$ is drawn from a uniform distribution. Table 1 shows the average of measure α of sources and the CPU time for 780 datasets at the output of the five methods. The three semi-nonnegative ICA methods, CG_{2steps}, LM^{+}_{sym} and A-LM^{+}_{sym}, outperform JADE and SOBI. This experiment shows that in this context the use of both constraints, namely nonnegativity and independence of the sources, allows us to achieve a good estimation accuracy. Concerning CPU time, JADE and SOBI appear to be the least expensive methods. As A-LM^{+}_{sym} is the least expensive among our semi-nonnegative ICA methods, it it may be a good compromise between accuracy and complexity.

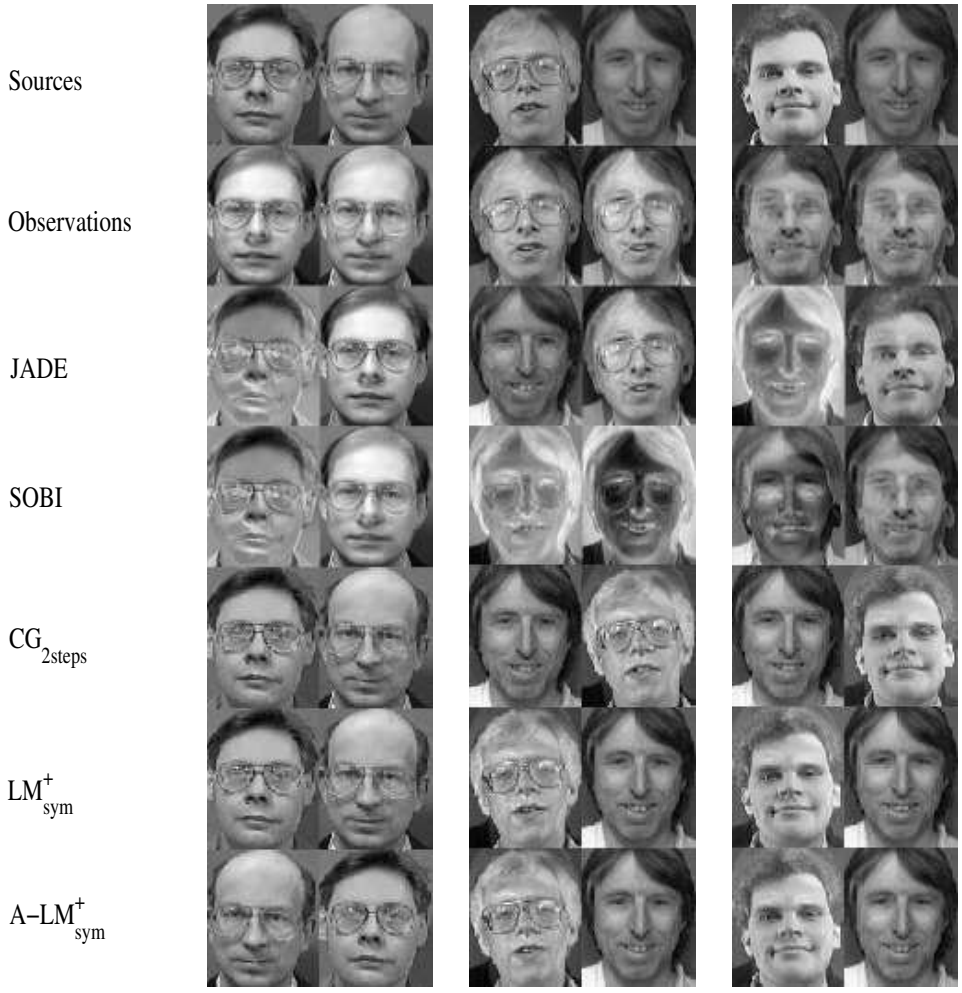
Figure 5 shows the extracted sources at the output of the five methods, for three experiments. The first line corresponds to sources and the second to observations. The two ICA methods, JADE and SOBI, fail to separate the two sources. On the other hand, using CG_{2steps}, LM^{+}_{sym} or A-LM^{+}_{sym} leads to a quasi-perfect separation.

10. Conclusion

Six algorithms have been proposed to compute the approximate CP decomposition of real three-way arrays, which are nonnegative and symmetric in two modes. Both constraints were explicitly exploited to achieve this decomposition. A change of variable into squares has been used to force the nonnegativity of two of the three loading matrices, leading to an unconstrained problem. In addition, two globalization strategies have been studied, namely line search and trust region. Regarding the former, a global plane search procedure has been considered, with one or two stepsizes optimally computed at each iteration. Moreover, all derivatives have been calculated in matrix form using the algebraic basis of matrix calculus and product operator properties. The computational complexity issue has been taken into account in the design phase of the algorithms, and has been evaluated for each algorithm, allowing to fairly compare their performance.

Then, various scenarios have been considered, aiming at testing the influence of i) an additive noise, which can stand for modeling errors, ii) the collinearity between factors, iii) the array rank and iv) the data size. The comparisons between our CG-like, Newton-like and LM-like methods (where semi-nonnegativity and semi-symmetry constraints are exploited), and classical

[1]Database accessible at http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html. The set of faces was taken between April 1992 and April 1994 at the Olivetti Research Laboratory in Cambridge, UK and used in [69].



(a) Legend (b) Test 1 (c) Test 2 (d) Test 3

Figure 4: Results for three simulations: (line 1) sources, (line 2) observations, (line 3) extracted sources by JADE, (line 4) by SOBI, (line 5) by CG_{2steps}, (line 6) by LM^{+}_{sym} and (line 7) by A-LM^{+}_{sym}.

CP algorithms (where no constraints are considered), showed that a better CP decomposition is obtained when these *a priori* are exploited. In particular in the context of high dimensions and high collinearity, we can observe that our methods outperform ELS-ALS and LM algorithms. Finally, based on our numerical analysis, the algorithms that seem to yield the best tradeoff between accuracy and complexity are $\text{LM}^{+}_{\text{sym}}$ for small dimensions, and CG_{2step} and $\text{A-LM}^{+}_{\text{sym}}$ for higher dimensions. Furthermore, the proposed algorithms have been evaluated in a practical context, namely semi-nonnegative ICA of face images. The results have shown the benefit gained by fully exploiting prior information, such as the nonnegativity of the mixing matrix.

Appendix A. The algebra of Kronecker and Khatri-Rao products and basic relationships

Some useful properties are recalled [40].

$$\begin{aligned}
\text{vec}(\boldsymbol{F}\boldsymbol{Z}) &= (\mathbf{I}_P \otimes \boldsymbol{F})\,\text{vec}(\boldsymbol{Z}) && \text{(A.1)}\\
\text{vec}(\boldsymbol{F}\boldsymbol{Z}) &= (\boldsymbol{Z}^{\mathsf{T}} \otimes \mathbf{I}_P)\,\text{vec}(\boldsymbol{F}) && \text{(A.2)}\\
\text{vec}(\boldsymbol{F}\boldsymbol{Z}) &= (\boldsymbol{Z}^{\mathsf{T}} \odot \boldsymbol{F})\mathbf{1}_N && \text{(A.3)}\\
\text{vec}(\boldsymbol{F}^{\mathsf{T}}) &= \boldsymbol{U}_{PN}\,\text{vec}(\boldsymbol{F}) && \text{(A.4)}\\
\text{vec}(\boldsymbol{F} \odot \boldsymbol{G}) &= \text{diag}(\text{vec}(\boldsymbol{F}) \otimes \mathbf{1}_K)(\boldsymbol{U}_{NP} \otimes \mathbf{I}_K)(\mathbf{1}_P \otimes \mathbf{I}_{NK})\text{vec}(\boldsymbol{G}) && \text{(A.5)}\\
\text{vec}(\boldsymbol{F} \odot \boldsymbol{G}) &= \text{diag}(\text{vec}(\mathbf{1}_P \otimes \boldsymbol{G}))(\mathbf{I}_{PN} \otimes \mathbf{1}_K)\,\text{vec}(\boldsymbol{F}) && \text{(A.6)}\\
\text{Tr}(\boldsymbol{F}\boldsymbol{Z}) &= \text{vec}(\boldsymbol{F}^{\mathsf{T}})^{\mathsf{T}}\text{vec}(\boldsymbol{Z}) && \text{(A.7)}\\
\boldsymbol{Z}(\boldsymbol{v} \boxdot \boldsymbol{e}) &= \boldsymbol{Z}\text{diag}(\boldsymbol{v})\,\boldsymbol{e} && \text{(A.8)}\\
\text{diag}(\text{vec}(\mathbf{1}_P \otimes \boldsymbol{F}))\text{vec}(\boldsymbol{L}^{\mathsf{T}}\boldsymbol{Z}) &= \text{vec}(\mathbf{1}_P \otimes \boldsymbol{F}) \boxdot \text{vec}(\boldsymbol{L}^{\mathsf{T}}\boldsymbol{Z}) && \text{(A.9)}\\
&= \text{vec}(\boldsymbol{L}^{\mathsf{T}}\boldsymbol{Z}) \boxdot \text{vec}(\mathbf{1}_P \otimes \boldsymbol{F}) && \text{(A.10)}\\
\text{vec}(\mathbf{1}_N \otimes \boldsymbol{Z}) &= (\boldsymbol{U}_{PN} \otimes \mathbf{I}_N)(\mathbf{1}_N \otimes \mathbf{I}_{NP})\text{vec}(\boldsymbol{Z}) && \text{(A.11)}\\
\text{vec}(\boldsymbol{Z} \boxdot \boldsymbol{H}) &= \text{diag}(\text{vec}(\boldsymbol{H}))\text{vec}(\boldsymbol{Z}) && \text{(A.12)}\\
&= \text{diag}(\text{vec}(\boldsymbol{Z}))\text{vec}(\boldsymbol{H}) && \text{(A.13)}\\
(\boldsymbol{F} \odot \boldsymbol{G})^{\mathsf{T}}(\boldsymbol{F} \odot \boldsymbol{G}) &= \boldsymbol{F}^{\mathsf{T}}\boldsymbol{F} \boxdot \boldsymbol{G}^{\mathsf{T}}\boldsymbol{G} && \text{(A.14)}
\end{aligned}$$

where $\boldsymbol{F} \in \mathbb{R}^{P\times N}, \boldsymbol{Z}, \boldsymbol{H} \in \mathbb{R}^{N\times P}, \boldsymbol{G} \in \mathbb{R}^{K\times N}, \boldsymbol{D} \in \mathbb{R}^{K\times M}, \boldsymbol{L} \in \mathbb{R}^{N\times NP}, \boldsymbol{w} \in \mathbb{R}^{N}$ and $\boldsymbol{v}, \boldsymbol{e} \in \mathbb{R}^{P}$.

Appendix B. Proof of lemma 1

Assume at this moment that matrix $\boldsymbol{A}$ is a real function in two real-valued matrices $\boldsymbol{A}_1 \in \mathbb{R}_+^{I_1\times P}$ and $\boldsymbol{A}_2 \in \mathbb{R}_+^{I_2\times P}$ with $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \boldsymbol{A}$. Then according to equations (A.5) and (A.6), we have:

$$\begin{aligned}
\text{vec}(\boldsymbol{Z}) &= \text{vec}(\boldsymbol{A}_1 \odot \boldsymbol{A}_2)\\
&= \text{diag}(\text{vec}(\mathbf{1}_{I_2} \otimes \boldsymbol{A}_2))(\mathbf{I}_{I_1 P} \otimes \mathbf{1}_{I_2})\text{vec}(\boldsymbol{A}_1) && \text{(B.1)}\\
&= \text{diag}(\text{vec}(\boldsymbol{A}_1) \otimes \mathbf{1}_{I_2})(\boldsymbol{U}_{I_2 P} \otimes \mathbf{I}_{I_2})(\mathbf{1}_{I_1} \otimes \mathbf{I}_{I_2 P})\text{vec}(\boldsymbol{A}_2) && \text{(B.2)}
\end{aligned}$$

where $\boldsymbol{A}_1 \in \mathbb{R}_+^{I_1\times P}$ and $\boldsymbol{A}_2 \in \mathbb{R}_+^{I_2\times P}$.
Then the differential of the vectorized form of $\boldsymbol{Z}$ is given by:

$$\text{dvec}(\boldsymbol{Z}) = D_{\boldsymbol{A}_1}\text{vec}(\boldsymbol{Z})\ \text{dvec}(\boldsymbol{A}_1) + D_{\boldsymbol{A}_2}\text{vec}(\boldsymbol{Z})\ \text{dvec}(\boldsymbol{A}_2) \qquad \text{(B.3)}$$

Then according to the last equation and equations (B.1) and (B.3), we have:

$$\mathrm{dvec}(\boldsymbol{Z}) = \mathrm{diag}(\mathrm{vec}(\mathbf{1}_{I_2} \otimes \boldsymbol{A}_2))(\mathbf{I}_{I_1 P} \otimes \mathbf{1}_{I_2})\,\mathrm{dvec}(\boldsymbol{A}_1) + \mathrm{diag}(\mathrm{vec}(\boldsymbol{A}_1) \otimes \mathbf{1}_{I_2})(\boldsymbol{U}_{I_2 P} \otimes \mathbf{I}_{I_2}) \times (\mathbf{1}_{I_1} \otimes \mathbf{I}_{I_2 P})\mathrm{dvec}(\boldsymbol{A}_2) \quad \text{(B.4)}$$

Now, by including the constraint $\boldsymbol{A}_1 = \boldsymbol{A}_2 = \boldsymbol{A}$ in the latter differential form and factorizing it w.r.t. $\mathrm{dvec}(\boldsymbol{Z})$, the gradient expression $D_{\boldsymbol{A}}\boldsymbol{Z}$ given in this lemma is proved.

Appendix C. Proof of lemma 2

As proceeded previously, we suppose at this moment that matrix $\boldsymbol{A}$ is is a real function in two real-valued matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ subject to $\boldsymbol{B}_1 = \boldsymbol{B}_2$. Then, according to equations A.12 and A.13, we have:

$$\begin{aligned} \mathrm{vec}(\boldsymbol{A}) = \mathrm{vec}(\boldsymbol{B}_1 \boxdot \boldsymbol{B}_2) &= \mathrm{diag}(\mathrm{vec}(\boldsymbol{B}_2))\,\mathrm{vec}(\boldsymbol{B}_1) && \text{(C.1)} \\ &= \mathrm{diag}(\mathrm{vec}(\boldsymbol{B}_1))\,\mathrm{vec}(\boldsymbol{B}_2) && \text{(C.2)} \end{aligned}$$

Then according to equations (C.1) and (C.2), the differential of $\boldsymbol{A}$ is given by:

$$\begin{aligned} \mathrm{dvec}(\boldsymbol{A}) &= D_{\boldsymbol{B}_1}\mathrm{vec}(\boldsymbol{A})\ \mathrm{dvec}(\boldsymbol{B}_1) + D_{\boldsymbol{B}_2}\mathrm{vec}(\boldsymbol{A})\ \mathrm{dvec}(\boldsymbol{B}_2) \\ &= \mathrm{diag}(\mathrm{vec}(\boldsymbol{B}_2))\,\mathrm{dvec}(\boldsymbol{B}_1) + \mathrm{diag}(\mathrm{vec}(\boldsymbol{B}_1))\,\mathrm{dvec}(\boldsymbol{B}_2) && \text{(C.3)} \end{aligned}$$

Now, by including the constraint $\boldsymbol{B}_1 = \boldsymbol{B}_2 = \boldsymbol{B}$ in the latter differential form and factorizing it w.r.t. $\mathrm{dvec}(\boldsymbol{B})$, the gradient expression $D_{\boldsymbol{B}}\boldsymbol{A}$ given in Lemma 2 is proved.

Appendix D. Proof of Lemma 3

According to the differential expression (18) and from equation (21), equations (24) and (25) are immediate. Now, we have

$$\begin{aligned} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) &= \mathrm{Tr}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}(\boldsymbol{A}\odot\boldsymbol{A})^\intercal) = \mathrm{Tr}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\boldsymbol{Z}^\intercal) = \mathrm{vec}\Big(\boldsymbol{C}^\intercal\boldsymbol{T}^{(3)}\Big)^\intercal\mathrm{vec}(\boldsymbol{Z}^\intercal) \\ &= \mathrm{vec}\Big(\boldsymbol{C}^\intercal\boldsymbol{T}^{(3)}\Big)^\intercal\boldsymbol{U}_{I^2P}\mathrm{vec}(\boldsymbol{Z}) = \mathrm{vec}\Big(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\Big)^\intercal\mathrm{vec}(\boldsymbol{Z}) \quad \text{(D.1)} \end{aligned}$$

Then:

$$\begin{aligned} \mathrm{d}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) &= \mathrm{vec}\Big(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\Big)^\intercal\,\mathrm{dvec}(\boldsymbol{Z}) \\ &= D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{Z}) = D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\,D_{\boldsymbol{A}}\boldsymbol{Z}\,\mathrm{dvec}(\boldsymbol{A}) \\ &= D_{\boldsymbol{Z}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\,D_{\boldsymbol{A}}\boldsymbol{Z}\,D_{\boldsymbol{B}}\boldsymbol{A}\,\mathrm{dvec}(\boldsymbol{B}) \\ &= D_{\boldsymbol{B}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{B}) \quad \text{(D.2)} \end{aligned}$$

Now using the results of both lemma 1 and lemma 2 we obtain:

$$D_{\boldsymbol{B}}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = 2\mathrm{vec}\Big(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\Big)^\intercal(\mathrm{diag}(\mathrm{vec}(\mathbf{1}_I\otimes\boldsymbol{A}))(\mathbf{I}_{IP}\otimes\mathbf{1}_N) + \mathrm{diag}(\mathrm{vec}(\boldsymbol{A})\otimes\mathbf{1}_I)\times (\boldsymbol{U}_{PI}\otimes\mathbf{I}_I)(\mathbf{1}_I\otimes\mathbf{I}_{IP}))\mathrm{diag}(\mathrm{vec}(\boldsymbol{B})) \quad \text{(D.3)}$$

With the aim of reducing the numerical complexity and dimensions of matrices used in the gradient expression, we simplify the expression using only small matrices:

$$D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = 2\left(\mathrm{vec}(\boldsymbol{B}) \boxdot (((\boldsymbol{A}\otimes \mathbf{1}_I{}^{\intercal}) \boxdot (\boldsymbol{M}+\boldsymbol{N}))^{\intercal}\mathbf{1}_I)\right)^{\intercal} \tag{D.4}$$

$$\tag{D.5}$$

with $\boldsymbol{M} = \mathrm{Mat}^{(I\times 1, IP)}(\mathrm{vec}\left(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\right))$ and $\boldsymbol{N} = \mathrm{Mat}^{(I\times I, P)}(\boldsymbol{M}^{\intercal})$.
This simplification is expressed using the algebraic basis of matrix calculus, some operator properties.

Let us now compute the gradient of the real function $g(\boldsymbol{B},\boldsymbol{C})$ w.r.t. $\boldsymbol{B}$.

$$g(\boldsymbol{B},\boldsymbol{C}) \quad = \quad \mathbf{1}_p{}^{\intercal}(\boldsymbol{Y}^{\boxdot 2} \boxdot \boldsymbol{C}^{\intercal}\boldsymbol{C})\mathbf{1}_p$$

where $\boldsymbol{Y} = \boldsymbol{A}^{\intercal}\boldsymbol{A}$. Then:

$$\begin{aligned} \mathrm{d}g(\boldsymbol{B},\boldsymbol{C}) &= D_{\boldsymbol{Y}} g(\boldsymbol{B},\boldsymbol{C})\, D_{\boldsymbol{A}}\boldsymbol{Y}\, D_{\boldsymbol{B}}\boldsymbol{A}\, \mathrm{dvec}(\boldsymbol{B}) \\ &= D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C})\, \mathrm{dvec}(\boldsymbol{B}) \end{aligned} \tag{D.6}$$

with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$ and $\boldsymbol{Y} = \boldsymbol{A}^{\intercal}\boldsymbol{A}$. By calculus, we obtain:

$$D_{\boldsymbol{Y}} g(\boldsymbol{B},\boldsymbol{C}) \quad = \quad 2\mathrm{vec}(\boldsymbol{Y} \boxdot \boldsymbol{C}^{\intercal}\boldsymbol{C}) \tag{D.7}$$

$$D_{\boldsymbol{A}}\boldsymbol{Y} \quad = \quad (\mathbf{I}_P \otimes \boldsymbol{A}^{\intercal}) + (\boldsymbol{A}^{\intercal}\otimes \mathbf{I}_P)\boldsymbol{U}PN \tag{D.8}$$

Now from both previous differential expressions and Lemma 2, we obtain the expression of $D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C})$. Then by multiplying equation (D.3) by a negative factor of two and adding it to equation (D.7), we obtain the gradient expression $D_{\boldsymbol{B}}\Psi(\boldsymbol{B},\boldsymbol{C})$.
We proceed in a similar manner to obtain the derivative w.r.t. $\boldsymbol{C}$: $D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$. We start by rewriting $f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$ and $g(\boldsymbol{B},\boldsymbol{C})$, to be able to derive in matrix form.

$$f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \mathrm{Tr}\left(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\boldsymbol{Z}^{\intercal}\right) = \mathrm{Tr}\left(\boldsymbol{Z}^{\intercal}\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\right) = \mathrm{vec}\left(\boldsymbol{T}^{(3)}\boldsymbol{Z}\right)^{\intercal}\mathrm{vec}(\boldsymbol{C}) \tag{D.9}$$

Then since we are interested in computing the gradient w.r.t. to $\boldsymbol{C}$, we will limit ourselves to compute the differential of $f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$ w.r.t. to $\boldsymbol{B}$:

$$\mathrm{d}f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \mathrm{vec}\left(\boldsymbol{T}^{(3)}\boldsymbol{Z}\right)^{\intercal}\mathrm{dvec}(\boldsymbol{C}) = D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\, \mathrm{dvec}(\boldsymbol{C}) \tag{D.10}$$

Let us now compute the gradient of the real function $g(\boldsymbol{B},\boldsymbol{C})$ w.r.t. $\boldsymbol{C}$.

$$\begin{aligned} g(\boldsymbol{B},\boldsymbol{C}) &= \mathrm{Tr}(\boldsymbol{Z}\boldsymbol{C}^{\intercal}\boldsymbol{C}\boldsymbol{Z}^{\intercal}) = \mathrm{Tr}(\boldsymbol{C}\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\boldsymbol{C}^{\intercal}) \\ &= \mathrm{vec}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\boldsymbol{C}^{\intercal})^{\intercal}\mathrm{vec}(\boldsymbol{C}^{\intercal}) \\ &= (\boldsymbol{U}_{KP}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\otimes \mathbf{I}_K)\,\mathrm{vec}(\boldsymbol{C}))^{\intercal}\boldsymbol{U}_{KP}\,\mathrm{vec}(\boldsymbol{C}) \\ &= \mathrm{vec}(\boldsymbol{C})^{\intercal}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\otimes \mathbf{I}_K)\,\mathrm{vec}(\boldsymbol{C}) \end{aligned} \tag{D.11}$$

Then since we are interested in computing the gradient w.r.t. to $\boldsymbol{C}$, we will limit ourselves to the differential of $g(\boldsymbol{B},\boldsymbol{C})$ w.r.t. to $\boldsymbol{C}$:

$$\mathrm{d}g(\boldsymbol{B},\boldsymbol{C}) = 2\mathrm{vec}(\boldsymbol{C})^{\intercal}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\otimes \mathbf{I}_K)\,\mathrm{dvec}(\boldsymbol{C}) = D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{C}) \tag{D.12}$$

Then premultiplying the term $D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$ by a negative factor of two and adding it to $D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C})$, we obtain the gradient expression $D_{\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$.

Appendix E. Proof of lemma 4

Using the same technique as that adopted to prove lemmas 1 and 2, the results of the latter, and using equations (A.1), (A.2) and (A.4), the vectorized expression of the matrix function $\boldsymbol{h}$ is modified to allow a derivation in matrix form.

$$\begin{aligned}
\boldsymbol{h}(\boldsymbol{B},\boldsymbol{C}) &= \text{vec}\Big(\boldsymbol{T}^{(3)} - \boldsymbol{C}(\boldsymbol{A}\odot\boldsymbol{A})^{\intercal}\Big) \\
&= \text{vec}\Big(\boldsymbol{T}^{(3)}\Big) - (\mathbf{I}_{I^2}\otimes\boldsymbol{C})\text{vec}((\boldsymbol{A}\odot\boldsymbol{A})^{\intercal}) \\
&= \text{vec}\Big(\boldsymbol{T}^{(3)}\Big) - (\mathbf{I}_{I^2}\otimes\boldsymbol{C})\boldsymbol{U}_{I^2P}\text{vec}(\boldsymbol{A}\odot\boldsymbol{A}) \qquad \text{(E.1)} \\
&= \text{vec}\Big(\boldsymbol{T}^{(3)}\Big) - ((\boldsymbol{A}\odot\boldsymbol{A})\otimes\mathbf{I}_K)\,\text{vec}(\boldsymbol{C}) \qquad \text{(E.2)}
\end{aligned}$$

Then, inserting the result of Lemma 1 in equation (E.1), equation (31) is readily obtained. Equation (32) then follows easily from equation (E.2)

Appendix F. Proof of Lemma 5

From the Jacobian expressions (31) and (32), we compute the term $\boldsymbol{J}^{\intercal}\boldsymbol{J}$, which yields:

$$\begin{aligned}
\boldsymbol{J}_{\boldsymbol{B}}{}^{\intercal}\boldsymbol{J}_{\boldsymbol{B}} &= 4\boldsymbol{W}^{\intercal}(\mathbf{I}_{I^2}\otimes\boldsymbol{C}^{\intercal}\boldsymbol{C})\boldsymbol{W} \qquad \text{(F.1)} \\
\boldsymbol{J}_{\boldsymbol{C}}{}^{\intercal}\boldsymbol{J}_{\boldsymbol{C}} &= (\boldsymbol{Z}^{\intercal}\boldsymbol{Z})\otimes\mathbf{I}_K \qquad \text{(F.2)} \\
\boldsymbol{J}_{\boldsymbol{B}}{}^{\intercal}\boldsymbol{J}_{\boldsymbol{C}} &= (\boldsymbol{J}_{\boldsymbol{B}}{}^{\intercal}\boldsymbol{J}_{\boldsymbol{C}})^{\intercal} \qquad \text{(F.3)}
\end{aligned}$$

where the second equation above can be found in [56], with $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, $\boldsymbol{Z} = \boldsymbol{A}^{\odot 2}$ and $\boldsymbol{W} = \boldsymbol{U}_{I^2P}[\text{diag}(\text{vec}(\boldsymbol{A}\otimes\mathbf{I}_I))(\mathbf{I}_{IP}\otimes\mathbf{1}_I) + \text{diag}(\text{vec}(\boldsymbol{A})\otimes\mathbf{1}_I)(\boldsymbol{U}_{PI}\otimes\mathbf{I}_N)(\mathbf{1}_I\otimes\mathbf{I}_{IP})]$. Based on the properties of Kronecker and Khatri-Rao products and using some matrix manipulations, the previous expression can be written in terms of matrices with dimensions less than or equal to max (IP, KP). This rearrangement permits in turn to deal with large dimensional problems. As a result, expressions (33), (34) and (35) are immediate.

Appendix G. Proof of Lemma 6

In order to compute the term $D^2_{\boldsymbol{B},\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$, we derive $(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\intercal}$ separately w.r.t. $\boldsymbol{B}$ and $\boldsymbol{A}$.

$$\begin{aligned}
(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\intercal} = 2\text{diag}(\text{vec}(\boldsymbol{B}))(\mathbf{1}_I{}^{\intercal}\otimes\mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes\mathbf{I}_I)\text{diag}(\text{vec}(\boldsymbol{A})\otimes\mathbf{1}_I) + (\mathbf{I}_{IP}\otimes\mathbf{1}_I{}^{\intercal}) \\
\text{diag}(\text{vec}(\mathbf{1}_I\otimes\boldsymbol{A}))\text{vec}\Big(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\Big) \qquad \text{(G.1)}
\end{aligned}$$

We shall first need the lemma below to derive $(D^{(1)}_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\intercal}$ w.r.t. $\boldsymbol{B}$..

Lemma 9. *Let $\boldsymbol{u}(\boldsymbol{B}) = \text{diag}(\text{vec}(\boldsymbol{B}))\boldsymbol{\omega}$ be a vector function in the matrix $\boldsymbol{B}$ of size $(I\times P)$ where $\boldsymbol{\omega}$ is an IP-dimensional vector, the gradient of $\boldsymbol{u}(\boldsymbol{B})$ w.r.t. $\boldsymbol{B}$ is then given by $D_{\boldsymbol{B}}\boldsymbol{u}(\boldsymbol{B}) = \text{diag}(\boldsymbol{\omega})$.*

The proof of this Lemma is immediate, using equation (A.12).

Now, by using the result of Lemma 9 in equation (G.1) and by inserting the properties (A.12) and (A.13) in equation (G.1) to derive w.r.t. $\boldsymbol{A}$, we have:

$$\begin{aligned}
&\mathrm{d}(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})^{\intercal}) = \mathrm{d}^2 f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \\
&2\mathrm{diag}(((\mathbf{I}_{IP}\otimes \mathbf{1}_I{}^{\intercal})\mathrm{diag}(\mathrm{vec}(\mathbf{1}_I\otimes \boldsymbol{A})) + (\mathbf{1}_I{}^{\intercal}\otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes \mathbf{I}_I)\mathrm{diag}(\mathrm{vec}(\boldsymbol{A})\otimes \mathbf{1}_I))\times \\
&\mathrm{vec}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{B}) + 2\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))((\mathbf{I}_{IP}\otimes \mathbf{1}_I{}^{\intercal})\mathrm{diag}(\mathrm{vec}(\boldsymbol{T}^{(3)}\boldsymbol{C}))(\boldsymbol{U}_{PI}\otimes \mathbf{I}_I)(\mathbf{1}_I\otimes \mathbf{I}_{IP}) \\
&+(\mathbf{1}_I{}^{\intercal}\otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes \mathbf{I}_I)\boldsymbol{U}_{PI}(\boldsymbol{U}_{PI}\otimes \mathbf{I}_I)(\mathbf{1}_I\otimes \mathbf{I}_{IP}))\,\mathrm{dvec}(\boldsymbol{A}) \qquad \text{(G.2)}
\end{aligned}$$

However $\boldsymbol{A} = \boldsymbol{B}^{\boxdot 2}$, then according to Lemma 2 we get:

$$\begin{aligned}
&\mathrm{d}(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})^{\intercal}) = \mathrm{d}^2 f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = \\
&2\mathrm{diag}(((\mathbf{I}_{IP}\otimes \mathbf{1}_I{}^{\intercal})\mathrm{diag}(\mathrm{vec}(\mathbf{1}_I\otimes \boldsymbol{A})) + (\mathbf{1}_I{}^{\intercal}\otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes \mathbf{I}_I)\mathrm{diag}(\mathrm{vec}(\boldsymbol{A})\otimes \mathbf{1}_I))\times \\
&\mathrm{vec}(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}))\,\mathrm{dvec}(\boldsymbol{B}) + 4\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))((\mathbf{I}_{IP}\otimes \mathbf{1}_I{}^{\intercal})\mathrm{diag}(\mathrm{vec}(\boldsymbol{T}^{(3)}\boldsymbol{C}))(\boldsymbol{U}_{PI}\otimes \mathbf{I}_I)(\mathbf{1}_I\otimes \mathbf{I}_{IP}) \\
&+(\mathbf{1}_I{}^{\intercal}\otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes \mathbf{I}_I)\boldsymbol{U}_{PI}(\boldsymbol{U}_{PI}\otimes \mathbf{I}_I)(\mathbf{1}_I\otimes \mathbf{I}_{IP}))\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))\,\mathrm{dvec}(\boldsymbol{B}) \\
=\;& D^2_{\boldsymbol{B},\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{B}) \qquad \text{(G.3)}
\end{aligned}$$

Recall that another expression (39) of $D^2_{\boldsymbol{B},\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$, written in terms of matrices of small dimensions, can be obtained using some matrix manipulations based on properties of Kronecker and Khatri-Rao products. Regarding the partial Hessian $D^2_{\boldsymbol{B},\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C})$, it is computed in the same manner.

Appendix H. Proof of Lemma 7

According to equation (25), the term $D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})$ is given by:

$$D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C}) = -2D^2_{\boldsymbol{C},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) + D^2_{\boldsymbol{C},\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}) \qquad \text{(H.1)}$$

with $D^2_{\boldsymbol{C},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = D_{\boldsymbol{C}}\left(D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})\right)^{\intercal}$ and $D^2_{\boldsymbol{C},\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}) = D_{\boldsymbol{C}}\left(D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C})\right)^{\intercal}$. Since according to Lemma 3, *i.e.* equation (28), $D_{\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$ does not depend on $\boldsymbol{C}$, then $D^2_{\boldsymbol{C},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}) = 0$. Thus the Hessian of Ψ given in (5) depends only on $D^2_{\boldsymbol{C},\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C})$. From equation (29) we have:

$$D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}) = 2\mathrm{vec}(\boldsymbol{C})^{\intercal}(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\otimes \mathbf{I}_K) \qquad \text{(H.2)}$$

Then:

$$\begin{aligned}
\mathrm{d}(D_{\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C}))^{\intercal} &= 2(\boldsymbol{Z}^{\intercal}\boldsymbol{Z}\otimes \mathbf{I}_K)\,\mathrm{dvec}(\boldsymbol{C}) \\
&= H_{\boldsymbol{C},\boldsymbol{C}} g(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{C}) = D^2_{\boldsymbol{C},\boldsymbol{C}}\Psi(\boldsymbol{B},\boldsymbol{C})\,\mathrm{dvec}(\boldsymbol{C})
\end{aligned} \qquad \text{(H.3)}$$

Hence the result.

Appendix I. Proof of Lemma 8

In order to compute the cross partial Hessian $D_{\boldsymbol{B},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})^{\intercal}$, we derive $(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\intercal}$ w.r.t. $\boldsymbol{C}$. Using the chain rules and equation (A.1), we obtain:

$$\begin{aligned}
(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\intercal} = -4\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))(\mathbf{1}_I{}^{\intercal}\otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP}\otimes \mathbf{I}_I)\mathrm{diag}(\mathrm{vec}(\boldsymbol{A})\otimes \mathbf{1}_I) + (\mathbf{I}_{IP}\otimes \mathbf{1}_I{}^{\intercal}) \\
\mathrm{diag}(\mathrm{vec}(\mathbf{1}_I\otimes \boldsymbol{A}))\mathrm{vec}\left(\boldsymbol{T}^{(3)\intercal}\boldsymbol{C}\right) \qquad \text{(I.1)}
\end{aligned}$$

$$(D_{\boldsymbol{B}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C}))^{\mathsf{T}} = -4\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))(\mathbf{1}_I{}^{\mathsf{T}} \otimes \mathbf{I}_{IP})(\boldsymbol{U}_{IP} \otimes \mathbf{I}_I)\mathrm{diag}(\mathrm{vec}(\boldsymbol{A}) \otimes \mathbf{1}_I) + (\mathbf{I}_{IP} \otimes \mathbf{1}_I{}^{\mathsf{T}})$$
$$\mathrm{diag}(\mathrm{vec}(\mathbf{1}_I \otimes \boldsymbol{A}))(\mathbf{I}_P \otimes \boldsymbol{T}^{(3)\mathsf{T}})\mathrm{vec}(\boldsymbol{C}) \qquad \text{(I.2)}$$

Hence the expression of $D^2_{\boldsymbol{B},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$. Moreover, as for $D_{\boldsymbol{B},\boldsymbol{C}} f_{\boldsymbol{T}^{(3)}}(\boldsymbol{B},\boldsymbol{C})$, the expression is reformulated, using the algebraic basis of matrix calculus and operator properties, in terms of small matrices (43)

$$D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C})^{\mathsf{T}} \quad = \quad 4\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))((\mathbf{I}_P \otimes \boldsymbol{A}) + \boldsymbol{U}_{PI}(\boldsymbol{A} \otimes \mathbf{I}_P))\mathrm{vec}(\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A} \odot \boldsymbol{C}^{\mathsf{T}}\boldsymbol{C})$$

So, we calculate:

$$\mathrm{d}(D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}))^{\mathsf{T}} \quad = \quad D_{\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}}(D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}))^{\mathsf{T}}\, D_{\boldsymbol{C}}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}\, \mathrm{dvec}(\boldsymbol{C}) \qquad \text{(I.3)}$$

We have:

$$D_{\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}}(D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}))^{\mathsf{T}} \quad = \quad 4\mathrm{diag}(\mathrm{vec}(\boldsymbol{B}))((\mathbf{I}_P \otimes \boldsymbol{A}) + \boldsymbol{U}_{PI}(\boldsymbol{A} \otimes \mathbf{I}_P))\mathrm{diag}(\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C})$$

and

$$D_{\boldsymbol{C}}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C} \quad = \quad (\mathbf{I}_P \otimes \boldsymbol{C}^{\mathsf{T}}) + (\boldsymbol{C}^{\mathsf{T}} \otimes \mathbf{I}_P)\boldsymbol{U}_{KP}$$

Inserting the two previous expressions $D_{\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}}(D_{\boldsymbol{B}} g(\boldsymbol{B},\boldsymbol{C}))$ and $D_{\boldsymbol{C}}\boldsymbol{C}^{\mathsf{T}}\boldsymbol{C}$, we obtain (43) and (44).

Acknowledgement

This work has been partly supported by the French ANR contract 10-BLAN-MULTIMODEL. We thank the reviewers for their thorough reading and helpful comments.

References

[1] J. CARROLL, J. CHANG, Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition, Psychometrika 35 (9) (1970) 267–283.

[2] S. HAJIPOUR SARDOUIE, L. ALBERA, M. B. SHAMSOLLAHI, I. MERLET, Canonical Polyadic Decomposition of Complex-Valued Multi-Way Arrays based on Simultaneous Schur Decomposition, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013.

[3] F. L. HITCHCOCK, The expression of a tensor or a polydiac as a sum of products, J. Math. Phys. 6 (1) (1927) 164–189.

[4] T. G. KOLDA, B. W. BADER, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500.

[5] X. LUCIANI, L. ALBERA, Semi-algebraic canonical decomposition of multi-way arrays and joint eigenvalue decomposition, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2011, pp. 4104–4107.

[6] Y. TAKANE, Applications of multidimensional scaling in psychometrics., In Rao, C. R., and Sinharay, S. (Eds.), Handbook of Statistics (Vol. 26): Pyschometrics, (pp. 359-400). Amsterdam: Elsevier., 2007.

[7] R. N. SHEPARD, Multidimensional scaling, tree-fitting and clustering, Science 210 (1980) 390–398.

[8] C. J. DOUGLAS, P. E. GREEN, An indscal-based approach to multiple correspondence analysis, Journal of Marketing Research 25 (25) (1988) 193.

[9] A. YEREDOR, Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation, IEEE Transactions On Signal Processing 50 (7) (2002) 1545–1553.

[10] F. HUSSON, J. PAGES, Indscal model: geometrical interpretation and methodology, Computational Statistics and Data Analysis 50 (2) (2006) 358 – 378. doi:DOI: 10.1016/j.csda.2004.08.005.

[11] A. KARFOUL, L. ALBERA, G. BIROT, Blind underdetermined mixture identification by joint canonical decomposition of ho cumulants, IEEE Transactions on Signal Processing 58 (2010) 638 – 649.

[12] K. DONG-GOO, S. YOUNGHUM, K. SUNGSU, L. SEONGDEOK, K. CHANGYEONG, Multiple object decomposition based on independent component analysis of multi-energy x-ray projections, International Conference on Image Processing ICIP (2010) 4173–4176.
[13] P. COMON, Independent Component Analysis, a new concept ?, Signal Processing, Elsevier 36 (3) (1994) 287–314.
[14] L. ALBERA, P. COMON, L. C. PARRA, A. KARFOUL, A. KACHENOURA, L. SENHADJI, Comparative performance study of independent component analysis methods on biomedical signals, in Handbook of Blind Source Separation, P. Comon and C. Jutten Eds, Academic Press, 2010.
[15] M. CONGEDO, C. GOUY-PAILLER, C. JUTTEN, On the blind source separation of human electroencephalogram by approximate joint diagonalization of second order statistics, Clinical Neurophysiology 119 (12) (2008) 2677–2686.
[16] A. KACHENOURA, L. ALBERA, L. SENHADJI, P. COMON, ICA: a potential tool for BCI systems, IEEE Signal Processing Magazine, special issue on Brain-Computer Interfaces 25 (1) (2008) 57–68.
[17] L. D. LATHAUWER, B. DE MOOR, J. VANDEWALLE, Fetal electrocardiogram extraction by blind source subspace separation, IEEE Transactions on Biomedical Engineering 47 (5) (2000) 567–572.
[18] R. VOLLGRAF and K. OBERMAYER, Quadratic optimization for simultaneous matrix diagonalization, IEEE Transactions on Signal Processing 54 (9) (2006) 3270–3278.
[19] A. ZIEHE, P. LASKOV, G. NOLTE and K.-R. MULLER, A fast algorithm for joint diagonalization with non-orthogonal transformation and its application to blind source separation, Journal of Machine Learning Research 5 (2004) 801–818.
[20] L. GIULIERI, H. GHENNIOUI, N. THIRION-MOREAU and E. MOREAU, Nonorthogonal joint diagonalization of spatial quadratic time-frequency matrices for source separation., IEEE Signal Processing Letters 12 (5) (2005) 415–418.
[21] J.-F. CARDOSO and A. SOULOUMIAC, Blind beamforming for non-gaussian signals, IEE Proceesings-F 140 (6) (1993) 362–370.
[22] A. BELOUCHRANI and A. CICHOCKI, Robust whitening procedure in blind source separation context, Electronics Letters 36 (2000), 2050–2053.
[23] S. DEGERINE and E. KANE, A comparative study of approximate joint diagonalization algorithms for blind source separation in presence of additive noise, IEEE Transactions on Signal Processing 55 (6) (2007) 3022–3031
[24] L. ALBERA, A. KACHENOURA, P. COMON, A. KARFOUL, F. WENDLING, L. SENHADJI, I. MERLET, ICA-based eeg denoising : a comparative analysis of fifteen methods, The special issue of the Bulletin of the Polish Academy of Science 60 (3) (2012) 737–777.
[25] F. ASANO, S. IKEDA, M. OGAWA, H. ASOH, N. KITAWAKI, Combined approach of array processing and independent component analysis for blind separation of acoustic signals, IEEE Transactions On Speech and Audio Processing 11 (3) (2003) 204–215.
[26] D. BARRY, D. FITZ, E. COYLE, B. LAWLOR, Single channel source separation using short-time independent component analysis, Audio Engineering Society, 119's convention (2005) 1–6.
[27] M. A. CASEY, A. WESTNER, Separation of mixed audio sources by independent component analysis, Tech. rep., Mistubishi electric research labs, cambridge, MA, USA (2000).
URL `http://www.merl.com/publications/TR2001-031/`
[28] J. C. BROWN, P. SMARAGDIS, Independent component analysis for automatic note extraction from musical trills, Acoustical Society of America 115 (5) (2004) 2295–2306.
[29] C. LADROUE, F. A. HOWE, J. R. GRIFFITHS, A. R. TATE, Independent component analysis for automated decomposition of in vivo magnetic resonance spectra, Magnetic Resonance in Medecine 50 (2003) 697–703.
[30] A. J. WRIGHT, G. FELLOW, T. J. BYRNES, K. S. OPSTAD, D. J. O. MCINTYRE, J. R. GRIFFITHS, B. A. BELL, C. A. CLARK, T. R. BARRICK, F. A. HOWE, Pattern recognition of mrsi data shows regions of glioma growth that agree with DTI markers of brain tumor infiltration., Magnetic Resonance in Medicine 62 (2009) 1646–1651.
[31] J. PULKKINEN, A.-M. HAKKINEN, N. LUNDBOM, A. PAETAU, R. A. KAUPPINEN, Y. HIKTUNEN, Independent component analysis to proton spectroscopic imaging data of human brain tumours, European Journal of Radiology 56 (2005) 160–164.
[32] P. CHEVALIER, A. CHEVREUIL, Application to telecommunications, in a handbook of blind source separation, P. Comon and C. Jutten Eds, Academic Press, 2010.
[33] A. TONAZZINI, L. BEDINI, E. SALERNO, Independent component analysis for document restoration, International Journal on Document Analysis and Recognition IJDAR (7) (2004) 17–27.
[34] A. HYVARINEN, Fast and robust fixed-point algorithms for independent component analysis, IEEE Transactions On Neural Networks 10, no. 3 (1999) 626–634.
[35] J. PULKKINEN, A. M. HAKKINEN, N. LUNDBOM, A. PAETAU, R. A. KAUPPINEN, Y. HILTUNEN, Independent component analysis to proton spectroscopic imaging data to human brain tumours, European Journal of

Radiology 56 (2) (2005) 160–164.
[36] F. SZABO DE EDELENYI, A. SIMONETTI, G. POSTMA, R. HUO, L. BUYDENS, Application of independent component analysis to 1H MR spectroscopic imaging exams of brain tumours, Analytica Chimica Acta 544 (1-2) (2005) 36–46.
[37] J.-F. CARDOSO, A. SOULOUMIAC, Jacobi angles for simultaneous diagonalization, SIAM Journal Matrix Analysis and Applications 17 (1) (1996) 161–164.
[38] J. HAO, X. ZOU, P. M. WILSON, N. P. DAVIES, Y. SUN, C. A. PEET, T. N. ARVANITIS, A comparative study of feature extraction and blind source separation of independent component analysis (ICA) on childhood brain tumour 1H magnetic resonance spectra, NMR in Biomedicine 22 (8) (2009) 809–818.
[39] P. McCULLAGH, Tensor Methods in Statistics, Chapman and Hall, Monographs on Statistics and Applied Probability, 1987.
[40] J. W. BREWER, Kronecker products and matrix calculus in system theory, IEEE Transactions On Circuits and Systems 25 (9) (1978) 114–122.
[41] P. COMON, X. LUCIANI, A. L. F. D. ALMEIDA, Tensor decomposition, alternating least squares and other tales, Journal of Chemometrics 23 (393-405) (2009) 393–405.
[42] L. D. LATHAUWER, B. DE MOOR, J. VANDEWALLE, A multilinear singular value decomposition, SIAM J. matrix Ana. Appl 21 (4) (2000) 1253–1278.
[43] P. COMON, Tensor decompositions, in: J. G. McWhirter, I. K. Proudler (Eds.), Mathematics in Signal Processing V, Clarendon Press, Oxford, UK, 2002, pp. 1–24.
[44] L. D. LATHAUWER, A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization, SIAM Journal in matrix Analysis and Applications 28 (3) (2006) 642–666.
[45] P. COMON, Tensors, usefulness and unexpected properties, in: 15th IEEE Workshop on Statistical Signal Processing (SSP'09), Cardiff, UK, 2009, pp. 781–788, keynote. hal-00417258. doi:10.1109/SSP.2009.5278471.
[46] P. COMON, J. M. F. T. BERGE, L. D. LATHAUWER, J. CASTAING, Generic and typical ranks of multi-way arrays, Linear Algebra and its Applications 430 (11-12) (2009) 2997–3007, hal-00410058.
[47] M. RAJIH, P. COMON, R. A. HARSHMAN, Enhanced line search : A novel method to accelerate PARAFAC, SIAM Journal on Matrix Analysis Appl. 30 (3) (2008) 1148–1171. doi:10.1137/06065577.
URL `http://link.aip.org/link/?SML/30/1128/1`
[48] A. STEGEMAN, J. M. F. T. BERGE, L. D. LATHAUWER, Sufficient conditions for uniqueness in CANDECOMP/PARAFAC and indscal with random component matrices, The Psychometric Society 71 (2) (2006) 219–229.
[49] M. CHU, F. DIELE, R. PLEMMONS, S. RAGNI, Optimality, computation and interpretation of nonnegative matrix factorizations, Tech. rep., Wake Forest University (2004).
URL `http://www.wfu.edu/ plemmons`
[50] J.-P. ROYER, N. THIRION-MOREAU, P. COMON, Computing the polyadic decomposition of nonnegative third order tensors, Signal Processing 91 (9) (2011) 2159–2171. doi:10.1016/j.sigpro.2011.03.006.
[51] P. PAATERO, A weighted non-negative least squares algorithm for three-way 'parafac' factor analysis, Chemometrics and Intelligent Laboratory Systems 38 (1997) 223–242.
[52] R. BRO, Multi-way analysis in the food industry: Models, algorithms, and applications, Ph.D. thesis, University of Amsterdam (1998).
[53] R. A. HARSHMAN, Foundation of PARAFAC procedure: Models and conditions for an 'explanatory' multi-mode factor analysis, UCLA working papers in Phonetics (16) (1970) 1–84.
[54] P. K. HOPKE, P. PAATERO, H. JIA, R. T. ROSS, R. A. HARSHMAN, Three-way (parafac) factor analysis: Examination and comparison of alternative computational methods as applied to ill-conditioned data, Chemometrics and Intelligent Laboratory Systems 43 (1-2) (1998) 25 – 42. doi:DOI: 10.1016/S0169-7439(98)00077-X.
[55] A. FRANC, Etude algebrique des multitableaux: Apport de l'algebre tensorielle, Ph.D. thesis, University of Montpellier II (1992).
[56] G. TOMASI, Practical and computational aspects inchemometric data analysis, Ph.D. thesis, University of Frederiksberg, Denmark (2006).
[57] K. MADSEN, H. B. NIELSEN, O. TINGLEFF, Methods for non-linear least squares problems, Informatics and Mathematical Modeling, Technical University of Denmark.
URL `Available online: http://www.imm.dtu.dk`
[58] A. HJORUNGNES, D. GESBERT, Complex-valued matrix differentiations: techniques and key results, IEEE Transactions on Signal Processing 55 (6) (2007) 2740–2746.
[59] J. TENDEIRO, M. B. DOSSE, J. M. F. T. BERGE, First and second-order derivatives for CP and indscal, Chemometrics and Intelligent Laboratory System 106 (2011) 27–36. doi:10.1016/j.chemolab.2010.05.013.
[60] J. R. MAGNUS, H. NEUDECKER, Matrix differential calculus with applications in statistics and econometrics, Wiley, 2007, third Edition.
[61] J. NOCEDAL, S. J. WRIGHT, Numerical optimization second edition, Springer, 2006.

[62] G. TOMASI, R. BRO, Parafac and missing values, Chemometrics and Intelligent Laboratory systems 75 (2005) 163–180.
[63] W. RAYENS, B. MITCHELL, Two-factor degeneracies and a stabilization of parafac, Chemometrics and Intelligent Laboratory Systems 38 (1997) 173–181.
[64] G. TOMASI, R. BRO, A comparison of algorithms for fitting the parafac model, Computational Statistics and Data Analysis 50 (7) (2006) 1700–1734.
[65] A. J. BELL, T. J. SEJNOWSKI, An information-maximization approach to blind separation and blind deconvolution, Neural Computation 7 (1995) 1129–1159.
[66] R. BRO, S. D. JONG, A fast non-negativity-constrained least squares algorithm, Journal of Chemometrics 11 (1999) 393–401.
[67] L. ALBERA, A. FERREOL, P. COMON, P. CHEVALIER, Blind Identification of Overcomplete Mixtures of sources (BIOME), Linear Algebra Applications 391C (2004) 3–30.
[68] P. CHEVALIER, A. FERREOL, L. ALBERA, G. BIROT, Higher order direction finding from arrays with diversely polarized antennas: The PD-2q-MUSIC algorithms, IEEE Transactions on Signal Processing 55 (11) (2007) 5337–5350.
[69] F. SAMARIA, A. HARTER, Parametrisation of a stochastic model for human face identification, Second IEEE Workshop on Applications of Computer Vision (1994), Sarasota, US.
[70] A. BELOUCHRANI, K. ABED-MERAIM, J.-F. CARDOSO and E. MOULINES, A blind source separation technique using second-order statistics, IEEE Transactions On Signal Processing 45 (2) (1997), 434–444.