DECOMPOSITION OF SEMI-NONNEGATIVE SEMI-SYMMETRIC TENSORS

A. Karfoul, J. Coloigner, L. Albera, P. Comon

Equipe SIGNAL

Rapport de recherche
ISRN I3S/RR–2010-11–FR

Novembre 2010
Résumé :

Mots clés :
Décomposition canonique, polyadique, factorisation non negative, tenseur, INDSCAL, Parafac, optimisation nonlinéaire.

Abstract:
The third order tensors we are interested in are real, both nonnegative and symmetric in two modes. The nonnegativity constraint is imposed by means of changes of variable into squares, leading to an unconstrained problem. In addition, a global plane search scheme is defined. The goal of the latter is to escape from local minima, which may be encountered in the search directions defined by the optimization method under consideration. This original scheme permits to compute algebraically and jointly two stepsizes, which is made possible thanks to the particular partial symmetry of tensors under study. Moreover, we provide a compact matrix form of derivatives of the objective function, which allows for a direct implementation of iterative algorithms such as gradient descent, Levenberg-Marquardt and Newton methods in matrix programming environments. Our numerical results show the benefit of our global plane search scheme, combined with a priori information such as nonnegativity and partial symmetry, especially in difficult contexts.

Key words:
Canonical decomposition, canonical polyadic, nonnegative matrix factorization, nonnegative tensor factorization, INDSCAL analysis, Parafac, nonlinear optimization.
Decomposition of semi-nonnegative semi-symmetric tensors

Ahmad Karfoul\textsuperscript{a}, Julie Coloigner\textsuperscript{b,c}, Laurent Albera\textsuperscript{b,c}, Pierre Comon\textsuperscript{d}

\textsuperscript{a}AL-Baath University, Faculty of Mech. and Elec. Eng., PB. 2244, Homs, Syria
\textsuperscript{b}Inserm, UMR 642, Rennes, F-35000, France
\textsuperscript{c}LTSI Laboratory, University of Rennes 1, Rennes, F-35000, France
\textsuperscript{d}I3S, University of Nice, 2000 route des Lucioles, BP 121, 06903 Sophia-Antipolis, France

Abstract

The third order tensors we are interested in are real, both nonnegative and symmetric in two modes. The nonnegativity constraint is imposed by means of changes of variable into squares, leading to an unconstrained problem. In addition, a global plane search scheme is defined. The goal of the latter is to escape from local minima, which may be encountered in the search directions defined by the optimization method under consideration. This original scheme permits to compute algebraically and jointly two stepsizes, which is made possible thanks to the particular partial symmetry of tensors under study. Moreover, we provide a compact matrix form of derivatives of the objective function, which allows for a direct implementation of iterative algorithms such as gradient descent, Levenberg-Marquardt and Newton methods in matrix programming environments. Our numerical results show the benefit of our global plane search scheme, combined with a priori information such as nonnegativity and partial symmetry, especially in difficult contexts.

Keywords: Canonical decomposition, canonical polyadic, nonnegative matrix factorization, nonnegative tensor factorization, INDSCAL analysis, Parafac, nonlinear optimization.

1. Introduction

Tensors of order $d$ are defined as multilinear maps, but once the bases of the underlying spaces are fixed, they may just be seen as arrays of coordinates [1]. The latter are often referred to as $d$-way arrays, since they have $d$ indices [2]. Therefore they can be seen as Higher Order (HO) equivalents of matrices, which are arrays of order 2. They often appear in the context of multiway factor analysis [2] when available data measurements can be arranged in a meaningful tensor form [1] such as in chemometrics [2], psychometrics [3], telecommunications [4, 5] and biomedical signal processing [6, 7, 8]. Even in the poor cases, that is, when the observation diversity is not sufficient, meaningful HO arrays can still be formed by resorting to HO Statistics (HOS) of the data [1]. In this case, HO arrays have a special algebraic structure since basic HOS (moments, cumulants, spectra, etc.) are nothing else but symmetric HO arrays. The latter are widely used in the context of Independent Component Analysis (ICA) [9, 10, 11, 12, 13, 14] and
more generally in the context of sensor array processing [15]. HO array factorizations permit to recover the latent factors in the available HO data arrays that explain the physical phenomena we observe. Among existing reliable HO array decompositions, the Canonical Polyadic (CP) decomposition turns out to be very attractive, due to its uniqueness properties, ensured under the sufficient conditions established by Kruskal [16], among others.

A polyadic decomposition of an array is a sum of rank-one terms that yields an exact fit [17]. The CP decomposition is defined as the minimal polyadic decomposition. The polyadic decomposition was rediscovered around 1970, in psychometrics [3] where the acronym CanDecomp is employed and in phonetics [18] where the term ParaFac (Parallel Factor analysis) is used instead.

Currently, CP decomposition is gaining importance in several applications such as exploratory data analysis [19], food industry [20], sensor array processing [21], telecommunications [4, 14], ICA [13, 11, 22] and recently in tracking the Direction of Arrival (DOA) and Direction Of Departure (DOD) of multiples targets in Multiple-Input Multiple-Output (MIMO) RADAR system [23]. Individuals Differences in SCALing (INDSCAL) analysis was proposed by Carroll and Chang [3] as a special case of CP decomposition for three-way arrays that are symmetric in two modes; i.e. when two of the loading matrices are equal. This situation can be encountered when dealing for instance with HOS. It is worth noticing that all algorithms dedicated to CP decomposition can be applied to INDSCAL, according the Caroll and Chang conjecture, but in practice they may fail when there is no unique solution [24]. On the other hand, various algorithms have been proposed imposing the symmetry constraint of the INDSCAL decomposition such as [22, App.A][25, 26] to cite a few. In addition, further constraints can be imposed to CP or INDSCAL decompositions, such as nonnegativity, especially when underlying components have a physical meaning. For example in image processing and computer vision, involved variables and parameters correspond to pixels and are indeed nonnegative [27].

Nonnegative tensor decomposition, or commonly speaking Nonnegative Tensor Factorization (NTF), is defined when all components in the tensor decomposition are constrained to be nonnegative. When this constraint is relaxed for some but not all components of this decomposition, the decomposition is called semi-nonnegative [27]. (Semi-) NTF is a straightforward terminology extension of the (semi-)Nonnegative Matrix Factorization (NMF). Most algorithms devised for (semi-) NMF can be generalized to (semi-) NTF [28] using simply the matricizing form of the considered HO array. The first work on NTF can be traced back to Carroll et al. [29] and later to Bro and Jong [30], who used an accelerated version of the Nonnegativity Constrained Least Squares (NNLS) technique, originally suggested by Lauwson and Hanson [31]. It is noteworthy that the NNLS technique is an active-set algorithm, which yields the least squares solution subject to the nonnegativity constraints, within a finite number of steps [32]. Furthermore, Paatero in [33] generalized his earlier 2-way positive matrix factorization (PMF) algorithm to the 3-way CP model referring to as PMF3. In fact since the pioneer works of Paatero and Tapper [34, 33] in 1994 on NMF, a variety of algorithms were proposed for NMF/NTF.

First, consider the multiplicative update algorithms originated with Lee & Seung [35, 36]. This class of techniques suffers from multiple drawbacks such as slowness and convergence to local minima. Indeed, it was shown that the multiplicative techniques require, when they converge, more iterations than the gradient descent and the Alternating Least Squares (ALS) algorithms. Furthermore, it was also shown that a stationary point to which these technique may converge is not necessarily a local minimum [37]. Authors in [38] succeeded in increasing the convergence speed, however the convergence to a local minima is still not guaranteed. On the other hand in [39], the problem of convergence to a local minimum is solved, at the price of a
high computational burden. Second, in [40, 41], nonnegativity is ensured by a projection step in gradient descent methods. More precisely, after each update rule, the updated matrices are projected to the nonnegative orthant by setting all negative elements to zero. The main drawback of such an algorithm is its slowness, which is merely related to the crucial choice of the step size. In addition, it is very sensitive to initialization [36]. Quasi-Newton methods [27, Chapter 6] perform comparatively better.

The third class is that of ALS algorithms. They transform the minimization problem into convex subproblems, where each one is solved for one factor while fixing the others to their last estimates. These algorithms are not guaranteed to converge to a global minimum (see [28] and the references therein). Recently a solution was proposed by Cichocki & Phan [28] that accelerates the convergence of the ALS algorithm. This solution is based on the minimization of a set of local cost functions with same global minima.

In this paper, we propose to compute the CP decomposition of third order real tensors, which are both nonnegative and symmetric in two out of three modes. The nonnegativity constraint is imposed by means of changes of variable into squares, leading to an unconstrained problem. In addition, a global two-dimensional line search scheme is described. The goal of the latter is to escape from local minima, which may be encountered in the search directions defined by the optimization method under consideration. This original scheme permits to compute algebraically and jointly two stepsizes, which is made possible thanks to the particular partial symmetry of tensors under study. Moreover, we provide a compact matrix form of derivatives of the considered objective function, which allows for a direct implementation of iterative algorithms such as Gradient Descent (GD), Levenberg-Marquardt (LM) and Newton methods in matrix programming environments, such as [42]. Our numerical results show the benefit behind our global line search scheme, combined with the use of some a priori information such as nonnegativity and symmetry, especially in difficult contexts.

2. Notations

Throughout this paper, vectors, matrices and tensors are denoted with bold lowercase letters \((a, b, \cdots)\), with bold uppercase letters \((A, B, \cdots)\) and with bold calligraphic letters \((\mathcal{A}, \mathcal{B}, \cdots)\), respectively. In addition, lower order parts of a given structure are considered as scalars and follow as a result the same notation as their structures. For instance, the entry with row \(i\) and column \(j\) in a matrix \(A\) is symbolized by \(A_{i,j}\) and the \((i_1, i_2, i_3)\)-th component of a third order array \(B\) is symbolized by \(B_{i_1,i_2,i_3}\). Sometimes we will use the MATLAB column/row notation to indicate submatrices of a given matrix or subarrays of a HO array, e.g. \(A_{1:p,q:}^\prime\). Upper bounds (i.e., \(n = 1, 2, \cdots, N\)) will be denoted by italic capital letters. \(\otimes\) denotes the Kronecker product while \(\odot\) stands for the Khatri-Rao product (column-wise Kronecker product), and \(\boxplus\) denotes the Hadamard product (element-wise product). Furthermore, the subscripts \(^2\) and \(^\dagger\) stand for the Moore-Penrose pseudoinverse and the transpose operators, respectively. In addition, the \((N \times N)\) identity matrix is denoted by \(I_N\) while \(\text{Tr}(Z)\) denotes the trace of the square matrix \(Z\). Besides, \(1_N\) stands for an \(N\)-dimensional vector of ones. Also, \(A^{\otimes 2} = A \odot A\) and \(A^{\boxplus 2} = A \boxplus A\). Finally \(|\cdot|\) and \(\|\cdot\|_F\) stand for the absolute value and the Frobenius norm, respectively.
3. Preliminaries and problem formulation

This section is devoted to some basic definition in multilinear algebra which are required for the problem formulation. Further related definitions can be found in [1, 43, 44, 45, 46].

Definition 1. The outer product $\mathbf{T} = \mathbf{u}^{(1)} \circ \cdots \circ \mathbf{u}^{(q)}$ of $q$ ($q \geq 2$) vectors $\mathbf{u}^{(i)} \in \mathbb{R}^{N_i}$ ($1 \leq i \leq q$) is a $q$-th order tensor of $\mathbb{R}^{N_1 \times \cdots \times N_q}$, whose elements are defined by $T_{n_1, \cdots, n_q} = u^{(1)}_{n_1} \cdots u^{(q)}_{n_q}$.

Definition 2. Each $q$-th ($q \geq 2$) order tensor $\mathbf{T}$ expressed as the outer product of $q$ vectors, is a $q$-th order rank-1 tensor.

Then the rank of a tensor always exists and is defined by:

Definition 3. The rank of a $q$-th order ($q \geq 2$) tensor $\mathbf{T} \in \mathbb{R}^{N_1 \times \cdots \times N_q}$, denoted by $\text{rk}(\mathbf{T})$, is the minimal number of rank-1 $q$-th order tensors that yield $\mathbf{T}$ in a linear combination.

Despite the similarity between matrices and tensors, they differ in a number of their properties [47]. For instance, contrary to the matrix case, tensor rank can exceed the smallest dimension [48]. From here on, only third-order tensors will be considered. From definitions 1 and 3, the CP decomposition of a 3-th order tensor is defined as follows [46, 49, 45]:

Definition 4. The CP decomposition of $\mathbf{T} \in \mathbb{R}^{I \times J \times K}$ is the linear combination of $P = \text{rk}(\mathbf{T})$ rank-1 third order arrays that yields $\mathbf{T}$ exactly:

$$\mathbf{T} = \sum_{p=1}^{P} \mathbf{a}_p \circ \mathbf{b}_p \circ \mathbf{c}_p$$

where the three matrices $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_P]$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_P]$ and $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_P]$ are called the loading matrices of $\mathbf{T}$. Sometimes we use the notation $\mathbf{T}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ to refer to the third array $\mathbf{T}$ with its loading matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

Then the INDSCAL decompositon is defined as follows [50]:

Definition 5. A 3-th order tensor $\mathbf{T} \in \mathbb{R}^{I \times J \times K}$ admits an INDSCAL decomposition if it can be decomposed according to definition 4 but with $\mathbf{A}$ being equal to $\mathbf{B}$.

It is clear from the latter definition that the INDSCAL decomposition is a special case of the CP decomposition for three-way arrays, which are symmetric in two of three modes. Let’s now define two basic operators, on which our subsequent results are based.

Definition 6. Let $\mathbf{T}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ be a 3-th order tensor of size $(I \times J \times K)$. Then the unfolding matrix, $\mathbf{T}^{(i)}$, associated with the $i$-th ($i \in \{1, 2, 3\}$) mode can be written using the HO array-to-matrix operator, named $\text{mat}^{(i)}$, as follows:

$$\mathbf{T}^{(1)}_{I \times JK} = \text{mat}^{(1)}(\mathbf{T}) = \mathbf{A}(\mathbf{C} \circ \mathbf{B})^t$$

$$\mathbf{T}^{(2)}_{J \times KL} = \text{mat}^{(2)}(\mathbf{T}) = \mathbf{B}(\mathbf{A} \circ \mathbf{C})^t$$

$$\mathbf{T}^{(3)}_{K \times IL} = \text{mat}^{(3)}(\mathbf{T}) = \mathbf{C}(\mathbf{B} \circ \mathbf{A})^t$$

(2)
Definition 7. Let $T$ be a 3-way array of size $(I \times J \times K)$. Then the components of the $1JK$-dimensional vector $\text{vec}(T)$ are defined by:

$$\text{vec}(T)_{i+j+(k-1)J} = T_{i,j,k}$$

Conversely, let $\text{unvec}$ be the inverse operator such as $\text{unvec}(\text{vec}(T)) = T$.

According to definition 5, the problem we tackle in this paper is defined as follows:

Problem 1. Given $T \in \mathbb{R}^{I \times J \times K}$, find its INDSCAL decomposition:

$$T = \sum_{p=1}^{P} z_p \odot z_p \odot c_p$$

subject to $Z = [z_1, \ldots, z_P]$ having nonnegative components.

As a way of including the nonnegative constraint in the required decomposition above, one can resort to the change of variable $Z = A \boxtimes A$ with $A \in \mathbb{R}^{I \times P}$ as originally proposed by Chu et al. to solve the NMF problem [51]. Then the above constrained problem 1 can be reformulated as an unconstrained one:

Problem 2. Let $T(Z, Z, C) \in \mathbb{R}^{I \times J \times K}$, find the following decomposition:

$$T = \sum_{p=1}^{P} (a_p \boxtimes a_p) \odot (a_p \boxtimes a_p) \odot c_p$$

Fitting the INDSCAL decomposition (5) is possible by minimizing the fitting error between the left and right hand sides of equation (5). Then the unconstrained optimization problem is totally characterized by the objective function $\Psi$ written in a matrix form as follows:

$$\Psi(A, C) = \| T^{(3)} - C(Z \odot Z)^\dag \|_F^2$$

with $Z = A \boxtimes A$. Minimizing this objective function to solve problem 2 is the main goal of this paper. To do so, several solutions are proposed, which are based on the use of first and second order derivatives of the considered cost function (see section 6), such as the GD, LM and Newton methods. However, these iterative algorithms, especially GD and Newton, suffer generally, even with a line search scheme, from local minima encountered in a given search direction. Then, looking for a good way to escape from these local minima and reaching the global optimum in that direction is quite appealing. This is the subject of the next section.

4. Global plane search procedure

In optimization, choosing an appropriate stepsize in the search direction is crucial. A line search procedure is defined by:

$$M^n = M^{n-1} + \mu^n M_{M} G_{M}^{n-1}$$

where $\mu_M$ is the stepsize to be computed, and $G_M$ a given search direction. A very simple line search scheme was proposed by Harshman and Bro [20, 18] wherein the stepsize was fixed to an arbitrary value changing with the number of iterations.
In this paper we shall be concerned by iterative algorithms of a special form, needing two stepsizes, \( \mu_A \) and \( \mu_C \):
\[
\begin{bmatrix}
A^{it} \\
C^{it}
\end{bmatrix} = \begin{bmatrix}
A^{it-1} \\
C^{it-1}
\end{bmatrix} + 
\begin{bmatrix}
\mu_A^i L_i & 0 \\
0 & \mu_C^i L_k
\end{bmatrix}
\begin{bmatrix}
G_A^{it-1} \\
G_C^{it-1}
\end{bmatrix}
\]
where \( G_A \) and \( G_C \) are two search directions specified by the iterative algorithm minimizing the objective function (6) w.r.t. its matrix variables \( A \) and \( C \), respectively. These directions could be for instance the opposite of the gradient of the objective function, as in the GD algorithm, or more refined directions as in Newton or quasi-Newton algorithms. Recent works \cite{49} showed that these stepsizes can be computed optimally in the case of nonsymmetric tensors without nonnegativity constraints.

For the sake of clarity the subscript \( it \) will be omitted in the rest of this section. Then the problem of global plane search is defined as follows:

**Problem 3.** Given two search directions \( G_A \) and \( G_C \) defined by an iterative algorithm minimizing the objective function (6), find the optimal stepsizes \( \mu_A \) and \( \mu_C \).

Solving problem 3 is performed by first inserting equation (7) in (6) and minimizing the following function:
\[
\Psi(\mu_A, \mu_C) = \| T^{(3)} - (C + \mu_C G_C)((A + \mu_A G_A)^{\boxtimes 2}) \|_F^2
\]
(8)
w.r.t. both \( \mu_A \) and \( \mu_C \). One can show that the latter can be rewritten as follows:
\[
\Psi(\mu_A, \mu_C) = \| F_0 + F_1 \mu_A + F_2 \mu_A^2 + F_3 \mu_A^3 + F_4 \mu_C + F_5 \mu_C \mu_A + F_6 \mu_C \mu_A^2 + \sum_{\alpha} F_{\alpha} \mu_C \mu_A^3 \|_F^2
\]
(9)

where:
\[
F_0 = T^{(3)} - CH_0 \\
F_1 = -CH_1 \\
F_2 = -CH_2 \\
F_3 = -CH_3 \\
F_4 = -CH_4 \\
F_5 = -G_C H_0 \\
F_6 = -G_C H_1 \\
F_7 = -G_C H_2 \\
F_8 = -G_C H_3 \\
F_9 = -G_C H_4
\]
(10)

with:
\[
H_0 = (K_0 \odot K_0)^T \\
H_1 = (K_0 \odot K_1 + K_1 \odot K_0)^T \\
H_2 = (K_0 \odot K_2 + K_1 \odot K_1 + K_2 \odot K_0)^T \\
H_3 = (K_1 \odot K_2 + K_2 \odot K_1)^T \\
H_4 = (K_2 \odot K_2)^T
\]
(11)

and finally:
\[
K_0 = A \boxtimes A \\
K_1 = A \boxtimes G_A + G_A \boxtimes A \\
K_2 = G_A \boxtimes G_A
\]
(12)

Note that equation (9) can be reduced to a compact form as follows:
\[
\Psi(\mu_A, \mu_C) = \| Fu \|_F^2 = u^T F^T F u = u^T Q u = \sum_{i,j} (Q \boxtimes uu^T)_{i,j}
\]
(13)

where \( F = [\mathrm{vec}(F_0), \mathrm{vec}(F_0^T), \mathrm{vec}(F_1), \mathrm{vec}(F_1^T), \mathrm{vec}(F_2), \mathrm{vec}(F_2^T), \mathrm{vec}(F_3), \mathrm{vec}(F_3^T), \mathrm{vec}(F_4), \mathrm{vec}(F_4^T)] \) is a \((I^2 K \times 10)\) matrix and \( u = [\mu_A \mu_A^2, \mu_C \mu_A^2, \mu_A^2 \mu_C, \mu_C, \mu_C \mu_A, \mu_C^2, \mu_A \mu_C^2, \mu_A \mu_C, \mu_A, \mu_C, 1]^T \) is a 10-dimensional vector and \( Q \) is a symmetric matrix. As it was expected after a simple glance at (8),
equation (13) shows that the objective function (13) is a second degree polynomial in \( \mu_C \). Thus, the main benefit of such a remark is that the optimal stepsize \( \mu_{opt}^C \) is the ratio of two coefficients, each being a polynomial in \( \mu_A \). Now, once \( \mu_{opt}^C \) is computed, its expression is injected in the equation \( \partial \Psi / \partial \mu_A = 0 \). Then, it can be shown that stationary points of (9) in \( \mu_A \) are among the roots of a 24-th degree polynomial in \( \mu_A \). Consequently, the global minimum \( \mu_{opt}^A \) can be obtained by rooting this polynomial and selecting the root yielding the smallest value of the objective. As in [22], it is noteworthy that the computation of the coefficients of the 24-th degree polynomial is computationally dominant, compared to rooting the polynomial itself.

An alternative way, avoiding such a computation, consists of considering \( \mu_C = \mu_A = \mu \). In that case, the objective (13) is a 10-th degree polynomial in \( \mu \) and the optimal stepsize \( \mu_{opt} \) is then computed by rooting its derivative. This alternative solution is more standard, but much less efficient (if we provisionally ignore the computational complexity), and will not be considered in the remainder.

As pointed out above, all considered algorithms in this paper are based on the first and the second derivatives of the considered cost function. Computing these derivatives in a compact matrix form can be attractive, rather than the element-by-element computation, especially from an implementation point of view when a matrix programming environment has to be used. Therefore, the following section describes a very practical way of computing these derivatives in compact matrix forms.

5. Toward a matrix derivation

This section is devoted to the computation of the gradient, the Jacobian and the Hessian of objective function (6) in a compact matrix form. Note that arranging in a compact matrix form the gradient of a scalar function in real-valued matrix variables was tackled before in [52, 53] to cite a few. An extension to the complex-valued matrix case was also introduced in [54, 55, 56]. The differential of \( \Psi \), seen as a function of two real-valued matrices \( A \) and \( C \), is given by:

\[
d\Psi(A, C) = D_A \Psi(A, C) d\text{vec}(A) + D_C \Psi(A, C) d\text{vec}(C)
\]

where \( D_A \Psi(A, C) = \partial \Psi / \partial \text{vec}(A) \) is defined as a vector representation of the matrix derivative \( D_A \Psi(A, C) = \partial \Psi / \partial A \). It is worth mentioning that the stationary points of \( \Psi \) can be found when the derivatives w.r.t. all independent variables vanish [53]. Thus, and from (14), it is sufficient to compute both \( D_A \Psi(A, C) \) and \( D_C \Psi(A, C) \) in order to find the extrema of \( \Psi \). Before starting, useful properties are recalled [52]:

\[
D_X Y = \Psi(A, C) = D_A \Psi(A, C) d\text{vec}(A) + D_C \Psi(A, C) d\text{vec}(C)
\]
Proof is given in appendix.

Lemma 1. Let \( Y = Z \odot Z \) (\( Z \in \mathbb{R}^{F \times P} \)), then the gradient of \( Y \) w.r.t. \( Z \) is given by:

\[
D_P Y = \partial \text{vec}(Y) / \partial \text{vec}(Z)^T = \text{diag}([\text{vec}(I_P \otimes Z) - (I_P \otimes I_P)](I_P \otimes I_P)(I_P \otimes I_P)) \text{vec}(Z) + \text{diag}([\text{vec}(Z) - (I_P \otimes I_P)](I_P \otimes I_P)(I_P \otimes I_P)) \text{vec}(Z)
\]

(28)

Proof is given in appendix.

Lemma 2. Let \( Z = A \odot A \) then the gradient of \( Z \) w.r.t. the matrix \( A \in \mathbb{R}^{F \times P} \) is given by \( D_A Z = \partial \text{vec}(Z) / \partial \text{vec}(A)^T = 2 \text{diag}(\text{vec}(A)) \).

Proof is given in appendix.

Lemma 3. Let \( u(B) = \text{diag}([\text{vec}(B)]) \omega \) be a vector function in the matrix \( B \) of size \((1 \times P)\) where \( \omega \) is an \( IP \)-dimensional vector; the gradient of \( u(B) \) w.r.t. \( B \) is then given by \( D_B u(B) = \text{diag}(\omega) \).

Proof is given in appendix.
5.1. Gradient computation

A compact matrix computation of both gradients \( D_A \Psi(A, C) \) and \( D_C \Psi(A, C) \) is given hereafter. From equation (6) we have:

\[
\Psi(A, C) = \text{Tr}(T^{(3)'}T^{(3)}) - 2f_{f^0}(Z, C) + g(Z, C)
\]

(29)

where \( f_{f^0}(A, C) = \text{Tr}(T^{(3)'}CY') \) and \( g(A, C) = \text{Tr}(YC'Y') \) with again \( Z = A \otimes A \) and \( Y = Z \otimes Z \). Then we get:

\[
d\Psi(A, C) = -2df_{f^0}(A, C) + dg(A, C)
\]

\[
= (-2D_f f_{f^0}(A, C) + D_y g(A, C))dvec(Y) +
\]

\[
(-2D_C f_{f^0}(A, C) + D_C g(A, C))dvec(C)
\]

(30)

Note that the term \( D_A \Psi(A, C) = D_y f_{f^0}(A, C)D_2 YD_A Z \) expresses the well-known chain rule [54].

**Lemma 4.** The gradient of the objective function \( \Psi \) (6) w.r.t. the matrix \( A \in \mathbb{R}^{l \times p} \) is given by:

\[
D_A \Psi(A, C) = -2D_y f_{f^0}(A, C)D_2 YD_A Z + D_y g(A, C)D_2 YD_A Z
\]

\[
D_A \Psi(A, C) = -2D_A f_{f^0}(A, C) + D_A g(A, C)
\]

(31)

where:

\[
D_A f_{f^0}(A, C) = 2(\text{vec}(C'T^{(3)})'U_{p,p} \text{diag}(\text{vec}(I_1 \otimes Z))(I_p \otimes I_1) + \text{vec}(C'T^{(3)})'U_{p,p} \times \text{diag}(\text{vec}(Z) \otimes I_1)(U_{p,p} \otimes I_1)(I_p \otimes I_{p,p})) \text{diag}(\text{vec}(A))
\]

(32)

and:

\[
D_A g(A, C) = 4\text{vec}(Y)'((\xi \otimes I_p) \text{diag}((\text{vec}(I_1 \otimes A)))(I_p \otimes I_1) + \text{diag}((\text{vec}(Z) \otimes I_1)) \times (U_{p,p} \otimes I_1)(I_f \otimes I_{p,p})) \text{diag}(\text{vec}(A))
\]

(33)

with \( Y = Z \otimes Z \) and \( \xi = C'C \).

Proof is given in appendix. The computation of \( D_C \Psi(A, C) \) follows the same strategy used to compute \( D_A \Psi(A, C) \).

**Lemma 5.** The gradient of \( \Psi \) (6) w.r.t. the matrix \( C \in \mathbb{R}^{k \times p} \) is given by:

\[
D_C \Psi(A, C) = -2D_C f_{f^0}(A, C) + D_C g(A, C) = -2\text{vec}(T^{(3)}Y)' + 2\text{vec}(C)'(YY' \otimes I_k)
\]

(34)

where \( Y = Z \otimes Z \).

Proof is given in appendix.
5.2. Computation of Jacobian

By stringing out $T^{(3)} - C(Z \otimes Z)^\top$ in a $P^2K$-th dimensional vector, the cost function can be then expressed as:

$$\Psi(A, C) = \|T^{(3)} - C(Z \otimes Z)^\top\|_F^2 = \|\text{vec}(T^{(3)} - C(Z \otimes Z)^\top)\|_F^2 = \|f(A, C)\|_F^2$$  \hfill (35)

where $f : \mathbb{R}^{I \times P} \times \mathbb{R}^{K \times P} \rightarrow \mathbb{R}^{P^2K}$ and $f(A, C) = \text{vec} \left( T^{(3)} - C(Z \otimes Z)^\top \right)$. The Jacobians of the vector function $f$ w.r.t. matrices $A$ and $C$ are respectively of size $\mathbb{R}^{P \times I}$ and $\mathbb{R}^{P \times K}$ and given by:

$$J_A = \frac{\partial f(A, C)}{\partial \text{vec}(A)} \text{ and } J_C = \frac{\partial f(A, C)}{\partial \text{vec}(C)^\top}$$  \hfill (36)

These two previous expressions can be easily found by computing the differential of $f$ (35) w.r.t. its variables $A$ and $C$.

**Lemma 6.** Jacobians of the vector function $f$ (35) w.r.t. to its matrix variables $A$ and $C$ are given by:

$$J_C = -(Z \otimes Z) \otimes I_K$$  \hfill (37)

$$J_A = -2(I_F \otimes C)U_{P \times I} \left\{ \text{vec}(Z) \otimes I_K \left( U_{IP} \otimes I_I \right) \left( I_I \otimes I_P \right) + \text{diag}(\text{vec}(I_I \otimes Z)) \left( I_P \otimes I_N \right) \text{diag}(\text{vec}(A)) \right\}$$  \hfill (38)

where $Z = A \, \boxdot \, A$.

Proof is given in appendix.

5.3. Hessian computation

The computation of the Hessian w.r.t. to both variables $A$ and $C$ is also given in a compact matrix form following the same technique used for the gradient computation, namely differentials. Since the Hessian of the objective function w.r.t. a given entry is nothing else but the derivative of the gradient of this objective function in that variable, deducing the Hessian expression is immediate using the differential of the previously computed gradient. Then we can write [55]:

$$d(D_A \Psi(A, C))^\top = D_A(D_A \Psi(A, C))^\top \text{dvec}(A) + D_C(D_A \Psi(A, C))^\top \text{dvec}(C)$$

$$= D_A(-2D_A f_{Dn}(A, C) + D_A g(A, C))^\top \text{dvec}(A) +$$

$$D_C(-2D_C f_{Dn}(A, C) + D_A g(A, C))^\top \text{dvec}(C)$$

$$= (-2H_{AA}f_{Dn}(A, C) + H_{AA}g(A, C))\text{dvec}(A) +$$

$$(-2H_{CA}f_{Dn}(A, C) + H_{CA}g(A, C))\text{dvec}(C)$$

$$= H_{AA}(A, C)\text{dvec}(A) + H_{CA}(A, C)\text{dvec}(C)$$  \hfill (39)

where $H_{AA}(A, C)$ denotes the Hessian of the scalar function $\Psi$ w.r.t. $A$. Likewise we deduce the Hessian of $\Psi$ (6) w.r.t. $C$, named $H_{CC}(A, C)$:

$$d(D_C \Psi(A, C))^\top = D_A(D_C \Psi(A, C))^\top \text{dvec}(A) + D_C(D_C \Psi(A, C))^\top \text{dvec}(C)$$

$$= H_{CA}(A, C)\text{dvec}(A) + H_{CC}(A, C)\text{dvec}(C)$$  \hfill (40)
In order to compute $H_{A,\Psi}(A, C)$, we should, according to equation (39), compute both $H_{A,f_{T^0}}$ and $H_{A,g_{T^0}}$. According to lemma 4 (i.e. equation (32)) we have:

\[(D_A f_{T^0}(A, C))^T = D_A^{(1)} f_{T^0}(A, C) + D_A^{(2)} f_{T^0}(A, C)\]  

(41)

with:

\[D_A^{(1)} f_{T^0}(A, C) = 2 \text{diag}(\text{vec}(A))(I_P \otimes 1_1^r)\text{diag}(\text{vec}(1_I \otimes Z))\text{vec}(T^{(3)} C)\]  

(42)

\[D_A^{(2)} f_{T^0}(A, C) = 2 \text{diag}(\text{vec}(A))(1_I^T \otimes I_P)U_{IP} \text{diag}(\text{vec}(Z \otimes 1_I)) \times \text{vec}(T^{(3)} C)\]  

(43)

Since the differential is a linear map, the Hessian of $f_{T^0}(A, C)$ w.r.t. $A$ can be then written as:

\[H_{A,A} f_{T^0}(A, C) = H_{A,A}^{(1)} f_{T^0}(A, C) + H_{A,A}^{(2)} f_{T^0}(A, C)\]  

(44)

where $H_{A,A} f_{T^0}(A, C) = D_A(D_A^{(0)} f_{T^0}(A, C))$ for $i \in \{1, 2\}$.

**Lemma 7.** The Hessians $H_{A,A}^{(1)} f_{T^0}(A, C)$ and $H_{A,A}^{(2)} f_{T^0}(A, C)$ in equation (44) are equal to:

\[H_{A,A}^{(1)} f_{T^0}(A, C) = 2 \text{diag}((I_P \otimes 1_1^r)\text{diag}(\text{vec}(1_I \otimes Z))\text{vec}(T^{(3)} C)) + 4 \text{diag}(\text{vec}(A))(I_P \otimes 1_I^r)\text{diag}(\text{vec}(A))\]  

(45)

\[H_{A,A}^{(2)} f_{T^0}(A, C) = 2 \text{diag}((1_I^T \otimes I_P)U_{IP} \otimes 1_I)\text{diag}(\text{vec}(Z \otimes 1_I))\text{vec}(Y^{(3)} C)) + 4 \text{diag}(\text{vec}(Z))(1_I^T \otimes I_P)U_{IP} \otimes 1_I)\text{diag}(\text{vec}(Z))\]  

(46)

Proof is given in appendix. Now, to compute $H_{A,A} g_{T^0}(A, C)$ we follow the same technique used to compute $H_{A,A} f_{T^0}(A, C)$. Indeed, according to Lemma 4, i.e. equation (33), we have:

\[(D_A g_{T^0}(A, C))^T = D_A^{(1)} g_{T^0}(A, C) + D_A^{(2)} g_{T^0}(A, C)\]  

(47)

with:

\[D_A^{(1)} g_{T^0}(A, C) = 4 \text{diag}(\text{vec}(A))(I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes Z))\text{vec}(Y)\]  

(48)

Using the linearity property of the differential, we have:

\[H_{A,A} g_{T^0}(A, C) = H_{A,A}^{(1)} g_{T^0}(A, C) + H_{A,A}^{(2)} g_{T^0}(A, C)\]  

(49)

**Lemma 8.** The Hessians $H_{A,A}^{(1)} g_{T^0}(A, C)$ and $H_{A,A}^{(2)} g_{T^0}(A, C)$ in equation (49) are equal to:

\[H_{A,A}^{(1)} g_{T^0}(A, C) = 4 \text{diag}((I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes A))\text{vec}(Y) + 8 \text{diag}(\text{vec}(A))(I_P \otimes 1_I^r)\text{diag}(\text{vec}(Y))U_{IP} \otimes 1_I)\text{vec}(A) + 8 \text{diag}(\text{vec}(A))(I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes A))\times \text{vec}(A)\times (I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes A))\times \text{vec}(Y) + 8 \text{diag}(\text{vec}(A))(I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes A))\times \text{vec}(A)\times (I_P \otimes 1_I^r)\text{diag}(\text{vec}(1_I \otimes A))\times \text{vec}(Y)\]  

(50)
and:

\[
H^{(2)}_{A,A}(A, C) = 4\text{diag}[(I_p \otimes I_p)(U_{IP} \otimes I_p)\text{diag}\{\text{vec}(Z) \otimes I_1\}(\xi \otimes I_p)\text{vec}(Y)] + 8\text{diag}(\text{vec}(A))(I_p \otimes I_p)(U_{IP} \otimes I_p)\text{diag}[(\xi \otimes I_1)\text{vec}(Y)] \\
(\xi \otimes I_p)\text{diag}(\text{vec}(1 \otimes Z))(I_{IP} \otimes I_1) + \text{diag}(\text{vec}(Z) \otimes I_1)(U_{IP} \otimes I_1) \\
(\xi \otimes I_p)\text{diag}(\text{vec}(A))
\]

(51)

Proof is given in appendix.

**Lemma 9.** The Hessian of $\Psi$ (6) w.r.t. the matrix $C \in \mathbb{R}^{K \times P}$ is given by:

\[
H_{CC} \Psi(A, C) = -2(Y^T Y \otimes I_K)
\]

(52)

where $Y = Z \otimes Z$ with $Z = A \boxtimes A$.

Proof is given in appendix.

6. Optimization methods based on first and second order derivatives

This section is devoted to the brief description of some derivative-based optimization algorithms used to solve problem 2. These algorithms are namely, the GD, LM and Newton algorithms.

6.1. Gradient descent method

The GD algorithm is well-known, and probably the simplest optimization algorithm. Beside its simplicity, it is cheap in terms of numerical complexity. The pseudo code of the GD_{1step} and GD_{2step} algorithms, equipped with one and two globally optimal stepsizes respectively, is described in Algorithm 1.

**Algorithm 1** Pseudo-code for the GD algorithm

Initialize both matrices $A$ and $C$.

Set $it = 1$ and set the maximal number $it_{max}$ of allowed iterations and the stop criterion threshold $\epsilon$ to a predefined value, respectively.

while $|\Psi(A^{it}, C^{it}) - \Psi(A^{it-1}, C^{it-1})|/|\Psi(A^{it-1}, C^{it-1})| > \epsilon$ or $it < it_{max}$ do

Compute the gradients $D_A \Psi(A^{it}, C^{it})$ and $D_C \Psi(A^{it}, C^{it})$ using lemmas 4 and 5.

Compute the stepsizes $\mu^A_{it}$ and $\mu^C_{it}$ using the global plane search scheme.

Update both matrices $A$ and $C$ using the following equations:

\[
A^{it+1} = A^{it} - \mu^A_{it} \nabla_A \Psi(A^{it}, C^{it})
\]
\[
C^{it+1} = C^{it} - \mu^C_{it} \nabla_C \Psi(A^{it}, C^{it})
\]

(53) (54)

where $\nabla_M \Psi(A, C) = \text{unvec}(D_M \Psi(A, C))$ with $M \in \{A, C\}$.

Set $it = it + 1$.

end while
6.2. Levenberg-Marquardt method

The LM algorithm is based on a first order linear approximation of the components of \( f \) in the neighborhood of \((A, C)\). Then the LM update rule is given by [57]:

\[
\text{vec}(M^{(i+1)}) = \text{vec}(M^{(i)}) - \left( J MF(A^{(i)}, C^{(i)}) J MF(A^{(i)}, C^{(i)}) + \nu I \right)^{-1} D M \Psi(A^{(i)}, C^{(i)})^T \tag{55}
\]

where \( J MF(A, C) \) is the Jacobian of \( f \) w.r.t. \( M \,(M \in \{A, C\}) \) and \( \nu \) is the damped parameter adjusting the well-conditioning of the approximated Hessian in the trust region [57]. We note from the previous equation that the Hessian of the considered cost function is approximated by the product of the transpose of the Jacobian \( f \) by itself. This approximation has two main benefits: first, if the Jacobian is full rank, then the positive definiteness property of the Hessian is already verified. Second, a full computation of the Hessian is not mandatory any more. Furthermore, the validity of such an approximation and consequently its non-singularity is always monitored using the damped parameter \( \nu \) which is computed at each iteration using the principle of the trust region. For more details on this technique one can refer to [57] and the references therein. The pseudo-code given in Algorithm 2 describes the main steps of the LM\(_{SYM}\) algorithm.

**Algorithm 2** Pseudo-code for the LM\(_{SYM}\) algorithm

<table>
<thead>
<tr>
<th>Initialize both matrices ( A ) and ( C ).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set ( it = 1 ) and set the maximal number ( it_{\text{max}} ) of allowed iterations and the stop criterion threshold ( \epsilon ) to a predefined value, respectively;</td>
</tr>
<tr>
<td><strong>while</strong> ( \Psi(A^{(i)}, C^{(i)}) - \Psi(A^{(i-1)}, C^{(i-1)}) |/| \Psi(A^{(i-1)}, C^{(i-1)}) &gt; \epsilon ) or ( it &lt; it_{\text{max}} ) <strong>do</strong></td>
</tr>
<tr>
<td>Compute the gradients ( D_A \Psi(A^{(i)}, C^{(i)}) ) and ( D_C \Psi(A^{(i)}, C^{(i)}) ) according to lemmas 4 and 5.</td>
</tr>
<tr>
<td>Compute the Jacobians ( J_A \phi(A^{(i)}, C^{(i)}) ) and ( J_C \phi(A^{(i)}, C^{(i)}) ).</td>
</tr>
<tr>
<td>Compute the damped parameter ( \nu ) based on the trust region technique [57].</td>
</tr>
<tr>
<td>Update both matrices ( A ) and ( C ) using the following equations:</td>
</tr>
</tbody>
</table>
| \[
\text{vec}(A^{(i+1)}) = \text{vec}(A^{(i)}) - (J_A f(A^{(i)}, C^{(i)}) J_A f(A^{(i)}, C^{(i)}) + \nu I)^{-1} D_A \Psi(A^{(i)}, C^{(i)})^T \tag{56}
\]
| \[
\text{vec}(C^{(i+1)}) = \text{vec}(C^{(i)}) - (J_C f(A^{(i)}, C^{(i)}) J_C f(A^{(i)}, C^{(i)}) + \nu I)^{-1} D_C \Psi(A^{(i)}, C^{(i)})^T \tag{57}
\]
| **end while** |

Set \( it = it + 1 \).

6.3. 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]. This approach requires the objective function \( \Psi \) (6) to be twice differentiable in its variables \( A \) and \( C \). We shall consider a simplified version of the Newton method, where the cross derivatives \( \partial \Psi / \partial A_a \partial C_b \) are neglected. This will be compensated by the fact that two distinct stepsizes will be computed for variables \( bA \) and \( C \). The update rule is then defined by:

\[
\text{vec}(M^{(i+1)}) = \text{vec}(M^{(i)}) - \mu_A^{(i)} H_{M,M} \Psi(A^{(i)}, C^{(i)})^{-1} D_M \Psi(A^{(i)}, C^{(i)})^T \tag{58}
\]

where \( M \in \{A, C\}, \mu_A \) and \( \mu_C \) are the stepsizes computed using the global plane search procedure. It is noteworthy that the Hessian must be positive definite to ensure convergence. Newton
method is then suitable when matrices are close to the solution. Indeed in this case the positive definiteness property of the Hessian is verified. However, if this property is missed or if the Hessian matrix is close to be singular, the algorithm may follow an ascent direction or may be excessively slow. A way that we propose to ensure positive definiteness and invertibility consists in modifying a bit the Hessian matrix. This slight modification should be performed in such a way the second order information in the Hessian is still preserved. This strategy is based on replacing all negative (or zero) eigenvalues of the Hessian matrix by one. Then the modified Hessian, named $\tilde{H}_{M,M}(A, C)$, is written by:

$$\tilde{H}_{M,M}(A, C) = H_{M,M}^{(1)}(A, C) + H_{M,M}^{(2)}(A, C)$$  \hspace{1cm} (59)$$

where $M \in \{A, C\}$, $H_{M,M}^{(1)}(A, C)$ stands for the sub-Hessian block matrix associated with the strictly positive eigenvalues and $H_{M,M}^{(2)}(A, C) = I$ is the sub-Hessian block matrix associated with the negative eigenvalues, which were set to one. Now replacing the previous equation in the Newton’s update rule (58), we state that this modified version of the Newton algorithm, named “mNewton”, is a combination between both the classical Newton method and the GD one. Indeed, when the number of strictly positive eigenvalues in the Hessian matrix is considerably bigger than the negative ones, the mNewton method behaves more as the classical Newton algorithm. Conversely, when eigenvalues of the Hessian are small, the roles of these two sub-matrices are swapped and the algorithm works more as a classical GD algorithm. The pseudo-code given in Algorithm 3 describes the main steps of the mNewton method, which uses two globally optimal stepsizes.

Algorithm 3 Pseudo-code for the mNewton algorithm

1. Initialize both matrices $A$ and $C$.
2. Set $it = 1$ and set the maximal number $itmax$ of allowed iterations and the stop criterion threshold $\epsilon$ to a predefined value, respectively.
3. while $\|\Psi(A^it, C^it) - \psi(A^{it-1}, C^{it-1})\|/\|\Psi(A^{it-1}, C^{it-1})\| > \epsilon$ or $it < itmax$ do
4. \hspace{0.5cm} Compute the gradients $D_A\Psi(A^it, C^it), D_C\Psi(A^it, C^it)$ and the Hessians $H_{A,A}\Psi(A^it, C^it)$ and $H_{C,C}\Psi(A^it, C^it)$ using lemmas 4, 5 and equation (39), respectively.
5. \hspace{0.5cm} Compute the modified Hessian matrices $\tilde{H}_{A,A}\Psi(A^it, C^it)$ and $\tilde{H}_{C,C}\Psi(A^it, C^it)$ as described in equation (59).
6. \hspace{0.5cm} Compute the stepsizes $\mu_A^it$ and $\mu_C^it$ using the global plane search scheme.
7. \hspace{0.5cm} Update both matrices $A$ and $C$ using the following equations:
8. \hspace{1.5cm} $vec(A^{it+1}) = vec(A^it) - \mu_A^it \tilde{H}_{A,A}\Psi(A^it, C^it)^{-1}D_A\Psi(A^it, C^it)$ \hspace{1cm} (60)
9. \hspace{1.5cm} $vec(C^{it+1}) = vec(C^it) - \mu_C^it \tilde{H}_{C,C}\Psi(A^it, C^it)^{-1}D_C\Psi(A^it, C^it)$ \hspace{1cm} (61)
10. \hspace{0.5cm} Set $it = it + 1$.

end while

7. Computer simulation

This section is devoted to study how much is appealing the use of our global line search procedure combined with prior informations in the considered tensors such as both semi-nonnegativity
and semi-symmetry. This analysis is performed in terms of convergence speed and accuracy of factor estimation for different Signal-to-Noise Ratios (SNR). Thus, the behavior of our four approaches, namely GD2step, GD1step, mNewton and LM\textsubscript{SYM}, where both semi-nonnegativity and semi-symmetry are taken into account is compared with that of the classical LM algorithm where neither symmetry nor nonnegativity constraints are used [1]. Therefore two real matrices $C \in \mathbb{R}^{10 \times 2}$ and $A \in \mathbb{R}^{8 \times 2}$ are randomly generated to build, according to equation (5), the third order array $T(Z, Z, C) \in \mathbb{R}^{8 \times 8 \times 10}$ where $Z = A \boxtimes A$. Then $T(Z, Z, C)$ is perturbed by a third-order array of noise, named $N$, dropped from a zero-mean normal distribution with unit standard deviation. The resulting noisy third-order array $Y$ has the following form:

$$Y = \frac{T}{\|T\|_F} + \sigma_N \frac{N}{\|N\|_F}$$  \hspace{1cm} (62)

where $\sigma_N$ is a scalar controlling the noise level. All considered algorithms in this study stop either when the relative error between the real and the estimated tensor, as defined in the above pseudo-codes, exhibits a value less than a predefined threshold of $10^{-12}$ or when the number of iterations exceeds 2000 iterations. Furthermore, we note that all results in this section are averaged over 50 Monte Carlo realizations. In addition, all methods are initialized in the same manner. That’s to say, for matrix $Z$ a random guess dropped from a zero-mean normal distribution with unit standard deviation is used while matrix $C$ is initialized by the most significant $P$ left singular vectors spanning the left subspace of $T^{(3)}$. As a performance criterion, we propose to measure the error between matrix $A$ and its estimate $A_{est}$. The performance criterion must be invariant to scale and permutation indeterminacies inherent to problem 1. Consequently, we choose the measure defined in[1] and given by:

$$D(A, A_{est}) = \min_{\Pi} \Psi(A, A_{est}, \Pi)$$  \hspace{1cm} (63)

where matrix $\Pi$ belongs to the set of permutations, and where:

$$\Psi(M, M_{est}) = \sum_{p=1}^{P} \|m_p - \frac{m_{p,est}'}{m_{p,est}'}m_{p,est}\|$$  \hspace{1cm} (64)

with $m_p$ and $m_{p,est}$ the $p$-th columns of $M$ and $M_{est}$, respectively.

As pointed out above, the first study consists in evaluating the behavior of the considered methods as a function of SNR. Figure 1 shows the median of the pseudo-distance between the real loading matrices and their estimates as a function of the SNR ranging for $\text{−}15$ dB to 30 dB. For a first glance on this figure we note, especially for low SNR value, how much is the gain that we obtain when exploiting both the semi-nonnegativity and the semi-symmetry constraints when solving problem 1. This derives straightforwardly from the comparison of the performance between LM\textsubscript{SYM} and mNewton and LM\textsubscript{SYM}. Next, we note among our proposed approaches that both the algorithms LM\textsubscript{SYM} and mNewton outperform the GD ones especially for weak SNR value.

Regarding the second study, it consists in evaluating the convergence speed of the considered algorithms for different SNR values (i.e. $\text{−}15, 5$ and 30 dB). Figures 2, 3 and 4 show also the median of the error between the real and the estimated loading matrices $A$ and $C$ as a function of the number of iterations for SNR values of $\text{−}15$ and 30 dB, respectively. We note from figure 2 the benefit we get when using both the semi-nonnegativity and the semi-symmetry constraints.
Figure 1: The median of the distance between the real loading matrices $A$ and $C$ and their estimates $A_{est}$ and $C_{est}$, respectively, as a function of SNR at the output of the classical LM algorithm compared to our proposed approaches, namely ii) GD$_{2step}$ (the GD algorithm with two optimal learning steps), GD$_{1step}$ (with $\mu_A = \mu_C$ optimally computed), iii) mNewton (the mNewton algorithm with two optimal learning steps $\mu_A$ and $\mu_C$ optimally computed) and iv) the LM$_{SYM}$ algorithm.

for low SNR values. Indeed although the classical LM algorithm goes faster during the optimization process, it is stuck in a local minimum while our proposed approaches keep going towards the global optimum, hence a better convergence as depicted in both figures 2(a) and 2(b). We also note from the latter figures how much the mNewton algorithm outperforms all the other considered algorithm especially the GD algorithms. Furthermore, things are going better for higher SNR values. In fact, a higher decomposition accuracy is obtained for SNR of 5 dB than the one when SNR is equal to $-15$ dB. Figure 3 confirms the remark stated in figure 2: the use of prior informations on the data permits the related algorithms combined with our global line search procedure to keep on its convergence towards the global optimum while the classical LM algorithm suffers from the local minimum encountered in its search direction. We note also a comparable performance of both the LM$_{SYM}$ and the GD$_{2step}$ for the estimation of matrix $C$ (see figure 3(b)) but with an obvious superiority of the LM$_{SYM}$ in estimating the loading matrix $A$ as depicted in figure 3(a). Furthermore, we state that even the GD$_{1step}$ with our global line search scheme is able to outperform the classical LM algorithm beyond 1800 iterations which confirms the interest behind exploiting the prior informations of the data.

Regarding the case when SNR is equal to 30 dB we note considerably a faster convergence rate for the classical LM algorithm. However, as previously mentioned, it suffers from lack of convergence whereas the use of the semi-symmetry and the semi-nonnegativity constraints makes better the convergence of its related algorithms. In result, we state that in more difficult situations (case of low SNR), our global lines search procedure combined with all prior informations about the tensor at hand is a good way to circumvent all convergence related problems such as local minimum, even if the convergence rate of those algorithms seems to be a bit slower.
Figure 2: The median of the distance between the real loading matrices $A$ and $C$ and their estimates $A_{est}$ and $C_{est}$, respectively, as a function the number of iterations at the output of the classical LM algorithm compared to our proposed approaches, namely ii) GD$_{2step}$ (the GD algorithm with two optimal learning steps), GD$_{1step}$ (with $\mu_A = \mu_C$ optimally computed), iii) mNewton (the mNewton algorithm with two optimal learning steps $\mu_A$ and $\mu_C$ optimally computed) and iv ) the LM$_{SYM}$ algorithm for SNR of $-15$ dB.

Figure 3: The median of the distance between the real loading matrices $A$ and $C$ and their estimates $A_{est}$ and $C_{est}$, respectively, as a function the number of iterations at the output of the classical LM algorithm compared to our proposed approaches, namely ii) GD$_{2step}$ (the GD algorithm with two optimal learning steps), GD$_{1step}$ (with $\mu_A = \mu_C$ optimally computed), iii) mNewton (the mNewton algorithm with two optimal learning steps $\mu_A$ and $\mu_C$ optimally computed) and iv ) the LM$_{SYM}$ algorithm for SNR of $5$ dB.
8. Conclusion

An original global plane search scheme was presented in this paper, and applies specifically in the case of semi-symmetric CP decompositions. Nonnegativity has been also imposed in the two symmetric modes, yielding a semi-nonnegative INDSCAL decomposition. But our global plane search scheme would also apply without this nonnegativity constraint. The nonnegativity constraint was parametrized, leading to an unconstrained optimization. In addition, we showed how compact matrix expressions of derivatives (i.e., gradient, Hessian and Jacobian) can be obtained in this rather complicated case. All the iterative approaches we proposed were analyzed and compared to the classical LM algorithm, wherein neither symmetry nor nonnegativity constraints are taken into account. This analysis was performed in terms of convergence speed, overfactoring and collinearity between factors for several SNR values. All reported numerical results showed the good behavior of the proposed approaches.
Appendix A. Proof of lemma 1

Assume at this moment that matrix $Y$ is a real function in two real-valued matrices $Z_1 \in \mathbb{R}^{I_1 \times P}$ and $Z_2 \in \mathbb{R}^{I_2 \times P}$ with $Z_1 = Z_2 = Z$. Then according to equations (18) and (19), we have:

\[
\text{vec}(Y) = \text{vec}(Z \otimes Z) = \text{diag}(\text{vec}(1_{I_1} \otimes Z_2))(1_{I_1} \otimes 1_{I_2})\text{vec}(Z_1)
\]

(A.1)

\[
= \text{diag}((\text{vec}(Z_1) \otimes 1_{I_2}))((U_{I_1} \otimes 1_{I_2})(1_{I_1} \otimes 1_{I_2})\text{vec}(Z_2))
\]

(A.2)

Then the differential of the vectorized form of $Y$ is given by:

\[
d\text{vec}(Y) = D_{Z_1}Yd\text{vec}(Z_1) + D_{Z_2}Yd\text{vec}(Z_2)
\]

(A.3)

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

\[
d\text{vec}(Y) = \text{diag}(\text{vec}(1_{I_1} \otimes Z_2))(1_{I_1} \otimes 1_{I_2})d\text{vec}(Z_1)
\]

(A.4)

Now, by including the constraint $Z_1 = Z_2 = Z$ in the latter differential form and factorizing it w.r.t. $d\text{vec}(Z)$, the gradient expression $D_Z Y$ given in the Lemma is proved.

Appendix B. Proof of lemma 2

As proceeded previously, we suppose at this moment that matrix $Z$ is a real function in two real-valued matrices $A_1$ and $A_2$ subject to $A_1 = A_2$. Then we have:

\[
\text{vec}(Z) = \text{vec}(A_1 \boxtimes A_2) = \text{diag}(\text{vec}(A_2))\text{vec}(A_1)
\]

(B.1)

\[
= \text{diag}(\text{vec}(A_1))\text{vec}(A_2)
\]

(B.2)

Then according to equations (B.1) and (16), the differential of $Z$ is given by:

\[
d\text{vec}(Z) = D_{A_1}d\text{vec}(A_1) + D_{A_2}d\text{vec}(A_2)
\]

(B.3)

Now, by including the constraint $A_1 = A_2 = A$ in the latter differential form and factorizing it w.r.t. $d\text{vec}(A)$, the gradient expression $D_A Z$ given in Lemma 2 is proved.

Appendix C. Proof of lemma 3

Proof. Using some properties of a vector multiplication by a diagonal matrix, we can write:

\[
u(B) = \text{diag}(\text{vec}(B))\omega = \text{diag}(\omega)\text{vec}(B)
\]

(C.1)

Then based on the differential of $\nu$ w.r.t. $B$, we get:

\[
d\nu(B) = \text{diag}(\omega)d\text{vec}(B) = D_B\nu(B)d\text{vec}(B)
\]

(C.2)

Hence the result.
Appendix D. Proof of lemma 4

According to the differential expression (14) and from equation (30), equation (31) is immediate. Now, we have

$$f_{\text{prev}}(A, C) = \text{Tr}(T^{(3)}'C(Z \otimes Z)') = \text{Tr}(T^{(3)}'CY')$$

$$= \text{vec}(C'T^{(3)})'\text{vec}(Y') = \text{vec}(C'T^{(3)})'U_{\theta ; \theta}^\top \text{vec}(Y) \quad (D.1)$$

Then:

$$d f_{\text{prev}}(A, C) = \text{vec}(C'T^{(3)})'\text{vec}(Y') = \text{vec}(C'T^{(3)})'U_{\theta ; \theta}^\top \text{vec}(Y)$$

$$= D_y f_{\text{prev}}(A, C)\text{dvec}(Y) = D_y f(A, C)D_Z Y d\text{vec}(Z)$$

$$= D_y f_{\text{prev}}(A, C)D_Z Y D_A \text{dvec}(A)$$

$$= D_A f_{\text{prev}}(A, C)\text{dvec}(A) \quad (D.2)$$

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

$$D_A f_{\text{prev}}(A, C) = 2\text{vec}(C'T^{(3)})'U_{\theta ; \theta}^\top (\text{diag}(\text{vec}(1_f \otimes Z)) \otimes 1_f) + \text{diag}(\text{vec}(Z) \otimes 1_f) \times$$

$$(U_{\theta ; \theta} \otimes 1_f)(1_f \otimes 1_f)\text{diag}(\text{vec}(A)) \quad (D.3)$$

Let’s now compute the gradient of the real function $g(A, C)$ w.r.t. $A$.

$$g(Z, C) = \text{Tr}((Z \otimes Z)C'(Z \otimes Z)') = \text{Tr}(Y\xi Y')$$

$$= \text{vec}(\xi' Y')'\text{vec}(Y') = (U_{\theta ; \theta}^\top \text{vec}(Y\xi))'U_{\theta ; \theta}^\top \text{vec}(Y)$$

$$= (U_{\theta ; \theta}^\top (\xi' \otimes 1_f)\text{vec}(Y))'U_{\theta ; \theta}^\top \text{vec}(Y)$$

$$= \text{vec}(Y)'(\xi' \otimes 1_f)\text{vec}(Y) \quad (D.4)$$

where $U_{\theta ; \theta}^\top U_{\theta ; \theta} = 1_f$. Then:

$$dg(A, C) = \text{vec}(Y)'(\xi' \otimes 1_f)\text{dvec}(Y)$$

$$= D_y g(A, C)\text{dvec}(Y) = D_y g(Z, C)D_Z Y D_A Z \text{dvec}(A)$$

$$= D_A g(A, C)\text{dvec}(A) \quad (D.5)$$

with $Z = A \bigtriangleup A$. Now using the results of both lemma 1 and lemma 2 we obtain:

$$D_A g(A, C) = 4\text{vec}(Y)'(\xi' \otimes 1_f)(\text{diag}(\text{vec}(1_f \otimes A)) \otimes 1_f) + \text{diag}(\text{vec}(A) \otimes 1_f) \times$$

$$(U_{\theta ; \theta} \otimes 1_f)(1_f \otimes 1_f)\text{diag}(\text{vec}(A)) \quad (D.6)$$

Then by multiplying equation (D.3) by a factor of two and summing it to equation (D.6), we obtain the gradient expression $D_A \Psi(A, C)$.

Appendix E. Proof of lemma 5

According to the differential expression (14) and from equation (30), equation (31) is immediate. Now, we have:

$$f_{\text{prev}}(A, C) = \text{Tr}(T^{(3)}'CY') = \text{Tr}(Y'T^{(3)}C) = \text{vec}(T^{(3)}Y') \text{vec}(C) \quad (E.1)$$
Then since we are interested in computing the gradient w.r.t. to $A$, we will limit ourself to compute the differential of $f_{F\Phi}(A, C)$ w.r.t. to $A$:
\[
d f_{F\Phi}(A, C) = \text{vec}(T^{(3)}Y^r)\text{dvec}(C) = D_c f_{F\Phi}(A, C)\text{dvec}(C)
\] (E.2)

Let's now compute the gradient of the real function $g(A, C)$ w.r.t. $C$.
\[
g(A, C) = \text{Tr}(YC'Y') = \text{Tr}(CY'YC')
\]
\[
= \text{vec}(Y'YC')^\top\text{vec}(C')
\]
\[
= (U_{KP}(Y'Y \otimes I_k)\text{vec}(C))^\top U_{KP}\text{vec}(C)
\]
\[
= \text{vec}(C)^\top(Y'Y \otimes I_k)\text{vec}(C)
\] (E.3)

Then since we are interested in computing the gradient w.r.t. to $C$, we will limit ourself to compute the differential of $g(A, C)$ w.r.t. to $C$:
\[
d g(A, C) = 2\text{vec}(C)^\top(Y'Y \otimes I_k)\text{vec}(C) = D_c g(A, C)\text{dvec}(C)
\] (E.4)

Then premultiplying the term $D_c f_{F\Phi}(A, C)$ by a factor of two and summing it to $D_c g(A, C)$, we obtain the gradient expression $D_c \Psi(A, C)$.

**Appendix F. Proof of lemma 6**

Based on the differential of a scalar function w.r.t. its variables (14), the differential of the vector function $f$ (35) is given by:
\[
\text{dvec}(f(A, C)) = df(A, C) = \frac{\partial f(A, C)}{\partial \text{vec}(A)}\text{dvec}(A) + J_c = \frac{\partial f(A, C)}{\partial \text{vec}(C)}\text{dvec}(C)
\]
\[
= f_A(A, C)\text{dvec}(A) + j_c f(A, C)\text{dvec}(C)
\] (F.1)

By comparing both equations (14) and (F.1), we note that the second is a direct generalization of the first to the case of vector functions. Now using the same technique adopted to prove both lemmas 2 and 1 and the results of these latter and using both equation (15), (16), (17), (18) and (19), equations (37) and (38) are immediate.

**Appendix G. Proof of lemma 7**

Using properties (23), (24) and (25), equation (42) can be written as following:
\[
D^{(1)}_{A} f_{F\Phi}(A, C) = 2\text{diag}\{\text{vec}(A)\}(I_p \otimes I_r)\text{vec}(T^{(3)}C) * (U_{pl} \otimes I_l) \times
\]
\[
(1_l \otimes I_P)\text{vec}(Z))
\] (G.1)

Now, since we are interested in computing the gradient w.r.t. to $A$, we will limit ourself to compute the differential of $D^{(1)}_{A} f_{F\Phi}(A, C)$ w.r.t. to $A$. Thus from both equations (42) and (G.1), the differential of $D^{(1)}_{A} f_{F\Phi}(A, C)$ w.r.t. to $A$ is given by:
\[
d(D^{(1)}_{A} f_{F\Phi}(A, C)) = \text{d}^2 f^{(1)}_{F\Phi}(A, C) =
\]
\[
2\text{diag}\{(I_p \otimes I_r)\text{diag}\{\text{vec}(1_l \otimes Z)\} \text{vec}(T^{(3)}C)\} \text{dvec}(A) +
\]
\[
2\text{diag}\{\text{vec}(A)\}(I_p \otimes I_r)\text{diag}\{\text{vec}(T^{(3)}C)\}(U_{pl} \otimes I_l)(1_l \otimes I_P)\text{dvec}(Z)
\] (G.2)
However $Z = A \boxplus A$, then according to lemma 3 we get:
\[
\begin{align*}
\frac{\partial^2}{\partial p^2}(A, C) &= 2 \text{diag} \{ (I_p \otimes I_f) \text{diag} \{ \text{vec}(I_f \otimes Z) \} \text{vec}(T^{(3)}) \} \text{vec}(A) + \\
&= 4 \text{diag} \{ \text{vec}(A) \} (I_p \otimes I_f) \text{diag} \{ \text{vec}(I_f \otimes Z) \} \text{vec}(T^{(3)}) \} + 4 \text{diag} \{ \text{vec}(A) \} \times \\
&= H_{A,A}^{(1)} f_{T \Theta}(A, C) \text{dvec}(A)
\end{align*}
\]
Hence the result.

Let's now compute $H_{A,A}^{(2)} f_{T \Theta}(A, C)$. Using properties (26) and (25), equation (43) can be written as following:
\[
D_{A}^{(2)} f_{T \Theta}(A, C) = 2 \text{diag} \{ \text{vec}(A) \} (I_f \otimes I_p) (U_p \otimes I_f) \text{diag} \{ \text{vec}(T^{(3)}) \} \times \\
(I_p \otimes I_f) \text{dvec}(A)
\]
Thus from both equations (43) and (G.4), the differential of $D_{A}^{(2)} f_{T \Theta}(A, C)$ w.r.t. to $A$ is given by:
\[
\begin{align*}
d(D_{A}^{(2)} f_{T \Theta}(A, C)) &= d^2 f_{T \Theta}(A, C) = \\
&= 2 \text{diag} \{ (I_f \otimes I_p) (U_p \otimes I_f) \text{diag} \{ \text{vec}(Z) \otimes I_f \} \text{vec}(T^{(3)}) \} \} \text{vec}(A) + \\
&= 2 \text{diag} \{ \text{vec}(A) \} (I_f \otimes I_p) (U_p \otimes I_f) \text{diag} \{ \text{vec}(T^{(3)}) \} \} \} (I_p \otimes I_f) \text{dvec}(A)
\end{align*}
\]
However $Z = A \boxplus A$, then according to lemma 3 we get:
\[
\begin{align*}
d^2 f_{T \Theta}(A, C) &= 2 \text{diag} \{ (I_f \otimes I_p) (U_p \otimes I_f) \text{diag} \{ \text{vec}(Z) \otimes I_f \} \text{vec}(T^{(3)}) \} \} \} \} + \\
&= 4 \text{diag} \{ \text{vec}(A) \} (I_f \otimes I_p) (U_p \otimes I_f) \text{diag} \{ \text{vec}(T^{(3)}) \} \} \} (I_p \otimes I_f) \text{dvec}(A)
\end{align*}
\]
\[
= H_{A,A}^{(2)} f_{T \Theta}(A, C) \text{dvec}(A)
\]
Hence the result.

Appendix H. Proof of lemma 8

For the lake of space, we'll explain the way to compute $H_{A,A}^{(1)} g(A, C)$ and the term $H_{A,A}^{(2)} g(A, C)$ is computed similarly. Note from equation (50) that $H_{A,A}^{(1)} g(A, C)$ depends directly and indirectly on matrix $A$. Indeed, since the matrix $A$ appears explicitly in equation (50), then $H_{A,A}^{(1)} g(A, C)$ depends directly on the matrix $A$. In addition, since $Z = A \boxplus A$ and $Y = Z \otimes Z$, $H_{A,A}^{(1)} g(A, C)$ depends indirectly on matrix $A$ via $Y$ and $Z$. Thus the computation of $H_{A,A}^{(1)} g(A, C)$ is performed as previously using the chain rule. Then, equation (48) can be re-written, using properties (26) and (25) in two equivalent ways:
\[
\begin{align*}
D_{A}^{(1)} g(A, C) &= 4 \text{diag} \{ (I_p \otimes I_f) \text{diag} \{ \text{vec}(I_f \otimes Z) \} (\xi \otimes I_p) \text{vec}(Y) \} \text{vec}(A) \\
&= 4 \text{diag} \{ \text{vec}(A) \} (I_p \otimes I_f) \text{diag} \{ (\xi \otimes I_f) \text{vec}(Y) \} (U_p \otimes I_p) \text{dvec}(A)
\end{align*}
\]
Now, based on the linearity property of the differential we have:

$$d(D^{(1)}_A g(A, C)) = d^2 g^{(1)}(A, C) =
\begin{align*}
4\text{diag}([I_P \otimes I_f']\text{diag}(\text{vec}(I_f \otimes Z)) & (\xi \otimes I_c)\text{vec}(Y)) d\text{vec}(A) + \\
4\text{diag}(\text{vec}(A)) & (I_P \otimes I_f')\text{diag}((\xi \otimes I_c)\text{vec}(Y)) (U_P \otimes I_f) \times \\
(1_f \otimes I_P) & \text{vec}(Z) + 4\text{diag}(\text{vec}(A)) (I_P \otimes I_f') \times \\
& \text{diag}(\text{vec}(I_f \otimes Z)) (\xi \otimes I_c) d\text{vec}(Y)
\end{align*}$$

Now using the results of both lemmas 1 and 2 we get:

$$d^2 g^{(1)}(A, C) = 4\text{diag}([I_P \otimes I_f']\text{diag}(\text{vec}(I_f \otimes A)) (\xi \otimes I_c)\text{vec}(Y)) + 8\text{diag}(\text{vec}(A)) \times \\
(1_f \otimes I_P)\text{diag}((\xi \otimes I_c)\text{vec}(Y)) (U_P \otimes I_f)(1_f \otimes I_P) \text{diag}(\text{vec}(A)) \times \\
8\text{diag}(\text{vec}(A)) (I_P \otimes I_f')\text{diag}(\text{vec}(I_f \otimes A)) (\xi \otimes I_c) (\text{diag}(\text{vec}(I_f \otimes Z)) (I_P \otimes I_f') \times \\
& \text{diag}(\text{vec}(Z) \otimes I_f) (U_P \otimes I_f)(1_f \otimes I_P))d\text{vec}(A)$$

Hence the result.

**Appendix I. Proof of lemma 9**

We found from equation (39) that $H_{C,C}\Psi(A, C)$ is given by:

$$H_{C,C}\Psi(A, C) = -2H_{C,F}f_{rn}(A, C) + H_{C,C}g(A, C) \quad \text{(I.1)}$$

Since the according to lemma 5 (i.e. equation (34)), $D_{C,F}f_{rn}(A, C)$ does not depend on $C$, then $H_{C,C}f_{rn}(A, C) = 0$. Thus the Hessian of $\Psi$ (6) depends only on $H_{C,C}g(A, C)$. From equation (34) we have:

$$D_{Cg}(A, C) = 2\text{vec}(C) (Y'Y \otimes I_c) \quad \text{(I.2)}$$

Then:

$$d(D_{Cg}(A, C)) = d^2 g(A, C) = 2(Y'Y \otimes I_c) d\text{vec}(C) = H_{C,C}g(A, C) d\text{vec}(C) = H_{C,C}\Psi(A, C) d\text{vec}(C) \quad \text{(I.3)}$$

Hence the result.

**References**


