Computation of low-rank tensor approximation under existence constraint via a forward-backward algorithm

Marouane Nazih, Khalid Minaoui, Elaheh Sobhani, Pierre Comon

Performance analysis of the Forward-Backward algorithm to compute the canonical polyadic tensor decomposition under coherence constraint

Marouane Nazih[a], Khalid Minaoui[a], Elaheh Sobhani[b], Pierre Comon[b]

[a] *LRIT Laboratory, associated unit to CNRST (URAC29), IT Rabat Center, Faculty of sciences in Rabat, Mohammed V University, Rabat, Morocco.*
[b] *GIPSA-lab, Univ. Grenoble Alpes, CNRS, Grenoble, France.*

Abstract

The Canonical Polyadic (CP) tensor decomposition has become an attractive mathematical tool in several fields during the last ten years. This decomposition is very powerful for representing and analyzing multidimensional data. The most attractive feature of the CP decomposition is its uniqueness, contrary to rank-revealing matrix decompositions, where the problem of rotational invariance remains. This paper presents the performance analysis of iterative descent algorithms for calculating the CP decomposition of tensors when columns of factor matrices are almost collinear – i.e. swamp problems arise. We propose in this paper a new and efficient proximal algorithm based on the Forward Backward splitting method. More precisely, the existence of the best low-rank tensor approximation is ensured thanks to a coherence constraint implemented via a logarithmic regularized barrier. Computer experiments demonstrate the efficiency and stability of the proposed algorithm in comparison to other iterative algorithms in the literature for the normal case, and also producing significant results even in difficult situations.

Keywords: CP decomposition; Tensor; Coherence constraint; Swamp; Forward Backward splitting; Low-rank approximation.

Email addresses: marouane.nazih1@gmail.com (Marouane Nazih), khalid.minaoui@um5.ac.ma (Khalid Minaoui), elaheh.sobhani@grenoble-inp.fr (Elaheh Sobhani), pierre.comon@grenoble-inp.fr (Pierre Comon)

1. INTRODUCTION

In numerous applications, signals or data may depend on several quantities such as spatial coordinates, velocity, time, frequency, temperature, etc. And are therefore naturally represented by higher-order arrays of numerical values, which are generally known as tensors. Basically, a *tensor* is considered as a mathematical object that possesses the properties of multi-linearity when changing the coordinate system [1]. For our purposes, it will be sufficient to see a tensor of order N as a multi-dimensional array in which every element is accessed via N indices. For instance, a first-order tensor is a vector, which is simply a column of numbers, a second-order tensor is a matrix, a third-order tensor appears as numbers arranged in a rectangular box (or a cube, if all modes have the same dimension), etc. Tensors of order higher than two possess properties that are not enjoyed by matrices and vectors.

We shall mainly focus on the so-called *Canonical Polyadic* (CP) decomposition of tensors [2]. Because of a rediscovery forty years later, it received other names such as Parafac [3] [4] or Candecomp [5]. The acronym "CP" can smartly stand for either "Canonical Polyadic" or "Candecomp/Parafac", as pointed out in [6] [7]. We shall assume this terminology. The CP decomposition has been already used in various fields [8] [9] [10] [11] [12] [13] [14]. The main interest in using the CP decomposition lies in its uniqueness, under rather mild hypotheses [15] [16]. Other tensor decompositions exist, but permit only compression and not parameter identification since they are not unique [17][18] [19].

Various CP decomposition algorithms can encounter problems of slowness or sometimes lack of convergence; such cases may be due to tensor degeneracies. These situations have been well categorized by Richard Harshman [20] into the following three cases: *Bottleneck* when two or more factors in one of the modes are almost collinear [21], *Swamp* when all modes have at least two quasi-collinear factors [21] [22] [23] – a general case of bottleneck, and *CP-degeneracies*

when some of the factors diverge to infinity while tend to cancelling each other, producing a better data fit [24] [25].

The traditional algorithm to compute the CP decomposition is ALS (Alternating Least Squares), which was initially proposed in [5]. Basically, the ALS is an iterative algorithm that progressively updates each unknown parameter individually in an alternating fashion starting from an initial guess. ALS continues until it can no longer improve the solution, or it reaches the allowed maximum number of iterations. As a result, the system to be solved is then turned into three simple least squares (LS) problems. The ALS algorithm converges towards a local minimum under mild conditions [26]. However, its convergence towards the global minimum can sometimes be slow, if ever reached. Furthermore, the convergence of the algorithm may, in certain cases, fall in *swamps* [22], where the convergence rate is very low and the error between two successive iterations remains unchanged.

Various solutions have been proposed to face the slowness of convergence of the ALS algorithm, for instance [27] [28] [12]. The idea is to produce a good initialization via a Generalized Eigenvalue Decomposition (GEVD) of two tensor slices. But such an initialization assumes that the two factor matrices are of full rank, and that the third one doesn't contain zero elements. Another way to increase the ALS convergence speed is to compress the tensor via a Tucker3 decomposition [29] [30]. In [12], the Tucker3 compression is applied as well as an initialization based on the proper analysis; this process has become a common practice to reduce the computational burden [7]. Other tricks to speed-up convergence include the Enhanced Line Search (ELS) method [21], which has demonstrated its effectiveness in cases of tensor affected by degenerative factors (Factor degeneracies), or to reduce ALS sensitivity to initialization [31].

The above algorithms enhance the convergence speed of the ALS algorithm but are still unable to handle swamp phenomenon. Recent studies [13] [32] [33] have shown that the introduction of coherence constraints overcome these phenomena. The solution proposed in [33] is a direct adaptation of ALS using the Dykstra projection algorithm over all correlation matrices. In the proposals

[13] [32], which consist in estimating the factor matrices in a simultaneous way, such methods have not only proved to be efficient and stable for calculating the CP decomposition in the normal case, but also in difficult cases, where the estimated factors are close to collinear.

This document presents a performance evaluation of the iterative descent algorithms for calculating CP decomposition such as the unconstrained gradient descent algorithm [11], the constrained gradient descent algorithm [13], the proximal gradient descent algorithm [32] and the accelerated proximal gradient descent algorithm [32] and where we propose another new algorithm links the logarithmic barrier function with proximal methods [34], which are now reliable and powerful optimization tools, leading to a variety of proximal algorithms, such as the Proximal Point algorithm, the Proximal Gradient algorithm, the Forward Backward algorithm, and several others including linearization and/or splitting [35]. We shall be particularly interested in Forward Backward algorithms, since they meet the assumptions of the novel CP decomposition formulation.

The paper is organized as follows. The section 2 states notation and basic properties of tensors. In section 3, formulation and conditions of CP approximation under coherence constraint are explained. In Section 4 we present the novel formulation for CP Decomposition as well as our new optimization algorithm. In Section 5 we report computer experiments, and eventually conclude in Section 6.

2. Preliminaries

2.1. Tensor notations

Outer (tensor) product. Given two vectors $\mathbf{u} \in \mathbb{C}^I$ and $\mathbf{v} \in \mathbb{C}^J$ their outer (tensor) product $\mathbf{u}\otimes\mathbf{v}$ is defined as the $I \times J$ matrix:

$$\mathbf{M} = \mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T.$$

A third-order *decomposable* tensor $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$ with entries T_{ijk} can be expressed as [36] [37] [38]:

$$T_{ijk} = u_i v_j w_k.$$

Using outer (tensor) product notation, such a decomposable tensor takes the form:

$$\mathcal{T} = \mathbf{u} \otimes \mathbf{v} \otimes \mathbf{w}, \tag{1}$$

where $\mathbf{u} \in \mathbb{C}^I$,$\mathbf{v} \in \mathbb{C}^J$ and $\mathbf{w} \in \mathbb{C}^K$. Above, $\otimes$ denotes the so-called tensor (outer) product. In view of the CP decomposition that will be introduced next, a decomposable tensor has a rank equal to 1, hence its alternative name is *rank-1 tensor*.

Frobenius norm . The Frobenius norm of a tensor is defined as:

$$\|\mathcal{T}\|_F^2 = \langle \mathcal{T}, \mathcal{T} \rangle = \sum_{i,j,k} \mid T_{ijk} \mid^2 . \tag{2}$$

We shall subsequently use $\|\mathcal{X} - \mathcal{Y}\|_F$ as a *distance* between two tensors.

Vectorization. Let $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$ be a tensor, then $vec\{\mathcal{T}\} \in \mathbb{C}^{IJK\times 1}$ represents the column vector defined by :

$$\big[vec\{\mathcal{T}\}\big]_{i+(j-1)I+(k-1)IJ} = T_{ijk} \tag{3}$$

2.2. Tensor decomposition

For our purposes, the word *tensor* will just designate a three-dimensional array of complex numbers, but the generalization to Nth-order tensors, $N \geq 3$, is quite straightforward.

CP decomposition. Any tensor $\mathcal{T}$ can be written as a sum of *decomposable tensors*. When the number of such decomposable terms is minimal, such a decomposition is called Canonical Polyadic (CP), and the number R of terms is called *tensor rank*. Hence a third-order tensor $\mathcal{T}$, of size $I \times J \times K$, admits a CP decomposition taking the form:

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r \mathcal{D}(r), \tag{4}$$

where coefficients λ_r can always be chosen to be real positive, and decomposable tensors $\mathcal{D}(r)$ to have unit norm.

Using Definition (1), the CP decomposition of $\mathcal{T}$ can also be written as:

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r (\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r), \tag{5}$$

It will be useful to use as a compact notation $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_R]$, the vector containing the scaling factors λ_r. Moreover, we define three matrices $\mathbf{A}$= $[\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_R] \in \mathbb{C}^{I \times R}$, $\mathbf{B}$= $[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_R] \in \mathbb{C}^{J \times R}$ and $\mathbf{C}$= $[\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_R] \in \mathbb{C}^{K \times R}$, which are often referred to as *factor matrices*. Note that (5) can be rewritten in terms of tensor entries as:

$$T_{ijk} = \sum_{r=1}^{R} \lambda_r \; a_{ir} b_{ir} c_{ir}. \tag{6}$$

Uniqueness. The CP decomposition (5) can be guaranteed to be unique under various sufficient conditions, all imposing an upper bound on tensor rank. Among them one involves the coherences of factor matrices [39]. However, when $R \leq K$, the condition below [40] is less restrictive:

$$R(R-1) \leq I(I-1)J(J-1) \tag{7}$$

Uniqueness signifies that there is only one set of decomposable tensors $\mathcal{D}(r)$ whose linear combination exactly equals tensor $\mathcal{T}$ in (4) [41] [38]. However, because permuting the terms $\{\lambda_r; \mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r\}$ does not change the sum (5), columns in matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ may be permuted. In addition, decomposable tensors are not defined in a unique way with three vectors, but up to scaling factors because $\mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c} = \alpha\mathbf{a} \otimes \beta\mathbf{b} \otimes \gamma\mathbf{c}$, as long as $\alpha\beta\gamma = 1$. That is why the product of the norms of the three vectors, λ, may be pulled outside the product as in (5); but this does not completely fix the scaling indeterminacy due to the sign ambiguous [11].

3. Tensor Approximation

3.1. Low rank

The problem we want to solve is the following: given a third-order tensor $\mathcal{X} \in \mathbb{C}^{I \times J \times K}$, compute its best approximation of rank R. In other words, for a given R, the goal is to minimize an objective function f of the form:

$$\begin{aligned} f(\mathbf{A}, \mathbf{B}, \mathbf{C}; \boldsymbol{\lambda}) &= \left\| \mathcal{X} - \widehat{\mathcal{X}} \right\|_F^2 \\ &= \left\| \mathcal{X} - \sum_{r=1}^{R} \lambda_r \left(\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r \right) \right\|_F^2 . \end{aligned} \tag{8}$$

As an alternative, we can also write the minimization of (8) using the vectorization defined in (3) with $\mathbf{x} = vec\{\mathcal{X}\}$ as:

$$\begin{aligned} \min_{\widehat{x}} f(\widehat{\mathbf{x}}) &= \min_{\widehat{\mathbf{x}}} \|\mathbf{x} - \widehat{\mathbf{x}}\|_F^2 \\ &= \min_{\mathbf{A},\mathbf{B},\mathbf{C},\boldsymbol{\lambda}} \left\| \mathbf{x} - \sum_{r=1}^{R} \lambda_r (\mathbf{a}_r \boxtimes \mathbf{b}_r \boxtimes \mathbf{c}_r) \right\|_F^2 , \end{aligned} \tag{9}$$

where symbol $\boxtimes$ represents the Kronecker product [42]. The above mentioned minimization is generally an ill-posed problem, but according to [38], consider a coherence constraint in the minimization (9) helps to overcome this ill-posedness.

3.2. Coherence of a matrix

One of the key concepts that we will use later in the conditioning of our problem is the coherence. It can be found in the literature under various significations [43] [44] [45]; the definition we shall assume is close to [46], and has been primarily utilized as a measure of the capability of algorithms - such as matching pursuit and basis pursuit - to precisely estimate the representation of a sparse signal. The coherence of a matrix $\mathbf{A}$ is defined as the maximum absolute value of the cross-correlations between its unit-norm columns:

$$\mu(\mathbf{A}) = \max_{i \neq j} |\mathbf{a}_i^H \mathbf{a}_j| \tag{10}$$

The coherence enjoys the following properties: (a) $0 \leq \mu(\mathbf{A}) \leq 1$, (b) $\mu(\mathbf{A}) = 0$ if and only if $\mathbf{a}_1, ..., \mathbf{a}_R$ are orthonormal, and (c) $\mu(\mathbf{A}) = 1$ if and only if $\mathbf{A}$ contains at least two collinear vectors, i.e. $\exists i \neq j, \lambda \neq 0 : \mathbf{a}_i = \lambda \mathbf{a}_j$. Because the L^∞ norm may be bounded by the L^{2p} for large p:

$$\|\mathbf{t}\|_\infty = \max_t\{t_k\} \leq \|\mathbf{t}\|_{2p} = \big(\sum_k t_k^{2p}\big)^{\frac{1}{2p}}, \forall t_k \in \mathbb{R}^+, \forall p \geq 1, \tag{11}$$

the coherence then admits a differentiable upper-bound:

$$\mu(\mathbf{A}) \leq \mu_p(\mathbf{A}) \stackrel{\text{def}}{=} \big(\sum_{i \neq j} |\mathbf{a}_i^H \mathbf{a}_j|^{2p}\big)^{\frac{1}{2p}} \tag{12}$$

3.3. Conditioning

Contrary to matrices, which can be decomposed into a sum of rank-1 terms in several ways, that unfortunately are not unique except if strong constraints such as non-negativity or orthogonality are imposed, the CP decomposition (4) of tensors with order higher than two, $N > 2$, is often unique under mild assumptions [3] [42][12]; this constitutes one of the attractive properties of the CP decomposition.

In [11], it has been proposed to fix the scaling factor $\boldsymbol{\lambda}$ so as to properly control the conditioning of the problem. More precisely, for given matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the optimal value $\boldsymbol{\lambda}_o$ minimizing the error f is determined by cancelling the gradient of (8) w.r.t. $\boldsymbol{\lambda}$, which results in the linear system:

$$\mathbf{G}\,\boldsymbol{\lambda}_o = \mathbf{s}, \tag{13}$$

where $\mathbf{G}$ is the Gram matrix of size $R \times R$ defined by: $\mathbf{G}_{pq} = \mathbf{a}_p^H \mathbf{a}_q\, \mathbf{b}_p^H \mathbf{b}_q\, \mathbf{c}_p^H \mathbf{c}_q$ and $\mathbf{s}$ is the R-dimensional vector defined by: $s_r = \Sigma_{ijk} T_{ijk} A_{ir} B_{jr} C_{kr}$. It can hence be concluded that coherences appear in the expression of $\mathbf{G}$, which reveals that coherence plays a key role in the problem conditioning.

3.4. Existence of the best approximation

It was shown in [39] that the optimization problem (8) is generally ill-posed. But under the constraint below

$$\mu(\mathbf{A})\,\mu(\mathbf{B})\,\mu(\mathbf{C}) \leq \frac{1}{R-1}, \tag{14}$$

the best rank-R approximation exists and the infimum of (8) is attained. This is because error (8) becomes coercive once condition (14) is satisfied [39], and since it is continuous, it must reach its minimum. Note that condition (14) is sufficient but not necessary. Moreover, it is not differentiable because of the presence of max operators, and this might be a difficulty in some numerical algorithms. To overcome this difficulty, the idea is to use an L^{2p} norm defined in (12) instead of L^{∞}. In fact, not only $\|\mathbf{x}\|_{\infty} \leq \|\mathbf{x}\|_{2p}$, $\forall p > 0$, but also $\|\mathbf{x}\|_{2p}$ becomes a good approximation of $\|\mathbf{x}\|_{\infty}$ for large p [13]. This leads us to assume the constraints:

$$C_p(\mathbf{x}) \stackrel{def}{=} 1 - R + \frac{1}{\mu_p(\mathbf{A})\,\mu_p(\mathbf{B})\,\mu_p(\mathbf{C})} \geqslant 0 \tag{15}$$

4. Proposed Method

4.1. Problem formulation

For the sake of simplicity of writing, we will stack factor matrices in a single vector defined as: $\mathbf{x} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$, and note that in this article, we have chosen to define the function g as the logarithmic term $-ln(C_p(\mathbf{x}))$ of the coherence constraint $C_p(\mathbf{x})$. Formally, we face a constrained optimization problem of the form:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t.} \quad g(\mathbf{x}) \geq 0. \end{aligned} \tag{16}$$

This optimization problem with inequality constraint can be solved by converting it to an unconstrained optimization problem with an appropriate penalty, thanks to Barrier's functions [47], which form a one-sided penalty function.

The cost function to be minimized at this point is composed of a sum of two terms, one of which concerns data fidelity – denoted by f – and the other one relates to a priori information on the target parameters which is in our case the

sufficient condition defined in (15) – denoted by g. This leads to:

$$\underset{\mathbf{x}\in\mathbb{R}^N}{\text{minimize}} \quad \underbrace{f(\mathbf{x})}_{data\,fidelity} + \eta \quad \underbrace{g(\mathbf{x})}_{constraint} \quad , \tag{17}$$

where the function g acts as a barrier in the interior point methods [48] and η is the penalty positive barrier parameter which decreases through iterations.

One of the approaches to estimate a solution of (17) when it is difficult to evaluate directly, or when it requires numerous iterations and more computing time to converge, is to introduce an intermediate variable which results in sub-problems that take a particular form called proximal operators.

Various methods to solve problems of the form (17) have been proposed in the literature based on proximal operators. Formally, let assume that h be a proper lower semi-continuous function which is bounded from below. The proximal operator $\mathbf{prox}_h : R^N \mapsto R^N$ of h is defined by :

$$\mathbf{prox}_h(v) = \arg\min_{\mathbf{x}}(h(\mathbf{x}) + \frac{1}{2\rho}\|\mathbf{x}-\mathbf{v}\|_2^2),$$

which admits a unique solution for every $\mathbf{v} \in R^N$ [34].

Principally, our objective function is explicitly written as follows:

$$\underset{\widehat{\mathbf{x}}\in\mathbb{R}^N}{\text{minimize}}\ \mathcal{F}(\widehat{\mathbf{x}}) = \|\mathbf{x}-\widehat{\mathbf{x}}\|_F^2 - \eta\ ln(C_p(\widehat{\mathbf{x}})) \tag{18}$$

Here, we propose to use the Forward Backward method [35] [49] to solve the problem described by (18). The general principle of our method is described in the following paragraph and is summarized in **Algorithm 1**.

Among non-convex splitting methods, one can find the works of [50] [51], where a monotonous descent of the objective value is imposed to ensure convergence, while on the other hand, there are some recent works [49] [52] that introduce a generic method for non-convex non-smooth problems based on Kurdyka-Lojasiewicz theory.

In our case, coercivity assumption verified in Section 3.4 with simple monitoring that imposes a monotonous descent of the objective value would be sufficient to ensure convergence. It can be demonstrated in [53] that the problem to be addressed (18) has at least one solution and that, for any $\mu \in]0, +\infty[$, its solutions are determined by the fixed point equation:

$$\mathbf{x}^{(k+1)} \leftarrow \underbrace{\mathbf{prox}_{\rho^{(k)}\eta\ g}}_{Backward\ step} \underbrace{(\mathbf{x}^{(k)} - \rho^{(k)}\nabla f(\mathbf{x}^{(k)}))}_{Forward\ step} \tag{19}$$

It is clear from (19) that the Forward Backward method consists of solving two main sub-problems, which can be rewritten as:

$$(sp_1)\ \mathbf{y}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \rho^{(k)}\nabla f(\mathbf{x}^{(k)})$$
$$(sp_2)\ \ \mathbf{x}^{(k+1)} \leftarrow \arg\min_{\mathbf{x}}(\eta\ g(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{y}^{(k+1)}\|_2^2)$$

The forward step addresses sub-problem (sp_1) and the backward step sub-problem (sp_2). We now describe in more details the algorithm outlined above.

4.2. Forward Step

This step yields an approximate solution, by only fitting the data - independently of the constraint. This step also involves two stages.

4.2.1. Descent

In this stage, a gradient steepest descent is executed in direction $\mathbf{d}^{(k)}$:

$$\mathbf{d}^{(k)} = -\nabla f(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) = -\nabla f(\mathbf{x}^{(k)}) \tag{20}$$

The gradient expressions needed to obtain the descent direction $\mathbf{d}^{(k)}$ take the form:

$$\frac{\partial f}{\partial \mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A$$

with

$$M^A_{pq} \stackrel{def}{=} \Sigma_{jk}\lambda_p B_{jp} C_{kp} C^*_{kq} B^*_{jq} \lambda^*_q$$
$$N^A_{ip} \stackrel{def}{=} \Sigma_{jk} T_{ijk} B^*_{jp} C^*_{kp} \lambda^*_p$$

The gradient expressions with respect to $\mathbf{B}$ and $\mathbf{C}$ can be obtained similarly to $\mathbf{A}$.

One should note that, although the gradient method may suffer from a slow convergence, particularly for large datasets [54], these shortcomings will be alleviated by the next "Backward" step that adjusts the search orientation based on the proximal operator of the coherence function g.

4.2.2. Backtracking

The second stage of the forward part consists of calculating the appropriate step-size $\rho^{(k)}$ according to the previously calculated direction $\mathbf{d}^{(k)}$. It is fundamental to adopt a good step-size strategy as it has an important influence on the convergence properties of the algorithm. It is possible to use a fixed step-size throughout the iterations, but this choice may be synonymous with low convergence speed if the step-size is chosen to be too small. Moreover, if a too large step-size is chosen, the optimization algorithm may diverge and one will be unable to correctly estimate the factor matrices.

On the other hand, the calculation of the step-size can be made in an exact way. The determination of the exact step-size of a function f corresponds to the Cauchy rule (21). It allows, as its name indicates, to determine the step-size that minimizes at most the function f at each step k, according to the calculated direction $\mathbf{d}^{(k)}$ as:

$$\rho^{(k)} = \arg\min_{\rho \in \mathbb{R}^+} \{f(\mathbf{x}^{(k)} + \rho \mathbf{d}^{(k)})\}. \tag{21}$$

However, the exact resolution may require too much computing time, and cannot be done with infinite precision anyway. Moreover, the additional efficiency that may be gained to the algorithm by an exact step rarely compensates the time spent to determine such a step.

Consequently, the usual strategies for the calculation of the step-size are often based on the verification of less restrictive conditions, which nevertheless guarantee the convergence of the algorithms. Among various methods searching

for the approached step-size, a simple and effective one is called *Backtracking*. This method produces a good approximation of the step-size $\rho^{(k)}$ at a lower cost, where it depends on two constants α and β, with $0 < \alpha < 0.5$ and $0 < \beta < 1$.

This method is called *Backtracking* [55] as it starts with a large step-size $\rho^{(k)}$ at the beginning, then this value is decreased by a factor β so that $\rho \leftarrow \rho * \beta$ until the following sufficient descent condition [56] is verified:

$$f(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < f(\mathbf{x}^{(k)}) + \alpha\rho^{(k)}\nabla f(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)} \tag{22}$$

Finally, the *Forward step* can be computed as:

$$\mathbf{y}^{(k+1)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{23}$$

4.3. Backward Step

This step represents the strength of the proposed algorithm. While the previous step focused on the function f related to data fidelity, this "Backward" step permits to better adjust the descent direction based on the function g associated with the coherence constraint. This is ensured firstly by applying the proximal operator to the final outcome of the previous "Forward" Step, i.e. $\mathbf{y}^{(k+1)}$, as:

$$\begin{aligned}\mathbf{z}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\eta\ g}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\eta\ g}(\mathbf{y}^{(k+1)}) \\ &= \arg\min_{\mathbf{x}} \underbrace{(\eta\ g(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{y}^{(k+1)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{24}$$

The solution of the proximal $\mathbf{prox}_g(\mathbf{y}^{(k+1)})$ allows us to benefit both from the minimization of function g as well as to be close to $\mathbf{y}^{(k+1)}$, this can be intuitively interpreted as a projection of $\mathbf{y}^{(k+1)}$ on the feasible set.

Our constraint function g is differentiable. This will allow us to calculate an approximation of the proximal operator of g through three main stages.

4.3.1. Gradient of $\mathcal{H}$

The gradient of $\mathcal{H}$ takes the form:

$$\nabla\mathcal{H}(\mathbf{x}) = \nabla\ \eta\ g(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{y}^{(k+1)}), \tag{25}$$

in which the gradient $\nabla g(X)$ is computed as:

$$\frac{\partial g}{\partial \mathbf{A}} = \frac{-1}{\mathcal{C}_p}\mathcal{L}_p^A \mathbf{A}[(\mathbf{A}^H\mathbf{A}) \boxdot \Omega^A - I], \tag{26}$$

where $\boxdot$ denotes the Hadamard entry-wise product,

$$\mathcal{L}_p^A \overset{def}{=} (\Sigma_{p<q}|\mathbf{a}_p^H\mathbf{a}_q|^{2p})^{\frac{-1}{2p}-1}\rho(\mathbf{B},p)^{-1}\rho(\mathbf{C},p)^{-1},$$

and $\Omega_{pq}^A \overset{def}{=} |\mathbf{a}_q^H\mathbf{a}_p|^{2p-2}$. Expressions for $\mathbf{B}$ and $\mathbf{C}$ are similar.

It is important to note that the minimization process is interrupted once we first encounter a better candidate improving $\mathbf{y}^{(k+1)}$, meaning that the new iterate $\mathbf{x}^{(k+1)}$ ensures the minimization of function g while remaining in a neighborhood of $\mathbf{y}^{(k+1)}$.

4.3.2. Relaxation

The Forward Backward (FB) offers some flexibility in the choice of $\mathbf{x}^{(k+1)}$ through a relaxation parameter $\tau^{(k)} \in [\epsilon, 1]$, that acts as an interpolation of the current point $\mathbf{z}^{(k+1)}$ and the previous point $\mathbf{x}^{(k)}$ as:

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \tau^{(k)}(\mathbf{prox}_{\rho^{(k)}\eta\ g}(\mathbf{y}^{(k+1)}) - \mathbf{x}_k) \\ &= \mathbf{x}_k + \tau^{(k)}(\mathbf{z}^{(k+1)} - \mathbf{x}_k),\end{aligned} \tag{27}$$

4.3.3. Descent control

The Forward Backward (FB) is not a monotonous algorithm [57]. This may have an impact on the general minimization problem (18), which may possibly yield a value $\mathcal{F}(\mathbf{x}_{k+1})$ larger than $\mathcal{F}(\mathbf{x}_k)$. To avoid this problem we append a descent validity test:

$$\mathbf{x}_{k+1} = \begin{cases}\mathbf{x}_{k+1} & \text{if } \mathcal{F}(\mathbf{x}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{y}_{k+1} & \text{otherwise.}\end{cases} \tag{28}$$

The monitoring described above allows for an easy migration to the gradient algorithm in case of failure of the "Backward" step to find a better candidate that improves the general direction of minimization.

Algorithm 1: Forward Backward (FB) algorithm to minimize (18)

Choose R satisfying (7);

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ to matrices with unit-norm columns, $\tau^{(k)}$,η, ρ ;

Calculate the optimal scaling factor $\boldsymbol{\lambda}^*$ by solving (13): $\mathbf{G}_0\boldsymbol{\lambda}^* = \mathbf{f}_0$;

for $k \geq 1$ *and subject to a stopping criterion,* **do**

1. Forward Step

   (a) Compute the descent direction $\mathbf{d}^{(k)}$ as the gradient according to (20) w.r.t. $\mathbf{x}_k$:
   $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}_k)$

   (b) Calculate a step-size ρ_k using the backtracking method such:
   $f(\mathbf{x}_k + \rho_k \mathbf{d}^{(k)}) < f(\mathbf{x}_k)$

   (c) Update
   $\mathbf{y}_{k+1} = \mathbf{x}_k + \rho_k \mathbf{d}^{(k)}$

2. Backward Step

   (a) Compute the approximate proximal operator of g at $\mathbf{y}_{k+1}$ using (24) such as:
   $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_k \eta\ g}(\mathbf{y}_{k+1})$

   (b) Relaxation
   $\mathbf{x}_{k+1} = \mathbf{x}_k + \tau^{(k)}(\mathbf{z}_{k+1} - \mathbf{x}_k)$

   (c) Monitoring
   $$\mathbf{x}_{k+1} = \begin{cases} \mathbf{x}_{k+1} & \text{if } \mathcal{F}(\mathbf{x}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{y}_{k+1} & \text{otherwise.} \end{cases}$$

3. Extract the three blocks of $\mathbf{x}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ and $\mathbf{C}_{k+1}$

4. Normalize the columns of $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ and $\mathbf{C}_{k+1}$

5. calculation of the optimal scaling factor $\boldsymbol{\lambda}^*$ using (13) such as: $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{S}$

end for

5. RESULTS AND DISCUSSIONS

In this section, we compare various algorithms based on computer experiments performed under two scenarios: (i) the first one, which is is a concrete example in which it is necessary to deal with the swamp phenomenon, i.e. where the factor matrices are highly correlated in all three modes and the constraint is not satisfied ($C_p \leq 0$) at convergence, and (ii) the second scenario addresses the normal case of the CP decomposition, where the limit point lies in the admissible region, i.e. when the constraint is satisfied ($C_p \geq 0$) at convergence, note also that in this second scenario we have provided three examples of tensors with different tensor sizes.

To see the interest of our proposed algorithm, we compare it to other CPD algorithms: i) The unconstrained gradient (UG) descent algorithm ii) The constrained gradient (CG) descent algorithm [13], iii) The constrained proximal gradient algorithm (PG) [32], iv) The constrained accelerated proximal gradient algorithm (APG)[32].

To illustrate the behaviour of our FB implementation, we evaluate the performance of each algorithm in terms of three criteria, namely accuracy, CPU time and congruence.

The best congruence sum requires finding the best permutation σ among the factor matrix columns, defined as:

$$\max_{\sigma} = \sum_{r=1}^{R} \frac{|a_r^H \widehat{a}_{\sigma(r)}|}{\|a_r\| \|\widehat{a}_{\sigma(r)}\|} \frac{|b_r^H \widehat{b}_{\sigma(r)}|}{\|b_r\| \|\widehat{b}_{\sigma(r)}\|} \frac{|c_r^H \widehat{c}_{\sigma(r)}|}{\|c_r\| \|\widehat{c}_{\sigma(r)}\|} \tag{29}$$

Note that all algorithms are initialized at each simulation with the same starting points, where data sets are generated by normally distributed random numbers with zero mean and unit variance, and all algorithms share the following common stopping criteria:

(i) The maximum number of iterations is fixed at 10^3.

(ii) The reconstruction error level, subsequently called *accuracy*, which is the

distance between the current tensor and the original tensor, for the best permutation σ.

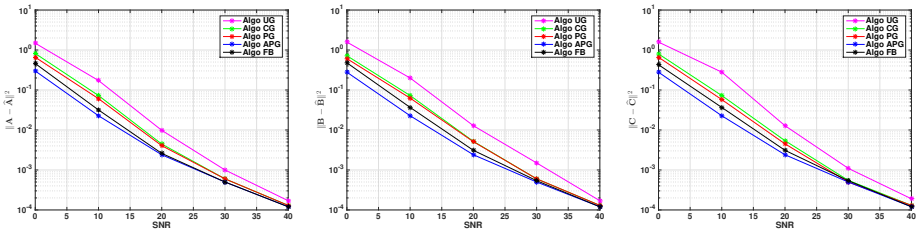
The simulations were performed in Matlab on a computer with Intel i5 CPU (2.6GHz) and 10GB memory running 64bit Mac OS, and the results are obtained by averaging results of 100 executions for all three examples.

5.1. Scenario 1

In this first scenario, we generate a random CP model of size $4 \times 3 \times 10$ with rank 5 and with coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$, which implies that $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0.91$ is larger than $1/(R-1) = 1/2$. This configuration addresses the problem of swamp, where all factor matrices are almost co-linear. In this experiment, η is initialized to 1, and is divided by 10 when $f(x)$ is reduced by less than 10^{-4} and the relaxation parameter $\tau^{(k)}$ is fixed to 0.3 thus allowing a tolerance of 30% for this relaxation. Figure 1 presents the estimation errors of the matrix as a function of SNR. From these results obtained with 100 different starting points, one can clearly observe that the unconstrained gradient (UG) algorithm produces less accurate results in all factor matrices compared to other algorithms, this explains the impact of the coherence constraint on the accuracy of the algorithms. On the other hand, it can also be observed that the proposed FB algorithm is more accurate than both the constrained gradient (CG) and the proximal gradient (PG) algorithms, while the accelerated proximal gradient (APG) algorithm remains the most accurate of all algorithms especially at a high noise level.

Figure 2 illustrates the best sum of congruences as a function of SNR. It can be seen that with the same initialization point, the results of the proposed FB algorithm and the APG algorithm are much more convincing, however, the APG algorithm performs slightly better than the FB algorithm particularly for high noise level. In addition, both the CG and PG algorithms produce comparable results, while the UG algorithm yields lower performance compared to all algorithms when overcoming the swamp phenomenon.

Figure 3 indicates that the UG algorithm produces average results in which



(a) Matrix A (b) Matrix B (c) Matrix C

Figure 1: Matrix estimation errors, with a random tensor of size $4 \times 3 \times 10$ and rank 4.

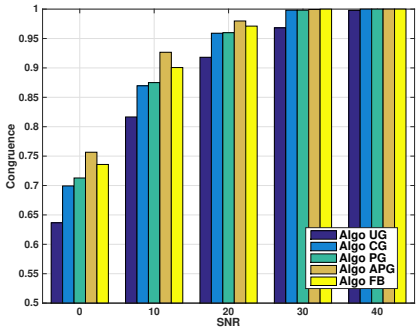


Figure 2: Congruence versus SNR results, $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$ up to an accuracy of 10^{-8} and results with 100 random initializations at each SNR.

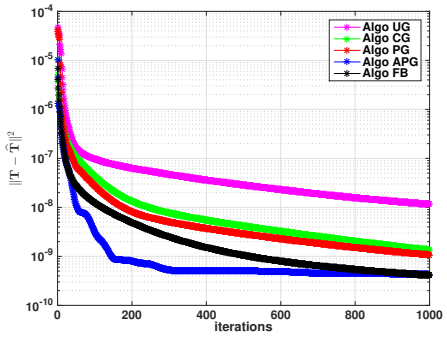


Figure 3: Reconstruction error (8) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 10$ and rank 4.

it only reaches an accuracy of 10^{-8}, whereas the proposed FB algorithm and the APG algorithm still performed much better than both the CG and PG algorithms as they converge faster in terms of number of iterations (especially to reach a precision of 10^{-9}), except that the APG achieves this accuracy in a few iterations compared to the FB algorithm. As a result, the APG algorithm still remains the preferred choice for the difficult case of CP decomposition.

This convergence speed result is also clearly presented in the Table 1, which lists the CPU time required for each algorithm to reach an accuracy of 10^{-8} and where we can notice a significant point in comparison to the previous results indicating that the PG algorithm is faster than the UG algorithm in terms of machine time, a deep inspection also reveals that the proximal algorithms present a high convergence speed, especially to address the problem of swamps.

5.2. Scenario 2

In this second scenario, we address the normal case of the CP decomposition through three different tensor sizes, for which the limit point lies in the admissible region, i.e. when the constraint is satisfied ($C_p \geq 0$) at convergence. It should be noted that the relaxation parameter $\tau^{(k)}$ is fixed to 0.7 for all examples of this scenario, thus allowing a tolerance of 70% for this relaxation, It should also be noted that this choice is made through a series of simulations for

Algorithm	SNR (dB)				
	0	**10**	**20**	**30**	**40**
UG	2.45	1.78	1.66	0.96	0.62
CG	2.04	1.52	1.23	0.72	0.54
PG	1.80	1.21	1.10	0.68	0.50
APG	**1.59**	**0.95**	**0.84**	**0.47**	**0.44**
FB	1.75	1.09	0.91	0.52	0.48

Table 1: CPU time (in seconds) versus SNR results, $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.97$ up to an accuracy of 10^{-8}. These are averaged results over 100 random initializations at each SNR.

that parameter.

5.2.1. Example 1

In this first example, we generate a random CP model of size $2 \times 2 \times 2$ with rank 2, η is initialized to 0.1, and is divided by 100 when $f(\mathbf{x})$ is reduced by less than 10^{-4}. Figure 4 represents the reconstruction error as a function of the number of iterations, from these results, we clearly observe the impact of the constraint on the performance of the algorithms in which we understand that the UG takes much iteration to achieve only an accuracy of 10^{-11}. In addition, we can also notice that the three proximal algorithms are more efficient than the CG with the advantage of the proposed FB algorithm that reaches an accuracy of 10^{-14} in 120 iterations compared to the APG that needs more iterations to reach the same accuracy.

In order to make a more accurate comparison, we evaluated the CPU time of these algorithms. By analysing the results of Table 2, we can clearly see that the proposed FB algorithm proves its efficiency in terms of both accuracy and CPU time compared to other algorithms.

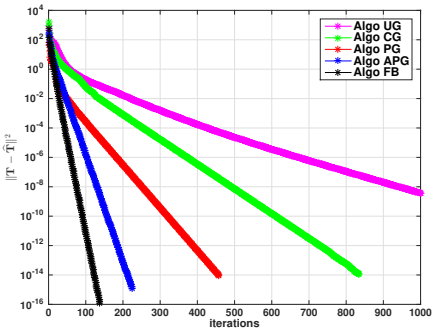


Figure 4: Reconstruction error (8) as a function of the number of iterations. For a tensor of size $2 \times 2 \times 2$ and rank 2.

UG	CG	PG	APG	FB
2.07	1.88	1.65	1.58	**1.47**

Table 2: CPU time (in seconds). For a tensor of size $2 \times 2 \times 2$ and rank 2 up to an accuracy of 10^{-8} and results with 100 random initializations.

5.2.2. Example 2

In this second example, we generate a random CP model of size $4 \times 3 \times 10$ with rank 4, η is initialized to 0.1, and is divided by 100 when $f(\mathbf{x})$ is reduced by less than 10^{-4}. Figure 5 reveals that the proposed FB algorithm requires less than 230 iterations to achieve an accuracy of 10^{-14} followed by APG algorithm that takes more than 300 iterations to achieve the same accuracy. We can also notice that although the CG algorithm produces good results compared to the PG algorithm in terms of accuracy, this performance is a little costly in terms of CPU time for these two algorithms as the Table 4 shows. On the other side, results in Table 4 still proves the efficiency of the proposed algorithm in terms of CPU time compared to other algorithms.

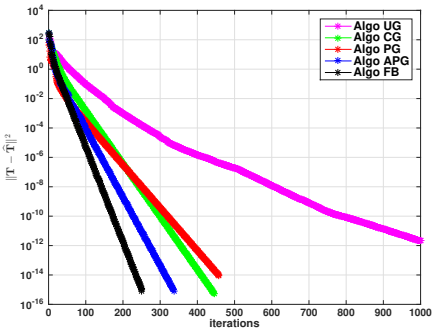


Figure 5: Reconstruction error (8) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 10$ and rank 4.

UG	CG	PG	APG	FB
2.17	1.78	1.73	1.64	**1.57**

Table 3: CPU time (in seconds). For a tensor of size $4 \times 3 \times 10$ and rank 4 up to an accuracy of 10^{-12} and results with 100 random initializations.

5.2.3. Example 3

In this third example, we generate a random CP model of size $40 \times 40 \times 40$ with rank 4, η is initialized to 0.5, and is divided by 100 when $f(\mathbf{x})$ is reduced by less than 10^{-4}. Figure 6 and Table 4 show that with tensors larger than the previous examples the unconstrained algorithm still requires more iterations and more CPU time to achieve an accuracy of 10^{-11}. On the other hand, the proposed FB algorithm significantly exceeds the UG and PG algorithms in terms of the number of iterations and also in terms of machine time. Furthermore, this improvement is slightly significant compared to the APG algorithm.

According to these simulation results, it is easy to see the impact of the coherence constraint in the accuracy of the results, and it is also observed that with different tensor sizes, the efficiency of the proposed algorithm is perceived. This makes the proposed algorithm a better choice to ensure both the accuracy and the convergence speed of the CP decomposition in the normal case.

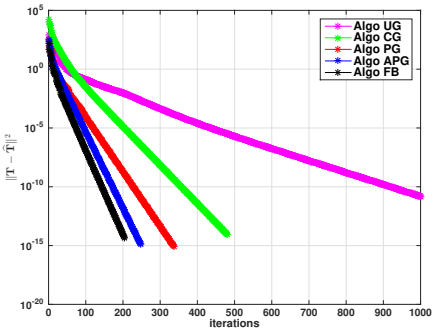


Figure 6: Reconstruction error (8) as a function of the number of iterations. For a tensor of size $40 \times 40 \times 40$ and rank 4.

UG	CG	PG	APG	FB
2.29	1.92	1.68	1.40	**1.46**

Table 4: CPU time (in seconds). For a tensor of size $40 \times 40 \times 40$ and rank 4 up to an accuracy of 10^{-10} and average results with 100 random initializations.

6. Conclusions

We have provided a novel formulation for the CP decomposition related to the logarithmic barrier function. We also described a new method to compute this decomposition for both normal and difficult cases, i.e. when the problem of swamp arises. This method is based on the Forward Backward splitting algorithm, with a simple and effective monitoring strategy capable of calculating the minimal CP decomposition of three-way arrays. We performed a complete comparison based on computer experiments, which emphasized the excellent performances of the proposed FB algorithm in terms of accuracy and convergence speed, compared to other iterative algorithms in the literature for the normal case. On the other hand, in more difficult cases, the accelerated proximal gradient remains the preferred choice to overcome the swamp phenomenon.

References

[1] P. Comon, Tensor decompositionsstate of the art and applications, keynote address in ima conf, Mathematics in Signal Processing, Warwick, UK.

[2] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. and Phys. 6 (1) (1927) 165–189.

[3] R. A. Harshman, et al., Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis.

[4] R. A. Harshman, Determination and proof of minimum uniqueness conditions for parafac1, UCLA Working Papers in phonetics 22 (111-117) (1972) 3.

[5] J. D. Carroll, J.-J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition, Psychometrika 35 (3) (1970) 283–319.

[6] H. A. L. Kiers, Towards a standardized notation and terminology in multiway analysis, J. Chemometrics 14 (2000) 105–122.

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

[8] R. Bro, Parafac. tutorial and applications, Chemometrics and intelligent laboratory systems 38 (2) (1997) 149–171.

[9] K. R. Murphy, C. A. Stedmon, D. Graeber, R. Bro, Fluorescence spectroscopy and multi-way techniques. parafac, Analytical Methods 5 (23) (2013) 6557–6566.

[10] C. Jutten, J. Herault, Blind separation of sources, part i: An adaptive algorithm based on neuromimetic architecture, Signal processing 24 (1) (1991) 1–10.

[11] A. Rouijel, K. Minaoui, P. Comon, D. Aboutajdine, Cp decomposition approach to blind separation for ds-cdma system using a new performance index, EURASIP Journal on Advances in Signal Processing 2014 (1) (2014) 128.

[12] N. D. Sidiropoulos, G. B. Giannakis, R. Bro, Blind parafac receivers for ds-cdma systems, IEEE Transactions on Signal Processing 48 (3) (2000) 810–823.

[13] S. Sahnoun, P. Comon, Joint source estimation and localization, IEEE Transactions on Signal Processing 63 (10) (2015) 2485–2495.

[14] X. Liu, S. Bourennane, C. Fossati, Denoising of hyperspectral images using the parafac model and statistical performance analysis, IEEE Transactions on Geoscience and Remote Sensing 50 (10) (2012) 3717–3724.

[15] J. B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, Linear algebra and its applications 18 (2) (1977) 95–138.

[16] A. Stegeman, N. D. Sidiropoulos, On kruskals uniqueness condition for the candecomp/parafac decomposition, Linear Algebra and its applications 420 (2-3) (2007) 540–552.

[17] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[18] M. A. O. Vasilescu, D. Terzopoulos, Multilinear analysis of image ensembles: Tensorfaces, in: European conference on computer vision, Springer, 2002, pp. 447–460.

[19] T. G. Kolda, Orthogonal tensor decompositions, SIAM Journal on Matrix Analysis and Applications 23 (1) (2001) 243–255.

[20] J. Kruskal, R. Harshman, M. Lundy, How 3-mfa data can cause degenerate parafac solutions, among other relationships, Multiway data analysis (1989) 115–122.

[21] M. Rajih, P. Comon, R. 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.

[22] B. C. Mitchell, D. S. Burdick, Slowly converging parafac sequences: swamps and two-factor degeneracies, Journal of Chemometrics 8 (2) (1994) 155–168.

[23] W. S. Rayens, B. C. Mitchell, Two-factor degeneracies and a stabilization of parafac, Chemometrics and Intelligent Laboratory Systems 38 (2) (1997) 173–181.

[24] R. A. Harshman, The problem and nature of degenerate solutions or decompositions of 3-way arrays, in: Talk at the Tensor Decompositions Workshop, Palo Alto, CA, American Institute of Mathematics, 2004.

[25] P. Paatero, Construction and analysis of degenerate parafac models, Journal of Chemometrics: A Journal of the Chemometrics Society 14 (3) (2000) 285–299.

[26] N. Li, S. Kindermann, C. Navasca, Some convergence results on the regularized alternating least-squares method for tensor decomposition, Lin. Algebra Appl. 438 (2) (2013) 796–812.

[27] E. Sanchez, B. R. Kowalski, Tensorial resolution: a direct trilinear decomposition, Journal of Chemometrics 4 (1) (1990) 29–45.

[28] S. Leurgans, R. Ross, R. Abel, A decomposition for three-way arrays, SIAM Journal on Matrix Analysis and Applications 14 (4) (1993) 1064–1083.

[29] C. A. Andersson, R. Bro, Improving the speed of multi-way algorithms:: Part i. tucker3, Chemometrics and intelligent laboratory systems 42 (1-2) (1998) 93–103.

[30] R. Bro, C. A. Andersson, Improving the speed of multiway algorithms: Part ii: Compression, Chemometrics and intelligent laboratory systems 42 (1-2) (1998) 105–113.

[31] V. Zarzoso, P. Comon, Robust independent component analysis, IEEE Trans. Neural Networks 21 (2) (2010) 248–261, hal-00457300. doi:10.1109/TNN.2009.2035920.

[32] M. Nazih, K. Minaoui, P. Comon, Using the proximal gradient and the accelerated proximal gradient as a canonical polyadic tensor decomposition algorithms in difficult situations, Signal Processing (2020) 107472.

[33] R. C. Farias, J. H. de Morais Goulart, P. Comon, Coherence constrained alternating least squares, in: 2018 26th European Signal Processing Conference (EUSIPCO), IEEE, 2018, pp. 613–617.

[34] N. Parikh, S. Boyd, et al., Proximal algorithms, Foundations and Trends® in Optimization 1 (3) (2014) 127–239.

[35] P. L. Combettes, J.-C. Pesquet, Proximal splitting methods in signal processing, in: Fixed-point algorithms for inverse problems in science and engineering, Springer, 2011, pp. 185–212.

[36] M. V. Catalisano, A. V. Geramita, A. Gimigliano, Ranks of tensors, secant varieties of Segre varieties and fat points, Linear Algebra Appl. 355 (2002) 263–285.

[37] V. D. Silva, L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM Journal on Matrix Analysis Appl. 30 (3) (2008) 1084–1127.

[38] P. Comon, Tensors: a brief introduction, IEEE Signal Processing Magazine 31 (3) (2014) 44–53.

[39] L.-H. Lim, P. Comon, Blind multilinear identification, IEEE Transactions on Information Theory 60 (2) (2014) 1260–1280.

[40] L. De Lathauwer, A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization, SIAM journal on Matrix Analysis and Applications 28 (3) (2006) 642–666.

[41] I. Domanov, L. De Lathauwer, On the uniqueness of the canonical polyadic decomposition of third-order tensors—part i: Basic results and uniqueness of one factor matrix, SIAM Journal on Matrix Analysis and Applications 34 (3) (2013) 855–875.

[42] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM review 51 (3) (2009) 455–500.

[43] E. Candes, J. Romberg, Sparsity and incoherence in compressive sampling, Inverse problems 23 (3) (2007) 969.

[44] D. L. Donoho, M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization, Proceedings of the National Academy of Sciences 100 (5) (2003) 2197–2202.

[45] E. J. Candès, T. Tao, The power of convex relaxation: Near-optimal matrix completion, arXiv preprint arXiv:0903.1476.

[46] R. Gribonval, M. Nielsen, Sparse representations in unions of bases, Ph.D. thesis, INRIA (2002).

[47] S. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[48] J. Gondzio, Interior point methods 25 years later, European Journal of Operational Research 218 (3) (2012) 587–601.

[49] H. Attouch, J. Bolte, B. F. Svaiter, Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods, Mathematical Programming 137 (1-2) (2013) 91–129.

[50] M. Fukushima, H. Mine, A generalized proximal point algorithm for certain non-convex minimization problems, International Journal of Systems Science 12 (8) (1981) 989–1000.

[51] Y. Nesterov, Gradient methods for minimizing composite functions, Mathematical Programming 140 (1) (2013) 125–161.

[52] H. Li, Z. Lin, Accelerated proximal gradient methods for nonconvex programming, in: Advances in neural information processing systems, 2015, pp. 379–387.

[53] J.-L. Starck, F. Murtagh, J. M. Fadili, Sparse image and signal processing: wavelets, curvelets, morphological diversity, Cambridge university press, 2010.

[54] E. Lee, S. Waziruddin, Applying gradient projection and conjugate gradient to the optimum operation of reservoirs 1, JAWRA Journal of the American Water Resources Association 6 (5) (1970) 713–724.

[55] P. Civicioglu, Backtracking search optimization algorithm for numerical optimization problems, Applied Mathematics and computation 219 (15) (2013) 8121–8144.

[56] M. Nazih, K. Minaoui, A progression strategy of proximal algorithm for the unconstrained optimization, in: 2018 4th International Conference on Optimization and Applications (ICOA), IEEE, 2018, pp. 1–6.

[57] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM journal on imaging sciences 2 (1) (2009) 183–202.