Using the Proximal Gradient and the Accelerated Proximal Gradient as a Canonical Polyadic tensor decomposition Algorithms in difficult situations

Marouane Nazih, Khalid Minaoui, Pierre Comon

Using the Proximal Gradient and the Accelerated Proximal Gradient as a Canonical Polyadic tensor decomposition Algorithms in difficult situations

Marouane Nazih[a], Khalid Minaoui[a], 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

Canonical Polyadic (CP) tensor decomposition is useful in many real-world applications due to its uniqueness, and the ease of interpretation of its factor matrices. This work addresses the problem of calculating the CP decomposition of tensors in difficult cases where the factor matrices in one or all modes are almost collinear – i.e. bottleneck or swamp problems arise. This is done by introducing a constraint on the coherences of the factor matrices that ensures the existence of a best low-rank approximation, which makes it possible to estimate these highly correlated factors. Two new algorithms optimizing the CP decomposition based on proximal methods are proposed. Simulation results are provided and demonstrate the good behaviour of these algorithms, as well as a better compromise between accuracy and convergence speed than other algorithms in the literature.

Keywords: CP decomposition; Tensor; Proximal gradient; Accelerated proximal gradient; Optimization

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

1. INTRODUCTION

In a wide range of applications, it is necessary to handle quantities with several indices. These quantities are often referred to as tensors. The definition of a tensor will generally depend on the scientific field in which it is used. Generally, a tensor is treated as a mathematical object that owns the property of multi-linearity when changing the coordinate system [1]. We consider that a tensor of order N represents a multidimensional array in which every element is addressed via N indices. For example, a scalar is a tensor of order 0, a vector is a tensor of order 1, a matrix is a second-order tensor and a cube of values is a third-order tensor, etc. In this paper, we focus on tensors of order higher than two since they possess properties which are not valid for matrices and vectors.

Among tensor decompositions, we shall be mainly interested in the so-called *Canonical Polyadic* (CP) decomposition [2], rediscovered forty years and named Parafac [3, 4] or Candecomp [5]. As pointed out in [6, 7], the acronym "CP" decomposition can smartly stand for either “Canonical Polyadic” or “CanDecomp/Parafac”, and we shall follow this terminology. The CP decomposition has been used in various fields, such as Chemometrics [8] [9], Telecommunications [10, 11, 12, 13] and in Denoising of Hyperspectral Images [14]. The most interesting feature of the CP decomposition is its essential uniqueness for orders strictly larger than two [15] [16], which permits parameter identification.

Another decomposition of interest is Tucker3 [17], and in particular its implementation as a High-Order Singular Value decomposition (HOSVD), where changes of bases are orthogonal. Both HOSVD and CP decomposition reduce to the usual Singular Value decomposition (SVD) in the case of matrices [18], which are second order tensors. HOSVD is mainly used for the purposes of compression since it is not unique.

There are many situations where CP decomposition algorithms may suffer from the absence or slowness of convergence, and which can be due to degeneracies of the tensor. These situations have been well classified by Richard Harshman [19] in the following three cases:

- **Bottleneck:** A bottleneck arises when two or more factors in one of the modes are almost collinear [20].
- **Swamp:** The swamp phenomenon occurs when all modes have at least two quasi-collinear factors [20] [21] [22], which can be seen as a general case of bottleneck.
- **CP-degeneracies:** CP-degeneracies can be considered as special case of swamps phenomenon, where some of the factors diverge and, at the same time, tend to cancel each other, leading to a better quality of the fit progresses [23] [24].

There are many algorithms for calculating the CP decomposition. Due to its simplicity of implementation, the most popular in the literature is the Alternating Least Squares (ALS) originally proposed in [5], which is an iterative optimization process that alternately estimates the factor matrices by updating each matrix individually while keeping the others fixed. In this way, the system to be solved is then turned into simple least square (LS) sub-problems. The ALS algorithm is known to converge towards a local minimum under mild conditions [25]. However, the ALS algorithm is highly sensitive to initialization, and its convergence to the global minimum can sometimes be slow, if ever met. In addition, the convergence of the algorithm may, in some cases, fall in swamps [21], where the convergence rate is very low and the error between two consecutive iterations does not decrease.

Several variants of the ALS algorithm have been proposed in the literature in order to reduce the slow convergence of the ALS algorithm. A proper solution has been proposed in [26, 27, 12]. This solution is also known as direct trilinear decomposition [26]; it consists in obtaining a first estimate of the factor matrices by constructing a Generalized Eigenvalue (GEVD) problem from two tensor slices. However, such an initialization requires that the two factor matrices are of full rank, and that the third one does not contain zero elements. In [28], the estimation of the factor matrices of the CP decomposition is linked to the simultaneous matrix diagonalization problem. Another way to improve the convergence speed of the ALS algorithm is to resort to a TUCKER3 compression [29, 30]; this is useful at initialization when the dimensions of the tensor are large. In [12], an algorithm that accelerates ALS convergence is proposed, and applies the TUCKER3 compression method followed by an initialization based on the proper analysis; this has now become a usual practice to reduce the computational burden [7]. The convergence of the ALS algorithm was also improved by the Enhanced Line Search (ELS) method [31], which has proven its usefulness when the tensor is affected by degenerative factors (Factor degeneracies), or to decrease its sensitivity to initialization [32].

The above algorithms improve the speed of the ALS algorithm but remain inadequate to overcome difficult cases when the factor matrices in one or all modes are highly collinear, i.e, when the problem of swamp or bottleneck arises. Recent works [13, 33] have demonstrated that the introduction of coherence constraints avoids this issue. The proposition in [33] is a direct modification of the ALS based on the Dykstra projection algorithm on all correlation matrices, while in proposal [13], which consists in simultaneously estimating factor matrices, this method has proven to be one of the most useful methods to overcome the difficulty where the estimated factors are highly collinear.

In this paper, we propose two algorithms to improve both accuracy and convergence speed of the CP decomposition in difficult situations, where swamp and bottleneck problems arise. These algorithms are based on proximal methods [34] [35], 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 accelerated proximal gradient algorithm, and several others including linearization and/or splitting. We shall be particularly interested in the proximal gradient algorithm and its accelerated version, since they fulfill the assumptions of the CP decomposition problem.

The rest of the paper is organized as follows. The next section presents notations and some properties of third-order tensors. In section 3, we define the coherence constraint ensuring the existence of a best low-rank approximation and then we introduce the proximal methods used in our context. In Section 4 we introduce our new optimization algorithms. In Section 5 we deal with analysis of the simulation results and finally Section 6 concludes the paper.

2. NOTATIONS AND PRELIMINARIES

2.1. Notations and definitions

Let us begin by introducing some key notations and definitions that will be used in this document. Tensors are denoted by calligraphic letters, e.g., $\mathcal{T}$, matrices are denoted by boldface capital letters, e.g., $\mathbf{M}$, vectors are denoted by boldface lowercase letters, e.g., $\mathbf{a}$ and Scalars are denoted by lowercase letters, e.g., a. In addition, the p^{th} column of matrix $\mathbf{A}$ is denoted by $\mathbf{a}_p$, the p^{th} element of a vector $\mathbf{a}$ is denoted by a_p, the entry of a matrix $\mathbf{A}$ in position (i,j) is denoted by A_{ij} and the entry of a tensor $\mathcal{T}$ in position (i,j,k) is denoted by T_{ijk}.

Definition 1. The outer product of two vectors $\mathbf{a} \in \mathbb{C}^I$ and $\mathbf{b} \in \mathbb{C}^J$ defines a matrix $\mathbf{M} \in \mathbb{C}^{I\times J}$

$$\mathbf{M} = \mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^T$$

Similarly, the outer product of three vectors $\mathbf{a} \in \mathbb{C}^I$, $\mathbf{b} \in \mathbb{C}^J$ and $\mathbf{c} \in \mathbb{C}^K$ produces a third order decomposable tensor $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$:

$$\mathcal{T} = \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c} \tag{1}$$

In Equation (1) above, $\otimes$ represents the tensor outer product, so that entry (i,j,k) of tensor $\mathcal{T}$ is defined by the product

$$T_{ijk} = a_i b_j c_k.$$

A tensor $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$ is said to be rank-1 (also referred to as a decomposable tensor [36, 37, 38]) if each of its elements can be represented as : $T_{ijk} = a_i b_j c_k$, in other words, if it can be expressed as the outer product of three vectors, which will be denoted in a compact form as in (1).

Definition 2. The scalar product between two tensors with the same size, $\mathcal{X}$, $\mathcal{Y}$ $\in \mathbb{C}^{I\times J\times K}$, is defined as:

$$\langle \mathcal{X}, \mathcal{Y}\rangle = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} X^{*}_{i,j,k} Y_{i,j,k} \tag{2}$$

where* stands for the complex conjugation.

Definition 3. The Frobenius norm $\|.\|_F$ of a tensor $\mathcal{T} \in \mathbb{C}^{I\times J\times K}$ is derived from the scalar tensor product:

$$\|\mathcal{T}\|_F = \sqrt{\langle \mathcal{T}, \mathcal{T}\rangle} = \sqrt{(\sum_{i,j,k} \mid T_{ijk} \mid^2)} \tag{3}$$

Consequently, the quadratic distance between two tensors $\mathcal{X}$ and $\mathcal{Y}$ of the same size $I\times J\times K$ can be determined by the quantity:

$$\|\mathcal{X} - \mathcal{Y}\|_F^2 \tag{4}$$

Definition 4. 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 :

$$\big[vec\{\mathcal{T}\}\big]_{i+(j-1)I+(k-1)IJ} = T_{ijk} \tag{5}$$

2.2. PRELIMINARIES

A tensor of order N is a mathematical entity defined on a product between N linear spaces, and once the bases of these spaces are fixed, then the tensor may be represented by a N-way array of coordinates [38]. For the sake of simplicity, we use the term tensor in a restricted sense, i.e. as a three-dimensional array of complex numbers (i.e. $N = 3$), but the generalization to Nth order tensors, $N \geq 3$, is straightforward. Let us consider a tensor $\mathcal{T}$ of order 3 with size $I\times J\times K$, its CP-decomposition is defined as follows:

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r \mathcal{D}(r), \tag{6}$$

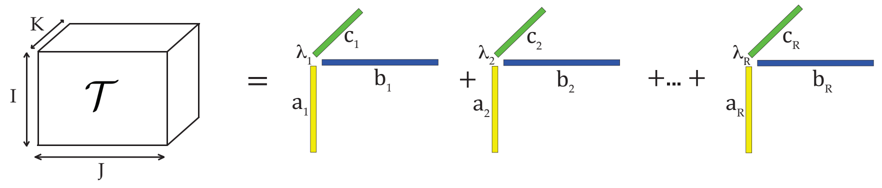


Figure 1: Visualization of the CP decomposition for a third-order tensor

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

From Definition (1) of decomposable tensors, another writing of the CP decomposition makes it more explicit :

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r (\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r), \tag{7}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_R]$ is a vector containing the scaling factors λ_r and the 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 $\mathbf{C}= [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_R] \in \mathbb{C}^{K\times R}$ represent the factor matrices. When the number R of terms in (7) is minimal, it is called the *rank* of tensor $\mathcal{T}$, and decomposition (7) is called the Canonical Polyadic Decomposition (CPD) of $\mathcal{T}$ [1]. Note that (7) can be rewritten in terms of tensor entries as:

$$T_{ijk} = \sum_{r=1}^{R} \lambda_r \; a_{ir} b_{ir} c_{ir}. \tag{8}$$

The CP decomposition is visualized for third-order tensors in Figure 1.

3. CP-DECOMPOSITION AND PROXIMAL OPERATOR

3.1. CP-DECOMPOSITION

3.1.1. Low rank

Even if the tensor of interest is of low-rank, it is often necessary to look for an approximation of low rank R of the observed tensor because of the presence of noise. In the latter case, the observed tensor is indeed generally of *generic* *rank*, strictly larger than R [38]. In the low-rank approximation problem [11], the goal is to minimize an objective function Υ of the form:

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

Alternatively, the minimization of (9) can be written using vectorization defined in (5) with $\mathbf{x} = vec\{\mathcal{X}\}$ as:

$$\begin{aligned}\min_{\widehat{x}}\Upsilon(\widehat{\mathbf{x}}) &= \min_{\widehat{\mathbf{x}}}\|\mathbf{x}-\widehat{\mathbf{x}}\|_F^2 \\ &= \min_{\mathbf{A},\mathbf{B},\mathbf{C},\boldsymbol{\lambda}}\left\|\mathbf{x}-\sum_{r=1}^{R}\lambda_r(\mathbf{a}_r\boxtimes\mathbf{b}_r\boxtimes\mathbf{c}_r)\right\|_F^2\end{aligned} \tag{10}$$

where symbol $\boxtimes$ represents the Kronecker product [39, 38].

3.1.2. Coherence

Let's introduce the concept of coherence that we will use later, when talking about the conditioning of the problem. The notion of coherence has received different names in the literature: the mutual incoherence of two dictionaries [40], the mutual coherence of two dictionaries [41], the coherence of a subspace projection [42], etc. The version here follows that of [43], which has been used particularly as a measure of the ability of algorithms such as basis pursuit and matching pursuit to accurately identify the correct representation of a sparse signal.

Mathematically, let $\mathbb{H}$ be a Hilbert space endowed with the scalar product $\langle\mathbf{a},\mathbf{b}\rangle = \mathbf{a}^H\mathbf{b}$, and $\mathbf{A}\subseteq\mathbb{H}$ be a finite set of elements $\mathbf{a}_i$ with unit norm. The coherence of $\mathbf{A}$ is defined as the maximum absolute value of the cross-correlations between the columns of $\mathbf{A}$:

$$\mu(\mathbf{A}) = \max_{i\neq j}|\mathbf{a}_i^H\mathbf{a}_j| \tag{11}$$

It is obvious that $0 \leq \mu(\mathbf{A}) \leq 1$, that $\mu(A) = 0$ if and only if $\mathbf{a}_1, ..., \mathbf{a}_R$ are orthonormal, and that $\mu(A) = 1$ if and only if $\mathbf{A}$ contains at least two collinear vectors, i.e. $\exists i \neq j, \lambda \neq 0 : \mathbf{a}_i = \lambda \mathbf{a}_j$. It should be noted that the L^∞ norm may be bounded by the L^{2p} norms for large p. This result will allow to bound the coherence by a differentiable quantity:

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

3.1.3. Conditioning of the problem

One of the interesting properties of tensors of order higher than two, $N > 2$, is that their CP decomposition is often unique, which is not the case of matrix decompositions [3] [39][12] (the decomposition of a matrix into a sum of rank-one matrices also exists, but it is not unique, unless some strong constraints are imposed, such as orthogonality or non-negativity).

Uniqueness means that there is only one set of rank-1 tensors whose sum exactly equals tensor $\mathcal{T}$ [44]. Note that permutation indeterminacy is inherent in a sum, and that scaling indeterminacy is inherent in the construction of rank-1 tensors [38]. When the decomposition is unique in that sense, factor matrices are sometimes said to be *essentially unique*. From the above definition of the CP decomposition, it is clear that decomposition (7) is insensitive to:

- Permutation of the rank-1 terms, which refers to the permutation indeterminacy.

- Scaling of vectors a_r, b_r and c_r, provided the produt of the scaling factors is equal to 1, which refers to the scaling indeterminacy.

In numerical algorithms, it is useful to fix indeterminacies. For instance, columns of factor matrices can be normalized and their norm stored in scaling factor $\boldsymbol{\lambda}$ [39]. Another approach described in [11] and used in our present paper consists in calculating the optimal value of the scaling factor $\boldsymbol{\lambda}$ to properly control the conditioning of the problem. For this purpose, and for a given matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the optimal value $\boldsymbol{\lambda}_o$ minimizing the error Υ is determined by cancelling the gradient of (9) w.r.t. $\boldsymbol{\lambda}$, which then results to the linear system:

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

where $\mathbf{G}$ is the Gram matrix of size $R \times R$ defined by: $G_{pq} = (\mathbf{a}_p \boxtimes \mathbf{b}_p \boxtimes \mathbf{c}_p)^H (\mathbf{a}_q \boxtimes \mathbf{b}_q \boxtimes \mathbf{c}_q)$ and f is the R-dimensional vector defined by: $f_r = \Sigma_{ijk} T_{ijk} A_{ir} B_{jr} C_{kr}$. Note that entries of $\mathbf{G}$ can preferably be obtained by $G_{pq} = \mathbf{a}_p^H \mathbf{a}_q\, \mathbf{b}_p^H \mathbf{b}_q\, \mathbf{c}_p^H \mathbf{c}_q$.

Equation (13) indicates that coherence plays a central role in the conditioning of the problem. And we can see that scalar products do not appear individually but via their products. Such a statement has profound consequences, particularly on the existence and uniqueness of the solution to problem (9).

3.1.4. Existence

It has been demonstrated in [45] that if

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

then the best rank-R approximation exists and that the infimum of (9) is attained. The reason is that the error (9) becomes coercive once (14) is satisfied, and then must reach its minimum since it is continuous.

Constraint (14) contains max operators, which are not differentiable. This difficulty can be circumvented by using L^{2p} norms defined in (12), and imposing differentiable constraints $C_p(\mathbf{x})$:

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

3.1.5. Uniqueness

There is a sufficient condition for ensuring the uniqueness of the CP-decomposition (7) in which the coherences are involved [45]. However, the use of the following condition[28] is less restrictive

$$R \le K \ \&\ R(R-1) \le I(I-1)J(J-1) \tag{16}$$

3.2. PROBLEM FORMULATION

As pointed out in (10), for the sake of simplicity, columns of factor matrices are gathered in a single vector $\mathbf{x} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$. The objective to be minimized consists of two terms: one related to 'data fidelity', which is defined in (9) by the cost function Υ, and another one linked to a priori information on model parameters, named 'constraint', which is presently defined in (15) and which will be represented by $\mathcal{G}$. One way of reformulating this constrained problem is to consider the unconstrained minimization of:

$$\mathcal{F}(\mathbf{x}) = \underbrace{\Upsilon(\mathbf{x})}_{data fidelity} + \underbrace{\mathcal{G}(\mathbf{x})}_{constraint} \tag{17}$$

where the function $\mathcal{G}$ acts as a barrier in the interior point methods [46] [47]; for example $\mathcal{G}$ may be defined as the logarithmic term $-ln(C_p(\mathbf{x}))$ weighed by a barrier parameter $\eta > 0$. But in order to be able to fully compare our proposed methods with the one proposed in [13] that is based on gradient descent algorithm, we have chosen to keep the same objective function, namely:

$$\mathcal{F}(\mathbf{x}) = \Upsilon(\mathbf{x}) + \eta \exp(-\gamma C_p(\mathbf{x})) \tag{18}$$

where γ is a parameter that controls the sharpness of the penalty $C_p(\mathbf{x})$ and η is a penalty weight, which decreases through iterations.

3.3. PROXIMAL MAPPING

Proximal mapping has a simple definition yet has long been a powerful tool in optimization, leading to a wide range of algorithms, such as the proximal-point algorithm, the proximal-gradient algorithm, and many other algorithms involving linearization and/or splitting.

Given a function $\mathcal{G}$, the proximal mapping (or proximal operator) [34] maps an input point $\mathbf{x}$ to the minimizer of $\mathcal{G}$ restricted to small proximity to $\mathbf{x}$. The definition of the proximal operator is recalled hereafter.

3.3.1. Definition

Let $\mathcal{G} \in \Gamma_0(\mathbb{R}^N)$[1], for every $\mathbf{x} \in \mathbb{R}^N$, the minimization problem (with parameter $\mu > 0$)

$$\underset{y \in \mathbb{R}^N}{\text{minimize}}\; \mathcal{G}(\mathbf{y}) + \frac{1}{2\mu}\|\mathbf{x} - \mathbf{y}\|_2^2 \tag{19}$$

admits a unique solution, which is denoted by $\mathbf{prox}_{\mu\mathcal{G}}(\mathbf{x})$.

The operator $\mathbf{prox}_{\mu\mathcal{G}} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ thus defined is the prox-operator of $\mathcal{G}$, and the parameter μ controls the stretch to which the proximal operator maps points towards the minimum of $\mathcal{G}$, with larger values of μ associated with mapped points near the minimum, and smaller values giving a smaller movement towards the minimum [34].
In other words, when we evaluate the prox-operator of $\mathcal{G}$, we are attempting to reduce the value of $\mathcal{G}$ by penalizing this reduction to not deviate too much from $\mathbf{x}$ (The parameter μ is like a "step-size" that controls how much we are penalized for moving away from $\mathbf{x}$).

The proximal mapping gets really handy for composite problem when the minimization has the form of $\Upsilon + \mathcal{G}$ with Υ convex and differentiable and $\mathcal{G}$ convex with "simple" proximal mapping in the sense that $\mathcal{G}$ does not require to be differentiable and that its proximal operator can be evaluated efficiently, i.e. its prox-operator admits a closed form [34]. This general composite problem can be solved with one of the proximal methods mentioned above, but those that reply to the assumptions of our specific problem (18), and which are the main proposals in this paper, are the proximal gradient method and the accelerated proximal gradient method. A natural strategy of those methods is firstly to reduce the value of Υ by using unconstrained iterative optimization methods such as descent methods or Newton's method, followed by the reduction of the

[1]is the class of lower semicontinuous convex functions from $\mathbb{R}^N$ to $]-\infty;+\infty]$ such that dom $\mathcal{G} \neq \varnothing$.

value of $\mathcal{G}$ by applying the prox-operator of $\mathcal{G}$ (using the same step-size) and repeat until convergence to a minimizer (under some further conditions). This strategy yields the following iteration:

$$\mathbf{x}^{(k+1)} = \mathbf{prox}_{\mu^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \mu^{(k)}\mathbf{d}^{(k)}) \tag{20}$$

This last strategy describes the general principle of the proximal gradient algorithm. The accelerated proximal gradient algorithm performs an extrapolation at each iteration, by taking into account information from current and previous iterations.

4. PROPOSED OPTIMIZATION METHOD

In this article, we propose two methods to overcome swamp and bottleneck problems based on the proximal gradient (PG) and the accelerated proximal gradient (APG) in order to solve the optimization problem defined in (18). The general principle of the proposed approaches is detailed in the following paragraphs and summarized in **Algorithm 1** and **Algorithm 2**. There are essentially of two basic steps:

- A gradient step associated with the data fidelity term (denoted by the function Υ).
- A proximal step related to the coherence constraint term (denoted by the function $\mathcal{G}$).

4.1. Gradient step

This step improves the approximate solution, by acting only on the data fidelity - regardless of the constraint. This step also involves two other stages.

(i) The first one consists in calculating the descent direction $\mathbf{d}^{(k)}$ of Υ, yielding the steepest decrease. In this paper, we propose to use gradient descent method to keep the minimization as simple as possible, and despite its slow convergence, especially for large data set [48], these drawbacks will be overcome by the second step of the proximal algorithm. The direction is defined as follows:

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{x}^{(k)}) \tag{21}$$

where gradient expressions required to determine the direction of descent $\mathbf{d}^{(k)}$ are of the form:

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^{A} - 2\mathbf{N}^{A} \tag{22}$$

with

$$\begin{aligned} M^{A}_{pq} &\overset{def}{=} \Sigma_{jk}\lambda_p B_{jp} C_{kp} C^{*}_{kq} B^{*}_{jq} \lambda^{*}_{q} \\ N^{A}_{ip} &\overset{def}{=} \Sigma_{jk} T_{ijk} B^{*}_{jp} C^{*}_{kp} \lambda^{*}_{p} \end{aligned} \tag{23}$$

Expressions for gradients w.r.t $\mathbf{B}$ and $\mathbf{C}$ are similar.

(ii) The second stage concerns the determination of the step-size $\rho^{(k)}$ along the chosen direction $\mathbf{d}^{(k)}$. Among many different methods of searching for a good step-size, *Backtracking*, is widely used. It depends on two parameters α and β, with $0 < \alpha < 0.5$ and $0 < \beta < 1$. The idea is to start with a sufficiently large step-size $\rho^{(k)}$ (e.g. $\rho = 1$) at the beginning, and then reduce it as $\rho \leftarrow \rho * \beta$, until the following Armijo condition [49] is verified:

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) \tag{4}$$

To conclude, the *gradient step* can be summarized as:

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{5}$$

4.2. Proximal step

Since the previous step concerns only the data fidelity term Υ, the *proximal step* should adjust the search orientation based on the constraint on coherences $\mathcal{G}$. And to do this, we apply the proximal algorithm to the previous point resulting from the previous step of the gradient, i.e. $\mathbf{z}^{(k)}$, as follows:

$$\begin{aligned}\mathbf{z}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) \\ &= \arg\min_{\mathbf{x}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{z}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{6}$$

This step indicates that $\mathbf{prox}_{\mathcal{G}}(\mathbf{z}^{(k)})$ is a point that compromises between minimizing $\mathcal{G}$ and being near to $\mathbf{z}^{(k)}$. In the general case, the function describing the constraints may not require to be differentiable but it is recommended to be simple in the sense that its proximal operator can be efficiently evaluated. Now it remains to calculate an approximation of the proximal operator of $\mathcal{G}$. We proceed in two stages.

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

$$\nabla\mathcal{H}(\mathbf{x}) = \nabla\mathcal{G}(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) \tag{24}$$

where the gradient $\nabla\mathcal{G}(X)$ is calculated as in [13]:

$$\frac{\partial\mathcal{G}}{\partial\mathbf{A}} = \frac{\gamma}{exp(\gamma\mathcal{C}_p)}\mathcal{L}_p^A\mathbf{A}[(\mathbf{A}^H\mathbf{A}) \boxdot \Omega^A - I] \tag{25}$$

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

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

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

The minimization in the *proximal step* is stopped as soon as we meet a good candidate that improves z_k, meaning that the new iterate z_{k+1} ensures the minimization of the function $\mathcal{G}$ as well as it does not deviate too much from the iterate z_k based on data fidelity.

Monitoring. the Accelerated Proximal Gradient (APG) algorithm first extrapolates a point y_k by a combination of the current point and the previous point as:

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$$

and then solves a proximal operator problem. However, for a bad extrapolation y_k, $\mathcal{F}(x_k)$ may be arbitrarily greater than $\mathcal{F}(x_{k+1})$; this may occur because APG is not a monotonous algorithm [50]. For this purpose, we introduce a monitor that ensures the descent property $\mathcal{F}(x^{(k+1)}) < \mathcal{F}(x^{(k)})$; it is defined as follows:

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{x}_k & \text{otherwise.} \end{cases} \tag{26}$$

5. RESULTS AND DISCUSSION

In this section, we test the performance of the proposed algorithms on three scenarios: (i) in the first one, the limit point lies in the admissible region, i.e. when the constraint is not active ($C_p \geq 0$) at convergence, (ii) the second scenario is a concrete example in which it is necessary to deal with the swamp problem, i.e. where the factor matrices are highly correlated in all three modes and the constraint is active ($C_p \leq 0$) at convergence, and (iii) the last scenario addresses the bottleneck problem, where the factors of the first mode are chosen to be co-linear.

To see the interest of our algorithms, we compare them to two other CPD algorithms: i) their counterpart without constraint and without calculating the optimal value of λ defined in (7), ii) the constrained gradient descent algorithm [13].

The performances are evaluated according to three criteria: the error between the calculated and actual factor matrices, the best sum of the "congruences" [51], and the execution speed. The best congruence sum requires finding

Algorithm 1: Proximal Gradient algorithm (PG) to minimize (18)

```
1 Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ to matrices with unit-norm columns ;
2 calculation of the optimal scaling factor $\boldsymbol{\lambda}^*$ using (13) such as: $\mathbf{G}_0\boldsymbol{\lambda}^* = \mathbf{f}_0$
    ;
3 for $k \geq 1$ and subject to a stopping criterion, do
      1. Gradient Step
          (a) Compute the descent direction $\mathbf{d}^{(k)}$ as the
              gradient according to (23) w.r.t. $\mathbf{x}_k$:
              $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{x}_k)$
          (b) Calculate a step-size $\rho_k$ using the backtracking method such:
              $\Upsilon(\mathbf{x}_k + \rho_k\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}_k)$
          (c) Update
              $\mathbf{z}_k = \mathbf{y}_k + \rho_k\mathbf{d}^{(k)}$
      2. Proximal Step
          (a) Compute the approximate proximal operator
              of $\mathcal{G}$ at $\mathbf{z}_k$ using (24) such as:
              $\mathbf{x}_{(k+1)} = \mathbf{prox}_{\rho_k\mathcal{G}}(\mathbf{z}_k)$
      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: $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{f}$
4 end for
```

Algorithm 2: Accelerated Proximal Gradient algorithm (APG) to minimize (18)

```
1 Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ to matrices with unit-norm columns, and the
    interpolation parameters to : $t_0 = 0$, $t_1 = 1$ ;
2 calculation of the optimal scaling factor $\boldsymbol{\lambda}^*$ using (13) such as: $\mathbf{G}_0\boldsymbol{\lambda}^* = \mathbf{f}_0$
    ;
3 for $k \geq 1$ and subject to a stopping criterion, do
4     $\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1})$;
5     $t_k = \frac{\sqrt{4t_k^2 + 1} + 1}{2}$
        1. Gradient Step
            (a) Compute the descent direction $\mathbf{d}^{(k)}$ as
                the gradient according to (23) w.r.t. $\mathbf{y}_k$:
                $\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{y}_k)$
            (b) Calculate a step-size $\rho_k$ using the backtracking method such:
                $\Upsilon(\mathbf{y}_k + \rho_k\mathbf{d}^{(k)}) < \Upsilon(\mathbf{y}_k)$
            (c) Update
                $\mathbf{z}_k = \mathbf{y}_k + \rho_k\mathbf{d}^{(k)}$
        2. Proximal Step
            (a) Compute the approximate proximal operator
                of $\mathcal{G}$ at $\mathbf{z}_k$ using (24) such as:
                $\mathbf{z}_{(k+1)} = \mathbf{prox}_{\rho_k\mathcal{G}}(\mathbf{z}_k)$
            (b) Monitoring
                $\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{x}_k) \\ \mathbf{x}_k & \text{otherwise.} \end{cases}$
        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: $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{f}$ using (13)
6 end for
```

the best permutation σ among the factor matrix columns; it is defined as :

$$\max_{\sigma} = \sum_{r=1}^{R} \frac{|a_r^H \widehat{a}_{\sigma(r)}|}{\|a_r\|\|\widehat{a}_{\sigma(r)}\|} \frac{|b_r^H \widehat{b}_{\sigma(r)}|}{\|b_r\|\|\widehat{b}_{\sigma(r)}\|} \frac{|c_r^H \widehat{c}_{\sigma(r)}|}{\|c_r\|\|\widehat{c}_{\sigma(r)}\|} \tag{27}$$

In order to obtain comparable results and to visualize the behaviour of each algorithm, the starting points for all methods – namely the unconstrained algorithm, denoted by "Algo Unconstrained", constrained algorithm resolved by gradient descent, denoted by "Algo Constrained Gradient", proximal gradient algorithm denoted by ("Algo PG") and the accelerated proximal gradient, denoted by ("Algo APG") – are initialized using the same initial points and share the same stopping criteria:

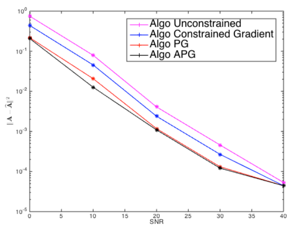
- The tolerance on the Frobenius norm of the gradient divided by the number of entries in the gradient is set at 10^{-8}.
- The maximum number of iteration is set to 10^3.

In all experiments, The computations are run in Matlab on a computer with Intel i5 CPU (2.6GHz) and 10GB memory running 64bit Mac OS. And the results are obtained from 100 Monte Carlo runs for all scenarios. At each iteration and for each SNR value, a new noise realization is drawn. Note also that we fix the same heuristic choices $\rho = 10$ and $\gamma = 5$ as in [13].

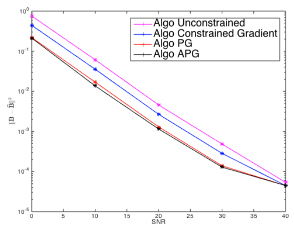
5.1. Simulation 1

In this experience, we generate a random CP model of size $4 \times 3 \times 6$ with rank 3 and with coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$, which implies that $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0.97$ is larger than $\frac{1}{R-1} = \frac{1}{2}$. This configuration describes the case where the constraint is active ($C_p \leq 0$) at convergence.

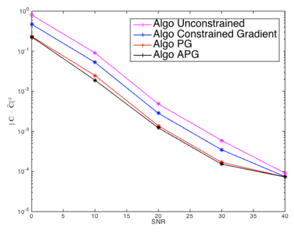
η is varied through iterations. More specifically in this first experiment, η is initialized to 1, and is divided by 10 when $\Upsilon(x)$ is reduced by less than 10^{-4}. Figure 2 illustrates the estimation errors of the matrix as a function of SNR. From these results obtained with 100 different initial points, we can see that the unconstrained algorithm produces poor results compared to other



(a) Matrix A



(b) Matrix B



(c) Matrix C

Figure 2: Matrix estimation errors, with a random tensor of size $4 \times 3 \times 6$ and rank 3.

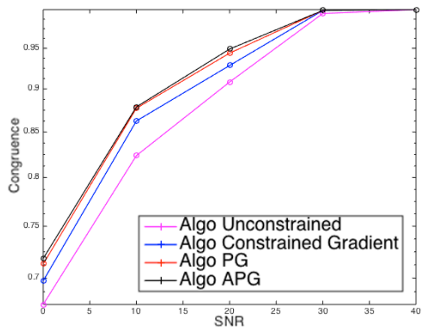
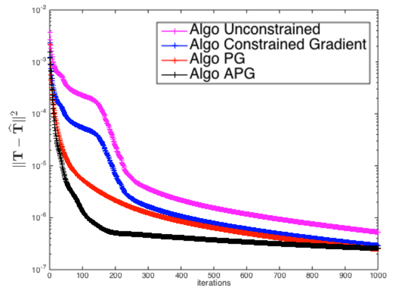


Figure 3: Congruence versus SNR results, $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$ up to a precision of 10^{-6} and results with 100 random initializations at each SNR.

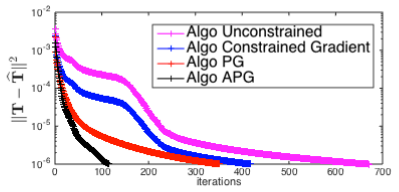
algorithms. On the other hand, we can also observe that the two proposed proximal algorithms are more accurate than the constrained gradient algorithm.

Figure 3 represents the best sum of congruences as a function of SNR. We can visualize that with the same initializations, the results are much more promising, especially at a high SNR, which confirms the accuracy of the proposed proximal algorithms.

In order to show a deeper difference between the Algorithms, we will examine the convergence speed of the tensor reconstruction according to the number of iterations, and we will also explore the CPU time to highlight the run time of each algorithm to achieve an accuracy of 10^{-6}. Figure 4 indicates that the accelerated proximal gradient algorithm converges faster than other algorithms



(a) Up to 1000 iteration



(b) Up to a precision of 10^{-6}

Figure 4: Reconstruction error (9) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 6$ and rank 3.

in terms of number of iterations, followed by proximal gradient algorithm, then by constrained gradient algorithm and finally the unconstrained algorithm which requires a lot of iterations to achieve an accuracy of 10^{-6}. This convergence speed result is also clearly presented in Table 1, which indicates the machine time required for each algorithm to achieve an accuracy of 10^{-6}, and where it is clearly observed that accelerated proximal gradient yields superior results to deal with the swamp phenomenon.

5.2. Simulation 2

In this second experience, we generate a random CP model of size $4 \times 3 \times 6$ with rank 3 and with coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.6$, which implies that $\mu(\mathbf{A})\mu(\mathbf{B})\mu(\mathbf{C}) = 0.216$ is less than $\frac{1}{R-1} = \frac{1}{2}$. This configuration describes the case where the constraint is not active ($C_p \geq 0$) at convergence. The starting points for all methods are randomly generated, followed by 5 iterations of the

	SNR				
Algorithms	0	10	20	30	40
Unconstrained	2.10	1.67	1.44	0.80	0.61
Constrained Gradient	1.83	1.22	1.20	0.64	0.45
PG	1.71	1.13	1.11	0.59	0.44
APG	**1.62**	**0.83**	**0.76**	**0.46**	**0.43**

Table 1: CPU time (in seconds) versus SNR results, $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$ up to a precision of 10^{-6} and results with 100 random initializations at each SNR.

Alternating Least Square (ALS). In this experiment, η is initialized to 0.1, and is divided by 100 when $\Upsilon(x)$ is reduced by less than 10^{-4}.

Figure 5 reports the Reconstruction error (9) as a function of the number of iterations. It can hence be observed that the accelerated proximal gradient algorithm converges faster than other algorithms in terms of number of iterations, followed by proximal gradient algorithm, then by the by constrained gradient algorithm, and finally by the unconstrained algorithm. A deeper inspection also reveals that the accelerated proximal algorithm, the proximal gradient algorithm and the constrained gradient algorithm yield 99% of correct estimations whereas the unconstrained algorithm yields 97%.

5.3. Simulation 3

In this last experience, we generate a random CP model of size $4 \times 3 \times 6$ with rank 3 and with coherences $\mu(\mathbf{A}) = 0.99$, $\mu(\mathbf{B}) = 0.6$, $\mu(\mathbf{C}) = 0.6$. This configuration addresses the problem of bottleneck, where we chose factors of the first mode (i.e, $\mathbf{A}$) to be almost co-linear. In this experiment, η is initialized to 0.1, and is divided by 100 when $\Upsilon(x)$ is reduced by less than 10^{-4}. Figure 6 illustrates the estimation errors of the matrix as a function of SNR. And from these results obtained with 100 different initial points, we can still observe that

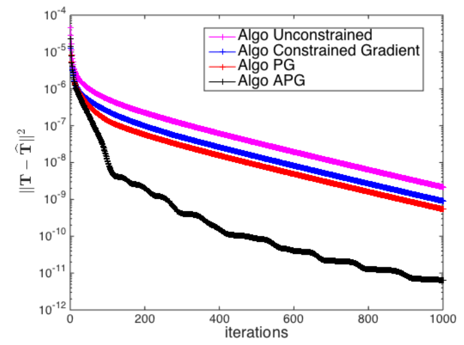
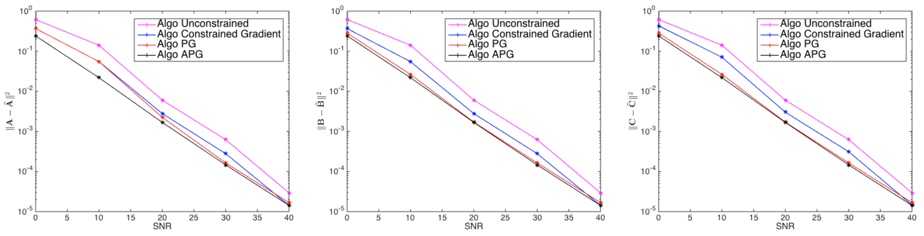


Figure 5: Reconstruction error (9) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 6$ and rank 3.



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

Figure 6: Matrix estimation errors, with a random tensor of size $4 \times 3 \times 6$ and rank 3.

the unconstrained algorithm produces mediocre results compared to other algorithms. On the other hand, the proximal gradient algorithm produces results similar to those of the constrained gradient algorithm in higher SNR values, particularly for the factors of the correlated matrix $\mathbf{A}$. This may be explained by the fact that in higher SNRs, the proximal gradient algorithm is sensitive to the starting points; however the accelerated proximal algorithm remains insensitive to starting points and provides more accuracy than other algorithms.

Figure 8 indicates that the accelerated proximal gradient algorithm converges faster than other algorithms in terms of number of iterations, where both proximal gradient and constrained gradient algorithms yield similar results succeeded by the unconstrained algorithm that requires many iterations.

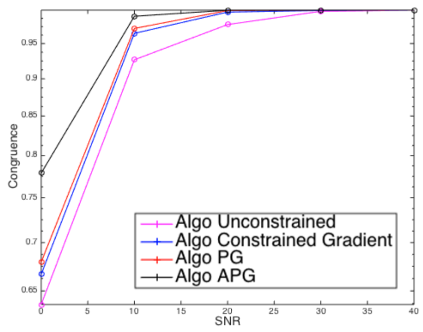


Figure 7: Congruence versus SNR results, $\mu(\mathbf{A}) = 0.99$, $\mu(\mathbf{B}) = 0.6$, $\mu(\mathbf{C}) = 0.6$ up to a precision of 10^{-8} and results with 100 random initializations at each SNR.

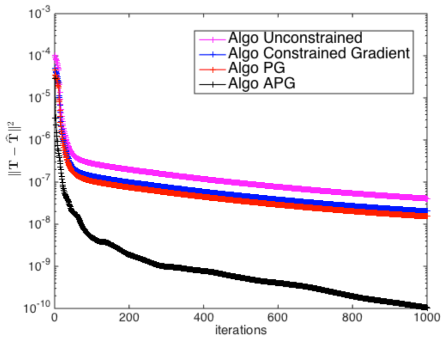


Figure 8: Reconstruction error (9) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 6$ and rank 3.

This convergence speed result is also clearly presented in Table 2, which indicates the CPU time required for each algorithm to reach an accuracy of 10^{-7}, and it should be noted that the proximal gradient algorithm is a slightly faster compared to the constrained gradient algorithm, while the accelerated proximal gradient algorithm remains the fastest to achieve an accuracy of 10^{-7}.

6. Conclusions

We have described two methods to overcome swamp and bottleneck problems, based on the proximal gradient (PG) and the accelerated proximal gra-

	SNR				
Algorithms	0	10	20	30	40
Unconstrained	1.85	1.66	1.19	0.95	0.62
Constrained Gradient	1.38	1.16	0.79	0.57	0.35
PG	1.37	1.15	0.77	0.52	0.36
APG	**1.35**	**1.12**	**0.74**	**0.47**	**0.30**

Table 2: CPU time (in seconds) versus SNR results, $\mu(\mathbf{A}) = 0.99$ and $\mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.6$ up to a precision of 10^{-7} and results with 100 random initializations at each SNR.

dient (APG), 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 proved the good behaviour of both algorithms in terms of accuracy and convergence speed, even in the presence of bad conditioning, compared to other iterative algorithms available in the literature.

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] G. H. Golub, Cf van loan, matrix computations, The Johns Hopkins.

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

[20] M. Rajih, P. Comon, R. A. Harshman, Enhanced line search: A novel method to accelerate parafac, SIAM journal on matrix analysis and applications 30 (3) (2008) 1128–1147.

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

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

[23] 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.

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

[25] 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.

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

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

[28] 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.

[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] 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`.
URL `http://link.aip.org/link/?SML/30/1128/1`

[32] 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`.

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

[40] 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.

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

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

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

[44] 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.

[45] L.-H. Lim, P. Comon, Blind multilinear identification, IEEE Transactions on Information Theory 60 (2) (2013) 1260–1280.

[46] M. H. Wright, Interior methods for constrained optimization, Acta numerica 1 (1992) 341–407.

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

[48] 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.

[49] L. Zhang, W. Zhou, D. Li, Global convergence of a modified fletcher–reeves conjugate gradient method with armijo-type line search, Numerische Mathematik 104 (4) (2006) 561–572.

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

[51] G. Tomasi, R. Bro, A comparison of algorithms for fitting the parafac model, Computational Statistics & Data Analysis 50 (7) (2006) 1700–1734.