# Computation of the nonnegative canonical tensor decomposition with two accelerated proximal gradient algorithms

Marouane Nazih, Khalid Minaoui, Elaheh Sobhani, Pierre Comon

# Monotone and non-monotone accelerated proximal gradient for nonnegative canonical polyadic tensor decomposition

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**

Tensors may be seen as multi-dimensional arrays that generalize vectors and matrices to more than two dimensions. Among tensor decompositions, we are especially interested in the Canonical Polyadic tensor decomposition, which is important in various real-world applications, for its uniqueness and ease of interpretation of its factor matrices.

In this research, we consider the estimation of factor matrices of the Nonnegative Canonical Polyadic (NCP) decomposition in a simultaneous way. Two proximal algorithms are proposed, the Monotone Accelerated Proximal Gradient (M-APG) and the Non-monotone Accelerated Proximal Gradient (Nm-APG) algorithms. These algorithms are implemented through a regularization function that incorporates previous iterations while using a monitoring capable of efficiently conducting this incorporation. Simulation results demonstrate better performance of the two proposed algorithms in terms of accuracy for the normal situation as well as for the bottleneck case when compared to other NCP algorithms in the literature.

*Keywords:* Canonical Polyadic Decomposition, Tensor, Non-convex Optimization, Accelerated Proximal gradient, Nonnegative Parafac.

*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 a large variety of applications, it is necessary to deal with quantities with multiple indices. These quantities are often expressed as tensors. Generally, a tensor is treated 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. In this present article, we focus mainly on tensors of order higher than two, since they possess properties which are not enjoyed by 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] or Candecomp [4]. As pointed out in [5],[6], the acronym "CP" decomposition can conveniently stands 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 [7, 8], Telecommunications [9, 10, 11], and also in other recent fields, such as data science, machine learning [12] and big data [13, 14]. The most remarkable characteristic of the CP decomposition is its essential uniqueness for orders strictly higher than two [15, 16], which enables parameter identification.

There are many algorithms for calculating CP decomposition. The most popular in the literature is the Alternating Least Squares (ALS) originally proposed in [4], which is an iterative optimization process that estimates the factor matrices in an alternating way. The ALS algorithm is designed to converge towards a local minimum under mild conditions [17]. However, the ALS algorithm is very sensitive to initialisation in some cases, and may suffer from

bottleneck or swamp phenomena [18, 19, 20] where the error between two consecutive iterations does not decrease, and results in a very low convergence rate. Various versions of the ALS algorithm [21, 10, 22, 23] have been proposed in the literature to speed up its convergence. These versions improved the ALS algorithm speed, but were still failing to increase the very low convergence rate caused by bottleneck and swamp phenomena. Recent works [11, 19, 24] have demonstrated that the introduction of appropriate constraints would avoid these issues. The proposition in [24] is a direct modification of the ALS, based on the Dykstra projection algorithm on all correlation matrices. In proposals [11, 19], the estimation of the factor matrices is performed in a simultaneous way under a coherence constraint on these factor matrices. Such methods have proven to be efficient and stable for calculating the CP decomposition in normal and also in difficult cases, especially when the factor matrices in one or all modes are almost collinear *i.e.*, bottleneck or swamp problems arise.

In the current study, we propose two algorithms, namely, the Monotone Accelerated Proximal Gradient (M-APG) and the Non-monotone Accelerated Proximal Gradient (Nm-APG) to improve the accuracy of Nonnegative Canonical Polyadic (NCP) decomposition. These algorithms are based on proximal methods [25] where we introduce a regularization function that penalizes the difference between the current and previous factor iterates, by using two different strategies capable of efficiently monitoring this regularization. We shall be particularly interested in the Accelerated Proximal Gradient (APG) algorithm, as it satisfies the assumptions of our NCP decomposition formulation problem.

The rest of the paper is organized as follows. Some notations and definitions are presented in Section 2. In Section 3 we describe properties of the NCP decomposition as well as some general definitions about proximal mapping. More details are also elaborated, where APG is explained for both convex and non-convex cases. In Section 5 we introduce our optimization algorithms for the NCP decomposition. In Section 6, we report computer results, and finally Section 7 concludes the paper.

## 2. 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 $T_{ijk}$. Operator $\otimes$ represents outer product of vectors, $\langle\ \rangle$ represents inner product, and symbols $[\![\ ]\!]$ are used to represent tensor decompositions.

*Definition 1.* 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 [26].

To simplify writing, we will use the term 'tensor' in a restricted sense, *i.e.*, as a three-dimensional array of real numbers (*i.e.*, $N = 3$). However, the generalization to $N^{th}$ order tensors, $N \geq 3$, is straightforward.

*Definition 2.* The outer product of two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^J$ defines a matrix $\mathbf{M} \in \mathbb{R}^{I \times J}$

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

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

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

In (1), the entry $(i, j, k)$ of tensor $\mathcal{T}$ is defined by the product

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

A tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ is said to be rank-1 (also referred to as a decomposable tensor [26]) 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 3.* The scalar product between two tensors with the same size, $\mathcal{X}$, $\mathcal{Y}$ $\in \mathbb{R}^{I\times J\times K}$, is defined as:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} X_{i,j,k}Y_{i,j,k}.$$

*Definition 4.* The Frobenius norm $\|.\|_F$ of a tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ is derived from the scalar tensor product:

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

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:

$$\|\mathcal{X} - \mathcal{Y}\|_F^2. \tag{3}$$

*Definition 5.* Let $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ be a tensor, then $vec\{\mathcal{T}\} \in \mathbb{R}^{IJK\times 1}$ represents the column vector defined by :

$$\big[vec\{\mathcal{T}\}\big]_{i+(j-1)I+(k-1)IJ} = T_{ijk}. \tag{4}$$

## 3. CP DECOMPOSITION

Let us consider a third order tensor $\mathcal{X}$ of size $I\times J\times K$, its CP decomposition is defined as follows:

$$\begin{aligned} \mathcal{X} &= \sum_{r=1}^{R} \lambda_r (\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r) \\ &= [\![\mathbf{A}, \mathbf{B}, \mathbf{C}; \boldsymbol{\lambda}]\!] \end{aligned} \tag{5}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_R]$ is a vector containing the scaling factors $\lambda_r$ and the three matrices $\mathbf{A}$= $[\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_R] \in \mathbb{R}^{I\times R}$, $\mathbf{B}$= $[\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_R] \in \mathbb{R}^{J\times R}$ and $\mathbf{C}$= $[\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_R] \in \mathbb{R}^{K\times R}$ are referred to as "factor matrices". When $R$ is minimal, then it is called the rank of $\mathcal{X}$, and we refer to the above expression as the CP Decomposition of $\mathcal{X}$ [1].

### 3.1. Low-rank approximation

Although the tensor of interest is of low-rank, it is frequently necessary to look for a low-rank approximation (*e.g.* rank $R$) of the observed tensor due to the presence of noise. In noisy case, the observed tensor is indeed generally of *generic rank*, strictly larger than $R$ [26]. In the low-rank approximation problem [9], the goal is to minimize an objective function $\Upsilon$ of the following form:

$$\begin{aligned} \Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}; \boldsymbol{\lambda}) &= \left\| \mathcal{X} - \widehat{\mathcal{X}} \right\|_F^2 \\ &= \left\| \mathcal{X} - \sum_{r=1}^{R} \lambda_r \left( \mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r \right) \right\|_F^2 . \end{aligned} \tag{6}$$

Alternatively, the minimization of (6) can be expressed using vectorization property (4) as:

$$\min_{\widehat{x}} \Upsilon(\widehat{\mathbf{x}}) = \min_{\widehat{\mathbf{x}}} \left\| \mathbf{x} - \widehat{\mathbf{x}} \right\|_F^2 , \tag{7}$$

where factor matrices are contained in a single vector $\widehat{\mathbf{x}} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$ and $\mathbf{x} = vec\{\mathcal{X}\}$.

### 3.2. Conditioning of the problem

One of the most practical properties of tensors of order greater than two ($N > 2$), lies in the uniqueness of their CP decomposition, contrary to matrix decompositions [3, 27, 10] (the decomposition of a matrix into a sum of rank-1 matrices also exists, but it is not unique, unless some strong constraints are imposed, such as orthogonality or nonnegativity).

From the definition of CP decomposition, it is obvious that the decomposition (5) is insensitive to:

- Permutation of the rank-1 terms (due to commutativity of addition), which refers to the permutation indeterminacy.
- Scaling of vectors $a_r$, $b_r$ and $c_r$, provided the product of the scaling factors is equal to 1, which corresponds to the scaling indeterminacy, inherent in tensor representations [26].

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 $\boldsymbol{\lambda}$ [27]. Another approach described in [9] and used in this study involves the calculation of the optimal value of the scaling factor $\boldsymbol{\lambda}$ to correctly control the conditioning of the problem. For that purpose, and for given matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, the optimal value $\boldsymbol{\lambda}_o$ minimizing the error $\Upsilon$ is determined by cancelling the gradient of (6) w.r.t. $\boldsymbol{\lambda}$, which then results in the following linear system:

$$\mathbf{G}\,\boldsymbol{\lambda}_o = \mathbf{s}, \tag{8}$$

where $\mathbf{G}$ is the Gram matrix of size $R \times R$ defined by:

$$G_{pq} = \mathbf{a}_p^T \mathbf{a}_q \times \mathbf{b}_p^T \mathbf{b}_q \times \mathbf{c}_p^T \mathbf{c}_q,$$

and $\mathbf{s}$ is the $R$-dimensional vector defined by:

$$\mathbf{s}_r = \Sigma_{ijk} T_{ijk} A_{ir} B_{jr} C_{kr}.$$

### *3.3. Nonnegative CP decomposition*

In most cases, the tensor components are nonnegative, so in order to extract these significant nonnegative components, the incorporation of a nonnegative constraint into the decomposition model is necessary. This gives rise to the Nonnegative Canonical Polyadic (NCP) decomposition, which is one of the most important tensor decomposition models with constraints [28, 29, 30, 31, 32]. This decomposition has received a great success in several real-world applications, such as the decomposition of hyperspectral data [28], the decomposition of electroencephalography (EEG) data [29], the decomposition of fluorescence excitation-emission matrix (EEM) data [30, 33], and also the decomposition of neural data [31], as well as many other multi-channel tensor data [34, 35].

Formally, given a nonnegative third order tensor $\mathcal{X}$ of size $I \times J \times K$, the NCP decomposition consists in solving the following minimization problem:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C};\boldsymbol{\lambda}} \quad & \|\mathcal{X} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}; \boldsymbol{\lambda}]\!]\|_F^2 \\ \mathbf{s.t.} \quad & \mathbf{A} \geq 0, \mathbf{B} \geq 0, \mathbf{C} \geq 0. \end{aligned} \tag{9}$$

where $\geq$ is understood entry-wise.

According to (9), the most natural way to perform this minimization is via the alternating approach. This approach consists of computing factor matrices by updating each of them individually while keeping the others fixed. In such a way, the problem to be solved is then transformed into three simple convex subproblems. Such an approach is the basis of the most commonly used methods to solve the NCP decomposition.

In our present work, we have performed the estimation of factor matrices in a simultaneous way (referred to as all-at-once way). Note that one of the major consequences of this approach is that the problem returns to a non-convex problem [36] as we can clearly observe in (7). To the best of our knowledge, proximal algorithms have not yet been provided in the literature to simultaneously estimate factor matrices of the NCP. This all-at-once computation was considered for the CP decomposition in [19] but under a coherence constraint. One such approach has also been used under the nonnegative constraint in [32] for which the minimization was performed using the Enhanced Line Search (ELS) and the alternate backtracking with ELS for the gradient and quasi-Newton algorithms. In addition the latter explicitly incorporates the nonnegative constraint of the factor matrices in the problem parameterization, contrary to our proposal in which we simply impose non-negative constraint by projecting all the entries onto the non-negative orthant.

## 4. PROXIMAL ALGORITHMS

### *4.1. Proximal operators*

Proximal operators are currently powerful and reliable optimization tools [37, 38], 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 $h$, the proximal operator (or proximal mapping) [37] maps an input point $\mathbf{x}$ to the minimizer of $h$ restricted to small proximity to $\mathbf{x}$. The

definition of the proximal operator is recalled hereafter.

*4.1.1. Definition*

The proximal map of a point $\mathbf{x} \in \mathbb{R}^N$ under a proper and closed function $h$ (with parameter $\theta > 0$):

$$\mathbf{prox}_{\theta h}(\mathbf{x}) = \underset{y \in \mathbb{R}^N}{\text{minimize}}\ h(\mathbf{y}) + \frac{1}{2\theta}\|\mathbf{x} - \mathbf{y}\|_2^2, \tag{10}$$

The operator $\mathbf{prox}_{\theta h} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ thus defined is the proximal operator of $h$, and the parameter $\theta$ controls the stretch to which the proximal operator maps points towards the minimum of $h$. Larger values of $\theta$ associated with mapped points near the minimum, whereas its smaller values giving a smaller movement towards the minimum [37].

As a result, we can see that the evaluation of a proximal operator can be a valuable sub-step in an optimization algorithm. As a general example, Let us consider the following optimization problem:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\textbf{minimize}}\ \ f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}), \tag{11}$$

where $g : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ and $h : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ are closed proper convex with $g$ is differentiable and $h$ is not necessarily differentiable but "simple" in the sense that its proximal operator can be evaluated efficiently, *i.e.*, its proximal operator admits a closed form [37]. A natural strategy of this method is to first reduce the value of $g$ by using iterative optimization methods such as gradient descent or Newton's method following a descent direction $\mathbf{d}_k$, then reduce the value of $h$ by applying the proximal operator to $h$, (using the same step-size) and repeat these two steps until convergence to a minimizer. This strategy yields the following iteration:

$$\mathbf{x}_{k+1} = \mathbf{prox}_{\theta_k h}(\mathbf{x}_k + \theta_k \mathbf{d}_k) \tag{12}$$

The mentioned strategy describes the general idea of the iterative proximal gradient method, while its accelerated version proposed by Beck and Teboulle in [39] involves an extrapolation at each iteration, by taking into account information from previous and current iterations.

First, we report the Accelerated Proximal Gradient (APG) method of Beck and Teboulle [39] in the convex case. This method consists of the following steps:

$$\begin{aligned} \mathbf{y}_k &= \mathbf{x}_k + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \\ \mathbf{x}_{k+1} &= \mathbf{prox}_{\theta_k h}(\mathbf{y}_k - \theta_k \nabla g(\mathbf{y}_k)), \\ t_{k+1} &= \frac{\sqrt{4(t_k)^2+1}+1}{2}. \end{aligned} \tag{13}$$

According to these steps, there is no guarantee that $f(\mathbf{x}_{k+1})$ will be inferior to $f(\mathbf{x}_k)$, this is due to the fact that APG is not a monotone algorithm. For that reason, Beck and Teboulle [39] have proposed a monotone APG consisting of the following steps:

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{14a}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\theta_k h}(\mathbf{y}_k - \theta_k \nabla g(\mathbf{y}_k)), \tag{14b}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2}, \tag{14c}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } f(\mathbf{z}_{k+1}) \leq f(\mathbf{x}_k) \\ \mathbf{x}_k & \text{otherwise.} \end{cases} \tag{14d}$$

*4.2. APG in the non-convex case*

In the recent research, the accelerated proximal gradient has been extended to solve general non-convex problems. Among these works, one can find those in [40, 41, 42], where a monotone descent of the objective value is imposed to ensure convergence, while on the other hand, the works in [43, 25] introduce a generic method for non-smooth non-convex problems based on Kurdyka Lojasiewicz theory.

In our present paper, we exploit the rigorous argument provided in [40] proving that the limit point of the sequence generated by the APG algorithm is a critical point of the objective function (6).

We present two APG-type algorithms for general non-convex problems [40], namely the monotone APG and the non-monotone APG, which we then adapt to our particular NCP decomposition problem.

#### 4.2.1. Monotone APG

The APG of Beck and Teboulle [39] is not guaranteed to converge for the non-convex case due to different reasons, one of them concerns the bad extrapolation of $\mathbf{y}_k$, and another one results from the fact that the sufficient descent condition is not ensured while it is essential to ensure the convergence to a critical point, however only the descent property, $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k)$, is guaranteed in (14).

To overcome these two difficulties, the use of an additional proximal gradient step of $\mathbf{x}_k$ as a monitor would be an appropriate choice due to its ability to ensure the required sufficient descent property [43] which is essential to guarantee convergence to a critical point and also to correct the bad extrapolation $\mathbf{y}_k$. Basically, the monotone APG algorithm consists of the following steps:

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{15a}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\theta_y h}(\mathbf{y}_k - \theta_k \nabla g(\mathbf{y}_k)), \tag{15b}$$

$$\mathbf{v}_{k+1} = \mathbf{prox}_{\theta_x h}(\mathbf{x}_k - \theta_k \nabla g(\mathbf{x}_k)), \tag{15c}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}, \tag{15d}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } f(\mathbf{z}_{k+1}) \leq f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{otherwise,} \end{cases} \tag{15e}$$

where $\theta_y$ and $\theta_x$ can be fixed constants satisfying $\theta_y < \frac{1}{L}$ and $\theta_x < \frac{1}{L}$ , or determined by backtracking line search method [44]. $L$ is the Lipschitz constant of $\nabla g$.

We can notice that the APG algorithm of Beck and Teboulle in the convex case ensures only the descent property for which $f(\mathbf{z}_{k+1})$ is compared to $f(\mathbf{x}_k)$ in (14), while its extension for the non-convex case ensures the sufficient descent property when $f(\mathbf{z}_{k+1})$ is compared to $f(\mathbf{v}_{k+1})$ (15), which means that:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \delta \|\mathbf{v}_{k+1} - \mathbf{x}_k\|^2, \tag{16}$$

where $\delta > 0$ is a small constant.

*4.2.2. Non-monotone APG*

In the preceding monotone APG algorithm, it is required to compute $\mathbf{v}_{k+1}$ at each iteration to control and correct $\mathbf{z}_{k+1}$, thus requiring more computation cost and CPU time. Moreover, we can directly accept $\mathbf{z}_{k+1}$ as $\mathbf{x}_{k+1}$ if it meets a particular criterion which indicates that $\mathbf{y}_k$ is a good extrapolation. Only afterwards, $\mathbf{v}_{k+1}$ is calculated when this criterion is not satisfied.

In contrast to the monotone APG, where (16) is guaranteed, in the case of non-monotone APG, $f(\mathbf{x}_{k+1})$ is allowed to be larger than $f(\mathbf{x}_k)$. More precisely, $\mathbf{x}_{k+1}$ is expected to yield an objective function value less than a relaxation of $f(\mathbf{x}_k)$, while not being too far from $f(\mathbf{x}_k)$. An appropriate way to define a relaxation parameter $c_k$ and still not to be too far from $f(\mathbf{x}_k)$ is to consider the average of $f(\mathbf{x}_1)$,...,$f(\mathbf{x}_{k-1})$,$f(\mathbf{x}_k)$ with exponentially decreasing weights as in [45]:

$$c_k = \frac{\Sigma_{j=1}^{k} \nu^{k-j} f(\mathbf{x}_j)}{\Sigma_{j=1}^{k} \nu^{k-j}}, \tag{17}$$

where $\nu \in [0, 1)$ controls the level of non-monotonicity. Practically, $c_k$ can be computed by the following efficient recursion:

$$\begin{aligned} q_{k+1} &= \nu q_k + 1, \\ c_{k+1} &= \frac{\nu q_k c_k + f(\mathbf{x}_{k+1})}{q_{k+1}}, \end{aligned} \tag{18}$$

where $q_1 = 1$ and $c_1 = f(\mathbf{x}_1)$.

Therefore, in the non-monotone APG, (16) gives rise to these two conditions by the different choices of $\mathbf{x}_{k+1}$:

$$f(\mathbf{z}_{k+1}) \leq c_k - \delta \|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2, \tag{19a}$$

$$f(\mathbf{v}_{k+1}) \leq c_k - \delta \|\mathbf{v}_{k+1} - \mathbf{x}_k\|^2, \tag{19b}$$

Condition (19a) is adopted as the criterion mentioned before. Note that when (19a) is verified, this means that $\mathbf{y}_k$ is a good extrapolation, which leads to a direct acceptance of $\mathbf{z}_{k+1}$ without computing $\mathbf{v}_{k+1}$. Otherwise, when (19a)

does not hold, which means that $\mathbf{y}_k$ is not an appropriate extrapolation. In this case, we are required to correct this bad extrapolation $\mathbf{z}_{k+1}$ by calculating the additional $\mathbf{v}_{k+1}$ using (15c) which satisfies (19b). In addition, by using the backtracking method, the $\mathbf{v}_{k+1}$ satisfying (19b) can be found in a finite number of steps.

The basic structure of the non-monotone APG algorithm is as follows:

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{20a}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\theta_y h}(\mathbf{y}_k - \theta_k \nabla g(\mathbf{y}_k)), \tag{20b}$$

$$\textbf{if} \;\; f(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2 \;\; \textbf{then} \tag{20c}$$

$$\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$$

$$\textbf{else}$$

$$\mathbf{v}_{k+1} = \mathbf{prox}_{\theta_x h}(\mathbf{x}_k - \theta_k \nabla g(\mathbf{x}_k)), \tag{20d}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } f(\mathbf{z}_{k+1}) \leq f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{otherwise} \end{cases} \tag{20e}$$

$$\textbf{end if}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}. \tag{20f}$$

## 5. PROPOSED METHOD

### *5.1. Proposed minimization*

In contrast to alternating approaches, where factor matrices are computed in turn, our approach addresses the Nonnegative CP decomposition problem through a variational approach in which all factor matrices are estimated simultaneously. In other words, we are looking for a solution to a minimization problem in which the function to be minimized is composed of two terms: The first one related to the properties of the noise, called the 'data fidelity term' which is defined in (6) by the cost function $\Upsilon$. The second one related to a priori information on model parameters, called 'regularization', which penalizes

the difference of a particular factor in two successive iterations, and which will be represented by $\mathcal{G}$. In [46], several numerical examples show that this regularization can help the algorithm to keep distance from the degenerate cases of the bottleneck and the swamps, *i.e.*, the degenerate regions, where convergence is slow. In addition, it has also been shown in [17] that the limit point obtained from the regularized NCP decomposition (22) is a critical point of the original minimization problem $\left\|\mathcal{X} - \widehat{\mathcal{X}}\right\|_F^2$.

As indicated in (7) and for the sake of simplicity, columns of factor matrices are contained in a single vector $\mathbf{x} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$, thus the objective to be minimized then has the following form:

$$\mathcal{F}(\mathbf{x}) = \underbrace{\Upsilon(\mathbf{x})}_{Fidelity} + \underbrace{\mathcal{G}(\mathbf{x})}_{Regularization}, \tag{21}$$

Which can be explicitly written as:

$$\mathcal{F}(\mathbf{x}) = \min_{\widehat{\mathbf{x}}_k \geq 0} \|\mathbf{x} - \widehat{\mathbf{x}}_k\|_F^2 + \eta_n \|\tilde{\mathbf{x}}_{k-1} - \widehat{\mathbf{x}}_k\|_F^2, \tag{22}$$

where $\tilde{\mathbf{x}}_{k-1}$ is the predecessor version of $\mathbf{x}_k$ in the previous iteration and $\eta_n$ is a penalty weight that controls the sharpness of the penalty, which decreases through iterations. In this paper, we propose two APG-type algorithms as solution to our optimization problem defined in (22) by exploiting the convergence analysis presented in [40].

The fundamental parts of our proposed methods will be described in the following paragraphs and will be clearly abstracted in **Algorithm 1** and **Algorithm 2**.

There are essentially two fundamental steps:

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

### *5.2. Gradient step*

This step improves the approximate solution, focusing only on the data fidelity with the exclusion of the penalty function. Two other steps are also

involved in these stages:

**(i)** First, we calculate the direction of the descent direction $\mathbf{d}^{(k)}$ of $\Upsilon$, leading to the direction of the steepest decrease, determined as follows:

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) = -\nabla\Upsilon(\mathbf{x}^{(k)}), \tag{23}$$

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

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A \tag{24}$$

with

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

The gradients expressions w.r.t $\mathbf{B}$ and $\mathbf{C}$ are similar.

**(ii)** The second stage involves the determination of the step-size $\rho^{(k)}$ according to the chosen direction $\mathbf{d}^{(k)}$. Among numerous methods of searching for a good step-size, *backtracking* is extensively used [44]. It depends on two parameters $\alpha$ and $\beta$, 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 [47] is verified:

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}. \tag{25}$$

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

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{26}$$

*5.3. Proximal step*

Since the preceding step concerns only the data fidelity term $\Upsilon$, the *proximal step* is expected to readjust the general search direction based on the penalty function $\mathcal{G}$. For this purpose, we apply the proximal algorithm to the previous point arising from the preceding step of the gradient, *i.e.* $\mathbf{z}^{(k)}$, as follows:

$$\begin{aligned}\mathbf{z}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{z}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{27}$$

This step indicates that $\mathbf{prox}_{\mathcal{G}}(\mathbf{z}^{(k)})$ is a point that compromises between minimizing $\mathcal{G}$ and being near to $\mathbf{z}^{(k)}$.

Now it remains to calculate the exact proximal operator of $\mathcal{G}$. In order to do that, the gradient of the function $\mathcal{H}$ is set to zero which gives us the closed form of the proximal operator for our regularized function $\mathcal{G}$.

*Gradient of $\mathcal{H}$.* The cancellation of the gradient of $\mathcal{H}$ yields :

$$\begin{aligned} \nabla\mathcal{H}(\mathbf{x}) = 0 &\implies \nabla\mathcal{G}(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0 \\ &\implies -2 * \eta_n(\tilde{\mathbf{x}}_{k-1} - \mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0, \end{aligned} \tag{28}$$

then, with a simple calculation, the analytical form of the proximal for our regularized function $\mathcal{G}$ would be of the following closed form:

$$\mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) = \frac{\frac{1}{\rho^{(k)}}\mathbf{z}^{(k)} + 2 * \eta_n\tilde{\mathbf{x}}_{k-1}}{\frac{1}{\rho^{(k)}} + 2 * \eta_n}. \tag{29}$$

*Projecting into nonnegative.* As the approaches of [48, 49, 50], the nonnegativity constraint is enforced by a simple projection onto the nonnegative orthant:

$$\mathbf{z}_{k+1} = \mathbf{max}(0, \mathbf{z}_{k+1})$$

*Extrapolation.* The APG algorithm initially extrapolates the point $\mathbf{x}_k$ by a combination of the current point $\mathbf{x}_k$ and the previous point $\mathbf{x}_{k-1}$ as:

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

With $t_0 = 0$, $t_1 = 1$ and according to the following update rule:

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}$$

As we have stated in the previous section, one of the main difficulties in the non-convex case lies in the bad extrapolation of $\mathbf{y}_k$, this prompts the use of an additional step of the proximal gradient of $\mathbf{x}_k$ as a monitor due to its ability to ensure the required sufficient descent property. The latter being essential to guarantee the convergence to a critical point and also to correct the bad

extrapolation of $\mathbf{y}_k$. This method refers to the Monotone Accelerated Proximal Gradient (M-APG) algorithm as summarized in **Algorithm 1**.

Note that computing $\mathbf{v}_{k+1}$ at each iteration to control and correct $\mathbf{z}_{k+1}$ is computationally expensive and costly in terms of CPU time. In order to provide a more efficient and less costly algorithm, we propose a second method in which we can directly accept $\mathbf{z}_{k+1}$ as $\mathbf{x}_{k+1}$, if it meets the following particular criterion:

$$\mathcal{F}(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2, \tag{30}$$

which indicates that $\mathbf{y}_k$ is a good extrapolation. Only afterwards, $\mathbf{v}_{k+1}$ is calculated when (30) is not satisfied. This second method is referred to as to the Non-monotone Accelerated Proximal Gradient (Nm-APG) algorithm and is summarized in **Algorithm 2**.

## 6. RESULTS AND DISCUSSIONS

In this section, we illustrate some experimental results for analysing the performance of the proposed algorithms with the most popular and widely applied methods for nonnegative CP decomposition in the literature.

A comparison is performed with conventional Multiplicative Updating (MU) [51], Hierarchical Alternative Least Squares (HALS) [48], Nonnegative Alternative Least Squares with Frobenius norm regularization based on block principal pivoting (ANLS-BPP) [49] and alternating Proximal Gradient (PG alternating) [50] algorithms [1].

In addition, we measure the performance of each algorithm based on three criteria, namely accuracy, CPU time and the best sum congruence, which is a meaningful criterion to compare two tensors of rank $R > 1$ [11]. As a matter of form, the best sum congruence involves finding the best permutation $\sigma$ among

[1] We would like to thank Deqing Wang, who sent us the MATLAB codes of the methods [51, 48, 49, 50]. Therefore, the author's original code was used to obtain the results presented in all figures in this section.

**Algorithm 1:** Monotone Accelerated Proximal Gradient (M-APG) to minimize (22)

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ by matrices with unit-norm columns, $\eta_0$, $t_0 = 0$, $t_1 = 1$ ;

Calculate the optimal scaling factor $\boldsymbol{\lambda}_0^*$ by solving (8): $\mathbf{G}_0\boldsymbol{\lambda}_0^* = \mathbf{s}_0$;

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

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

$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2}$

1. **Gradient Step**

   (a) Compute the descent direction $\mathbf{d}_{\mathbf{y}}^{(k)}$ w.r.t. $\mathbf{y}_k$: $\mathbf{d}_{\mathbf{y}}^{(k)} = -\nabla\Upsilon(\mathbf{y}_k)$

   (b) Compute the descent direction $\mathbf{d}_{\mathbf{x}}^{(k)}$ w.r.t. $\mathbf{x}_k$: $\mathbf{d}_{\mathbf{x}}^{(k)} = -\nabla\Upsilon(\mathbf{x}_k)$

   (c) Calculate step sizes $\rho_{\mathbf{y}}^{(k)}$ and $\rho_{\mathbf{x}}^{(k)}$ using the backtracking method:

$$\Upsilon(\mathbf{y}_k + \rho_{\mathbf{y}}^{(k)}\mathbf{d}_{\mathbf{y}}^{(k)}) < \Upsilon(\mathbf{x}_k) + \alpha\rho_{\mathbf{y}}^{(k)}\nabla\Upsilon(\mathbf{x}_k)^T\mathbf{d}_{\mathbf{y}}^{(k)}$$
$$\Upsilon(\mathbf{x}_k + \rho_{\mathbf{x}}^{(k)}\mathbf{d}_{\mathbf{x}}^{(k)}) < \Upsilon(\mathbf{x}_k) + \alpha\rho_{\mathbf{x}}^{(k)}\nabla\Upsilon(\mathbf{x}_k)^T\mathbf{d}_{\mathbf{x}}^{(k)}$$

   (d) Update : $\mathbf{z}_k = \mathbf{y}_k + \rho_{\mathbf{y}}^{(k)}\mathbf{d}_{\mathbf{y}}^{(k)}$ and $\mathbf{v}_k = \mathbf{x}_k + \rho_{\mathbf{x}}^{(k)}\mathbf{d}_{\mathbf{x}}^{(k)}$

2. **Proximal Step**

   (a) Compute the proximal operator of $\mathcal{G}$ at $\mathbf{z}_k$ and $\mathbf{v}_k$ using (29) such as: $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_{\mathbf{y}}^{(k)}\mathcal{G}}(\mathbf{z}_k)$ and $\mathbf{v}_{k+1} = \mathbf{prox}_{\rho_{\mathbf{x}}^{(k)}\mathcal{G}}(\mathbf{v}_k)$

   (b) Projecting

   $\mathbf{z}_{k+1} = \mathbf{max}(0, \mathbf{z}_{k+1})$ and $\mathbf{v}_{k+1} = \mathbf{max}(0, \mathbf{v}_{k+1})$

   (c) Monitoring

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

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 $\boldsymbol{\lambda}_{k+1}^*$ using (8) such as: $\mathbf{G}_{k+1}\boldsymbol{\lambda}_{k+1}^* = \mathbf{s}_{k+1}$

**end for**

**Algorithm 2:** Non-monotone Accelerated Proximal Gradient (Nm-APG) to minimize (22)

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$, $\eta_0$, $t_0 = 0$, $t_1 = 1$, $\delta = \nu = 0.2$, $q_1 = 1$, $c_1 = \mathcal{F}(\mathbf{x}_1)$;

Calculate the optimal scaling factor $\boldsymbol{\lambda}_0^*$ by solving (8): $\mathbf{G}_0\boldsymbol{\lambda}_0^* = \mathbf{s}_0$ ;

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

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

$t_{k+1} = \frac{\sqrt{4(t_k)^2+1}+1}{2}$

1. **Gradient Step**

   (a) Compute the descent direction $\mathbf{d}_{\mathbf{y}}^{(k)}$ w.r.t. $\mathbf{y}_k$: $\mathbf{d}_{\mathbf{y}}^{(k)} = -\nabla\Upsilon(\mathbf{y}_k)$

   (b) Calculate step sizes $\rho_{\mathbf{y}}^{(k)}$ and using the backtracking method such: $\Upsilon(\mathbf{y}_k + \rho_{\mathbf{y}}^{(k)}\mathbf{d}_{\mathbf{y}}^{(k)}) < \Upsilon(\mathbf{x}_k) + \alpha\rho_{\mathbf{y}}^{(k)}\nabla\Upsilon(\mathbf{x}_k)^T\mathbf{d}_{\mathbf{y}}^{(k)}$

   (c) Update : $\mathbf{z}_k = \mathbf{y}_k + \rho_{\mathbf{y}}^{(k)}\mathbf{d}_{\mathbf{y}}^{(k)}$

2. **Proximal Step**

   (a) Compute the proximal operator of $\mathcal{G}$ at $\mathbf{z}_k$ using (29) such as: $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_{\mathbf{y}}^{(k)}\mathcal{G}}(\mathbf{z}_k)$

   (b) Projecting

   $\mathbf{z}_{k+1} = \mathbf{max}(0, \mathbf{z}_{k+1})$

   (c) Monitoring

   **if** $\mathcal{F}(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2$ **then** $\mathbf{x}_{k+1} = \mathbf{z}_{k+1}$

   **else** Compute the proximal operator of $\mathcal{G}$ at $\mathbf{v}_k$ using (29) such as:

   $\mathbf{v}_{k+1} = \mathbf{prox}_{\rho_{\mathbf{x}}^{(k)}\mathcal{G}}(\mathbf{x}_k + \rho_{\mathbf{x}}^{(k)}\mathbf{d}_{\mathbf{x}}^{(k)}) = \mathbf{prox}_{\rho_{\mathbf{x}}^{(k)}\mathcal{G}}(\mathbf{v}_k)$

   $\mathbf{v}_{k+1} = \mathbf{max}(0, \mathbf{v}_{k+1})$

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

3. $q_{k+1} = \nu q_k + 1$, and $c_{k+1} = \frac{\nu q_k c_k + f(\mathbf{x}_{k+1})}{q_{k+1}}$,

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

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



6. Calculation of the optimal scaling factor $\boldsymbol{\lambda}_{k+1}^*$ using (8) such as: $\mathbf{G}_{k+1}\boldsymbol{\lambda}_{k+1}^* = \mathbf{s}_{k+1}$

**end for**

the columns of factor matrices by maximizing:

$$\max_{\sigma} \sum_{r=1}^{R} \frac{|a_r^H \widehat{a}_{\sigma(r)}|}{\|a_r\| \|\widehat{a}_{\sigma(r)}\|} \frac{|b_r^H \widehat{b}_{\sigma(r)}|}{\|b_r\| \|\widehat{b}_{\sigma(r)}\|} \frac{|c_r^H \widehat{c}_{\sigma(r)}|}{\|c_r\| \|\widehat{c}_{\sigma(r)}\|} \tag{31}$$

In order to obtain comparable results, all algorithms are initialized with the same initial points which are generated with positive uniformly distributed random numbers. Also, in order to clearly visualize each algorithm's behaviour, we consider only the maximum number of iterations, which is fixed to 600.

In all experiments, The computations are run in Matlab on a computer with Intel i5 CPU (2.7GHz) and 8GB memory running 64bit Mac OS. In addition, the results are obtained from 50 Monte Carlo runs for all experiments.

### *6.1. Experiment 1*

In this experience, we generate a random NCP model of size $3 \times 4 \times 6$ with rank 3 with and with coherences [2] $0.4 \leq \mu(\mathbf{A}) \leq 0.6$, $0.4 \leq \mu(\mathbf{B}) \leq 0.6$ and $0.4 \leq \mu(\mathbf{C}) \leq 0.6$. $\eta$ is varied through iterations. More precisely in this first experiment, $\eta$ is initialized to 1, and is divided by 100 when $\Upsilon(x)$ is reduced by less than $10^{-4}$.

Figure 1 reports the reconstruction error (6) as a function of the number of iterations. It can hence be observed that the Non-monotone Accelerated Proximal Gradient (Nm-APG) and Monotone Accelerated Proximal Gradient (M-APG) algorithms are more accurate than other algorithms in terms of number of iterations, followed by alternating proximal gradient (PG alternating) algorithm, then by ANLS-BPP algorithm, and finally MU and HALS algorithms.

A further examination in Table 1 also reveals that the Nm-APG algorithm, the M-APG algorithm and the PG-alternating algorithm yield 99% of correct estimations whereas the ANLS-BPP algorithm yields 97% and finally MU and HALS algorithms produce 95%. From this result, we can clearly see that the two proposed algorithms are more efficient in terms of accuracy.

[2] The coherence of a matrix $\mathbf{A}$ is expressed as the the maximum absolute value of the cross-correlations between the columns of $\mathbf{A}$ as : $\mu(\mathbf{A}) = \max_{i \neq j} |\mathbf{a}_i^H \mathbf{a}_j|$.

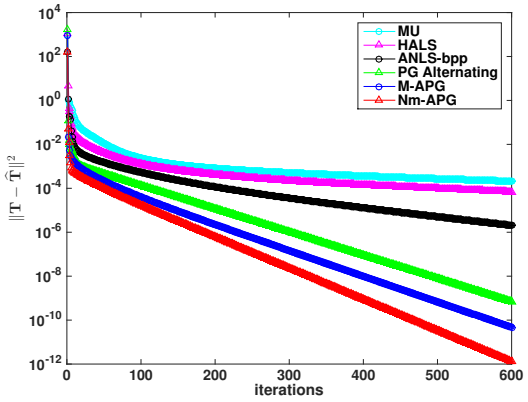


Figure 1: Reconstruction error as a function of the number of iterations. For the experiment 1.

In order to show a deeper comparison between the algorithms, we explore the CPU time criterion to highlight the run time of each algorithm. Table 1 indicates that the M-APG algorithm requires more machine time compared to the other algorithms, this drawback comes at the expense of accuracy and is expected due to the high complexity of the monitoring used in the M-APG algorithm which demands the computation of additional proximal at each iteration. On the other hand, one can clearly understand the time reduction in its non-monotone version which avoids the computation of the additional proximal at each iteration.

Table 1: Performances of the NCP algorithms for experiment 1.

| Algorithms | AbsErr | Time | Congruence |
|---|---|---|---|
| MU | 2.1240e-04 | 2.66 | 0.95 |
| HALS | 6.6760e-05 | 2.60 | 0.95 |
| ANLS-bpp | 2.0701e-06 | 4.30 | 0.97 |
| PG alternating | 6.8699e-10 | 3.77 | 0.99 |
| M-APG | 4.6690e-11 | 7.04 | 0.99 |
| Nm-APG | 1.2813e-12 | 5.25 | 0.99 |

*6.2. Experiment 2*

In this second experience, we generate a random CP model of size $4 \times 3 \times 10$ with rank 4 and with coherences $\mu(\mathbf{A}) = 0.9$, $\mu(\mathbf{B}) = 0.6$ and $\mu(\mathbf{C}) = 0.6$. This configuration addresses the problem of bottleneck, where we chose factors of the first mode (*i.e.*, $\mathbf{A}$) to be almost collinear. In this experiment, $\eta$ is initialized to 1.5, and is divided by 10 when $\Upsilon(x)$ is reduced by less than $10^{-2}$.

Figure 2 illustrates the reconstruction error as a function of the number of iterations. From these results, we can still observe that both MU and HALS algorithms produce mediocre results compared to other algorithms. On the other hand, The proposed Nm-APG algorithm has also superior results over other algorithms in the bottleneck case.

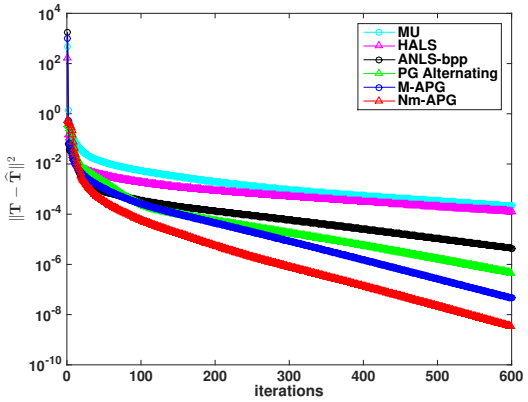


Figure 2: Reconstruction error as a function of the number of iterations. For the experiment 2.

Table 2 also reports the accuracy of all proximal algorithms by remarking that Nm-APG, M-APG and PG-alternating algorithms provide 99% of correct estimations with a precision higher than $10^{-6}$.

By analysing the execution time of each algorithm in Table 2, we can indeed notice that although the Nm-APG algorithm requires less machine time than the M-APG algorithm, the two proposed algorithms still require a little more machine time than the other algorithms. However, it should be noted that our

simultaneous way of estimating factor matrices has provided better accuracy to the detriment of the machine time. It is also important to point out that in many applications of nonnegative CP decomposition, accuracy is more important than execution time; this is the case for the decomposition of hyperspectral images or electroencephalography data.

In the light of these results, we can conclude that proximal algorithms seem to be appropriate choices for nonnegative CP decomposition problem. Moreover, our choice of estimating factor matrices in a simultaneous way proves its efficiency in terms of accuracy compared to the alternating way. Nevertheless, the alternating way remains a good choice when execution time becomes a priority for some other applications, such as multichannel tensor data where the estimation of multicode CDMA needs to be processed in real time.

Table 2: Performances of the NCP algorithms for experiment 2.

| Algorithms | AbsErr | Time | Congruence |
|---|---|---|---|
| MU | 2.1417e-04 | 2.67 | 0.95 |
| HALS | 1.2933e-04 | 2.55 | 0.95 |
| ANLS-bpp | 4.4461e-06 | 4.91 | 0.96 |
| PG alternating | 4.6119e-07 | 3.71 | 0.99 |
| M-APG | 4.5559e-08 | 8.11 | 0.99 |
| Nm-APG | 3.4007e-09 | 6.26 | 0.99 |

## 7. Conclusions

We have opted to estimate factor matrices of the Nonnegative Canonical Polyadic (NCP) decomposition in a simultaneous way, the proposals are based on both Monotone Accelerated Proximal Gradient (M-APG) and Non-monotone Accelerated Proximal Gradient (Nm-APG), with a simple and comprehensive monitoring strategy capable of computing the minimum NCP decomposition of

three-way arrays. We performed a thorough comparison with other NCP algorithms in the literature based on computational experiments, which have shown the very high performance of the two proposed algorithms in terms of accuracy for the normal situation as well as for the bottleneck case when compared to other NCP algorithms in the literature. Although the proposed algorithms have high performance, at the same time they are computationally more expensive in terms of machine time.

## References


[1] P. Comon, Tensor decompositions, state 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] 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.

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

[6] 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.

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

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

[9] 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.

[10] 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.

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

[12] A. Shashua, T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, in: Proceedings of the 22nd international conference on Machine learning, 2005, pp. 792–799.

[13] Q. Zhang, L. T. Yang, Z. Chen, P. Li, An improved deep computation model based on canonical polyadic decomposition, IEEE Transactions on Systems, Man, and Cybernetics: Systems 48 (10) (2017) 1657–1666.

[14] Q. Zhang, L. T. Yang, Z. Chen, P. Li, High-order possibilistic c-means algorithms based on tensor decompositions for big data in iot, Information Fusion 39 (2018) 72–80.

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

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

[19] 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.

[20] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, K. Huang, Block-randomized stochastic proximal gradient for low-rank tensor factorization, IEEE Transactions on Signal Processing 68 (2020) 2170–2185.

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

[22] 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.

[23] 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.

[24] 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.

[25] Q. Li, Y. Zhou, Y. Liang, P. K. Varshney, Convergence analysis of proximal gradient with momentum for nonconvex optimization, in: International Conference on Machine Learning, PMLR, 2017, pp. 2111–2119.

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

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

[28] M. A. Veganzones, J. E. Cohen, R. C. Farias, J. Chanussot, P. Comon, Nonnegative tensor CP decomposition of hyperspectral data, IEEE Transactions on Geoscience and Remote Sensing 54 (5) (2015) 2577–2588.

[29] D. Wang, Y. Zhu, T. Ristaniemi, F. Cong, Extracting multi-mode ERP features using fifth-order nonnegative tensor decomposition, Journal of neuroscience methods 308 (2018) 240–247.

[30] N. J. Rodriguez-Fernandez, F. Aires, P. Richaume, Y. H. Kerr, C. Prigent, J. Kolassa, F. Cabot, C. Jimenez, A. Mahmoodi, M. Drusch, Soil moisture retrieval using neural networks: Application to SMOS, IEEE Transactions on Geoscience and Remote Sensing 53 (11) (2015) 5991–6007.

[31] A. H. Williams, T. H. Kim, F. Wang, S. Vyas, S. I. Ryu, K. V. Shenoy, M. Schnitzer, T. G. Kolda, S. Ganguli, Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis, Neuron 98 (6) (2018) 1099–1115.

[32] J.-P. Royer, N. Thirion-Moreau, P. Comon, Computing the polyadic decomposition of nonnegative third order tensors, Signal Processing 91 (9) (2011) 2159–2171.

[33] X. Vu, C. Chaux, N. Thirion-Moreau, S. Maire, E. M. Carstea, A new penalized nonnegative third-order tensor decomposition using a block coordinate proximal gradient approach: Application to 3d fluorescence spectroscopy, Journal of Chemometrics 31 (4) (2017) e2859.

[34] E. Acar, B. Yener, Unsupervised multiway data analysis: A literature survey, IEEE transactions on knowledge and data engineering 21 (1) (2008) 6–20.

[35] M. Mørup, Applications of tensor (multiway array) factorizations and decompositions in data mining, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1 (1) (2011) 24–40.

[36] X. Fu, N. Vervliet, L. De Lathauwer, K. Huang, N. Gillis, Computing large-scale matrix and tensor decomposition with structured factors: A unified nonconvex optimization perspective, IEEE Signal Processing Magazine 37 (5) (2020) 78–94.

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

[38] 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.

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

[40] H. Li, Z. Lin, Accelerated proximal gradient methods for nonconvex programming, Advances in neural information processing systems 28 (2015) 379–387.

[41] 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.

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

[43] 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.

[44] 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.

[45] H. Zhang, W. W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, SIAM journal on Optimization 14 (4) (2004) 1043–1056.

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

[47] 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.

[48] A. Cichocki, R. Zdunek, A. H. Phan, S.-i. Amari, Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation, John Wiley & Sons, 2009.

[49] H. Kim, H. Park, Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method, SIAM journal on matrix analysis and applications 30 (2) (2008) 713–730.

[50] D. Wang, F. Cong, T. Ristaniemi, Higher-order nonnegative candecomp/parafac tensor decomposition using proximal algorithm, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 3457–3461.

[51] N. Gillis, F. Glineur, Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization, Neural computation 24 (4) (2012) 1085–1105.