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

Marouane Nazih, Khalid Minaoui, Elaheh Sobhani, Pierre Comon

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

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

Multidimensional signal analysis has become an important part of many signal processing problems. This type of analysis allows to take advantage of different diversities of a signal in order to extract useful information. This paper focuses on the design and development of multidimensional data decomposition algorithms called Canonical Polyadic (CP) tensor decomposition, a powerful tool in a variety of real-world applications due to its uniqueness and ease of interpretation of its factor matrices. More precisely, it is desired to compute simultaneously the factor matrices involved in the CP decomposition of a real nonnegative tensor, under nonnegative constraints. For this purpose, 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 via a regularization function with a simple control strategy capable of efficiently taking advantage of previous iterations. Simulation results demonstrate better performance of the two proposed algorithms in terms of accuracy when compared to other nonnegative CP algorithms in the literature.

Keywords: Canonical Polyadic Decomposition, Tensor, Non-convex

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

Optimization, Accelerated Proximal gradient, Nonnegative Parafac.

1. INTRODUCTION

Tensors have interested mathematicians and physicists since the 19th century. In physics, they offer a practical language for expressing some natural laws independently of the coordinate system. A famous example is the one established by Einstein's theory of general relativity, whose fundamental equations are expressed in terms of tensors. In our framework, the motivation to use tensors is different, as elaborated below.

In our context, a tensor of order N just represents a multidimensional array where each element is accessible via N indices. Thus, a first-order tensor is a vector, a second-order tensor is a matrix, and a zero-order tensor is a scalar. The analysis of tensors of order greater than two is a matter of multilinear algebra.

Since the sixties, a growing interest in tensors was observed in many scientific and engineering communities. On can mention the early works in psychometrics that applied tensor techniques for data analysis purposes [1],[2], and later, the works on blind source separation that exploited the tensor structure of higher order cumulants [3], while numerous works using tensor models were emerging in a wide variety of applications ranging from component analysis in chemistry [4] to estimation and location of radiating sources [5]. Today, the growing list of tensor applications includes problems in computer vision [6], biomedical engineering [7], modeling and identification of dynamical systems [8], big data [9], and data mining [10]. This renewed interest in tensor models is mainly due to their ability to exploit an additional structure of the problem compared to traditional matrix models. A typical example of this superiority is the estimation of excitation/emission spectra from fluorescence data in chemometrics using high-order tensor decomposition techniques [4]. In fact, the main advantage of tensor models in these applications is the ability to uniquely identify the parameters that characterize them, under much weaker assumptions than matrix models.

In this paper, we will mainly focus on the so-called *Canonical Polyadic* (CP)

decomposition [11]. One of the most remarkable features of this decomposition is its essential uniqueness for orders strictly greater than two [12, 13], which enables parameter identification. The CP decomposition has been used in various fields, such as Chemometrics [14, 15], Telecommunications [16, 17, 18], and also in other recent fields, such as big data [19, 20] as well as in many machine learning tasks, including regression analysis with Tensor Regression (TR) [21], supervised classification with Support Tensor Machine (STM) [22, 23] instead of the popular Support Vector Machine (SVM), data pre-processing with tensor dictionary learning (also known as "sparse coding") [24, 25] and unsupervised classification with High-order restricted Boltzmann machines [26].

Many algorithms have been developed to calculate the CP decomposition. The most popular in the literature is the Alternating Least Squares (ALS) originally proposed in [2], in which each factor matrix is updated alternatingly as a subproblem. The ALS algorithm is well designed to converge to a local minimum under mild conditions [27]. However, the ALS algorithm remains inappropriate to compute the CP decomposition under positivity constraints, and also in some situations where factors matrices in one or all modes are collinear, i.e., bottleneck or swamp phenomena [28, 29]. When such phenomena occur, the error between two consecutive iterations does not significantly decrease, leading to a very low convergence rate. However, the bottleneck phenomenon is less often encountered when decomposing nonnegative tensors, because the problem is well-posed [30, 31].

Different variants of the ALS algorithm [17, 32, 33] have been specifically developed in the literature to compute the Nonnegative Canonical Polyadic (NCP) decomposition. These include, for example, the Hierarchical Alternative Least Squares (HALS) method, which was designed for large-scale tensor data [34, 35]. The Alternating Non-negative Least Squares method (ANLS) which is a powerful sub-processing for NCP, taking advantage of the efficiency of many Non-Negative Least Squares methods (NNLS) such as the Active Set (AS) [36] and the Block Principal Pivot (BPP) [37]. Nevertheless, NNLS often suffer from rank deficiency due to the sparse effect induced by the projection

onto the nonnegative orthant, yielding zero components in factor matrices [38]. Recently, some methods involving proximal methods [39, 40] have been shown to be effective for such problems, including the alternating proximal gradient method [41, 42, 43] which has gained popularity for NMF and third-order tensor decomposition due to its stable convergence.

In this paper, 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 [44], 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 in the non-convex case. Indeed, unlike the alternating approach where factor matrices are computed alternately, leading to simple convex problems, our approach (sometimes referred to as *all-at-once*) consists in computing all factor matrices simultaneously, which then leads to a continuous non-convex optimization 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. Some general definitions on proximal mapping and further details are elaborated in Section 4, where APG is explained for 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

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, and $\langle \cdot, \cdot \rangle$ represents the Euclidean inner product.

Definition 1. Tensors are mathematical objects derived from multilinear algebra that generalize scalars and vectors. For our purposes, a tensor of order N will merely refer to a multidimensional array in which each element is accessible via N indices $\{i_1, \ldots, i_N\}$, $1 \leq i_n \leq I_n$, for all n, $1 \leq n \leq N$. Such a tensor is a table of size $I_1 \times \cdots \times I_N$, which we denote:

$$\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}. \tag{1}$$

The entries of tensor $\mathcal{T}$ are denoted by $T_{i_1,\ldots,i_N}$. The nth dimension of tensor $\mathcal{T}$, I_n, is sometimes referred to as the nth mode in the literature.

Definition 2. 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 tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$:

$$\mathcal{T} = \mathbf{a} \otimes \mathbf{b} \otimes \mathbf{c}. \tag{2}$$

The entry (i, j, k) of the simple tensor defined above in (2) is 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 [3]) if each of its elements can be represented as : $T_{ijk} = a_i b_j c_k$; in other words, any tensor expressed as the outer product of three vectors, which will be denoted in a compact form as in (2), is rank-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:

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

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:

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

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

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

Definition 5. The vectorization operation $vec\{.\}$ maps a tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$ to a vector $vec\{\mathcal{T}\}$ of size IJK, defined by:

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

3. CP DECOMPOSITION

The CP decomposition of a 3rd-order tensor $\mathcal{X}$ of size $I \times J \times K$ is defined as follows:

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r (\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r) \tag{6}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_R]$ is a vector containing real positive scaling factors λ_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 $\mathcal{X}$ [45, 3].

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 cases, the observed tensor is indeed generally of *generic rank*, strictly larger than R [3]. In the low-rank approximation problem [16], the goal is to minimize an objective function Υ of the following form:

$$\Upsilon(\mathbf{A}, \mathbf{B}, \mathbf{C}; \boldsymbol{\lambda}) = \left\| \mathcal{X} - \widehat{\mathcal{X}} \right\|_F^2 \tag{7}$$

where $\widehat{X} = \sum_{r=1}^{R} \lambda_r \, (\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r)$. Alternatively, the minimization of (7) can be expressed using vectorization property (5) as:

$$\min_{\hat{x}} \Upsilon(\hat{x}) = \min_{\hat{x}} \|\mathbf{x} - \hat{\mathbf{x}}\|_F^2 \tag{8}$$

with $\hat{\mathbf{x}} = \sum_r \lambda_r \mathbf{a}_r \boxtimes \mathbf{b}_r \boxtimes \mathbf{c}_r$, where $\boxtimes$ represents the Kronecker product [46].

It is worth noting that scaling indeterminacies are present in the expressions of $\hat{\mathcal{X}}$ or $\hat{\mathbf{x}}$; this fact is well known, see *e.g* [3] and references therein. Yet, in numerical optimization, it is desirable to fix these indeterminacies. This is the reason why it is often chosen to impose vectors $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r\}$ to have a unit norm (regardless of the norm chosen): λ_r are then unique, and vectors $\{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r\}$ are unique up to sign flip when the CP decomposition is unique.

Next, the optimal value $\boldsymbol{\lambda}_o$ minimizing the error Υ can be determined by cancelling the gradient of (7) w.r.t. $\boldsymbol{\lambda}$, which then results in the following linear system:

$$\mathbf{G}\,\boldsymbol{\lambda}_o = \mathbf{s}, \tag{9}$$

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:

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

3.2. Nonnegative CP decomposition

From now on, let us limit ourselves to the case where the components of the tensor are non-negative. 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 [47, 48, 49, 50, 51]. This decomposition has received a great success in several real-world applications, such as the decomposition of hyperspectral data [47], the decomposition of electroencephalography (EEG)

data [48], the decomposition of fluorescence excitation-emission matrix (EEM) data [49, 52], and also the decomposition of neural data [50], as well as many other multi-channel tensor data [53, 54].

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:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{C};\boldsymbol{\lambda}} \quad & \left\| \mathcal{X} - \sum_{r=1}^{R} \lambda_r \left(\mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r \right) \right\|_F^2 \\ \textbf{s.t.} \quad & \mathbf{a}_r \geq 0, \mathbf{b}_r \geq 0, \mathbf{c}_r \geq 0, \ r = 1, ..., R. \end{aligned} \tag{10}$$

where $\geq$ is understood entry-wise.

According to (10), the most natural way to perform this minimization is via an alternating approach. Such an approach consists of updating one factor matrix at a time while keeping the others fixed. In such a way, the problem to be solved is then transformed into three simple convex sub-problems. This is the basis of the most commonly used methods to solve the NCP decomposition.

In the present work, we compute all factor matrices in a simultaneous way. Note that one of the major consequences of this approach is that the problem turns into a non-convex one [55] as we can clearly observe in (8). 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 [29, 56] but under a coherence constraint. One such approach has also been used under the nonnegative constraint in [51] for which the minimization was performed using the Enhanced Line Search (ELS) and the alternating 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 impose the non-negative constraint to all entries by resorting to proximal algorithms.

4. PROXIMAL ALGORITHMS

4.1. Proximal operators

Proximal operators are currently powerful and reliable optimization tools [39, 40], 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) [39] 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 $\rho > 0$) is defined as:

$$\mathbf{prox}_{\rho h}(\mathbf{x}) = \underset{y \in \mathbb{R}^N}{\text{minimize}}\ h(\mathbf{y}) + \frac{1}{2\rho}\|\mathbf{x} - \mathbf{y}\|_2^2, \tag{11}$$

The operator $\mathbf{prox}_{\rho h} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is the proximal operator of h, and parameter ρ controls the extent to which the proximal operator maps points towards the minimum of h, with larger values of ρ associated with mapped points near the minimum, and smaller values giving a smaller movement towards the minimum [39, p.125].

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:

$$\underset{\mathbf{x} \in \mathbb{R}^n}{\textbf{minimize}}\ \ f(\mathbf{x}) = g(\mathbf{x}) + h(\mathbf{x}), \tag{12}$$

where $g : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ and $h : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ are closed proper convex and g is differentiable where h is not necessarily differentiable but rather "simple" in the sense that its proximal operator can be evaluated efficiently, *i.e.*, its proximal operator admits a closed form [39]. 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:

$$\mathbf{x}_{k+1} = \mathbf{prox}_{\rho_k h}(\mathbf{x}_k + \rho_k \mathbf{d}_k) \tag{13}$$

The mentioned strategy describes the general idea of the iterative proximal gradient method, while its accelerated version proposed by Beck and Teboulle in [57] 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 [57] in the convex case. This method consists of the following steps:

$$\begin{aligned} \mathbf{y}_k &= \mathbf{x}_k + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \\ \mathbf{x}_{k+1} &= \mathbf{prox}_{\rho_k h}(\mathbf{y}_k - \rho_k \nabla g(\mathbf{y}_k)), \\ t_{k+1} &= \frac{\sqrt{4(t_k)^2+1}+1}{2}. \end{aligned} \tag{14}$$

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 [57] have proposed a monotone APG consisting of the following steps:

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

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

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

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

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

[58, 59, 60], where a monotone descent of the objective value is imposed to ensure convergence, while on the other hand, the works in [61, 44] introduce a generic method for non-smooth non-convex problems based on Kurdyka-Lojasiewicz theory.

In the present paper, we exploit the rigorous argument provided in [58] proving that the limit point of the sequence generated by the APG algorithm is a critical point of the objective function (7).

We present two APG-type algorithms for general non-convex problems [58], 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 [57] is not guaranteed to converge in 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 (15).

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 [61], 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:

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1}-1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{16a}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_y h}(\mathbf{y}_k - \rho_y \nabla g(\mathbf{y}_k)), \tag{16b}$$

$$\mathbf{v}_{k+1} = \mathbf{prox}_{\rho_x h}(\mathbf{x}_k - \rho_x \nabla g(\mathbf{x}_k)), \tag{16c}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}, \tag{16d}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } f(\mathbf{z}_{k+1}) \leq f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{otherwise,} \end{cases} \tag{16e}$$

where ρ_y and ρ_x can be either fixed constants satisfying $\rho_y < \frac{1}{L}$ and $\rho_x < \frac{1}{L}$, where L is the Lipschitz constant of ∇g, or determined by backtracking line search method [62].

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 (15), while its extension to the non-convex case ensures the sufficient descent property when $f(\mathbf{z}_{k+1})$ is compared to $f(\mathbf{v}_{k+1})$ (16), which means that:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - \delta\|\mathbf{v}_{k+1} - \mathbf{x}_k\|^2, \tag{17}$$

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 $\mathbf{z}_{k+1}$, which involves larger computational cost and CPU time. Moreover, we can directly accept $\mathbf{z}_{k+1}$ as $\mathbf{x}_{k+1}$ if it meets a particular criterion indicating that $\mathbf{y}_k$ is a good extrapolation. Only afterwards, $\mathbf{v}_{k+1}$ is calculated whenever this criterion is not met.

In contrast to the monotone APG, where (17) 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 so as to remain in the vicinity of $f(\mathbf{x}_k)$ 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 [63]:

$$c_k = \frac{\Sigma_{j=1}^{k} \nu^{k-j} f(\mathbf{x}_j)}{\Sigma_{j=1}^{k} \nu^{k-j}}, \tag{18}$$

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

$$\begin{aligned} q_{k+1} &= \nu q_k + 1, \\ c_{k+1} &= \frac{\nu q_k c_k + f(\mathbf{x}_{k+1})}{q_{k+1}}, \end{aligned} \tag{19}$$

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

Therefore, in the non-monotone APG, (17) gives rise to these two conditions by the different choices of $\mathbf{x}_{k+1}$:

$$f(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2, \tag{20a}$$

$$f(\mathbf{v}_{k+1}) \leq c_k - \delta\|\mathbf{v}_{k+1} - \mathbf{x}_k\|^2, \tag{20b}$$

Condition (20a) is adopted following the criterion mentioned before. More precisely, when (20a) 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 (20a) does not hold, which means that $\mathbf{y}_k$ is not an appropriate extrapolation. In that case, we are required to correct this bad extrapolation $\mathbf{z}_{k+1}$ by calculating the additional variable $\mathbf{v}_{k+1}$ using (16c), which satisfies (20b). In addition, by using the backtracking method, variables $\mathbf{v}_{k+1}$ satisfying (20b) can be found in a finite number of steps.

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

$$\mathbf{y}_k = \mathbf{x}_k + \frac{t_{k-1}}{t_k}(\mathbf{z}_k - \mathbf{x}_k) + \frac{t_{k-1} - 1}{t_k}(\mathbf{x}_k - \mathbf{x}_{k-1}), \tag{21a}$$

$$\mathbf{z}_{k+1} = \mathbf{prox}_{\rho_y h}(\mathbf{y}_k - \rho_y \nabla g(\mathbf{y}_k)), \tag{21b}$$

$$\textbf{if } \ f(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2 \ \textbf{ then} \tag{21c}$$

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

$$\textbf{else}$$

$$\mathbf{v}_{k+1} = \mathbf{prox}_{\rho_x h}(\mathbf{x}_k - \rho_x \nabla g(\mathbf{x}_k)), \tag{21d}$$

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1} & \text{if } f(\mathbf{z}_{k+1}) \leq f(\mathbf{v}_{k+1}) \\ \mathbf{v}_{k+1} & \text{otherwise} \end{cases} \tag{21e}$$

$$\textbf{end if}$$

$$t_{k+1} = \frac{\sqrt{4(t_k)^2 + 1} + 1}{2}. \tag{21f}$$

5. PROPOSED METHOD

5.1. Proposed minimization

In contrast to the alternating approach, where factor matrices are computed in turn, our approach addresses the Nonnegative CP decomposition problem by estimating all factor matrices 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 (7) by the cost function Υ. 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 [64], several numerical examples show that this regularization can also help the algorithm to stay away from degenerate cases of bottlenecks or swamps, *i.e.*, from regions where convergence is slow. In addition, it has also been shown in [27] that the limit point obtained from the regularized NCP decomposition (23) is a critical point of the original minimization problem $\left\|\mathcal{X}-\widehat{\mathcal{X}}\right\|_F^2$.

As indicated in (8) and for the sake of simplicity, columns of factor matrices are arranged in a single vector $\mathbf{x} = vec\{[\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]\}$, so that the objective to be minimized takes the form:

$$\mathcal{F}(\mathbf{x}) = \underbrace{\Upsilon(\mathbf{x})}_{Fidelity} + \underbrace{\mathcal{G}(\mathbf{x})}_{Regularization} , \tag{22}$$

which can be explicitly written as:

$$\mathcal{F}(\mathbf{x}) = \min_{\widehat{\mathbf{x}}_k \geq 0} \|\mathbf{x} - \widehat{\mathbf{x}}_k\|_F^2 + \eta_k \|\tilde{\mathbf{x}}_{k-1} - \widehat{\mathbf{x}}_k\|_F^2 , \tag{23}$$

where $\tilde{\mathbf{x}}_{k-1}$ is the predecessor version of $\mathbf{x}_k$ in the previous iteration and η_k is a penalty weight that controls the sharpness of the penalty, which decreases through iterations. We propose two APG-type algorithms as a solution to the optimization problem defined in (23) by exploiting the convergence analysis presented in [58].

The core of the latter two algorithms is described in the following paragraphs, and labelled **Algorithm 1** and **Algorithm 2**, respectively.
There are essentially two fundamental steps:

- A *gradient step* associated with the data fidelity term (function Υ).
- A *proximal step* related to the penalty term (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 Υ, leading to the direction of the steepest decrease, determined by:

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{A}^{(k)}, \mathbf{B}^{(k)}, \mathbf{C}^{(k)}) = -\nabla\Upsilon(\mathbf{x}^{(k)}), \tag{24}$$

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

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A \tag{25}$$

with

$$M^A_{pq} \overset{def}{=} \sum_{jk} \lambda_p B_{jp} C_{kp} C^*_{kq} B^*_{jq} \lambda^*_q$$

$$N^A_{ip} \overset{def}{=} \sum_{jk} T_{ijk} B^*_{jp} C^*_{kp} \lambda^*_p$$

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 [62]. 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 [65] is verified:

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) + \alpha\rho^{(k)}\nabla\Upsilon(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}. \tag{26}$$

To conclude, the *gradient step* is executed:

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{27}$$

5.3. Proximal step

Since the preceding step concerns only the data fidelity term Υ, 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:

$$\begin{aligned}\mathbf{z}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) \\ &= \underset{\mathbf{x}}{\operatorname{argmin}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{z}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})}\end{aligned} \tag{28}$$

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

Now it remains to calculate the exact proximal operator of $\mathcal{G}$. In order to do that, the gradient of function $\mathcal{H}$ is set to zero, which yields 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 :

$$\begin{aligned}\nabla\mathcal{H}(\mathbf{x}) = 0 &\implies \nabla\mathcal{G}(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0 \\ &\implies -2\eta_k(\tilde{\mathbf{x}}_{k-1} - \mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0.\end{aligned} \tag{29}$$

Then, with a simple calculation, the analytical form of the proximal of $\mathcal{G}$ is obtained:

$$\mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) = \frac{\frac{1}{\rho^{(k)}}\mathbf{z}^{(k)} + 2\eta_k\tilde{\mathbf{x}}_{k-1}}{\frac{1}{\rho^{(k)}} + 2\eta_k}. \tag{30}$$

Projecting onto the nonnegative orthant. As the approaches of [34, 36, 66], 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 assuming the 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 is referred to as the Monotone Accelerated Proximal Gradient (M-APG) algorithm, and is 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:

$$\mathcal{F}(\mathbf{z}_{k+1}) \leq c_k - \delta\|\mathbf{z}_{k+1} - \mathbf{y}_k\|^2, \tag{31}$$

which indicates that $\mathbf{y}_k$ is a good extrapolation. Only afterwards, $\mathbf{v}_{k+1}$ is calculated when (31) is not satisfied. This second method is referred to as the Non-monotone Accelerated Proximal Gradient (Nm-APG) algorithm, and is summarized in **Algorithm 2**.

6. RESULTS AND DISCUSSIONS

In this section, we provide some experimental results illustrating the performance of the proposed algorithms, compared to the most popular methods dedicated to nonnegative CP decomposition in the literature.

Algorithm 1: Monotone Accelerated Proximal Gradient (M-APG) to minimize (23)

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ by matrices with unit-norm columns, η_0, $t_0 = 0$, $t_1 = 1$;

Calculate the optimal scaling factor $\boldsymbol{\lambda}_0^*$ by solving (9) : $\mathbf{G}_0\boldsymbol{\lambda}_0^* = \mathbf{s}_0$;

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 (30) 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 (9) 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 (23)

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$, η_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 (9) : $\mathbf{G}_0\boldsymbol{\lambda}_0^* = \mathbf{s}_0$;

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 (30) 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 (30) 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 (9) such as: $\mathbf{G}_{k+1}\boldsymbol{\lambda}_{k+1}^* = \mathbf{s}_{k+1}$

end for

A comparison is performed with conventional Multiplicative Updating (MU) [67], Hierarchical Alternative Least Squares (HALS) [34], Nonnegative Alternative Least Squares with Frobenius norm regularization based on block principal pivoting (ANLS-BPP) [36], alternating Proximal Gradient (PG alternating) [66] and gradient in a simultaneous way (Gradient all-at-once) [51] algorithms.

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 $R > 1$ [18]. More precisely, the best sum congruence involves finding the best permutation σ among the columns of factor matrices by maximizing:

$$\max_{\sigma} \sum_{r=1}^{R} \frac{|a_r^H \widehat{a}_{\sigma(r)}|}{\|a_r\| \|\widehat{a}_{\sigma(r)}\|} \frac{|b_r^H \widehat{b}_{\sigma(r)}|}{\|b_r\| \|\widehat{b}_{\sigma(r)}\|} \frac{|c_r^H \widehat{c}_{\sigma(r)}|}{\|c_r\| \|\widehat{c}_{\sigma(r)}\|} \tag{32}$$

In order to obtain comparable results, all algorithms are initialized with the same initial points, randomly generated uniformly in $[0, 1]$. Also, in order to clearly visualize the behavior of each algorithm, we execute no more than 600 iterations.

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

Figure 1a reports the reconstruction error (7) as a function of the number of iterations. It can hence be observed that the Non-monotone Accelerated

[1] 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 : $\mu(\mathbf{A}) = \max_{i \neq j} |\mathbf{a}_i^H \mathbf{a}_j|$.

Proximal Gradient (Nm-APG) and Monotone Accelerated Proximal Gradient (M-APG) algorithms are more accurate than other algorithms for a given number of iterations, followed by alternating proximal gradient (PG alternating) algorithm, then by Gradient all-at-once algorithm, and finally ANLS-BPP, MU and HALS algorithms.

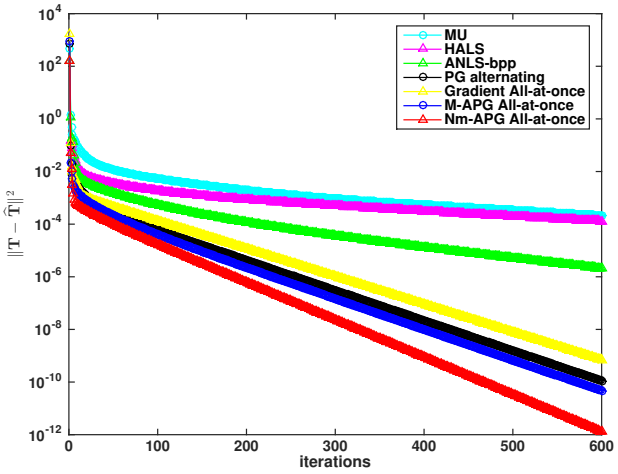
This conclusion is based on an equal number of iterations. However, no conclusion can be drawn on the speed of algorithms, since one iteration of each algorithm has not the same computational complexity; therefore a study of the CPU time of all algorithms is required. For this purpose, figure 1b reports some results for a given CPU time instead. At 2s for example, the Gradient, the PG alternating and the proposed Nm-APG algorithms produce the best result in terms of reconstruction error which is around 10^{-7} compared to 10^{-6} for the M-APG and ANLS-BPP algorithms. On the other hand, the MU and HALS algorithms still perform poorly compared to all other algorithms.

Another evaluation can be conducted using the congruence (32). An inspection of the results reported in table 1 also reveals that the Nm-APG algorithm, the M-APG algorithm, the Gradient 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%.

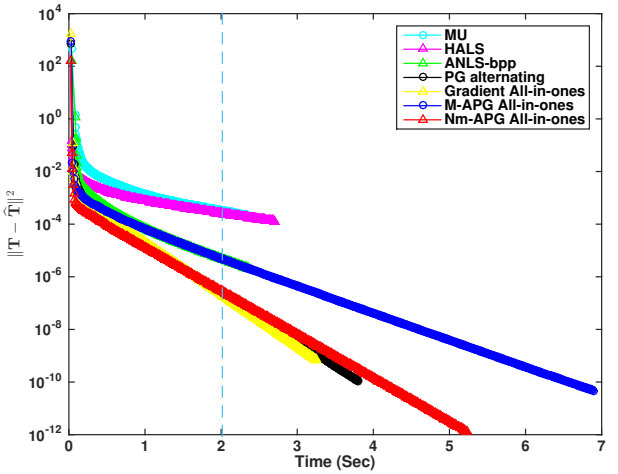
In order to show a deeper comparison between the two proposed algorithms, table 1 also indicates that the M-APG algorithm requires more CPU time compared to the Nm-APG algorithm, which is expected due to the high complexity of the monitoring used in the M-APG algorithm that demands the computation of 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.

6.2. Experiment 2

In this second experiment, 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 indicates that the first mode (*i.e.*, $\mathbf{A}$) is chosen to be almost collinear. In this



(a) Reconstruction error as a function of the number of iterations



(b) Reconstruction error as a function of the CPU time

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

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

Algorithms	AbsErr	Time	Congruence
MU	$2,13.10^{-04}$	2.36	0.95
HALS	$1,28.10^{-04}$	2.70	0.95
ANLS-bpp	$2,05.10^{-06}$	2.36	0.97
PG alternating	$6,86.10^{-10}$	3.79	0.99
Gradient-all-in-ones	$6,70.10^{-10}$	3.32	0.99
M-APG	$4,54.10^{-11}$	6.89	0.99
Nm-APG	$1,24.10^{-12}$	5.23	0.99

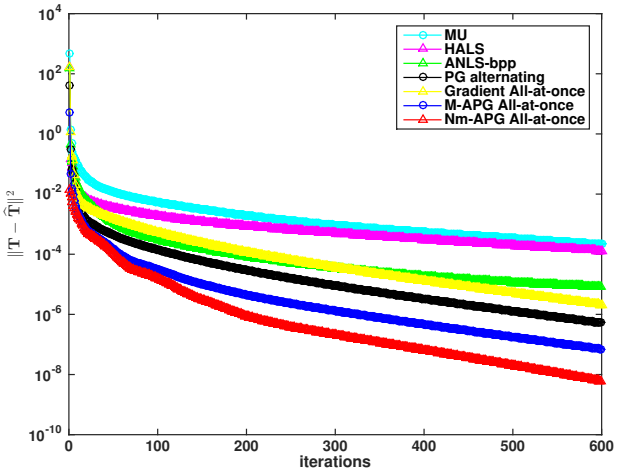
experiment, η is initialized to 1.5, and is divided by 10 when $\Upsilon(x)$ is reduced by less than 10^{-2}.

Figure 2a 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 produces better results than other algorithms, when dealing with the difficult situation of first mode collinearity.

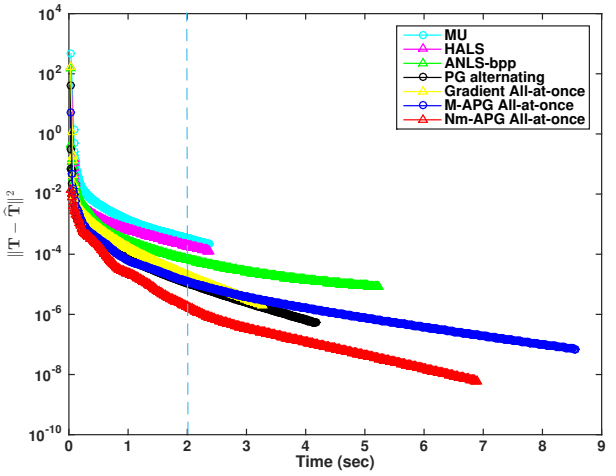
Moreover, figure 2b shows that in terms of CPU time and accuracy, the proposed Nm-APG algorithm still performs better than all other algorithms. Indeed, for a fixed time of 2s, the Nm-APG is around 10^{-6} compared to 10^{-5} for the Gradient, the PG alternating and the M-APG algorithms followed by ANLS-bpp which is about 10^{-4} then MU and HALS algorithms with low accuracy of around 10^{-3}.

The accuracy in terms of congruence criteria is also reported in table 2, which shows that Nm-APG, M-APG, Gradient and PG-alternating algorithms provide 99% of correct estimations whereas the ANLS-BPP algorithm yields 96% and finally MU and HALS algorithms that produce 95%.

In light of these results, it can be claimed that proximal algorithms are appropriate choices for the nonnegative CP decomposition problem. In addition,



(a) Reconstruction error as a function of the number of iterations



(b) Reconstruction error as a function of the CPU time

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

the choice of estimating factor matrices in a simultaneous way proves its efficiency in terms of accuracy compared to the alternating way. Finally, one may also point out that the proposed Non-monotone Accelerated Proximal Gradient algorithm represents a suitable choice, especially when the collinearity of the modes needs to be faced.

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

Algorithms	AbsErr	Time	Congruence
MU	$2,13.10^{-04}$	2.36	0.95
HALS	$1,28.10^{-04}$	2.43	0.95
ANLS-bpp	$8,15.10^{-06}$	5.23	0.96
PG alternating	$5,21.10^{-07}$	4.17	0.99
Gradient-all-in-ones	$2,05.10^{-06}$	3.26	0.99
M-APG	$7,03.10^{-08}$	8.56	0.99
Nm-APG	$5,95.10^{-09}$	6.86	0.99

7. Conclusions

We chose to estimate factor matrices of the Nonnegative Canonical Polyadic (NCP) decomposition in a simultaneous way. For this purpose, we have opted for proximal algorithms, 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 available in the literature based on computational experiments, which have shown the high performance of the proposed Nm-APG algorithm in terms of accuracy for the normal situation as well as for the difficult situation of first-mode collinearity.

References

[1] L. R. Tucker, Implications of factor analysis of three-way matrices for measurement of change, Problems in measuring change 15 (122-137) (1963) 3.

[2] 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.

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

[4] R. Bro, Multi-way analysis in the food industry-models, algorithms, and applications, in: MRI, EPG and EMA, Proc ICSLP 2000, Citeseer, 1998.

[5] F. Raimondi, P. Comon, O. Michel, S. Sahnoun, A. Helmstetter, Tensor decomposition exploiting diversity of propagation velocities: Application to localization of icequake events, Signal Processing 118 (2016) 75–88.

[6] 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.

[7] A. G. Mahyari, D. M. Zoltowski, E. M. Bernat, S. Aviyente, A tensor decomposition-based approach for detecting dynamic network states from eeg, IEEE Transactions on Biomedical Engineering 64 (1) (2016) 225–237.

[8] G. Favier, A. Y. Kibangou, T. Bouilloc, Nonlinear system modeling and identification using volterra-PARAFAC models, International Journal of Adaptive Control and Signal Processing 26 (1) (2012) 30–53.

[9] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, G. Min, A tensor-based approach for big data representation and dimensionality reduction, IEEE transactions on emerging topics in computing 2 (3) (2014) 280–291.

[10] T. G. Kolda, J. Sun, Scalable tensor decompositions for multi-aspect data mining, in: 2008 Eighth IEEE international conference on data mining, IEEE, 2008, pp. 363–372.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] H. Zhou, L. Li, H. Zhu, Tensor regression with applications in neuroimaging data analysis, Journal of the American Statistical Association 108 (502) (2013) 540–552.

[22] I. Kotsia, W. Guo, I. Patras, Higher rank support tensor machines for visual recognition, Pattern Recognition 45 (12) (2012) 4192–4203.

[23] K. Makantasis, A. D. Doulamis, N. D. Doulamis, A. Nikitakis, Tensor-based classification models for hyperspectral data analysis, IEEE Transactions on Geoscience and Remote Sensing 56 (12) (2018) 6884–6898.

[24] M. Ghassemi, Z. Shakeri, A. D. Sarwate, W. U. Bajwa, Stark: Structured dictionary learning through rank-one tensor recovery, in: 2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), IEEE, 2017, pp. 1–5.

[25] L. Albera, H. Becker, A. Karfoul, R. Gribonval, A. Kachenoura, S. Bensaid, L. Senhadji, A. Hernandez, I. Merlet, Localization of spatially distributed brain sources after a tensor-based preprocessing of interictal epileptic eeg data, in: 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), IEEE, 2015, pp. 6995–6998.

[26] T. D. Nguyen, T. Tran, D. Phung, S. Venkatesh, Tensor-variate restricted boltzmann machines, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[27] 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.

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

[29] 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 171 (2020) 107472.

[30] L.-H. Lim, P. Comon, Nonnegative approximations of nonnegative tensors, Journal of Chemometrics: A Journal of the Chemometrics Society 23 (7-8) (2009) 432–441.

[31] Y. Qi, P. Comon, L.-H. Lim, Semialgebraic geometry of nonnegative tensor rank, SIAM Journal on Matrix Analysis and Applications 37 (4) (2016) 1556–1580.

[32] 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.

[33] 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.

[34] 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.

[35] A. Cichocki, A.-H. Phan, Fast local algorithms for large scale nonnegative matrix and tensor factorizations, IEICE transactions on fundamentals of electronics, communications and computer sciences 92 (3) (2009) 708–721.

[36] 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.

[37] J. Kim, H. Park, Fast nonnegative matrix factorization: An active-set-like method and comparisons, SIAM Journal on Scientific Computing 33 (6) (2011) 3261–3281.

[38] M. Jouni, M. Dalla Mura, P. Comon, Some issues in computing the CP decomposition of nonnegative tensors, in: International Conference on Latent Variable Analysis and Signal Separation, Springer, 2018, pp. 57–66.

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

[40] 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.

[41] Y. Xu, W. Yin, A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion, SIAM Journal on imaging sciences 6 (3) (2013) 1758–1789.

[42] N. Guan, D. Tao, Z. Luo, B. Yuan, Nenmf: An optimal gradient method for nonnegative matrix factorization, IEEE Transactions on Signal Processing 60 (6) (2012) 2882–2898.

[43] Y. Xu, Alternating proximal gradient method for sparse nonnegative tucker decomposition, Mathematical Programming Computation 7 (1) (2015) 39–70.

[44] 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.

[45] P. Comon, Tensor decompositions, state of the art and applications, in: J. G. McWhirter, I. K. Proudler (Eds.), Mathematics in Signal Processing V, Clarendon Press, Oxford, UK, 2002, pp. 1–24, arxiv:0905.0454. Keynote address at IMA Conf., Dec 2000, Warwick, UK.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

[52] 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.

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

[54] 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.

[55] 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.

[56] M. Nazih, K. Minaoui, E. Sobhani, P. Comon, Computation of low-rank tensor approximation under existence constraint via a forward-backward algorithm, Signal Processing 188 (2021) 108178.

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

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

[59] 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.

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

[61] 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.

[62] 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.

[63] 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.

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

[65] 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.

[66] 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.

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