Computation of the regularized canonical polyadic decomposition of tensors using the accelerated proximal gradient with momentum

Marouane Nazih, Khalid Minaoui, Pierre Comon

Computation of the regularized canonical polyadic decomposition of tensors using the accelerated proximal gradient with momentum

Marouane Nazih[1], Khalid Minaoui[1], and Pierre Comon[2]

[1] LRIT Laboratory, associated unit to CNRST (URAC29), IT Rabat Center, Faculty of sciences in Rabat, Mohammed V University, Rabat, Morocco, marouane_nazih@um5.ac.ma,
[2] GIPSA-lab, Univ. Grenoble Alpes, CNRS, Grenoble, France.

Abstract. Tensors may be seen as multidimensional arrays that generalize vectors and matrices to more than two dimensions. Among tensor decompositions, we are especially interested in the Canonical Polyadic (CP) tensor decomposition, which is important in numerous real-world applications, for its uniqueness and ease of interpretation of its factor matrices. In this research, we propose a new Canonical Polyadic (CP) model based on the Accelerated Proximal Gradient (PG) algorithm and through the introduction of a regularization function that incorporates the previous iterations; using a new strategy capable of efficiently conducting this incorporation. Simulation results demonstrate the better performance of the proposed approach in terms of accuracy and rapidity compared to other algorithms in the literature, especially when the swamp phenomenon occurs.

Keywords: Canonical Polyadic Decomposition, Tensor, Non-convex Optimization, Accelerated Proximal gradient, Low-rank approximation.

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 multilinearity when changing the coordinate system [1]. For our purposes, it will be sufficient to see a tensor of order N as a multidimensional 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 stand for either "Canonical Polyadic" or "CanDecomp/Parafac", and we shall follow this terminology. The CP decomposition has been used in various fields, such as Chemometrics [7, 8], Telecommunications [9–11], and also in other newer fields, such as data science, machine learning [12] and big data [14, 13]. The most attractive 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 alternately through individual updates of each matrix while keeping the others fixed. Hence the system to be solved is then transformed to three simple least square (LS) sub-problems. The ALS algorithm is designed to converge towards a local minimum under mild conditions [33]. However, the ALS algorithm is very sensitive to initialisation in some cases, and may suffer from swamp phenomena [19], where the error between two consecutive iterations does not decrease, resulting in a very low convergence rate. Various versions of the ALS algorithm [20, 10, 21, 6] have been proposed in the literature to reduce the slow convergence of the ALS algorithm. These versions improved the ALS algorithm speed but were still unable to overcome difficult case of swamp problem. Recent works [11, 22, 23] have demonstrated that the introduction of appropriate constraints would avoid these issues. The proposition in [23] is a direct modification of the ALS ; Alternating way ; based on the Dykstra projection algorithm on all correlation matrices, while in proposals [11, 22], which consist in simultaneously estimating factor matrices ; All-in-ones way ; through 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 when estimated factors are close to collinear, i.e., swamp problem arises.

In the present work, we propose an algorithm to improve both the accuracy and the convergence speed of Canonical Polyadic (CP) decomposition especially in the difficult case of the swamp. This algorithm is based on proximal methods [24] and through the introduction of a regularization function that penalizes the difference between the current and previous factor iterates using a new strategy 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 CP decomposition formulation problem.

The rest of the paper is organized as follows. In the next section, we present some notations and definitions for tensors. In section 3 we describe properties of the CP decomposition as well as some general definitions about proximal mapping. In Section 4 we introduce our new optimization algorithm. In Section 5 we deal with analysis of the simulation results and finally Section 6 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}.

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

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

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

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

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

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

A tensor $\mathcal{T} \in \mathbb{C}^{I \times J \times K}$ is said to be rank-1 (also referred to as a decomposable tensor [25]) if each of its elements can be represented as : $T_{ijk} = a_i b_j c_k$, in other words, if it can be expressed as the outer product of three vectors, which will be denoted in a compact form as in (1).

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

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

where* stands for the complex conjugation.

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

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

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

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

Definition 4 Let $\mathcal{T} \in \mathbb{C}^{I \times J \times K}$ be a tensor, then $vec\{\mathcal{T}\} \in \mathbb{C}^{IJK \times 1}$ represents the column vector defined by :

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

3 CP-DECOMPOSITION AND PROXIMAL ALGORITHMS

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 [25].

To simplify writing, we will use the term "tensor" in a restricted sense, i.e., as a three-dimensional array of complex numbers (i.e., $N = 3$). However, the generalization to Nth order tensors, $N \geq 3$, is straightforward. Let's consider a tensor $\mathcal{X}$ of order 3 with size $I \times J \times K$, its CP-decomposition is defined as follows:

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

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

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_R]$ is a vector containing the scaling factors λ_r and the three matrices $\mathbf{A}= [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_R] \in \mathbb{C}^{I \times R}$, $\mathbf{B}= [\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_R] \in \mathbb{C}^{J \times R}$ and $\mathbf{C}= [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_R] \in \mathbb{C}^{K \times R}$ 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 Canonical Polyadic Decomposition (CPD) of $\mathcal{X}$ [1].

3.1 Low rank

Although the tensor of interest is of low rank, it is frequently necessary to look for a low-rank (*e.g.* rank R) approximation of the observed tensor due to the presence of noise. In the latter case, the observed tensor is indeed generally of *generic rank*, strictly larger than R [25]. In the low-rank approximation problem [9], the goal is to minimize an objective function Υ of the form:

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

Alternatively, the minimization of (8) can be expressed using vectorization property where $\mathbf{x} = vec\{\mathcal{X}\}$ as:

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

where symbol $\boxtimes$ represents the Kronecker product [26].

3.2 Coherence

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

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

3.3 Conditioning of the problem

One of the attractive properties of tensors of order greater than two, $N > 2$, lies in the uniqueness of their CP decomposition, contrary to matrix decompositions [3] [26][10] (the decomposition of a matrix into a sum of rank-one matrices also exists, but it is not unique, unless some strong constraints are imposed, such as orthogonality or non-negativity).

From the definition of CP decomposition, it is obvious that the decomposition (7) is insensitive to:

- Permutation of the rank-1 terms, which refers to the permutation indeterminacy.
- Scaling of vectors a_r, b_r and c_r, provided the product of the scaling factors is equal to 1, which corresponds to the scaling indeterminacy.

In numerical algorithms, it is useful to fix indeterminacies. For instance, columns of factor matrices can be normalized and their norm stored in scaling factor $\boldsymbol{\lambda}$ [26]. 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 Υ is determined by cancelling the gradient of (8) w.r.t. $\boldsymbol{\lambda}$, which then results in the linear system:

$$\mathbf{G}\,\boldsymbol{\lambda}_o = \mathbf{s} \tag{11}$$

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

$$G_{pq} = (\mathbf{a}_p \boxtimes \mathbf{b}_p \boxtimes \mathbf{c}_p)^H (\mathbf{a}_q \boxtimes \mathbf{b}_q \boxtimes \mathbf{c}_q)$$

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

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

Note that entries of $\mathbf{G}$ can preferably be obtained by:

$$G_{pq} = \mathbf{a}_p^H \mathbf{a}_q \cdot \mathbf{b}_p^H \mathbf{b}_q \cdot \mathbf{c}_p^H \mathbf{c}_q.$$

3.4 Proximal mapping

Proximal mapping are now powerful and reliable optimization tools, 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. These methods have been used effectively in unconstrained CP decomposition, which can ensure that the CP converges to a fixed point [33, 18]. We shall be particularly interested in the Accelerated Proximal Gradient (APG) algorithm [33], as it satisfies the assumptions of our CP decomposition formulation problem.

Given a function $\mathcal{G}$, the proximal mapping (or proximal operator) [27] maps an input point $\mathbf{x}$ to the minimizer of $\mathcal{G}$ restricted to small proximity to $\mathbf{x}$. The definition of the proximal operator is recalled hereafter.

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

$$\mathbf{prox}_{\theta\mathcal{G}}(\mathbf{x}) = \underset{y\in\mathbb{R}^N}{\text{minimize}}\ \mathcal{G}(\mathbf{y}) + \frac{1}{2\theta}\|\mathbf{x}-\mathbf{y}\|_2^2 \tag{12}$$

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

The operator $\mathbf{prox}_{\theta\mathcal{G}} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ thus defined is the prox-operator of $\mathcal{G}$, and the parameter θ controls the stretch to which the proximal operator maps points towards the minimum of $\mathcal{G}$, with larger values of θ associated with mapped points near the minimum, and smaller values giving a smaller movement towards the minimum [27].

The proximal mapping gets really handy for composite problem when the minimization has the form of $\Upsilon + \mathcal{G}$ with Υ convex and differentiable and $\mathcal{G}$ convex with "simple" proximal mapping, i.e. its prox-operator admits a closed form [27]. Recent work has expanded these proximal methods to general non-smooth and non-convex problems. Among these works, one can find those in [24, 28, 29], where a monotonic descent of the objective value is imposed to ensure convergence, while on the other hand, the works in [30, 31] which introduce a generic method for non-smooth non-convex problems based on Kurdyka Lojasiewicz theory.

This general composite problem can be solved with one of the proximal methods mentioned above, but the one that reply to the assumptions of our specific problem (15), and which is the main proposal in this paper, is the accelerated proximal gradient method. A natural strategy of this method is firstly to reduce the value of Υ by using unconstrained iterative optimization methods such as descent methods or Newton's method, followed by the reduction of the value of $\mathcal{G}$ by applying the prox-operator of $\mathcal{G}$ (using the same step-size) and repeat until convergence to a minimizer (under some further conditions). This strategy yields the following iteration:

$$\mathbf{x}^{(k+1)} = \mathbf{prox}_{\theta^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \theta^{(k)}\mathbf{d}^{(k)}) \tag{13}$$

This last strategy describes the general principle of the iterative proximal method. In our present paper, we exploit the rigorous argument provided in [24] proving that the limit point of the sequence generated by the accelerated proximal gradient algorithm is the critical point of the objective function (8).

4 PROPOSED METHOD

4.1 Problem formulation

In contrast to the alternative approach where factor matrices are estimated alternately. Our approach addresses the 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 an optimization problem and more precisely to a minimization problem in which the function to be minimized is composed of two terms: one related to the properties of the noise, called the "data fidelity term" which is defined in (8) by the cost function Υ, and the other related to the a priori information on model parameters, called "regularization", which is currently defined as a regularization function that penalizes the difference between the current and previous factor iterates and which will be represented by $\mathcal{G}$. In [32], several numerical examples show that this regularization can help the algorithm to keep distance from the degenerate swamps, i.e., the degenerate regions where convergence is slow, in addition, it has also been shown in [33] that the limit point obtained from the regularized CP decomposition (15) is a critical point of the original minimization problem $\left\|\mathcal{X} - \widehat{\mathcal{X}}\right\|_F^2$.

As indicated in (9), 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:

$$\mathcal{F}(\mathbf{x}) = \underbrace{\Upsilon(\mathbf{x})}_{Fidelity} + \underbrace{\mathcal{G}(\mathbf{x})}_{Regularization} . \tag{14}$$

And can be explicitly written as:

$$\mathcal{F}(\mathbf{x}) = \min_{\widehat{\mathbf{x}}_k} \|\mathbf{x} - \widehat{\mathbf{x}_k}\|_F^2 + \eta_n \|\tilde{\mathbf{x}}_{k-1} - \widehat{\mathbf{x}}_k\|_F^2 , \tag{15}$$

where $\tilde{\mathbf{x}}_{k-1}$ is the predecessor version of $\mathbf{x}_k$ in the previous iteration and η_n is a penalty weight that controls the sharpness of the penalty, which decreases through iterations. In this paper, we exploit the convergence analysis of the proposed method in [24] using the accelerated proximal gradient (APG) as a solution to the optimization problem defined in (15).

The general principle of the proposed approach is summarized in **Algorithm 1** and detailed in the following paragraphs, where there are essentially two basic steps:

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

4.2 Gradient step

This step improves the approximate solution, focusing only on the data fidelity - independently 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 as follows:

$$\mathbf{d}^{(k)} = -\nabla\Upsilon(\mathbf{x}^{(k)}) \tag{16}$$

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

$$\frac{\partial\Upsilon}{\partial\mathbf{A}} = 2\mathbf{A}\mathbf{M}^A - 2\mathbf{N}^A \tag{17}$$

with

$$\begin{aligned} M^A_{pq} &\overset{def}{=} \Sigma_{jk}\lambda_p B_{jp} C_{kp} C^*_{kq} B^*_{jq} \lambda^*_q \\ N^A_{ip} &\overset{def}{=} \Sigma_{jk} T_{ijk} B^*_{jp} C^*_{kp} \lambda^*_p \end{aligned} \tag{18}$$

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. It depends on two parameters α and β, with $0 < \alpha < 0.5$ and $0 < \beta < 1$. Now, 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 [34] is verified:

$$\Upsilon(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}^{(k)}) \tag{19}$$

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

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}. \tag{20}$$

4.3 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, we apply the proximal algorithm to the previous point arising from the preceding step of the gradient, i.e. $\mathbf{z}^{(k)}$, as follows:

$$\begin{aligned} \mathbf{z}^{(k+1)} &= \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{x}^{(k)} + \rho^{(k)}\mathbf{d}^{(k)}) = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) \\ &= \arg\min_{\mathbf{x}} \underbrace{(\mathcal{G}(\mathbf{x}) + \frac{1}{2\rho^{(k)}}\|\mathbf{x} - \mathbf{z}^{(k)}\|_2^2)}_{\mathcal{H}(\mathbf{x})} \end{aligned} \tag{21}$$

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 $\mathcal{G}$. For this, 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 gradient of $\mathcal{H}$ takes the form:

$$\begin{aligned} \nabla\mathcal{H}(\mathbf{x}) = 0 &\implies \nabla\mathcal{G}(\mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0 \\ &\implies -2 * \eta_n(\tilde{\mathbf{x}}_{k-1} - \mathbf{x}) + \frac{1}{\rho^{(k)}}(\mathbf{x} - \mathbf{z}^{(k)}) = 0, \end{aligned} \tag{22}$$

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

$$\mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}^{(k)}) = \frac{\frac{1}{\rho^{(k)}}\mathbf{z}^{(k)} + 2 * \eta_n\tilde{\mathbf{x}}_{k-1}}{\frac{1}{\rho^{(k)}} + 2 * \eta_n}. \tag{23}$$

Momentum The Accelerated Proximal Gradient (APG) algorithm will then extrapolate the point $\mathbf{v}_k$ by a combination of the current point and the previous point as :

$$\mathbf{v}_k = \mathbf{z}_{(k+1)} + \gamma_k(\mathbf{z}_{(k+1)} - \mathbf{x}_k),$$

where γ_k is the momentum that is adapted to each iteration and through a monitor that also ensures the descent property $\mathcal{F}(x^{(k+1)}) < \mathcal{F}(x^{(k)})$; it is defined, for t$\in$ (0,1), as follows:

$$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1}, \gamma_k \leftarrow t\gamma_k & \text{if } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{v}_k) \\ \mathbf{v}_k, \gamma_k \leftarrow \dfrac{\gamma_k}{t} & \text{otherwise.} \end{cases} \tag{24}$$

5 RESULTS AND DISCUSSIONS

In this section, we present the simulations results of the proposed method using two scenarios: (i) the first one, which represents the normal case where factor matrices are uncorrelated, while in (ii) the second scenario which is a specific case where the swamp problem needs to be addressed, i.e., the factor matrices are highly correlated in all three matrices.

To see the advantage of the proposed algorithm, we compare it to three other CPD algorithms: i) the Alternating Least Squares (ALS) [4], ii) the Levenberg-Marquardt (LM) [6] and iii) the unconstrained version with gradient (UG) [35].

Besides, we evaluate the performance of each algorithm according to two criteria, namely accuracy and CPU time.

Note that all algorithms are initialized at each simulation with the same starting points and share common stopping criteria which are as follows:

Algorithm 1: Accelerated Proximal gradient (APG) with adaptive momentum to minimize (15)

Initialize $(\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_0)$ to matrices with unit-norm columns, γ_0, η_0. ;
Calculate the optimal scaling factor $\boldsymbol{\lambda}^*$ by solving (11): $\mathbf{G}_0\boldsymbol{\lambda}^* = \mathbf{s}_0$;
for $k \geq 1$ *and subject to a stopping criterion* **do**

1. Gradient Step
   (a) Compute the descent direction $\mathbf{d}^{(k)}$ as the gradient according to (16) w.r.t. $\mathbf{x}_k$: $\mathbf{d}^{(k)} = -\nabla f(\mathbf{x}_k)$
   (b) Calculate a step-size ρ_k using the backtracking method such: $\Upsilon(\mathbf{x}_k + \rho_k \mathbf{d}^{(k)}) < \Upsilon(\mathbf{x}_k)$
   (c) Update $\mathbf{z}_k = \mathbf{x}_k + \rho_k \mathbf{d}^{(k)}$
2. Proximal Step
   (a) Compute the proximal operator of g at $\mathbf{z}_k$ using (23) such as: $\mathbf{z}_{k+1} = \mathbf{prox}_{\rho^{(k)}\mathcal{G}}(\mathbf{z}_k)$
   (b) Momentum $\mathbf{v}_k = \mathbf{z}_{(k+1)} + \gamma_k(\mathbf{z}_{(k+1)} - \mathbf{x}_k)$
   (c) Monitor
   $$\mathbf{x}_{k+1} = \begin{cases} \mathbf{z}_{k+1}, \gamma_k \leftarrow t\gamma_k & \text{if } \mathcal{F}(\mathbf{z}_{k+1}) < \mathcal{F}(\mathbf{v}_k) \\ \mathbf{v}_k, \gamma_k \leftarrow \dfrac{\gamma_k}{t} & \text{otherwise.} \end{cases}$$
3. Extract the three blocks of $\mathbf{x}_{k+1}$: $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ and $\mathbf{C}_{k+1}$
4. Normalize the columns of $\mathbf{A}_{k+1}$, $\mathbf{B}_{k+1}$ and $\mathbf{C}_{k+1}$
5. calculation of the optimal scaling factor $\boldsymbol{\lambda}^*$ using (11) such as: $\mathbf{G}\boldsymbol{\lambda}^* = \mathbf{s}$

end

(i) The maximum number of iterations is fixed at 10^3.
(ii) Reconstruction error which is the magnitude of the difference between the current tensor and the original tensor.

5.1 Scenario 1

In this first scenario, we randomly generate a three-dimensional tensor of size $10\times20\times30$ with rank 8 and with mutual coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.5$. Note that in this first setting, η is initialized to 0.5, and is divided by 100 when $\Upsilon(x)$ is reduced by less than 10^{-4}. Figure 1 illustrates the reconstruction error as a function of the number of iterations, from which it can clearly be observed that in the normal case, the ALS algorithm remains the most efficient of all the algorithms followed by the proposed algorithm then the LM algorithm and finally the unconstrained algorithm which requires more iterations to reach an accuracy of 10^{-15}.

To ensure a detailed comparison, we evaluated the CPU time of these algorithms, by analysing the results of Table 1, we can still see that the ALS algorithm remains the fastest of all these algorithms. In addition, the proposed accelerated proximal gradient algorithm also maintains its efficiency in terms of CPU time compared to the LM and UG algorithms.

Table 1. CPU time (in seconds). For a tensor of size $10 \times 20 \times 30$ and rank 8 up to a precision of 10^{-15} and results with 100 random initializations.

Algorithms	CPU time (in seconds)
ALS	**1.47**
LM	2.45
UG	5.41
proposed	2.33

5.2 Scenario 2

In this second scenario, we randomly generate a tensor of size $4\times3\times6$ with rank 4 and with mutual coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$. Note that in this second setting, η is initialized to 1, and is divided by 10 when $\Upsilon(x)$ is reduced by less than 10^{-4}. Figure 2 and table 2 show that the ALS algorithm requires more iterations and more machine time to reach an accuracy of 10^{-12}, we can clearly see that the objective function stays in a value and does not decrease for a long time, this slowed convergence, characterized by a flat curve in the error plot, refers to the swamp phenomenon. On the other hand, the proposed algorithm can reduce the swamp by taking about half the number of iterations and CPU time to achieve the same accuracy.

From these simulation results, one can observe the impact of the penalty through the accelerated proximal gradient algorithm in the accuracy of the results. This makes the proposed algorithm a better choice to ensure both the

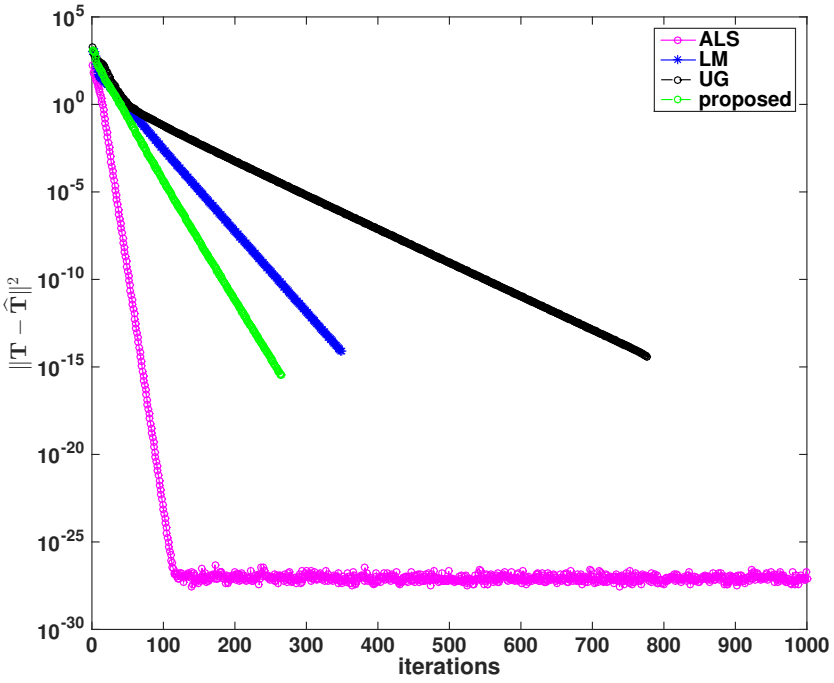

Fig. 1. Reconstruction error (8) as a function of the number of iterations. For a tensor of size $10 \times 20 \times 20$ with rank 8 and with mutual coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.5$.

accuracy and the convergence speed of the CP decomposition in the difficult case of swamp phenomenon.

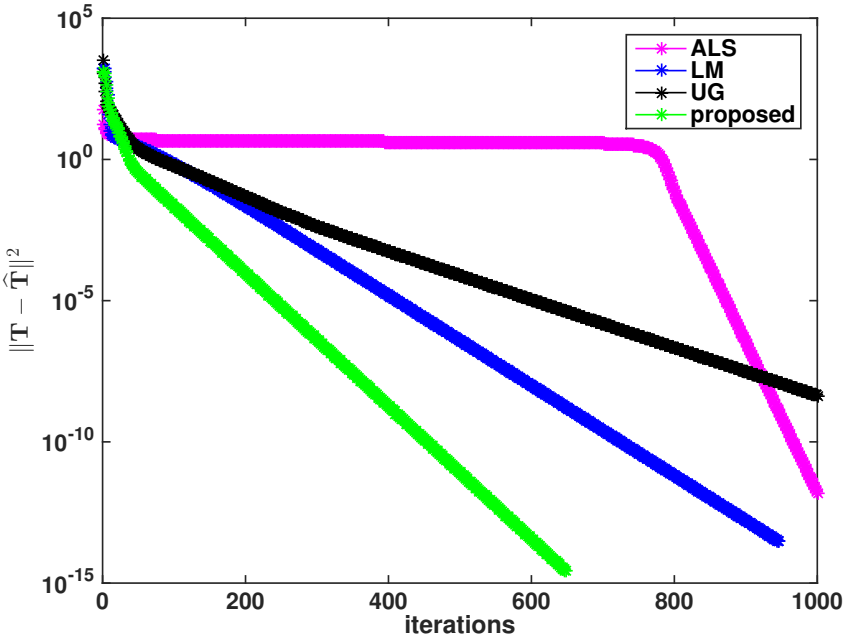

Fig. 2. Reconstruction error (8) as a function of the number of iterations. For a tensor of size $4 \times 3 \times 6$ with rank 4 and with mutual coherences $\mu(\mathbf{A}) = \mu(\mathbf{B}) = \mu(\mathbf{C}) = 0.99$.

6 Conclusions

We have designed a method for calculating the Canonical Polyadic decomposition based on the Accelerated Proximal Gradient algorithm and through the introduction of a regularization function that penalizes the difference between the current and previous factor iterates using a new strategy capable of efficiently monitoring this regularization. We performed a complete comparison based on computer experiments, which proved the good performance of the proposed algorithm in terms of accuracy and convergence speed, compared to other iterative algorithms, especially when the swamp phenomenon occurs.

References

1. Comon, P.: Tensor decompositions: state of the art and applications. Math. Signal Process. V (Coventry, 2000), 1-24.

Table 2. CPU time (in seconds). For a tensor of size $4 \times 3 \times 6$ and rank 4 up to a precision of 10^{-6} and results with 100 random initializations.

Algorithms	CPU time (in seconds)
ALS	5.85
LM	4.78
UG	9.91
proposed	**3.41**

2. Hitchcock, Frank L.: The expression of a tensor or a polyadic as a sum of products. Journal of Mathematics and Physics 6.1-4 (1927): 164-189.
3. Harshman, Richard A.: Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis." (1970): 1-84.
4. Carroll, J. Douglas, and Jih-Jie Chang.: Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition. Psychometrika 35.3 (1970): 283-319.
5. Kiers, Henk AL.: Towards a standardized notation and terminology in multiway analysis. Journal of Chemometrics: A Journal of the Chemometrics Society 14.3 (2000): 105-122.
6. Comon, Pierre, Xavier Luciani, and Andr LF De Almeida.: Tensor decompositions, alternating least squares and other tales. Journal of Chemometrics: A Journal of the Chemometrics Society 23.78 (2009): 393-405.
7. Bro, Rasmus.: PARAFAC. Tutorial and applications. Chemometrics and intelligent laboratory systems 38.2 (1997): 149-172.
8. Murphy, Kathleen R., et al.: Fluorescence spectroscopy and multi-way techniques. PARAFAC. Analytical Methods 5.23 (2013): 6557-6566.
9. Rouijel, Awatif, et al.: 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. Sidiropoulos, Nicholas D., Georgios B. Giannakis, and Rasmus Bro.: Blind PARAFAC receivers for DS-CDMA systems. IEEE Transactions on Signal Processing 48.3 (2000): 810-823.
11. Sahnoun, Souleymen, and Pierre Comon.: Joint source estimation and localization. IEEE Transactions on Signal Processing 63.10 (2015):2485-2495.
12. Shashua, Amnon, and Tamir Hazan.: Non-negative tensor factorization with applications to statistics and computer vision. Proceedings of the 22nd international conference on Machine learning. 2005.
13. Zhang, Qingchen, et al.: High-order possibilistic c-means algorithms based on tensor decompositions for big data in IoT. Information Fusion 39 (2018): 72-80.
14. Zhang, Qingchen, et al.: An improved deep computation model based on canonical polyadic decomposition. IEEE Transactions on Systems, Man, and Cybernetics: Systems 48.10 (2017): 1657-1666.
15. Kruskal, Joseph B.: 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. Stegeman, Alwin, and Nicholas D. Sidiropoulos.: On Kruskals uniqueness condition for the Candecomp/Parafac decomposition. Linear Algebra and its applications 420.2-3 (2007): 540-552.

17. Li, Na, Stefan Kindermann, and Carmeliza 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. Navasca, Carmeliza, Lieven De Lathauwer, and Stefan Kindermann. "Swamp reducing technique for tensor decomposition." 2008 16th European Signal Processing Conference. IEEE, 2008.
19. Mitchell, Ben C., and Donald S. Burdick.: Slowly converging PARAFAC sequences: swamps and twofactor degeneracies. Journal of Chemometrics 8.2 (1994): 155-168.
20. Sanchez, Eugenio, and Bruce R. Kowalski.: Tensorial resolution: a direct trilinear decomposition. Journal of Chemometrics 4.1 (1990): 29-45.
21. De Lathauwer, Lieven.: 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.
22. Nazih Marouane, Khalid Minaoui, and Pierre 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.
23. Farias, Rodrigo Cabral, Jos Henrique de Morais Goulart, and Pierre Comon.: Coherence Constrained Alternating Least Squares. 2018 26th European Signal Processing Conference (EUSIPCO). IEEE, 2018.
24. Li, Qunwei, et al.: Convergence analysis of proximal gradient with momentum for nonconvex optimization. arXiv preprint arXiv:1705.04925 (2017).
25. Comon, Pierre.: Tensors: a brief introduction. IEEE Signal Processing Magazine 31.3 (2014): 44-53.
26. Kolda, Tamara G., and Brett W. Bader.: Tensor decompositions and applications. SIAM review 51.3 (2009): 455-500.
27. Parikh, Neal, and Stephen Boyd.: Proximal algorithms. Foundations and Trends in optimization 1.3 (2014): 127-239.
28. Fukushima, Masao, and Hisashi Mine.: A generalized proximal point algorithm for certain non-convex minimization problems. International Journal of Systems Science 12.8 (1981): 989-1000.
29. Nesterov, Yu.: Gradient methods for minimizing composite functions. Mathematical Programming 140.1 (2013): 125-161.
30. Attouch, Hedy, Jrme Bolte, and Benar Fux Svaiter.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forwardbackward splitting, and regularized GaussSeidel methods. Mathematical Programming 137.1-2 (2013): 91-129.
31. Li, Huan, and Zhouchen Lin.: Accelerated proximal gradient methods for nonconvex programming. Advances in neural information processing systems. 2015.
32. Paatero, Pentti.: Construction and analysis of degenerate PARAFAC models. Journal of Chemometrics: A Journal of the Chemometrics Society 14.3 (2000): 285-299.
33. Li, Na, Stefan Kindermann, and Carmeliza Navasca.: Some convergence results on the regularized alternating least-squares method for tensor decomposition. Linear Algebra and its Applications 438.2 (2013): 796-812.
34. Nazih Marouane, and Khalid Minaoui.: A progression strategy of proximal algorithm for the unconstrained optimization. 2018 4th International Conference on Optimization and Applications (ICOA). IEEE, 2018.
35. Comon, P., Minaoui, K., Rouijel, A. and Aboutajdine, D.: Performance index for tensor polyadic decompositions. 21st European Signal Processing Conference (EUSIPCO 2013). IEEE, 2013.