FBS4: A Forward-Backward Splitting algorithm for constrained tensor decomposition

Elaheh Sobhani, Pierre Comon, Christian Jutten, Massoud Babaie-Zadeh

FBS4: A Forward-Backward Splitting algorithm for constrained tensor decomposition

Elaheh Sobhani, Pierre Comon, Christian Jutten, and Massoud Babaie-Zadeh

***Abstract*—Tensors (multi-way arrays) are very practical in various applications such as chemometrics, text mining, medical image and signal processing, where desired parameters are estimated by tensor decomposition. It has been shown that *constrained* tensor decomposition performs better than unconstrained in parameter identification problems. Most tensor decomposition algorithms are based on Alternating Least Squares (ALS), and for constrained decomposition it is needed to solve a constrained minimization in each step of ALS. Over the past decade, some algorithms based on ALS have been proposed for constrained (mostly non-negative) tensor decomposition, and applied Alternating Direction Method of Multipliers (ADMM) or proximal methods to handle the constraint. Although ADMM based method performs efficiently in various cases, there is no convergence guarantee for this method in case of non-convex constraint. On the other hand, proximal based methods proposed so far suffer from lack of expected accuracy in the decomposition, while there is a convergence proof for these kinds of methods even in case of non-convexity. In this paper, an algorithm is proposed based on ALS for constrained tensor decomposition, which utilizes a particular proximal method called Forward-Backward Splitting to handle the constraint. We call this algorithm FBS4, which stands for "Forward-Backward Splitting with Smart initialization for tensor CP decompositions under non-negativity, Sparsness or Simplex constraints". FBS4 is theoretically one step ahead compared to ADMM-based methods, since (i) the provided convergence analysis of FBS4 holds true for both convex and non-convex constraints such as sparsity; (ii) in practice FBS4 enables to manage a large range of constraints such as non-negativity, simplex set and sparsity; (iii) computer results show that FBS4 achieves state-of-the-art performances; (iv) FBS4 algorithm is simpler and faster compared to other algorithms based on proximal approaches.**

***Index Terms*—Tensor decomposition, Proximal, Forward-Backward Splitting, constraint, non-negative, simplex, sparsity.**

I. Introduction

Considering vectors and matrices as one-way and two-way arrays, respectively, the concept can be extended to *tensors* as multi-way arrays which can have more than two dimensions. Although vectors/matrices can be seen as tensors of order one or two, tensors usually refer to an array with more than two ways [1], [2].

Tensors have appeared in many applications so far, such as Blind Source Separation (BSS) based on data cumulants [1], component analysis in chemistry [3], estimation and localization of sources (Direction Of Arrival (DOA)) [4]. In addition, the importance of tensors has been revealed in many tasks of machine learning including classification [2], data fusion [5], topic modeling [6], [7] and the estimation of parameters in mixture models [6].

In order to recover latent variables by means of tensors, typically data tensor is decomposed into some multi-way arrays (vectors, matrices or tensors) which can be interpreted as the desired latent variables of the problem. There are two prevalent decompositions in the literature: Tucker decomposition [8] and Canonical Polyadic (CP) decomposition [1], also referred to as CANDECOMP or PARAFAC [9] in some communities. The major part of literature on tensors is devoted to algorithms for these decompositions and their variants [1], [2], [5].

According to the application, it is preferred to add some constraints to the tensor decomposition which results normally in much more accurate and reasonable solutions. Non-negativity, belonging to simplex space, orthogonality and sparsity are some examples of key constraints often introduced in applications such as medical image and signal processing [10], probability estimation in topic modeling [7], [11] and dictionary learning [12].

Generally, the algorithms of constrained tensor decomposition are inspired from constrained matrix decomposition (factorization). For instance, the algorithms mentioned in [10] for Non-negative Tensor Factorization (NTF) are the extensions of Non-negative Matrix Factorization (NMF). Moreover, many of the existing algorithms are based on Alternating Optimization (AO) [2] or its special case, Alternating Least Squares (ALS) [5], in which the data fidelity term in AO is the least square error.

Over the past decade, some algorithms [13]–[15] have been proposed for constrained tensor decomposition based on AO or ALS, where in each step, a constrained minimization over one parameter is carried out. In each step, Alternating Direction Method of Multipliers (ADMM) [16] or proximal methods [17] have been applied to solve the constrained minimization.

Although Alternating Optimization-Alternating Direction Method of Multipliers (AO-ADMM) [13] shows better performance for constrained tensor decomposition [18] compared to traditional approaches, there is no convergence guarantee in case of non-convex constraints such as sparsity. On the other hand, it is proved that the algorithms based on proximal method [14], [15], [19] converge to a critical point of the problem even for non-convex constraints, but the performances of these kinds of algorithms have not yet achieved that of AO-ADMM.

In this paper, an algorithm for constrained tensor decomposition is proposed, which is based on ALS. In each step of this new algorithm, a constrained minimization is solved

E. Sobhani, P. Comon and C. Jutten are with GIPSA-lab, Univ. Grenoble Alpes, CNRS, Grenoble, France, e-mail: first name.last name@grenoble-inp.fr.

E. Sobhani and M. Babaie-Zadeh are with the Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran, e-mail: sobhani.es@gmail.com; mbzadeh@yahoo.com.

by means of a specific proximal approach called *Forward-Backward Splitting* [17], [20]. We call this algorithm FBS4, which stands for "**F**orward-**B**ackward **S**plitting with **S**mart initialization for tensor CP decompositions under non-negativity, **S**parsness or **S**implex constraints". Compared to recent algorithms, e.g. [13], [14], the contributions of this paper are of high practical and theoretical interests for many scientists, since:

- it achieves much better performance compared to state-of-the-art methods including [14],
- it is simpler to understand and implement,
- it is able to easily manage a large range of constraints,
- there is a convergence proof, which remains valid even for non-convex constraints, unlike AO-ADMM.

The paper is organized as follows. In Section II, some preliminaries about tensor, CP decomposition and proximal methods are reviewed. Some recent algorithms for constrained tensor decomposition utilizing ADMM or proximal methods are studied in Section III. FBS4 is described in Section IV, as well as a convergence analysis. Section V eventually reports computer results.

Notation. Vectors, matrices and tensors are denoted with bold lowercases (*e.g.* $\boldsymbol{a}$), bold uppercases (*e.g.* $\boldsymbol{A}$) and bold calligraphic letters (*e.g.* $\boldsymbol{\mathcal{T}}$), respectively. The tensor (outer) product and Khatri-Rao product are denoted by $\otimes$ and $\odot$, respectively.

II. Preliminary

A. Tensor and CP decomposition

Tensors can be considered as a multi-linear map from a vector space to another one [1], or they are simply multi-way (multi-dimensional or multi-index) numerical arrays [5], [10]. The order of an array refers to the number of its ways [21]. Vectors and matrices are one-way and two-way arrays, respectively, but usually tensor refers to an array with three or more ways [2].

A decomposable tensor of order N is a tensor product of N vectors [1], *i.e.*, $\boldsymbol{\mathcal{D}} = \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(2)} \ldots \otimes \boldsymbol{a}^{(N)}$. According to [9], any tensor can be written as a linear combination of finite number of decomposable tensors,

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

where $\boldsymbol{\mathcal{T}}$ is a tensor of order N, and $\boldsymbol{\mathcal{D}}(r) = \boldsymbol{a}_r^{(1)} \otimes \boldsymbol{a}_r^{(2)} \ldots \otimes \boldsymbol{a}_r^{(N)}$. In compact form, (1) can be represented as $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, in which $\boldsymbol{\lambda}$ is a *coefficient vector* of size R containing the values of λ_r and $\boldsymbol{a}_r^{(1)}, \boldsymbol{a}_r^{(2)}, \ldots \boldsymbol{a}_r^{(N)}$ are the r^{th} columns of N *loading matrices* $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$, respectively. The N matrices $\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}$ are called mode-1, mode-2. …, mode-N loading matrices, respectively, since their columns are responsible for the construction of the first, second, …, N^{th} dimension of $\boldsymbol{\mathcal{T}}$ [5].

For the sake convenience, tensors are sometimes transformed into matrices [1]. This transformation is called *unfolding* or *flattering* and can be done in each mode. The resulting matrix in mode n is called *mode-n unfolding* [1] and is denoted by $\boldsymbol{\mathcal{T}}^{(n)}$.

For each tensor, the minimum value of R for which (1) holds is called tensor rank [1]. Therefore, the rank of a decomposable tensor is one. The decomposition described in (1) is called *Polyadic decomposition* [9] or also *CANDECOMP* or *PARAFAC* [1].

Based on the number of knowns and unknowns in (1), the expected rank of a tensor is defined as:

$$R^{\circ} \triangleq \left\lceil \frac{D}{1 - N + \sum_i n_i} \right\rceil,$$

where N is the order of the tensor of dimensions $n_1 \times \ldots \times n_N$ and $D = \prod_i n_i$. It has been shown in [22] that $R \leq R^{\circ} - 1$ ensures almost surely the uniqueness of decomposition (1). If this polyadic decomposition is unique, it can be called *Canonical Polyadic (CP)* decomposition [1].

As in practice, data stored in the form of a tensor are usually corrupted by noise, then the best rank-R approximation must be estimated. Although low-rank approximation is useful, generally it is ill-posed [23], [24], since the set of tensors of rank at most R is not closed [1]. Therefore, imposing some constraints such as non-negativity in the CP decomposition is proposed in the literature to overcome this difficulty. Some prevalent constraints are listed below:

- **Non-negativity:** All the loading matrices and the coefficient vector are supposed to be non-negative even if the tensor to decompose is not completely non-negative due to noise. There is a rich literature on the tensor decomposition under this constraint [5].
- **Simplex:** This constraint usually appears in the models involving a probabilistic analysis [6], [7], [11]. All or part of the columns/rows of the loading matrices and/or the coefficient vector in (1) should belong to a simplex set defined by: $\mathcal{S} \triangleq \{\boldsymbol{x} : \boldsymbol{x} \geq 0, \|\boldsymbol{x}\|_1 = 1\}$
- **Orthogonality:** it can be imposed between columns of loading matrices or between decomposable tensors, $\boldsymbol{\mathcal{D}}_r$, in (1) [1], [25]. The former constraint is widely used in blind source separation after standardization [26].
- **Sparsity:** In some applications [12], it may be needed to impose sparsity constraints on loading matrices or coefficient vectors. The exact sparsity constraint is also known as *cardinality* constraint, and defines a non-convex set [13]. Cardinality of a vector can be measured by the ℓ_0 pseudo-norm (or the counting norm), which is the number of non-zero entries. Since ℓ_0 is a non-convex function, sometimes its convex approximation such as the ℓ_1 norm [20] or Smoothed ℓ_0 (SL0) [27] is used instead.

B. Proximal concept and approach

In the sequel, some properties of functions and definitions are required, which are defined as follows:

* **Lower semi-continuity [28]:** Suppose $\overline{\mathbb{R}} = [-\infty, +\infty]$. The function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ is called *lower semi-continuous* on $\mathbb{R}^n$ if $\liminf_{\boldsymbol{x} \to \overline{\boldsymbol{x}}} f(\boldsymbol{x}) \geq f(\overline{\boldsymbol{x}})$ holds for every $\overline{\boldsymbol{x}} \in \mathbb{R}^n$.

* **Sub-gradient [28, Definition 8.3]:** Suppose $\overline{\mathbb{R}} = [-\infty, +\infty]$. For a function $f : \mathbb{R}^n \mapsto \overline{\mathbb{R}}$ and a point $\overline{\boldsymbol{x}}$ for which $f(\overline{\boldsymbol{x}})$ is finite, a vector $\boldsymbol{v} \in \mathbb{R}^n$ is said to be the sub-gradient of f at $\overline{\boldsymbol{x}}$, *i.e.* $\boldsymbol{v} \in \partial f(\overline{\boldsymbol{x}})$, if $f(\boldsymbol{x}) \geq f(\overline{\boldsymbol{x}}) + < \boldsymbol{v}, \boldsymbol{x} - \overline{\boldsymbol{x}} > + o(|\boldsymbol{x} - \overline{\boldsymbol{x}}|)$.
* **Kurdyka-Lojasiewicz property (KL) [29]:** denote the domain of a function f by $\text{dom}(f)$. The function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ has the Kurdyka-Lojasiewicz property at $\boldsymbol{x}^* \in \text{dom}(f)$ if there exist $\eta \in (0, +\infty]$, a neighborhood U of $\boldsymbol{x}^*$ and a continuous concave function $\varphi : [0, \eta) \mapsto \mathbb{R}_+$ such that:
  1) $\varphi(0) = 0$,
  2) φ is differentiable on $(0, \eta)$,
  3) $\varphi'(y) \geq 0$ for all $y \in (0, \eta)$,
  4) The Kurdyka-Lojasiewicz inequality,

$$\varphi'(f(\boldsymbol{x}) - f(\boldsymbol{x}^*))\text{dist}(0, \partial f(\boldsymbol{x})) \geq 1,$$

  holds for all $\boldsymbol{x} \in U \cap [f(\boldsymbol{x}^*) < f < f(\boldsymbol{x}^*) + \eta]$, where $dist$ denotes the distance function. In the rest of paper, this property is referred to as "KL", in short.

In the rest of this section, proximal concept and approaches are reviewed.

*1) **Proximity operator**:* The projection of a vector $\boldsymbol{x} \in \mathbb{R}^N$ onto a closed convex set $\mathcal{S} \subset \mathbb{R}^N$ is a classical problem in signal processing which can be formulated as [17, (31,42,141)]:

$$\underset{\boldsymbol{y} \in \mathbb{R}^N}{\text{argmin}} \left\{ i_{\mathcal{S}}(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right\}, \tag{2}$$

where $i_{\mathcal{S}}$ is the indicator function defined by:

$$i_{\mathcal{S}}(\boldsymbol{y}) \triangleq \begin{cases} 0 & if \quad \boldsymbol{y} \in \mathcal{S} \\ \infty & if \quad \boldsymbol{y} \notin \mathcal{S} \end{cases}$$

Let $\boldsymbol{\Gamma}_0(\mathbb{R}^N)$ be the class of lower semi-continuous convex functions $f : \mathbb{R}^N \mapsto (-\infty, +\infty]$, with $\text{dom}(f) \neq \varnothing$. Then $i_{\mathcal{S}}$ belongs to $\boldsymbol{\Gamma}_0(\mathbb{R}^N)$.

According to the proposition of Moreau in 1962 [30], the definition of *Proximity operator* is obtained by replacing $i_{\mathcal{S}}(\boldsymbol{y})$ in (2) with any arbitrary function in $\boldsymbol{\Gamma}_0(\mathbb{R}^N)$:

Definition 1 (Proximity operator [17])**.** *For every $\boldsymbol{x} \in \mathbb{R}^N$, the unique solution of the following minimization problem:*

$$\underset{\boldsymbol{y} \in \mathbb{R}^N}{\text{argmin}}\, f(\boldsymbol{y}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2, \tag{3}$$

is defined as the proximity operator of the function $f \in \boldsymbol{\Gamma}_0(\mathbb{R}^N)$, and it is denoted by $prox_f(\boldsymbol{x})$. Thus, the proximity operator of f is $prox_f : \mathbb{R}^N \mapsto \mathbb{R}^N$, and it is characterized by:

$$\boldsymbol{p} = prox_f(\boldsymbol{x}) \Leftrightarrow (\boldsymbol{x} - \boldsymbol{p}) \in \partial f(\boldsymbol{p}), \quad \forall (\boldsymbol{x}, \boldsymbol{p}) \in \mathbb{R}^N \times \mathbb{R}^N.$$

Note that $\partial f(\boldsymbol{p})$ is replaced by $\nabla f(\boldsymbol{p})$ for differentiable f.

The above definition indicates that $\text{prox}_f(\boldsymbol{x})$ is a point that minimizes f and simultaneously is as close as possible to $\boldsymbol{x}$. Therefore, $\text{prox}_f(\boldsymbol{x})$ is also called a *proximal point of $\boldsymbol{x}$ with respect to f* [20].

Algorithm 1 Forward-Backward Splitting [17], [31]

Input: β, $\boldsymbol{x}_0 \in \mathbb{R}^N$
1: Fix $\epsilon \in (0, \min\{1, \frac{1}{\beta}\})$
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: $\quad \gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$
4: $\quad \boldsymbol{y}_k = \boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)$
5: $\quad \alpha_k \in [\epsilon, 1]$
6: $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k(\text{prox}_{\gamma_k g}(\boldsymbol{y}_k) - \boldsymbol{x}_k)$
7: **end for**

*2) **Forward-Backward Splitting**:* In many signal processing applications, the cost function to be minimized is the sum of two functions where one of them is usually non-differentiable or even non-convex. By following a proximal approach, these kinds of problems can be solved by means of a particular algorithm called *Forward-Backward Splitting*. Let us now explain this algorithm.

Theorem 1 (Forward-Backward Splitting [17], [29])**.** *Suppose $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ is a proper lower semi-continuous function which has KL property and is bounded from below. If f can be spilt into two parts as $f = h + g$, where g is lower semi-continuous and $h : \mathbb{R}^N \mapsto \mathbb{R}$ is a finite valued, differentiable function with a β-Lipschitz continuous gradient,* i.e.*, $\exists \beta$ such that:*

$$\|\nabla h(\boldsymbol{x}) - \nabla h(\boldsymbol{y})\|_2 \leq \beta \|\boldsymbol{x} - \boldsymbol{y}\|_2,$$

then it can be shown [31] that the minimizer of f satisfies the following fixed point equation:

$$\boldsymbol{x} = prox_{\gamma g}(\boldsymbol{x} - \gamma \nabla h(\boldsymbol{x})), \tag{4}$$

where $\gamma \in (0, +\infty)$.

Equation (4) suggests an iterative approach, called the *Forward-Backward Splitting* algorithm:

$$\boldsymbol{x}_{k+1} = \text{prox}_{\gamma g}(\boldsymbol{x}_k - \gamma_k \nabla h(\boldsymbol{x}_k)), \tag{5}$$

where the values of γ_k should be chosen from a suitable bounded interval.

Several variations of implementing Forward-Backward Splitting exist, and are reported in [17]. Two of them are restated (Algorithms 1, and 2) to which we will refer in the rest of the paper. In Algorithm 1, a relaxation parameter, λ_k, has been introduced and cannot exceed 1. Algorithm 2 is proposed in [32], [33] as a proximal gradient algorithm. It is usually expected to have *global convergence* when the cost function to be minimized is convex [29], which means that the algorithm generates a converging sequence to the solution regardless of the starting point. It has been shown that every sequence generated by Algorithms 1 and 2 converges to a solution of $\min f$ [31], [34].

However, if the objective function is non-convex, the monotonicity of the sequences generated by descent methods will be broken and oscillatory behaviors may appear [29]. In order to achieve convergence in this kind of cases, it is necessary to limit ourselves to functions with some particular properties,

Algorithm 2 Beck-Teboulle proximal gradient algorithm [17]

Input: β, $\boldsymbol{x}_0 \in \mathbb{R}^N$

Set $\boldsymbol{z}_0 = \boldsymbol{x}_0$ and $t_0 = 1$
for $k = 0, 1, 2, \ldots$ **do**
  $\boldsymbol{y}_k = \boldsymbol{z}_k - \beta^{-1}\nabla h(\boldsymbol{z}_k)$
  $\boldsymbol{x}_{k+1} = \text{prox}_{\beta^{-1}g}(\boldsymbol{y}_k)$
  $t_{k+1} = \dfrac{1+\sqrt{4t_k^2+1}}{2}$
  $\lambda_k = 1 + \dfrac{t_k - 1}{t_{k+1}}$
  $\boldsymbol{z}_{k+1} = \boldsymbol{x}_k + \lambda_k(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$
end for

such as KL [29]. It has been proved that the sequence $\boldsymbol{x_k}$ generated by Theorem 1 converges to a critical point of $f = h + g$, if the mentioned sequence is bounded [29].

In many applications (including FBS4 in this paper), the function g is an indicator function of a particular set, i_C, and its proximity operator is a projection onto that set [17]. If the desired set is non-convex, the projection onto it may not result to a unique point. It has been proved [29] that in spite of the multivalued projection, the convergence property of Theorem 1 is not influenced. Note that this interesting conclusion is valid only if the assumptions of Theorem 1 are satisfied, the most important being the KL property satisfied by $h + i_C$.

III. Related works

In this section, some papers in which constrained (mostly non-negativity) CP decomposition is investigated by means of a proximal approach are reviewed. FBS4 will be compared to all of them in Section V.

A. Alternating Optimization-Alternating Direction Method of Multipliers (AO-ADMM) [13]

As mentioned in [17], [20], Alternating Direction Method of Multipliers (ADMM) [16] can be considered as a special case of proximal method. In [13], constrained CP decomposition by means of ADMM is discussed.

Since minimizing the cost function of the CP decomposition over all loading matrices is a non-convex problem, a common strategy to transform it to a convex function is Alternating Optimization (AO) [2], in which by fixing all the loading matrices (initializing or using their previous estimation) except one of them, one tries to minimize the cost function over just one loading matrix. AO-ADMM attempts to minimize the cost function overeach loading matrix by ADMM. The details of AO-ADMM for constrained CP decomposition can be found in [7], [11], [13]. The proposed algorithm in [13] is capable of applying several constraints on loading matrices, such as non-negativity, sparsity, smoothness, cardinality, etc.

The convergence of AO is mentioned in [35], [36]. Although the convergence of AO-ADMM for convex constraints, such as non-negativity, has been proved in [13], the convergence of ADMM is not guaranteed for non-convex constraints such as cardinality (ℓ_0 pseudo-norm) [13], [16].

B. Alternating Proximal Gradient (APG) [14]

In [14], three kinds of updates based on the proximal concept are introduced, namely *original, proximal and prox-linear*, for updating the unknown variables of a particular type of constrained optimization, including constrained tensor decomposition.

Prox-linear is similar to Forward-Backward Splitting in several respects, since the required assumption for its convergence is the Lipschitz continuous gradient of differentiable part of cost function and the KL property. In addition, for some usual constraints such as belonging to a particular set (i.e. indicator function), the prox-linear update reduces to the projection onto the set, as Forward-Backward Splitting.

The main difference between APG and the method proposed herein (FBS4) is the algorithm chosen for implementing the proximal approach. In fact, APG is based on Algorithm 2, while it shall be seen in Section IV that the algorithm utilized in FBS4 is Algorithm 1.

C. Fast Non-negative Tensor Factorization-APG (FastNTF-APG) [19]

FastNTF-APG [19] is a modified version of APG [14] dedicated to non-negative tensor decomposition. In [19], it is mentioned that, contrary to classical algorithms of Non-negative Tensor Factorization (NTF), which suffer from slow convergence especially in practical applications, FastNTF-APG speeds up NTF and overcomes this bottleneck by combining APG with low rank approximation.

D. Block Coordinate Variable Metric Forward-Backward (BC-VMFB) [15], [37], [38]

The cost function proposed in [15], [38] includes two main parts: data fidelity and regularization terms. The latter is capable of considering hard constraints such as non-negativity and regularizations such as sparsity. The method used for solving the resulting minimization is Block Coordinate Variable Metric Forward-Backward (BC-VMFB) [37].

BC-VMFB consists of two main steps: a gradient step related to data fidelity, which is assumed to be differentiable with β-Lipschitz gradient, and a proximal step linked to the regularization term, for which a new proximity operator should be calculated. This proximity operator of the function φ is associated with a symmetric positive definite matrix $\boldsymbol{P}$ by the following definition [15]:

$$\text{prox}_{\boldsymbol{P},\varphi}(\boldsymbol{v}) = \underset{\boldsymbol{u}}{\text{argmin}}\ \frac{1}{2}\|\boldsymbol{u}-\boldsymbol{v}\|_{\boldsymbol{P}}^2 + \varphi(\boldsymbol{v})$$

where $\|\boldsymbol{x}\|_{\boldsymbol{P}}^2 = \langle \boldsymbol{x}, \boldsymbol{P}\boldsymbol{x}\rangle$, and $\langle\cdot,\cdot\rangle$ is the inner product. In the above definition of proximity operator, $\boldsymbol{P}$ is called *preconditioning* matrix [15]. Definition 1 of proximity operator given in Section II-B may be obtained from the above definition if $\boldsymbol{P}$ is the identity matrix. It is observed empirically in [39] that utilizing preconditioning matrix speeds up the convergence of Proximal Alternating Linearized Minimization (PALM).

IV. Forward-Backward Splitting with Smart initialization (FBS4)

In this section, we describe FBS4, the method we propose for constrained (*e.g.* non-negativity) CP decomposition. First, in Section IV-A, the cost function, its solution by means of Forward-Backward Splitting and FBS4 algorithm, based on Algorithm 1 are introduced. Then, some prevalent constraints such as non-negativity, sparsity, and explain FBS4 algorithm for these particular constraints in Section IV-B are discussed. In Section IV-C, the convergence theorem for FBS4 is briefly quoted, whereas the complete convergence analysis can be found in Appendix A.

A. Formulation and algorithm

Consider the N-th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$ of rank R. Assume that $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]$, where $\boldsymbol{\lambda} \in \mathbb{R}_+^R$ and $\boldsymbol{A}^{(n)} \in \mathbb{R}^{I_n \times R}$. A general problem of constrained CP decomposition of $\boldsymbol{\mathcal{T}}$ can be formulated as follows:

$$\min_{\boldsymbol{\lambda}, \boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(N)}]\!]\|_F^2 \qquad (6)$$
$$\text{s.t. } \mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)}),\ 1 \leq n \leq N$$

where $\|\cdot\|_F$ is the Frobenius norm and $\mathcal{C}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}), \mathcal{C}_{\boldsymbol{A}^{(n)}}(\boldsymbol{A}^{(n)})$ are, respectively the constraints over vector $\boldsymbol{\lambda}$ (including abovementioned constraint, *i.e.* $\boldsymbol{\lambda} \in \mathbb{R}_+^R$, such as belonging to a simplex set) and matrix $\boldsymbol{A}^{(n)}$.

As mentioned in Section III-A, a common strategy is to solve (6) via AO. Moreover, a constrained optimization can be transformed into an unconstrained one by adding the indicator function of the constraint set to the cost function. To be more precise, at the n^{th} step of AO for solving (6), we have:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \big\|\boldsymbol{\mathcal{T}} - [\![\boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(n)}, \ldots, \boldsymbol{A}^{(N)}]\!]\big\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}), \qquad (7)$$

where $i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)})$ is defined as follows:

$$i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}) = \begin{cases} 0 & if \quad \boldsymbol{A}^{(n)} \in \mathcal{C}_{\boldsymbol{A}^{(n)}} \\ \infty & if \quad \boldsymbol{A}^{(n)} \notin \mathcal{C}_{\boldsymbol{A}^{(n)}} \end{cases}.$$

The vector $\boldsymbol{\lambda}$ is omitted in (7), since it can be calculated by normalizing loading matrices ($\boldsymbol{A}^{(n)}$). Otherwise, vector $\boldsymbol{\lambda}$ as one of the unknown variables can be optimized in one of the steps of AO.

Define $\boldsymbol{W} \triangleq (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$. Then by the mode-$n$ unfolding of (7), we have:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2} \|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)} \boldsymbol{W}\|_F^2 + i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}). \qquad (8)$$

Note that $i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)})$ is a lower semi-continuous function (see Appendix A) and $\frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$ is finite valued, differentiable and β-Lipschitz continuous gradient where $\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$ denotes the spectral norm[1] of matrix $\boldsymbol{W}\boldsymbol{W}^T$ (see Appendix A for calculations[2]). Since, $\boldsymbol{W}\boldsymbol{W}^T$ is a symmetric matrix, its singular values are the squared of those of $\boldsymbol{W}$. Moreover, the cost function in (8) is proper, lower semi-continuous with KL property [29]. Consequently, all the required assumptions of Theorem 1 are satisfied for (8), and according to this theorem, the minimizer of (8) is the convergence point of the following fixed point equation:

$$\boldsymbol{A}^{(n)} = \text{prox}_{\gamma i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}} \{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T})\}.$$

Since the proximity operator of $\gamma i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}$ is the projection onto $\mathcal{C}_{\boldsymbol{A}^{(n)}}$, we have:

$$\boldsymbol{A}^{(n)} = \text{proj}_{\mathcal{C}_{\boldsymbol{A}^{(n)}}} \{\boldsymbol{A}^{(n)} - \gamma(\boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T - \boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T})\}. \quad (9)$$

Algorithm 3 Algorithm of FBS4

Input: $\boldsymbol{\mathcal{T}}$, $\mathcal{C}_{\boldsymbol{A}^{(n)}}$, initial $\boldsymbol{A}_0^{(n)},\ n \in [1, \ldots, N]$
Output: Estimated $\boldsymbol{A}^{(n)},\ n \in [1, \ldots, N]$

1: **repeat**
2: **for** $n = 1, 2, \ldots, N$ **do**
3: $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$
4: $\beta = \{\max(\text{singular value}(\boldsymbol{W}))\}^2$
5: set $\gamma = \frac{1}{\beta}$ and $\alpha = 1$
6: **for** $k = 0, 1, 2, \ldots$ **do**
7: $\boldsymbol{Y} = \boldsymbol{A}_k^{(n)} - \gamma(\boldsymbol{A}_k^{(n)}(\boldsymbol{W}\boldsymbol{W}^T) - \boldsymbol{\mathcal{T}}^{(n)}\boldsymbol{W}^T)$
8: $\boldsymbol{A}_{k+1}^{(n)} = \boldsymbol{A}_k^{(n)} + \alpha(\text{proj}_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{Y}) - \boldsymbol{A}_k^{(n)})$
9: **end for**
10: **end for**
11: **until** some termination criterion

FBS4 is described in Algorithm 3. Although, Algorithm 3 is based upon Algorithm 1, we experimentally find the proper values of some parameters of Algorithm 1 such as γ_k and α_k to obtain best results. As it is expressed in Algorithm 3, we ignore ϵ and fix the value of $\gamma_k = \frac{1}{\beta}$ in all iterations over k. In addition, we remove the effect of α_k by setting it to 1 in linear updating of the estimated variable (Line 8 in Algorithm 3).

We now review the main theoretical and practical advantages of Algorithm 3. Firstly, it can be applied even on non-convex and non-smooth constraints such as cardinality [29], and this does not affect its convergence. Secondly, it does not require any critical setting of the parameters. The only parameters to be set are γ and α, where the suggested values ($\gamma = \frac{1}{\beta}$ and $\alpha = 1$) in Algorithm 3 are almost always suitable. Thirdly, compared to state-of-the-art methods such as APG and BC-VMFB, FBS4 is easy enough to understand and implement. Fourthly, contrary to APG or BC-VMFB, FBS4 works with variables in matrix form, so there is no need to vectorize loading matrices. This brings an advantage in working with large dimension tensors. Fifthly, FBS4 is smartly initialized, which permits a faster convergence. Last but not least, as it is developed in the next section, FBS4 can be adapted to many different constraints.

B. Some constraints

As mentioned before, the FBS4 is not limited to any particular constraint. Any constraint whose indicator function satisfies the assumptions of Theorem 1 can be considered. In

[1]Spectral norm of a matrix is defined as its maximum singular value [40].

[2]It is mentioned in [19] that the Lipschitz constant is $\|\boldsymbol{W}\boldsymbol{W}^T\|_F$, but as we prove in Appendix A, the Lipschitz constant is in fact the $\|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma$.

this section, the required modifications of Algorithm 3 are described, for some widespread constraints.

1) Non-negativity: Non-negativity is one of the most common constraints in the literature. In many applications such as image processing [10] or chemometrics [41], the non-negativity of loading matrices is essential and helpful in the performance of tensor decomposition. Non-negative constraints of (6) are expressed as: $\boldsymbol{A}^{(n)} \in \mathbb{R}_+^{I_n \times R}$ and $\boldsymbol{\lambda} \in \mathbb{R}_+^R$.

The projection to the non-negative orthant is done with the $\max$ operator, thereby line 8 of Algorithm 3 would be $\boldsymbol{A}_{k+1}^{(n)} = \boldsymbol{A}_k^{(n)} + \alpha(\max(\boldsymbol{Y}, 0) - \boldsymbol{A}_k^{(n)})$. In other words, the proximity operator of the non-negative constraint retains the non-negative elements of the array and replaces its negative values with zero.

2) Simplex set: In applications involving probability estimation or distribution approximation, the simplex (or probability simplex) constraint unavoidably appears [7], [11]. A vector $\boldsymbol{x} \in \mathbb{R}^P$ belongs to the simplex set $\boldsymbol{\mathcal{S}}^P$ if $\{\boldsymbol{x}^T\mathbf{1} = 1,\ x_i \geq 0,\ i \in [1, \ldots, P]\}$. This constraint can be written as $\{\boldsymbol{A}^{(n)}(:, j) \in \boldsymbol{\mathcal{S}}^{I_n},\ j \in [1, \ldots, R],\ n \in [1, \ldots, N]\}$ and $\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}^R$. An algorithm for projecting a vector onto the simplex set is proposed in [42]. Therefore, the line 8 of Algorithm 3 would include a projection algorithm to the simplex, whose input is $\boldsymbol{Y}$.

An efficient way to apply the simplex constraint to all the columns of loading matrices $\boldsymbol{A}^{(n)}, n \in [1, \ldots, N]$ and to the coefficient vector $\boldsymbol{\lambda}$ is to first combine $\boldsymbol{\lambda}$ with one of the loading matrices, let us say $\boldsymbol{A}^{(N)}$. By combination, we mean $\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)} = \boldsymbol{A}^{(N)}\text{Diag}(\boldsymbol{\lambda})$, where $\text{Diag}(\boldsymbol{\lambda})$ is a diagonal matrix containing $\boldsymbol{\lambda}$ on its diagonal. Then, the simplex constraint may be applied to every column of every matrix $\boldsymbol{A}^{(n)}, n \in [1, \ldots, N-1]$, and only to the vectorization of matrix $\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)}$. Each entry λ_r of $\boldsymbol{\lambda}$ is eventually obtained by normalizing the rth column of matrix $\boldsymbol{A}_{\boldsymbol{\lambda}}^{(N)}$ with respect to ℓ_1 norm, and the resulting normalized matrix yields an estimation of $\boldsymbol{A}^{(N)}$. It can then be proved easily that estimated $\boldsymbol{\lambda}$ and every column of loading matrix $\boldsymbol{A}^{(N)}$ indeed lie in the simplex.

3) Sparsity with ℓ_0 pseudo-norm (cardinality): In some applications such as two dimensional dictionary learning, a constraint on the number of non-zero elements of loading matrices (or cardinality) is needed [12]. This constraint is measured with ℓ_0 pseudo-norm and known as *sparsity* in compressive sensing [43].

In penalized form, an ℓ_0 pseudo-norm term is added to the cost function [44]. Although ℓ_0 is non-convex, the resulted cost function can be solved by means of Theorem 1. In order to use Algorithm 3 with a cardinality constraint, the proximity operator of ℓ_0 should be calculated. It is mentioned in [20], [29] that the proximity operator of ℓ_0 is a function called *hard-thresholding*, defined as follows:

$$f_\eta^h(y) \triangleq \begin{cases} y & |y| > \eta \\ 0 & |y| \leq \eta \end{cases}.$$

With this in mind, the line 8 of Algorithm 3 includes applying hard-thresholding on each element of $\boldsymbol{Y}$ $([f^h_{\sqrt{2\gamma}}(Y_{(i,j)})]_{(i=1,j=1)}^{(I_n,R)})$.

4) Sparsity with ℓ_1 norm: A common and convex approximation of ℓ_0 pseudo-norm is ℓ_1 norm. The resulting cost function obtained by replacing ℓ_0 with ℓ_1 norm is called *LASSO regression* [43], [44]. In [20], it is stated that according to (3), the proximity operator of $\ell_1(\boldsymbol{x})$ is $f^s_{\sqrt{2\gamma}}$,where f_η^s is called *soft-thresholding*, and is defined as:

$$f_\eta^s(y) \triangleq \begin{cases} y - \eta & y > \eta \\ 0 & |y| \leq \eta \\ y + \eta & y < -\eta \end{cases}.$$

C. Convergence guarantee

The overall convergence of Algorithm 3 has been proved in [45] (convergence of Proximal Alternating Linearized Minimization (PALM)). Based on Theorem 1, the convergence of each step of AO is next discussed.

The convergence of Theorem 1 is studied in [14], [29], [46]. In [14], the convergence analysis is proved provided that the cost function is convex and continuous. Since we look for weaker assumptions, we ignore the proof of [14] in this section.

In this section, we restate the convergence theorem of Forward-Backward Splitting from [29], [46], which is not only applicable for continuous and convex cost functions, but is also usable for non-smooth and non-convex functions (or non-convex set of constraint).

Theorem 2 (Convergence of Forward-Backward Splitting [29], [46])**.** *Suppose $f : \mathbb{R}^N \mapsto \mathbb{R} \cup \{+\infty\}$ is a proper lower semi-continuous function which has KL property and is bounded from below. Assume that f can be spilt into two parts as $f = h + g$, where g is lower semi-continuous and $h : \mathbb{R}^N \mapsto \mathbb{R}$ is a finite valued, differentiable function with a β-Lipschitz continuous gradient.*

If the sequence generated by Algorithm 3 is bounded, then this sequence will converge to a critical point of f. In addition, by choosing $\gamma \in [0, \frac{1}{\beta}]$, the values of the cost function are not increasing.

Proof. See Appendix A. Note that the proof of Theorem 2 is provided by means of [29], [46] but with much more details, which makes it significantly easier to follow. □

V. Experimental results

In this section, experimental results obtained by decompositions of third order tensors are reported, either generated synthetically or coming from real data sets.

We compare FBS4 with those described in Section III, namely AO-ADMM [13], APG [14], FastNTF-APG [19] and BC-VMFB [15][3]. Algorithms are first tested on artificially generated non-negative third order tensors, constructed from factors belonging to the simplex or not. In some cases, the tensors are corrupted with additive noise. Next, experiments on real data sets are described, namely colored images and amino acid fluorescence data sets [3], under non-negativity constraint.

[3]We would like to thank the corresponding authors of FastNTF-APG [19] and BC-VMFB [15], Guoxu Zhou and Caroline Chaux, respectively, who sent us the MATLAB codes of their methods. The MATLAB codes of AO-ADMM [13] and APG [14] are made available by the authors at [47] and [48], respectively. Therefore, the original codes of authors have been used to obtain the results reported in all figures of this section.

All computer experiments reported in this section have been executed on a laptop with a processor of 3.1 GHz Intel Core i5, 16 GB RAM, running macOS Mojav and MATLAB 2019a.

A. Synthetic data

Algorithms are examined on three kinds of synthetic third order tensors:

1- Small tensors of size $10 \times 10 \times 10$ and of rank $R = 6$,
2- Medium tensors of size $50 \times 50 \times 50$ and of rank $R = 10$,
3- Large tensors of size $200 \times 200 \times 200$ and of rank $R = 20$.

According to the considered constraints (non-negativity or simplex), we generate randomly (uniform distribution) loading matrices and coefficient vector. Then, the noiseless tensor $\boldsymbol{\mathcal{T}}_o$ is computed via (1). In order to work in a noisy context, a noise tensor is added, $\boldsymbol{\mathcal{T}}_n$, with Gaussian i.i.d. entries with zero mean and unit variance, of same size as $\boldsymbol{\mathcal{T}}_o$, and weighted by parameter σ. The variance σ^2 of the additive noise is adjusted such that we reach a desired Signal to Noise Ratio (SNR) according to the following relation:

$$\text{SNR} = 10 \log_{10} \frac{\frac{1}{M} \sum_{i,j,k} \boldsymbol{\mathcal{T}}_o(i,j,k)^2}{\frac{1}{M} \sum_{i,j,k} \sigma^2 \boldsymbol{\mathcal{T}}_n(i,j,k)^2},$$

where M is the total number of elements of tensors $\boldsymbol{\mathcal{T}}_o$ or $\boldsymbol{\mathcal{T}}_n$.

Let us discuss two technical points about the practical implementation:

* In order to compare the performances of algorithms fairly, we force all the algorithms to run for an identical time.
* Instead of random initialization, which is usual in the literature, we have used a *smart initialization* by means of [49]. To be more precise, first, we decompose the desired tensor thanks to the closed-form described in [49]. Then, we use the resulting loading matrices as initialization for Algorithm 3. In some cases (*e.g.* large tensors), we use a few iterations of AO-ADMM as a smart initialization.

*1) **Non-negative constraint**:* The entries of loading matrices and coefficient vector $\boldsymbol{\lambda}$ are independently drawn from a uniform distribution (without column normalization). Tensors are then built using (1). The size of generated matrices are 10×6, 50×10 and 200×20 for small, medium and large tensors, respectively. The size of the coefficient vector is equal to the rank of tensors, R, in all cases (R being equal to either 6, 10 or 20).

Denote by $\boldsymbol{\mathcal{T}}$ the desired tensor to be decomposed, $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_o + \sigma \boldsymbol{\mathcal{T}}_n$. After decomposing $\boldsymbol{\mathcal{T}}$, the estimation of $\boldsymbol{\mathcal{T}}_o$ can be calculated through (1), as a rank-R approximation, which we call $\hat{\boldsymbol{\mathcal{T}}}$. The relative reconstruction error is computed as follows:

$$\epsilon(\hat{\boldsymbol{\mathcal{T}}}) = \frac{\|\hat{\boldsymbol{\mathcal{T}}} - \boldsymbol{\mathcal{T}}_o\|_F^2}{\|\boldsymbol{\mathcal{T}}_o\|_F^2} \qquad (10)$$

Figure 1 (Left) shows the relative reconstruction error of the noiseless small tensor (of size $10 \times 10 \times 10, R = 6$). Figure. 1 (Right) relates to the relative reconstruction error when the small tensor to be decomposed is noisy, and SNR $= 20$dB. With non-negative constraints, in order to be able to track the causes of performance differences, all the algorithms are initialized smartly with the method of [49].

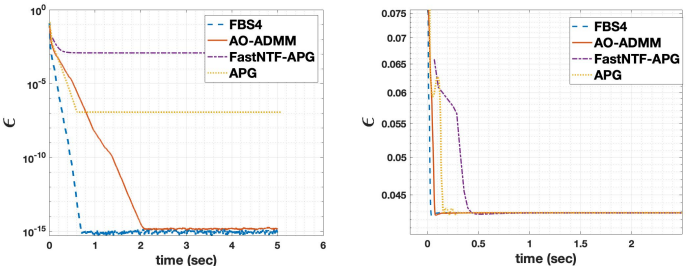

Fig. 1: Left: Small tensor of size $10 \times 10 \times 10, R = 6$, noiseless, Right: Small tensor of size $10 \times 10 \times 10, R = 6$, SNR $= 20$dB

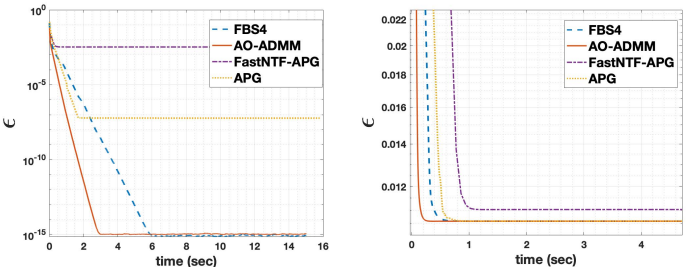

Fig. 2: Left: Medium tensor of size $50 \times 50 \times 50, R = 10$, noiseless, Right: Medium tensor of size $50 \times 50 \times 50, R = 10$, SNR $= 20$dB

Most of the methods have succeeded to achieve acceptable relative errors as demonstrated in Fig. 1 (Left). However, BC-VMFB, in spite of our effort to adjust its parameters properly and our consulting the corresponding author about parameters, results in a relative error around 0.2 or 0.3. Therefore, in order to be able to distinguish the differences between errors of other methods, we have been forced to omit BC-VMFB from the comparisons.

As it is shown in Fig. 1 (Left), FBS4 and AO-ADMM outperform other methods in decomposing noiseless small tensors. Moreover, FBS4 converges faster than AO-ADMM. Although APG and FastNTF-APG are based on Forward-Backward Splitting as FBS4, their performance is weaker than FBS4. This is because their algorithm are based on Algorithm 2, while FBS4 is based on Algorithm 1. The different behaviors of Algorithm 1 and Algorithm 2 have been explained in [17]. In addition, since FastNTF-APG tries to decompose a low-rank approximation of the desired tensor, it can be expected that FastNTF-APG returns less accurate decomposition than FBS4 and APG. For noisy small tensor, as it is shown in Fig. 1 (Left), all the methods perform almost the same, but FBS4 converges a bit faster than others.

Figure 2 (Left) shows the result obtained after decomposing and reconstructing a noiseless medium tensor of size $50 \times 50 \times 50, R = 10$, and Fig. 2 (Right) depicts the relative reconstruction error after decomposing a noisy medium tensor of size $50 \times 50 \times 50, R = 10$ with SNR $= 20$dB. In these experiments, as before, all the algorithms are initialized smartly with the method of [49] to have fair comparisons. The results for medium tensors (Fig. 2 (Left and Right)) are almost the same as those of small tensors (Fig. 1 (Left and Right))

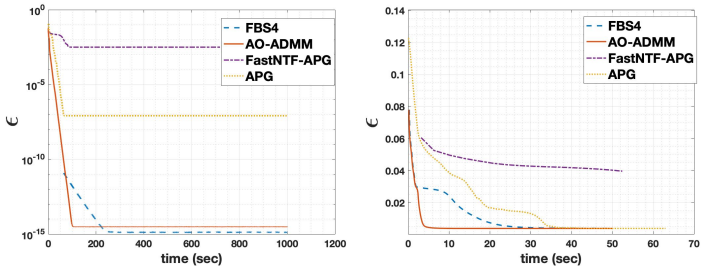

Fig. 3: Left: Large tensor of size $200 \times 200 \times 200, R = 20$, noiseless, Right: Large tensor of size $200 \times 200 \times 200, R = 20$, SNR $= 20$dB

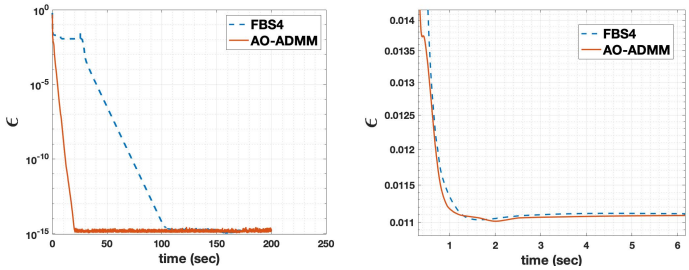

Fig. 4: Left: The reconstruction error $(50 \times 50 \times 50, R = 10$, noiseless), Right: The reconstruction error $(50 \times 50 \times 50, R = 10,$ SNR $= 20$dB)

except that AO-ADMM converges faster than FBS4.

Figures 3 (Left and Right) are related to large tensors of size $200 \times 200 \times 200, R = 20$, in which FBS4 is initialized with 1000 iterations and two seconds of running AO-ADMM, respectively. All the other algorithms initialized smartly with the method of [49]. Comparing Fig. 3 (Left) to the results for noiseless medium tensors, FBS4 performs a bit better than AO-ADMM, but instead AO-ADMM converges faster than FBS4. Moreover, FBS4 and AO-ADMM outperform the other two methods, APG and FastNTF-APG. However, in noisy context, almost all the algorithms achieve the same level of relative reconstruction error and AO-ADMM converges faster than the others. Parameter γ in Algorithm 3 is set to $\frac{1.9}{\beta}$ for Fig. 3 (Right).

*2) **Simplex constraint**:* The entries of loading matrices and coefficient vector $\boldsymbol{\lambda}$ are independently drawn from a uniform distribution, and columns are normalized with respect to the ℓ_1 norm. Then tensor $\boldsymbol{\mathcal{T}}_o$ is computed via (1). For space reasons, we report in this section only the simulation results for medium tensors of size $50 \times 50 \times 50$, $R = 10$ under simplex constraint. The size of generated matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ are 50×10 and the size of the coefficient vector $\boldsymbol{\lambda}$ is R.

To the best of our knowledge, the only algorithm in the literature that considers simplex constraints for tensor decomposition is AO-ADMM [13].

The relative reconstruction error is computed by (10). Similarly, the relative error of the estimation of loading matrices and coefficient vector can be computed. If we denote the first mode loading matrix and its estimation by $\boldsymbol{A}$ and $\hat{\boldsymbol{A}}$, respectively, the relative error of estimation $\boldsymbol{A}$ is as follows:

$$\epsilon(\hat{\boldsymbol{A}}) = \frac{\|\hat{\boldsymbol{A}} - \boldsymbol{A}\|_F^2}{\|\boldsymbol{A}\|_F^2}.$$

The relative error of other loading matrices and coefficient vector is computed in the same way.

It is hard to assess the relative error made on loading matrices, because of scaling and permutation ambiguities of tensor decomposition [1]. Under the simplex constraint, we resolve scaling ambiguity by normalizing the columns of loading matrices and coefficient vector with respect to the ℓ_1 norm. In addition, we use the Hungarian algorithm [50], [51] to fix the permutation ambiguity.

Figures 4-8 (Left) correspond to the performance of FBS4 and AO-ADMM for the decomposition of a noiseless medium tensor of size $50 \times 50 \times 50, R = 10$ under the simplex constraint. As shown in Figs. 4-8 (Left), AO-ADMM converges

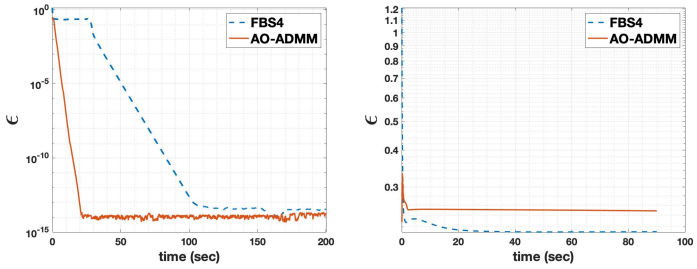

Fig. 5: Left: The error of first mode loading matrix $\boldsymbol{A}$ $(50 \times 50 \times 50, R = 10,$ noiseless), Right: The error of first mode loading matrix $\boldsymbol{A}$ $(50 \times 50 \times 50, R = 10,$ SNR $= 20$dB)

faster than FBS4, but both of these methods achieve almost the same value of relative error.

Figures 4-8 (Right) correspond to the performances of FBS4 and AO-ADMM for the decomposition of a medium tensor of size $50 \times 50 \times 50, R = 10$ with SNR $= 20$dB under the simplex constraint. As shown in Figs. 4-8 (Right), FBS4 performs better than AO-ADMM, for estimating loading matrices and the coefficient vector $\boldsymbol{\lambda}$, although the reconstruction error (Fig. 4, right) achieved by the two algorithms is almost the same. Both algorithms are initialized smartly by the method of [49] in all of the experiments of this section.

B. Real data

In this section, the performances of the algorithms mentioned in Sections III and IV are evaluated on two real data sets, namely *Amino acids* and *Color images*.

The Amino acids data set consists of five samples, each of which contains different amounts of three amino acids, namely Tryptophan (Trp), Tyrosine (Tyr), Phenylalanine (Phe) [52]. In fluorescence spectrometry, the excitation spectrum (the light that is absorbed by the sample) and the emission spectrum (the light emitted by the sample) can be measured simultaneously. This kind of measurement ends in a specific matrix known as Excitation Emission Matrix (EEM) [3].

In the well-known Amino acids data set, the excitation and emission spectrum of EEM are measured, respectively over wave-length range of $(240$ nm $- 300$ nm$)$ and $(250$ nm $- 450$ nm$)$, every nanometer. Therefore, the Amino acids data set is a non-negative real tensor of size $201 \times 61 \times 5$. Although there are some small negative values in the data, the tensor is supposed to be non-negative, since it is constructed from non-negative values which are the intensity measurements of EEM [52].

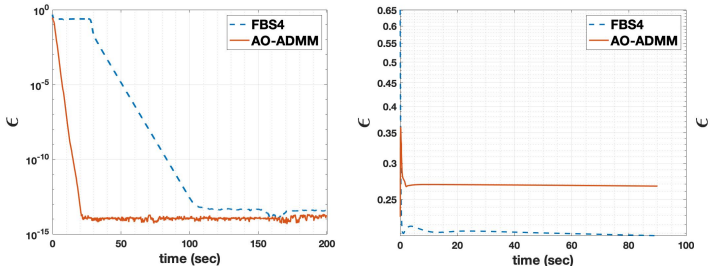

Fig. 6: Left: The error of second mode loading matrix $\boldsymbol{B}$ ($50\times50\times50, R=10$, noiseless), Right: The error of second mode loading matrix $\boldsymbol{B}$ ($50\times50\times50, R=10, \text{SNR}=20\text{dB}$)

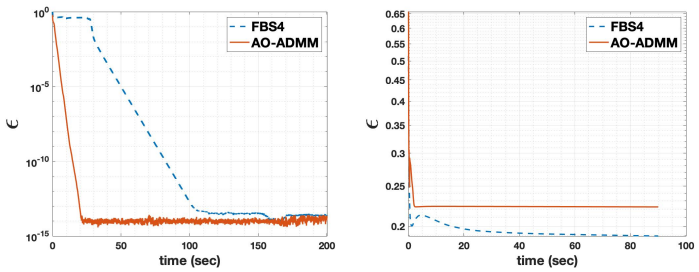

Fig. 7: Left: The error of third mode loading matrix $\boldsymbol{C}$ ($50\times50\times50, R=10$, noiseless), Right: The error of third mode loading matrix $\boldsymbol{C}$ ($50\times50\times50, R=10, \text{SNR}=20\text{dB}$)

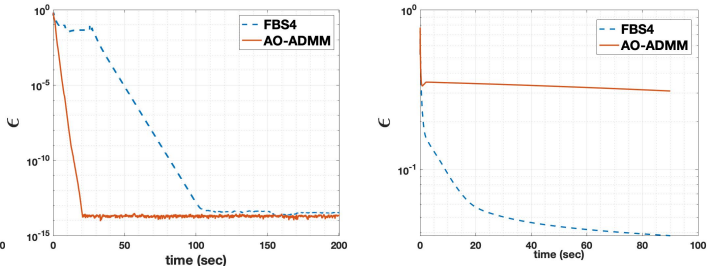

Fig. 8: Left: The error of the coefficient vector $\boldsymbol{\lambda}$ ($50\times50\times50, R=10$, noiseless), Right: The error of the coefficient vector $\boldsymbol{\lambda}$ ($50\times50\times50, R=10, \text{SNR}=20\text{dB}$)

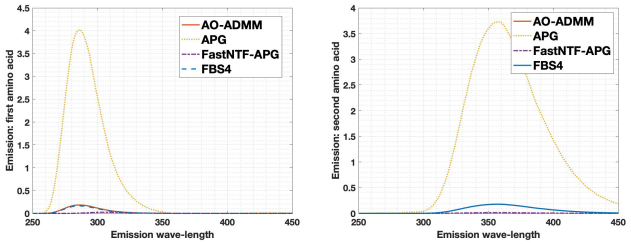

Fig. 9: Left: Estimated emission of first amino acid, Right: Estimated emission of second amino acid

The other real data set contains color images, which are stored in arrays of size height $\times$ width $\times 3$, where "height" and "width" correspond to the size of RGB images in pixels. As the intensities are non-negative values, the tensor of a color image is non-negative.

*1) **Amino acid**:* The Amino acids data set is available on the web [53]. It is discussed in [3] that these data is describable with CP of rank $R=3$. By decomposing Amino acids tensor, three loading matrices are obtained: Emission of size 201×3, Excitation of size 61×3, and Amount matrix of size 5×3.

We have applied Several non-negative CP decomposition algorithms of Sections III and IV with the smart initialization provided by [49] on Amino acid data. Then, each column of estimated Emission/Excitation matrix is plotted versus emission/excitation wave-length (Figs. 9-11).

By comparing the known fluorescence spectrometry results [3] of Trp, Tyr and Phe with Figs. 9-11, it can be interpreted that FBS4 and AO-ADMM result in the closest diagrams to the expected ones. Therefore, these two methods are the most reliable ones.

*2) **Image reconstruction**:* Color images are the most accessible examples of real data tensors. In this section, algorithms of Section III and IV are applied on colored version of Lena image (Fig. 12 (Right)) under non-negativity constraint and with the smart initialization provided by [49].

In this experiment, the performances of algorithms are compared using Peak Signal to Noise Ratio (PSNR), which is computed based on the original image and its noisy version (or its reconstruction): Given an original image, Im_{org} of size $P\times Q$ and its reconstruction, Im_{rec}, PSNR is defined as:

$$\text{PSNR(dB)} = 10\log_{10}\frac{\text{Max}_I^2}{\text{MSE}},$$
$$\text{MSE} = \frac{\sum_{p=1}^{P}\sum_{p=1}^{P}(\text{Im}_{org}(p,q)-\text{Im}_{rec}(p,q))^2}{PQ},$$

where Max_I is the maximum possible pixel value, which is 255 in this experiment over Lena image.

Unlike Amino acids data set, the proper rank of color images as a tensor is not known. This makes it hard to fit the best decomposition that results in the minimum level of relative reconstruction error. However, in Fig. 12 (Left), a range of ranks is considered, and for each of them, the PSNR of reconstructed images generated by the algorithms is compared. As shown in Fig. 12 (Left), the larger the chosen rank, the higher the resulting PSNR. Nevertheless, increasing rank of CP decomposition algorithms is very costly in terms of computation complexity and execution time. Therefore, there is a trade-off between increasing rank of decomposition (which results in higher PSNR) and running time.

In order to compare the algorithms more precisely, the values of the PSNR corresponding to rank $R=20$ are available in Table I. In addition, the reconstructed images by FBS4 algorithm and FastNTF-APG are depicted in Fig. 13. It can be concluded from Table I and Fig. 13 that by choosing rank $R=20$, FBS4 and FastNTF-APG result in the best and the worst PSNR, respectively.

This experiment can be viewed as an image compression, since from visual perspective, the reconstructed image has the same quality as the original image. In other words, instead of storing $128\times128\times3=49152$ pixel values of the colored Lena image, one can reconstruct it with an acceptable PSNR from the three factor matrices, that is, from

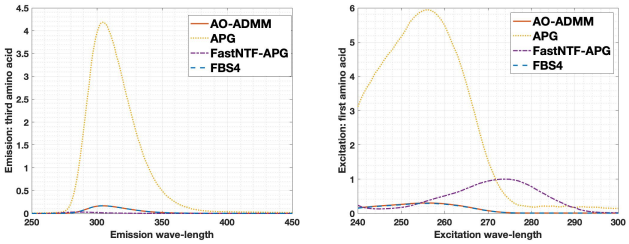

Fig. 10: Left: Estimated emission of third amino acid, Right: Estimated excitation of first amino acid

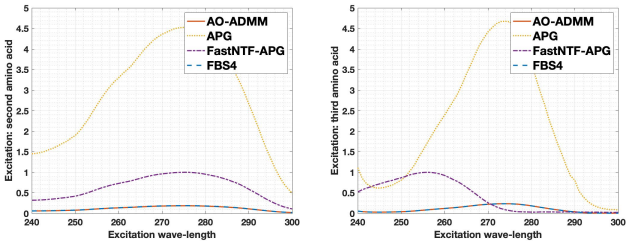

Fig. 11: Left: Estimated excitation of second amino acid, Right: Estimated excitation of third amino acid

TABLE I: PSNR of reconstructed Lena image for rank $R = 20$

Method	AO-ADMM	APG	FastNTF-APG	FBS4
PSNR (dB)	29.92	29.81	29.00	**29.94**

$2 \times (128 \times 20) + 3 \times 20 = 5180$ elements, which represents a compression around 89.46%.

VI. Conclusion

In this paper, an algorithm, FBS4, was proposed based on ALS for constrained tensor decomposition, utilizing a particular proximal method called Forward-Backward Splitting [17]. A convergence analysis was provided for FBS4 which, unlike the convergence of AO-ADMM [13], guarantees the convergence towards a critical point for both convex and non-convex constraints such as sparsity.

We compared practically FBS4 algorithm to AO-ADMM, APG [14], FastNTF-APG [19] and BC-VMFB [15]. The computer experiments showed that FBS4 algorithm performs better than proximal based methods (APG, FastNTF-APG and BC-VMFB) especially in noiseless context, and it is also competitive with AO-ADMM. Although, the experiments had been dedicated to non-negativity and simplex set constraints, FBS4 can be easily adapted for managing many different constraints such as cardinality. In fact, sparsity was not explicitly addressed in computer experiments of the paper, due to lack of space. However, it has been described in Sec IV how FBS4 can be applied with such a constraint.

Appendix A
Convergence Analysis of FBS4

Let us define $h(\boldsymbol{A}^{(n)})$ and $g(\boldsymbol{A}^{(n)})$ as follows:

$$h(\boldsymbol{A}^{(n)}) \triangleq \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2$$
$$g(\boldsymbol{A}^{(n)}) \triangleq i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}^{(n)}),$$

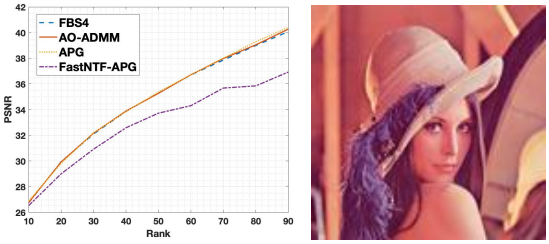

Fig. 12: Left: Reconstruction performances of algorithms versus each of chosen rank, Right: Original image ($128 \times 128 \times 3$)

Fig. 13: Left: Reconstructed image by FastNTF-APG with PSNR $= 29.00$dB (chosen rank $R = 20$), Right: Reconstructed image by FBS4 with PSNR $= 29.94$dB (chosen rank $R = 20$)

where $\boldsymbol{W} = (\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})^T$.

Since $h(\boldsymbol{A}^{(n)})$ has quadratic form, the "Lipschitz" constant of its gradient can be calculated. The gradient of $h(\boldsymbol{A}^{(n)})$ is computed as follows:

$$\begin{aligned} h(\boldsymbol{A}^{(n)}) &= \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W}\|_F^2 \\ &= \frac{1}{2}\,\mathrm{trace}\{(\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W})^T(\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}^{(n)}\boldsymbol{W})\} \\ &\Rightarrow \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) = -\boldsymbol{W}\boldsymbol{\mathcal{T}}^{(n)^T} + \boldsymbol{A}^{(n)}\boldsymbol{W}\boldsymbol{W}^T \end{aligned}$$

In calculating the gradient, the following relations have been used:

$$\nabla_{\boldsymbol{A}}\,\mathrm{trace}\{\boldsymbol{A}\boldsymbol{B}\} = \boldsymbol{B}^T,$$
$$\nabla_{\boldsymbol{A}}\,\mathrm{trace}\{\boldsymbol{A}\boldsymbol{B}\boldsymbol{A}^T\boldsymbol{C}\} = \boldsymbol{C}^T\boldsymbol{A}\boldsymbol{B}^T + \boldsymbol{C}\boldsymbol{A}\boldsymbol{B}.$$

Now, the Lipschitz constant of $\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)})$ can be calculated as follows: $h(\boldsymbol{A}^{(n)})$ is β-Lipschitz gradient, *i.e.*,

$$\begin{aligned} &\|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y})\|_F \le \beta\|\boldsymbol{X} - \boldsymbol{Y}\|_F \\ &\Rightarrow \beta = \max \frac{\|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y})\|_F}{\|\boldsymbol{X} - \boldsymbol{Y}\|_F} \\ &\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{X}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{Y}) = (\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T \\ &\Rightarrow \beta = \max \frac{\|(\boldsymbol{X} - \boldsymbol{Y})\boldsymbol{W}\boldsymbol{W}^T\|_F}{\|\boldsymbol{X} - \boldsymbol{Y}\|_F}. \end{aligned}$$

Note that the definition of "spectral norm" of a matrix is:

$$\|\boldsymbol{A}\|_\sigma \triangleq \max \frac{\|\boldsymbol{A}\boldsymbol{X}\|_F}{\|\boldsymbol{X}\|_F},$$

which is equal to the maximum singular value of $\boldsymbol{A}$. Therefore, we have:

$$\beta = \|\boldsymbol{W}\boldsymbol{W}^T\|_\sigma.$$

The problem is $\min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)})$. By means of a well-known lemma, called "Descent Lemma" [29], we want to

show that instead of $\min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)})+g(\boldsymbol{A}^{(n)})$, the following problem can be solved:

$$\boldsymbol{A}_{k+1}^{(n)} = \underset{\boldsymbol{A}^{(n)}}{\operatorname{argmin}}\, h(\boldsymbol{A}_k^{(n)})+ \operatorname{trace}\{(\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)})\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)^T}\} + \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}^{(n)}), \tag{11}$$

where $\boldsymbol{A}_k^{(n)}$ is the k^{th} estimation of $\boldsymbol{A}^{(n)}$ and $\mu \in (0, \frac{1}{\beta})$. In [29], "Descent Lemma" is mentioned as follows:

Lemma 1 (Descent Lemma [29] (Lemma 3.1))**.** *let $f : \mathbb{R}^d \to \mathbb{R}$ be a continuously differentiable function with gradient ∇f assumed β-Lipschitz continuous. Then,*

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{\beta}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2,\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$$

By considering $\mu \in (0, \frac{1}{\beta})$, obviously, the following equation resulted from "Descent Lemma" can be written:

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{1}{2\mu}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2. \tag{12}$$

In order to show that minimizing (11) is equivalent to $\min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)})$, it is needed to rewrite (12) for matrix variables, $\boldsymbol{A}^{(n)}$ and $\boldsymbol{A}_k^{(n)}$, instead of $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively:

$$h(\boldsymbol{A}^{(n)}) \leq h(\boldsymbol{A}_k^{(n)}) + \langle \nabla h(\boldsymbol{A}_k^{(n)}), \boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\rangle + \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 \tag{13}$$

Note that Euclidean norm is equal to Frobenius norm for matrix variables, and inner product of matrices $\langle \nabla h(\boldsymbol{A}_k^{(n)}), \boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\rangle$ is equal to $\operatorname{trace}\{(\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)})(\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}))^T\}$. By adding $g(\boldsymbol{A}^{(n)})$ to the both sides of (13), we have:

$$h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)}) \leq h(\boldsymbol{A}_k^{(n)}) + \langle \nabla h(\boldsymbol{A}_k^{(n)}), \boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\rangle + \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}^{(n)}) \tag{14}$$

It is obvious that instead of $\min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)})$, the right hand side of (14) can be minimized equivalently. It is good to see that how minimizing (14) is equal to standard form of proximal problems:

$$\begin{aligned}
&\min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)}) \\
&\quad\equiv \min_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}) + \langle \nabla h(\boldsymbol{A}_k^{(n)}), \boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\rangle \\
&\quad+ \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}^{(n)}) \\
&\quad\equiv \min_{\boldsymbol{A}^{(n)}} \langle \nabla h(\boldsymbol{A}_k^{(n)}), \boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\rangle \\
&\quad+ \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}^{(n)}).
\end{aligned} \tag{15}$$

By simple calculations, it can be seen that (15) is equal to:

$$\min_{\boldsymbol{A}^{(n)}} \frac{1}{2}\|\boldsymbol{A}^{(n)} - (\boldsymbol{A}_k^{(n)} - \mu\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}))\|_F^2 + \mu g(\boldsymbol{A}^{(n)}). \tag{16}$$

Let us review the definition of the standard form of a proximal problem:

$$\operatorname{prox}_{\mu g}(\boldsymbol{x}) = \underset{y}{\operatorname{argmin}}\, \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \mu g(\boldsymbol{y}).$$

So, (16) can be rewritten as:

$$\boldsymbol{A}_{k+1}^{(n)} = \operatorname{prox}_{\mu g}(\boldsymbol{A}_k^{(n)} - \mu\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)})).$$

Let us go back to (15), which can be rewritten as:

$$\boldsymbol{A}_{k+1}^{(n)} = \underset{\boldsymbol{A}^{(n)}}{\operatorname{argmin}} \operatorname{trace}\{(\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)})(\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}))^T\} + \frac{1}{2\mu}\|\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}^{(n)}). \tag{17}$$

Since $\boldsymbol{A}_{k+1}^{(n)}$ is the minimizer of (17), we have:

$$\operatorname{trace}\{(\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)})(\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}))^T\} + \frac{1}{2\mu}\|\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 + g(\boldsymbol{A}_{k+1}^{(n)}) \leq g(\boldsymbol{A}_k^{(n)}). \tag{18}$$

To prove that $\boldsymbol{x}^*$ is a critical point of f, we have to show that:

$$\frac{\partial f}{\partial \boldsymbol{x}}|_{\boldsymbol{x}=\boldsymbol{x}^*} = 0.$$

We call the cost function of (17) "f_e", and according to (17), $\boldsymbol{A}_{k+1}^{(n)}$ is a critical point of f_e. FONC of f_e reveals the following relation:

$$0 \in \frac{\partial f_e}{\partial \boldsymbol{A}^{(n)}}|_{\boldsymbol{A}^{(n)}=\boldsymbol{A}_{k+1}^{(n)}} \Rightarrow$$
$$0 \in \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}) + \frac{1}{\mu}(\boldsymbol{A}^{(n)} - \boldsymbol{A}_k^{(n)}) + \partial g(\boldsymbol{A}^{(n)})|_{\boldsymbol{A}^{(n)}=\boldsymbol{A}_{k+1}^{(n)}}. \tag{19}$$

Note that since g is a non-derivative function, its sub-gradient, ∂g is used. By means of Lemma 1 for $h(\boldsymbol{A}^{(n)})$, we have:

$$h(\boldsymbol{A}_{k+1}^{(n)}) \leq h(\boldsymbol{A}_k^{(n)}) + \operatorname{trace}\{(\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)})\nabla h(\boldsymbol{A}_k^{(n)})^T\} + \frac{\beta}{2}\|\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2. \tag{20}$$

Recall that the cost function in Theorem 1 is $f(\boldsymbol{A}^{(n)}) = h(\boldsymbol{A}^{(n)}) + g(\boldsymbol{A}^{(n)})$. By adding (18) and (20), we have

$$\begin{aligned}
h(\boldsymbol{A}_{k+1}^{(n)}) + g(\boldsymbol{A}_{k+1}^{(n)}) &\leq h(\boldsymbol{A}_k^{(n)}) + g(\boldsymbol{A}_k^{(n)}) \\
&\quad - (\frac{1}{2\mu} - \frac{\beta}{2})\|\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 \\
\Rightarrow f(\boldsymbol{A}_{k+1}^{(n)}) &\leq f(\boldsymbol{A}_k^{(n)}) \\
&\quad - (\frac{1}{2\mu} - \frac{\beta}{2})\|\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2.
\end{aligned} \tag{21}$$

Since $0 \leq \mu \leq \frac{1}{\beta}$, or equivalently $(\frac{1}{2\mu} - \frac{\beta}{2}) \geq 0$, then $f(\boldsymbol{A}_{k+1}^{(n)}) \leq f(\boldsymbol{A}^{(n)}$. On the other hand, the cost function is bounded from below, since it is a summation of the norm of a bounded amount ($h(\boldsymbol{A}_k^{(n)}) = \frac{1}{2}\|\boldsymbol{\mathcal{T}}^{(n)} - \boldsymbol{A}_k^{(n)}\boldsymbol{W}\|_F^2$, $\boldsymbol{A}_k^{(n)}$ is bounded as it is a projection over a bounded space in each iteration) and a projection over a bounded space

$(g(\boldsymbol{A}_k^{(n)}) = i_{\mathcal{C}_{\boldsymbol{A}^{(n)}}}(\boldsymbol{A}_k^{(n)}))$. Therefore, the cost function is decreasing and bounded from below, and consequently it proves that the cost function converges.

From convergence of the cost function, we want to show that $\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)} \to 0$. To this end, sum up (21) for $k = 0, \ldots, \infty$:

$$\sum_{k=0}^{\infty}(\frac{1}{2\mu}-\frac{\beta}{2})\|\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}\|_F^2 \leq f(\boldsymbol{A}_0^{(n)}) - f(\boldsymbol{A}_\infty^{(n)})$$

Since the right hand side is bounded and $(\frac{1}{2\mu} - \frac{\beta}{2}) \geq 0$, then $\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)} \to 0$. Now let us mention the following remark to which we will refer in the sequel:

Remark 1 (Converging subsequence)**.** *Since $\{\boldsymbol{A}_k^{(n)}\}_{k=0}^{\infty}$ is a bounded sequence (as it is a projection over bounded space), therefore according to Bolzano-Weierstrass theorem [54], there exists a converged subsequence like $\{\boldsymbol{A}_{k_j}^{(n)}\}_{j=0}^{\infty}$ which converges to $\boldsymbol{A}^{(n)^*}$.*

Now, it is needed to show that $\boldsymbol{A}^{(n)^*}$ is a critical point of the cost function. Let define:

$$\boldsymbol{U}_j \triangleq \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_j}^{(n)}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_{j-1}}^{(n)}) - \frac{1}{\mu}(\boldsymbol{A}_{k_j}^{(n)} - \boldsymbol{A}_{k_{j-1}}^{(n)}).$$

It can be shown that $\boldsymbol{U}_j \in \frac{\partial f}{\partial \boldsymbol{A}^{(n)}}|_{\boldsymbol{A}_{k_j}^{(n)}}$, because by adding $\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k+1}^{(n)})$ to the two sides of (19), we have:

$$\begin{aligned}\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k+1}^{(n)}) &\in \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k+1}^{(n)}) + \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}) \\ &\quad + \frac{1}{\mu}(\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}) + \partial g(\boldsymbol{A}_{k+1}^{(n)}) \Rightarrow \\ \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k+1}^{(n)}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_k^{(n)}) &- \frac{1}{\mu}(\boldsymbol{A}_{k+1}^{(n)} - \boldsymbol{A}_k^{(n)}) \\ &\in \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k+1}^{(n)}) + \partial g(\boldsymbol{A}_{k+1}^{(n)}).\end{aligned}$$

By considering k as k_{j-1}, we have $\boldsymbol{U}_j \in \frac{\partial f}{\partial \boldsymbol{A}^{(n)}}|_{\boldsymbol{A}_{k_j}^{(n)}}$. Now, we want to show that $\boldsymbol{U}_j \to 0$:

$$\begin{aligned}\|\boldsymbol{U}_j\|_F &= \|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_j}^{(n)}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_{j-1}}^{(n)}) \\ &\quad - \frac{1}{\mu}(\boldsymbol{A}_{k_j}^{(n)} - \boldsymbol{A}_{k_{j-1}}^{(n)})\|_F \Rightarrow \\ \|\boldsymbol{U}_j\|_F &\overset{a}{\leq} \|\nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_j}^{(n)}) - \nabla_{\boldsymbol{A}^{(n)}} h(\boldsymbol{A}_{k_{j-1}}^{(n)})\|_F \\ &\quad + \frac{1}{\mu}\|\boldsymbol{A}_{k_j}^{(n)} - \boldsymbol{A}_{k_{j-1}}^{(n)}\|_F \Rightarrow \\ &\overset{b}{\leq} \beta\|\boldsymbol{A}_{k_j}^{(n)} - \boldsymbol{A}_{k_{j-1}}^{(n)}\|_F + \frac{1}{\mu}\|\boldsymbol{A}_{k_j}^{(n)} - \boldsymbol{A}_{k_{j-1}}^{(n)}\|_F \quad (22)\end{aligned}$$

"a" is because of triangular inequality, and "b" is because of gradient Lipschitz property of $h(\boldsymbol{A}^{(n)})$. According to the Remark. 1, "Converging subsequence", the right hand side of (22) converges to zero, so $\boldsymbol{U}_j \to 0$. Here, a definition and a lemma that will be helpful to complete the proof are reviewed.

Definition 2 (Definition of graph [45])**.** *A graph of a function, say $q : \mathbb{R}^d \to (-\infty, +\infty]$, is $\{(\boldsymbol{u}, t) \in \mathbb{R}^{d+1} : q(\boldsymbol{u}) = t\}$.*

Lemma 2 ([45])**.** *Let $\{(\boldsymbol{X}^k, \boldsymbol{U}^k)\}_{k\in\mathbb{N}}$ be a sequence in graph of the sub-gradient of a function, ∂F, that converges to $(\boldsymbol{X}, \boldsymbol{U})$ as $k \to \infty$. If $F(\boldsymbol{X}^k)$ converges to $F(\boldsymbol{X})$ as $k \to \infty$, then $(\boldsymbol{X}, \boldsymbol{U}) \in$ graph(∂F).*

So far, a converging subsequence $\boldsymbol{A}_{k_j}^{(n)}$ for the cost function is considered, and that $\boldsymbol{U}_j \in \partial f$ was showed. Therefor, $\{(\boldsymbol{A}_{k_j}^{(n)}, \boldsymbol{U}_j)\}_{j=0}^{\infty}$ is a sequence in graph of the sub-gradient of the cost function. In Remark. 1, "Converging subsequence", we mentioned that the sub-sequent $\boldsymbol{A}_{k_j}^{(n)}$ converges to $\boldsymbol{A}^{(n)^*}$, and in (22) we proved that $\boldsymbol{U}_j \to 0$. So, $\{(\boldsymbol{A}_{k_j}^{(n)}, \boldsymbol{U}_j)\}_{j=0}^{\infty}$ converges to $(\boldsymbol{A}^{(n)^*}, 0)$. According to Lemma 2, $(\boldsymbol{A}^{(n)^*}, 0) \in$ graph(∂f). In the other words, $0 \in \frac{\partial f}{\partial \boldsymbol{A}^{(n)}}|_{\boldsymbol{A}^{(n)^*}}$ which means that $\boldsymbol{A}^{(n)^*}$ is a critical point of the cost function. This completes the proof.

ACKNOWLEDGMENT

The authors would like to thank Laurent Condat for useful discussions about the simplex constraint.

REFERENCES

[1] P. Comon, "Tensors: a brief introduction," *IEEE Sig. Proc. Magazine*, vol. 31, no. 3, pp. 44–53, 2014.

[2] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[3] R. Bro, "Multi-way analysis in the food industry," *Models, Algorithms, and Applications. Academish proefschrift. Dinamarca*, 1998.

[4] F. Raimondi, P. Comon, O. Michel, S. Sahnoun, and A. Helmstetter, "Tensor decomposition exploiting diversity of propagation velocities: Application to localization of icequake events," *Signal Processing*, vol. 118, pp. 75–88, 2016.

[5] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE signal processing magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[6] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Machine Learning Research*, vol. 15, pp. 2773–2832, Aug. 2014.

[7] E. Sobhani, P. Comon, C. Jutten, and M. Babaie-Zadeh, "Text mining with constrained tensor decomposition," in *International Conference on Machine Learning, Optimization, and Data Science (LOD)*. Springer, 2019, pp. 219–231.

[8] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[9] F. L. Hitchcock, "The expression of a tensor or a polyadic as a sum of products," *Journal of Mathematics and Physics*, vol. 6, no. 1-4, pp. 164–189, 1927.

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

[11] E. Sobhani, P. Comon, and M. Babaie-Zadeh, "Data mining with tensor decompositions," in *Gretsi*, Lille, Aug. 26-29 2019.

[12] S.-H. Hsieh, C.-S. Lu, and S.-C. Pei, "2d sparse dictionary learning via tensor decomposition," in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2014, pp. 492–496.

[13] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, vol. 64, no. 19, pp. 5052–5065, 2016.

[14] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[15] X. Vu, C. Chaux, N. Thirion-Moreau, and S. Maire, "A proximal approach for nonnegative tensor decomposition," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2017, pp. 201–210.

[16] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[17] P. L. Combettes and 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, ch.10.

[18] W. Wang, L. Albera, L. Senhadji, and J.-C. Pesquet, "An alternating direction method of multipliers for constrained joint diagonalization by congruence," in *Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Cancun, Mexico, Dec. 2015, pp. 197–200.

[19] Y. Zhang, G. Zhou, Q. Zhao, A. Cichocki, and X. Wang, "Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation," *Neurocomputing*, vol. 198, pp. 148–154, 2016.

[20] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[21] P. Comon, "Tensor decompositions—state of the art and applications, keynote address in ima conf," *Mathematics in Signal Processing, Warwick, UK*, vol. 375, 2000.

[22] L. Chiantini, G. Ottaviani, and N. Vannieuwenhoven, "An algorithm for generic and low-rank specific identifiability of complex tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 4, pp. 1265–1287, 2014.

[23] C. J. Hillar and L.-H. Lim, "Most tensor problems are np-hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, pp. 1–39, 2013.

[24] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1084–1127, 2008.

[25] T. G. Kolda, "Orthogonal tensor decompositions," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 243–255, 2001.

[26] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[27] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ_0 norm," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 289–301, 2008.

[28] R. T. Rockafellar and R. J.-B. Wets, *Variational analysis*. Springer Science & Business Media, 2009, vol. 317.

[29] H. Attouch, J. Bolte, and 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*, vol. 137, no. 1-2, pp. 91–129, 2013.

[30] J.-J. Moreau, "Fonctions convexes duales et points proximaux dans un espace hilbertien," *Comptes Rendus Acad. Sciences Paris Serie A Math.*, vol. 255, pp. 2897–2899, 1962.

[31] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Modeling & Simulation*, vol. 4, no. 4, pp. 1168–1200, 2005.

[32] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE transactions on image processing*, vol. 18, no. 11, pp. 2419–2434, 2009.

[33] ——, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[34] H. H. Bauschke, P. L. Combettes *et al.*, *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2011, vol. 408.

[35] E. Sobhani, M. Sadeghi, M. Babaie-Zadeh, and C. Jutten, "A robust ellipse fitting algorithm based on sparsity of outliers," in *25th European Sig. Proc. Conference (EUSIPCO)*, Kos, Greece, 2017, pp. 1195–1199.

[36] A. Beck and L. Tetruashvili, "On the convergence of block coordinate descent type methods," *SIAM journal on Optimization*, vol. 23, no. 4, pp. 2037–2060, 2013.

[37] E. Chouzenoux, J.-C. Pesquet, and A. Repetti, "A block coordinate variable metric forward–backward algorithm," *Journal of Global Optimization*, vol. 66, no. 3, pp. 457–485, 2016.

[38] X. Vu, C. Chaux, N. Thirion-Moreau, S. Maire, and 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*, vol. 31, no. 4, p. e2859, 2017.

[39] H. Chang, S. Marchesini, Y. Lou, and T. Zeng, "Variational phase retrieval with globally convergent preconditioned proximal algorithm," *SIAM Journal on Imaging Sciences*, vol. 11, no. 1, pp. 56–93, 2018.

[40] D. S. Bernstein, *Matrix Mathematics*. Princeton Univ. Press, 2005.

[41] L.-H. Lim and P. Comon, "Nonnegative approximations of nonnegative tensors," *Jour. Chemometrics*, vol. 23, pp. 432–441, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1002/cem.1244

[42] L. Condat, "Fast projection onto the simplex and the ℓ_1 ball," *Mathematical Programming*, vol. 158, no. 1-2, pp. 575–585, 2016.

[43] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[45] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2014.

[46] M. Sadeghi and M. Babaie-Zadeh, "Iterative sparsification-projection: Fast and robust sparse signal approximation," *IEEE Transactions on Signal Processing*, vol. 64, no. 21, pp. 5536–5548, 2016.

[47] K. Huang, *AO-ADMM-Code*. [Online]. Available: https://www.catalyzex.com/paper/arxiv:1506.04209#clicktoread

[48] Y. Xu and W. Yin, *APG-Code*. [Online]. Available: https://xu-yangyang.github.io/BCD/

[49] S. E. Leurgans, R. T. Ross, and R. B. Abel, "A decomposition for three-way arrays," *SIAM Journal on Matrix Analysis and Applications*, vol. 14, no. 4, pp. 1064–1083, 1993.

[50] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[51] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia: SIAM, 2009. [Online]. Available: https://epubs.siam.org/doi/book/10.1137/1.9781611972238

[52] H. A. Kiers, "A three–step algorithm for Candecomp/Parafac analysis of large data sets with multicollinearity," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 12, no. 3, pp. 155–171, 1998.

[53] R. Bro, *Amino acids*. [Online]. Available: http://www.models.life.ku.dk/nwaydata1

[54] A. W. Knapp, *Basic real analysis*. Springer Science & Business Media, 2005.