Received: 16 January 2020 | Revised: 9 February 2021 | Accepted: 23 February 2021
DOI: 10.1002/nla.2373

RESEARCH ARTICLE

WILEY

Accelerating block coordinate descent for nonnegative tensor factorization

Andersen Man Shun Ang[1] | **Jeremy E. Cohen**[2] | **Nicolas Gillis**[1] | **Le Thi Khanh Hien**[1]

[1]Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, Mons, Belgium

[2]INRIA, CNRS, Rennes, France

Correspondence
Andersen Man Shun Ang, Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, 7000 Mons, Belgium.
Email: manshun.ang@umons.ac.be

Funding information
ERC starting grant, Grant/Award Number: 679515; Fonds De La Recherche Scientifique - FNRS; Fonds Wetenschappelijk Onderzoek - Vlaanderen, Grant/Award Number: O005318F-RG47

Abstract
This paper is concerned with improving the empirical convergence speed of block-coordinate descent algorithms for approximate nonnegative tensor factorization (NTF). We propose an extrapolation strategy in-between block updates, referred to as heuristic extrapolation with restarts (HER). HER significantly accelerates the empirical convergence speed of most existing block-coordinate algorithms for NTF, in particular for challenging computational scenarios, while requiring a negligible additional computational budget.

KEYWORDS
block-coordinate descent, Nesterov extrapolation, nonconvex optimization, nonnegative tensor factorization

1 | INTRODUCTION

In this paper, we consider the approximate nonnegative tensor canonical polyadic decomposition (CPD) problem, which we refer to as nonnegative tensor factorization (NTF). A N-way array or Nth order tensor $\mathcal{T}$ is a multidimensional array in the product $\mathbb{R}^{I_1\times\cdots\times I_N}$ of the vector spaces $\mathbb{R}^{I_i}$ for $i = 1, 2, \dots, N$. A vector $x \in \mathbb{R}^{I_1}$ is a first-order tensor, and a matrix $M \in \mathbb{R}^{I_1\times I_2}$ is a second-order tensor. The goal of NTF is to approximate a tensor $\mathcal{T}$ by a structured tensor $\mathcal{X}$. Using the squared Frobenius norm as a distance metric, defined as $\|\mathcal{X}\|_F^2 = \sum_{j_1,j_2,\dots,j_N} \mathcal{X}^2_{j_1 j_2 \dots j_N}$, NTF is the following optimization problem:

$$\min_{a_p^{(i)}\geq 0, 1\leq i\leq N, 1\leq p\leq r} \left\| \mathcal{T} - \sum_{p=1}^{r} \bigotimes_{i=1}^{N} a_p^{(i)} \right\|_F^2, \tag{1}$$

where $\otimes$ is a tensor product over N real vector spaces $\mathbb{R}^{I_1}, \dots, \mathbb{R}^{I_N}$ defined as follows:

$$\left[\bigotimes_{i=1}^{N} a_p^{(i)}\right]_{j_1,j_2,\dots,j_N} := \prod_{i=1}^{N} a_p^{(i)}(j_i), \quad \text{where } a_p^{(i)} \in \mathbb{R}^{I_i} \text{ for } i = 1, 2, \dots, N \text{ and } p = 1, 2, \dots, r.$$

In other words, NTF is a low* nonnegative rank approximation problem, since by definition any nonnegative rank r tensor $\mathcal{X}$ of order N can be parameterized as

$$\mathcal{X} = \sum_{p=1}^{r} \bigotimes_{i=1}^{N} a_p^{(i)} \text{ where } a_p^{(i)} \in \mathbb{R}_+^{I_i} \text{ for } i = 1, 2, \dots, N \text{ and } p = 1, 2, \dots, r.$$

Intuitively, using NTF to approximate a tensor means using a part-based decomposition to summarize its content with a few "simple" rank-one tensors $a_p^{(1)} \otimes a_p^{(2)} \otimes \cdots \otimes a_p^{(N)}$ where the components $a_p^{(i)}$ are entrywise nonnegative, with $1 \le p \le r$ and $1 \le i \le N$. This idea finds numerous applications in diverse areas, among which chemometrics or psychometrics are historical examples, see References 3-5.

In this paper, we use the Frobenius norm to measure the error of approximation. It is arguably the most widely used measure, mostly because it has some nice properties (in particular, the subproblems in each block of variables is a convex quadratic problem; see below) and it corresponds to the maximum likelihood estimator in the presence of i.i.d. Gaussian noise. NTF is a nonconvex optimization problem. Moreover, no closed-form solution is known to solve NTF; in fact, the problem is NP-hard already for the matrix case, that is, for $N = 2$; see Reference 6. Therefore, there has been a large amount of works dedicated to solving NTF using various optimization heuristics; see Section 2 for a review of the state-of-the-art algorithms. However, unlike unconstrained approximate tensor factorization (TF), NTF is well-posed in the sense that there always exists an optimal solution; see Reference 7. Moreover, a solution $\mathcal{X}^*$ to NTF is almost always† unique for $N > 2$, and the solution to (1) also has exactly rank r; see Reference 8.

1.1 | Outline and contribution

This paper focuses on computing solutions to NTF as fast as possible. We derive new Block-Coordinate Descent (BCD) algorithms for NTF, that aim at being faster than existing BCD algorithms. To achieve this empirical speed-up in convergence speed, an extrapolation scheme "à la Nesterov" is used every time after a block has already been updated, before switching to another block. The proposed Heuristic Extrapolation with Restarts (HER) algorithm consists of the following steps:

1. Initialize $A^{(i)} = [a_1^{(i)}, \dots, a_r^{(i)}]$ and pairing variables $\hat{A}^{(i)}$ for $1 \le i \le N$.
2. Loop over the blocks $A^{(i)}$ $(1 \le i \le N)$:
   (a) Update $A^{(i)}$ by minimizing (1) where the other blocks are fixed and take the value of the pairing variables $\hat{A}^{(j)}$ $(j \ne i)$. For example, one can take a gradient step (see Section 2.5 for more sophisticated strategies). Keep the previous value of $A^{(i)}$ in memory as $A_{\text{old}}^{(i)}$.
   (b) Update the pairing variable using extrapolation:
$$\hat{A}^{(i)} = \max\left(0, A^{(i)} + \beta(A^{(i)} - A_{\text{old}}^{(i)})\right).$$
3. If the reconstruction error F has increased, reject the extrapolation and reset pairing variables $\hat{A}^{(i)} = A^{(i)}$ for $1 \le i \le N$; otherwise, update $A^{(i)} = \hat{A}^{(i)}$ for $1 \le i \le N$.
4. Update the parameter β; see Section 3.1.4 for the details.
5. If convergence criterion is not met, go back to 2.

This approach has been scarcely studied,[9-11] while extrapolation is a rather well-understood method to accelerate both convex and nonconvex single-block descent algorithms; see for instance References 12,13. The main novelty of this paper is to tackle a nonconvex optimization problem using BCD with extrapolation between the block update, as opposed to inside each block update such as in Reference 14 or after each outer loop as in Reference 10. This in-between extrapolation comes at almost no additional computational cost.

Extrapolated BCD algorithms are shown to be considerably faster than their standard counterparts in various difficult cases. These algorithms were observed to be slower than existing BCD algorithms only for extremely sparse tensors.

*Low means much smaller than the generic rank of tensors in the considered tensor space.[1,2]
†The set of "bad" $\mathcal{T}$ form an hypersurface.

Therefore, a contribution of this work is to experimentally show that using in-between block extrapolation allows to accelerate any BCD algorithm for dense NTF. This opens interesting questions for other optimization problems usually solved by BCD, for which such an extrapolation scheme may be applicable.

1.2 | Context

Let us provide a brief historical note about tensor decomposition. The origin of tensor decomposition can be traced back to the work of Hitchcock,[15,16] whereas the idea of using multiway analysis is credited to the work of References 17,18. Since then, especially after the work of Tucker in the field of psychometrics,[19,20] tensor decomposition has spread and become more and more popular in other fields such as chemometrics,[21] signal processing,[22,23] data mining,[24,25] and many more. We refer the readers to References 26-30 and references therein for comprehensive reviews of the applications of tensor decomposition. It is important to note that NTF is just one of many tensor decomposition models. Some other types of tensor decomposition or format include PARAFAC (i.e., unconstrained approximate TF), Tucker/HOSVD,[20,31] and Tensor Train,[32] to name a few. We focus on NTF in this paper.

Nonnegative matrix factorization (NMF), a key problem in machine learning and data analysis, is a special case of NTF when $N=2$. First introduced in Reference 33, it started becoming widely used after the seminal work of Reference 34, and NMF has since then been deeply studied and well documented with variety of applications such as document classification, image processing, audio source separation and hyperspectral unmixing; see References 35-37 and the references therein for more details. On the other hand, there are classes of data for which being represented by tensors is more natural. For example, a third-order tensor is preferably used to connect excitation-emission spectroscopy matrices in chemometrics,[21] and RGB color images or 3D light field displays are generated as tensors;[38] see Reference 35 for more examples. NTF was first introduced in Reference 39 for fitting the latent class model in statistics. It has also been applied in model selection problem, sparse image coding in computer version,[40] sound source separation,[41] image decomposition,[42] text mining,[43] among others; see References 7,35,44-46 and reference therein for more applications of NTF.

1.3 | Notation

Below we recall some important notations in tensor algebra. First of all, the Kronecker product[47] of two matrices $A \in \mathbb{R}^{I_1 \times J_1}$ and $B \in \mathbb{R}^{I_2 \times J_2}$ is defined as follows:

$$A \boxtimes B = \left[\begin{array}{c|c|c} [A]_{11}B & \cdots & [A]_{1J_1}B \\ \hline \vdots & \ddots & \vdots \\ \hline [A]_{I_1 1}B & \cdots & [A]_{I_1 J_1}B \end{array}\right]. \tag{2}$$

Moreover, the Kronecker product of several matrices can be deduced from the above definition by associativity. The Khatri–Rao product $A \odot B$ is the columns-wise Kronecker product. Setting $A = [a_1, \ldots, a_{J_1}]$ and $B = [b_1, \ldots, b_{J_1}]$,

$$A \odot B = \left[a_1 \boxtimes b_1, \ldots, a_{J_1} \boxtimes b_{J_1}\right]. \tag{3}$$

The Hadamard product (element-wise product) is denoted $A \circledast B$.

1.3.1 | Compact decomposition notations

There exist several complementary notations to parameterize a low-rank tensor. In particular, grouping components $a_p^{(i)}$ as columns of factor matrices $A^{(i)} = [a_1^{(i)}, \ldots, a_r^{(i)}]$, the following notations are equivalent:

$$\mathcal{X} = \sum_{p=1}^{r} \bigotimes_{i=1}^{N} a_p^{(i)}, \tag{4}$$

$$= \left[A^{(1)}, \ldots, A^{(N)}\right], \tag{5}$$

$$= \mathcal{I}_r \times_1 A^{(1)} \times_2 \cdots \times_N A^{(N)}, \tag{6}$$

$$\underset{\text{def}}{:=} \left(\bigotimes_{i=1^a}^{N} A^{(i)} \right) \mathcal{I}_r, \tag{7}$$

where $\otimes_a$ is a tensor product of linear maps induced by the tensor product $\otimes$ of vectors.

Equation (4) is the so-called Kruskal notation, Equation (5) makes use of the n-mode product $\times_p$ (see Reference 26), and Equation (6) uses the fact that linear applications on tensor spaces of finite dimensions also form a tensor space with tensor product $(A \otimes_a B)(x \otimes y) := Ax \otimes By$. Because (6) exhibits this tensor product structure, we will make use of this compact formulation rather than the others.

1.3.2 | Tensor unfoldings and useful formula

To derive partial derivatives of the NTF cost with respect to factors matrices, it is convenient to switch from a tensor formulation to a matrix description of the problem. More precisely, the following relationships hold:

$$\mathcal{X} = \left(\bigotimes_{i=1^a}^{N} A^{(i)} \right) \mathcal{I}_r \quad \equiv \quad \forall i \in [1, N], X_{[i]} = A^{(i)} \left(\bigodot_{\substack{l \neq i \\ l=N}}^{1} A^{(l)} \right)^T, \tag{8}$$

where unfoldings $X_{[i]}$ of a rank-one tensor $\mathcal{X}$ are defined as follows:

$$X_{[i]} := a^{(i)} \otimes \left(\boxtimes_{\substack{l \neq i \\ l=N}}^{1} a^{(l)} \right) \in \mathbb{R}^{\mathcal{D}}, \mathcal{D} = I_i \times \prod_{l \neq i} I_l. \tag{9}$$

Unfoldings of a general tensor are obtained by linearity of the unfolding maps. Note that several nonequivalent definitions are used in the tensor signal processing community; see References 26 and 48.

2 | THE STATE-OF-THE-ART ALGORITHMS FOR SOLVING NTF

Below, we provide an overview of various techniques to solve NTF, which can be reformulated as follows

$$\min_{A^{(i)} \geq 0, 1 \leq i \leq N} F(A^{(1)}, \ldots, A^{(N)}),$$

where

$$F(A^{(1)}, \ldots, A^{(N)}) = \frac{1}{2} \left\| \mathcal{T} - \left(\bigotimes_{i=1^a}^{N} A^{(i)} \right) \mathcal{I}_r \right\|_F^2. \tag{10}$$

As a foreword, let us mention that there exist a wide range of algorithmic solutions for NTF (as for most of the tensor decomposition problems), that show different performances depending on the task at hand.

2.1 | Algorithms for exact PARAFAC

First of all, although the focus of this paper is approximate decompositions, it is interesting to point out that several algebraic techniques based on eigendecompositions have been proposed to deal with exact unconstrained TFs[49-51] whenever such a factorization exists. However, by design, these algorithms are typically not robust to noise, or may even be numerically unstable; see Reference 52. These techniques are sometimes used for initialization. Moreover, to the best of our knowledge, there does not exist an algorithm that computes NTF exactly for any rank using algebraic techniques. In fact, even in the matrix case, that is, for NMF, such techniques do not exist. Therefore, in this paper, we do not discuss exact NTF algorithms; this is a direction for further research.

2.2 | Algorithms for approximate unconstrained tensor decomposition

Because the TF model (that is, without nonnegativity constraints) has some interesting identifiability properties, it may occur that for well-conditioned tensors,[53] approximate NTF can be computed with high precision by using a TF solver. Solving the TF problem is however harder in theory (the tensor low-rank unconstrained approximation problem is ill-posed, see Reference 54) and not really easier in practice than solving NTF. Actually, many algorithms that solve NTF are inspired from TF solvers and have similar complexity. Therefore we do not discuss TF solvers in what follows, and assume the reader is interested in solving NTF with specific algorithms that make use of the properties of the NTF problem.

2.3 | All-at-once optimization

A first class of widely used methods to solve NTF are all-at-once gradient-based methods. Indeed, it is quite straightforward to compute the gradients of F with respect to each matrix $A^{(i)}$. Let us denote

$$B^{(i)} = A^{(N)} \odot \cdots \odot A^{(i+1)} \odot A^{(i-1)} \odot \cdots \odot A^{(1)}. \tag{11}$$

Then the gradient of F with respect to $A^{(i)}$ is

$$\nabla_{A^{(i)}} F = \left(A^{(i)} \left(B^{(i)}\right)^T - \mathcal{T}_{[i]} \right) B^{(i)}. \tag{12}$$

Therefore, there is no obstacle to using any constrained gradient-based algorithm to (try to) find a stationary point of the non-convex NTF problem. To the best of our knowledge, the oldest all-at-once algorithm for NTF is a Gauss–Newton approach,[55] but many approaches have been tested, including:

- Second-order optimization: Using the fact that surrogates of the Hessian of F are heavily structured, second-order information can be used to solve NTF at a reasonable cost.[56] Limited-memory BFGS has also been employed when scalability is required.[57] To enforce the nonnegativity constraints, one can for instance square the variables, or use a variational approach (such a log-barrier).
- Primal-dual optimization: The alternating direction method of multipliers has been tested for NTF, with however less-promising results than its block-coordinate counterpart discussed below, see Reference 58.
- Conjugate gradient: It has been reported that conjugate gradient can be used to solve NTF by squaring the variables; see.[59]

2.4 | BCD methods

Other than the above-mentioned algorithms, BCD has become a standard and efficient scheme for solving NTF, mainly because (1) it essentially has cheap computation cost in each block update (BCD fixes all blocks except for one), (2) BCD can make use of recent developments in convex constrained optimization to efficiently solve NTF with respect to each block, and (3) under some suitable assumptions, many first-order BCDs and their accelerated versions have convergence guarantee in the context of general block-separable *nonconvex composite optimization* problem that subsumes NTF as a special case, see for example, References 60 and 61 and the references therein. Below, we review several block coordinate methods for solving NTF; we list these algorithms in Table 1.

TABLE 1 Several block coordinate methods for solving nonnegative tensor factorization

Algorithms	AO-AS[27]	AO-ADMM[58]	AO-Nesterov[62]	A-HALS[36]	APG[60]	iBPG[61]
Section	Section 2.5.1	Section 2.5.2	Section 2.5.3	Section 2.5.4	Section 2.6.1	Section 2.6.2

2.5 | Alternating optimization (AO) framework

When solving NTF using BCDs, the blocks of variables that are alternatively updated must be chosen. It turns out that F is a quadratic function with respect to each matrix $A^{(i)}$ and therefore the optimization problem

$$\min_{A^{(i)}\geq 0} F(A^{(1)}, \ldots, A^{(N)}), \tag{13}$$

is a linearly constrained quadratic programming problem referred to as Nonnegative Least Squares (NNLS). In particular, it is strictly convex if and only if $B^{(p)}$ is full column-rank. Therefore, it is quite natural to consider $A^{(i)}$ as the blocks in a BCD. The AO framework, which is a standard procedure to solve NTF, alternatively (exactly/inexactly) solves (13) for each block. We describe the AO framework in Algorithm 1. Note that the objective function of AO methods decreases after each block update. Depending on how the matrix-form NNLS problem (14) is solved, various implementations of AO algorithms can be obtained. Some of them are very efficient for solving NTF, they are surveyed below.

Algorithm 1. Alternating optimization framework

```
1: Input: a nonnegative $N$-way tensor
2: Output: nonnegative factors $A^{(1)}, A^{(2)}, \ldots, A^{(N)}$.
3: Initialization: $\left(A_0^{(1)}, \ldots, A_0^{(N)}\right)$.
4: for $k = 1, \ldots$ until some criteria is satisfied do
5:     for $i = 1, \ldots, N$ do
6:         Update $A_k^{(i)}$ as an exact/inexact solution of
               $$\min_{A^{(i)}\geq 0} F\left(A_k^{(1)}, \ldots, A_k^{(i-1)}, A^{(i)}, A_{k-1}^{(i+1)}, \ldots, A_{k-1}^{(N)}\right). \tag{14}$$
         ($A_{k-1}^{(i)}$ can be used as the initial point for the algorithm used to solve (14).)
7:     end for
8: end for
```

2.5.1 | AO-AS – solving NNLS with Active Set

When $A_k^{(i)}$ is updated by an exact solution of the NNLS problem (14), we obtain an alternating NNLS algorithm, usually referred to as ANLS in the literature. To solve exactly the NNLS subproblem (14), active set (AS) methods are usually rather effective and popular; see Reference 27. We will refer to AO-AS as the ANLS algorithm where the NNLS subproblems are solved with AS.

2.5.2 | AO-ADMM—solving NNLS with ADMM

Designed to tackle a wide range of constrained tensor decomposition problems and various loss functions, AO-ADMM[58] applied to NTF boils down to using several steps of a primal-dual algorithm, the Alternating Direction Method of Multipliers (ADMM), to solve the cascaded NNLS problems. Therefore, AO-ADMM for NTF problem (1) is a variant of the AO framework that solves (14) inexactly.

2.5.3 | AO-Nesterov

When Nesterov's accelerated gradient method is applied to solve the NNLS problem in (14), we obtain AO-Nesterov; see References 62,63.

2.5.4 | A-HALS

The hierarchical alternating least square (HALS) algorithm was introduced for solving the NMF problem $\min_{W\geq 0, H\geq 0} \|M - WH^T\|_F^2$ (that is, NTF when $N = 2$), and has been widely used for solving NMF as it performs extremely well in practice; see for example References 35,36.

HALS cyclically updates each column of the factor matrix $A^{(i)}$ by solving an NNLS problem with respect to that column while fixing the others. The optimal solution of this NNLS subproblem can be written in closed form. A-HALS, which is short for accelerated HALS, was proposed in Reference 64 to accelerate HALS. A-HALS repeats updating each factor matrix several times before updating the other ones. Hence A-HALS can be considered as a variant of the AO framework where each NNLS is inexactly solved itself by a BCD with closed-form updates. Let us briefly describe A-HALS for solving NTF. The NNLS problem (14) of A-HALS is inexactly solved by repeating cyclically updating the columns of $A^{(i)}_{k-1}$. In particular, let $M = X_{(i)}$, $W = A^{(i)}_{k-1}$, and $H = A^{(N)}_{k-1} \odot \dots A^{(i+1)}_{k-1} \odot A^{(i-1)}_{k} \dots \odot A^{(1)}_{k}$. The jth column of $A^{(i)}_{k-1}$ is updated by

$$W_{:,j} = \frac{\max\left(0, MH^T_{j,:} - \sum_{l\neq j} W_{:,l}H_{l,:}H^T_{j,:}\right)}{\|H_{j,:}\|^2}.$$

It is worth noting that A-HALS for NTF has subsequential convergence guarantee (i.e., every limit point is a stationary point of the objective function), see Reference 65, section 7.

2.6 | Block proximal gradient-type methods

The NNLS problem (14) does not have a closed-form solution. From Equation (12), we see that F, when restricted to $A^{(i)}$, is a $L^{(i)}$-smooth function, that is, the gradient $\nabla_{A^{(i)}}F$ is Lipschitz continuous with the constant $L^{(i)} = \left\|\left(B^{(i)}\right)^T B^{(i)}\right\|$, where $B^{(i)}$ is defined in (11). This property can be employed to replace the objective function in the NNLS problem (14) by its quadratic majorization function, that leads to a new minimization problem which has a closed-form solution. This minimization–majorization approach, in the literature of block-separable composite optimization problem with the block-wise L-smooth property, is known as proximal gradient block coordinate descent method (see e.g., References 61). Considering the NTF problem, the closed-form solution of minimizing the majorization function is a projected gradient step. Applying Nesterov-type acceleration for the proximal gradient step improves the convergence of the BCD algorithm. Below we review the two recent accelerated proximal gradient BCD methods that were proposed for solving the general composite optimization problem.

2.6.1 | APG – An alternating proximal gradient method for solving NTF

APG was proposed by Xu and Yin;[60] see the appendix of References 66 and 60, section 3.2, for the algorithm pseudocode. APG *cyclically* update each block (a.k.a each factor matrix) $A^{(i)}$ by calculating an extrapolation point $\hat{A}^{(i)}_{k-1} = A^{(i)}_{k-1} + w^{(i)}_{k-1}\left(A^{(i)}_{k-1} - A^{(i)}_{k-2}\right)$ (here $w^{(i)}_{k-1}$ is some extrapolation parameter) and embedding this point in a projected gradient step

$$A^{(i)}_k = \max\left(0, \hat{A}^{(i)}_{k-1} - \frac{1}{L^{(i)}_{k-1}}\left(\hat{A}^{(i)}_{k-1}\left(B^{(i)}_{k-1}\right)^T - \mathcal{T}_{[i]}\right) B^{(i)}_{k-1}\right).$$

After all blocks are updated, APG needs a restarting step, that is, if the objective function has increased then the projected gradient step would be redone by using the previous values of all blocks instead of using the extrapolation points.

2.6.2 | iBPG – An inertial block proximal gradient method

Recently proposed in Reference 61, iBPG computes two different extrapolation points $\hat{A}_{k-1}^{(i,1)}$ and $\hat{A}_{k-1}^{(i,2)}$: one is for evaluating the gradient and the other one for adding inertial force. iBPG updates one matrix factor using a projected gradient step

$$A_k^{(i)} = \max\left(0, \hat{A}_{k-1}^{(i,2)} - \frac{1}{L_{k-1}^{(i)}}\left(\hat{A}_{k-1}^{(i,1)}\left(B_{k-1}^{(i)}\right)^T - \mathcal{T}_{[i]}\right)B_{k-1}^{(i)}\right),$$

see the appendix of Reference 66 for the algorithm pseudocode. Furthermore, similar to A-HALS, iBPG allows updating each matrix factor some times before updating another one—this feature would help save some computational costs since some common expressions can be re-used when repeating updating the same block. iBPG does not require a restarting step which make it suitable for solving large-scale NTF problems where evaluating the objective functions is costly.

3 | MAKING BCD SIGNIFICANTLY FASTER WITH HER

With modern machine learning applications of NTF in mind, for which input tensor sizes can be extremely large and NTF should be provided as a low-level routine, there would be a definite economical and scientific gain to speeding up NTF algorithms. Radically different approaches exist in the literature to speed up existing algorithms for solving NTF, such as parallel computing,[67,68] compression, and sketching.[69,70] The combinations and relationships between these methods is poorly understood. In this paper, we focus on the acceleration of BCD using extrapolation.

Extrapolation inside BCD algorithms such as the workhorse ALS algorithm has been studied in the tensor decomposition literature as an empirical trick to speed up computations and avoid swamps, see Section 3.5. However, in a recent work on rank-one approximations of rank-two tensors, Gong, Mohlenkamp, and Young[71] used gradient flow to study transverse stability, and provided a deeper analysis of the optimization landscape of tensor low-rank approximation which further supports the use of extrapolation.

As reviewed in Section 2.6, we have seen that APG and iBPG accelerate block proximal gradient methods by using extrapolation points in the projected gradient step to update each factor matrix. In another line of works, AO (Algorithm 1) was accelerated by using extrapolation *between* each block update (rather than inside the block update as in APG and iBPG); in other words, each factor matrix is updated by the extrapolation between previous updated factors. In the literature of tensor decomposition, the second type of extrapolation has been used to accelerate alternating least squares algorithms for solving CPD. Those works will be reviewed in Section 3.5. In the following, we introduce HER—a novel extrapolation scheme that can be categorized into the class of accelerated AO algorithms using extrapolation between block update.

3.1 | Heuristic extrapolation with restarts

HER was first proposed for solving NMF in Reference 11, and found to be extremely effective on NTF in a preliminary work.[72] The sketch of HER was given in the introduction and its pseudocode is given in Algorithm 2. In the following, we elaborate on HER with more details.

3.1.1 | Update step—line 6

It is clear that Algorithm 2 has the form of an alternating optimization framework in which the key optimization subproblem (15) is a NNLS problem. As reviewed in Section 2, some efficient algorithms for the NNLS problem (15) include AS, ADMM, Nesterov's accelerated gradient, or A-HALS. The main difference between AO and HER is that HER does not use the latest values of the other blocks $A^{(j)}$ $(j \neq i)$ but employs the latest values of their extrapolation $\hat{A}^{(j)}$ $(j \neq i)$. For convenience, we refer to $\left\{\hat{A}_k^{(i)}, i = 1, \dots, N\right\}_{k\geq 0}$ as the extrapolation sequence.

Algorithm 2. HER

```
1: Input: a nonnegative $N$-way tensor
2: Output: nonnegative factors $A^{(1)}, A^{(2)}, \dots, A^{(N)}$.
3: Initialization: Choose $\beta_0 \in (0,1)$, $\eta \geq \bar{\gamma} \geq \gamma \geq 1$ and two sets of initial factor matrices $\left(A_0^{(1)}, \dots, A_0^{(N)}\right)$ and $\left(\hat{A}_0^{(1)}, \dots, \hat{A}_0^{(N)}\right)$. Set $\bar{\beta}_0 = 1$ and $k = 1$.
4: for $k = 1, \dots$ until some criteria is satisfied do
5:     for $i = 1, \dots, N$ do
6:         Update step Let $A_k^{(i)}$ be an exact/inexact solution of
               $$\min_{A^{(i)} \geq 0} F\left(\hat{A}_k^{(1)}, \dots, \hat{A}_k^{(i-1)}, A^{(i)}, \hat{A}_{k-1}^{(i+1)}, \dots, \hat{A}_{k-1}^{(N)}\right). \quad (15)$$
7:         Extrapolation step
               $$\hat{A}_k^{(i)} = \max\left(0, A_k^{(i)} + \beta_{k-1}(A_k^{(i)} - A_{k-1}^{(i)})\right). \quad (16)$$
8:     end for
9:     Compute $\hat{F}_k := F\left(\hat{A}_k^{(1)}, \hat{A}_k^{(2)}, \dots, \hat{A}_k^{(N-1)}, A_k^{(N)}\right)$.
10:    if $\hat{F}_k > \hat{F}_{k-1}$ then
11:        Set $\hat{A}_k^{(i)} = A_k^{(i)}$, $i = 1, \dots, N$                    % abandon the sequence $\hat{A}_k^{(i)}$
12:        Set $\bar{\beta}_k = \beta_{k-1}$, $\beta_k = \beta_{k-1}/\eta$.                % Update $\bar{\beta}$, decrease $\beta$
13:    else
14:        Set $A_k^{(i)} = \hat{A}_k^{(i)}$, $i = 1, \dots, N$.                   % keep the sequence $\hat{A}_k^{(i)}$
15:        Set $\bar{\beta}_k = \min\{1, \bar{\beta}_{k-1}\bar{\gamma}\}$, $\beta_k = \min\{\bar{\beta}_{k-1}, \beta_{k-1}\gamma\}$.   % Increase $\bar{\beta}$ and $\beta$
16:    end if
17: end for
```

3.1.2 | Extrapolation step—line 7

After the update of $A_k^{(i)}$, the same block of the extrapolation sequence $\hat{A}_k^{(i)}$ is updated by extrapolating $A_k^{(i)}$ along the direction $A_k^{(i)} - A_{k-1}^{(i)}$, see (16). Note that $\hat{A}_k^{(i)}$ produced by (16) is always feasible. It is possible to remove the projection in (16), but we do not consider such approach in this work. Note that, regarding feasibility, $A_k^{(i)}$ produced by line 6 of Algorithm 2 is always feasible regardless of the feasibility of $\hat{A}_k^{(i)}$.

3.1.3 | The restart mechanism—lines 9–16

After the update-extrapolate process on all the blocks, a restart procedure is carried out to decide whether or not we replace $A^{(i)}$ $(1 \leq i \leq N)$ with the extrapolation sequence. The command in line 14 of Algorithm 2 has the same spirit with the update in (21) where the factor matrices are updated by *the extrapolation between block update*.

It may raise a question why $F\left(A_{k-1}^{(1)}, \dots, A_{k-1}^{(N)}\right)$ does not appear in the restarting condition—line 10. The answer is due to the practicality of the algorithm. As stated in Reference 11, using F as the restart criterion is computationally much more expensive than using the approximate $\hat{F}$. When computing $\hat{F}$, no explicit computation is required; instead, one may reuse already computed components from the updates of $A^{(N)}$ and $\hat{A}^{(N)}$. This creates an important reduction of computational complexity. For example, let us consider an order-N NTF problem with factor matrices $\left\{A^{(i)}\right\}_{i=1,2,\dots,N}$ with size $I_1 \times r, I_2 \times r, \dots$ up to $I_N \times r$. Reusing already computed components (such as gradient) in the update of the last block $(A^{(N)}, \hat{A}^{(N)})$, we can compute $\hat{F}(\hat{A}^1, \dots, \hat{A}^{N-1}, A^N)$ under $I_N r^{N-1}$ flops. However, if we compute $F(A^1, \dots, A^N)$, it takes $\prod_{i=1}^N I_i$ flops. If $r^N \ll \prod_i I_i$, then such reduction in complexity from $\prod_{i=1}^N I_i$ to $I_N r^{N-1}$ is significant even when N is low. Furthermore, we can even rotate the tensor such that I_N is the mode with the smallest size among all the modes. In fact, computing the cost function naively can be as costly as one block update, and thus using $\hat{F}$ instead of F as the restart criterion is important, since restart using F requires computing the cost function at each iteration, while restart using $\hat{F}$ is much cheaper.

Moreover, note that if the iterates sequence is converging, then the extrapolated sequence also converges to the same limit point. Therefore, since F is a continuous map, if convergence of the iterates is observed then the surrogate cost $\hat{F}$ will asymptotically converge to the same final value as F. Although we did not characterize how fast this convergence happens, this justifies to use $\hat{F}$ as a surrogate at least near a stationary point.

3.1.4 | The extrapolation parameters in lines 9–16

The extrapolation weight β_k is computed within the restart mechanism of lines 9–16 of Algorithm 2, and it is updated using four parameters; see Table 2.

In the initialization stage, we set the upper bound for β as $\overline{\beta}_0 = 1$, pick $\beta_0 \in (0,1)$, and select η, γ and $\overline{\gamma}$ such that $1 < \overline{\gamma} \le \gamma \le \eta$. The parameter $\overline{\beta}$, which is initialized as 1, is called the upper bound parameter for β. This parameter is used to limit the growth of β; see below for more details. The parameter γ is called the (multiplicative) growth rate of β: when the error decreases, β is updated with $\gamma\beta$. Similarly $\overline{\gamma}$ is the (multiplicative) growth rate of $\overline{\beta}$. Finally, η is called the decay rate of β. This value is used to update β with β/η when the error increases. The parameters $(\gamma, \overline{\gamma}, \eta)$ are fixed constants, while β and $\overline{\beta}$ are updated depending on the restart condition.

3.2 | The update of β

HER updates β_k as

$$\beta_{k+1} = \begin{cases} \beta_k/\eta & \text{if } \hat{F}_{k+1} > \hat{F}_k \\ \min\{\gamma\beta_k, \overline{\beta}_k\} & \text{if } \hat{F}_{k+1} \le \hat{F}_k \end{cases}, \tag{17}$$

which is explained as follows :

- If restart occurs, that is, if $\hat{F}_{k+1} > \hat{F}_k$, we assume it is caused by an over-sized β_k (recall that, for $\beta_k = 0$, decrease is guaranteed by the update in line 6) and we shrink the value of β for the next iteration using the decay parameter η as in (14).
- Otherwise, $\hat{F}_{k+1} \le \hat{F}_k$, and we assume β_k can safely be increased. We grow β for the next iteration as $\gamma\beta$. To prevent β grow indefinitely, we use an upper bound $\overline{\beta}$ as in (14).

3.3 | The update of $\overline{\beta}$

HER updates $\overline{\beta}_k$ as follows

$$\overline{\beta}_{k+1} = \begin{cases} \beta_k & \text{if } \hat{F}_{k+1} > \hat{F}_k \\ \min\{\overline{\gamma}\overline{\beta}_k, 1\} & \text{if } \hat{F}_{k+1} \le \hat{F}_k \end{cases}. \tag{18}$$

Symbol	Name	Setting	Range	Requires tuning?
β_k	Extrapolation weight	Update as (14)	$[0,1]$	Yes for β_0
γ	Growth rate of β	Constant	$[\overline{\gamma}, \eta]$	Yes
η	Decay rate of β	Constant	$[\gamma, \infty)$	Yes
$\overline{\gamma}$	Growth rate of $\overline{\beta}$	Constant	$[1, \gamma]$	Yes
$\overline{\beta}_k$	Upper bound for β	Update as (15)	$[\beta_k, 1]$	No, $\overline{\beta}_0 = 1$

TABLE 2 Parameters in the heuristic extrapolation with restarts scheme

The explanations are as follows:

- If there is no restart, that is, if $\hat{F}_{k+1} \leq \hat{F}_k$, $\overline{\beta}$ is increased if $\overline{\beta}$ is smaller than 1.
- Otherwise $\hat{F}_{k+1} > \hat{F}_k$ and $\overline{\beta}_{k+1}$ is set to β_k to prevent β_{k+1} growing larger than β_k too fast in the future. In fact, β_k indicates a too large value for β since the error has increased.

Let us make a few remarks:

- The relationships between the parameters in HER is as follows:

$$0 < \beta_k \leq \overline{\beta}_k \leq 1 < \overline{\gamma} \leq \gamma \leq \eta < \infty. \tag{19}$$

  By construction, $\beta_k \leq \overline{\beta}_k \leq 1$, while $\overline{\gamma} \leq \gamma$ ensures that $\overline{\beta}$ increases slower than β, while $\gamma \leq \eta$ ensures that β is decreased faster.
- We have observed that HER is more effective if the NNLS subproblems (15) are solved with relatively high precision. Empirically Figure 6 suggests to use HER with repeated projected gradient steps rather than just a single step. The suffix 50 after the algorithms' name in Figure 6 means that we run 50 iterations for the algorithms to solve (15).
- A drawback of the HER approach is the parameter tuning. There are four parameters to tune: $\beta_0, \gamma, \overline{\gamma}, \eta$. However HER is not too sensitive for reasonable values of the parameters; see Figure 1 for an illustration. Therefore, all the experiments in this paper are executed with no parameter tuning, even in difficult cases when data are ill-conditioned or rank is very high; namely we will use $\beta_0 = 0.5$, $\gamma = 1.05$, $\overline{\gamma} = 1.01$ and $\eta = 1.5$.
- In the implementation, we initialize $\hat{A}_0^{(i)} = A_0^{(i)}$, $i = 1, \ldots, N$.

3.4 | Discussion on convergence

Unfortunately, the HER framework to accelerate BCD methods for NTF cannot be guaranteed to converge using current existing proofs in the literature. The main distinction of HER that makes it very efficient but difficult to prove convergence is its *dynamic and flexibility* in the update of extrapolation parameters. Specifically, HER performs a very

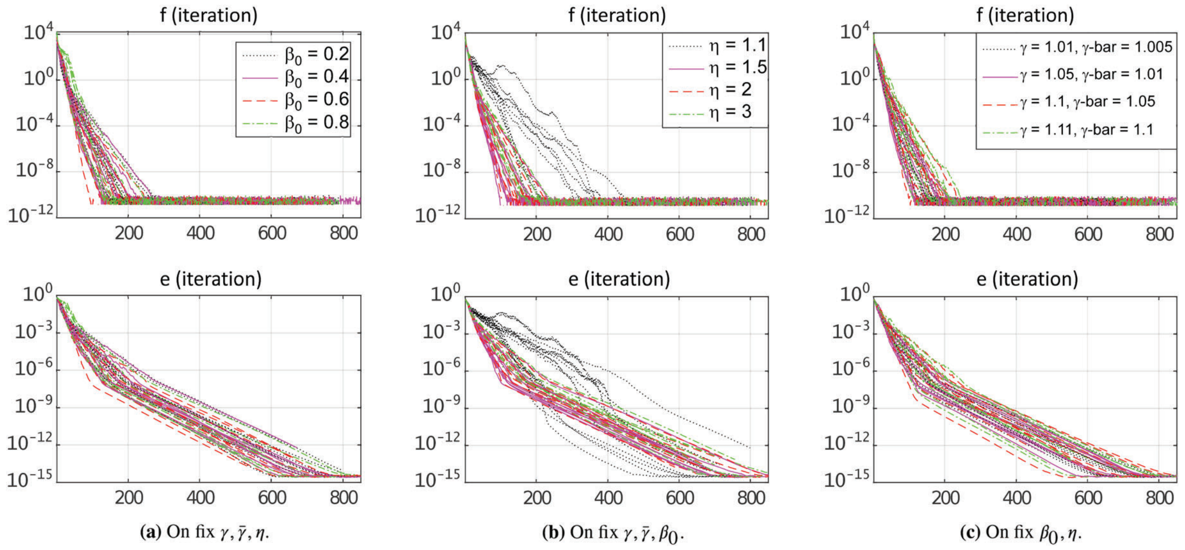

FIGURE 1 Comparison of heuristic extrapolation with restarts (HER) with different parameters on the same nonnegative tensor factorization problems: a rank-10 factorization on noiseless tensors generated by random with size $50 \times 50 \times 50$. For each set of parameters, the decomposition is repeated 10 times over 10 different data tensors and initializations; see Section 4.1 for more details. The top plots representing f display the error of the approximation, and the bottom plots representing e display the distance to the ground truth factors; see (23) and (24). The default set of hyper-parameters are $[\beta_0 = 0.5, \gamma = 1.05, \overline{\gamma} = 1.01, \eta = 1.5]$. The results here showed that HER is not very sensitive to its parameters as all the curves are not deviating away from each other, except for the case $\eta = 1.1$, suggesting that η should not be too small.

aggressive extrapolation strategy: it extrapolates after each update of a block variable and uses that extrapolated point for the evaluation of the next block. Moreover, as long as the objective function decreases, it increases the extrapolation parameter β (as discussed in the previous section). The accelerated block proximal gradient methods using extrapolation inside the block update such as APG and iBPG do not have such strategy. Hence, although these methods have strong convergence guarantee, the way they choose extrapolation parameters is more conservative. Our extensive experiments strongly confirm the efficacy of dynamic and flexibility in choosing extrapolation parameters of HER when extrapolation between each block update is used and monotonicity of the objective function is taken into account. Studying convergence guarantees for the HER framework would be very challenging and is an important direction of further research.

3.5 | Related works

Extrapolated AO algorithms can be traced back to a seminal work by Harshman,[4] in which extrapolation was seen as a way to speed up the convergence of ALS. In this subsection we review two works on extrapolated AO algorithm: an older one[9] and a recent one.[10] To differentiate the extrapolation parameter used in HER (denoted as β_k), we denote the extrapolation parameter used in these works as ω_k.

3.5.1 | General description

Before we provide the details about the algorithms, let us point out a very important observation: unlike Algorithm 3.1, these algorithms use a global extrapolation parameter shared among all the blocks, and the extrapolation is conducted after all the blocks have been updated. That is, they first update all the blocks variables, then stack all the blocks together to form a vector x, and extrapolate it as

$$x_{k+1} = x_{k+\frac{1}{2}} + \omega_k (x_{k+\frac{1}{2}} - x_k), \tag{20}$$

where $x_{k+\frac{1}{2}}$ is the vector obtained by stacking all the blocks $A_k^{(1)}, \dots, A_k^{(N)}$ after all of them have been updated. The algorithms[9,10] follow (17), and they differ in the way ω_k is computed, which we discuss below.

3.5.2 | Extrapolated AO algorithms with Bro's sequence

Bro revisited and optimized the seminal work by Harshman,[4] and came up with an extrapolation scheme with convincing empirical speed-ups for computing CPD: the extrapolation parameter is tuned as

$$\omega_k = k^{\frac{1}{h(k)}} - 1,$$

where $h(k)$ is a recursive function so that $h(k+1) = h(k)$ if the error has not increased for more than four iterations, $h(k+1) = 1 + h(k)$ otherwise, and $h(1) = 3$. Moreover, no extrapolation is performed in the first few iterations because of stability issues. Furthermore, when the error increases, the extrapolation is not performed, that is, the extrapolated sequence is abandoned, as for HER.

Note that there is no particular modification of the Bro extrapolation scheme for aNCPD. In this paper we implement Bro-AHALS, Bro-ADMM and Bro-Nesterov—the three versions of Bro's accelerated methods in which we respectively use the same strategy using A-HALS (see Section 2.5.4), ADMM and Nesterov's accelerated gradient method for solving the NNLS problem (14) inexactly.

3.5.3 | Extrapolated AO algorithms with gradient ratio and line search

Recently two heuristic approaches similar to Bro's have been proposed,[10] and compute ω_k as follows

$$\text{Gradient ratio (GR): } \omega_k = \nabla_x F_k(x)|_{x=x_{k+\frac{1}{2}}} / \nabla_x F_{k-1}(x)|_{x=x_{k-1}}, \tag{21}$$

$$\text{Linear search (LS): } \omega_k = \underset{\omega}{\operatorname{argmin}} \, F\left(x_{k+\frac{1}{2}} + \omega(x_{k+\frac{1}{2}} - x_{k-1})\right), \tag{22}$$

where $\nabla_x F$ is the gradient of F with respect to x. As for Bro's accelerated algorithms, we implement in this paper GR-AHALS, GR-ADMM, GR-Nesterov, LS-AHALS, LS-ADMM, and LS-Nesterov where we correspondingly use the same strategy as for A-HALS (see Section 2.5.4), ADMM and Nesterov's accelerated gradient method for (14).

Note that GR and LS are designed for aCPD but not aNCPD, and similar to Bro's approach,[9] there is no modification of the GR and LS schemes for the nonnegative decomposition case.

3.5.4 | The modified implementation of Bro, GR, and LS

The algorithms Bro, GR, and LS use a global extrapolation parameter shared among all the blocks, which is different from the extrapolation parameter β_k used in HER that is tuned independently for each block. Preliminary tests have showed that HER is always speeding up BCD algorithms much faster than Bro, GR, and LS (see Figure 7 in the next section). Such superiority can be explained in part by the fact that the block-wise tuning of β_k in HER gives HER much more degrees of freedom than Bro, GR and LS. Hence, to make a fair comparison between the different extrapolation strategies, we make the following modifications so that Bro, GR, and LS have the same algorithmic structure as HER.

First, the update of x_k is performed block wise, that is, one $A^{(i)}$ at a time. Next, we extrapolate the blocks right after they have been updated, using the same extrapolation coefficients as described in Bro, GR, and LS. It is important to note that, in the original algorithms, all the block variables are extrapolated with the same "global" extrapolation coefficient. That is, the extrapolation coefficients for every block $A^{(i)}$ in the original algorithm are the same. In the modification here, we "split" the global extrapolation coefficient into block-specific extrapolation coefficient. For example, in GR, the update-then-extrapolate step is performed for all i as

$$\text{Update: } \quad A^{(i)}_{k+\frac{1}{2}} \text{ using (14)}, \tag{23}$$

$$\text{Extrapolate: } \quad A^{(i)}_{k+\frac{1}{2}} + \frac{\|\nabla_{A^{(i)}} F_k\|}{\|\nabla_{A^{(i)}} F_{k-1}\|}(A^{(i)}_{k+\frac{1}{2}} - A^{(i)}_k), \tag{24}$$

where $\nabla_{A^{(i)}} F_k$ is the gradient of F with respect to block $A^{(i)}$ at iteration k (see (12)), and $A^{(i)}_{k+\frac{1}{2}}$ is the block $A^{(i)}$ at iteration k just after the update, that is, we extrapolate the block right after it has been updated, as in HER. Moreover, (21) uses the ratio between the norm of the gradient of the current block A and the norm of the gradient of the same block in the last iteration. We use the same strategy on splitting the global extrapolation coefficient into block-specific extrapolation coefficient in Bro and LS. That is, the term $\frac{\|\nabla_{A^{(i)}} F_k\|}{\|\nabla_{A^{(i)}} F_{k-1}\|}$ in Equation (21) is replaced by $\left(k^{\frac{1}{h(k)}} - 1\right)$ in Bro, and for LS, the extrapolation parameter ω_k is computed by solving a minimization subproblem: consider the update of the ith block at iteration k, we have

$$\omega_k = \underset{\omega}{\operatorname{argmin}} \, F\left(A^{(1)}_k, \dots, A^{(i-1)}_k, A^i_k + \omega(A^i_k - A^i_{k-1}), A^{(i+1)}_{k-1}, \dots, A^{(N)}_{k-1}\right). \tag{25}$$

By expanding F in terms of ω, (22) can be expressed as a second-order polynomial in ω, and hence a closed-form solution for ω exists.

3.5.5 | Computational cost on Bro, GR, and LS compared with HER

The per-iteration cost in both GR and LS schemes is much larger than that of Bro. Both Bro, GR, and LS have restart, but Bro's extrapolation parameter is basically a scalar computation, while GR has multiple matrix–matrix multiplications and LS even has to solve a minimization sub-problem:

- Here we solve (22), which is a second-order polynomial in ω, exactly.
- For (19), for a third-order NCPD problem, we need to minimize a sixth-order polynomial in ω.

- In general, for a N-order NCPD problem, we need to minimize a $2N$-order polynomial in ω. Let $x = [A^{(1)}, \dots, A^{(N)}]$ denotes the stacking of the block into vector[‡], then

$$\begin{aligned}\beta_k &= \arg\min_{\omega} F\left(x_{k+\frac{1}{2}} + \omega(x_{k+\frac{1}{2}} - x_{k-1})\right)\\ &= \arg\min_{\omega} F\left([A^{(1)}_{k+\frac{1}{2}}, \dots, A^{(N)}_{k+\frac{1}{2}}] + \omega\left(\left[A^{(1)}_{k+\frac{1}{2}}, \dots, A^{(N)}_{k+\frac{1}{2}}\right] - [A^{(1)}_{k-1}, \dots, A^{(N)}_{k-1}]\right)\right)\\ &\overset{(6)}{=} \arg\min_{\omega} \frac{1}{2}\left\| \mathcal{T} - \left(\bigotimes_{i=1^a}^{N}\left(A^{(i)}_{k+\frac{1}{2}} + \omega\left(A^{(i)}_{k+\frac{1}{2}} - A^{(i)}_{k-1}\right)\right)\right)\mathcal{I}_r\right\|_F^2 .\end{aligned}$$

  We can see the cost of computing the coefficients for the polynomial can potentially be very high. Due to such reason, in the original paper, the ω_k in (19) is solved approximately using cubic line search in the Poblano toolbox.
- As pointed out in Reference 11, exact line search has bad performance in NMF, so we believe this is the same for LS, for both the original form and the modified form.

In general, the per-iteration cost of the extrapolation step in Bro's extrapolation is negligible, while for GR is higher than one ALS, and for LS it is much higher than one ALS. For this reason, we only run the modified LS in the numerical tests, that is, (22), and we will see that, often LS performs the worst in the experiments, see for example Figure 7.

3.5.6 | Remarks on Bro, GR and LS compared with HER

In the numerical experiments, we will compare the original form of Bro, GR, LS, as well as the modified version. There are several remarks on Bro, GR and LS.

- There are two sequences $A^{(i)}$ and $\hat{A}^{(i)}$ used in HER, while there is no auxiliary sequence in Bro, GR and LS: that is, the extrapolation in these approaches is conducted on the same block $A^{(i)}$.
- By splitting of the extrapolation coefficient into block extrapolation coefficients, the original GR and LS are improved as the ω_k in the new GR and LS are more adapted to each block variable.
- As Bro, GR and LS are designed for aCPD but not aNCPD, and similar to Bro's approach,[9] there is no modification of the GR and LS schemes for aNCPD. This means that there is no guarantee on feasibility of the iterates produced by these methods for aNCPD.

4 | EXPERIMENTS

In this section, we empirically prove the efficacy of HER by extensively test its performance on a rich set of synthetic datasets as well as real data sets. As presented in Section 3.1, HER is a scheme to accelerate AO algorithms by using extrapolation between block update; and as such, by using HER, we can derive several different algorithms corresponding to the solver we use for the NNLS problem (15). We stress out that HER can be used in combination with any BCD algorithm to make it faster. In this section, we combined it with the most well-known algorithms to tackle NTF, namely AS, ADMM, Nesterov accelerated gradient and AHALS for solving (15), which we denote by HER-AS, HER-ADMM, HER-Nesterov, and HER-AHALS, respectively. We call HER-AO the set of these algorithms. Table 3 lists the algorithms that we implement and test in our experiments. Note that our goal is not to compare these various algorithms, but to show that HER can accelerate all of them significantly.

All experiments are run with MATLAB (v.2015a) on a laptop with 2.4GHz CPU and 16GB RAM. The code is available from https://angms.science/research.html.

Remark 1. In this paper, we focus on dense NTF problems, for which the input tensor has mostly positive entries. However, the HER framework can also be applied to sparse tensors. This was in fact done for NMF in Reference 11 with similar

[‡]Here it is not the Kruskal notation (4).

TABLE 3 Algorithms for solving nonnegative tensor factorization

Algorithms	Reference
HER-AS, HER-ADMM, HER-Nesterov, HER-AHALS	Section 3.1
AO-AS, AO-ADMM, AO-Nesterov, AHALS	Section 2.5
Bro-ADMM, Bro-Nesterov, Bro-AHALS	Section 3.5
GR-ADMM, GR-Nesterov, GR-AHALS	Section 3.5
LS-ADMM, LS-Nesterov, LS-AHALS	Section 3.5
APG, iBPG	Section 2.6

conclusions, that is, HER can accelerate algorithms significantly for sparse data sets as well. The problem with sparse data is not the algorithm itself, but rather its implementation. Handling sparse data also means dealing with extremely large dataset, which we cannot deal with our current implementation in Matlab. This could be fixed by integrating the proposed HER framework within a toolbox which features efficient sparse tensor manipulations and contractions, see for instance Reference 73 and the references therein. We leave an efficient implementation of the HER framework for very sparse and large tensors for future works.

4.1 | Setup

4.1.1 | Performance measurement

Two important factors in the evaluation of the performance of an algorithm are the data fitting error and the factor fitting error that are computed as follows. We use the value of the objective function

$$f_k := F\left(A_k^{(1)}, A_k^{(2)}, \dots, A_k^{(N-1)}, A_k^{(N)}\right), \tag{26}$$

to represent the data fitting error. Supposing the ground truth factor matrices $A_{\text{true}}^{(i)}$, $i=1 \dots, N$ are available, then we compute the factor fitting error e_k as

$$e_k := \frac{1}{N}\sum_{i=1}^{N} \frac{\left\|\text{normalize}(A_{\text{true}}^{(i)}) - \text{normalize}(A_k^{(i)})\Pi\right\|_F}{\left\|\text{normalize}(A_{\text{true}}^{(i)})\right\|_F}. \tag{27}$$

Here normalize$(\cdot)$ is the column-wise normalization step (i.e., the ith column of normalize(A) is set to $\frac{A(:,j)}{\|A(:,j)\|_2}$), and Π is the permutation matrix computed through the Hungarian algorithm. The use of Π is to remove the permutation degree of freedom for matching the columns of $A^{(i)}$ to the column of $A_{\text{true}}^{(i)}$, and the use of normalization is to remove the scaling degree of freedom for matching the columns of $A^{(i)}$ to the column of $A_{\text{true}}^{(i)}$.

4.1.2 | Generate a synthetic data

To generate a synthetic tensor, we first generate ground truth factor matrices $A_{\text{true}}^{(i)} \in \mathbb{R}_+^{I_i \times r}$, $i = 1, \dots N$ whose entries are sampled from i.i.d. uniform distributions in the interval $[0, 1]$. The tensor $\mathcal{T}^{\text{clean}} \in \mathbb{R}_+^{I_1 \times \cdots \times I_N}$ is then constructed from $A_{\text{true}}^{(i)}$, $i=1, \dots N$. Finally, we form a synthetic data $\mathcal{T}$ by adding some noise to $\mathcal{T}^{\text{clean}}$, $\mathcal{T} = \max(0, \mathcal{T}^{\text{clean}} + \sigma\mathcal{E})$, where $\sigma \geq 0$ is the noise level, and $\mathcal{E} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a tensor whose entries are sampled from a unitary centered normal distribution.

Initialization, number of runs, and plots
For each run of an algorithm, we use a random initialization, that is, the initial factor matrices $A_0^{(i)}$, $i=1, \dots, N$, are generated by sampling uniform distributions in [0,1]. Note that, testing a specific data tensor, we use the same initialization in one run of all algorithms. We run all the algorithms 20 times with 20 different initializations. We stop one run of an algorithm when the maximum time (which is chosen before running the algorithms) is reached.

TABLE 4 List of experiments on nonnegative tensor factorization

Figures	Test description	$[I_1, I_2, I_3, r, \sigma]$
Synthetic data		
2	Cube size, low rank, noiseless	$[50, 50, 50, 10, 0]$
2	Unbalanced size, low rank, noiseless	$[150, 10^3, 50, 12, 0]$
3	Unbalanced size, larger rank, noiseless	$[150, 10^3, 50, 25, 0]$
4	Large cube size, low rank, noisy	$[500, 500, 500, 10, 0.01]$
5	Unbalanced size, low rank, noisy, ill-condition	$[150, 10^3, 50, 12, 0.001]$
6	HER-AO-gradients compared with APG and iBPG	$[150, 10^3, 50, 10, 0.01]$
7	Comparing {HER,Bro,GR,LS}-AHALS	$[50, 50, 50, 10, 0]$ $[150, 10^3, 50, 12, 0.01]$ $[150, 10^3, 50, 25, 0.01]$
Real data		
8	Two HSI images : PaviaU and Indian Pine	$[610, 340, 103, 10]$ $[145, 145, 200, 15]$
9	Big data : black-and-white video sequence	$[153, 238, 1.4 \times 10^4, \{10, 20, 30\}]$

In presenting the results, we plot $f - f_{\min}$; and if the ground truth is known, we also plot $e - e_{\min}$. Here $f_{\min}$ and $e_{\min}$ are respectively the minimal value of all the data fitting errors and the factor fitting errors obtained across all algorithms on all runs. In noiseless settings ($\sigma = 0$), exact factorization is possible, so we set $f_{\min} = 0$. In order to have a better observation of the performance of the algorithms, we plot the curves with respect to both time and iterations[§]. We remark that, "an iteration" for AO algorithms means the counter k of the outer loop after all blocks being updated. Regarding the time evaluation, we record the time stamp for each iteration, and then perform a linear interpolation to synchronize the time curves. Note that such linear interpolation does not reflect 100% truly the real convergence behavior as it is just an linear estimate, but we consider such estimate to be accurate enough.

In our experiment, we emphasize on plotting the median curves of the 20 runs (which are the thick curves in the upcoming figures), because there may be large deviations between different runs.

4.1.3 | Solving the NNLS problem (14) and (15)

When using AS, ADMM, Nesterov's accelerated gradient descent algorithm or AHALS to solve (14) or (15), in our implementation, we terminate the solver when the number of iterations reaches 50 or when $\|A_s^{(i)} - A_{s-1}^{(i)}\| \leq 10^{-2}\|A_0^{(i)} - A_1^{(i)}\|$, where s is the iteration counter of the solver.

Parameter set up for HER

We use the following set of parameters for HER-AO (unless otherwise specified): $\beta_0 = 0.5, \gamma = 1.05, \overline{\gamma} = 1.01, \eta = 1.5$, and (16) is used for the extrapolation point.

List of experiments

Table 4 lists the figures that report our diverse experiments on synthetic data and real datasets. All the experiments have $N = 3$ and the input tensor is dense.

Complete numerical experimental results

We refer the interested reader to the appendix of the arXiv version of the current paper[66] for all the numerical experiments as reported in Table 4. The conclusions remain the same: HER significantly accelerate the convergence of BCD algorithms, while the HER-BCD outperforms both iBPG and APG.

[§]We do not report the number of MTTKRP (Matricized tensor times Khatri-Rao product) as all the algorithms in the experiments (except for AS) share the same number of MTTKRP (which is N for an tensor with order N), so the performance in terms of number of MTTKRP is contained implicitly in the plot with respect to the iterations.

4.2 | Experiments on synthetic datasets

As listed in Table 4, the experiments on synthetic data sets are designed to simulate different kinds of situations that may occur in real applications, which includes : low rank, larger rank, noiseless, noisy, tensor with balanced size (cubic tensor), tensor with unbalanced size (rectangular tensor), and ill-conditioned tensor.

Figures 2–5 illustrate that HER-ADMM and HER-AHALS significantly outperform their counterparts AO-ADMM and AHALS in term of both f_k and e_k. We stress that the improvement is often of several orders of magnitude (at least 10^4 in most cases). We observe the same result for HER-AS and HER-Nesterov vs AO-AS and AO-Nesterov.

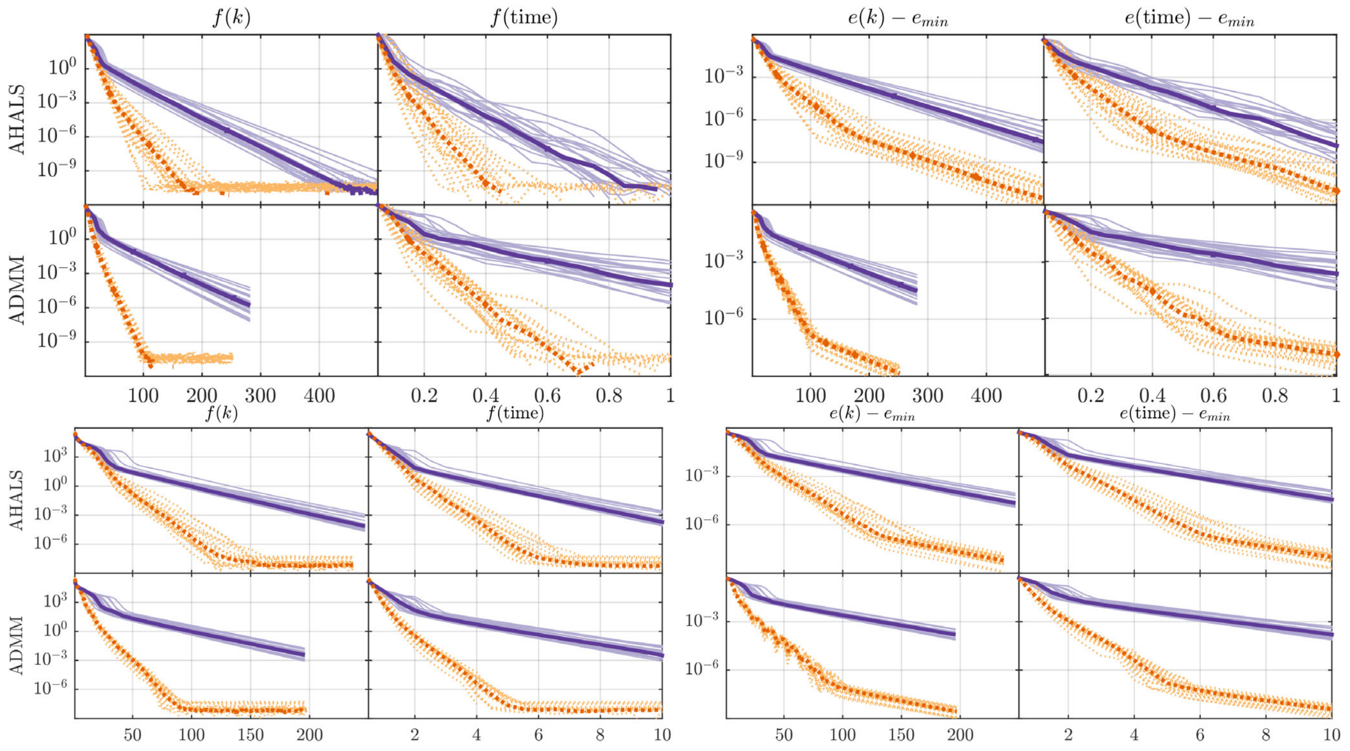

FIGURE 2 Convergence of algorithms : A-HALS and AO-ADMM without heuristic extrapolation with restarts (HER) (solid purple) and with HER (dotted orange), on standard test case (top) : $[I_1, I_2, I_3, r] = [50, 50, 50, 10]$ and unbalanced sizes (bottom) $[I_1, I_2, I_3, r] = [150, 10^3, 50, 12]$. The results show that HER improves the convergence significantly, the convergence in both f and e for HER-accelerated methods are already multiple-order of magnitude better than the un-accelerated algorithms. Notice that due to a higher per-outer-iteration cost, ADMM-based algorithms (AO-ADMM and HER-AO-ADMM) run fewer number of outer-iteration than the AHALS-based algorithms. See the appendix of Reference 66 for the results on other algorithms where we observe a similar behavior

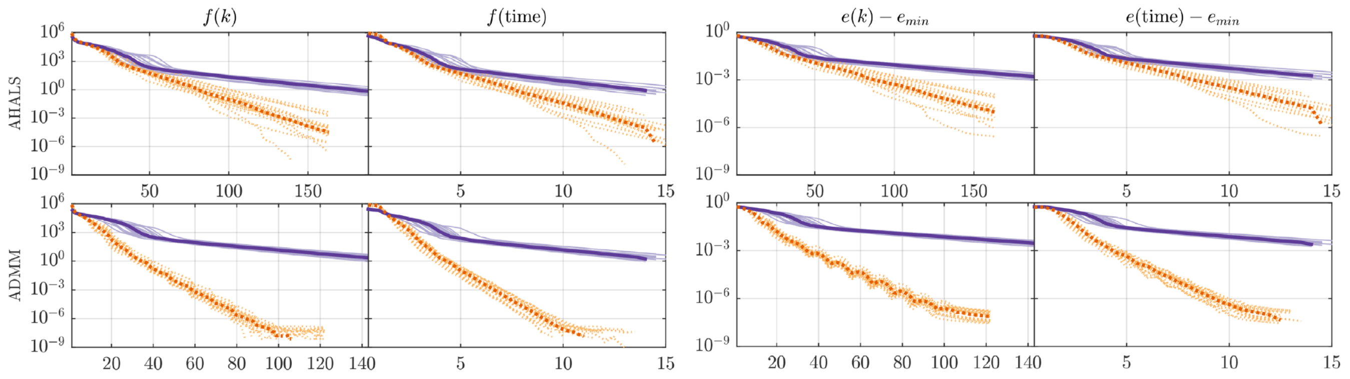

FIGURE 3 On large rank $[I_1, I_2, I_3, r] = [150, 10^3, 50, 25]$. For the plot set up, see Figure 2. The results show that heuristic extrapolation with restarts improves the convergence speed significantly. See Figure 2 for the plot set up, and the appendix of Reference 66 for the results on other algorithms

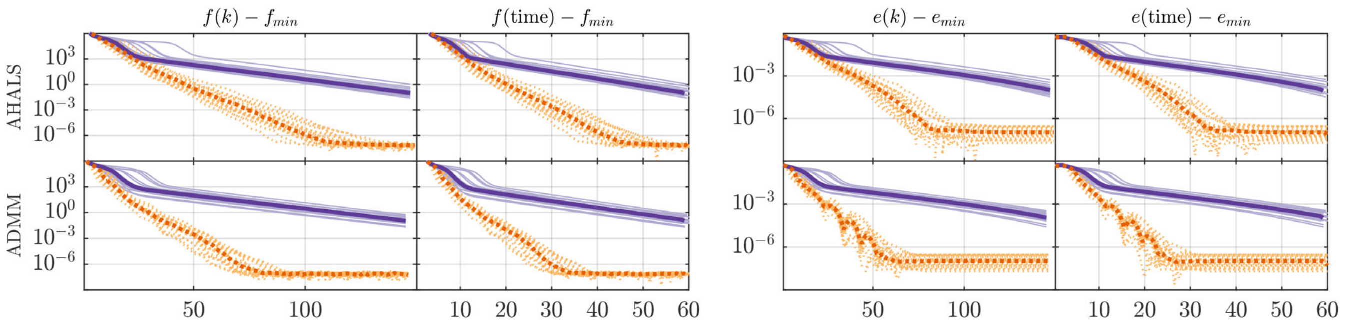

FIGURE 4 On big and noisy case $I_1 = I_2 = I_3 = 500$, $[r, \sigma] = [10, 0.01]$. The results show that heuristic extrapolation with restarts improves the convergence speed significantly. See Figure 2 for the plot set up, and the appendix of Reference 66 for the results on other algorithms

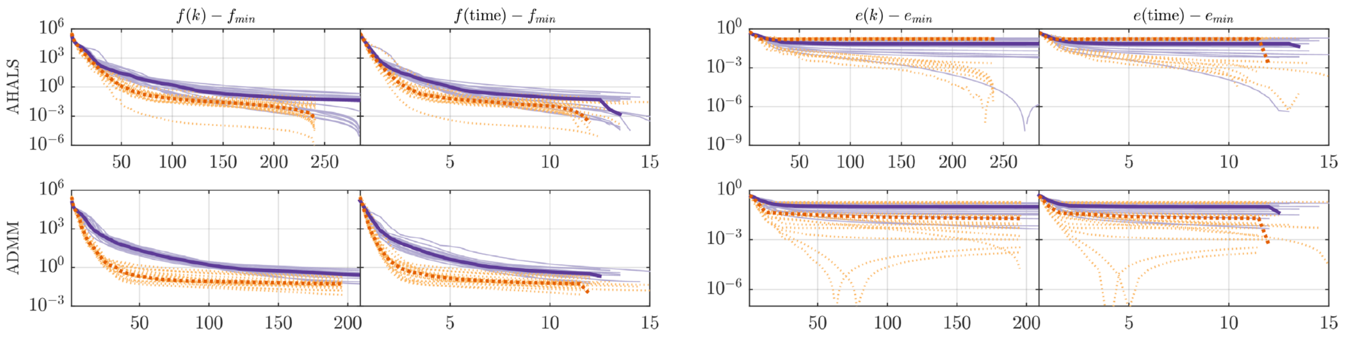

FIGURE 5 On ill-conditioned case $[I_1, I_2, I_3, r, \sigma] = [150, 10^3, 50, 12, 0.01]$, where $A_i(:,1) = 0.99A_i(:,2) + 0.01A_i(:,1)$ for $i \in \{1, 2, 3\}$. For the plot set up, see Figure 2. The results show that heuristic extrapolation with restarts improves the convergence speed. See Figure 2 for the plot set up, and the appendix of Reference 66 for the results on other algorithms

Compared with APG and iBPG, we observe from Figure 6 that HER-Nesterov outperforms both APG and iBPG in term of f and significantly outperforms them in term of e. From extensive experiments, we observe that HER, the scheme that makes use of the extrapolation between block update scheme, shows much better performance than APG and iBPG, the accelerated block proximal gradient methods that use Nesterov-type extrapolation inside each block update.

Compared with Bro-AHALS, GR-AHALS and LS-AHALS, Figure 7 shows that our HER-AHALS performs the best in the three experimental settings (only median is plotted here). Note that since the acceleration frameworks Bro, GR, and LS are not designed for NTF, it is possible the iterates produced by these frameworks are infeasible. Here we only compare HER-AHALS with Bro- AHALS, GR- AHALS, and LS- AHALS; the comparison of these methods where AHALS is replaced with AO-ADMM and AO-ADMM are available in the appendix of Reference 66, and similar conclusions are drawn, namely that HER outperforms the other accelerations.

4.3 | On real data

4.3.1 | Two hyperspectral images

We test the performance of the algorithms on two hyperspectral images (HSI) PaviaU and Indian Pines [#]. They are nonnegative third-order tensor; PaviaU has size $[610, 340, 103]$ with $r = 10$ and Indian Pines has size $[145, 145, 200]$ with $r = 15$. The r chosen are commonly used in practice.

We perform minimal preprocessing on the raw data : NaN or negative values (if any) are replaced by zero. Hence, it is possible the preprocessed data contains many zeros and being ill-conditioned. Figure 8 reports the performance of

[#] Data available from http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes

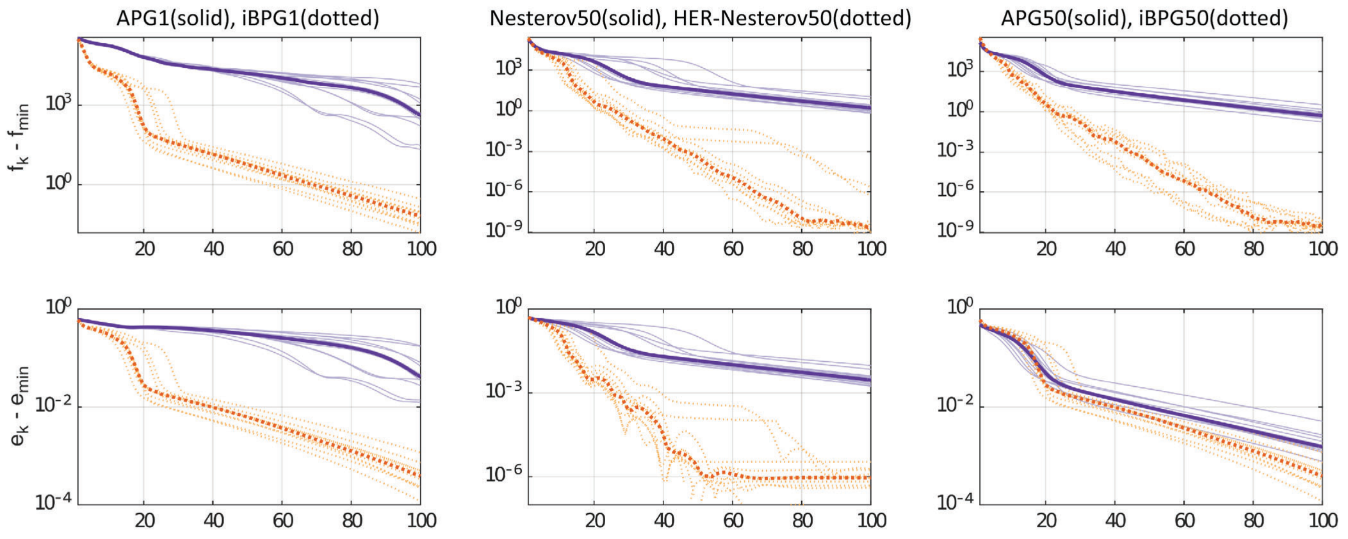

FIGURE 6 Comparing gradient algorithms on $[I_1, I_2, I_3, r, \sigma] = [150, 10^3, 50, 10, 0.01]$. Suffix number denotes the maximum number of inner iterations. Result shows heuristic extrapolation with restarts (HER) works for both inexact and exact block-coordinate descent using gradient. Here HER-Nesterov50 and HER-PGD50 are the best algorithms in both f and e. We do not plot the time plot here as they are similar to the iteration plot

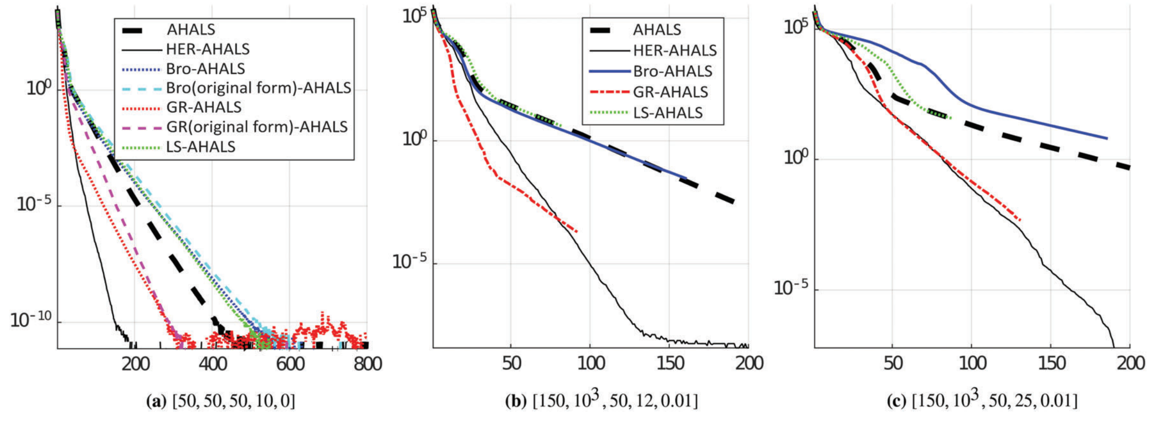

FIGURE 7 Comparing AHALS with different acceleration frameworks on synthetic datasets on 3 setting of $[I_1, I_2, I_3, r, \sigma]$. The curves are the median in $f(k) - f_{\min}$. The x-axis is the number of iteration, and all algorithm run with same run time limited. These results show that that (1) HER-AHALS performs better than all other algorithms, and (2) the modified Bro and GR algorithms perform better than their original counterpart; see subplot(a), while we do not show the result by Bro and GR in their original form in other subplots because they perform worse. Note that LS and GR run less number of iterations due to their larger per-iteration cost. Bro's approach has lower per-iteration cost, but it is even slower than vanilla AHALS. See the appendix of Reference 66 for more results. (a) [50, 50, 50, 10, 0]; (b) [150, 10^3, 50, 12, 0.01]; (c) [150, 10^3, 50, 25, 0.01]

HER-AHALS, HER-ADMM and their counterparts AO-AHALS and AO-ADMM on the two datasets. As there are no ground truth factors, we only show f in the results.

We observe that there are multiple swamps, which are common for real datasets as the data are highly ill-conditioned (the condition numbers of the metricized preprocessed data tensor along all modes are $[593, 642, 1009]$ for Indian Pines and $[944, 462, 8083]$ for PaviaU). Nevertheless, considering the "best case" among the trials, HER-AHALS and HER-ADMM provide solutions with error $10^8 - 10^{10}$ times smaller than the best case of their unaccelerated counterparts. To compare with other algorithms, the readers can view the results in the appendix of Reference 66. We observe that iBPG, APG and the AO (AO-AHALS and AO-ADMM) algorithms accelerated by GR, Bro, and LS schemes are much slower than our AO (AO-AHALS and AO-ADMM) algorithms accelerated by HER. GR-AO and Bro-AO (for AO being AO-AHALS or AO-ADMM) even sometimes diverge.

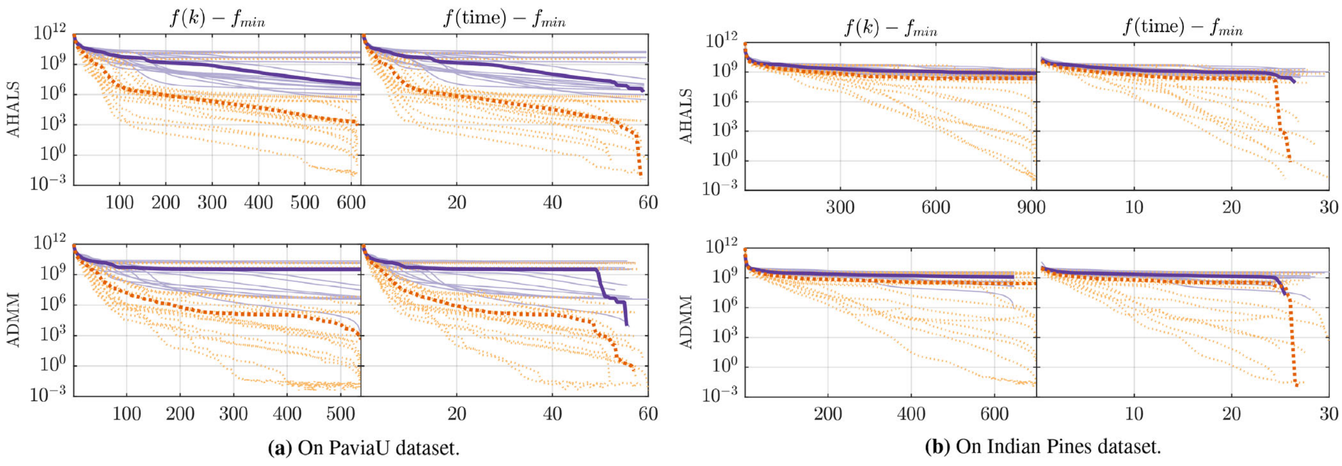

FIGURE 8 The results on HSI data. For the plot set up, see Figure 2. Results show heuristic extrapolation with restarts improve convergences. See the appendix of Reference 66 for more results

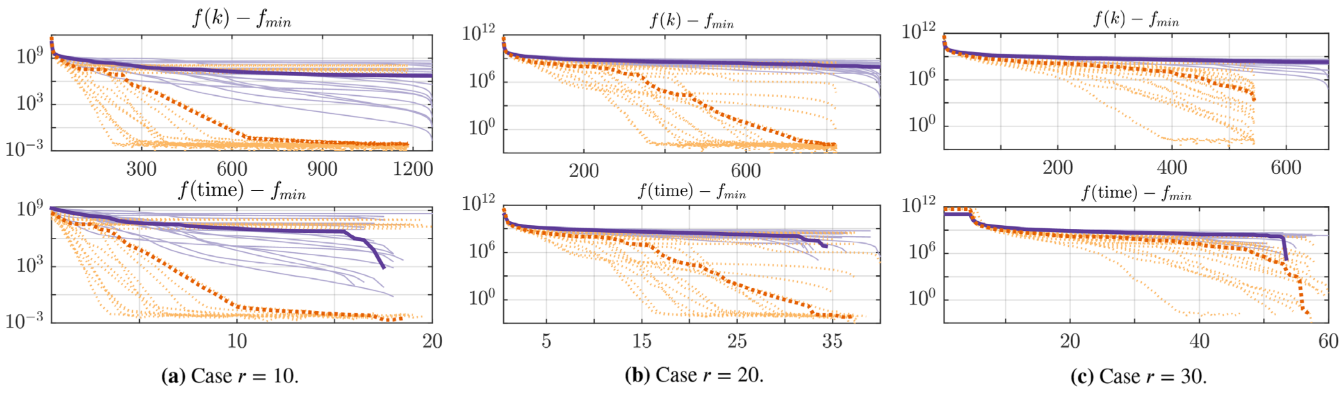

FIGURE 9 On video data [153, 238, 14, 000] for three values of r. Results show heuristic extrapolation with restarts improve convergence and works well with Tucker-based compression. (a) Case $r = 10$; (b) Case $r = 20$; (c) Case $r = 30$

4.3.2 | On big data : video sequences

We test HER-AHALS on the video data of the UCSD Anomaly Dataset.[74] Constructed by combining all the frame images of 70 surveillance video in the dataset, we have a tensor with sizes $153 \times 238 \times 14{,}000$, where the first two modes are the screen resolution and the third mode is the number of frame. No pre-processing is performed on the raw data. Data of such size is too big to store in our computer memory, so we perform compression using Tucker decomposition, based on the built-in function from the Tensor toolbox.[75] We compare AHALS and HER-AHALS with $r \in \{10, 20, 30\}$. Results in Figure 9 shows that HER-AHALS performs much better than AHALS. For the details on how HER works with Tucker compression, see the appendix of Reference 66. As a conclusion for this section, we give some remarks on HER-AO. From our extensive experiments, we observe that HER-AS has inferior performance than others when the data is either big in size, high rank, or ill-conditioned. When the data has small size, all HER-AO algorithms have similar performance, and they all outperform their un-accelerated counterpart algorithms in term of both time and iteration. Among HER-AO algorithms, we highly recommend HER-AHALS for NTF as it shows good performance in all experiments.

5 | DISCUSSION AND CONCLUSION

In this paper, we have proposed an extrapolation strategy in-between block updates, referred to as HER, for improving the empirical convergence speed of BCD algorithms for approximate NTF. HER significantly accelerates the empirical convergence speed of most existing block-coordinate algorithms for dense NTF, in particular for challenging computational scenarios, while requiring a negligible additional computational budget. The core of HER is to apply a special extrapolation-restart mechanism that aims to reduce the computational cost of restart while making sure the restart criterion follows the standard function restarts. The performance of HER was verified by the experiments reported in this paper. In all scenarios, HER-AHALS provides among the best results hence we recommend its use in practice.

Future works include deriving theoretical convergence for HER, and to apply it on other challenging applications.

ACKNOWLEDGEMENTS

This work was supported by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen (FWO) under EOS Project no O005318F-RG47, and by the European Research Council (ERC starting grant no 679515).

CONFLICT OF INTEREST

This study does not have any conflicts to disclose.

AUTHOR CONTRIBUTIONS

Andersen Man Shun Ang, Jeremy E. Cohen, Nicolas Gillis, and Le Thi Khanh Hien designed the study. Andersen Man Shun Ang, Jeremy E. Cohen and Nicolas Gillis developed the methodology. Andersen Man Shun Ang collected the data, performed the analysis. Andersen Man Shun Ang, Jeremy E. , Nicolas Gillis, and Le Thi Khanh Hien wrote the manuscript.

DATA AVAILABILITY STATEMENT

Datasets in the experiments come from publicly available sources, which can be find online. The code will be public too, see https://angms.science/research.html

ORCID

Andersen Man Shun Ang https://orcid.org/0000-0002-8330-758X
Nicolas Gillis https://orcid.org/0000-0001-6423-6897

REFERENCES

1. Friedland S. On the generic and typical ranks of 3-tensors. Linear Algebra Appl. 2012;436(3):478–97.
2. Chiantini L, Ottaviani G. On generic identifiability of 3-tensors of small rank. SIAM J Matrix Anal Appl. 2012;33(3):1018–37.
3. Carroll JD, Chang JJ. Analysis of individual differences in multidimensional scaling via an n-way generalization of “Eckart-Young” decomposition. Psychometrika. 1970;35(3):283–319.
4. Harshman RA. Foundations of the PARAFAC procedure: model and conditions for an “explanatory” multi-mode factor analysis. UCLA Work Pap Phon. 1970;11(16):1–84.
5. Kiers HAL. Towards a standardized notation and terminology in multiway analysis. J Chemom. 2000;14(3):105–22.
6. Vavasis SA. On the complexity of nonnegative matrix factorization. SIAM J Optim. 2010;20(3):1364–77.
7. Lim LH, Comon P. Nonnegative approximations of nonnegative tensors. J Chemom. 2009;23(7-8):432–41.
8. Qi Y, Comon P, Lim LH. Uniqueness of nonnegative tensor approximations. IEEE Trans Inf Theory. 2016;62(4):2170–83.ArXiv:1410.8129.
9. Bro R. Multi-way analysis in the food industry: models, algorithms, and applications. Amsterdam, The Netherlands: University of Amsterdam; 1998.
10. Mitchell D, Ye N, De Sterck H. Nesterov acceleration of alternating least squares for canonical tensor decomposition: momentum step size selection and restart mechanisms. Numer Linear Algebra Appl. 2020;27(4):1–24.
11. Ang AMS, Gillis N. Accelerating nonnegative matrix factorization algorithms using extrapolation. Neural Comput. 2019;31(2):417–39.
12. Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math Program. 2016;156(1-2):59–99.
13. Su W, Boyd S, Candès EJ. A differential equation for modeling Nesterov’s accelerated gradient method: theory and insights. J Mach Learn Res. 2016;17(1):5312–54.
14. Liavas AP, Kostoulas G, Lourakis G, Huang K, Sidiropoulos ND. Nesterov-based alternating optimization for nonnegative tensor factorization: algorithm and parallel implementation. IEEE Trans Signal Process. 2017;66(4):944–53.
15. Hitchcock FL. The expression of a tensor or a polyadic as a sum of products. J Math Phys. 1927;6(1-4):164–89.
16. Hitchcock FL. Multiple invariants and generalized rank of a P-way matrix or tensor. J Math Phys. 1928;7(1-4):39–79.
17. Cattell RB. εParallel proportional profilesε and other principles for determining the choice of factors by rotation. Psychometrika. 1944;9(4):267–83.
18. Cattell RB. The three basic factor-analytic designs–their interrelations and derivatives. Psychol Bull. 1952;10(49):499–520.

19. Tucker LR. The extension of factor analysis to three-dimensional matrices. Contributions to mathematical psychology. New York, NY: Holt, Rinehart and Winston; 1964. p. 110–27.
20. Tucker LR. Some mathematical notes on three-mode factor analysis. Psychometrika. 1966;31(3):279–311.
21. Smilde AK, Bro R, Geladi P. Multi way analysis–applications in chemical sciences. West Sussex, England: Wiley; 2004.
22. De Lathauwer L, De Moor B. From matrix to tensor: multilinear algebra and signal processing. Proceedings of the Digest of the 4th IMA International Conference on Mathematics in Signal Process. University of Warwick, Coventry, England; 1996. p. 1–11.
23. Cichocki A, Mandic D, De Lathauwer L, Zhou G, Zhao Q, Caiafa C, et al. Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Signal Process Mag. 2015;32(2):145–63.
24. Bader BW, Berry MW, Browne M. Discussion tracking in Enron email using PARAFAC. Survey of text mining II. London: Springer; 2008. p. 147–63.
25. Mørup M. Applications of tensor (multiway array) factorizations and decompositions in data mining. Wiley Interdiscip Rev Data Min Knowl Discov. 2011;1(1):24–40.
26. Kolda T, Bader B. Tensor decompositions and applications. SIAM Rev. 2009;51(3):455–500.
27. Kim J, He Y, Park H. Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J Glob Optim. 2014;58(2):285–319.
28. Kroonenberg PM. Applied multiway data analysis. Vol 702. West Sussex, England: Wiley; 2008.
29. Anandkumar A, Ge R, Hsu D, Kakade SM, Telgarsky M. Tensor decompositions for learning latent variable models. J Mach Learn Res. 2014;15(1):2773–832.
30. Sidiropoulos ND, De Lathauwer L, Fu X, Huang K, Papalexakis EE, Faloutsos C. Tensor decomposition for signal processing and machine learning. IEEE Trans Signal Process. 2017;65(13):3551–82.
31. De Lathauwer L, De Moor B, Vandewalle J. A multilinear singular value decomposition. SIAM J Matrix Anal Appl. 2000;21(4):1253–78.
32. Oseledets IV. Tensor-train decomposition. SIAM J Sci Comput. 2011;33(5):2295–317.
33. Paatero P, Tapper U. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics. 1994;5(2):111–26.
34. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;401(6755):788–91.
35. Cichocki A, Zdunek R, Phan AH, Amari S. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. Hoboken, NJ: John Wiley & Sons; 2009.
36. Gillis N. The why and how of nonnegative matrix factorization. Regularization, optimization, kernels, and support vector machines. Volume Machine Learning and Pattern Recognition Series, 12; 2014. p. 257–91. London: Chapman and Hall/CRC.
37. Fu X, Huang K, Sidiropoulos ND, Ma W. Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. IEEE Signal Process Mag. 2019;36(2):59–80.
38. Wetzstein G, Lanman D, Hirsch M, Raskar R. Tensor displays: Compressive Light Field Synthesis using Multilayer Displays with Directional Backlighting. SIGGRAPH '12: ACM SIGGRAPH 2012 Emerging Technologies. New York, NY: Association for Computing Machinery; 2012, 2012. p. 24, 1–1.
39. Carroll JD, De Soete G, Pruzansky S. Multiway data analysis. Amsterdam, The Netherlands: North-Holland Publishing Co.; 1989.p. 463–72.
40. Shashua A, Hazan T. Non-negative tensor factorization with applications to statistics and computer vision. Proceedings of the 22nd International Conference on Machine Learning. Bonn, Germany; 2005. p. 792–799.
41. Fitzgerald D, Cranitch M, Coyle E. Non-negative tensor factorisation for sound source separation. Proceedings of the IET Conference. Dublin, Ireland; January 2005. p. 8–12.
42. Welling M, Weber M. Positive tensor factorization. Pattern Recogn Lett. 2001;22(12):1255–61.
43. Chi EC, Kolda TG. On tensors, sparsity, and nonnegative factorizations. SIAM J Matrix Anal Appl. 2012;33(4):1272–99.
44. Bro R, De Jong S. A fast non-negativity-constrained least squares algorithm. J Chemom. 1997;11(5):393–401.
45. Friedlander MP, Hatz K. Computing non-negative tensor factorizations. Optim Methods Softw. 2008;23(4):631–47.
46. Hazan T, Polak S, Shashua A. Sparse image coding using a 3D non-negative tensor factorization. Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV'05). Beijing, China; vol. 1 2005. p. 50-57.
47. Brewer J. Kronecker products and matrix calculus in system theory. IEEE Trans Circuits Syst. 1978;25(9):772–81.
48. Cohen JE. About notations in multiway array processing; 2015. arXiv preprint.
49. De Lathauwer L, De Moor B, Vandewalle J. Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition. SIAM J Matrix Anal Appl. 2004;26(2):295–327.
50. Luciani X, Albera L. Semi-algebraic canonical decomposition of multi-way arrays and joint eigenvalue decomposition. Processing of the 2011 IEEE International Conference on Acoustics, Speech and Signal (ICASSP). IEEE; 2011. p. 4104–4107.
51. Domanov I, De Lathauwer L. Canonical polyadic decomposition of third-order tensors: reduction to generalized eigenvalue decomposition. SIAM J Matrix Anal Appl 2014;35(2):636–660.
52. Beltrán C, Breiding P, Vannieuwenhoven N. Pencil-based algorithms for tensor rank decomposition are not stable. SIAM J Matrix Anal Appl. 2019;40(2):739–73.
53. Vannieuwenhoven N. Condition numbers for the tensor rank decomposition. Linear Algebra Appl. 2017;535:35–86.
54. Silva VD, Lim LH. Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM J Matrix Analysis Appl. 2008;30(3):1084–127.
55. Paatero P. A weighted non-negative least squares algorithm for three-way εPARAFACε factor analysis. Chemom Intell Lab Syst. 1997;38(2):223–42.

56. Vervliet N. Compressed sensing approaches to large-scale tensor decompositions. Belgium: Katholieke Universiteit Leuven; 2018.
57. Acar E, Dunlavy DM, Kolda TG. A scalable optimization approach for fitting canonical tensor decompositions. J Chemom. 2011;25(2):67–86.
58. Huang K, Sidiropoulos ND, Liavas AP. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. IEEE Trans Signal Process. 2016;64(19):5052–65.
59. Royer JP, Thirion-Moreau N, Comon P. Nonnegative 3-way tensor factorization taking in to account possible missing data. Proceedings of the 2012 20th European Signal Processing Conference (EUSIPCO). Bucharest, Romania: IEEE; 2012. p. 71–75.
60. Xu Y, Yin W. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. SIAM J Imag Sci. 2013;6(3):1758–89.
61. Hien LTK, Gillis N, Patrinos P. Inertial block proximal method for non-convex non-smooth optimization. Proceedings of the 37th International Conference on Machine Learning (ICML); 2020. p. 1–11.
62. Zhang Y, Zhou G, Zhao Q, Cichocki A, Wang X. Fast nonnegative tensor factorization based on accelerated proximal gradient and low-rank approximation. Neurocomputing. 2016;198:148–54.
63. Guan N, Tao D, Luo Z, Yuan B. NeNMF: an optimal gradient method for nonnegative matrix factorization. IEEE Trans Signal Process. 2012;60(6):2882–98.
64. Gillis N, Glineur F. Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. Neural Comput. 2012;24(4):1085–105.
65. Razaviyayn M, Hong M, Luo Z. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. SIAM J Optim. 2013;23(2):1126–53.
66. Ang AMS, Cohen JE, Gillis N, Khanh Hien LT. Accelerating block coordinate descent for nonnegative tensor factorization; 2020. arXiv e-prints. arXiv:2001.04321.
67. Ravindran N, Sidiropoulos ND, Smith S, Karypis G. Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition. Proceedings of the 2014 48th Asilomar Conference on Signals, Systems and Computers. California, CA: IEEE; 2014. p. 581-585.
68. Ballard G, Hayashi K, Ramakrishnan K. Parallel nonnegative cp decomposition of dense tensors. Proceedings of the 2018 IEEE 25th International Conference on High Performance Computing (HiPC). Bengaluru, India; 2018. p. 22-31.
69. Vervliet N, Debals O, De Lathauwer L. Exploiting efficient representations in large-scale tensor decompositions. SIAM J Sci Comput. 2019;41(2):A789–815.
70. Battaglino C, Ballard G, Kolda TG. A practical randomized CP tensor decomposition. SIAM J Matrix Anal Appl. 2018;39(2):876–901.
71. Gong X, Mohlenkamp MJ, Young TR. The optimization landscape for fitting a rank-2 tensor with a rank-1 tensor. SIAM J Appl Dyn Syst. 2018;17(2):1432–77.
72. Ang A, Cohen J, Gillis N. Accelerating approximate nonnegative canonical polyadic decomposition using extrapolation. Proceedings of the GRETSI 2019 - XXVIIéme Colloque francophone de traitement du signal et des images. Lille, France: Gretsi; 2019. p. 1–4.
73. Smith S, Karypis G. Tensor-matrix products with a compressed sparse tensor. Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms. Austin, TX; 2015. p. 1–7.
74. Mahadevan V, Li W, Bhalodia V, Vasconcelos N. Anomaly detection in crowded scenes. Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco, CA: IEEE; 2010. p. 1975–1981.
75. Bader BW, Kolda TG. Efficient MATLAB computations with sparse and factored tensors. SIAM J Sci Comput. 2007;30(1):205–31.

How to cite this article: Man Shun Ang A, Cohen JE, Gillis N, Thi Khanh Hien L. Accelerating block coordinate descent for nonnegative tensor factorization. *Numer Linear Algebra Appl.* 2021;28:e2373. https://doi.org/10.1002/nla.2373