Received: 7 June 2019 | Revised: 30 November 2019 | Accepted: 30 January 2020

DOI: 10.1002/nla.2297

RESEARCH ARTICLE

WILEY

Nesterov acceleration of alternating least squares for canonical tensor decomposition: Momentum step size selection and restart mechanisms

Drew Mitchell[1] | **Nan Ye**[2] | **Hans De Sterck**[3]

[1]School of Mathematical Sciences, Monash University, Melbourne, Australia

[2]School of Mathematics and Physics, University of Queensland, Brisbane, Australia

[3]Department of Applied Mathematics, University of Waterloo, Waterloo, Canada

Correspondence
Hans De Sterck, Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada.
Email: hdesterck@uwaterloo.ca

Summary

We present Nesterov-type acceleration techniques for alternating least squares (ALS) methods applied to canonical tensor decomposition. While Nesterov acceleration turns gradient descent into an optimal first-order method for convex problems by adding a momentum term with a specific weight sequence, a direct application of this method and weight sequence to ALS results in erratic convergence behavior. This is so because ALS is accelerated instead of gradient descent for our nonconvex problem. Instead, we consider various restart mechanisms and suitable choices of momentum weights that enable effective acceleration. Our extensive empirical results show that the Nesterov-accelerated ALS methods with restart can be dramatically more efficient than the stand-alone ALS or Nesterov's accelerated gradient methods, when problems are ill-conditioned or accurate solutions are desired. The resulting methods perform competitively with or superior to existing acceleration methods for ALS, including ALS acceleration by nonlinear conjugate gradient, nonlinear generalized minimal residual method, or limited-memory Broyden-Fletcher-Goldfarb-Shanno, and additionally enjoy the benefit of being much easier to implement. We also compare with Nesterov-type updates where the momentum weight is determined by a line search (LS), which are equivalent or closely related to existing LS methods for ALS. On a large and ill-conditioned $71\times1,000\times900$ tensor consisting of readings from chemical sensors to track hazardous gases, the restarted Nesterov-ALS method shows desirable robustness properties and outperforms any of the existing methods we compare with by a large factor. There is clear potential for extending our Nesterov-type acceleration approach to accelerating other optimization algorithms than ALS applied to other nonconvex problems, such as Tucker tensor decomposition.

KEYWORDS

alternating least squares, canonical tensor decomposition, Nesterov acceleration, nonlinear acceleration, nonlinear optimization, nonlinear preconditioning

This is a revised and extended version of paper arXiv:1810.05846v1 (2018).

1 INTRODUCTION

Nesterov's accelerated gradient descent (GD) method is a celebrated method for speeding up the convergence rate of GD, achieving the optimal convergence rate obtainable for first-order methods on convex problems.[1] Nesterov's method is an extrapolation method that specifies carefully tuned weight sequences for the so-called momentum term which updates an iterate in the direction of the previous update. With those tailored weight sequences, optimal convergence rates can be proved for convex problems.

Recent work has seen extensions of Nesterov's accelerated gradient method in several ways: either the method is extended to nonconvex optimization problems,[2,3] or Nesterov's approach is applied to accelerate convergence of methods that are not directly of GD-type, such as the alternating direction method of multipliers (ADMM) applied to convex problems.[4]

In this paper our goal is to extend Nesterov extrapolation to accelerating the alternating least squares (ALS) method for computing the canonical approximation of a tensor by a sum of R rank-1 tensors—the so-called canonical polyadic (CP) decomposition of a tensor.[5] As such, this paper attacks two challenges at the same time: we develop Nesterov-accelerated algorithms for a nonconvex problem— the CP tensor decomposition problem—, and we do this by accelerating ALS steps instead of GD steps. Since we accelerate ALS instead of GD for our nonconvex CP decomposition problem, we cannot simply use the standard sequences for the Nesterov momentum weights that lead to optimal convergence of Nesterov's accelerated gradient method applied to convex problems; in fact, we will illustrate that using these sequences leads to erratic or divergent convergence behaviour. Instead, we investigate in this paper choices for selecting the momentum step size combined with restarting mechanisms that lead to effective acceleration methods for ALS applied to tensor decomposition. Our approach is partially inspired by recent related work on acceleration methods for nonlinear systems[6] and on restarting mechanisms that improve Nesterov convergence for convex problems.[4,7,8] Our approach and goals are most closely related to (independent) recent work by Ang and Gillis,[9] who consider Nesterov-type acceleration for nonnegative matrix factorization, and also propose step size and restart mechanisms, which are different from ours.

The Nesterov acceleration methods for ALS with our proposed step size and restart mechanisms are attractive because they are simple to implement, and we show in extensive numerical tests that they are competitive with other recently proposed nonlinear acceleration methods for ALS applied to CP tensor decomposition, most of which are much more involved in terms of implementation. We also believe our step size and restart strategies are of broader interest and may be applied to Nesterov-type acceleration of other simple optimization methods of alternating or (block) coordinate descent-type applied to potentially nonconvex problems.

As another contribution of this paper, we also establish links between Nesterov-type acceleration of ALS and other existing acceleration methods for ALS. We establish links between Nesterov-type extrapolation and the well-known nonlinear acceleration methods of Anderson acceleration and the nonlinear generalized minimal residual method (NGMRES).[10–13] In particular, we show that the Nesterov extrapolation formula is equivalent to Anderson acceleration with window size 1, and closely related to NGMRES with window size 1. In our numerical results, we compare with NGMRES acceleration of ALS. Furthermore, if the Nesterov momentum weight is determined by a LS, Nesterov acceleration of ALS is closely related to LS methods for ALS that go back to the 1970s and have more recently been enhanced.[14–17] We note that acceleration of ALS by NGMRES, nonlinear conjugate gradient (NCG), and Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) also employs LSs.[12,18,19] To explore the link with LS methods,[14–17] we also compare in our numerical results with Nesterov weights determined by a cubic LS. Line searches are common in acceleration methods for ALS, but are not commonly considered when Nesterov acceleration is applied to optimization algorithms in the literature, so we argue based on our numerical results that Nesterov momentum weights determined by a LS may be a useful algorithmic approach. Finally, we also explain how Nesterov acceleration of ALS can be interpreted as using ALS as a nonlinear preconditioner for Nesterov's accelerated gradient formula, following a general framework for nonlinear preconditioning of optimization methods that was previously applied to the NCG and LBFGS methods.[19]

1.1 Nesterov's accelerated gradient method

Consider the problem of minimizing a function $f(x)$,

$$\min_x f(x). \tag{1}$$

Nesterov's accelerated GD starts with an initial guess x_1. For $k \geq 1$, given x_k, a new iterate x_{k+1} is obtained by first adding a multiple of the *momentum* $x_k - x_{k-1}$ to x_k to obtain an auxiliary variable y_k, and then performing a GD step at y_k. The update equations at iteration $k \geq 1$ are as follows:

$$y_k = x_k + \beta_k(x_k - x_{k-1}), \tag{2}$$

$$x_{k+1} = y_k - \alpha_k \nabla f(y_k), \tag{3}$$

where the GD step length α_k and the momentum weight β_k are suitably chosen numbers, and $x_0 = x_1$ so that the first iteration is simply GD.

There are a number of ways to choose the α_k and β_k so that Nesterov's accelerated GD converges at the optimal $O(1/k^2)$ in function value for smooth convex functions. For example, when $f(x)$ is a convex function with L-Lipschitz gradient, by choosing $\alpha_k = \frac{1}{L}$, and β_k as

$$\lambda_0 = 0, \quad \lambda_k = \frac{1 + \sqrt{1 + 4\lambda_{k-1}^2}}{2}, \tag{4}$$

$$\beta_k = \frac{\lambda_{k-1} - 1}{\lambda_k}. \tag{5}$$

one obtains the following $O(1/k^2)$ convergence rate:

$$f(x_k) - f(x^*) \leq \frac{2L\|x_1 - x^*\|}{k^2}, \tag{6}$$

where x^* is a minimizer of f. See, for example, Su et al.[8] for more discussion on the choices of momentum weights.

It will be useful for later considerations in this paper to rewrite the expressions for Nesterov acceleration in Equations (2) and (3) solely in terms of x_k or y_k. For the x_k variables, we obtain

$$\begin{aligned} x_k &= q_{\nabla,k-1}(y_{k-1}), \\ &= q_{\nabla,k-1}(x_{k-1} + \beta_{k-1}(x_{k-1} - x_{k-2})), \end{aligned} \tag{7}$$

where the function $q_{\nabla,k-1}(\cdot)$ is introduced to represent the result of one steepest-descent step with step size α_{k-1}:

$$q_{\nabla,k-1}(x) = x - \alpha_{k-1}\nabla f(x). \tag{8}$$

Similarly, for the y_k variables, we can write

$$y_k = q_{\nabla,k-1}(y_{k-1}) + \beta_k \left(q_{\nabla,k-1}(y_{k-1}) - q_{\nabla,k-2}(y_{k-2})\right). \tag{9}$$

Recently, there are a number of works that apply Nesterov's acceleration technique to nonconvex problems. A modified Nesterov-accelerated GD method has been developed that enjoys the same convergence guarantees as GD on nonconvex optimization problems, and maintains the optimal first-order convergence rate on convex problems.[2] A Nesterov-accelerated proximal gradient algorithm was developed that is guaranteed to converge to a critical point, and maintains the optimal first-order convergence rate on convex problems.[3]

Nesterov's accelerated gradient method is known to exhibit oscillatory behavior on convex problems. An interesting discussion on this is provided by Su et al.[8] which formulates an ODE as the continuous time analogue of Nesterov's method. Such oscillatory behavior happens when the method approaches convergence, and can be alleviated by restarting the algorithm using the current iterate as the initial solution, usually resetting the sequence of momentum weights to its initial state close to 0. An explanation has been provided of why resetting the momentum weight to a small value is effective using the ODE formulation of Nesterov's accelerated GD.[8] The use of adaptive restarting has been explored for convex problems,[7] and Nguyen et al.[6] explored the use of adaptive restarting and adaptive momentum weights for nonlinear systems of equations resulting from finite element approximation of PDEs. Our work is the first study of a general Nesterov-accelerated ALS scheme.

1.2 | Nesterov extrapolation as a generic nonlinear acceleration scheme

We can now generalize the Nesterov extrapolation approach and apply it to accelerate other optimization methods than steepest descent. Considering a generic iterative optimization method with update formula

$$x_k = q(x_{k-1}),$$

we replace the steepest-descent operator $q_{\nabla,k-1}(\cdot)$ by the generic update function $q(\cdot)$ in Nesterov update formulas Equation (7) or Equation (9) to obtain

$$x_k = q\left(x_{k-1} + \beta_{k-1}(x_{k-1} - x_{k-2})\right), \tag{10}$$

$$y_k = q(y_{k-1}) + \beta_k\left(q(y_{k-1}) - q(y_{k-2})\right). \tag{11}$$

This extrapolation approach has been used previously to accelerate, for example, ADMM, where $q(\cdot)$ represents an ADMM update.[4] Nesterov's technique has also been used to accelerate an approximate Newton method.[20]

In this paper, we consider Nesterov-type acceleration of ALS for CP tensor decomposition. We write $q_{\text{ALS}}(\cdot)$ for the ALS update function, and obtain accelerated formulas

$$x_k = q_{\text{ALS}}\left(x_{k-1} + \beta_{k-1}(x_{k-1} - x_{k-2})\right), \tag{12}$$

and

$$y_k = q_{\text{ALS}}(y_{k-1}) + \beta_k\left(q_{\text{ALS}}(y_{k-1}) - q_{\text{ALS}}(y_{k-2})\right). \tag{13}$$

Replacing gradient directions by update directions provided by ALS is essentially also an approach that has been taken to obtain nonlinear acceleration of ALS by NGMRES, NCG, and LBFGS;[12,18,19] in the case of Nesterov's method the procedure is extremely simple and easy to implement.

In most of what follows, we will focus on the update formula in the y variables, and we will simplify notation for the case of ALS acceleration by defining

$$\bar{y}_k = q_{\text{ALS}}(y_{k-1}) \tag{14}$$

and writing the update formula for Nesterov-accelerated ALS as

$$y_k = \bar{y}_k + \beta_k\left(\bar{y}_k - \bar{y}_{k-1}\right). \tag{15}$$

Recently, the application of Nesterov acceleration to ALS for canonical tensor decomposition was considered by Wang et al.[21] However, they only tried the vanilla Nesterov technique with a standard Nesterov momentum sequence β_k and without restarting or LS mechanisms and not surprisingly they fail to obtain acceleration of ALS. The main goal of this paper is to investigate suitable choices for the momentum weights β_k in Equation (15), and for restart mechanisms that enable effective convergence patterns when applying the Nesterov-ALS method of Equation (15).

1.3 | Canonical tensor decomposition

Tensor decomposition has wide applications in machine learning, signal processing, numerical linear algebra, computer vision, natural language processing and many other fields.[5] This paper focuses on the CP decomposition of tensors,[5] which is also called the CANDECOMP/PARAFAC decomposition. CP decomposition approximates a given tensor $T \in \mathbb{R}^{I_1\times\cdots\times I_N}$

by a low-rank tensor composed of a sum of R rank-1 terms, $\tilde{T} = \sum_{i=1}^{R} a_1^{(i)} \circ \cdots \circ a_N^{(i)}$, where $\circ$ is the vector outer product. Specifically, defining the factor matrices $A_i = (a_i^{(1)}, \dots, a_i^{(R)})$, we minimize the error in the Frobenius norm by considering the objective function

$$f(A_1, \dots, A_N) = \frac{1}{2}\left\| T - \sum_{i=1}^{R} a_1^{(i)} \circ \cdots \circ a_N^{(i)} \right\|_F^2. \tag{16}$$

Finding efficient methods for computing tensor decomposition is an active area of research, but the ALS algorithm is still considered one of the most efficient algorithms for CP decomposition.[22] Alternative optimization methods for CP that have been considered in the literature include quasi-Newton methods such as NCG and LBFGS,[22] nonlinear least-squares approaches using Gauss–Newton and Levenberg–Marquardt algorithms,[23–25] and stochastic GD.[26]

ALS finds a CP decomposition in an iterative way. In each iteration, ALS sequentially updates a block of variables at a time by minimizing expression (16), while keeping the other blocks fixed: first A_1 is updated, then A_2, and so on. Updating a factor matrix A_i is a linear least-squares problem that can be solved in closed form. The ALS update equations for the A_i can be derived by considering expressions for the gradient of $f(A_1, \dots, A_N)$. For example, for a tensor of order 3, with factor matrices A, B, C of sizes $I \times R$, $J \times R$, $K \times R$, expressions for the gradient of f are given by Acar et al.[22]

$$\frac{\partial f}{\partial A} = -T_1(B \odot C) + A\left((B^T B) * (C^T C)\right), \tag{17}$$

$$\frac{\partial f}{\partial B} = -T_2(A \odot C) + B\left((A^T A) * (C^T C)\right), \tag{18}$$

$$\frac{\partial f}{\partial C} = -T_3(A \odot B) + C\left((A^T A) * (B^T B)\right), \tag{19}$$

where $\odot$ denotes the Khatri-Rao product (the "matching columnwise" Kronecker product[5,22]), $*$ denotes component-wise multiplication, and T_i is the matricized version of tensor T in direction i.[5,22] For example, in the first equation, T_1 is of size $I \times JK$, and $B \odot C$ is of size $JK \times R$. The $I \times R$ matrix $T_1(B \odot C)$ is called a "matricized-tensor times Khatri-Rao product" (often abbreviated as "MTTKRP").

The update equations for ALS can be derived from setting each of the gradient components in (17)–(19) equal to zero:[22]

$$T_1(B \odot C) = A\left((B^T B) * (C^T C)\right), \tag{20}$$

$$T_2(A \odot C) = B\left((A^T A) * (C^T C)\right), \tag{21}$$

$$T_3(A \odot B) = C\left((A^T A) * (B^T B)\right). \tag{22}$$

In each ALS iteration, the A, B, C factor matrices are sequentially updated by solving these systems in order for A, B, and C. It can be shown that these are the normal equations for minimizing $f(A, B, C)$ in A, B, and C, respectively, while keeping the other factor matrices constant.[22] For example, Equation (20) are the normal equations for minimizing the least-squares functional $f(A, B, C) = \frac{1}{2}\|T - \tilde{T}\|_F^2 = \frac{1}{2}\|T_1 - A(B \odot C)^T\|_F^2$ with respect to A, keeping B and C fixed.

Collecting the matrix elements of the A_i's in a vector x, we use $q_{ALS}(x)$ to denote the updated variables after performing one full ALS iteration starting from x.

When the CP decomposition problem is ill-conditioned, ALS can be slow to converge,[22] and recently a number of methods have been proposed to accelerate ALS. This includes acceleration by previously mentioned methods such as NGMRES,[12] NCG,[18] and LBFGS.[19] An approach has also been proposed based on the Aitken–Stefensen acceleration technique.[21] These acceleration techniques can substantially improve ALS convergence speed when problems are ill-conditioned or an accurate solution is required.

1.4 | Main approach and contributions of this paper

As explained above, our basic approach is to apply Nesterov acceleration to ALS in a manner that is equivalent to replacing the gradient update in the second step of Nesterov's method, Equation (3), by an ALS step. However, applying this

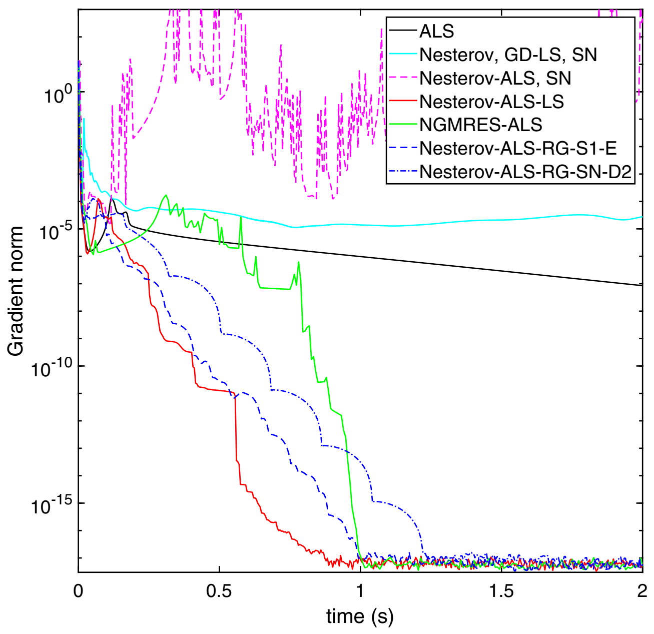


FIGURE 1 Convergence of the gradient norm as a function of runtime for a standard ill-conditioned synthetic canonical polyadic tensor decomposition problem of a tensor of size $50 \times 50 \times 50$ (parameters $s = 50$, $c = 0.9$, $R = 3$, $l_1 = 1$, $l_2 = 1$, see Section 4.3 for the problem description). After an initial transient period, alternating least squares (ALS) (black) converges at a slow linear rate for this problem, and Nesterov's accelerated gradient method (cyan) converges even more slowly. Direct application of Nesterov acceleration to ALS fails to converge (magenta). To remedy this, we propose a family of methods that rely on restarting mechanisms to stabilize Nesterov-ALS (blue curves), which are competitive with previously proposed nonlinear acceleration methods for ALS, such as nonlinear generalized minimal residual method (NGMRES)-ALS (green), but are much easier to implement. We also compare with a Nesterov-ALS method with line search (red). This paper will show numerically that the proposed Nesterov-ALS methods are competitive with or outperform existing nonlinear acceleration methods for ALS, on a set of representative test problems

procedure directly fails for several reasons. First, it is not clear to which extent the β_k momentum weight sequence of (5), which guarantees optimal convergence for gradient acceleration in the convex case, applies at all to our case of ALS acceleration for a nonconvex problem. Second, and more generally, it is well-known that optimization methods for non-convex problems require mechanisms to safeguard against "bad steps," especially when the solution is not close to a local minimum.[27] The main contribution of this paper is to propose and explore restart-based safeguarding mechanisms for Nesterov acceleration applied to ALS, along with momentum weight selection. This leads to a family of acceleration methods for ALS that are competitive with or outperform previously described highly efficient nonlinear acceleration methods for ALS. We also compare with choosing the momentum weight using LSs, which is another way to obtain a safeguarding mechanism and is equivalent to LS methods for ALS that go back to the 1970s.[14–17]

As further motivation for the problem we address and for our approach, Figure 1 illustrates the convergence difficulties that ALS may experience for ill-conditioned CP tensor decomposition problems, and how nonlinear acceleration may allow to remove these convergence difficulties. For the standard ill-conditioned synthetic test problem that is the focus of Figure 1 (see Section 4 for the problem description), ALS converges slowly (black curve). It is known that standard gradient-based methods such as GD, NCG, or LBFGS that do not rely on ALS, perform more poorly than ALS,[22] so it is no surprise that applying Nesterov's accelerated gradient method to the problem (e.g., with the GD step length α_k determined by a standard cubic LS,[12,18,19,22] cyan curve) leads to worse performance than ALS. Nonlinear acceleration of ALS, however, can substantially improve convergence,[12,18,19] and we pursue this using Nesterov acceleration in this paper. However, as expected, applying Nesterov acceleration (with the standard Nesterov momentum weight sequence) directly to ALS for our nonconvex problem, by replacing the gradient step in the Nesterov formula by a step in the ALS direction, does not work and leads to erratic convergence behavior (magenta curve).

As the main contribution of this paper, we consider restart mechanisms to stabilize the convergence behavior of Nesterov-accelerated ALS, and we study how two key parameters, the momentum step and the restart condition, should be set. The blue curves in Figure 1 show two examples of the acceleration that can be provided by two variants of the family of restarted Nesterov-ALS methods we consider. One of these variants (Nesterov-ALS-RG-SN-D2) uses Nesterov's sequence for the momentum weights, and another successful variant simply always uses momentum weight one (Nesterov-ALS-RG-S1-E). The naming scheme for the Nesterov-ALS variants that we consider will be explained in Section 4. Note that, for our nonconvex problem, the cascading convergence pattern of our restarted variant with Nesterov's sequence for the momentum weights, Nesterov-ALS-RG-SN-D2, is very similar to the convergence patterns observed in the work of Su et al.[8] on restart mechanisms for Nesterov acceleration (using Nesterov's sequence) in the convex setting. Extensive numerical tests to be provided in Section 4 show that the best-performing Nesterov-ALS scheme is achieved when using the gradient ratio as momentum weight (as in Nguyen et al.[6]), and restarting when the objective value increases. We also compare with determining the Nesterov momentum weight β_k in each iteration using a cubic LS (red curve). The resulting Nesterov-ALS-LS method (which is similar to classical LS methods for ALS[14–17]) is competitive with or superior to other recently developed nonlinear acceleration techniques for ALS that use LSs, such as NGMRES-ALS (green curve), with the advantage that Nesterov-ALS-LS is much easier to implement. However, the LSs may require multiple evaluations of $f(x)$ and its gradient and can be expensive.

The convergence theory of Nesterov's accelerated gradient method for convex problems does not apply in our case due to the nonconvex setting of the CP problem, and because we accelerate ALS steps instead of gradient steps. In fact, in the context of nonlinear convergence acceleration for ALS, few theoretical results on convergence are available.[12,18,19] We will, however, demonstrate numerically, for representative synthetic and real-world test problems, that our Nesterov-accelerated ALS methods are competitive with or outperform existing acceleration methods for ALS. In particular, some of the Nesterov-ALS methods substantially outperform other acceleration methods for ALS when applied to a large real-world ill-conditioned 71×1,000×900 tensor.

The remainder of this paper is structured as follows. Section 2 presents our general Nesterov-ALS scheme and discusses its instantiations, focusing on the choice of momentum weights and restarting mechanisms. Section 3 discusses and establishes links between the Nesterov-accelerated ALS methods we consider, and existing acceleration methods for ALS. In Section 4, we perform an extensive experimental study of our algorithm by comparing it with a number of acceleration schemes on several benchmark datasets. Section 5 concludes the paper.

2 | NESTEROV-ACCELERATED ALS METHODS

We consider Nesterov-type acceleration of ALS as in Equation (13), but a direct application of a standard Nesterov momentum weight sequence β_k for convex problems as in Equations (4) and (5) does not work. A typical behavior is illustrated by the magenta curve in Figure 1, which suggests that the algorithm gets stuck in a highly suboptimal region. Such erratic behavior arises here, and not in Nesterov's accelerated GD for convex problems, because the ALS update we use for our nonconvex problem is very different from GD. Below we propose a general restart method to safeguard against bad steps, and investigate suitable choices for the momentum weights β_k.

2.1 | Nesterov-ALS with restart

Our general Nesterov-ALS scheme with restart is shown in Algorithm 1. Besides incorporating the momentum term in the update rule (line 12), there are two other important ingredients in our algorithm: adaptive restarting (line 5-7), and adaptive momentum weight β_k (line 9). The precise expressions we use for restarting and computing the momentum weight are explained in the following subsections. In each iteration k of the algorithm we compute a new update according to the update rule (12) with momentum term (line 12). Before computing the update, we check whether a restart is needed (line 5) due to a bad current iterate. When we restart, we discard the current bad iterate (line 6), and compute a simple ALS update instead (ALS always reduces $f(x)$ and is thus well-behaved), by setting β_k equal to zero (line 7) such that

Algorithm 1. Nesterov-ALS with restart

```
1:  initialize $x_1, x_2 \leftarrow q_{ALS}(x_1)$
2:  $i \leftarrow 2$                                  ▷ $i$ is the number of iterates since *restarting*
3:  $\beta_1 = 0$
4:  **for** $k = 2, 3, \dots$ **do**                  ▷ $k$ is the number of iterates since the start of the algorithm
5:      **if** (restart condition met) **and** ($\beta_{k-1} \neq 0$) **then**
6:          $x_k = x_{k-1}$                          ▷ discard the current bad step
7:          $\beta_k = 0, i \leftarrow 1$             ▷ force ALS step this iteration
8:      **else**
9:          compute $\beta_k$ using $i$ and/or previous iterates
10:     **end if**
11:     **exit** loop if termination criterion is met
12:     $x_{k+1} = q_{ALS}(x_k + \beta_k(x_k - x_{k-1})), i \leftarrow i + 1$
13: **end for**
```

(line 12) computes an ALS update. Note that, when a bad iterate is discarded, we do not decrease the iteration index k by 1, but instead set the current iterate x_k equal to the previously accepted iterate x_{k-1}, which then occurs twice in the sequence of iterates. We wrote the algorithm down this way because we can then use k to count work (properly keeping track of the cost to compute the rejected iterate), but the algorithm can of course also be written without duplicating the previous iterate when an iterate is rejected. The index i keeps track of the number of iterates since restarting, which is used for some of our strategies to compute the momentum weight β_k, see Section 2.2. The $\beta_{k-1} \neq 0$ condition is required in (line 5), which checks whether a restart is needed, to make sure that each restart (computing an ALS iteration) is followed by at least one other iteration before another restart can be triggered (because otherwise the algorithm could get stuck in the same iterate).

Various termination criteria may be used. In our experiments, we terminate when the gradient 2-norm reaches a set tolerance:

$$\|\nabla f(x_k)\| / n_X \leq \text{tol},$$

Here n_X is the number of variables in the low-rank tensor approximation.

The momentum weight β_k and the restart condition need to be specified to turn the scheme into concrete algorithms. We discuss the choices used in Sections 2.2 and 2.3.

2.2 | Momentum weight choices for Nesterov-ALS with restart

Naturally, we can ask whether a momentum weight sequence that guarantees optimal convergence for convex problems is applicable in our case. We consider the momentum weight rule defined in Equation (5), but adapted to take restart into account in Algorithm 1:

$$\beta_k \leftarrow (\lambda_{i-1} - 1)/\lambda_i, \tag{23}$$

where λ_i is defined in Equation (4). Restart is taken into account by using i instead of k as the index on the RHS.

Following Nguyen et al.,[6] we also consider using the *gradient ratio* as the momentum weight

$$\beta_k \leftarrow \frac{\|\nabla f(x_k)\|}{\|\nabla f(x_{k-1})\|}. \tag{24}$$

This momentum weight rule can be motivated as follows.[6] When the gradient norm drops significantly, that is, when convergence is fast, the algorithm performs a step closer to the ALS update, because momentum may not really be needed and may in fact be detrimental, potentially leading to overshoots and oscillations. When the gradient norm does not change much, that is, when the algorithm is not making much progress, acceleration may be beneficial and a β_k closer to 1 is obtained by the formula.

Finally, since we observe that Nesterov's sequence Equation (5) produces β_k values that are always of the order of 1 and approach 1 steadily as k increases, we can simply consider a choice of $\beta_k = 1$ for our nonconvex problems, where we rely on the restart mechanism to correct any bad iterates that may result, replacing them by an ALS step. Perhaps surprisingly, the numerical results to be presented below show that this simplest of choices for β_k may work well, if combined with suitable restart conditions.

2.3 | Restart conditions for Nesterov-ALS

One natural restarting strategy is *function restarting* (e.g., O'donoghue and Candes[7,8] for its use in the convex setting), which restarts when the algorithm fails to sufficiently decrease the function value. We consider condition

$$f(x_k) > \eta f(x_{k-d}). \tag{25}$$

Here, we normally use $d = 1$, but $d > 1$ can be used to allow for *delay*. We normally take $\eta = 1$, but we have found that it sometimes pays off to allow for modest increase in $f(x)$ before restarting, and a value of $\eta > 1$ facilitates that. If $d = 1$ and $\eta = 1$, the condition guarantees that the algorithm will make some progress in each iteration, because the ALS step that is carried out after a restart is guaranteed to decrease $f(x)$. However, requiring strict decrease may preclude accelerated iterates (the first accelerated iterate may always be rejected in favor of an ALS update), so either $\eta > 1$ or $d > 1$ allows for a few accelerated iterates to initially increase $f(x)$, after which they may decrease $f(x)$ in further iterations in a much faster way than ALS, potentially resulting in substantial acceleration of ALS. While function restarting (with $d = 1$ and $\eta = 1$) has been observed to significantly improve convergence for convex problems, no theoretical convergence rate has been obtained.[7,8]

Following Su et al.,[8] we also consider the *speed restarting* strategy which restarts when

$$\|x_k - x_{k-1}\| < \|x_{k-1} - x_{k-2}\|. \tag{26}$$

Intuitively, this condition means that the speed along the convergence trajectory, as measured by the change in x, drops. Su et al.[8] showed that speed restarting leads to guaranteed improvement in convergence rate for convex problems.

Another natural strategy is to restart when the gradient norm satisfies

$$\|\nabla f(x_k)\| > \eta \|\nabla f(x_{k-d})\|, \tag{27}$$

where, as above, η can be chosen to be equal to or greater than 1. This *gradient restarting* strategy (with $\eta = 1$) has been used in conjunction with gradient ratio momentum weight by Nguyen et al.,[6] and a similar condition on the residual has been used for Nesterov acceleration of ADMM for convex problems by Goldstein et al..[4]

When we use a value of $\eta > 1$ in the above restart conditions, we have found in our experiments that it pays off to allow for a larger η immediately after the restart, and then decrease η in subsequent steps. In particular, in our numerical tests below, we set $\eta = 1.25$, and decrease η in every subsequent step by 0.02, until η reaches 1.15.

2.4 | Nesterov-ALS with line search

To compare our numerical results with existing acceleration methods for ALS that use a LS in the direction of the ALS update,[12,14–17] we also consider an approach where the momentum weight β_k in the Nesterov extrapolation formula (12) is determined by a LS. In Section 3.2 we explain the equivalence of this approach with existing LS methods to accelerate ALS.[12,14–17] The LS to determine the momentum weight β_k safeguards against bad steps introduced by the $\beta_k(x_k - x_{k-1})$ term, so an additional restart mechanism is not needed. (Note that ALS itself always reduces $f(x)$ and is not prone to introducing bad steps; if the LS does not find a suitable extrapolation point, it simply returns $\beta_k = 0$ and the ALS step is accepted, which can be considered as a restart mechanism built-in into the LS.) For the LS approach, we determine β_k in each iteration as an approximate solution of

$$\beta_k \approx \arg\min_{\beta \geq 0} f(x_k + \beta(x_k - x_{k-1})). \tag{28}$$

We use the standard Moré-Thuente cubic LS that has been used extensively for tensor decomposition methods.[12,18,19,22] This inexact LS finds a value of β_k that satisfies the Wolfe conditions, which impose a sufficient descent condition and a curvature condition. Each iteration of this iterative LS requires the computation of the function value, $f(x)$, and its gradient. As such, the LS can be quite expensive. In our numerical tests, we use the following LS parameters: 10^{-4} for the descent condition, 10^{-2} for the curvature condition, a starting search step length of 1, and a maximum of 20 LS iterations.

3 | RELATION WITH EXISTING ACCELERATION METHODS

In this section we elucidate the relation of the Nesterov acceleration methods for ALS discussed in this paper with other existing convergence acceleration methods for ALS.

3.1 | NGMRES and Anderson acceleration

The NGMRES[12,28] for minimizing $f(y)$ accelerates convergence of a nonlinear iteration $y_k = q(y_{k-1})$ by considering the w-step extrapolation formula

$$\hat{y}_k = \bar{y}_k + \sum_{j=k-w}^{k-1} \beta_j(\bar{y}_k - y_j). \tag{29}$$

where $\bar{y}_k = q(y_{k-1})$, in combination with a LS to stabilize the iteration when the iterate is far from a stationary point,

$$y_k = \bar{y}_k + \alpha_k(\hat{y}_k - \bar{y}_k), \tag{30}$$

where α_k is determined by a LS. The expansion coefficients β_j are computed in each step by solving a small $w \times w$ linear least-squares problem that minimizes a linearization of $\|\nabla f(\hat{y}_k)\|_2$. NGMRES was originally proposed as a convergence accelerator for solving a system of nonlinear algebraic equations $g(y) = 0$, and it was shown to be essentially equivalent to the GMRES method in the case of a linear system $Ay = b$.[10,11] Note that the nonlinear iteration $y_k = q(y_{k-1})$ can also be seen as an inner-iteration nonlinear preconditioner for NGMRES, see also Section 3.3 below.

NGMRES is closely related to the classical Anderson acceleration method for solving $g(y) = 0$,[13] which uses the expansion formula

$$\hat{y}_k = \bar{y}_k + \sum_{j=k-w}^{k-1} \beta_j(\bar{y}_k - \bar{y}_j), \tag{31}$$

where the β_j are also determined to approximately minimize $\|g(\hat{y}_k)\|_2$. Anderson acceleration can be viewed as a multi-secant method[29] and also reduces to GMRES for the case of linear systems.[30]

It can be seen immediately that the form of the Nesterov extrapolation formula (15) is equivalent to the Anderson extrapolation formula (31) with one step ($w = 1$), and is also closely related to NGMRES with one step (Equation (29)). The difference, of course, is that the extrapolation weight β_k in Nesterov's formula (15) is usually determined differently than for Anderson acceleration and NGMRES.

3.2 | Line search methods for ALS

As early as 1970, Harshman[14] described enhancing ALS convergence for CP tensor decomposition by an over-relaxation in the direction of the ALS update:

$$y_k = \bar{y}_k + \beta(\bar{y}_k - y_{k-1}), \tag{32}$$

where $\bar{y}_k = q_{ALS}(y_{k-1})$ and a value between 1.2 and 1.3 was recommended for the over-relaxation parameter β. This extrapolation step is of the same form as NGMRES with window size one, see Equation (29), and is similar to (but not the same as) Nesterov extrapolation formula (15) and Anderson acceleration with $w = 1$. In later work, β in Equation (32) was determined in each step using a LS, obtaining a value of β that approximately minimizes $f(y_k)$. In particular, the so-called *enhanced LS* approach[15] computes the optimal β in Equation (32) in an accurate manner, which is possible because, for finding the optimal CP decomposition in the 2-norm, the objective is a polynomial function of β. Enhanced LS was later extended to broader tensor optimization problems and to exact plane search.[17] Chen et al.[16] consider one-step extrapolation according to the update formula

$$y_k = \bar{y}_k + \beta_k(\bar{y}_k - \bar{y}_{k-1}), \tag{33}$$

where again $\bar{y}_k = q_{ALS}(y_{k-1})$, which is of the same form as the Nesterov-type update we consider in this paper, Equation (15) (and, thus, the same as Anderson acceleration with one step), but Chen et al.[16] determine β_k using a LS.

Chen et al.[16] also considered two variants of two-step extrapolation formula

$$y_k = \bar{y}_k + \beta_k(\gamma_0\bar{y}_k + \gamma_1\bar{y}_{k-1} + \gamma_2\bar{y}_{k-2}), \tag{34}$$

where the coefficients γ_j in the extrapolation are predetermined based on a so-called geometric or algebraic ansatz. This approach is similar to NGMRES or Anderson acceleration with window size two, except that the latter methods aim to determine optimal extrapolation coefficients by solving, in each iteration, a $w \times w$ linear least-squares problem for a window size of w. NGMRES and Anderson, thus, adapt the extrapolation coefficients each iteration to minimize the residual of the gradient, for general window size, whereas the methods of Chen et al.[16] are limited to two-step extrapolation and use extrapolation coefficients that are fixed over the iterations and are determined in an ad-hoc way. Numerically it has been shown[12] that NGMRES with window size greater than 1 (i.e., multistep extrapolation) can be much faster than (enhanced) LS of the form (32) (i.e., one-step extrapolation); it was found that NGMRES window size $w = 20$ is a good choice for efficient acceleration of ALS for CP tensor decomposition.[12] Finally, we note that acceleration of ALS by NGMRES, NCG, and LBFGS also employs LSs as a globalization mechanism to guard against erratic iterates.[12,18,19]

While the main focus of this paper is on investigating momentum weights β_k and restart mechanisms for Nesterov update formula (15) for ALS, we also compare with a version of update formula (15) where β_k is determined by a LS, similar to existing methods for accelerating ALS convergence that use LSs.[12,14–17]

3.3 | Nonlinear preconditioning: left preconditioning and right preconditioning

Applying Nesterov's acceleration approach to ALS as explained in Section 1.2 can also be interpreted in the context of nonlinear preconditioning[13] for optimization methods. In recent work[19] a general framework for nonlinear preconditioning of optimization methods was formulated and applied to nonlinear preconditioning of NCG and LBFGS. This framework extends the concepts of linear preconditioning for linear systems

$$Ay = b$$

to genuinely nonlinear preconditioners for nonlinear optimization. In the context of linear systems, the concept of *left preconditioning* multiplies the linear system with a nonsingular preconditioning matrix P, aiming to improve the convergence of iterative methods (such as GMRES) applied to the preconditioned system

$$PAy = Pb.$$

Left preconditioning can be seen as taking linear combinations of the equations to be solved. In the *right preconditioning* approach, on the other hand, a linear change of the variables $y = Pz$ is considered by means of a nonsingular preconditioning matrix P, and the iterative method is then applied to the preconditioned system

$$APz = b.$$

We now explain how these ideas can be extended to minimizing nonlinear functions $f(y)$ using genuinely nonlinear preconditioners, where the preconditioner is not given by a linear coordinate transformation that is encoded by a matrix multiplication, but by a genuinely nonlinear transformation.

3.3.1 | Nonlinear left preconditioning

In the linear case, left preconditioning of, for example, the GMRES method, can be understood as a combination of two methods: an inner preconditioning iteration

$$y_{k+1} = y_k - P(Ay_k - b), \tag{35}$$

where P is the left preconditioning matrix, is combined with the outer GMRES iteration (where each iteration of the combined method typically uses one inner update and one outer update).[10–12] One can say that inner iteration (35) preconditions the GMRES outer iteration, speeding up the convergence of GMRES, or, in an alternative view, one can say that GMRES is an outer iteration that accelerates the convergence of inner iteration (35). Possible choices for preconditioning matrix P include, for example, the lower triangular part of A (Gauss-Seidel), or the constant diagonal preconditioner αI (Richardson iteration), with a suitably chosen α.

It is instructive to consider the case of solving a linear system $Ay = b$ where A is Symmetric Positive Definite (SPD). In this case, solving $Ay = b$ is equivalent to minimizing $f(y) = \frac{1}{2}y^T Ay - b^T y$. The gradient $g(y)$ of $f(y)$ is given by

$$g(y) = \nabla f(y) = Ay - b. \tag{36}$$

In this case, it can be seen that inner iteration preconditioner (35) using Richardson iteration is equivalent to steepest descent:

$$y_{k+1} = y_k - \alpha \nabla f(y_k). \tag{37}$$

More generally, when using a preconditioning matrix P in the case of an SPD system, the expression $P(Ay_k - b) = P\nabla f(y_k) = Pg(y_k)$ in (35) can be called the preconditioned gradient. Note that the preconditioned gradient can be obtained from the preconditioning iteration (35) by

$$Pg(y_k) = -(y_{k+1} - y_k). \tag{38}$$

In words, the preconditioned gradient vector is given by the update vector of the linear preconditioning iteration.

Conceptually, this can be generalized to nonlinear preconditioning as follows. Standard nonlinear optimization methods such as NCG or LBFGS use gradient directions to minimize a nonlinear function $f(y)$. However, if a simple nonlinear iterative method for minimizing $f(y)$ is available that converges faster than GD, and, in a sense, provides better search directions than the gradient, we can use this iterative method as an inner preconditioning iteration for NCG or LBFGS, and make use of the more suitable search directions provided by this method. For example, in the case of ALS for canonical tensor decomposition, we can use the nonlinear iteration

$$y_{k+1} = q_{ALS}(y_k), \tag{39}$$

as the inner iteration preconditioner of NCG or LBFGS. In analogy with the linear SPD case, we can interpret the direction provided by ALS as a nonlinearly preconditioned gradient direction, $\mathcal{P}(g; y)$, given by

$$\mathcal{P}(g; y_k) = -(y_{k+1} - y_k) = -(q_{\mathrm{ALS}}(y_k) - y_k), \tag{40}$$

in analogy with (38) and (35). To bring this idea into practice, we simply replace the gradient direction in the update formulas for NCG or LBFGS by the nonlinearly preconditioned gradient $\mathcal{P}(g; y)$, that is, by the update vector that the nonlinear preconditioner (used as an inner iteration) provides. In a sense, standard NCG and LBFGS can be understood as using steepest descent (gradient directions) as the inner iteration. In left-nonlinearly preconditioned NCG and LBFGS, we replace the steepest-descent inner iteration by the nonlinear preconditioner iteration (e.g., ALS). Another way to understand this is that standard NCG and LBFGS are iterative methods that work on solving $g(y) = 0$, that is, their update formulas drive the gradient to zero (equivalent to solving $Ay - b = 0$ in the linear case). Nonlinearly preconditioned NCG or LBFGS work on solving the nonlinearly preconditioned equation $\mathcal{P}(g; y) = 0$ (which is really the fixed-point equation $y - q(y) = 0$), that is, their update formulas drive the nonlinearly preconditioned gradient to zero (equivalent to solving $PAy - Pb = 0$ in the linear case). Further details and numerical results can be found in a recent paper establishing this general formalism of nonlinear left preconditioning for optimization.[19] In this formalism, we can say that the nonlinear preconditioner is used as an inner iteration for the outer-iteration NCG and LBFGS methods, or, alternatively, we can say that NCG and LBFGS are used as nonlinear convergence accelerators for the inner iteration (e.g., ALS).

This nonlinear left preconditioning can in principle be applied to any nonlinear optimization method (not just NCG or LBFGS), and it can clearly also be applied to Nesterov's accelerated GD method as in Equation (9), when, for example, using ALS as the nonlinear preconditioner for CP tensor decomposition. In fact, in the case of Nesterov's method, the nonlinear left preconditioning procedure is very simple: it basically amounts to replacing the gradient update $-\alpha_k \nabla f(y_k)$ in Equation (3) by $q_{\text{ALS}}(y_k) - y_k$, the step provided by ALS, and directly results in the nonlinearly preconditioned formulas of Equation (10) or Equation (11), so the Nesterov acceleration formula (13) for ALS can also be interpreted as using ALS as a nonlinear preconditioner for Nesterov's accelerated gradient formula (9).

3.3.2 | Nonlinear right preconditioning—transformation preconditioning for optimization

Similarly, nonlinear preconditioning techniques can be derived for optimization that rely on a nonlinear change of variables,[19] inspired by right preconditioning in the linear case. In the so-called transformation preconditioning approach[19] (which is a form of right preconditioning), nonlinearly preconditioned versions of NCG and LBFGS are derived as follows. One first considers minimization of $f(y)$ using a linear change of variables

$$y = Cz,$$

and then applies standard NCG and LBFGS (in the z variable) to minimize

$$\hat{f}(z) = f(Cz).$$

Transforming the resulting update formulas back to the original y variables using

$$\nabla_z \hat{f}(z) = \nabla_z f(Cz) = C^T \nabla_y f(y), \tag{41}$$

introduces the linear preconditioning matrix

$$P = CC^T, \tag{42}$$

in expressions like the scalar product of gradients:

$$\nabla_z \hat{f}(z)^T \nabla_z \hat{f}(z) = \nabla_y f(y)^T CC^T \nabla_y f(y) = \nabla_y f(y)^T P \nabla_y f(y). \tag{43}$$

The resulting linearly preconditioned NCG and LBFGS methods are well-known.[31,32] However, in recent work this approach was extended to nonlinear preconditioning by replacing the linearly preconditioned gradient $P\nabla_y f(y_k)$ in expressions like (43) by the nonlinearly preconditioned gradient direction $\mathcal{P}(g; y_k)$ given by the update vector that is provided by the nonlinear preconditioning iteration (39): $\mathcal{P}(g; y_k) = -(y_{k+1} - y_k) = -(q_{ALS}(y_k) - y_k)$, as in (38). This gives different nonlinearly preconditioned iterations for NCG and LBFGS than the nonlinear left preconditioning approach discussed above. An attractive feature of this nonlinear transformation preconditioning is that it reduces to well-known linear preconditioning techniques for NCG and LBFGS in the case of the linear change of variables $y = Cz$.[31,32] In practice, both left and transformation nonlinear preconditioning may lead to dramatic improvements in convergence for NCG and LBFGS.[19] In this paper, we use the transformation preconditioning versions of NCG-ALS and LBFGS-ALS in the numerical results. In the case of Nesterov's method, which is a simple iteration that does not involve scalar products of gradients, the two procedures of nonlinear left preconditioning and transformation preconditioning give the same result, that is, both approaches lead to the nonlinearly preconditioned formulas of Equation (10) or (11).

More broadly, nonlinear preconditioning has a long but not widely known history in computational science, and is currently an active area of ongoing research,[13] including in the optimization context.[19] In our numerical results, we compare Nesterov acceleration of ALS—that is, Nesterov's method with ALS as nonlinear preconditioner—with NCG and LBFGS acceleration of ALS—that is, NCG and LBFGS with ALS as nonlinear preconditioner. Numerical results will show that Nesterov-ALS is often competitive with the more sophisticated LBFGS-ALS. In addition, Nesterov acceleration is attractive because it is much easier to implement than LBFGS acceleration.

Abbreviation	Explanation
RF	function restarting as in Equation (25)
RG	gradient restarting as in Equation (27)
RX	speed restarting as in Equation (26)
SN	Nesterov step as in Equation (5)
SG	gradient ratio step as in Equation (24)
S1	constant step 1
Dd	delay $d > 1$ in restart condition
E	$\eta \neq 1$ in restart condition

TABLE 1 Abbreviations used in naming convention for restarted Nesterov-ALS variants

4 | NUMERICAL TESTS

We evaluated our Nesterov-ALS algorithm with various choices for momentum weights and restart mechanisms on a set of synthetic CP test problems that have been carefully designed and used in many papers, and three real-world datasets of different sizes and originating from different applications. All numerical tests were performed in Matlab, using the Tensor Toolbox[33] and the Poblano Toolbox for optimization.[34]

4.1 | Naming convention for Nesterov-ALS schemes

We use the following naming conventions for the restarting strategies and momentum weight strategies defined in Section 2. For the restarted Nesterov-ALS schemes, we append Nesterov-ALS with the abbreviations in Table 1 to denote the restarting scheme used, and the choice for the momentum weight β_k.

For example, Nesterov-ALS-RF-SG means using restarting based on function value (RF) and momentum step based on gradient ratio (SG). For most tests we do not use delay (i.e., $d = 1$ in Equation (25) or (27)), and η is usually set to 1 in Equation (25) or (27). Appending Dn or E to the name indicates that a delay $d = n > 1$ is used, and that $\eta \neq 1$ is used, respectively. The LS Nesterov-ALS scheme we compare with is denoted Nesterov-ALS-LS.

4.2 | Baseline algorithms

We compare our proposed Nesterov-ALS schemes with the recently proposed nonlinear acceleration methods for ALS using GMRES,[12] NCG,[18] and LBFGS.[19] These methods will be denoted in the result figures as GMRES-ALS, NCG-ALS, and LBFGS-ALS, respectively.

4.3 | Synthetic test problems and results

We use the synthetic tensor test problems considered by[22] and used in many papers as a standard benchmark test problem for CP decomposition.[12,18,19] As described in more detail elsewhere,[12] we generate six classes of random three-way tensors with highly collinear columns in the factor matrices. We add two types of random noise to the tensors generated from the factor matrices (homoscedastic and heteroscedastic noise[12,22]), and then compute low-rank CP decompositions of the resulting tensors.

Due to the high collinearity, the problems are ill-conditioned and ALS is slow to converge.[22] All tensors have equal size $s = I_1 = I_2 = I_3$ in the three tensor dimensions. The six classes differ in their choice of tensor sizes ($s = 20, 50, 100$), decomposition rank ($R = 3, 5$), and noise parameters l_1 and l_2 ($l_1 = 0, 1$ and $l_2 = 0, 1$), in combinations that are specified in Table A1.

To compare how various methods perform on these synthetic problems, we generate 10 random tensor instances with an associated random initial guess for each of the six problem classes, and run each method on each of the 60 test

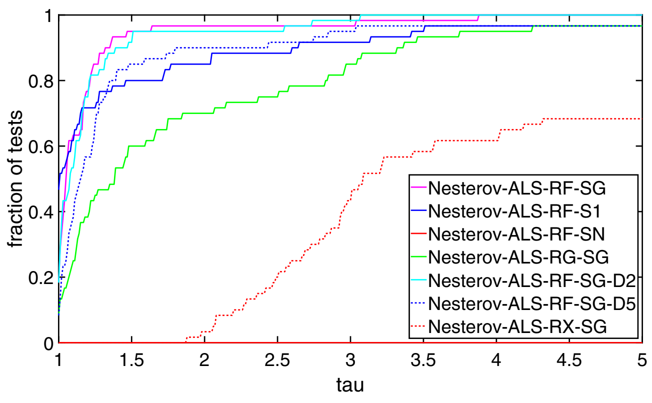


FIGURE 2 Synthetic test problems. τ-plot comparing the optimal restarted algorithm, Nesterov-ALS-RF-SG, with other variants of restarted Nesterov-ALS. ALS, alternating least squares; RF, function value; SG, momentum step based on gradient ratio

problems, with a convergence tolerance tol $= 10^{-9}$. We then present so-called τ-plot performance profiles,[12] as explained below, to compare the relative performance of the methods over the test problem set.

Optimal restarted Nesterov-ALS method. Our extensive experiments on both the synthetic and real-world datasets (as indicated in tests below and in Appendix B1) suggest that the optimal restarted Nesterov-ALS method is the one using function restarting (RF) and gradient ratio momentum steps (SG), that is, Nesterov-ALS-RF-SG. As a comparison, the study of Nguyen et al.[6] suggests that gradient restarting and gradient ratio momentum weights work well for accelerating GD in the context of nonlinear system solving.

Figure 2 shows the performance of our optimal Nesterov-ALS-RF-SG method on the synthetic test problems, with an ablation analysis that compares it with those variants obtained by varying one hyperparameter of Nesterov-ALS-RF-SG at a time. In this τ-plot, we display, for each method, the fraction of the 60 problem runs for which the method execution time is within a factor τ of the fastest method for that problem. For example, for $\tau = 1$, the plot shows the fraction of the 60 problems for which each method is the fastest. For $\tau = 2$, the plot shows, for each method, the fraction of the 60 problems for which the method reaches the convergence tolerance in a time within a factor of two of the fastest method for that problem, etc. As such, the area between curves is a measure for the relative performance of the methods, with the curves at the top performing the best.

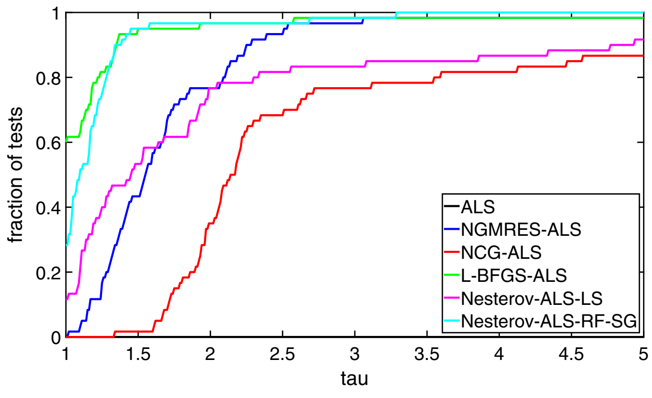
We can see that several variants have comparable performance to Nesterov-ALS-RF-SG, so the optimal choice of restart mechanism and momentum weight is not very sensitive. For these tests, changing the delay parameter has least effect on the performance. Interestingly, this is then followed by changing the momentum weight to be a constant of 1. It thus appears that, for our nonconvex problem, the simple choice of $\beta_k = 1$ combined with a suitable restart mechanism leads to a highly performant acceleration method. This is followed by changing function restarting to gradient restarting and speed restarting, respectively. More detailed numerical results comparing Nesterov-ALS-RF-SG with a broader variation of restarted Nesterov-ALS methods are shown in Appendix B1, further confirming that Nesterov-ALS-RF-SG generally performs the best among the family of restarted Nesterov-ALS methods we have considered, for the synthetic test problems.

Fig. 3 compares Nesterov-ALS-RF-SG with the line search version Nesterov-ALS-LS (equivalent or similar to existing LS methods to accelerate ALS[14–17]), and with several other existing accelerated ALS methods, namely, GMRES-ALS, NCG-ALS, and LBFGS-ALS. For high accuracy (top panel), Nesterov-ALS-RF-SG performs similarly to the best performing of the existing methods we compare with, LBFGS-ALS.[19] It performs substantially better than Nesterov-ALS-LS (it avoids the expensive LSs). Nevertheless, Nesterov-ALS-LS is competitive with the existing NGMRES-ALS,[12] and superior to NCG-ALS.[18]

For low accuracy (bottom panel), Nesterov-ALS-RF-SG and Nesterov-ALS-LS still perform very well. ALS is now more competitive, which is not unexpected, since ALS is often efficient at reducing the initial error quickly, but then may converge slowly later on for difficult problems.

4.4 | The Enron dataset and results

The Enron dataset is a subset of the corporate e-mail communications that were released to the public as part of the 2002 Federal Energy Regulatory Commission (FERC) investigations following the Enron bankruptcy. After various steps



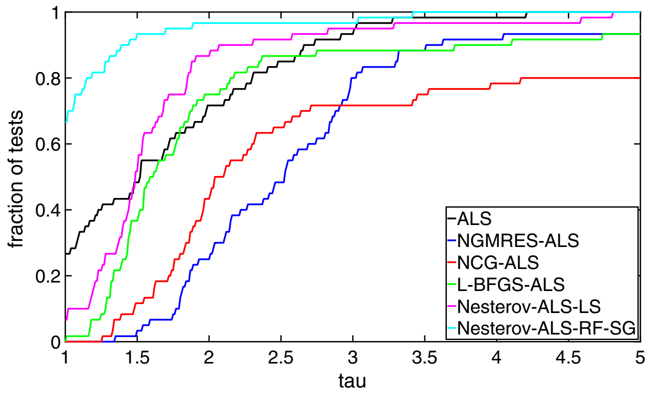


FIGURE 3 Comparisons of the following algorithms on synthetic test problems: the optimal restarted algorithm (Nesterov-ALS-RF-SG), the line search restarted algorithm (Nesterov-ALS-LS), and existing accelerated ALS methods. (top) τ-plot for high accuracy tolerance value tol $= 10^{-9}$. (bottom) τ-plot for low accuracy tolerance value tol $= 10^{-5}$. ALS, alternating least squares; LS, line search; RF, function value; SG, momentum step based on gradient ratio

of preprocessing,[35] a sender × receiver × month tensor of size 105×105×28 was obtained. We perform rank-10 CP decompositions for Enron. Figure 4 shows gradient norm convergence for one typical test run, and a τ-plot for 60 runs with different random initial guesses and convergence tolerances tol $= 10^{-7}\,\|\nabla f(x_0)\|$ (high accuracy, middle panel) and tol $= 10^{-5}\,\|\nabla f(x_0)\|$ (lower accuracy, bottom panel). For this well-conditioned problem, ALS converges fast and does not need acceleration. In fact, the acceleration overhead makes ALS faster than any of the accelerated methods. This is consistent with previously reported results[12,18,19,22] for well-conditioned problems.

4.5 | The claus dataset and results

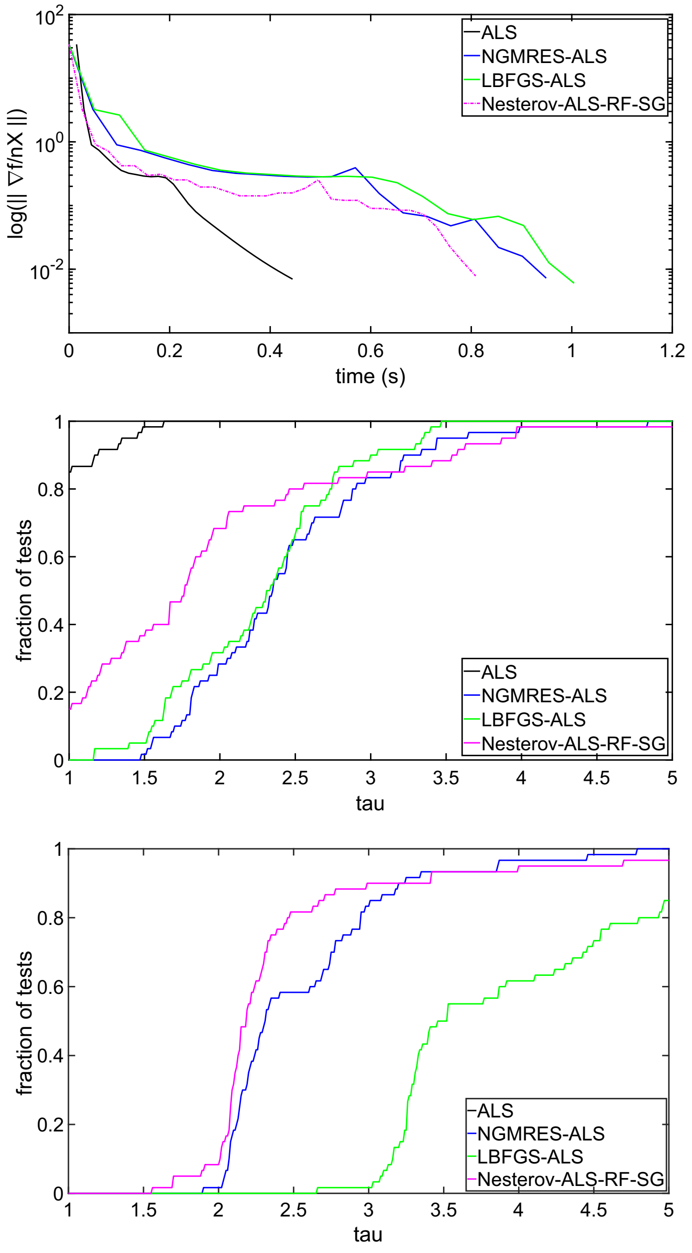
The claus dataset is a 5×201×61 tensor consisting of fluorescence measurements of five samples containing three amino acids, taken for 201 emission wavelengths and 61 excitation wavelengths. Each amino acid corresponds to a rank-1 component.[36] We perform a rank-3 CP decomposition for claus. Figure 5 shows gradient norm convergence for one test run, and a τ-plot for 60 runs with different random initial guesses and convergence tolerances tol $= 10^{-7}\,\|\nabla f(x_0)\|$ (high accuracy, middle panel) and tol $= 10^{-4}\,\|\nabla f(x_0)\|$ (low accuracy, bottom panel). For this medium-conditioned problem, substantial acceleration of ALS can be obtained if high accuracy is desired, and Nesterov-ALS-RF-SG performs as well as the best methods we compare with, but it is much easier to implement. For low accuracy, ALS is more competitive, but Nesterov-ALS-RF-SG still outperforms it.

4.6 | The Gas3 dataset and results

Gas3 is relatively large and has multiway structure. It is a 71×1,000×900 tensor consisting of readings from 71 chemical sensors used for tracking hazardous gases over 1,000 time steps.[37] There were three gases, and 300 experiments were performed for each gas, varying fan speed and room temperature. We perform a rank-5 CP decomposition for Gas3.

Figure 6 shows gradient norm convergence for one typical test run, and a τ-plot for 20 runs with different random initial guesses and convergence tolerance tol $= 10^{-7}\,\|\nabla f(x_0)\|$ (high accuracy, middle panel) and tol $= 10^{-4}\,\|\nabla f(x_0)$ (low accuracy, bottom panel).

FIGURE 4 Comparison of algorithms on the Enron data. (top) Gradient convergence plot. (middle) τ-plot for high accuracy tolerance value tol $= 10^{-7}\,\|\nabla f(x_0)\|$. (bottom) τ-plot for low accuracy tolerance value tol $= 10^{-5}\,\|\nabla f(x_0)\|$.



For this highly ill-conditioned problem, ALS converges slowly, and NGMRES-ALS, NCG-ALS and LBFGS-ALS behave erratically. For high accuracy, our newly proposed Nesterov-ALS-RF-SG very substantially outperforms all other methods (not only for the convergence profile shown in the top panel, but for the large majority of the 20 tests with random initial guesses). Nesterov-ALS-RF-SG performs much more robustly for this highly ill-conditioned problem than any of the other accelerated methods, and reaches high accuracy much faster than any other method. We were initially surprised that Nesterov-ALS-RF-SG performs so much better than the other accelerated ALS methods we compare with, and have so far not found a clear indication why this is the case. One possible explanation is that the LS employed in NGMRES-ALS, NCG-ALS and LBFGS-ALS may suffer robustness issues due to the ill-conditioning of the problem. For low accuracy, ALS is clearly the fastest: it is efficient at reducing the initial error quickly, but converges slowly later on for this difficult problem.

The other tests in this paper indicate that our proposed Nesterov-ALS-RF-SG acceleration method is competitive with leading existing acceleration methods for ALS, and, additionally, this result gives some initial indication that Nesterov-ALS-RF-SG is surprisingly robust for highly ill-conditioned problems. This will need to be investigated further when these methods, which will be made available as part of the Poblano toolbox for optimization, will be applied by us and others to other demanding CP decomposition problems.

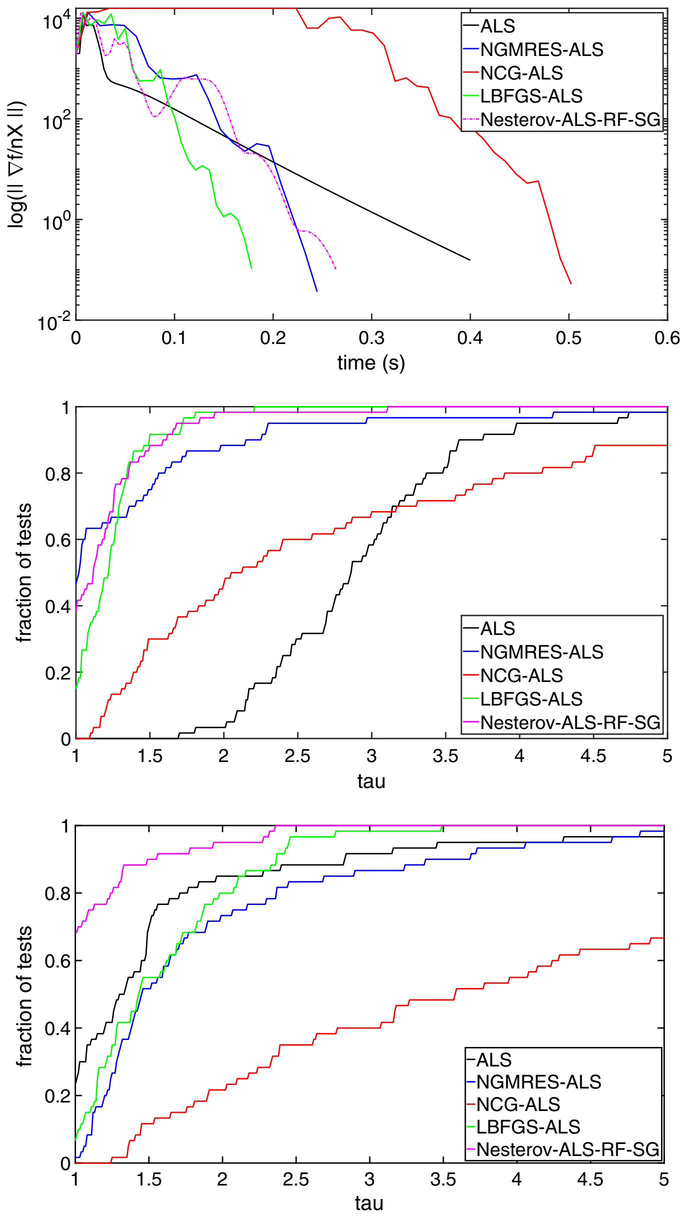


FIGURE 5 Comparison of algorithms on the Claus data. (top) Gradient convergence plot. (middle) τ-plot for high accuracy tolerance value tol $= 10^{-7}\ \|\nabla f(x_0)\|$. (bottom) τ-plot for low accuracy tolerance value tol $= 10^{-4}\ \|\nabla f(x_0)\|$.

4.7 | Discussion on per-iteration computational cost

It is important to compare the computational cost per iteration of our Nesterov-accelerated ALS methods with plain ALS iterations. For simplicity, we discuss the case of dense order-3 tensors of size $I \times J \times K$. For $R \ll I, J, K$, the dominant cost in each ALS step is the formation of the three MTTKRPs on the left-hand side in Equations (20)–(22), which takes approximately $O(IJKR)$ arithmetic operations. At the end of an ALS iteration, the function value can be computed at virtually no cost by noting that

$$f(A,B,C) = \frac{1}{2}\|T - \tilde{T}\|_F^2 = \frac{1}{2}\|T_3 - C(A \odot B)^T\|_F^2 = \frac{1}{2}\|T_F^2 - T_3(A \odot B) \cdot C + \frac{1}{2}\|C(A \odot B)^T\|_F^2,$$

where $\|T\|_F^2$ is precomputed, the MTTKRP $T_3(A \odot B)$ is reused, and $\cdot$ is the dot product between two matrices (i.e., the sum of the entries of the element-wise multiplication of the matrices). The third term equals $\frac{1}{2}\|\tilde{T}\|_F^2$, and, due to the structure of the rank-1 terms, it can be computed in $O(2R^2(I + J + K))$ operations, which is negligible compared to the cost of the MTTKRPs in ALS. In fact, this efficient computation is the default algorithm for computing the function value in the ALS implementation of the Tensor Toolbox.[33]

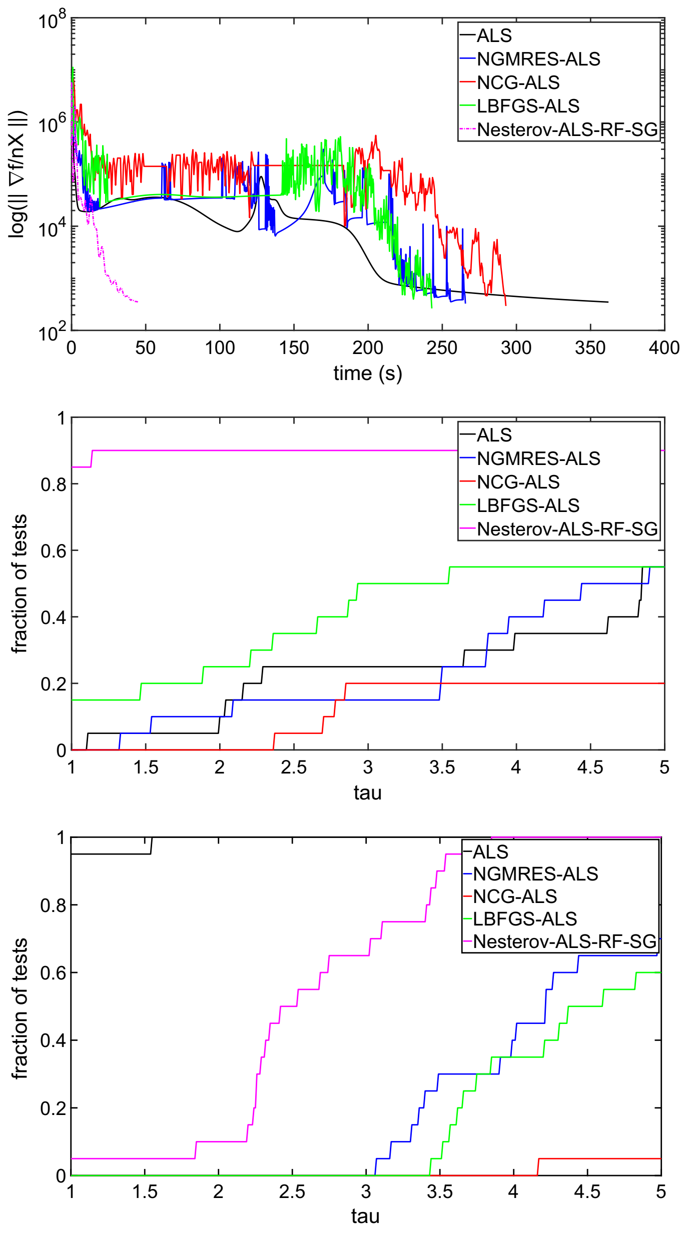


FIGURE 6 Comparison of algorithms on the Gas3 data. (top) Gradient convergence plot. (middle) τ-plot for high accuracy tolerance value tol $= 10^{-7}\ \|\nabla f(x_0)\|$. (bottom) τ-plot for low accuracy tolerance value tol $= 10^{-4}\ \|\nabla f(x_0)\|$

Computing the three MTTKRPs is also the dominant cost when calculating the gradient of f, according to Equations (17)–(19), and following a gradient computation the function value comes virtually for free, so the dominant cost of a function + gradient evaluation is also of $O(IJKR)$, the same cost as an ALS iteration.

In our Nesterov-accelerated ALS methods, we use the gradient and/or function value to compute the restart condition and/or the step length, so we perform one function + gradient evaluation per iteration in the implementation of all our methods, in addition to the cost of one ALS step per iteration. For this reason, each of our Nesterov-accelerated ALS iterations is about twice as expensive as a regular ALS iteration. Still, the numerical results in the previous sections show clearly that the Nesterov acceleration tends to reduce the number of iterations required for (accurate) convergence by much more than a factor of 2, making the accelerated methods clear winners over ALS (despite the doubled cost per iteration), for difficult problems or when accurate solutions are required.

Note that it is possible to avoid the doubling of the per-iteration cost by considering acceleration variants that only rely on function values for the restart mechanism, and determine the step length without using gradient information. In that respect, our accelerated method with function restart and constant momentum weight one is attractive. Note also that the restarted acceleration methods in Ang and Gillis[9] for Nonnegative Matrix Factorization use more elaborate strategies for determining step lengths that also do not require gradient information.

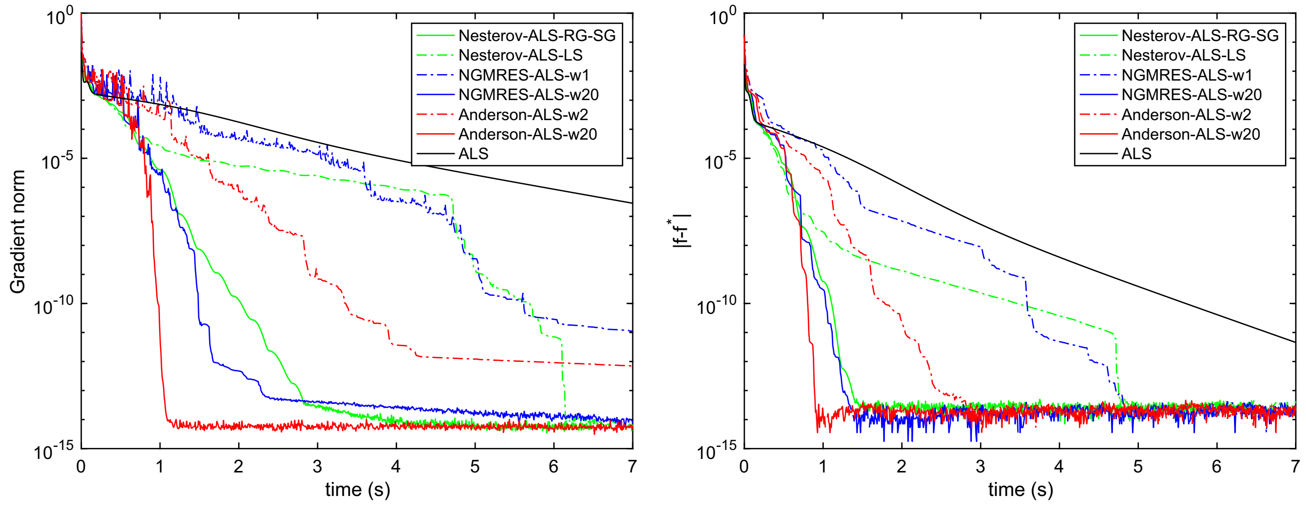


FIGURE 7 Convergence comparison with existing line search (LS) methods for synthetic test problem 2 from Table A1. (left) Gradient norm as a function of time. (right) Convergence of the objective, f, to its minimum value, f^*, as a function of time. One of our proposed Nesterov-accelerated methods with restart, Nesterov-ALS-RG-SG (green), is compared with Harshman's extrapolation with LS[14] (equivalent to NGMRES-ALS-w1, blue-dashed), Chen et al.'s one-step extrapolation with LS[16] (equivalent to Nesterov-ALS-LS, green-dashed), and Chen et al.'s two-step extrapolation with LS[16] (similar to our Anderson acceleration with window size 2, Anderson-ALS-w2, red-dashed). Our proposed restarted Nesterov method (green) converges substantially faster than the existing LS methods (dashed). Also, for Harshman's extrapolation with LS (blue-dashed), the result for NGMRES-ALS-w20 (blue) shows that a window size for NGMRES greater than 2 leads to much faster convergence. Similarly, for Chen et al.'s two-step extrapolation with LS (similar to Anderson-ALS-w2, red-dashed, the result for Anderson-ALS-w20 (red) shows that a window size for Anderson acceleration greater than 2 leads to much faster convergence. These convergence plots are for a specific example, but they are typical for the relative performance of the methods. . ALS, alternating least squares; LS, line search; RF, function value; SG, momentum step based on gradient ratio

Finally, it is interesting to observe that, due to the LSs, the per-iteration cost of our NGMRES-ALS, Anderson-ALS and LBFGS-ALS methods is typically four to five times the cost of 1 ALS iteration, and the per-iteration cost of NCG-ALS is typically six to seven times the cost of 1 ALS iteration. Still, some of these methods achieve very large reductions in the number of iterations required for convergence, and are often among the most efficient, in particular, LBFGS-ALS and NGMRES-ALS/Anderson-ALS.

4.8 | Further comparison with LS methods

It is interesting to further compare numerical results with the existing LS methods that were discussed in Section 3.2. Figure 7 compares convergence for synthetic test problem 2 from Table A1, for several methods. The left panel shows the gradient norm as a function of time, and the right panel the convergence of the objective, f, to its minimum value, f^*, as a function of time. We compute f^* as the smallest value of f obtained by any algorithm after performing sufficiently many iterations to make ∇f converge to machine accuracy.

One of our proposed Nesterov-accelerated methods with restart, Nesterov-ALS-RG-SG (green), is compared with several existing LS methods.

We first compare with NGMRES-ALS with window size 1, NGMRES-ALS-w1 (blue-dashed), which is equivalent to Harshman's extrapolation with LS.[14,15] We use the Moré-Thuente cubic LS rather than an exact LS,[15] but it is known that the exact LS is expensive since it requires $2N - 1$ function evaluations,[12,22] whereas the cubic LS typically only requires about 2 or 3 function evaluations.[22]

We also compare with Nesterov extrapolation with LS, Nesterov-ALS-LS (green-dashed), which is equivalent with Chen et al.'s 1-step extrapolation with LS,[16] except that we don't use an exact LS. Finally, we compare with Anderson acceleration with window size 2, Anderson-ALS-w2 (red-dashed), which uses the same form of extrapolation as Chen et al.'s two-step extrapolation,[16] but determines optimal expansion coefficients by solving a least-squares problem in each step, whereas Chen et al.'s method makes an ad-hoc choice for the expansion coefficients that is the same in all iterations. As such, we can use Anderson-ALS-w2 as an (optimistic) proxy for estimating the performance of Chen et al.'s two-step extrapolation.[16]

The results in Figure 7 show that our proposed Nesterov-accelerated method with restart, Nesterov-ALS-RG-SG (green), converges much faster than the three existing LS methods (dashed). Also, for Harshman's extrapolation with LS (blue-dashed, NGMRES-ALS-w1), the result for NGMRES-ALS-w20 (blue) shows that a window size for NGMRES greater than 2 leads to much faster convergence. Similarly, for Chen et al.'s two-step extrapolation with LS (similar to Anderson-ALS-w2, red-dashed), the result for Anderson-ALS-w20 (red) shows that a window size for Anderson acceleration greater than 2 leads to much faster convergence. These convergence plots are for a specific example, but they are typical for the relative performance of the methods.

Note, finally, that the problem in Figure 7 is an ill-conditioned problem, so $|f - f^*|$ reaches machine accuracy before the gradient does.

5 | CONCLUSION

We have proposed Nesterov-ALS methods with effective choices for momentum weights and restart mechanisms that are simple and easy to implement as compared to several existing nonlinearly accelerated ALS methods, such as GMRES-ALS, NCG-ALS, and LBFGS-ALS.[12,18,19] The optimal variant, using function restarting and gradient ratio momentum weight, is competitive with or superior to stand-alone ALS and GMRES-ALS, NCG-ALS, LBFGS-ALS, and LS ALS.[14–17]

Simple nonlinear iterative optimization methods like ALS and coordinate descent (CD) are widely used in a variety of application domains. There is clear potential for extending our approach to accelerating such simple optimization methods for other nonconvex problems. A specific example is Tucker tensor decomposition.[5] NCG, NGMRES, and LBFGS acceleration have been applied to Tucker decomposition,[19,38] using a manifold approach to maintain the Tucker orthogonality constraints, and this approach can directly be extended to the Nesterov acceleration methods discussed in this paper.

More generally, we have formulated a Nesterov-type acceleration approach that can effectively accelerate optimization algorithms different from GD (such as ALS) and for nonconvex problems (such as CP tensor decomposition), using simple choices for momentum weights (as simple as setting them equal to 1) and suitable restart mechanisms. The proposed methods are competitive in terms of performance with any existing acceleration methods for ALS and are very simple to implement, and initial tests on a challenging real-world problem indicate desirable robustness properties.

Matlab code implementing the proposed Nesterov acceleration methods is freely available at https://github.com/hansdesterck/nonlinear-preconditioning-for-optimization. The code includes implementations of all Nesterov-ALS variants, as well as NCG-ALS, NMGRES-ALS, LBFGS-ALS, and Anderson-ALS. The implementations are generic in that any suitable nonlinear preconditioner can be provided, not just ALS, and the nonlinearly preconditioned methods can be applied to any suitable optimization problem, not just canonical tensor decomposition.

ACKNOWLEDGEMENTS

The work of H. D. S. was partially supported by an NSERC Discovery Grant. This work does not have any conflicts of interest.

ORCID

Hans De Sterck https://orcid.org/0000-0002-1641-932X

REFERENCES

1. Nesterov Y. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. Soviet Math Doklady. 1983;27:372–376.
2. Ghadimi S, Lan G. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. Math Program. 2016;156(1-2):59–99.
3. Li H, Lin Z. Accelerated proximal gradient methods for nonconvex programming. Advances in Neural Information Processing Systems, 2015; p. 379–387.
4. Goldstein T, O'Donoghue B, Setzer S, Baraniuk R. Fast alternating direction optimization methods. SIAM J Imag Sci. 2014;7(3):1588–1623.
5. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51(3):455–500.
6. Nguyen N, Fernandez P, Freund R, Peraire J. Accelerated residual methods for the iterative solution of systems of equations. SIAM J Sci Comput. 2018;40(5):A3157–A3179.
7. O'donoghue B, Candes E. Adaptive restart for accelerated gradient schemes. Found Comput Math. 2015;15(3):715–732.

8. Su W, Boyd S, Candes EJ. A differential equation for modeling Nesterov's accelerated gradient method: theory and insights. J Mach Learn Res. 2016;17(153):1–43.
9. Ang AMS, Gillis N. Accelerating nonnegative matrix factorization algorithms using extrapolation. Neural Comput. 2019;31(2):417–439.
10. Washio T, Oosterlee CW. Krylov subspace acceleration for nonlinear multigrid schemes. Electr Trans Numer Anal. 1997;6(271-290):3–1.
11. Oosterlee CW, Washio T. Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows. SIAM J Sci Comput. 2000;21(5):1670–1690.
12. De Sterck H. A Nonlinear GMRES optimization algorithm for canonical tensor decomposition. SIAM J Sci Comput. 2012;34(3):A1351–A1379.
13. Brune PR, Knepley MG, Smith BF, Tu X. Composing scalable nonlinear algebraic solvers. SIAM Rev. 2015;57(4):535–565.
14. Harshman RA. Foundations of the PARAFAC procedure: models and conditions for an" explanatory" multimodal factor analysis; 1970.
15. Rajih M, Comon P, Harshman RA. Enhanced line search: A novel method to accelerate PARAFAC. SIAM J Matrix Anal Appl. 2008;30(3):1128–1147.
16. Chen Y, Han D, Qi L. New ALS methods with extrapolating search directions and optimal step size for complex-valued tensor decompositions. IEEE Trans Signal Process. 2011;59(12):5888–5898.
17. Sorber L, Domanov I, Van Barel M, De Lathauwer L. Exact line and plane search for tensor optimization. Comput Optimiz Appl. 2016;63(1):121–142.
18. De Sterck H, Winlaw M. A nonlinearly preconditioned conjugate gradient algorithm for rank-R canonical tensor approximation. Numer Linear Algebra Appl. 2015;22(3):410–432.
19. De Sterck H, Howse A. Nonlinearly preconditioned L-BFGS as an acceleration mechanism for alternating least squares, with application to tensor decomposition. Numer Linear Algebra Appl. 2018;25(6):e2202.
20. Ye H, Zhang Z. Nesterov's acceleration for approximate Newton. arXiv preprint arXiv:171008496; 2017.
21. Wang X, Navasca C, Kindermann S. On accelerating the regularized alternating least-squares algorithm for tensors. Electr Trans Numer Anal. 2018;48:1–14.
22. Acar E, Dunlavy DM, Kolda TG. A scalable optimization approach for fitting canonical tensor decompositions. J Chemom. 2011;25(2):67–86.
23. Paatero P. A weighted non-negative least squares algorithm for three-way ?PARAFAC?factor analysis. Chemom Intell Lab Syst. 1997;38(2):223–242.
24. Tomasi G, Bro R. A comparison of algorithms for fitting the PARAFAC model. Comput Stat Data Anal. 2006;50(7):1700–1734.
25. Phan AH, Tichavsky P, Cichocki A. Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC. SIAM J Matrix Anal Appl. 2013;34(1):126–147.
26. 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):3551–3582.
27. Nocedal J, Wright S. Numerical Optimization. Springer Science & Business Media; Berlin, Heidelberg / Germany: 2006.
28. De Sterck H. Steepest descent preconditioning for nonlinear GMRES optimization. Numer Linear Algebra Appl. 2013;20(3):453–471.
29. Hr F, Saad Y. Two classes of multisecant methods for nonlinear acceleration. Numer Linear Algebra Appl. 2009;16(3):197–221.
30. Walker HF, Ni P. Anderson acceleration for fixed-point iterations. SIAM J Numer Anal. 2011;49(4):1715–1735.
31. Hager WW, Zhang H. A survey of nonlinear conjugate gradient methods. Pacific J Optimiz. 2006;2(1):35–58.
32. Luenberger DG, Ye Y. Linear and Nonlinear Programming. Vol 2. New York, NY: Springer, 1984.
33. Bader BW, Kolda TG. MATLAB Tensor Toolbox Version 2.6; 2015.
34. Dunlavy DM, Kolda TG, Acar E. Poblano v1.0: A matlab toolbox for gradient-based optimization. Sandia National Laboratories. Tech. Rep. SAND2010-1422, Albuquerque, NM and Livermore, CA; 2010.
35. Chi EC, Kolda TG. On tensors, sparsity, and nonnegative factorizations. SIAM J Matrix Anal Appl. 2012;33(4):1272–1299.
36. Andersen CM, Bro R. Practical aspects of PARAFAC modeling of fluorescence excitation-emission data. J Chemom A J Chemom Soc. 2003;17(4):200–215.
37. Vergara A, Fonollosa J, Mahiques J, Trincavelli M, Rulkov N, Huerta R. On the performance of gas sensor arrays in open sampling systems using inhibitory support vector machines. Sens Actuat B Chem. 2013;185:462–477.
38. De Sterck H, Howse A. Nonlinearly preconditioned optimization on grassmann manifolds for computing approximate tucker tensor decompositions. SIAM J Sci Comput. 2016;38(2):A997–A1018.

How to cite this article: 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;e2297. https://doi.org/10.1002/nla.2297

APPENDIX A: PARAMETERS FOR SYNTHETIC CP TEST PROBLEMS

Table A1 lists the parameters for the standard ill-conditioned synthetic test problems used in the paper.[22] The specific choices of parameters for the six classes in Table A1 correspond to test problems (7)–(12) in De Sterck.[12] All tensors have equal size $s = I_1 = I_2 = I_3$ in the three tensor dimensions, and have high collinearity c. The six classes differ in their choice of tensor sizes (s), decomposition rank (R), and noise parameters l_1 and l_2.

APPENDIX B: DETAILED COMPARISONS FOR DIFFERENT RESTARTING STRATEGIES

Figures B1–B3 show τ-plots for variants of the restarted Nesterov-ALS schemes, for the case of function restart (RF, Figure B1), gradient restart (RG, Figure B2), and speed restart (RX, Figure B3), applied to the synthetic test problems.

For each of the restart mechanisms, several of the restarted Nesterov-ALS variants typically outperform ALS, NCG-ALS[18] and NGMRES-ALS.[12]

Several of the best-performing restarted Nesterov-ALS variants are also competitive with the best existing nonlinear acceleration method for ALS we compare with, LBFGS-ALS,[19] and they are much easier to implement.

TABLE A1 List of parameters for synthetic canonical polyadic test problems

Problem	s	c	R	l_1	l_2
1	20	0.9	3	0	0
2	20	0.9	5	1	1
3	50	0.9	3	0	0
4	50	0.9	5	1	1
5	100	0.9	3	0	0
6	100	0.9	5	1	1

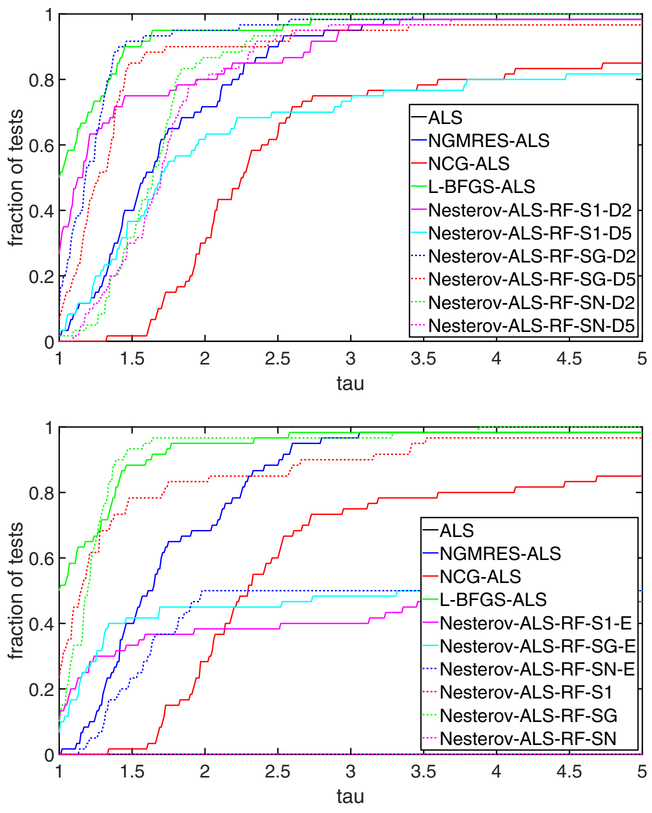


FIGURE B1 Synthetic test problems. τ-plots comparing variants of function restart. (top) Variants with delay. (bottom) Variants without delay

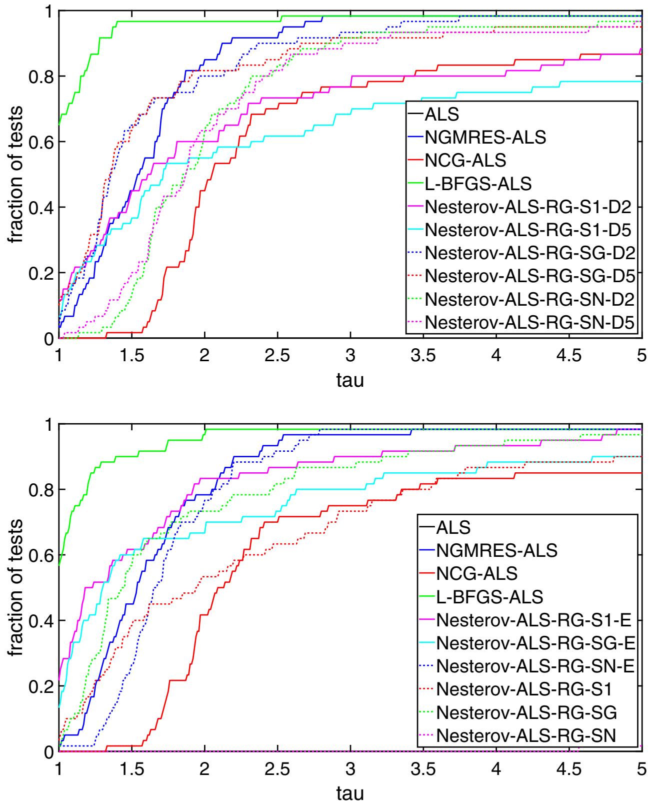


FIGURE B2 Synthetic test problems. τ-plots comparing variants of gradient restart. (top) Variants with delay. (bottom) Variants without delay

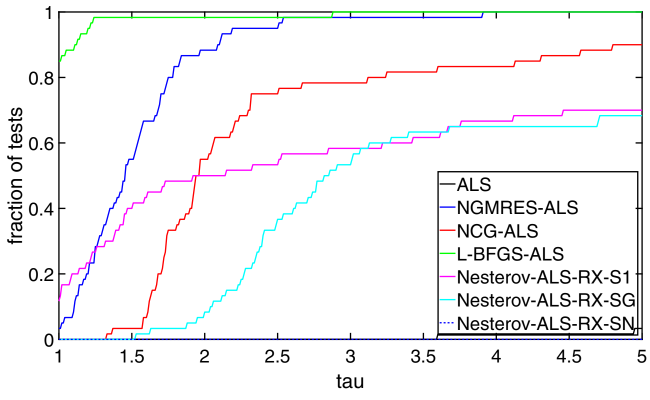


FIGURE B3 Synthetic test problems. τ-plot comparing variants of speed restart.

Among the restart mechanisms tested, function restart (Figure B1) substantially outperforms gradient restart (Figure B2), and, in particular, speed restart (Figure B3).

The τ-plots confirm that Nesterov-ALS-RF-SG, using function restarting and gradient ratio momentum weight, consistently performs as one of the best methods, making it our recommended choice for ALS acceleration.