A Nonlinearly Preconditioned Conjugate Gradient Algorithm for Rank-R Canonical Tensor Approximation

Hans De Sterck[†‡] Manda Winlaw[†‡]

Abstract

Alternating least squares (ALS) is often considered the workhorse algorithm for computing the rank-R canonical tensor approximation, but for certain problems its convergence can be very slow. The nonlinear conjugate gradient (NCG) method was recently proposed as an alternative to ALS, but the results indicated that NCG was usually not faster than ALS. To improve the convergence speed of NCG, we consider a nonlinearly preconditioned nonlinear conjugate gradient (PNCG) algorithm for computing the rank-R canonical tensor decomposition. Our approach uses ALS as a nonlinear preconditioner in the NCG algorithm. We demonstrate numerically that the convergence acceleration mechanism in PNCG often leads to important pay-offs for difficult tensor decomposition problems, with convergence that is significantly faster and more robust than for the stand-alone NCG or ALS algorithms. We consider several approaches for incorporating the nonlinear preconditioner into the NCG algorithm that have been described in the literature previously and have met with success in certain application areas. However, it appears that the nonlinearly preconditioned NCG approach has received relatively little attention in the broader community and remains underexplored both theoretically and experimentally. Thus, we provide a concise overview of several PNCG variants and their properties that have only been described in a few places scattered throughout the literature. We also systematically compare the performance of these PNCG variants for the tensor decomposition problem, and draw further attention to the usefulness of nonlinearly preconditioned NCG as a general tool. In addition, we obtain a new convergence result for one of the PNCG variants under suitable conditions, building on known convergence results for non-preconditioned NCG.

1 Introduction

In this paper, we consider a nonlinearly preconditioned nonlinear conjugate gradient (PNCG) algorithm for computing a canonical rank-R tensor approximation using the Frobenius norm as a distance metric. The current workhorse algorithm for computing the canonical tensor decomposition is the alternating least squares (ALS) algorithm [1, 2, 3]. The ALS method is simple to understand and implement, but for certain problems its convergence can be very slow [4, 3]. In [5], the nonlinear conjugate gradient (NCG) method is considered as an alternative to ALS for solving canonical tensor decomposition problems. However, [5] found that NCG is usually not faster than ALS. In this paper, we show how incorporating ALS as a nonlinear preconditioner into the NCG algorithm (or, equivalently, accelerating ALS by the NCG algorithm) may lead to significant convergence acceleration for difficult canonical tensor decomposition problems.

Our approach is among extensive, recent, research activity on nonlinear preconditioning for nonlinear iterative solvers [6, 7, 8, 9, 10], including nonlinear GMRES and NCG. This work builds on original contributions dating back as far as the 1960s [11, 12, 13, 14], but much of this early work is not well-known in the broader community and large parts of the landscape remain unexplored experimentally and theoretically [10]; the recent paper [10] gives a comprehensive overview of the state of the art in nonlinear preconditioning and provides interesting new directions.

In this paper we, consider nonlinear preconditioning of NCG for the canonical tensor decomposition problem. We consider several approaches for incorporating the nonlinear preconditioner into the NCG algorithm that are described in the literature (see [15, 12, 16, 7, 10]). Early references to nonlinearly preconditioned NCG include [15] and [12]. Both propose the NCG algorithm as a solution method for solving nonlinear elliptic partial differential equations (PDEs) and while both present NCG algorithms that include a possible nonlinear preconditioner, [12] actually uses a block nonlinear SSOR method as the nonlinear preconditioner in their numerical experiments. Hager and Zhang's survey paper [17] describes a linearly preconditioned NCG algorithm, but does not discuss general nonlinear preconditioning for NCG. More

[†]Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, N2L 3G1, Canada (hdesterck@uwaterloo.ca, mwinlaw@uwaterloo.ca)

[‡]This work was supported by NSERC of Canada and was supported in part by the Scalable Graph Factorization LDRD Project, 13-ERD-072, under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

recent work on nonlinearly preconditioned NCG includes [7] and the recent overview paper [10] on nonlinear preconditioning also briefly mentions nonlinearly preconditioned NCG, but discusses a different variant than [15], [12], [16] and [7]. In Section 3, the differences between the PNCG variants of [15, 12, 16, 7, 10] will be explained.

As mentioned above, we apply the PNCG algorithm to the tensor decomposition problem which can be described as follows. Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ be a N-way or Nth-order tensor of size $I_1 \times I_2 \times \ldots \times I_N$. We are interested in finding a canonical rank-R tensor, $\mathcal{A}_R \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, as an approximation to $\mathcal{X}$ by minimizing the following function:

$$f(\mathcal{A}_R) = \frac{1}{2}\|\mathcal{X} - \mathcal{A}_R\|_F^2, \text{ where } \mathcal{A}_R = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \ldots \circ \mathbf{a}_r^{(N)} = [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]. \tag{1}$$

and $\|\cdot\|_F$ denotes the Frobenius norm of the N-dimensional array. The canonical tensor $\mathcal{A}_R$ is the sum of R rank-one tensors, with the rth rank-one tensor composed of the outer product of N column vectors $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n},\ n = 1, \ldots, N$. The decomposition of $\mathcal{X}$ into $\mathcal{A}_R$ is known as the canonical tensor decomposition and is commonly referred to as the CP decomposition [1, 2]. The canonical tensor decomposition is commonly used as a data analysis technique in a wide variety of fields including chemometrics, signal processing, neuroscience and web analysis.

The ALS algorithm for CP decomposition was first proposed in papers by Carroll and Chang [1] and Harshman [2]. An overview of the ALS algorithm can be found in [3]. The ALS method is simple to understand and implement, but can take many iterations to converge. It is not guaranteed to converge to a global minimum or even a stationary point of (1) [3]. We can only guarantee that the objective function in (1) is nonincreasing at every step of the ALS algorithm. As well, if the ALS algorithm does converge to a stationary point, the stationary point can be heavily dependent on the starting guess. A number of algorithms have been proposed as alternatives to the ALS algorithm. See [5, 3, 4, 18] and the references therein for examples. Inspired by the nonlinearly preconditioned nonlinear GMRES method of [9], we propose in this paper to accelerate the NCG approach of [5] by considering the use of ALS as a nonlinear preconditioner for NCG.

The remainder of the paper is structured as follows. In Section 2, we introduce the standard NCG algorithm for unconstrained continuous optimization. Section 3 gives a concise description of several variants of the PNCG algorithm that we collect from the literature and describe systematically, and it discusses their relation to the PCG algorithm in the linear case, followed by a brief convergence discussion highlighting our new convergence result. In Section 4 we follow the experimental procedure of Tomasi and Bro [4] to generate test tensors that we use to systematically compare the PNCG variants we have described with the standard ALS and NCG algorithms. Section 5 concludes.

2 Nonlinear Conjugate Gradient Algorithm

The NCG algorithm for continuous optimization is an extension of the CG algorithm for linear systems. The CG algorithm minimizes the convex quadratic function

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}, \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a SPD matrix. Equivalently, the CG algorithm can be viewed as an iterative method for solving the linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. The NCG algorithm is adapted from the CG algorithm and can be applied to any unconstrained optimization problem of the form

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \tag{3}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function bounded from below. The NCG algorithm is a line search algorithm that generates a sequence of iterates $\mathbf{x}_i,\ i \geq 1$ from the initial guess $\mathbf{x}_0$ using the recurrence relation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k. \tag{4}$$

The parameter $\alpha_k > 0$ is the step length and $\mathbf{p}_k$ is the search direction generated by the following rule:

$$\mathbf{p}_{k+1} = -\mathbf{g}_{k+1} + \beta_{k+1}\mathbf{p}_k, \ \ \mathbf{p}_0 = -\mathbf{g}_0, \tag{5}$$

where β_{k+1} is the update parameter and $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is the gradient of f evaluated at $\mathbf{x}_k$. In the CG algorithm, α_k and β_{k+1} are defined as

$$\alpha_k^{CG} = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}, \qquad \beta_{k+1}^{CG} = \frac{\mathbf{r}_{k+1}^T\mathbf{r}_{k+1}}{\mathbf{r}_k^T\mathbf{r}_\mathbf{k}}, \tag{6, 7}$$

where $\mathbf{r}_k = \nabla\phi(\mathbf{x}_k) = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ is the residual. In the nonlinear case α_k is determined by a line search algorithm and β_{k+1} can assume various different forms. We consider three different forms in this paper, given by

$$\beta_{k+1}^{FR} = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}, \qquad \beta_{k+1}^{PR} = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{g}_k}, \qquad \beta_{k+1}^{HS} = \frac{\mathbf{g}_{k+1}^T (\mathbf{g}_{k+1} - \mathbf{g}_k)}{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{p}_k}. \tag{8, 9,10}$$

Fletcher and Reeves [19] first showed how to extend the conjugate gradient algorithm to the nonlinear case. By replacing the residual, $\mathbf{r}_k$, with the gradient of the nonlinear objective f, they obtained a formula for β_{k+1} of the form β_{k+1}^{FR}. The variant β_{k+1}^{PR} was developed by Polak and Ribière [20] and the Hestenes-Stiefel [21] formula is given by Equation (10). For all three versions, it can easily be shown that, if a convex quadratic function is optimized using the NCG algorithm and the line search is exact then $\beta_{k+1}^{FR} = \beta_{k+1}^{PR} = \beta_{k+1}^{HS} = \beta_{k+1}^{CG}$ where β_{k+1}^{CG} is given by Equation (7), see [22].

3 Preconditioned Nonlinear Conjugate Gradient Algorithm

In this section we give a concise description of several variants of PNCG that have been proposed in a few places in the literature but have not been discussed and compared systematically in one place, briefly discuss some of their relevant properties, and prove a new convergence property for one of the variants. Before we introduce PNCG we briefly describe the PCG algorithm for linear systems. We do this because it will be useful for interpreting some of the variants for β_{k+1} in the PNCG algorithm.

The PCG algorithm is derived from the CG algorithm by introducing a linear change of variables. Consider a change of variables from $\mathbf{x}$ to $\widehat{\mathbf{x}}$ via a nonsingular matrix $\mathbf{C}$ such that $\widehat{\mathbf{x}} = \mathbf{C}\mathbf{x}$. The new objective function is

$$\widehat{\phi}(\widehat{\mathbf{x}}) = \frac{1}{2}\widehat{\mathbf{x}}^T(\mathbf{C}^{-T}\mathbf{A}\mathbf{C}^{-1})\widehat{\mathbf{x}} - (\mathbf{C}^{-T}\mathbf{b})^T\widehat{\mathbf{x}}, \tag{11}$$

and by applying the CG algorithm to (11) we arrive at the PCG algorithm. Preconditioning the CG algorithm is commonly used in numerical linear algebra to speed up convergence [23]. In the CG algorithm, the properties of the matrix $\mathbf{A}$, particularly the eigenvalue distribution of $\mathbf{A}$, determine the convergence speed of the algorithm. In the PCG algorithm, it is the properties of $\mathbf{C}^{-T}\mathbf{A}\mathbf{C}^{-1}$ that determine the convergence speed of the algorithm and if $\mathbf{C}$ is chosen to improve the properties of $\mathbf{C}^{-T}\mathbf{A}\mathbf{C}^{-1}$ over $\mathbf{A}$ then we can significantly speed up the convergence of PCG over CG.

We can also apply a linear change of variables, $\widehat{\mathbf{x}} = \mathbf{C}\mathbf{x}$, to the NCG algorithm as is explained in review paper [17]. The linearly preconditioned NCG algorithm expressed in terms of the original variable $\mathbf{x}$ can be described by the following equations:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{12}$$

$$\mathbf{p}_{k+1} = -\mathbf{P}\mathbf{g}_{k+1} + \breve{\beta}_{k+1}\mathbf{p}_k, \;\; \mathbf{p}_0 = -\mathbf{P}\mathbf{g}_0, \tag{13}$$

where $\mathbf{P} = \mathbf{C}^{-1}\mathbf{C}^{-T}$. The formulas for $\breve{\beta}_{k+1}$ remain the same as before (Equations (8)-(10)), except that $\mathbf{g}_k$ and $\mathbf{p}_k$ are replaced by $\mathbf{C}^{-T}\mathbf{g}_k$ and $\mathbf{C}\mathbf{p}_k$, respectively. Thus, we obtain linearly preconditioned versions of the β_{k+1} parameters of Equations (8)-(10):

$$\breve{\beta}_{k+1}^{FR} = \frac{\mathbf{g}_{k+1}^T \mathbf{P}\mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{P}\mathbf{g}_k}, \qquad \breve{\beta}_{k+1}^{PR} = \frac{\mathbf{g}_{k+1}^T \mathbf{P}(\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k^T \mathbf{P}\mathbf{g}_k}, \qquad \breve{\beta}_{k+1}^{HS} = \frac{\mathbf{g}_{k+1}^T \mathbf{P}(\mathbf{g}_{k+1} - \mathbf{g}_k)}{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T \mathbf{p}_k}. \tag{14, 15,16}$$

If we use the linearly preconditioned NCG algorithm with these $\breve{\beta}_{k+1}$ formulas to minimize the convex quadratic function, $\phi(\mathbf{x})$, defined in Equation (2), using an exact line search, where $\mathbf{g}_k = \mathbf{r}_k$, then the algorithm is the same as the PCG algorithm.

Suppose instead, we wish to introduce a nonlinear transformation of $\mathbf{x}$. In particular, suppose we consider a nonlinear iterative optimization method such as Gauss-Seidel. Let $\overline{\mathbf{x}}_k$ be the preliminary iterate generated by one step of a nonlinear iterative method, i.e., we write

$$\overline{\mathbf{x}}_k = P(\mathbf{x}_k), \tag{17}$$

which we will use as a nonlinear preconditioner. Now define the direction generated by the nonlinear preconditioner as

$$\overline{\mathbf{g}}_k = \mathbf{x}_k - \overline{\mathbf{x}}_k = \mathbf{x}_k - P(\mathbf{x}_k). \tag{18}$$

In nonlinearly preconditioned NCG, one considers the nonlinearly preconditioned direction, $\overline{\mathbf{g}}_k$, instead of the gradient, $\mathbf{g}_k$, in formulating the NCG method [15, 12, 16, 7, 10]. This idea can be motivated by the linear preconditioning of CG, where $\mathbf{g}_k = \mathbf{r}_k$ is replaced by the preconditioned gradient $\mathbf{P}\mathbf{g}_k = \mathbf{P}\mathbf{r}_k$ in

certain parts of the algorithm. This corresponds to replacing the Krylov space for CG, which is formed by the gradients $\mathbf{g}_k = \mathbf{r}_k$, with the left-preconditioned Krylov space for PCG, which is formed by the preconditioned gradients $\mathbf{P}\mathbf{g}_k = \mathbf{P}\mathbf{r}_k$ [23]. In a similar way, we replace the nonlinear gradients $\mathbf{g}_k$ with the nonlinearly preconditioned directions $\overline{\mathbf{g}}_k$.

Thus, our nonlinearly preconditioned NCG algorithm is given by the following equations:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{19}$$

$$\mathbf{p}_{k+1} = -\overline{\mathbf{g}}_{k+1} + \overline{\beta}_{k+1}\mathbf{p}_k, \;\; \mathbf{p}_0 = -\overline{\mathbf{g}}_0. \tag{20}$$

The formulas for $\overline{\beta}_{k+1}$ in Equation (20) are modified versions of the β_{k+1} from Equations (8)-(10) that incorporate $\overline{\mathbf{g}}_k$. However, there are several different ways to modify the β_{k+1} to incorporate $\overline{\mathbf{g}}_k$, leading to several different variants of $\overline{\beta}_{k+1}$. Table 1 summarizes the variants of $\overline{\beta}_{k+1}$ we consider in this paper for PNCG.

	Fletcher-Reeves	Polak-Ribière	Hestenes-Stiefel
$\widetilde{\beta}_{k+1}$	$\widetilde{\beta}^{FR}_{k+1} = \dfrac{\overline{\mathbf{g}}^T_{k+1}\overline{\mathbf{g}}_{k+1}}{\overline{\mathbf{g}}^T_k\overline{\mathbf{g}}_k}$	$\widetilde{\beta}^{PR}_{k+1} = \dfrac{\overline{\mathbf{g}}^T_{k+1}(\overline{\mathbf{g}}_{k+1} - \overline{\mathbf{g}}_k)}{\overline{\mathbf{g}}^T_k\overline{\mathbf{g}}_k}$	$\widetilde{\beta}^{HS}_{k+1} = \dfrac{\overline{\mathbf{g}}^T_{k+1}(\overline{\mathbf{g}}_{k+1} - \overline{\mathbf{g}}_k)}{(\overline{\mathbf{g}}_{k+1} - \overline{\mathbf{g}}_k)^T\mathbf{p}_k}$
$\widehat{\beta}_{k+1}$	$\widehat{\beta}^{FR}_{k+1} = \dfrac{\mathbf{g}^T_{k+1}\overline{\mathbf{g}}_{k+1}}{\mathbf{g}^T_k\overline{\mathbf{g}}_k}$	$\widehat{\beta}^{PR}_{k+1} = \dfrac{\mathbf{g}^T_{k+1}(\overline{\mathbf{g}}_{k+1} - \overline{\mathbf{g}}_k)}{\mathbf{g}^T_k\overline{\mathbf{g}}_k}$	$\widehat{\beta}^{HS}_{k+1} = \dfrac{\mathbf{g}^T_{k+1}(\overline{\mathbf{g}}_{k+1} - \overline{\mathbf{g}}_k)}{(\mathbf{g}_{k+1} - \mathbf{g}_k)^T\mathbf{p}_k}$

Table 1: Variants of $\overline{\beta}_{k+1}$ for the Nonlinearly Preconditioned Nonlinear Conjugate Gradient Algorithm (PNCG).

The first set of $\overline{\beta}_{k+1}$ variants we consider are the $\widetilde{\beta}_{k+1}$ shown in row 1 of Table 1. The $\widetilde{\beta}_{k+1}$ formulas are derived by replacing all occurrences of $\mathbf{g}_k$ with $\overline{\mathbf{g}}_k$ in the formulas for β_{k+1}, Equations (8)-(10). This is a straightforward generalization of the β_{k+1} expressions in Equations (8)-(10), and the systematic numerical comparisons to be presented in Section 4 indicate that these choices lead to efficient PNCG methods. The PR variant of this formula is used in [10] in the context of PDE solvers.

However, suppose we want the $\overline{\beta}_{k+1}$ formulas to reduce to the PCG update formulas in the linear case. Indeed, suppose we apply the PNCG algorithm to the convex quadratic problem, (2), with an exact line search, using a symmetric stationary linear iterative method such as symmetric Gauss-Seidel or Jacobi as a preconditioner. We begin by writing the stationary iterative method in general form as

$$\overline{\mathbf{x}}_k = P(\mathbf{x}_k) = \mathbf{x}_k - \mathbf{P}\mathbf{r}_k, \tag{21}$$

where the SPD preconditioning matrix $\mathbf{P}$ is often written as $\mathbf{M}^{-1}$ and $\mathbf{r}_k = \mathbf{g}_k$. The search direction $\overline{\mathbf{g}}_k$ from Equation (18) simply becomes

$$\overline{\mathbf{g}}_k = \mathbf{x}_k - \overline{\mathbf{x}}_k = \mathbf{P}\mathbf{r}_k = \mathbf{P}\mathbf{g}_k. \tag{22}$$

This immediately suggests a generalization of the linearly preconditioned NCG parameters, $\breve{\beta}_{k+1}$, to the case of nonlinear preconditioning: replacing all occurrences of $\mathbf{P}\mathbf{g}_k$ with $\overline{\mathbf{g}}_k$ we obtain the expressions, $\widehat{\beta}_{k+1}$, in row 2 of Table 1. Expressions of this type have been used in [15, 12, 16, 7]. It is clear that the PNCG algorithm with this second set of expressions, reduces to the PCG algorithm in the linear case, since the $\widehat{\beta}_{k+1}$ reduce to the $\breve{\beta}_{k+1}$ in the case of a linear preconditioner, and the $\breve{\beta}_{k+1}$ in turn reduce to the β_{k+1} from the PCG algorithm when solving an SPD linear system. Thus, for the PNCG method, we have two sets of $\overline{\beta}_{k+1}$ formulas and we will use both the $\widetilde{\beta}_{k+1}$ and $\widehat{\beta}_{k+1}$ formulas in our numerical tests.

Next we investigate aspects of convergence of the PNCG algorithm. For the NCG algorithm without preconditioning, global convergence can be proved for the Fletcher-Reeves method applied to a broad class of objective functions, in the sense that

$$\lim_{k\to\infty} \inf \|\mathbf{g}_k\| = 0, \tag{23}$$

when the line search satisfies the strong Wolfe conditions (see [22] for a general discussion on NCG convergence). Global convergence cannot be proved in general for the Polak-Ribière or Hestenes-Stiefel variants. Nevertheless, these methods are also widely used and may perform better than Fletcher-Reeves in practice. Global convergence can be proved for variants of these methods in which every search direction $\mathbf{p}_k$ is guaranteed to be a descent direction ($\mathbf{g}^T_k\mathbf{p}_k < 0$), and in which the iteration is restarted periodically with a steepest-descent step.

General convergence results for the PNCG algorithm are also difficult to obtain. However, with the use of the following theorem we will be able to establish global convergence for a restarted version of the $\widehat{\beta}_k^{FR}$ variant of the PNCG algorithm under suitable conditions. The proof of the theorem relies on showing that the PNCG search directions $\mathbf{p}_k$ obtained using $\widehat{\beta}_k^{FR}$ are descent directions when the nonlinear preconditioner produces descent directions. To show this we follow the proof technique of Lemma 5.6 in [22].

Theorem 1. *Consider the PNCG algorithm with $\overline{\beta}_{k+1} = \widehat{\beta}_{k+1}^{FR}$ and where α_k satisfies the strong Wolfe conditions. Let $P(\mathbf{x})$ be a nonlinear preconditioner such that $-\overline{\mathbf{g}}(\mathbf{x}_k) = P(\mathbf{x}_k) - \mathbf{x}_k$ is a descent direction for all k, i.e., $-\mathbf{g}_k^T\overline{\mathbf{g}}_k < 0$. Suppose the objective function f is bounded below in $\mathbb{R}^n$ and f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} := \{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{x}_0)\}$, where $\mathbf{x}_0$ is the starting point of the iteration. Assume also that the gradient $\mathbf{g}_k$ is Lipshitz continuous on $\mathcal{N}$. Then,*

$$\sum_{k\geq 0} \cos^2\theta_k \|\mathbf{g}_k\|^2 < \infty, \qquad \cos\theta_k = \frac{-\mathbf{g}_k^T\mathbf{p}_k}{\|\mathbf{g}_k\|\|\mathbf{p}_k\|}. \tag{24, 25}$$

Proof. We show that $\mathbf{p}_k$ is a descent direction, i.e., $\mathbf{g}_k^T\mathbf{p}_k < 0 \;\forall\; k$. Then condition (24) follows directly from Theorem 3.2 of Nocedal and Wright [22] which states that condition (24) holds for any iteration of the form $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$ provided that the above conditions hold for α_k, f and $\mathbf{g}_k$, and where $\mathbf{p}_k$ is a descent direction.

Instead of proving that $\mathbf{g}_k^T\mathbf{p}_k < 0$ directly, we will prove the following:

$$-\frac{1}{1-c_2} \leq \frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_k^T\overline{\mathbf{g}}_k} \leq \frac{2c_2-1}{1-c_2}, \quad k \geq 0, \tag{26}$$

where $0 < c_2 < \frac{1}{2}$ is the constant from the curvature condition of the strong Wolfe conditions:

$$|\mathbf{g}_{k+1}^T\mathbf{p}_k| \leq c_2|\mathbf{g}_k^T\mathbf{p}_k|. \tag{27}$$

Note, that the following two conditions hold

$$-1 < \frac{2c_2-1}{1-c_2} < 0, \qquad -2 < -\frac{1}{1-c_2} < -1. \tag{28, 29}$$

Condition (28) holds because the function $t(\xi) = (2\xi-1)/(1-\xi)$ is monotonically increasing on the interval $[0, \frac{1}{2}]$, $t(0) = -1$ and $t(\frac{1}{2}) = 0$ and $c_2 \in (0, \frac{1}{2})$. Similarily, condition (29) holds because the function $t(\xi) = -1/(1-\xi)$ is monotonically decreasing on the interval $[0, \frac{1}{2}]$, $t(0) = -1$ and $t(\frac{1}{2}) = -2$ and $c_2 \in (0, \frac{1}{2})$. Also note that since $-\overline{\mathbf{g}}_k$ is a descent direction, $\mathbf{g}_k^T\overline{\mathbf{g}}_k > 0$. So, if (26) holds then $\mathbf{g}_k^T\mathbf{p}_k < 0$ and $\mathbf{p}_k$ is a descent direction.

We use an inductive proof to show that (26) is true. For $k = 0$, we use the definition of $\mathbf{p}_0$ to get,

$$\frac{\mathbf{g}_0^T\mathbf{p}_0}{\mathbf{g}_0^T\overline{\mathbf{g}}_0} = \frac{-\mathbf{g}_0^T\overline{\mathbf{g}}_0}{\mathbf{g}_0^T\overline{\mathbf{g}}_0} = -1. \tag{30}$$

Then, from the (28) and (29) we get

$$-\frac{1}{1-c_2} < \frac{\mathbf{g}_0^T\mathbf{p}_0}{\mathbf{g}_0^T\overline{\mathbf{g}}_0} < \frac{2c_2-1}{1-c_2}. \tag{31}$$

Now suppose that (26) holds for $l = 1, \ldots, k$. We need to show that (26) is true for $k+1$. Using the definition of $\mathbf{p}_{k+1}$ we have,

$$\frac{\mathbf{g}_{k+1}^T\mathbf{p}_{k+1}}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = \frac{\mathbf{g}_{k+1}^T\left(-\overline{\mathbf{g}}_{k+1} + \widehat{\beta}_{k+1}^{FR}\mathbf{p}_k\right)}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = -1 + \widehat{\beta}_{k+1}^{FR}\frac{\mathbf{g}_{k+1}^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}}. \tag{32}$$

From the Wolfe condition, Equation (27), and the inductive hypothesis, which implies that $\mathbf{g}_k^T\mathbf{p}_k < 0$, we can write

$$c_2\mathbf{g}_k^T\mathbf{p}_k \leq \mathbf{g}_{k+1}^T\mathbf{p}_k \leq -c_2\mathbf{g}_k^T\mathbf{p}_k. \tag{33}$$

Combining this with Equation (32), we have

$$-1 + c_2\widehat{\beta}_{k+1}^{FR}\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} \leq \frac{\mathbf{g}_{k+1}^T\mathbf{p}_{k+1}}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} \leq -1 - c_2\widehat{\beta}_{k+1}^{FR}\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} \tag{34}$$

So,

$$-1 + c_2\widehat{\beta}_{k+1}^{FR}\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = -1 + c_2\left(\frac{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}}{\mathbf{g}_k^T\overline{\mathbf{g}}_k}\right)\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = -1 + c_2\left(\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_k^T\overline{\mathbf{g}}_k}\right) \geq -1 - \frac{c_2}{1-c_2} = -\frac{1}{1-c_2},$$

and

$$-1 - c_2\widehat{\beta}_{k+1}^{FR}\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = -1 - c_2\left(\frac{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}}{\mathbf{g}_k^T\overline{\mathbf{g}}_k}\right)\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_{k+1}^T\overline{\mathbf{g}}_{k+1}} = -1 - c_2\left(\frac{\mathbf{g}_k^T\mathbf{p}_k}{\mathbf{g}_k^T\overline{\mathbf{g}}_k}\right) \leq -1 + \frac{c_2}{1-c_2} = \frac{2c_2-1}{1-c_2}.$$

□

We can now easily establish that convergence holds for a restarted version of the PNCG algorithm with $\widehat{\beta}_{k+1}^{FR}$ if a nonlinear preconditioner is used that produces descent directions: If we use the steepest decent direction as the search direction on every mth iteration of the algorithm and then restart the PNCG algorithm with $\mathbf{p}_{m+1} = -\overline{\mathbf{g}}_{m+1} = -\mathbf{x}_m + P(\mathbf{x}_m)$, then Equation (24) of Theorem 1 is satisfied for the combined process and (23) holds since $\cos\theta_k = 1$ for the steepest descent steps [22]. Thus we are guaranteed overall global convergence for this method. Note that the proof for global convergence of NCG using β_{k+1}^{FR} without restarting (Theorem 5.7 in [22]) does not carry over to the case of unrestarted PNCG with $\widehat{\beta}_{k+1}^{FR}$.

The convergence result requires that the nonlinearly preconditioned directions $-\overline{\mathbf{g}}_k = P(\mathbf{x}_k) - \mathbf{x}_k$ be descent directions. If one assumes a continuous preconditioning function $P(\mathbf{x})$ such that $-\overline{\mathbf{g}}(\mathbf{x}) = P(\mathbf{x}) - \mathbf{x}$ is a descent direction for all $\mathbf{x}$ in a neighbourhood of an isolated local minimizer $\mathbf{x}^*$ of a continuously differentiable objective function $f(\mathbf{x})$, then this implies that the nonlinear preconditioner satisfies the fixed-point condition $\mathbf{x}^* = P(\mathbf{x}^*)$, which is a natural condition for a nonlinear preconditioner. It is often the case in nonlinear optimization that convergence results only hold under restrictive conditions and are mainly of theoretical value. In practice, numerical results may show satisfactory convergence behaviour for much broader classes of problems. Our numerical results will show that this is also the case for PNCG applied to canonical tensor decomposition: While the ALS preconditioner satisfies the fixed-point property, it is not guaranteed to produce descent directions. Nevertheless, convergence was generally observed numerically for all the PNCG variants we considered, with the $\widetilde{\beta}_{PR}$ variant producing the fastest results in most cases.

4 Numerical Results

To test our PNCG algorithm we randomly generate artificial tensors of different sizes, ranks, collinearity, and heteroskedastic and homoskedastic noise, which constitute standard test problems for CP decomposition [4]. We then compare the speed and accuracy of the CP factorization using the PNCG algorithm with results from using the ALS and NCG algorithms.

4.1 Problem Description

The artificial tensors are generated using the methodology of [4]. All the tensors we consider are 3-way tensors. Each dimension has the same size but we consider tensors of three different sizes, $I = 20$, 50 and 100. The factor matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$, and $\mathbf{A}^{(3)}$ are generated randomly so that the collinearity of the factors in each mode is set to a particular level C where,

$$C = \frac{\mathbf{a}_r^{(n)T}\mathbf{a}_s^{(n)}}{\|\mathbf{a}_r^{(n)}\|\|\mathbf{a}_s^{(n)}\|}, \tag{35}$$

for $r \neq s,\ r,s = 1,\ldots,R$ and $n = 1,2,3$. As in [5], the values of C we consider are 0.5 and 0.9, where higher values of C make the problem more difficult. We consider two different values for the rank, $R = 3$ and $R = 5$. For each combination of R and C we generate 20 different sets of factor matrices. Once we have converted these factors into tensors and added noise our goal is to recover these underlying factors using the different optimization algorithms. From a given set of factor matrices we are able to generate nine test tensors by adding different levels of homoskedastic and heteroskedastic noise. Homoskedastic noise refers to noise with constant variance whereas heteroskedastic noise refers to noise with differing variance. The noise ratios we consider for homoskedastic and heteroskedastic noise are $l_1 = 1, 5, 10$ and $l_2 = 0, 1, 5$, respectively, see [4, 5]. For each size, $I = 20$, 50 and 100 we generate 720 test tensors since we consider 2 different ranks, 2 different collinearity values, 20 sets of factor matrices for each combination of C and R and 9 different levels of noise.

4.2 Results

All numerical experiments where performed on a Linux Workstation with a Quad-Core Intel Xeon 3.16GHz processor and 8GB RAM. We use the NCG algorithm from the Poblano toolbox for MATLAB [24] which uses the Morè-Thuente line search algorithm. We use the same line search algorithm for the PNCG algorithm. The line search parameters are as follows: 10^{-4} for the sufficient decrease condition tolerance, 10^{-2} for the curvature condition tolerance, an initial step length of 1 and a maximum of 20 iterations. The ALS algorithm we use is from the tensor toolbox for MATLAB [25]; however, we use a different normalization of the factors and we use the gradient norm as a stopping condition instead of the relative function change. It is often useful to assume that the columns of the factor matrices, $\mathbf{A}^{(n)}$, are normalized to length one with the weights absorbed into a vector $\boldsymbol{\lambda} \in \mathbb{R}^k$ where $\mathcal{X} \approx \sum_{r=1}^{k} \lambda_r a_r^{(1)} \circ \ldots \circ a_r^{(N)}$. In our ALS algorithm the factors are normalized such that $\boldsymbol{\lambda}$ is distributed evenly over all the factors. Also note that, while the gradient norm is used as a stopping condition for the ALS algorithm, the calculation of the gradient is not included in the timing results for the ALS algorithm. For all three algorithms, ALS, NCG and PNCG, there are three stopping conditions; all are set to the same value for each algorithm. They are as follows: 10^{-9} for the gradient norm divided by the number of variables, $\|\mathbf{G}(\mathcal{A}_R)\|_2/N$ where N is the number of variables in $\mathcal{X}$, 10^4 for the maximum number of iterations and 10^5 for the maximum number of function evaluations. Also note, that for a given test tensor, each algorithm starts from the same initial guess with components chosen randomly from a uniform distribution between 0 and 1.

For the PNCG and NCG algorithms we only consider the β_{k+1}^{PR} variants since the results for β_{k+1}^{FR} and β_{k+1}^{HS} are similar. We also only show results for $C = 0.9$. When the collinearity is 0.5, it is known that the problem is relatively easy [4, 5, 9], so we don't necessarily expect the preconditioned algorithm to outperform the standard algorithm, and the additional time needed to perform the preconditioning may actually slow the algorithm down relative to the original algorithm. In fact, for all combinations of I and R, when $C = 0.5$ the ALS algorithm is the fastest algorithm and for a given formula for β (PR, FR, or HS) the NCG algorithm is faster than both PNCG variants. Thus, the case where $C = 0.9$ is more interesting for investigating the performance of a preconditioned algorithm. The timing results presented are written in the form $a \pm b$ where a is the mean time and b is the standard deviation. The numbers in brackets represent the number of CP decompositions that converge out of a possible 180 since for a given value of I, R and C there are twenty test tensors each with nine different combinations of homoskedastic and heteroskedastic noise added to them. All timing calculations are performed for the converged runs only.

The timing results may be dominated by a small number of difficult problems. Including the standard deviation helps to describe the effects of this bias; however, the timing results don't account for the problems where the algorithm fails to converge within the prescribed resource limit. One way to overcome this is to use the performance profiles suggested by Dolan and Moré in [26]. Suppose that we want to compare the performance of a set of algorithms or solvers $\mathcal{S}$ on a test set $\mathcal{P}$. Suppose there are n_s algorithms and n_p problems. For each problem $p \in \mathcal{P}$ and algorithm $s \in \mathcal{S}$ let $t_{p,s}$ be the computing time required to solve problem p using algorithm s. In order to compare algorithms we use the best performance by any algorithm as a baseline and define the performance ratio as $r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$. Although we may be interested in the performance of algorithm s on a given problem p, a more insightful analysis can be performed if we can obtain an overall assessment of the algorithm's performance. We can do this by defining the following: $\rho_s(\tau) = \frac{1}{n_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}$. For algorithm $s \in \mathcal{S}$, $\rho_s(\tau)$ is the fraction of problems p for which the performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best ratio (which equals one). Thus, $\rho_s(\tau)$ is the cumulative distribution function for the performance ratio and we refer to it as the performance profile. By visually examining the performance profiles of each algorithm we can compare the algorithms in $\mathcal{S}$. In particular, algorithms with large fractions $\rho_s(\tau)$ are preferred since algorithms with high values of ρ_s relative to the other algorithms indicate robust solvers. Also note that the value of ρ_s at $\tau = 1$ is the fraction of wins for each solver.

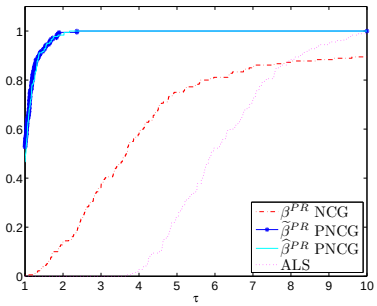
For all three values of I, Table 2 summarizes the timing results and Figure 1 plots the performance profiles for $I = 20$ and 100 for the algorithms in $\mathcal{S} = \{$ALS, NCG with β^{PR}, PNCG with $\widetilde{\beta}^{PR}$, PNCG with $\widehat{\beta}^{PR}\}$. From Table 2 we can see that for $I = 20$ and 50, and both values of R, the PNCG algorithm with the $\widetilde{\beta}^{PR}$ variant is the fastest. The ALS algorithm is the slowest and both PNCG algorithms are faster than the NCG algorithm, thus, nonlinear preconditioning significantly speeds up the NCG algorithm for these values of I. This is also true for $I = 100$ and $R = 3$. However, in the case when $I = 100$, $R = 5$, Table 2 shows that the ALS algorithm is the fastest on average.

Next, we examine the performance profiles in Figure 1. Since $\rho_s(1)$ indicates what fraction of the 180 trials each algorithm is the fastest, we see from Figure 1(a) that when $I = 20$ and $R = 3$, PNCG with $\widetilde{\beta}_{PR}$ is the fastest algorithm in the largest percentage of runs. When $\tau = 6$ approximately 50% of the 180 NCG runs are within six times the fastest time and approximately 80% of the ALS runs are within six times the fastest time. However, as τ increases to 10 we notice that approximately all of the ALS and PNCG runs are within ten times the fastest time but only 90% of the NCG runs are within ten times the fastest

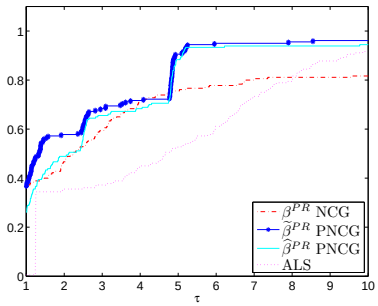
time. This suggests that the NCG algorithm without nonlinear preconditioning is not nearly as robust as the other algorithms. In Figure 1(b), R increases to 5. In this case, we also find that the NCG algorithm is not as robust as the other algorithms. The same is true when $I = 100$ and $R = 3$. This can be seen from Figure 1(c). In the case where $I = 100$, $R = 5$, the timing results indicated that the ALS algorithm was the fastest algorithm on average. However, Figure 1(d) shows that PNCG with the $\widetilde{\beta}^{PR}$variant is the fastest in the most runs. Both variants of the PNCG algorithm are more robust than the NCG algorithm, while ALS is the most robust in this case. In general, we can say that, while PNCG appears significantly faster than ALS for all difficult ($C = 0.9$) problems when the number of factors R and the tensor size I are relatively small, ALS becomes competitive again with PNCG when R and I are large. Note, however, that the line search parameters in the NCG and PNCG algorithms were the same for every problem, and it may be possible to improve both the NCG and PNCG results by fine-tuning these parameters. The main conclusion from our numerical tests is that nonlinear preconditioning can dramatically improve the speed and robustness of NCG: PNCG is significantly faster and more robust than NCG for all difficult ($C = 0.9$) CP problems we tested.

	Optimization Method	Mean Time (Seconds)		
		$I = 20$	$I = 50$	$I = 100$
$R = 3$	ALS	5.3182 ± 1.1356 **(180)**	5.1981 ± 0.3444 **(180)**	47.35 ± 4.30 **(180)**
	NCG - β^{PR}	3.5328 ± 2.7377 **(167)**	4.4516 ± 1.9664 **(179)**	94.98 ± 89.65 **(180)**
	PNCG - $\widetilde{\beta}^{PR}$	**0.9676 ± 0.2020 (180)**	**1.6320 ± 1.1064 (180)**	**28.23± 30.94 (180)**
	PNCG - $\widehat{\beta}^{PR}$	0.9979 ± 0.3077 **(180)**	1.6676 ± 0.7855 **(180)**	34.87 ± 46.97 **(180)**
$R = 5$	ALS	13.8499 ± 5.8256 **(106)**	10.4698 ± 3.0988 **(159)**	**57.11 ± 5.53 (180)**
	NCG - β^{PR}	7.0099 ± 4.3507 **(83)**	14.6827 ± 10.1787 **(142)**	124.54 ± 95.94 **(178)**
	PNCG - $\widetilde{\beta}^{PR}$	**2.7751 ± 1.9319 (109)**	**7.4386 ± 12.2583 (155)**	103.77 ± 257.10 **(178)**
	PNCG - $\widehat{\beta}^{PR}$	4.1549 ± 5.0031 **(108)**	10.4150 ± 25.0737 **(156)**	151.79 ± 356.29 **(180)**

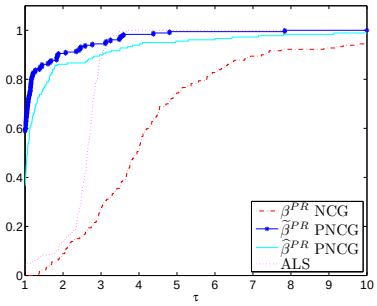
Table 2: Speed Comparison



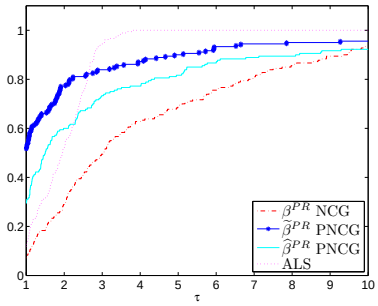
(a) $I = 20$, $R = 3$, Collinearity= 0.9



(b) $I = 20$, $R = 5$, Collinearity= 0.9



(c) $I = 100$, $R = 3$, Collinearity= 0.9



(d) $I = 100$, $R = 5$, Collinearity= 0.9

Figure 1: Performance profiles for the algorithms in $\mathcal{S}$ with $I = 20$ and $I = 100$.

5 Conclusion

We have proposed an algorithm for computing the canonical rank-R tensor decomposition that applies ALS as a nonlinear preconditioner to the NCG algorithm. We consider the ALS algorithm as a preconditioner because it is the standard algorithm used to compute the canonical rank-R tensor decomposition but it is known to converge very slowly for certain problems, for which acceleration by NCG is expected to be beneficial. We have considered several approaches for incorporating the nonlinear preconditioner into the NCG algorithm that have been described in the literature [15, 12, 16, 7, 10], corresponding to two different sets of preconditioned formulas for the standard FR, PR and HS update parameter, β, namely the $\widetilde{\beta}$ and $\widehat{\beta}$ formulas. If we use the $\widehat{\beta}$ formulas and apply the PNCG algorithm using a SPD preconditioner to a convex quadratic function using an exact line search, then the PNCG algorithm simplifies to the PCG algorithm. Also, we proved a new convergence result for one of the PNCG variants under suitable conditions, building on known convergence results for non-preconditioned NCG when line searches are used that satisfy the strong Wolfe conditions.

Following the methodology of [4] we create numerous test tensors and perform extensive numerical tests comparing the PNCG algorithm to the ALS and NCG algorithms. We consider a wide range of tensor sizes, ranks, factor collinearity and noise levels. Results in [5] showed that ALS is normally faster than NCG. In this paper, we show that NCG preconditioned with ALS (or, equivalently, ALS accelerated by NCG) is often significantly faster than ALS by itself, for difficult problems. When the collinearity is 0.9, the PNCG algorithm is often the fastest algorithm. The performance profiles of each algorithm also show that for the more difficult problems, PNCG is consistently both more robust and faster than the NCG algorithm. For our optimization problems, we generally obtain convergent results for all of the six variants of the PNCG algorithm we considered. It is interesting that for the PDE problems of [10], out of the $\widetilde{\beta}$ variants, only $\widetilde{\beta}^{PR}$ was found viable. It appears that the $\widehat{\beta}$ variants were not investigated in [10]. We did find for our test tensors that the $\widetilde{\beta}^{PR}$ formula, which does not reduce to PCG in the linear case, converges the fastest for most cases.

The PNCG algorithm discussed in this paper is formulated under a general framework. While this approach has met with success previously in certain application areas [15, 12, 16, 7, 10] and may offer promising avenues for further applications, it appears that the nonlinearly preconditioned NCG approach has received relatively little attention in the broader community and remains underexplored both theoretically and experimentally. It will be interesting to investigate the effectiveness of PNCG for other nonlinear optimization problems. Other nonlinear least-squares optimization problems for which ALS solvers are available are good initial candidates for further study. However, as with PCG for SPD linear systems [23], it is fully expected that devising effective preconditioners for more general nonlinear optimization problems will be highly problem-dependent while at the same time being crucial for gaining substantial performance benefits.

References

[1] Carroll JD, Chang JJ. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*. 1970; **35** : 283–319.

[2] Harshman RA. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*. 1970; **16** : 1–84.

[3] Kolda TG, Bader BW. Tensor decompositions and applications. *SIAM Review*. 2009; **51** : 455–500.

[4] Tomasi G, Bro R. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics and Data Analysis*. 2006; **50** : 1700–1734.

[5] Acar A, Dunlavy DM, Kolda TG. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*. 2011; **25** : 67–86.

[6] Fang H, Saad Y. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Application*. 2009; **16** : 197–221.

[7] Zibulevsky M, Elad M. L1-L2 optimization in signal and image processing. *IEEE Signal Processing Magazine*. 2010; **27** : 76–88.

[8] Walker H, Ni P. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*. 2011; **49** : 1715–1735.

[9] De Sterck H. A Nonlinear GMRES Optimization Algorithm for Canonical Tensor Decomposition. *SIAM Journal on Scientific Computing*. 2012; **34** : A1351–A1379.

[10] Brune P, Knepley MG, Smith BF, Tu X. Composing scalable nonlinear algebraic solvers. 2013. `http://www.mcs.anl.gov/papers/P2010-0112.pdf`

[11] Anderson D. Iterative procedures for nonlinear integral equations. *Journal of the Association for Computing Machinery*. 1965; **12** : 547–560.

[12] Concus P, Golub GH, O'Leary DP. Numerical solution of nonlinear elliptical partial differential equations by a generalized conjugate gradient method. *Computing*. 1977; **19** : 321–339.

[13] Pulay P. Convergence acceleration of iterative sequences: The case of SCF iteration. *Chemical Physics Letters*. 1980; **73** : 393–398.

[14] Oosterlee C, Washio T. Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows. *SIAM Journal on Scientific Computing*. 2000; **21** : 1670–1690.

[15] Bartels R, Daniel JW. A conjugate gradient approach to nonlinear elliptic boundary value problems in irregular regions. In: *Conference on the Numerical Solution of Differential Equations*, edited by Watson G, vol. 363 of *Lecture Notes in Mathematics*, pp. 1–11. Springer Berlin Heidelberg. 1974.

[16] Mittelmann HD. On the efficient solution of nonlinear finite element equations I. *Numerische Mathematik*. 1980; **35** : 277–291.

[17] Hager WW, Zhang H. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*. 2006; **2** : 35–58.

[18] Grasedyck L, Kressner D, Tobler C. A literature survey of low-rank tensor approximation techniques. 2013. `http://arxiv.org/abs/1302.7121`

[19] Fletcher R, Reeves CM. Function minimization by conjugate gradients. *Computer Journal*. 1964; **7** : 149–154.

[20] Polak E, Ribière G. Note sur la convergence de méthodes de directions conjugées. *Revue Françasie d'Informatique et de Recherche Opérationnelle*. 1969; **16** : 35–43.

[21] Hestenes MR, Stiefel E. Method of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*. 1952; **49** : 409–436.

[22] Nocedal J, Wright SJ. *Numerical Optimization*. New York: Springer, 2nd ed. 2006.

[23] Saad Y. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd ed. 2003.

[24] Dunlavy DM, Kolda TG, Acar E. Poblano v1.0: A MATLAB Toolbox for Gradient-Based Optimization. *Tech. Rep. SAND2010-1422*, Sandia National Laboratories, Albuquerque, NM and Livermore, CA. 2010.

[25] Bader BW, Kolda TG. MATLAB Tensor Toolbox Version 2.5. Available online. 2012. `http://www.sandia.gov/~tgkolda/TensorToolbox/`

[26] Dolan ED, Moré JJ. Benchmarking optimization software with performance profiles. *Math Program*. 2002; **91** : 201–213.