SIAM J. SCI. COMPUT.
Vol. 34, No. 3, pp. A1351–A1379

A NONLINEAR GMRES OPTIMIZATION ALGORITHM FOR CANONICAL TENSOR DECOMPOSITION*

H. DE STERCK†

Abstract. A new algorithm is presented for computing a canonical rank-R tensor approximation that has minimal distance to a given tensor in the Frobenius norm, where the canonical rank-R tensor consists of the sum of R rank-one tensors. Each iteration of the method consists of three steps. In the first step, a tentative new iterate is generated by a stand-alone one-step process, for which we use alternating least squares (ALS). In the second step, an accelerated iterate is generated by a nonlinear generalized minimal residual (GMRES) approach, recombining previous iterates in an optimal way, and essentially using the stand-alone one-step process as a preconditioner. In particular, the nonlinear extension of GMRES we use that was proposed by Washio and Oosterlee in [*Electron. Trans. Numer. Anal.*, 15 (2003), pp. 165–185] for nonlinear partial differential equation problems (which is itself related to other existing acceleration methods for nonlinear equation systems). In the third step, a line search is performed for globalization. The resulting nonlinear GMRES (N-GMRES) optimization algorithm is applied to dense and sparse tensor decomposition test problems. The numerical tests show that ALS accelerated by N-GMRES may significantly outperform stand-alone ALS when highly accurate stationary points are desired for difficult problems. Further comparison tests show that N-GMRES is competitive with the well-known nonlinear conjugate gradient method for the test problems considered and outperforms it in many cases. The proposed N-GMRES optimization algorithm is based on general concepts and may be applied to other nonlinear optimization problems.

Key words. canonical tensor decomposition, alternating least squares, GMRES, nonlinear optimization

AMS subject classifications. 15A69, 65F10, 65K10, 65F08

DOI. 10.1137/110835530

1. Introduction.

In this paper, we present a new algorithm for computing a canonical rank-R tensor approximation that has minimal distance to a given tensor in the Frobenius norm, where the canonical rank-R tensor consists of the sum of R rank-one tensors. N-way tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is an N-dimensional array of size $I_1 \times \cdots \times I_N$ [16]. The size of mode n is I_n $(n = 1, \ldots, N)$. Let $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ be a canonical rank-R tensor given by

$$\mathcal{A}_R = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \cdots \circ \mathbf{a}_r^{(N)} = [\![\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]. \tag{1.1}$$

Canonical tensor $\mathcal{A}_R$ is a sum of R rank-one tensors, with the rth rank-one tensor composed of the outer product of N column vectors $\mathbf{a}_r^{(n)} \in \mathbb{R}^{I_n}$, $n = 1, \ldots, N$. For each mode $n = 1, \ldots, N$, the R vectors $\mathbf{a}_r^{(n)}$, $r = 1, \ldots, R$, form the columns of the mode-n factor matrix $\mathbf{A}^{(n)}$, and the double-bracket notation on the right of (1.1) is used to denote the canonical tensor by the factor matrices.

This paper concerns numerical algorithms for the following optimization problem.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section May 26, 2011; accepted for publication (in revised form) April 2, 2012; published electronically May 24, 2012. This work was sponsored by the Natural Sciences and Engineering Research Council of Canada and by Lawrence Livermore National Laboratory under subcontract B594099.
http://www.siam.org/journals/sisc/34-3/83553.html
†Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario N2L 361, Canada (hdesterck@uwaterloo.ca).

Optimization Problem I. *Given tensor* $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$*, find rank-*$R$ *canonical tensor* $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ *that minimizes*

$$f(\mathcal{A}_R) = \frac{1}{2} \|\mathcal{T} - \mathcal{A}_R\|_F^2. \tag{1.2}$$

Here, $\|.\|_F$ denotes the Frobenius norm of the N-dimensional array, which is the square root of the sum of the squares of all the array elements.

We briefly recall some properties of canonical tensor decomposition, mainly referring to review article [16], which also contains extensive references to original papers that the reader may consult for more details. The exact decomposition of a data tensor $\mathcal{T}$ into a canonical tensor is often called a CP decomposition of the tensor, with the C standing for "CANDECOMP" and the "P" standing for "PARAFAC," after the names originally given to this decomposition in early papers on the subject [5, 14]. The smallest number of rank-one tensors that generate $\mathcal{T}$ as their sum is called the tensor rank of tensor $\mathcal{T}$ [16]. If, rather than an exact decomposition, $\mathcal{T}$ is *approximately* decomposed into a low-rank canonical tensor $\mathcal{A}_R$ of specified rank R that is smaller than the rank of $\mathcal{T}$, then the resulting tensor $\mathcal{A}_R$ is called an approximate CP decomposition or CP factorization (with the qualifier "approximate" often omitted). CP decompositions are used for data analysis in many applications such as chemometrics, signal processing, neuroscience, and web analysis. Many criteria exist for specifying the type of approximation that is sought for approximate CP decompositions (see, e.g., [26]). In this paper, we focus on the specific (and practically relevant) case of computing an approximate CP decomposition $\mathcal{A}_R$ that minimizes the Frobenius distance between the data tensor and $\mathcal{A}_R$; i.e., we seek to minimize objective function (1.2). Contrary to the case of best rank-R matrix approximation, the rank-one terms of the best rank-R CP tensor approximation cannot be solved for sequentially but must be found simultaneously, since a best rank-R CP approximation cannot be obtained by truncating a best rank-S approximation with $S > R$ [16].

It is well known that Optimization Problem I is a nonconvex optimization problem and as such may exhibit multiple local minima. Even when scaling and permutation indeterminacies are removed, CP optimization may still exhibit multiple local minima for some problems, and depending on the initial guess, iterative methods for approximate CP decomposition may converge to different stationary points. It is also known that some tensors can fail to have best rank-R approximations [16, 3]. On the other hand, exact CP decomposition has been shown to be unique up to scaling and permutation under mild conditions that relate the ranks of the factor matrices with the tensor rank [16].

A multitude of algorithms and approaches exist for computing approximate CP decompositions; see, for example, [26, 16, 2] and references therein. A standard numerical method for computing the CP approximation of Optimization Problem I is the alternating least squares (ALS) method, which was already proposed in early papers on CP decomposition [5, 14]. ALS is simple to understand and implement and often performs adequately, but its convergence can also be very slow, depending on the problem and the initial condition. Alternatives to ALS are described in, for example, [26, 6, 7]; see also the discussion in [16, 2]. Even though ALS is a simple algorithm, it has proven difficult over the years to develop alternatives that significantly improve on it in a way that is robust over large classes of problems. As a result, ALS-type algorithms are still often considered the "workhorse" algorithms of choice in practice.

In recent work on CP decomposition, Acar, Dunlavy, and Kolda [2] consider first-order optimization algorithms for Optimization Problem I. In particular, they

employ the nonlinear conjugate gradient (N-CG) method and compare it with ALS and nonlinear least squares algorithms. In order to formulate first-order optimization methods, they derive expressions for the gradient of objective function (1.2) in a systematic way. At any (local) minimum of Optimization Problem I the following conditions hold.

First-order optimality equation I.

$$\nabla f(\mathcal{A}_R) = \mathbf{g}(\mathcal{A}_R) = 0. \tag{1.3}$$

Detailed expressions for the gradient vector $\mathbf{g}(\mathcal{A}_R)$ will be given below. Acar, Dunlavy, and Kolda [2] then apply a standard N-CG optimization method with line search and obtain convergence speeds that are competitive with ALS and sometimes better, depending on the problem.

In this paper we also follow a first-order optimization approach for Optimization Problem I, but we propose using a different optimization method. In particular, we propose using a nonlinear GMRES approach to recombine iterates provided by a standard iterative method for Optimization Problem I (we use ALS as the GMRES preconditioner in this paper), combined with line search for globalization. The resulting N-GMRES optimization algorithm will be shown numerically to accelerate ALS convergence significantly in many cases, especially when stationary points need to be determined accurately for difficult problems, and we show this for two classes of dense and sparse test problems. We also compare our algorithm to N-CG and to ALS enhanced with line search [26, 22]. The N-GMRES optimization algorithm proposed is easy to implement as a wrapper around any iterative solution method for Optimization Problem I. It is based on general concepts and may be applied to different nonlinear optimization problems as a potential alternative to other first-order optimization methods like N-CG.

The nonlinear GMRES acceleration method we propose using in our algorithm for nonlinear optimization is the nonlinear extension of GMRES that was developed by Washio and Oosterlee in [27] to accelerate multigrid solvers for nonlinear PDE systems (see also [20, 19] for further applications of their method in the nonlinear PDE systems context). Their method is a generalization of Saad and Schultz's celebrated GMRES method for linear equation systems [24] to the nonlinear case. In the N-GMRES optimization algorithm we propose, we apply this nonlinear GMRES approach to combine ALS-generated iterates with the goal of driving the residual of the (nonlinear) first-order optimality equations to zero. A line search method is used to provide globalization. As with GMRES for linear equation systems, the method that generates the iterates can be seen as a preconditioner for GMRES, or, from an alternative viewpoint, GMRES can be interpreted as an acceleration mechanism for the method that generates the iterates [27, 20]. It will be explained in detail below how this interpretation carries over to the present context of nonlinear optimization for CP decomposition. Note that early descriptions of nonlinear iterate acceleration ideas similar to the method used in [27] appear in so-called Anderson mixing [1] and Pulay mixing [21] for solving nonlinear equation systems, and the relationship of this type of method with GMRES is also further elucidated in [12, 11, 28, 23], along with more recent applications of these ideas to nonlinear equation systems and fixed-point problems. This type of nonlinear iterate acceleration has thus been considered several times before in the context of solving nonlinear systems of equations [1, 21, 27, 20, 12, 11, 28, 23]. However, its combination in this paper with a line search and an advanced nonlinear preconditioner leads to a tensor decomposition optimization algorithm that is new. Moreover, the resulting preconditioned nonlinear

optimization approach, which can be applied to optimization problems other than tensor decomposition, is also new in the optimization context, to the best of our knowledge.

Figures 1.1–1.3 give a preview of what the N-GMRES optimization algorithm proposed in this paper tries to achieve. The figures present convergence plots for a dense test problem that is a standard test case for CP decomposition from the literature [26, 2]. A three-way data tensor with size $50 \times 50 \times 50$ is generated starting from a rank-three canonical tensor with random factor matrices modified to have pre-specified collinearity c, and noise is added. (See Test Problem I in section 3 for a detailed description of the test case.) A rank-three CP approximation is sought

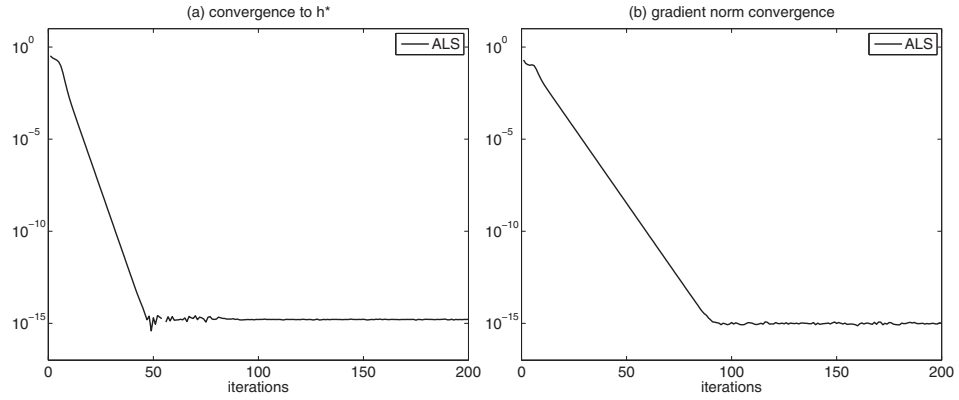

FIG. 1.1. *ALS convergence plots for an "easy" instance of Test Problem* I *(parameters* $s = 50$, $c = 0.5$, $R = 3$, $l_1 = 1$, $l_2 = 1$*; see section* 3 *for full problem description).* (a) *Convergence of* $|h(\mathcal{A}_R^{(i)}) - h^*|$*, where* h^* *is the value of the normalized distance measure,* (1.4)*, in the stationary point the method converges to.* (b) *Convergence of the normalized gradient of the objective function,* $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$*, indicating convergence to a stationary point. ALS is a fast method for this "easy" problem, and N-GMRES acceleration is not required.*

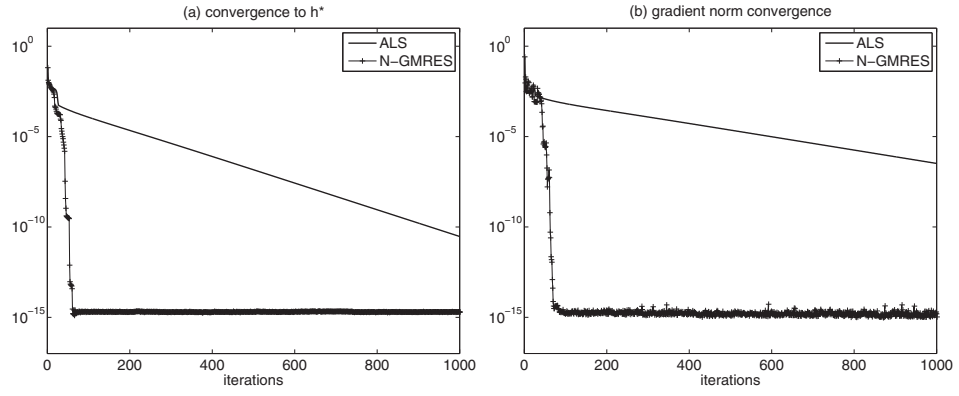

FIG. 1.2. *ALS convergence plots for a "difficult" instance of Test Problem* I *(parameters* $s = 50$, $c = 0.9$, $R = 3$, $l_1 = 1$, $l_2 = 1$*).* (a) *Convergence of* $|h(\mathcal{A}_R^{(i)}) - h^*|$*.* (b) *Convergence of* $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$*. ALS is slow for this "difficult" problem, but our proposed N-GMRES optimization algorithm significantly reduces the number of iterations.*

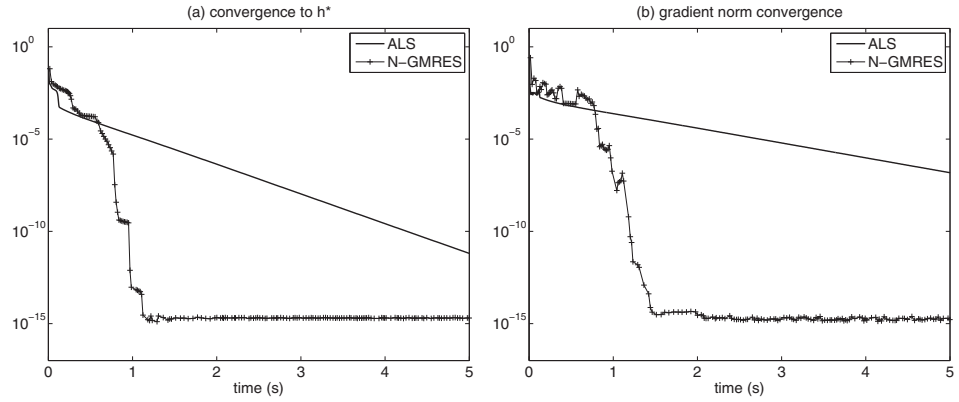

FIG. 1.3. *Same as Figure* 1.2*, but now convergence is plotted as a function of execution time. Even though N-GMRES iterations take more time per iteration, N-GMRES still substantially accelerates ALS, especially if accurate approximations are desired.*

starting from a random initial guess. In order to track accuracy during the progress of the iterative methods, we define a measure of the normalized distance between data tensor $\mathcal{T}$ and rank-R approximation $\mathcal{A}_R^{(i)}$ in iteration i as

$$h(\mathcal{A}_R^{(i)}) = \frac{\|\mathcal{T} - \mathcal{A}_R^{(i)}\|_F}{\|\mathcal{T}\|_F} \tag{1.4}$$

and define the optimal distance

$$h^* = h(\mathcal{A}_R^*), \tag{1.5}$$

where $\mathcal{A}_R^*$ is the stationary point that the method converges to in the test. The accuracy of the approximation as the iterative method progresses is then tracked by measuring $|h(\mathcal{A}_R^{(i)}) - h^*|$. We also track convergence of $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$, the norm of the gradient of the objective function normalized by the norm of the data tensor, which gives information about convergence to a stationary point.

Figure 1.1 shows convergence plots for a case with collinearity $c = 0.5$. It is known that this case is "easy" for ALS, and the plots confirm that ALS converges quickly to a stationary point. It is also known that factor matrices with nearly collinear columns constitute problems that are more difficult for ALS [26, 2]. Figure 1.2 shows convergence plots for a case with $c = 0.9$, and we can indeed observe that ALS converges slowly. The plots also show how the N-GMRES optimization algorithm that is proposed in this paper significantly speeds up ALS and reduces the number of iterations dramatically. Of course, Figure 1.2 is only part of the story, because the N-GMRES iterations are more expensive than simple ALS iterations. However, the timing plots in Figure 1.3 show that significant gains can still be made if this extra cost is taken into account, especially when accurate results are desired.

The rest of this paper is organized as follows. In section 2 we describe the proposed N-GMRES optimization algorithm for nonlinear optimization problems and explain how it is applied to compute rank-R canonical tensor decompositions. We also explain the interpretation of the N-GMRES acceleration method as a preconditioned GMRES nonlinear optimization method. In section 3 we test the N-GMRES optimization algorithm on dense tensor test cases, and in section 4 we discuss sparse test problems. Conclusions are formulated in section 5.

2. N-GMRES optimization algorithm for canonical tensor decomposition. In this section, we describe the proposed N-GMRES optimization algorithm for nonlinear optimization problems and explain how it is applied to compute rank-R canonical tensor decompositions. We start with a description of the N-GMRES optimization approach and situate this discussion in the context of a general nonlinear optimization problem, since the approach is general. We then compare the N-GMRES approach for optimization to GMRES for linear systems, explaining how the dual viewpoint of preconditioned GMRES and GMRES acceleration applies in the nonlinear optimization context. In our descriptions we closely follow the derivations and presentation of Washio and Oosterlee's nonlinear GMRES for PDE problems from [27, 20], which we propose using as the main ingredient in our nonlinear optimization algorithm. We give extensive details because these details give precise insight into how the GMRES method generalizes to the nonlinear optimization problem. This is followed by a discussion of how N-GMRES is applied to CP optimization, giving details on the first-order optimality equations for CP and the line search algorithm we use in the numerical results of subsequent sections.

2.1. N-GMRES optimization algorithm. Consider the following nonlinear optimization problem with associated first-order optimality equations.

Optimization Problem II.

$$\textit{Find } \mathbf{u}^* \textit{ that minimizes } f(\mathbf{u}). \tag{2.1}$$

First-order optimality equation II.

$$\nabla f(\mathbf{u}) = \mathbf{g}(\mathbf{u}) = 0. \tag{2.2}$$

Assume that we have $i+1$ previous iterates $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{i-1}, \mathbf{u}_i$ and that we have a one-step (nonlinear) iterative update process $M(\cdot)$ that generates a new approximation from an existing approximation:

$$\mathbf{u}_{new} = M(\mathbf{u}_{old}). \tag{2.3}$$

The goal of the N-GMRES optimization algorithm will be to accelerate the convergence of iterative update process $M(\cdot)$.

Every iteration of the N-GMRES optimization algorithm consists of three steps.

Step I. In the first step, we generate a preliminary new iterate $\bar{\mathbf{u}}_{i+1}$ from the last iterate $\mathbf{u}_i$ using one-step iterative update process $M(\cdot)$:

$$\bar{\mathbf{u}}_{i+1} = M(\mathbf{u}_i). \tag{2.4}$$

Step II. In the second step, we find an accelerated iterate $\hat{\mathbf{u}}_{i+1}$ that is obtained by recombining previous iterates as follows:

$$\hat{\mathbf{u}}_{i+1} = \bar{\mathbf{u}}_{i+1} + \sum_{j=0}^{i} \alpha_j \left(\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j\right). \tag{2.5}$$

The N-GMRES algorithm seeks to determine the unknown coefficients α_j such that the two-norm of the gradient of the objective function evaluated at the accelerated iterate is small. We call $\mathbf{g}(\hat{\mathbf{u}}_{i+1})$ the residual at $\hat{\mathbf{u}}_{i+1}$, and the objective is thus to determine the α_j such that the residual is minimized in the two-norm: we attempt to minimize $\|\mathbf{g}(\hat{\mathbf{u}}_{i+1})\|_2$. However, $\mathbf{g}(.)$ is a nonlinear function of the α_j (more

precisely, it is a high-order polynomial in the α_j), and we proceed by linearization to allow for inexpensive computation of coefficients α_j that may approximately minimize $\|\mathbf{g}(\hat{\mathbf{u}}_{i+1})\|_2$. Using the approximations

$$\begin{aligned}\mathbf{g}(\hat{\mathbf{u}}_{i+1}) &\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \left.\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right|_{\bar{\mathbf{u}}_{i+1}} \alpha_j \left(\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j\right) \\ &\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \alpha_j \left(\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j)\right), \end{aligned} \tag{2.6}$$

we arrive at minimization problem

$$\begin{gathered}\text{find coefficients } (\alpha_0, \ldots, \alpha_i) \text{ that minimize} \\ \left\| \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \alpha_j \left(\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j)\right) \right\|_2 .\end{gathered}$$

This is a standard least squares problem that can be solved, for example, by using the normal equations. Defining

$$\begin{aligned}\boldsymbol{\alpha} &= (\alpha_0, \ldots, \alpha_i)^T, \\ \mathbf{p}_j &= \mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j), \\ \mathbf{P} &= [\mathbf{p}_0 | \ldots | \mathbf{p}_j] ,\end{aligned}$$

we minimize $\|\mathbf{P}\,\boldsymbol{\alpha} + \mathbf{g}(\bar{\mathbf{u}}_{i+1})\|_2$, which leads to normal equation system

$$\mathbf{P}^T \mathbf{P}\, \boldsymbol{\alpha} = -\mathbf{P}^T\, \mathbf{g}(\bar{\mathbf{u}}_{i+1}). \tag{2.7}$$

Step III. In the third step, we perform a line search that optimizes objective function $\mathbf{f}(\mathbf{u})$ on a half line starting at preliminary iterate $\bar{\mathbf{u}}_{i+1}$, which was generated in Step I, and connecting it with accelerated iterate $\hat{\mathbf{u}}_{i+1}$, which was generated in Step II:

$$\begin{gathered}\text{find } \beta^* \in [0, \infty) \text{ that minimizes} \\ \mathbf{f}(\bar{\mathbf{u}}_{i+1} + \beta(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1})).\end{gathered} \tag{2.8}$$

This line search step is necessary, because without it erratic convergence behavior can occur, especially as long as iterates are far from a stationary point (see [18]). In this case, the linearization steps in (2.6) may incur large errors, resulting in accelerated iterates $\hat{\mathbf{u}}_{i+1}$ that may be bad approximations. The result of the line search is finally selected as the new iterate $\mathbf{u}_{i+1}$:

$$\mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1} + \beta^*(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}). \tag{2.9}$$

In practice, to limit the computational cost and especially the memory cost of the N-GMRES acceleration, we implement the algorithm with a windowed acceleration process with window size w, in which N-GMRES-accelerated iterates are generated based on the w last iterations.

Figure 2.1 gives a schematic representation of the N-GMRES optimization algorithm, and Algorithm 1 describes it in pseudocode. (Note of course that the w

initial iterates required in the pseudocode description can naturally be generated by applying the algorithm with a window size that gradually increases from one up to w, starting from a single initial guess.)

Algorithm 1: N-GMRES optimization algorithm (window size w)

Input: w initial iterates $\mathbf{u}_0, \ldots, \mathbf{u}_{w-1}$.

$i = w - 1$
repeat
  STEP I: *(generate preliminary iterate by one-step update process $M(\cdot)$)*
    $\bar{\mathbf{u}}_{i+1} = M(\mathbf{u}_i)$
  STEP II: *(generate accelerated iterate by nonlinear GMRES step)*
    $\hat{\mathbf{u}}_{i+1} = gmres(\mathbf{u}_{i-w+1}, \ldots, \mathbf{u}_i; \bar{\mathbf{u}}_{i+1})$
  STEP III: *(generate new iterate by line search process)*
    $\mathbf{u}_{i+1} = linesearch(\bar{\mathbf{u}}_{i+1} + \beta(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}))$
  $i = i + 1$
until *convergence criterion is satisfied*

We now make some additional remarks about the N-GMRES optimization algorithm proposed and its application to canonical tensor decomposition.

First, for Step I in the algorithm, we use the ALS algorithm as the one-step update process $M(\cdot)$. The ALS algorithm for CP decomposition will be described in section 2.3, along with specific expressions for the gradient of the objective function and the first-order optimality equations of Optimization Problem I. The one-step update process $M(\cdot)$ can be interpreted as the preconditioner of the nonlinear GMRES procedure, as will be explained in section 2.2.

Second, Steps I and II in the N-GMRES optimization algorithm are entirely analogous to the nonlinear GMRES approach for PDE systems that was proposed by Washio and Oosterlee in [27]. In their applications, they employ nonlinear GMRES to drive the residuals of the PDEs to zero and use multigrid as the preconditioner [27, 20, 19]. In the present context of canonical tensor decomposition as a nonlinear optimization problem, we drive the residual of the first-order optimality equations (the gradient of the objective function) to zero and use ALS as the preconditioner.

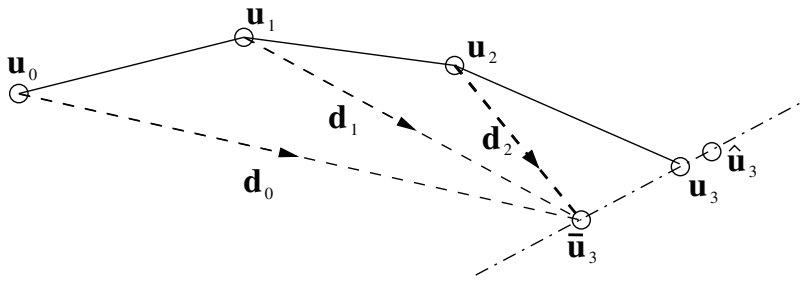

FIG. 2.1. *Schematic representation of one iteration of the N-GMRES optimization algorithm. Given previous iterations $\mathbf{u}_0$, $\mathbf{u}_1$, and $\mathbf{u}_2$, new iterate $\mathbf{u}_3$ is generated as follows. In Step* I, *preliminary iterate $\bar{\mathbf{u}}_3$ is generated by the one-step update process $M(\cdot)$: $\bar{\mathbf{u}}_3 = M(\mathbf{u}_2)$. In Step* II, *the nonlinear GMRES step, accelerated iterate $\hat{\mathbf{u}}_3$ is obtained by determining the coefficients α_j in $\hat{\mathbf{u}}_3 = \bar{\mathbf{u}}_3 + \alpha_0\mathbf{d}_0 + \alpha_1\mathbf{d}_1 + \alpha_2\mathbf{d}_2$ such that the gradient of the objective function in $\hat{\mathbf{u}}_3$ is approximately minimized. In Step* III, *the new iterate, $\mathbf{u}_3$, is finally generated by a line search that minimizes the objective function $\mathbf{f}(\bar{\mathbf{u}}_3 + \beta(\hat{\mathbf{u}}_3 - \bar{\mathbf{u}}_3))$.*

Third, Step III in the N-GMRES optimization algorithm performs a role analogous to the selection mechanism proposed by Washio and Oosterlee in [27] to reduce erratic convergence. They select either the preliminary iterate or the accelerated one, based on ingenious but somewhat ad hoc criteria that consider changes in the norms of the residuals and in the solution updates. Instead, in the present context of nonlinear optimization problems, we can exploit the final goal of minimizing the objective function to control erratic behavior, and rather than selecting either the preliminary iterate or the accelerated one, we perform a line search on the line connecting the two. Not only does this control erratic behavior, but it also provides an improved new iterate, at the cost of a line search.

Fourth, it may be beneficial to restart the N-GMRES acceleration with window size one when indications arise that the current window contains iterates that harm the acceleration process. In [27] the GMRES procedure is restarted based on criteria similar to those used to choose between the preliminary or accelerated iterate. In the context of N-GMRES for nonlinear optimization problems, we propose the following simple criterion: we restart whenever the search direction of the line search, $\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}$, does not point in a descent direction (so we set the window size back to one whenever $\mathbf{g}(\bar{\mathbf{u}}_{i+1})^T (\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}) > 0$). The motivation for this is simple: we expect the acceleration mechanism to be effective close to a stationary point. In that case, the accelerated iterate is expected to be an improvement over the preliminary iterate, and the search direction is expected to be a descent direction. If this is not the case, this is taken as an indication that the current N-GMRES sequence does not help, and N-GMRES is restarted. Our numerical tests have indicated that this simple approach indeed tends to be beneficial for speed of convergence, for the test problems we considered. Note that windowing with restarts can be implemented efficiently in an elegant way, and we refer the reader to the detailed pseudocode in [27] for this.

Fifth, another practical point is that the normal equation system in (2.7) may become ill conditioned since the vectors $\mathbf{p}_j$ may become nearly linearly dependent. One way to deal with this, as suggested in [27], is to add a small multiple of the identity matrix, $\delta \mathbf{I}$, to the normal equation operator. This was sufficient for the numerical tests we performed, and [27] provides further theoretical justification for this fix. As an alternative, one can monitor the singular values of P and restart when the condition number of P exceeds a large value (e.g., 10^6).

Sixth, for the line search of Step III, we use in our numerical experiments the Moré–Thuente line search method [17] as implemented in the Poblano toolbox for MATLAB [10]. This line search method was also used in the N-CG method for canonical tensor decomposition in [2]. This is a rather sophisticated line search method that employs multiple function and gradient evaluations to improve the solution. If the search direction is not a descent direction, no search is performed and the original point is kept.

Finally, we briefly discuss the computational cost of the N-GMRES optimization algorithm. In terms of memory cost, for window size w the algorithm requires storing w previous solution vectors and residual vectors. In the numerical results presented below we will see that, for the CP decomposition test problems we consider, a window size of approximately $w = 20$ is optimal in terms of computing time. This requires substantial additional memory, however, and memory can obviously be traded for speed. In terms of the computing time, the dominant costs in each iteration of N-GMRES nonlinear optimization algorithm 1 are applying $M(\cdot)$ in Step I, computing the gradient of the preliminary iterate, $\mathbf{g}(\bar{\mathbf{u}}_{i+1})$, in Step II, and the function and gra-

dient evaluations in the line search of Step III. Since the optimization problems we solve in this paper are quite involved and the ALS, function, and gradient evaluations are very expensive, these calculations strongly dominate all others in practice. The relevant extra costs of the N-GMRES optimization approach compared to just applying the one-step update process $M(\cdot)$ are thus the additional function and gradient evaluations, and the cost of building and solving the normal equation system (2.7) and computing the accelerated iterate is negligible in practice. The number of function and gradient evaluations per line search depend on the line search method used and the accuracy parameters one chooses for the line search calculations, which in turn influence the number of overall iterations required to achieve a certain accuracy for the stationary point. It is thus difficult to say much in general about the additional cost of the line search step. The numerical results to be presented below show that, for two relevant classes of test problems for canonical tensor decomposition, N-GMRES optimization can be significantly faster than stand-alone iteration by applying $M(\cdot)$, especially when high accuracy is desired. We will at this point just give one example. For the N-GMRES CP calculation shown in Figure 1.3, up to an accuracy of $|h - h^*| < 10^{-10}$ (which required 54 iterations), about 40% of the time is spent in the ALS iterations (54 calls), 15% in the calculation of $\mathbf{g}(\bar{\mathbf{u}}_{i+1})$ (54 calls), and about 30% in the function and gradient evaluations in the line search procedure (137 calls to evaluate $\mathbf{f}$ and $\mathbf{g}$). The average number of function/gradient evaluations per line search was about 2.5. While the generality of this single example is of course limited, it does illustrate that N-GMRES optimization algorithm 1 mainly adds overhead in terms of additional function and residual evaluations for the GMRES and the line search steps. But it is precisely these additional calculations that potentially lead to a great reduction in the number of iterations and overall execution time, as we will illustrate with extensive numerical tests below.

2.2. Interpretation as a GMRES method: Acceleration and preconditioning. In this subsection we briefly recall how the preconditioned GMRES method for linear equation system $\mathbf{A}\,\mathbf{u} = \mathbf{b}$ can be interpreted as an acceleration mechanism for a stationary iterative method and interpret N-GMRES as a GMRES method highlighting the role of preconditioning, following the presentation of [27] and referring to it extensively. Note that this is not the standard way to present GMRES.

Consider stationary iterative method

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{M}^{-1}\,\mathbf{r}_i \tag{2.10}$$

for $\mathbf{A}\,\mathbf{u} = \mathbf{b}$. Here, $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\,\mathbf{u}_i$ is the ith residual, and matrix $\mathbf{M}$ is an approximation of $\mathbf{A}$ that has an easily computable inverse, i.e., $\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$. For example, $\mathbf{M}$ can be chosen to correspond to Gauss–Seidel iteration. Consider a sequence of iterates $\mathbf{u}_0, \ldots, \mathbf{u}_i$ generated by update formula (2.10), starting from some initial guess $\mathbf{u}_0$. Note that the residuals of these iterates are related as $\mathbf{r}_i = \mathbf{b} - \mathbf{A}\,\mathbf{u}_i = (\mathbf{I} - \mathbf{A}\mathbf{M}^{-1})\,\mathbf{r}_{i-1} = (\mathbf{I} - \mathbf{A}\mathbf{M}^{-1})^i\,\mathbf{r}_0$. This motivates the definition of the following vector spaces:

$$\begin{aligned}
V_{1,i+1} &= span\{\mathbf{r}_0, \ldots, \mathbf{r}_i\},\\
V_{2,i+1} &= span\{\mathbf{r}_0, \mathbf{A}\mathbf{M}^{-1}\,\mathbf{r}_0, (\mathbf{A}\mathbf{M}^{-1})^2\,\mathbf{r}_0\}, \ldots, (\mathbf{A}\mathbf{M}^{-1})^i\,\mathbf{r}_0\}\\
&= K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0), \qquad (2.11)\\
V_{3,i+1} &= span\{\mathbf{M}\,(\mathbf{u}_1 - \mathbf{u}_0), \mathbf{M}\,(\mathbf{u}_2 - \mathbf{u}_1), \ldots, \mathbf{M}\,(\mathbf{u}_{i+1} - \mathbf{u}_i)\},\\
V_{4,i+1} &= span\{\mathbf{M}\,(\mathbf{u}_{i+1} - \mathbf{u}_0), \mathbf{M}\,(\mathbf{u}_{i+1} - \mathbf{u}_1), \ldots, \mathbf{M}\,(\mathbf{u}_{i+1} - \mathbf{u}_i)\}. \qquad (2.12)
\end{aligned}$$

Vector space $V_{2,i+1}$ is the so-called Krylov space $K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$ of order $i+1$, generated by matrix $\mathbf{A}\mathbf{M}^{-1}$ and vector $\mathbf{r}_0$. It is easy to see that the vector spaces defined above are equal: $V_{1,i+1} = V_{2,i+1} = V_{3,i+1} = V_{4,i+1}$ (see [27] for a short proof). We know that $\mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_i) \in K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$. The GMRES procedure can be seen as a way to accelerate stationary iterative method (2.10) by recombining iterates (or, equivalently, by reusing residuals). In particular, we seek a better approximation $\hat{\mathbf{u}}_{i+1}$, with $\mathbf{M}(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i)$ in the Krylov space $K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$, such that $\hat{\mathbf{r}}_{i+1} = \mathbf{b} - \mathbf{A}\,\hat{\mathbf{u}}_{i+1}$ has minimal two-norm. In other words, we seek coefficients β_j in $\mathbf{M}(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i) = \sum_{j=0}^{i} \beta_j \mathbf{M}(\mathbf{u}_{i+1} - \mathbf{u}_j)$, or, equivalently, coefficients α_j in

$$\hat{\mathbf{u}}_{i+1} = \mathbf{u}_{i+1} + \sum_{j=0}^{i} \alpha_j (\mathbf{u}_{i+1} - \mathbf{u}_j), \tag{2.13}$$

such that $\|\hat{\mathbf{r}}_{i+1}\|_2$ is minimized (which leads to a small least squares problem).

We now explain two complementary viewpoints of GMRES for linear systems. We have presented the method as a way to accelerate simple one-step stationary iterative method (2.10). A second, more customary, viewpoint is to see GMRES in terms of preconditioning. With $\mathbf{M} = \mathbf{I}$, the approach described above reduces to nonpreconditioned GMRES. Preconditioned GMRES can then also be derived by applying nonpreconditioned GMRES (as described above, but with $\mathbf{M} = \mathbf{I}$) to the preconditioned linear equation system $\mathbf{A}\mathbf{M}^{-1}(\mathbf{M}\mathbf{u}) = \mathbf{b}$. In this viewpoint, the matrix $\mathbf{M}^{-1}$ is called the preconditioner matrix, because its role is viewed as preconditioning the spectrum of the linear system operator such that the (nonpreconditioned) GMRES method applied to $(\mathbf{A}\mathbf{M}^{-1})\mathbf{y} = \mathbf{b}$ becomes more effective. By extension, it is also customary (in the first viewpoint) to say that the stationary iterative method preconditions GMRES (in the sense of, for example, Gauss–Seidel or multigrid being used as preconditioners for GMRES). In this viewpoint, the role of the stationary iterative method is to generate a preconditioned residual that builds the Krylov space. Note also that in the presentation above, all iterates $\mathbf{u}_j$ for $j = 0, \ldots, i$ (for instance, in the right-hand side of (2.13)) refer to iterates generated by stationary iterative method (2.10), without acceleration. However, the formulas remain valid when we use accelerated iterates instead; this merely changes the values of the coefficients α_j but leads to the same accelerated iterates [27]. The reason is that all the GMRES optimization does is produce improved iterates with residuals that are optimal elements of the Krylov spaces, but the Krylov spaces are still those generated by the residuals of the stationary iterative method, and themselves do not change, due to linearity.

The N-GMRES optimization algorithm presented in subsection 2.1 can now be interpreted as a preconditioned GMRES method as follows. With $\mathbf{u}_{i+1}$ the preliminary $i+1$st iterate generated by (2.10) and the $\mathbf{u}_j$ with $j \leq i$ the previous accelerated iterates, the expression for the accelerated iterate $\hat{\mathbf{u}}_{i+1}$ we seek in (2.13) for the linear case corresponds directly to the expression in (2.5) for the nonlinear case. The definition of $V_{2,i+1}$ in (2.11) does not provide a nonlinear generalization for the Krylov space in which we seek an optimal approximation in the linear case, but the definition of $V_{4,i+1}$ (2.12) does: the image of the linear Krylov space $K_{i+1}(\mathbf{A}\mathbf{M}^{-1}, \mathbf{r}_0)$ under the preconditioning matrix $\mathbf{M}^{-1}$ is $span\{(\mathbf{u}_{i+1} - \mathbf{u}_0), (\mathbf{u}_{i+1} - \mathbf{u}_1), \ldots, (\mathbf{u}_{i+1} - \mathbf{u}_i)\}$, and this can trivially be generalized to the nonlinear case: the nonlinear generalization of the Krylov space is precisely given by this vector space. And it is precisely in this space that we seek an optimal vector, both in the linear and nonlinear cases (see (2.13) and

(2.5), respectively). Stationary iterative method (2.10) is the preconditioning process for GMRES in the linear case, responsible for generating a preliminary approximation with the help of a residual calculation and a preconditioning matrix, and in the same way one-step approximation method (2.4) is the preconditioner for GMRES in the nonlinear case. In the present case of the N-GMRES optimization algorithm applied to canonical tensor decomposition, we can say that ALS is used as the preconditioner of N-GMRES. The alternative viewpoint is that N-GMRES accelerates ALS.

This reasoning (as in [27]) shows that the nonlinear acceleration mechanism used in N-GMRES is mathematically "essentially equivalent" to GMRES in the linear case (using the terminology of [28]; see also [27, 20, 12, 11]) in the sense that it minimizes the two-norm of the residual over the Krylov space, just like GMRES does. For this reason, we call our nonlinear optimization approach N-GMRES. Note, however, that our nonlinear version does not share with the GMRES algorithm an efficient use of the Arnoldi iteration to generate orthogonal bases for Krylov spaces of increasing order [24], which sets GMRES apart from other methods for solving $\mathbf{A}\,\mathbf{u} = \mathbf{b}$ that are based on two-norm minimization of the residual. It thus does not have this important property of GMRES. For nonlinear problems, such an advantageous implementation is not possible, but the general ideas of optimal acceleration that are embodied by GMRES remain powerful and can still be applied, as we do in the N-GMRES optimization method.

2.3. First-order optimality equations for the CP optimization problem. In this subsection, we briefly recall the first-order optimality equations for CP Optimization Problem I, following [2]. The gradient of objective function (1.2) can be written as a vector of matrices

$$\nabla f(\mathcal{A}_R) = \vec{\mathbf{G}}(\mathcal{A}_R) = (\mathbf{G}^{(1)}, \ldots, \mathbf{G}^{(N)}), \tag{2.14}$$

with $\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times R}$ $(n = 1, \ldots, N)$. The matrices $\mathbf{G}^{(n)}$ for $n = 1, \ldots, N$ are given by

$$\mathbf{G}^{(n)} = -\mathbf{T}_{(n)}\bar{\mathbf{\Phi}}^{(n)} + \mathbf{A}^{(n)}\,\bar{\mathbf{\Gamma}}^{(n)}, \tag{2.15}$$

with variables defined as follows.

With $\mathcal{T}$ and $\mathcal{A}_R \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, define size parameters

$$K = \prod_{l=1}^{N} I_l, \tag{2.16}$$

$$\bar{K}^{(n)} = \prod_{\substack{l=1 \\ l \neq n}}^{N} I_l. \tag{2.17}$$

Then $\mathbf{T}_{(n)} \in \mathbb{R}^{I_n \times \bar{K}^{(n)}}$ is the mode-n matricization of $\mathcal{T}$, obtained by stacking the n-mode fibers of $\mathcal{T}$ in its columns in a regular way; see [16]. Similar to size parameters K and $\bar{K}^{(n)}$, we define matrices $\mathbf{\Phi} \in \mathbb{R}^{K \times R}$ and $\bar{\mathbf{\Phi}}^{(n)} \in \mathbb{R}^{\bar{K}^{(n)} \times R}$ by

$$\mathbf{\Phi} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(1)}, \tag{2.18}$$

$$\bar{\mathbf{\Phi}}^{(n)} = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}, \tag{2.19}$$

where $\odot$ is the Khatri–Rao product [16]. Finally, the matrices $\mathbf{\Gamma} \in \mathbb{R}^{R\times R}$ and $\bar{\mathbf{\Gamma}}^{(n)} \in \mathbb{R}^{R\times R}$ are defined by

$$\mathbf{\Gamma} = (\mathbf{A}^{(1)T}\mathbf{A}^{(1)}) * \cdots * (\mathbf{A}^{(N)T}\mathbf{A}^{(N)}), \tag{2.20}$$

$$\bar{\mathbf{\Gamma}}^{(n)} = (\mathbf{A}^{(1)T}\mathbf{A}^{(1)}) * \cdots * (\mathbf{A}^{(n-1)T}\mathbf{A}^{(n-1)}) * (\mathbf{A}^{(n+1)T}\mathbf{A}^{(n+1)}) * \cdots * (\mathbf{A}^{(N)T}\mathbf{A}^{(N)}), \tag{2.21}$$

where $*$ means elementwise multiplication.

The first-order optimality equations are then given by

$$\mathbf{G}^{(n)} = 0 = -\mathbf{T}_{(n)}\bar{\mathbf{\Phi}}^{(n)} + \mathbf{A}^{(n)}\,\bar{\mathbf{\Gamma}}^{(n)}, \qquad n = 1, \ldots, N. \tag{2.22}$$

They offer an easy way to describe the ALS method for CP optimization: an ALS iteration proceeds by sequentially solving for each of the factor matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$ using the corresponding optimality equation $\mathbf{G}^{(n)} = 0$, updating the matrices $\bar{\mathbf{\Phi}}^{(n)}$ and $\bar{\mathbf{\Gamma}}^{(n)}$ along the way. As such, it is an example of a nonlinear block Gauss–Seidel method to solve the nonlinear equation system provided by the first-order optimality equations. The MATLAB Tensor Toolbox [4] implements efficient methods for computing the product $\mathbf{T}_{(n)}\bar{\mathbf{\Phi}}^{(n)}$ both for sparse and dense tensors.

Note that the local minima of objective function (1.2) are not isolated, since there is a scaling indeterminacy in each of the rank-one terms: there is ample freedom to rescale the vectors $\mathbf{a}_r^{(n)}$ in (1.1) without changing the rank-one product. We deal with this by normalizing the $\mathbf{a}_r^{(n)}$ as follows. There is also a permutation indeterminacy in the order of the rank-one terms, which we deal with by imposing a specific order as we now explain. If no special care is taken during ALS iterations, it may happen that vectors in some modes diverge while others approach zero size, even if the process is converging to the desired CP decomposition; this is indeed possible due to the scaling indeterminacy. This type of anomalous behavior can be avoided by normalizing the columns of the factor matrices. After each complete ALS iteration, for each rank-one term we first normalize all factor vectors to size one and then distribute the product of the normalization factors evenly to all factor vectors (using the nth root). We also order the rank-one terms in order of decreasing product of normalization factors. We apply this normalization and reordering after each ALS iteration, but also after each calculation of new accelerated or line search iterates in the N-GMRES optimization procedure. This controls possible erratic behavior in the order of the rank-one terms and balances the relative sizes of the factor vectors as they are stacked in the iterate vectors $\mathbf{u}_i$, which is expected to be beneficial for the convergence of the N-GMRES acceleration process. This consistent normalization thus appears to take care of the scaling and permutation indeterminacies present in the CP optimization problem, at least for the problems we have tested, as our numerical results confirm. Note that this normalization is also used for N-CG in [2] and in our N-CG comparison runs. In [2] an alternative regularization approach is discussed, but this is not considered here.

2.4. Application of N-GMRES to CP optimization: Further particulars and parameter choices for numerical tests. In this subsection we discuss some further particulars of our application of N-GMRES to CP optimization and discuss some parameters for the numerical results to be presented in the next two sections. As we mentioned before, and as in [2], we utilize the Moré–Thuente line search method [17] as implemented in the Poblano toolbox for MATLAB [10]. For all experiments,

the Moré–Thuente line search parameters used were as follows: 10^{-4} for the function value tolerance, 10^{-2} for the gradient norm tolerance, a starting search step length of 1, and a maximum of 20 iterations. These values were also used for the N-CG tests in [2], and we use them for our N-CG comparison runs as well. We use the N-CG variant with the Polak–Ribière update formula [18]. As suggested in [27], the parameter δ in the term $\delta \mathbf{I}$ added to normal equation matrix $\mathbf{P}^T\mathbf{P}$ in (2.7) was chosen as ϵ times the size of the largest diagonal element of $\mathbf{P}^T\mathbf{P}$, with $\epsilon = 10^{-12}$. We normally choose the N-GMRES window size w equal to 20, which is confirmed to be a good choice in numerical tests described below. All initial guesses are determined uniformly randomly, as in [22], and when we compare different methods they are given the same random initial guess. All numerical tests were run on a laptop with a quad-core 2.2 GHz Intel Core i7 processor and 8GB of 1333 MHz DDR3 memory. MATLAB version 7.11.0.584 (R2010b) 64-bit (maci64) was used for all tests.

3. Numerical results: Dense tensor test problem. In this section and the next, we present extensive numerical tests for the N-GMRES optimization algorithm, compared with ALS and the N-CG algorithm of [2], accessed via the MATLAB Tensor Toolbox [4]. In this section we present a class of dense test problems, and the next section deals with a sparse problem class.

Our dense test problem generates three-way data tensors of various sizes starting from a canonical tensor with specified rank and random factor matrices that are modified to have prespecified collinearity c, and noise is added. This is a standard CP test problem from [26] and was also used in [2]. The collinearity between different columns of factor matrix $\mathbf{A}^{(n)}$ is defined as

$$c = \frac{\mathbf{a}_r^{(n)T}\,\mathbf{a}_s^{(n)}}{\|\mathbf{a}_r^{(n)}\|_2\,\|\mathbf{a}_s^{(n)}\|_2}. \tag{3.1}$$

Collinearity close to 1 is known to lead to slow ALS convergence [26].

TEST PROBLEM I. *Generate a 3-way data tensor $\mathcal{T}$ of noise-free rank R, size $s \times s \times s$, collinearity c, and noise levels l_1 and l_2 as follows. First generate an $R \times R$ matrix $\mathbf{K}$ that has diagonal elements 1 and off-diagonal elements c, and compute the Cholesky factor $\mathbf{C}$ of $\mathbf{K}$. Then generate n uniformly random $s \times R$ matrices, orthonormalize their columns using the QR decomposition, and multiply on the right with C. Then let $\mathcal{T}_R$ be the canonical rank-R tensor generated by these matrices as factor matrices. Two types of noise are added to $\mathcal{T}_R$. Generate tensors $\mathcal{N}_1$ and $\mathcal{N}_2 \in \mathbb{R}^{s\times s\times s}$ with elements drawn from the standard normal distribution. An intermediate tensor $\hat{\mathcal{T}}$ is generated as $\hat{\mathcal{T}} = \mathcal{T}_R + (100/l_1 - 1)^{-1/2}\,\|\mathcal{T}_R\|_F\,\mathcal{N}_1/\|\mathcal{N}_1\|_F$, and finally $\mathcal{T}$ is obtained as $\mathcal{T} = \hat{\mathcal{T}} + (100/l_2 - 1)^{-1/2}\,\|\hat{\mathcal{T}}\|_F\,(\mathcal{N}_2 * \hat{\mathcal{T}})/\|\mathcal{N}_2 * \hat{\mathcal{T}}\|_F$, where $*$ denotes elementwise multiplication.*

We perform a series of tests computing a rank-R CP decomposition of the tensors $\mathcal{T}$ for various values of s, c, R, l_1, and l_2. Note that in this paper we do not study so-called overfactoring effects [26, 2], in which numerical methods may produce approximations that do not give physically relevant information when the CP-rank R is chosen larger than some rank intrinsic to the data tensor. Our reason for not considering overfactoring is that in this paper we accelerate ALS by the N-GMRES optimization approach, and we have observed that we almost always converge to the same stationary point as ALS. Since the overfactoring properties of ALS have been studied extensively elsewhere [26, 2], we do not consider them here.

Before performing tests for a wide range of parameter values s, c, R, l_1, and l_2, we look at a specific case and study the choice of N-GMRES window size w. Figure 3.1

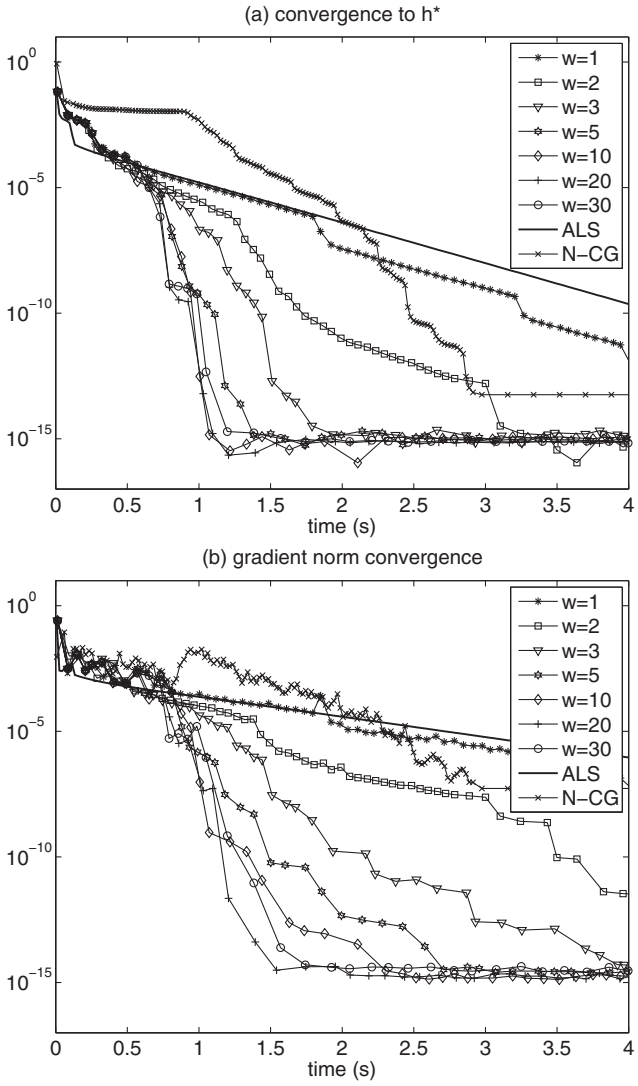

FIG. 3.1. *Test Problem* I *with parameters* ($s = 50, c = 0.9, R = 3, l_1 = 1, l_2 = 1$). *Convergence plots as a function of the N-GMRES window size,* w. *(Every fourth iteration is plotted.)* (a) *Convergence of* $|h(\mathcal{A}_R^{(i)}) - h^*|$, *where* h^* *is the value of the normalized distance measure,* (1.4), *in the stationary point the method converges to.* (b) *Convergence of the normalized gradient of the objective function,* $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$, *indicating convergence to a stationary point. Window size* $w = 20$ *emerges as a good choice for fast convergence when high accuracy is required.*

shows convergence results for an instance of Test Problem I with parameters $s = 50$, $c = 0.9$, $R = 3$, $l_1 = 1$, and $l_2 = 1$. This constitutes a "difficult" case with $c = 0.9$, for which ALS converges rather slowly. It is the same case as in Figures 1.2–1.3. Figure 3.1 shows the effect of the window size w on convergence speed. Several observations can be made. The choice $w = 1$ does not converge much faster than ALS, but $w = 2$ already leads to significant gains when high accuracy is desired, and $w = 3$ already performs reasonably similarly to the best choice. (Note again that these convergence plots take into account the extra costs of the N-GMRES optimization and line search

TABLE 3.1

Test Problem I. *Parameter values for convergence test cases. Test cases* 1–6 *are the "easy" cases, with collinearity* $c = 0.5$*, while test cases* 7–12 *are the "difficult" cases, with collinearity* $c = 0.9$.

1	$s = 20$, $c = 0.5$, $r = 3$, $l_1 = 1$, $l_2 = 1$
2	$s = 20$, $c = 0.5$, $r = 5$, $l_1 = 10$, $l_2 = 5$
3	$s = 50$, $c = 0.5$, $r = 3$, $l_1 = 1$, $l_2 = 1$
4	$s = 50$, $c = 0.5$, $r = 5$, $l_1 = 10$, $l_2 = 5$
5	$s = 100$, $c = 0.5$, $r = 3$, $l_1 = 1$, $l_2 = 1$
6	$s = 100$, $c = 0.5$, $r = 5$, $l_1 = 10$, $l_2 = 5$
7	$s = 20$, $c = 0.9$, $r = 3$, $l_1 = 0$, $l_2 = 0$
8	$s = 20$, $c = 0.9$, $r = 5$, $l_1 = 1$, $l_2 = 1$
9	$s = 50$, $c = 0.9$, $r = 3$, $l_1 = 0$, $l_2 = 0$
10	$s = 50$, $c = 0.9$, $r = 5$, $l_1 = 1$, $l_2 = 1$
11	$s = 100$, $c = 0.9$, $r = 3$, $l_1 = 0$, $l_2 = 0$
12	$s = 100$, $c = 0.9$, $r = 5$, $l_1 = 1$, $l_2 = 1$

in each iteration.) The optimal performance appears to occur, for this test problem, when $w \approx 20$ is chosen. Fast convergence of $h(\mathcal{A}_R^{(i)})$ to the optimal value h^* (see (1.4)) is already obtained with high accuracy for $w \approx 10$, but Figure 3.1(b) shows that $w \approx 20$ leads to faster convergence to a stationary point with high accuracy. Convergence plots as a function of iteration (not shown due to space constraints) show the same pattern, indicating that speed differences are almost entirely due to differing iteration counts and not to differences in the amount of work to build and solve normal equation system (2.7) for varying window size. We use window size $w = 20$ in the numerical tests below.

Tables 3.2–3.4 show convergence results for a series of tests with the wide range of parameter values s, c, R, l_1, and l_2 given in Table 3.1. Ten trials with different random initial guesses were performed for each of the 12 test cases of Table 3.1.

Table 3.2 compares the average number of iterations and the average times (in seconds) that the ALS, N-GMRES, and N-CG (from [2]) methods need to reduce the accuracy measures $|h(\mathcal{A}_R^{(i)}) - h^*|$ and $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ to 10^{-3}. The $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2 / \|\mathcal{T}\|_F$ measure gives information about convergence to a stationary point, and the $|h(\mathcal{A}_R^{(i)}) - h^*|$ measure gives information about convergence to the best fit, which is an important criterion in practice. The numbers in brackets give the number of trials in which the convergence tolerance was successfully met within a prespecified number of iterations or function/gradient evaluations. The details of how these tests are set up are explained below. Table 3.2 compares the three methods for a situation where low-accuracy results are considered sufficient. The smallest average times appear in bold. Table 3.2 shows that ALS is normally the fastest when low accuracy is sufficient, except for some of the difficult problems, where N-GMRES or N-CG may be somewhat faster on average in terms of the $|h(\mathcal{A}_R^{(i)}) - h^*|$ measure. This is consistent with what many others have observed for CP decomposition and other problems: ALS-type algorithms tend to be more efficient than other algorithms when low accuracy is sufficient (see, for example, [26, 2]).

Tables 3.3 and 3.4 show convergence results when higher accuracy is required (10^{-6} and 10^{-10}). Table 3.3, for medium accuracy 10^{-6}, shows that ALS is (not unexpectedly) still fastest for all easy cases ($c = 0.5$), but N-GMRES or N-CG are faster for most of the difficult cases ($c = 0.9$), especially in the gradient measure. Table 3.4, for high accuracy 10^{-10}, confirms that ALS remains faster than N-GMRES

TABLE 3.2

Test Problem I. *Average number of iterations and time (in seconds) until the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure and relative gradient norm measure are reduced to* 10^{-3}*, for the test cases with parameters given in Table* 3.1. *(Tests* 7–12 *are the "difficult" test cases.) Ten trials with random initial guesses are performed for each test case. The numbers in brackets indicate how many times out of ten the method considered was able to find a solution satisfying the required accuracy within a maximum number of iterations or function/gradient evaluations. Only these successful runs are taken into account for the iteration and time averages. The smallest average times appear in bold. ALS is generally fastest for these low-accuracy tests, except for some of the difficult problems, where N-GMRES or N-CG may be somewhat faster on average in terms of the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure.*

	ALS		N-GMRES		N-CG	
	It	Time	It	Time	It	Time
	$\lvert h - h^*\rvert$ accuracy 10^{-3}					
1	15	**0.05** (10)	13	0.13 (10)	36	0.16 (10)
2	12	**0.05** (10)	10	0.12 (10)	45	0.24 (10)
3	12	**0.06** (10)	10	0.15 (10)	29	0.27 (10)
4	19	**0.10** (10)	16	0.26 (10)	47	0.47 (10)
5	11	**0.23** (10)	10	0.63 (10)	38	2.15 (10)
6	36	**0.87** (10)	31	2.18 (10)	50	3.20 (10)
7	511	1.75 (10)	308	3.94 (10)	211	**0.95** (10)
8	26	**0.10** (10)	19	0.20 (10)	169	0.67 (8)
9	306	1.41 (9)	72	**1.28** (9)	244	1.94 (10)
10	15	**0.08** (10)	10	0.21 (10)	151	1.82 (10)
11	281	5.77 (10)	69	**4.80** (10)	279	11.80 (10)
12	17	**0.41** (10)	12	1.12 (10)	144	7.66 (10)
	Relative gradient norm accuracy 10^{-3}					
1	19	**0.07** (10)	15	0.16 (10)	49	0.20 (10)
2	21	**0.08** (10)	15	0.17 (10)	59	0.30 (10)
3	17	**0.08** (10)	12	0.19 (10)	38	0.33 (10)
4	27	**0.14** (10)	21	0.36 (10)	63	0.58 (10)
5	17	**0.34** (10)	12	0.82 (10)	47	2.50 (10)
6	24	**0.56** (10)	17	1.47 (10)	62	3.72 (10)
7	65	**0.22** (10)	39	0.43 (10)	121	0.57 (10)
8	63	**0.24** (10)	39	0.46 (10)	197	0.78 (10)
9	66	**0.31** (10)	43	0.71 (10)	151	1.23 (10)
10	60	**0.31** (10)	34	0.74 (10)	210	2.46 (10)
11	75	**1.53** (10)	52	3.29 (10)	142	6.43 (10)
12	64	**1.51** (10)	39	4.32 (10)	169	8.79 (10)

for all easy cases ($c = 0.5$), but N-GMRES is substantially faster for all difficult cases on average ($c = 0.9$). We explain below why N-CG results are not retained in this table.

The tests reported in Tables 3.2–3.4 have been carefully designed as follows. First it has to be noted that optimization problem (1.2) is difficult: convergence histories may depend significantly on the random initial guess, multiple local minima may exist, and different methods may converge to different stationary points (even when starting from the same initial guess). In order to average out some of these effects and gather sufficient data to compare the performance of the methods in light of this variability, we have performed 10 trials for each of the test cases 1–12 in Table 3.1. Within each trial, the random initial guess for ALS, N-GMRES, and N-CG is the same, but each trial features a different random initial guess (with random seeds chosen arbitrarily). ALS is executed for a maximum of 20,000 iterations, N-GMRES for a maximum of 5,000, and N-CG for a maximum of 7,500 function/gradient evaluations. These numbers are chosen sufficiently large that the methods converge to high accuracy in almost all cases, but their purpose is to stop computation when a method fails to

TABLE 3.3

Test Problem I. *Average number of iterations and time (in seconds) until the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure and relative gradient norm measure are reduced to* 10^{-6} *for the test cases with parameters given in Table* 3.1. *(Tests* 7–12 *are the "difficult" test cases.) Ten trials with random initial guesses are performed for each test case. For these midaccuracy tests, N-GMRES or N-CG are fastest for most of the "difficult" cases, while ALS is fastest for the "easy" cases.*

	ALS		N-GMRES		N-CG	
	It	Time	It	Time	It	Time
	$\lvert h - h^*\rvert$ accuracy 10^{-6}					
1	22	**0.08** (10)	17	0.17 (10)	53	0.21 (10)
2	23	**0.09** (10)	15	0.18 (10)	61	0.30 (10)
3	19	**0.09** (10)	14	0.21 (10)	41	0.35 (10)
4	29	**0.15** (10)	22	0.38 (10)	65	0.60 (10)
5	20	**0.41** (10)	13	0.86 (10)	50	2.61 (10)
6	47	**1.12** (10)	36	2.63 (10)	62	3.73 (10)
7	1234	4.23 (10)	330	4.19 (10)	425	**1.75** (9)
8	582	2.19 (9)	189	2.16 (9)	557	**2.09** (8)
9	1127	5.17 (9)	101	**1.87** (9)	619	4.83 (10)
10	485	**2.49** (10)	175	3.52 (10)	472	5.17 (10)
11	1100	22.57 (10)	93	**6.85** (10)	561	22.68 (10)
12	493	**11.73** (10)	207	20.57 (10)	367	17.91 (10)
	Relative gradient norm accuracy 10^{-6}					
1	37	**0.13** (10)	21	0.21 (10)	83	0.31 (10)
2	44	**0.17** (10)	22	0.26 (10)	99	0.45 (10)
3	34	**0.16** (10)	19	0.30 (10)	61	0.47 (10)
4	50	**0.26** (10)	28	0.49 (10)	99	0.83 (10)
5	35	**0.71** (10)	18	1.29 (10)	71	3.41 (10)
6	68	**1.61** (10)	43	3.30 (10)	85	4.76 (10)
7	984	3.37 (10)	326	4.15 (10)	441	**1.88** (10)
8	1384	5.22 (10)	216	**2.47** (10)	1041	3.80 (10)
9	795	3.65 (10)	87	**1.58** (10)	521	4.06 (10)
10	1156	5.93 (10)	215	**4.31** (10)	905	9.31 (10)
11	841	17.26 (10)	86	**6.23** (10)	489	19.82 (10)
12	1163	27.69 (10)	257	**24.75** (10)	570	26.81 (10)

converge. The tables report average iteration numbers and average times for reaching the convergence tolerance specified in the tables. If for a certain trial a method fails to reach this tolerance within the prescribed number of iterations, the iteration count and time of this failed run is not counted in the averages, and the number of successful runs (used to compute the averages) is noted in brackets after the average time. For Test Problem I, all three methods converge to a stationary point (measured by the relative gradient measure) for all parameter sets and all trials. It is known that N-CG with Moré–Thuente line search converges only up to an accuracy of roughly the square root of machine epsilon (i.e., $\sim 10^{-7}$) due to loss of accuracy in checking the Wolfe sufficient decrease condition; see [13]. We do indeed observe this in our numerical tests, and for this reason we do not include N-CG in the high-accuracy comparisons with tolerance 10^{-10} (Table 3.4). For each trial, we determine a single h^* by finding the overall minimum value of normalized distance measure $h(\mathcal{A}_R^{(i)})$ (see (1.4)) for the three methods over all iterations i. We make sure that this corresponds to a stationary point by verifying convergence of $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$. Within a given trial, it may sometimes happen that not all three methods converge to stationary points with the same value of h. In such a case, only the method that converges to the single minimal h^* will converge to high accuracy in the $|h(\mathcal{A}_R^{(i)}) - h^*|$ measure, and the other runs will appear as failed runs in the top half of the tables (because

TABLE 3.4

Test Problem I. *Average number of iterations and time (in seconds) until the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure and relative gradient norm measure are reduced to* 10^{-10} *for the test cases with parameters given in Table* 3.1. *(Tests* 7–12 *are the "difficult" test cases.) Ten trials with random initial guesses are performed for each test case. For these high-accuracy tests, N-GMRES is faster than ALS for the "difficult" cases, while ALS remains fastest for the "easy" cases.*

	ALS		N-GMRES	
	It	Time	It	Time
	$\lvert h - h^*\rvert$ accuracy 10^{-10}			
1	34	**0.12** (10)	20	0.21 (10)
2	39	**0.15** (10)	20	0.23 (10)
3	31	**0.15** (10)	18	0.28 (10)
4	45	**0.23** (10)	27	0.46 (10)
5	32	**0.65** (10)	17	1.18 (10)
6	62	**1.48** (10)	42	3.16 (10)
7	2320	7.96 (10)	351	**4.57** (10)
8	1483	5.59 (9)	224	**2.55** (9)
9	2213	10.16 (9)	117	**2.35** (9)
10	1226	6.30 (10)	215	**4.31** (10)
11	2186	44.88 (10)	113	**10.60** (10)
12	1232	29.33 (10)	256	**24.67** (10)
	Relative gradient norm accuracy 10^{-10}			
1	61	**0.21** (10)	28	0.36 (10)
2	76	**0.29** (10)	32	0.49 (10)
3	59	**0.27** (10)	25	0.50 (10)
4	81	**0.42** (10)	37	0.79 (10)
5	59	**1.20** (10)	24	2.33 (10)
6	98	**2.32** (10)	51	5.02 (10)
7	2065	7.08 (10)	346	**4.47** (10)
8	3566	13.44 (10)	266	**3.30** (10)
9	1796	8.25 (10)	107	**2.10** (10)
10	2654	13.63 (10)	243	**5.26** (10)
11	1930	39.63 (10)	107	**9.07** (10)
12	2628	62.64 (10)	279	**29.16** (10)

they do not converge to a stationary point with the lowest value of h in the trial). For Test Problem I, this happens four times: in three trials of test case 8, and in one trial of test case 9. In each of these cases, ALS and N-GMRES converge to the same stationary point, but N-CG converges to a different one. For the three trials of test case 8, ALS/N-GMRES converge to a stationary point with lower h value than N-CG in two of the three cases, and it is the other way around in the third case. For the trial of test case 9, N-CG converges to a stationary point with lower h value than ALS/N-GMRES. This is reflected, for example, in the "success" numbers in the top half of Table 3.3, rows 7 and 8 (the "(8)" and "(9)" indicate that there were one and two failures to reach the convergence tolerance).

For further illustration, Figure 3.2 shows convergence plots for one trial of test case 7 from Tables 3.2–3.4 (this is one of the difficult test cases). For this problem (and the random initial guess used), convergence of N-GMRES sets in only after more than 150 iterations. Figure 3.2(b) gives some indication as to why this is the case. N-GMRES accelerates ALS, but ALS needs to get over a bump in the size of the gradient before it gets close enough to a stationary point for the N-GMRES acceleration to become effective. This points out a fundamental property of the N-GMRES acceleration procedure that is the topic of this paper: N-GMRES is not expected to offer a systematic improvement in the global convergence properties of the preconditoning process it accelerates. The only potential help with global convergence

would come from the line search in Step III of the algorithm, but this will be effective only when the search direction jointly determined by ALS and N-GMRES is suitable, and, as we argued before, we can only expect this to be the case consistently when the process approaches a stationary point. So N-GMRES still mainly relies on the preconditioning process to guide convergence on the global scale and kicks in when close enough to a stationary point. In the case of Figure 3.2, N-CG appears to perform better initially in getting close to a stationary point (see especially Figure 3.2(c)). Indeed, N-CG works in a way that is fundamentally different from N-GMRES: N-CG does not accelerate an underlying one-step method but follows its own strategy for exploring the global search space, with globalization provided by its line search.

Figure 3.2 also illustrates the convergence stalling of N-CG with Moré–Thuente line search around accuracy $\sim 10^{-7}$, which we observe in almost all our tests. This can be explained by loss of accuracy in checking the Wolfe sufficient decrease condition in the line search; see [13]. It is interesting to note that N-GMRES does not suffer from this convergence stalling, even though we use the same Moré–Thuente line search. This different behavior of the two methods can be understood as follows. For N-GMRES, the line search step is only an insurance to guard against large steps in unfruitful directions when far away from the solution, and if the iterate predicted by the nonlinear acceleration mechanism converges to the solution, the line search

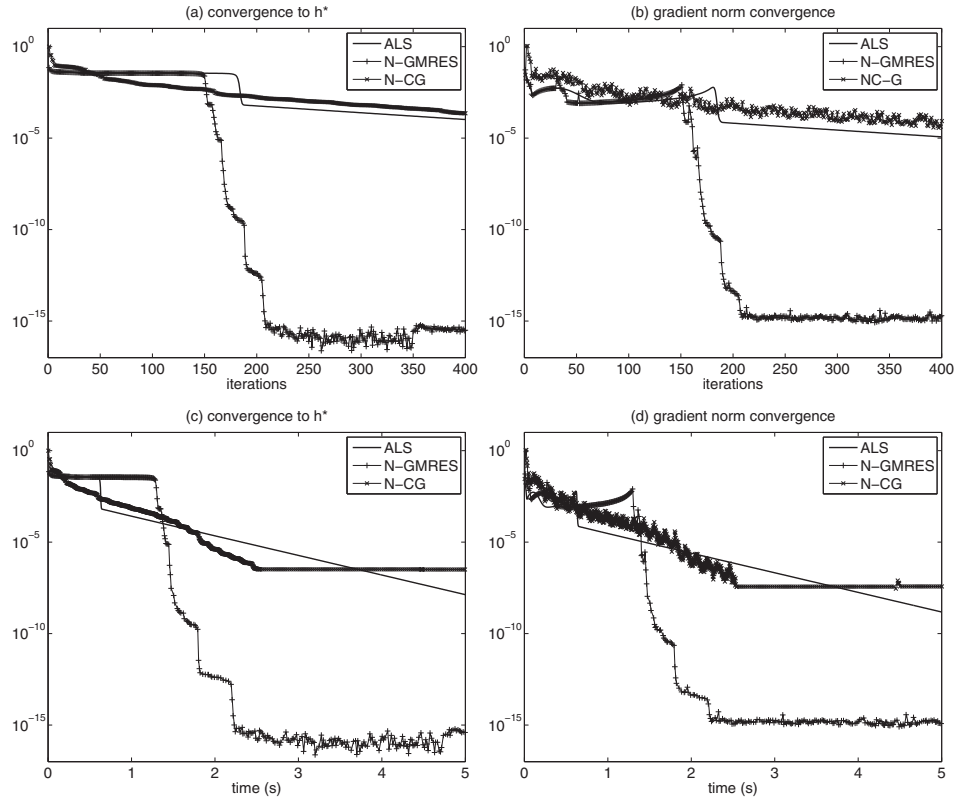

FIG. 3.2. *Test Problem* I. *Convergence plots for a trial of test case* 7 *from Tables* 3.2–3.4. *Panel* (b) *shows that N-GMRES convergence kicks in only when ALS has gotten over a "bump" in the gradient size.*

is essentially inactive because it is not needed (a step length of 1, up to the point produced by the nonlinear acceleration mechanism, is selected by default if that point satisfies the Wolfe conditions). N-CG, in contrast, only produces a step direction, and the step length provided by the line search is important in every step, also close to convergence. If a step length cannot be determined accurately, the method cannot converge with high accuracy. This difference points to a potential advantage of N-GMRES over N-CG with Moré–Thuente line search when high accuracy is required. Note, however, that it is possible to implement a line search procedure for N-CG that does not suffer from this loss in precision and satisfies modified Wolfe conditions [13].

We finally compare the performance of the three algorithms with respect to execution time and the relative gradient convergence measure using the convergence profiles of Dolan and Moré [9]; see Figure 3.3. For each method, we plot the fraction of total test runs for which the method is within a factor τ of the best time in the trial, for a given accuracy of $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$. The left panels of Figure 3.3 deal with the easy test cases of Test Problem I, namely, cases 1–6 from Tables 3.2–3.4 (60 trials in total), and the right panels deals with the difficult cases (also 60 trials in total). Plotted values for each method at $\tau = 1$ show for which fraction of the 60 trials each of the methods is the fastest. For example, from Figure 3.3(a) one can read off that ALS is the fastest for accuracy 10^{-3} in all of the 60 trials for the easy problems, while Figure 3.3(d) shows that N-GMRES is the fastest for accuracy 10^{-6} in 80% of the 60 trials for the difficult problems (with N-CG the fastest in about 15%, and ALS in about 5%). Values for increasing τ give information on how much more slowly than the fastest method each method converges (per trial) in a cumulative way. For example, the values at $\tau = 3$ in Figure 3.3(d) for the difficult problems shows that almost 100% of the N-GMRES runs reach the required accuracy within three times the fastest time (for each trial), wheres ALS and N-CG each reach that accuracy within three times the fastest time in about 75% of the trials. These τ-profiles thus allow us to compare the methods in an overall sense [9, 13]. They complement the comparison in Tables 3.2–3.4. For example, outliers (which occur in trials in which one of the methods converges much more slowly than in most of the other trials) can have a significant influence on the averages in Tables 3.2–3.4, but they have less influence on the values in the τ-profiles for small τ. As such, both the tables and the τ-profiles give complementary information. The left panels of Figure 3.3 show that ALS is the top curve for all accuracies for the easy problems (it is the method that solved the most problems in a time that was within a factor τ of the best time), so it is clearly the fastest method for the easy problems, by the measure of these τ-profiles. For the difficult problems (right panels), ALS is still clearly the fastest for low accuracy (10^{-3}), but for medium and high accuracy (10^{-6} and 10^{-10}) N-GMRES is clearly the fastest according to the τ-profile measure. N-GMRES is also faster than N-CG in all plots.

3.1. Comparison to ALS with enhanced line search. Line search methods have been considered before to speed up ALS [26, 22], and it is interesting to compare our N-GMRES approach, which also uses ALS and line search as two of its three components, with these line search-enhanced ALS methods. In particular, we compare our approach to the ALS with enhanced line search method (ALS-ELS) from [22]. For the trivial case of $w = 1$ (i.e., without the nonlinear acceleration step in Algorithm 1), N-GMRES is indeed similar to ALS-ELS: ALS-ELS produces a sequence of iterates $u_0, u_1, u_2, \ldots$ in which $u_{2i+1} = ALS(u_{2i})$ and u_{2i} is obtained by an *exact* line search in the direction $u_{2i-1} - u_{2i-2}$, while N-GMRES with $w = 1$ (no nonlin-

ear acceleration mechanism) produces a sequence of iterates $u_0, u_1, u_2, \ldots$ in which $u_{2i+1} = ALS(u_{2i})$ and u_{2i} is obtained by an *approximate* line search in the direction $u_{2i-1} - u_{2i-2}$ (using Moré–Thuente line search to get a new point that satisfies the

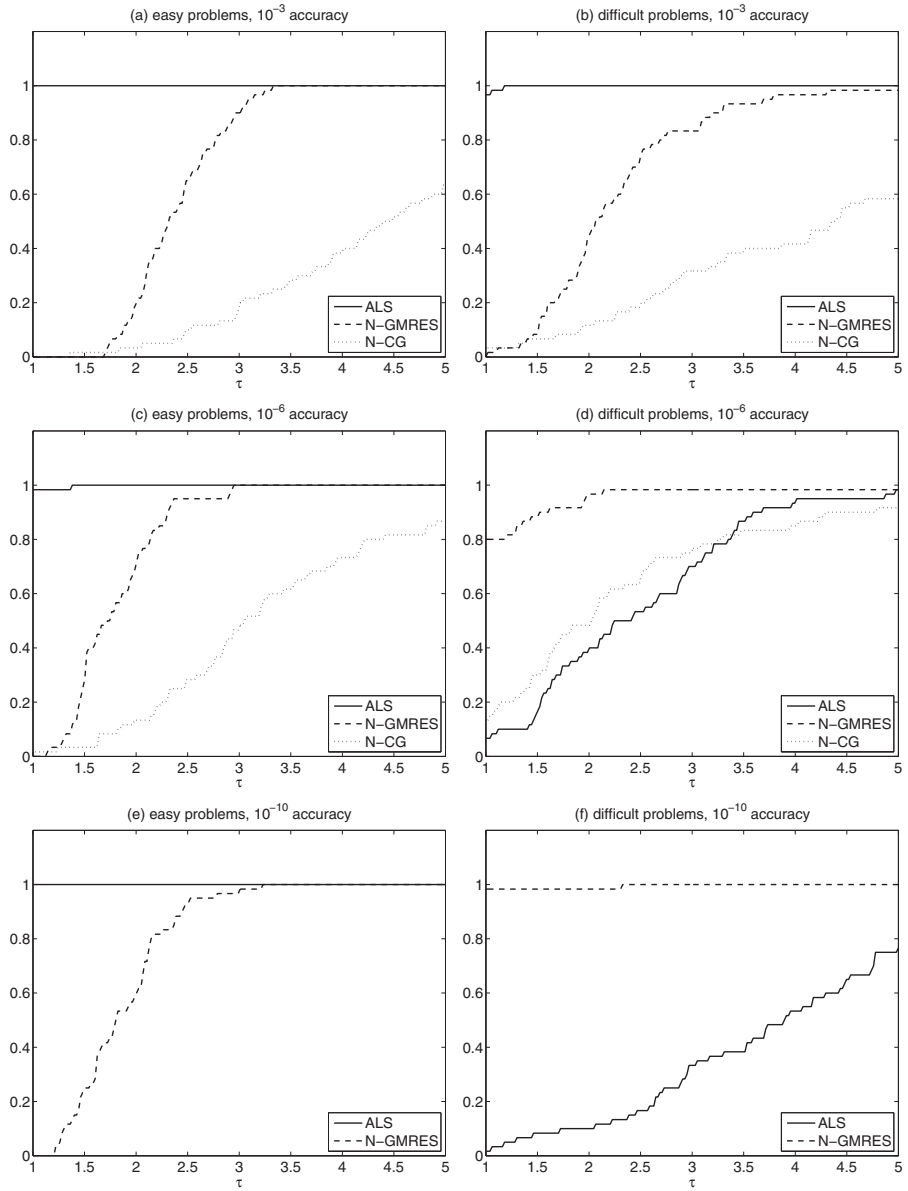

FIG. 3.3. *Test Problem* I. *τ-profiles for ALS, N-CG, and N-GMRES for the tests of Tables* 3.2–3.4 *(relative gradient measure). For each method, we plot the fraction of total test runs for which the method is within a factor τ of the best time in the trial for a given accuracy of $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$. The left panels deal with the* 60 *trials of the "easy" test cases of Test Problem* I, *namely, cases* 1–6 *from Tables* 3.2–3.4, *and the right panels deal with the "difficult" test cases* (7–12). *ALS is the top curve and can thus be called the fastest for the "easy" test problems and for the "difficult" test problems at low accuracy (10^{-3}), but N-GMRES is the fastest for the "difficult" test cases at medium and high accuracy (10^{-6} and 10^{-10}).*

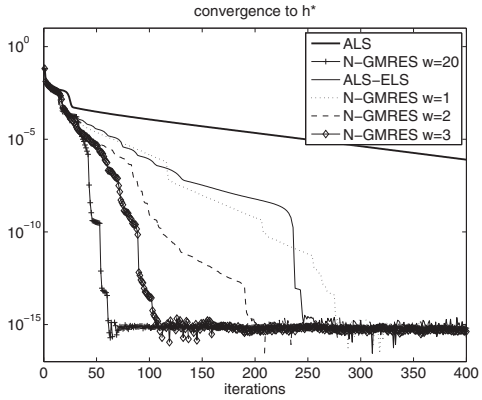

FIG. 3.4. *Test Problem* I. *Comparison of N-GMRES, ALS and ALS with Enhanced Line Search (ALS-ELS) for the problem of Figure* 1.2 *(parameters* $s = 50$, $c = 0.9$, $R = 3$, $l_1 = 1$, $l_2 = 1$*); convergence of* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *in terms of number of iterations is shown. ALS-ELS significantly speeds up ALS and performs similarly to N-GMRES with window size* $w = 1$*, as expected. N-GMRES with window sizes larger than* $w = 1$ *further improves on ALS-ELS in a significant way.*

Wolfe conditions). (Note that an exact line search is possible for CP optimization by computing the minimum of a high-order polynomial, due to the multilinear nature of Optimization Problem I, (1.2); see [22].) We can thus expect that ALS-ELS and the trivial $w = 1$ case of N-GMRES may perform similarly and that activating the nonlinear acceleration mechanism in N-GMRES ($w \geq 2$), which is its main component, will further improve significantly over both the trivial N-GMRES case with $w = 1$ and ALS-ELS. We have tested this using the MATLAB code of ALS-ELS that is provided online by the authors of [22]. Figure 3.4 shows a comparison between ALS, ALS-ELS (with exact line search), N-GMRES with $w = 1$ (with approximate line search and without nonlinear acceleration mechanism), and the full N-GMRES with $w \geq 2$. As expected, ALS-ELS and N-GMRES with $w = 1$ both improve over ALS in a similar

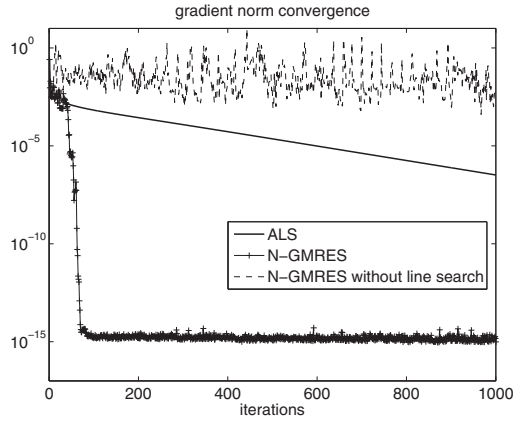

FIG. 3.5. *Test Problem* I. *Comparison of N-GMRES, ALS, and N-GMRES without the line search step for the problem of Figure* 1.2 *(parameters* $s = 50$, $c = 0.9$, $R = 3$, $l_1 = 1$, $l_2 = 1$*); convergence of the relative gradient in terms of number of iterations is shown. N-GMRES without line search (dashed line) shows erratic convergence. As explained in section* 2*, the line search step is necessary for globalization, since erratic convergence behavior is likely to occur without it.*

way, but further significant improvement is made by activating the nonlinear acceleration mechanism in N-GMRES with $w \geq 2$. Note that we show only iteration plots because the MATLAB code provided by the authors of [22] is not efficient for large problems. (For example, the N-GMRES runs of Figure 3.4 converged in about 1s, while the ALS-ELS run took about 150s.)

We conclude this section with a final note. In Figure 3.5 we illustrate why the line search step in N-GMRES (Step III in Algorithm 1) is essential: without it, erratic convergence behavior is indeed likely to occur.

4. Numerical results: Sparse tensor test problem. In this section, we present numerical results for a simple sparse test problem.

TEST PROBLEM II. *Standard finite difference Laplacian tensor on a regular grid of size s^d in d dimensions. This test problem results in an N-way sparse data tensor $\mathcal{T}$ with $N = 2\,d$ and nonzero elements of value $2\,d$ and -1. For example, for $d = 2$, $\mathcal{T} \in \mathbb{R}^{s\times s\times s\times s}$, and the nonzero tensor elements are $t(i,j,i,j) = 2\,d = 4$ for $i = 1,\ldots,s$ and $j = 1,\ldots,s$, and $t(i,j,i+1,j) = -1$, $t(i,j,i,j+1) = -1$, $t(i,j,i-1,j) = -1$, and $t(i,j,i,j-1) = -1$ (with the usual exceptions at boundary points of the d-dimensional regular grid).*

This is an academic test problem, but it provides sparse tensors that are suitable for testing our numerical method.

As in the previous section, we first consider the effect of varying the N-GMRES window size. Figure 4.1 shows convergence plots for an instance of Test Problem II with parameters $d = 3$ and $s = 6$. Tensor $\mathcal{T}$ is a six-way tensor of size $6\times6\times6\times6\times6\times6$, and we seek a CP decomposition with $R = 3$. Figure 4.1, with N-GMRES window size $w = 20$, shows that ALS is rather slow for this problem. "N-GMRES significantly reduces the number of iterations." Figure 4.2 shows convergence plots as a function of time for this test problem, for varying window size w. For this test problem, as in the previous section, window size $w = 20$ also appears a suitable choice, and we use it for the remaining numerical tests in this section.

Tables 4.2–4.4 show convergence results for a series of sparse Test Problem II runs with the wide range of parameter values N, s, and R given in Table 4.1. Ten trials with different random initial guesses were performed for each of the five test cases

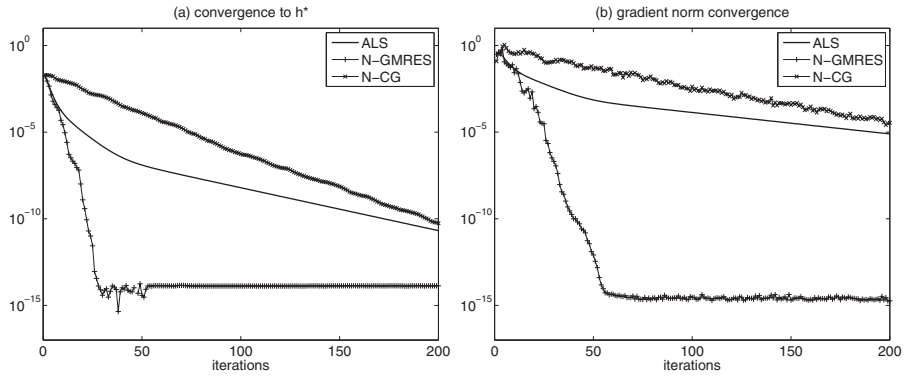

FIG. 4.1. *Test Problem* II. *Parameters are $d = 3$ (and thus $N = 6$), $s = 6$, and $R = 3$, leading to a 6-way tensor of size $6 \times 6 \times 6 \times 6 \times 6 \times 6$ for which an $R = 3$ CP approximation is sought. Convergence plots for ALS, N-CG, and N-GMRES with window size $w = 20$.*

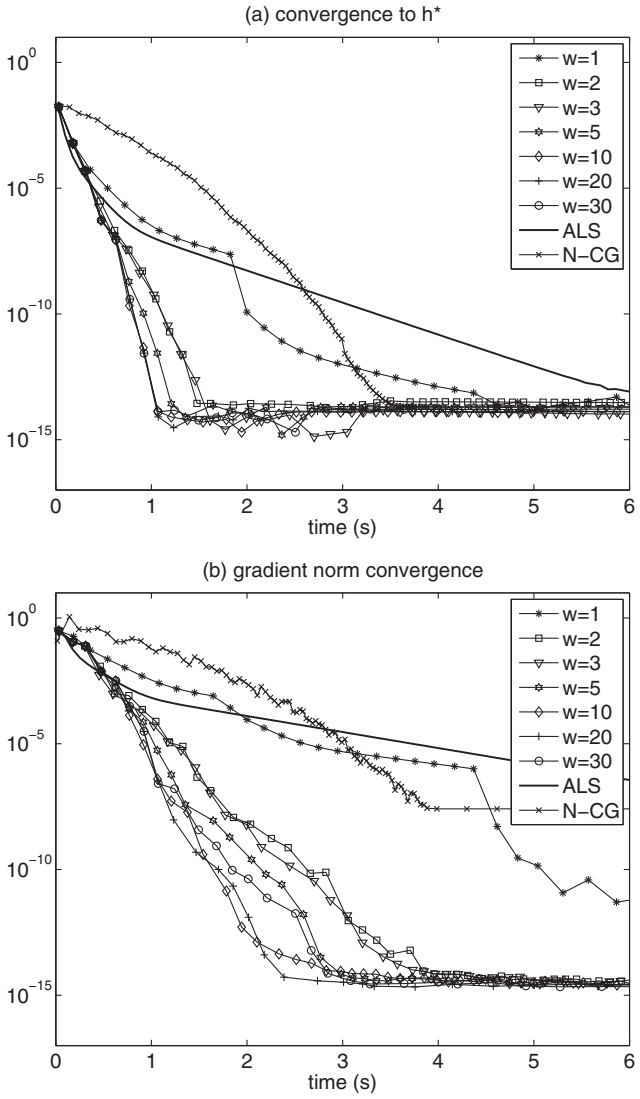

FIG. 4.2. *Test Problem* II *($N = 6, s = 6, R = 3$). Convergence plots as a function of the N-GMRES window size, w. (Every fourth iteration is represented by a plotting symbol.) Window size $w = 20$ emerges as a good choice for fast convergence when high accuracy is required.*

TABLE 4.1
Test Problem II. *Parameter values for convergence test cases.*

1	$N = 4$, $s = 8$, $R = 6$
2	$N = 4$, $s = 16$, $R = 3$
3	$N = 6$, $s = 4$, $R = 2$
4	$N = 6$, $s = 8$, $R = 5$
5	$N = 8$, $s = 4$, $R = 2$

of Table 4.1. The tables report the average number of iterations and the average times (in seconds) needed to reach low, medium, and high convergence tolerances, and Figure 4.3 shows the τ-profiles.

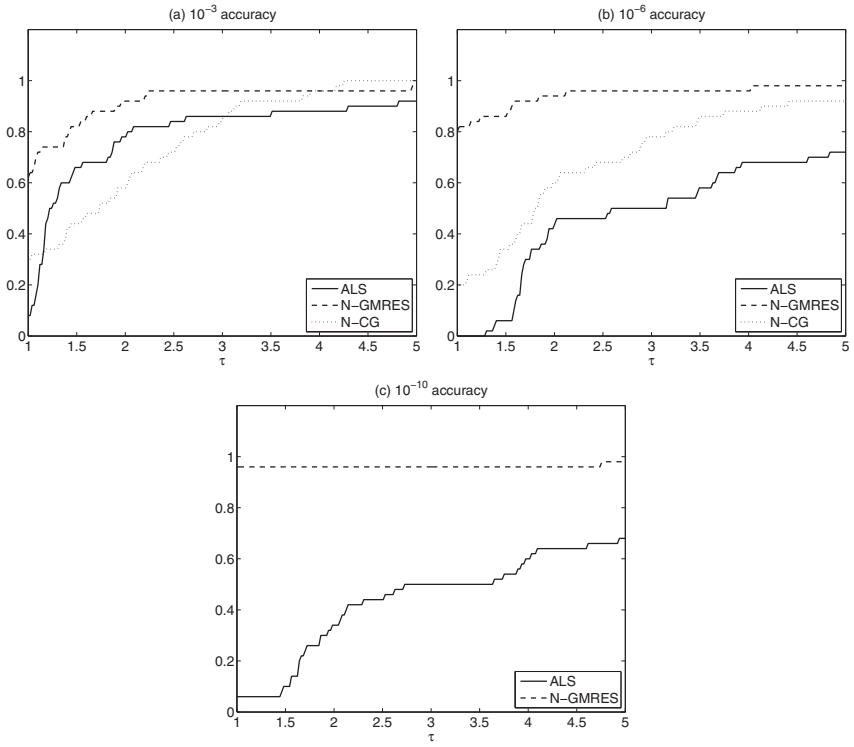

FIG. 4.3. *Test Problem* II. *τ-profiles for ALS, N-CG, and N-GMRES for the tests of Tables* 4.2–4.4 *(relative gradient measure). For each method, we plot the fraction of total test runs for which the method is within a factor τ of the best time in the trial, for a given accuracy of $\|\mathbf{g}(\mathcal{A}_R^{(i)})\|_2/\|\mathcal{T}\|_F$ (there are a total of* 50 *trials). For low accuracy (10^{-3}), N-GMRES, ALS, and N-CG perform largely similarly (N-GMRES is somewhat faster). For medium and high accuracy (10^{-6} and 10^{-10}), N-GMRES is clearly faster.*

TABLE 4.2

Test Problem II. *Average number of iterations and time (in seconds) until the $|h(\mathcal{A}_R^{(i)}) - h^*|$ measure and relative gradient norm measure are reduced to 10^{-3} for the test cases with parameters given in Table* 4.1. *Ten trials with random initial guesses are performed for each test case. The numbers in brackets indicate how many times out of ten the method considered was able to find a solution satisfying the required accuracy within a maximum number of iterations or function/gradient evaluations. Only these successful runs are taken into account for the iteration and time averages. The smallest average times appear in bold. ALS is generally fastest for these low-accuracy tests in the $|h(\mathcal{A}_R^{(i)}) - h^*|$ measure, while N-GMRES or N-CG are somewhat faster for the gradient measure.*

	ALS		N-GMRES		N-CG	
	It	Time	It	Time	It	Time
	$\|h - h^*|$ accuracy 10^{-3}					
1	30	0.57 (5)	23	0.69 (5)	35	**0.38** (9)
2	14	**0.16** (10)	9	0.22 (10)	19	0.28 (10)
3	11	**0.16** (10)	8	0.23 (10)	31	0.58 (10)
4	13	**0.40** (10)	11	0.82 (10)	41	3.62 (10)
5	9	**0.21** (10)	7	0.32 (10)	55	2.03 (10)
	Relative gradient norm accuracy 10^{-3}					
1	121	2.27 (10)	36	1.07 (10)	81	**0.72** (10)
2	169	1.95 (10)	53	1.26 (10)	111	**1.15** (10)
3	35	0.51 (10)	15	**0.44** (10)	70	1.08 (10)
4	164	5.13 (10)	60	**4.75** (10)	131	9.01 (10)
5	45	1.02 (10)	18	**0.87** (10)	58	2.12 (10)

TABLE 4.3

Test Problem II. *Average number of iterations and time (in seconds) until the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure and relative gradient norm measure are reduced to* 10^{-6} *for the test cases with parameters given in Table* 4.1. *N-GMRES or N-CG are fastest for most of these midaccuracy tests.*

	ALS		N-GMRES		N-CG	
	It	Time	It	Time	It	Time
	$\lvert h - h^*\rvert$ accuracy 10^{-6}					
1	198	3.71 (5)	49	1.46 (5)	76	**0.69** (9)
2	201	2.32 (7)	58	1.35 (7)	100	**1.06** (10)
3	28	0.41 (10)	13	**0.36** (10)	55	0.94 (10)
4	135	**4.21** (7)	54	4.22 (7)	98	7.40 (7)
5	33	0.76 (10)	15	**0.70** (10)	89	3.24 (10)
	Relative gradient norm accuracy 10^{-6}					
1	1030	19.40 (10)	319	9.81 (10)	236	**1.62** (10)
2	695	7.99 (10)	74	**1.74** (10)	261	2.16 (10)
3	70	1.02 (10)	23	**0.64** (10)	124	1.58 (10)
4	686	21.46 (10)	83	**6.63** (10)	307	14.80 (10)
5	92	2.10 (10)	25	**1.20** (10)	187	4.92 (10)

TABLE 4.4

Test Problem II. *Average number of iterations and time (in seconds) until the* $|h(\mathcal{A}_R^{(i)}) - h^*|$ *measure and relative gradient norm measure are reduced to* 10^{-10} *for the test cases with parameters given in Table* 4.1. *N-GMRES is fastest for all these high-accuracy tests.*

	ALS		N-GMRES	
	It	Time	It	Time
	$\lvert h - h^*\rvert$ accuracy 10^{-10}			
1	460	8.62 (5)	66	**1.97** (5)
2	636	7.32 (7)	76	**1.78** (7)
3	51	0.75 (10)	18	**0.50** (10)
4	459	14.39 (7)	71	**5.50** (7)
5	64	1.48 (10)	20	**0.96** (10)
	Relative gradient norm accuracy 10^{-10}			
1	915	17.17 (8)	114	**4.16** (9)
2	1420	16.34 (10)	92	**2.46** (10)
3	117	1.70 (10)	34	**1.02** (10)
4	1488	46.57 (10)	326	**39.14** (10)
5	155	3.56 (10)	35	**1.78** (10)

Overall, the results show that, for low accuracy, ALS, N-GMRES, and N-CG perform similarly, while for medium and high accuracy N-GMRES is the fastest. We have found that for Test Problem II it happens more often that ALS, N-GMRES, and N-CG converge to stationary points with different values of h than for Test Problem I. This happens for 15 out of the 50 trials for the test cases of Table 4.1: 6 out of 10 times for test case 1, 3 out of 10 times for test case 2, and 6 out of 10 times for test case 4. For these, N-CG converges to a stationary point with a lower value of h in 11 out of 15 times (5 out of 6 for test case 1, 3 out of 3 for test case 2, and 3 out of 6 for test case 4). ALS and N-GMRES always converge to a stationary point with the same value of h. This explains the sometimes low number of successful runs in brackets in the top halves of Tables 4.2–4.4. Also, there is one trial for test case 1 where ALS and N-GMRES fail to converge beyond 10^{-7} in the relative residual measure, and ALS fails to converge beyond 10^{-9} in another trial of test case 1.

In summary, we can say for Test Problem II that, for low accuracy, the three methods perform similarly, while for medium and high accuracy, N-GMRES is the fastest method most of the time.

5. Conclusion. We have presented a new optimization algorithm for computing a canonical rank-R tensor approximation that has minimal distance to a given tensor in the Frobenius norm. The optimization algorithm uses a nonlinear version of GMRES iterate recombination that was proposed by Washio and Oosterlee for systems of nonlinear PDEs [27], which is itself related to other existing acceleration methods for nonlinear equation systems. We apply this nonlinear GMRES acceleration to the ALS method, with the goal of efficiently driving the gradient of the CP objective function to zero, and combine it with a line search for globalization. The resulting three-step N-GMRES optimization algorithm can be interpreted as an acceleration process of a one-step stand-alone method, or, alternatively, the stand-alone method can be considered as a preconditioning process for N-GMRES. We have explained how N-GMRES can be applied to the (approximate) canonical tensor decomposition problem.

Extensive numerical tests on dense and sparse tensors with varying sizes and dimensions (up to 8) show that N-GMRES with ALS preconditioning in many cases outperforms pure ALS when high accuracy is required, while ALS remains faster for low accuracy and easy problems. N-GMRES is also competitive with the N-CG method studied in [2] and appears to outperform it significantly in some cases. In cases where ALS-preconditioned N-GMRES is slower than pure ALS, it is rarely so by much, and in the other cases the potential speed gain is substantial. For this reason, it may be a good strategy to put an N-GMRES acceleration wrapper around ALS if one does not know in advance whether ALS would converge slowly (which may depend on the problem and on the initial guess). Generally speaking, one may expect that adding an N-GMRES acceleration wrapper makes ALS convergence more robust.

It is a question of extensive current interest how Krylov methods can be generalized to tensor computations; see, for example, [25]. Our work presents the application of a nonlinear Krylov-type method to a tensor optimization problem, and we obtain acceleration that is significant in many cases. Our approach uses a nonlinear generalization of Krylov techniques, and it is perhaps not surprising that this seems to be a promising approach for tensor minimization problems, which are indeed inherently nonlinear (more specifically, multilinear).

A promising avenue for further research is to explore how the proposed N-GMRES optimization algorithm can be used to accelerate existing methods other than ALS for canonical tensor decomposition. Similarly, it would also be interesting to investigate how the N-GMRES optimization algorithm can accelerate ALS-type algorithms (or other algorithms) for other tensor approximation problems—for example, the best rank-(R_1, R_2, R_3) approximation of a tensor; see, e.g., [15]. Furthermore, the nonlinear GMRES optimization algorithm proposed in this paper is based on general concepts and may be applied to other nonlinear optimization problems. In a follow-up paper [8], we have investigated steepest-descent preconditioning as a general way to make the N-GMRES optimization method proposed here broadly applicable to nonlinear optimization problems.

Acknowledgment. The research was conducted during a sabbatical visit at the Algorithms and Complexity Department of the Max Planck Institute for Informatics in Saarbruecken, whose hospitality is greatly acknowledged.

REFERENCES

[1] D. G. Anderson, *Iterative procedures for nonlinear integral equations*, J. A. C. M., 12 (1965), pp. 547–560.

[2] E. Acar, D. M. Dunlavy, and T. G. Kolda, *A scalable optimization approach for fitting canonical tensor decompositions*, J. Chemometrics, 25 (2011), pp. 67–86.

[3] V. De Silva and L.-H. Lim, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM. J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.

[4] B. W. Bader and T. G. Kolda, *MATLAB Tensor Toolbox Version* 2.4, http://csmr.ca.sandia.gov/∼tgkolda/TensorToolbox/, 2010.

[5] J. D. Carroll and J. J. Chang, *Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[6] L. De Lathauwer, *A link between the canonical decomposition in multilinear algebra and simultaneous matrix diagonalization*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 642–666.

[7] L. De Lathauwer, B. De Moor, and J. Vandewalle, *Computation of the canonical decomposition by means of a simultaneous generalized Schur decomposition*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 295–327.

[8] H. De Sterck, *Steepest descent preconditioning for nonlinear GMRES optimization*, Numer. Linear Algebra Appl., to appear.

[9] E. D. Dolan and J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.

[10] D. M. Dunlavy, T. G. Kolda, and E. Acar, *Poblano* v1.0*: A MATLAB Toolbox for Gradient-Based Optimization*, Technical report SAND2010-1422, Sandia National Laboratories, Albuquerque, NM, and Livermore, CA, 2010.

[11] H. Fang and Y. Saad, *Two classes of multisecant methods for nonlinear acceleration*, Numer. Linear Algebra Appl., 16 (2009), pp. 197–221.

[12] D. R. Fokkema, G. L. G. Sleijpen, and H. A. Van der Vorst, *Accelerated inexact Newton schemes for large systems of nonlinear equations*, SIAM J. Sci. Comput., 19 (1998), pp. 657–674.

[13] W. W. Hager and H. Zhang, *A new conjugate gradient method with guaranteed descent and an efficient line search*, SIAM J. Optim., 16 (2005), pp. 170–192.

[14] R. A. Harshman, *Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[15] M. Ishteva, L. De Lathauwer, P. Absil, and S. Van Huffel, *Differential-geometric Newton method for the best rank-(R_1, R_2, R_3) approximation of tensors*, Numer. Algorithms, 51 (2009), pp. 179–194.

[16] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Rev., 51 (2009), pp. 455–500.

[17] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Software, 20 (1994), pp. 286–307.

[18] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, Berlin, 2006.

[19] C. W. Oosterlee, *On multigrid for linear complementarity problems with application to American-style options*, Electron. Trans. Numer. Anal., 15 (2003), pp. 165–185.

[20] C. W. Oosterlee and T. Washio, *Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows*, SIAM J. Sci. Comput., 21 (2000), pp. 1670–1690.

[21] P. Pulay, *Convergence acceleration of iterative sequences: The case of SCF iteration*, Chem. Phys. Lett., 73 (1980), pp. 393–398.

[22] M. Rajih, P. Comon, and R. A. Harshman, *Enhanced line search: A novel method to accelerate PARAFAC*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1128–1147.

[23] T. Rohwedder and R. Schneider, *An analysis for the DIIS acceleration method used in quantum chemistry calculations*, J. Math. Chem., 49 (2011), pp. 1889–1914.

[24] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[25] B. Savas and L. Eldén, *Krylov-type methods for tensor computations* I, Linear Algebra Appl., to appear.

[26] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the PARAFAC model*, Comput. Statist. Data Anal., 50 (2006), pp. 1700–1734.

[27] T. Washio and C. W. Oosterlee, *Krylov subspace acceleration for nonlinear multigrid schemes*, Electron. Trans. Numer. Anal., 6 (1997), pp. 271–290.

[28] H. F. Walker and P. Ni, *Anderson acceleration for fixed-point iterations*, SIAM J. Numer. Anal., 49 (2011), pp. 1715–1735.