Steepest Descent Preconditioning for Nonlinear GMRES Optimization

Hans De Sterck*

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

SUMMARY

Steepest descent preconditioning is considered for the recently proposed nonlinear generalized minimal residual (N-GMRES) optimization algorithm for unconstrained nonlinear optimization. Two steepest descent preconditioning variants are proposed. The first employs a line search, while the second employs a predefined small step. A simple global convergence proof is provided for the N-GMRES optimization algorithm with the first steepest descent preconditioner (with line search), under mild standard conditions on the objective function and the line search processes. Steepest descent preconditioning for N-GMRES optimization is also motivated by relating it to standard non-preconditioned GMRES for linear systems in the case of a standard quadratic optimization problem with symmetric positive definite operator. Numerical tests on a variety of model problems show that the N-GMRES optimization algorithm is able to very significantly accelerate convergence of stand-alone steepest descent optimization. Moreover, performance of steepest-descent preconditioned N-GMRES is shown to be competitive with standard nonlinear conjugate gradient and limited-memory Broyden-Fletcher-Goldfarb-Shanno methods for the model problems considered. These results serve to theoretically and numerically establish steepest-descent preconditioned N-GMRES as a general optimization method for unconstrained nonlinear optimization, with performance that appears promising compared to established techniques. In addition, it is argued that the real potential of the N-GMRES optimization framework lies in the fact that it can make use of problem-dependent nonlinear preconditioners that are more powerful than steepest descent (or, equivalently, N-GMRES can be used as a simple wrapper around any other iterative optimization process to seek acceleration of that process), and this potential is illustrated with a further application example.

KEY WORDS: nonlinear optimization, GMRES, steepest descent, preconditioning, conjugate gradient method, BFGS method

1. Introduction

In recent work on canonical tensor approximation [1], we have proposed an algorithm that accelerates convergence of the alternating least squares (ALS) optimization method for the canonical tensor approximation problem considered there. The algorithm proceeds by linearly

*Correspondence to: hdesterck@uwaterloo.ca

recombining previous iterates in a way that approximately minimizes the residual (the gradient of the objective function), using a nonlinear generalized minimal residual (GMRES) approach. The recombination step is followed by a line search step for globalization, and the resulting three-step non-linear GMRES (N-GMRES) optimization algorithm is shown in [1] to significantly speed up the convergence of ALS for the canonical tensor approximation problem considered.

As explained in [1] (which we refer to as Paper I in what follows), for the tensor approximation problem considered there, ALS can also be interpreted as a preconditioner for the N-GMRES optimization algorithm. The question then arises what other types of preconditioners can be considered for the N-GMRES optimization algorithm proposed in Paper I, and whether there are universal preconditioning approaches that can make the N-GMRES optimization algorithm applicable to nonlinear optimization problems more generally. In the present paper, we propose such a universal preconditioning approach for the N-GMRES optimization algorithm proposed in Paper I, namely, steepest descent preconditioning. We explain how updates in the steepest descent direction can indeed naturally be used as a preconditioning process for the N-GMRES optimization algorithm. In fact, we show that steepest descent preconditioning can be seen as the most basic preconditioning process for the N-GMRES optmization method, in the sense that applying N-GMRES to a quadratic objective function with symmetric positive definite (SPD) operator, corresponds mathematically to applying standard non-preconditioned GMRES for linear systems to the linear system corresponding to the quadratic objective function. We propose two variants of steepest descent preconditioning, one with line search and one with a predefined small step. We give a simple global convergence proof for the N-GMRES optimization algorithm with our first proposed variant of steepest descent preconditioning (with line search), under standard mild conditions on the objective function and for line searches satisfying the Wolfe conditions. The second preconditioning approach, without line search, is of interest because it is more efficient in numerical tests, but there is no convergence guarantee. Numerical results are employed for a variety of test problems demonstrating that N-GMRES optimization can significantly speed up stand-alone steepest descent optimization. Performance of N-GMRES with steepest descent preconditioning is compared with the standard steepest descent method and with two well-known and widely used nonlinear optimization methods, namely the nonlinear conjugate gradient (N-CG) method and the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method.

1.1. Numerical Methods for Unconstrained Nonlinear Optimization

We consider the following unconstrained nonlinear optimization problem with associated first-order optimality equations:

OPTIMIZATION PROBLEM I:

$$\text{find } \mathbf{u}^* \text{ that minimizes } f(\mathbf{u}). \tag{1.1}$$

FIRST-ORDER OPTIMALITY EQUATIONS I:

$$\nabla f(\mathbf{u}) = \mathbf{g}(\mathbf{u}) = 0. \tag{1.2}$$

Many of the most widely used nonlinear unconstrained optimization methods are formulated either within the line search or within the trust region frameworks [2]. In the line search

framework, the algorithm computes a search direction $\mathbf{p}_k$ and searches along this direction for a new iterate $\mathbf{u}_{k+1}$ with a lower function value, starting from the current iterate, $\mathbf{u}_k$. Two prototypical choices for the search direction are provided by the steepest descent direction, $\mathbf{p}_k = -\nabla f(\mathbf{u}_k)$, and the Newton direction, $\mathbf{p}_k = -(\nabla^2 f(\mathbf{u}_k))^{-1} \nabla f(\mathbf{u}_k)$. The steepest descent method is simple (it only requires computation of the gradient) and is inexpensive per step, but its convergence can be very slow. Newton's method has a quadratic local convergence rate, but it is expensive per step (it requires computing the Hessian matrix and inverting it) and fast convergence only kicks in once iterates are sufficiently close to the solution. For both methods, the line search process is an essential 'globalization' mechanism that guides the algorithm to the solution, guarding against erratic behaviour and divergence. Line search procedures typically generate a limited number of trial step lengths until a step length is found that satisfies certain conditions on sufficient decrease in function value and gradient size. A well-known set of line search conditions are the so-called Wolfe conditions [2]. These line search conditions are also often used in convergence analysis. For example, when line searches are employed that satisfy the Wolfe conditions, global convergence can be proved for steepest descent and for Newton's method with suitably modified Hessian matrix (see Chapter 3 of [2]).

As an alternative to the full Newton method, so-called quasi-Newton methods attempt to attain a superlinear convergence rate while avoiding computation of the Hessian. In place of the true Hessian $\nabla^2 f(\mathbf{u}_k)$, they use an approximation that is updated after each step using gradient information, often satisfying a secant equation. One of the most popular methods in this class is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, which updates the Hessian with a rank-two matrix in every step. Storing the full Hessian is memory-inefficient for large problems, and a limited-memory version of BFGS (L-BFGS) has been developed which stores only a few vectors that implicitly represent the approximate Hessian. Another approach within the line search framework is provided by nonlinear versions of the conjugate gradient method, which was originally developed for solving symmetric positive definite systems of linear equations. Like our N-GMRES algorithm, N-CG is a generalization to nonlinear optimization of a Krylov method for linear equations. N-CG is attractive because it does not require matrix storage, and it improves significantly on steepest descent in terms of convergence speed.

An alternative to the line search globalization mechanism is provided by the trust region approach. This approach is not considered in this paper, but we mention it for completeness. In the trust region approach, a quadratic model function is constructed in each step whose behaviour near the current iterate $\mathbf{u}_k$ is similar to that of the objective function, $f(\mathbf{u})$. A minimizer of the quadratic model is sought in a trust region about $\mathbf{u}_k$ with radius Δ_k. If this minimizer does not produce a sufficient decrease, Δ_k is reduced, until a suitable minimizer $\mathbf{u}_{k+1}$ is found, after which the quadratic model function is computed for $\mathbf{u}_{k+1}$ in the next step of the iterative procedure. Popular approaches for building the quadratic model function are to use the Hessian (trust-region Newton method) or approximations to it (trust region quasi-Newton methods).

Our new N-GMRES method uses line search as the globalization mechanism. In this paper, we compare our N-GMRES method with the N-CG and L-BFGS line search methods, which are two well-established algorithms for nonlinear optimization that are widely used in practice.

1.2. The N-GMRES Optimization Algorithm from [1]

The N-GMRES optimization algorithm proposed in Paper I for accelerating ALS for canonical tensor approximation consists of three steps that can be summarized as follows. (Fig. 1 gives a schematic representation of the algorithm, and it is described in pseudo-code in Algorithm 1.) In the first step, a preliminary new iterate $\bar{\mathbf{u}}_{i+1}$ is generated from the last iterate $\mathbf{u}_i$ using a one-step iterative update process $M(.)$, which can be interpreted as a preconditioning process (see Paper I and below). ALS preconditioning is used for $M(.)$ in Paper I. In the second step, an accelerated iterate $\hat{\mathbf{u}}_{i+1}$ is obtained by linearly recombining previous iterates in a window of size w, $(\mathbf{u}_{i-w+1}, \ldots, \mathbf{u}_i)$, using a nonlinear GMRES approach. (The details of this step will be recalled in Section 2 below.) In the third step, a line search is performed that minimizes 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, to obtain the new iterate $\mathbf{u}_{i+1}$.

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 $M(.)$)*
    $\bar{\mathbf{u}}_{i+1} = M(\mathbf{u}_i)$
  STEP II: *(generate accelerated iterate by nonlinear GMRES step)*
    $\hat{\mathbf{u}}_{i+1} = \text{gmres}(\mathbf{u}_{i-w+1}, \ldots, \mathbf{u}_i; \bar{\mathbf{u}}_{i+1})$
  STEP III: *(generate new iterate by line search process)*
    if $\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}$ *is a descent direction*
      $\mathbf{u}_{i+1} = \text{linesearch}(\bar{\mathbf{u}}_{i+1} + \beta(\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}))$
    else
      $\mathbf{u}_{i+1} = \bar{\mathbf{u}}_{i+1}$
    end
  $i = i + 1$
until *convergence criterion satisfied*

(Note that the w initial iterates required in Algorithm 1 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. Also, as in [1], we perform a restart and reset the window size back to 1 whenever $\hat{\mathbf{u}}_{i+1} - \bar{\mathbf{u}}_{i+1}$ is not a descent direction.)

The second step in the N-GMRES optimization algorithm (Step II in Algorithm 1) uses the nonlinear extension of GMRES for solving nonlinear systems of equations that was proposed by Washio and Oosterlee in [3] in the context of nonlinear partial differential equation (PDE) systems (see also [4] and [5] for further applications to PDE systems). It is a nonlinear extension of the celebrated GMRES method for iteratively solving systems of linear equations [6, 7]. Washio and Oosterlee's nonlinear extension is related to Flexible GMRES as described in [8], and is also related to the reduced rank extrapolation method [9]. An early description of this

type of nonlinear iterate acceleration ideas for solving nonlinear equation systems appears in so-called Anderson mixing, see, e.g., [10, 11]. More recent applications of these ideas to nonlinear equation systems and fixed-point problems are discussed in [10, 11]. In Paper I we formulated a nonlinear GMRES optimization algorithm for canonical tensor decomposition that uses this type of acceleration as one of its steps, combined with an ALS preconditioning step and a line search for globalization. The type of nonlinear iterate acceleration in Step II of Algorithm 1 has thus been considered several times before in the context of solving nonlinear systems of equations, but we believe that its combination with a line search to obtain a general preconditioned nonlinear optimization method as in Algorithm 1 (see Paper I) is new in the optimization context. In the present paper we show how this N-GMRES optimization approach can be applied to a broad class of sufficiently smooth nonlinear optimization problems by using steepest descent preconditioning. We establish theoretical convergence properties for this approach and demonstrate its effectiveness in numerical tests.

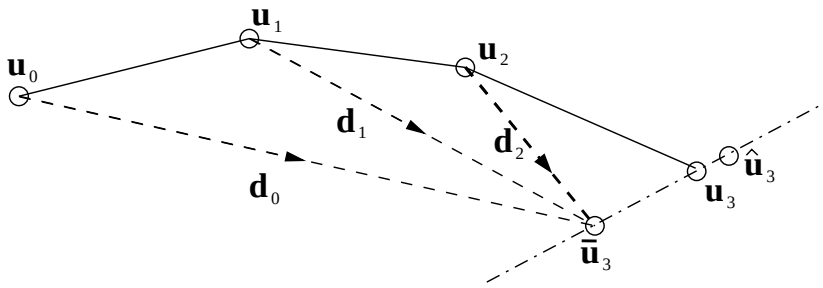

Figure 1. Schematic representation of one iteration of the N-GMRES optimization algorithm (from [1]), see Algorithm 1. Given previous iterations $\mathbf{u}_0$, $\mathbf{u}_1$ and $\mathbf{u}_2$, new iterate $\mathbf{u}_3$ is generated as follows. In Step I, preliminary iterate $\bar{\mathbf{u}}_3$ is generated by the one-step update process $M(.)$: $\bar{\mathbf{u}}_3 = M(\mathbf{u}_2)$. 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 $f(\bar{\mathbf{u}}_3 + \beta(\hat{\mathbf{u}}_3 - \bar{\mathbf{u}}_3))$.

The rest of this paper is structured as follows. In Section 2 we propose two types of steepest descent preconditioners for N-GMRES Optimization Algorithm 1. We briefly recall the details of the nonlinear GMRES optimization step, give a motivation and interpretation for steepest descent preconditioning that relate it to non-preconditioned GMRES for SPD linear systems, and give a simple proof for global convergence of the N-GMRES optimization algorithm using steepest descent preconditioning with line search. In Section 3 we present extensive numerical results for N-GMRES optimization with the two proposed steepest descent preconditioners, applied to a variety of nonlinear optimization problems, and compare with stand-alone steepest descent, N-CG and L-BFGS. Finally, Section 4 concludes.

2. Steepest Descent Preconditioning for N-GMRES Optimization

In this section, we first propose two variants of steepest descent preconditioning. We then briefly recall the details of the nonlinear GMRES recombination step (Step II in Algorithm 1), and relate N-GMRES optimization to standard non-preconditioned GMRES for linear systems

in the case of a simple quadratic optimization problem with SPD operator. Finally, we give a simple global convergence proof for the N-GMRES optimization algorithm using steepest descent preconditioning with line search.

2.1. Steepest Descent Preconditioning Process

We propose a general steepest descent preconditioning process for Step I of N-GMRES Optimization Algorithm 1 with the following two variants:

STEEPEST DESCENT PRECONDITIONING PROCESS:

$$\bar{\mathbf{u}}_{i+1} = \mathbf{u}_i - \beta \frac{\nabla f(\mathbf{u}_i)}{\|\nabla f(\mathbf{u}_i)\|} \quad \text{with}$$

$$\text{OPTION A:} \qquad \beta = \beta_{sdls}, \tag{2.1}$$

$$\text{OPTION B:} \qquad \beta = \beta_{sd} = \min(\,\delta\,,\, \|\nabla f(\mathbf{u}_i)\|\,). \tag{2.2}$$

For Option A, β_{sdls} is the step length obtained by a line search procedure. For definiteness, we consider a line search procedure that satisfies the Wolfe conditions (see below). We refer to the steepest descent preconditioning process with line search (2.1) as the *sdls* preconditioner. For Option B, we predefine the step β_{sd} as the minimum of a small positive constant δ, and the norm of the gradient. In the numerical results to be presented further on in the paper, we use $\delta = 10^{-4}$, except where noted. We refer to the steepest descent preconditioning process with predefined step β_{sd} (2.2) as the *sd* preconditioner. These two Options are quite different, and some discussion is in order.

Preconditioning process A can be employed as a stand-alone optimization method (it can converge by itself), and N-GMRES can be considered as a wrapper that accelerates this stand-alone process. We will show below that N-GMRES with preconditioning process A has strong convergence properties, but it may be expensive because the line search may require a significant number of function and gradient (f/g) evaluations. However, the situation is very different for preconditioning process B. Here, no additional f/g evaluations are required, but convergence appears questionable. It is clear that preconditioning process B cannot be used as a stand-alone optimization algorithm; in most cases it would not converge. It can, however, still be used as a preconditioning process for N-GMRES. As is well-known and will be further illustrated below, preconditioners used by GMRES for linear systems do not need to be convergent by themselves, and this suggests that it may be interesting to consider this for N-GMRES optimization as well. As will be motivated further below, the role of the N-GMRES preconditioning process is to provide new 'useful' directions for the nonlinear generalization of the Krylov space, and the iteration can be driven to convergence by the N-GMRES minimization, even if the preconditioner is not convergent by itself. However, for this to happen in the three-step N-GMRES optimization algorithm with preconditioning process B, it is required that $\bar{\mathbf{u}}_{i+1}$ eventually approaches $\mathbf{u}_i$ and the step length β_{sd} approaches 0. For this reason, we select $\beta_{sd} = \|\nabla f(\mathbf{u}_i)\|$ as soon as $\|\nabla f(\mathbf{u}_i)\| \leq \delta$. The initial step length β_{sd} is chosen to be not larger than a small constant because the linear case (see below) suggests that a small step is sufficient to provide a new direction for the Krylov space, and because the minimization of the residual is based on a linearization argument (see also below), and small steps tend to lead to small linearization errors.

2.2. N-GMRES Recombination Step

Before relating steepest-descent preconditioned N-GMRES to non-preconditioned GMRES for linear systems, we first recall from [1] some details of the N-GMRES recombination step, Step II in Algorithm 1. In this step, we find an accelerated iterate $\hat{\mathbf{u}}_{i+1}$ that is obtained by recombining previous iterates as follows:

$$\hat{\mathbf{u}}_{i+1} = \bar{\mathbf{u}}_{i+1} + \sum_{j=0}^{i} \alpha_j \left(\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j\right). \tag{2.3}$$

The unknown coefficients α_j are determined by the N-GMRES algorithm in such a way that the two-norm of the gradient of the objective function evaluated at the accelerated iterate is small, in the following sense. In general, $\mathbf{g}(.)$ is a nonlinear function of the α_j, and linearization is used to allow for inexpensive computation of coefficients α_j that may approximately minimize $\|\mathbf{g}(\hat{\mathbf{u}}_{i+1})\|_2$. Using the approximations

$$\begin{aligned}
\mathbf{g}(\hat{\mathbf{u}}_{i+1}) &\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \left.\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right|_{\bar{\mathbf{u}}_{i+1}} \alpha_j \left(\bar{\mathbf{u}}_{i+1} - \mathbf{u}_j\right) \\
&\approx \mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \alpha_j \left(\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j)\right)
\end{aligned} \tag{2.4}$$

one arrives at minimization problem

$$\text{find coefficients } (\alpha_0, \ldots, \alpha_i) \text{ that minimize}$$
$$\|\mathbf{g}(\bar{\mathbf{u}}_{i+1}) + \sum_{j=0}^{i} \alpha_j \left(\mathbf{g}(\bar{\mathbf{u}}_{i+1}) - \mathbf{g}(\mathbf{u}_j)\right)\|_2. \tag{2.5}$$

This is a standard least-squares problem that can be solved, for example, by using the normal equations, as explained in [3, 1]. (In this paper, we solve the least-squares problem as described in [1].) In a windowed implementation with window size w, the memory cost incurred by N-GMRES acceleration is the storage of w previous approximations and residuals. The dominant parts of the CPU cost for each acceleration step are the cost of building and solving the least-squares system (which can be done in approximately $2nw$ flops if the normal equations are used and some previous inner products are stored, see [3]), and nw flops to compute the accelerated iterate. For problems with expensive objective functions, this cost is often negligible compared to the cost of the f/g evaluations in the line searches [1]. An alternative to the normal equations approach would be to use QR decomposition, which has better stability properties. As is explained in [11] for a similar algorithm applied to fixed-point iterations, computations can be organized such that information can be reused from previous steps: the QR factorization in step i can efficiently be obtained from the QR factorization in step $i-1$ in $O(nw)$ operations, using standard QR factor-updating techniques.

2.3. Motivation and Interpretation for Steepest Descent Preconditioning

Consider a standard quadratic minimization problem with objective function

$$f(\mathbf{u}) = \frac{1}{2}\,\mathbf{u}^T A \mathbf{u} - \mathbf{b}^T \mathbf{u}, \tag{2.6}$$

where A is SPD. It is well-known that its unique minimizer satisfies $A\mathbf{u} = \mathbf{b}$. Now consider applying the N-GMRES optimization algorithm with steepest descent preconditioner to the quadratic minimization problem. The gradient of f at approximation $\mathbf{u}_i$ is given by

$$\nabla f(\mathbf{u}_i) = A\mathbf{u}_i - \mathbf{b} = -\mathbf{r}_i \quad \text{with} \quad \mathbf{r}_i = \mathbf{b} - A\mathbf{u}_i, \tag{2.7}$$

where $\mathbf{r}_i$ is defined as the residual of the linear system $A\mathbf{u} = \mathbf{b}$ in $\mathbf{u}_i$. N-GMRES steepest descent preconditioner (2.1)-(2.2) then reduces to the form

$$\bar{\mathbf{u}}_{i+1} = \mathbf{u}_i + \beta \frac{\mathbf{r}_i}{\|\mathbf{r}_i\|}, \tag{2.8}$$

where $\bar{\mathbf{u}}_{i+1}$ is the preliminary new iterate, see Fig. 1, and it can easily be shown that this corresponds to the stationary iterative method that generates the Krylov space in non-preconditioned linear GMRES applied to $A\mathbf{u} = \mathbf{b}$. We now briefly show this because it provides further insight (recalling parts of the discussion in [3, 1]). (Note also that GMRES reduces to the MINRES algorithm in the case of SPD matrices, see, for example, [6].)

We first explain how preconditioned GMRES for $A\mathbf{u} = \mathbf{b}$ works. Consider so-called stationary iterative methods for $A\mathbf{u} = \mathbf{b}$ of the following form:

$$\mathbf{u}_{i+1} = \mathbf{u}_i + M^{-1}\,\mathbf{r}_i. \tag{2.9}$$

Here, matrix M is an approximation of A that has an easily computable inverse, i.e., $M^{-1} \approx A^{-1}$. For example, M can be chosen to correspond to Gauss-Seidel or Jacobi iteration, or to a multigrid cycle [3].

Consider a sequence of iterates $\mathbf{u}_0, \ldots, \mathbf{u}_i$ generated by update formula (2.9), starting from some initial guess $\mathbf{u}_0$. Note that the residuals of these iterates are related as

$$\mathbf{r}_i = \mathbf{b} - A\,\mathbf{u}_i = (I - AM^{-1})\,\mathbf{r}_{i-1} = (I - AM^{-1})^i\,\mathbf{r}_0. \tag{2.10}$$

This motivates the definition of the following vector spaces:

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

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 [3]:

Lemma 2.1
$V_{1,i+1} = V_{2,i+1} = V_{3,i+1} = V_{4,i+1}$.

Proof. First, $V_{1,i+1} = V_{2,i+1}$ by (2.10), which directly shows that $\mathbf{r}_j \in V_{2,i+1}$ for all j, and $(\mathbf{A}\mathbf{M}^{-1})^j\,\mathbf{r}_0 \in V_{1,i+1}$ for all j follows by recursion.
Second, $V_{1,i+1} = V_{3,i+1}$ because $M\,(\mathbf{u}_{i+1} - \mathbf{u}_i) = \mathbf{r}_i$, by (2.9).
Third, $V_{3,i+1} = V_{4,i+1}$ because, for $k < i+1$, $\mathbf{u}_{i+1} - \mathbf{u}_k = \sum_{j=k+1}^{i+1}(\mathbf{u}_j - \mathbf{u}_{j-1})$, and $\mathbf{u}_k - \mathbf{u}_{k-1} = (\mathbf{u}_{i+1} - \mathbf{u}_{k-1}) - (\mathbf{u}_{i+1} - \mathbf{u}_k)$. □

Expression (2.9) shows that $M(\mathbf{u}_{i+1} - \mathbf{u}_i) \in K_{i+1}(AM^{-1}, \mathbf{r}_0)$. The GMRES procedure can be seen as a way to accelerate stationary iterative method (2.9), by recombining iterates (or, equivalently, by reusing residuals). In particular, we seek a better approximation $\hat{\mathbf{u}}_{i+1}$, with $M(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i)$ in the Krylov space $K_{i+1}(AM^{-1}, \mathbf{r}_0)$, such that $\hat{\mathbf{r}}_{i+1} = \mathbf{b} - A\,\hat{\mathbf{u}}_{i+1}$ has minimal two-norm. In other words, we seek optimal coefficients β_j in

$$M(\hat{\mathbf{u}}_{i+1} - \mathbf{u}_i) = \sum_{j=0}^{i} \beta_j\, M(\mathbf{u}_{i+1} - \mathbf{u}_j),$$

and it is easy to show that this corresponds to seeking optimal coefficients α_j in

$$\hat{\mathbf{u}}_{i+1} = \mathbf{u}_{i+1} + \sum_{j=0}^{i} \alpha_j\,(\mathbf{u}_{i+1} - \mathbf{u}_j), \tag{2.11}$$

such that $\|\hat{\mathbf{r}}_{i+1}\|_2$ is minimized (which leads to a small least-squares problem equivalent to (2.5)). Note that $V_{1,i+1}$ and $V_{2,i+1}$ do not easily generalize to the nonlinear case, but the image of $V_{4,i+1}$ under M^{-1}, $span\{\mathbf{u}_{i+1}-\mathbf{u}_0, \mathbf{u}_{i+1}-\mathbf{u}_1, \ldots, \mathbf{u}_{i+1}-\mathbf{u}_i\}$, does generalize naturally and is taken as the 'generalized Krylov space' that is used to seek the approximation in the nonlinear case.

Up to this point, we have presented GMRES as a way to accelerate one-step stationary iterative method (2.9). A more customary way, however, to see GMRES is in terms of preconditioning. The approach described above reduces to 'non-preconditioned' GMRES when one sets $M = I$. Applying non-preconditioned GMRES to the preconditioned linear equation system $AM^{-1}(M\mathbf{u}) = \mathbf{b}$ also results in the expressions for preconditioned GMRES derived above. In this viewpoint, the matrix M^{-1} is called the preconditioning matrix, because its role is viewed as to pre-condition the spectrum of the linear system operator such that the (non-preconditioned) GMRES method applied to $(AM^{-1})\mathbf{y} = \mathbf{b}$ becomes more effective. It is also sometimes said that the stationary iterative process preconditions GMRES (for example, Gauss-Seidel, Jacobi or multigrid can precondition GMRES [3]). We can summarize that the role of the stationary iterative method is to generate preconditioned residuals that build the Krylov space. (Note that considering right-preconditioning [7] leads to the desired equivalence here, similar as in the case of Flexible GMRES [8], where right-preconditioning enables the use of preconditioners that vary from step to step, as in the case of N-GMRES.)

In the presentation above, all iterates $\mathbf{u}_j$ for $j = 0, \ldots, i$ (for instance, in the right-hand side of (2.11)) refer to the unaccelerated iterates generated by stationary iterative method (2.9). However, the formulas remain valid when accelerated iterates are used instead; this does change the values of the coefficients α_j, but leads to the same accelerated iterates [3]. This is so because the Krylov spaces generated in the two cases are identical due to linearity (see (2.10)), and consequently GMRES selects the same optimal improved iterate.

This brings us to the point where we can compare steepest-descent preconditioned N-GMRES applied to quadratic objective function (2.6) with SPD operator A, to non-preconditioned linear GMRES applied to $A\mathbf{u} = \mathbf{b}$. Assume we have w previous iterates $\mathbf{u}_i$ and residuals $\mathbf{r}_i$. Stationary iterative process (2.9) without preconditioner ($M = I$) would add a vector to the Krylov space which has the same direction as the vector that would be added to it by the steepest descent preconditioning process (2.8). This means that the

accelerated iterate $\hat{\mathbf{u}}_{i+1}$ produced by N-GMRES with steepest descent preconditioner applied to quadratic objective function (2.6) with SPD operator A is the same as the accelerated iterate $\hat{\mathbf{u}}_{i+1}$ produced by linear GMRES with identity preconditioner applied to $A\mathbf{u} = \mathbf{b}$. This motivates our proposal to use steepest descent preconditioning as the natural and most basic preconditioning process for the N-GMRES optimization algorithm applied to general nonlinear optimization problems.

Note that, in the case of linear systems, the efficiency of GMRES as an acceleration technique for stationary iterative methods can be understood in terms of how optimal polynomials can damp modes that are slow to converge [3, 7]. In the case of N-GMRES for nonlinear optimization, if the approximation is close to a stationary point and the nonlinear residual vector function $\mathbf{g}(.)$ can be approximated well by linearization, then it can be expected that the use of the subspace $span\{\mathbf{u}_{i+1} - \mathbf{u}_0, \mathbf{u}_{i+1} - \mathbf{u}_1, \ldots, \mathbf{u}_{i+1} - \mathbf{u}_i\}$ for acceleration may give efficiency similar to the linear case [3]. Also, restarting N-GMRES is analogous to restarting GMRES [7]. Note finally that the above also explains why a small step is allowed in the *sd* preconditioner of (2.2) (basically, in the linear case, the size of the coefficient does not matter for the Krylov space, in exact arithmetic), and the linearization argument of (2.4) indicates that a small step may be beneficial.

2.4. Convergence Theory for N-GMRES Optimization with Steepest Descent Preconditioning

We now formulate and prove a convergence theorem for N-GMRES Optimization Algorithm 1 using steepest descent preconditioning with line search (2.1). We assume that all line searches provide step lengths that satisfy the Wolfe conditions [2]:

SUFFICIENT DECREASE CONDITION:

$$f(\mathbf{u}_i + \beta_i \mathbf{p}_i) \leq f(\mathbf{u}_i) + c_1\, \beta_i\, \nabla f(\mathbf{u}_i)^T \mathbf{p}_i, \tag{2.12}$$

CURVATURE CONDITION:

$$\nabla f(\mathbf{u}_i + \beta_i \mathbf{p}_i)^T\, \mathbf{p}_i \geq c_2\, \nabla f(\mathbf{u}_i)^T \mathbf{p}_i, \tag{2.13}$$

with $0 < c_1 < c_2 < 1$. Condition (2.12) ensures that large steps are taken only if they lead to a proportionally large decrease in f. Condition (2.13) ensures that a step is taken that is large enough to sufficiently increase the gradient of f in the line search direction (make it less negative). Global convergence (in the sense of convergence to a stationary point from any initial guess) can then be proved easily using standard approaches [12, 2].

Theorem 2.2 (Global convergence of N-GMRES optimization algorithm)
Consider N-GMRES Optimization Algorithm 1 with steepest descent line search preconditioning (2.1) for Optimization Problem I, and assume that all line search solutions satisfy the Wolfe conditions, (2.12) and (2.13). Assume that objective function f is bounded below in $\mathbb{R}^n$ and that f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} = \{\mathbf{u} : f(\mathbf{u}) \leq f(\mathbf{u}_0)\}$, where $\mathbf{u}_0$ is the starting point of the iteration. Assume also that the gradient ∇f is Lipschitz continuous on $\mathcal{N}$, that is, there exists a constant L such that $\|\nabla f(\mathbf{u}) - \nabla f(\hat{\mathbf{u}})\| \leq L\|\mathbf{u} - \hat{\mathbf{u}}\|$ for all $\mathbf{u}, \hat{\mathbf{u}} \in \mathcal{N}$. Then the sequence of N-GMRES iterates $\{\mathbf{u}_0, \mathbf{u}_1, \ldots\}$ is convergent to a fixed point of Optimization Problem I in the sense that

$$\lim_{i \to \infty} \|\nabla f(\mathbf{u}_i)\| = 0. \tag{2.14}$$

Proof. Consider the sequence $\{\mathbf{v}_0, \mathbf{v}_1, \ldots\}$ formed by the iterates $\mathbf{u}_0$, $\bar{\mathbf{u}}_1$, $\mathbf{u}_1$, $\bar{\mathbf{u}}_2$, $\mathbf{u}_2$, ... of Algorithm I, but with $\bar{\mathbf{u}}_i$ removed if $\hat{\mathbf{u}}_i - \bar{\mathbf{u}}_i$ is not a descent direction in Step III of the algorithm ($\bar{\mathbf{u}}_i$ equals $\mathbf{u}_i$ in this case). Then all iterates $\mathbf{v}_i$ are of the form $\mathbf{v}_i = \mathbf{v}_{i-1} + \beta_{i-1}\mathbf{p}_{i-1}$, with $\mathbf{p}_{i-1}$ a descent direction and β_{i-1} such that the Wolfe conditions are satisfied. According to Theorem 3.2 of [2] (p. 38, Zoutendijk's Theorem), we have that

$$\sum_{i=0}^{\infty} \cos^2\theta_i \, \|\nabla f(\mathbf{v}_i)\|^2 < \infty, \tag{2.15}$$

with

$$\cos\theta_i = \frac{-\nabla f(\mathbf{v}_i)^T \mathbf{p}_i}{\|\nabla f(\mathbf{v}_i)\| \, \|\mathbf{p}_i\|}, \tag{2.16}$$

which implies that

$$\lim_{i\to\infty} \cos^2\theta_i \, \|\nabla f(\mathbf{v}_i)\|^2 = 0. \tag{2.17}$$

Consider the subsequence $\{\|\nabla f(\mathbf{u}_i)\|\}$ of $\{\|\nabla f(\mathbf{v}_i)\|\}$. Since all the $\mathbf{u}_i$ are followed by a steepest descent step in the algorithm, the θ_i corresponding to all the elements of $\{\|\nabla f(\mathbf{u}_i)\|\}$ satisfy $\cos\theta_i = 1$. Therefore, it follows from (2.17) that $\lim_{i\to\infty} \|\nabla f(\mathbf{u}_i)\| = 0$, which concludes the proof. □

Note that the notion of convergence (2.14) we prove in Theorem 2.2 for N-GMRES optimization with steepest descent line search preconditioning is stronger than the type of convergence that can be proved for some N-CG methods [12, 2], namely,

$$\lim_{i\to\infty} \inf \|\nabla f(\mathbf{u}_i)\| = 0. \tag{2.18}$$

Note also that in Theorem 2.2 we do not prove that sequence $\{\|\nabla f(\bar{\mathbf{u}}_i)\|\}$ converges to 0. It may be possible to prove this under the stated conditions by using more advanced tools, but it is also possible that stronger conditions are required to guarantee that $\{\|\nabla f(\bar{\mathbf{u}}_i)\|\} \to 0$. In any case, we are able to prove the strong global convergence result (2.14) for the iterates $\mathbf{u}_i$ of N-GMRES optimization with steepest descent line search preconditioning under the conditions of Theorem 2.2: sequence $\{\|\nabla f(\mathbf{u}_i)\|\}$ converges to 0.

3. Numerical Results

We now present extensive numerical results for the N-GMRES optimization algorithm with steepest descent preconditioners (2.1) and (2.2), compared with stand-alone steepest descent optimization, N-CG and L-BFGS.

In all tests, we utilize the Moré-Thuente line search method [13] and the N-CG and L-BFGS optimization methods as implemented in the Poblano toolbox for MATLAB [14]. For all experiments, the Moré-Thuente line search parameters used were as follows: function value tolerance $c_1 = 10^{-4}$ for (2.12), gradient norm tolerance $c_2 = 10^{-2}$ for (2.13), starting search step length $\beta = 1$, and a maximum of 20 f/g evaluations are used. These values were also used for the N-CG and L-BFGS comparison runs. Note that the Moré-Thuente line search is designed to compute a line search step length that satisfies the Wolfe conditions. We use

the N-CG variant with Polak-Ribière update formula, and the two-loop recursion version of L-BFGS [2]. 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. The L-BFGS window size is chosen equal to 5 (we found that larger window sizes tend to harm L-BFGS performance for the tests we considered). All initial guesses are determined uniformly randomly with components in the interval $[0, 1]$, and when we compare different methods they are given the same random initial guess. All numerical tests were run on a laptop with a dual-core 2.53 GHz Intel Core i5 processor and 4GB of 1067 MHz DDR3 memory. MATLAB version 7.11.0.584 (R2010b) 64-bit (maci64) was used for all tests.

3.1. Test Problem Description

We first describe the seven test problems we consider. In what follows, all vectors are chosen in $\mathbb{R}^n$, and all matrices in $\mathbb{R}^{n\times n}$.

PROBLEM A. (QUADRATIC OBJECTIVE FUNCTION WITH SPD DIAGONAL MATRIX.)

$$f(\mathbf{u}) = \frac{1}{2}\,(\mathbf{u}-\mathbf{u}^*)^T D\,(\mathbf{u}-\mathbf{u}^*) + 1, \qquad \text{with } D = \operatorname{diag}(1, 2, \ldots, n). \tag{3.1}$$

This problem has a unique minimizer $\mathbf{u}^*$ in which $f^* = f(\mathbf{u}^*) = 1$. We choose $\mathbf{u}^* = (1, \ldots, 1)$. Note that $\mathbf{g}(\mathbf{u}) = D(\mathbf{u}-\mathbf{u}^*)$, and the condition number of D is given by $\kappa = n$. It is well-known that for problems of this type large condition numbers tend to lead to slow convergence of the steepest descent method due to a zig-zag effect. Problem A can be used to show how methods like N-CG and N-GMRES improve over steepest descent and mitigate this zig-zag effect.

PROBLEM B. (PROBLEM A WITH PARABOLOID COORDINATE TRANSFORMATION.)

$$f(\mathbf{u}) = \frac{1}{2}\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*)^T D\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*) + 1, \qquad \text{with } D = \operatorname{diag}(1, 2, \ldots, n) \text{ and } \mathbf{y}(\mathbf{x}) \text{ given by} \quad y_1(\mathbf{x}) = x_1 \text{ and } y_i(\mathbf{x}) = x_i - 10\,x_1^2 \ (i = 2, \ldots, n). \tag{3.2}$$

This modification of Problem A still has a unique minimizer $\mathbf{u}^*$ in which $f^* = f(\mathbf{u}^*) = 1$. We choose $\mathbf{u}^* = (1, \ldots, 1)$. The gradient of $f(\mathbf{u})$ is given by $\mathbf{g}(\mathbf{u}) = D\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*) - 20\,(u_1 - u_1^*)\,(\sum_{j=2}^{n}(D\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*))_j)\,[1, 0, \ldots, 0]^T$. This modification of Problem A increases nonlinearity (the objective function is now quartic in $\mathbf{u}$) and changes the level surfaces from ellipsoids into parabolically skewed ellipsoids. As such, the problem is more difficult for nonlinear optimization methods. For $n = 2$, the level curves are modified from elliptic to 'banana-shaped'. In fact, the objective function of Problem B is a multi-dimensional generalization of Rosenbrock's 'banana' function.

PROBLEM C. (PROBLEM B WITH A RANDOM NON-DIAGONAL MATRIX WITH CONDITION

NUMBER $\kappa = n$.)

$$f(\mathbf{u}) = \frac{1}{2}\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*)^T T\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*) + 1, \tag{3.3}$$
$$\text{with } T = Q\,\mathrm{diag}(1,2,\ldots,n)\,Q^T, \text{ where } Q \text{ is a}$$
$$\text{random orthogonal matrix and } \mathbf{y}(\mathbf{x}) \text{ is given by}$$
$$y_1(\mathbf{x}) = x_1 \text{ and } y_i(\mathbf{x}) = x_i - 10\,x_1^2 \ (i = 2,\ldots,n).$$

This modification of Problem B still has a unique minimizer $\mathbf{u}^*$ in which $f^* = f(\mathbf{u}^*) = 1$. We choose $\mathbf{u}^* = (1,\ldots,1)$. The gradient of $f(\mathbf{u})$ is given by $\mathbf{g}(\mathbf{u}) = T\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*) - 20\,(u_1 - u_1^*)\,(\sum_{j=2}^n (T\,\mathbf{y}(\mathbf{u}-\mathbf{u}^*))_j)\,[1,0,\ldots,0]^T$. The random matrix Q is the Q factor obtained from a QR-factorization of a random matrix with elements uniformly drawn from the interval $[0,1]$. This modification of Problem B introduces nonlinear 'mixing' of the coordinates (cross-terms) and further increases the difficulty of the problem.

PROBLEM D. (EXTENDED ROSENBROCK FUNCTION, PROBLEM (21) FROM [15].)

$$f(\mathbf{u}) = \frac{1}{2}\sum_{j=1}^{n} t_j^2(\mathbf{u}), \text{ with } n \text{ even and}$$
$$t_j = 10\,(u_{j+1} - u_j^2) \qquad (j \text{ odd}),$$
$$t_j = 1 - u_{j-1} \qquad (j \text{ even}).$$

Note that $\mathbf{g}(\mathbf{u})$ can easily be computed using $g_k(\mathbf{u}) = \sum_{j=1}^n t_j\,\partial t_j/\partial u_k$ $(k = 1,\ldots,n)$.

PROBLEM E. (EXTENDED POWELL SINGULAR FUNCTION, PROBLEM (22) FROM [15].)

$$f(\mathbf{u}) = \frac{1}{2}\sum_{j=1}^{n} t_j^2(\mathbf{u}), \text{ with } n \text{ a multiple of 4 and} \tag{3.4}$$
$$t_{4i-3} = u_{4i-3} + 10\,u_{4i-2},$$
$$t_{4i-2} = \sqrt{5}\,(u_{4i-1} - u_{4i}),$$
$$t_{4i-1} = (u_{4i-2} - 2\,u_{4i-1})^2,$$
$$t_{4i} = \sqrt{10}\,(u_{4i-3} - u_{4i})^2 \quad \text{for } i = 1,\ldots,n/4.$$

PROBLEM F. (TRIGONOMETRIC FUNCTION, PROBLEM (26) FROM [15].)

$$f(\mathbf{u}) = \frac{1}{2}\sum_{j=1}^{n} t_j^2(\mathbf{u}), \text{ with}$$
$$t_j = n - \left(\sum_{i=1}^{n} \cos u_i\right) - j\,(1 - \cos u_j) - \sin u_j.$$

PROBLEM G. (PENALTY FUNCTION I, PROBLEM (23) FROM [15].)

$$f(\mathbf{u}) = \frac{1}{2}\left(\left(\sum_{j=1}^{n} t_j^2(\mathbf{u})\right) + t_{n+1}^2(\mathbf{u})\right), \text{ with}$$
$$t_j = \sqrt{10^{-5}}\,(u_j - 1) \qquad (j = 1, \ldots, n),$$
$$t_{n+1} = \left(\sum_{i=1}^{n} u_i^2\right) - 0.25.$$

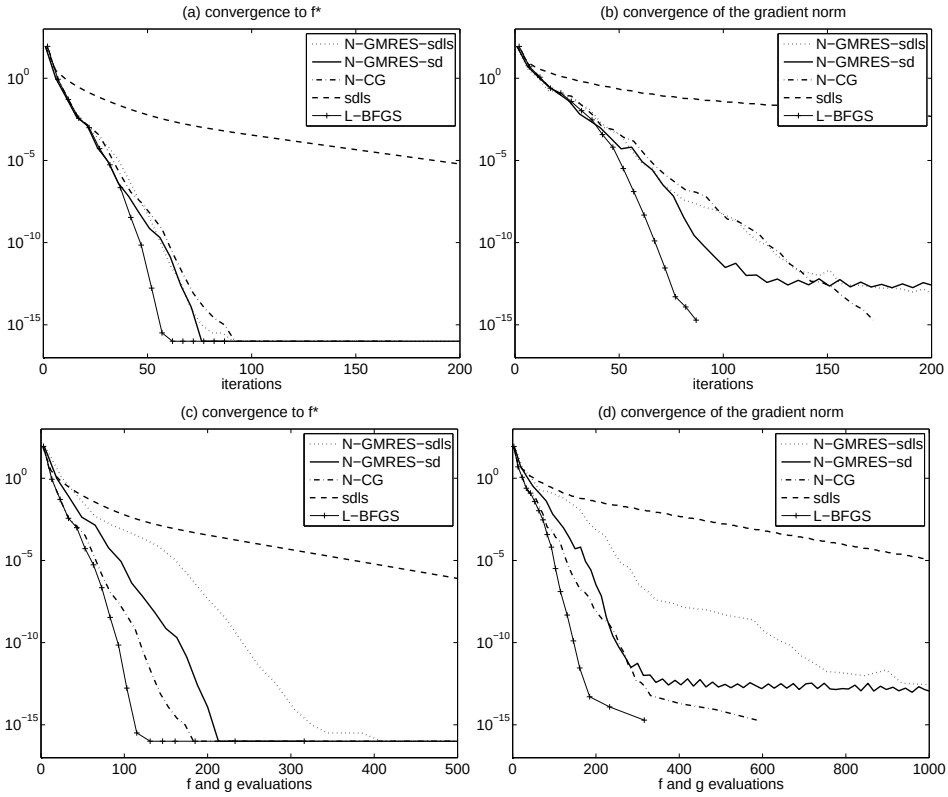

Figure 2. Problem A ($n = 100$). Convergence histories of the 10-logarithms of $|f(\mathbf{u}_i) - f^*|$ and $\|\mathbf{g}(\mathbf{u}_i)\|$ as a function of iterations and f/g evaluations. N-GMRES-sdls is the N-GMRES optimization algorithm using steepest descent preconditioning with line search, N-GMRES-sd is the N-GMRES optimization algorithm using steepest descent preconditioning with predefined step, N-CG is the Polak-Ribière nonlinear conjugate gradient method, L-BFGS is the limited-memory Broyden-Fletcher-Goldfarb-Shanno method, and sdls is the stand-alone steepest descent method with line search.

3.2. Numerical Results for Problems A–C

Before presenting average performance comparisons for the different methods applied to Problems A–C in Table I, we first present some convergence plots for instances of Problems

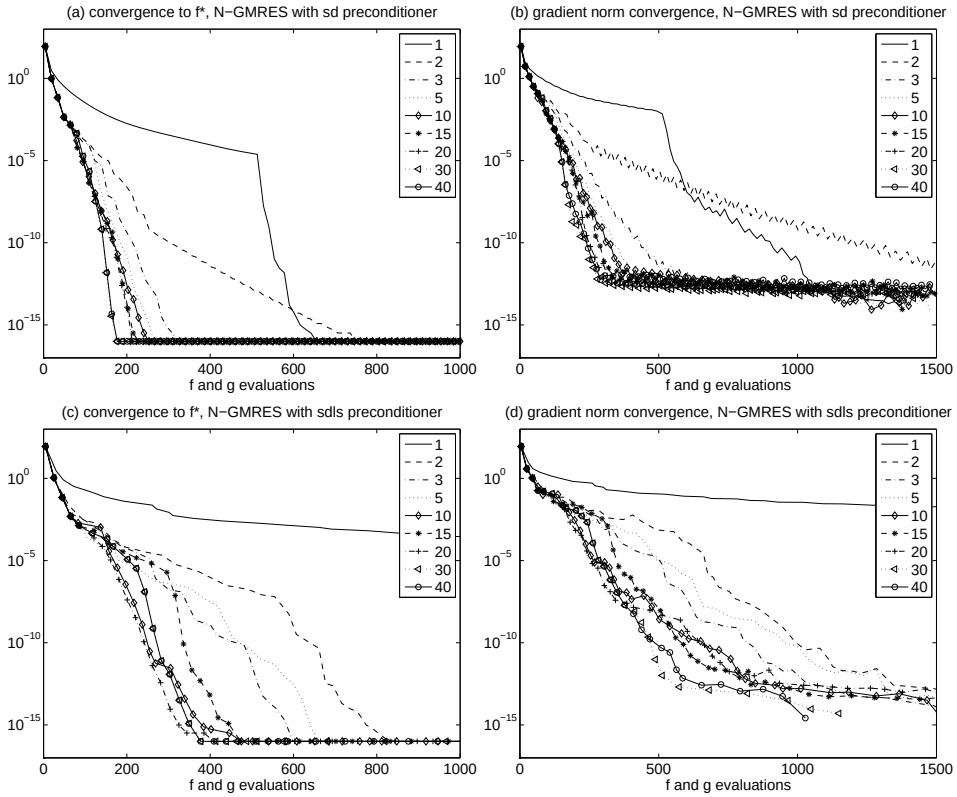

Figure 3. Problem A ($n = 100$). Effect of varying window size w on $|f(\mathbf{u}_i) - f^*|$ and $\|\mathbf{g}(\mathbf{u}_i)\|$ convergence for N-GMRES-sdls and N-GMRES-sd optimization as a function of f/g evaluations. Window size $w = 20$ emerges as a suitable choice, leading to rapid convergence. These results give some general indication that, if sufficient memory is available, $w = 20$ may be a good choice. However, if memory is scarce, $w = 3$ already provides good results, especially for N-GMRES-sd.

A–C. Fig. 2 shows results for an instance of Problem A. (To make the plots less dense and avoid cluttered plotting symbols, we only plot data points corresponding to every fifth iteration.) We see that stand-alone steepest descent with line search (sdls) converges slowly, which is expected because the condition number of matrix D is $\kappa = 100$. Both N-GMRES optimization using steepest descent preconditioning with line search (2.1) (N-GMRES-sdls) and N-GMRES optimization using steepest descent preconditioning with predefined step (2.2) (N-GMRES-sd) are significantly faster than stand-alone sdls, in terms of iterations and f/g evaluations, confirming that the N-GMRES acceleration mechanism is effective, and steepest descent is an effective preconditioner for it. As could be expected, the preconditioning line searches of N-GMRES-sdls add significantly to its f/g evaluation cost, and N-GMRES-sd is more effective. N-GMRES accelerates steepest descent up to a point where performance becomes competitive with N-CG and L-BFGS. It is important to note that convergence profiles like the ones presented in Fig. 2 tend to show significant variation depending on the random initial guess. The instances presented are arbitrary and not hand-picked with a special purpose in

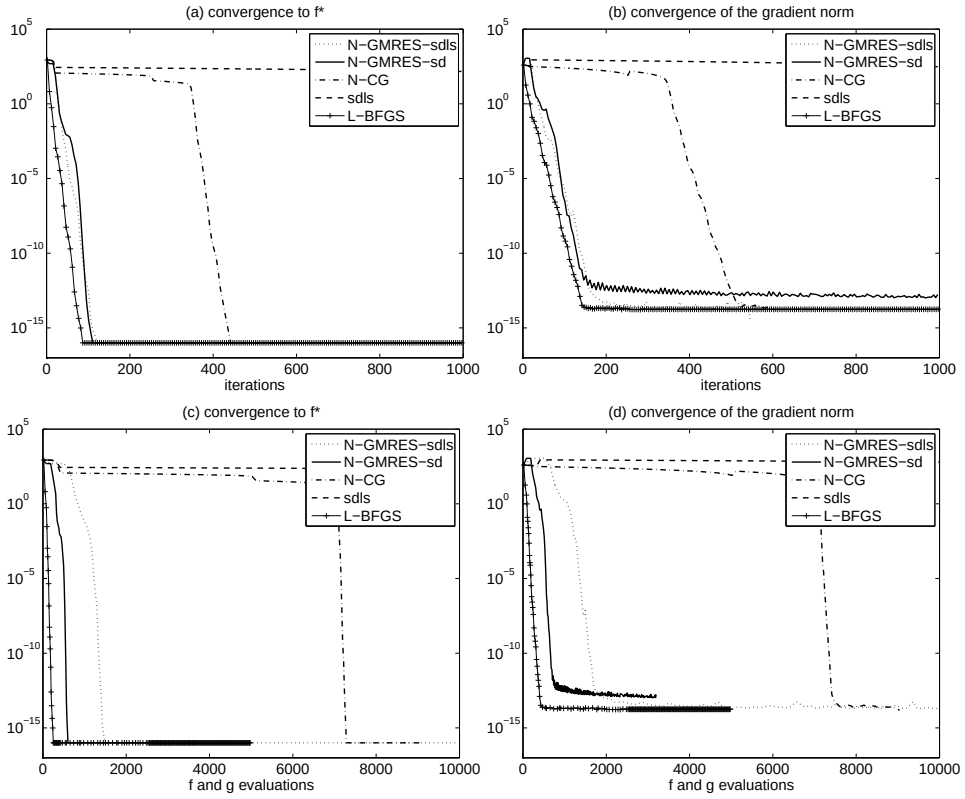

Figure 4. Problem C ($n = 100$). Convergence comparison.

mind (they simply correspond to random seed 0 in our MATLAB code) and we show them because they do provide interesting illustrations and show patterns that we have verified to be quite general over many random instances. However, they cannot reliably be used to conclude on detailed relative performance of various methods. For this purpose, we provide tables below that compare performance averaged over a set of random trials.

Fig. 3 shows the effect of varying the window size w on $|f(\mathbf{u}_i) - f^*|$ and $\|\mathbf{g}(\mathbf{u}_i)\|$ convergence for N-GMRES-sdls and N-GMRES-sd optimization as a function of f/g evaluations, for an instance of Problem A. Window size $w = 20$ emerges as a suitable choice if sufficient memory is available, leading to rapid convergence. However, window sizes as small as $w = 3$ already provide good results, especially for N-GMRES-sd. This indicates that satisfactory results can be obtained with small windows, which may be useful if memory is scarce. We use window size $w = 20$ for all numerical results in this paper.

Fig. 4 shows results for an instance of Problem C, which is a modification of Problem A introducing a nonlinear coordinate transformation (as in Problem B) and random nonlinear mixing of the coordinate directions. The figure shows that stand-alone sdls is very slow, confirms that N-GMRES-sdls and N-GMRES-sd significantly speed up steepest descent, and shows that N-GMRES-sd and L-BFGS perform much better than N-CG for this problem.

problem	N-GMRES-sdls	N-GMRES-sd	N-CG	L-BFGS
A n=100	242	111	84	73
A n=200	406	171	127	104
B n=100	1200	395	198	170
B n=200	1338	752	606	321
C n=100	926(1)	443	13156(7)	151
C n=200	1447	461	26861(9)	204

Table I. Average number of f/g evaluations needed to reach $|f(\mathbf{u}_i) - f^*| < 10^{-6}$ for 10 instances of Problems A–C with random initial guess and with different sizes. Numbers in brackets give the number of random trials (out of 10) that did not converge to the required tolerance within 1500 iterations (if any).

Table I confirms the trends that were already present in the specific instances of test problems A and C that were shown in Figures 2 and 4. The table gives the average number of f/g evaluations that were needed to reach $|f(\mathbf{u}_i) - f^*| < 10^{-6}$ for 10 random instances of Problems A–C with different sizes. For Problems A and B, N-GMRES-sdls and N-GMRES-sd consistently give f/g evaluation counts that are of the same order of magnitude as N-CG. N-GMRES-sd comes close to being competitive with N-CG. L-BFGS is the fastest method for all problems in Table I. For the more difficult Problem C, both N-GMRES-sdls, N-GMRES-sd and L-BFGS are significantly faster than N-CG, which appears to have convergence difficulties for this problem. N-GMRES-sd is clearly faster than N-GMRES-sdls for all tests.

In these comparisons, it is also interesting to consider the computational cost per iteration in addition to the cost of the f/g evaluations. The cost of the f/g evaluations may dominate in many cases, but sometimes the other costs are not negligible. As discussed before, N-GMRES requires approximately $3nw$ operations per step to solve the least-squares problem and update the solution, L-BFGS requires approximately $5nw$ operations per step (the two-loop recursion version, see [2]), and N-CG requires approximately $4n$ operations. We use somewhat larger window size for N-GMRES than for L-BFGS, and their additional costs per iteration are expected to be comparable (within a factor of 2 or so). The additional cost per N-CG iteration is expected to be about 10 times smaller.

3.3. Numerical Results for Problems D–G

Figure 5 gives convergence plots for a single instance of Problem D. It confirms the observations from Figures 2 and 4: for this standard test problem from [15], stand-alone sdls again is very slow, and N-GMRES-sdls and N-GMRES-sd significantly speed up steepest descent convergence. N-GMRES-sdls and N-GMRES-sd have iteration and f/g counts that are of the same order of magnitude as N-CG and L-BFGS, and in particular N-GMRES-sd is competitive with N-CG and L-BFGS. Convergence plots for instances of Problems E–G show similar behaviour and are not presented.

Figure 6 gives convergence plots for an instance of Problem D investigating the effect of varying the steepest descent parameter δ in β_{sd} of (2.2) for N-GMRES-sd optimization. It can be seen that there is a rather broad range about our choice of $\delta = 10^{-4}$ that appears suitable, but choosing δ too large (like 1) or too small (like 10^{-7}) leads to much decreased performance.

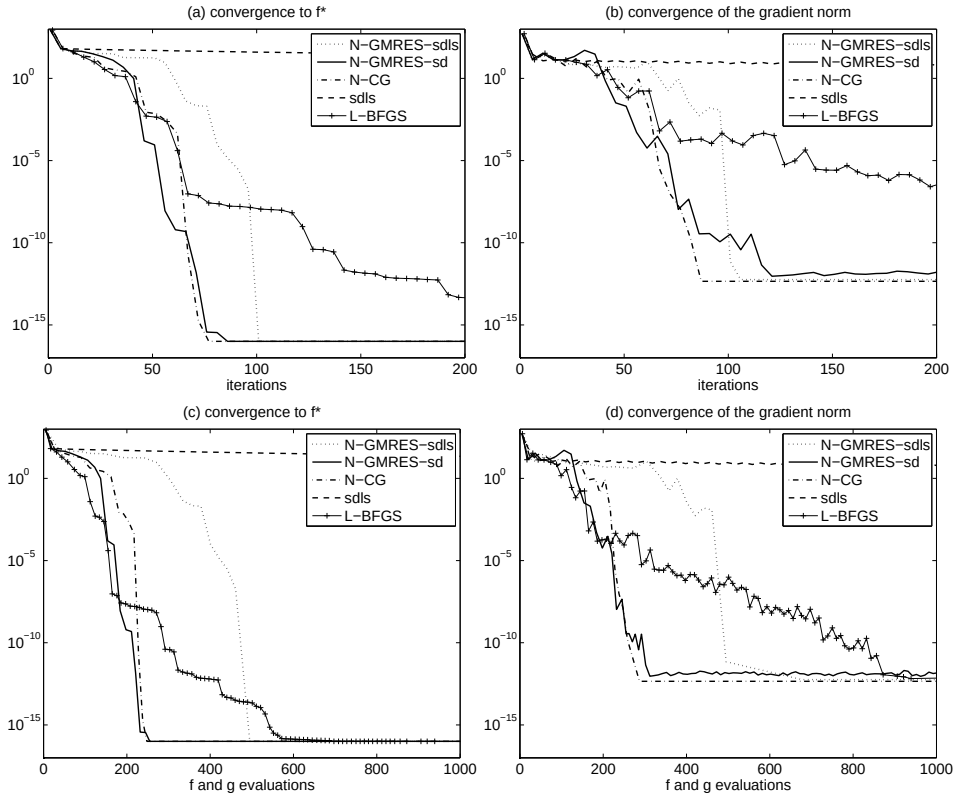

Figure 5. Problem D ($n = 1000$). Convergence comparison.

We have found $\delta = 10^{-4}$ to be a suitable choice for the test problems reported on in this paper.

problem	N-GMRES-sdls	N-GMRES-sd	N-CG	L-BFGS
D n=500	525	172	222	166
D n=1,000	445	211	223	170
D n=50,000	461	251	236	216
D n=100,000	661	220	237	243
E n=100	294	259	243	358
E n=200	317	243	240	394
E n=50,000	832	494	496	1592
E n=100,000	933	650	556	1752

Table II. Average number of f/g evaluations needed to reach $|f(\mathbf{u}_i) - f^*| < 10^{-6}$ for 10 instances of Problems D and E with random initial guess and with different sizes. Numbers in brackets give the number of random trials (out of 10) that did not converge to the required tolerance within 500 iterations (if any).

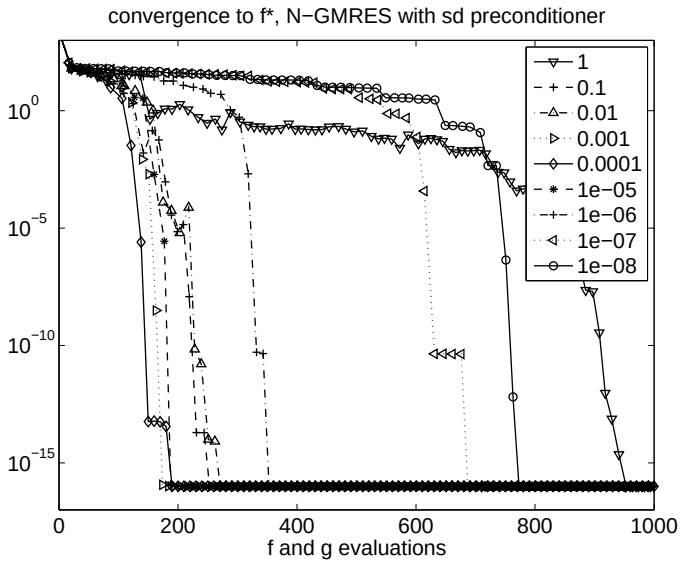

Figure 6. Problem D ($n = 1000$). Effect of varying steepest descent parameter δ on $|f(\mathbf{u}_i) - f^*|$ convergence for N-GMRES-sd optimization as a function of f/g evaluations. Steepest descent parameter $\delta = 10^{-4}$ emerges as a suitable choice, leading to rapid convergence.

Tables II and III on f/g evaluation counts for Problems E–G again confirm the trends that were observed before. N-GMRES-sdls and N-GMRES-sd give f/g evaluation counts that are of the same order of magnitude as N-CG and L-BFGS, and N-GMRES-sd in particular is competitive with N-CG and L-BFGS. Table II includes some tests with larger problem size.

problem	N-GMRES-sdls	N-GMRES-sd	N-CG	L-BFGS
F n=200	140	102(1)	102	92
F n=500	206(1)	175(1)	135	118
G n=100	1008(2)	152	181	358
G n=200	629(1)	181	137	240

Table III. Average number of f/g evaluations needed to reach $|f(\mathbf{u}_i) - f^*| < 10^{-6}$ for 10 instances of Problems F and G with random initial guess and with different sizes. Numbers in brackets give the number of random trials (out of 10) that did not converge to the required tolerance within 500 iterations (if any).

3.4. Numerical Results for a Tensor Optimization Problem

We conclude this section with some numerical results for a difficult tensor optimization problem, in particular, the canonical tensor approximation problem of Figures 1.2 and 1.3 in Paper I ([1]). In this problem, a rank-three canonical tensor approximation (with 450 variables) is sought for a three-way data tensor of size $50 \times 50 \times 50$. The data tensor is generated starting from a canonical tensor with specified rank and random factor matrices that are modified to

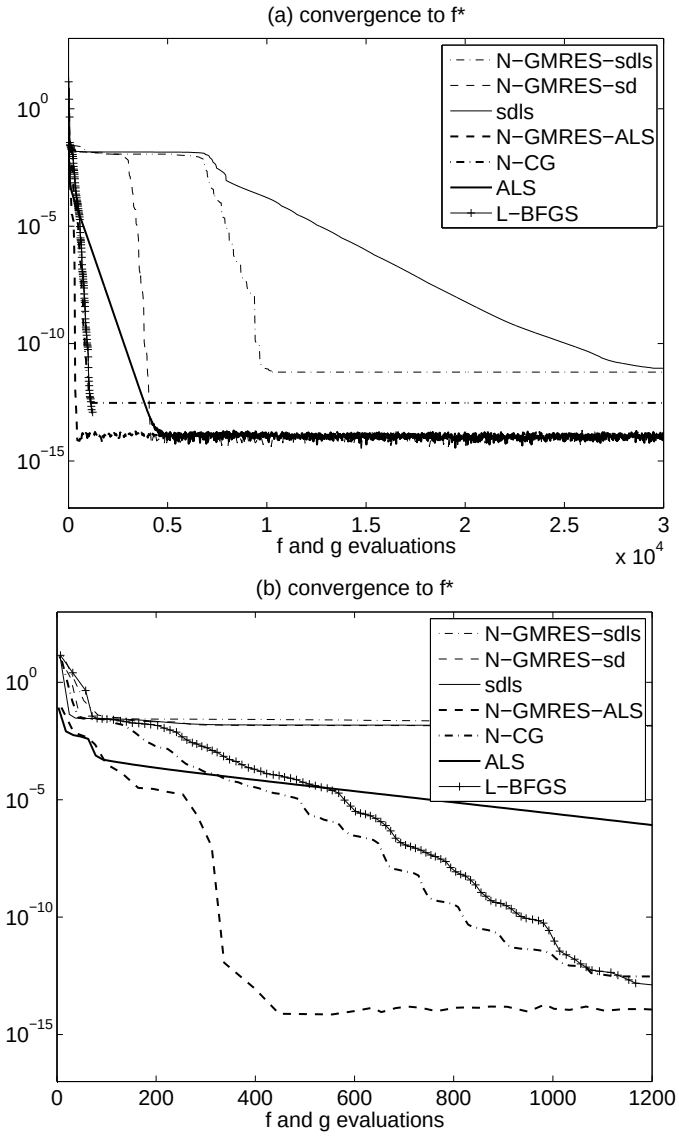

Figure 7. Convergence histories of the 10-logarithm of $|f(\mathbf{u}_i) - f^*|$ as a function of f/g evaluations, for the canonical tensor approximation problem of Figures 1.2 and 1.3 in [1]. Panel (a) shows that stand-alone sdls is very slow for this problem, and N-GMRES-sdls and N-GMRES-sd significantly speed up steepest descent. However, for this difficult problem, it is beneficial to use a more powerful nonlinear preconditioner. Using the ALS preconditioner in stand-alone fashion already provides faster convergence than N-GMRES-sdls and N-GMRES-sd. The zoomed view in Panel (b) shows that N-CG and L-BFGS are faster than stand-alone ALS when high accuracy is required, but N-GMRES preconditioned with the powerful ALS preconditioner is the fastest method by far, beating N-CG and L-BFGS by a factor of 2 to 3. This illustrates that the real power of the N-GMRES optimization algorithm may lie in its ability to employ powerful problem-dependent nonlinear preconditioners (ALS in this case).

have pre-specified column collinearity, and noise is added. This is a standard canonical tensor decomposition test problem [16]. See Paper I for a full description of the problem and for pointers to background information on tensor decomposition.

This problem is interesting because it illustrates how N-GMRES allows for the use of powerful problem-specific nonlinear preconditioners ($M(.)$ in Step I of Algorithm 1), as an alternative to the generic steepest-descent based preconditioners that are the main topic of this paper. In particular, as in [1], we use the alternating least squares (ALS) optimization method for the canonical tensor approximation problem as the nonlinear preconditioning process. It is also interesting to note that ALS is in fact a nonlinear block Gauss-Seidel iteration for the first-order optimality equations of the canonical tensor optimization problem, and as such its use in N-GMRES constitutes a direct generalization to the nonlinear case of GMRES preconditioned by Gauss-Seidel.

Panel (a) of Fig. 7 shows how stand-alone steepest descent (sdls) is very slow for this tensor decomposition problem: it requires more than 30,000 f/g evaluations. (The tensor calculations are performed in MATLAB using the Tensor Toolbox [17]. For this problem, we use $\delta = 10^{-3}$ in (2.2).) The GMRES-sdls and N-GMRES-sd convergence profiles confirm once more one of the main messages of this paper: steepest-descent preconditioned N-GMRES speeds up stand-alone steepest descent very significantly. However, steepest descent preconditioning (which we have argued is in some sense equivalent to non-preconditioned GMRES for linear systems) is not powerful enough for this difficult problem, and a more advanced preconditioner is required. Indeed, Panel (a) of Fig. 7 shows that the stand-alone ALS process is already more efficient than steepest-descent preconditioned N-GMRES. Panel (b) indicates, however, that N-GMRES preconditioned by ALS is a very effective method for this problem: it speeds up ALS very signficantly, and is much faster than N-CG and L-BFGS, by a factor of 2 to 3. (Panel (b) of Fig. 7 illustrates the findings from extensive tests comparing ALS, N-CG and ALS-preconditioned N-GMRES that were reported in Paper I and [16].)

4. Conclusion

In this paper, we have proposed and studied steepest descent preconditioning as a universal preconditioning approach for the N-GMRES optimization algorithm that we recently introduced in the context of a canonical tensor approximation problem and ALS preconditioning [1] (Paper I). We have considered two steepest descent preconditioning process variants, one with a line search, and the other one with a predefined step length. The first variant is significant because we showed that it leads to a globally convergent optimization method, but the second variant proved more efficient in numerical tests, with no apparent degradation in convergence robustness. Numerical tests showed that the two steepest-descent preconditioned N-GMRES methods both speed up stand-alone steepest descent optimization very significantly, and are competitive with standard N-CG and L-BFGS methods, for a variety of test problems. These results serve to theoretically and numerically establish steepest-descent preconditioned N-GMRES as a general optimization method for unconstrained nonlinear optimization, with performance that appears promising compared to established techniques.

However, we argue that the real potential of the N-GMRES optimization framework lies in the fact that it can use problem-dependent nonlinear preconditioners that are more powerful than steepest descent. Preconditioning of N-CG in the form of (linear) variable transformations

is an area of active research [18]. However, it is interesting to note that our N-GMRES optimization framework naturally allows for a more general type of preconditioning: any nonlinear optimization process $M(.)$ can potentially be used as a nonlinear preconditioner in the framework, or, equivalently, N-GMRES can be used as a simple wrapper around any other iterative optimization process $M(.)$ to seek acceleration of that process. The potential of this approach was illustrated by applying N-GMRES with nonlinear block Gauss-Seidel preconditioning to a difficult tensor optimization problem (as in Paper I), significantly outperforming N-CG and L-BFGS.

In the case of GMRES for linear systems, non-preconditioned GMRES (or: GMRES with the identity preconditioner) is often just a starting point. For many difficult problems it converges too slowly, and there is a very extensive and ever expanding research literature on developing advanced problem-dependent preconditioners that in many cases speed up convergence very significantly. In the same way, the present paper is likely not more than a starting point in theoretically and numerically establishing the N-GMRES optimization method with general steepest descent preconditioning process. As the results shown in Fig. 7 already indicate, we expect that the real power of the N-GMRES optimization framework will turn out to lie in its ability to use powerful problem-dependent nonlinear preconditioners. This suggests that further exploring N-GMRES optimization with advanced preconditioners may lead to efficient numerical methods for a variety of nonlinear optimization problems.

Acknowledgments

This work was supported by NSERC of Canada and was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. 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. De Sterck H. A nonlinear GMRES optimization algorithm for canonical tensor decomposition. Submitted to *SIAM Journal on Scientific Computing* 2011; arXiv:1105.5331.
2. Nocedal J. and Wright SJ. *Numerical optimization*, Second Edition. Springer, Berlin, 2006.
3. Washio T and Oosterlee CW. Krylov subspace acceleration for nonlinear multigrid schemes. *Electronic Transactions on Numerical Analysis* 1997; **6**:271–290.
4. Oosterlee CW and Washio T. Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows. *SIAM Journal on Scientific Computing* 2000; **21**:1670–1690.
5. Oosterlee CW. On multigrid for linear complementarity problems with application to American-style options. *Electronic Transactions on Numerical Analysis* 2003; **15**:165–185.
6. Saad Y and Schultz MH. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1986; **7**:856–869.
7. Saad Y. *Iterative Methods for Sparse Linear Systems*, Second Edition. SIAM, Philadelphia, 2003.
8. Saad Y. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing* 1993; **14**:461–469.
9. Smith DA, Ford WF, and Sidi A. Extrapolation methods for vector sequences. *SIAM Review* 1987; **29**:199–234.
10. Fang H and Saad Y. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications* 2009; **16**:197–221.

11. Walker H and Ni P. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis* 2011; **49**:1715–1735.
12. Gilbert JC and Nocedal J. Global convergence properties of conjugate gradient methods for optimization. *SIAM Journal on Optimization* 1992; **2**:21–42.
13. Moré JJ and Thuente DJ. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software* 1994; **20**:286–307.
14. Dunlavy DM, Kolda TG, and Acar E. *Poblano v1.0: A* MATLAB *Toolbox for Gradient-Based Optimization*. Technical Report SAND2010-1422, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, March 2010.
15. Moré JJ, Garbow BS, and Hillstrom KE. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software* 1981; **7**:17–41.
16. Acar E, Dunlavy DM, and Kolda TG. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics* 2011; **25**:67–86.
17. Bader BW and Kolda TG. MATLAB Tensor Toolbox Version 2.4. http://csmr.ca.sandia.gov/ tgkolda/TensorToolbox [March 2010].
18. Hager WW and Zhang H. A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization* 2006; **2**:35–58.