SIAM J. SCI. COMPUT.
Vol. 29, No. 6, pp. 2329–2351

ON THE PERFORMANCE OF TENSOR METHODS FOR SOLVING ILL-CONDITIONED PROBLEMS*

BRETT W. BADER† AND ROBERT B. SCHNABEL‡

Abstract. This paper investigates the performance of tensor methods for solving small- to large-scale systems of nonlinear equations where the Jacobian matrix at the root is ill-conditioned or singular. This condition occurs on many classes of problems, such as identifying or approaching turning points in path-following problems. The singular case has been studied more than the highly ill-conditioned case, for both Newton and tensor methods. It is known that Newton-based methods do not work well with singular problems because they converge linearly to the solution and, in some cases, with poor accuracy. On the other hand, direct tensor methods have performed well on singular problems and have superlinear convergence on such problems under certain conditions. This behavior originates from the use of a special, restricted form of the second-order term included in the local tensor model that provides information lacking in a (nearly) singular Jacobian. With several implementations available for large-scale problems, tensor methods now are capable of solving larger problems. We compare the performance of tensor methods and Newton-based methods for small- to large-scale problems over a range of conditionings, from well-conditioned to ill-conditioned to singular. Previous studies with tensor methods concerned only the ends of this spectrum. Our results show that tensor methods are increasingly superior to Newton-based methods as the problem grows more ill-conditioned.

Key words. nonlinear equations, tensor methods, Newton's method, ill-conditioned problems, singular problems

AMS subject classification. 65H10

DOI. 10.1137/040607745

1. Introduction.

This paper examines two classes of methods for solving the following nonlinear equations problem:

$$\text{given } F : \mathbb{R}^n \to \mathbb{R}^n, \quad \text{find } x_* \in \mathbb{R}^n \text{ such that } F(x_*) = 0, \tag{1.1}$$

where it is assumed that $F(x)$ is at least once continuously differentiable. General systems of nonlinear equations defined by (1.1) arise in many practical situations, including systems produced by finite-difference or finite-element discretizations of boundary value problems for ordinary differential equations (ODEs) and partial differential equations (PDEs).

As a subset of the general nonlinear equations problem (1.1), there is a class of important problems where $F'(x_*)$ is singular or, at least, very ill-conditioned. Such examples arise in bifurcation tracking and path-following problems where the goal is to locate turning points, such as the ignition and extinction points in chemical combustion. Resolving these features is important to engineers, who, for instance, may

*Received by the editors May 4, 2004; accepted for publication (in revised form) March 15, 2007; published electronically October 5, 2007. This work was supported in part by Air Force Office of Scientific Research grant F49620-00-1-0162 and Army Research Office grant DAAG55-98-1-0176. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

http://www.siam.org/journals/sisc/29-6/60774.html

†Computational Sciences Department, Sandia National Laboratories, Albuquerque, NM 87185-1318 (bwbader@sandia.gov).

‡Department of Computer Science, University of Colorado, Boulder, CO 80309. Current address: School of Informatics, Indiana University, Bloomington, IN 47405 (schnabel@indiana.edu).

be designing control systems for such applications and may need to know important operating boundaries.

Standard methods for solving (1.1), such as Newton's method, base each iteration upon a local, linear model $M(x_k + d)$ of the function $F(x)$ around the current iterate $x_k \in \mathbb{R}^n$. Standard methods work well for problems where the Jacobian at the solution, $F'(x_*)$, is well-conditioned, but they face difficulties when the Jacobian is singular, or even nearly singular, at the solution. Many authors have analyzed the behavior of Newton's method on singular problems and have proposed acceleration techniques as remedies (see, e.g., Decker, Keller, and Kelley [9]; Decker and Kelley [10, 11, 12]; Griewank [19]; Griewank and Osborne [20]; Kelley and Suresh [22]; and Reddien [28]). Their collective analysis shows that, from many starting points, Newton's method is locally q-linearly convergent with constant converging to $\frac{1}{2}$ on singular problems where the second-order term $F''(x_k)$ contains appropriate null space information. Acceleration techniques can improve this behavior; however, they require a priori knowledge that the problem is singular, which is not practical for general problem solving.

Tensor methods, however, do not require a priori knowledge of whether the problem is singular or not. These methods were introduced by Schnabel and Frank in [34] and base each iteration on a simplified quadratic model of $F(x)$ such that the quadratic term is a low-rank secant approximation that augments the standard linear model. Tensor methods also have been extended by other authors to utilize iterative solvers, making the methods appropriate for solving large-scale problems (see [1], [4], and [17]).

The analysis in [16] proves that direct tensor methods have quadratic convergence on nonsingular problems and a superlinear convergence rate on problems where the Jacobian matrix at the solution is singular. Specifically, when the rank of the Jacobian at the root is $n-1$, "practical" tensor methods (i.e., those using secant approximations for the tensor term) have three-step superlinear convergence behavior with q-order $\frac{3}{2}$. In practice, one-step superlinear convergence frequently is observed on these problems, which makes the method even more attractive. The second-order term provides higher order information in recent step directions, which aids in cases where the Jacobian is (nearly) singular at the solution. As the iterates approach the solution, the Jacobian lacks information in the null space direction, and the second-order term supplies useful information for a better quality step. Computational evidence in [34] on small problems shows that tensor methods show 21–23% average improvement (in terms of nonlinear iterations and function evaluations) over standard methods on nonsingular problems and 40–43% improvement on singular problems with $\text{rank}(F'(x_*)) = n - 1$.

While tensor methods have encouraging theoretical and computational results on singular problems, less is known about their performance relative to Newton's method on ill-conditioned problems. Do tensor methods outperform Newton's method due to the close relationship of ill-conditioned matrices with singular matrices? Or do tensor methods exhibit superior behavior only when the problem is truly singular? Does the computational performance of Newton's method degrade gradually as the problem becomes more singular, or sharply at the singularity? The performance comparison over a spectrum of ill-conditioned problems was previously unknown. Thus, this paper examines the performance of tensor methods versus standard methods as the problem becomes more ill-conditioned. We consider tensor methods using direct factorizations of the Jacobian matrix for small-scale problems in addition to Krylov-based iterative tensor methods for large-scale problems.

The organization of this paper is as follows. Because this research involves meth-

ods for solving small- to large-scale problems, this paper includes background for both types in section 2. Specifically, we review direct methods for solving small-scale problems, and we review Krylov-based iterative methods for solving large-scale problems, including the relevant algorithms from [1] and [17]. Section 3 presents numerical results on several small- and large-scale ill-conditioned test problems to examine the performance of tensor methods on problems over a range of conditionings, from well-conditioned to singular. Finally, section 4 summarizes the numerical results and provides some concluding remarks.

Throughout this paper, a subscript k refers to the current iterate of a nonlinear solver. We denote the Jacobian $F'(x)$ by $J(x)$ and abbreviate $J(x_k)$ as J_k. Similarly, $F(x_k)$ is abbreviated often as F_k. When the context is clear, we may drop the subscript k while still referring to the "current" values at an iteration.

2. Algorithms. In this section, we introduce the relevant methods for solving systems of nonlinear equations. We start with methods that use a direct factorization of the Jacobian matrix, and then we discuss inexact methods that use Krylov subspace projection techniques. General references for these topics in nonlinear solvers include [14], [21], [26], and [30].

2.1. Standard methods. In this paper, we denote by standard methods the class of methods for solving (1.1) that uses a linear approximation to $F(x)$ at each iterate around the current iterate $x_k \in \mathbb{R}^n$. Most notable among these methods is Newton's method, which uses the linear local model

$$M_N(x_k + d) = F_k + J_k d, \tag{2.1}$$

where $d \in \mathbb{R}^n$ is the step and $J_k \in \mathbb{R}^{n\times n}$ is either the current Jacobian matrix or an approximation to it. A root of this local model provides the Newton step

$$d_N = -J_k^{-1} F_k,$$

which is used to reach the next trial point. Thus, Newton's method is defined when J_k is nonsingular and consists of updating the current point with the Newton step,

$$x_+ = x_k + d_N. \tag{2.2}$$

If the Jacobian $J(x_k)$ is Lipschitz continuous in a neighborhood containing the root x_* and $J(x_*)$ is nonsingular, then the sequence of iterates produced by (2.2) converges locally and q-quadratically to x_*. That is, there exist constants $\delta > 0$ and $c \geq 0$ such that the sequence of iterates x_k produced by Newton's method obeys

$$\|x_{k+1} - x_*\| \leq c \|x_k - x_*\|^2$$

if $\|x_0 - x_*\| \leq \delta$.

When these standard approaches use direct factorizations of the Jacobian matrix, we will refer to these methods as direct methods. Due to the storage and linear algebra costs, direct methods are practical only for solving small, dense problems. If the matrix is sparse, then sparse linear algebra techniques may be used to solve larger problems but are still limited by the size of the problem.

2.2. Tensor methods. Tensor methods [34] solve (1.1) by including a second-order term in the local model at each iteration. The local tensor model has the general form

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} T_k dd, \tag{2.3}$$

where $T_k \in \mathbb{R}^{n\times n\times n}$ is the tensor term at x_k and is selected so that the model interpolates a small number p of function values in the recent history of iterates. By choosing the smallest T_k in the Frobenius norm, T_k has rank p and $T_k dd$ is both simple in form and inexpensive to find. Because (2.3) may not have a root, one solves the minimization subproblem

$$\min_{d\in\mathbb{R}^n} \|M_T(x_k+d)\|_2\,, \tag{2.4}$$

and a root or minimizer of the model is the tensor step.

The additional cost of forming, storing, or solving the model is minor compared to Newton's method. Specifically, the additional cost of a direct tensor method is about n^2p multiplications (QR implementation) and $2p$ vectors of length n in storage. For our numerical experiments, we will consider only the simplest case of $p = 1$. Computational evidence in [34] suggests that additional past iterates add little benefit to the computational performance of the tensor method.

2.3. Newton–Krylov methods. Up to this point, we have discussed direct methods for the solution of small, dense problems, such that the local model is solved with direct factorizations of the Jacobian matrix. Standard direct methods, such as Newton's method, are impractical on large-scale problems because of their high linear algebra costs and large memory requirements. Thus, most large systems often are solved successfully using a class of "inexact" Newton methods [13]:

$$x_{k+1} = x_k + d_k, \quad \text{where} \quad F'(x_k)d_k = -F(x_k) + r_k, \quad \|r_k\| \le \eta_k \|F(x_k)\|\,, \tag{2.5}$$

such that the local model typically is solved only approximately at each step using a less expensive approach. These "inexact" steps then locate the next trial point. The convergence rate of inexact Newton methods depends on the forcing sequence η_k [13]. If η_k is constant, then convergence is linear. Successively better approximations to the linear model at each iteration (i.e., η_k getting smaller as k increases) preserve the rapid convergence behavior of Newton's method when nearing the solution. The computational savings reflected in this less expensive inner iteration is usually partially offset with more outer iterations, but the overall savings still is quite significant on large-scale problems by avoiding the direct methods that solve the local model exactly.

The most common methods for approximately solving the local Newton model are Krylov-based methods, which iteratively solve the linear system projected onto the Krylov subspace $\mathcal{K}$. A linear Krylov subspace method is a projection method that seeks an approximate solution x_m to the linear system $Ax = b$ from an m-dimensional affine subspace $x_0 + \mathcal{K}_m$. Here $\mathcal{K}_m$ is the subspace

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2r_0, \ldots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$ is the residual at an initial guess x_0. A popular Krylov subspace method is the generalized minimum residual method (GMRES), which computes a solution $x_m \in x_0 + \mathcal{K}_m$ such that the residual norm over all vectors in $x_0 + \mathcal{K}_m$ is minimized. That is, at the mth step, GMRES finds x_m such that $\|b - Ax_m\|_2$ is minimized for all $x_m \in x_0 + \mathcal{K}_m$.

Newton-GMRES is one specific method in the class of Newton–Krylov methods. Here, the linear system is the Newton equation $J_k d = -F_k$, and the system is solved via GMRES according to the tolerance η_k in (2.5). Krylov subspace methods have

the appeal of not requiring an explicit Jacobian due to their exclusive use of Jacobian-vector products, which may be calculated by a finite-difference directional derivative. For this reason and others, Newton-GMRES is a popular algorithm for solving large-scale problems, and it will be the standard large-scale Newton-based algorithm for comparisons in our numerical experiments.

Newton–Krylov methods have been considered by many authors, including Brown and Saad [6, 7], Chan and Jackson [8], and Brown and Hindmarsh [5]. Their computational results show that Newton–Krylov methods can be quite effective for many classes of problems in the context of systems of PDEs and ODEs.

2.4. Tensor-Krylov methods. To accommodate large problems and avoid the expensive direct solution of the local model, the three tensor-Krylov methods described in [1] combine the concepts from direct tensor methods with concepts from inexact Newton methods [13] using Krylov-based linear solvers. The tensor-Krylov methods calculate an inexact tensor step from a specially chosen Krylov subspace that facilitates the solution of a minimization subproblem at each iteration. Here, we give a very brief overview of these methods.

All large-scale tensor methods in this paper consider only a rank-one tensor model, which only interpolates the function value at the previous iterate. Thus, the rank-p model of (2.3) reduces to

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2, \tag{2.6}$$

such that

$$a_k \in \mathbb{R}^n = \frac{2(F_{k-1} - F_k - J_k s_k)}{(s_k^T s_k)^2}, \tag{2.7}$$

$$s_k \in \mathbb{R}^n = x_{k-1} - x_k. \tag{2.8}$$

In each of the tensor-Krylov methods, the tensor step is found by approximately solving the minimization subproblem

$$\min_{d \in \mathcal{K}_m} \left\| F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2 \right\|_2, \tag{2.9}$$

where $\mathcal{K}_m$ is a specially chosen Krylov subspace that facilitates the solution of the quadratic model. The three methods described in [1] differ in their choice of $\mathcal{K}_m$, and they are identified by this characteristic difference. TK2 and TK2+ use a Krylov-based local solver that starts with an initial block of two vectors (TK2+ also augments the Krylov subspace in a special way). Similarly, TK3 uses an initial block of size three. The three methods share the ability to calculate an approximate tensor step that satisfies the tensor model to within a specified tolerance. Their cost per nonlinear iteration exceeds that of Newton-GMRES by at most $10n+4mn+6m^2$ multiplications (in comparison, GMRES costs $\mathcal{O}(nm^2)$ multiplications). The methods can be readily combined with either left or right preconditioning. More details of these Krylov-subspace methods for solving the tensor model may be found in [1].

2.5. Tensor-GMRES method. Another large-scale tensor method is that of Feng and Pulliam [17], which uses Krylov subspace projection techniques for solving the local tensor model. In particular, it uses GMRES to first find the approximate Newton step $d_N = d_0 + V_m y_m$. The columns of V_m form an orthonormal basis for the Krylov subspace $\mathcal{K}_m$ generated by the corresponding Arnoldi process, and the Hessenberg matrix H_m is also generated from the Arnoldi process. Given these key

matrices, their tensor-GMRES algorithm proceeds to solve a projected version of the tensor model (2.6) along a subspace that spans the Newton step direction and the Krylov subspace from the Newton step solution. (That is, the approximate tensor step is in the span of the Krylov subspace $\mathcal{K}_m^N$ and d_0, or, equivalently, the span of the matrix $[V_m, d_0]$.) Thus, their algorithm solves the least-squares problem

$$\min_{d \in \{d_0\} \cup \mathcal{K}_m^N} \left\| F_k + J_k d + \tfrac{1}{2} P a (s^T d)^2 \right\|, \tag{2.10}$$

where P is the projection matrix

$$P = Y(Y^T Y)^{-1} Y^T, \quad \text{where } Y = J_k[V_m, d_0]. \tag{2.11}$$

The analysis in [17] shows that the same superlinear convergence properties for the unprojected tensor model considered in [16] also hold for the projected tensor model (2.10). The complete tensor-GMRES algorithm for solving (2.10) at the kth nonlinear iteration is listed in [17].

Despite the algorithm's difficult algebra, the design actually is rather straightforward. The algorithm may be viewed as an extension of Newton-GMRES, where the inexact Newton step is calculated via GMRES in the standard way. The tensor step is calculated subsequently using the Krylov subspace information generated for the Newton step. In this way, the method also is consistent with preconditioning techniques and a matrix-free implementation, which makes it appealing for general use.

The extra work and storage beyond GMRES for computing the tensor step are quite small. The extra work is at most $4mn + 5n + 2m^2 + \mathcal{O}(m)$ multiplications plus a single Jacobian-vector product for evaluating the tensor term a_k. The extra storage amounts to two extra n-vectors for a and s plus a few smaller working vectors of length m.

The results in [17] show the superlinear convergence behavior of tensor-GMRES on three singular and nearly singular problems, where the Newton-GMRES method exhibits linear convergence due to a lack of sufficient first-order information. The margin of improvement (in terms of reduction of nonlinear iterations over Newton's method) varied from 20% to 55% on the simpler problems and from 32% to 60% on the more difficult Euler problem. Running times and the total number of Jacobian-vector products for each method were not reported in [17], but from our own experience with the algorithm, we assume that these performance metrics are correlated with the number of nonlinear iterations.

3. Numerical experiments on ill-conditioned problems.

3.1. Small problems solved with direct methods. This section investigates the performance of direct tensor methods as well as Newton's method on a set of small problems that include a parameter for adjusting the ill-conditioning of the Jacobian matrix at the root. The results show that Newton's method requires increasingly more iterations as the problem ill-conditioning grows, whereas the direct tensor methods are only mildly affected.

Following the approach in [34], we created ill-conditioned problems by modifying nonsingular test problems to be of the form

$$\hat{F}(x, \lambda) = F(x) - \lambda F'(x_*) A (A^T A)^{-1} A^T (x - x_*), \tag{3.1}$$

where $F(x)$ is the standard nonsingular test function, x_* is its root, $A \in \mathbb{R}^{n \times j}$ is an arbitrary matrix that has full column rank with $1 \le j \le n$, and $\lambda \in [0, 1]$ is a

parameter for ill-conditioning. We denote $\hat{J}(x,\lambda)$ as the Jacobian matrix of $\hat{F}(x,\lambda)$ with respect to x:

$$\hat{J}(x,\lambda) \equiv \frac{\partial}{\partial x}\hat{F}(x,\lambda).$$

These new test problems are similar to problems in continuation or homotopy methods, except that $\hat{F}(x,\lambda)$ has the same root as $F(x)$ (i.e., x_*) for all values of λ. While the problem becomes harder to solve as λ approaches 1, the idea is not to follow the path over a sequence of values of λ. Rather, we solve the modified problem for each value of λ from the same starting point across all tests and record the number of iterations required to reach the solution.

One special quality of this modified problem is that if $\lambda = 1$ and $F'(x_*)$ has full rank, then the rank of $\hat{J}(x_*,\lambda)$ equals $\operatorname{rank}(F'(x_*)) - \operatorname{rank}(A) = n - \operatorname{rank}(A)$. Thus, as λ approaches 1, the rank deficiency of $\hat{J}(x_*,\lambda)$ approaches the rank of A. Stated another way, a set of the smallest singular values of $\hat{J}(x_*,\lambda)$ equal to the rank of A will approach 0 as λ approaches 1.

Different sets of singular and ill-conditioned problems may be created using the matrix A in (3.1); adding more independent columns to A serves to decrease the rank of $\hat{J}(x_*,1)$. We routinely used the matrices

$$A \in \mathbb{R}^{n\times 1}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \end{pmatrix},$$

$$A \in \mathbb{R}^{n\times 2}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & -1 & 1 & \dots & \pm 1 \end{pmatrix},$$

and

$$A \in \mathbb{R}^{n\times 3}, \quad A^T = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & -1 & 1 & \dots & \pm 1 \\ 1 & 1 & -1 & \dots & \pm 1, \end{pmatrix}$$

because they provide "balanced" problems by acting equally on the whole Jacobian $F'(x_*)$. Letting A equal the unit vector e_1, for example, would operate only on the $(1,1)$ element of $F'(x)$.

We chose six problems for testing from the standard small-dimensional test set of Moré, Garbow, and Hillstrom [25] and modified them according to (3.1). Table 3.1 lists the problems used in our numerical tests, along with the corresponding dimensions. For all problems, the starting vector x_0 was the standard starting point for each problem published in [25]. Except for the cases mentioned below, these starting points did not require a global strategy to reach the solution (i.e., the full step was accepted at all iterations).

We chose the parameter λ in (3.1) to asymptotically approach $\lambda = 1$ by using the values $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \dots$. Thus, for $j = 0$ the problem is the original, unmodified problem, and each subsequent value of j makes the problem more ill-conditioned. At some point, round-off errors in the evaluation of $\hat{F}(x,\lambda)$ as well as the numerical precision of the root x_* make the numerical solution indistinguishable from the case of $\lambda = 1$. All subsequent values of λ would produce the same result, so we collected only the results up to these points and indicated the case of $\lambda = 1$ at the rightmost extent of our plots. Due to the numerical precision of the solutions, the Jacobian at x_* may not be fully singular.

TABLE 3.1
Ill-conditioned test problems.

Problem	Size
Broyden tridiagonal	50
Broyden banded	50
Discrete integral equation	50
Discrete boundary value function	50
Rosenbrock's function	2
Brown almost linear	10

We used the following two stopping conditions in all these tests: $\|F(x_k)\|_\infty \leq 10^{-12}$ and $\|x_k - x_{k-1}\|_\infty \leq 10^{-12}$, which are the same conditions Eisenstat and Walker used when analyzing inexact Newton methods in [15]. In many practical applications, less stringent convergence tolerances are commonly used, but these tight tolerances were used in this experiment (and later experiments) to differentiate results at higher condition numbers and to allow asymptotic convergence behavior to become evident. To help gauge what can be expected with looser stopping tolerances, we have included convergence histories for a few problems (see, e.g., Figure 3.1) where the performance difference is still present but less striking.

As a prelude to these results and to help explain what is happening in these experiments, we first provide a graphical description. Figure 3.1 shows the typical iteration profiles for different values of λ that we observed. This figure graphs the function value at each iteration for the tensor and Newton methods on a typical problem approaching rank $n-1$ with various values of $\lambda = 0, 0.9, 0.99, 0.999, \ldots, 1$. All of the tensor method profiles are bunched together on the left, requiring few iterations even as λ nears 1, whereas the profiles of Newton's method are spread out and require increasingly more iterations for convergence as the problem becomes more ill-conditioned. This plot of iteration profiles is typical for all problems. Thus, while all of the tensor runs display superlinear convergence throughout the iterations, it is evident that Newton's method converges linearly for a number of iterations before accelerating to quadratic convergence. With increasing ill-conditioning, the region of quadratic convergence for Newton's method shrinks in size, acting as if the problem were singular outside of this region.

Figure 3.2 shows the relative performance of Newton's method versus a rank-one tensor method on the six problems, in the case where each Jacobian at the solution approaches rank $n-1$. The condition number $\hat{J}(x_*, \lambda)$ is plotted on the abscissa, while the number of iterations for both methods is plotted on the ordinate axis. One tensor method iteration is just slightly more expensive than a Newton iteration. There were no linesearches in all cases except on Rosenbrock's function at $\lambda = 0$, which explains the unusual spike on the left in that plot. Also, both methods arrived at different solutions in the Brown almost linear problem at $\lambda = 0$, and thus these points were not included in that plot. For all other points, both methods arrived at the same x_*.

The six plots of Figure 3.2 show that as the ill-conditioning of a particular problem grows, Newton's method requires more iterations, whereas the tensor method is very mildly affected. This is a key result. Previously, it was suggested that tensor methods behave reasonably well on ill-conditioned problems, but there was no numerical testing of this conjecture.

However, because the condition number is just a single value representing the ratio of the largest to the smallest magnitude singular values, it does not capture the singular value spectrum of the Jacobian. Say, for instance, that there are multiple

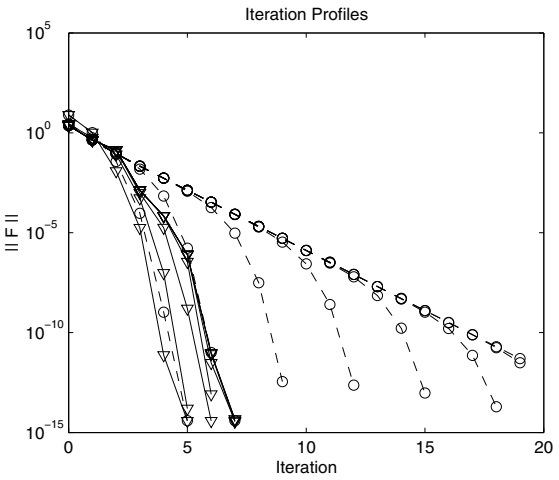

FIG. 3.1. *Superimposed iteration profiles of the tensor method (∇, solid lines) and Newton's method ($\circ$, dashed lines) when solving the modified Broyden tridiagonal function as it approaches rank $n-1$. For both methods and their corresponding set of iteration profiles, problem difficulty (λ) increases from left to right, making the problem more ill-conditioned and requiring more iterations to solve.*

singular values approaching zero, creating a very ill-conditioned matrix. One might speculate that a tensor method using a rank-one tensor term would be less advantageous in this case because it could "handle" only one direction, perhaps the direction associated with the smallest singular value, leaving other (near) singular directions to impede convergence. Figures 3.3 and 3.4 consider exactly these cases where the Jacobians approach rank $n-2$ and $n-3$, respectively, while still using a rank-one tensor model. Two problems were not included in these figures for different reasons. The modified Rosenbrock function had excessive linesearches that obscured the results in the rank $n-2$ problems, and the rank $n-3$ problem is not possible with the Rosenbrock function due to its small dimension of $n=2$. For the Brown almost linear problem, each method arrived at a solution that was different from the root x_* used in (3.1). For these reasons, these two problems were not included in Figures 3.3 and 3.4.

It is evident that the results for higher rank deficiencies are not as striking as in Figure 3.2, but the general trend still remains—tensor methods perform better than Newton's method on ill-conditioned problems, even when the Jacobian approaches a rank deficiency greater than the rank of the tensor term. A few peculiarities exist in the results that warrant explanation. We attribute the "humps" in both curves to the random paths each method takes to arrive at the solution, and we believe this shape is coincidental. The "spikes" in both methods for the two discrete problems in Figure 3.4 are due to linesearches occurring at a specific value of λ. The linesearches force a few more outer iterations to eventually solve the problems.

3.2. Medium-scale problems solved with inexact methods. This subsection investigates the performance of the Newton-GMRES, tensor-GMRES, and tensor-Krylov methods on three ill-conditioned problems of moderate size and complexity: the Bratu problem, a modified discrete boundary value function, and a modified Broyden tridiagonal problem. The tests on the ill-conditioned problems have a design

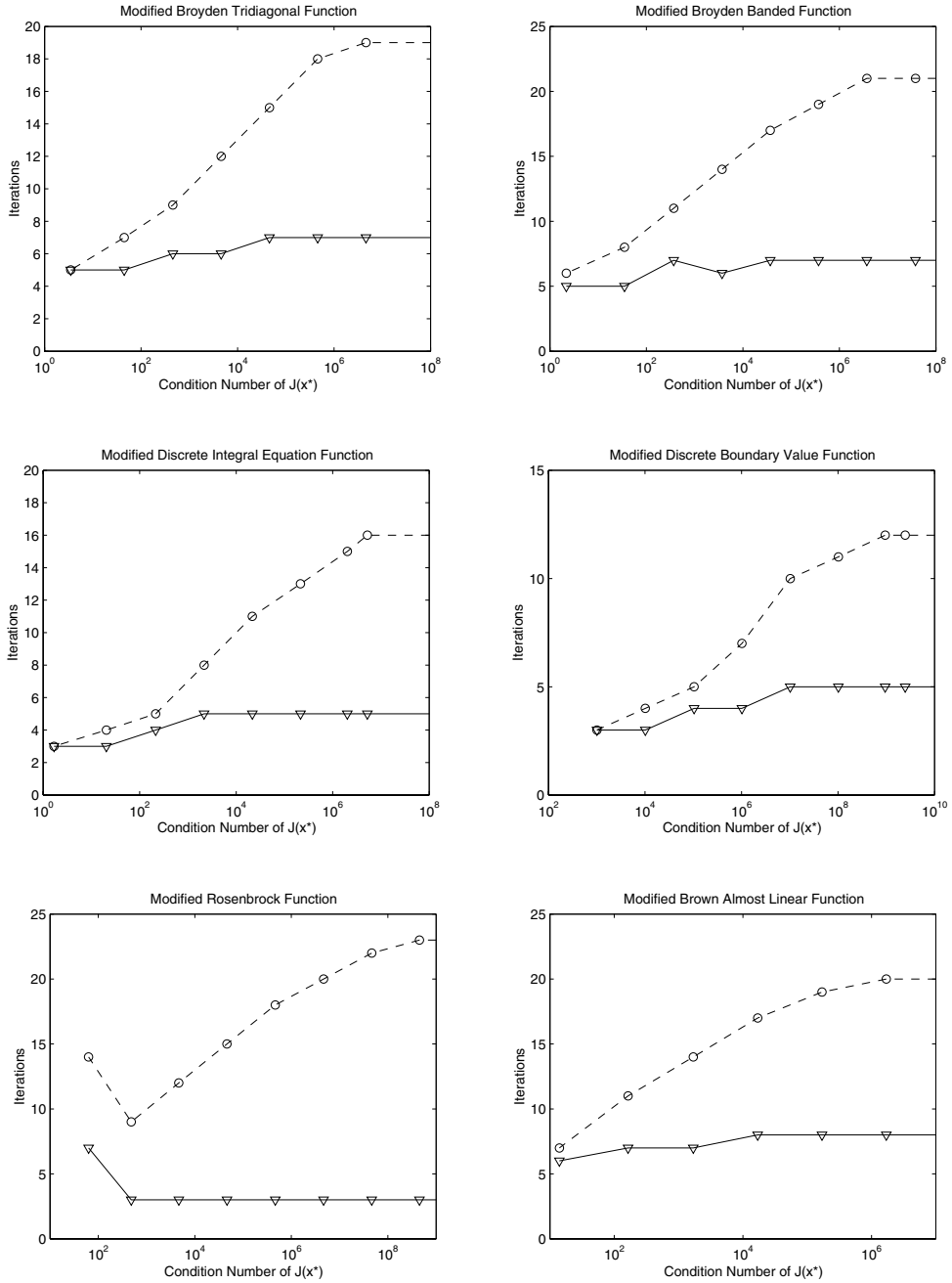

FIG. 3.2. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (∘, dashed line) on variably ill-conditioned test problems that approach rank* $n-1$.

similar to that of the study performed in section 3.1, and we investigate to what extent the benefits of using direct tensor methods for solving small ill-conditioned/singular problems that were shown in section 3.1 extend to inexact versions of tensor methods for larger problems.

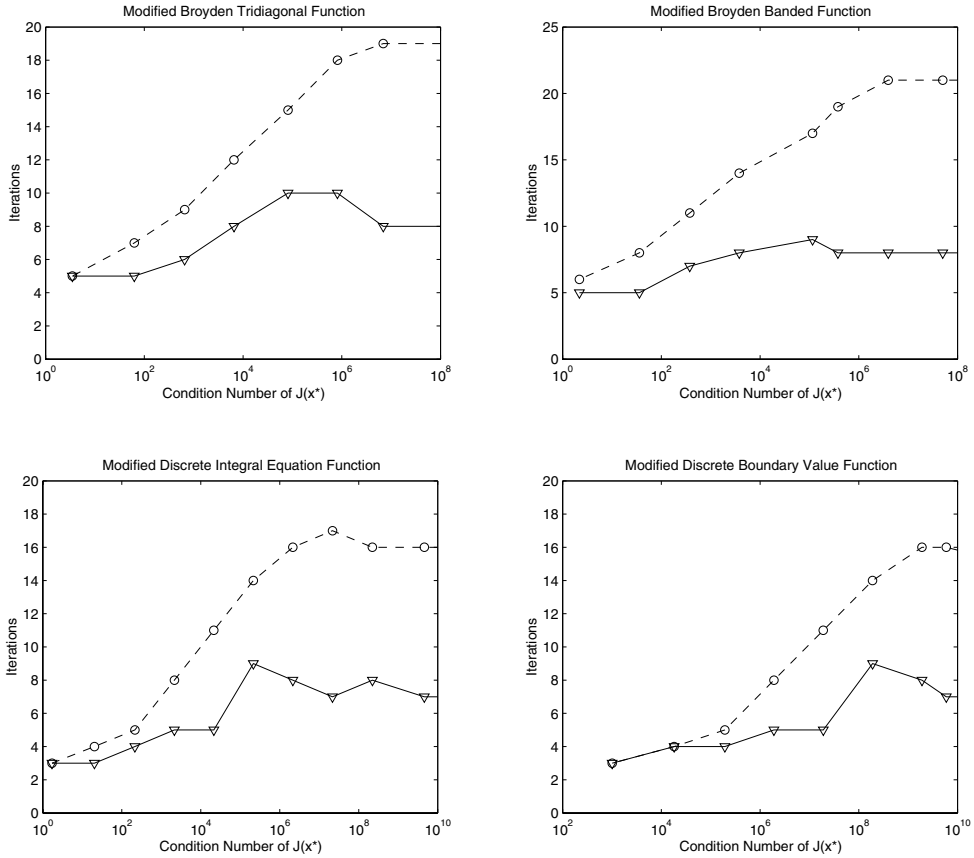

FIG. 3.3. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (○, dashed line) on variably ill-conditioned test problems that approach rank* $n-2$.

We implemented all of the methods in MATLAB using double precision arithmetic. For objective comparisons, we used the same level of basic linear algebra routines in MATLAB in all of our tests. Thus, the results in this section do not reflect the most efficient implementations that are available, but the statistics that we collected (nonlinear iterations, function evaluations) are invariant to optimal implementations.

For the numerical tests, we used the same stopping conditions and parameters as in section 3.1. For the inner method, we solved the local model to a constant relative tolerance of 10^{-4}, which requires uniformly close approximations to Newton and/or tensor steps at each iterate and results in fast local q-linear convergence. Using a constant relative tolerance may not be ideal in practice (see [15] for alternatives), but it provides for more unbiased comparisons among the methods. Changing this tolerance affects the fastest eventual convergence rate that may be achieved (linear convergence for inexact Newton methods [13]). Specifically, a tighter (or looser) tolerance leads to a faster (slower) convergence rate and does not change the relative performance of these methods with respect to each other. The effect may be seen in the steepest slope that the convergence curve may achieve as the nonlinear iterates near the solution.

In practice, the Krylov subspace dimension should be kept small to reduce arithmetic and storage costs, which may be accomplished via restarted Krylov methods

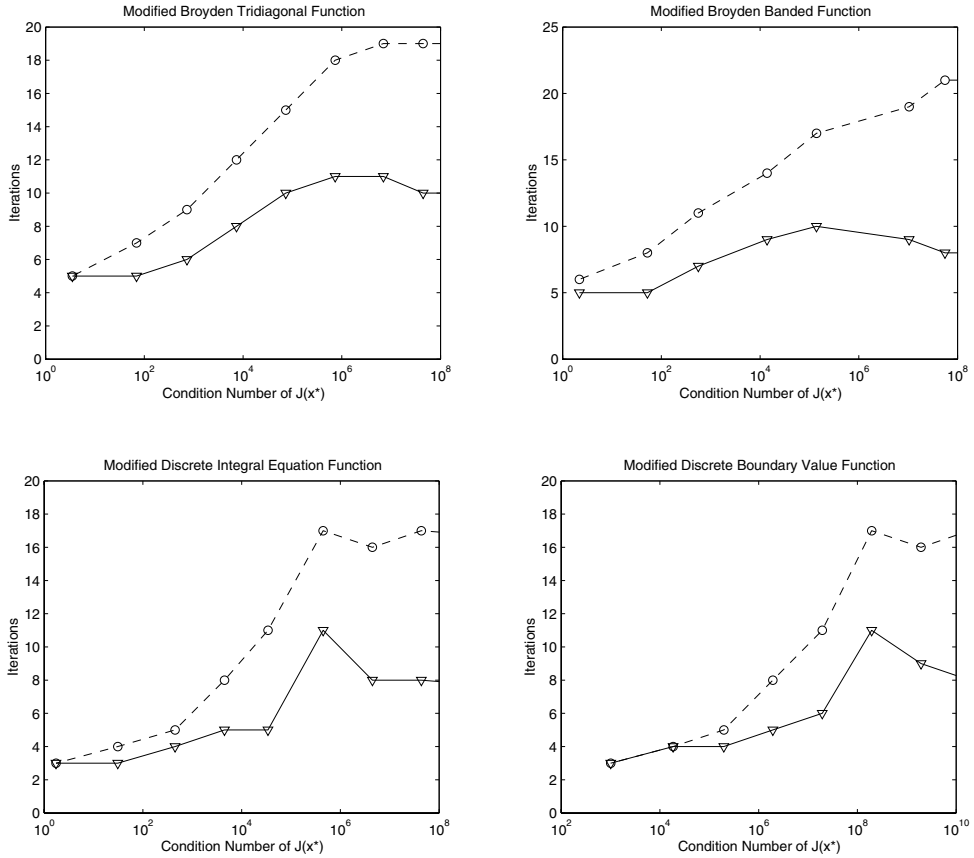

FIG. 3.4. *Comparison of a rank-one tensor method (∇, solid line) with Newton's method (○, dashed line) on variably ill-conditioned test problems that approach rank $n-3$.*

and preconditioning. However, to eliminate any ill effects of small Krylov subspaces preventing convergence in the local model and affecting the outer iterations, the maximum Krylov subspace was set to the problem dimension, $m_{max} = n$, and the solver was not restarted. We used preconditioners that are appropriate for the problem, and they will be discussed with each problem.

In the next three subsections, we present numerical results on the ill-conditioned problems. The results of the tensor-Krylov method TK2 are virtually identical to those of TK2+, so we do not include them here.

3.2.1. Bratu problem. The Bratu problem is a simplified model for nonlinear diffusion phenomena occurring, for example, in semiconductors and combustion, where the source term is related to the Arrhenius law for modeling exothermic reactions. The following version is taken from the set of nonlinear model problems collected by Moré [24]. The problem is the nonlinear PDE in u,

$$-\nabla^2 u = \lambda e^u \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \tag{3.2}$$

where $\nabla^2 = \sum_{i=1}^{n} \partial^2/\partial x_i^2$ is the Laplace operator, $\lambda \in \mathbb{R}$ is a parameter, Ω is the bounded domain $(0,1) \times (0,1)$, and $\partial\Omega$ is the boundary of Ω.

Problem (3.2) has a unique solution for $\lambda \leq 0$, but for $\lambda > 0$, there may be zero, one, or two solutions (cf. [18]). The critical value $\lambda^* = 6.80812$ is a limit point such that for $0 < \lambda < \lambda^*$, problem (3.2) has two solutions and for $\lambda > \lambda^*$, it has no solutions. Also, the problem at the limit point is singular with a rank of $n - 1$, and as λ approaches the limit point, the discretized problem becomes harder to solve. To investigate the effects of ill-conditioning on the inexact algorithms, we increased λ over the range $\lambda \in [5, 6.806652]$, which increased the condition number of $J(x_*)$ from about 10^3 to 10^6.

When testing the Bratu problem, the initial approximate solution was zero on a uniform grid of size 31×31. The Laplace operator was discretized using centered differences (5-point stencil), and our preconditioner was also the centered differences discretization of the Laplace operator. Note that the preconditioner is not exact due to omitting the forcing term in (3.2). We computed Jacobian-vector products using first-order forward differences. Hence, the number of function evaluations is the sum of the total number of Arnoldi iterations, the number of linesearch backtracks (if any), and the total number of nonlinear iterations. Thus, the number of function evaluations provides a relative measure of overall work for each algorithm.

Figure 3.5 presents the results of these tests, comparing the number of function evaluations computed for each method using the three choices of preconditioning (none, left, and right preconditioning). If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots span the range of 3×10^0 to about 10^4 instead of 10^3 to 10^6 for the Jacobian itself. The results in the figure are similar to those in figures in section 3, where Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned and difficult. The tensor-Krylov methods required from four to seven outer iterations, whereas Newton-GMRES required from four to 13 outer iterations. That is, Newton-GMRES required almost double the number of tensor-Krylov iterations on the most ill-conditioned problem. The three plots in Figure 3.5 uphold the prior conclusion that increased ill-conditioning only mildly increases the number of outer iterations for tensor methods.

In all cases, when the problem is not overly difficult (e.g., the experiments at a condition number of 10^3 in Figure 3.5), Newton-GMRES is more efficient in terms of nonlinear iterations than the tensor methods. Here, the number of iterations for all of the methods is equal, but GMRES is more efficient at solving the local Newton model, which results in fewer total function evaluations. When the problem is more difficult, however, the tensor methods are superior to Newton-GMRES by a factor of up to 1.9. Approaching the limit point even more closely would have given even more of an advantage to the tensor-Krylov methods.

When comparing only the tensor methods, the tensor-GMRES method of Feng and Pulliam [17] is more efficient than the TK3 and TK2+ methods. It appears that the step produced from the projected tensor model of the Feng–Pulliam method is nearly the same as the steps calculated from the more precise local models of the tensor-Krylov methods, and it is less expensive to compute. This behavior may be understood more clearly by investigating the iteration history of all of the methods, which we describe next.

To investigate the step quality of each tensor-Krylov method, we tested the Bratu problem with $\lambda = 6.806652$, which has a condition number of roughly 1.5×10^6. Using right preconditioning, the results in the first plot of Figure 3.6 show the faster outer (nonlinear) iterations of the three tensor methods, which display very similar performance. All methods start by exhibiting linear convergence until the respective

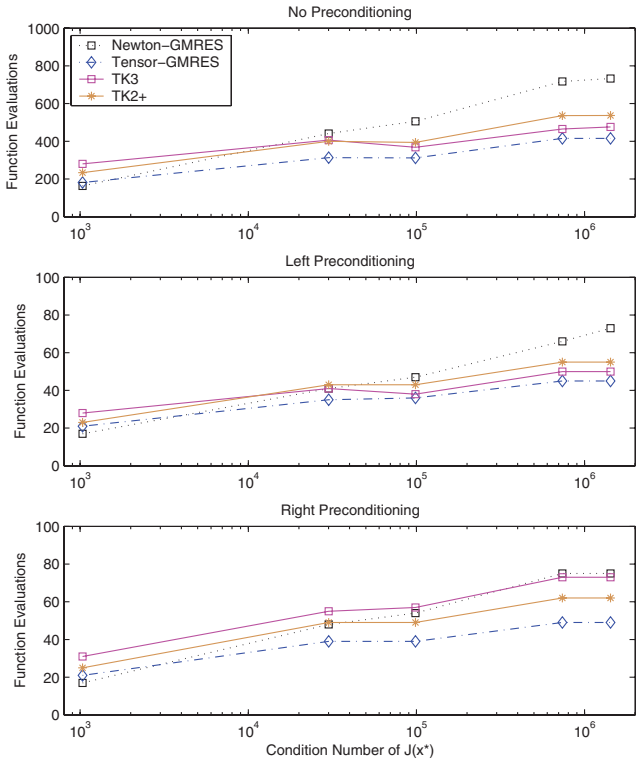

FIG. 3.5. *Effects of ill-conditioning on the inexact algorithms as seen in the Bratu problem.*

method can overcome the near singularity and accelerate convergence. The tensor methods accelerate convergence sooner than Newton-GMRES (iteration 2 versus iteration 5), and this is typical behavior for ill-conditioned problems—Newton-type methods branch into superlinear convergence later and later as the ill-conditioning grows. Because the forcing term for the inner iterative method is constant (instead of decreasing each iteration), all methods exhibit asymptotic linear convergence, as evidenced by their straight trajectories near the solution.

When we consider function evaluations in the bottom plot, the tensor-Krylov methods separate from the Feng–Pulliam method. The block size of the method is a clear indicator of the relative efficiency of the method. Specifically, tensor-GMRES, which uses the scalar (block size one) implementation of GMRES, is more efficient than TK2+ (block size two) and TK3 (block size three). The Bratu problem is unique among our tests in that the steps computed from a projected local tensor model are of roughly the same quality as the steps from TK2+ and TK3. Therefore, the number of outer iterations is the same, but because tensor-GMRES is more efficient in solving the local model, tensor-GMRES has the advantage.

3.2.2. Modified discrete boundary value problem. The discrete boundary value problem is a simple test problem from [25]. The standard discrete boundary value problem is

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + \tfrac{1}{2}h^2(x_i + t_i + 1)^3, \quad 1 \le i \le n, \tag{3.3}$$

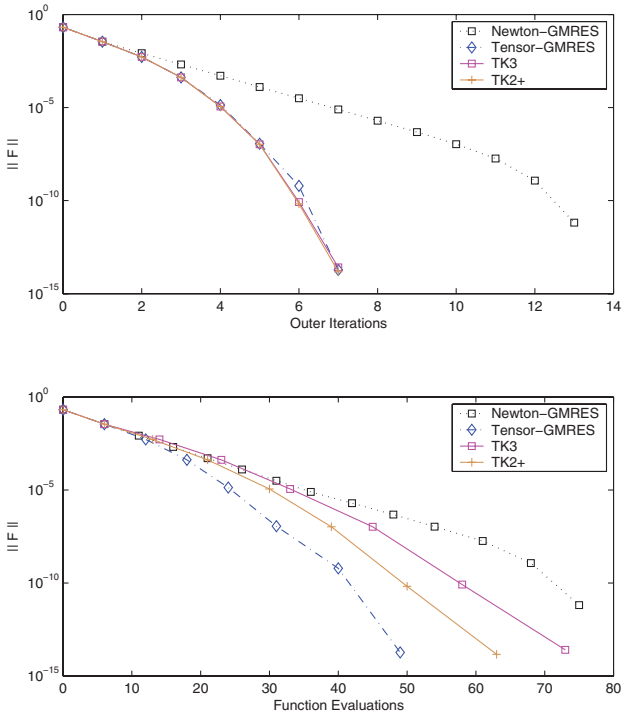

FIG. 3.6. *Example iteration history on the Bratu problem at* $\lambda = 6.806652$.

where $h = \frac{1}{n+1}$, $t_i = ih$, and $x_0 = x_{n+1} = 0$. The initial approximate solution is zero on a problem size of $n = 100$ equations. For our tests, we have modified (3.3) in accordance with (3.1) of section 3.1.

When testing this problem, we used a preconditioner that corresponds to the Jacobian of (3.3) but without the term t_i. We computed Jacobian-vector products using an analytic evaluation of the Jacobian, and we tallied the number of "function evaluation equivalents," which is the sum of function evaluations and Jacobian-vector products. If these products were approximated by first-order forward differences, as is generally the case with complex problems, then "function evaluation equivalents" would be equal to the number of function evaluations. This number also provides a relative measure of overall work for each algorithm.

Figure 3.7 presents the results of these tests, comparing the number of function evaluation equivalents computed for each method using the three choices of preconditioning (none, left, and right preconditioning) for the various values of $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \ldots$, in (3.1). If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots span the range of 1×10^0 to about 2×10^6 (left preconditioning) or 2×10^8 (right preconditioning) instead of 4×10^3 to 2×10^9 for the Jacobian, not counting the singular case.

The results for the modified discrete boundary value problem are similar to those of the Bratu problem in Figure 3.5: Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned, whereas the number of function evaluations required by the tensor methods increases to a lesser extent. It

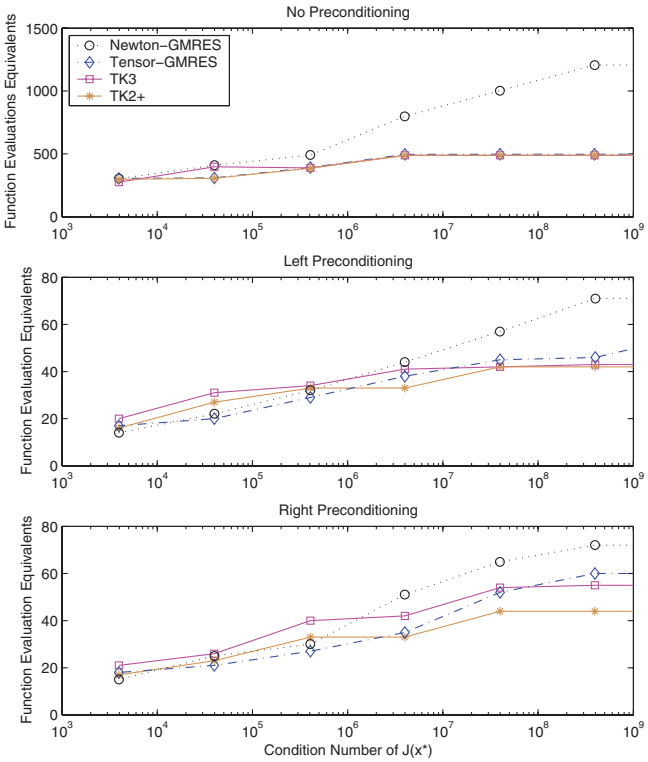

FIG. 3.7. *Effects of ill-conditioning on the inexact algorithms as seen in the discrete boundary value problem.*

is interesting to note that the tensor methods are virtually identical in performance without preconditioning and resemble the results in section 3.1. However, once preconditioning is used, the tensor methods are affected by the ill-conditioning of the problem, yet still to a lesser extent than Newton-GMRES.

When comparing only the tensor methods while using left and right preconditioning, the tensor-GMRES method of Feng and Pulliam is more efficient than the TK3 and TK2+ methods on the easier problems but less efficient on the hardest problems, which is different from the experience with the Bratu problem above. We explain this relative difference by noting that tensor-GMRES required more nonlinear iterations as the problem grew more ill-conditioned. Because the tensor-Krylov methods generally require more Arnoldi (inner) iterations to solve the local tensor model due to the less efficient block-Arnoldi process, any computational savings must come from fewer nonlinear (outer) iterations.

To better illustrate this behavior, Figure 3.8 presents the iteration history of all methods for one test case in Figure 3.7. We present the results of the modified problem using right preconditioning with $\lambda = 1$, which corresponds to the rightmost set of points in the bottom plot of Figure 3.7. Here, the top plot of Figure 3.8 shows that Newton-GMRES has linear convergence, and after one good step on the second iteration, tensor-GMRES appears to have linear convergence that is slightly faster than that of Newton-GMRES. The TK2+ and TK3 methods appear to have superlinear convergence with nearly the same steps. This profile indicates that the

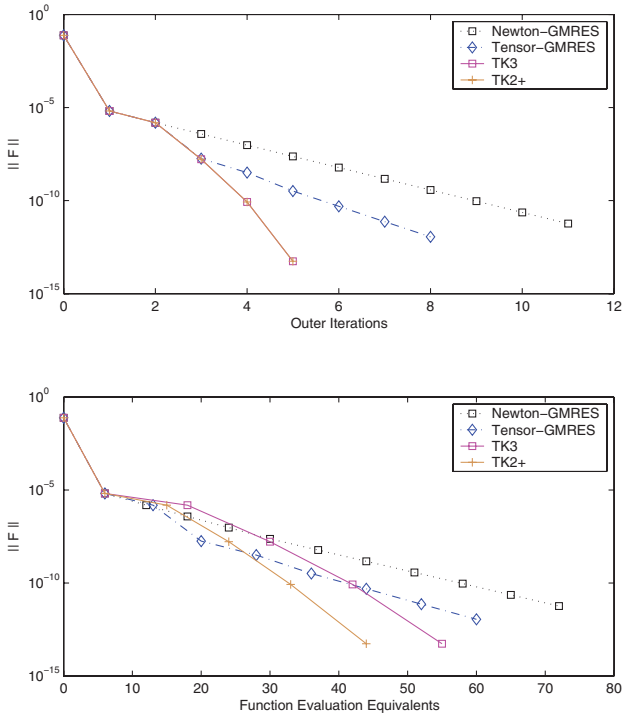

FIG. 3.8. *Example iteration history on the discrete boundary value problem at* $\lambda = 1$.

tensor step found from a projected tensor model is not as good for this problem as the tensor-Krylov steps which use the full tensor model. That is, the projection of the tensor model loses some information that is important for achieving superlinear convergence.

When we consider function evaluations in the bottom plot of Figure 3.8, two features are evident. First, the slopes of the Newton-GMRES and tensor-GMRES lines are nearly identical, which indicates that the cost of solving the projected tensor model in terms of function evaluations is roughly the same as the cost of solving the Newton model in Newton-GMRES. Second, the TK2+ and TK3 trajectories separate, which indicates that the block-2+ local solver in TK2+ is more efficient than the block-3 solver in TK3.

3.2.3. Modified Broyden tridiagonal problem. The Broyden tridiagonal problem is another test problem from [25]. The function is defined as

$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1, \quad 1 \le i \le n, \tag{3.4}$$

where $x_0 = x_{n+1} = 0$. Here again, we have modified the problem to be ill-conditioned, but in this case, we modify the last function to be

$$f_n(x) = (1 - \lambda)\left[(3 - 2x_n)x_n - x_{n-1} + 1\right] + \lambda\left[(3 - 2x_n)x_n - x_{n-1} + 1\right]^2.$$

This modification is similar to the problem studied by Feng and Pulliam in [17] and makes the problem more ill-conditioned as λ asymptotically approaches one, and singular at $\lambda = 1$ with a rank $n - 1$. We used the values $\lambda = 1 - 10^{-j}$, $j = 0, 1, 2, \ldots,$

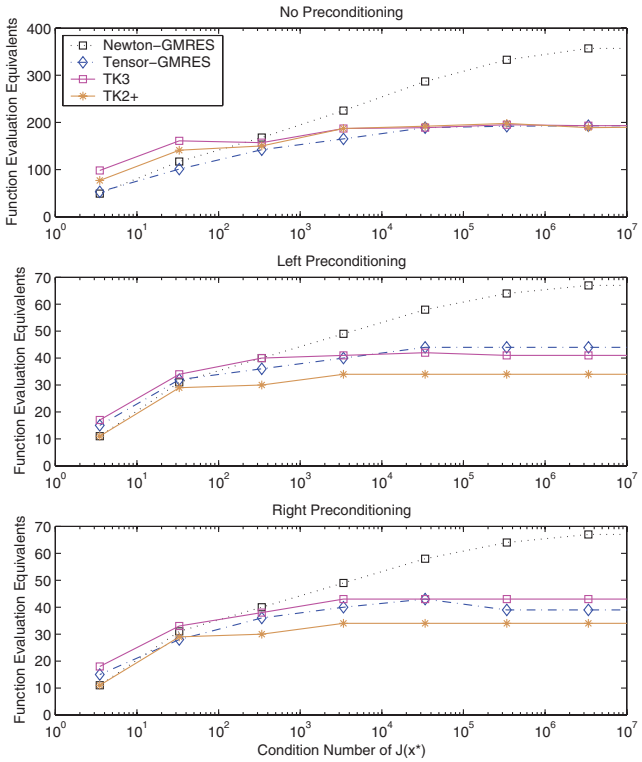

FIG. 3.9. *Effects of ill-conditioning on the inexact algorithms as seen in the modified Broyden tridiagonal problem.*

until the results were indistinguishable from $\lambda = 1$ due to round-off error, at which point we just use $\lambda = 1$. For our tests, we set $n = 100$ and used $(-1, \ldots, -1)^T$ as the starting vector. We used the Jacobian of (3.4) as our preconditioner for all values of λ.

Figure 3.9 presents the results of the ill-conditioning tests over the span of problems using $\lambda \in [0, 1]$, which affects the condition number of $J(x_*)$. The three plots represent the three choices of preconditioning (none, left, and right preconditioning), and each compares the number of function evaluation equivalents computed for the different methods. If one considers the condition number of the preconditioned Jacobian at the root (i.e., $M^{-1}J(x_*)$ or $J(x_*)M^{-1}$), then the condition numbers in the bottom two plots (not counting the singular case) span the range of 1×10^0 to 2×10^6 instead of 3×10^0 to 3×10^6 for the Jacobian.

Here again the results show that Newton-GMRES requires increasingly more function evaluations as the problem becomes more ill-conditioned. The tensor methods are affected somewhat at the low condition numbers, but there is a plateau where the number of function evaluations required by the tensor methods are no longer affected by the ill-conditioning of the problem. Near $\lambda = 1$, Newton-GMRES requires almost twice the number of function evaluations as the tensor methods.

When comparing only the tensor methods while using left and right preconditioning, the tensor-GMRES method of Feng and Pulliam is roughly comparable to TK3. The TK2+ method is more efficient on all problems except one, where tensor-GMRES is the best. In comparison with Newton-GMRES on the easier problems with left or

right preconditioning, TK2+ requires the same number of or fewer function evaluation equivalents.

Figure 3.10 presents the iteration history of all methods for the rightmost test case ($\lambda = 1$) in the right preconditioning plot in Figure 3.9. The top plot of Figure 3.10 shows that Newton-GMRES has linear convergence on this singular problem, and the tensor methods have superlinear convergence. Once again, the two tensor-Krylov methods have the same quality of steps, but the tensor-GMRES method no longer shares the same trajectory as TK2+ and TK3, which indicates that the projected tensor model loses some critical directional information.

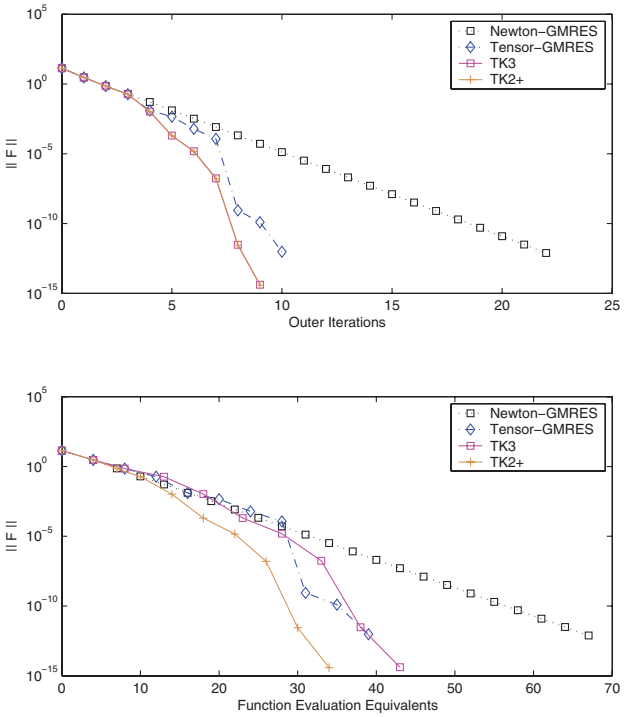

FIG. 3.10. *Example iteration history on the modified Broyden tridiagonal problem at* $\lambda = 1$.

The bottom plot of Figure 3.10 considers the function evaluation equivalents on the x-axis. It shows that tensor-GMRES is generally more efficient than TK3 at solving the local model. Thus, while TK3 is more efficient in total nonlinear iterations, tensor-GMRES is more efficient at solving the local model, which accounts for the difference. Another feature evident in the bottom plot is that the TK2+ and TK3 trajectories separate. As with the discrete boundary value problem above, this indicates that the block-2+ local solver in TK2+ is more efficient than the block-3 solver in TK3.

3.3. Large-scale problem solved with inexact methods. For our largest, most difficult problem, we consider the counterflow jet problem described in [27]. This fluid flow problem offers steady-state multiplicity and exhibits a pitchfork bifurcation, where the Jacobian is singular (rank $n - 1$).

The two-dimensional planar counterflow jet problem consists of a rectangular channel with two opposing inlet flows on the y-axis and two outlet flows on the x-axis. This fluid flow problem is set up using a particular spatial discretization of the continuity and Navier–Stokes equations describing laminar, isothermal flow of an incompressible Newtonian fluid. The particular spatial discretization that we use is from the finite element reacting flow code MPSalsa [33].

The finite-element discretization utilized 6,321 bilinear elements corresponding to 18,963 unknowns and had a fixed aspect ratio of 1.0. For reasons related to coupling external solvers to MPSalsa, the aspect ratio was constrained to the geometry in the original mesh file and could not be scaled with a parameter to achieve different aspect ratios as was done in [27]. Consequently, not all experimental conditions are the same.

As the Reynolds number of the injected fluid changes, multiple steady states may exist. The point at which a single steady state changes to multiple steady states is a bifurcation. We used the Library of Continuation Algorithms (LOCA) [32, 31] to identify a pitchfork bifurcation at $Re^* = 872.20306885$ for our particular geometry and mesh. To investigate the effects of ill-conditioning on the methods, we increased Re from 100 to Re^*, which increased the condition number of $J(x_*)$ from about 10^4 to 10^{15}. Differences in performance among the algorithms were not seen until $Re = 872.19$.

We implemented the algorithms in NOX [23], which is a C++ object-oriented nonlinear solver package developed at Sandia National Laboratories. For objective comparisons, all of the methods, including Newton-GMRES and tensor-GMRES, used the same Arnoldi process (modified Gram–Schmidt) as the tensor-Krylov methods.

We used an explicit Jacobian, which our PDE code computed efficiently by a combination of analytic evaluation and numerical differentiation, and enabled the option for maximum accuracy in the Jacobian. We employed right preconditioning in all cases using an ILUT preconditioner [29].

We used a standard backtracking linesearch procedure for Newton-GMRES and used the complete tensor-GMRES algorithm in [17], including their globalization. For the tensor-Krylov methods, we report results of the standard linesearch, but the curvilinear linesearch [3] provided similar performance. For selecting the linesearch parameter at each trial step, we used interval halving (dividing the search direction in half at each inner iteration).

To get the Jacobian as close to singular as possible, the problem was solved as accurately as possible. We used the following stopping condition: $\|F(x_k)\|_2 \leq 10^{-15} \|F(x_0)\|_2$. In all other respects, we used the same conditions and parameters as before. The initial approximation was the zero vector for all cases.

Figure 3.11 shows that the tensor-Krylov methods require 6–10 iterations to solve the range of problems. On the other hand, Newton-GMRES requires nearly twice the iterations at the pitchfork bifurcation, and tensor-GMRES failed to solve that problem. All methods had identical performance until very near the pitchfork bifurcation as the condition exceeded 10^{10}. TK2+ required four linesearch iterations at 10^{11}, hence the higher number of function evaluations.

Figure 3.12 shows the convergence history for all methods at the pitchfork bifurcation. The tensor-Krylov methods share roughly the same level of performance. Tensor-GMRES, on the other hand, stalls after about 12 orders of magnitude improvement in $\|F\|$. This is not due to excessive linesearches because in most instances a full step is taken; tensor-GMRES just loses directional information here. Newton-GMRES behaves as we have seen previously with smaller problems—roughly linear

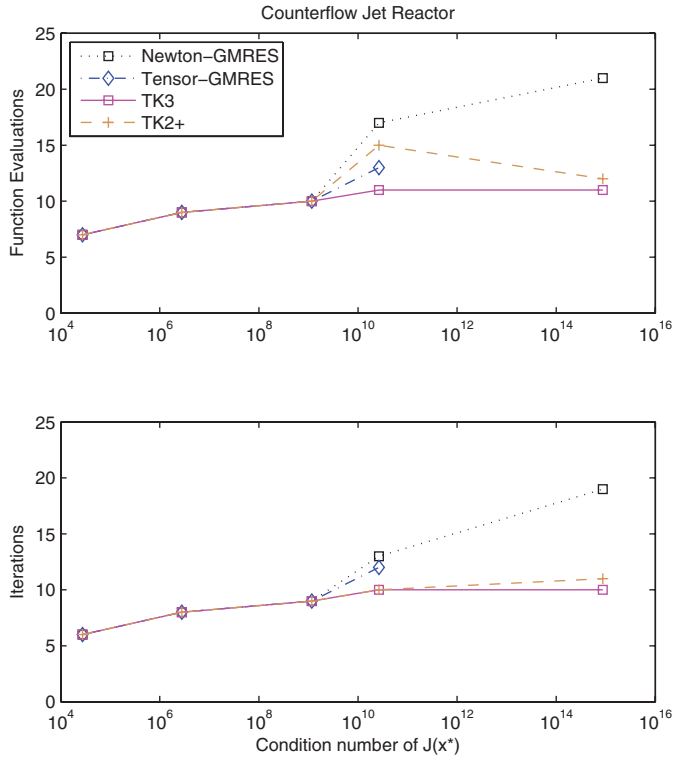

FIG. 3.11. *Effects of ill-conditioning in the counterflow jet reactor problem.*

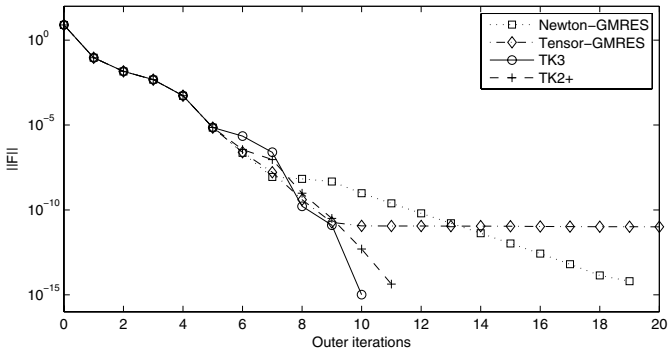

FIG. 3.12. *Iteration history of the counterflow jet reactor problem at the pitchfork bifurcation.*

convergence that is slower than that of the tensor methods (tensor-GMRES notwithstanding).

4. Summary and conclusions. This paper has investigated the performance of tensor methods (both direct and Krylov-based inner solvers) on small- to large-scale problems over a range of conditionings, from well-conditioned to ill-conditioned to singular. Our results showed that tensor methods outperform Newton-based methods as the problems become more ill-conditioned. Prior to this investigation, studies on

direct tensor methods focused only on singular problems or on general problems that are well-conditioned.

Specifically, our results show that, eventual quadratic convergence notwithstanding, the performance of Newton's method will degrade as the ill-conditioning grows, whereas tensor methods appear to be relatively unaffected or only mildly affected (in the case of larger rank deficiencies). Newton-based methods do not handle these singular problems well because they converge linearly to the solution and, in some cases, with poor accuracy.

For the large-scale methods, despite the use of an iterative inner method with an approximate solve, the tensor-Krylov methods appear to retain superlinear convergence properties on ill-conditioned problems. On the other hand, Newton-GMRES is affected by the ill-conditioning and branches into superlinear convergence later and later as the problems become more ill-conditioned. Thus, tensor methods are especially useful for large-scale problems that are highly ill-conditioned or singular, where Newton-based algorithms exhibit very slow convergence.

In terms of the various tensor methods, TK3 and TK2+ generally have the best performance in terms of nonlinear iterations. When considering total function evaluations, then TK2 and TK2+ outperform TK3. Feng and Pulliam's tensor-GMRES method is very competitive but could not solve the large-scale problem at the bifurcation point. A thorough comparison of these methods on other fluid flow problems is in [2].

There are many important and practical problems that have ill-conditioned or singular Jacobian matrices at the solution, such as large PDE problems that exhibit shocks, turning points, and/or pitchfork bifurcations. The conclusions of this research indicate that concepts from tensor methods may benefit algorithms for bifurcation tracking and stability analysis. We intend to investigate these applications in future research.

Acknowledgments. We thank Roger Pawlowski for the counterflowing jet problem in MPSalsa, and we are grateful to Eric Phipps for assistance with finding the pitchfork bifurcation using LOCA. We also thank the two anonymous referees for carefully reading the manuscript and suggesting important improvements.

REFERENCES

[1] B. W. Bader, *Tensor-Krylov Methods for Solving Large-Scale Systems of Nonlinear Equations*, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, 2003.

[2] B. W. Bader, *Tensor-Krylov methods for solving large-scale systems of nonlinear equations*, SIAM J. Numer. Anal., 43 (2005), pp. 1321–1347.

[3] B. W. Bader and R. B. Schnabel, *Curvilinear linesearch for tensor methods*, SIAM J. Sci. Comput., 25 (2003), pp. 604–622.

[4] A. Bouaricha, *Solving Large Sparse Systems of Nonlinear Equations and Nonlinear Least Squares Problems Using Tensor Methods on Sequential and Parallel Computers*, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, 1992.

[5] P. N. Brown and A. C. Hindmarsh, *Reduced storage methods in stiff ODE systems*, J. Appl. Math. Comput., 31 (1989), pp. 40–91.

[6] P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[7] P. N. Brown and Y. Saad, *Convergence theory of nonlinear Newton–Krylov algorithms*, SIAM J. Optim., 4 (1994), pp. 297–330.

[8] T. F. Chan and K. R. Jackson, *The use of iterative linear-equation solvers in codes for large systems of stiff IVPs for ODEs*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 378–417.

[9] D. W. Decker, H. B. Keller, and C. T. Kelley, *Convergence rates for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 296–314.

[10] D. W. Decker and C. T. Kelley, *Newton's method at singular points.* I, SIAM J. Numer. Anal., 17 (1980), pp. 66–70.
[11] D. W. Decker and C. T. Kelley, *Newton's method at singular points.* II, SIAM J. Numer. Anal., 17 (1980), pp. 465–471.
[12] D. W. Decker and C. T. Kelley, *Convergence acceleration for Newton's method at singular points*, SIAM J. Numer. Anal., 19 (1981), pp. 219–229.
[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
[14] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[15] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[16] D. Feng, P. D. Frank, and R. B. Schnabel, *Local convergence analysis of tensor methods for nonlinear equations*, Math. Programming, 62 (1993), pp. 427–459.
[17] D. Feng and T. H. Pulliam, *Tensor-GMRES method for large systems of nonlinear equations*, SIAM J. Optim., 7 (1997), pp. 757–779.
[18] R. Glowinski, H. B. Keller, and L. Reinhart, *Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 793–832.
[19] A. Griewank, *On solving nonlinear equations with simple singularities or nearly singular solutions*, SIAM Review, 27 (1985), pp. 537–563.
[20] A. Griewank and M. R. Osborne, *Analysis of Newton's method at irregular singularities*, SIAM J. Numer. Anal., 20 (1983), pp. 747–773.
[21] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
[22] C. T. Kelley and R. Suresh, *A new acceleration method for Newton's method at singular points*, SIAM J. Numer. Anal., 20 (1983), pp. 1001–1009.
[23] T. Kolda and R. Pawlowski, *NOX: An Object-Oriented Nonlinear Solver Package*, http://trilinos.sandia.gov/packages/nox/ (24 May 2007).
[24] J. J. Moré, *A collection of nonlinear model problems*, in Computational Solution of Nonlinear Systems of Equations, Lectures in Appl. Math. 26, AMS, Providence, RI, 1990, pp. 723–762.
[25] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.
[26] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[27] R. P. Pawlowski, A. G. Salinger, J. N. Shadid, and T. J. Mountziaris, *Bifurcation and stability analysis of laminar isothermal counterflowing jets*, J. Fluid Mech., 551 (2006), pp. 117–139.
[28] G. W. Reddien, *On Newton's method for singular problems*, SIAM J. Numer. Anal., 15 (1978), pp. 993–996.
[29] Y. Saad, *ILUT: A dual threshold incomplete ILU preconditioner*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[30] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.
[31] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. T. Phipps, and L. A. Romero, *Bifurcation tracking algorithms and software for large scale applications*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 15 (2005), pp. 1015–1032.
[32] A. G. Salinger, N. M. Bou-Rabee, R. P. Pawlowski, E. D. Wilkes, E. A. Burroughs, R. B. Lehoucq, and L. A. Romero, *LOCA* 1.0*: Library of Continuation Algorithms: Theory and Implemenation Manual*, Technical report SAND2002-0396, Sandia National Laboratories, Albuquerque, NM, 2002.
[33] A. G. Salinger, K. D. Devine, G. L. Hennigan, H. K. Moffat, S. A. Hutchinson, and J. N. Shadid, *MPSalsa, A Finite Element Computer Program for Reacting Flow Problems, Part* 2*—User's Guide*, Technical report SAND96-2331, Sandia National Laboratories, Albuquerque, NM, 1996.
[34] R. B. Schnabel and P. D. Frank, *Tensor methods for nonlinear equations*, SIAM J. Numer. Anal., 21 (1984), pp. 815–843.