SANDIA REPORT

SAND2004-1837
Unlimited Release
Printed August 2004

Tensor-Krylov Methods for Solving Large-Scale Systems of Nonlinear Equations

Brett W. Bader

Prepared by Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2004-1837
Unlimited Release
Printed August 2004

Tensor-Krylov Methods for Solving Large-Scale Systems of Nonlinear Equations

Brett W. Bader
Computational Sciences Department
Sandia National Laboratories
P.O. Box 5800, MS-0316
Albuquerque, NM 87185-0316

Abstract

This paper develops and investigates iterative tensor methods for solving large-scale systems of nonlinear equations. Direct tensor methods for nonlinear equations have performed especially well on small, dense problems where the Jacobian matrix at the solution is singular or ill-conditioned, which may occur when approaching turning points, for example. This research extends direct tensor methods to large-scale problems by developing three tensor-Krylov methods that base each iteration upon a linear model augmented with a limited second-order term, which provides information lacking in a (nearly) singular Jacobian. The advantage of the new tensor-Krylov methods over existing large-scale tensor methods is their ability to solve the local tensor model to a specified accuracy, which produces a more accurate tensor step. The performance of these methods in comparison to Newton-GMRES and tensor-GMRES is explored on three Navier–Stokes fluid flow problems. The numerical results provide evidence that tensor-Krylov methods are generally more robust and more efficient than Newton-GMRES on some important and difficult problems. In addition, the results show that the new tensor-Krylov methods and tensor-GMRES each perform better in certain situations.

Acknowledgment

Special thanks is owed to Prof. Robert Schnabel for his guidance and supervision of this research. We thank Tamara Kolda and Roger Pawlowski of Sandia National Laboratories for the opportunity to code these algorithms into NOX and for the use of their computing resources for the numerical tests.

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 while the author was in the Department of Computer Science at the University of Colorado, Boulder.

While at Sandia, the author also received support from DOE's Office of Science MICS program and Sandia's Laboratory Directed Research and Development program under LDRD 04-1024.

Contents

Figures

Tables

Tensor-Krylov Methods for Solving Large-Scale Systems of Nonlinear Equations

1 Introduction

This paper describes a new class of methods for solving the nonlinear equations problem

$$\text{given } F : \mathbb{R}^n \to \mathbb{R}^n, \quad \text{find } x_* \in \mathbb{R}^n \text{ such that } F(x_*) = 0, \tag{1}$$

where it is assumed that $F(x)$ is at least once continuously differentiable. Large-scale systems of nonlinear equations defined by (1) arise in many practical situations, including systems produced by finite-difference or finite-element discretizations of boundary value problems for ordinary and partial differential equations.

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 current practical approaches for solving large problems involves approximately solving a local linear model and then using these "inexact" steps to locate the next point.

Two inexact versions of tensor methods already exist for solving large problems. Bouaricha [5] describes an implementation of a tensor method using Krylov subspace methods for linear equations, which involves constructing an inexact tensor step from the approximate solutions of two linear systems (with the same Jacobian matrix). In addition, Feng and Pulliam [16] have developed a "tensor-GMRES" method, which first finds the Newton-GMRES step and then solves for an approximate tensor step.

We propose three variants of a new approach for solving the large-scale nonlinear equations problem (1). These new methods are an extension of the class of standard tensor methods [26], which 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. Specifically, the new algorithms are an amalgamation of various techniques, including tensor methods for nonlinear equations [26], Krylov subspace techniques [8], and an inexact solver framework [13], that make them well-suited for large-scale problems. Given the parallels to Newton-Krylov methods [8], the new algorithms may be called "tensor-Krylov" methods. In a manner similar to Newton-GMRES, 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 step.

The key feature of these new methods is that the step satisfies the local tensor model to within a specified tolerance, making it possible to control the quality of the

step. In addition, the new tensor-Krylov methods are aptly suited to target problems where the Jacobian at the root is singular or, at least, very ill-conditioned. Newton-based methods do not handle singular problems well because they converge linearly to the solution and, in some cases, with poor accuracy [10, 11, 12, 19]. On the other hand, tensor methods are superlinearly convergent on singular problems under mild conditions [15].

Because the tensor-Krylov methods borrow elements from both direct tensor methods and linear Krylov subspace methods, these topics are reviewed before introducing the tensor-Krylov formulations. Section 2 reviews direct tensor methods for solving small-scale systems of nonlinear equations and includes background on pertinent large-scale methods, namely linear Krylov subspace methods, Newton-GMRES, and existing large-scale tensor methods. Then section 3 describes three different approaches for solving the local tensor model using a Krylov-based method and wraps these local Krylov solvers into a large-scale method with options for various global strategies. With the complete tensor-Krylov nonlinear solver fully discussed, section 4 describes several fluid flow benchmark problems that serve as ambitious test problems. Finally, section 5 makes some concluding remarks and discusses directions for future research.

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 on J_k, F_k, a_k, and s_k while still referring to the current values at an iteration.

2 Background and review

In this section, we introduce standard methods for solving systems of nonlinear equations. We provide a brief review of standard methods in section 2.1 and a short introduction to tensor methods in section 2.2. We extend these methods to large-scale problems in sections 2.3 and 2.4 by reviewing Newton-Krylov methods and existing large-scale tensor methods, respectively. General references for topics in nonlinear solvers include [14], [18], and [23].

2.1 Standard methods

In this paper, we will refer to a class of methods, which we will call standard methods, for solving (1) that are based on a linear local model. Most notable among these methods is Newton's method, which bases each iteration upon a linear local model $M_N(x_k + d)$ of the function $F(x)$ around the current iterate $x_k \in \mathbb{R}^n$:

$$M_N(x_k + d) = F(x_k) + J(x_k)d, \tag{2}$$

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

$$d_N = -J(x_k)^{-1} F(x_k),$$

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,

$$x_{k+1} = x_k + d_N. \tag{3}$$

Due to to large arithmetic and storage costs, implementations of Newton's method using direct factorizations of $J(x_k)$ are not practical for large-scale problems.

2.2 Tensor methods

Tensor methods solve (1) by including more information in the local model than Newton's method. By solving this augmented local model, tensor methods tend to generate steps of better quality than standard methods, thus reaching the solution faster. The local tensor model has the generic form

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} T_k dd, \tag{4}$$

where $T_k \in \mathbb{R}^{n \times n \times n}$ is a tensor, which includes second-order information and is where these methods get their name. This term is selected so that the model interpolates $p \leq \sqrt{n}$ previous function values in the recent history of iterates, which makes T_k a rank p tensor. Most often p is 1 or 2, but computational evidence in [26] suggests that $p > 1$ actually adds little to the computational performance of the direct method.

For this paper, we focus on the case of $p = 1$ because the tensor-Krylov methods only use one secant update. In this case, the tensor model about x_k reduces to

$$M_T(x_k + d) = F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2, \tag{5}$$

where

$$a_k \in \mathbb{R}^n = \frac{2(F_{k-1} - F_k - J_k s_k)}{(s_k^T s_k)^2}, \tag{6}$$

$$s_k \in \mathbb{R}^n = x_{k-1} - x_k. \tag{7}$$

After forming the model, we use it to determine the step to the next trial point. Because (5) may not have a root, one solves the minimization subproblem

$$\min_{d \in \mathbb{R}^n} \| M_T(x_k + d) \|_2, \tag{8}$$

and a root or minimizer of the model is the tensor step. Due to the special form of (5), the solution of (8) in the nonsingular case reduces to solving a quadratic equation followed by solving a system of $n-1$ linear equations in as many unknowns.

A practical approach for solving (8), which relates to the presentation of tensor-Krylov methods in section 3, uses two orthogonal transformations to reduce the problem to two subproblems that are more easily solved. Briefly, the first transformation finds an orthogonal $Q_1 \in \mathbb{R}^{n\times n}$, such that $s/\|s\|$ is the last column and permits a change in variables

$$d = Q_1 \hat{d}.$$

The second transformation finds an orthogonal $Q_2 \in \mathbb{R}^{n\times n}$, such that $Q_2 J_k Q_1$ is upper triangular. Thus, applying the two transformations to (5) and setting it equal to zero, yields the following triangular system of n equations in n unknowns

$$Q_2 F + Q_2 J Q_1 \hat{d} + \tfrac{1}{2} Q_2 a \|s\|^2 \hat{d}_n^2 = 0, \tag{9}$$

where $\hat{d}_n \in \mathbb{R}$ is the quadratic variable.

Then, breaking (9) into two smaller problems, the solution to (8) continues by first solving for $\hat{d}_n$ by minimizing the quadratic equation appearing in the last row of (9) and choosing the smaller magnitude minimizer if there are two. Using the value of $\hat{d}_n$ in (9), a triangular linear system of size $(n-1)\times(n-1)$ is revealed. Finally, the complete solution to (8) is found by solving this resultant system for the remaining components of $\hat{d}$ and then reversing the variable space transformation from the first step, $d = Q_1 \hat{d}$.

2.3 Newton-Krylov methods

Up to this point, this review has discussed direct methods for the solution of small, dense problems, such that the local model is solved using direct factorizations of the Jacobian matrix. Large, sparse systems are often successfully solved using a class of "inexact" Newton methods:

$$x_{k+1} = x_k + d_k, \quad \text{where} \quad J(x_k)d_k = -F(x_k) + r_k, \quad \|r_k\| \le \eta_k \|F(x_k)\|, \tag{10}$$

where the local model typically is solved only approximately at each step using a less expensive approach. Successively better approximations at each iteration 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.

The most common methods for approximately solving the local Newton model are Krylov-based methods. 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 Krylov 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) [24], 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, where the linear system $J_k d = -F_k$ is solved via GMRES according to the tolerance η in (10). Krylov methods have the appeal of requiring almost no matrix storage 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 algorithm for comparisons in our numerical experiments.

2.4 Previous large-scale, sparse tensor methods

The large-scale tensor methods described in this paper are not the first tensor methods aimed at large-scale problems. Two other methods have been proposed, and we discuss them now.

Bouaricha [5] describes a large-scale implementation of a tensor method using Krylov subspace methods for linear equations (GMRES and FOM), which involves constructing an inexact tensor step from the approximate solutions of $J^{-1}F_k$ and $J^{-1}F_{k-1}$. The approach involves finding the values $s^T J^{-1}F_k$ and $s^T J^{-1}a_k$, which are used to calculate an approximate value of $s^T d_T$, which multiplies $J^{-1}a_k$ in the final computation of the step. More precise details regarding an efficient implementation may be found in [1] or [5].

Despite some favorable results in [5], we have found Bouaricha's method to be not as competitive on more practical problems. The two main disadvantages of this method stem from the fact that two linear systems must be solved for each outer iteration and that an accurate value of β is not calculated, which may lead to spurious steps. Due to these theoretical disadvantages and based upon our own numerical experience with the algorithm in [1], we will not consider any numerical comparisons with Bouaricha's algorithm.

Feng and Pulliam [16] describe another large-scale tensor method, which they call "tensor-GMRES." It uses Krylov subspace projection techniques for solving the Newton equations; and, in particular, it uses GMRES to find the approximate Newton step $d_N = d_0 + V_m y_m$. The Arnoldi process in GMRES generates a Hessenberg

matrix H_m and an orthonormal basis for the Krylov subspace $\mathcal{K}_m$ in the columns of V_m. Given these key matrices, their algorithm proceeds to solve a projected version of the tensor model (5) along a subspace that spans the Newton step direction (i.e., 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

$$\min_{d \in \{d_0\} \cup \mathcal{K}_m^N} \left\| F_k + J_k d + \tfrac{1}{2} P a (s^T d)^2 \right\|, \tag{11}$$

where P is the projection matrix $P = Y(Y^T Y)^{-1} Y^T$ and $Y = J_k[V_m, d_0]$.

The algorithm has some difficult algebra (details may be found in [16]), but the design is actually 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 is also consistent with preconditioning techniques and a matrix-free implementation, which makes it appealing for general use.

The extra work and storage beyond the GMRES method is actually quite small, and the analysis in [16] shows that the same superlinear convergence properties for the unprojected tensor model considered in [15] also hold for the projected tensor model in (11). These properties are evident in the numerical results of [16], which show the superlinear convergence behavior of tensor-GMRES on the 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) spanned 20–55% on the simpler problems and 32–60% on the more difficult Euler problem.

However, there are a few potential disadvantages related to the Feng and Pulliam method. The variable space restriction on d in the minimization problem (11) illustrates a possible disadvantage, particularly when using preconditioners or restarted GMRES. The norm of the projected tensor model is only minimized to the extent that the Krylov subspace for the Newton-GMRES step is large enough to capture important directions in the tensor step. For example, consider using an *exact* preconditioner, i.e., J_k itself. One iteration of GMRES solves the Newton equations exactly, and the Newton step direction is along v_1, the first basis vector in V_m. Then, according to the Feng-Pulliam method, the approximate tensor step that solves (11) could only be a scalar multiple of the direction v_1 (assuming that $d_0 = 0$). A similar example may be developed when using restarted GMRES in the Feng-Pulliam method—if GMRES converges soon after a restart, then the orthonormal basis V_m is smaller than before the restart. A smaller basis may lead to a tensor step that solves (11) with more error due to fewer degrees of freedom.

Despite these hypothetical examples, it is unclear whether solving (11) in a smaller variable space will adversely affect the practical performance of this method when using preconditioners or restarted GMRES. Because the Newton step tends to un-

dershoot (or overshoot) when first-order information is lacking in the local model, the solution to the tensor model is often nearly along the Newton direction, so the subspace restriction on d might not be a problem. The fact remains, however, that the Feng-Pulliam method solves the *projected* tensor model (11), which loses some information in the projection. In addition, the relative stopping tolerance η_k in the Newton-GMRES step has no direct relationship with the error in the tensor model.

3 Tensor-Krylov methods

The new tensor-Krylov methods differ from previous large-scale tensor methods in their ability to solve the local tensor model to a specified tolerance. Using either the methods of Bouaricha or Feng and Pulliam, the residual error $\|M_T(x_k + d)\|$ must be computed explicitly, making it difficult to assess the quality of the approximate tensor step that they compute. In addition, the new methods avoid the costly solution of two linear systems (as in Bouaricha's method) and compute the solution to the full tensor model, as opposed to a projected tensor model (as in the Feng-Pulliam method).

In the same manner that GMRES is an algorithm for solving linear systems and Newton-GMRES is the nonlinear solver, we make a distinction between the solver for the local tensor model and the nonlinear solver. In this section, we describe three procedures for iteratively solving the local tensor model that use the concepts from linear Krylov subspace methods. We restrict ourselves to the rank-one tensor model in (5)–(7), which only interpolates the function value at the previous iterate. Because (5) may or may not have a root, we seek a solution to the minimization problem

$$\min_{d \in \mathcal{K}_m} \|M_T(x_k + d)\|_2 = \min_{d \in \mathcal{K}_m} \left\| F_k + J_k d + \tfrac{1}{2} a_k (s_k^T d)^2 \right\|_2 , \tag{12}$$

where $\mathcal{K}_m$ is a specially chosen Krylov subspace that facilitates the solution of the quadratic model. The three tensor-Krylov methods differ in their choice of $\mathcal{K}_m$, which becomes their signature difference and dictates the algorithm. We differentiate the three variants by the size of their initial block subspace, identifying them as block-2, block-2+, and block-3. The reason for considering three variants is related to their complexity and usefulness as a block algorithm. The block-3 method is the most straightforward and most likely the best block implementation, while the block-2 methods are more complex but may work better in scalar implementations.

In sections 3.1–3.3, we start with a description of the three Krylov-based techniques for solving the local tensor model. Due to space considerations, we describe only the block-2 method in detail and refer to [1] for more detailed information on the other two methods. Section 3.1 covers all aspects of the block-2 method that are important to a nonlinear equations solver, including block-Krylov subspace issues, residual calculation, stopping conditions, preconditioning and scaling techniques, computation of the Newton step, and cost. Section 3.4 wraps the local solver into a complete

tensor-Krylov algorithm for solving large-scale systems of nonlinear equations, and section 3.5 discusses the global strategies for the tensor-Krylov algorithm.

3.1 Block-2 method

The block-2 algorithm proceeds in a block-Krylov-like fashion, operating on a matrix of initial vectors V instead of the single residual vector of a linear system. In particular, this method uses a block-Krylov subspace composed of two initial vectors.

We begin by rearranging the local tensor model and noting that it looks like a linear system involving a linear combination of two right-hand sides:

$$Jd = -F_k - \tfrac{1}{2}a\beta^2, \tag{13}$$

where $\beta \equiv s^T d$. The right-hand side spans only the directions F_k and a; the vector s appears in the inner product $\beta = s^T d$, which is a scalar multiple for a that is unknown. Thus, the premise of the block-2 method is that we start with the initial block Krylov subspace $\mathcal{K}_0 = span\{a, F_k\}$ and build $\mathcal{K}_m = span\{a, F_k, Ja, JF_k, J^2a, J^2F_k, \dots\}$ to solve (12). Specifically, we consider the block of initial vectors

$$R_0 = [(Jd_0 + F_k),\ a], \tag{14}$$

where $d_0 \in \mathbb{R}^n$ is some initial guess for the step. Because the starting matrix R_0 uses the residual $F_k + Jd_0$, which depends on d_0, the block-2 method may be restarted with successively better initial guesses in a manner similar to restarted GMRES.

The first step of the algorithm computes the QR-factorization of R_0,

$$R_0 = VR = [v_1, v_2]R, \tag{15}$$

where $V \in \mathbb{R}^{n\times 2} = [v_1, v_2]$ is unitary and $R \in \mathbb{R}^{2\times 2}$ is upper triangular. A block-Arnoldi process then creates additional columns of an orthonormal basis V_m that spans the block-Krylov subspace

$$span\{V, JV, J^2V, J^3V, \dots\}. \tag{16}$$

There are several block-Arnoldi versions available for implementation, and the particular variant is not critical to the implementation of the tensor-Krylov method. The standard procedure works on a whole block $V \in \mathbb{R}^{n\times t}$ and adds t vectors—$t = 2$ in this case—to the subspace at a time. This procedure may work well when considering cache memory performance and may be considered in future research. However, we decided to implement the single-vector version of block-Arnoldi to more closely correspond with the scalar implementation of GMRES. The version in Algorithm 3.1 is very similar to the standard Arnoldi algorithm, which operates on a single vector at a time and is due to Ruhe [20] (see also [23]) for the symmetric case (block Lanczos).

Algorithm 3.1: BLOCK ARNOLDI PROCESS—RUHE'S VARIANT

1. Choose t initial orthonormal vectors $\{v_i\}_{i=1,\ldots,t}$.
2. Choose a number of Arnoldi iterations to perform and set to m.
3. For $k = 1, \ldots, m$:
   (a) Set $j := k + t - 1$
   (b) Compute $w := Jv_k$
   (c) For $i = 1, 2, \ldots, j$
      i. $h_{ik} := (w, v_i)$
      ii. $w := w - h_{ik}v_i$
   (d) $h_{j+1,k} := \|w\|_2$
   (e) If $h_{j+1,k} \neq 0$, then set $v_{j+1} := w/h_{j+1,k}$;
   Else if $t = 1$, then Stop;
   Else set $t := t - 1$ and continue.

The first step of the algorithm is to multiply a single vector, v_1, by the Jacobian matrix J and orthonormalize the resulting vector w against all j vectors $v_1, \ldots, v_j$ ($j = t$ at the first iteration) in the orthonormal basis, building the subspace one vector at a time. Thus, a vector from the initial block $\{v_i\}_{i=1,\ldots,t}$ is multiplied by J every t steps. The last step 3e avoids a division by zero and is commonly referred to as the breakdown condition. In the scalar case ($t = 1$), a breakdown condition indicates that the solution is in the subspace spanned by the k basis vectors computed thus far. Here in the block case, we modify the usual condition to reduce the block dimension by one until it eventually reduces to the scalar case.

After m steps on the initial matrix $V \in \mathbb{R}^{n\times 2}$ defined in (15), the block-Arnoldi process produces an orthogonal matrix $V_{m+2} \in \mathbb{R}^{n\times(m+2)}$ and a matrix $\bar{H}_m \in \mathbb{R}^{(m+2)\times m}$ whose nonzero entries are the elements h_{ik} computed in the process. It is important to note that $\bar{H}_m$ is banded upper Hessenberg with two subdiagonals. The orthonormal basis V_{m+2} and the matrix $\bar{H}_m$ have an important relationship,

$$JV_m = V_{m+2}\bar{H}_m. \tag{17}$$

Continuing with the solution to (12), let the approximate solution at the mth iteration be

$$d = d_0 + V_m y, \tag{18}$$

where $V_m \in \mathbb{R}^{n\times m}$ is an orthonormal basis for the Krylov subspace generated in (16) and $y \in \mathbb{R}^m$ is unknown. Substituting (18) into the tensor model yields

$$\begin{aligned} M_T(x_k + d) &= F_k + Jd + \tfrac{1}{2}a(s^T d)^2 \\ &= F_k + Jd_0 + JV_m y + \tfrac{1}{2}a(s^T d_0 + s^T V_m y)^2 \\ &= r_0 + JV_m y + \tfrac{1}{2}a(s^T d_0 + s^T V_m y)^2, \end{aligned} \tag{19}$$

where $r_0 = F_k + Jd_0$, which is also the residual for the Newton model when using the initial guess d_0. Having r_0 permits the calculation of the approximate Newton step later in the algorithm.

Let $Q_1 \in \mathbb{R}^{m\times m}$ be an orthogonal matrix that has $V_m^T s / \left\|V_m^T s\right\|$ in the last column, and let the vector $\hat{y} \in \mathbb{R}^m$ be defined by the following transformation

$$y = Q_1 \hat{y}. \tag{20}$$

Then, using Q_1 and $\hat{y}$, the simplification of (19) continues

$$\begin{aligned} M_T(x_k + d) &= r_0 + JV_m y + \tfrac{1}{2}a(s^T d_0 + s^T V_m y)^2 \\ &= r_0 + JV_m Q_1 \hat{y} + \tfrac{1}{2}a(s^T d_0 + (V_m^T s)^T Q_1 \hat{y})^2 \\ &= r_0 + JV_m Q_1 \hat{y} + \tfrac{1}{2}a(s^T d_0 + (\left\|V_m^T s\right\| \hat{y}_m)^2, \end{aligned} \tag{21}$$

where $\hat{y}_m$ is the mth element of $\hat{y}$ and limits the quadratic part to a single unknown. After the next step, we will discuss a good choice for efficiently constructing the orthogonal matrix Q_1 that retains a desirable structure for solving this problem.

From (14) and (15), let $\bar{r}_1 \in \mathbb{R}^{m+2}$ and $\bar{a} \in \mathbb{R}^{m+2}$ denote the first and second columns of R, respectively, padded with m zeros to a length $m+2$. These definitions along with (17) permit a change in the function space of (21),

$$\begin{aligned} M_T(x_k + d) &= r_0 + JV_m Q_1 \hat{y} + \tfrac{1}{2}a(s^T d_0 + (\left\|V_m^T s\right\| \hat{y}_m)^2 \\ &= r_0 + V_{m+2}\bar{H}_m Q_1 \hat{y} + \tfrac{1}{2}a(s^T d_0 + (\left\|V_m^T s\right\| \hat{y}_m)^2 \\ &= V_{m+2}\left(\bar{r}_1 + \bar{H}_m Q_1 \hat{y} + \tfrac{1}{2}\bar{a}(s^T d_0 + (\left\|V_m^T s\right\| \hat{y}_m)^2\right). \end{aligned} \tag{22}$$

Because the column-vectors of V_{m+2} are orthonormal, the original least-squares problem of (12) may be simplified:

$$\min_{d\in\mathcal{K}_m} \left\|M_T(x_k + d)\right\|_2 = \min_{d\in\mathcal{K}_m} \left\|V_{m+2}^T M_T(x_k + d)\right\|_2, \tag{23}$$

where

$$V_{m+2}^T M_T(x_k + d) = \bar{r}_1 + \bar{H}_m Q_1 \hat{y} + \tfrac{1}{2}\bar{a}(s^T d_0 + (\left\|V_m^T s\right\| \hat{y}_m)^2. \tag{24}$$

At this point, we want to preserve some structure of the problem by requiring that the product $\bar{H}_m Q_1$ does not need expensive updates to transform it to upper triangular form. In other words, we want the banded Hessenberg structure of $\bar{H}_m$ to be retained after multiplication with Q_1, adding at most another subdiagonal. That restriction may be accomplished with an orthonormal Q_1 that is itself a Hessenberg matrix. Efficiently constructing such an orthogonal matrix Q_1 in this algorithm involves some careful algebra, which we now discuss.

Let the orthonormal matrix Q_1 be represented by an orthogonal matrix times a diagonal scaling matrix:

$$Q_1 = \hat{Q}_1 D.$$

The diagonal scaling matrix has diagonal entries $D_{ii} = \frac{1}{\|\hat{Q}_1[:,i]\|}$ so that the columns in Q_1 have unit length. The last column in $\hat{Q}_1$ is the m-dimensional vector $V_m^T s$, and constructing an orthogonal Hessenberg matrix using this last column involves only two updates on the mth iteration: the (m,m) and $(m,m-1)$ elements. We define $\hat{Q}_1$ recursively:

$$\hat{Q}_1 = \begin{pmatrix} \hat{Q}_1[1:m-1;1:m-2] & \hat{Q}_1[1:m-1;m-1] & V_{m-1}^T s \\ 0 & \frac{-(V_{m-1}^T s)^2}{v_m^T s} & v_m^T s \end{pmatrix}, \tag{25}$$

where the initial $\hat{Q}_1[1;1] = v_1^T s$. The matrix Q_1 may be represented by three vectors: the mth column of $\hat{Q}_1$ holding $V_m^T s$, the subdiagonal of $\hat{Q}_1$, and the entries in the diagonal scaling matrix. In addition, because only the two elements in the last row are new on the mth iteration, the product $\bar{H}_m Q_1$ has only two newly updated columns, which can be computed in $2(m+1)+2$ multiplications. Using a simple example with $m = 4$, we may represent this graphically

$$\bar{H}_m Q_1 = \begin{pmatrix} x & x & x & \star \\ x & x & x & \star \\ x & x & x & \star \\ & x & x & \star \\ & & x & \star \\ & & & \star \end{pmatrix} \begin{pmatrix} x & x & x & x \\ x & x & x & x \\ & x & x & x \\ & & \star & \star \end{pmatrix} = \begin{pmatrix} x & x & (x+\star) & (x+\star) \\ x & x & (x+\star) & (x+\star) \\ x & x & (x+\star) & (x+\star) \\ x & x & (x+\star) & (x+\star) \\ & x & (x+\star) & (x+\star) \\ & & \star & \star \end{pmatrix},$$

where a $\star$ represents a new number on the mth iteration and an x represents a nonzero from a previous iteration.

Given that the matrix $\bar{H}_m$ has two subdiagonals and that the matrix product $\bar{H}_m Q_1$ has three subdiagonals, the structure of (24) is

$$V_{m+2}^T M_T(x_k + d) = \begin{pmatrix} \star \\ \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} + \begin{pmatrix} \star & \star & \star & \cdots & \star & \star \\ \star & \star & \star & & \star & \star \\ \star & \star & \star & & \star & \star \\ \star & \star & \star & & \star & \star \\ & \star & \star & & \star & \star \\ & & \star & \cdots & \star & \star \\ & & & \ddots & \vdots & \vdots \\ & & & & \star & \star \end{pmatrix} \hat{y} + \begin{pmatrix} \star \\ \star \\ \\ \\ \\ \\ \\ \\ \end{pmatrix} (s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2.$$

Eliminating the subdiagonals of $\bar{H}_m Q_1$ may be accomplished with a series of Givens rotations or Householder reflections. Let $Q_2 \in \mathbb{R}^{(m+2)\times(m+2)}$ be the product of all Givens rotations or Householder reflections applied to the system, and let the variables $\tilde{r}_1$, $\tilde{a}$, and $\tilde{H}_m$ denote the following transformed vectors

$$\tilde{r}_1 \equiv Q_2 \bar{r}_1, \tag{26}$$

$$\tilde{a} \equiv Q_2 \bar{a}, \tag{27}$$

$$\tilde{H}_m \equiv Q_2 \bar{H}_m Q_1. \tag{28}$$

Using (26)–(28), we may premultiply (24) by Q_2 as a step toward the least-squares solution:

$$Q_2 V_{m+2}^T M_T(x_k + d) = \tilde{r}_1 + \tilde{H}_m \hat{y} + \tfrac{1}{2}\tilde{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2, \tag{29}$$

which has the following structure:

$$Q_2 V_{m+2}^T M_T(x_k + d) = \begin{pmatrix} \star \\ \star \\ \star \\ \vdots \\ \star \\ \star \end{pmatrix} + \begin{pmatrix} \star & \star & \star & \cdots & \star \\ & \star & \star & & \star \\ & & \star & \cdots & \star \\ & & & \ddots & \vdots \\ & & & & \star \\ & & & & \end{pmatrix} \hat{y} + \begin{pmatrix} \star \\ \star \\ \star \\ \vdots \\ \star \\ \star \\ \star \end{pmatrix} (s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2.$$

This system has $m+2$ equations in m unknowns, and the last three rows are quadratic equations in the variable $\hat{y}_m$.

By the orthonormality of Q_2 and V_{m+2}, minimizing $\left\|Q_2 V_{m+2}^T M_T(x_k + d)\right\|$ is the same as minimizing $\|M_T(x_k + d)\|$. Thus, the solution to (12) involves minimizing the last three rows of (29), which requires finding the optimum value for $\hat{y}_m$:

$$\min_{\hat{y}_m \in \mathbb{R}} \left\| \tilde{r}_1[m : m+2] + \tilde{H}_m[m : m+2, m]\hat{y}_m + \tfrac{1}{2}\tilde{a}[m : m+2](s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2 \right\|. \tag{30}$$

Problem (30), which has a closed-form solution, involves minimizing a quartic equation in a single unknown. The minimizers correspond to the critical points of the quartic equation in the objective function of (30) are thus among the real roots of a cubic equation found by differentiating the quartic equation with respect to $\hat{y}_m$. Thus, (30) can have one or two minimizers, in which case we choose the minimizer that makes $\beta = (s^T d_0 + \|s\| \hat{y}_m)$ have smaller magnitude. This choice is not necessarily the global minimizer of (30). Justification for choosing the smaller magnitude minimizer comes both from the step itself and from considering the sequence of iterates. Choosing the smaller magnitude minimizer is consistent with the approach used in direct tensor methods and results in the inexact tensor step being closer to the inexact Newton step. Also, if we consider the sequence of values $\{\beta_j\}$ for the first j iterations, then this sequence converges to a single number when choosing the smaller magnitude minimizer. If we were to choose the global minimizer, then the sequence $\{\beta_j\}$ could oscillate between two numbers, and thus the residual error would not necessarily be monotonically decreasing (β enters into the residual calculation via $\frac{1}{2}a\beta^2$). Hence, the smaller magnitude minimizer has more theoretical appeal and is used here.

As a simpler alternative for determining $\hat{y}_m$, we mention an approach that would approximately minimize $\left\|Q_2 V_{m+2}^T M_T(x_k + d)\right\|$. Instead of solving a quartic equation, we solve the single quadratic equation in the mth row of (29), choosing the root such that $\beta = (s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)$ has smaller magnitude. If the equation does not have

a root, then we choose the value of $\hat{y}_m$ that minimizes the quadratic equation. The difference between $\hat{y}_m$ found in this manner and $\hat{y}_m$ found by minimizing (30) is usually negligible once the relative residual decreases by about two orders of magnitude.

Once the minimizer $\hat{y}_m$ is determined, the remaining elements of $\hat{y}$ may be found by computing a single right-hand side using the value $\hat{y}_m$ and solving the resultant $(m-1) \times (m-1)$ linear system found by neglecting the last 3 rows of (29).

At this stage, the vector $\hat{y}$ contains the coefficients for the linear combination of basis vectors $\{v_i\}$. Thus, the approximate tensor step that solves (12) is

$$d_T = d_0 + V_m Q_1 \hat{y}.$$

The decision for stopping the Arnoldi process so that the approximate step solves the tensor model to a specified tolerance appears before the computation of the explicit step, which is at an inconvenient location. GMRES has a similar dilemma but uses an efficient approach in its least-squares solution. With GMRES, the least-squares error $\|b - Ax\|_2$ is equal to the last element of $Qe_1 \|b\|$, where Q is the product of all Givens rotations to transform the Hessenberg matrix to upper triangular form and e_1 is the unit vector $(1, 0, 0, \dots)^T$. Similarly, the last 3 rows of (29) pertain to the least-squares error of the local tensor model and may be used in stopping conditions.

There are two possible implementations for computing a stopping condition in these Krylov-based methods, and they are fundamentally similar. Both may be checked without explicitly computing the approximate step d_m after each step in the Arnoldi process.

The practical stopping condition that is used in our numerical tests is similar to GMRES in that it involves computing the norm of the remaining rows below the triangular part of $\tilde{H}_m$. That is, we neglect the contribution from the quadratic equation in row m of (29) and only calculate the norm of the last two rows when computing the least-squares error. This computation does not include any contribution from $\tilde{H}_m$ because its last two rows contain only zeroes. Thus, the practical stopping condition may be simplified to

$$\left\| \tilde{r}_1[m+1:m+2] + \tilde{a}[m+1:m+2]\tilde{\beta}_m^2 \right\| \leq \eta \left\| F(x_k) \right\|_2, \tag{31}$$

where η is the relative stopping tolerance and the norm covers only the last two rows of the vectors $\tilde{r}_3$ and $\tilde{a}$. We point out that (31) requires the computation of $\tilde{\beta}_m$ at each iteration m. Calculating $\tilde{\beta}_m$ is an $\mathcal{O}(1)$ calculation using the first-order condition for a minimizer of $\left| q(\tilde{\beta}) \right|$.

Another stopping condition, which is briefly mentioned here but covered in more detail in [1], considers how close the residual norm at the approximate step d_m comes to the minimum residual norm at the exact step d_T. In other words, the comparison is

$$\left\| M_T(x_k + d_m) \right\| - \left\| M_T(x_k + d_T) \right\| \leq \eta \left\| F(x_k) \right\|. \tag{32}$$

A practical implementation of (32) is straightforward. The residual error $\|M_T(x_k + d_m)\|$ equals the minimum value calculated in (30), and the current estimate of $\|M_T(x_k + d_T)\|$ at the mth iteration equals the value of the quadratic equation on the mth row.

Of the two conditions, (31) is a more demanding test than (32), and an implementation using (31) may require more iterations before satisfying the same relative tolerance.

Algorithm 3.2 describes the whole process for computing the approximate tensor step d_T. The algorithm is a basic implementation that progressively updates $\bar{H}_m$ to $\tilde{H}_m$ after each step in the Arnoldi process.

Algorithm 3.2: BLOCK-2 ITERATIVE TENSOR METHOD

1. Choose a relative residual tolerance $\eta \in [0, 1)$ and maximum subspace dimension m_{max}.
2. Given the local tensor model $M_T(x_k + d) = F_k + Jd + \frac{1}{2}a(s^T d)^2$, previous function value F_{k-1}, and initial guess d_0, form the block of initial vectors $R_0 = [(Jd_0 + F_k), a]$, where $a = \frac{2(F_{k-1} - F_k - Js)}{(s^T s)^2}$ and $s = x_{k-1} - x_k$.
3. Perform a partial QR-factorization on R_0, such that $R_0 = [v_1, v_2]R = VR$.
4. For $m = 1, 2, \ldots, m_{max}$ do:
   (a) Let the two columns of R, appended with m zeroes to a length $m + 2$, be labeled $\bar{r}_1$ and $\bar{r}_2$, respectively.
   (b) Form the vector Jv_m and orthogonalize it against the previous $v_1, \ldots, v_{m+1}$ vectors via the Block Arnoldi Process, Algorithm 3.1:

$$
\begin{aligned}
w &:= Jv_m \\
h_{i,m} &:= (w, v_i), \ \ i = 1, 2, \ldots, m + 1 \\
w &:= w - \sum_{i=1}^{m+1} h_{i,m} v_i \\
h_{m+2,m} &:= \|w\|_2 \\
v_{m+2} &:= w / h_{m+2,m}.
\end{aligned}
$$

   (c) Define $\bar{H}_m$ to be the $(m + 2) \times m$ upper banded Hessenberg matrix whose nonzero entries are the coefficients $h_{ij}, 1 \leq i \leq m + 3, 1 \leq j \leq m$, and define $V_m = [v_1, v_2, \ldots, v_m]$.
   (d) Let $Q_1 \in \mathbb{R}^{m \times m}$ be an orthogonal Hessenberg matrix that has $V_m^T s / \|V_m^T s\|$ in the last column and be computed via (25), and let the vector $\hat{y} \in \mathbb{R}^m$ be defined by the following transformation $y = Q_1 \hat{y}$.
   (e) Let $\bar{h}_{m-1}$ and $\bar{h}_m$ denote the two newly updated columns of the matrix-matrix product $\bar{H}_m Q_1$.

(f) Using Householder reflections, transform the $(m+2) \times m$ system $\bar{r}_1 + \bar{H}_m Q_1 \hat{y} + \frac{1}{2}\bar{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2$ into $\tilde{r}_1 + \tilde{H}_m \hat{y} + \frac{1}{2}\tilde{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2$, which involves applying all previous reflections to $\bar{h}_{m-1}$ and $\bar{h}_m$, followed by two new reflections to zero three elements in $\bar{h}_{m-1}$ and two elements in $\bar{h}_m$. Apply these reflections to the vectors $\tilde{r}_1$ and $\tilde{a}$.

(g) Find the minimizer $\hat{y}_m$ of (30) such that $\beta = (s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)$ has smaller magnitude and minimizes the least-squares error in (30).

(h) Let ρ_m represent the error estimate of solving the local tensor model. Either set ρ_m equal to the norm of the last 2 rows of (29), or set ρ_m equal to the norm of the last 3 rows of (29) minus the absolute value of the mth row.

(i) If $\rho_m \leq \eta \left\|F_k\right\|$, then proceed to step 5 to calculate the approximate step.

5. Form the approximate solution:

(a) Find the remaining $m-1$ elements of the vector $\hat{y}$ by solving the first $m-1$ rows of the linear system

$$\tilde{H}_m \hat{y} = -\tilde{r}_1 - \tilde{h}_m \hat{y}_m - \tfrac{1}{2}\tilde{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2,$$

where $\tilde{h}_m$ is the mth column of $\tilde{H}_m Q_1$.

(b) Form the approximate step $d_T = d_0 + V_m Q_1 \hat{y}$.

Just as with the Newton-GMRES algorithm, Algorithm 3.2 may be implemented matrix-free. Jacobian-vector products may be approximated by

$$J(x)v \approx \frac{F(x+\sigma v) - F(x)}{\sigma}.$$

In addition, we may apply preconditioning to accelerate convergence of the iterative methods. Consider a matrix M that approximates the current Jacobian J in some manner and is simple enough to permit inexpensive solutions to linear systems of the form $Mx = b$. Then, given M, the following left-preconditioned tensor model can be formed and solved:

$$\min_{d \in \mathcal{K}_m} \left\|M^{-1}F_k + M^{-1}Jd + \tfrac{1}{2}M^{-1}a(s^T d)^2\right\|. \tag{33}$$

The iterative tensor algorithms outlined above requires only minor modifications to incorporate left preconditioning—replace the call to Jv with $M^{-1}Jv$ in the Arnoldi process and premultiply all occurrences of F_k and a by M^{-1}. A separate subroutine that computes the action of M^{-1} times a vector is all that is needed.

Right preconditioning transforms the variable space. Given a preconditioner M, the following right-preconditioned tensor model can be formed and solved:

$$\min_{u \in \mathcal{K}_m} \left\|F_k + JM^{-1}u + \tfrac{1}{2}a(s^T M^{-1}u)^2\right\|, \tag{34}$$

where the approximate step d is found from the solution of $Md = u$. Once again, the iterative tensor algorithm requires only minor modifications—replace the call to Jv with $JM^{-1}v$ in the Arnoldi process and replace the starting vector s in R_0 with $M^{-T}s$. If the matrix M^{-1} is not explicitly stored, then $M^{-T}s$ may be difficult to compute. The algorithm may be modified to avoid this step, however.

Of the two forms, right preconditioning is mildly preferred over left preconditioning because the norm of the error $\|M_T(x_k + d)\|$, which enters directly into the stopping conditions, is unaffected with right preconditioning. In addition, right preconditioning guarantees that the inexact Newton step is a descent direction on the function, which is of paramount importance to a linesearch global strategy. With left preconditioning (with either GMRES or the Krylov-based tensor methods), the Newton step $d_N \in \mathcal{K}$ is no longer guaranteed to be a descent direction on $F(x)$, which has dire consequences in linesearch global strategies because backtracking along a step presupposes that the direction is a descent direction for eventual step acceptance.

Scaling is of particular importance when solving systems of nonlinear equations, as noted in [14], and is a subject that is closely related to preconditioning. We only mention here that variable and function scaling is possible in the Krylov-based tensor method and may be implemented in a manner similar to preconditioning.

An important remark about Algorithm 3.2 is that the Newton model is carried through the procedure, so the Newton step is calculated readily at the end of the algorithm and permits greater flexibility with the global strategy. In a manner similar to GMRES, we solve the linear least-squares problem

$$\min_{\hat{y} \in \mathbb{R}^m} \left\| \tilde{r}_1 + \tilde{H}_m \hat{y} \right\|,$$

which involves a back substitution with the upper triangular matrix $\tilde{H}_m$ and right-hand side $-\tilde{r}_1$. Then the approximate Newton step is given by

$$d_N = d_0 + V_m Q_1 \hat{y}.$$

Calculating the Newton step in addition to the tensor step adds a minimal cost. It involves a back substitution ($\frac{1}{2}m^2$ multiplications), matrix-vector product (m^2 multiplications), and a linear combination of basis vectors (nm multiplications), which is the dominant cost.

The cost of Algorithm 3.2 is very similar to the cost of GMRES. The extra work beyond GMRES involves the following:

1. Computation of one extra Jacobian-vector product to get a,
2. Partial QR-factorization of the $n \times 2$ matrix of initial vectors R_0 ($4n$ multiplications to get the second column of both Q and R),
3. Orthogonalization against one extra vector in the Gram–Schmidt process ($2nm$ multiplications),

4. Formation of an orthogonal Q_1 (nm multiplications for $V_m^T s$),

5. Computation of the new columns in the matrix-matrix product $\bar{H}_m Q_1$ (m^2 multiplications),

6. Orthogonal transformations involving Q_2 to form $\tilde{H}_m$ ($8m^2$ multiplications, if using Householder reflections),

7. Matrix-vector multiplication $Q_1 \hat{y}$ (m^2 multiplications).

Thus, when considering only the leading terms, the total cost beyond GMRES is $4n + 3nm + 6m^2$ multiplications plus one Jacobian-vector product. The bulk of the cost of GMRES is due to Gram–Schmidt orthogonalization in the Arnoldi process, $\mathcal{O}(nm^2)$, so the extra cost of the iterative Krylov-based tensor method is minor. This method compares favorably with the tensor-GMRES of Feng and Pulliam, which costs $5n + 4nm + 2m^2$ multiplications plus one extra Jacobian-vector product beyond GMRES.

3.2 Block-2+ method

It is conceivable that the block-2 method above could generate a Krylov subspace for a step that minimizes (12) but does not include any information in the direction s, thereby neglecting any contribution from the second-order term $\frac{1}{2}a(s^T d)^2$. The aim of the the block-2+ method is to explicitly include the direction s in the subspace so that the inner product $s^T d$ is fully captured in the Krylov subspace while still working with a block of dimension two. Here we only mention the discriminating feature of this method and refer to [1] or [2] for the algorithmic details.

The problem that we are solving changes to finding the step d that solves the minimization problem

$$\min_{d \in \{s\} \cup \mathcal{K}_m} \left\| F_k + Jd + \tfrac{1}{2}a(s^T d)^2 \right\|_2 . \tag{35}$$

The procedure is basically the same as the block-2 method in section 3.1 but with some extra algebra and a special technique for augmenting the standard block-Krylov subspace with the new direction s at *each* Arnoldi iteration. This approach contrasts with the usual implementation of augmented Krylov subspace methods[9] and is discussed in [1].

3.3 Block-3 method

The block-3 algorithm for solving (12) proceeds in a block-Krylov-like fashion, operating on a matrix of three initial vectors instead of the single residual r_0 of a linear

system. By choosing three vectors, we may include information on the three known vectors in the local tensor model (s, a, and F_k) and allow a transformation of the variable space and function space in a manner similar to the method of orthogonal transformations of section 2.2. To that end, we consider the block of initial vectors

$$R_0 = [s,\ (Jd_0 + F_{k-1}),\ (Jd_0 + F_k)]. \tag{36}$$

The rationale for choosing these specific vectors is as follows. The vector s is listed first in order to isolate the inner product $s^T d$ (via $\|s\| v_1^T d$) and later create a single quadratic equation in a single unknown. The second vector is the residual involving the previous function value F_{k-1} and is needed for computing the tensor term a, (6). The third vector is the residual of the Newton equations, and it may be placed as the second or third column in R_0. Collectively, these three vectors are chosen specifically to compute the tensor term a later in the algorithm in addition to fully characterizing the local tensor model (i.e., the three known vectors F_k, a, s) with this initial subspace.

The block-3 algorithm is procedurally different from the block-2 algorithm because it uses orthogonal transformations and permutation matrices to switch rows and columns to isolate a quadratic equation in the mth row. After the block-Arnoldi process adds a basis vector and an extra column to $\bar{H}_m$, we perform a series of plane rotations to put the matrix $\bar{H}_m$ in upper triangular form. In its current ordering, the quadratic equation would not be isolated to a single variable in the first row and should be switched to the mth row. So we permute the first row and column with the mth row and column to facilitate an easier solution. After all of the orthogonal transformations and row/column permutations, the structure of the simplified problem is

$$QP_L V_{m+3}^T M_T(x_k + d) = \begin{pmatrix} \star \\ \star \\ \star \\ \vdots \\ \star \\ \star \\ \star \\ \star \end{pmatrix} + \begin{pmatrix} \star & \star & \star & \cdots & \star \\ & \star & \star & & \star \\ & & \star & \cdots & \star \\ & & & \ddots & \vdots \\ & & & & \star \end{pmatrix} \hat{y} + \begin{pmatrix} \star \\ \star \\ \star \\ \vdots \\ \star \\ \star \\ \star \\ \star \end{pmatrix} (s^T d_0 + \|s\| \hat{y}_m)^2.$$

Otherwise, the block-3 algorithm is similar to the block-2 algorithm above.

3.4 Tensor-Krylov methods

With the introduction of the Krylov-based iterative methods for solving the local tensor model in sections 3.1–3.3, we return to solving the general nonlinear equations problem (1). The following algorithm outlines the tensor-Krylov algorithm, which at every outer iteration calls a Krylov-based iterative method for solving the local tensor model.

Algorithm 3.3: TENSOR-KRYLOV METHOD

1. Given the nonlinear equations problem $F(x)$, choose a starting point x_0 and set the maximum iteration counter k_{max}.
2. For $k = 0, 1, 2, \ldots, k_{max}$, do:
   (a) Choose a forcing term tolerance $\eta_k \in [0, 1)$.
   (b) If $k = 0$, then calculate the Newton-GMRES step d_N according to the relative tolerance η_k and proceed to step 2e.
   (c) Form the local tensor model $M_T(x_k + d) = F_k + Jd + \frac{1}{2}a(s^T d)^2$, where $F_k = F(x_k)$, $F_{k-1} = F(x_{k-1})$, $J = F'(x_k)$, $s = x_{k-1} - x_k$, and $a = \frac{2(F_{k-1} - F_k - Js)}{(s^T s)^2}$.
   (d) Compute the inexact tensor step d_T according to the relative tolerance η_k by approximately solving the local tensor model according to the methods of sections 3.1, 3.2, or 3.3.
   (e) Set $x_{k+1} = x_k + \lambda d$, where d and λ are chosen according to a linesearch strategy that uses the directions d_T and/or d_N.
   (f) If x_{k+1} is an acceptable approximation to a root of $F(x)$, then stop and signal a success.

When referring to Algorithm 3.3 that uses a specific Krylov-based local solver in step 2d as defined in sections 3.1, 3.2, or 3.3 (i.e., the block-2, block-2+, or block-3 methods), we will abbreviate the method as TK2, TK2+, and TK3, respectively.

3.5 Global strategy and step selection

Algorithm 3.3 needs a robust strategy for global convergence if neither the full tensor step nor Newton step is satisfactory in step 2e. While step 2e uses a linesearch strategy, a trust region strategy is still viable, albeit less straightforward. Here we discuss details regarding a linesearch implementation in the tensor-Krylov method.

The standard tensor linesearch of [26] and the TENSOLVE linesearch of [5, 6] are straightforward applications of backtracking along the tensor step, if it is a descent direction, or otherwise along the Newton direction. The curvilinear linesearch implementation of [3] requires a little adaptation. Because the curvilinear linesearch for tensor methods has posted encouraging results and has a nice theoretical basis, we will focus primarily on this linesearch implementation in the tensor-Krylov algorithm.

The curvilinear step $d_T(\lambda)$ is the solution of the modified tensor model $\lambda F + Jd + \frac{1}{2}a(s^T d)^2$, where λ is the linesearch parameter. Thus, in the tensor-Krylov algorithm, the local tensor model is likewise changed and recomputed. Fortunately, the scalar λ is carried through the process in a straightforward manner, irrespective of method,

as will now be discussed. For this discussion, we focus on the block-2 method in Algorithm 3.2, but the procedure applies to the block-2+ and block-3 methods in the obvious way. The only trick to this implementation involves the scaling of the initial guess d_0; all other aspects are intuitive.

The derivation involves changing the block of initial vectors in (14) to include λ in the Newton residual,

$$R_0 = [\lambda(Jd_0 + F_k),\ a]. \tag{37}$$

The scalar λ follows through the steps of Algorithm 3.2 and changes (29) to

$$Q_2 V_{m+2}^T M_T(x_k + d) = \lambda \tilde{r}_1 + \tilde{H}_m \hat{y} + \tfrac{1}{2}\tilde{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2, \tag{38}$$

which only differs by λ multiplying $\tilde{r}_1$. Up to this point in the algorithm, the presence of λ does not require any new computations. To be more precise, the basis vectors in V and the matrix $\tilde{H}_m$ are unchanged. The presence of λ in (38) does change the calculation of the vector $\hat{y}_m$, and the corresponding change in (30) is

$$\min_{\hat{y}_m \in \mathbb{R}} \left\| \lambda \tilde{r}_1 + \tilde{H}_m \hat{y} + \tfrac{1}{2}\tilde{a}(s^T d_0 + \left\|V_m^T s\right\| \hat{y}_m)^2 \right\|.$$

The remaining elements of $\hat{y}$ are found by solving the triangular system with a right-hand side modified by λ and the newly computed $\hat{y}_m$. Finally, the change in (37) corresponds to scaling both F_k and d_0 by λ, so the curvilinear step changes to

$$d_T(\lambda) = \lambda d_0 + V_m Q_1 \hat{y}, \tag{39}$$

where $\hat{y}$ is also a function of λ, as noted above.

We reiterate that the scalar λ may multiply $\tilde{r}_1$ after all orthogonal transformations, so the initial work in generating a Krylov basis and performing the subsequent orthogonal transformations to calculate d_T is not repeated for computing $d_T(\lambda)$. That is, the matrix of basis vectors V_m used in (39) contains the same vectors from the original computation of d_T; only the vector $\hat{y}$ depends on λ.

The additional cost of the curvilinear linesearch per trial is an extra backsolve per λ-value (an extra $\frac{1}{2}m^2$ multiplications) plus a linear combination of basis vectors (nm multiplications), which is the dominant cost. However, if using right preconditioning, one application of the preconditioner must be used, which increases the cost further. While these costs are more than in the other linesearches, they are probably still less than the cost of evaluating $F(x)$ and certainly less than the cost of evaluating $J(x)$ or the total cost of generating the Krylov subspace for the original computation of d_T. Alternative implementations that use other simplifications or approximations are discussed in [1].

It should be noted that other large-scale tensor methods, such as the tensor-GMRES method of Feng and Pulliam [16], could employ the curvilinear linesearch even though these other methods have subtle differences in calculating an inexact

tensor step. This is because the curvilinear step is calculated from a simple scalar multiplication of the function value in the local tensor model and may be carried through the algebra of the step calculation to arrive at a parametric form of the curvilinear step.

4 Computational results and discussion

This section describes implementations of the three tensor-Krylov methods and presents some numerical results and comparisons on several challenging problems. We set up the numerical experiments to closely correspond to those in [28], and the tests are aimed at comparing the tensor-Krylov methods with both Newton-GMRES and the tensor-GMRES method of Feng and Pulliam. More comprehensive numerical tests may be found in [1], and results on several ill-conditioned problems are included in [4].

We implemented the algorithms in a software package called NOX [17], which is a C++ object-oriented nonlinear solver package being developed at Sandia National Laboratories. For objective comparisons, we implemented all of the methods, including Newton-GMRES and tensor-GMRES, and used the same Arnoldi process (modified Gram–Schmidt) as in our tensor-Krylov methods. That choice granted us more control over the algorithm and assured us of a controlled experiment. Thus, the results in this section do not reflect the most efficient implementations that are available.

We solved the problems using the conditions and parameters in [28] as a guide. For a successful termination, we required $\|F(x_k)\| \leq \varepsilon_F \|F(x_0)\|$, where $\varepsilon_F = 10^{-2}$. In addition, we also required the more stringent weighted step length stopping condition,

$$\frac{1}{\sqrt{n}} \|W d_k\| < 1,$$

where n is the total number of unknowns, d_k is the full Newton or tensor step, and W is a diagonal scaling matrix with entries

$$W_{ii} = \frac{1}{\varepsilon_r |x_{k_i}| + \varepsilon_a},$$

in which x_{k_i} is the ith element of the current solution x_k, ε_r is a relative tolerance (10^{-3}), and ε_a is an absolute tolerance (10^{-8}). The step length criterion is necessary to resolve finer details of the fluid flow and transport by requiring that each ith element of the Newton or tensor step be small relative to its current value x_{k_i}. Also, if the test problem required more than 200 nonlinear (outer) iterations or if there was a linesearch failure (i.e., $f(x_c + \lambda d) \leq f(x_c) + \alpha\lambda\nabla f(x_c)^T d$, where $f(x) \equiv \frac{1}{2}\|F(x)\|_2$ and $\alpha = 10^{-4}$, could not be satisfied with $\lambda > 10^{-12}$ in at most 40 backtracks), then we declared a failure for the run.

We allowed the local solver (i.e., GMRES or its tensor-Krylov equivalent) to iterate a total of 250 times to satisfy a relative tolerance of $\eta = 10^{-4}$. If the local solver did not satisfy the desired tolerance or finish in the allotted count, we used the step computed so far and tested for step acceptance with our global strategies. In nearly all of our numerical experiments, the local method successfully computed a step that satisfied the relative tolerance, so restarting was not an issue.

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 [22], and we performed no variable or function scaling in the problems. The initial approximation was the zero vector for all cases.

There are multiple choices for the global strategy, and we used the standard tensor linesearch for all tensor methods to facilitate an unbiased comparison of nonlinear algorithms. Due to the promising results of the curvilinear linesearch on small-scale problems in [3], we also used the curvilinear linesearch with the TK2 and TK3 methods to study the benefit of this linesearch on large-scale implementations. We used the λ-halving procedure (dividing λ by two at each inner iteration) for selecting the linesearch parameter at each trial step. Quadratic backtracking was an option but generally required more iterations and function evaluations than λ-halving in preliminary tests on these problems, so it was not used.

All tests were performed on a dual processor desktop computer (Intel 686 architecture) at Sandia National Laboratories. The computer had 1GB of RAM, which was more than sufficient for our problems, which used no more than 17% of the memory. However, the computer was not dedicated to these tests, so the timing statistics we provide are very approximate and could be off by 10% or more relative to each other.

All of the tables in this section have standard column labels, which we now describe. The Reynolds number or Rayleigh number is listed in the first column with the appropriate label. The second column lists the nonlinear method according to the abbreviations discussed above (in addition to abbreviating the Feng-Pulliam tensor-GMRES method as TG and the Newton-GMRES method as NG). The third column indicates the particular linesearch procedure used. The tables include comparisons with both the standard tensor linesearch and the curvilinear linesearch. The column entitled "Fail" indicates any failures, i.e., if there was a linesearch failure (LSF) or if the maximum iteration count was reached (MAX). The "Itn" and "Feval" columns record the number of nonlinear (outer) iterations and function evaluations, respectively. The next two columns list the number of calls to the linesearch procedure and the total number of backtrack steps that were performed. The "Arnoldi" column records the total number of Arnoldi (inner) iterations, e.g., GMRES steps. Because an iteration of the Arnoldi process includes a Jacobian-vector product, this is the time-limiting step and closely correlates with time. The last column lists an approximate time to solution, but as mentioned above, these times have error of 10% or more.

The columns of greatest interest in this study are the nonlinear iterations total, the number of Arnoldi iterations, and the approximate time.

4.1 Test results on fluid flow benchmark problems

The test problems are three CFD benchmark problems described in [28] that are used for verification of fluid flow codes and solution algorithms: the thermal convection problem, the lid driven cavity problem, and the backward-facing step problem.

The problems are set up using a particular spatial discretization of the governing steady-state transport equations for momentum and heat transfer in flowing fluids. These governing PDEs are given below. The unknown quantities in these equations are the fluid velocity vector ($\mathbf{u}$), the hydrodynamic pressure (P), and the temperature (T).

$$\text{Conservation of mass:} \quad \nabla \cdot \mathbf{u} = 0 \tag{40}$$

$$\text{Momentum transport:} \quad \rho\, \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \mathbf{T} - \rho \mathbf{g} = 0 \tag{41}$$

$$\text{Energy transport:} \quad \rho C_p \mathbf{u} \cdot \nabla T + \nabla \cdot \mathbf{q} = 0 \tag{42}$$

In these equations, $\mathbf{g}$ is the gravity vector, and ρ and C_p are the density and specific heat at constant pressure of the bulk fluid, respectively. The constitutive equations for the stress tensor $\mathbf{T}$ and heat flux $\mathbf{q}$ are

$$\begin{aligned} \mathbf{T} &= -P\mathbf{I} + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T), \\ \mathbf{q} &= -\kappa \nabla T, \end{aligned}$$

where μ is the dynamic viscosity and κ is the thermal conductivity of the fluid.

The particular spatial discretization of (40)–(42) that we use is from a finite element reacting flow code called MPSalsa [25] developed at Sandia National Laboratories. MPSalsa generates an algebraic system of equations by a pressure-stabilized Petrov–Galerkin finite element formulation of the low Mach number Navier–Stokes equations with heat transport. This scheme uses equal-order interpolation of velocity and pressure, and we enabled the option for streamline upwinding to limit oscillations due to high grid Reynolds numbers. Since the publication of [28], the pressure-stabilized streamline upwinding Petrov–Galerkin formulation in MPSalsa has been changed to a Galerkin least squares–type method [27]. This stabilization method is slightly less dissipative, and the nonlinear convergence behavior for difficult problems (e.g., the lid driven cavity problem) is less robust at higher Reynolds numbers. Consequently, this change precludes direct comparisons with results in [28].

To complete a problem's specification, boundary conditions are imposed on the governing PDEs, which we discuss in the subsections that follow. The three problems differ only in their boundary conditions and in whether they use (40)–(42) or only (40)–(41). The next three subsections describe the test problems and test results.

4.1.1 Thermal convection problem

This problem consists of the thermal convection (or buoyancy driven) flow of a fluid in a differentially heated square cavity in the presence of gravity. It requires the solution of (40)–(42) on the unit square with the following Dirichlet and Neumann boundary conditions:

$$\begin{aligned} T = T_{cold},\ \mathbf{u} = 0 \quad &\text{at} \quad x = 0, \\ T = T_{hot},\ \mathbf{u} = 0 \quad &\text{at} \quad x = 1, \\ \frac{\partial T}{\partial y} = 0,\ \mathbf{u} = 0 \quad &\text{at} \quad y = 0, 1. \end{aligned}$$

Once the governing equations and boundary conditions are nondimensionalized, two parameters appear: the Prandtl number (Pr) and the Rayleigh number (Ra). In our experiments, we fixed Pr = 1 and increased the Rayleigh number up to 10^7, which increases the nonlinear effects of the convection terms and makes the solution more difficult to obtain. The range in [28] is Ra = 10^3 to 10^6, but we shifted the range to include 10^7 to explore the effectiveness of tensor methods on more difficult problems. We used a 100 × 100 equally spaced mesh, which has 40,804 unknowns. On this size mesh, it is unclear whether the choice of Ra = 10^7 admits a physically accurate and/or stable solution. However, we are interested only in the relative performance of the numerical methods on this problem, which remain valid comparisons.

The results of the thermal convection problem in Table 1 provide the most convincing evidence for the benefit of tensor-Krylov methods. It is also the only test problem of the three that includes heat transport in its formulation, which may or may not be a factor. The results show that Newton-GMRES performs a little better than the other methods on the two easiest problem difficulties, which has been typical. Yet as the problem grows more difficult, the tensor methods outperform Newton-GMRES. In other tests that we have performed on this problem, the trend is even more evident, showing a clear degradation in performance by Newton-GMRES over the transition from easy to difficult problems. The tensor-Krylov methods are much less affected by the transition, especially when using the curvilinear linesearch.

Tensor-GMRES also does well on this problem at the mid-range difficulties and is more efficient than the tensor-Krylov methods using the same standard linesearch, especially in terms of function evaluations. However, tensor-GMRES and Newton-GMRES are unable to solve the most difficult problem at Ra = 10^7. All tensor-Krylov methods except TK3 with the standard linesearch are able to solve the hardest problem. Among the tensor-Krylov methods, TK2 is slightly more efficient than TK2+, and both are more efficient than TK3, which requires more Arnoldi iterations at all difficulties.

The tensor-Krylov methods with the curvilinear linesearch are able to solve more difficult problems than Newton-GMRES or tensor-GMRES. Moreover, there is a clear difference between the standard linesearch and the curvilinear linesearch among the

Table 1. Results of the thermal convection problem over a range of Rayleigh numbers. All methods use a constant forcing term of $\eta_k = 10^{-4}$ and right preconditioning.

Ra	Method	Linesearch	Fail	Itn	Feval	LS's	Btrk	Arnoldi	Time
1e+04	NG	Standard	.	6	8	1	1	577	3.0e+02
1e+05	NG	Standard	.	10	32	6	21	1031	5.3e+02
1e+06	NG	Standard	.	34	215	30	180	3825	2.1e+03
1e+07	NG	Standard	Max	200	2065	200	1864	23955	1.5e+04
1e+04	TG	Standard	.	7	9	1	1	677	3.4e+02
1e+05	TG	Standard	.	11	34	7	22	1169	6.2e+02
1e+06	TG	Standard	.	36	218	32	181	4039	2.6e+03
1e+07	TG	Standard	Max	200	1793	200	1592	23852	1.5e+04
1e+04	TK2	Standard	.	8	10	1	1	990	5.2e+02
1e+05	TK2	Standard	.	15	60	9	44	2088	1.2e+03
1e+06	TK2	Standard	.	37	294	33	256	5838	3.9e+03
1e+07	TK2	Standard	.	138	1606	134	1467	23202	1.7e+04
1e+04	TK2+	Standard	.	8	10	1	1	992	5.2e+02
1e+05	TK2+	Standard	.	15	60	9	44	2078	1.2e+03
1e+06	TK2+	Standard	.	37	293	32	255	5818	4.1e+03
1e+07	TK2+	Standard	.	159	1866	155	1706	26869	1.9e+04
1e+04	TK3	Standard	.	8	11	2	2	1034	5.6e+02
1e+05	TK3	Standard	.	19	81	13	61	2857	1.8e+03
1e+06	TK3	Standard	.	42	355	37	312	7631	5.5e+03
1e+07	TK3	Standard	Max	200	2683	200	2482	40963	3.1e+04
1e+04	TK2	Curvilinear	.	8	10	1	1	990	5.2e+02
1e+05	TK2	Curvilinear	.	11	35	6	23	1537	9.4e+02
1e+06	TK2	Curvilinear	.	21	118	17	96	3195	2.1e+03
1e+07	TK2	Curvilinear	.	55	439	51	383	9160	6.6e+03
1e+04	TK3	Curvilinear	.	8	11	2	2	1020	5.5e+02
1e+05	TK3	Curvilinear	.	12	36	6	23	1749	1.1e+03
1e+06	TK3	Curvilinear	.	22	120	18	97	3878	2.9e+03
1e+07	TK3	Curvilinear	.	57	465	53	407	11385	8.9e+03

tensor-Krylov methods. As the Rayleigh number increases and the problem becomes more difficult to solve, the performance margin increasingly favors the curvilinear linesearch. For instance, on the hardest problem, TK2 with the standard tensor linesearch finishes in about 2.6 times the time of the corresponding curvilinear linesearch implementation. TK3 with the standard tensor linesearch could not solve the problem in 200 nonlinear iterations.

4.1.2 Backward-facing step problem

This problem consists of a rectangular channel with a 1×30 aspect ratio in which a reentrant backward-facing step (i.e., a sudden expansion in the channel width) is simulated by injecting fluid with a fully developed parabolic velocity profile in the upper half of the inlet boundary and imposing a zero velocity on the lower half. The channel geometry and flowing fluid produce recirculation zones beneath the entering flow on the lower wall and, for sufficiently high Re, farther downstream on the upper wall. This problem requires the solution of (40)–(41) on the unit square with the following Dirichlet boundary conditions:

$$\begin{aligned} \mathbf{u} = 24y(\tfrac{1}{2} - y)U_0\hat{x} \quad &\text{at} \quad x = 0,\ 0 \le y \le \tfrac{1}{2}, \\ \mathbf{u} = 0 \quad &\text{at} \quad x = 0,\ -\tfrac{1}{2} \le y < 0, \\ \mathbf{u} = 0 \quad &\text{at} \quad y = -\tfrac{1}{2}, \tfrac{1}{2}, \\ \mathbf{T}_{xx} = \mathbf{T}_{xy} = 0 \quad &\text{at} \quad x = 30, \end{aligned}$$

where $\hat{x}$ is the unit vector in the x-direction. Once the governing equations and boundary conditions are nondimensionalized, the Reynolds number (Re) appears, which is a measure of inertial forces to viscous forces. In our experiments, we increased the Reynolds number up to 800, which increases the nonlinear inertial terms in the momentum equation and makes the solution more difficult to obtain. Beyond Re = 800, it is not clear that the problem is stable and admits a physical solution. All solutions for this problem were computed on a 20×400 unequally spaced mesh, which has 25,263 unknowns.

The plots in Figure 1 show that all of the methods require about 10–12 iterations, on average, to solve, with Newton-GMRES requiring considerably more iterations in some cases. These results are somewhat typical: the Newton-based method required gradually more work to solve the problems as they increased in difficulty, whereas the tensor methods are less affected. Here, however, Newton's method is more erratic, having slight difficulty at Re = 300 and 400, improvements at Re = 500 and 600, and then more difficulty on the three hardest problems. On the other hand, all of the tensor-Krylov methods share almost the exact same level of performance in terms of nonlinear iterations. Tensor-GMRES requires slightly more nonlinear iterations than the tensor-Krylov methods, but its local solve with with GMRES is more efficient. Thus, for this problem, tensor-GMRES is more efficient than the tensor-Krylov methods by a small margin, and TK2 and TK2+ are more efficient than TK3 by about the

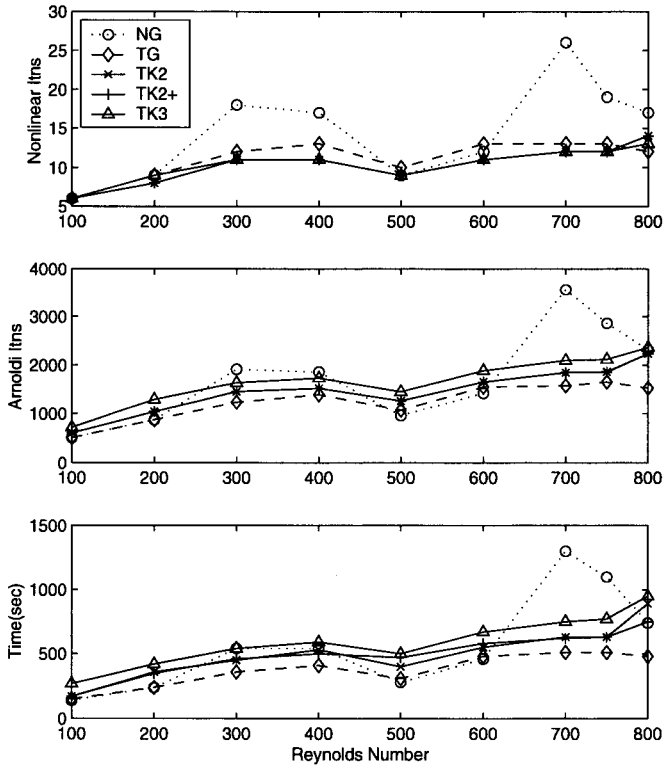

Figure 1. BFS problem results for the following methods: TK3 ($\triangle$, solid line), TK2 ($\times$, solid line), TK2+ ($+$, solid line), TG ($\diamond$, dashed line), and NG ($\circ$, dotted line).

same amount. Incidentally, there appears to be no distinct difference between TK2 and TK2+.

4.1.3 Lid driven cavity problem

The lid driven cavity problem consists of the confined flow of a fluid in a square cavity driven by a moving upper boundary. It requires the solution of (40)–(41) on the unit square with the following Dirichlet boundary conditions:

$$\begin{aligned} \mathbf{u} &= 0 \quad \text{at} \quad x = 0, 1, \\ \mathbf{u} &= 0 \quad \text{at} \quad y = 0, \\ \mathbf{u} &= U_0 \hat{x} \quad \text{at} \quad y = 1. \end{aligned}$$

The Reynolds number (Re) appears in the nondimensionalized problem and is a measure of inertial forces to viscous forces. In our experiments, we increased the Reynolds number, which increases the nonlinear inertial terms in the momentum

Table 2. Results of the lid driven cavity problem over a range of Reynolds numbers. All methods use a constant forcing term of $\eta_k = 10^{-4}$ and right preconditioning.

Re	Method	Linesearch	Fail	Itn	Feval	LS's	Btrk	Arnoldi	Time
500	NG	Standard	.	9	10	0	0	635	2.0e+02
1000	NG	Standard	.	11	14	2	2	838	2.8e+02
1500	NG	Standard	LSF	29	525	26	495	2887	1.6e+03
2000	NG	Standard	.	21	59	14	37	1770	6.2e+02
500	TG	Standard	.	10	12	1	1	709	2.7e+02
1000	TG	Standard	.	15	24	7	8	1207	4.1e+02
1500	TG	Standard	Max	200	2723	198	2522	22248	1.1e+04
2000	TG	Standard	Max	200	2162	200	1961	23662	1.1e+04
500	TK2	Standard	.	11	15	3	3	1092	3.9e+02
1000	TK2	Standard	.	20	64	14	43	2181	8.7e+02
1500	TK2	Standard	Max	200	2462	199	2261	24756	1.2e+04
2000	TK2	Standard	Max	200	2561	200	2360	30244	1.5e+04
500	TK2+	Standard	.	11	15	3	3	1082	3.8e+02
1000	TK2+	Standard	.	25	97	19	71	2778	1.2e+03
1500	TK2+	Standard	.	120	1019	114	898	15556	7.1e+03
2000	TK2+	Standard	Max	200	2644	200	2443	31687	1.5e+04
500	TK3	Standard	.	12	15	1	2	1302	5.1e+02
1000	TK3	Standard	.	15	35	6	19	1716	7.8e+02
1500	TK3	Standard	.	46	233	39	186	5685	2.4e+03
2000	TK3	Standard	Max	200	2152	200	1951	30006	1.4e+04
500	TK2	Curvilinear	.	11	15	3	3	1098	4.0e+02
1000	TK2	Curvilinear	.	14	19	3	4	1409	5.1e+02
1500	TK2	Curvilinear	Max	200	2213	199	2012	26243	1.3e+04
2000	TK2	Curvilinear	Max	200	2485	200	2284	34339	1.9e+04
500	TK3	Curvilinear	.	11	13	1	1	1183	4.6e+02
1000	TK3	Curvilinear	.	15	28	6	12	1768	7.0e+02
1500	TK3	Curvilinear	.	19	41	12	21	2227	8.9e+02
2000	TK3	Curvilinear	Max	200	2226	200	2025	28720	1.5e+04

equation and makes the solution more difficult to obtain. We used a 100×100 equally spaced mesh, which has 30,603 unknowns.

The results of the lid driven cavity in Table 2 paint a different picture. For Reynolds numbers greater than 2×10^3, the problem was too difficult to solve for all methods under these testing conditions. Newton-GMRES is the most efficient on the three problems that it does solve, having one linesearch failure at Re = 1500. The tensor methods require more nonlinear iterations than Newton-GMRES. We have encountered this phenomenon before on various formulations of the lid driven cavity problem, which suggests that the Newton step is typically a better direction than the tensor direction, possibly because the secant approximation in the tensor method is a poor approximation to any local second-order information.

On the two problems that tensor-GMRES solves, it is also faster than the tensor-

Krylov methods. This analysis is in line with the previous observation about tensor methods. Because tensor-GMRES may be viewed as a hybrid method between Newton-GMRES and the tensor-Krylov methods, one could expect its performance in terms of nonlinear iterations to fall somewhere in the range between the two. In terms of robustness, however, tensor-GMRES could not solve the two tougher problems, whereas TK3 and TK2+ could solve the case Re = 1500.

Among the tensor-Krylov methods, TK3 is the most efficient and most robust. On this test problem, there is a significant difference between TK2 and TK2+. At Re = 1000, TK2 is faster than TK2+, but TK2+ was able to solve the problem at Re = 1500.

The benefit of the curvilinear linesearch is also evident on the lid driven cavity problem. Comparing TK2 and TK3, one can see that the curvilinear linesearch requires less work than the corresponding run with the standard tensor linesearch.

4.2 Summary of results considering restarts

This section summarizes some of the conclusions from more extensive testing performed in [1]. Numerical tests using restarts clearly indicate that the tensor-Krylov methods needed a larger subspace over which to solve the local tensor model. TK2 and TK2+ appeared to be better at restarting than TK3. We believe this is because in a single restart cycle the block-2 methods have a polynomial expansion of the block-Krylov subspace that contains higher orders of the Jacobian matrix. That is, after m iterations, the block-2 methods have terms in the Krylov subspace up to $J^{\frac{m}{2}-1}$, whereas the block-3 method has terms up to $J^{\frac{m}{3}-1}$. For comparison, GMRES includes terms up to J^{m-1}.

In addition, the tensor-Krylov methods tend to stall more frequently than GMRES(m). That is, restarting the method does not always appreciably improve the step after another m iterations and so more restarts are needed to refine the step. This behavior may be attributed to the block-Krylov style of the Arnoldi process, which retains a constant vector in R_0 at each restart (i.e., s and/or a, depending upon the algorithm), keeping part of the subspace unchanged. Restarts rely on a new and different subspace to make progress.

One alternative for addressing these issues is incomplete orthogonalization [7, 21], which would require modifications to the algorithms but may be a better strategy for coping with difficult problems. We may investigate this idea further in future research.

Tests in [1] also provide some evidence that tensor-GMRES loses effectiveness when restarting because of the smaller subspace. A smaller subspace provides less information in the projection of the tensor term (i.e., $Pa(s^Td)^2$) as well as including a smaller basis for solving the minimization problem in (11), where $Pa(s^Td)^2$ acts

as another right-hand side but the Krylov subspace generated by GMRES starts with $F_k + J_k d_0$. Overall, tensor-GMRES and the tensor-Krylov methods are fairly similar—sometimes tensor-GMRES is better due to its efficient use of GMRES(m) and sometimes tensor-Krylov methods are better due to a more accurate tensor step.

5 Summary and conclusions

The main objective of this research was to combine approaches based on direct tensor methods and Krylov subspace methods into an effective large-scale nonlinear equations solver. We developed three Krylov-based methods for iteratively solving the local tensor model, and we incorporated these three local solvers into an inexact nonlinear solver framework for different versions of a "tensor-Krylov" method, which we denoted TK2, TK2+, and TK3. The new tensor-Krylov methods are especially effective at solving large-scale problems that possess Jacobians at the solution that are highly ill-conditioned or singular. Algorithms based on Newton's method exhibit very slow convergence on such problems.

The new methods proposed in this paper solve the local tensor model in a novel fashion. Their costs per iteration are similar to GMRES, requiring only one Jacobian-vector product at each iteration and $O(nm)$ additional arithmetic operations beyond GMRES per solve. Relative to previous iterative tensor methods, they are the only methods that produce an approximate tensor step that solves the local tensor model to within a specified accuracy. In addition, these methods can compute an exact solution to the tensor model in at most n iterations (in exact arithmetic). The new tensor-Krylov methods can also utilize much of the technology developed for Newton-Krylov methods, including preconditioning and restarting.

Our numerical results suggest that the new tensor-Krylov methods, as well as the tensor-GMRES method, clearly have some big advantages over Newton-GMRES in many cases, especially as the problem becomes more difficult or more ill-conditioned. In addition, the tensor-Krylov methods have some potential advantages over tensor-GMRES that make them likely to be beneficial on some important problems.

We see many different research questions at this point that we would like to explore. We mention two future extensions here. First, the current tensor-Krylov and tensor-GMRES implementations need to manipulate data structures that are inaccessible in many linear solver packages, so we would like to simplify these methods and investigate better ways to incorporate stand-alone linear algebra packages. Second, changing the current scalar implementation of the block-Arnoldi method to a true block implementation (i.e., simultaneously multiplying a block of vectors by the Jacobian) may improve memory efficiency and make the tensor-Krylov methods even more economical and attractive. These two changes would make the methods more

accessible and possibly more efficient, respectively, than their current implementations.

References

[1] Brett W. Bader. *Tensor-Krylov methods for solving large-scale systems of nonlinear equations.* PhD thesis, University of Colorado, Boulder, Department of Computer Science, 2003.

[2] Brett W. Bader and Allison H. Baker. Augmenting GMRES at every iteration. In preparation.

[3] Brett W. Bader and Robert B. Schnabel. Curvilinear linesearch for tensor methods. *SIAM J. Sci. Comput.*, 25(2):604–622, 2003.

[4] Brett W. Bader and Robert B. Schnabel. On the performance of tensor methods for ill-conditioned problems. In preparation.

[5] Ali Bouaricha. *Solving large sparse systems of nonlinear equations and nonlinear least squares problems using tensor methods on sequential and parallel computers.* PhD thesis, University of Colorado, Boulder, Department of Computer Science, 1992.

[6] Ali Bouaricha and Robert B. Schnabel. Algorithm 768: TENSOLVE: A software package for solving systems of nonlinear equations and nonlinear least-squares problems using tensor methods. *ACM Transactions of Mathematical Software*, 23:174–195, 1997.

[7] Peter N. Brown and Alan C. Hindmarsh. Reduced storage methods in stiff ODE systems. *J. Appl. Math. Comput.*, 31:40–91, 1989.

[8] Peter N. Brown and Yousef Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Statist. Comput.*, 11:450–481, 1990.

[9] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numer. Linear Algebra Appl.*, 4(1):43–66, 1997.

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

[11] D. W. Decker and C. T. Kelley. Newton's method at singular points I. *SIAM J. Numer. Anal.*, 17:66–70, 1980.

[12] D. W. Decker and C. T. Kelley. Newton's method at singular points II. *SIAM J. Numer. Anal.*, 17:465–471, 1980.

[13] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM J. Numer. Anal.*, 19:400–408, 1982.

[14] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[15] Dan Feng, Paul D. Frank, and Robert B. Schnabel. Local convergence analysis of tensor methods for nonlinear equations. *Math. Programming*, 62:427–459, 1993.

[16] Dan Feng and Thomas H. Pulliam. Tensor-GMRES method for large systems of nonlinear equations. *SIAM J. Optim.*, 7:757–779, 1997.

[17] Tamara Kolda and Roger Pawlowski. NOX: An object-oriented nonlinear solver package. http://software.sandia.gov/nox/.

[18] Jorge Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[19] G. W. Reddien. On Newton's method for singular problems. *SIAM J. Numer. Anal.*, 15:993–996, 1978.

[20] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Mathematics of Computations*, 33:680–687, 1979.

[21] Yousef Saad. Krylov subspace methods for solving unsymmetric linear systems. *Math. Comp.*, 37:105–126, 1981.

[22] Yousef Saad. ILUT: A dual threshold incomplete ILU preconditioner. *Numer. Linear Algebra Appl.*, 1:387–402, 1994.

[23] Yousef Saad. Analysis of augmented Krylov subspace methods. *SIAM J. Matrix Anal. Appl.*, 18:435–449, 1997.

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

[25] 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.

[26] Robert B. Schnabel and Paul D. Frank. Tensor methods for nonlinear equations. *SIAM J. Numer. Anal.*, 21:815–843, 1984.

[27] J. N. Shadid. A fully-coupled Newton-Krylov solution method for parallel unstructured finite element fluid flow, heat and mass transfer simulations. *Int. J. Comput. Fluid Dyn.*, 12:199–211, 1999.

[28] John N. Shadid, Ray S. Tuminaro, and Homer F. Walker. An inexact Newton method for fully coupled solution of the Navier–Stokes equations with heat and mass transport. *J. Comput. Phys.*, 137:155–185, 1997.

DISTRIBUTION:

5	MS 0316	Brett W. Bader, 9233
2	MS 9018	Central Technical Files, 8945-1
1	MS 0899	Technical Library, 9616