Tensor-Krylov Methods for Large Nonlinear Equations[*]

Ali Bouaricha[†]
MCS Division, Argonne National Laboratory, Argonne, IL 60439, USA

Abstract. In this paper, we describe tensor methods for large systems of nonlinear equations based on Krylov subspace techniques for approximately solving the linear systems that are required in each tensor iteration. We refer to a method in this class as a tensor-Krylov algorithm. We describe comparative testing for a tensor-Krylov implementation versus an analogous implementation based on a Newton-Krylov method. The test results show that tensor-Krylov methods are much more efficient and robust than Newton-Krylov methods on hard nonlinear equations problems.

Key words. tensor-Krylov methods, Newton-Krylov methods, nonlinear equations, sparse problems, ill-conditioned and singular problems

[*]Part of this work was performed while the author was research associate at CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique).

[†]Research supported in part by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

1. Introduction

In this paper we introduce tensor-Krylov methods for solving the sparse nonlinear equations problem

$$\text{given } F : \Re^n \rightarrow \Re^n, \text{ find } x_* \in \Re^n \text{ such that } F(x_*) = 0, \tag{1.1}$$

where it is assumed that n is large (say, $n > 100$) and $F(x)$ is a least once continuously differentiable. Large systems of nonlinear equations arise frequently in many practical applications including various network-flow problems and equations produced by finite-difference or finite-element discretizations of boundary values problems for ordinary and partial differential equations. In many situations, $F'(x_*)$ is ill-conditioned or singular. In such situations, tensor methods are especially intended to improve upon the efficiency of standard algorithms based on Newton's method.

Tensor methods for large, sparse systems of nonlinear equations were introduced in Bouaricha and Schnabel [3]. These methods base each iteration on a quadratic model of $F(x)$ that has the form

$$M(x_c + d) = F(x_c) + F'(x_c)d + \frac{1}{2} T_c dd, \tag{1.2}$$

where x_c is the current iterate, and $T_c \in R^{n \times n \times n}$ is the tensor term at x_c. The notation $T_c dd$ is defined as follows.

Definition 1.1. Let $T \in R^{n \times n \times n}$. Then T is composed of n horizontal faces $H_i \in R^{n \times n}, i = 1, \cdots, n$, where $H_i[j,k] = T[i,j,k]$. For $v, w \in R^n, Tvw \in R^n$ with

$$Tvw[i] = v^T H_i w = \sum_{j=1}^{n} \sum_{k=1}^{n} T[i,j,k] v[j] w[k].$$

The tensor term is selected so that the model interpolates a very small number, p, of function values from previous iterations. This interpolation process results in T_c being a rank p tensor, which is crucial to the efficiency of the tensor method. After the model (1.2) is formed, the problem

$$\text{find } d \in R^n \text{ that minimizes } \| M(x_c + d) \|_2 \tag{1.3}$$

is solved; that is, at each iteration of tensor methods, a minimizer of the model is used if no root exists. Methods for forming the tensor term and solving the tensor model for sparse systems of nonlinear equations are reviewed in more detail in the next section. The tensor method requires no more derivative or function information per iteration than Newton's method, and its storage requirement is not appreciably more than for Newton's method.

In tests reported in [3], the tensor method virtually never is less efficient than a standard method based upon a linear (Newton) model, and usually is more efficient.

One of the major contributions of tensor methods has been its great robustness. Experimental results in [2], [3] have shown that the tensor method solves a considerable number of problems that the standard Newton method does not, and the reverse is virtually never true.

The tensor methods for large, sparse nonlinear equations described in Bouaricha [1] use sparse direct methods for solving the linear systems of equations that arise in each tensor iteration. Unfortunately, for problems where the factorization of the Jacobian matrix is too expensive, the slow asymptotic performance and large storage requirements of direct methods make them impractical. Thus, for these systems robust and fast iterative solvers such as Krylov algorithms must be used.

Newton-Krylov methods for nonlinear equations have been studied by Brown and Saad ([5], [4]). They show that their methods are remarkably effective on large systems of partial differential equations. Their success has motivated us to use Krylov subspace algorithms in conjunction with the tensor methods developed in [1].

The tensor methods developed in [1] require at least $p+1$ back solves, where p is the number of interpolated function values from previous iterations. Usually, the cost of these solves is relatively small compared with the cost of the Jacobian factorization. Unfortunately, this is not the case when iterative methods are used. If $p = 1$, say, then it appears that the cost of a tensor-Krylov iteration will be approximately twice that of a Newton-Krylov iteration. However, by reformulating the tensor step computation in [1], we show in §5 that the cost of one of the two solves can be made cheaper, as a result of a good initial estimate of the solution.

The aim of this paper is to show that tensor-Krylov methods are much more robust than Newton-Krylov methods in solving large and complex systems of nonlinear equations and that it may still be preferable to use Newton-Krylov methods when the nonlinear equations problem is not too difficult to solve.

The remainder of this paper is organized as follows. In §2 we review tensor methods for large, sparse nonlinear equations that were introduced in [1]. In §3 we review Krylov subspace methods and their main properties. In particular, we review the *generalized minimum residual method* (GMRES) [10], which will be used in conjunction with our tensor methods. In §4 we review the Newton-Krylov methods for large nonlinear equations that were described in [5]. In §5 we introduce tensor-Krylov algorithms for large nonlinear equations. We present some numerical results in §6. Finally, in §7 we make some concluding remarks.

2. Overview of Tensor Methods for Large, Sparse Nonlinear Equations

Tensor methods are general–purpose methods intended especially for problems where the Jacobian at the solution is singular or ill–conditioned. Each iteration is based on a quadratic model of the nonlinear function (1.2). The choice of T_c causes the second–order term $T_c dd$ in (1.2) to have a simple and useful form. We use the tensor term to allow the local model $M(x_c + d)$ to interpolate values of the function $F(x)$ at past iterates; that is, the model should satisfy

$$F(x_{-k}) \;=\; F(x_c) \;+\; F'(x_c)s_k \;+\; \frac{1}{2}T_c s_k s_k, \qquad k \;=\; 1,\cdots,p,$$

where

$$s_k \;=\; x_{-k} \;-\; x_c, \qquad k \;=\; 1,\cdots,p.$$

The past points $x_{-1},\cdots,x_{-p}$ are selected so that the set of directions s_k from x_c to the selected points must be strongly linearly independent; we require that each direction s_k make an angle of at least 45 degrees with the subspace spanned by the previously selected past directions. In practice, p is usually 1 or 2. The procedure of finding linearly independent directions is implemented easily by using a modified Gram-Schmidt method.

After selecting the linearly independent past directions, we form the tensor term. Schnabel and Frank [11] choose T_c to be the smallest matrix that satisfies the interpolation conditions;

that is,

$$\min_{T_c \in R^{n\times n\times n}} \quad \| T_c \|_F \tag{2.1}$$

$$\text{subject to } T_c s_k s_k \;=\; 2\,(F(x_{-k}) \;-\; F(x_c) \;-\; F'(x_c)s_k),$$

where $\| T_c \|_F$, the Frobenius norm of T_c is defined by

$$\| T_c \|_F{}^2 \;=\; \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \; (T_c[i,j,k])^2.$$

The solution to (2.1) is the sum of p rank–one tensors whose horizontal faces are symmetric,

$$T_c \;=\; \sum_{k=1}^{p} \; a_k s_k s_k,$$

where a_k is the $k-$th column of $A \in R^{n\times p}$, A defined by $A = ZM^{-1}$, Z is an $(n \times p)$ matrix whose columns are $Z_j = 2(F(x_{-j}) - F(x_c) - F'(x_c)s_j)$ and M is a $(p \times p)$ matrix defined by $M(i,j) = ({s_i}^T s_j)^2$, $1 \le i,\; j \le p$.

If we use the tensor term derived above, the tensor model (1.2) becomes

$$M(x_c \;+\; d) \;=\; F(x_c) \;+\; F'(x_c)d \;+\; \frac{1}{2} \sum_{k=1}^{p} \; a_k \; \{d^T s_k\}^2. \tag{2.2}$$

The simple form of the second term in (2.2) is the key to being able to efficiently form, store, and solve the tensor model. The cost of forming the tensor term and the tensor model is $O(n^2 p)$ arithmetic operations. The additional storage required is $4p$ n-vectors. In the remainder of this paper, we will denote $F(x_c)$ and $F'(x_c)$ by F and J, respectively, for simplicity.

Once the tensor model is formed, a step $d \in \Re^n$ is computed such that

$$M(x_c \;+\; d) \;=\; F \;+\; Jd \;+\; \frac{1}{2} \sum_{k=1}^{p} \; a_k \; \{d^T s_k\}^2 \;=\; 0. \tag{2.3}$$

Bouaricha and Schnabel [3] show that the solution of (2.3) can be reduced to the solution of a system of p quadratic equations in p unknowns, plus the solution of $p + 1$ systems of linear equations that all involve the same matrix. This matrix is either $J(x_c)$ if it is nonsingular and well conditioned, or $J(x_c)$ augmented by p dense rows and columns if $J(x_c)$ is singular or ill–conditioned. They also show that their algorithm efficiently solves the generalization of (2.3),

$$\text{find } \; d \in R^n \text{ that minimizes } \| M(x_c \;+\; d) \|_2. \tag{2.4}$$

The basic approach is illustrated by the case when the Jacobian matrix is nonsingular and the tensor model has a root. In this case, premultiplying (2.3) by ${s_i}^T J^{-1}, i = 1, \cdots, p$, gives the p quadratic equations in the p unknowns $\beta_i = {s_i}^T d$,

$${s_i}^T J^{-1} F + \; \beta_i \;+\; \frac{1}{2} \sum_{k=1}^{p} \; ({s_i}^T J^{-1} a_k) {\beta_k}^2 \;=\; 0, \quad i \;=\; 1, \cdots, p. \tag{2.5}$$

These equations can be solved for $\beta_i, i = 1, \cdots, p$, and then from (2.3) the equation

$$F + \ Jd \ + \ \frac{1}{2}\sum_{k=1}^{p} \ a_k{\beta_k}^2 \ = \ 0$$

can be solved for d. The entire process requires the solution of $p+1$ systems of linear equations in the matrix J to compute $J^{-1}F$ and $J^{-1}a_k, k = 1, \cdots, p$ (or, alternatively, $J^{-1}(F + \frac{1}{2}\sum_{k=1}^{p} a_k{\beta_k}^2)$ and $J^{-T}s_i, i = 1, \cdots, p$) and the solution of the small system of quadratics (2.5).

The preceding paragraph indicated how to solve (2.4) efficiently when the Jacobian matrix is nonsingular and the tensor model has a root. Now we address the more general problem of solving (2.4) efficiently whether or not the model has a root, when the Jacobian matrix is nonsingular. We do this by considering the equivalent minimization problem to (2.4),

$$\min_{d \in \ R^n} \ \ || \ Q \ M(x_c \ + \ d) \ ||_2, \tag{2.6}$$

where Q is an $n \times n$ orthogonal matrix that has the structure

$$Q = \begin{bmatrix} U^T \\ Z^T \end{bmatrix},$$

with

$U \in \Re^{n \times p}$; $U = J^{-T}S[S^T(J^TJ)^{-1}S]^{-\frac{1}{2}}$, S an $(n \times p)$ matrix whose columns are $s_i, i = 1, \cdots, p$
$Z \in \Re^{n \times (n-p)}$ is an orthonormal basis for the orthogonal complement of the subspace spanned by the columns of $J^{-T}S$.

Note that $Z^TJ^{-T}S = 0$. If we define $W = [S^T(J^TJ)^{-1}S]$, $\beta = S^Td$, and

$$q(\beta) = \ S^TJ^{-1}F \ + \ \beta \ + \ \frac{1}{2}S^TJ^{-1}A\beta^2, \tag{2.7}$$

where β^2 denotes the vector in $\Re^p$ whose i-th component is $(\beta_i)^2$, then

$$QM(x_c + \ d) \ = \ \begin{bmatrix} W^{-\frac{1}{2}}q(\beta) \\ Z^T \ M(x_c \ + \ d) \end{bmatrix}. \tag{2.8}$$

The following lemma is the key to showing that (2.6) can be solved efficiently through (2.8).

Lemma 2.1. For any $\beta \in \Re^p$, there exists a $d \in \Re^n$ such that $Z^TM(x_c + d) = 0$ and $S^Td = \beta$.
Proof. Let

$$d = \ (J^TJ)^{-1}SW^{-1}\beta \ + \ J^{-1}Z \ t, \tag{2.9}$$

where t is arbitrary vector $\in \Re^{n-p}$. Then

$$S^Td = \ S^T(J^TJ)^{-1}SW^{-1}\beta \ + \ S^TJ^{-1}Z \ t \ = \ \beta,$$

from the definitions of W and Z, and

$$\begin{aligned} Z^T M(x_c + d) &= Z^T F + Z^T J[(J^T J)^{-1} S W^{-1}\beta + J^{-1} Z\, t] + \tfrac{1}{2} Z^T A\beta^2 \\ &= Z^T F + t + \tfrac{1}{2} Z^T A\beta^2. \end{aligned}$$

Thus the choice

$$t = -Z^T[F + \frac{1}{2}A\beta^2]$$

in (2.9) yields a value of d for which $Z^T M(x_c + d) = 0$ and $S^T d = \beta$ are both satisfied.□

Since for any β, we are able to find a step d such that $Z^T M(x_c + d) = 0$ and $S^T d = \beta$, Lemma 2.1 and (2.8) show that problem (2.6) can be reduced to the minimization problem in p variables

$$\min_{\beta \in \Re^p} \| W^{-\frac{1}{2}} q(\beta) \|_2. \tag{2.10}$$

Furthermore, once the value of β that solves (2.10) is determined, we can obtain the solution d to (2.6) efficiently as follows. From (2.8) and Lemma 2.1, d_* must satisfy

$$\begin{aligned} M(x_c + d_*) &= Q^T \begin{bmatrix} W^{-\frac{1}{2}} q(\beta) \\ 0 \end{bmatrix} \\ &= U W^{-\frac{1}{2}} q(\beta). \end{aligned}$$

From this equation and the definition of U we have

$$F + J d_* + \frac{1}{2}A\beta^2 = J^{-T} S W^{-1} q(\beta)$$

and, hence,

$$d_* = -J^{-1}[\, F + \frac{1}{2}A\beta^2 - J^{-T} S W^{-1} q(\beta)]. \tag{2.11}$$

Therefore, once we know β, we simply calculate the value of $q(\beta)$ and substitute these two values into Equation (2.11) to obtain the value of d_*.

An algorithm that solves (2.4) is summarized as follows.

Algorithm 2.2. Solving the Sparse Tensor Model

Let $J \in R^{n\times n}$ be sparse, $F \in R^n$, S, $A \in R^{n\times p}$.

1. Form the $q(\beta)$ equation (i.e, Equation (2.7)) by calculating $J^{-T}S$ as follows: factor J, and solve $J^T y_j = s_j, j = 1, \cdots, p$.
2. Form the positive definite matrix $W \in R^{p\times p}$, where $W_{ij} = [{s_i}^T (J^T J)^{-1} s_j]$, $1 \le i, j \le p$, as follows: $W_{ij} = (J^{-T} s_i)^T (J^{-T} s_j) = {y_i}^T y_j$.
3. Perform a Cholesky decomposition of W (i.e., $W = LL^T$) resulting in $L \in R^{p\times p}$, a lower triangular matrix.

4. Use UNCMIN [12], an unconstrained minimization software package, to solve

$$\min_{\beta \in R^p} \; || \, L^{-1} \; q(\; \beta \;) ||_2{}^2, \tag{2.12}$$

or solve (2.12) in closed form if $p = 1$.

5. Substitute the value of the solution to Equation (2.12), β_*, and $q(\beta_*)$ into the following equation for d,

$$d \; = \; -J^{-1}[F \; + \; \frac{1}{2}A\beta_*{}^2 \; - \; J^{-T}SW^{-1}q(\beta_*)]$$

to obtain the tensor step d; this involves one additional solve, since the factorization of J is already calculated.

The total cost of this process is the factorization of the sparse matrix J, $p+1$ back solves using this factorization, the unconstrained minimization of a function of p variables, and some lower-order ($O(n)$) costs.

For further information we refer to [1].

3. Krylov Subspace Methods

The underlying idea of Krylov subspace methods when applied to a linear system of equations, $Ax = b$, is to generate an approximate solution to the original problem from the Krylov subspace Span$\{b, Ab, \cdots, A^{m-1}b\}$. Therefore, the original problem of size N is approximated by one of dimension m, typically much smaller that N.

Consider the linear system

$$Ax \; = \; b, \tag{3.1}$$

where $A \in \Re^{n\times n}$ and $b \in \Re^n$. Let x_0 be an initial guess of the solution x_* of (3.1), and let r_0 be the initial residual $r_0 \; = \; b \; - \; Ax_0$. If the unknown x is decomposed as $x \; = \; x_0 \; + \; z$, then it is immediate that the new unknown z must satisfy

$$Az \; - \; r_0 \; = \; 0. \tag{3.2}$$

A Krylov subspace method obtains an approximation $z^{(m)}$ to the system (3.2) by applying a projection process to the system onto the Krylov subspace $K_m \; = \; \text{Span}\{r_0, \; Ar_0, \; ..., \; A^{m-1}r_0\}$. That is, the approximation $z^{(m)}$ must be satisfy

$$z^{(m)} \in K_m$$

and

$$(Az^{(m)} \; - \; r_0) \; \perp \; v_j, \; j \; = \; 1, \cdots, m,$$

where $v_j, \; j \; = \; 1, \cdots, m$ form a basis for K_m. If $V_m \; = \; [v_1, \cdots, v_m]$ is any basis of K_m, then $z^{(m)}$ can be expressed as $z^{(m)} \; = \; V_m \; \cdot \; y^{(m)}$, where $y^{(m)}$ is the solution to the $m \times m$ system

$$V_m{}^T A V_m \; \cdot \; y^{(m)} \; - \; V_m{}^T r_0 \; = \; 0, \tag{3.3}$$

and the approximate solution $x^{(m)}$ of the system (3.1) is related to $z^{(m)}$ by $x^{(m)} \; = \; x_0 \; + \; z^{(m)}$.

For further information we refer to [8], [10], and [9].

3.1. GMRES Method

We first review Arnoldi's algorithm for building an orthonormal basis of the Krylov subspace K_m, on which the GMRES method is based.

Arnoldi's algorithm builds an orthonormal basis $v_1, \cdots, v_m$ of $K_m = \mathrm{Span}\{r_0,\ Ar_0, \cdots, A^{m-1}r_0\}$ by the recurrence

$$h_{k+1,k}v_{k+1} = Av_k - \sum_{i=1}^{k} h_{ik}v_i \tag{3.4}$$

starting with $v_1 = r_0/\|\ r_0\ \|$ and choosing $h_{ik},\ i = 1, \cdots, k+1$, in such a way that v_{k+1} is orthogonal to $v_1, \cdots, v_k$ and $\|\ v_{k+1}\ \| = 1$. The algorithm is as follows.

Algorithm 3.1. Arnoldi's Method

```
1. Start: Compute $r_0 = b - Ax_0$, and take $v_1 = r_0/\| r_0 \|$.

2. Iterate: For $j = 1, 2, \cdots, m$, do
        $h_{i,j} = (Av_j,\ v_i),\ i = 1, 2, \cdots, j$
        $\hat{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$
        $h_{j+1,j} = \| \hat{v}_{j+1} \|_2$
        $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$.
```

It is clear that after m steps of Arnoldi's algorithm we have an orthonormal basis of K_m, $[v_1, \cdots, v_m]$, and an $(m+1) \times m$ Hessenberg matrix $\bar{H}_m$ whose nonzero elements are the h_{ij} defined by Step 2 of Algorithm 3.1. Furthermore, the vector v_i and the matrix $\bar{H}_k$ satisfy the important relation

$$AV_k = V_{k+1}\bar{H}_k. \tag{3.5}$$

The GMRES method seeks a solution of the least squares problem

$$\min_{z_m \in K_m} \|\ b - A(x_0 + z_m)\ \| = \min_{z_m \in K_m} \|\ r_0 - Az_m\ \|. \tag{3.6}$$

If we set $z_m = V_m y$, $v_1 = r_0/\|r_0\|$ and $\beta = \|\ r_0\ \|$, then the least squares problem (3.6) is equivalent to

$$\begin{aligned} &\min\nolimits_{y \in R_m} \|\ \beta v_1 - AV_m y\ \| \\ =\ &\min\nolimits_{y \in R_m} \|\ V_{m+1}(\beta e_1 - \bar{H}_m y)\ \| \\ =\ &\min\nolimits_{y \in R_m} \|\ \beta e_1 - \bar{H}_m y\ \|. \end{aligned} \tag{3.7}$$

Hence, the solution of the least squares problem (3.6) is given by

$$x_m = x_0 + V_m y_m, \tag{3.8}$$

where y_m minimizes problem (3.7).

We now give the following structure of the method.

Algorithm 3.2. GMRES Method

```
1. Start: Choose $x_0$, and compute $r_0 = b - Ax_0$ and $v_1 = r_0 / || r_0 ||$.

2. Iterate: For $j = 1, 2, \cdots, k, \cdots$,until satisfied, do
        $h_{i,j} = (Av_j, v_i), i = 1, 2, \cdots, j$
        $\hat{v}_{j+1} = Av_j = \sum_{i=1}^{j} h_{i,j} v_i$
        $h_{j+1,j} = || \hat{v}_{j+1} ||$
        $v_{j+1} = \hat{v}_{j+1} / h_{j+1,j}$.

3. Form the approximate solution:
   $x_k = x_0 + V_k y_k$, where $y_k$ minimizes (3.7).
```

A typical implementation of the GMRES method limits m to a fixed value m_{max} to minimize the storage allocation. It is possible that $m = m_{max}$ in the Arnoldi process, and yet $|| b - Ax_m ||_2$ is still greater than ϵ_n, where ϵ_n is a stopping criterion. One way to deal with this case is to set x_0 equal to $x^{(m)}$ and restart the Arnoldi process, effectively restarting the Krylov method. This procedure does not always guarantee convergence but does appear to work well in practice.

4. Newton-Krylov Methods

In this section we review the nonlinear version of the GMRES method described in Brown and Saad [5], which combines it with a Newton iteration, for solving nonlinear systems of equations. A method in this class is referred to as a Newton-Krylov algorithm.

The Newton-GMRES method has the following general form:

Algorithm 4.1. Newton-GMRES Method

```
1. Choose $x_0$, and an initial guess for $x_*$.

2. For $k = 0, 1, \cdots$, until convergence,

    • Choose $\eta_k \in [0, 1)$.
    • Compute a vector $\delta_k$ satisfying

          $J(x_k)\delta_k = -F(x_k) + r_k$,          (4.1)

      with $\frac{|| r_k ||}{|| F(x_k) ||} \leq \eta_k$, using a GMRES method.
    • Compute a next iterate $x_{k+1}$ using a backtracking line search global strategy, i.e.,
      $x_{k+1} = x_k + \lambda\delta_k$, where $\lambda$ is the line search damping parameter.
```

The residual r_k represents the amount by which δ_k fails to satisfy the Newton equation $J(x_k)\delta_k = -F(x_k)$. The forcing sequence $\eta_k \in [0, 1)$ is used to control the level of accuracy. Brown and Saad [4] show that if the sequence $\eta_k \to 0$, then if $J(x_*)$ is nonsingular and $\eta_k \leq \eta_{max} < 1$, the iterates generated by Algorithm 4.1 converge to the solution superlinearly; the convergence is quadratic if $\eta_k = O(||F(x_k)||)$.

5. Tensor-Krylov Methods

In this section, we describe how we construct tensor-Krylov algorithms for nonlinear equations problems. In particular, we emphasize the key differences between tensor-Krylov methods and the tensor methods reviewed in §2.

In tensor-Krylov methods, we form the $q(\beta)$ equation by only forming $J_c^{-1}A$, where J_c is the Jacobian at the current iterate, as follows. Let a_i, $i = 1, \cdots, p$ denote the columns of the tensor matrix A. Thus, to calculate $J_c^{-1}A$, we need to solve the linear systems of equations $J_c y_i = a_i, i = 1, \cdots, p$, for y_i. Note, however, that each column a_i is a linear combination of F_c, $J_c s_{-i}$, and F_{-i}, where F_c is the function value at the current iterate x_c, $s_{-i} = x_{-i} - x_c$ is the direction from x_c to the past iterate x_{-i}, and F_{-i} is the function value at x_{-i}. If $p = 1$ for example, then

$$a_1 = \frac{2}{(s_{-1}^T s_{-1})^2}[F_{-1} - F_c - J_c s_{-1}].$$

Therefore, to compute $J_c^{-1}a_1$, we need to compute $J_c^{-1}F_{-1}$ and $J_c^{-1}F_c$, where $J_c^{-1}F_c$ is the Newton direction. In computing $J_c^{-1}F_{-1}$, an advantage of using iterative methods is that if we have a good starting guess, the cost of solving the system of linear equations will be a lot less. Therefore, if we start with the vector $J_{-1}^{-1}F_{-1}$, which was computed approximately at the previous iteration, as an initial guess to solve the system of equations $J_c x = F_{-1}$, then an iterative method such as Krylov method might eliminate most of the cost of that calculation.

The other key difference between the tensor-Krylov methods presented in this paper and the tensor methods reviewed in §2 is in the way the tensor step is computed. In tensor-Krylov methods, we omit the expression $-(J_c^T J_c)^{-1} S W^{-1} q(\beta)$ from the tensor step calculation (2.13), in the case where the $q(\beta)$ equation has a minimizer. Leaving this expression in would require another linear system of equations to be solved, thereby making the cost of a tensor-Krylov iteration prohibitive. It can be proven that this new way of computing the tensor step will retain the same convergence properties of the tensor method because, in the neighborhood of the solution x_*, the above expression is negligible and therefore has no effect on the tensor method. In our experiments, omitting this term generally has a negligible effect on the number of iterations required by the tensor method.

An iteration of tensor-Krylov algorithms for large nonlinear equations is given in Algorithm 5.1. For simplicity, we assume throughout the remainder of this paper that $p = 1$.

Algorithm 5.1. Tensor-Krylov Methods for Large Nonlinear Equations

Let $J_c \in R^{n \times n}$ be the Jacobian at the current iterate x_c, $F_c \in R^n$ the function value at x_c, $F_{-1} \in R^n$ the function value at the previous iterate x_{-1}, $s_{-1} \in R^n$ the step from x_c to x_{-1} ($s_{-1} = x_{-1} - x_c$), $a_1 \in R^n$ the tensor term ($a_1 = \frac{2}{(s_{-1}^T s_{-1})^2}[F_{-1} - F_c - J_c s_{-1}]$), and d_n and d_t the Newton and tensor steps, respectively.

1. Form the $q(\beta)$ equation

$$\frac{1}{2} s_{-1}^T J_c^{-1} a_1 \beta^2 \;+\; \beta \;+\; s_{-1}^T J_c^{-1} F_c \;=\; 0.$$

- Solve $J_c d_n = F_c$, using a preconditioned (restarted) GMRES method starting from $d_s = 0$.
- Solve $J_c y = F_{-1}$, using a preconditioned (restarted) GMRES method starting from $y_s = J_{-1}^{-1} F_{-1}$, where J_{-1} is the Jacobian at x_{-1}.
- Calculate $J_c^{-1} a_1 = \dfrac{2}{(s_{-1}^T s_{-1})^2}[y - d_n - s_{-1}]$.

2. Solve $q(\beta) = 0$, for β.
3. Calculate d_t as follows:

$$d_t \;=\; -J_c^{-1} F_c \;-\; \frac{1}{2} J_c^{-1} a_1 {\beta_*}^2,$$

where β_* is the solution obtained in Step 2.
4. Select the next iterate x_+, using a line search global strategy as follows:

```
$f(x_c) = \frac{1}{2}||F(x_c)||_2^2$
$x_+^t = x_c + d_t$
$f(x_+^t) = \frac{1}{2}||F(x_+^t)||_2^2$
$slope = -||F_c||_2 - \frac{1}{2}{F_c}^T a_1 {\beta_*}^2$
    if ($slope < 0$) then
        if $f(x_+^t) < f(x_c) + 10^{-4} \cdot slope$ then
            $x_+ = x_+^t$
        else
            Find an acceptable $x_+^n$ in the Newton direction $d_n$,
            using a line search algorithm (Algorithm A6.3.1, page 325 [6])
            Find an acceptable $x_+^t$ in the tensor direction $d_t$,
            using a line search algorithm (Algorithm A6.3.1, page 325 [6])
            if $f(x_+^n) < f(x_+^t)$ then
                $x_+ = x_+^n$
            else
                $x_+ = x_+^t$
            endif
        endif
    else
        Find an acceptable $x_+^n$ in the Newton direction $d_n$,
        using a line search algorithm (Algorithm A6.3.1, page 325 [6])
        $x_+ = x_+^n$
    endif
```

6. Numerical Testing

In this section, we present some results of testing tensor-Krylov methods on a set of nonlinear partial differential equations. We also compare these results with those obtained by Newton-Krylov methods [5] on the same problems.

All our computations were performed on a Sun SPARCstation 10, using double–precision arithmetic.

The tensor-Krylov and Newton-Krylov programs terminate successfully if $|| F(x_+) ||_\infty$ is less than 10^{-9}. The test

$$\max_i \left\{ \frac{|x_+[i] - x_c[i]|}{\max\{|x_+[i], 1|\}} \right\} \leq 10^{-9}$$

is used to determine whether the algorithms have converged or stalled at x_+. The iteration limit is set to 500.

We ran the tensor-Krylov method with $p = 1$. The reasons for this choice are that previous computational results obtained in [1] showed that the tensor method with $p = 1$ is generally about as effective as the tensor method that allows $p \geq 1$, and that the method is considerably simpler and cheaper to implement in this case. Note that the tensor method will have the same convergence properties whether $p = 1$ or $p \geq 1$.

In the test results of the remainder of this section, the following notations are used:

- K_{dim} - dimension of the Krylov subspace
- *precond* - type of preconditioner used
- *itns* - number of nonlinear iterations
- *fevals* - number of function evaluations
- *nli* - number of linear iterations within the Krylov method
- *nb* - number of backtracks within the line search algorithm
- F - line search method cannot locate a point lower than the current one – algorithm fails to converge
- IL - iteration limit exceeded

We use two stopping criteria for the linear Krylov method at iteration k of the nonlinear equations algorithm. We require that the l_2 norm of the residual be at least as small as $1.0e - 3$ and that the residual reduction be at least $1.0e - 2$, that is,

$$\frac{||\text{ residual at current Krylov iterate }||}{||\text{ residual at initial Krylov iterate }||} \leq 1.0e - 2.$$

The maximum linear solve iterations, m_{max}, is set to 50. If m_{max} is reached, but either the l_2 norm of the residual is bigger than $1.0e - 3$ or the residual reduction is bigger than $1.0e - 2$, we use the last computed Krylov step. The default size of the Krylov subspace is set to 10.

Test Problem 1. The Bratu problem [5]

$$-\Delta u + \alpha u_x + \lambda e^u = f \tag{6.1}$$

We solve (6.1) over the unit square of R^2 with Dirichlet boundary conditions. We discretize this problem using a five-point finite differencing and obtain a large system of nonlinear equations of size N, where $N = n_x^2$ and n_x is the number of meshpoints in each direction. We choose f so that the solution of the discretized problem is the constant unity. We run this problem with $N = 1024$ ($n_x = 32$), with $\alpha = 10.0$, and with different values of λ ranging from -5.0 to 10^{12}.

Test Problem 2. The driven cavity problem (given in stream-vorticity formulation) [5]

$$\nu \Delta \omega \; + \; (\psi_{x_2} \omega_{x_1} \; - \; \psi_{x_1} \omega_{x_2}) \; = \; 0 \quad \text{in} \quad \Omega \tag{6.2}$$

$$-\,\Delta \psi \; = \; \omega \quad \text{in} \quad \Omega \tag{6.3}$$

$$\psi \; = \; 0 \quad \text{on} \quad \partial\Omega \tag{6.4}$$

$$-\,\frac{\partial \psi}{\partial n}(x_1, x_2) \mid_{\partial\Omega} \; = \; \begin{cases} 1 & \text{if } x_2 = 1 \\ 0 & \text{if } 0 \le x_2 < 1 \end{cases} \tag{6.5}$$

Here $\Omega = \{(x_1, x_2) : 0 < x_1 < 1, 0 < x_2 < 1\}$, and the viscosity ν is the reciprocal of the Reynolds number R_e. After discretization by five-point finite differencing, we obtain a system of nonlinear equations. In this test we chose $N = 3969$ ($n_x = 63$). We tried six different values for R_e: 500, 1000, 1500, 2000, 3000, and 5000.

Test Problem 3. The incompressible Navier-Stokes and thermal energy problem (given in stream-vorticity formulation) [13]

$$\frac{\partial^2 \Omega}{\partial x^2} \; + \; Ax^2 \frac{\partial^2 \Omega}{\partial y^2} \; = \; \Pi_1 \frac{\partial \Phi}{\partial x} \; + \; \Pi_2 \frac{\partial \Phi}{\partial y} \tag{6.6}$$

$$\frac{\partial^2 \Psi}{\partial x^2} \; + \; Ax^2 \frac{\partial^2 \Psi}{\partial y^2} \; = \; \frac{-Ax^2}{4} \Omega \tag{6.7}$$

$$Ax \frac{\partial \Psi}{\partial y} \frac{\partial \Phi}{\partial x} \; - \; Ax \frac{\partial^2 \Psi}{\partial x} \frac{\partial^2 \Phi}{\partial y} \; = \; \frac{\partial^2 \Phi}{\partial x^2} \; + \; Ax^2 \frac{\partial^2 \Phi}{\partial y^2} \tag{6.8}$$

Equations (6.6)–(6.8) describe buoyancy induced natural convection in an inclined two–dimensional rectangular cavity over the computational domain $(-1, 1) \times (-1, 1)$. In the above equations

$$\Pi_1 \; \equiv \; -Ax \frac{R_a \; cos\Theta}{2}, \quad \Pi_2 \; \equiv \; (Ax)^2 \frac{R_a \; sin\Theta}{2},$$

Ax is the aspect ratio of a rectangular cavity, Θ is the angle of inclination, and R_a is the Raleigh number. In this formulation the Prandtl number has been assumed to be infinite so that the inertial terms in the momentum equation can be neglected. We took $Ax = 1, \Theta = \pi/2$, and $R_a = 10^3, 10^4, 10^5,$ and 10^6. We used Dirichlet boundary conditions for the stream function, Ψ; a mixed Dirichlet/Neumann condition for temperature, Φ; and a first–order approximation of the stream function-vorticity relationship at the boundary for the vorticity boundary conditions, where the vorticity is represented by Ω in equations (6.6)–(6.8). We used central difference approximations to obtain a system of nonlinear equations. We ran this problem with $N = 1024$.

Since the tensor method requires two solves per iteration, the use of block iterative algorithms (see e.g., [15, 14]) may be preferable than standard iterative methods in this case. A clear advantage is that under certain conditions of the residual block and the Jacobian matrix, block iterative methods achieve finite termination in $[n/2]$ iterations in solving the two tensor solves. This makes block iterative algorithms mathematically attractive. Block iterative methods can also help reduce the effect of the sequential inner products in parallel environments. Thus, they are of a great practical value in applications involving several right sides but they are not as well studied from the theoretical point of view. Consequently, we used standard iterative methods in this paper.

We experimented with three Krylov solvers: the generalized minimum residual (GMRES) [10], conjugate gradient squared (CGS) [16] , and the Bi-CGSTAB [17] methods. There appears to be no significant difference in performance among the three methods on the test problems described above. As a result, we discuss the test results only when the GMRES method was used in conjunction with the tensor and Newton methods. Furthermore, we have tested a wide variety of standard preconditioners which include Jacobi, red-black Gauss-Seidel, polynomial (Neumann series expansion to approximate the inverse of a matrix), incomplete Cholesky, incomplete LU, and multigrid (one "V" cycle) preconditioners. The multigrid preconditioner, *mg*, appears to be the most effective and the most robust one. Therefore, we report only the multigrid test results.

We tested both the Newton-GMRES and tensor-GMRES methods on the test problems above. Tables 1, 2, and 3 show the test results for problems 1, 2, and 3, respectively. Figures 1–12 show the contour plots of the streamlines and equivorticities for different values of the Reynolds number. The contour levels plotted for the streamlines are

$$\begin{aligned}\psi \quad = \quad & -0.1, -0.08, -0.06, -0.04, -0.02, 0.0, \\ & 0.00005, 0.0001, 0.0005, 0.001, 0.0025.\end{aligned}$$

The contour levels plotted for the equivorticities are

$$\omega \;=\; -5.0, -3.0, -1.0, 1.0, 3.0, 5.0.$$

Figures 13–20 show the streamlines for different values of the Raleigh number. The contour levels plotted for the streamlines are

$$\psi \;=\; -1.0, -0.9, -0.8, -0.7, -0.6, -0.5, 0.0, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.$$

On the basis of Tables 1–3, the following observations can be made. Table 1 shows that if we pick λ to be a very large value, then the Jacobian at the solution becomes nearly singular. This is exactly the case in which the tensor method is intended to improve upon the performance of Newton's method due to the faster local convergence properties of the tensor method [1, 7]. Clearly, the tensor-GMRES method outperforms the Newton-GMRES method on this test problem for large values of λ in nonlinear iterations, function evaluations, and GMRES iterations.

When the default Krylov subspace is used, both methods have failed to solve test problems 2 and 3 for high Reynolds and Raleigh numbers. A possible reason for the convergence failures of Newton's method is that the Krylov subspace dimension is not large enough to generate a good approximate solution $x^{(m)}$ to effectively restart the Arnoldi process. This results in the GMRES method not generating sufficiently good descent directions for the nonlinear iteration. The convergence failures of the tensor-Krylov method, however, are due to the fact that the coefficients of the $q(\beta)$ equation, $J_c^{-1}F_c$ and/or $J_c^{-1}a_1$, are not solved within the required accuracies in the early nonlinear iterations. As a result, the tensor-Krylov method generates poor tensor directions, which cause the line search to fail in locating a next iterate at some point of the

Table 1: Test results for problem 1

			Newton-GMRES				Tensor-GMRES			
λ	K_{dim}	*precond*	*itns*	*fevals*	*nli*	*nb*	*itns*	*fevals*	*nli*	*nb*
-5.0	10	mg	5	6	15	0	5	6	23	0
1.0	10	mg	5	6	14	0	5	6	21	0
1.0e+3	10	mg	11	12	77	0	9	10	95	0
1.0e+6	10	mg	20	21	84	0	13	14	82	0
1.0e+9	10	mg	32	43	118	10	16	29	89	9
1.0e+12	10	mg	30	32	186	1	9	20	84	8

Table 2: Test results for problem 2

			Newton-GMRES				Tensor-GMRES			
R_e	K_{dim}	*precond*	*itns*	*fevals*	*nli*	*nb*	*itns*	*fevals*	*nli*	*nb*
500	10	mg	10	14	166	3	11	19	281	5
1000	10	mg	11	16	198	4	15	33	415	11
1500	10	mg	12	17	335	4	15	33	642	11
2000	15	mg	F	–	–	–	36	55	2075	12
3000	20	mg	72	193	3459	120	35	122	2563	59
5000	50	mg	IL	–	–	–	100	498	8814	308

algorithm execution. To overcome this problem, we increased the Krylov subspace dimensions between 10 and 50, which yielded the results in Tables 2 and 3.

It can be concluded from these two tables that as the Reynolds and Raleigh numbers grow larger, both Newton-GMRES and tensor-GMRES methods produce poor descent directions, as is clearly indicated by the high number of line search backtracks. Nevertheless, the tensor-GMRES method converges for all R_e and R_a, whereas the Newton-GMRES method fails to solve test problem 2 for $R_e = 2000$ and $R_e = 5000$ and test problem 3 for $R_a = 1.0e + 6$. These results suggest that a poor tensor-GMRES step is in general better that a poor Newton-GMRES step. An important observation that can be made from Tables 1, 2, and 3 is that the average cost of a tensor-GMRES iteration is approximately 1.5 that of a Newton-Krylov iteration. This is due to the fact that the $J_c y = F_{-1}$ solve required by the tensor iteration generally converges in about half of the number of GMRES iterations required by the $J_c d_n = F_c$ solve. This result is likely attributed to the good starting guess ${J_{-1}}^{-1} F_{-1}$.

Over the three test problems, the Newton-Krylov method is very competitive up to the values of $\lambda = 1.0e + 3$, $R_e = 1500$, and $R_a = 1.0e + 5$. Then, the tensor-Krylov method becomes much more efficient and robust than the Newton-Krylov method as the test problems become increasingly harder to solve. These results are consistent with those obtained in [1]. We anticipate that similar conclusions would be obtained on other test problems as well. Therefore, our recommendation is to use tensor-Krylov methods for difficult problems to solve, and Newton-Krylov methods otherwise.

7. Conclusions

We have developed tensor methods for large systems of nonlinear equations based on Krylov subspace projection methods for approximately solving the linear systems that are required. The numerical test results show that tensor-Krylov methods are much more efficient and robust than Newton-Krylov methods on nonlinear equations problems that are difficult to solve.

In order to firmly establish the conclusion above, additional testing is required, including rank-deficient test problems. We expect the efficiency advantage of tensor-Krylov methods to be much larger on such problems [1]. Several issues remain to be investigated. Among them is the possibility of combining tensor-Krylov and Newton-Krylov methods with the two-dimensional trust region global strategy developed in [2]. This global approach was shown to be much more robust than line search methods [2]. Thus, its integration with tensor-Krylov methods is expected to make these methods even more robust. Finally, we intend to implement the algorithms discussed in this paper in a software package.

Table 3: Test results for problem 3

			Newton-GMRES				Tensor-GMRES			
R_a	K_{dim}	*precond*	*itns*	*fevals*	*nli*	*nb*	*itns*	*fevals*	*nli*	*nb*
1.0e+3	10	mg	9	14	74	4	7	12	95	3
1.0e+4	10	mg	17	30	215	12	13	40	270	19
1.0e+5	15	mg	35	81	482	45	27	121	615	73
1.0e+6	25	mg	F	–	–	–	90	555	3867	382

Acknowledgments. I thank Bobby Schnabel for initiating this research and for his numerous suggestions. I am grateful to Ray Tuminaro for discussing several issues related to Krylov methods and their preconditioning. I also thank the referees, and Jorge Moré and Gail Pieper from the Mathematics and Computer Science Division at Argonne National Laboratory for criticism that helped me improve the paper.

References

[1] 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, Computer Science Department, University of Colorado at Boulder, 1992.

[2] A. Bouaricha and R. B. Schnabel. TENSOLVE: A software package for solving systems of nonlinear equations and nonlinear least squares problems using tensor methods. Preprint MCS-P463-0894, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

[3] A. Bouaricha and R. B. Schnabel. Tensor methods for large, sparse systems of nonlinear equations. Preprint MCS-P473-1094, Mathematics and Computer Science Division, Argonne National Laboratory, 1994.

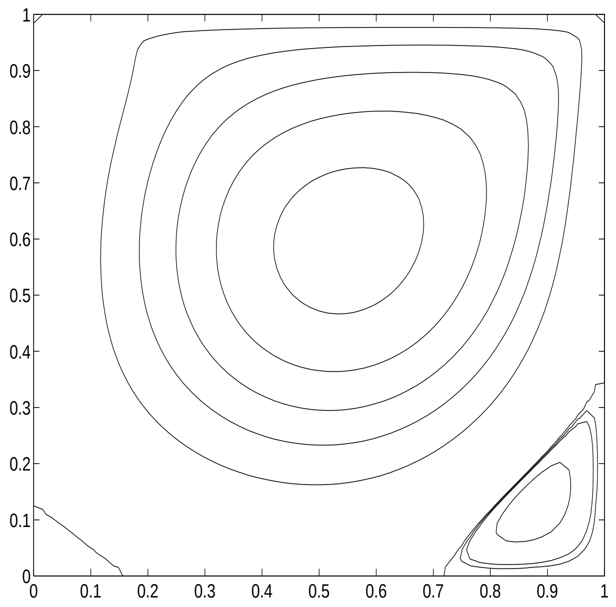

Figure 1: Streamlines for Reynolds number 500

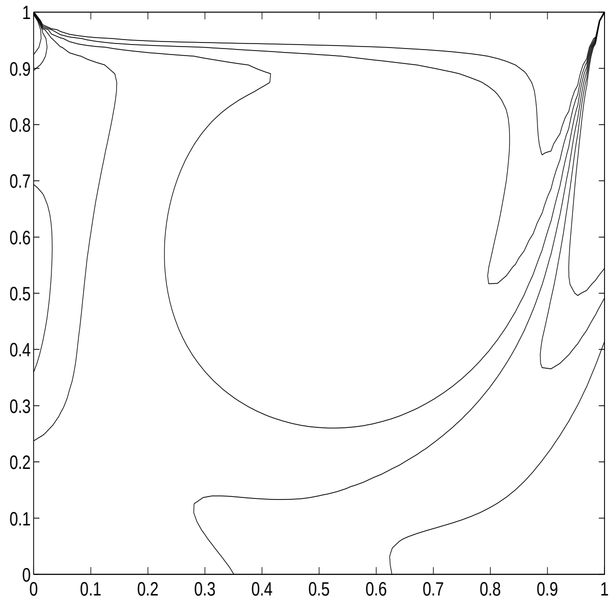

Figure 2: Equivorticity for Reynolds number 500

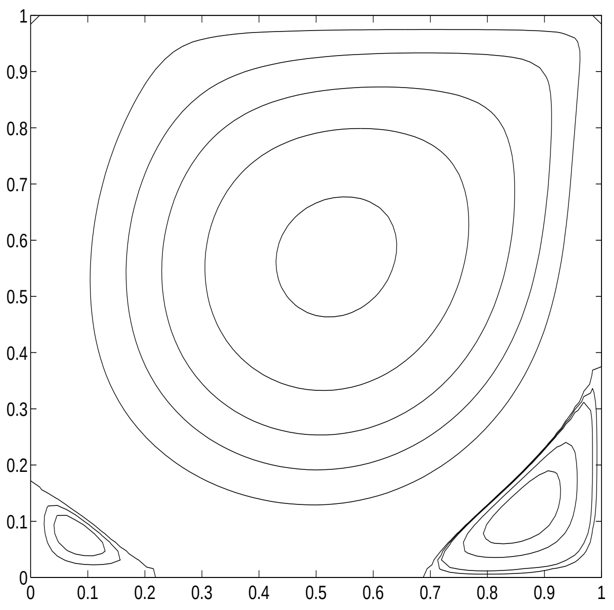

Figure 3: Streamlines for Reynolds number 1000

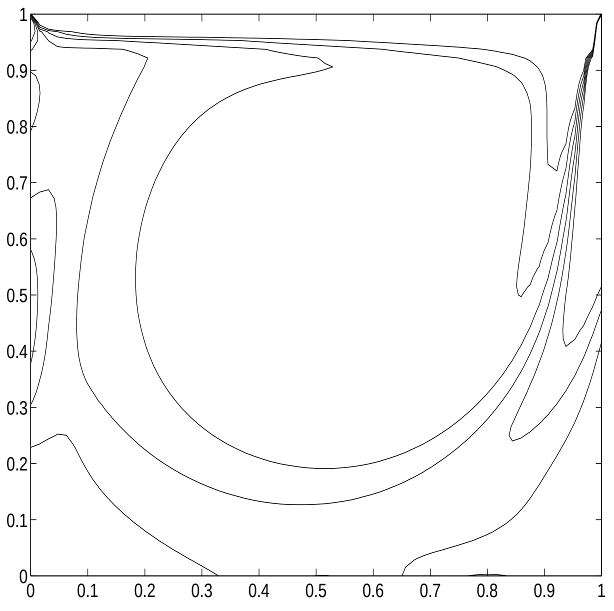

Figure 4: Equivorticity lines for Reynolds number 1000

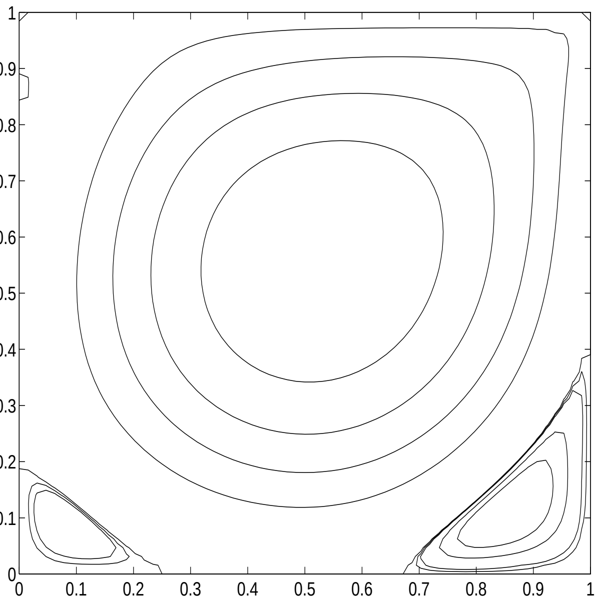

Figure 5: Streamlines for Reynolds number 1500

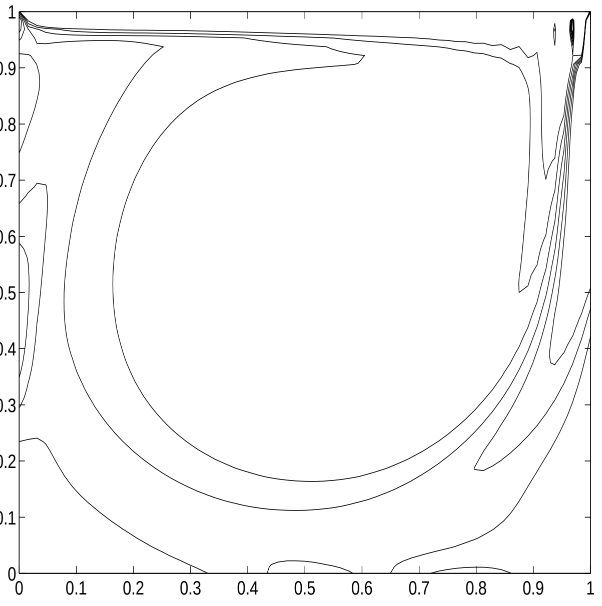

Figure 6: Equivorticity for Reynolds number 1500

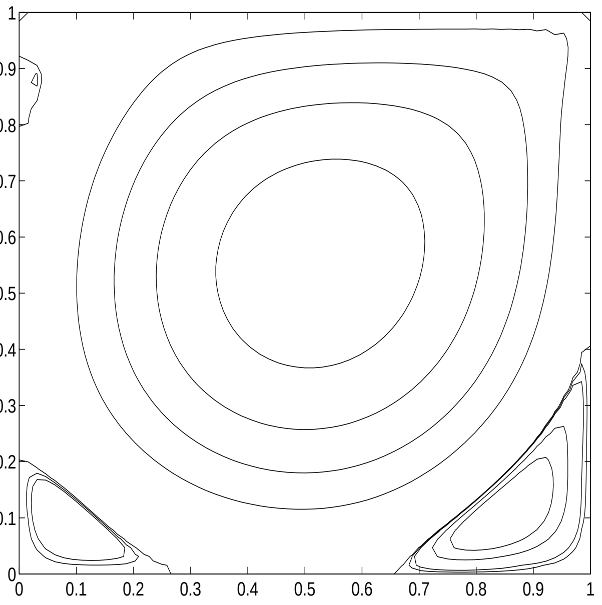

Figure 7: Streamlines for Reynolds number 2000

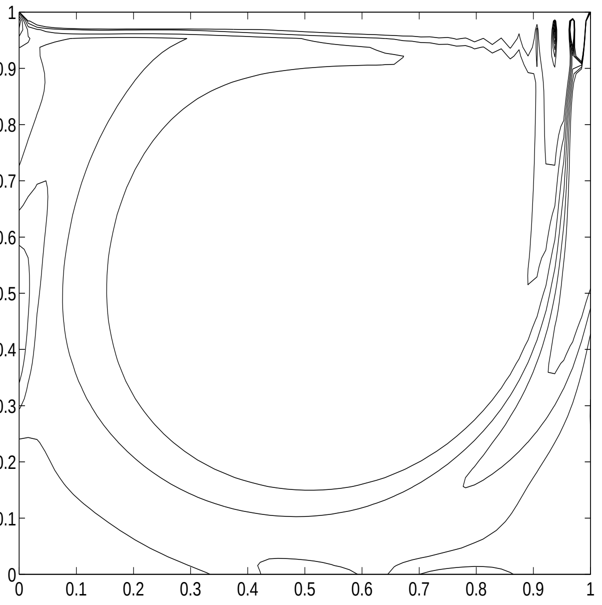

Figure 8: Equivorticity lines for Reynolds number 2000

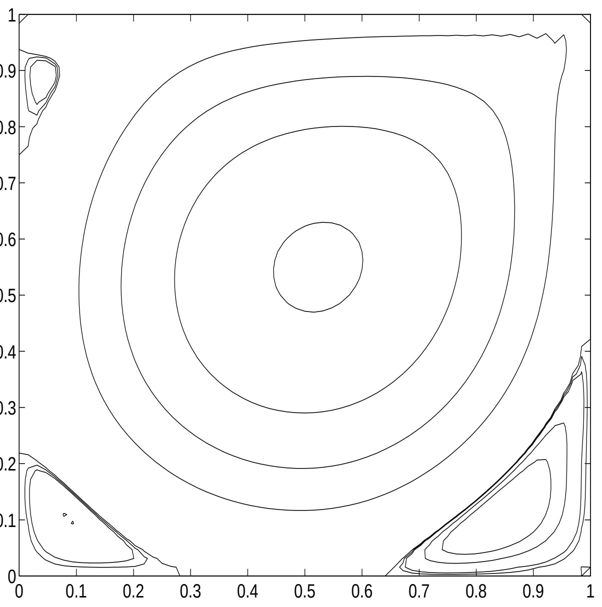

Figure 9: Streamlines for Reynolds number 3000

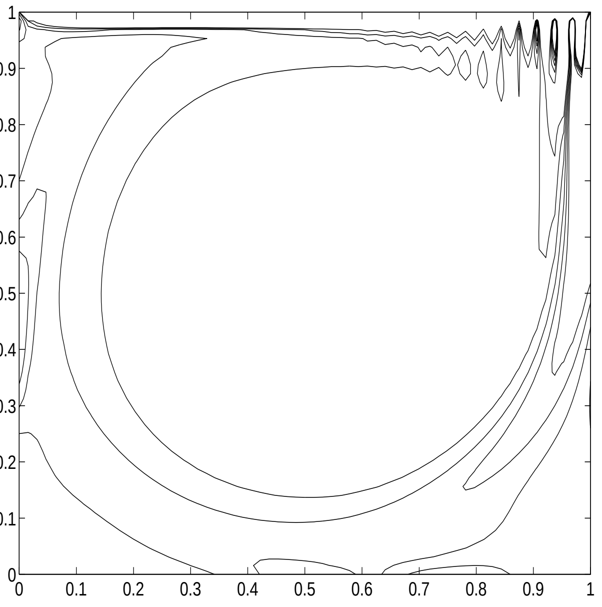

Figure 10: Equivorticity lines for Reynolds number 3000

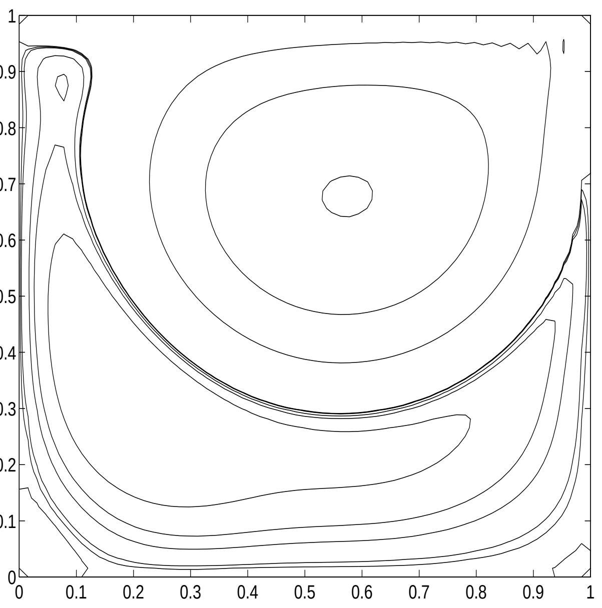

Figure 11: Streamlines for Reynolds number 5000

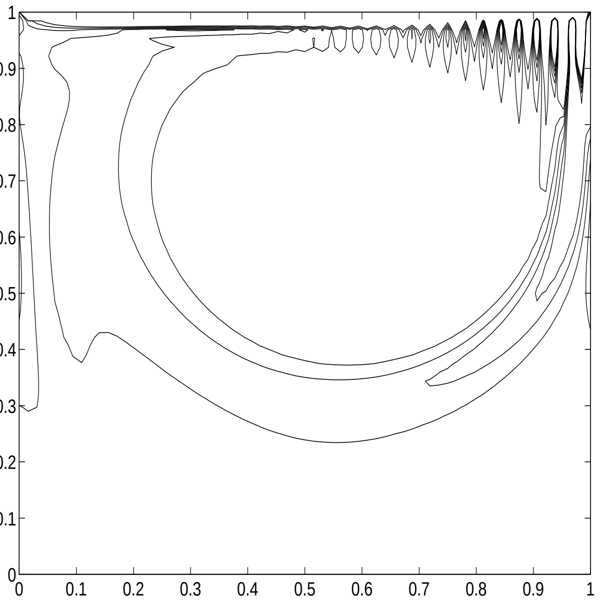

Figure 12: Equivorticity lines for Reynolds number 5000

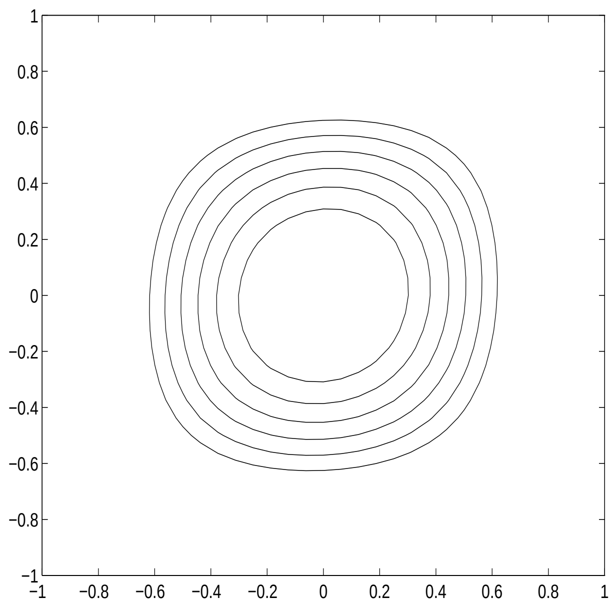

Figure 13: Streamlines for Raleigh = 1000

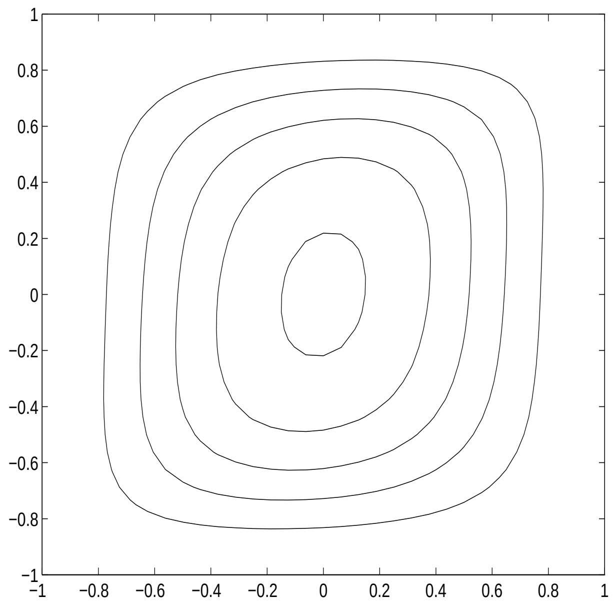

Figure 14: Streamlines for Raleigh = 10,000

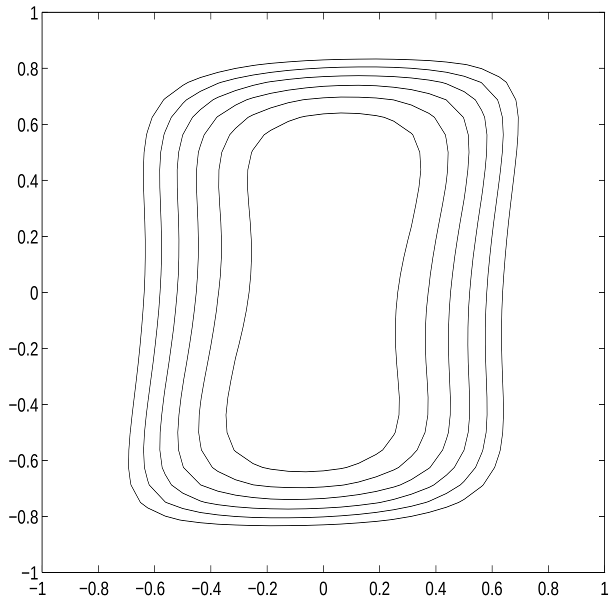

Figure 15: Streamlines for Raleigh = 100,000

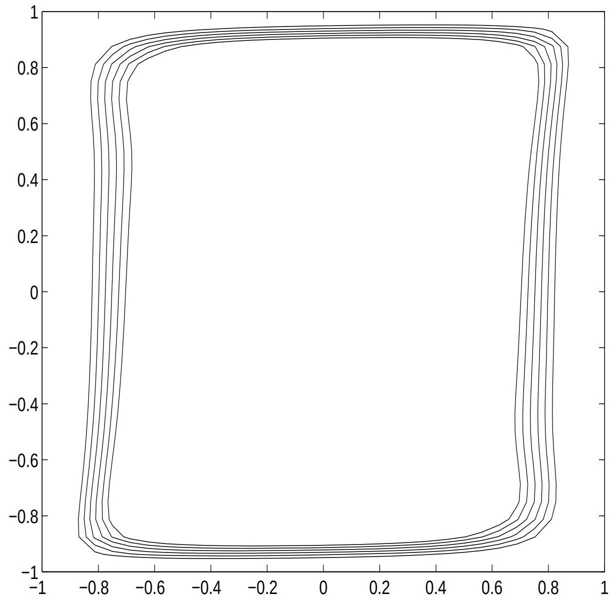

Figure 16: Streamlines for Raleigh = 1,000,000

[4] P. N. Brown and Y. Saad. Globally convergent techniques in nonlinear Newton-Krylov algorithms. Technical Report L-316, Computing and Mathematics Research Division, Lawrence Livermore National Laboratory, 1989.

[5] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Stat. Comp.*, 11:450–481, 1990.

[6] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, N.J., 1983.

[7] D. Feng, P. Frank, and R. B. Schnabel. Local convergence analysis of tensor methods for nonlinear equations. *Math. Prog.*, 62:427–459, 1993.

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

[9] Y. Saad and M. H. Schultz. Conjugate gradient-like algorithms for solving nonsymmetric linear systems. *Math. Comp.*, 44:417–424, 1985.

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

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

[12] R. B. Schnabel, J. E. Koontz, and B. E. Weiss. A modular system of algorithms of unconstrained minimization. *ACM Trans. Math. Softw.*, 11:419–440, 1985.

[13] J. N. Shadid and R. S. Tuminaro. Sparse iterative algorithm software for large-scale MIMD machines: An initial discussion and implementation. Technical Report Sand91-0059, Sandia National Laboratories, Albuquerque, N.M., 1991.

[14] V. Simonsini and E. Gallopoulos. Convergence properties of block GMRES for solving systems with multiple right-hand sides. Technical Report 1316, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, 1992.

[15] V. Simonsini and E. Gallopoulos. An iterative method for nonsymmetric systems with multiple right hand-hand sides. *SIAM J. Sci. Stat. Comput.*, 16:917–933, 1995.

[16] P. Sonneveld. CGS: A fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 10:36–52, 1989.

[17] H. Van Der Vost. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.