SIAM REVIEW
Vol. 48, No. 4, pp. 700–721

Globalization Techniques for Newton–Krylov Methods and Applications to the Fully Coupled Solution of the Navier–Stokes Equations*

Roger P. Pawlowski†
John N. Shadid†
Joseph P. Simonis‡
Homer F. Walker‡

Abstract. A Newton–Krylov method is an implementation of Newton's method in which a Krylov subspace method is used to solve approximately the linear subproblems that determine Newton steps. To enhance robustness when good initial approximate solutions are not available, these methods are usually *globalized*, i.e., augmented with auxiliary procedures (*globalizations*) that improve the likelihood of convergence from a starting point that is not near a solution. In recent years, globalized Newton–Krylov methods have been used increasingly for the fully coupled solution of large-scale problems. In this paper, we review several representative globalizations, discuss their properties, and report on a numerical study aimed at evaluating their relative merits on large-scale two- and three-dimensional problems involving the steady-state Navier–Stokes equations.

Key words. Newton's method, inexact Newton methods, Newton iterative methods, Newton–Krylov methods, globalized Newton methods, backtracking, line search, trust-region methods, dogleg methods, fully coupled solution methods, Navier–Stokes equations

AMS subject classifications. 65H10, 65F10

DOI. 10.1137/S0036144504443511

1. Introduction.

Krylov subspace methods constitute a broad and widely used class of iterative linear algebra methods that includes, most notably, the classical conjugate gradient method for symmetric positive-definite systems [23] and more re-

*Received by the editors April 28, 2004; accepted for publication (in revised form) December 5, 2005; published electronically November 2, 2006.
http://www.siam.org/journals/sirev/48-4/44351.html

†Sandia National Laboratories, MS 0316, P.O. Box 5800, Albuquerque, NM 87185-0316 (rppawlo@sandia.gov, jnshadi@sandia.gov). The work of these authors was partially supported by the U.S. Department of Energy ASCI program and the U.S. Department of Energy Office of Science MICS program at Sandia National Laboratories under contract DE-AC04-94AL85000.

‡Department of Mathematical Sciences, Worcester Polytechnic Institute, Worcester, MA 01609-2280 (jpsimoni@wpi.edu, walker@wpi.edu). The work of these authors was partially supported by Sandia National Laboratories under the ASCI program and by the Sandia National Laboratories Computer Science Research Institute (contract 16099 with WPI). The work of the fourth author was also supported in part by the Center for Simulation of Accidental Fires and Explosions funded at the University of Utah by the U.S. Department of Energy under contracts LLNL B341493 and B524196.

cently developed methods for nonsymmetric linear systems such as GMRES[1] [40], which are of particular interest here, and also Bi-CGSTAB [51], TFQMR [15], and related methods. An extensive discussion of these methods is beyond the scope of this work; we refer the reader to the surveys [16] and [20] and the books [19], [39], and [52].

A *Newton–Krylov method* (see, e.g., [3], [26], [28]) is an implementation of Newton's method in which a Krylov subspace method is used to solve approximately the linear systems that characterize steps of Newton's method. Specifically, if we seek a zero of a residual function $F : \mathbb{R}^n \to \mathbb{R}^n$ and if $u \in \mathbb{R}^n$ is a current approximate solution, then a Krylov subspace method is applied to the *Newton equation*

$$F'(u)s = -F(u), \tag{1.1}$$

where $F'(u) \in \mathbb{R}^{n\times n}$ is the Jacobian (matrix) of F at u. A Newton–Krylov method that uses a specific Krylov subspace method is often designated by appending the name of the method to "Newton," as in "Newton-GMRES" or "Newton-BiCGSTAB." (The term "truncated Newton method" is also widely used when the Krylov subspace method is the conjugate gradient method; cf. [10] and [32].) Krylov subspace methods have special advantages in the solution of (1.1). In particular, most of these methods, including those named above, require only products of $F'(u)$ with vectors[2] and thus allow "matrix-free" Newton–Krylov implementations, in which these products are evaluated or approximated without creating or storing $F'(u)$. (See, e.g., [28].)

A Newton–Krylov method is usually implemented as an *inexact Newton method* [9], the basic form of which is as follows.

ALGORITHM IN. INEXACT NEWTON METHOD [9].

Let u_0 be given.
For $k = 0, 1, \ldots$ (until convergence) do:
Choose $\eta_k \in [0, 1)$ and s_k such that

$$\|F(u_k) + F'(u_k)\, s_k\| \le \eta_k \|F(u_k)\|. \tag{1.2}$$

Set $u_{k+1} = u_k + s_k$.

In the Newton–Krylov context, one chooses for each k a *forcing term* $\eta_k \in [0, 1)$ (cf. [13]) and then applies the Krylov subspace method until an iterate s_k satisfies the *inexact Newton condition* (1.2). The forcing terms determine the local convergence properties of the method: by choosing $\{\eta_k\}$ appropriately, one can achieve desirably fast rates of local convergence, up to the rate of exact Newton's method (typically quadratic) [9]. Additionally, by reducing the likelihood of *oversolving*, i.e., obtaining unproductive accuracy in approximately solving (1.1), well-chosen forcing terms may significantly improve the efficiency of the method and, in some applications, the robustness as well ([13], [47], [50]).

Newton–Krylov methods, like all Newton-like methods, must usually be *globalized*, i.e., augmented with certain auxiliary procedures (*globalizations*) that increase the likelihood of convergence when good initial approximate solutions are not available. Globalizations are typically structured to test whether a step gives satisfactory progress toward a solution and, if necessary, to modify it to obtain a step that does give satisfactory progress. There are two major categories of globalizations:[3] *backtracking* (*line-search*, *damping*) methods, in which step lengths are adjusted (usually

[1] For convenience in what follows, we do not usually distinguish between GMRES and its restarted version GMRES(m).

[2] Some Krylov subspace methods require products of $F'(u)^T$ as well.

[3] See [11, Chap. 6] for a general discussion of classical globalizations.

shortened) to obtain satisfactory steps; and *trust-region* methods, in which a step from an approximate solution u is ideally chosen to minimize the norm of $F(u)+F'(u)s$, the *local linear model* of F, within a specified "trust region." (More specifically, the trust-region step is ideally $\arg\min_{\|s\|\leq\delta}\|F(u) + F'(u)s\|$, where $\delta > 0$ is the *trust-region radius*.) Both backtracking and trust-region methods have strong theoretical support; see, e.g., [11] and [12]. Backtracking methods are relatively easy to implement; however, each step direction is restricted to be that of the initial trial step, which may be a weak descent direction, especially if the Jacobian is ill-conditioned [50]. Trust-region methods have the potential advantage of producing modified steps that may be stronger descent directions than the initial trial step; however, their implementation may be problematic. In general, it is not feasible to compute the ideal trust-region step accurately, and popular ways of approximating this step require products of the transpose of the Jacobian with vectors. These products may be difficult or impractical to compute in some applications, especially in the Newton–Krylov context, in which the Jacobian may not be known. Additionally, a step produced by a Newton–Krylov method (or any iterative method) may not be well suited for use in these popular approaches unless it solves (1.1) fairly accurately, and the necessary accuracy may be difficult to determine a priori. We comment further on these issues in section 2.4.

The purpose of this paper is to review several representative globalizations of Newton–Krylov methods, discuss their properties, and report on extensive numerical experiments with Newton-GMRES implementations that demonstrate their relative merits in large-scale applications involving the steady-state Navier–Stokes equations. Our main goal is to provide an accessible introduction to the methods of interest and a thorough study of their performance on an important class of large-scale problems. We also offer pointers to publicly available, downloadable software implementations used in our tests and report new experimental data on the numerical solution of several three-dimensional benchmark problems. This work is meant to be a cohesive study of a representative variety of practically effective techniques rather than an exhaustive survey. We do not cover other robust solution techniques such as homotopy, continuation, pseudotransient continuation, or mesh sequencing methods but refer the reader to [54], [55], [1], [30], [6], [27], [28], and the references in those works.

In an earlier study [47], we considered a backtracking globalization from [12] and showed in experiments that it significantly improves the robustness of a Newton-GMRES method on the applications of interest here, especially when combined with adaptively determined forcing terms from [13]. Here, we extend that study to include the backtracking globalization of [12] and [47], a certain refinement of that globalization, a line-search procedure from [31], and a *dogleg* trust-region implementation [37], [11]. This dogleg implementation is feasible because our testing environment allows evaluation of products of the transpose of the Jacobian with vectors. Further aspects of the implementation and associated issues are discussed in section 2.4.

The test problems are two- and three-dimensional (2D, 3D) versions of three benchmark flow problems, viz., the thermal convection, backward-facing step, and lid-driven cavity problems. These problems are all large scale, with between 25,263 and 1,042,236 equations and unknowns in our tests; consequently, all numerical experiments were necessarily run on parallel computers. An important aspect of this study is describing the algorithmic features that were used, beyond the basic globalized Newton–Krylov methods, to make the implementations effective on these platforms in the problem regimes of interest.

In section 2 below, we discuss the numerical methods of interest and outline their theoretical support. In section 3, we introduce the governing PDEs and the

discretized equations. In section 4, the test problems and the computing environment are described. We summarize the test results in section 5, comment further on failure and robustness in section 6, and draw conclusions in section 7. For the proofs of some of the theoretical results in section 2, the reader is referred to [34], which also contains complete details of the test results.

2. The Numerical Methods. An important note is that the discussion of algorithms and theoretical results in this section is valid in the general inexact Newton setting and is not restricted to the Newton–Krylov context. In sections 2.1–2.2, the norm $\|\cdot\|$ is arbitrary. In sections 2.3–2.4, it is assumed to be an inner-product norm but is otherwise arbitrary.

2.1. The Forcing Terms. Although the focus of this study is on globalization procedures, previous studies (see [13], [47], [50]) have shown that the forcing terms may affect the robustness of a Newton–Krylov method, globalization notwithstanding. Accordingly, we consider two choices of the forcing terms here to assess their effects on the globalizations of interest. The first is a small constant, $\eta_k = 10^{-4}$ for each k, which should produce close approximations of exact Newton steps. The second is an adaptive forcing term, called "Choice 1" in [13] and determined as follows: Select $\eta_0 \in [0, 1)$ and set

$$\eta_k = \frac{\Big|\|F(u_k)\| - \|F(u_{k-1}) + F'(u_{k-1})\, s_{k-1}\|\Big|}{\|F(u_{k-1})\|}, \qquad k = 1, 2, \ldots. \tag{2.1}$$

In keeping with practice elsewhere (see [13], [47], [36]), we follow (2.1) with the safeguard

$$\eta_k \leftarrow \min\Big\{\eta_{\max}, \max\{\eta_k, \eta_{k-1}^{(1+\sqrt{5})/2}\}\Big\} \text{ whenever } \eta_{k-1}^{(1+\sqrt{5})/2} > 0.1, \tag{2.2}$$

which is intended to prevent the forcing terms from becoming too small too quickly away from a solution and also to keep them below a prescribed $\eta_{\max} \in [0, 1)$. (In our implementations, we used $\eta_0 = .01$ and $\eta_{\max} = .9$.) The exponent $(1 + \sqrt{5})/2$ is related to a local convergence rate associated with these forcing terms; see the remark following Theorem 2.3 below.

To state briefly the local convergence properties of Algorithm IN with these two choices of the forcing terms, we formulate the following assumption:

ASSUMPTION 2.1.

(a) *$F : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable in a neighborhood of $u_* \in \mathbb{R}^n$ such that $F(u_*) = 0$ and $F'(u_*)$ is nonsingular.*

(b) *F is Lipschitz continuously differentiable at u_*, i.e., there is a constant Γ for which $\|F'(u) - F'(u_*)\| \le \Gamma\|u - u_*\|$ for all u sufficiently near u_*.*

With this assumption, we have the results below from [9] and [13].

THEOREM 2.2 (see [9, Thm. 2.3]). *Suppose that Assumption* 2.1(a) *holds, and let $\{u_k\}$ be a sequence produced by Algorithm* IN *with $0 \le \eta_k \le \eta_* < 1$ for each k. If u_0 is sufficiently near u_*, then $\{u_k\}$ converges to u_* and, provided $u_k \ne u_*$ for all k,*

$$\limsup_{k\to\infty} \|u_{k+1} - u_*\|_* / \|u_k - u_*\|_* \le \eta_*, \tag{2.3}$$

where $\|v\|_* \equiv \|F'(u_*)v\|$ for each $v \in \mathbb{R}^n$.

It follows that if $\eta_k = 10^{-4}$ for each k, then, under Assumption 2.1(a), Algorithm IN exhibits fast local q-linear convergence[4] to a solution u_*; specifically, (2.3) holds with $\eta_* = 10^{-4}$.

THEOREM 2.3 (see [13, Thm. 2.2]). *Suppose that Assumption* 2.1 *holds, and let* $\{u_k\}$ *be a sequence produced by Algorithm* IN *with each* η_k *given by* (2.1). *If* u_0 *is sufficiently near* u_*, *then* $\{u_k\}$ *converges to* u_* *with*

$$\|u_{k+1} - u_*\| \le \gamma \|u_k - u_*\| \|u_{k-1} - u_*\|, \qquad k = 1, 2, \ldots, \tag{2.4}$$

for a constant γ *independent of* k.

As observed in [13], it follows from (2.4) that the convergence is q-superlinear, two-step q-quadratic, and of r-order $(1+\sqrt{5})/2$. Also, the conclusions of the theorem still hold if each η_k is determined by (2.1) followed by the safeguard (2.2).

2.2. The Backtracking Methods. We consider the following general backtracking method from [12].

ALGORITHM INB. INEXACT NEWTON BACKTRACKING METHOD [12].

Let u_0, $\eta_{\max} \in [0, 1)$, $t \in (0, 1)$, *and* $0 < \theta_{\min} < \theta_{\max} < 1$ *be given.*
For $k = 0, 1, \ldots$ *(until convergence) do:*
  Choose initial $\eta_k \in [0, \eta_{\max}]$ *and* s_k *such that*
    $$\|F(u_k) + F'(u_k)\, s_k\| \le \eta_k \|F(u_k)\|.$$
  While $\|F(u_k + s_k)\| > [1 - t(1 - \eta_k)]\, \|F(u_k)\|$ *do:*
    Choose $\theta \in [\theta_{\min}, \theta_{\max}]$.
    Update $s_k \leftarrow \theta s_k$ *and* $\eta_k \leftarrow 1 - \theta(1 - \eta_k)$.
  Set $u_{k+1} = u_k + s_k$.

In Algorithm INB, the backtracking globalization resides in the while-loop, in which steps are tested and shortened as necessary until the acceptability condition

$$\|F(u_k + s_k)\| \le [1 - t(1 - \eta_k)]\, \|F(u_k)\| \tag{2.5}$$

holds. As noted in [12], if F is continuously differentiable, then this globalization produces a step for which (2.5) holds after a finite number of passes through the while-loop; furthermore, the inexact Newton condition (1.2) still holds for the final s_k and η_k. The condition (2.5) is a "sufficient-decrease" condition on $\|F(u_k + s_k)\|$. To illuminate it further, we follow [12] and [47] and define

$$\begin{aligned} ared_k &\equiv \|F(u_k)\| - \|F(u_k + s_k)\|, \\ pred_k &\equiv \|F(u_k)\| - \|F(u_k) + F'(u_k)\, s_k\|; \end{aligned} \tag{2.6}$$

these are, respectively, the *actual reduction* in $\|F\|$ and the *predicted reduction* given by the local linear model. It is easily verified that (1.2) is equivalent to $pred_k \ge (1 - \eta_k)\|F(u_k)\|$ and (2.5) is equivalent to $ared_k \ge t(1 - \eta_k)\|F(u_k)\|$. Thus, if (1.2) requires the predicted reduction to be at least $(1-\eta_k)\|F(u_k)\|$, then (2.5) requires the actual reduction to be at least the fraction t of that amount. In our implementation, we used $t = 10^{-4}$ so that, consistent with recommendations in [11], only a very modest decrease in $\|F\|$ is required for a step to be accepted.

Theoretical support for Algorithm INB is provided in [12] by the following result.

[4] For definitions of the kinds of convergence referred to here, see [11].

THEOREM 2.4 (see [12, Thm. 6.1]). *Assume that F is continuously differentiable. If $\{u_k\}$ produced by Algorithm* INB *has a limit point u_* such that $F'(u_*)$ is nonsingular, then $F(u_*) = 0$ and $u_k \to u_*$. Furthermore, the initial s_k and η_k are accepted without modification in the while-loop for all sufficiently large k.*

A consequence of the theorem is that if $\{u_k\}$ converges to a solution u_* such that $F'(u_*)$ is nonsingular, then, under Assumption 2.1, the convergence is ultimately governed by the initial η_k's. In particular, if $\eta_k = 10^{-4}$ for each k, then (2.3) holds with $\eta_* = 10^{-4}$, and if each η_k is given by (2.1) followed by (2.2), then an inequality (2.4) holds for sufficiently large k and a γ independent of k.

Restricting each step-length reduction factor θ to lie in $[\theta_{\min}, \theta_{\max}]$ is known as *safeguarded backtracking*. In our implementation, we used the common choices $\theta_{\min} = \frac{1}{10}$ and $\theta_{\max} = \frac{1}{2}$ (cf. [11]). We also followed standard practice in choosing $\theta \in [\theta_{\min}, \theta_{\max}]$ to minimize a low-degree polynomial that interpolates values of $\|F\|$. Specifically, in this study, we tested two possibilities. The first is that used in [47], viz., to determine $\theta \in [\theta_{\min}, \theta_{\max}]$ to minimize a quadratic polynomial $p(t)$ that satisfies $p(0) = \frac{1}{2}\|F(u_k)\|^2$, $p(1) = \frac{1}{2}\|F(u_k + s_k)\|^2$, and $p'(0) = \frac{d}{dt}\frac{1}{2}\|F(u_k + ts_k)\|^2\big|_{t=0}$. The second is a refinement of this idea from [11], as follows: On the first step-length reduction, θ is chosen to minimize an interpolating quadratic polynomial as before. On subsequent reductions, θ is chosen to minimize over $[\theta_{\min}, \theta_{\max}]$ a *cubic* polynomial $p(t)$ for which $p(0)$, $p(1)$, and $p'(0)$ have the same values as before and additionally $p(\theta_{\rm prev}^{-1}) = \frac{1}{2}\|F(u_k + \theta_{\rm prev}^{-1} s_k)\|^2$, where $\theta_{\rm prev}$ is the step-length reduction factor used in the previous reduction and $\|F(u_k + \theta_{\rm prev}^{-1} s_k)\|$ has been retained from that reduction. Formulas for the minimizers of these polynomials are given in [11, Chap. 6].

2.3. The Moré–Thuente Line-Search Method. This line-search procedure from [31] is intended for the unconstrained minimization of a general functional $f : \mathbb{R}^n \to \mathbb{R}^1$ and is adapted to the present setting by taking $f(u) \equiv \frac{1}{2}\|F(u)\|^2$. It differs from the backtracking globalizations of section 2.2 primarily in being more directly focused on approximately minimizing $\|F\|$ in a given search direction and by allowing steps that are longer than the initial trial step if such steps seem warranted by potential further decrease in $\|F\|$. To set the context, we embed it within an inexact Newton method as follows.

ALGORITHM INMTL. INEXACT NEWTON MORÉ–THUENTE LINE-SEARCH METHOD.

Let u_0 and $\eta_{\max} \in [0, 1)$ be given.
For $k = 0, 1, \ldots$ (until convergence) do:
  Choose $\eta_k \in [0, \eta_{\max}]$ and initial s_k such that

$$\|F(u_k) + F'(u_k)\, s_k\| \le \eta_k \|F(u_k)\|.$$

  Apply the Moré–Thuente line search [31] *to determine a final s_k.*
  Set $u_{k+1} = u_k + s_k$.

To describe the Moré–Thuente line search, we define for a particular k

$$\phi(\lambda) \equiv f(u_k + \lambda s_k) = \frac{1}{2}\|F(u_k + \lambda s_k)\|^2. \tag{2.7}$$

We assume for this discussion that F is continuously differentiable, which implies that f and ϕ are as well. Since the initial s_k is an inexact Newton step from u_k, it is also a *descent direction* for $\|\cdot\|$ and f in that $\phi'(0) < 0$ (see [4, Prop. 3.3], [12, Lem. 7.1]).

The goal of the line search is to find a $\lambda > 0$ satisfying the two inequalities

$$\phi(\lambda) \leq \phi(0) + \alpha\phi'(0)\lambda, \tag{2.8}$$

$$|\phi'(\lambda)| \leq \beta|\phi'(0)|, \tag{2.9}$$

where α and β are given parameters in $(0, 1)$. Once such a λ has been determined by the line search, the initial s_k is updated by $s_k \leftarrow \lambda s_k$ to determine the final s_k. Inequalities (2.8) and (2.9) are sometimes known as the *(strong) Wolfe conditions* [31, Chap. 3]. (The weak Wolfe conditions, sometimes known as the *Goldstein–Armijo conditions* [11, Chap. 6], consist of (2.8) and the inequality $\phi'(\lambda) \geq \beta\phi'(0)$.) Inequality (2.8) is a sufficient-decrease condition on $\|F(u_k + \lambda s_k)\|$; cf. (2.5). Since $\phi'(0) < 0$ and $0 < \alpha < 1$, (2.8) holds for sufficiently small $\lambda > 0$. Inequality (2.9) does *not* hold for small $\lambda > 0$, and its primary function is to prevent steps from being too short to give adequate progress toward a solution. Additionally, (2.9) is sometimes called a *curvature condition* since it implies $\phi'(\lambda) - \phi'(0) \geq (1 - \beta)|\phi'(0)| > 0$, from which it follows that the average curvature of ϕ on $(0, \lambda)$ is positive [31]. Such conditions are used in optimization to ensure that certain quasi-Newton updates inherit positive-definiteness (see, e.g., [33]).

In our context, $\phi(\lambda) \geq 0$ for all λ, and one easily verifies that there is at least one $\lambda > 0$ that satisfies both (2.8) and (2.9) provided $\beta \geq \alpha$, which is usually the case in practice. (In our implementation, we used the typical values $\alpha = 10^{-4}$ and $\beta = .9999$.) If $\beta < \alpha$, then there may be no $\lambda > 0$ that satisfies both (2.8) and (2.9). However, if the set of λ satisfying (2.8) contains a local minimizer of ϕ, then this set contains solutions of (2.9) as well, and small values of β may serve to restrict solutions of (2.8)–(2.9) to be near this minimizer.

The line search generates iterates that lie within "intervals of uncertainty" and are additionally constrained to be within $[\lambda_{\min}, \lambda_{\max}]$ for specified $0 < \lambda_{\min} < \lambda_{\max}$. In typical practice, fixed $\lambda_{\min}$ and $\lambda_{\max}$ are used for all k. (We used $\lambda_{\min} = 10^{-12}$ and $\lambda_{\max} = 10^6$ in our implementation.) The procedures for updating the intervals of uncertainty and determining successive line-search iterates within them are complex and are explained in detail in [31] (see also [34]). The possible outcomes of the line search, as explained in [31], are as follows:

1. The iterates increase monotonically and reach $\lambda_{\max}$ after a finite number of iterations; with $\lambda = \lambda_{\max}$, (2.8) holds but (2.9) may not hold.
2. The iterates decrease monotonically and reach $\lambda_{\min}$ after a finite number of iterations; neither (2.8) nor (2.9) is guaranteed to hold with $\lambda = \lambda_{\min}$.
3. A value of $\lambda \in (\lambda_{\min}, \lambda_{\max})$ is reached for which (2.8) holds and (2.9) also holds with $\beta = \alpha$.
4. A value of $\lambda \in (\lambda_{\min}, \lambda_{\max})$ is reached for which both (2.8) and (2.9) hold.

Note that if one of the first two outcomes occurs, then both (2.8) and (2.9) *may* hold, but this is not guaranteed. Note also that if (2.9) holds with $\beta = \alpha$, then it also holds with $\beta \geq \alpha$. Thus if $\beta \geq \alpha$, then the third and fourth outcomes become one.

Our global convergence result for Algorithm INMTL is Theorem 2.5 below. The proof is given in [34, App. 1].

THEOREM 2.5. *Suppose that u_0 is given and that F is Lipschitz continuously differentiable on $\mathcal{L}(u_0) \equiv \{u : \|F(u)\| \leq \|F(u_0)\|\}$, i.e., F is differentiable on $\mathcal{L}(u_0)$ and there is a $\Gamma \geq 0$ such that $\|F'(v) - F'(u)\| \leq \Gamma\|v - u\|$ for all $u \in \mathcal{L}(u_0)$ and nearby $v \in \mathcal{L}(u_0)$. Assume that $\{u_k\}$ is produced by Algorithm* INMTL *and, for each k, the λ determined by the Moré–Thuente line search satisfies* (2.8) *and* (2.9). *If $\{u_k\}$ has a subsequence $\{u_{k_j}\}$ such that $F'(u_{k_j})$ is nonsingular for each j and $\{\|F'(u_{k_j})^{-1}\|\}$ is*

bounded, then $F(u_k) \to 0$. *If* $\{u_k\}$ *has a limit point* u_* *such that* $F'(u_*)$ *is nonsingular, then* $F(u_*) = 0$ *and* $u_k \to u_*$.

In contrast to Theorem 2.4 above and Theorem 2.7 below, Theorem 2.5 provides no assurance that initial inexact Newton steps are ultimately accepted without modification as the iterates near a solution. Indeed, such an assurance cannot be made without further assumptions. For example, one must require $\alpha < 1/2$ to ensure that exact Newton steps are acceptable without modification near a solution (see, e.g., [11, Thm. 6.3.4]). The following lemma generalizes this observation to the inexact Newton context; see [34, App. 1] for a proof.

LEMMA 2.6. *Suppose that* $\{u_k\}$ *produced by Algorithm* INMTL *converges to* u_* *for which Assumption* 2.1(a) *holds. Then* (2.8) *holds with* $\lambda = 1$ *for all sufficiently large* k *if*

$$\alpha < (1 - \limsup_{k\to\infty} \eta_k)/2 \tag{2.10}$$

and only if

$$\alpha \le 1/[2(1 - \liminf_{k\to\infty} \eta_k)] \,. \tag{2.11}$$

Additionally, (2.9) *holds with* $\lambda = 1$ *for all sufficiently large* k *if*

$$\beta > \limsup_{k\to\infty} \eta_k (1 + \limsup_{k\to\infty} \eta_k)/(1 - \limsup_{k\to\infty} \eta_k) \,. \tag{2.12}$$

If something is known about $\limsup_{k\to\infty} \eta_k$, then (2.10) and (2.12) may provide useful guidance in specifying α and β. However, if $\limsup_{k\to\infty} \eta_k \ge \sqrt{2} - 1$, then the bound on the right-hand side of (2.12) is not less than 1, and (2.12) is not helpful.

Lemma 2.6 can be used to advantage with the forcing terms considered here. With the adaptive Choice 1 forcing terms, it is easy to show that $\limsup_{k\to\infty} \eta_k = 0$ under the assumptions of the lemma. Thus, in this case, the lemma implies that, for any $\alpha \in (0, \frac{1}{2})$ and $\beta \in (0, 1)$, initial inexact Newton steps are ultimately accepted without modification and an inequality (2.4) holds for sufficiently large k and a γ independent of k. With $\eta_k = 10^{-4}$ for each k, it follows from the lemma that, under the scarcely more restrictive conditions $0 < \alpha < (1 - 10^{-4})/2$ and $10^{-4}(1 + 10^{-4})/(1 - 10^{-4}) < \beta < 1$, initial inexact Newton steps are again ultimately accepted and the convergence obeys (2.3) with $\eta_* = 10^{-4}$. The values $\alpha = 10^{-4}$ and $\beta = .9999$ used in our implementation generously satisfy (2.10) and (2.12) in either case.

2.4. The Dogleg Method. The traditional dogleg method (cf. [37], [11]) determines, at the kth Newton iterate u_k, a step along the *dogleg curve* Γ_k^{DL}. This is the piecewise linear curve connecting 0, the "Cauchy point" s_k^{CP} (defined to be the minimizer of the local linear model norm in the steepest descent direction), and the Newton step $s_k^{\mathrm{N}} = -F'(u_k)^{-1}F(u_k)$. The dogleg curve has the desirable properties that, as a point s traverses the curve from 0 to s_k^{CP} to s_k^{N}, $\|s\|$ is monotone increasing and $\|F(u_k) + F'(u_k)\, s\|$ is monotone decreasing (see, e.g., [11]). Consequently, if $\delta > 0$ is a given trust-region radius, then there is a unique $s_k \in \Gamma_k^{\mathrm{DL}}$ such that $s_k = \arg\min_{s\in\Gamma_k^{\mathrm{DL}}, \|s\|\le\delta} \|F(u_k) + F'(u_k)\, s\|$, and this s_k is characterized as follows: If $\|s_k^{\mathrm{N}}\| \le \delta$, then $s_k = s_k^{\mathrm{N}}$; otherwise, s_k is the unique point on the dogleg curve satisfying $\|s_k\| = \delta$. If s_k so chosen is acceptable, then the next iterate is $u_{k+1} = u_k + s_k$; if not, then δ is reduced and a new s_k is similarly determined.

In this study, we use a straightforward adaptation of the traditional method, outlined in general form below, that is suitable for implementation as a Newton–Krylov method. In this adaptation, each Newton step s_k^{N} is replaced by an inexact Newton step s_k^{IN}, and the corresponding dogleg curve Γ_k^{DL} connects 0, the Cauchy point s_k^{CP}, and s_k^{IN}. We note that the computation of s_k^{CP} requires the product of $F'(u_k)^T$ with a vector. As indicated in section 1, these products can be evaluated by our test codes. However, they may not be readily available in other circumstances, especially those involving "matrix-free" Newton–Krylov implementations in which the Jacobian is not created. A Newton-GMRES dogleg adaptation that does not require these products is described in [3]; in this, each Cauchy point s_k^{CP} is replaced by an approximation determined using quantities generated by GMRES.

ALGORITHM INDL. INEXACT NEWTON DOGLEG METHOD.

Let u_0, $\eta_{\max} \in [0,1)$, $t \in (0,1)$, $0 < \theta_{\min} < \theta_{\max} < 1$, *and* $0 < \delta_{\min} \le \delta$ *be given.*

For $k = 0, 1, \ldots$ *(until convergence) do:*

Choose $\eta_k \in [0, \eta_{\max}]$ *and* s_k^{IN} *such that*

$$\|F(u_k) + F'(u_k)\, s_k^{\mathrm{IN}}\| \le \eta_k \|F(u_k)\|.$$

Evaluate s_k^{CP} *and determine* $s_k \in \Gamma_k^{\mathrm{DL}}$.

While $ared_k < t \cdot pred_k$ *do:*

If $\delta = \delta_{\min}$, *stop; else choose* $\theta \in [\theta_{\min}, \theta_{\max}]$.

Update $\delta \leftarrow \max\{\theta\delta, \delta_{\min}\}$.

Redetermine $s_k \in \Gamma_k^{\mathrm{DL}}$.

Set $u_{k+1} = u_k + s_k$ *and update* δ.

The procedure for determining each $s_k \in \Gamma_k^{\mathrm{DL}}$ is as follows:

- If $\|s_k^{\mathrm{IN}}\| \le \delta$, then $s_k = s_k^{\mathrm{IN}}$.
- Otherwise, if $\|s_k^{\mathrm{CP}}\| \ge \delta$, then $s_k = (\delta/\|s_k^{\mathrm{CP}}\|)\, s_k^{\mathrm{CP}}$.
- Otherwise, $s_k = (1-\tau)s_k^{\mathrm{CP}} + \tau s_k^{\mathrm{IN}}$, where $\tau \in (0,1)$ is uniquely determined so that $\|s_k\| = \delta$.

This procedure always determines s_k uniquely and is standard for dogleg implementations. However, there are several issues that arise as a consequence of using s_k^{IN} in place of s_k^{N}. First, for any $\eta_k \in (0, \eta_{\max}]$, no matter how small, $\|F(u_k) + F'(u_k)\, s\|$ may not be monotone decreasing as s traverses Γ_k^{DL} from s_k^{CP} to s_k^{IN}; consequently, s_k may not minimize the local linear model norm along Γ_k^{DL} within the trust region. However, if η_k is small, then this nonmonotonicity can occur only in a small neighborhood of s_k^{IN} and is not a serious concern. Second, unless η_k is sufficiently small, $\|s\|$ may not be monotone increasing as s traverses Γ_k^{DL} from s_k^{CP} to s_k^{IN}; in this case, if $\|s_k^{\mathrm{CP}}\| > \delta$, then Γ_k^{DL} may have up to *three* points of intersection with the trust-region boundary: one between 0 and s_k^{CP} and one or two between s_k^{CP} and s_k^{IN}. Thus s_k may not be uniquely characterized by the property $\|s_k\| = \delta$. Third, and perhaps of greatest concern, if η_k is sufficiently large to allow

$$\eta_k \|F(u_k)\| \ge \|F(u_k) + F'(u_k)\, s_k^{\mathrm{IN}}\| \ge \|F(u_k) + F'(u_k)\, s_k^{\mathrm{CP}}\|$$

and $\|s_k^{\mathrm{IN}}\| \le \delta \le \|s_k^{\mathrm{CP}}\|$, then the procedure specifies $s_k = s_k^{\mathrm{IN}}$, even though the step $(\delta/\|s_k^{\mathrm{CP}}\|)s_k^{\mathrm{CP}}$ may (depending on δ, s_k^{CP}, and s_k^{IN}) give greater reduction of the local linear model norm along Γ_k^{DL} within the trust region. Although Algorithm INDL was effective in our tests (see section 5), these issues remain a potential cause for concern. In recent work [35], we have explored alternative strategies that may mitigate them.

In the while-loop, $ared_k$ and $pred_k$ are defined as in (2.6). In our implementation, we used $t = 10^{-4}$ as in the case of Algorithm INB. In updating δ within the while-loop,

we used a simple reduction $\delta \leftarrow .25\delta$. (Alternatives based on minimizing interpolating polynomials are outlined in [11].) In the final update of δ following the while-loop, we used a procedure similar to that in [11], in which the trust region is shrunk if $\|F(u_k + s_k)\|$ and $\|F(u_k) + F'(u_k)\, s_k\|$ do not agree well, expanded if they agree especially well, and left unchanged otherwise. The specific procedure, in which $0 < \rho_s < \rho_e < 1$, $0 < \beta_s < 1 < \beta_e$, and $\delta_{\max} > \delta_{\min}$, is as follows:

- If $ared_k/pred_k < \rho_s$ and $\|s_k^{IN}\| < \delta$, then $\delta \leftarrow \max\{\|s_k^{IN}\|, \delta_{\min}\}$.
- Otherwise, if $ared_k/pred_k < \rho_s$, then $\delta \leftarrow \max\{\beta_s\delta, \delta_{\min}\}$.
- Otherwise, if $ared_k/pred_k > \rho_e$ and $\|s_k\| = \delta$, then $\delta \leftarrow \min\{\beta_e\delta, \delta_{\max}\}$.

In our implementation, we used $\rho_s = 0.1$, $\rho_e = 0.75$, $\beta_s = .25$, $\beta_e = 4.0$, $\delta_{\min} = 10^{-6}$, and $\delta_{\max} = 10^{10}$. The initial δ was determined after the computation of s_0^{IN} as follows: If $\|s_0^{IN}\| < \delta_{\min}$, then $\delta = 2\delta_{\min}$; otherwise, $\delta = \|s_0^{IN}\|$.

We conclude this subsection with a convergence theorem for the general Algorithm INDL. The proof is given in [34, App. 1]; the theorem also follows from [35, Thm. 2.1]. The theorem affirms a notable property of Algorithm INDL shared with other trust-region methods, viz., that every limit point of $\{u_k\}$ produced by the algorithm must be a *stationary point* of $\|F\|$.[5] (It is possible for line-search methods to produce iterates that converge to nonstationary points at which the Jacobian is singular; see [37] and [5].) Additionally, as in the case of Theorem 2.4, a particular consequence of the theorem is that if $\{u_k\}$ converges to a solution u_* such that $F'(u_*)$ is nonsingular, then the convergence is ultimately governed by the η_k's. Thus, as before, if $\eta_k = 10^{-4}$ for each k, then (2.3) holds with $\eta_* = 10^{-4}$, and if each η_k is given by (2.1) followed by (2.2), then (2.4) holds for sufficiently large k and a γ independent of k.

THEOREM 2.7. *Assume that F is continuously differentiable. If u_* is a limit point of $\{u_k\}$ produced by Algorithm* INDL*, then u_* is a stationary point of $\|F\|$. If additionally $F'(u_*)$ is nonsingular, then $F(u_*) = 0$ and $u_k \to u_*$; furthermore, $s_k = s_k^{IN}$ for all sufficiently large k.*

3. The Discretized Equations. In this section, the governing transport PDEs of interest are presented briefly. These equations describe the conservation of momentum and mass along with the thermal energy equation for a variable density low Mach number flow. The physical transport mechanisms include diffusion, convection, a volumetric continuity constraint, external surface forces set by pressure boundary conditions, and buoyancy forces that are incorporated in our examples by using a Boussinesq approximation [7].

The transport equations are (3.1)–(3.3) below. In these, the unknown quantities are the fluid velocity vector $\mathbf{u}$, the hydrodynamic pressure P, and the temperature T:

$$\text{Momentum:} \quad \rho\mathbf{u}\cdot\nabla\mathbf{u} + \nabla P - \nabla\cdot\{\mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\} - \rho\mathbf{g} = \mathbf{0}, \tag{3.1}$$

$$\text{Total Mass:} \quad \nabla\cdot\mathbf{u} = 0, \tag{3.2}$$

$$\text{Energy:} \quad \rho C_p\mathbf{u}\cdot\nabla T - \nabla\cdot\kappa\nabla T = 0. \tag{3.3}$$

In (3.1)–(3.3), ρ, μ, $\mathbf{g}$, κ, and C_p are, respectively, the density, the dynamic viscosity, the gravity vector, the thermal conductivity, and the specific heat at constant pressure. More information on this system of equations can be found in [46].

To complete the system, boundary conditions are imposed on (3.1)–(3.3) by taking combinations of Dirichlet conditions on $\mathbf{u}$, P, and T and specified stress and heat flux conditions. In section 4.1, we discuss the specific boundary conditions for each test problem.

[5] A vector $u \in \mathbb{R}^n$ is a stationary point of $\|F\|$ if $\|F(u)\| \le \|F(u) + F'(u)\, s\|$ for every $s \in \mathbb{R}^n$.

To obtain an algebraic system of equations $F(u) = 0$, a stabilized finite-element formulation of (3.1)–(3.3) is employed. This formulation, which follows [25] and [48], allows equal order interpolation of velocity and pressure and also provides stabilization of the convection operators to limit oscillations due to high grid Reynolds and Péclet number effects. To form F', the finite-element equations are linearized. The discrete form of these linearized terms is then determined by expanding the unknowns $\mathbf{u}$, P, and T and the weighting function Φ in terms of a linear finite-element basis. The resulting Newton equation (1.1) is a fully coupled nonsymmetric linear system.

4. The Test Problems and Algorithm Evaluation Framework.

4.1. The Test Problems. The three test problems described below are standard benchmark problems used in the verification of fluid flow codes and solution algorithms [47]. In the numerical tests, we employed both 2D and 3D forms of these problems.

4.1.1. The Thermal Convection Problem [8]. This problem describes the flow of a fluid driven by thermal convection in a differentially heated square box in the presence of gravity. It requires the solution of (3.1)–(3.3) on the unit square in $\mathbb{R}^2$ or the unit cube in $\mathbb{R}^3$. When the equations and boundary conditions are suitably nondimensionalized, two nondimensional parameters appear: the Rayleigh number Ra and the Prandtl number Pr. As Ra increases for fixed Pr, the nonlinear effects of the convection terms increase and the solution becomes increasingly difficult to obtain. In this study, we took Pr = 1 and varied Ra. The boundary conditions are no-slip on all surfaces and specified temperature values $T = T_{cold}$ and $T = T_{hot}$ on the $x = 0$ and $x = 1$ surfaces, respectively. The other surfaces are insulated, i.e., zero Neumann conditions. All solutions in two dimensions were computed on a 100×100 equally spaced mesh, which resulted in 40,804 unknowns for the discretized problem. In three dimensions, solutions were computed on a $32 \times 32 \times 32$ equally spaced grid, resulting in 179,685 unknowns for the discretized problem. Figure 4.1 depicts representative solutions of the problem.

4.1.2. The Backward-Facing Step Problem [17]. This problem involves the simulation of flow through a rectangular channel that is initially constricted and subsequently expands over a reentrant backward-facing step. It requires the solution of

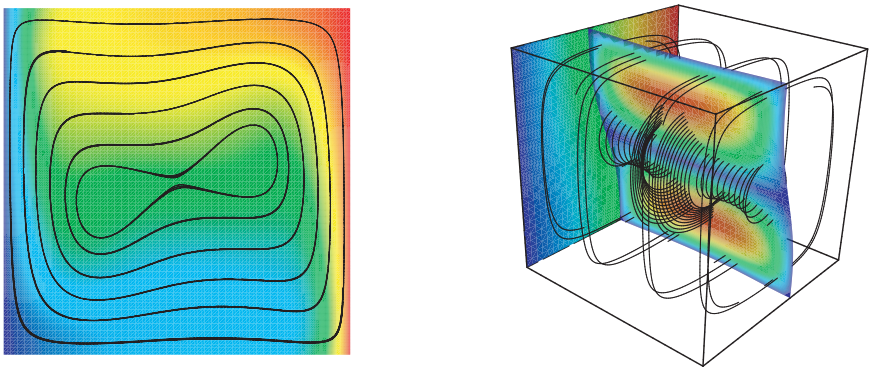

Fig. 4.1 *Thermal convection problem. Left: 2D model; color contour plot of temperature with streamlines (*$\mathrm{Ra} = 10^5$*). Right: 3D model; side-wall color contour plot of temperature, temperature isosurface colored by velocity vector magnitude, and streamlines (*$\mathrm{Ra} = 10^3$*).*

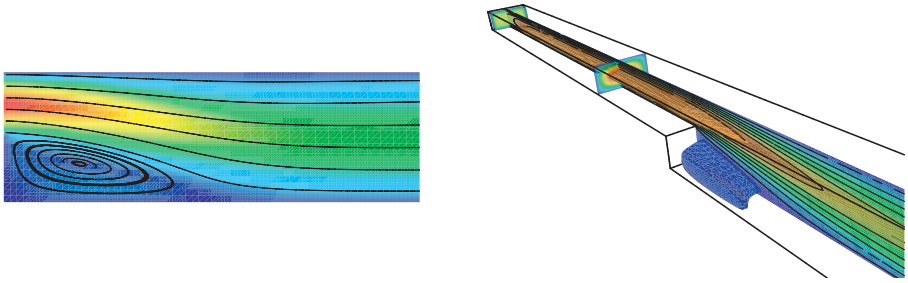

Fig. 4.2 *Backward-facing step problem. Left: 2D model; color contour plot of velocity vector magnitude with streamlines (*$\mathrm{Re} = 500$*). Right: 3D model; isosurfaces of positive (orange) and negative (blue) x-velocity with a color contour plot and isolines on a central slice plane (*$\mathrm{Re} = 200$*).*

(3.1)–(3.2). The nondimensional parameter is the Reynolds number Re. As Re is increased, the nonlinear convection components of the equations become more dominant and the problem becomes more difficult. As the fluid flows downstream, it produces a recirculation zone on the lower channel wall and, for sufficiently high Reynolds numbers, a second recirculation zone farther downstream on the upper wall.

In our 2D problem, the flow was computed only in the expanded portion of the channel, which had a 1×30 aspect ratio. Flow entering from the constricted portion was simulated by introducing a parabolic velocity profile, $\mathbf{u} = (24y(0.5 - y), 0)^T$, in the upper half of the inlet boundary and imposing zero velocity on the lower half. The remaining boundary conditions were no-slip on the solid surfaces and zero-stress on the outflow boundary. The discretization was a 20×400 unequally spaced mesh (with a finer mesh near the step), which resulted in 25,263 unknowns.

In three dimensions, we computed the flow over the entire domain, with the step placed one-fourth of the distance along the channel. The height-length and height-width ratios for the constricted portion of the channel were 1×20 and 1×8, respectively. For the expanded portion, these ratios were 1×30 and 1×4, respectively. The inlet velocity was set at $\mathbf{u} = (U_0, 0, 0)^T$; the remaining boundary conditions were as in the 2D problem. To provide an example of a larger-scale problem, we used a finer discretization on this problem than on the others, viz., a $20 \times 400 \times 30$ unequally spaced mesh. (As in the 2D case, the mesh was refined near the step.) This resulted in 1,042,236 unknowns. Figure 4.2 depicts representative solutions of the problem.

4.1.3. The Lid-Driven Cavity Problem [18], [43]. The third test problem addresses a confined flow in the unit square in $\mathbb{R}^2$ or the unit cube in $\mathbb{R}^3$ driven by a moving upper boundary with velocity U_0. As in the previous problem, the equations are (3.1)–(3.2), and the nondimensional parameter is the Reynolds number Re. All surfaces use no-slip boundary conditions. The problem becomes more difficult as Re increases, with increasingly prominent regions of countercirculation appearing in the corners of the domain.

For the 2D tests, we took Re up to 10,000. For the 3D tests, we used only values of Re up to 1,000 since there is evidence that the stability of the solution is questionable for $\mathrm{Re} > 700$. The 2D problem was discretized on a 100×100 equally spaced grid, which resulted in 30,603 unknowns for the discretized problem. In three dimensions, the discretization was a $32 \times 32 \times 32$ equally spaced grid, resulting in 143,748 unknowns. Figure 4.3 depicts representative solutions of the problem.

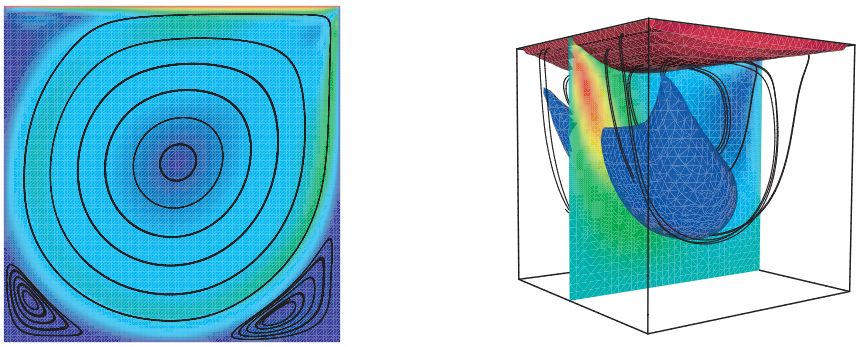

Fig. 4.3 *Lid-driven cavity problem. Left: 2D model; color contour plot of velocity vector magnitude with streamlines (*$\mathrm{Re} = 5000$*). Right: 3D model; isosurfaces of positive (red) and negative (blue) x-velocities with a color contour plot of y-velocity on a central slice plane and a few streamlines (*$\mathrm{Re} = 200$*).*

4.2. The Algorithm Evaluation Framework. For our tests, we used implementations of the algorithms in section 2 provided by the NOX nonlinear solver package [29]. NOX is a C++ object-oriented library designed for the efficient solution of nonlinear systems. It offers various globalized Newton-based solvers, including those in section 2, and other techniques such as tensor and Broyden methods. The GMRES implementation and preconditioners for the linear subproblems were from the AztecOO package [22], an extension of the Aztec library [49], which provides an easy-to-use framework for the efficient parallel solution of linear systems through an extensive suite of Krylov solvers and preconditioners. We give more details below about the particular GMRES algorithm and preconditioners used in our tests.

The parallel finite-element reacting flow code MPSalsa [46] was used to set up the finite-element discretization described in section 3 and to invoke the solvers, calling the Chaco [21] graph partitioning tool to partition the finite-element mesh into subdomains and assign subdomains to processors. For a detailed description of parallel finite-element data structures and a discussion of the strong link between partitioning quality and parallel efficiency, see [45].

In our tests, we used the restarted GMRES implementation in AztecOO with a restart value (maximum Krylov subspace size) of 200 and a maximum number of GMRES iterations equal to 600 (i.e., three restarts). Products of the Jacobian F' or, when needed, its transpose with vectors were evaluated by analytically evaluating F' as indicated in section 3 and subsequently using the stored value of F' or its transpose to perform the necessary matrix-vector multiplications. (However, products of F' with vectors needed by the Moré–Thuente line-search iterations were, in most instances, approximated with finite differences of F-values. See section 5 for details.)

In large-scale PDE applications such as those of interest here, preconditioning is a very important factor in GMRES performance and is the key to scalability on massively parallel machines. The preconditioners available in AztecOO include numerous algebraic and polynomial preconditioners and also additive-Schwarz domain decomposition preconditioners, which use incomplete LU factorizations for subdomain solves and allow variable levels of overlap. In our experiments, we used an additive-Schwarz preconditioner with one level of overlap and an ILUT(*fill-in*,*drop*) incomplete factorization [38] for each subdomain solve. We used *fill-in* $= 1$ and *drop* $= 0$, resulting in no additional fill-in and no entries dropped due to small magnitude.

In all of our tests, we imposed left-sided diagonal scaling. In this scaling, at the kth step and for $i = 1, \ldots, n$, $F_i(u_k)$ was scaled by the inverse of the sum of the absolute values of the entries in the ith row of $F'(u_k)$. This scaling was very important to success in many cases. It was used not only in the GMRES solve but also throughout the algorithm by incorporating it as a diagonal scaling matrix in the inner product.

Successful termination of the Newton–Krylov iterations was declared if $\|F(u_k)\| \leq \varepsilon_F \|F(u_0)\|$, where $\varepsilon_F = 10^{-2}$ in our tests, *and* the step-length criterion $\frac{1}{n}||Ws_k||_2 < 1$ was also satisfied, where n is the total number of unknowns and W is a diagonal weighting matrix with entries $W_{ii} = 1/(\varepsilon_r |u_k^{(i)}| + \varepsilon_a)$, in which $u_k^{(i)}$ is the ith component of u_k and $\varepsilon_r = 10^{-3}$ and $\varepsilon_a = 10^{-8}$ in our tests. (We also enforced safeguards to prevent this step-length criterion from being prematurely satisfied; see [29].) In our experience, this second criterion is typically more stringent and is necessary to ensure that finer physical details of the flow and transport are adequately resolved. The weighting matrix assures that all variables are treated equitably in deciding when to terminate. This weight-matrix definition is similar to one used to dynamically control time-step sizes that is standard in general purpose ODE packages such as LSODE [24]. Finally, all tests on the 2D problems were done using 8 processors; tests on the 3D thermal convection and lid-driven cavity problems were done using 30 processors; and tests on the 3D backward-facing step problem were done using 100 processors. In all tests, the number of processors corresponded to the number of subdomains used in the additive-Schwarz preconditioners.

5. Results. We performed extensive numerical tests involving the application of the methods in section 2 to the benchmark problems in section 4. For comparison, we also tested a nonglobalized method, i.e., a method taking full, unmodified inexact Newton steps. For each method, we used both small constant (10^{-4}) forcing terms and adaptive (Choice 1) forcing terms, as discussed in section 2.1. In every test case, the initial approximate solution was the zero vector.

We first consider the robustness of the methods. Table 5.1 shows the numbers of method failures on the test problems for the following parameter values:

2D and 3D thermal convection: Ra = 10^3, 10^4, 10^5, 10^6,
2D and 3D backward-facing step: Re = 100, 200, ..., 700, 750, 800,
2D lid-driven cavity: Re = 1,000, 2,000, ..., 10,000,
3D lid-driven cavity: Re = 100, 200, ..., 1,000.

In Table 5.1 and in what follows, "INB-Q" refers to Algorithm INB with each $\theta \in [\theta_{\min}, \theta_{\max}]$ determined by minimizing a quadratic interpolating polynomial. "INB-

Table 5.1 *Failure totals: The 2nd, 4th, and 6th columns show numbers of failures with the adaptive forcing terms, with numbers of failures for the constant forcing terms in square brackets; the 3rd, 5th, and 7th columns show total numbers of failures.*

Method	2D problems		3D problems		All problems	
INB-Q	0 [10]	10	0 [0]	0	0 [10]	10
INB-QC	1 [9]	10	0 [0]	0	1 [9]	10
INMTL	1 [9]	10	1 [2]	3	2 [11]	13
INDL	0 [10]	10	0 [0]	0	0 [10]	10
INFS	15 [18]	33	4 [10]	14	19 [28]	47

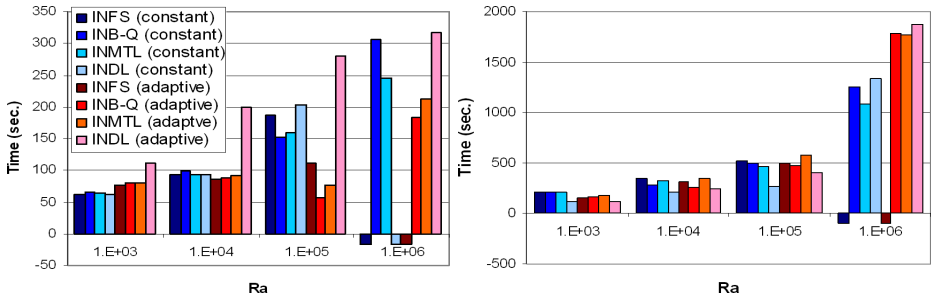

Fig. 5.1 *Thermal convection problem timings (in seconds): 2D (left) and 3D (right).*

QC" refers to the method that minimizes an interpolating quadratic on the first step-length reduction and then minimizes interpolating cubics on all subsequent step-length reductions (see section 2.2). "INFS" designates the nonglobalized method that takes full, unmodified inexact Newton steps.

The results in Table 5.1 indicate that, overall, each of the globalizations significantly improved robustness in these experiments. The INB methods and the INDL method suffered fewer failures overall than the INMTL method, but by only a modest margin. More significantly, the adaptive forcing terms considerably improved the robustness of all methods, including the INFS method; for the globalized methods, the improvement was dramatic. This outcome is consistent with the study in [47], which included results for other adaptive and constant forcing terms in addition to those considered here. Overall, the combination of adaptive forcing terms and globalization was very effective on these problems, although it did not lead to success in every case.

The more detailed results graphed in Figures 5.1–5.3, in which failure is indicated by a negative bar, show that the robustness benefit of globalization varied considerably among the test cases. For example, it was striking for the 2D backward-facing step problem, for which there were only two cases in which a globalized method failed. In contrast, it was only marginal with small constant forcing terms on the 2D lid-driven cavity problem; in these tests, all of the globalized methods failed for $\mathrm{Re} > 1{,}000$. No globalization was always superior to the others in these tests. For example, the INB and INMTL methods succeeded in every case of the 2D thermal convection problem, while the INDL method failed in one instance; however, the INDL method was the only method to succeed in every case of the 2D backward-facing step problem. We discuss Figures 5.1–5.3 further at the end of this section.

A remark is in order concerning the INMTL method. Each line-search iteration requires evaluating ϕ defined by (2.7) and also ϕ', and the latter evaluation requires multiplication of a vector by F' evaluated at a new point. For all of the test problems except one, these F'-products were satisfactorily approximated using finite differences of F-values. The exception was the 3D backward-facing step problem, in which errors in the finite-difference approximations at times caused the algorithm to fail. For this problem, it was necessary to evaluate each of these products analytically, at the cost of a fresh Jacobian evaluation and a matrix-vector multiplication, in order to obtain the level of success shown in Table 5.1 and Figure 5.2.

We next consider the efficiency of the methods. Table 5.2 shows various statistics for a selected set of test cases. This set includes all cases considered in the robustness study in which all of the globalized methods succeeded. Additionally, since at least one globalized method failed on the 2D lid-driven cavity problem for each $\mathrm{Re} > 1{,}000$,

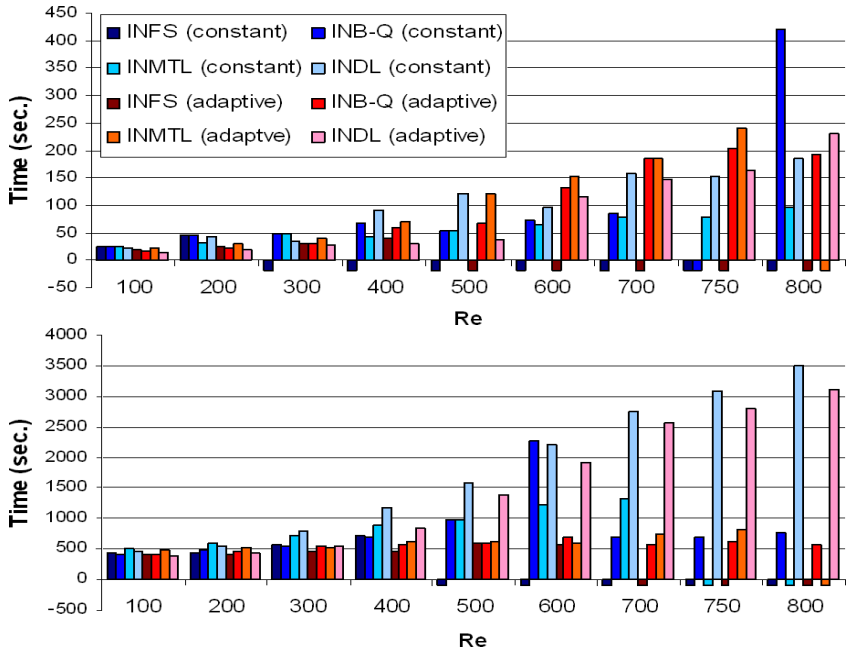

Fig. 5.2 *Backward-facing step problem timings (in seconds): 2D (top) and 3D (bottom).*

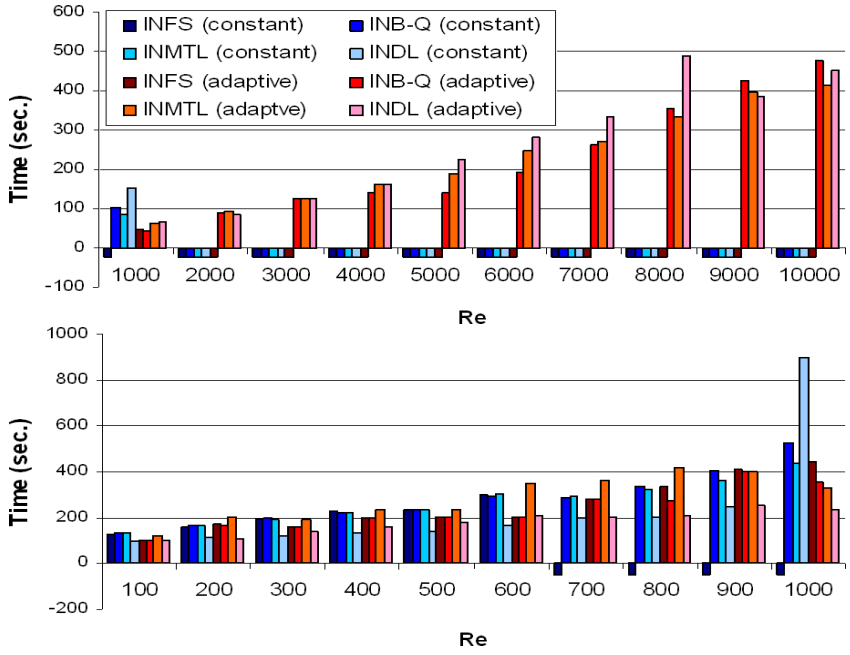

Fig. 5.3 *Lid-driven cavity problem timings (in seconds): 2D (top) and 3D (bottom).*

Table 5.2 *Efficiency study: The columns show results with the adaptive forcing terms, with values for the constant forcing terms in square brackets.*

Method	Inexact Newton steps	Backtracks per per inexact Newton step	Total function evals.	Total GMRES iterations	GMRES iterations per inexact Newton step	Normalized time
INB-Q	16.0 [9.23]	0.13 [0.18]	19.2 [11.7]	997.1 [1502]	62.2 [163]	0.77 [1]
INB-QC	16.0 [9.20]	0.13 [0.20]	19.2 [11.8]	989.6 [1498]	61.9 [163]	0.77 [1.02]
INMTL	15.2 [8.67]	0.17 [0.17]	43.4 [25.1]	979.7 [1408]	64.3 [162]	0.90 [0.96]
INDL	17.0 [10.7]	NA [NA]	18.9 [12.6]	1454 [1799]	85.3 [168]	0.83 [1.01]

it includes cases of this problem with $100 \leq \mathrm{Re} \leq 1{,}000$. (We did not include the nonglobalized INFS method in this study because its high incidence of failure would have made the test set undesirably small.) The specific cases considered are

$$\begin{aligned} \text{2D thermal convection:} &\quad \mathrm{Ra} = 10^3,\ 10^4,\ 10^5, \\ \text{3D thermal convection:} &\quad \mathrm{Ra} = 10^3,\ 10^4,\ 10^5,\ 10^6, \\ \text{2D and 3D backward-facing step:} &\quad \mathrm{Re} = 100,\ 200, \ldots, 700, \\ \text{2D and 3D lid-driven cavity:} &\quad \mathrm{Re} = 100,\ 200, \ldots, 1{,}000. \end{aligned}$$

In Table 5.2, all table values are geometric means, except for numbers of backtracks per inexact Newton step, which are arithmetic means. Mean run times are relative to that of the INB-Q method with constant forcing terms.

Table 5.2 indicates that, for each choice of the forcing terms, the globalized methods performed rather similarly on this test set. There are some differences in performance; in particular, the INMTL method required more function evaluations than the other methods, while the INDL method required more GMRES iterations. However, in view of the modest size of the test set, these differences may not have much significance. In contrast, notable differences are seen with each method when different forcing terms were used. Compared to the small constant forcing terms, the adaptive forcing terms resulted in greatly reduced mean numbers of GMRES iterations per inexact Newton step and significantly reduced mean numbers of total GMRES iterations. On the other hand, the small constant forcing terms required significantly fewer inexact Newton steps on average. In the balance, the adaptive forcing terms yielded considerably better run times than the small constant forcing terms (although only slight improvement with the INMTL method).

The bar graphs in Figures 5.1–5.3 show run times and method failures (indicated by a negative bar) for all problems considered in the robustness study (Table 5.1) and for all methods except the INB-QC method, which performed very similarly to the INB-Q method. Methods using the small constant forcing terms are shown in shades of blue; methods using the adaptive forcing terms are shown in shades of red. The results for the thermal convection problem in Figure 5.1 show failures at only the largest Rayleigh number Ra of 10^6. At this Ra, the nonglobalized INFS method always failed; in the 2D case, the INDL method with constant forcing terms also failed. For the backward-facing step problem, Figure 5.2 shows many more failures. At low values of the Reynolds number Re, all methods succeeded. As the problems became more difficult with increasing Re, the INFS method began to fail in both the 2D and 3D cases, first with the small constant forcing terms and, for higher Re, with the adaptive forcing terms as well. At the two largest Re values, failures of the globalized methods were observed. The only methods that converged over the entire Re range were the

INB-Q method with adaptive forcing terms and the INDL method with both adaptive and small constant forcing terms. The 2D lid-driven cavity problem (Figure 5.3, top) was more difficult to solve. No method succeeded with small constant forcing terms beyond Re = 1,000; however, all globalized methods in combination with adaptive forcing terms attained convergence over the entire range. In the 3D cases (Figure 5.3, bottom), the methods exhibited excellent convergence, with only the INFS method with small constant forcing terms failing for higher Re.

6. Additional Remarks on Failure and Robustness. In practice, globalized Newton–Krylov methods can fail in a number of ways, and there are a variety of factors that contribute to success or failure. We describe below several general failure modes and comment on the extent to which these were observed in our tests.

- *Fatal near-stagnation.* The method achieves sufficient residual norm reduction at each step to continue but not enough *in toto* to succeed before reaching the maximum allowable number of steps. This mode accounted for most of the failures in our tests: 26 of 33 failures for the backtracking/line-search methods and all 10 failures for the dogleg method. In our tests, we specified generous maximum allowable numbers of steps: 300 for the 2D lid-driven cavity problem and 200 in all other cases. In no failure case did it seem likely that increasing these numbers would have led to success.
- *Globalization failure.* The globalization fails to determine an acceptable step at some Newton–Krylov iteration. For example, a backtracking routine might fail to produce an acceptable step within the maximum allowable number of step-length reductions. In our tests, such failures accounted for 7 of 33 failures for the backtracking and line-search methods.
- *Divergence.* The iterates fail to converge. This is likely to be manifested in unbounded growth of the iterates, which we observed in some of our tests involving the Newton-GMRES method with no globalization. We did not observe this in any of our tests with the globalized methods. However, it can occur with globalized methods, as can other forms of divergence [12, p. 400].
- *Component failure.* Failure occurs with one or more "components" of the algorithm, such as the Krylov solver, the preconditioner, or the function evaluation routine. We saw no failures of this type in our tests.

Many ancillary factors may affect the robustness of a globalized Newton–Krylov method on problems such as those of interest here. The following are several that we have found to be influential.

- *The nonlinear nature of the continuous PDE problem.* Nonlinear PDE systems often have characteristic parameters, such as the Reynolds number, the Rayleigh number, and the Prandtl number in our test problems. These parameters may strongly affect problem difficulty (see section 4.1), and the limits of practical solvability may occur near critical parameter values at which a steady-state solution becomes unstable.
- *The discretization of the PDE problem.* Failure of the spatial discretization to adequately reflect the underlying physics of the continuous problem can cause convergence difficulties for globalized Newton–Krylov methods. For example, in the case of strong convection (large Reynolds numbers) in our prototype problems, common spatial discretizations, such as centered finite-difference or Galerkin finite-element methods, become unstable when the computational mesh is too coarse, exhibiting nonphysical spatial oscillations and producing ill-conditioned Jacobian matrices [14]. This ill-conditioning is likely to

result in poor convergence of the Krylov solver and, in turn, the Newton–Krylov method. (Schemes such as the stabilized finite-element method used in our tests attempt to suppress spurious oscillations on coarser grids and also produce Jacobians that are better conditioned.) Additionally, ill-chosen spatial discretizations can produce spurious nonphysical solutions (see [44], [53]), and Newton–Krylov iterates may converge to these from an unfortunate initial guess.

- *The convergence of the Krylov solver and preconditioning.* Effective preconditioning (see, e.g., [39] and [2]) is essential for good Krylov solver performance on large, complex nonlinear discretized PDE systems. Among various issues that make the linear subproblems challenging, a particular concern is the ill-conditioning that results from using very fine mesh spacing or very large aspect-ratio cells or elements, and suitable preconditioning is necessary to address this. Also, as indicated in section 4.2, appropriate preconditioning is central to the scalability of the Krylov solver and, therefore, the Newton–Krylov method on massively parallel platforms.
- *Scaling.* Proper scaling of variables can be an important contributor to method success. In our tests, we found it effective to implement scaling consistently throughout the algorithm by incorporating it into a certain weighted norm defined by row-sums of the Jacobian matrix (see section 4.2). This technique significantly improved the robustness of all methods in our tests.
- *Accuracy.* Finally, the accuracy of computations within the algorithm is important. As noted in section 5, finite-difference approximations of Jacobian-vector products within the Moré–Thuente line search were insufficiently accurate for good success in one case; analytic evaluations were necessary. Also, in preliminary experimentation, we found that high-accuracy Jacobian evaluations used in producing matrix-vector products in GMRES solves resulted in much better method performance than certain cheaper but less accurate alternatives that were available in our codes. These high-accuracy evaluations were used in obtaining the results reported in section 5.

7. Conclusions. We have considered several representative globalizations intended to improve the robustness of a Newton–Krylov method: two variants of a backtracking method (with step-length reduction by quadratic and quadratic/cubic minimization) from [12], a line-search procedure from [31], and a dogleg implementation of a trust-region method (see [37], [11]). These methods all have strong global convergence properties, as indicated by the theoretical results outlined in section 2.

We extensively tested Newton-GMRES implementations of these methods on large-scale benchmark problems involving the steady-state 2D and 3D Navier–Stokes equations; the results are given in section 5. Each of the globalizations considered here significantly improved robustness in our tests. Overall, the backtracking methods and the dogleg method were most robust, with the backtracking methods producing slightly better run times. (The two backtracking step-length reduction strategies produced very similar results.) The line-search procedure of [31] performed almost as well. These overall results notwithstanding, no method was better than the others in every test, and the only methods to succeed in every case were the backtracking method (with quadratic minimization only) and the dogleg method, with each using adaptive forcing terms.

The use of adaptive forcing terms resulted in major improvements in the robustness of all methods, including the method with no globalization. For the globalized

methods, the improvement was dramatic. Using adaptive forcing terms also contributed significantly to the efficiency of the globalized methods.

Among the globalizations considered here, the backtracking method may be a first choice for implementation because of its simplicity as well as its effectiveness in our tests. In our backtracking tests, we saw no reason to prefer the step-length reduction strategy using both cubic and quadratic minimization over the simpler strategy that uses only quadratic minimization. We stress, though, that no method was uniformly superior in our experiments. Additionally, our test set was limited to a particular class of problems, and results on other types of problems may differ. Ideally, one would have several globalizations available to determine which works best in a particular application. Similarly, while the adaptive forcing terms greatly improved the performance of the globalized methods in these experiments, no particular choice of the forcing terms is best for all problems. (Indeed, the small constant value of 10^{-4} was the most effective forcing term among a number of alternatives in a 3D chemical vapor deposition reactor simulation described in [47].) Thus it would be ideal to have available several forcing term choices as well as several globalizations to determine the most effective combination.

Finally, as noted in additional remarks on failure and robustness in section 6, there are many factors other than the globalization and forcing terms that may affect the performance of a Newton–Krylov method on problems such as those of interest here, and these should be considered in formulating problems and algorithms to solve them. Additionally, there are other robust solution techniques, such as homotopy, continuation, pseudotransient continuation, and mesh-sequencing, that should be kept in mind as possible alternatives to globalized Newton–Krylov methods. In particular, if the goal is to traverse or map out a complex nonlinear solution set as problem parameters vary, then the most appropriate methods may be continuation methods, which can follow stable and unstable solution branches and track critical points as parameters are varied. If the goal is to find a stable steady-state solution within a complex nonlinear landscape, then pseudotransient continuation can also be used. We note that, for large-scale problems, Newton–Krylov methods are often used as nonlinear solvers within these methods (cf. [42], [41], [53]).

REFERENCES

[1] E. Allgower and K. Georg, *Continuation and path following*, Acta Numer., 1993 (2) (1993), pp. 1–64.

[2] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

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

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

[5] R. H. Byrd, M. Marazzi, and J. Nocedal, *On the convergence of Newton iterations to non-stationary points*, Math. Program., 99 (2004), pp. 127–148.

[6] K. A. Cliffe, A. Spence, and S. J. Tavener, *The numerical analysis of bifurcation problems with application to fluid mechanics*, Acta Numer., 2000 (9) (2000), pp. 39–131.

[7] I. G. Currie, *Fundamental Mechanics of Fluids*, McGraw-Hill, New York, 1974.

[8] G. D. V. Davis and C. P. Jones, *Natural convection in a square cavity: A comparison exercise*, Internat. J. Numer. Methods Fluids, 3 (1983), pp. 227–248.

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

[10] R. S. Dembo and T. Steihaug, *Truncated Newton algorithms for large-scale optimization*, Math. Programming, 26 (1983), pp. 190–212.

[11] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall Ser. Comput. Math., Prentice-Hall, Englewood Cliffs, NJ, 1983.
[12] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.
[13] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
[14] C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics*, Comput. Phys. 2, Springer-Verlag, Berlin, Heidelberg, 1988.
[15] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.
[16] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numer., 1992 (1) (1992), pp. 57–100.
[17] D. K. Gartling, *A test problem for outflow boundary conditions—flow over a backward facing step*, Internat. J. Numer. Methods Fluids, 11 (1990), pp. 953–967.
[18] U. Ghia, K. N. Ghia, and C. T. Shin, *High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.
[19] G. H. Golub and C. V. Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.
[20] M. H. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations*, Acta Numer., 1997 (6) (1997), pp. 271–397.
[21] B. Hendrickson and R. Leland, *The Chaco User's Guide–Version* 1.0, Tech. Report Sand93-2339, Sandia National Laboratories, Albuquerque, NM, 1993.
[22] M. Heroux, *AztecOO: Object-Oriented Aztec Linear Solver Package*, http://software.sandia.gov/trilinos/packages/aztecoo/index.html.
[23] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–435.
[24] A. C. Hindmarsh, *LSODE and LSODEI: Two new initial value ordinary differential equation solvers*, ACM Signum Newsletter, 15 (1980), pp. 10–11.
[25] T. J. R. Hughes, L. P. Franca, and G. M. Hulbert, *A new finite element formulation for computational fluid dynamics:* VII. *The Galerkin/Least-Squares method for advective-diffusive equations*, Comput. Methods Appl. Mech. Engrg., 73 (1989), pp. 173–189.
[26] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, Fundam. Algorithms 1, SIAM, Philadelphia, 2003.
[27] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Numer. Anal., 35 (1998), pp. 508–523.
[28] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton–Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.
[29] T. G. Kolda and R. P. Pawlowski, *NOX Nonlinear Solver Project*, http://software.sandia.gov/nox.
[30] M. Kubicek and M. Marek, *Computational Methods in Bifurcation Theory and Dissipative Structures*, Springer Ser. Comput. Phys., Springer-Verlag, New York, 1983.
[31] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Trans. Math. Soft., 20 (1984), pp. 286–307.
[32] S. G. Nash, *Truncated Newton Methods*, Ph.D. thesis, Computer Science Department, Stanford University, Stanford, CA, 1982.
[33] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[34] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker, *Globalization Techniques for Newton–Krylov Methods and Applications to the Fully-Coupled Solution of the Navier–Stokes Equations*, Tech. Report Sand2004-1777, Sandia National Laboratories, Albuquerque, NM, 2003.
[35] R. P. Pawlowski, J. N. Shadid, J. P. Simonis, and H. F. Walker, *Inexact Newton dogleg methods*, Tech. Report MS-03-02-18, WPI Math. Sciences Dept., Worcester, MA, 2005. Submitted to SIAM J. Numer. Anal.
[36] M. Pernice and H. F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.
[37] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz, ed., Gordon and Breach, London, 1970, pp. 87–114.
[38] Y. Saad, *ILUT: A dual threshold incomplete* ILU *factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[39] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, MA, 1996.
[40] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[41] A. Salinger, E. Burroughs, R. Pawlowski, E. Phipps, and L. Romero, *Bifurcation analysis algorithms and software for large-scale applications*, Internat. J. Bifur. Chaos, 15 (2005), pp. 1015–1032.
[42] A. G. Salinger, R. B. Lehoucq, R. P. Pawlowski, and J. N. Shadid, *Computational bifurcation and stability studies of the* 8:1 *thermal cavity problem*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 1059–1073.
[43] R. Schreiber and H. B. Keller, *Driven cavity flows by efficient numerical techniques*, J. Comput. Phys., 49 (1983), pp. 310–333.
[44] R. Schreiber and H. B. Keller, *Spurious solutions in driven cavity calculations*, J. Comput. Phys., 49 (1983), pp. 165–172.
[45] J. N. Shadid, S. A. Hutchinson, G. L. Hennigan, H. K. Moffat, K. D. Devine, and A. G. Salinger, *Efficient parallel computation of unstructured finite element reacting flow solutions*, Parallel Comput., 23 (1997), pp. 1307–1325.
[46] J. N. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger, *MPSalsa: A Finite Element Computer Program for Reacting Flow Problems, Part* 1: *Theoretical Development*, Tech. Report Sand95-2752, Sandia National Laboratories, Albuquerque, NM, 1996.
[47] J. N. Shadid, R. S. Tuminaro, and H. F. Walker, *An inexact Newton method for fully-coupled solution of the Navier–Stokes equations with heat and mass transport*, J. Comput. Phys., 137 (1997), pp. 155–185.
[48] T. E. Tezduyar, *Stabilized finite element formulations for incompressible flow computations*, Adv. Appl. Mech., 28 (1992), pp. 1–44.
[49] R. S. Tuminaro, M. Heroux, S. A. Hutchinson, and J. N. Shadid, *Aztec User's Guide—Version* 2.1, Tech. Report Sand99-8801J, Sandia National Laboratories, Albuquerque, NM, 1999.
[50] R. S. Tuminaro, H. F. Walker, and J. N. Shadid, *On backtracking failure in Newton–GMRES methods with a demonstration for the Navier–Stokes equations*, J. Comput. Phys., 180 (2002), pp. 549–558.
[51] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.
[52] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, UK, 2003.
[53] H. F. Walker, *An adaptation of Krylov subspace methods to path following problems*, SIAM J. Sci. Comput., 21 (1999), pp. 1191–1198.
[54] L. T. Watson, *Globally convergent homotopy algorithms for nonlinear systems of equations*, Nonlinear Dynamics, 1 (1990), pp. 143–191.
[55] L. T. Watson, M. Sosonkina, R. C. Melville, A. P. Morgan, and H. F. Walker, *HOMPACK90: A suite of* Fortran 90 *codes for globally convergent homotopy algorithms*, ACM Trans. Math. Soft., 23 (1997), pp. 514–549.