AIAA JOURNAL
Vol. 32, No. 12, December 1994

Comparison of Standard and Matrix-Free Implementations of Several Newton–Krylov Solvers

Paul R. McHugh* and Dana A. Knoll†
Lockheed Idaho Technologies Company, Idaho Falls, Idaho 83415

Fully coupled Newton's method is combined with conjugate gradient-like iterative algorithms to form inexact Newton-Krylov algorithms for solving the steady, incompressible, Navier–Stokes and energy equations in primitive variables. Finite volume differencing is employed using the power law convection-diffusion scheme on a uniform but staggered mesh. The well-known model problem of natural convection in an enclosed cavity is solved. Three conjugate gradient-like algorithms are selected from a class of algorithms based upon the Lanczos biorthogonalization procedure; namely, the conjugate gradients squared algorithm, the transpose-free quasi-minimal-residual algorithm, and a more smoothly convergent version of the biconjugate gradients algorithm. A fourth algorithm is based upon the Arnoldi process, namely the popular generalized minimal residual algorithm (GMRES). The performance of a standard inexact Newton's method implementation is compared with a matrix-free implementation. Results indicate that the performance of the matrix-free implementation is strongly dependent upon grid size (number of unknowns) and the selection of the conjugate gradient-like method. GMRES appeared to be superior to the Lanczos based algorithms within the context of a matrix-free implementation.

Nomenclature

$\boldsymbol{A}$	= general system matrix
a	= perturbation constant
$\boldsymbol{b}$	= general right-hand side vector
d	= damping constant
$\boldsymbol{F}$	= vector of discrete governing equations
f	= individual discrete governing equation
Gr	= Grashof number, $L^3\beta g\Delta T/\nu_0^2$
g	= gravitational acceleration in negative y direction
$\boldsymbol{J}$	= Jacobian matrix
$\boldsymbol{K}_k$	= Krylov subspace of dimension k
k	= dimension of Krylov subspace
m	= inner iteration counter
N	= dimension of linear system
n	= Newton iteration counter
$\boldsymbol{P}$	= preconditioning matrix
Pr	= Prandtl number, ν/α
p	= dimensionless pressure, $p^*/\rho_0 V^2$
Ra	= Rayleigh number, $Gr\,Pr$
R_n^i	= scaled inner iteration residual norm
R_n^o	= relative update for nth Newton iteration
T	= dimensionless temperature, $(T^* - T_0)/\Delta T$
u	= dimensionless velocity, x direction, u^*/V
V	= characteristic velocity scale, ν_0/L
v	= dimensionless velocity, y direction, v^*/V
$\boldsymbol{w}$	= arbitrary Krylov vector
$\boldsymbol{x}$	= state variable vector
x, y	= Cartesian coordinate variables, x^*/L, y^*/L
α	= thermal diffusivity
β	= coefficient of thermal expansion
γ	= tolerance for inner linear
ΔT	= characteristic temperature difference, $T_n - T_0$
ε	= perturbation
ν	= kinematic viscosity
ρ	= density

Subscripts

h	= hot wall value
i	= vector component number
n	= Newton iteration number
0	= reference value

Superscripts

i	= inner iteration
n	= Newton iteration number
o	= outer iteration
*	= dimensional value

Operators

$[A]^T$	= transpose of $\boldsymbol{A}$
$\|\boldsymbol{b}\|$	= Euclidean norm of $\boldsymbol{b}$
$\|b\|_\infty$	= L_∞ norm of $\boldsymbol{b}$
$\bar{m}$	= average value of m

Introduction

THE use of robust, fully coupled algorithms to solve the Navier–Stokes equations is growing in popularity mainly due to the rapid advances in computer speed and available memory. An "inexact" Newton's method refers to the use of an iterative solver to approximately solve the linear systems arising from a Newton linearization of the governing equations.[1,2] The primary advantage in the use of an iterative solver is reduced memory requirements. Additionally, the tolerance of the linear equation solver can be relaxed when far from the true solution and then tightened as the true solution is approached. Preconditioned conjugate gradient-like iterative algorithms have been successfully used in this fashion.[3–10]

The linear systems arising on each Newton step are of the form $\boldsymbol{J}\delta\boldsymbol{x} = -\boldsymbol{F}(\boldsymbol{x})$. True conjugate gradient methods compute approximations to $\delta\boldsymbol{x}$ in the affine space $\delta\boldsymbol{x}_0 + \boldsymbol{K}_k$.[11] They are characterized by an optimality condition and short vector recurrence relationships.[12] In many of these Krylov projection methods, the Jacobian matrix appears only in matrix–vector products of the form $\boldsymbol{J}\boldsymbol{w}$. This becomes very important in the context of an inexact Newton

Received June 2, 1993; presented as Paper 93-3332 at the 11th AIAA Computational Fluid Dynamics Conference, Orlando, FL, July 6–9, 1993; revision received March 29, 1994; accepted for publication April 1, 1994.

*Department of Engineering Analysis, Idaho National Engineering Laboratory; currently Senior Engineer, Computational Fluid Dynamics Team, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415. Member AIAA.

†Department of Engineering Analysis, Idaho National Engineering Laboratory; currently Senior Engineering Specialist, Computational Fluid Dynamics Team, EG&G Idaho, Inc., P.O. Box 1625, Idaho Falls, ID 83415.

iteration because these products may be approximated by finite differences as follows [8–10]:

$$\boldsymbol{J}\boldsymbol{w} \approx \frac{\boldsymbol{F}(\boldsymbol{x} + \varepsilon \boldsymbol{w}) - \boldsymbol{F}(\boldsymbol{x})}{\varepsilon} \tag{1}$$

The existence of this approximation is significant because it suggests the possibility of matrix-free implementations of Newton's method, thereby circumventing the main drawback associated with its use. The performance of this matrix-free implementation compared with the standard implementation is the focus of this paper.

Note that for symmetric matrices short vector recurrence relationships arise naturally, resulting in constant work and storage requirements on each iteration. For nonsymmetric matrices, however, short recurrences do not exist,[13] and so the work and storage requirements increase with the iteration number, making the use of true conjugate gradient methods impractical. However, some problems allow successful application of true conjugate gradient algorithms to the normal equations. Disadvantages in this approach, however, are that the condition number of the new system is made much worse, and matrix–vector multiplications with the transpose are required. Working with the transpose is undesirable for several reasons: first, the transpose is not always readily available; second, the efficiency of matrix–vector multiplications with the transpose may be reduced on vector/parallel computers; and third, working with the transpose eliminates the option of the aforementioned matrix-free implementations of Newton's method. For these reasons we chose to concentrate on the performance of conjugate gradient-like algorithms.

Conjugate gradient-like algorithms are derived by either relaxing the optimality condition or sacrificing short vector recursion relationships.[14] The optimality condition may be relaxed by allowing periodic algorithm restarts and by artificially truncating the recursion (i.e., the new direction vector is orthogonal to only the previous s direction vectors). Additionally, economical vector recursions can also be obtained, at the expense of optimality, by using the Lanczos biorthogonalization procedure (i.e., using three-term recursions to build a pair of biorthogonal bases).[14]

The restarted generalized minimal residual (GMRES) algorithm, based on the use of the Arnoldi process, is derived by relaxing the optimality condition.[15] Note that the full GMRES algorithm maintains optimality but does not exhibit short vector recursions. Consequently, the storage requirements grow linearly and the work quadratically with the iteration number.[7] Thus, it is often necessary to use the restarted version, GMRES(k), where k indicates the selected dimension of the Krylov subspace. In this case, the algorithm is restarted after k iterations. Restarts may cause the GMRES algorithm to exhibit very slow convergence, but GMRES will not encounter the breakdowns that are possible with the algorithms described later.[16]

Algorithms derived by relaxing the optimality condition using the nonsymmetric Lanczos procedure include the biconjugate gradient (BCG) algorithm,[17,18] a more smoothly convergent version of the biconjugate gradient algorithm (Bi-CGSTAB) and its variants,[19] the conjugate gradients squared (CGS) algorithm,[20] and a set of algorithms based on the quasi-minimal-residual idea.[21] Compared with the Arnoldi-based method (i.e., GMRES), these Lanczos-based methods require less work and storage per iteration.[22]

The first of the Lanczos-based methods developed, BCG, suffers from three main shortcomings[22]:

1) It requires matrix–vector multiplications with the matrix transpose.

2) It lacks the minimization property inherent in Arnoldi-based methods, resulting in irregular convergence properties.

3) Algorithm breakdown is possible.

The first shortcoming was overcome with the development of CGS. This algorithm is based upon the equivalent polynomial form of the BCG algorithm and is derived by squaring the BCG polynomial recurrences. This process results in an algorithm that avoids use of the matrix transpose and doubles the rate of convergence of BCG without increasing the amount of work.[20] Unfortunately, the CGS algorithm sometimes exhibits more irregular convergence behavior than BCG.[22] With the goal of obtaining more smoothly convergent

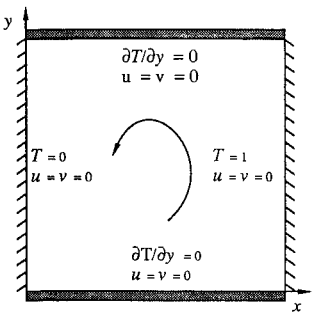

Fig. 1 Geometry and boundary conditions for the model problem.

CGS-like solutions, several new algorithms have been developed: Van der Vorst modified the BCG polynomial recurrences and used local steepest decent steps in the Bi-CGSTAB algorithm,[19] and Freund applied the quasi-minimal-residual idea to the CGS algorithm to arrive at the transpose-free quasi-minimal-residual (TFQMR) algorithm.[21] Note that CGS, Bi-CGSTAB, and TFQMR may still encounter algorithm breakdown. The unpreconditioned versions of these three algorithms roughly require the same amount of work and storage per iteration.[21] However, the preconditioned TFQMR algorithm requires twice as many operations of the preconditioner (four vs two) as the preconditioned versions of the other two algorithms. This is because the TFQMR algorithm produces two solution estimates per iteration, whereas CGS and Bi-CGSTAB produce only one solution estimate per iteration.

This paper investigates the use of CGS, TFQMR, Bi-CGSTAB, and GMRES(20) within both standard and matrix-free implementations of inexact Newton's method. The resulting algorithms are used to solve the model problem described next.

Model Problem Description

The model problem considered here is natural convection in an enclosed square cavity as illustrated in Fig. 1. The flow is assumed incompressible. The coupling between the momentum and energy equations occurs through the buoyancy force term in the momentum equation, using the Boussinesq approximation.[23] In conservative and dimensionless form the governing equations can be expressed as follows:

Continuity:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2}$$

Momentum:

$$\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{3}$$

$$\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} = -\frac{\partial p}{\partial y} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + GrT \tag{4}$$

Energy:

$$\frac{\partial uT}{\partial x} + \frac{\partial vT}{\partial y} = \frac{1}{Pr}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) \tag{5}$$

where Gr is the Grashof number, Pr is the Prandtl number ($= 0.71$), and the Rayleigh number Ra is given by $Ra = GrPr$ ($= 10^4$). The gravity vector is assumed pointing in the negative y direction. Boundary conditions for this problem are specified in Fig. 1.

Finite volume differencing using the power law scheme of Patankar[24] is used to discretize these governing equations on a uniform, staggered grid where velocities are located on cell faces and pressures and temperatures are located at cell centers.

Numerical Solution Algorithm

The numerical solution algorithm used here is based on Newton's method. Implementation is simplified using a numerically evaluated Jacobian, while robustness is improved using mesh sequencing and a damped Newton iteration.[5] The linear systems that arise

on each Newton iteration are solved using preconditioned conjugate gradient-like iterative algorithms. Consequently, the resulting solution algorithm is referred to as an "inexact" Newton's method. Both standard and matrix-free implementations of inexact Newton's method are discussed next.

Newton's Method

Newton's method is a robust technique for solving systems of nonlinear equations of the form

$$\boldsymbol{F}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})]^T = 0 \tag{6}$$

where the state variable $\boldsymbol{x}$ can be expressed as

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T \tag{7}$$

Application of Newton's method requires the solution of the linear system,

$$\boldsymbol{J}^n \delta \boldsymbol{x}^n = -\boldsymbol{F}(\boldsymbol{x}^n) \tag{8}$$

where the elements of the Jacobian $\boldsymbol{J}$ are defined by

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{9}$$

and the new solution approximation is obtained from

$$\boldsymbol{x}^{n+1} = \boldsymbol{x}^n + d\delta \boldsymbol{x}^n \tag{10}$$

The constant d in Eq. (10) is used to damp the Newton updates. The damping strategy is designed to prevent the calculation of nonphysical variable values (i.e., negative temperature) and to scale large variable updates when the solution is far from the true solution. However, damping was not necesssary to obtain the results presented here.

The convergence criteria for the Newton iteration is based upon a relative update defined by

$$R_n^o = \max_{\text{all } i} \left[\frac{|\delta x_i^n|}{\max\{|x_i^n|, 1\}} \right] \tag{11}$$

where the superscript on R_n^o refers to the outer Newton iteration and the subscript indicates the dependence on the Newton iteration. Convergence is then assumed when $R_n^o < 1 \times 10^{-6}$. This means that six digits of accuracy are required when the magnitude of the state variable is greater than one, and six decimal places of accuracy are required when the magnitude of the state variable is less than one.

We employ a natural ($uvpT$) ordering of the variables, where the u-momentum equation is solved for the u velocity, the v-momentum equation is solved for the v velocity, the continuity equation is solved for the pressure, and the energy equation is solved for the temperature. Cells are numbered from left to right and then bottom to top. This ordering and our finite volume differencing scheme results in a sparse banded Jacobian matrix. We exploit this sparse banded structure by storing only the nonzero diagonal bands of the Jacobian matrix.

Preconditioning

In this paper, several conjugate gradient-like iterative algorithms are used to approximately solve Eq. (8) on each Newton step, giving rise to an inexact Newton iteration. We employ right preconditioning to improve the performance of the conjugate gradient-like algorithms on each Newton step. The right preconditioned linear system then takes the form

$$\boldsymbol{J}^n \boldsymbol{P}^{-1} \boldsymbol{P} \delta \boldsymbol{x}^n = -\boldsymbol{F}(\boldsymbol{x}^n) \tag{12}$$

so that within the conjugate gradient-like algorithm $\boldsymbol{J}^n$ is replaced with $\boldsymbol{J}^n\boldsymbol{P}^{-1}$ and $\delta \boldsymbol{x}^n$ is replaced with $\boldsymbol{P}\delta \boldsymbol{x}^n$. Effective preconditioning requires that $\boldsymbol{P}$ reasonably approximate $\boldsymbol{J}$ and that systems of the form $\boldsymbol{P}\boldsymbol{w} = \boldsymbol{b}$, which arise within the conjugate gradient-like iteration, can be solved efficiently. Selection of an effective preconditioner is a very important but sometimes difficult task. In this paper the focus is on the performance of the standard vs the matrix-free implementations of inexact Newton's method. Therefore, we restrict our attention to only incomplete lower-upper (ILU) factorization, specifically ILU(0) preconditioning.[14,25] This means that the preconditioning matrix assumes the same nonzero sparsity pattern as the Jacobian matrix. In our implementation, however, we take advantage of the banded structure of our Jacobian matrix and store nonzero diagonals. We assume then that our preconditioner has the same number of nonzero diagonals.

A difficulty, which arises when the continuity equation is solved for pressure, warrants a short discussion with regard to preconditioning. Because pressure does not explicitly appear in this equation, zeros appear on the main diagonal in every row of the Jacobian matrix representing the continuity equation. These zero diagonal entries reduce the number of effective preconditioners that can be derived from the Jacobian matrix. However, incomplete lower-upper (ILU) factorization can still be used as an effective preconditioner in this case because fill-in resulting from the incomplete factorization will generate nonzero entries in all of the diagonal rows except those corresponding to a finite volume with faces adjacent to both left and bottom boundaries.[6] In our model problem, the only cell with faces adjacent to both a left and a bottom boundary is located in the lower left corner. We handle this problem by simply fixing the pressure to zero in that cell, which is justified for this model problem and incompressible flow because pressure is determined only up to an additive constant.[6,26]

Standard Inexact Newton Iteration

In our stantard inexact Newton iteration, the accuracy of the iterative solve is controlled by an inner convergence criteria similar to that proposed by Averick and Ortega[1] and Dembo et al.[2] Specifically, the inner iteration is assumed converged when

$$R_n^i = \frac{\|\boldsymbol{J}^n \delta \boldsymbol{x}^n + \boldsymbol{F}(\boldsymbol{x}^n)\|}{\|\boldsymbol{F}(\boldsymbol{x}^n)\|} < \gamma_n \tag{13}$$

The selection of the best value of γ_n is based upon previous results.[10,27] When starting from a flat initial guess, γ_n is allowed to vary on each Newton iteration according to

$$\gamma_n = \left(\tfrac{1}{2}\right)^{\text{Min}\{n,10\}} \tag{14}$$

When starting from a reasonably good initial guess (i.e., the interpolated solution from a coarser grid), then γ_n is set equal to 10^{-2}.

Matrix-Free Inexact Newton Iteration

The conjugate gradient-like algorithms used to solve Eq. (8) require the Jacobian matrix only in the form of matrix–vector products of the form $\boldsymbol{J}\boldsymbol{w}$, which may be approximated by Eq. (1). Use of this approximation leads to a matrix-free inexact Newton iteration. When preconditioning is employed, $\boldsymbol{J}$ is replaced by $\boldsymbol{J}\boldsymbol{P}^{-1}$, and the finite difference approximation to $\boldsymbol{J}\boldsymbol{P}^{-1}\boldsymbol{w}$ is computed as

$$\boldsymbol{J}\boldsymbol{P}^{-1}\boldsymbol{w} \approx \frac{\boldsymbol{F}(\boldsymbol{x} + \varepsilon \boldsymbol{P}^{-1}\boldsymbol{w}) - \boldsymbol{F}(\boldsymbol{x})}{\varepsilon} \tag{15}$$

Implementation of this approximation is accomplished by replacing matrix-times-a-vector operations within the conjugate gradient-like algorithms with calls to a routine that uses Eq. (15). Thus, each time this routine is called during the inner iteration, the discrete governing equations vector must be evaluated with perturbed values of the state variable. Since these function evaluations can be expensive, the number of required inner iterations should be kept as low as possible. This can be accomplished by limiting the maximum number of inner iterations and by using effective preconditioning.

Equation (15) indicates that the accuracy of the matrix-free approximation is strongly dependent upon the vector $\boldsymbol{w}$. Since this vector changes within the inner conjugate gradient-like iteration, the accuracy of the matrix-free approximation is subject to some

uncertainty. In our matrix-free implementation, the perturbation constant ε is chosen as follows:

$$\varepsilon = \frac{1}{N}\sum_{i=1}^{N}\varepsilon_i \tag{16}$$

where

$$\varepsilon_i = a \cdot x_i + a \tag{17}$$

and where x_i is the ith component of the state vector of dimension N, and a is a perturbation constant whose magnitude is on the order of the square root of computer roundoff. We note for completeness that alternative options exist for determing ε.[8–10] With the matrix-free approximation, the inner iteration is assumed converged when

$$R_n^i = \frac{\|\{[F(x^n + \varepsilon\delta x^n) - F(x^n)]/\varepsilon\} + F(x^n)\|}{\|F(x^n)\|} < \gamma_n \tag{18}$$

where γ_n is defined earlier.

Table 1 Comparison with benchmark solution of De Vahl Davis[28]

	Benchmark solution	Current solution 10 × 10 grid	20 × 20 grid	40 × 40 grid
u_{max}	22.7859	22.3569	22.7304	22.7172
y	0.177	0.15	0.175	0.1875
v_{max}	27.6296	25.3276	27.5293	27.6181
x	0.881	0.85	0.875	0.8875

Results

The advantages and disadvantages of the matrix-free implementation are studied with respect to performance and robustness. The computer memory advantages of the matrix-free implementation are obvious. The potential performance advantage lies in reducing the CPU cost of forming and using the Jacobian matrix, without inhibiting or degrading convergence. Performance is studied using both Lanczos-based iterative solvers (CGS, TFQMR, and Bi-CGSTAB) and an Arnoldi-based iterative solver [GMRES(20)]. Note that in applications with a large number of equations, where the cost of forming the Jacobian matrix is a significant fraction of the total CPU time, the potential advantages of the matrix-free implementation are very appealing.

Performance data are obtained for the steady-state solution of the natural convection test problem with $Ra = 10^4$ and $Pr = 0.71$. All computations were run on an IBM RISC/6000 model 320 workstation. Calculations were initiated from a fat initial guess (i.e., $u = v = 0, T = 0.5$). Selected data from the solution to this problem are compared with the benchmark solution of De Vahl Davis[28] in Table 1. Note that the velocity data from Ref. 28 were multiplied by the factor Pr^{-1} to account for the different choices in velocity scaling. Additionally, the positions were adjusted to account for the reversed circulation direction in Ref. 28. Table 1 presents the maximum horizontal velocity component (u_{max}), its corresponding y location along the line $x = 0.5$, the maximum vertical velocity component (v_{max}), and its x location along the line $y = 0.5$. Data from three different grids of increasing refinement are compared with the benchmark solution. Reasonable agreement is obtained between the benchmark solution and the 40 × 40 grid solution (< 1% difference).

Tables 2–5 present the required number of Newton iterations n, the average number of inner iterations per Newton iteration $\bar{m}$, the required CPU time, and the number of times the maximum inner iteration limit m_{max} was encountered. The maximum inner iteration

Table 2 Comparison of standard and matrix-free implementations on a 10 × 10 grid ($m_{max} = 20$)

Iterative solver	Standard implementation n	$\bar{m}$	CPU time, s	m_{max}, hits	Matrix-free implementation n	$\bar{m}$	CPU time, s	m_{max}, hits
CGS	7	7	2.3	0	9	9	6.4	2
TFQMR	8	7	2.9	0	8	12	7.2	3
Bi-CGSTAB	8	7	2.5	0	7	8	4.4	1
GMRES(20)	8	8	2.6	0	8	8	3.3	0

Table 3 Comparison of standard and matrix-free implementations on a 20 × 20 grid ($m_{max} = 40$)

Iterative solver	Standard implementation n	$\bar{m}$	CPU time, s	m_{max}, hits	Matrix-free implementation n	$\bar{m}$	CPU time, s	m_{max}, hits
CGS	8	17	16.7	0	—	—	—	—
TFQMR	8	21	24.4	0	10	34	85.4	6
Bi-CGSTAB	8	18	17.3	0	10	27	60.3	4
GMRES(20)	9	25	16.3	3	9	25	26.3	3

Table 4 Comparison of standard and matrix-free implementations on a 40 × 40 grid ($m_{max} = 80$)

Iterative solver	Standard implementation n	$\bar{m}$	CPU time, s	m_{max}, hits	Matrix-free implementation n	$\bar{m}$	CPU time, s	m_{max}, hits
CGS	9	55	188.2	3	—	—	—	—
TFQMR	10	61	305.7	4	58	79	4313.4	57
Bi-CGSTAB	13	69	322.0	8	—	—	—	—
GMRES(20)	19	74	320.0	16	21	74	631.2	18

Table 5 Comparison of standard and matrix-free implementations on a 40 × 40 grid using a 10 × 10, 20 × 20, and 40 × 40 mesh sequence (m_{max} = 20, 40, and 80, respectively)

Iterative solver	Standard implementation				Matrix-free implementation			
	n	$\bar{m}$	CPU time, s	m_{max}, hits	n	$\bar{m}$	CPU time, s	m_{max}, hits
CGS	5	62	131.7	0	—	—	—	—
TFQMR	5	70	195.1	3	29	80	2249.9	29
Bi-CGSTAB	7	71	196.6	4	28	79	1841.1	26
GMRES(20)	15	80	290.9	15	16	80	538.7	16

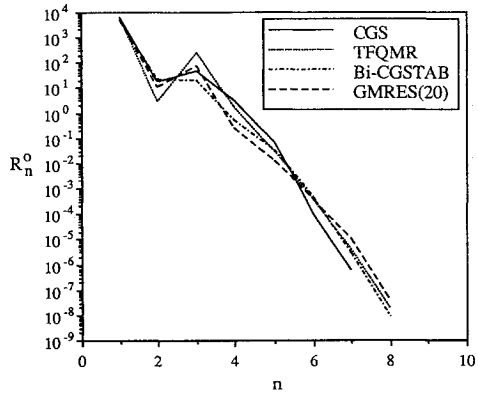

Fig. 2 Standard inexact Newton iteration convergence behavior (10 ×10 grid).

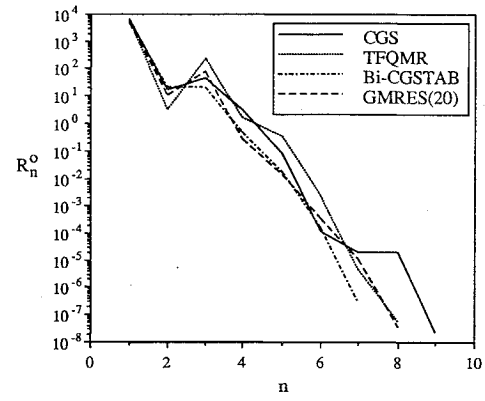

Fig. 3 Matrix-free inexact Newton iteration convergence behavior (10 × 10 grid).

limit was set equal to the square root of the number of unknowns. Note that the implementation used to generate these results is not practical in the sense that a new ILU(0) preconditioner was formed each Newton iteration, which in turn requires the formation of the Jacobian matrix. A more practical implementation might use the same preconditioner for several Newton iterations or a less expensive preconditioner that does not require forming the complete Jacobian matrix. However, the purpose of this article is to investigate the effects of the matrix-free approximation. With this goal in mind, a more practical implementation was not necessary, and so ILU(0) was selected as the only preconditioner. Consequently, CPU times should not be used as a basis for comparing the two implementations but rather as a basis for comparing the performance of the different iterative solvers. Convergence behavior is used as a basis of comparison for the standard and matrix-free implementations. Note that the TFQMR algorithm provides an upper bound for the residual norm that was not used in this study. Use of this upper bound could make the TFQMR algorithm less expensive because the calculation of the residual norm could be postponed until this upper bound was small enough. In our implementation, however, we chose to compute the residual norm on each iteration to provide a more accurate convergence check as well as to provide information regarding convergence history.

Table 2 presents performance data for a coarse 10 × 10 grid solution for each of the inner iteration algorithms. Corresponding convergence plots are shown for both the standard and the matrix-free implementations in Figs. 2 and 3, respectively. These figures plot the maximum relative Newton update R_n^o as a function of the Newton iteration count. The performance of the different iterative solvers is similar for both the standard implementation and the matrix-free implementations for this coarse grid.

Tables 3 and 4 investigate the effect of grid refinement. Use of the Lanczos-based iterative algorithms with the matrix-free approximation led to a marked degradation in performance as the grid was refined, whereas the Arnoldi-based method (GMRES) performed similarly for both implementations. In fact, for the 40 × 40 grid (Table 4) no solutions were obtained using the matrix-free approximation with CGS or Bi-CGSTAB. The use of mesh sequencing in Table 5 led to a solution using Bi-CGSTAB but still did not enable a solution using CGS.

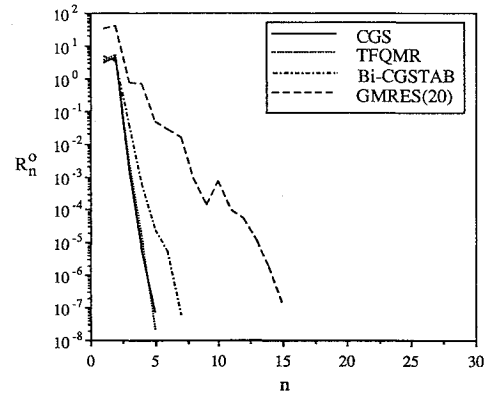

Fig. 4 Standard inexact Newton iteration convergence behavior (40 ×40 grid).

Convergence plots for the 40 × 40 grid solutions in Table 5 are shown in Figs. 4 and 5. Figure 6 is a plot of R_n^i vs the inner iteration number for the first Newton iteration on the 40 × 40 grid corresponding to the standard implementation solutions in Table 5. Note that the first iteration was chosen because then each of the iterative algorithms are roughly solving the same linear system. Note from Fig. 4, the relatively slow convergence obtained using GMRES(20) for the standard implementation. This behavior follows from the choice for the dimension of our Krylov subspace (k = 20). Twenty iterations was not sufficient to satisfy the inner iteration convergence criteria [Eq. (13)]. This necessitated periodic algorithm restarts, which in turn slowed the convergence of the GMRES algorithm. This observation is evidenced by the large number of m_{max} hits encountered in Tables 4 and 5 and the convergence-flattening trend shown in Fig. 6 for the GMRES(20) curve. GMRES(20) convergence is excellent during the first 20 iterations but thereafter begins to flatten or stall as more periodic algorithm restarts are needed. Increasing the dimension of the Krylov subspace would improve performance, but it would also further increase algorithm memory requirements.

Several additional observations can be gleaned from Fig. 6. Notice the rather erratic convergence behavior of the CGS algorithm

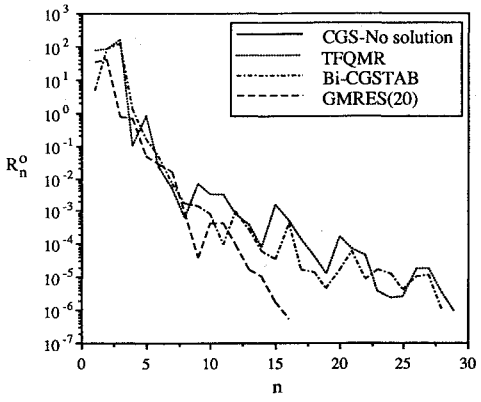

Fig. 5 Matrix-free inexact Newton iteration convergence behavior (40 × 40 grid).

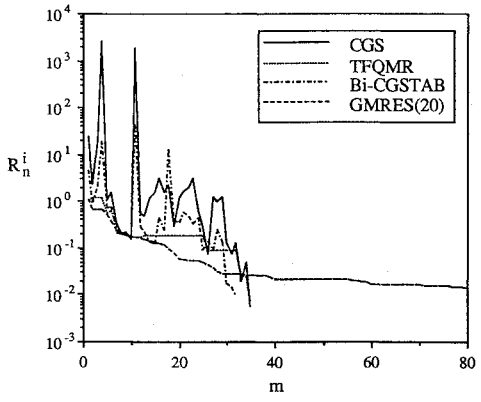

Fig. 6 Inner iteration convergence on the first standard Newton iteration (40 × 40 grid).

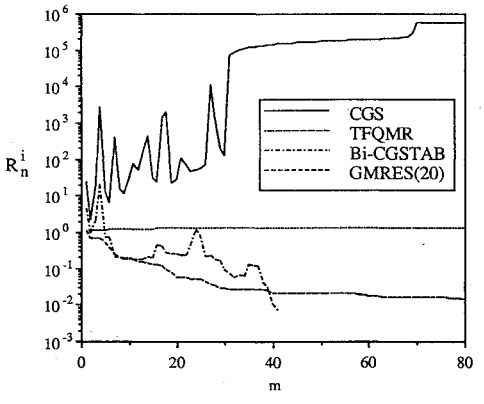

Fig. 7 Inner iteration convergence on the first matrix-free Newton iteration (40 × 40 grid).

that was alluded to previously. In spite of this behavior, the tabulated data show that when the CGS algorithm converged it was more CPU efficient than the other algorithms. Figure 6 shows that, for this problem, TFQMR is more successful than Bi-CGSTAB at controlling the erratic CGS convergence behavior but at a noticeably higher CPU cost as can be seen in the tabulated results. The TFQMR convergence curve tends to temporarily stall or flatten out when CGS is displaying very erratic convergence behavior. The Bi-CGSTAB convergence curve, although more controlled than the CGS curve, still exhibits some erratic convergence behavior.

These observations may lend some insight into the relatively poor performance of matrix-free implementation when the Lanczos-based methods are used. This performance is illustrated in Fig. 5, which once again corresponds to the solutions presented in Table 5. Recall that no solution was obtained using the CGS algorithm with the matrix-free approximation. In the case of GMRES(20), replacing the standard implementation with the matrix-free approximation resulted in nearly identical convergence behavior. This suggests that Eq. (1) yielded acceptable approximations for the needed matrix–vector products. In contrast, the convergence behavior using TFQMR and Bi-CGSTAB degraded appreciably when the standard implementation was replaced with the matrix-free approximation. In an attempt to relate this behavior to the observations cited earlier, consider the convergence behavior of the iterative solvers as shown in Fig. 7 for the first Newton iteration. The erratic convergence behavior of CGS coupled with the use of Eq. (15) results in very poor approximations for the needed matrix–vector products. The CGS algorithm could not recover from the erratic jumps and eventually returned a very bad Newton update that led to divergence. Once again, the TFQMR algorithm is observed to stall out when the CGS iteration is behaving badly. During this Newton iteration, TFQMR stalls with a value of R_n^i near 1. This behavior, although resulting in poor convergence, does not cause divergence of the algorithm. During this first Newton iteration it is fortuitous that Bi-CGSTAB converged, because during later iterations it most often encountered the $m_{\max}$ limit as shown in Table 5. In addition, note that a solution could not be obtained using the matrix-free approximation with Bi-CGSTAB on the 40 × 40 grid starting from a flat initial guess. In that case, behavior similar to that of CGS over several Newton iterations led to divergence.

Recall that the accuracy of the matrix-free approximation in Eq. (1) is dependent upon the vector w. In the case of the Lanczos-based algorithms, the characteristics of this vector may vary wildly as evidenced by the sometimes erratic CGS convergence behavior. In the case of GMRES, however, an orthonormal basis is constructed for the Krylov subspace so that only normalized vectors appear in matrix–vector products. Presumably, this is one feature that enables Eq. (1) to generate acceptable approximations for the required matrix–vector products needed within the GMRES algorithm.

Summary and Conclusions

Inexact Newton algorithms were used to solve the well-known problem of natural convection in an enclosed square cavity, which is assumed governed by the incompressible Navier–Stokes and energy equations. Coupling between the momentum and energy equations occurred through the buoyancy force terms in the momentum equations using the Boussinesq approximation.[23] These equations were solved with $Ra = 10^4$ and $Pr = 0.71$ on several staggered, finite volume grids of increasing refinement. Several conjugate gradient-like algorithms were selected from a class of algorithms based upon the Lanczos biorthogonalization process. These included CGS, TFQMR, and Bi-CGSTAB. A fourth algorithm was based upon the Arnoldi process, namely the restarted GMRES algorithm. We chose the dimension of the Krylov subspace for the restarted GMRES algorithm to be 20. Right ILU(0) preconditioning was used to improve the performance of the iterative solvers. Both standard and matrix-free implementations were investigated.

In general, GMRES(20) outperformed the Lanczos-based methods when the matrix-free approximation was used. GMRES was able to maintain an acceptable level of performance when the standard implementation was replaced with the matrix-free approximation. In contrast, the Lanczos-based methods were not able to maintain the same level of performance. Among these methods, CGS was found to be poorly suited to matrix-free implementations of inexact Newton's method because of its erratic convergence behavior. TFQMR and Bi-CGSTAB performed better than CGS because of their smoother convergence behavior but still suffered a notable drop in performance when the matrix-free approximation was used.

Standard implementations using the Lanczos-based algorithms seemed to outperform the standard implementation using GMRES(20) when the grid was refined (number of unknowns increased). Convergence of the GMRES(20) algorithm was not insured within 20 iterations, the selected dimension of the Krylov subspace. Consequently, periodic algorithm restarts were necessary, leading to slower convergence. The GMRES(20) iteration frequently encountered the upper limit for the number of inner iterations on the finest grid. This resulted in the return of mediocre Newton updates and slower convergence of the outer Newton iteration compared with the use of the Lanczos-based methods.

Acknowledgments

This work was supported through the INEL Long Term Research Initiative in Computational Mechanics under Department of Energy Idaho Field Office Contract DE-AC07-94ID13223. The authors thank Peter Brown for suggesting the use of SPARSKIT[29] and Youcef Saad for granting permission to use the GMRES algorithm from that package. The suggestions and comments made by the referees were very helpful and are greatly appreciated.

References

[1]Averick, B. M., and Ortega, J. M., "Solutions of Nonlinear Poisson-Type Equations," *Applied Numerical Mathematics*, Vol. 8, No. 6, 1991, pp. 443–455.

[2]Dembo, R. S., Eisenstat, S. C., and Steihaug, T., "Inexact Newton Methods," *SIAM Journal of Numerical Analysis*, Vol. 9, No. 2, 1982, pp. 400–408.

[3]Einset, E. O., and Jensen, K. F., "A Finite Element Solution of Three-Dimensional Mixed Convection Gas Flows in Horizontal Channels Using Preconditioned Iterative Matrix Methods," *International Journal of Numerical Methods in Fluids*, Vol. 14, No. 7, 1992, pp. 817–841.

[4]Dahl, O., and Wille, S. O., "An ILU Preconditioner with Couple Node Fill-In for Iterative Solution of the Mixed Finite Element Formulation of the 2D and 3D Navier–Stokes Equations," *International Journal of Numerical Methods in Fluids*, Vol. 15, No. 5, 1992, pp. 525–544.

[5]McHugh, P. R., and Knoll, D. A., "Fully Implicit Solution of the Benchmark Backward Facing Step Problem Using Finite Volume Differencing and Inexact Newton's Method," *Benchmark Problems For Heat Transfer Codes*, edited by B. Blackwell, and D. W. Pepper, ASME HTD-Vol. 222, American Society of Mechanical Engineers, New York, Nov. 1992, pp. 77–87.

[6]Chin, P., D'Azevedo, E. F., Forsyth, P. A., and Tang, W.-P., "Preconditioned Conjugate Gradient Methods for the Incompressible Navier–Stokes Equations," *International Journal of Numerical Methods in Fluids*, Vol. 15, No. 3, 1992, pp. 273–295.

[7]Ern, A., Giovangigli, V., Keyes, D., and Smooke, M. D., "Towards Polyalgorithmic Linear System Solvers for Nonlinear Elliptic Problems," *SIAM Journal of Scientific Computing*, Vol. 15, No. 3, 1994, pp. 681–703.

[8]Gear, C. W., and Saad, Y., "Iterative Solution of Linear Equations in ODE Codes," *SIAM Journal of Scientific Statistical Computation*, Vol. 4, 1983, pp. 583–601.

[9]Brown, P. N., and Hindmarsh, A. C., "Matrix-Free Methods for Stiff Systems of ODE's," *SIAM Journal of Numerical Analysis*, Vol. 23, No. 3, 1986, pp. 610–638.

[10]Brown, P. N., and Saad, Y., "Hybrid Krylov Methods for Nonlinear Systems of Equations," *SIAM Journal of Scientific Statistical Computation*, Vol. 11, No. 3, 1990, pp. 450–481.

[11]Saad, Y., and Schultz, M. H., "Conjugate Gradient-Like Algorithms for Solving Nonsymmetric Linear Systems," *Mathematics of Computation*, Vol. 44, N170, 1985, pp. 417–424.

[12]Ashby, S., Manteuffel, T., and Saylor, P., "A Taxonomy for Conjugate Gradient Methods," *SIAM Journal of Numerical Analysis*, Vol. 27, No. 6, 1990, pp. 1542–1568.

[13]Faber, V., and Manteuffel, T., "Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method," *SIAM Journal of Numerical Analysis*, Vol. 21, No. 2, 1984, p. 352.

[14]Ashby, S., Manteuffel, T., and Saylor, P., *Preconditioned Polynomial Iterative Methods, A Tutorial*, Univ. of Colorado, Denver, CO, April 1992.

[15]Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific Statistical Computation*, Vol. 7, No. 7, 1986, p. 856.

[16]Freund, R. W., Golub, G. H., and Nachtigal, N. M., "Iterative Solution of Linear Systems," Numerical Analysis Project, Computer Science Dept., Stanford Univ., Manuscript NA-91-05, Stanford, CA, Nov. 1991.

[17]Lanczos, C., "Solution of Systems of Linear Equations by Minimized Iterations," *Journal of Research of the National Bureau of Standards*, Vol. 49, 1952, pp. 33–53.

[18]Fletcher, R., "Conjugate Gradient Methods for Indefinite Systems," *Proceedings of the Dundee Conference on Numerical Analysis*, edited by G. A. Watson, Lecture Notes in Mathematics 506, Springer-Verlag, Berlin, 1975, pp. 73–89.

[19]Van der Vorst, H. A., "Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems," *SIAM Journal of Scientific Statistical Computation*, Vol. 13, No. 2, 1992, pp. 631–644.

[20]Sonneveld, P., "CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems," *SIAM Journal of Scientific Statistical Computation*, Vol. 10, No. 1, 1989, pp. 36–52.

[21]Freund, R. W., "A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems," *SIAM Journal of Scientific Computing*, Vol. 14, No. 2, 1993, pp. 470–482.

[22]Tong, C. H., "A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems," Sandia National Lab., SAND91-8240, UC-404, Livermore, CA, Jan. 1992.

[23]Landau, L. D., and Lifshitz, E. M., *Fluid Mechanics*, 2nd ed., Pergamon, Elmsford, N.Y., Vol. 6, 1987.

[24]Patankar, S. V., *Numerical Heat Transfer and Fluid Flow*, Hemisphere, New York, 1980, pp. 90–100.

[25]Sangback, M., and Chronopoulos, A. T., "Implementation of Iterative Methods for Large Sparse Nonsymmetric Linear Systems on a Parallel Vector Machine," *International Journal of Supercomputer Applications*, Vol. 4, No. 4, 1990, pp. 9–24.

[26]Gresho, P. M., "Some Current CFD Issues Relevant to the Incompressible Navier–Stokes Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 87, Nos. 2–3, 1991, pp. 201–252.

[27]McHugh, P. R., and Knoll, D. A., "Fully Coupled Finite Volume Solutions of the Incompressible Navier–Stokes and Energy Equations Using an Inexact Newton Method," *International Journal of Numerical Methods in Fluids* (to be published).

[28]De Vahl Davis, G., "Natural Convection of Air in a Square Cavity: A Benchmark Numerical Solution," *International Journal of Numerical Methods in Fluids*, Vol. 3, No. 2, 1983, pp. 249–264.

[29]Saad, Y., "SPARSKIT, A Basic Tool Kit for Sparse Matrix Computations," RIACS TR 90.20, 1991.