SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 398–418

A MULTIGRID-PRECONDITIONED NEWTON–KRYLOV METHOD FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS*

M. PERNICE† AND M. D. TOCCI‡

Abstract. Globalized inexact Newton methods are well suited for solving large-scale systems of nonlinear equations. When combined with a Krylov iterative method, an explicit Jacobian is never needed, and the resulting matrix-free Newton–Krylov method greatly simplifies application of the method to complex problems. Despite asymptotically superlinear rates of convergence, the overall efficiency of a Newton–Krylov solver is determined by the preconditioner. High-quality preconditioners can be constructed from methods that incorporate problem-specific information, and for the incompressible Navier–Stokes equations, classical pressure-correction methods such as SIMPLE and SIMPLER fulfill this requirement. A preconditioner is constructed by using these pressure-correction methods as smoothers in a linear multigrid procedure. The effectiveness of the resulting Newton–Krylov-multigrid method is demonstrated on benchmark incompressible flow problems.

Key words. Newton–Krylov methods, multigrid preconditioning, pressure-correction smoothers

AMS subject classifications. 65H10, 65F10, 65N55

PII. S1064827500372250

1. Introduction.

Efficient solution of the steady-state incompressible Navier–Stokes equations

$$\begin{aligned} (uu)_x + (uv)_y - \tfrac{1}{Re}\Delta u + p_x &= f_1, \\ (uv)_x + (vv)_y - \tfrac{1}{Re}\Delta v + p_y &= f_2, \\ u_x + v_y &= 0, \end{aligned} \tag{1.1}$$

where Re is the Reynolds number, has been a problem of central importance in computational science and engineering since its inception. Methods for solving (1.1) can also be leveraged to solve the systems of nonlinear equations that arise when an implicit method is used to solve the unsteady incompressible Navier–Stokes equations. Early efforts were constrained by limited memory capacity and were often based on strategies that involved solving a series of simplified and lower-dimensional problems. The simplifications generally arose from application of operator-splitting strategies, and solutions of individual equations were often built from iterative methods based on line solves. Pressure-correction algorithms such as SIMPLE [29] and SIMPLER [28] are typical of these approaches. While frugal in their use of memory, these methods lack theoretical foundations and converge slowly. Moreover, these convergence rates degrade with increasing problem size. Despite these drawbacks, pressure-correction methods are still in widespread use in production engineering

*Received by the editors May 18, 2000; accepted for publication (in revised form) November 29, 2000; published electronically July 10, 2001. This work was supported by the University of Utah Center for the Simulation of Accidental Fires and Explosions and was funded by the Department of Energy, Lawrence Livermore National Laboratory, under subcontract B341493.
http://www.siam.org/journals/sisc/23-2/37225.html
†Computer and Computational Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (pernice@lanl.gov). This author was also supported by Reaction Engineering International, Salt Lake City, UT, through funding from the Department of Energy Office of Energy Research under grant DE–FG03–96ER82268, and by the University of Utah Center for High Performance Computing.
‡The MathWorks, Inc., Natick, MA 01760 (mtocci@mathworks.com).

codes. Consequently, there is great interest in exploring strategies for accelerating the convergence of pressure-correction methods while minimizing changes in software and data structures.

Recent decades have seen considerable progress in the development of new approaches for solving large-scale nonlinear problems, in particular, globalized inexact Newton methods [11, 12, 13]. The major shortcoming of classical Newton methods is the need to solve a large-scale system of linear equations at each iteration. Inexact Newton methods address this by using an iterative method to solve this system of equations to less than full accuracy. Inexact Newton methods can be considerably more expensive than the classical pressure-correction techniques, but rapid increases in both the speed and capacity of computing resources make it practical to consider inexact Newton methods for incompressible flow problems. At the same time, the desire to address new classes of problems sometimes makes it necessary to consider these methods.

Since their introduction, inexact Newton methods have been successfully applied to a large variety of nonlinear problems: integral equation descriptions of equilibrium states of liquid surfaces [3], multiphase flow in porous media [9, 43], radiation-diffusion problems [32, 27, 5], reacting flows [20, 21, 35], aerodynamics calculations [7, 6], and incompressible flow problems [25, 19, 22, 23]. While by no means exhaustive, these applications represent a cross-section of preconditioning strategies currently in use. Incomplete LU (ILU) factorizations [26, 33] are popular choices, but they require information about the Jacobian that may be difficult to determine in a matrix-free inexact Newton method. Depending on the amount of element fill-in that is allowed, ILU preconditioners can have high storage requirements. Calculating an ILU preconditioner can also be computationally expensive, and this cost is multiplied by the number of times the preconditioner is updated during the nonlinear solution process.

A notable trend reflected in this list of applications of Newton–Krylov methods is the use of multigrid methods as preconditioners [43, 32, 5, 22, 23]. While multigrid methods are highly efficient solvers on their own, they also serve as excellent preconditioners, and their use in this context makes the performance and robustness of the multigrid method less sensitive to the selection of components such as intergrid transfers and coarse grid solvers. Multigrid methods can be tailored to specific problems by selection of an appropriate smoother; it has been known for some time that SIMPLE can be used as a multigrid smoother [36, 37], and it has recently been demonstrated that SIMPLER can also be used as a smoother in a multigrid procedure [30]. In contrast to prior work on Newton–Krylov-multigrid (NK-MG) methods [22, 23], pressure-correction methods work directly on the primitive variable formulation (1.1) and so are readily extendible to problems in three dimensions. Also in contrast to prior work applying Newton–Krylov methods to incompressible flow problems [25], a multigrid preconditioner represents a considerable savings in storage and setup time over an ILU-based preconditioner. The combination of an NK-MG method with a pressure-correction smoother may also be regarded as a means for accelerating the convergence of the pressure-correction scheme; however, patterns of use of the pressure-correction methods must be reexamined in order to use them effectively in this context.

This paper is organized as follows. First, some notation is introduced. Following this, the relevant algorithmic components are described in section 3. The effectiveness of combining Newton–Krylov methods with multigrid preconditioners equipped with

pressure-correction smoothers is examined in section 4. These results are summarized and some conclusions are drawn in section 5.

2. Notation. Pressure-correction methods have traditionally been developed for staggered grid discretizations [29, 28], and that convention is adhered to in this work. In this arrangement of variables, originally presented in [18], the domain is divided into a number of cells, with the horizontal component of velocity centered on vertical cell faces, vertical components of velocity centered on horizontal cell faces, and pressure located at cell centers. A second-order centered discretization of the Navier–Stokes equations (1.1) on a staggered grid produces a set of nonlinear equations $F(\mathbf{u}, p) = 0$, which may be written in block matrix form as

$$F(\mathbf{u}, p) = \mathcal{Q}[\mathbf{u}] \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{2.1}$$

where

$$\mathcal{Q}[\mathbf{u}] = \begin{pmatrix} \mathbf{Q}[\mathbf{u}] & \nabla^h \\ \nabla^h \cdot & 0 \end{pmatrix}.$$

In this,

$$\mathbf{Q}[\mathbf{u}] = \begin{pmatrix} Q_1[\mathbf{u}] & 0 \\ 0 & Q_2[\mathbf{u}] \end{pmatrix} \tag{2.2}$$

is the discrete momentum operator, ∇^h is a discrete gradient operator, $\nabla^h \cdot$ is a discrete divergence operator,

$$\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix}$$

is the discrete velocity field, and

$$\mathbf{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}$$

represents the body force.

Implicit methods for solving the time dependent version of the Navier–Stokes equations can be accommodated with a few minor modifications. If a backward Euler method is used to discretize the time derivative, the discrete momentum operator becomes

$$\mathbf{Q}[\mathbf{u}] = \begin{pmatrix} Q_1[\mathbf{u}] + u/\Delta t & 0 \\ 0 & Q_2[\mathbf{u}] + v/\Delta t \end{pmatrix},$$

and the set of nonlinear equations to be solved at each time step is

$$F(\mathbf{u}, p) = \mathcal{Q}[\mathbf{u}] \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} - \frac{1}{\Delta t} \begin{pmatrix} \mathbf{u}^n \\ 0 \end{pmatrix} - \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix},$$

where $\mathbf{u}^n$ is the velocity at the previous time step.

3. Algorithms. This section describes the basic ingredients that are used to construct the NK-MG method. The discussion begins with a description of the inexact Newton method that will be used. This is followed by a discussion of classical pressure-correction methods and their use as smoothers in the multigrid preconditioner.

3.1. Inexact Newton methods. Newton's method for solving a system of nonlinear equations

$$F(x) = 0$$

requires, at the kth step, the solution of the linear *Newton equation*

$$F'(x_k)\, s_k = -F(x_k), \tag{3.1}$$

where x_k is the current approximate solution and F' is the Jacobian matrix of the system. Once the Newton step s_k is determined, the current approximation is updated via

$$x_{k+1} = x_k + s_k.$$

This process is continued until a satisfactory solution is found, which is usually judged by making $\|F(x_k)\|$ or $\|s_k\|$ (or both) sufficiently small. Traditionally, Newton's method was considered to be inappropriate for the solution of large-scale systems of nonlinear equations because of the high computational and storage costs of solving (3.1). However, Newton's method will still converge even if (3.1) is not solved exactly; indeed, under some circumstances an exact solution of (3.1) is undesirable. This can be seen by noting that (3.1) is obtained from a linearization of $F(x_k) = 0$ around x_k, which is valid only in a neighborhood of x_k. If the solution of (3.1) produces an s_k that is too large, there may be poor agreement between F and its local linear model, and the effort expended to accurately compute s_k may be wasted. Examples that illustrate this behavior appear in [13, 35].

Newton iterative methods relax the requirement to solve (3.1) exactly to an *inexact Newton condition* [11]

$$\|F(x_k) + F'(x_k)\, s_k\| \le \eta_k \|F(x_k)\|, \tag{3.2}$$

in which the "forcing term" $\eta_k \in (0, 1)$ can be specified either statically or dynamically [13] to enhance efficiency and convergence. The optimal choice of η_k is somewhat problem-specific; see, for example, [35] for some comparative studies. Nevertheless, superlinear and even quadratic convergence of the inexact Newton method can be obtained under certain choices of the forcing terms [11, 13]. In this work, an initial η_k is dynamically determined using a slight modification of the strategy from [13] labeled Choice 1. There, safeguards that prevent η_k from getting too small too quickly are disabled once η_k drops below a certain threshold. In this work, these safeguards are retained throughout the computation.

There are many ways to compute an inexact Newton step s_k that satisfies (3.2), and the efficiency of an inexact Newton method is strongly affected by this choice. Krylov subspace methods [15] are especially well suited for this purpose since they require only matrix-vector products $F'(x_k)v$. This further specialization of inexact Newton methods leads to the class of methods referred to as *Newton–Krylov methods*. The matrix-vector products needed in a Newton–Krylov method may be approximated with finite differences of function values

$$F'(x_k)v \approx \frac{F(x_k + \epsilon v) - F(x_k)}{\epsilon}, \tag{3.3}$$

and so the Jacobian F' never needs to be explicitly formed. Because the Jacobian matrix is neither formed nor stored, this approach is frequently referred to as a

matrix-free Newton–Krylov method. While this greatly reduces storage requirements and simplifies implementation, the differencing parameter ϵ must be chosen carefully. Furthermore, some information about the Jacobian is still needed to construct a preconditioner. Finally, a matrix-free Newton–Krylov method generally requires more nonlinear iterations than a Newton–Krylov method that uses the Jacobian directly.

Among Krylov subspace methods, GMRES [34] is generally preferred, since it minimizes the residual at every iteration. Unfortunately, in order to enforce this, the work and storage requirements per iteration grow linearly with the number of iterations. In practice, this is dealt with by restarting the method, which can potentially slow down convergence or even cause divergence if restarting is done too frequently. Alternatives such as BiCGSTAB [38] and transpose-free QMR [14] can be used; however, they do not share the minimum residual property and, when used in a matrix-free Newton–Krylov method, are generally not as robust as GMRES [24], provided a sufficiently large restart value is used.

Finally, another traditional objection to using Newton's method for large-scale problems is the need to find a good initial approximation x_0. Newton's method (and its inexact counterpart) can fail to converge if x_0 is not chosen carefully. Fortunately, classical strategies for improving the likelihood of convergence from a poor initial approximation also apply to inexact Newton methods [12]. The backtracking globalization strategy given in Algorithm Inexact Newton Backtracking (INB) from [12] is employed in this work.

ALGORITHM 3.1 (INB [12]).
Let x_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*

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

  While $\|F(x_k + s_k)\| > [1 - t(1 - \eta_k)]\, \|F(x_k)\|$ *do:*
    Choose $\theta \in [\theta_{\min}, \theta_{\max}]$.
    Update $s_k \longleftarrow \theta s_k$ *and* $\eta_k \longleftarrow 1 - \theta(1 - \eta_k)$.
  Set $x_{k+1} = x_k + s_k$.

In this, backtracking is invoked when the trial Newton iterate $x_k + s_k$ fails to adequately reduce $\|F\|$. This is determined by comparing the *actual reduction* in $\|F\|$,

$$\|F(x_k)\| - \|F(x_k + s_k)\|,$$

with the *predicted reduction*

$$\|F(x_k)\| - \eta_k \|F(x_k)\| = (1 - \eta_k)\|F(x_k)\|.$$

If the actual reduction exceeds some fraction t of the predicted reduction, then the step is accepted. The value $t = 10^{-4}$ is used to judge sufficient reduction, leading to acceptance of a step that produces even minimal reduction in $\|F\|$. If this criterion is not met, the step is damped by a factor θ that is chosen to minimize a quadratic that interpolates $\|F\|$ in the direction of s_k. Backtracking safeguard values $\theta_{\min}$ and $\theta_{\max}$ provide bounds on the step reduction and are given by 0.1 and 0.5, respectively. Once the Newton step is damped, the predicted reduction in $\|F\|$ is then approximated by noting

$$\begin{aligned}\|F(x_k+\theta s_k)\| &\approx \|F(x_k)+\theta F'(x_k)s_k\| \\ &\le \theta\|F(x_k)+F'(x_k)s_k\|+(1-\theta)\|F(x_k)\| \\ &\le (1-\theta(1-\eta_k))\|F(x_k)\|.\end{aligned}$$

This leads to the adjustment in the forcing term indicated in Algorithm INB, which is used either in the next pass through the backtracking loop or as an initial value in the selection of the forcing term for the next Newton iteration.

3.2. Pressure-correction methods. The pressure-correction methods SIMPLE and SIMPLER can be used to incorporate information that is specific to the incompressible Navier–Stokes equations into the preconditioner. These methods are described in this section. In the following, the dependence of the momentum transport operator (2.2) on $\mathbf{u}$ is suppressed.

3.2.1. SIMPLE. The SIMPLE algorithm, introduced in [29], begins by approximately solving the momentum equations, and then uses the discretized form of the continuity equation to derive an equation whose solution is used both to update the pressure field and to correct the velocity field so that mass is conserved. To begin with, the discrete momentum transport operator $\mathbf{Q}$ is updated to reflect the current approximate solution $\mathbf{u}^{(n)}$ to the momentum equations. The resulting linearized momentum equations can then be solved to determine an intermediate velocity field $\mathbf{u}^{(n+\frac{1}{2})}$ using the current approximate pressure field $p^{(n)}$:

$$\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} = \mathbf{f} - \nabla^h p^{(n)}. \tag{3.4}$$

The resulting velocity field $\mathbf{u}^{(n+\frac{1}{2})}$ does not satisfy the continuity equation, and the residual of this equation is used to compute a correction δp to the pressure field whose gradient is also used to correct $\mathbf{u}^{(n+\frac{1}{2})}$.

To derive an equation for δp, let

$$\mathbf{D} = \operatorname{diag}\mathbf{Q}, \tag{3.5}$$

and introduce the approximations

$$\begin{array}{rcccl}\mathbf{u}^{(n+\frac{1}{2})} & \approx & \mathbf{D}^{-1}\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} & = & \mathbf{D}^{-1}(\mathbf{f}-\nabla^h p^{(n)}), \\ \mathbf{u}^{(n+1)} & \approx & \mathbf{D}^{-1}\mathbf{Q}\mathbf{u}^{(n+1)} & = & \mathbf{D}^{-1}(\mathbf{f}-\nabla^h p^{(n+1)}).\end{array} \tag{3.6}$$

Subtracting these equations and changing the approximation to equality leads to

$$\delta\mathbf{u} \equiv \mathbf{u}^{(n+1)} - \mathbf{u}^{(n+\frac{1}{2})} = -\mathbf{D}^{-1}\nabla^h \delta p, \tag{3.7}$$

where $\delta p = p^{(n+1)} - p^{(n)}$. Applying the discrete divergence operator to these equations gives

$$\nabla^h \cdot (\mathbf{u}^{(n+1)} - \mathbf{u}^{(n+\frac{1}{2})}) = \nabla^h \cdot \mathbf{D}^{-1}\nabla^h \delta p.$$

Finally, requiring $\nabla^h \cdot \mathbf{u}^{(n+1)} = 0$ leads to

$$S\delta p = \nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})}, \tag{3.8}$$

where

$$S = -\nabla^h \cdot \mathbf{D}^{-1}\nabla^h \tag{3.9}$$

is symmetric. Equation (3.8) is a generalized Poisson equation which must be solved for the pressure correction δp. Since the intended correction (3.7) should be 0 at locations where the velocity field is specified, (3.8) is supplemented with homogeneous boundary conditions at these locations. In particular, for problems where the velocity field is specified at the boundaries, δp is determined only up to an additive constant, and S is positive semidefinite. Summarizing, we have the following algorithm.

ALGORITHM 3.2 (SIMPLE).

Determine $\mathbf{u}^{(n+\frac{1}{2})}$ *by solving* (3.4).

Find the pressure correction δp *from* (3.8).

Calculate the velocity corrections $\delta\mathbf{u}$ *using* (3.7).

Update the pressure

$$p^{(n+1)} = p^{(n)} + \delta p$$

and the velocities

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n+\frac{1}{2})} + \delta\mathbf{u}.$$

Remark 3.1. In practice the pressure update is damped by a factor $\alpha \in (0, 1]$:

$$p^{(n+1)} = p^{(n)} + \alpha\delta p.$$

Remark 3.2. SIMPLEC is a variation that replaces the entries in the diagonal matrices D_i with absolute rowsums from Q_i [39]. This variation is used in this work.

Remark 3.3. Observe that

$$\begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \begin{pmatrix} \mathbb{I} & -\mathbf{D}^{-1}\nabla^h \\ 0 & \alpha\mathbb{I} \end{pmatrix} \begin{pmatrix} \mathbf{Q} & 0 \\ \nabla^h\cdot & S \end{pmatrix}^{-1} \mathbf{r}^{(n)}, \tag{3.10}$$

where $\mathbb{I}$ is an identity operator of the appropriate dimension and

$$\mathbf{r}^{(n)} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} - \mathcal{Q} \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix}$$

is the residual of the system. This representation can be derived by employing the notion of transforming smoothers; see [42, 41].

Remark 3.4. When SIMPLE is used as a solver, the inversion in (3.10) is not computed exactly. This practice is followed when using SIMPLE as a smoother: the momentum equations are approximately solved with five sweeps of the point Gauss–Seidel method, and the pressure correction equation is solved with twenty sweeps of point Gauss–Seidel. These values were empirically determined to strike a good balance between cost and effectiveness of the preconditioner.

3.2.2. SIMPLER. SIMPLER is a variation due to Patankar [28]. It is similar to SIMPLE, but it determines $p^{(n+1)}$ from $\mathbf{u}^{(n)}$ and uses a separate potential field ϕ to enforce continuity in a manner similar to projection methods [8, 2].

As with SIMPLE, each cycle begins with an update of the momentum transport operator (2.2) to reflect the latest approximate solution $\mathbf{u}^{(n)}$. For the next iteration the pressure and velocity field should satisfy

$$\mathbf{Q}\mathbf{u}^{(n+1)} = \mathbf{f} - \nabla^h p^{(n+1)}.$$

To determine an equation for $p^{(n+1)}$, introduce the splitting $\mathbf{Q} = \mathbf{D} - (\mathbf{L} + \mathbf{U})$, where $\mathbf{D}$ is again given by (3.5) and $-(\mathbf{L} + \mathbf{U})$ are the off-diagonal elements of $\mathbf{Q}$. It follows that

$$\begin{aligned} \mathbf{u}^{(n+1)} &= \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n+1)} - \nabla^h p^{(n+1)}) \\ &\approx \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)} - \nabla^h p^{(n+1)}). \end{aligned} \tag{3.11}$$

Taking the divergence of both sides of (3.11), requiring that $\nabla^h \cdot \mathbf{u}^{(n+1)} = 0$, changing the approximation to equality, and rearranging terms lead to

$$S p^{(n+1)} = -\nabla^h \cdot \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)}), \tag{3.12}$$

where S is again given by (3.9).

Once the updated pressure field is known, the updated velocity field is found by first solving

$$\mathbf{Q}\mathbf{u}^{(n+\frac{1}{2})} = \mathbf{f} - \nabla^h p^{(n+1)} \tag{3.13}$$

and then correcting $\mathbf{u}^{(n+\frac{1}{2})}$ so that the updated velocity field conserves mass. This is done using the gradient of an auxiliary variable ϕ. To determine an equation for ϕ, the following *ansatz* is made for the correction:

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n+\frac{1}{2})} - \mathbf{D}^{-1}\nabla^h \phi. \tag{3.14}$$

An equation for ϕ is then determined by requiring that $\nabla^h \cdot \mathbf{u}^{(n+1)} = 0$:

$$\begin{aligned} 0 &= \nabla^h \cdot \mathbf{u}^{(n+1)} \\ &= \nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})} + S\phi, \end{aligned} \tag{3.15}$$

where S is again given by (3.9). Summarizing, we have the following algorithm.

ALGORITHM 3.3 (SIMPLER).
Determine $p^{(n+1)}$ *by solving* (3.12).
Determine $\mathbf{u}^{(n+\frac{1}{2})}$ *by solving* (3.13).
Determine ϕ *by solving* (3.15).
Correct $\mathbf{u}^{(n+\frac{1}{2})}$ *using* (3.14).

Remark 3.5. The correction (3.14) may be described in terms of a projection:

$$\begin{aligned} \mathbf{u}^{(n+1)} &= \mathbf{u}^{(n+\frac{1}{2})} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})} \\ &= \left(\mathbb{I} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h \cdot\right)\mathbf{u}^{(n+\frac{1}{2})} \\ &\equiv \mathbb{P}\mathbf{u}^{(n+\frac{1}{2})}. \end{aligned}$$

From (3.9) it follows that $\mathbb{P}^2 = \mathbb{P}$. Thus $\mathbb{P}$ is actually a *projection* (though it is not an *orthogonal* projection with respect to the standard inner product).

3.2.3. Alternative description. In order to employ SIMPLER as either a preconditioner or a smoother in a linear multigrid method, the method must be recast in a form that operates on residuals, analogous to (3.10). To derive such a representation, observe that

$$\begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} = \mathcal{M}^{-1} \begin{pmatrix} \mathbf{f} \\ -\nabla^h \cdot \mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L} + \mathbf{U})\mathbf{u}^{(n)}) \end{pmatrix}, \tag{3.16}$$

where

$$\mathcal{M} = \begin{pmatrix} \mathbf{Q} & \nabla^h \\ 0 & S \end{pmatrix}.$$

The residual $\mathbf{r}^{(n)}$ may be partitioned into components r_1, r_2 that correspond, respectively, to the momentum and continuity equations. Then note that

$$\mathbf{Q}\mathbf{u}^{(n)} + \nabla^h p^{(n)} + r_1 = \mathbf{f}$$

so that

$$\mathbf{D}^{-1}(\mathbf{f} + (\mathbf{L}+\mathbf{U})\mathbf{u}^{(n)}) = \mathbf{u}^{(n)} + \mathbf{D}^{-1}\nabla^h p^{(n)} + \mathbf{D}^{-1} r_1.$$

Thus (3.16) may be rewritten as

$$\begin{aligned}
\begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} &= \mathcal{M}^{-1}\left[\begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h \cdot & -S \end{pmatrix}\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h \cdot \mathbf{D}^{-1} & 0 \end{pmatrix}\mathbf{r}^{(n)}\right] \\
&= \mathcal{M}^{-1}\left[\begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} + (\mathcal{M} - \mathcal{Q})\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} - \begin{pmatrix} 0 & 0 \\ \nabla^h \cdot \mathbf{D}^{-1} & 0 \end{pmatrix}\mathbf{r}^{(n)}\right] \\
&= \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h \cdot \mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)}.
\end{aligned}$$

Finally, add the projection step to obtain the complete update:

$$\begin{aligned}
\begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} &= \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \mathbb{I} \end{pmatrix}\begin{pmatrix} \mathbf{u}^{(n+\frac{1}{2})} \\ p^{(n+1)} \end{pmatrix} \\
&= \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \mathbb{I} \end{pmatrix}\left[\begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h \cdot \mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)}\right].
\end{aligned}$$

Now note that

$$\begin{aligned}
\mathbb{P}\mathbf{u}^{(n)} &= \mathbf{u}^{(n)} + \mathbf{D}^{-1}\nabla^h S^{-1}\nabla^h \cdot \mathbf{u}^{(n)} \\
&= \mathbf{u}^{(n)},
\end{aligned}$$

provided that each iterate $\mathbf{u}^{(n)}$ is calculated to accurately satisfy the continuity equation. Thus

$$\begin{pmatrix} \mathbf{u}^{(n+1)} \\ p^{(n+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{(n)} \\ p^{(n)} \end{pmatrix} + \begin{pmatrix} \mathbb{P} & 0 \\ 0 & \alpha\mathbb{I} \end{pmatrix}\mathcal{M}^{-1}\begin{pmatrix} \mathbb{I} & 0 \\ -\nabla^h \cdot \mathbf{D}^{-1} & \mathbb{I} \end{pmatrix}\mathbf{r}^{(n)}, \tag{3.17}$$

where a damping factor $\alpha \in (0,1]$ has been included as in Remark 3.1.

Remark 3.6. Remark 3.4 also applies to (3.17).

Remark 3.7. In order to satisfy the condition $\mathbb{P}\mathbf{u}^{(n)} = \mathbf{u}^{(n)}$, it is necessary to compute the action of $\mathbb{P}$ to high accuracy. In light of (3.15), it is apparent that

$$\|\nabla^h \cdot \mathbf{u}^{(n+1)}\| = \|\nabla^h \cdot \mathbf{u}^{(n+\frac{1}{2})} + S\phi\|,$$

so that accurate computation of the action of $\mathbb{P}$ can be accomplished by solving (3.15) to a tight absolute tolerance; here an absolute tolerance of $\epsilon_{abs} = 10^{-10}$ was used. Since S is symmetric and positive semidefinite, a preconditioned conjugate gradient method can be used. The preconditioner was a multigrid method composed

of volume-averaged restriction, piecewise constant prolongation, symmetric Gauss–Seidel smoothing, V(1,1) cycles, and a Galerkin coarse grid version of (3.9). This latter choice was made necessary by the fact that $\mathbf{D}$ is not conveniently available on the coarser grids, and was made easy to implement because of the choices made for intergrid transfers. The semidefiniteness was treated by requiring ϕ to have a 0 average value, which amounts to a least-squares projection against the null space of S. These choices led to a highly efficient solver with convergence rates of less than 0.1 that were observed to be independent of grid size.

3.3. Multigrid preconditioners with pressure-correction smoothers. Solving the Newton equations (3.1) with a preconditioned iterative method requires computing the action of the inverse of a preconditioner M,

$$z = M^{-1}r, \tag{3.18}$$

at every iteration of the linear solver. In this, r is some vector in the Krylov subspace generated by the Jacobian $F'(x_k)$, M, and the initial residual $F(x_k)$. In a matrix-free Newton–Krylov method, constructing M can be a challenge since the matrix entries of the Jacobian are not available. A good choice for solving the Navier–Stokes equations is

$$M = \mathcal{Q}[\mathbf{u}_k],$$

where $\mathbf{u}_k$ is the current Newton approximation to the velocity field. This captures most of the important features of the Jacobian. Moreover, $\mathcal{Q}[\mathbf{u}_k]$ is readily available since it is generally formed and stored at the beginning of each nonlinear iteration so that it can be used in (2.1) to evaluate F. Alternatively, it is possible to construct M from a lower-order discretization of the Navier–Stokes system. This entails a higher setup cost since it requires a rediscretization, but it can also lead to a system (3.18) that is easier to solve. This does not sacrifice accuracy in the solution, since convergence of the overall procedure is determined by $\|F\|$, which is based on a higher-order discretization.

Once M is chosen, a means for approximating the action of its inverse (3.18) is needed. Multigrid methods [4, 41] provide an efficient means for performing this operation. These approaches compute the solution to a problem discretized on a given grid by approximately solving related problems on coarser grids. The following is a recursive version of a V-cycle, where the h is used to indicate different grid levels and h_c denotes the coarsest grid that is used.

ALGORITHM 3.4 (MG-V(h, M^h, z^h, r^h)).
If $h = h_c$ *then:*
  Solve $M^h z^h = r^h$.
else
  Presmooth $z^h \longleftarrow z^h + B(r^h - M^h z^h)$ ν_1 *times.*
  Restrict $r^{2h} = I_h^{2h}(r^h - M^h z^h)$.
  Set $z^{2h} = 0$.
  MG-V($2h$, M^{2h}, z^{2h}, r^{2h}*).*
  Correct $z^h = z^h + I_{2h}^h z^{2h}$.
  Postsmooth $z^h \longleftarrow z^h + B(r^h - M^h z^h)$ ν_2 *times.*

A multigrid algorithm is constructed by supplying the following components:

- a *grid coarsening* strategy,

- a *restriction* operation I_h^{2h} to transfer the problem from a fine grid to a coarser grid,
- a *prolongation* operation I_{2h}^h to transfer the problem from a coarse grid to a finer grid,
- a *coarse grid operator* M^{2h},
- a *coarse grid solver* for solving the problem on the coarsest grid, and
- a *smoother* B that effectively reduces high frequency components of the error.

Grid coarsening is readily achieved by defining each coarse cell to be a union of underlying fine cells. Restriction and prolongation of both velocity and pressure on a staggered grid are accomplished via bilinear interpolation; these are rather standard choices [40]. While coarse grid operators M^{2h} can be computed using a Galerkin coarse grid approximation [44], for simplicity they are determined here by recomputing $M^{2h} = \mathbf{Q}[\mathbf{u}_k^{2h}]$ on the coarser grid. The solution on the coarsest grid h_c is obtained with a fixed number of sweeps of the smoother. Finally, the action of the smoother B is given by either (3.10) or (3.17), once again with $\mathbf{Q}$ fixed at the current inexact Newton approximation.

One additional consideration is required when SIMPLER is used as a smoother. While the projection $\mathbb{P}$ is used to ensure $\nabla^h \cdot \mathbf{u}^h = 0$ to machine accuracy on each grid level, the interpolated coarse grid correction $I_{2h}^h \mathbf{u}^{2h}$ does not satisfy this constraint. An additional application of the projection $\mathbb{P}$ must be applied to the coarse grid correction, so that the velocity portion of the coarse grid correction becomes

$$\mathbf{u}^h = \mathbf{u}^h + \mathbb{P} I_{2h}^h \mathbf{u}^{2h}.$$

4. Numerical evaluations. The combination of multigrid methods, Newton–Krylov methods, and pressure-correction methods offer a wide range of algorithmic choices. Since these combinations embed iterative methods within iterative methods, it is difficult to predict the most effective combination. Numerical experimentation is necessary to help determine which combinations merit further investigation. This section illustrates effective combinations of these approaches. Except where noted, all reported execution times were measured on a MIPS R10000 processor with a 1 MB L2 cache running at 195 Mhz using the MipsPro 7.2 Fortran compiler and level 2 optimization. Solutions to steady-state problems were obtained using NITSOL [31] on a 128×128 grid using an initial approximation $x_0 = 0$ and a GMRES restart value of 100. Such a large restart value is too large for practical computation and is used here only to study the behavior of the preconditioner without the influence of restarting. The nonlinear iterations were terminated when $\|F(x_k)\| \leq 10^{-6}\|F(x_0)\|$.

4.1. Flow in a lid driven cavity. The driven cavity problem is a classic difficult fluid dynamics benchmark problem. The governing equations consist of the incompressible Navier–Stokes equations (1.1) defined on the unit square $\Omega = [0,1]^2$ with boundary conditions

$$\begin{aligned} u = 1,\ v = 0, \quad & y = 1, \\ u = v = 0 \quad & \text{elsewhere on } \partial\Omega. \end{aligned}$$

In applying the NK-MG method to this problem, convergence difficulties were encountered when the preconditioner was based on the same second-order differences used to discretize the problem. Consequently, all results reported here use a first-order upwind discretization to construct the operator used by the preconditioner. Solutions

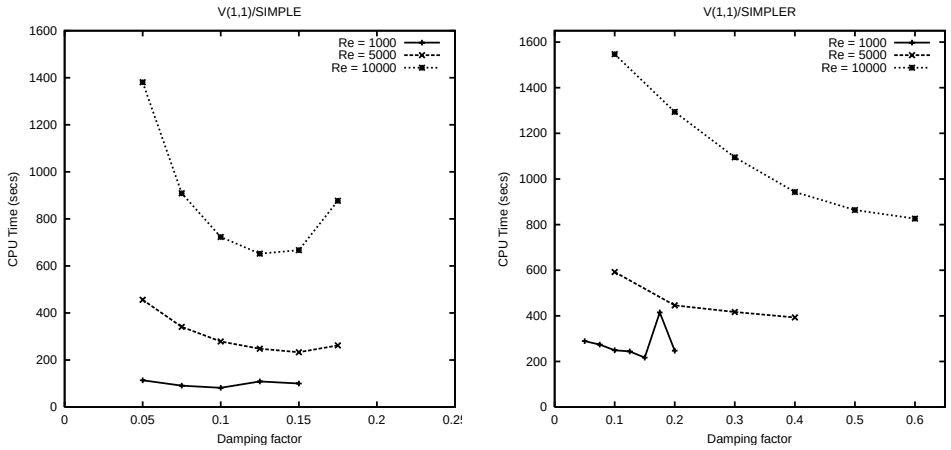

FIG. 4.1. *Sensitivity of multigrid preconditioner to variation in pressure damping factor for Reynolds numbers* 1000, 5000, *and* 10,000.

for Reynolds numbers up to 10,000 were obtained on a 128×128 grid and were within 10% of results published elsewhere [17].

The sensitivity of the performance of the multigrid preconditioners to the pressure damping factor α was investigated. The results for V(1,1) cycles and 2 grid coarsenings are summarized in Figure 4.1; results for larger numbers of pre- or postsmoothing sweeps and for more grid coarsenings are qualitatively similar.

In general, SIMPLER smoothing allowed larger values for the damping factor α. Increasing α beyond those shown in Figure 4.1 caused the inexact Newton method to fail to converge in 200 iterations. Sensitivity to α increased as the Reynolds number was increased. For $Re = 1000$, performance varies by around 25%, excluding the result obtained for SIMPLER smoothing with $\alpha = 0.175$. For this case, backtracking was invoked four times, twice as many as the other cases tested. At $Re = 5000$ performance with SIMPLE smoothing varied by 95%, whereas performance with SIMPLER smoothing varied by 50%; at $Re = 10,000$ the variations were 112% and 87%, respectively.

While multigrid preconditioning with SIMPLE smoothing produced solutions in less CPU time than when SIMPLER smoothing was used, this was the result of a larger number of less expensive iterations. This is due to the need to compute $\mathbb{P}$ to high accuracy: profiling showed that the net cost of this operation was around 20% of the calculation. Note that, by doing so, the preconditioner takes on all the responsibility for enforcing continuity. This is in contrast to the SIMPLE-based preconditioners, where the corrections that are designed to drive the velocity field to conserve mass are not computed with very high accuracy, and the outer Newton iteration is responsible for enforcing $\nabla \cdot \mathbf{u} = 0$. Some consequences of this division of labor are illustrated in Table 4.1, which compares the number of times the preconditioner was applied for the optimal value of α determined above. At lower Reynolds numbers, multigrid preconditioning with SIMPLE smoothing outperforms SIMPLER smoothing by a substantial margin. This gap narrows at $Re = 10,000$, however, as the total number of applications of the SIMPLER-based multigrid preconditioner grows at a slower rate.

SIMPLER smoothing in the multigrid preconditioner also appears to scale better with problem size. This is illustrated in Table 4.2, which shows the usual figures-of-

TABLE 4.1
Number of applications of multigrid preconditioner (NPRE) *and CPU times* (T) *for different smoothers.*

	SIMPLE		SIMPLER	
Re	NPRE	T	NPRE	T
1000	107	82	96	217
5000	283	233	166	393
10,000	834	719	333	826

TABLE 4.2
Algorithmic scaling for different smoothers, $Re = 10,000$. NNI: *number of nonlinear iterations;* NLI: *total number of linear iterations;* NBT: *number of backtracks;* T: *CPU time in seconds.*

	SIMPLE			SIMPLER		
	32×32	64×64	128×128	32×32	64×64	128×128
NLI	623	535	808	305	288	305
NNI	32	23	24	31	30	28
NBT	7	6	6	7	6	6
T	31	108	719	47	187	826

merit for evaluating the performance of an inexact Newton method. These results were obtained with the optimal value for α found above. As problem size increases, multigrid with SIMPLE smoothing requires fewer nonlinear iterations but an increasing number of linear iterations than when SIMPLER smoothing is used. While many factors contribute to the peformance of a globalized Newton–Krylov method as problem size increases, the final arbiter is CPU time. In this regard, the CPU time for the case of SIMPLER scales more favorably, owing to the fact that NLI is nearly constant as problem size is increased.

Finally, performance of the linear solver is examined. When a variable forcing term is used, it is difficult to simply characterize the performance of the linear solver. Table 4.3 summarizes some performance statistics for the linear solver over the entire inexact Newton solve. The minimum number of iteration counts occurs early in the computation, when a conservative value for the forcing term is used. During the course of the iteration, a more aggressive value for the forcing term is attempted, generally leading to a higher iteration count for solving the Newton equations. The maximum values always occurred in the last or next-to-last Newton iteration. Overall, the performance of the linear solver with SIMPLER smoothing is better than when SIMPLE smoothing is used, allowing a smaller restart value to be used. However, this is offset by the higher cost per iteration.

4.2. Buoyancy driven flow. Natural convection in an enclosed cavity is a standard benchmark problem that is frequently used to test different numerical schemes and solution methods [10]. The governing equations consist of the incompressible Navier–Stokes equations (1.1) coupled to an energy transport equation,

$$(uT)_x + (vT)_y - \frac{1}{RePr}\Delta T = 0,$$

together with a body force on the fluid that, under the Boussinesq approximation, is proportional to the temperature

$$\mathbf{f} = \begin{pmatrix} 0 \\ \frac{Ra}{Re^2 Pr} T \end{pmatrix},$$

TABLE 4.3
Performance statistics for solving the Newton equations with multigrid preconditioning and pressure-correction smoothers. The preconditioner is a V(1,1) cycle with two grid coarsenings.

	SIMPLE			SIMPLER		
Re	Min	Avg	Max	Min	Avg	Max
1000	1	7.5	22	1	6.4	21
5000	1	22.6	65	1	9.4	37
10,000	1	33.7	164	1	10.9	72

TABLE 4.4
Performance of NK-MG with SIMPLE-based smoothers. Multigrid preconditioning used a $V(\nu_1,\nu_2)$ cycle and three grid coarsenings.

	V(1,1)	V(2,1)	V(2,2)	V(4,2)	V(4,4)
NLI	285	221	193	155	133
NNI	19	18	18	19	16
NBT	4	3	3	3	2
T	313	308	329	368	400

where Pr is the Prandtl number and Ra is the Rayleigh number. Following [25], Re is fixed at 1, Pr is fixed at 0.71, and the Rayleigh number is varied. The problem is defined on the unit square $\Omega = [0,1]^2$ with boundary conditions

$$\begin{aligned} u = v = 0 \quad &\text{on } \partial\Omega, \\ T(0,y) = 0,\ T(1,y) = 1, \quad &y \in [0,1], \\ T_y(x,0) = T_y(x,1) = 0, \quad &x \in [0,1]. \end{aligned}$$

The additional transport equation is readily accommodated by the SIMPLE and SIMPLER smoothers. Five sweeps of the point Gauss–Seidel method are applied to the discretized energy equation prior to solving the momentum equations. The updated temperature field is then incorporated as a source term before solving the equation governing the vertical component of momentum.

Results for $Ra = 100,000$ using SIMPLE smoothing in the multigrid preconditioner are summarized in Table 4.4, which contains the usual figures-of-merit for evaluating a Newton–Krylov method. Use of a multigrid preconditioner reduces both the number of nonlinear iterations and the number of backtracking steps that were needed. Reducing the number of backtracking steps when employing a variable forcing term is particularly desirable, since each backtracking step increases the forcing term and delays onset of superlinear convergence behavior. In general, V(2,1) cycles in the preconditioner seem to provide a performance "sweet spot." Use of more pre- and/or postsmoothing sweeps decreases NLI, but these reductions are not always enough to offset the resulting higher cost of the preconditioner. Finally, note that use of a multigrid preconditioner improves the performance of GMRES to the extent that a smaller restart value can be used. Thus the high-quality multigrid preconditioner has an additional benefit of reducing the storage costs associated with the Newton–Krylov method. Similar results were obtained for Rayleigh numbers ranging from 20,000 to 500,000.

Results for the same problem that were obtained with the SIMPLER-based preconditioners are summarized in Table 4.5, which reports the same figures-of-merit that appear in Table 4.4. Here only two grid coarsenings were used. Note that the SIMPLER-based preconditioners generally require fewer linear iterations than their SIMPLE-based counterparts. Yet, as observed for the driven cavity problem, the CPU

TABLE 4.5
Performance of the inexact Newton method with SIMPLER-based preconditioners. Column labels are the same as for Table 4.4.

	V(1,1)	V(2,1)	V(2,2)	V(4,2)	V(4,4)
NLI	191	179	178	155	223
NNI	20	19	22	18	20
NBT	3	3	5	3	4
T	464	552	676	790	1418

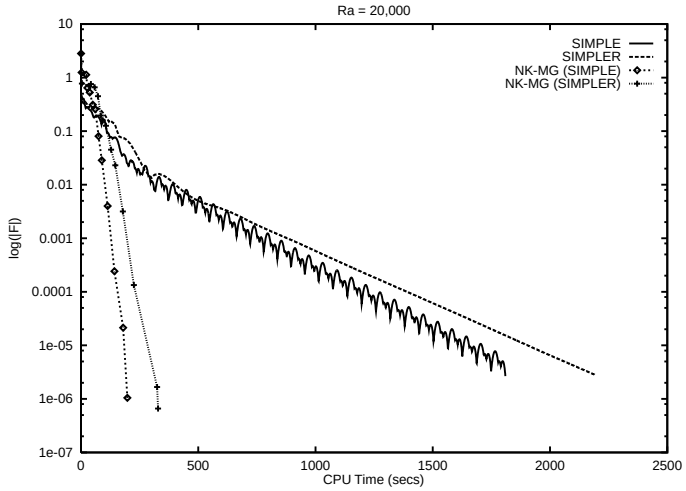

FIG. 4.2. *Convergence histories for SIMPLE, SIMPLER, and the NK-MG methods using these pressure-correction solvers as smoothers. V*(2,1) *cycles are used with SIMPLE smoothing, while V*(1,1) *cycles are used with SIMPLER smoothing.*

times are somewhat higher. Another interesting trend to note is that, while increasing the number of pre- or postsmoothing sweeps generally decreases the total number of linear iterations, it also increases the total execution time. This reflects the tradeoff between the cost and effectiveness of the preconditioner. Both of these observations can be attributed to the cost of enforcing $\nabla^h \cdot \mathbf{u} = 0$ in the preconditioner.

Next, the performance of the NK-MG method using different smoothers is compared with stand-alone versions of the pressure-correction methods. Figure 4.2 shows this comparison for $Ra = 20,000$. The NK-MG method with SIMPLE smoothing is nine times faster than SIMPLE as a stand-alone solver; the NK-MG method with SIMPLER smoothing is nearly seven times faster than SIMPLER as a solver. With SIMPLE smoothing, the largest number of linear iterations taken in an inexact Newton step was 27, so these same results could be obtained by storing only a maximum of 27 Krylov subspace basis vectors. Using a smaller restart value of 10 increases the solution time by roughly a third. On the other hand, when SIMPLER smoothing was used, the largest number of linear iterations taken in an inexact Newton step was 46; using a smaller restart value of 10 increases the solution time by almost 15%.

Dynamic selection of forcing terms as described in [13] employs a safeguard that prevents η_k from getting too small too quickly. More specifically, Choice 1 from [13] uses the safeguard

$$\eta_k \leftarrow \max\{\eta_k, \eta_{k-1}^{(1+\sqrt{5})/2}\} \text{ if } \eta_{k-1}^{(1+\sqrt{5})/2} > \text{threshold}. \tag{4.1}$$

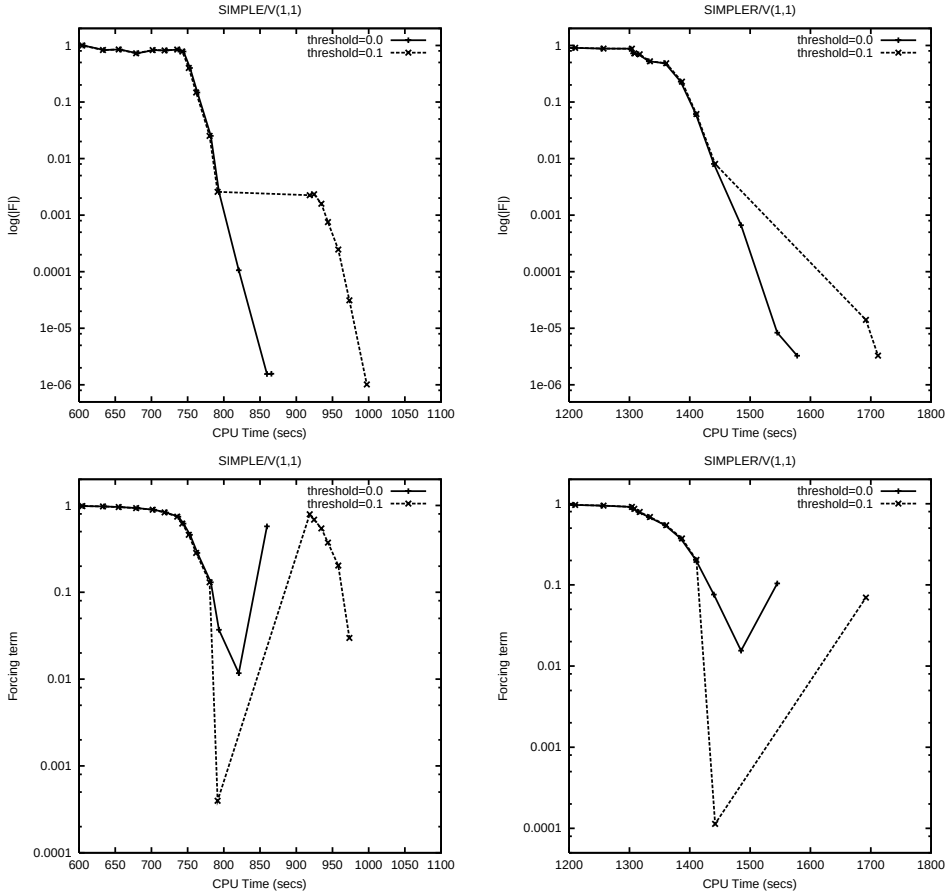

FIG. 4.3. *Convergence histories and forcing terms of the NK-MG method with and without safeguard* (4.1). *When the threshold is set to* 0*, the safeguard is never disabled. SIMPLE smoothing is on the left, while SIMPLER smoothing is on the right.*

The motivation for this safeguard is that it is possible to request too much accuracy in the solution of the Newton equations, which can happen, for example, when there is serendipitously good agreement between the linear model and the nonlinear function. This safeguard is disabled once the forcing term drops below the threshold value. The choice of a threshold value is somewhat arbitrary, and the implementation of Algorithm INB in NITSOL allows a user to specify this while supplying a default value of 0.1 [31]. The safeguard can be retained by using a threshold value of 0.

While previous studies of the performance of Newton-GMRES with dynamic selection of forcing terms [13, 35, 31] disabled this safeguard, it was found that, when using a multigrid preconditioner, doing so sometimes led to undesirable behavior, as illustrated in Figure 4.3. These plots show the final stages in the solution of the buoyancy-driven flow problem at $Ra = 1,000,000$. The top two plots show $\|F(x_k)\|$, while the bottom two plots show the corresponding η_k. The bottom plots show that, with the safeguard disabled at a threshold of 0.1, a very small value for η_k is generated. For SIMPLE smoothing, the linear solver fails to converge in 100 iterations. Further, sufficient reduction in $\|F\|$ is not achieved, which triggers backtracking, increases the

forcing term, and delays convergence of the nonlinear iterations. (The increase in η that is seen on the last iteration when the safeguard is retained is the result of a second safeguard implemented in NITSOL to prevent unnecessary work in reducing $\|F\|$ beyond the requested tolerance [31].) When SIMPLER smoothing is used, backtracking is not triggered, but when (4.1) is disabled, a smaller η is produced in the next to last iteration. While this saves one nonlinear iteration over the case where (4.1) is retained, overall it results in 49 more linear iterations and a higher overall cost to solve the problem.

4.3. Unsteady thermal convection. The issues that are important in a time dependent simulation are somewhat different than was the case for the steady-state case. One major difference is that unlike the steady-state case, there will usually be a very good initial guess for the Newton iteration. This means that globalization strategies are not as important as they were for steady-state problems. For the set of runs shown in this section, SIMPLE was used as the smoother in a multigrid preconditioner, point Gauss–Seidel relaxation was replaced by red-black Gauss–Seidel relaxation, and V(4,2) cycles were used. All of the tests were run over ten time steps using a constant time step size. The test problem is the threee-dimensional analogue of the natural convection problem described in section 4.2. The problem is defined on the unit square $\Omega = [0,1]^3$ with boundary conditions

$$\begin{aligned} u = v = w = 0 \quad & \text{on } \partial\Omega, \\ T(0,y,z) = 0,\ T(1,y,z) = 1, \quad & y,z \in [0,1], \\ T_y(x,0,z) = T_y(x,1,z) = 0, \quad & x,z \in [0,1], \\ T_z(x,y,0) = T_z(x,y,1) = 0, \quad & x,y \in [0,1]. \end{aligned}$$

Before presenting the numerical results, the structure of the computer program that is used is described. The package PETSc [1] is used to implement the inexact Newton solver that was described in section 3.1. In order to define the grid, the package SAMRAI [16] is used. SAMRAI is an object oriented framework for implementing structured adaptive mesh refinement (SAMR). Although adaptive mesh refinement is not used for the work presented in this paper, it is a feature that will be vital in future work.

In order to use PETSc to solve nonlinear systems that are defined on SAMRAI grids, either the data must be copied into a format that PETSc understands or the PETSc vector functions must be extended to act on SAMRAI grids. The former strategy will likely be very inefficient, as large amounts of data must be moved in memory. Instead, SAMRAI provides an interface to PETSc that redirects PETSc vector operations, such as norms and dot products, to act directly on the data that is stored in SAMRAI format. This solver structure is used to solve the natural convection problem and test several aspects of the algorithms.

First the effect of using a better initial iterate on the performance of the linear and nonlinear solver is examined. For this time integrator, the initial conditions are used as the initial iterate for the first time step, and then a linear predictor based on the previous two time steps is used for the remaining time steps in a simulation. The linear predictor should perform much better than the initial iterate for the first time step. The results shown in Table 4.6 are for a $128 \times 128 \times 128$ grid, and they show that the solvers have to work much harder to solve the initial time step because of the poor initial iterate. Also, for all the results shown here, the backtracking scheme was never activated.

TABLE 4.6
Performance of the linear and nonlinear solvers per time step for the three-dimensional natural convection problem on a grid of $128 \times 128 \times 128$.

		First time step		10 time steps	
Ra	Δt	NNI	NLI	NNI	NLI
10^6	10^{-4}	5	14	4.1	10.4
10^7	10^{-5}	9	24	4.0	8.0
10^8	10^{-6}	8	19	2.8	4.2

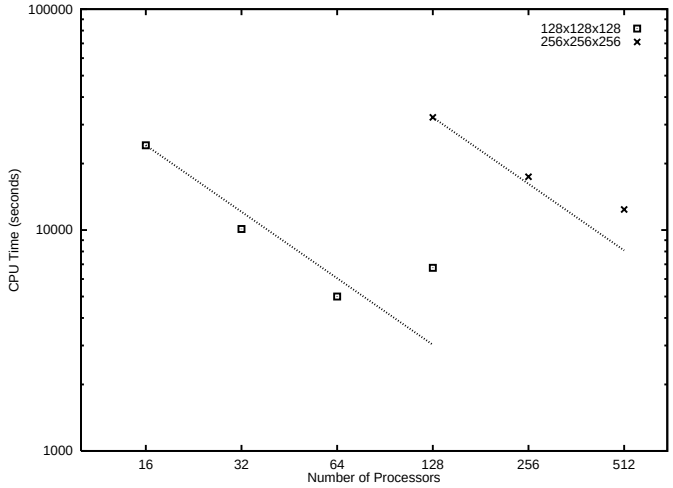

FIG. 4.4. *Parallel performance on an SGI Origin* 2000 *for the natural convection problem.*

The parallel performance of these algorithms is also very important, so the same test problem was run on the SGI Origin 2000 at Los Alamos National Laboratory with $Ra = 10^6$ and $\Delta t = 10^{-4}$ for the $128 \times 128 \times 128$ problem and $Ra = 10^6$ and $\Delta t = 10^{-5}$ for the $256 \times 256 \times 256$ problem. The results, shown in Figure 4.4, indicate that you can get very good scalability for the small problem until the problem size on each processor becomes too small. Then increasing the problem size allows us to get good scalability up to 512 processors. This type of behavior is typical of parallel algorithms for solving PDEs.

5. Summary and conclusions. The pressure-correction methods SIMPLE and SIMPLER have been adapted for use in preconditioning a matrix-free Newton–Krylov method applied to the incompressible Navier–Stokes equations. Application to both two- and three-dimensional problems, as well as unsteady problems, was demonstrated. NK-MG methods with pressure-correction smoothers substantially accelerate the convergence of the pressure-correction methods, albeit at an increase in storage cost, and their performance scales well as problem size is increased. The performance of both methods was found to be sensitive to damping the pressure update; analysis of this behavior will be the subject of future work and will be facilitated by the development of a new correction-oriented version of SIMPLER. SIMPLE-based preconditioning was found to require less CPU time. While SIMPLER-based preconditioning generally led to fewer total linear iterations to complete the inexact Newton solve, this reduction in iteration counts was offset by the additional expense of enforcing continuity at each step. However, SIMPLER-based preconditioning displayed

some favorable robustness trends with respect to problem size and Reynolds number, so further investigation of ways to reduce the cost per iteration is warranted. Finally, these methods were found to be straightforward to parallelize, and encouraging initial parallel results have been obtained.

Acknowledgments. Thanks to Dana Knoll of Los Alamos National Laboratory for pointing out that **Q** could be directly reused in the Newton–Krylov preconditioner. The comments of an anonymous referee led to numerous improvements in the presentation. The computations presented in this paper were conducted on facilities provided by the University of Utah Center for High Performance Computing and by Los Alamos National Laboratory.

REFERENCES

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *The Portable Extensible Toolkit for Scientific Computing, Version* 2.0.24, http://www.mcs.anl.gov/petsc, 1999.

[2] J. B. Bell, P. Colella, and H. M. Glaz, *A second-order projection method for the incompressible Navier-Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[3] M. J. Booth, A. G. Schlijper, L. E. Scales, and A. D. J. Haymet, *Efficient solution of liquid state integral equations using the Newton-GMRES algorithm*, Comput. Phys. Comm., 119 (1999), pp. 122–134.

[4] A. Brandt, *Multilevel adaptive solution to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[5] P. N. Brown, B. Chang, F. Graziani, and C. S. Woodward, *Implicit solution of large-scale radiation-material energy transfer problems*, in Iterative Methods in Scientific Computation IV, D. R. Kincaid and A. C. Elster, eds., Series in Computational and Applied Mathematics 5, IMACS, New Brunswick, NJ, 1999, pp. 343–356.

[6] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[7] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton-Krylov-Schwarz methods in CFD*, in Proceedings of an International Workshop on Numerical Methods for the Navier-Stokes Equations, F. Hebeker and R. Rannacher, eds., Vieweg Verlag, Braunschweig, 1994, pp. 17–30.

[8] A. J. Chorin, *Numerical solution of the Navier-Stokes equations*, Math. Comp., 22 (1968), pp. 745–762.

[9] C. N. Dawson, H. Klie, M. F. Wheeler, and C. S. Woodward, *A parallel, implicit, cell-centered method for two-phase flow with a Newton-Krylov solver*, Computational Geosci., 1 (1997), pp. 215–249.

[10] G. de Vahl Davis, *Natural convection of air in a square cavity: A benchmark numerical solution*, Internat. J. Numer. Methods Fluids, 3 (1983), pp. 249–264.

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

[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] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[15] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, (1992), pp. 87–100.

[16] X. Garaizar, R. Hornung, and S. Kohn, *The use of object oriented design patterns in the SAMRAI structured AMR framework*, in Proceedings of the SIAM Workshop on Object Oriented Methods for Interoperable Scientific and Engineering Computing, M. Henderson, C. Anderson, and S. Lyons, eds., SIAM, Philadelphia, 1998.

[17] 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.

[18] F. H. Harlow and J. E. Welch, *Numerical study of large amplitude free surface motions*, Phys. Fluids, 8 (1965), pp. 2182–2189.
[19] P. G. Jacobs, V. A. Mousseau, P. R. McHugh, and D. A. Knoll, *Newton–Krylov–Schwarz techniques applied to the two-dimensional incompressible Navier–Stokes and energy equations*, in Domain Decomposition Methods in Scientific and Engineering Computing, D. Keyes and J. Xu, eds., AMS, Providence, RI, 1994, pp. 503–507.
[20] D. A. Knoll and P. R. McHugh, *An inexact Newton algorithm for solving the tokamak edge plasma fluid equations on multiply connected domains*, J. Comput. Phys., 116 (1995), pp. 281–291.
[21] D. A. Knoll, P. R. McHugh, and D. E. Keyes, *Newton-Krylov methods for low-mach-number compressible combustion*, AIAA J., 34 (1995), pp. 961–967.
[22] D. A. Knoll and V. A. Mousseau, *On Newton-Krylov-multigrid methods for the incompressible Navier-Stokes equations*, J. Comput. Phys., 163 (2000), pp. 262–267.
[23] D. A. Knoll and W. J. Rider, *A multigrid preconditioned Newton–Krylov method*, SIAM J. Sci. Comput., 21 (1999), pp. 691–710.
[24] P. R. McHugh and D. A. Knoll, *Comparison of standard and matrix-free implementations of several Newton-Krylov solvers*, AIAA J., 32 (1994), pp. 2394–2400.
[25] P. R. McHugh and D. A. Knoll, *Fully coupled finite volume solutions of the incompressible Navier–Stokes and energy equations using an inexact Newton method*, Internat. J. Numer. Methods Fluids, 19 (1994), pp. 439–455.
[26] J. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[27] V. A. Mousseau, D. A. Knoll, and W. J. Rider, *Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys., 160 (2000), pp. 743–765.
[28] S. V. Patankar, *A calculation procedure for two-dimensional elliptic situations*, Numer. Heat Transfer, 4 (1981), pp. 409–425.
[29] S. V. Patankar and D. B. Spalding, *A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows*, Int. J. Heat Mass Transfer, 15 (1972), pp. 1787–1806.
[30] M. Pernice, *A hybrid multigrid method for the steady-state incompressible Navier-Stokes equations*, Electron. Trans. Numer. Anal., 10 (2000), pp. 74–91.
[31] M. Pernice and H. F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci Comput., 19 (1998), pp. 302–318.
[32] W. J. Rider, D. A. Knoll, and G. L. Olson, *A multigrid Newton-Krylov method for multimaterial equilibrium radiation diffusion*, J. Comput. Phys., 152 (1999), pp. 164–191.
[33] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[34] Y. Saad and M. H. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[35] 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.
[36] G. J. Shaw and S. Sivaloganathan, *On the smoothing properties of the SIMPLE pressure-correction algorithm*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 441–461.
[37] S. Sivaloganathan and G. J. Shaw, *A multigrid method for recirculating flows*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 417–440.
[38] 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.
[39] J. P. van Doormaal and G. D. Raithby, *Enhancements of the SIMPLE method for predicting incompressible fluid flows*, Numer. Heat Transfer, 7 (1984), pp. 147–163.
[40] S. P. Vanka, *Block implicit multigrid solution of Navier-Stokes equations in primitive variables*, J. Comput. Phys., 65 (1986), pp. 138–158.
[41] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley and Sons, New York, 1991.
[42] G. Wittum, *Multigrid methods for Stokes and Navier-Stokes equations*, Numer. Math., 54 (1989), pp. 543–563.
[43] C. S. Woodward, *A Newton-Krylov multigrid solver for variably saturated flow problems*, in Proceedings of the Twelfth International Conference on Computation Methods in Water Resources, Vol. 2, Computational Mechanics Publications, Southhampton, 1998, pp. 609–616.

[44] S. Zeng and P. Wesseling, *An Efficient Algorithm for the Computation of Galerkin Coarse Grid Approximation for the Incompressible Navier-Stokes Equations*, Tech. report 92-40, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1992.