UNIVERSITY OF MARYLAND
INSTITUTE FOR ADVANCED COMPUTER STUDIES
DEPARTMENT OF COMPUTER SCIENCE

A PARALLEL BLOCK MULTI-LEVEL PRECONDITIONER FOR THE 3D INCOMPRESSIBLE NAVIER–STOKES EQUATIONS

CS-TR #4415 / UMIACS TR #2002-95

HOWARD ELMAN[†], V. E. HOWLE[‡], JOHN SHADID[§], AND RAY TUMINARO[¶]

OCTOBER 2002

Abstract. The development of robust and efficient algorithms for both steady-state simulations and fully-implicit time integration of the Navier–Stokes equations is an active research topic. To be effective, the linear subproblems generated by these methods require solution techniques that exhibit robust and rapid convergence. In particular, they should be insensitive to parameters in the problem such as mesh size, time step, and Reynolds number. In this context, we explore a parallel preconditioner based on a block factorization of the coefficient matrix generated in an Oseen nonlinear iteration for the primitive variable formulation of the system. The key to this preconditioner is the approximation of a certain Schur complement operator by a technique first proposed by Kay, Loghin, and Wathen [25] and Silvester, Elman, Kay, and Wathen [45]. The resulting operator entails subsidiary computations (solutions of pressure Poisson and convection–diffusion subproblems) that are similar to those required for decoupled solution methods; however, in this case these solutions are applied as preconditioners to the coupled Oseen system. One important aspect of this approach is that the convection–diffusion and Poisson subproblems are significantly easier to solve than the entire coupled system, and a solver can be built using tools developed for the subproblems. In this paper, we apply smoothed aggregation algebraic multigrid to both subproblems. Previous work has focused on demonstrating the optimality of these preconditioners with respect to mesh size on serial, two-dimensional, steady-state computations employing geometric multi-grid methods; we focus on extending these methods to large-scale, parallel, three-dimensional, transient and steady-state simulations employing algebraic multigrid (AMG) methods. Our results display nearly optimal convergence rates for steady-state solutions as well as for transient solutions over a wide range of CFL numbers on the two-dimensional and three-dimensional lid-driven cavity problem.

[†]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. elman@cs.umd.edu. This work was supported in part by the National Science Foundation under grant DMS0208015.

[‡]V. E. HowleSandia National Laboratories, PO Box 969, MS 9217 Livermore, CA 94551, vehowle@sandia.gov.

[§]John ShadidSandia National Laboratories, PO Box 5800, MS 1111, Albuquerque, NM 87185, jnshadi@cs.sandia.gov.

[¶]Ray TuminaroSandia National Laboratories, PO Box 969, MS 9217, Livermore, CA 94551, rstumin@sandia.gov.

A PARALLEL BLOCK MULTI-LEVEL PRECONDITIONER FOR THE 3D INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

HOWARD ELMAN†, V. E. HOWLE‡, JOHN SHADID§, AND RAY TUMINARO¶

Abstract. The development of robust and efficient algorithms for both steady-state simulations and fully-implicit time integration of the Navier–Stokes equations is an active research topic. To be effective, the linear subproblems generated by these methods require solution techniques that exhibit robust and rapid convergence. In particular, they should be insensitive to parameters in the problem such as mesh size, time step, and Reynolds number. In this context, we explore a parallel preconditioner based on a block factorization of the coefficient matrix generated in an Oseen nonlinear iteration for the primitive variable formulation of the system. The key to this preconditioner is the approximation of a certain Schur complement operator by a technique first proposed by Kay, Loghin, and Wathen [25] and Silvester, Elman, Kay, and Wathen [45]. The resulting operator entails subsidiary computations (solutions of pressure Poisson and convection–diffusion subproblems) that are similar to those required for decoupled solution methods; however, in this case these solutions are applied as preconditioners to the coupled Oseen system. One important aspect of this approach is that the convection–diffusion and Poisson subproblems are significantly easier to solve than the entire coupled system, and a solver can be built using tools developed for the subproblems. In this paper, we apply smoothed aggregation algebraic multigrid to both subproblems. Previous work has focused on demonstrating the optimality of these preconditioners with respect to mesh size on serial, two-dimensional, steady-state computations employing geometric multi-grid methods; we focus on extending these methods to large-scale, parallel, three-dimensional, transient and steady-state simulations employing algebraic multigrid (AMG) methods. Our results display nearly optimal convergence rates for steady-state solutions as well as for transient solutions over a wide range of CFL numbers on the two-dimensional and three-dimensional lid-driven cavity problem.

1. Introduction.

Recently, the development of efficient iterative methods for the fully-implicit solution of the Navier–Stokes equations has seen considerable activity. Significant increases in computing power due to large-scale parallel systems coupled with a decade of work on efficient parallel CFD algorithms (e.g., [16], [28], [42], [47]) have now begun to make large-scale implicit calculations tractable in time frames that are consistent with engineering analysis and scientific exploration. Further, an enhanced need to model stiff nonlinear multiple-time-scale PDE systems such as the Navier–Stokes equations coupled with additional transport/reaction physics has increased interest in fully-implicit solution techniques.

The use of fully-implicit solvers allows the time stepping algorithm to resolve the appropriate time scales of interest (the dynamical modes) as opposed to the much stiffer short time scale physics [2], [29]. The ability to produce a stable integrator for large time steps can also be employed in a nontime-accurate mode within pseudo-transient methods [9], [27]. Further, similar iterative method components can often be utilized in direct-to-steady-state solution methods for appropriate applications.

The robustness and versatility of the fully-implicit schemes, however, come with a significant cost. These methods place a heavy burden on the development of robust nonlinear and linear solution methods for the large-scale systems produced at each

†Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742. elman@cs.umd.edu. This work was supported in part by the National Science Foundation under grant DMS0208015.

‡Sandia National Laboratories, PO Box 969, MS 9217 Livermore, CA 94551, vehowle@sandia.gov.

§Sandia National Laboratories, PO Box 5800, MS 1111, Albuquerque, NM 87185, jnshadi@sandia.gov.

¶Sandia National Laboratories, PO Box 969, MS 9217, Livermore, CA 94551, rstumin@sandia.gov.

time step. For this reason many solvers have relied on decoupled solution strategies. Often, transient schemes combine semi-implicit methods with fractional-step (operator splitting) approaches or use fully-decoupled solution strategies. In these cases, the motivation is to reduce memory usage and to produce a simplified equation set for which efficient solution strategies already exist. Unfortunately, these simplifications place significant limitations on the broad applicability of these methods. For example, fractional-step methods such as pressure projection [1], [6], [14] and operator splitting [35] require time step limitations based on the explicit part of the time integration process as well as on the stability and accuracy associated with the decoupled physics [8], [17], [24], [29], [39], [40], [49]. This restriction can severely limit the step size, and direct-to-steady-state simulations with these methods are not possible.

Fully-decoupled solution strategies (e.g., the SIMPLE [37], SIMPLER [36], and PISO [22] class of methods) use a successive substitution (or Picard) iteration to simplify the coupled systems of equations. Nonlinearities at each time step are resolved by an outer nonlinear iteration. Unfortunately, while this technique should improve time step limitations, steps are frequently reduced to facilitate the nonlinear iteration. Convergence of these decoupled methods can often be problematic. In particular, the nonlinear iteration has only a linear rate of convergence and in practice can often exhibit very slow convergence. In addition, since all the equations have been decoupled artificially, this strategy can sometimes result in non-convergence for difficult problems in which the essential coupling of the physics has been violated (see for example [11], [12], and the references contained therein). The intent of fully-coupling the PDEs in the time integration and nonlinear solver is to preserve the inherently strong coupling of the physics with the goal to produce a more robust solution methodology in the process.

Much of the previous work on parallel fully-coupled solution methods demonstrate considerable success for the solution of the incompressible Navier–Stokes equations (e.g., [11], [16], [41], [42]). In these studies, high parallel efficiencies are attained using preconditioned Krylov methods with additive Schwarz domain decomposition preconditioning and effective sub-domain solvers based on incomplete factorizations. While parallel scaling and robustness are encouraging, the algorithmic scaling is non-optimal since the number of linear iterations increases with increasing problem size or an increase in the number of sub-domains [16], [42]. Attempts at mitigating this poor scaling often consider two-level domain decomposition schemes which accelerate convergence by solving a projected version of the problem on a very coarse grid with a direct solver. This coarse grid correction is then interpolated to the fine grid and combined with the more traditional Schwarz preconditioner. These methods exhibit optimal convergence scaling as demonstrated for coupled solution of Navier–Stokes and Navier–Stokes with thermal energy transport [16], [43], [53]. The principal drawback is that on large three-dimensional problems with many sub-domains, the cost of the coarse grid direct solver becomes prohibitive, and the method becomes sub-optimal in terms of CPU time. Therefore true multi-level preconditioning methods, which can deliver nearly optimal scaling for these coupled solution methods, are still an open research issue.

The current view towards producing optimal coupled solution techniques for the incompressible Navier–Stokes equations is based on using preconditioners that approximate the Jacobian (or an approximate Jacobian for a quasi-Newton method) of the coupled system with some simplified block-partitioned system of equations. These methods include approximate block LU factorization techniques [7], [13], [25], [46] and physics-based preconditioning [30], [38]. When applied to a system of PDEs, there are many similarities among these preconditioners. They are all motivated by a "divide and conquer" approach to constructing a preconditioner. The general goal is to approximately invert separate scalar systems rather than the fully-coupled systems. This reduction to scalar systems is motivated by the desire to apply a composition of multi-level solves on the separate equations to precondition the coupled system effectively.

In this manuscript, we focus on the evaluation of an efficient fully-implicit time integration and direct-to-steady-state solution method using a parallel coupled solver for the incompressible Navier–Stokes equations. This solver is based on an Oseen nonlinear iteration with a multigrid method for the linear subproblems. The Oseen iteration is a successive substitution approach that retains the pressure velocity coupling and relaxes (by means of the nonlinear iteration) the coupling of the convection operator (see Section 2). Since part of the Jacobian coupling that is fully utilized within a Newton scheme is retained, studying the Oseen equations serves as an intermediate step towards the development of a fully-coupled multi-level solution process. Additionally, preconditioners for the Oseen system can be employed within a Newton code. This is particularly natural in the matrix-free Newton-Krylov setting [26]. It is in this context that our study of the Oseen iteration nonlinear solver and the Kay, Loghin, and Wathen [25] and Silvester, Elman, Kay, and Wathen [45] preconditioner is carried out. Previous work with these methods has demonstrated optimality with respect to mesh size on serial, two-dimensional, steady-state computations using geometric multigrid; we focus on extending these methods to large-scale, parallel, three-dimensional, transient and steady-state simulations with algebraic multi-grid (AMG) methods.

The remainder of this paper is organized as follows. Section 2 provides background on the Oseen iteration and the approximate block preconditioners. In Section 3 we describe in some detail the algebraic multigrid methods that are used for the component scalar solvers for the preconditioner systems. Section 4 provides a brief overview of the MAC discretization of Navier–Stokes equations and the parallel implementation of the nonlinear and linear solvers. Details of the numerical experiments and the results of these experiments are described in Section 5. Concluding remarks are provided in Section 6.

2. Background. We are concerned with the incompressible form of the Navier–Stokes equations

$$\begin{array}{rcl} \alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \operatorname{grad})\, \mathbf{u} + \operatorname{grad} p & = & \mathbf{f} \\ -\operatorname{div} \mathbf{u} & = & 0 \end{array} \quad \text{in } \Omega \subset \mathbb{R}^3, \tag{2.1}$$

where $\mathbf{u}$ satisfies suitable boundary conditions on $\partial\Omega$, say Dirichlet conditions $\mathbf{u} = \mathbf{g}$. The value $\alpha = 0$ corresponds to the steady-state problem and $\alpha = 1$ to the transient case.

Our focus is on solution algorithms for the systems of equations that arise after linearization of the system (2.1). We will use a nonlinear iteration derived by lagging the convection coefficient in the quadratic term $(\mathbf{u} \cdot \operatorname{grad})\, \mathbf{u}$. For the steady-state problem, this procedure starts with some initial guess $\mathbf{u}^{(0)}$ for the velocities and then computes updated velocities and pressures by solving the *Oseen equations*

$$\begin{array}{rcl} -\nu \nabla^2 \mathbf{u}^{(m)} + (\mathbf{u}^{(m-1)} \cdot \operatorname{grad})\, \mathbf{u}^{(m)} + \operatorname{grad} p^{(m)} & = & \mathbf{f} \\ -\operatorname{div} \mathbf{u}^{(m)} & = & 0. \end{array} \tag{2.2}$$

For transient problems, a strategy of this type can be combined with an implicit time discretization, see [48], [54]. For example, a variant of the backward Euler discretization uses a first order time discretization for $\mathbf{u}_t$ and treats all other terms implicitly except the lagged convection coefficient. This gives the time-stepping strategy

$$\begin{aligned} \frac{\mathbf{u}^{(m)}-\mathbf{u}^{(m-1)}}{\Delta t} - \nu\nabla^2\mathbf{u}^{(m)} + (\mathbf{u}^{(m-1)}\cdot\mathrm{grad})\,\mathbf{u}^{(m)} + \mathrm{grad}\,p^{(m)} &= \mathbf{f} \\ -\mathrm{div}\,\mathbf{u}^{(m)} &= 0. \end{aligned} \tag{2.3}$$

This iteration can also be used to solve the steady-state problem by integrating in time until a steady solution is obtained. Customarily, when large time steps are used (or equivalently, large CFL numbers) and no error control is applied, this scheme is termed a pseudo-transient method [9], [27].

For both (2.2) and (2.3), a stable finite difference or finite volume discretization leads to a linear system of equations of the form

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}, \tag{2.4}$$

which must be solved at each step. For the steady problem, the matrix F has block diagonal form in which each individual diagonal block consists of a discretization of a convection–diffusion operator

$$-\nu\nabla^2 + (\mathbf{w}\cdot\mathrm{grad})\,, \tag{2.5}$$

where $\mathbf{w} = \mathbf{u}^{(m-1)}$. For the transient problem, the blocks of F represent discretizations of the operator

$$\frac{1}{\Delta t}I - \nu\nabla^2 + (\mathbf{w}\cdot\mathrm{grad})\,, \tag{2.6}$$

which arises from implicit time discretization of the time-dependent convection–diffusion equation.

The strategy we employ for solving (2.4) is derived from the block factorization

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix} \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix},$$

where $S = BF^{-1}B^T$ is the Schur complement. This implies that

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix}, \tag{2.7}$$

which, in turn, suggests a preconditioning strategy for (2.4). If it were possible to use the matrix

$$\mathcal{Q} = \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix} \tag{2.8}$$

as a right-oriented preconditioner, then the preconditioned operator would be the one given in (2.7). All the eigenvalues have the value 1, and it can be shown that this operator contains Jordan blocks of at most 2, and consequently that at most two iterations of a preconditioned GMRES iteration would be needed to solve the system [33].

When any preconditioner $\mathcal{Q}$ is used in a Krylov subspace iteration, each step requires the application of $\mathcal{Q}^{-1}$ to a vector. To see the computational issues involved for the particular choice (2.8), it is useful to express $\mathcal{Q}^{-1}$ in factored form

$$\begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -S^{-1} \end{pmatrix}.$$

This shows that two nontrivial operations are required to apply $\mathcal{Q}^{-1}$: application of S^{-1} to a vector in the discrete pressure space, and application of F^{-1} to a vector in the discrete velocity space. These tasks, especially the first one, are too expensive for a practical computation. However, an effective preconditioner can be derived by replacing these two operations with inexpensive approximations.

Applying the action of F^{-1} to a vector v entails solving the discrete convection–diffusion equation, i.e., solving $Fx = v$ where F is a discrete version of (2.5) or (2.6). For this computation, we will use a multigrid iteration, as outlined in the next section.

The key component for the preconditioner is the availability of an accurate and inexpensive approximation to the action of the inverse of the Schur complement operator $BF^{-1}B^T$. Here, we will use a strategy developed in [25] and [45]. To derive it, we begin with the convection–diffusion operator of (2.5). (The treatment of the transient version (2.6) is identical.) Suppose there is an analogous operator

$$(-\nu\nabla^2 + (\mathbf{w} \cdot \mathrm{grad}))_p$$

defined on the pressure space. It is not necessary to ascribe any physical meaning to this operator; it will only be used to construct an algorithm. Suppose in addition that the convection–diffusion operators formally commute with the gradient operator, i.e.,

$$(-\nu\nabla^2 + (\mathbf{w} \cdot \mathrm{grad}))\nabla = \nabla(-\nu\nabla^2 + (\mathbf{w} \cdot \mathrm{grad}))_p\,. \tag{2.9}$$

A discrete version of this (posited) relation, using the discrete versions of the operators given in (2.4) together with a discretization F_p of the convection–diffusion operator on the pressure space, is

$$FB^T = B^T F_p\,. \tag{2.10}$$

A straightforward algebraic manipulation then gives

$$BF^{-1}B^T = (BB^T)F_p^{-1}. \tag{2.11}$$

In reality, the formal relation (2.9) is not valid except in special cases (such as constant $\mathbf{w}$). However, we can still take the matrix on the right side of (2.11) as an approximation to the Schur complement, leading to the preconditioner

$$\mathcal{Q} = \begin{pmatrix} F & B^T \\ 0 & -\hat{S} \end{pmatrix} \tag{2.12}$$

for (2.4), where $\hat{S} = (BB^T)F_p^{-1}$. Application of $\hat{S}^{-1}$ to a vector is now a relatively straightforward operation, entailing application of the action of $(BB^T)^{-1}$ (i.e., solving a system of equations with coefficient matrix BB^T), followed by a matrix-vector product by F_p. The matrix BB^T is essentially a scaled discrete Laplacian, and there are many approaches for solving the required systems. We will again use algebraic multigrid methods for these computations.[1]

To implement this methodology, it is necessary to construct the matrix F_p, i.e., a discrete convection–diffusion operator on the pressure space. This requires a convention for specifying boundary conditions associated with this operator. Our strategy has been to choose conditions that ensure that the resulting operator is elliptic over the discrete pressure space [10]. When (2.1) is posed with Dirichlet boundary conditions, F_p is defined using Neumann boundary conditions; if a component $\partial\Omega$ is an outflow boundary, then Dirichlet conditions would be used for F_p. (Similar conditions also apply to A_p if that needs to be defined.) The issues involved here appear to be essentially the same as what is required for the pressure Poisson equation in other settings [15, Sect. 3.8.2]. Note, however, that here the choice of pressure boundary conditions only affects the algorithm used to solve the discrete equations (i.e., the definition of the preconditioner) and is unrelated to the accuracy of the underlying solution method.

We highlight some aspects of using the preconditioner of (2.12). Considerable empirical evidence for two-dimensional problems indicates that it is effective, leading to convergence rates that are independent of mesh size, only mildly dependent on Reynolds numbers for steady problems, and essentially independent of Reynolds numbers in the transient case [13], [25], [45]. A proof that convergence rates are independent of the mesh is given in [31]. As observed above, each step of a Krylov subspace iteration then requires a Poisson solve on the pressure space and a convection–diffusion solve on the velocity space. Both of these operations can be performed or approximated using multigrid methods.

3. Multigrid. It is well known that multigrid methods are among the most effective methods for solving discrete partial differential equations, see e.g, [5], [19], [51]. In this study we employ a particular multilevel method called an *algebraic multigrid method* (AMG). These methods require no mesh (or geometric) information and therefore are attractive for solving problems in complex domains discretized with unstructured meshes. Although multigrid methods have been developed for the incompressible Navier–Stokes equations (see, for example, [4], [60]), there has been only a modest amount of work on using algebraic multigrid in this setting. One reason for this is the strong coupling inherent in the complex block structure of the discretized governing PDE system as described in Section 2. A key advantage of the block preconditioning approach is that the resulting component block solvers require separate solutions of equations with coefficient matrices F (a discrete convection–diffusion operator) and A_p (a discrete Laplacian), each of which is amenable to solution by AMG.

We begin by briefly recalling the philosophy behind traditional (geometric) multigrid methods. The basic idea is to capture errors by utilizing multiple resolutions in

[1]This derivation is essentially a full description of the preconditioner for the finite-difference discretization that we will use in Section 5. A more careful derivation, applicable in particular to finite element methods, leads to the approximation

$$BF^{-1}B^T \approx (BM_v^{-1}B^T)F_p^{-1}M_p = A_pF_p^{-1}M_p$$

where M_v and M_p are the *mass matrices* corresponding to the L_2 representation of the finite element bases. $A_p = BM_v^{-1}B^T$ represents a scaled discrete Laplacian operator on the pressure space, and this leads to the more general definition $\hat{S} = A_pF_p^{-1}M_p$. We will not discuss this more general formulation here. It introduces no serious computational difficulties but enables an extension of this approach to handle stable finite element discretizations; see [25] and [45] for details. For finite differences on a uniform grid of width h, $M_p = h^2I$ and $BB^T = A_pM_p$.

an iterative scheme. High energy (or oscillatory) components are effectively reduced through a simple smoothing procedure, while low energy (or smooth) components are tackled using an auxiliary lower resolution version of the problem (coarse grid). The idea is applied recursively on the next coarser level. In standard multigrid, this is accomplished by generating a hierarchy of meshes, $\mathcal{G}_k$, corresponding to differing resolutions. Grid transfer (i.e., interpolation and restriction) operators are defined to move data (residuals and corrections) between meshes, and discretizations are constructed on all the meshes. On coarse meshes, it is common to employ the same discretization technique (often the same subroutine) that is used on the finest mesh. However, it is also possible to project the fine grid operator algebraically via

$$A_{k+1} = P_k^T A_k P_k \tag{3.1}$$

where P_k interpolates a solution from grid $\mathcal{G}_k$ to $\mathcal{G}_{k+1}$, P_k^T restricts a solution from grid $\mathcal{G}_{k+1}$ to $\mathcal{G}_k$, and A_k is the discretization on $\mathcal{G}_k$. In this paper, we only use restriction which is the transpose of interpolation. However, this does not have to be the case and for highly nonsymmetric problems it is often more appropriate to consider alternatives. This is planned for future work. A sample multilevel iteration is given in Figure 3.1 to solve

$$A_1 u_1 = b_1. \tag{3.2}$$

```
// Solve A_k u_k = b_k
procedure multilevel(A_k, b_k, u_k, k)
      u_k = S_k(A_k, b_k, u_k);
      if ( k ≠ Nlevel)
            r_k = b_k − A_k u_k ;
            A_{k+1} = P_k^T A_k P_k;
            u_{k+1} = 0;
            multilevel(A_{k+1}, P_k^T r_k, u_{k+1}, k + 1);
            u_k = u_k + P_k u_{k+1};
            u_k = S_k(A_k, b_k, u_k);
```

FIG. 3.1. *Multigrid V cycle consisting of 'Nlevel' grids to solve* $A_1 u_1 = b_1$.

To specify the method fully, the smoothers S_k and the grid transfers P_k must be defined for each level k.. The key to fast convergence is the complementary nature of these two operators. That is, errors not reduced by S_k must be well interpolated by P_k. In our implementation, we employ a standard Gauss–Seidel smoother for the S_k when solving the Poisson operator. For the convection–diffusion operator, we present experiments with a few different choices. These experiments are discussed in Section 5.

An algebraic multigrid algorithm has the same structure as a standard multigrid algorithm (e.g., Figure 3.1). The main difference is that no grid hierarchy is supplied and so a notion of a mesh must be developed from matrix data. This mesh must then be coarsened, and finally grid transfer operators P_k must be deduced, from purely algebraic principles. We will use one particular approach, called *smoothed aggregation*. This is an algebraic multigrid technique for determining the operators P_k that interpolate the aggregated graph to its refinement given only the $n \times n$ discretization matrices A_k. We give a brief description of a simplified smoothed aggregation scheme for scalar partial differential equations. More details can be found in [52], [56], [57], and [59].

The key feature of AMG methods is that no mesh information is supplied. Instead, a matrix graph is defined, and this graph effectively occupies the role of the mesh used in traditional multigrid methods (with the exception that no coordinates are associated with a matrix graph). Specifically, define the matrix graph

$$\mathcal{G}_k = \{V_k, E_k\}$$

with vertices

$$V_k = \{1, 2, \ldots n\}$$

and undirected edges

$$E_k = \{(i,j) : i,j \in V_k, \quad j \le i, \quad A_k(i,j) \ne 0\}.$$

For this discussion, it is assumed that A_k is structurally symmetric with nonzero diagonal entries. In our notation, (i,j) and (j,i) refer to the same undirected edge. To produce the 'next' mesh within the multigrid hierarchy, $\mathcal{G}_k$ must be automatically coarsened. In smoothed aggregation, $\mathcal{G}_k$ is coarsened by grouping or aggregating neighboring vertices together. Each aggregate will effectively become a mesh point on the next coarser mesh. Formally, an aggregate corresponds to a set agg_k such that

$$agg_p \cap agg_j = \emptyset \qquad p \ne j$$

and

$$V_k = \bigcup_{j=1}^{m} agg_j$$

where m is the total number of aggregates and $\emptyset$ is the empty set. For details on aggregation algorithms, we refer the reader to [59] and [52]. In this paper, it is sufficient to consider an ideal aggregate, agg_k, as comprising a single central vertex and all of its immediate neighbors. In practice, it is not possible to a coarsen a graph completely with ideal aggregates. This is further discussed at the end of this section.

Using the above aggregates, a simple interpolation operator can be defined corresponding to piecewise constants. Specifically, a value at a coarse grid point is interpolated by assigning it to all fine grid vertices within its corresponding aggregate. This interpolation is referred to as the tentative prolongator and is represented by an $n \times m$ matrix $\tilde{P}_k$, where n is the dimension of A_k and m is the total number of aggregates. Each row of $\tilde{P}_k$ corresponds to a grid point, and each column corresponds to an aggregate. Formally, the entries are given by[2]

$$\tilde{P}_k(i,j) = \begin{cases} 1 & \text{if } i \in agg_j \\ 0 & \text{if } i \notin agg_j. \end{cases}$$

The main point is that this simple prolongator is easily constructed without geometric information. Unfortunately, however, using $\tilde{P}_k$ within a multigrid algorithm gives rise

[2]For specific applications such as elasticity problems more complicated tentative prolongators are defined based on rigid body motions.

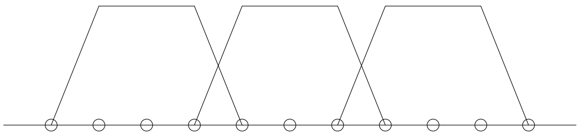

FIG. 3.2. *Three piecewise constant basis functions associated with three aggregates. Each function corresponds to a single prolongator column.*

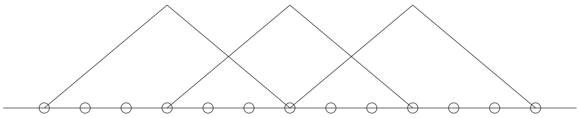

FIG. 3.3. *Three smoothed basis functions. Each corresponds to a single prolongator column.*

to suboptimal (not mesh independent) convergence. Instead, a more robust method is realized by smoothing the piecewise constant basis functions.

The main idea of smoothed aggregation is to smooth the basis functions (i.e., the matrix columns) and thereby lower the energy (i.e., essentially reduce $\|P_k\|_{A_k}$) associated with $\tilde{P}_k$. We omit the theory details and refer the interested reader to the smoothed aggregation references. Specifically, a simple damped Jacobi iteration is applied

$$P_k = (I - \alpha D_k^{-1} A_k)\tilde{P}_k \tag{3.3}$$

where D_k is the diagonal of A_k, and α is a damping parameter. Typically, α is taken as $\frac{4}{3\rho(D_k^{-1}A_k)}$ where $\rho(\cdot)$ denotes the spectral radius. This smoothing step is critical to obtaining h-independent multigrid convergence [3], [56]. Figure 3.2 illustrates the piecewise constant basis functions (or matrix columns) associated with $\tilde{P}_k$. Figure 3.3 illustrates the effect of smoothing by depicting the basis functions (or matrix columns) associated with P_k when A_k is a Laplace operator. Intuitively, it should be no surprise that in this example the multigrid method using piecewise linear interpolation[3] is superior to that using piecewise constant interpolation. It is important to notice that the aggregates in Figure 3.2 are ideal aggregates. That is, they are comprised of a central vertex and its immediate neighbors (i.e., they have a diameter of three). If the diameter is greater than three, the smoothed basis functions have a region where they are locally constant (i.e., the hat functions have a plateau). This leads to slower multigrid convergence due to poorer interpolation properties. When the diameter is less than three, the leftmost and rightmost smoothed basis functions in Figure 3.3 will overlap. This implies that the coarse grid discretization matrix obtained via $P_k^T A_k P_k$ will have additional nonzeros. This can cause the multigrid iteration cost to very quickly increase. Though this example is simple, the situation in higher dimensions and on unstructured grids is identical. In practice, multigrid schemes with convergence/cost properties similar to the ideal aggregate case are achieved using good aggregation heuristics that keep the number of nonideal aggregates to a minimum and prevent nonideal aggregates from becoming too small or large. We refer the reader to [59] for more details.

[3]In general, smoothed aggregation does not reproduce linear interpolation nor is this necessary to obtain mesh independent convergence.

The basic idea and most of the theory for smoothed aggregation has been developed for symmetric positive definite systems. For the nonsymmetric system with coefficient matrix F, we make one modification to the algorithm described here. Specifically, we replace A_k in (3.3) by the symmetric part of F (i.e., $(F+F^T)/2$) and estimate the spectral radius of the symmetric part of F. In this way, the smoothing of the prolongator maintains a sense of energy minimization. We have found that this procedure is quite effective when an incomplete LU factorization [23] is used as a smoother. For the Reynolds numbers that we have considered, the resulting multigrid procedure is quite efficient. Our numerical results (Section 5) demonstrate convergence for the F solve within about 25 multigrid iterations. However, for highly convective flows it should be possible to further improve the multigrid by considering more sophisticated generalizations of smoothed aggregation to nonsymmetric problems which allow for different restriction schemes [18], [58].

4. Implementation. For the steady-state Oseen equations in three dimensions, the structure of the convection-diffusion operator F is a 3×3 block diagonal matrix corresponding to the three velocity components $[u, v, w]$. That is,

$$F = \begin{pmatrix} -\nu\nabla^2 + \mathbf{w}\cdot\nabla & & \\ & -\nu\nabla^2 + \mathbf{w}\cdot\nabla & \\ & & -\nu\nabla^2 + \mathbf{w}\cdot\nabla \end{pmatrix} \tag{4.1}$$

where $\mathbf{w} = [u, v, w]$. The matrix B^T is a simple gradient operator applied to the pressure unknowns. A marker-and-cell (MAC) finite difference scheme [20] is used to discretize the saddle-point linear subproblem $[F\ B;\ B^T\ 0]$. This discretization is stable and first-order accurate in a discrete H_1-norm [34]. All of our experiments are on a uniform mesh of width h. Pressures are on the cell centers and velocities are on the cell faces. In two dimensions, we have N^2 cells and approximately $3N^2$ degrees of freedom. In three dimensions, we have N^3 cells and approximately $4N^3$ degrees of freedom. The operator F_p needed for the preconditioner is also a convection-diffusion operator (2.5) but on the pressure space. Specifically, in three dimensions, the F_p operator on a pressure vector p corresponds to

$$\begin{aligned} F_p p &= (-\nu\nabla^2 + (\mathbf{w}\cdot\mathrm{grad}))_p p \\ &= -\nu(p_{xx} + p_{yy} + p_{zz}) + u p_x + v p_y + w p_z. \end{aligned} \tag{4.2}$$

The discrete Laplacian term is the usual seven point stencil. Discretization of the convection terms uses velocities at the cell edges. Finally, the operator A_p also required by the preconditioner is a standard seven point Laplace operator with Neumann boundary conditions. Since this operator is singular, the constant vector is projected out of the right hand side and the resulting A_p solution. This singularity also makes solution of the coarse grid equations somewhat more difficult than usual, and we handle the coarse grid system by iteration.

The implementation of the preconditioned Krylov subspace solution algorithm was done using the software packages *Petra* and *Trilinos* developed at Sandia National Laboratories [21, 32]. Petra provides fundamental construction and support for many basic linear algebra functions and facilitates matrix construction on parallel distributed machines. Each processor constructs the subset of matrix rows assigned to it via a partitioning, and a local matrix-vector product is defined. Once F and B are defined, a global matrix-vector product for the saddle point linear system $S = [F\ B;\ B^T\ 0]$ is defined using the matrix-vector products for the individual systems. Petra handles all the distributed parallel matrix details (e.g. local indices

$\mathbf{u}^{(0)}$ = initial condition or initial guess
$p^{(0)}$ = initial condition or initial guess

form B and A_p and then set up MG for A_p subproblem

time loop

for $m = 1$,Ntimesteps

$\quad \mathbf{u}^{(m)} = \mathbf{u}^{(m-1)}, \quad p^{(m)} = p^{(m-1)}$

$\quad$ ***nonlinear loop***

$\quad$ while $||\mathcal{F}(\mathbf{u}^{(m)}, \mathbf{u}^{(m-1)}, p^{(m)}, \mathbf{u}^{(m)})|| > \epsilon_{oseen}$

$\quad\quad \mathbf{u}_{lag} = \mathbf{u}_k^{(m)}$

$\quad\quad$ /* Set up $K\mathbf{u}^{(m)} = b$ corresponding to $\mathcal{F}(\mathbf{u}^{(m)}, \mathbf{u}^{(m-1)}, p^{(m)}, \mathbf{u}_{lag}) = 0$ */

$\quad\quad$ form F, F_p and $K = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}$ and then set up MG for F subproblem

$\quad\quad$ ***linear solve***

$\quad\quad$ GMRESR on $K\mathbf{u}^{(m)} = b$ until $||r_k||/||r_0|| < \epsilon_{saddle}$

$$\text{Saddle\_Precondition}() = \begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & -F_p A_p^{-1} \end{pmatrix}$$

$\quad\quad\quad$ where

$\quad\quad\quad\quad A_p^{-1}$ = CG preconditioned AMG on Laplacian until $||r_k||/||r_0|| < \epsilon_A$
$\quad\quad\quad\quad F^{-1}$ = GMRES preconditioned AMG on F until $||r_k||/||r_0|| < \epsilon_F$

FIG. 4.1. *Implementation Pseudo-code.*

versus global indices, communication for matrix-vector products, etc.). Construction of the preconditioner follows in a similar fashion. That is, the individual components are defined and then grouped together to form the preconditioner. All of the Krylov methods (i.e. those for the saddle point solve and for the F and A_p subsystems) are supplied by Trilinos [21], a high-performance parallel solver library that makes available linear and nonlinear solvers along with several preconditioning options. The multigrid preconditioning for the subsystems is done by ML [50], a multigrid preconditioning package, which we access through Trilinos.

Once all of the matrices and matrix-vector products are defined, we can use Trilinos to solve the incompressible Navier–Stokes equations using our block preconditioner with specific choices of linear solvers for the saddle-point problem and the convection–diffusion and pressure Poisson subproblems. To solve the saddle-point linear problem associated with each Oseen iteration, we use GMRESR. GMRESR is a variation on GMRES proposed by van der Vorst and Vuik [55] allowing the preconditioner to vary at each iteration. For the pressure Poisson problem, A_p, we use CG preconditioned with algebraic multigrid, and for the convection–diffusion problem, F, we use GMRES preconditioned with algebraic multigrid. For transient and pseudo-transient problems, we use backward Euler for the time-stepping loop. These choices are summarized in Figure 4.1 to solve the nonlinear problem

$$\mathcal{F}(\mathbf{u}^{(m)}, \mathbf{u}^{(m-1)}, p^{(m)}, \mathbf{u}^{(m)}) = 0$$

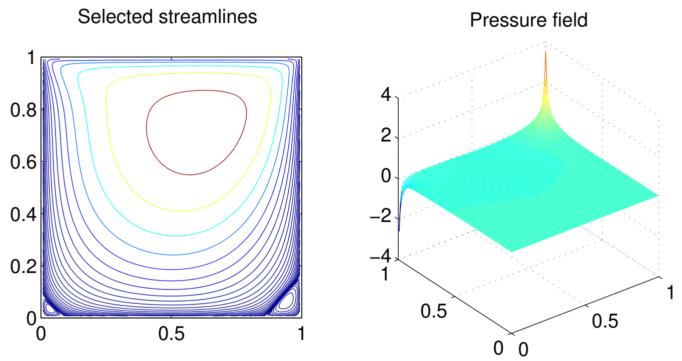

FIG. 5.1. *Sample velocity field and pressure field from 2D lid driven cavity.* $h = 1/128$, $Re = 100$.

at each time step where

$$\mathcal{F}(\mathbf{u}^{(m)}, \mathbf{u}^{(m-1)}, p^{(m)}, \mathbf{w}) = \begin{pmatrix} \alpha \frac{\mathbf{u}^{(m)} - \mathbf{u}^{(m-1)}}{\Delta t} - \nu \nabla^2 \mathbf{u}^{(m)} + (\mathbf{w} \cdot \mathrm{grad})\, \mathbf{u}^{(m)} + \mathrm{grad}\, p^{(m)} - \mathbf{f} \\ -\mathrm{div}\, \mathbf{u}^{(m)} \end{pmatrix}$$

and $\alpha = 1$ for transient problems and $\alpha = 0$ for steady-state problems.

5. Numerical Results.

Numerical experiments are performed on the lid driven cavity problem in two and three dimensions. Specifically, we consider a square region with unit length sides in two dimensions and a cube with unit length sides in three dimensions. Velocities are zero on all edges except the top (lid), which has a driving velocity of one. The two-dimensional lid driven cavity is a well-known benchmark for fluids problems. It contains many features of harder flows. The three-dimensional problem is less well studied and is actually a much more difficult problem. Lid driven cavity flows exhibit unsteady solutions and multiple solutions at high enough Reynolds numbers. In two dimensions, unsteady solutions appear around Reynolds number 7000 to 10,000. In three dimensions, these unsteady solutions can occur at much lower Reynolds number, $Re < 1000$ [44]. Figure 5.1 shows the velocity field and pressure field for an example solution to a two-dimensional lid driven cavity problem with $h = 1/128$.

Results are presented for both steady-state and transient problems. In all presented results, the values for ϵ_{oseen}, ϵ_{saddle}, ϵ_F, ϵ_A, and $NtimeSteps$ in Figure 4.1 are defined as follows. The relative stopping tolerances for the nonlinear and saddle-point problems are $\epsilon_{oseen} = 10^{-5}$ and $\epsilon_{saddle} = 10^{-2}$ as For experiments using 'exact' solutions of the convection–diffusion and pressure Poisson subproblems, we have relative stopping tolerances $\epsilon_F = \epsilon_A = 10^{-10}$. All time-stepping studies employ backward Euler and take ten time steps (i.e., $NtimeSteps = 10$) using a constant time step. All two-dimensional results were obtained in serial on a DEC Alpha ES40. All three-dimensional results were obtained on 100 processors of Sandia's ASCI Red machine. Each of Red's compute nodes consists of two Intel Pentium II Xeon Core processors with a peak performance of 333 MFLOPs each.

5.1. Steady-State Results.

We first explore the performance of the algebraic multigrid solver for the discrete convection–diffusion equations. Performance on the simple Poisson subproblems is optimal and well understood. For the convection–diffusion subproblem $Fx = v$, we explore two multigrid choices: the smoothing operator and the grid transfer operator. In Table 5.1, both ILU and symmetric Gauss–Seidel smoothers are considered within smoothed aggregation multigrid. The ILU and symmetric Gauss–Seidel smoothers are actually used in conjunction with Schwarz domain decomposition ideas. In particular, each processor performs one iteration of the smoother on the subdomain defined by the matrix partitioning (independent of the others) and performs communication between smoothing iterations. These subdomains include one level of overlap (i.e., the processor-based subdomains are expanded by one layer of equations around the subdomain perimeter) though only solution values from the non-overlapped regions are used in the preconditioner. In the case of symmetric Gauss–Seidel, we compare using one and four iterations of symmetric Gauss–Seidel (referred to as 1-Gauss–Seidel and 4-Gauss–Seidel respectively), performed before and after the coarse grid correction on each level of the V-cycle. For ILU, one ILU sweep is performed before and after the coarse grid correction on each V-cycle level. Tables 5.1 and 5.2 show the average multigrid iteration counts and CPU times required to solve the convection–diffusion subproblems arising in the block preconditioner. The timings include the entire time within the Krylov solver and the algebraic multigrid preconditioners. They do not, however, include algebraic multigrid setup times. These will be discussed later in this section.

	1-Gauss–Seidel		4-Gauss–Seidel		ILU	
	iters	AMG time	iters	AMG time	iters	AMG time
$Re = 20$	14	2.89	10	5.37	11	3.95
$Re = 50$	15	3.06	11	5.53	12	4.03
$Re = 100$	15	3.40	11	5.96	12	3.96
$Re = 200$	65	14.11	116*	62.12	12	4.82

TABLE 5.1

*Smoothed aggregation multigrid performance on 3D steady-state problems corresponding to $N = 64$ and $P = 100$. Average times (seconds) and iterations per convection–diffusion subproblem are given. (*Note: In the Re=200, 4-GS case, some of the convection–diffusion subproblems reached the maximum number of iterations of 200 without converging.)*

Table 5.2 gives the same information as Table 5.1, except using unsmoothed aggregation for the grid transfers. This corresponds to simple piecewise-constant interpolation. We see in Table 5.1 that the symmetric Gauss–Seidel smoother has difficultly converging when the Reynolds number is too large for the smoothed aggregation method. This is due to the grid transfers, which are built ignoring non-symmetric information. The resulting coarse grid discrete operators (constructed via (3.1)) can correspond to unstable discretizations for which the Gauss–Seidel method is divergent. This occurs on the coarsest grid for $Re = 200$ in Table 5.1 and helps explain why four Gauss–Seidel iterations perform worse than a single Gauss–Seidel iteration. Though unsmoothed aggregation generally gives poorer grid transfers (and non-mesh independent convergence), the coarse discretization stability problem does not arise. Thus, in some high Reynolds number cases, unsmoothed aggregation can actually perform better than smoothed aggregation. We are continuing to explore this issue and are working on combinations of smoothed and unsmoothed aggregation to handle convection–diffusion flows. For the remainder of the experiments in this paper we use smoothed aggregation with ILU smoothing in the solution of the discrete convection–diffusion equations as it is the most robust and gives good solution times.

In Table 5.3 we illustrate the breakdown of time spent within the saddle-point

	1-Gauss–Seidel		4-Gauss–Seidel		ILU	
	iters	AMG time	iters	AMG time	iters	AMG time
$Re = 20$	45	10.30	27	12.26	32	7.05
$Re = 50$	53	12.03	30	13.04	36	7.59
$Re = 100$	64	11.60	35	15.53	43	9.86
$Re = 200$	76	14.35	37	16.40	49	10.47

TABLE 5.2

Unsmoothed aggregation multigrid performance on 3D steady-state problems corresponding to $N = 64$ and $P = 100$. Average times (seconds) and iterations per convection–diffusion subproblem are given.

linear subproblem for a three-dimensional steady-state $Re = 100$ calculation. In each

	1-Gauss–Seidel	4-Gauss–Seidel	ILU
AMG setup	59.41	59.72	59.33
ILU factorization	N/A	N/A	74.58
matrix-vector products	116.08	623.48	183.85
smoother	423.79	1022.80	629.95
grid transfers	38.64	31.14	30.24
total	637.92	1737.10	977.95

TABLE 5.3

Breakdown of the total time spent in various parts of the solution of the saddle-point subproblem over a complete nonlinear, 3D, steady-state problem corresponding to $Re = 100$ and $N = 64$. Different AMG smoothers are shown for solution of the convection–diffusion subproblem.

case the multigrid times for separate solutions of the pressure Poisson and convection–diffusion subproblems are lumped together. In all cases, one symmetric Gauss–Seidel iteration is performed before and after the coarse grid correction within each V-cycle level for the solution of the pressure Poisson subproblem while results are shown with three different smoothers for the solution of the convection–diffusion subproblem. Overall, it is clear that the multigrid setup time is small. The grid transfer time is also small. Most of the time is spent computing the ILU factorization, applying the smoother, and performing matrix-vector products (this includes both residual calculations and within the Krylov solver). For the rest of this paper, we use the ILU smoother for the convection–diffusion subproblems and one symmetric Gauss–Seidel iteration before and after the coarse grid correction for the pressure Poisson subproblem.

We now begin to explore the performance of the block preconditioner. Table 5.4 demonstrates h-independent (i.e., mesh-independent) convergence on the two-dimensional steady-state problem. This table displays the average number of iterations per linear saddle-point subproblem of the Oseen iteration.

For moderate Reynolds numbers, 10 to 15 algebraic multigrid iterations are required to reach convergence on the convection–diffusion and pressure Poisson subproblems. In the $Re = 1000$ example, 15 to 30 iterations were required for convergence in the convection–diffusion subproblem. The number of saddle-point problem iterations is h-independent, which is in agreement with theory [31]. As expected, the number of iterations grows moderately with increasing Reynolds number.

Table 5.5 demonstrates h-independence on the three-dimensional steady-state problem. For the three-dimensional problems in this table, the convection–diffusion

N	8	16	32	64	128	256
$Re = 100$	12	14	15	16	16	17
$Re = 300$	18	22	25	27	27	30
$Re = 1000$	26	39	44	50	56	57

TABLE 5.4

2D steady-state results demonstrating h-independence. Average number of iterations to solve each linear saddle-point subproblem are shown. The convection–diffusion and pressure Poisson subproblems are solved exactly.

and pressure Poisson subproblems required 10 to 25 algebraic multigrid iterations for convergence to the given tolerance. As mentioned above, nonlinear difficulties for the three-dimensional lid driven cavity occur at much lower Reynolds numbers than in the two-dimensional case. In three dimensions, the nonlinear Oseen solver failed to converge for Reynolds numbers above 200 and converges quite poorly for Reynolds number 200 (see Tables 5.8 and 5.9). We will consider Newton's method (in conjunction with Oseen preconditioners) in a future work to address these difficulties.

N	8	16	32	64
$Re = 20$	8	9	9	10
$Re = 100$	13	15	17	18
$Re = 200$	17	20	22	23

TABLE 5.5

3D steady-state results demonstrating h-independence. Average number of iterations to solve each linear saddle-point subproblem are shown. The convection–diffusion and pressure Poisson subproblems are solved exactly.

In Tables 5.6–5.9 we compare steady-state solutions in which the convection–diffusion and pressure Poisson subproblems are solved exactly and inexactly within the preconditioner. For the exact solutions, the subproblems are solved to a tolerance of $\epsilon_F = \epsilon_A = 10^{-10}$. (This is how the results for Tables 5.4–5.5 were generated.) For the inexact solutions, we perform three, five, or seven iterations. All problems in Tables 5.6–5.9 were run with $N = 64$ (producing approximately one million degrees of freedom for the three-dimensional problems).

$N = 64$	A_p Exact, F Exact	A_p 3, F Exact	A_p 3, F 7	A_p 3, F 5	Oseen Steps
$Re = 100$	16	16	18	20	8
$Re = 300$	27	27	31	36	11
$Re = 1000$	56	57	255	290*	19*

TABLE 5.6

The average number of iterations per linear saddle-point subproblem are shown for exact vs. inexact solutions in the 2D steady-state problem. The last column shows the number of nonlinear iterations required for each solution. (Note: the "Ap 3, F 5" example took 32 Oseen steps and reached the maximum 300 saddle-point iterations.)*

Table 5.6 shows the average number of iterations per linear saddle-point problem and Table 5.7 shows the total CPU time to solution in the two-dimensional case. Tables 5.8 and 5.9 show the same information for the three-dimensional case. For moderate Reynolds numbers, solving the convection–diffusion and pressure Poisson

$N = 64$	A_p Exact, F Exact	A_p 3, F Exact	A_p 3, F 7	A_p 3, F 5
$Re = 100$	66.46	52.18	44.72	40.29
$Re = 300$	164.16	130.33	110.28	103.98
$Re = 1000$	1072.83	930.45	1675.38	2674.04

TABLE 5.7

Total CPU time to solution (in seconds) is shown for exact vs. inexact solutions in the 2D steady-state problem.

subproblems inexactly increases the average number of iterations per linear saddle-point problem, however the total time to solution improves due to the less expensive convection–diffusion and pressure Poisson solutions. In the $Re = 1000$ two-dimensional case, the convection–diffusion problem is much more difficult, and solving it inexactly increases the total time to solution. This is due to the GMRES/multigrid solver, which initially converges very slowly and then proceeds quite rapidly to the solution. Thus, while only approximately thirty iterations are required to obtain a solution, very little progress is made after just seven iterations. We expect stronger multigrid smoothers to resolve this difficulty but have not pursued this here.[4] It should also be noted that it may be possible to reuse Krylov vectors from previous convection–diffusion solutions to accelerate the overall convergence for the current convection–diffusion subproblem.

$N = 64$	A_p Exact, F Exact	A_p 3, F Exact	A_p 3, F 7	A_p 3, F 5	Oseen Steps
$Re = 20$	10	11	13	15	6
$Re = 100$	18	18	23	28	13
$Re = 200$	23	24	31	37	90

TABLE 5.8

Average number of iterations per linear saddle-point subproblem are shown for exact vs. inexact solutions in the 3D steady-state problem.

$N = 64$	A_p Exact, F Exact	A_p 3, F Exact	A_p 3, F 7	A_p 3, F 5
$Re = 20$	566.59	494.62	408.37	390.82
$Re = 100$	2503.07	2045.33	1487.17	1435.23
$Re = 200$	24,761.95	24,178.14	15,349.72	11,750.80

TABLE 5.9

Total CPU time to solution (in seconds) is shown for exact vs. inexact solutions in the 3D steady-state problem.

In the preceding steady-state examples, the difficulties encountered with large Reynolds number are largely due to the poor performance of the nonlinear Oseen iteration. One method of avoiding this difficulty is by introducing time stepping.

[4]In our case, the ILU method does not smooth certain modes on coarse grids. Modifications to ILU for multigrid smoothers discussed in [51] may improve the method as well as alternative grid transfers that better capture nonsymmetry.

In the next section, we examine the performance of the block preconditioner in the context of transient and pseudo-transient problems.

5.2. Transient Solver Results. In this section, transient solver performance is demonstrated. In our first set of experiments, a moderate range of CFL numbers are considered. Our main emphasis is to demonstrate how convergence of the method is relatively insensitive to CFL number. This implies that artificially small time increments are not required for the solver. Instead, time steps can be chosen based entirely on accuracy concerns and the time scales associated with the physics being resolved.

In all of the tables in this section, ten time steps are performed and averages are reported within each of the columns. Specifically, "Time" is the average time per step, "Oseen Steps" indicates the average number of nonlinear steps per time step, "Linear Solves" denotes the average number of linear saddle-point iterations per Oseen step, and "Ap" and "F" show the average number of multigrid iterations for each pressure Poisson, A_p, and convection–diffusion, F, subproblem. Table 5.10 illustrates performance for the case where the convection–diffusion and pressure Poisson subproblems are solved exactly.

$N = 64$	CFL	Time (secs)	Oseen Steps	Linear Solves	Ap	F
$Re = 500$	0.1	83.03	2	2	20	2
$Re = 500$	0.5	85.55	2	2	20	2
$Re = 500$	1	79.30	2	2	20	2
$Re = 500$	10	110.00	2	2	20	2
$Re = 500$	50	92.76	2	2	20	3
$Re = 500$	100	103.69	3	2	20	3

TABLE 5.10

Transient solver results on a 3D problem corresponding to $Re = 500$ for various CFL numbers. The columns show total CPU time to solutions (in seconds), the number of Oseen steps required for convergence of the nonlinear problem, the number of iterations required for convergence in the linear saddle-point problem and in the pressure Poisson and convection–diffusion subproblems.

We conclude this section with some results for very large CFL numbers. Table 5.11 illustrates performance for 'exact' solution of the convection–diffusion and pressure Poisson subproblem. These results are intended to be indicative of a pseudo-transient solver, where time stepping is introduced to improve the nonlinear Oseen iteration, and very large time steps are chosen to step quickly to steady-state. Ten pseudo-time steps are taken, and the results given are averaged per time step and per solve as in the previous table. Once again, good convergence rates are observed for the linear solvers and the iteration counts are relatively insensitive to CFL numbers. In this case, the nonlinear Oseen method performs acceptably and solutions are obtained for larger Reynolds numbers. The physical relevance of these higher Reynolds number solutions is unclear, as these Reynolds numbers approach regimes where the three-dimensional lid driven cavity no longer exhibits steady flows. The use of higher Reynolds numbers in this table and the one that follows is intended to demonstrate that our method does not preclude solving problems with higher Reynolds numbers when they are appropriate and physically relevant. It should be noted that this Oseen performance is achieved with very large time steps. However, as progressively larger time increments are chosen, the Oseen method eventually struggles as in the steady-state case.

$N = 64$	CFL	Time (secs)	Oseen Steps	Linear Solves	Ap	F
$Re = 500$	5000	238.55	5	5	18	5
$Re = 500$	10000	269.62	5	6	18	6
$Re = 500$	50000	403.88	6	9	19	8
$Re = 1000$	5000	212.30	5	5	18	4
$Re = 1000$	10000	235.97	5	6	18	5
$Re = 1000$	50000	524.83	7	10	19	9

TABLE 5.11

Pseudo-transient solver results on a 3D problem corresponding to $Re = 500$ and $Re = 1000$ for various large CFL numbers. The columns show total CPU time to solutions (in seconds), the number of Oseen steps required for convergence of the nonlinear problem, the number of iterations required for convergence in the linear saddle-point problem and in the pressure Poisson and convection–diffusion subproblems.

Table 5.12 explores the effects of 'inexact' solution of the subproblems. For the inexact solutions, the convection–diffusion subproblem is 'solved' with five iterations, and the pressure Poisson subproblem is 'solved' with three iterations. The last two columns of the table report the number of iterations required for an exact solution to the subproblems or the number of iterations specified in an inexact solution.

$N = 64$	Exact / Inexact	Time (secs)	Oseen Steps	Linear Solves	Ap	F
$Re = 500$	exact	403.88	6	9	19	8
$Re = 500$	inexact	357.70	6	12	3	5
$Re = 1000$	exact	524.83	7	10	19	9
$Re = 1000$	inexact	395.92	7	13	3	5

TABLE 5.12

Exact vs. inexact pseudo-transient solver results on a 3D problem corresponding to $Re = 500$ and $Re = 1000$ with $CFL = 50000$. The columns show total CPU time to solutions (in seconds), the number of Oseen steps required for convergence of the nonlinear problem, and iterations required for convergence in the linear saddle-point problem. In the exact solution cases, the last two columns show the number of iterations required for convergence of the pressure Poisson and convection–diffusion subproblems. In the inexact solution cases, these columns report the number iterations taken for the subproblems.

Tables 5.10–5.12 demonstrate that the multigrid method and the saddle-point preconditioner require very few iterations for these transient computations, and that the iteration counts are relatively insensitive to the CFL number. Unlike the large Reynolds number steady-state simulations, good nonlinear convergence is also obtained with the Oseen iteration for the transient and pseudo-transient problems. Thus a pseudo-transient strategy to obtain steady-state results would appear to mitigate, to some degree, the relatively slow convergence of the Oseen iteration at higher Reynolds numbers.

6. Conclusions. The multilevel block preconditioner presented and examined in this paper has been developed for linear systems arising from the implicit solution of the incompressible Navier–Stokes equations. The block preconditioner approximates the Schur complement (corresponding to pressure unknowns) using a convection–diffusion operator in the pressure space. This method requires component scalar block solvers that have similarities to pressure projection schemes and existing decoupled solution strategies. These component solves are based on a set of momentum convection-diffusion equations and a pressure Poisson-type problem. Unlike the pressure projection and fully-decoupled solution methods, the technique considered here does not suffer from overly restrictive time-step limitations for stability and the essential nonlinear coupling of the velocity and pressure variables can be retained. An important aspect of this preconditioner is the relative ease of implementation using existing software kernels.

In this study we have demonstrated mesh independent convergence in 2D and 3D of the saddle-point solver based on the Kay, Loghin, and Wathen [25] and Silvester, Elman, Kay, and Wathen [45] block preconditioner. The convergence of the saddle-point problem for transient problems was demonstrated to be fairly uniform over a wide range of Reynolds numbers and for CFL conditions (time steps size) that varied from time-accurate to pseudo-transient solutions. For steady-state problems a mild degradation is observed with increasing Reynolds number. This study extends the current literature by providing, three dimensional steady results and both steady and transient 2D and 3D results. These have been obtained with both serial and parallelalgorithms. Additionally, we have provided new results on the application of parallel smoothed aggregation AMG solvers to the momentum and pressure Poisson-type component block systems. This technique has been demonstrated to be an effective solver for these systems over a wide range of Reynolds numbers and CFL conditions.

While the overall results were obtained by employing an Oseen nonlinear iteration we believe they are more broadly applicable. Specifically this study is intended as a first step towards applying similar ideas within a more robust nonlinear solver such as Newton's method.

REFERENCES

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

[2] J. U. Brackbill and B. I. Cohen, eds., *Multiple time scales*, Academic Press, Orlando, 1985.

[3] J. Bramble, J. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.

[4] A. Brandt and N. Dinar, *Multigrid solutions to elliptic flow problems*, in Numerical Methods for Partial Differential Equations, S. V. Parter, ed., Academic Press, New York, 1979, pp. 53–147.

[5] W. L. Briggs, V. E. Henson, and S. McCormick, *A Multigrid Tutorial, Second Edition*, SIAM, Philadelphia, 2000.

[6] A. J. Chorin, *A numerical method for solving incompressible viscous problems*, J. Comp. Phys., 2 (1967), p. 12.

[7] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices*, SIAM J. Sci. Comput., 18 (1997), pp. 1657–1675.

[8] R. Codina, *Pressure stability in fractional step finite element methods for incompressible flow*, J. Comp. Phys., 170 (2001), pp. 112–140.

[9] T. S. Coffey, C. T. Kelley, and D. E. Keyes, *Pseudo-transient continuation and differential-algebraic-equations*, Submitted to SIAM J. Sci. Comput., (2002).

[10] E. Dean and R. Glowinski, *On some finite element methods for the numerical simulation of incompressible viscous flow*, in Incompressible Computational Fluid Dynamics, M. D. Gunzburger and R. Y. Nicolaides, eds., Cambridge University Press, New York, 1993, pp. 17–65.

[11] G. B. Deng, J. Piquet, P. Queutey, and M. A. Visonneau, *A new fully coupled solution of the Navier–Stokes equations*, Int. J. Num. Meth. Fluids, 19 (1994), pp. 605–639.

[12] G. B. Deng, J. Piquet, X. Vasseur, and M. A. Visonneau, *A new fully coupled method for computing turbulent flows*, Computers and Fluids, 30 (2001), pp. 445–472.

[13] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations*, Numer. Math., 90 (2002), pp. 665–688.

[14] P. M. Gresho, *On the theory of semi-implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix I: Theory*, Int. J. Num. Meth. Fluids., 11 (1990), p. 587.

[15] P. M. Gresho and R. Sani, *Incompressible Flow and the Finite Element Method*, Wiley, Chichester, 1998.

[16] W. D. Gropp, D. K. Kaushil, D. E. Keyes, and B. F. Smith, *High-performance parallel implicit CFD*, Par. Computing., 27 (2001), pp. 337–362.

[17] J. L. Guermond and L. Quartapelle, *On stability of convergence of projection methods based on pressure poisson equation*, Int. J. Num. Meth. Fluids, 26 (1998), pp. 1039–1053.

[18] H. Guillard and P. Vanek, *An aggregation multigrid solver for convection-diffusion problems on unstructured meshes*, Tech. Rep. Center for Comp. Math Report 130, Univ. of Denver, June 1998.

[19] W. Hackbusch, *Multigrid Methods and Applications*, vol. 4 of Computational Mathematics, Springer–Verlag, Berlin, 1985.

[20] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, The Physics of Fluids, 8 (1965), pp. 2182–2189.

[21] M. A. Heroux, *Trilinos/Petra: linear algebra services package*, Tech. Rep. SAND2001-1494W, Sandia National Laboratories, 2001.

[22] R. Issa, *Solution of the implicitly discretized fluid flows equations by operator splitting*, J. Comp. Phys, 62(1) (1986), pp. 40–65.

[23] J.A.Meijerink and H. V. der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix*, Math. of Comp., 31 (1977), pp. 148–162.

[24] G. E. Karniadakis, M. Israeli, and S. A. Orszag, *High-order splitting methods for the incompressible Navier–Stokes equations*, J. Comp. Phys., 97 (1991), pp. 414–443.

[25] D. Kay, D. Loghin, and A. J. Wathen, *A preconditioner for the steady-state Navier–Stokes equations.* To appear in SIAM J. Sci. Comput., 2002.

[26] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM, Philadelphia, 1995.

[27] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Num. Anal., 35 (1998), pp. 508–523.

[28] D. Keyes, A. Ecer, J. Periaux, and N. Satofuka, eds., *Parallel Computational Fluid Dynamics: Towards Teraflops, Optimization, and Novel Formulations: Proceedings of the Parallel CFD'99 Conference*, Elsevier, New York, 2000.

[29] D. A. Knoll, L. Chacon, L. G. Margolin, and V. A. Mousseau, *Nonlinearly consistent approximations for multiple time scale systems*, submitted to J. Comp. Phys, (April 2002).

[30] D. A. Knoll and V. A. Mousseau, *On Newton–Krylov multigrid methods for the incompressible Navier–Stokes equations*, J. Comp. Phys, 163 (2000), pp. 262–267.

[31] D. Loghin, *Analysis of preconditioned picard iterations for the Navier–Stokes equations*, Tech. Rep. 01/10, Oxford University Computing Laboratory, 2001.

[32] K. Long and M. Heroux, *The Trilinos Solver Framework*, in 2002 ACTS Workshop Proceedings, Berkeley, CA, Sept 3–6, 2002, O. Marques and T. Drummond, eds., 2002.

[33] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[34] R. A. Nicolaides, *Analysis and convergence of the MAC scheme I*, SIAM J. Numer. Anal., 29 (1992), pp. 1579–1591.

[35] E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow*, Cambridge Univ. Press, Cambridge, 2001.

[36] S. V. Patankar, *Numerical heat transfer and fluid flow*, Hemisphere Pub. Corp, New York, 1980.

[37] S. V. Patankar and D. A., *A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows*, Int. J. Heat and Mass Trans., 15 (1972), pp. 1787–1806.

[38] M. Pernice and M. D. Tocci, *A multigrid- preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations*, SIAM J. Sci. Comput., 123 (2001), pp. 398–418.

[39] J. B. Perot, *An analysis of the fractional step method*, J. Comp. Phys., 108 (1993), pp. 51–58.

[40] A. Quateroni, F. Saleri, and A. Veneziani, *Factorization methods for the numerical approximation of Navier–Stokes equations*, Comput. Meth. Appl. Mech. Eng., 188 (2000), pp. 505–526.

[41] J. N. Shadid, *A fully-coupled Newton–Krylov solution method for parallel unstructured finite element fluid flow, heat and mass transfer simulations*, Int. J. CFD, 12 (1999), pp. 199–211.

[42] J. N. Shadid, S. A. Hutchinson, G. L. Hennigan, H. K. Moffat, K. D. Devine, and

A. G. Salinger, *Efficient parallel computation of unstructured finite element reacting flow solutions*, Par. Computing., 23 (1997), pp. 1307–1325.

[43] J. N. Shadid, R. S. Tuminaro, K. D. Devine, and P. T. Lin, *Parallel preconditioned Krylov solvers for unstructured finite element reacting flow solutions*, To be submitted to J. Comp. Phys, (2002).

[44] P. N. Shankar and M. D. Deshpande, *Fluid mechanics in the driven cavity*, Annu. Rev. Fluid Mech, 32 (2000), pp. 93–136.

[45] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, J. Comp. Appl. Math., 128 (2001), pp. 261–279.

[46] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems. Part II: Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994).

[47] H. D. Simon, ed., *Parallel CFD'90; Parallel Computational Fluid Dynamics: Implementation and Results*, MIT Press, 1992.

[48] A. Smith and D. Silvester, *Implicit algorithms and their linearisation for the transient incompressible Navier–Stokes equations*, IMA J. Numer. Anal., 17 (1997), pp. 527–545.

[49] J. C. Strikwerda and Y. S. Lee, *The accuracy of the fractional step method*, SIAM J. Numer. Anal., 37 (1999), pp. 37–47.

[50] C. Tong, R. Tuminaro, K. Devine, and J. Shadid, *Design of a Multilevel Preconditioning Module for Unstructured Calculations*, Tech. Rep. in preparation, Sandia National Laboratories, Albuquerque NM, 87185, 2000.

[51] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, UK, 2001.

[52] R. Tuminaro and C. Tong, *Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines*, in SuperComputing 2000 Proceedings, J. Donnelley, ed., 2000.

[53] R. S. Tuminaro, C. H. Tong, J. N. Shadid, K. D. Devine, and D. M. Day, *On a multilevel preconditioning module for unstructured mesh Krylov solvers: two-level Schwarz*, Comm. Num. Meth. Eng., 18 (2002), pp. 383–389.

[54] S. Turek, *A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: from fully implicit non-linear schemes to semi-implicit projection methods*, Int. J. Numer. Meth. Fluids, 22 (1996), pp. 987–1011.

[55] H. A. van der Vorst and C. Vuik, *GMRESR: a family of nested GMRES methods*, Numerical Linear Algebra with Applications, 1 (1994), pp. 369–386.

[56] P. Vanek, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numerische Mathematik, 88 (2001), pp. 559–579.

[57] P. Vanek, M. Brezina, and R. Tezaur, *Two-grid method for linear elasticity on unstructured meshes*, SIAM J. Sci. Comp., 21 (1999), pp. 900–923.

[58] P. Vanek, A. Janka, and H. Guillard, *Convergence of the algebraic multigrid based on smoothed aggregation II: Extension to a petrov-galerkin method*, Tech. Rep. Report no. 3683, INRIA, May 1999.

[59] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[60] P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, West Sussex, 1992.