A COMPARISON OF PARALLEL BLOCK MULTI-LEVEL PRECONDITIONERS FOR THE INCOMPRESSIBLE NAVIER–STOKES EQUATIONS

ROBERT SHUTTLEWORTH*

Abstract. Over the past several years, considerable effort has been placed on developing efficient solution algorithms for the incompressible Navier–Stokes equations. The effectiveness of these methods requires that the solution techniques for the linear subproblems generated by these algorithms exhibit robust and rapid convergence; These methods should be insensitive to problem parameters such as mesh size and Reynolds number. This study concerns a preconditioner derived from a block factorization of the coefficient matrix generated in a Newton nonlinear iteration for the primitive variable formulation of the system. This preconditioner is based on the approximation of the Schur complement operator using a technique proposed by Kay, Loghin, and Wathen [11] and Silvester, Elman, Kay, and Wathen [18]. It is derived using subsidiary computations (solutions of pressure Poisson and convection–diffusion–like subproblems) that are significantly easier to solve than the entire coupled system, and a solver can be built using tools, such as smooth aggregation multigrid for the subproblems. We discuss a computational study performed using MPSalsa, a stabilized finite element code, in which parallel versions of these preconditioners from the pressure convection-diffusion preconditioners are compared with an overlapping Schwarz domain decomposition preconditioner. Our results show nearly ideal convergence rates for a wide range of Reynolds numbers on two-dimensional problems with both enclosed and in/out flow boundary conditions on both structured and unstructured meshes.

1. Introduction. We are concerned with the incompressible Navier–Stokes equations

$$\begin{aligned} -\nu\nabla^2\mathbf{u} + (\mathbf{u}\cdot\operatorname{grad})\,\mathbf{u} + \operatorname{grad} p &= \mathbf{f} \\ -\operatorname{div}\mathbf{u} &= 0 \end{aligned} \quad \text{in } \Omega \subset \mathbb{R}^2, \tag{1}$$

where the velocity, $\mathbf{u}$, satisfies suitable boundary conditions on $\partial\Omega$. Our focus is on developing robust, scalable, and efficient solution algorithms for the systems of equations that arise after linearization of the system (1).

The block coefficient matrix that results from linearization and discretization of the incompressible Navier-Stokes equations, has the form

$$A = \begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix}. \tag{2}$$

The strategies we employ for solving (2) are derived from the LDU block factorization of this coefficient matrix

$$A = \begin{pmatrix} I & 0 \\ \hat{B}F^{-1} & I \end{pmatrix} \begin{pmatrix} F & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & F^{-1}B^T \\ 0 & I \end{pmatrix} \tag{3}$$

where $S = C + \hat{B}F^{-1}B^T$ is the Schur complement (of F in A). Use of the exact Schur complement is not feasible, but replacement of S with carefully derived approximations leads to efficient *preconditioning* strategies for use with iterative solvers.

The remainder of this paper is organized as follows. Section 2 gives some brief background on the Newton iteration and provides an overview on the type of discretization and resulting coefficient matrix used for our numerical experiments. Section 3 describes the pressure convection-diffusion preconditioner and the merits of this choice of preconditioner. Section 4 provides an overview of 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.

*Applied Mathematics and Scientific Computing Program and Center for Scientific Computation and Mathematical Modeling, University of Maryland, College Park, MD 20742. rshuttle@math.umd.edu. This work was partially supported by the National Science Foundation under grant DMS0208015, the Department of Energy under grant DOEG0204ER25619, and the ASC Program at Sandia National Laboratories under contract number DE-AC04-94AL85000. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration.

2. Background. Our focus is on solution algorithms for the systems of equations that arise after linearization of the system (1). We will use a nonlinear iteration based on an inexact Newton-Krylov method to solve this problem. If we write the nonlinear problem to be solved as $F(x) = 0$, where $F : \mathbf{R}^n \to \mathbf{R}^n$. At the k^{th} step of Newton's method, the solution of the linear *Newton equation*

$$J(x_k)s_k = -F(x_k), \tag{4}$$

where x_k is the current solution and $J(x_k)$ denotes the Jacobian matrix of F at x_k. Once the Newton step, s_k, is determined, the approximation is updated via

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

Newton-Krylov methods relax the requirement to solve (4) exactly. A Krylov subspace method is used to compute an iterate, s_k, that satisfies the *inexact Newton condition*,

$$\|F(x_k) + J(x_k)s_k\| \leq \eta_k \|F(x_k)\|, \tag{5}$$

where the *forcing term*, $\eta_k \in [0, 1]$. If $\eta_k = 0$, this is Newton's method. There are many different means for choosing η_k, for a discussion of the merits of each of these choices, see [2]. For our computational study, we have chosen η_k to be a constant. We solve for the Newton step, s_k, using the Krylov subspace method, GMRES [14].

We will discretize the Navier-Stokes equations with a stabilized finite element method for which the Jacobian system at the kth step that arises from Newton's method has the form

$$\begin{pmatrix} F_k & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} \mathbf{\Delta u_k} \\ \Delta p_k \end{pmatrix} = \begin{pmatrix} \mathbf{f_k} \\ g_k \end{pmatrix} \tag{6}$$

where F_k resembles a discrete convection-diffusion operator, B^T is a discrete gradient operator, $\hat{B}$ is a small perturbation of the discrete divergence operator, and C is an operator that stabilizes the finite element discretization.

The right hand side of the above expression represents the k^{th} residual of the nonlinear system, which has the form

$$\begin{pmatrix} \mathbf{f_k} \\ g_k \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix} - \begin{pmatrix} F_k & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} \mathbf{u_k} \\ p_k \end{pmatrix}. \tag{7}$$

The condition (5) then becomes

$$\left\| \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix} - \begin{pmatrix} F_k & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} \mathbf{u_k} \\ p_k \end{pmatrix} \right\| \leq \eta_k \left\| \begin{pmatrix} \mathbf{f_k} \\ g_k \end{pmatrix} \right\|. \tag{8}$$

3. Pressure Convection-Diffusion Preconditioner. The strategy we will use to solve (2) is derived by grouping the block diagonal and upper triangular factors shown in (3). The block factorization of the coefficient matrix is then

$$\begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix} = \begin{pmatrix} I & 0 \\ \hat{B}F^{-1} & I \end{pmatrix} \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}. \tag{9}$$

This implies that

$$\begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ \hat{B}F^{-1} & I \end{pmatrix}, \tag{10}$$

which suggests a preconditioning strategy for (2). If it were possible to use the matrix

$$\mathcal{Q} = \begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix} \tag{11}$$

as a right-oriented preconditioner, then the preconditioned operator would be the matrix on the right in (10). The efficacy of this preconditioner choice can be seen by analyzing the following generalized eigenvalue problem:

$$\begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \lambda \begin{pmatrix} F & B^T \\ 0 & S \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix},$$

whose eigenvalues are identically one. The preconditioned operator contains Jordan blocks of dimension at most 2, so at most two iterations of a preconditioned GMRES iteration would be needed to solve the system [13].

To apply this preconditioner $\mathcal{Q}$ in a Krylov subspace iteration, at each step the application of $\mathcal{Q}^{-1}$ to a vector is needed. By expressing $\mathcal{Q}$ in factored form,

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

the computational issues involved for the particular choice (11) can be seen. Two potentially difficult operations are required to apply $\mathcal{Q}^{-1}$ to a vector: S^{-1} must be applied to a vector in the discrete pressure space, and F^{-1} must be applied to a vector in the discrete velocity space. The application of F^{-1} can be performed relatively cheaply using iterative techniques, such as multigrid. However, applying S^{-1} to a vector is too expensive. So, we will discuss how an effective preconditioner can be built by replacing this operation with an inexpensive approximation. This gives rise to the pressure convection-diffusion preconditioning strategy.

To derive the pressure convection-diffusion strategy, we will begin by defining a new operator, denoted F_p, used in approximating the Schur complement, using the following premise:

Suppose that we begin with a discrete version of a convection-diffusion operator derived by linearizing the nonliner term in (1),

$$(\nu\nabla^2 + (\mathbf{w} \cdot \text{grad})).$$

Here $\mathbf{w}$ can be viewed as an approximation to the velocity from a previous nonlinear iteration. Suppose that there is an analogous operator that is defined on the pressure space,

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

Consider the commutator of these operators with the gradient:

$$\epsilon = (\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))\nabla - (\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))_p\nabla. \tag{12}$$

Supposing that ϵ is small, then we can rewrite (12) as

$$(\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))\nabla \approx \nabla(\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))_p$$
$$\nabla(\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))_p^{-1} \approx (\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))^{-1}\nabla$$

Then after multiplying on both sides of the above equation by the divergence operator

$$\nabla^2(\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))_p^{-1} \approx \nabla \cdot (\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))^{-1}\nabla \tag{13}$$

In discrete form, this becomes

$$\begin{aligned}(Q_p^{-1}A_p)(Q_p^{-1}F_p)^{-1} &\approx (Q_p^{-1}B)(Q_v^{-1}F)^{-1}(Q_v^{-1}B^T)\\ A_pF_p^{-1} &\approx Q_p^{-1}(BF^{-1}B^T)\\ A_pF_p^{-1} &\approx Q_p^{-1}S\end{aligned}$$

where here F represents a discrete convection-diffusion operator on the velocity space, F_p is the discrete convection-diffusion operator on the pressure space, A_p is a discrete Laplacian operator, Q_v the velocity mass matrix, and Q_p is the pressure mass matrix. This suggests a suitable Schur complement approximation for a finite element discretization when $C = 0$. In the case of stabilized finite element discretizations, the discrete version of (13) is

$$C + BF^{-1}B^T \approx A_pF_p^{-1}Q_p.$$

The preconditioner for (2) then becomes

$$\begin{pmatrix} F & B^T \\ \hat{B} & -C \end{pmatrix} \approx \begin{pmatrix} F & B^T \\ 0 & A_pF_p^{-1}Q_p \end{pmatrix}.$$

Applying the inverse of $A_pF_p^{-1}Q_p$ to a vector requires solving a system of equations with a discrete Laplacian operator, then multiplication by the matrix F_p, and solving a system of equations with the pressure mass matrix. Both the convection-diffusion-like system (with coefficient matrix F), and the Schur complement system (with coefficient matrix $A_pF_p^{-1}Q_p$), solves can be approximated using multigrid with little deterioration of effectiveness.

Considerable empirical evidence for two-dimensional problems indicates that this preconditioning strategy is effective, leading to convergence rates that are independent of mesh size and mildly dependent on Reynolds numbers for steady problems [4, 7, 11, 18]. A proof that convergence rates are independent of the mesh is given in [12]. One drawback is that many industrial codes do not provide the F_p operator. Means for generating this operator directly from the coefficient matrix can be found in [5].

4. Implementation. A pressure stabilized, streamwise upwinded Petrov Galerkin least squares finite element scheme [19] with $Q_1 - Q_1$ elements is used to discretize the incompressible Navier-Stokes equations. This scheme couples together the momentum equation with the incompressiblity constraint, thus giving rise to a nonlinear, coupled, nonsymmetric system of equations. This stabilized discretization is first-order accurate and is provided by the MPSalsa [15] chemically reactive fluid flow code developed at Sandia National Laboratories. One advantage for using this finite element scheme is that the velocity and pressure degrees of freedom are defined at the same grid points, so equal order interpolants for both velocity and pressure are used.

For the Newton equations, the structure of F is a 2×2 block matrix with the form

$$F = \begin{pmatrix} -\nu\Delta + u^{(n-1)} \cdot \nabla + (u_1^{(n-1)})_x & (u_1^{(n-1)})_y \\ (u_2^{(n-1)})_x & -\nu\Delta + u^{(n-1)} \cdot \nabla + (u_2^{(n-1)})_y \end{pmatrix}. \tag{14}$$

For the A_p operator required by this strategy, we choose it by taking $1/\nu$ times the symmetric part of F_p. This generates a Laplacian type operator suitable for the use in this preconditioning strategy. For Q_p, we use a lumped version of the pressure mass matrix. All of these operators, including the F_p operator, are generated by the application code, MPSalsa. For problems with inflow boundary conditions, such as the flow over a diamond obstruction, we specify Dirichlet boundary conditions on the inflow boundary for all of the preconditioning operators. For singular operators found in

problems with enclosed flow, we pin all of the operators as the stabilization matrix is pinned [3, ch. 8].

One key to the pressure convection-diffusion style of preconditioning is that it requires two subsidiary computations (solutions of pressure Poisson and convection-diffusion subproblems) that are significantly easier to solve than the entire coupled system. Both of these computations are amenable to being solved with multigrid methods. We employ smoothed aggregation algebraic multigrid (AMG) for these computations because AMG does not require mesh or geometric information, and thus are attractive for problems posed on complex domains or unstructured meshes. More details on AMG can be found at [22, 24].

The implementation of the preconditioned Krylov subspace solution algorithm was done using *Trilinos* [10]. This project is an effort at Sandia National Laboratories to develop parallel solution algorithms in an object-oriented collection of software packages for the solution of large-scale multiphysics simulations. The pressure convection-diffusion preconditioner detailed in this study is implemented in the package *Meros*. This package provides scalable block preconditioning for the Navier-Strokes equations. One advantage of using Trilinos is its capability to seamlessly use other Trilinos packages for core operations, such as *Epetra* for matrix vector products, *AztecOO* for the Krylov subspace iteration, and *ML* for algebraic multigrid preconditioning. Meros uses the Epetra package for basic linear algebra functions. Epetra also facilitates matrix construction on parallel distributed machines. Each processor constructs the subset of matrix rows assigned to it via the static domain decomposition partitioning generated by stand-alone library, CHACO [9], and a local matrix-vector product is defined. Epetra handles all the distributed parallel matrix details (e.g. local indices versus global indices, communication for matrix-vector products, etc.). Once the matrices F, B, $\hat{B}$, and C are defined, a global matrix-vector product for (6) is defined using the matrix-vector products for the individual systems. Construction of the preconditioner follows in a similiar 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 solving (6), for the F, and Schur complement approximation subsystems) are supplied by AztecOO [21] which we access through Trilinos [10]. The multigrid preconditioning for the subsystems is done by ML [20], an algebraic multigrid preconditioning package, which we also 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 Jacobian system, the convection–diffusion, and pressure Poisson subproblems. For the pressure Poisson problem, we use CG preconditioned with four levels of algebraic multigrid, and for solving the system with coefficient matrix F we use GMRES preconditioned with three levels of algebraic multigrid. For the pressure Poisson problem, a multilevel smoother polynomial was used for the smoothing operations and for the convection-diffusion problem,a block Gauss Seidel smoother was used [1]. To solve the linear problem associated with each Newton iteration, we use GMRESR, a variation on GMRES proposed by van der Vorst and Vuik [23] allowing the preconditioner to vary at each iteration. GMRESR is required because we used a multigrid preconditioned Krylov subspace method to generate approximate solutions in the subsidiary computations (pressure Poisson and convection-diffusion-like) of the preconditioner, so the preconditioner is not a fixed linear operator.

In our experiments, we compare the pressure convection-diffusion preconditioner with a one-level Schwarz domain decomposition preconditioner [16]. This preconditioner does not vary from iteration to iteration (as the pressure convection-diffusion does), so GMRES can be used as the outer solver. Domain decomposition methods are based upon computing approximate solutions on subdomains. Robustness can be improved by increasing the coupling between processors, thus expanding the original subdomains to include unknowns outside of the processor's assigned nodes. Again, the original Jacobian system matrix is partitioned into subdomains using CHACO, whereas AztecOO is used to implement the one-level Schwarz method and automatically construct the overlapping

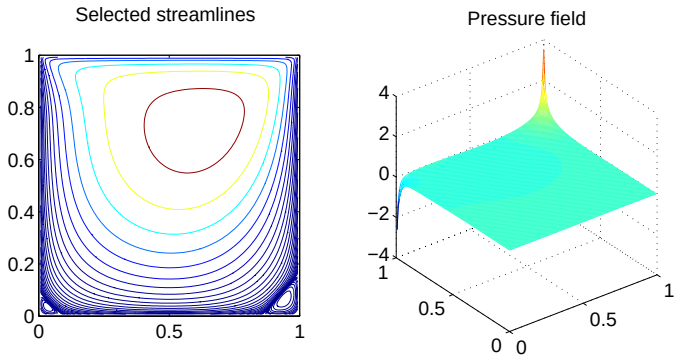

FIG. 1. *Sample velocity field and pressure field from 2D lid driven cavity.* $h = 1/128$, $Re = 100$.

submatrices. Instead of solving the submatrix systems exactly we use an incomplete factorization technique on each subdomain (processor). For our experiments, we used an ILUT with a fill-in of 1.0 and a drop tolerance of 0.0. Therefore, the ILU factors have the same number of nonzeros as the original matrix with no entries dropped [16].

5. Numerical Results. For our computational study, we have focused our efforts on two benchmark flow problems, the lid driven cavity problem and flow over a diamond obstruction. For the driven cavity, we consider a two-dimensional model of a square region with unit length sides. Velocities are zero on all edges except the top (lid), which has a driving velocity of one. This problem is then discretized on a uniform mesh of width h. In two dimensions, we have approximately $3/h^2$ unknowns, i.e. $1/h^2$ pressure and $2/h^2$ velocity unknowns.

For the diamond obstruction, we consider a two dimensional rectangular region with unit length width and a channel length of seven units, where the fluid flows in one side of a channel, then around the obstruction and out the other end of the channel. Velocities are zero along the top and bottom of the channel, and along the diamond obstruction. The flow is set with a parabolic inflow condition and a natural outflow condition, i.e. $\frac{\partial u_x}{\partial x} = p$ and $\frac{\partial u_y}{\partial y} = 0$. This model is discretized on an unstructured mesh.

The two-dimensional lid driven cavity is a well-known benchmark for fluids problems because it contains many features of harder flows, such as recirculations. The lid driven cavity poses challenges to both linear and nonlinear solvers and exhibits unsteady solutions and multiple solutions at high Reynolds numbers. In two dimensions, unsteady solutions appear around Reynolds number 7000 to 10,000 [17]. Figure 1 shows the velocity field and pressure field for an example solution to a two-dimensional lid driven cavity problem with $h = 1/128$.

The two-dimensional flow over a diamond obstruction also poses many difficulties for both linear and nonlinear solvers. It contains many important features appearing in realistic flows, including an unstructured mesh with inflow and outflow conditions. In two dimensions, unsteady solutions appear around Reynolds number 50 [8].

For the lid driven cavity problem, we terminate the nonlinear iteration when the relative solution error is 10^{-5}, i.e.

$$\left\| \begin{pmatrix} \mathbf{f} - (F\mathbf{u} + B^T p) \\ g - (\hat{B}\mathbf{u} - Cp) \end{pmatrix} \right\| \leq 10^{-5} \left\| \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix} \right\|. \tag{15}$$

The tolerance for (8), the Jacobian solver, is fixed at 10^{-5}. For the diamond obstruction, the nonlinear solution error (15) is 10^{-3} and the tolerance for (8) is 10^{-3}. For both problems, we employ inexact solves on the subsidiary pressure Poisson type and convection-diffusion subproblems. For

solving the system with coefficient matrix A_p, we use six iterations of algebraic multigrid preconditioned CG and for the convection-diffusion-like subproblem, with coefficient matrix F, we fix a tolerance of 10^{-2}, i.e. this iteration is terminated when $\|(y - F\mathbf{u})\| \leq 10^{-2}\|y\|$ [6]. We compare this method to a one-level overlapping Schwarz domain decomposition preconditioner that uses GMRES to solve the Jacobian system (8) at each step using the same tolerances. For both preconditioners, we use a Krylov subspace of 300 and a maximum number of iterations of 3000. All two-dimensional results were obtained in parallel on Sandia's Institutional Computing Cluster (ICC). Each of this cluster's compute nodes are dual Intel 3.6 GHz Xenon processors with 2GB of RAM.

5.1. Experimental Results.

5.1.1. Lid Driven Cavity Problem. We first compared the performance of the pressure convection-diffusion preconditioner to the domain decomposition preconditioner on the 2D lid driven cavity problem generated by MPSalsa. In the first column of Table 1, we list the Reynolds number followed by three mesh sizes in column two. In columns three and four, we list the total CPU time and the average number of outer linear iterations per Newton step for the pressure convection-diffusion and domain decomposition preconditioners, respectively. For the pressure convection-diffusion preconditioner, we notice iteration counts that are largely independent of mesh size for a given Reynolds number. As the mesh is refined, we do notice an increase in the computational time for a given Reynolds number. This is due to a number of reasons, including an increase in communication time amongst processors and an increase in the difficulty of solving the linear system with coefficient matrix F to a specified tolerance for higher Reynolds numbers. The domain decomposition preconditioner does not display mesh independent convergence behavior as the mesh is refined. However, there is much less computational effort involved in one iteration of preconditioning with domain decomposition than in one iteration of preconditioning with pressure convection-diffusion. For the fine meshes, we notice the CPU time for the pressure convection-diffusion preconditioner is a factor of 5 faster than domain decomposition.

Re Number	Mesh Size	Pressure Convection-Diffusion		Domain Decomposition		Procs
		iters	time	iters	time	
$Re = 10$	64×64	26.0	19.7	79.4	19.4	1
	128×128	26.0	68.9	220.6	91.4	4
	256×256	32.0	114.9	1018.6	596.6	16
$Re = 100$	64×64	34.2	30.2	86.5	26.4	1
	128×128	35.9	59.8	300.3	150.2	4
	256×256	41.3	156.1	1603.9	1326.6	16
$Re = 500$	64×64	109.2	490.2	89.7	44.4	1
	128×128	92.2	606.0	334.9	258.8	4
	256×256	98.0	948.7	5433.1	4543.9	16

TABLE 1

Comparison of the iteration counts and CPU time for the pressure convection-diffusion and domain decomposition preconditioners for the lid driven cavity problem.

5.1.2. Flow over a Diamond Obstruction. We will now compare the pressure convection-diffusion preconditioner to the domain decomposition preconditioner on the flow over a diamond obstruction problem generated by MPSalsa. In the first column of Table 2, we list the Reynolds number followed by the number of unknowns for four problem sizes in column two. In columns three and four, we list the total CPU time and the average number of outer linear iterations per Newton step for the pressure convection-diffusion and domain decomposition preconditioners, respectively. We see many similiar trends to the results from the driven cavity problem, mainly iteration counts

that are largely independent of mesh size for a given Reynolds number and an increase in the computational time as the mesh size is refined. The domain decomposition preconditioner does not display mesh independent convergence behavior as the mesh is refined. For Re 10 and Re 25, the pressure convection-diffusion preconditioner was faster in all cases. For Re 40, it was faster for all meshes except for the small problems with 62,000 unknowns run on one processor. Note that the GMRES solver preconditioned with domain decomposition stagnated and did not converge to a solution for the problems with 4 million unknowns. The pressure convection-diffusion preconditioner converged without difficulty on this problem. On modest sized problems where both methods converged, the pressure convection-diffusion preconditioner ranged from 4 to 15 times faster than domain decomposition.

Re Number	Unknowns	Pressure Convection-Diffusion		Domain Decomposition		Procs
		iters	time	iters	time	
$Re = 10$	62K	20.5	138.8	110.8	186.6	1
	256K	22.5	266.2	284.6	1657.4	4
	1M	22.9	501.0	1329.0	7825.5	16
	4M	29.4	1841.7	NC	NC	64
$Re = 25$	62K	32.9	248.0	101.7	198.8	1
	256K	35.9	480.6	273.8	1583.1	4
	1M	38.3	956.9	1104.8	7631.5	16
	4M	52.0	4189.8	NC	NC	64
$Re = 40$	62K	54.6	565.8	70.4	267.2	1
	256K	70.1	1280.9	203.9	1420.7	4
	1M	65.4	2011.7	997.1	8188.2	16
	4M	79.8	9387.9	NC	NC	64

TABLE 2

Comparison of the iteration counts and CPU time for the pressure convection-diffusion and domain decomposition preconditioners for the flow over a diamond obstruction. NC stands for no convergence.

6. Conclusions. The pressure convection-diffusion multilevel block preconditioner presented and studied in this paper has been developed for linear systems generated from the solution of the incompressible Navier-Stokes equations. The key to this preconditioner is the approximation of the Schur complement using a convection-diffusion operator defined on the pressure space. This method requires block solves (convection diffusion and pressure Poisson-type) that are significantly easier to solve than the entire coupled system.

In this study, we have demonstrated asymptotic convergence that is mesh independent in 2D for problems generated by an application code, MPSalsa, over a range of Reynolds numbers and problems discretized on structured and unstructured meshes with inflow and outflow conditions. For the steady-state problems, the iteration counts see a slight degradation for increasing Reynolds number.

While the overall results are promising, the CPU times can be greatly improved by employing a more efficient convection-diffusion solver. In the future, we intend to further expand this technique to time dependent problems and problems posed on more complex domains.

7. Acknowledgements. I would like to thank Howard Elman, my thesis advisor at the University of Maryland, for all of his invaluable comments and encouragement throughout this project. I would also like to thank Vicki Howle, John Shadid, and Ray Tuminaro, of Sandia National Laboratories, for their guidance and willingness to work with me on these block preconditioners.

REFERENCES

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, *Parallel multigrid smoothing: Polynomial versus Gauss-Seidel*, Journal of Computational Physics, 188 (2003), pp. 593–610.

[2] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[3] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.

[4] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1299–1316.

[5] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM Journal on Scientific Computing, to appear (2006).

[6] H. C. Elman, V. E. Howle, J. Shadid, and R. Tuminaro, *A parallel block multi-level preconditioner for the 3D incompressible Navier-Stokes equations*, Journal of Computational Physics, 180 (2003).

[7] 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.

[8] B. Fornberg, *Computing incompressible flows past blunt bodies–a historical overview*, Numerical Methods for Fluid Dynamics IV, IV (1993).

[9] B. Hendrickson and R. Leland, *A users guide to Chaco, version 1.0.*, Tech. Report SAND93-2339, Sandia National Laboratories, 1993.

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

[11] D. Kay, D. Loghin, and A. J. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 237–256.

[12] D. Loghin, A. Wathen, and H. Elman, *Preconditioning techniques for Newton's method for the incompressible Navier-Stokes equations*, BIT, 43 (2003), pp. 961–974.

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

[14] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[15] J. Shadid, A. Salinger, R. Schmidt, T. Smith, S. Hutchinson, G. Hennigan, K. Devine, and H. Moffat., *MPSalsa version 1.5: A finite element computer program for reacting flow problems*, tech. report, Sandia National Laboratories, 1998.

[16] J. Shadid, R. Tuminaro, K. Devine, G. Hennigan, and P. Lin, *Performance of fully-coupled domain decomposition preconditioners for finite element transport/reaction simulations*, Journal of Computational Physics, 205 (2005), pp. 24–47.

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

[18] 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.

[19] T. E. Tezduyar, *Stabilized finite element formulations for incompressible flow computations*, Advances in Applied Mechanics, 28 (1991), pp. 1–44.

[20] C. Tong, R. Tuminaro, K. Devine, J. Shadid, and D. Day, *On a multilevel preconditioning module for unstructured mesh Krylov solvers: Two-level Schwarz*, Comm. Num. Meth. Eng., 18 (2002), pp. 363–389.

[21] R. Tuminaro, M. Heroux, S. Hutchinson, and J. Shadid, *Official Aztec user's guide: Version 2.1*, Tech. Report Sand99-8801J, Sandia National Laboratories, Albuquerque NM, 87185, Nov 1999.

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

[23] 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.

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