FAST SOLVERS FOR FINITE DIFFERENCE APPROXIMATIONS FOR THE STOKES AND NAVIER-STOKES EQUATIONS

by

Dongho Shin and John C. Strikwerda

Computer Sciences Technical Report #1034

August 1991

Fast Solvers for Finite Difference Approximations for the Stokes and Navier-Stokes Equations

Dongho Shin † and John C. Strikwerda‡

Abstract. We consider several methods for solving the linear equations arising from finite difference discretizations of the Stokes equations. The two best methods, one presented here for the first time, apparently, and a second, presented by Bramble and Pasciak, are shown to have computational effort that grows slowly with the number of grid points. The methods work with second-order accurate discretizations. Computational results are shown for both the Stokes and incompressible Navier-Stokes at low Reynolds number.

Key words. Stokes equations, Navier-Stokes equations, finite difference schemes, iterative methods

AMS(MOS) subject classifications. 65N20, 65F10

1. Introduction.

The steady-state Stokes equations in R^d are

$$\begin{aligned} \nabla^2 \vec{u} - \vec{\nabla} p &= \vec{f} \\ \vec{\nabla} \cdot \vec{u} &= g \end{aligned} \qquad \text{in} \qquad \Omega \subset R^d. \tag{1.1}$$

In almost all applications the function g in the second equation of (1.1) is zero, but the methods discussed here do not require that g be zero, so we include this slightly more general case. We consider only the the Dirichlet boundary condition

$$\vec{u} = \vec{b} \qquad \text{on} \qquad \partial\Omega.$$

The methods discussed here should be easy to extend to other boundary conditions. The velocity $\vec{u}$ is a vector of dimension d and the pressure p is a scalar. The domain of our computational examples is the unit square in R^2. In current research we are using these methods on more general domains. For a discussion of the mathematical theory related to the Stokes equations see the book by Temam [16].

Let A_h, G_h and D_h be the matrices generated by discretizations of the differential operators $(-\nabla^2), \vec{\nabla}$ and $(-\vec{\nabla}\cdot)$ respectively. The discretization of (1.1) may be written as

$$\begin{pmatrix} A_h & G_h \\ D_h & 0 \end{pmatrix} \begin{pmatrix} u_h \\ p_h \end{pmatrix} = \begin{pmatrix} f_h \\ g_h \end{pmatrix}. \tag{1.2}$$

† Department of Mathematics, University of Wisconsin-Madison, Madison, WI 53706.

‡ Department of Computer Sciences and Center for the Mathematical Sciences, University of Wisconsin-Madison, Madison, WI 49506. The work of this author was supported by the U.S.Army Research Office under grants DAAL03-87-K-0028 and DAAL03-91-G-0094.

In some formulations of the difference equations, e.g., staggered grids and finite element formulations, the matrix D_h is the transpose of G_h, i.e., $D_h = G_h^T$. However, in many cases this is not true, either because of boundary conditions or because of the difference schemes. In this paper we do not assume that $D_h = G_h^T$. Note that

$$\det \begin{pmatrix} A_h & G_h \\ D_h & 0 \end{pmatrix} = \det(A_h^{-1}) \det(-Q_h)$$

where

$$Q_h = D_h A_h^{-1} G_h.$$

Hence (1.2) is solvable if A_h and Q_h are invertible.

The methods we discuss here are based on the operator Q_h. We note that

$$u_h = A_h^{-1}(f_h - G_h p_h) \tag{1.3}$$

from the first row in (1.2). Using the second row, we have

$$D_h A_h^{-1}(f_h - G_h p_h) = g_h.$$

Thus (1.2) can be solved by first solving

$$Q_h p_h = h_h \tag{1.4}$$

for p_h where

$$h_h = D_h A_h^{-1} f_h - g_h.$$

After p_h is obtained, u_h can be recovered from (1.3). The operator Q_h is the Schur complement of the system (1.2).

The operator Q_h often has several rather desirable properties. As we show in the next section, Q_h is close to being a symmetric, positive definite operator. Moreover, in many cases the eigenvalues of Q_h can be bounded independently of the mesh spacing. In these cases, one can use the conjugate gradient method to solve (1.4), and the number of conjugate gradient iterations required to solve (1.4) should be relatively independent of the grid parameters. We call the iterative method based on solving (1.4) by the conjugate gradient method the pressure equation method, and refer to it as the PE method.

The PE method requires that A_h needs to be inverted in each iteration of the conjugate gradient method. This must be done efficiently in order for the overall method to be efficient. Multigrid methods or preconditioned conjugate gradient methods are two possible methods. The price for inversion of A_h would be essentially independent of the grid size when the multigrid method is used, and would grow slowly if a preconditioned conjugate gradient method were used.

The Uzawa iterative method, see [1], can be viewed as solving equation (1.4) by a fixed point iteration. This method can be written as,

$$\begin{aligned} A_h u_h^{\nu+1} &= -G_h p_h^{\nu} + f_h \\ p_h^{\nu+1} &= p_h^{\nu} - \gamma(D_h u_h^{\nu+1} - g_h). \end{aligned} \tag{1.5}$$

The method converges for γ in some interval $(0, \bar{\gamma})$ depending on the scaling of the operators.

A potential disadvantage of these methods is the necessity of inverting A_h at each iteration. There have been a number of iterative methods that avoid the inversion of the operator A_h as required by the Uzawa method. We describe only a few here. For other related methods see [8], and [1].

Bramble and Pasciak [3] proposed an iterative method using a preconditioned conjugate gradient method to solve finite element approximations to the Stokes equations. To avoid the inversion of A_h, Bramble and Pasciak used a preconditioner A_{h0}^{-1}. With the preconditioner, (1.2) is transformed to

$$M_h \begin{pmatrix} u_h \\ p_h \end{pmatrix} = \hat{f}_h \tag{1.6}$$

where

$$M_h = \begin{pmatrix} A_{h0}^{-1}A_h & A_{h0}^{-1}G_h \\ D_h A_{h0}^{-1}(A_h - A_{h0}) & D_h A_{h0}^{-1} G_h \end{pmatrix} \quad \text{and} \quad \hat{f}_h = \begin{pmatrix} A_{h0}^{-1} f_h \\ D_h A_{h0}^{-1} f_h - g_h \end{pmatrix}.$$

They assumed that $G_h^* = D_h$ and

$$0 < ((A_h - A_{h0})u_h, u_h) \leq \alpha (A_h u_h, u_h) \tag{1.7}$$

for all $u_h \neq 0$ and for some α with $0 < \alpha < 1$. If (1.7) is satisfied, then M_h is symmetric and positive definite with the inner product

$$\left[\begin{pmatrix} u_h \\ p_h \end{pmatrix}, \begin{pmatrix} v_h \\ q_h \end{pmatrix} \right] = (A_h u_h, v_h) - (A_{h0} u_h, v_h) + (p_h, q_h)$$

where $(\cdot,\cdot)$ is the usual inner product in the discrete space.

Under an assumption equivalent to the inf-sup condition (see [2]) which implies that the condition number $\kappa(M_h)$ of Q_h is bounded by a constant independent of h, they showed that

$$0 < C_1 \left\| \begin{pmatrix} u_h \\ p_h \end{pmatrix} \right\|^2 \leq \left[M_h \begin{pmatrix} u_h \\ p_h \end{pmatrix}, \begin{pmatrix} u_h \\ p_h \end{pmatrix} \right] \leq C_2 \left\| \begin{pmatrix} u_h \\ p_h \end{pmatrix} \right\|^2 \tag{1.8}$$

for some positive constants C_1 and C_2 and for all $(u_h, p_h)^t$. This implies that $\kappa(M_h)$ is bounded by a constant independent of h and the conjugate gradient is a good method to solve (1.6). We refer to the above iterative method suggested by Bramble and Pasciak as the BP method in more detail.

Strikwerda [14] avoided the inversion of A_h by using one step of successive-over-relaxation. If A_h is written as

$$A_h = \Delta_h - L_h - U_h$$

where Δ_h is the diagonal of A_h and L_h and U_h are strictly lower and upper triangular matrices respectively, then the method introduced in [14] is

$$\begin{aligned} u_h^{\nu+1} &= u_h^\nu - \omega \Delta_h^{-1}(\Delta_h u_h^\nu - L_h u_h^{\nu+1} - U_h u_h^\nu + G_h p_h^\nu - f_h) \\ p_h^{\nu+1} &= p_h^\nu - \gamma(D_h u_h^{\nu+1} - g_h). \end{aligned} \tag{1.9}$$

We refer to this method as the SOR method.

The number of iterations required by the SOR method is, at best, proportional to h^{-1} where h denotes mesh size [14], and this requires a great amount of time to get a solution for small mesh sizes. For example, Strikwerda and Scarbnick [12] pointed out that the SOR method was quite slow when they used domain decomposition methods. An advantage of the SOR method is the relative simplicity of coding the algorithm. We include this method in our study as a representative of iterative methods that use either time-marching or SOR-like methods to solve the Stokes equations, see Roach [11]. Although there is a great variety of such methods, they all take a number of time steps or iterations that is proportional to h^{-1} at best.

The PE method is the fastest of the methods we compare here. Both the PE and the BP method have work that is proportional to the number of grid points, but the PE method is faster. In part this is because the PE method needs to invert A_h just once in each conjugate gradient step, while the BP method needs to operate with A_{h0}^{-1} twice. The other reason is that the inner product used in the BP method requires considerable work to compute. This extra work cancels out the advantage of using the preconditioner. The exact comparison of efficiency is done in the section 4.

The PE method doesn't require parameters. This is a significant advantage over the SOR method for which good values of the parameters ω and γ in the SOR method can be hard to find. The BP method also needs a scaling parameter in the preconditioning and, in our experiments, the method was very sensitive to the scaling parameter. In the two subsequent sections, we discuss the PE method and the BP method.

2. Analysis of the Pressure Equation Method.

To analyze the PE method, we first examine the analogous problem for partial differential equations. Define the operator Q for p in $L^2(\Omega)/R$ as

$$Qp := \vec{\nabla} \cdot \vec{\psi}$$

where

$$\nabla^2 \vec{\psi} = \vec{\nabla} p \qquad \text{with} \qquad \vec{\psi}|_{\partial\Omega} = 0.$$

Q can be expressed symbolically as $(-\vec{\nabla}\cdot)(-\nabla^2)^{-1}(\vec{\nabla})$. Crozier [7] has proved the following theorem, see also [9].

Theorem 2.1. *If Ω is a connected, bounded domain in R^2 with smooth boundary, then the operator Q is a bounded, positive definite operator, with bounded inverse, on $L^2(\Omega)/R$.*

The norm of Q is actually bounded by 1. So the above theorem can be expressed mathematically as

$$0 < C||p||^2 \leq (Qp, p) \leq ||p||^2 \tag{2.1}$$

for some positive constant C and for all p. Moreover the operator Q is self-adjoint.

Even more can be said about the eigenvalues of Q. The eigenvalue 1 occurs with infinite multiplicity. This is on the orthogonal compliment of the harmonic functions in $L^2(\Omega)$. We

conjecture, based on some evidence, that the rest of the eigenvalues are clustered around one-half.

Conjecture. *The operator Q has the eigenvalue 1 with an infinite multiplicity, and the remaining eigenvalues have a cluster point at 1/2 with no other cluster point.*

If Q_h is a consistent and regular finite difference approximation to Q, then one can expect that Q_h is positive definite and has its condition number bounded by a constant independent of h.

If one uses the usual central difference scheme for D_h and G_h, then Q_h is symmetric. However, if central difference formulas are used for D_h and G_h then the scheme is not regular, see [13], and Q_h will either be singular or be nearly singular.

If the regularized central difference scheme (see [13]) is used for D_h and G_h, then the symmetry of Q_h is lost. However Q_h is close to being symmetric. As our numerical solutions show, the ordinary conjugate gradient method works very well.

The following is the conjugate gradient method we used to find the pressure p_h, see [15]. Let (u_h^0, p_h^0) be an initial solution with u_h^0 having the true boundary values. Let $s_h^0 = r_h^0 = h_h - Q_h p_h^0$ where s_h^ν and r_h^ν denote the search vectors and residual vectors, respectively. Define $q_h^0 = Q_h r_h^0$. The conjugate gradient method for the PE method is

$$
\begin{aligned}
p_h^{\nu+1} &= p_h^\nu + \alpha_\nu s_h^\nu \\
r_h^{\nu+1} &= r_h^\nu - \alpha_\nu q_h^\nu \\
s_h^{\nu+1} &= r_h^{\nu+1} + \beta_\nu s_h^\nu \\
q_h^{\nu+1} &= Q_h r_h^{\nu+1} + \beta_\nu q_h^\nu \\
\alpha_\nu &= \frac{(r_h^\nu, r_h^\nu)}{(s_h^\nu, q_h^\nu)} \\
\beta_\nu &= \frac{(r_h^{\nu+1}, r_h^{\nu+1})}{(r_h^\nu, r_h^\nu)}
\end{aligned}
$$

When A_h is inverted, the boundary values must be assigned to obtain a unique solution. The residual vector r_h in the conjugate gradient method is defined to be $h_h - Q_h p_h$ and initially $r_h^0 = D_h A_h^{-1}(f_h - G_h p_h^0) - g_h$. The first row in the equation (1.2) implies that the boundary values of $A_h^{-1}(f_h - G_h p_h^0)$ have to be the boundary values of u_h, the velocity field of the solution. But, in later steps, when one needs to evaluate $Q_h r_h$, the zero boundary values should be used for A_h^{-1}.

The multigrid process using V-cycles was used to invert A_h. The ordinary Gauss-Seidel iteration was used as the smoother. The number of relaxations in each node of the multigrid was 2. Injection was used to go to a coarser level and interpolation was used to go to a finer level. The residual was computed just before the injection process and at the end of the V-cycles. For the multigrid terminology, refer to [6].

3. The Bramble-Pasciak Method.

The conjugate gradient method applied to (1.6) is defined as the following, refer to [3] for details. Let z_h^0 be an initial approximation to the solution pair $(u_h^0, p_h^0)^t$ with the true boundary values assigned for u_h^0. With $s_h^0 = r_h^0 = \hat{f}_h - M_h z_h^0$, define for $\nu \geq 0$

$$\begin{aligned} \alpha_\nu &= \frac{[r_h^\nu, s_h^\nu]}{[M_h s_h^\nu, s_h^\nu]}, \\ z_h^{\nu+1} &= z_h^\nu + \alpha_\nu s_h^\nu, \\ r_h^{\nu+1} &= \hat{f}_h - M_h z_h^{\nu+1}, \\ \beta_\nu &= -\frac{[M_h s_h^\nu, r_h^{\nu+1}]}{[M_h s_h^\nu, s_h^\nu]}, \\ s_h^{\nu+1} &= r_h^{\nu+1} + \beta_\nu s_h^\nu. \end{aligned} \tag{3.1}$$

Note that, from [15],

$$\alpha_\nu = \frac{[r_h^\nu, r_h^\nu]}{[M_h s_h^\nu, s_h^\nu]} \quad \text{and} \quad \beta_\nu = \frac{[r_h^{\nu+1}, r_h^{\nu+1}]}{[r_h^\nu, r_h^\nu]}. \tag{3.2}$$

Since M_h is positive-definite, (3.2) shows that α_ν and β_ν are nonnegative. This fact can be used to test a good candidates for A_{h0}^{-1}. One possible choice for A_{h0}^{-1} is to let it be one V-cycle for solving

$$A_h u_h = f_h \tag{3.3}$$

when the boundary values of u_h are specified. However this choice of A_{h0}^{-1} may not satisfy (1.7). A better choice is to take

$$A_{h0}^{-1} = \sigma A_{h1}^{-1}$$

where A_{h1}^{-1} is one V-cycle for solving (3.3) and σ is a scaling factor. If σ is chosen improperly, then there is a chance for M_h to be indefinite. This is detected in computation by checking on the positivity of α_ν and β_ν. By changing the value σ, one is able to find a A_{h0}^{-1} satisfying (1.7).

The parameter σ is not hard to find since it is larger than and close to 1 by the following argument. Since $A_{h1}^{-1} A_h \approx I_h$, $\sigma A_{h1}^{-1} A_h \approx I_h$ also for σ near 1. Note $A_h - A_{h1} \approx 0$. To get $A_h - \sigma^{-1} A_{h1} > 0$, σ needs to be larger than and close to 1.

The following comments explain how we implemented the BP method. Some care must be taken to insure good efficiency. From (1.6) and the definition of M_h, the residual vector is

$$r_h = \begin{pmatrix} A_{h0}^{-1}(f_h - A_h u_h - G_h p_h) \\ D_h A_{h0}^{-1}(f_h - A_h u_h - G_h p_h) + D_h u_h - g_h \end{pmatrix}.$$

To compute r_h, first set and save the vector

$$w_h := f_h - A_h u_h - G_h p_h \tag{3.4}$$

for later use. Next the system

$$A_{h0} \hat{w}_h = w_h \tag{3.5}$$

is solved for $\hat{w}_h$ with zero boundary condition, we then have

$$r_h = \begin{pmatrix} r_I \\ r_{II} \end{pmatrix} := \begin{pmatrix} \hat{w}_h \\ D_h(\hat{w}_h + u_h) - g_h \end{pmatrix}. \tag{3.6}$$

In this way the initial residual r_h^0 is computed. Also set $s_h^0 = r_h^0$.

In subsequent iterations, the inner product $[r_h, s_h]$ is computed as

$$\begin{aligned} [r_h, s_h] &= ((A_h - A_{h0})r_I, s_I) + (r_{II}, s_{II}) \\ &= (A_h r_I - w_h, s_I) + (r_{II}, s_{II}). \end{aligned} \tag{3.7}$$

where $s_h = (s_I, s_{II})^t$. The last expression is used to compute $[r_h, s_h]$. Note that A_{h0} is not used explicitly.

$$\begin{aligned} \left[M_h \begin{pmatrix} s_I \\ s_{II} \end{pmatrix}, \begin{pmatrix} s_I \\ s_{II} \end{pmatrix}\right] &= (A_h A_{h0}^{-1}(A_h s_I + G_h s_{II}) - (A_h s_I + G_h s_{II}), s_I) \\ &\quad + (D_h A_{h0}^{-1}(A_h - A_{h0})s_I + D_h A_{h0}^{-1} G_h s_{II}, s_{II}). \end{aligned}$$

To simplify this expression, we set

$$t_h := A_h s_I + G_h s_{II} \tag{3.8}$$

and solve

$$A_{h0}\hat{t}_h = t_h \tag{3.9}$$

for $\hat{t}_h$ with zero boundary condition. If m_I and m_{II} are defined to be $A_h\hat{t}_h - t_h$ and $D_h(\hat{t}_h - s_I)$ respectively, then

$$\left[M_h \begin{pmatrix} s_I \\ s_{II} \end{pmatrix}, \begin{pmatrix} s_I \\ s_{II} \end{pmatrix}\right] = (m_I, s_I) + (m_{II}, s_{II}). \tag{3.10}$$

If the vector $(m_I, m_{II})^t$ is saved, then $[M_h s_h, r_h]$ is computed as

$$(m_I, r_I) + (m_{II}, r_{II}). \tag{3.11}$$

In this whole process, we need to evaluate A_{h0}^{-1}, and never need to evaluate A_{h0} itself. The special forms of α_ν and β_ν in (3.1) were chosen to be easily computable.

4. Analysis of Efficiency.

In this section we estimate the total number of significant operations, which we designate as TSO, for each iterative method. We use these estimates to compare the efficiency of each of these methods. We take as a representative case the Stokes equations on a square in R^2 or cube in R^3. If $N+1$ is the number of grid points in a coordinate direction in R^d, then $(N-1)^d$ is the number of interior grid points. TSO_S, TSO_P and TSO_B are the TSO for the SOR method, the PE method, and the BP Method, respectively. $Iter_S, Iter_P$, and $Iter_B$ are defined similarly.

Let N_A, N_G, and N_D be the number of multiplications per grid point to apply A_h, G_h and D_h, respectively. If $u_h = (u_1, \ldots, u_d)^t$, then

$$(A_h u_h)_{l,m} = ((\nabla_h^2 u_1)_{l,m}, \ldots, (\nabla_h^2 u_d)_{l,m})^t. \tag{4.1}$$

We used the usual second-order accurate discrete Laplacian for ∇_h^2. Since A_h involves d scalar Laplacians, $N_A \approx (2d+1)d$. The regularized central differencing was used to find any first derivative with respect to any direction, and this needs 4 points to evaluate. Each of $(G_h p_h)_{l,m}$ and $(D_h u_h)_{l,m}$ needs d first derivatives to be evaluated, so $N_G \approx 4d$ and $N_D \approx 4d$. We consider our "cost" to be the number of multiplications required.

Lemma 4.1. *$TSO_S \approx Iter_S \cdot d(2d+9) \cdot (N-1)^d$.*

Proof. From (1.9),

$$\begin{aligned} TSO_S &\approx Iter_S \cdot (N_A + N_G + N_D) \cdot (N-1)^d \\ &\approx Iter_S \cdot (2d^2 + d + 8d) \cdot (N-1)^d \\ &\approx Iter_S \cdot (2d^2 + 9d) \cdot (N-1)^d. \spadesuit \end{aligned}$$

Lemma 4.2. *One V-cycle for the scalar second-order Laplacian costs approximately $N_V(N-1)^d$ where $N_V = 2^d(2^d-1)^{-1}(10d+6)$.*

Proof. Going down along a V-cycle, we do 2 smoothing processes, 1 residual finding, and 1 injection at each level. On the way up, we do 2 smoothing processes and 1 interpolation at each level. So, in a V-cycle, altogether 4 smoothing processes, 1 residual finding, 1 injection and 1 interpolation at each level are needed. On the finest level, smoothing costs $(2d+1)(N-1)^d$, computing the residual is about the same, injection and interpolation together cost at most $(N-1)^d$ operations.

Thus one V-cycle costs

$$(5(2d+1)+1) \cdot (N-1)^d \cdot \left(1 + \frac{1}{2^d} + (\frac{1}{2^d})^d + \ldots + (\frac{1}{2^d})^{\#\text{ of levels}}\right)$$

where d is the dimension of our domain. The above number is approximately

$$\begin{aligned} &\frac{1}{1-2^{-d}} \cdot (10d+6) \cdot (N-1)^d \\ =&\frac{2^d}{2^d-1}(10d+6)(N-1)^d. \spadesuit \end{aligned}$$

Lemma 4.3. *$TSO_P \approx Iter_P \cdot d(8 + \bar{v}N_V) \cdot (N-1)^d$, where $\bar{v}$ is the average number of V-cycles required per iteration.*

Proof. One needs to apply the matrix Q_h in each conjugate gradient iteration. From (4.1), we see that A_h^{-1} consists of d multigrid operations. So, we have by Lemma 4.2,

$$\begin{aligned} TSO_P &\approx Iter_P \cdot (N_G + N_D + d\bar{v} \cdot N_V) \cdot (N-1)^d \\ &\approx Iter_P \cdot (8d + d\bar{v}N_V) \cdot (N-1)^d \spadesuit \end{aligned}$$

Lemma 4.4. $TSO_B \approx Iter_B \cdot 2d(4d + 10 + N_V) \cdot (N-1)^d$.

Proof. In each iteration, the main effort is in finding $r_h, [r_h, s_h]$ and $[M_h s_h, s_h]$ from (3.1). By Lemma 4.2 and the equations from (3.4) to (3.6), the cost to get r_h is

$$(N_A + N_G + d \cdot N_V + N_D)(N-1)^d.$$

Evaluating $[r_h, s_h]$ costs

$$N_A \cdot (N-1)^d$$

by (3.7). The cost of evaluating $[M_h s_h, s_h]$ is

$$(2N_A + N_G + dN_V + N_D)(N-1)^d$$

by the equations from (3.8) to (1.6).

Adding these costs, we obtain

$$\begin{aligned} TSO_B &\approx Iter_B \cdot (4N_A + 2N_G + 2N_D + 2dN_V) \cdot (N-1)^d \\ &\approx Iter_B \cdot (8d^2 + 4d + 16d + 2dN_V) \cdot (N-1)^d \\ &\approx Iter_B \cdot (8d^2 + 20d + 2dN_V) \cdot (N-1)^d . \spadesuit \end{aligned}$$

By (1.8) and (2.1), $Iter_P$ and $Iter_B$ are bounded by some constants not depending on mesh size. Moreover, $Iter_S$ is proportional to N at best. For the test case considered in section 6 we find, for $N = 64$ and $d = 2$, $Iter_S \approx 8(N-1)$, $Iter_P = 12$, and $Iter_B = 17$. Also, $\bar{v}$ was about 2 for the PE method. So, $TSO_S \approx 208(N-1)^3$, $TSO_P \approx 1856(N-1)^2$, and $TSO_B \approx 3581(N-1)^2$.

We see that the PE is the fastest method, with the BP method being about twice as much work. The SOR method is 7 times as much work as the PE method for the one case considered here and is even less efficient as N increases. The numerical results in section 6 also show that based on CPU time, for this test case, the PE method is more than 7 times faster and the BP method is about 4 times faster than the SOR method, agreeing with our analysis.

5. The Numerical Experiments.

For the numerical experiment, we used the Stokes equations of the form

$$\nabla^2 u - \frac{\partial p}{\partial x} = -2\pi^2 \sin \pi x \sin \pi y + \pi \sin \pi x \sin \pi y,$$

$$\nabla^2 v - \frac{\partial p}{\partial y} = -2\pi^2 \cos \pi x \cos \pi y - \pi \cos \pi x \cos \pi y,$$

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

Table 3
Errors and CPU time for the SOR method.

N	*iter*	u	p	*time*
32	275	6.33(−5)	2.74(−3)	8.546
48	399	2.48(−5)	1.26(−3)	28.150
64	511	1.39(−5)	6.98(−4)	65.214

by the PE method and the BP method are essentially independent of mesh size, which supports (1.7) and (2.1).

The next table, Table 4 shows the accuracy of the PE method, the BP method, and the SOR method. The order of accuracy was obtained from the formula $\log(\text{error}(h_1)/\text{error}(h_2))/\log(h_2/h_1)$ where h_1, h_2 are mesh sizes with $h_1 < h_2$. All numerical solutions show that they are second-order accurate.

Table 4
Order of accuracy for the computed solutions.

	PE		BP		SOR	
N_1, N_2	u	p	u	p	u	p
64, 48	2.1	2.0	1.9	2.1	2.1	1.9
64, 32	2.2	2.0	2.2	2.0	2.4	2.0
48, 32	2.3	1.9	2.5	1.9	2.6	2.0

7. Navier-Stokes Equations.

The steady-state Navier-Stokes equations in R^d are of the form

$$\begin{aligned} -R^{-1}\nabla^2\vec{u} + (\vec{u}\cdot\vec{\nabla})\vec{u} + \vec{\nabla}p &= \vec{f}, \\ \vec{\nabla}\cdot\vec{u} &= g \qquad \text{in} \qquad \Omega \subset R^d \end{aligned} \tag{7.1}$$

where R is the Reynolds number. We consider the Dirichlet boundary condition

$$\vec{u} = \vec{b} \qquad \text{on} \qquad \partial\Omega.$$

There are several possible extensions of the PE method from the Stokes equations to the Navier-Stokes equations, depending on how one linearizes the first equation in (7.1). To apply the PE method efficiently to (7.1), we used the following algorithm which worked for R up to about 100.

(1) Start with an initial solution $\vec{u}^0, p^0$.

(2) Given the solution $\vec{u}^\nu$, let

$$\begin{aligned} \vec{d}^\nu &:= (\vec{u}^\nu\cdot\vec{\nabla}_h)\vec{u}^\nu, \\ \vec{f}_1^\nu &:= \vec{f} - \vec{d}^\nu. \end{aligned}$$

where $\vec{\nabla}_h$ is a finite discretization of $\vec{\nabla}$, then (7.1) can be expressed as

$$\begin{aligned} -R^{-1}\nabla^2\vec{u} + \vec{\nabla}p &= \vec{f}_1^{\nu}, \\ \vec{\nabla}\cdot\vec{u} &= g. \end{aligned} \tag{7.2}$$

(3) The system (7.2) gives an equation for pressure p which is

$$Q_h p_h = h_h^{\nu} \tag{7.3}$$

where the function h_h^{ν} is generated by $\vec{f}_1^{\nu}$ and g_h. Apply the PE method to (7.3), i.e., do several conjugate gradient iterations to update $p^{\nu+1}$ from p^{ν}.

(4) Let

$$\vec{f}_2^{\nu} = \vec{f} - \vec{\nabla}p^{\nu+1},$$

then the first equation in (7.1) is the so-called convection diffusion equation

$$-R^{-1}\nabla^2\vec{u} + (\vec{u}\cdot\vec{\nabla})\vec{u} = \vec{f}_2^{\nu}. \tag{7.4}$$

To update $\vec{u}^{\nu+1}$, solve (7.4) for $\vec{u}$. We discuss the solution procedure later. Go to step (2).

For our numerical experiment, we used the Navier-Stokes equations of the form

$$\begin{aligned} -R^{-1}\nabla^2 u + uu_x + vu_y + p_x &= f_1 \\ -R^{-1}\nabla^2 v + uv_x + vv_y + p_y &= f_2 \\ u_x + v_y &= 0 \end{aligned}$$

on $0 < x, y < 1$ where

$$\begin{aligned} f_1 &= 2R^{-1}\pi^2 \sin\pi x \sin\pi y + 0.5\pi\sin(2\pi x) - \pi \sin\pi x \sin\pi y \\ f_2 &= 2R^{-1}\pi^2 \cos\pi x \cos\pi y - 0.5\pi\sin(2\pi y) + \pi \cos\pi x \cos\pi y. \end{aligned}$$

The values of u and v are specified on the boundary.

The exact solution is given by

$$\begin{aligned} u &= \sin\pi x \sin\pi y, \\ v &= \cos\pi x \cos\pi y, \\ p &= \cos\pi x \sin\pi y. \end{aligned}$$

Because of the nonlinearity of (7.4), the Full Approximation Scheme (FAS) was used for multigrid solver. See [5] for a description of FAS. Moreover the full weighting was used in the fine-to-coarse transfers of both the solution and the residual functions. To employ a stable discretization, upwind differencing was used for the first derivatives in (7.4) when the mesh size h was larger than $2/RU$ where U is the maximum value of $\vec{u}$ on the domain, see [10]. Otherwise, the central differencing was used to get the overall second-order accuracy. In [4], the authors mentioned that it is better to employ upwind differencing only in the

on $0 < x, y < 1$ with u and v specified on the boundary.

The exact solution is given by

$$\begin{aligned} u &= \sin \pi x \sin \pi y, \\ v &= \cos \pi x \cos \pi y, \\ p &= \cos \pi x \sin \pi y. \end{aligned}$$

The discretization used a uniform grid with the same number of grid points in each direction. The second order accurate five-point Laplacian was used to approximate ∇^2 for all the iterative methods.

We employed, for all the iterative methods, the regularized central difference (see [13]) given by

$$\begin{aligned} \frac{\partial p}{\partial x} &\approx \delta_{x0} p_h - \frac{h^2}{6} \delta_{x-} \delta_{x+}^2 p_h, \\ \frac{\partial p}{\partial y} &\approx \delta_{y0} p_h - \frac{h^2}{6} \delta_{y-} \delta_{y+}^2 p_h, \\ \frac{\partial u}{\partial x} &\approx \delta_{x0} u_h - \frac{h^2}{6} \delta_{x+} \delta_{x-}^2 u_h, \\ \frac{\partial v}{\partial y} &\approx \delta_{y0} v_h - \frac{h^2}{6} \delta_{y+} \delta_{y-}^2 v_h, \end{aligned}$$

where h is the grid spacing and δ_{x0}, δ_{x+}, and δ_{x-} are the centered, forward, and backward difference operators in the x-direction. The operators δ_{y0}, δ_{y+} and δ_{y-} are defined similarly for the y-direction.

To obtain the pressure on the boundary, we used the quadratic interpolation, e.g.,

$$p_{0m} = 2p_{1m} - p_{2m},$$

for all the iterative methods.

The SOR method was stopped when the quantities

$$||u_h^{n+1} - u_h^n||, \quad ||v_h^{n+1} - v_h^n||, \quad ||p_h^{n+1} - p_h^n|| \tag{5.1}$$

were all less than $5 \cdot 10^{-5}, 10^{-4}$ and $2 \cdot 10^{-4}$ for mesh sizes 1/32, 1/48 and 1/64 respectively. These values were chosen because the quantities in (5.1) could not be made much smaller than these values. We did not investigate why these quantities could not be made smaller, but presume that it is due to the use of single precision arithemetic. As will be seen, the use of higher precision would not alter our conclusions. The norms of u_h and v_h in (5.1) were the discrete L^2 norms, and the norm for p_h was the L^2 norm in its quotient space (see [14]). The relaxation parameters ω and γ were given by

$$\omega = 2/(1 + c_0 h), \qquad \gamma = c_1 h$$

where c_0=3.14 and c_1=4.5. See [13], [14] for more details.

The PE method was stopped when the residual was less than 10^{-6}. In each conjugate gradient iteration of the PE method, the multigrid process using V-cycles was used to invert A_h. We found that to achieve good overall accuracy it was only necessary to do enough V-cycles to reduce the residual in the L^2 norm to less than 10^{-4}. Each multigrid process to solve $A_h u_h = f_h$ for u_h was stopped when either the number of V-cycles was 4 or the residual error was less than 10^{-4}. The maximum number of V-cycles was chosen to be 4 since the residual error didn't change significantly after 4 V-cycles. Because the reduction factor of the error is small in the multigrid process, more than 4 V-cycles would rarely be needed. With these stopping criteria, the average number of V-cycles needed in each conjugate gradient iteration was 2.

The BP method was stopped when the residuals were less than $3 \cdot 10^{-4}, 10^{-4}$ and $3 \cdot 10^{-5}$ for mesh sizes 1/32, 1/48 and 1/64 respectively. These values were chosen since, similar to the SOR method, the residuals decreased to values a little bit smaller than these values, but could not be made much smaller. Again, this is probably due to the precision of the computer arithmetic. In the BP method, several values were run for σ, the value of 1.2 worked well.

6. Test Results.

Tables 1, 2, and 3 show the errors for the PE method, the BP method and the SOR method. The column labeled "*time*" shows the CPU time required for the total computation.

Table 1
Errors and CPU time for the PE method.

N	*iter*	u	p	*time*
32	12	6.46(−5)	2.71(−3)	1.617
48	12	2.35(−5)	1.25(−3)	4.347
64	12	1.38(−5)	6.91(−4)	8.362

Table 2
Errors and CPU time for the BP method.

N	*iter*	u	p	*time*
32	14	6.34(−5)	3.04(−3)	2.843
48	16	2.19(−5)	1.36(−3)	8.558
64	17	1.19(−5)	7.83(−4)	17.162

By comparing CPU times, one can see that the PE method is most efficient, and the BP method takes about twice as much effort, and the SOR method is least efficient, taking about 7 times as much time as the PE method. Note that the number of iterations taken

relaxation sweeps, central differencing in the residual transfers, but we obtained the best numerical solution when the same differencing was used in both relaxation sweeps and residual transfers. Also, the computation of $\vec{f}_2^{\nu}$ at coarser levels used upwind differencing.

Table 5 and Table 6 show the error and accuracy of the solution when R is 30. Notice that the method is second-order accurate.

Table 5
Errors for $R = 30$.

N	u	p
32	7.34(−4)	3.65(−4)
48	2.01(−4)	1.55(−4)
64	9.35(−5)	8.94(−5)

Table 6
Accuracy of the solution for $R = 30$.

N_1, N_2	u	p
64, 48	2.7	1.9
64, 32	3.0	2.0
48, 32	3.2	2.1

8. Conclusions.

The pressure equation method has been shown to be an efficient numerical method for solving the steady Stokes equations. Since the work is essentially proportional to the number of grid points, the efficiency of this method is exceptional. We have also shown that the method advocated by Bramble and Pasciak is not as efficient for the finite difference schemes used here.

The pressure equation method has been extended to the Navier-Stokes equations for low Reynolds numbers. Research is continuing on improving this method. Work is also being done on applying the method to time-dependent problems and using the method with domain decomposition.

REFERENCES

[1] K. Arrow, L. Hurwitz and H. Uzawa, *Studies in Nonlinear Programming*, Stanford University Press, Stanford, 1958.

[2] A. K. Aziz & I. Babuška, *"Survey lectures on the mathematical foundations of the finite element method, Part I"*, in *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, A. K. Aziz, ed., Academic Press, New York, 1972, pp. 1-362.

[3] J. H. Bramble and J. E. Pasciak, *"A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems"*, Math. Comp., 50 (1988), pp. 1-18.

[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, Inc., New York, 1979, pp. 53-148.

[5] A. Brandt, *"Guide to multigrid development"*, in *Multigrid Methods*, W. Hackbusch and U. Trottenberg, ed., Springer-Verlag, New York, NY, 1981, pp. 220-312.

[6] W. L. Briggs, *Multigrid Tutorial*, Lancaster Press, Lancaster, Pennsylvania, 1987.

[7] M. Crozier, *Approximation et methodes iteratives de resolution d'inequations variationnelles et de problemes non lineares*, IRIA cahier no 12., 1974.

[8] M. Fortin and R. Glowinski, *Resolution Numerique de Problèmes aux Limites par des Methodes de Langrangien Augment*, 1981.

[9] V. Girault and P. A. Raviart, *Finite Element Approximation of the Navier-Stokes Equations*, Lecture Notes in Mathematics, 749, Springer-Verlag, New York, NY, 1979.

[10] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, New York, NY, 1980.

[11] P. Roach, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1972.

[12] J. C. Strikwerda and C. D. Scarbnick, *"A domain decomposition method for incompressible viscous flow"*, SIAM J. Sci. Stat. Comput., to appear (1991).

[13] J. C. Strikwerda, *"Finite difference methods for the Stokes and Navier-Stokes equations"*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 56-68.

[14] J. C. Strikwerda, *"An iterative method for solving finite difference approximations to the Stokes equations"*, SIAM J. Numer. Anal., 21 (1984), pp. 447-458.

[15] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*, Wadsworth & Brooks/Cole, Pacific Grove, CA, 1989.

[16] R. Temam, *Navier-Stokes Equations*, Elsevier Science Publishing Company, Inc., New York, NY, 1984.