INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 22, 755–770 (1996)

MULTIGRID AND KRYLOV SUBSPACE METHODS FOR THE DISCRETE STOKES EQUATIONS

HOWARD C. ELMAN

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, U.S.A.

SUMMARY

Discretization of the Stokes equations produces a symmetric indefinite system of linear equations. For stable discretizations a variety of numerical methods have been proposed that have rates of convergence independent of the mesh size used in the discretization. In this paper we compare the performance of four such methods, namely variants of the Uzawa, preconditioned conjugate gradient, preconditioned conjugate residual and multigrid methods, for solving several two-dimensional model problems. The results indicate that multigrid with smoothing based on incomplete factorization is more efficient than the other methods, but typically by no more than a factor of two. The conjugate residual method has the advantage of being independent of iteration parameters.

KEY WORDS Stokes; multigrid; Krylov subspace; conjugate gradient; conjugate residual; Uzawa

1. INTRODUCTION

Consider the system of partial differential equations

$$\begin{aligned} -\Delta u + \nabla p = f \quad &\text{and} \quad -\operatorname{div} u = 0 \quad \text{on } \Omega, \\ u = 0 \quad &\text{on } \partial\Omega, \\ \int_\Omega p = 0, \end{aligned} \tag{1}$$

where Ω is a simply connected bounded domain in $\mathbb{R}^d$, $d = 2$ or 3. This system, the *Stokes equations*, is a fundamental problem arising in computational fluid dynamics (see e.g. References 1–4); u is the d-dimensional velocity vector defined in Ω, and p represents pressure.

Discretization of (1) by finite difference or finite element techniques leads to a linear system of equations of the form

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{2}$$

where A is a set of uncoupled discrete Laplacian operators and C is a positive semidefinite matrix. We consider here only *stable* discretizations, i.e. those for which the condition number of the Schur complement matrix $BA^{-1}B^T + C$ is bounded independently of the mesh size used in the discretization. For finite element discretizations with $C = 0$ this is a consequence of the inf-sup condition and upper bound

$$\gamma \leqslant \inf_q \sup_v \frac{(q, \operatorname{div} v)}{|v|_1 \|q\|_0}, \qquad \frac{|(q, \operatorname{div} v)|}{|v|_1 \|q\|_0} \leqslant \Gamma,$$

CCC 0271–2091/96/080755–16

Received 28 June 1994
Revised 24 June 1995

where γ and Γ are independent of the mesh size. Here $|\cdot|_1$ and $\|\cdot\|_0$ denote the H^1-seminorm and Euclidean norm respectively on the discrete velocity and pressure spaces and the bounds are taken over all v and q in the appropriate discrete spaces.[1–4]

In recent years a variety of iterative algorithms have been devised for solving the discrete Stokes equations. In this paper we compare the performance of four such methods:

(i) a variant of the Uzawa method
(ii) a preconditioned conjugate gradient (PCG) method applied to a transformed version of (2)
(iii) a preconditioned conjugate residual (PCR) method
(iv) multigrid (MG).

The Uzawa method is the first among these to have been devised[5] and is often advocated as an efficient solution technique (see e.g. References 1–3). The convergence factor associated with it is proportional to $(\kappa - 1)/(\kappa + 1)$, where κ is the condition number of the Schur complement $BA^{-1}B^{\mathrm{T}} + C$ (see Section 2.5). The conjugate gradient method, developed by Bramble and Pasciak,[6] has a convergence factor proportional to $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$ but a larger cost per step than the Uzawa method. The preconditioned conjugate residual method was developed by Rusten and Winther[7] and Silvester and Wathen[8,9] and its convergence behaviour is determined by the properties of the indefinite matrix. For multigrid we consider versions derived from two smoothing strategies: a variant of the distributive Gauss–Seidel (MG/DGS) method of Brandt and Dinar[10] and a technique based on incomplete factorization (MG/ILU) studied by Wittum.[11]

These methods all have the property that for an appropriate choice of preconditioners (or, for multigrid, smoothers) their convergence rates are independent of the mesh size used in the discretization. The actual costs of using them depend on both the convergence rate and the cost per iteration. Our goal in this paper is to compare the costs, in operation counts, of using each of the methods to solve four discrete versions of (1). For convergence to be independent of mesh size, the first three methods (*Krylov subspace methods*) require a preconditioning operator spectrally equivalent to the discrete Laplacian. In an effort to unify the comparison of these ideas with multigrid, we also implement this preconditioner using a multigrid method for the associated Poisson equation. The benchmark problems are derived from the Stokes equations (1) on the two-dimensional unit square, discretized by either finite differences or one of three low-order mixed finite element schemes.

Our main observations are as follows. For problems where it is applicable, one version of multigrid, using incomplete factorization, requires the fewest iterations and operations, but it is only marginally faster, i.e. by factors of approximately 1·5–2, than the Krylov subspace methods and the distributive Gauss–Seidel method. Among the Krylov subspace methods the conjugate residual method is slightly slower than the conjugate gradient method and in some cases the Uzawa method, but it has the advantage of not requiring any parameter estimates.

An outline of the rest of the paper is as follows. In Section 2 we present the solution algorithms and give an overview of their convergence properties. In Section 3 we specify the benchmark problems and the computational costs per iteration of each of the solution methods. In Section 4 we present the numerical comparison.

2. OVERVIEW OF METHODS

In this section we present the four algorithms under consideration and outline their convergence properties. The first three methods depend on a preconditioning operator Q_A that approximates the matrix A of (2). We assume that Q_A is symmetric positive definite (SPD) and that

$$\eta_1 \leqslant \frac{(v, Av)}{(v, Q_A v)} \leqslant \eta_2, \tag{3}$$

where η_1 and η_2 are independent of the mesh size used in the discretization. In addition, finite element discretizations of (1) have a mass matrix M associated with the pressure discretization.* The preconditioner will also include an SPD approximation Q_M of M. Discussions of computational costs will be made in terms of various matrix operations together with inner products and 'AXPYs', i.e. vector operations of the form $y \leftarrow \alpha x + y$.

2.1. *The inexact Uzawa method*

We use the following 'inexact' version of the Uzawa algorithm[12] which starts with $u_0 \equiv 0$ and an arbitrary initial guess p_0:

```
for $i = 0$ until convergence, do
    $u_{i+1} = u_i + Q_A^{-1}[f - (Au_i + B^{\mathrm{T}}p_i)]$
    $p_{i+1} = p_i + \alpha Q_M^{-1}(Bu_{i+1} - Cp_i)$
enddo
```

Here α is a scalar parameter that must be determined prior to the iteration.

In the 'exact' version of this algorithm, $Q_A = A$ and the first step is equivalent to solving the linear system $Au_{i+1} = f - B^{\mathrm{T}}p_i$. When $Q_M = I$, the exact algorithm is then a fixed parameter first-order Richardson iteration applied to the Schur complement system $(BA^{-1}B^{\mathrm{T}} + C)p = BA^{-1}f$; Q_M is a preconditioner for this iteration. The inexact Uzawa algorithm (4) replaces the exact computation of $A^{-1}(f - B^{\mathrm{T}}p_i)$ with an approximation.

2.2. *A preconditioned conjugate gradient method*

Let $\mathscr{A}$ denote the coefficient matrix of (2). Premultiplication of (2) by the matrix

$$\mathscr{T} = \begin{pmatrix} Q_A^{-1} & 0 \\ BQ_A^{-1} & -I \end{pmatrix}$$

produces the equivalent system

$$\begin{pmatrix} Q_A^{-1}A & Q_A^{-1}B^{\mathrm{T}} \\ BQ_A^{-1}A - B & BQ_A^{-1}B^{\mathrm{T}} + C \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} Q_A^{-1}f \\ BQ_A^{-1}f \end{pmatrix}. \tag{5}$$

Let $\mathscr{M} = \mathscr{T}\mathscr{A}$ denote the coefficient matrix of this system. The conjugate gradient method (CG) developed in Reference 6 requires that the bilinear form

$$\left[\begin{pmatrix} v_1 \\ q_1 \end{pmatrix}, \begin{pmatrix} v_2 \\ q_2 \end{pmatrix}\right] \equiv ((A - Q_A)v_1, v_2) + (q_1, q_2) \tag{6}$$

define an inner product. Equivalently, the preconditioning operator Q_A must satisfy (3) with $\eta_1 > 1$. It is shown in Reference 6 that $\mathscr{M}$ is SPD with respect to the inner product (6), so that CG in this inner product is applicable. The matrix

$$\mathscr{G} = \begin{pmatrix} I & 0 \\ 0 & Q_M \end{pmatrix} \tag{7}$$

is also SPD with respect to (6), so that this can be used as a preconditioner.

* If the finite element solution is expressed using a given basis $\{\phi_i\}$ as $p = \sum_i \delta_i \phi_i$, then $\|p\|_{L_2} = (\delta, M\delta)^{1/2}$.

Let

$$X_0 = \begin{pmatrix} u_0 \\ p_0 \end{pmatrix}, \qquad R_0 = \begin{pmatrix} f - (Au_0 + B^T p_0) \\ -(Bu_0 - Cp_0) \end{pmatrix}$$

denote an arbitrary guess for the solution and the associated residual. An implementation of PCG is given below. Except for the non-standard inner product, it is the standard implementation as given e.g. Reference 13, in p. 529. It is more efficient then the version given in Reference 6. The preconditioner Q_A is implicitly incorporated into the inner product. The use of the preconditioner (7) is new.

```
$\hat{R}_0 = \mathcal{T}R_0, \quad \tilde{R}_0 = \mathcal{G}^{-1}\hat{R}_0$
$P_0 = \tilde{R}_0, \quad \mathcal{M}P_0 = \mathcal{T}\mathcal{A}P_0$
$\alpha_0^{(n)} = [\hat{R}_0, \tilde{R}_0], \quad \alpha_0^{(d)} = [P_0, \mathcal{M}P_0,], \quad \alpha_0 = \alpha_0^{(n)}/\alpha_0^{(d)}$
$X_1 = X_0 + \alpha_0 P_0$
$R_1 = R_0 - \alpha_0\mathcal{A}P_0, \quad \hat{R}_1 = \hat{R}_0 - \alpha_0\mathcal{M}P_0, \quad \tilde{R}_1 = \mathcal{G}^{-1}\hat{R}_1$
for $i = 1$ until convergence, do
    $\beta_{i-1}^{(n)} = [\hat{R}_i, \tilde{R}_i], \quad \beta_{i-1}^{(d)} = \alpha_{i-1}^{(n)}, \quad \beta_{i-1} = \beta_{i-1}^{(n)}/\beta_{i-1}^{(d)}$
    $P_i = \tilde{R}_i + \beta_{i-1}P_{i-1}, \quad \mathcal{M}P_i = \mathcal{T}\mathcal{A}P_i$
    $\alpha_i^{(n)} = \beta_{i-1}^{(n)}, \quad \alpha_i^{(d)} = [P_i, \mathcal{M}P_i], \quad \alpha_i = \alpha_i^{(n)}/\alpha_i^{(d)}$
    $X_{i+1} = X_i + \alpha_i P_i, \quad R_{i+1} = R_i - \alpha_i\mathcal{A}P_i$
    $\hat{R}_{i+1} = \hat{R}_i - \alpha_i\mathcal{M}P_i, \quad \tilde{R}_{i+1} = \mathcal{G}^{-1}\hat{R}_{i+1}$
enddo
```

To help identify operation counts, we describe the computation of $\{\alpha_i\}$ and $\{\beta_i\}$ in more detail. Letting

$$R_i = \begin{pmatrix} r_i \\ s_i \end{pmatrix}, \qquad \hat{R}_i = \begin{pmatrix} \hat{r}_i \\ \hat{s}_i \end{pmatrix}, \qquad \tilde{R}_i = \begin{pmatrix} \hat{r}_i \\ \tilde{s}_i \end{pmatrix},$$

we have $\beta_{i-1}^{(n)} = [\hat{R}_i, \tilde{R}_i] = (\hat{r}_i, A\hat{r}_i - r_i) + (\hat{s}_i, \tilde{s}_i)$; similarly, if

$$P_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}, \qquad \mathcal{A}P_i = \begin{pmatrix} v_i \\ w_i \end{pmatrix}, \qquad \mathcal{M}P_i = \begin{pmatrix} Q_A^{-1}v_i \\ BQ_A^{-1}v_i - w_i \end{pmatrix}, \tag{8}$$

then $\alpha_i^{(d)} = [P_i, \mathcal{M}P_i] = (c_i, AQ_A^{-1}v_i - v_i) + (d_i, BQ_A^{-1}v_i - w_i)$. Q_A is referenced only in the construction of $Q_A^{-1}v$ in (8), so that only the action of the inverse of Q_A is required. Moreover, although the vectors $A\hat{r}_i, Ac_i$ (for v_i) and $AQ_A^{-1}v_i$ are used, the first two of these can be computed using an AXPY. Consequently, only one matrix–vector product by A is needed.

2.3. *The preconditioned conjugate residual method*

Since $\mathscr{A}$ is symmetric, variants of the conjugate residual method are applicable. Let X_0 denote the initial guess and R_0 its residual. The following algorithm implements the Orthomin version of PCR with preconditioner $\mathscr{Q}$:[14,*]

```
$\tilde{R}_0 = \mathscr{Q}^{-1}R_0, \quad P_0 = \tilde{R}_0, \quad S_0 = \mathscr{Q}^{-1}\mathscr{A}P_0$
$\alpha_0^{(n)} = (\tilde{R}_0, \mathscr{A}P_0), \quad \alpha_0^{(d)} = (\mathscr{A}P_0, S_0), \quad \alpha_0 = \alpha_0^{(n)}/\alpha_0^{(d)}$
$X_1 = X_0 + \alpha_0 P_0, \quad R_1 = R_0 - \alpha_0\mathscr{A}P_0, \quad \tilde{R}_1 = \tilde{R}_0 - \alpha_0 S_0$
for $i = 1$ until convergence, do
    $\beta_{i-1}^{(n)} = -(\mathscr{A}\tilde{R}_i, S_{i-1}), \quad \beta_{i-1}^{(d)} = \alpha_{i-1}^{(d)}$
    $P_i = \tilde{R}_i + \beta_{i-1}P_{i-1}, \quad \mathscr{A}P_i = \mathscr{A}\tilde{R}_i + \beta_{i-1}\mathscr{A}P_{i-1}, \quad S_i = \mathscr{Q}^{-1}\mathscr{A}P_i$
    $\alpha_i^{(n)} = (\tilde{R}_i, \mathscr{A}P_i), \quad \alpha_i^{(d)} = (\mathscr{A}P_i, S_i), \quad \alpha_i = \alpha_i^{(n)}/\alpha_i^{(d)}$
    $X_{i+1} = X_i + \alpha_i P_i, \quad R_{i+1} = R_i - \alpha_i\mathscr{A}P_i, \quad \tilde{R}_{i+1} = \tilde{R}_i - \alpha_i S_i$
enddo
```

Any symmetric positive definite $\mathscr{Q}$ could be used as a preconditioner. As in Reference 8, we use

$$\mathscr{Q} = \begin{pmatrix} Q_A & 0 \\ 0 & Q_M \end{pmatrix}.$$

2.4. *Multigrid*

As is well known, multigrid methods combine iterative methods to smooth the error with correction derived from a coarse grid computation. We use V-cycle multigrid for 'transformed systems'. Our description follows References 11 and 16. See References 17 and 18 for other multigrid methods derived from the squared system associated with (2).

Let $-\Delta_p$ denote the Laplace operator defined on the pressure space, with Neumann boundary conditions,[19] and let A_p be a discrete approximation to $-\Delta_p$ defined on the pressure grid. Consider the following transformed version of (2):

$$\begin{pmatrix} A & B^T \\ B & -C \end{pmatrix}\begin{pmatrix} I & B^T \\ 0 & -A_p \end{pmatrix}\begin{pmatrix} \hat{u} \\ \hat{p} \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} I & B^T \\ 0 & -A_p \end{pmatrix}\begin{pmatrix} \hat{u} \\ \hat{p} \end{pmatrix}. \tag{9}$$

The coefficient matrix in (9) is

$$\tilde{\mathscr{A}} = \begin{pmatrix} A & W \\ B & G \end{pmatrix}, \tag{10}$$

where $W = AB^T - B^T A_p$ and $G = BB^T + CA_p$. For appropriate discretizations of (1) (see Section 3), W is of low rank, with non-zero entries only in rows corresponding to mesh points next to $\partial\Omega$. When $C = 0$, G can also be viewed as a discretization of $-\Delta_p$. The splitting

$$\tilde{\mathscr{A}} = \mathscr{S} - \mathscr{R} \tag{11}$$

* It is possible for this version of PCR to break down, with $\alpha_i = 0$. The Orthodir version, which uses a three-term recurrence to generate P_i, is guaranteed not to break down; it requires two additional AXPYs. Our implementation switches from the Orthomin to Orthodir direction update if $|\alpha_i| < 10^{-4}$, as described in Reference 15. In the experiments discussed in Section 4, this switch never took place.

then induces a stationary iteration applicable to (2), namely

$$\begin{pmatrix} u_{k+1} \\ p_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ p_k \end{pmatrix} + \begin{pmatrix} I & B^{\mathrm{T}} \\ 0 & -A_p \end{pmatrix} \mathscr{S}^{-1} \begin{pmatrix} f - (Au_k + B^{\mathrm{T}} p_k) \\ -(Bu_k - Cp_k) \end{pmatrix}. \tag{12}$$

This is used as the smoother for the multigrid solver for (2). Specific choices for $\mathscr{S}$ are given in Section 3.2.

Let R_u denote a restriction operator mapping velocity vectors in the fine grid (of width h) to the coarse grid (of width $2h$), let R_p similarly denote the restriction operator for the discrete pressure space and let P_u and P_p denote prolongation operators from the coarse spaces to the fine spaces. (For simplicity we are omitting explicit mention of h in this notation.) One step of V-cycle multigrid for solving (2), starting with initial guess u^0, p^0 and using $g = 0$, is as follows:

```
$(u^1, p^1) = \mathbf{MG}(u^0, p^0, f, g, k_1, k_2, h)$
if $h < h_0$, then   % Recursive call
    Starting with $u^0, p^0$, perform $k_1$ smoothing steps (12), producting $u^{1/3}, p^{1/3}$
    $r^{1/3} = f - (Au^{1/3} + B^{\mathrm{T}} p^{1/3}), \quad s^{1/3} = g - (Bu^{1/3} - Cp^{1/3})$
    $r_c^{1/3} = R_u r^{1/3}, \quad s_c^{1/3} = R_p s^{1/3}$
    $(u_c^{2/3}, p_c^{2/3}) = \mathbf{MG}(0, 0, r_c^{1/3}, s_c^{1/3}, k_1, k_2, 2h)$
    $u^{2/3} = u^{1/3} + P_u u_c^{2/3}, \quad p^{2/3} = p^{1/3} + P_p p_c^{2/3}$
    Starting with $u^{2/3}, p^{2/3}$, perform $k_2$ smoothing steps (12), producing $u^1, p^1$
else   % Coarse grid solve when $h = h_0$
    Solve $\begin{pmatrix} A & B^{\mathrm{T}} \\ B & -C \end{pmatrix} \begin{pmatrix} u^1 \\ p_1 \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$ directly
end if
```

We also use V-cycle multigrid derived from the discrete Laplacian as a preconditioner to approximate the action of A^{-1} for the Krylov subspace methods; this is defined analogously and we omit the details.[20] For all multigrid methods we use bilinear interpolation to define P_u and P_p, and $R_u = P_u^{\mathrm{T}}, R_p = P_p^{\mathrm{T}}$. The discrete operators at each level are derived from the discretization on the associated grid.

2.5. *Convergence properties*

We briefly outline some convergence properties of these methods; see the primary references for derivations of bounds. Each of the methods generates a sequence of iterates $u_i \approx u, p_i \approx p$ such that if e_i is a representation of the error, then $\lim_{i\to\infty}(\|e_i\|/\|e_0\|)^{1/i} = \rho$ for some norm $\|\cdot\|$. We refer to ρ as the *convergence factor.*

We are assuming that the discretization and choice of Q_M are such that

$$\lambda_1 \leqslant \frac{(q, (BA^{-1}B^{\mathrm{T}} + C)q)}{(q, Q_M q)} \leqslant \lambda_2, \tag{13}$$

where λ_1 and λ_2, and therefore $\kappa \equiv \lambda_2/\lambda_1$, are bounded independently of the mesh size of the discretization. This is the case, for example, when Q_M is a suitable approximation of the mass matrix in finite element discretization.[21,22] Note that κ is the spectral condition number of $Q_M^{-1}(BA^{-1}B^{\mathrm{T}} + C)$.[2]

The exact Uzawa algorithm has a convergence factor $\rho\ (I - \alpha Q_M^{-1}(BA^{-1}B^T + C))$. This is smallest for the choice $\alpha = 2/(\lambda_1 + \lambda_2)$, in which case it has the value $(\kappa - 1)/(\kappa + 1)$. Thus the convergence factor for the Uzawa algorithm is independent of the mesh. It is shown in Reference 12 that the performance of the inexact Uzawa algorithm is close to that of the exact one if the iterate u_{i+1} satisfies

$$\|f - B^{\mathrm{T}}p_i - Au_{i+1}\|_2 < \tau \|Bu_i - Cp_i\|_{Q_A^{-1}}, \tag{14}$$

where τ is independent of the mesh size.

The PCG method is analyzed in Reference 6, Theorem 1, where it is shown that the condition number of the coefficient matrix $\mathcal{M}$ of (5) is bounded by a constant proportional to κ. Thus standard results for CG[13] imply that the bound on the convergence factor for this method is proportional to $(\sqrt{\kappa}-1)/(\sqrt{\kappa}+1)$. The constant of proportionality depends on how close η_1 is to η_2 in (3), i.e. how well Q_A approximates A.

The PCR method is analyzed in References 7 and 8. The analysis shows that the eigenvalues of the preconditioned matrix $\mathcal{Q}^{-1}\mathcal{A}$ are contained in two intervals $[-a, -b] \cup [c, d]$, where a, b, c are d are positive constants that are independent of the mesh size. The sizes of the intervals depend on κ and the accuracy with which Q_A approximates A. It follows from the convergence analysis of CR[15,23] that the convergence factor for the preconditioned algorithm is independent of the mesh size. For example, it is shown[15] that if $d - c = a - b > 0$, then the convergence factor is bounded by $2[(1-\sqrt{\beta})/(1+\sqrt{\beta})]^{1/2}$, where $\beta = bc/ad$.

It is shown in Reference 24 that for finite difference discretization of (1) (see Section 3.1), two-grid variants of multigrid are convergent with a convergence rate independent of the mesh size. The analysis applies to the ILU smoothing of Section 3.2, although it requires that the prolongation be based on bidquadratic interpolation. In practice, bilinear interpolation has been observed to be sufficient.[11] Fourier analysis in Reference 10 also suggests that MG/DGS has a convergence rate independent of the mesh size.

Remark 1. Several other proposed methods share properties with the version of PCG under consideration. In particular, Verfürth[21] has shown that PCG applied directly to the Schur complement system has a convergence factor proportional to ρ_{CG}; however, this method requires accurate computation of the action of A^{-1} at each CG step.[25] Bank *et al.*[26] present a method making use of $Q_A \approx A$, with the convergence rate dependent on the accuracy of this approximation, but using an additional inner iteration on the pressure space.

3. SOLUTION COSTS

In this section we outline the computational costs required to solve four benchmark problems on $\Omega = (0, 1) \times (0, 1)$ for each of the solution methods of Section 2.

3.1. Benchmark problems

We use four discretizations to produce test problems: 'marker and cell' finite differences and three mixed finite element strategies.

1. *Finite differences.*[27] This consists of the usual five-point operator for each of the discrete Laplacian operators of (1), together with centred differences for the first derivatives ∇p and div u. For the discretization to be stable, it is necessary to use staggered grids in $\bar{\Omega}$. Figure 1 shows such grids on a mesh of width $h = \frac{1}{4}$. In order to define the velocity discretizations at grid points next to $\partial\Omega$, certain values outside Ω must be extrapolated; for example, this is needed to approximate $\partial^2 u_1/\partial y^2$ for points '$\times$' next to the bottom of $\partial\Omega$.

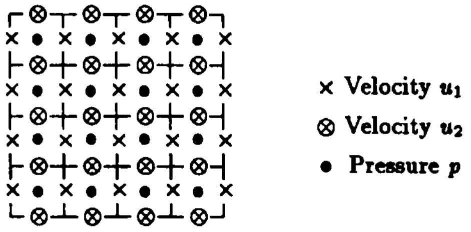

Figure 1. Staggered grids for finite difference discretization

2. *Linear/constant finite elements.* This choice consists of continuous piecewise linear velocities on a mesh of width h and piecewise constant pressures on a mesh of width $2h$. The discrete pressures are not required to be continuous. The coarser pressure grid ensures that the inf-sup condition holds.[4] We refer to this as the $P_1(h)P_0(2h)$ discretization.
3. *Piecewise linear finite elements.* Here continuous piecewise linear velocities on a mesh of width h are paired with continuous piecewise linear pressures on a mesh of width $2h$. The inf-sup condition is also satisfied. We call this the $P_1(h)P_1(2h)$ discretization.
4. *Stabilized piecewise linear finite elements.* A stable discretization using piecewise linear velocities and pressures on a single mesh can be obtained using a stabilization matrix $C = \beta h^2 A_n$, where A_n is the discrete Laplace operator defined on the pressure space, subject to Neumann boundary conditions.[28] This technique is equivalent to mini-element discretization[29] after elimination of the internal degrees of freedom. We use $\beta = 0{\cdot}025$ as recommended in Reference 30. We refer to this discretization as $P_1(h)P_1(h)$. The usual hat functions are used as the bases for linear velocities and pressures.

The coefficient matrix $\mathscr{A}$ of (2) for all these problems, as well as B^T, C and $BA^{-1}B^T + C$, is rank-deficient by one; the latter three matrices share a constant null vector. As a result, the discrete pressure solutions are uniquely defined only up to a constant. In exact arithmetic the solution methods under consideration correct the initial guess with quantities orthogonal to the null space of $\mathscr{A}$, so that the component of the null space in the computed solution is the same as in the initial guess. For the analysis the lower bound of (13) refers to the smallest non-zero eigenvalue.

Note that our goal in considering these problems is to compare the performance of the different solution strategies on a variety of problems. We highlight some properties of each of the problems as follows.

1. Finite differences, stable, #(pressure unknowns) $\approx$ #(velocity grid points).
2. Finite elements, stable, discontinuous pressures, #(pressure unknowns) $\approx \frac{1}{2}$#(velocity grid points).
3. Finite elements, stable, continuous pressure, #(pressure unknowns) $\approx \frac{1}{4}$ #(velocity grid points).
4. Finite elements, requires stabilization, continuous pressures, #(pressure unknowns) $\approx$ #(velocity grid points).

We are not comparing the accuracy achieved by the discretizations, but remark only that the three finite element discretizations display the same asymptotic convergence rates. See Reference 4, pp. 29 and 50 for comments on the accuracy of finite element discretization and Reference 31 for analysis of the finite difference scheme.

3.2. *Preconditioners and smoothers*

The Uzawa, PCR and PCG methods require choices of Q_A and Q_M. For all cases, Q_A consists of one step of V-cycle multigrid derived from the discrete Laplacian. The smoothing is based on damped point Jacobi iteration (so that Q_A is symmetric), with optimal damping parameter $\omega = \frac{3}{4}$. For the three finite element discretizations, Q_M is chosen to be the diagonal of the mass matrix M.[22] (In the case of the $P_1(h)P_0(2h)$ discretization, $Q_M = M$.) Although there is no mass matrix for finite differences, a natural analogue in two dimensions is $M = h^2I$ and this is used for Q_M with finite differences.

We consider two multigrid smoothing strategies. The first is a variant of the distributive Gauss–Seidel (DGS) iteration introduced by Brandt and Dinar.[10] The splitting operator of (11) is given by

$$\mathscr{S} = \begin{pmatrix} S_A & 0 \\ B & S_G \end{pmatrix},$$

so that the smoother (12) has the form

$$\begin{aligned} \tilde{u}_{k+1} &= S_A^{-1}[f - (Au_k + B^{\mathrm{T}}p_k)], \\ \tilde{p}_{k+1} &= S_G^{-1}[-B(u_k + \tilde{u}_{k+1}) + Cp_k], \\ u_{k+1} &= u_k + \tilde{u}_{k+1} + B^{\mathrm{T}}\tilde{p}_{k+1}, \\ p_{k+1} &= p_k - A_p\tilde{p}_{k+1}. \end{aligned}$$

For S_A we use the point Gauss-Seidel matrix derived from red–black ordering of the velocity grid. (That is, if $A = D - L - U$ with red–black ordering, then $S_A = D - L$.) For finite differences, $S_G = (1/\omega)T$, where T is the tridiagonal part of G and $\omega = \frac{3}{4}$; that is, S_G corresponds to a damped one-line Jacobi splitting. For $P_1(h)P_1(h)$ finite elements, S_G is the block Jacobi matrix derived from a two-line ordering of the underlying grid.[32] We refer to this multigrid method as MG/DGS.

The other multigrid smoother is the incomplete LU factorization (ILU) used by Wittum.[11] We use an ILU factorization of the matrix $\tilde{\mathscr{A}}$ of (10), with no fill-in in the factors. The ordering for $\tilde{\mathscr{A}}$ is problem-dependent. For finite differences it is derived from an uncoupled red–black ordering of the underlying grid. That is, the grid values for u_1 were listed first, in red–black ordering, followed by those for u_2 and then those for p. (See also Remark 4 below.) For $P_1(h)P_1(h)$ finite elements, $\tilde{\mathscr{A}}$ is ordered according to an uncoupled *lexicographic* ordering of the grid vectors. We denote this method by MG/ILU.

In choosing preconditioners and smoothers, we have attempted to use methods that are suitable for vector and parallel computers. Thus we are using point Jacobi smoothing for multigrid preconditioning, red–black Gauss–Seidel and line Jacobi for the DGS iteration and a red–black ordering for MG/ILU applied to finite differences. With the $P_1(h)P_1(h)$ discretization the operator G in the DGS method is a 19-point operator that has block property A for a two-line ordering of the pressure grid,* so that the two-line Jacobi splitting can be implemented efficiently in parallel. The ILU smoother used with this problem is not efficient on parallel computers. Our multigrid strategies do not address the issue of idleness of parallel processors for coarse grid computations; see References 35 and 35 for discussions of this point for the discrete Poisson equation.

Parameters are required for the Uzawa, PCG and multigrid methods and for the multigrid preconditioner. These are as follows.

Uzawa. The optimal value of α for the exact Uzawa method, determined empirically, is used for the inexact version. This requires computation of the extreme eigenvalues of $Q_M^{-1}(BA^{-1}B^{\mathrm{T}} + C)$.

* That is, G can be partitioned as a block tridiagonal matrix in which the block diagonal S_G is a set of decoupled blocks, each of which reflects connections within pairs of horizontal lines in the grid; see Reference 33, p. 445.

PCG. As noted in Section 2.3, the preconditioner must be scaled so that $\eta_1 > 1$ in (3). From the results of Reference 6 it is desirable to have η_1 close to unity. In all tests the scaling is chosen so that $1 < \eta_1 < 1{\cdot}02$. This requires computation of the smallest eigenvalue of $Q_A^{-1}A$.*

Multigrid. For the coarse mesh size h_0 in multigrid computations we chose the one of $h_0 = \frac{1}{2}$ and $h_0 = \frac{1}{4}$ that produced lower iteration counts. This turned out to be $h_0 = \frac{1}{2}$ for preconditioners and $h_0 = \frac{1}{4}$ for solvers. The coarse grid solution is obtained using Cholesky factorization for the preconditioners and singular value decomposition for the solvers.

Remark 2. For the Uzawa method the choice of Q_A does not guarantee that condition (14) is satisfied. The results of References 12 and 36 as well as those of 4 suggest that with multigrid for Q_A, (14) may be too stringent.

Remark 3. The effectiveness of the multigrid solvers depends on the fact that the commutator W in (10) is zero away from the boundary of Ω. This is true for the finite difference and stabilized $P_1(h)P_1(h)$ discretizations, where pressures and velocities are defined on the same grid, but not for the $P_1(h)P_1(2h)$ discretization. Our experiments indicate that a simple implementation of multigrid for $P_1(h)P_1(2h)$ is ineffective. See Reference 37, p. 248 for a discussion of this issue. For the $P_1(h)P_0(2h)$ discretization it is difficult to define the discrete pressure Poisson operator A_p and we have not tested multigrid in this case. It is possible to define versions of multigrid for these discretizations by grouping the unknowns in a special manner.[38] For example, for $P_1(h)P_1(2h)$ let the velocity grid be organized into four types of points, namely those at which the pressure unknowns are centred and the horizontal, vertical and diagonal neighbours of those points. The two velocity components are then each blocked into four subsidiary sets according to this reordering of grid points. A similar idea can be devised for the $P_1(h)P_0(2h)$ discretization. We have not examined these ideas.

Remark 4. For MG/ILU applied to the finite difference discretization, we also tested several alternative ordering strategies, including an uncoupled lexicographic ordering (i.e. like that used for $P_1(h)P_1(h)$) as well as several 'coupled' lexicographic orderings. For the latter strategies, velocity and pressure unknowns are not separated from one another.[39] The performances of MG/ILU for all these orderings were very close. For example, for $h = \frac{1}{32}$ as in Table IV below, the smallest average iteration count with one smoothing step was $10\frac{1}{3}$ and the largest was $11\frac{2}{3}$.

Remark 5. Better performance of MG/DGS and multigrid applied to the Poisson equation can be obtained with red–black Gauss–Seidel iteration for S_G and Q_A. In order to significantly improve performance, however, it is necessary to perform more relaxation steps at points near the boundary than at interior points.[40–42] This has negligible effect on the computational costs but makes implementation somewhat more complicated.

3.3. *Iteration costs*

We identify the costs per iteration of each of the methods by first specifying the 'high-level' operations of which they are composed and then determining the costs of each of these operations. High-level operations are defined to be matrix–vector products, inner products (denoted '(,)' in the tables of this section) and AXPYs. Note that each of the techniques under consideration is formulated with essentially the same set of these operations; consequently, we expect operation counts to give a good idea of their comparative performance.

* In the experiments described in Section 4, these were computed using a power method applied to $Q_A^{-1}A - I$; 5–10 steps were needed to obtain an estimate accurate to three significant digits.

Table I. High-level operations for all solution algorithms

	Matrix–vector product			AXPY	(,)
Uzawa	$1\,A$	$1\,B^T$	$1\,Q_A^{-1}$	$1\,(n_p)$	$1\,(n_u+n_p)$
	$1\,B$	$1\,C$	$1\,Q_M^{-1}$		
PCG	$1\,A$	$1\,B^T$	$1\,Q_A^{-1}$	$4\,(n_u+n_p)$	$3\,(n_u+n_p)$
	$2\,B$	$1\,C$	$1\,Q_M^{-1}$	$2\,(n_u)$	
PCR	$1\,A$	$1\,B^T$	$1\,Q_A^{-1}$	$5\,(n_u+n_p)$	$4\,(n_u+n_p)$
	$1\,B$	$1\,C$	$1\,Q_M^{-1}$		
Multigrid	$(1+k_1+k_2)A$		$1\,R_u$		
preconditioner	$(k_1+k_2)S_A^{-1}$		$1\,P_u$		
Multigrid solver	$1\,A$	$1\,B^T$	$1\,R_u$		$1\,(n_u+n_p)$
(excluding	$1\,B$	$1\,C$	$1\,R_p$		
smoother)	$1\,P_u$	$1\,P_p$			
	$1\,A$	$2\,B^T$	$1\,A_p$		
DGS smoother	$1\,B$	$1\,C$	$1\,S_A^{-1}$		
			$1\,S_G^{-1}$		
ILU smoother	$1\,A$	$2\,B^T$	$1\,A_p$		
	$1\,B$	$1\,C$	$1\,\mathcal{S}^{-1}$		

Table II. Costs for matrix–vector products

	Fin. diff.	$P_1(h)P_0(2h)$	$P_1(h)P_1(2h)$	$P_1(h)P_1(h)$
A	$10n^2$	$10n^2$	$10n^2$	$10n^2$
B, B^T	$4n^2$	$4n^2$	$8n^2$	$12n^2$
C	0	0	0	$5n^2$
Q_M^{-1}	$1n^2$	$0{\cdot}25n^2$	$0{\cdot}25n^2$	$1n^2$
S_A^{-1} (Jacobi)	$2n^2$	$2n^2$	$2n^2$	$2n^2$
S_A^{-1} (Gauss–Seidel)	$6n^2$	$6n^2$	$6n^2$	$6n^2$
S_G^{-1}	$3n^2$	—	—	$9n^2$
A_p	$5n^2$	—	—	$5n^2$
R_u, P_u	$6n^2$	$4{\cdot}5n^2$	$4{\cdot}5n^2$	$4{\cdot}5n^2$
R_p, P_p	$3n^2$	—	—	$2{\cdot}25n^2$
$\mathcal{S}^{-1}$	$19n^2$	—	—	$41n^2$

Table III. Cost factors

		Uzawa	PCR	PCG	MG/DGS	MG/ILU
Finite	$k_1 = k_2 = 1$	84	107	109	148	175
differences	$k_1 = k_2 = 2$	116	139	141	244	297
$P_1(h)P_0(2h)$	$k_1 = k_2 = 1$	79	98	101	—	—
	$k_2 = k_2 = 2$	111	130	133	—	—
$P_1(h)P_1(2h)$	$k_1 = k_2 = 1$	86	104	111	—	—
	$k_2 = k_2 = 2$	118	136	143	—	—
$P_1(h)P_1(h)$	$k_1 = k_2 = 1$	101	124	134	247	333
	$k_2 = k_2 = 2$	133	156	166	421	591

The high-level operations are shown in Table I. Matrix–vector products include operations with matrices that define the problem or method, such as A or R_u, as well as preconditioning and smoothing operators such as Q_A^{-1} and S_A^{-1}. The latter computations are themselves built from other matrix operations and some of these are also identified in the table. All multigrid entries correspond to operations performed on one grid level. For multigrid solvers the smoothing operations are presented separately; these operations would be performed k_1 times during pre-smoothing and k_2 times during post-smoothing. The lengths of the vector operations are listed in parentheses. We are assuming that one inner product will be used in the convergence test and the counts in the table include this.

The costs of matrix–vector products are estimated to be the number of non-zeros in the matrices used. This is roughly one-half the number of 'FLOPS' required and is also proportional to the number of memory references. These costs, for discretizations in which the velocity unknowns come from an $n \times n$ grid, are shown in Table II. The costs of vector operations are taken to be the length of the vectors.

Combining the data of Table I and II gives an estimate for the cost per iteration for each of the solution methods under consideration. These numbers are all proportional to n^2 and we present in Table III the cost factors obtained by omitting this factor, rounded to the nearest integer. For the multigrid methods (preconditioners and solvers) the cost of one full multigrid step is estimated as $\frac{4}{3}$ times the cost of the computations on the finest grid; this is approximately the cost of full recursive multigrid in two dimensions.

4. EXPERIMENTAL RESULTS

We now present the results of numerical experiments for solving (2). All experiments were performed in Matlab on a Sparc-10 workstation. For each solution algorithm we solved three problems derived from three choices of f consisting of uniformly distributed random numbers in $[-1, 1]$. The initial guess in all cases was $u_0 = 0, p_0 = 0$. The stopping criterion was

$$\|R_i\|_2 / \|R_0\|_2 < 10^{-6},$$

where

$$R_i = \begin{pmatrix} f \\ 0 \end{pmatrix} - \begin{pmatrix} A & B^T \\ B & -C \end{pmatrix} \begin{pmatrix} u_i \\ p_i \end{pmatrix}.$$

We found that performance was essentially in the asymptotic range for $h = \frac{1}{32}$ and all results are for this mesh size.

We present three types of data: iteration counts, estimates for convergence factors and plots of residual norms as functions of operation counts. The iteration counts are averages over three runs of the number of steps needed to satisfy the stopping criterion; these are shown in Table IV. The estimates for asymptotic convergence factors are the averages of $(\|\bar{R}_{5+i}\|_2 / \|\bar{R}_5\|_2)^{1/i}$ over all steps after step 5; here $\bar{R}_k$ represents the average of the kth residual norm over the three runs. These are shown in Table V. We chose step 5 rather than step 0 because performance was often better in the first few steps than later, when asymptotic behaviour is seen. Finally, Figures 2–5 plot the averages of the residual norms against operation counts.

We make the following observations on these results.

1. Where multigrid was tested, it requires the smallest number of iterations and has the smallest convergence factors. MG/ILU is superior to MG/DGS in these measures. These observations agree with those of Reference 11. In addition, where it is applicable, MG/ILU requires the smallest number of operations. (See Remark 5, however.)
2. The versions of the Krylov subspace methods and MG/DGS tested are roughly equal in cost.

Table IV. Iterations

		Uzawa	PCR	PCG	MG/DGS	MG/ILU
Finite differences	$k_1 = k_2 = 1$	31	38	27	22	12
	$k_1 = k_2 = 2$	29	32	21	14	9
$P_1(h)P_0(2h)$	$k_1 = k_2 = 1$	29	38	28	—	—
	$k_2 = k_2 = 2$	28	33	22	—	—
$P_1(h)P_1(2h)$	$k_1 = k_2 = 1$	89	54	36	—	—
	$k_2 = k_2 = 2$	89	50	29	—	—
$P_1(h)P_1(h)$	$k_1 = k_2 = 1$	39	44	30	20	8
	$k_1 = k_2 = 2$	38	39	24	10	7

Table V. Estimates of convergence factor

		Uzawa	PCR	PCG	MG/DGS	MG/ILU
Finite difference	$k_1 = k_2 = 1$	0·63	0·68	0·66	0·59	0·39
	$k_1 = k_2 = 2$	0·60	0·63	0·52	0·48	0·31
$P_1(h)P_0(2h)$	$k_1 = k_2 = 1$	0·64	0·68	0·69	—	—
	$k_2 = k_2 = 2$	0·60	0·67	0·54	—	—
$P_1(h)P_1(2h)$	$k_1 = k_2 = 1$	0·82	0·78	0·74	—	—
	$k_2 = k_2 = 2$	0·84	0·80	0·68	—	—
$P_1(h)P_1(h)$	$k_1 = k_2 = 1$	0·69	0·74	0·67	0·56	0·24
	$k_1 = k_2 = 2$	0·71	0·76	0·59	0·33	0·21

3. The performances of all these methods are very close. In terms of operation counts the ratio of costs of the most expensive and least expensive method is no worse than 2·2.
4. No Krylov subspace method is clearly superior to the others. PCG exhibits a somewhat faster convergence rate than PCR and the Uzawa algorithm is surprisingly competitive with the other two methods. This appears to derive from the dependence of PCG and PCR on both the spectral condition number κ from (13) and the accuracy of the preconditioning Q_A as an approximation to

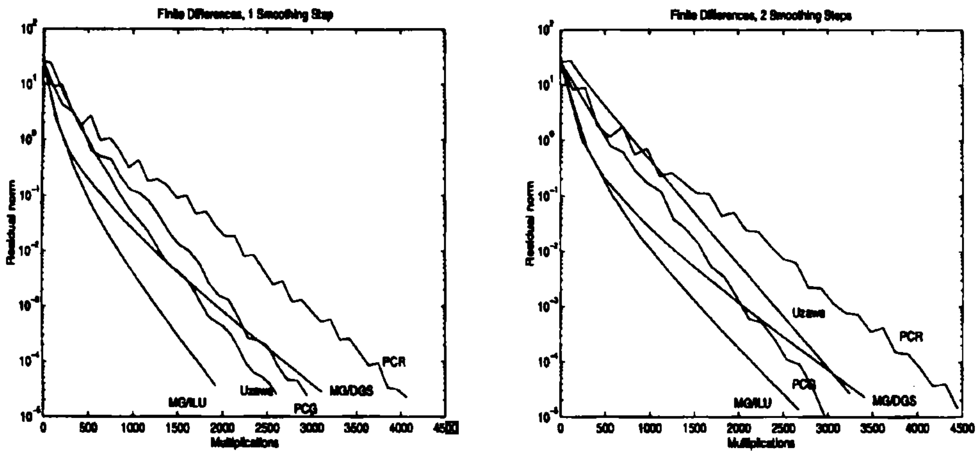

Figure 2. Operation counts for finite difference discretization

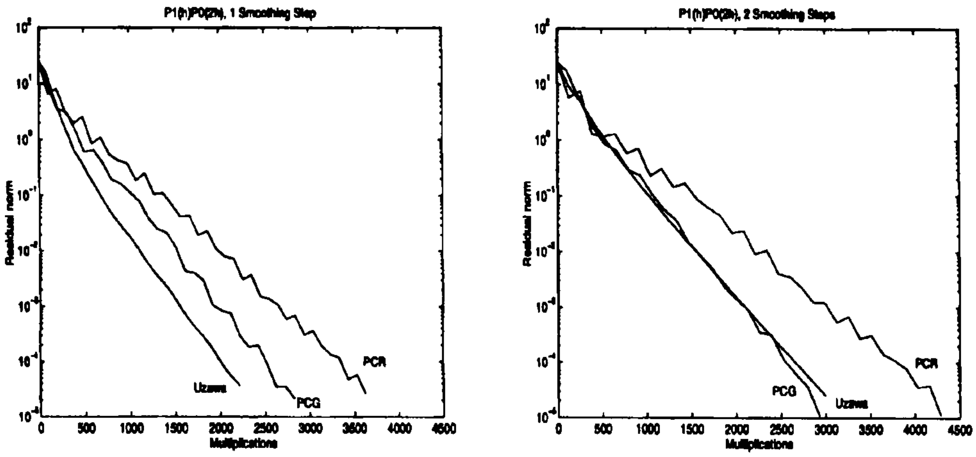

Figure 3. Operation counts for $P_1(h)P_0(2h)$ finite element discretization

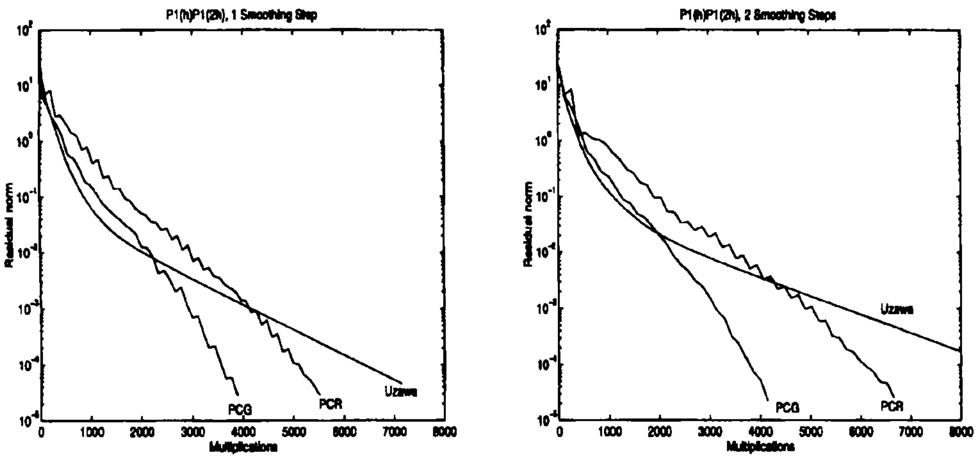

Figure 4. Operation counts for $P_1(h)P_1(2h)$ finite element discretization

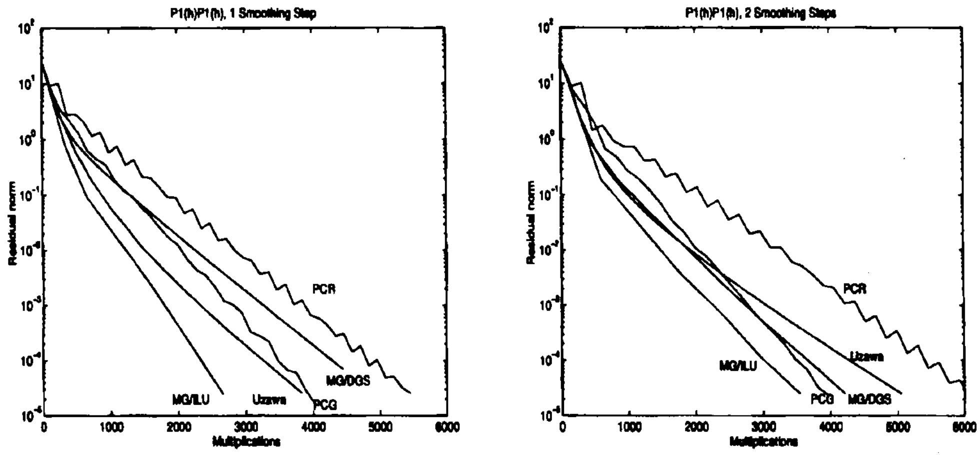

Figure 5. Operation counts for $P_1(h)P_1(h)$ finite element discretization

A; for both these methods the iteration counts go down in all cases when the number of smoothing steps in Q_A increases. The Uzawa method appears to be less sensitive to the accuracy of Q_A. The values of κ for the three problems are:

finite difference	4·14,	$P_1(h)P_1(2h)$	22·71,
$P_1(h)P_0(2h)$	4·87,	$P_1(h)P_1(h)$	9·91.

The Uzawa method is least effective for the $P_1(h)P_1(2h)$ discretization, which has the largest condition number.

5. The Uzawa and PCG methods depend on choices of iteration parameters. These can be estimated relatively inexpensively (e.g. using a coarse grid for the Uzawa method and a few steps of the power method for PCG), but this increases the cost of these methods and makes implementing them considerably more difficult. In contrast, PCR is independent of parameters, except for those needed for the multigrid preconditioning, and is therefore easier to implement. Thus there is a trade-off between these methodologies: PCR converges slightly more slowly than PCG and often the Uzawa method, but it has a simpler implementation.
6. For each of the solution strategies except PCG it is less expensive to use one smoothing step than two.

ACKNOWLEDGEMENTS

The author wishes to thank David Silvester for a careful reading of a preliminary version of this paper, and Andy Wathen and Achi Brandt for some helpful remarks.

This work was supported by the U.S. Army Research Office under grant DAAL-0392-G-0016 and by the National Science Foundation under grant ASC-8958544.

REFERENCES

1. F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer, New York, 1991.
2. M. Fortin and R. Glowinski, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, North-Holland, New York, 1983.
3. R. Glowinski, *Numerical Methods for Nonlinear Variational Problems*, Springer, New York, 1984.
4. M. Gunzburger, Finite Element Methods for Viscous Incompressible Flows, Academic, San Diego, CA, 1989.
5. K. Arrow, L. Hurwicz and H. Uzawa, *Studies in Nonliner Programming*, Stanford University Press, Stanford, CA, 1958.
6. J. H. Bramble and J. E. Pasciak, 'A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems', *Math. Comput.*, **50**, 1–17 (1988).
7. T. Rusten and R. Winther, 'A preconditioning iterative method for saddle point problems', *SIAM J. Matr. Anal. Appl.*, **13**, 887–904.
8. D. Silvester and A. Wathen, 'Fast iterative solution of stabilized Stokes systems. Part II: Using block preconditioners', *SIAM J. Numer. Anal.*, **31**, 1352–1367 (1994).
9. A. Wathen and D. Silvester, 'Fast iterative solution of stabilized Stokes systems. Part I: Using simple diagonal preconditioners', *SIAM J. Numer. Anal.*, **30**, 630–649 (1993).
10. A. Brandt and N. Dinar, 'Multigrid solutions to elliptic flow problems', in S. V. Parter (ed.), *Numerical Methods for Partial Differential Equations*, Academic, New York, 1979, pp. 53–147.
11. G. Wittum, 'Multi-grid methods for the Stokes and Navier–Stokes equations', *Numer. Math.*, **54**, 543–564, 1989.
12. H. C. Elman and G. H. Golub, 'Inexact and preconditioned Uzawa algorithms for saddle point problems', *Tech. Rep. UMIACS-TR-93-41*, Institute for Advanced Computer Studies, University of Maryland, College Park, MD, 1993; *SIAM J. Numer. Anal.*, **30**, 1645–1661 (1994).
13. G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd edn, Johns Hopkins University Press, Baltimore, MD, 1989.
14. S. F. Ashby, T. A. Manteuffel and P. E. Saylor, 'A taxonomy for conjugate gradient methods', *SIAM J. Numer. Anal.*, **27**, 1542–1568 (1990).
15. R. Chandra, S. C. Eisenstat and M. H. Schultz, 'The modified conjugate residual method for partial differential equations', in R. Vichnevetski (ed.), *Advances in Computer Methods for Partial Differential Equations II*, IMACS, New Brunswick, 1977, pp. 13–19.
16. P. Wesseling, *An Introduction to Multigrid Methods*, Wiley, New York, 1992.

17. J. Pitkäranta and T. Saarinen, 'A multigrid version of a simple finite element method for the Stokes problem', *Math. Comput.*, **45**, 1–14 (1985).
18. R. Verfürth, 'A multilevel algorithm for mixed problems', *SIAM J. Numer. Anal.*, **21**, 264–271 (1984).
19. P. M. Gresho and R. L. Saint, 'On pressure boundary conditions for the incompressible Navier–Stokes equations', *Int. j. numer. methods fluids*, 7, 1111–1145 (1987).
20. S. F. McCormick (ed.), *Multigrid Methods*, SIAM, Philadelphia, PA, 1987.
21. R. Verfürth, 'A combined conjugate gradient–multigrid algorithm for the numerical solution of the Stokes problem', *IMA J. Numer. Anal.*, 4, 441–455 (1984).
22. A. J. Wathen, 'Realistic eigenvalue bounds for the Galerkin mass matrix', *IMA J. Numer. Anal.*, 7, 449–457 (1987).
23. D. B. Szyld and O. B. Widlund, 'Variational analysis of some conjugate gradient methods', *East-West J. Numer. Math.*, **1**, pp. 51–74 (1993).
24. G. Wittum 'On the convergence of multi-grid methods with transforming smoothers', *Numer. Math.*, **57**, 15–38 (1990).
25. A. Ramage and A. J. Wathen, 'Iterative solution techniques for the Navier–Stokes equations', *Tech. Rep. 93-01*, School of Mathematics: University of Brisol, 1993; *Int. j. numer. methods fluids*, **19**, 67–83 (1994).
26. R. E. Bank, B. D. Welfert and H. Yserentant, 'A class of iterative methods for solving saddle point problems', *Numer. Math.*, **56**, 645–666 (1990).
27. F. H. Harlow and J. E. Welch, 'Numerical calculation of time-dependent of time-dependent viscous incompressible flow of fluid with free surface', *Phys. Fluids*, **8**. 2182–2189 (1965).
28. F. Brezzi and J. Pitkäranta, 'On the stabilization of finite element approximations of the Stokes problem', in W. Hackbusch (ed.), *Efficient Solutions of Elliptic Systems*, Braunschweig, NNFM Vol. 10, Vieweg, 1984, pp. 11–19.
29. D. Arnold, F. Brezzi and M. Fortin, 'A stable finite element for the Stokes equations', *Calcolo*, **21**, 337–344 (1984).
30. D. Silvester, 'Optimal low order finite element methods for incompressible flow', *Comput. Methods Appl. Mech. Eng.*, **111**, 357–368 (1994).
31. R. A. Nicolaides, 'Analysis and convergence of the MAC scheme I', *SIAM J. Numer. Anal.*, **29**, 1579–1591 (1992).
32. S. V. Parter, 'On "two-line" iterative methods for the Laplace and biharmonic difference equations', *Numer. Math.*, **1**, 240–252 (1959).
33. D. M. Young, *Iterative Solution of Large Linear Systems*, Academic, New York, 1970.
34. N. Decker, 'Note on the parallel efficiency of the Frederickson–McBryan multigrid algorithm', *SIAM J. Sci. Stat. Comput.*, **12**, 208–220 (1991).
35. P. O. Frederickson and O. A. McBryan, 'Normalized convergence rates for the PSMG method', *SIAM J. Sci. Stat. Comput.*, **12**, 221–229 (1991).
36. B. D. Welfert, 'Convergence of inexact Uzawa algorithms for saddle point problems', *Tech. Rep.*, Mathematics Department, University of Arizona, 1993.
37. W. Hackbusch, *Multi-Grid Methods and Applications*, Springer, Berlin, 1985.
38. A. Brandt, personal communication, 1995.
39. S. P. Vanka, 'Block-implicit multigrid solution of Navier–Stokes in primitive variables', *J. Comput. Phys.*, **65**, 138–158 (1986).
40. A. Brandt, 'Rigorous quantitative analysis of multigrid, I: Constant coefficients two-level cycle with l_2-norm', *SIAM J. Numer. Anal.*, **31**, 1695–1730 (1994).
41. A. Niestegge and K. Witsch, 'Analysis of a multigrid Stokes solver', *Appl. Math. Comput.*, **35**, 291–303 (1990).
42. I. Yavneh, 'Multigrid methods for incompressible flows', *Ph.D. Thesis*, Weizmann Institute of Science, Rehovot, 1991.