AN ORTHOGONAL PROJECTION ALGORITHM FOR LINEAR STOKES PROBLEMS

R. BRAMLEY
CENTER FOR SUPERCOMPUTING RESEARCH AND DEVELOPMENT
UNIVERSITY OF ILLINOIS - URBANA *

Abstract. A projection–based algorithm for solving symmetric indefinite systems is analyzed and applied to linear Stokes problems in CFD. Unlike other projection–based methods in CFD, this is a strictly linear algebraic approach that avoids the need to define boundary conditions for a pressure Poisson equation. The leading principal block need not be positive definite, and unlike reduced Hessian methods of nonlinear programming a null space basis matrix need not be created. No assumptions about the rank of the constraint block of the matrix are necessary. Advantages of using CG to solve the projected system are shown, and an approximation and preconditioning method is analyzed. After comparison with the preconditioned CG-accelerated Uzawa algorithm, the most commonly used iterative method for these problems, numerical results are given for problems arising in a large scale finite element industrial CFD code.

1. Introduction. An analysis is made here of an algorithm for solving symmetric indefinite systems

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{1}$$

where $B \in \Re^{n\times q}$ is arbitrary and $A \in \Re^{n\times n}$ is symmetric and positive definite on null(B^T), the kernel of B^T. The algorithm, originally proposed for generalized eigenvalue problems with A symmetric positive definite by Sameh and Wisniewski [22, 26], applies the conjugate gradient (CG) algorithm to the restriction of A to null(B^T) as defined by an orthogonal projector P onto null(B^T). Previous related work rejected such an approach as impractical, but the development of stable and effective sparse least squares solution methods over the last few years, combined with larger computer memories, has made implementation possible. The situation is similar to the use of barrier methods for optimization, which were rejected in the 1960's as numerically impractical but are now commonly used for large sparse linear and quadratic programming problems.

The algorithm has distinct advantages over existing solvers. In particular the matrix B need not be full rank, the matrix A need be positive definite on null(B^T), a basis matrix for null(B^T) need not be explicitly created, systems with coefficient matrix A do not need to be solved, the condition number of the derived coefficient matrix is no worse than that of A, and the possible number of distinct nonzero eigenvalues is $n - q$ instead of n. When applied to linear Stokes problems in computational fluid dynamics, no artificial boundary conditions need be defined on the pressure variable as is required with pressure Poisson methods. Instead, the purely linear algebraic approach taken here allows boundary conditions to be defined only on the velocity, as occurs naturally

* Work supported by grants NSF-MIP-8410110,DOE DE-FG02-85ER25001, AT&T-AFFL-67-SAMEH,NSF CCR-900000N(NCSA) NSF-CCR-8717942,AFOSR-85-0211, and Digital Equipment Corp.

with the primitive variable formulation of the Navier–Stokes equations. Furthermore, the residual norm updated by CG is the residual of (1) and not of the dual problem as is the case for CG-accelerated Uzawa methods.

Section 2 gives the origins of problems of the form (1). After deriving the projection method in Section 3, a review of related methods is given in Section 4. Section 5 then gives properties and practical implementation details and analytically compares the algorithm to commonly used Uzawa–type methods. A theorem is proven showing when approximation for the pseudoinverse of the projection system coefficient matrix is accurate, leading to a preconditioning method. Numerical results for linear Stokes problems from a finite element CFD code are given in Section 6.

2. Origins of the Problem. Systems with the form (1) arise in a variety of applications, all of which can be interpreted in terms of minimizing a quadratic functional $\frac{1}{2}u^TAu - f^Tu$, subject to the linear constraints $B^Tu = g$. One application is of primary interest here: minimization of a nonlinear function subject to equality constraints, particularly applied to the solution of linear Stokes problems arising in the mixed finite element solution of computational fluid dynamics (CFD) problems.

The standard equality constrained nonlinear programming problem is

$$\left\{ \begin{array}{l} \min\ f(x) \\ \text{subject to } g(x) = 0 \end{array} \right. \tag{2}$$

where $f(x) : \Re^n \to \Re$ and $g(x) : \Re^n \to \Re^q$ are twice continuously differentiable. Let x^* be a local minimum for (2). A common and efficient solution method models $f(x)$ by a quadratic function and $g(x)$ by a linear function and then solves the resulting quadratic programming problem

$$\left\{ \begin{array}{l} \min\ \nabla f(x)^Tu + \frac{1}{2}u^TAu \\ \text{subject to } g(x) + \nabla g(x)^Tu = 0 \end{array} \right. \tag{3}$$

for a search direction u. The new estimate $\bar{x}$ is then given by $\bar{x} = x + \lambda u$, where λ is a suitable stepsize. In order to account for possible curvature of the constraint function $g(x)$, A is an approximation to the Hessian with respect to x of the Lagrangian function $L_{xx}(x,p) = f(x) + p^Tg(x)$ rather than of $f(x)$ alone. For nonlinear programming problems, A is usually a quasi-Newton updated matrix, with the update chosen numerically to keep A "safely " positive definite either by using an augmented Lagrangian approach [17, 2] or by using a partial update suggested by Powell [21]. General references for such sequential quadratic programming (SQP) methods can be found in [12, 7]. Optimality conditions for (3) are the existence of a Lagrange multiplier $p \in \Re^q$ such that

$$\begin{array}{rcl} Au + \nabla g(x)p & = & -\nabla f(x) \\ \nabla g(x)^Tu & = & -g(x) \end{array} \tag{4}$$

Letting $B = \nabla g(x)$, (4) is clearly of the form (1).

One particular application area is solving the linear Stokes problem for the incompressible flow of a Newtonian fluid:

$$\begin{aligned} \alpha u - \nabla\cdot(\mu_e \nabla u) + \nabla p &= f \\ \nabla\cdot u &= 0 \end{aligned} \tag{5}$$

where $\alpha > 0$ is a constant and suitable boundary conditions have been defined on the velocity vector u. Boundary conditions for the pressure p are not imposed by the physical formulation of the Navier–Stokes equations, but are often computationally imposed for reasons described later. In (5), $\mu_e \geq 0$ is the "effective" viscosity which can include Reynolds-averaged turbulence modeling effects. In the simplest case $\alpha = 1/\delta t$ with δt the time step and $\mu_e = 1/R$, where the constant R is the fluid Reynolds number:

$$\begin{aligned} \left(\frac{1}{\delta t}I - \frac{1}{R}\nabla^2\right) u + \nabla p &= f \\ \nabla\cdot u &= 0 \end{aligned} \tag{6}$$

When discretized with a Galerkin finite element method satisfying the (LBB) condition [16], (6) has the form (1), where A is the discrete version of $\left(\frac{1}{\delta t}I - \frac{1}{R}\nabla^2\right)$ and B is the gradient operator ∇. Important details of the derivation and of boundary conditions have been omitted here, and the reader is referred to [16, 8, 25] for more information.

There are five characteristics about the systems of the form (1) that the sample application entails:

1. The systems can be huge and sparse. For three-dimensional CFD problems, some currently used codes solve problems with $n = \mathcal{O}(10^5)$ or larger.
2. The smallest eigenvalue of the symmetric matrix A is frequently kept bounded away from zero. For CFD application this can be seen from (6) when δt is small and/or R is large, both important cases in applications. For nonlinear programming methods in general this occurs when quasi-Newton updating is used.
3. The matrix B has significantly fewer columns than A does. For conforming finite element methods with P_1–P_2 elements in three-dimensional CFD applications $q \approx \frac{1}{7}n$ is typical.
4. At a local minimum x^*, the Hessian of the Lagrangian $\nabla_{xx}L(x^*, p^*)$ in (3) which A approximates is typically not positive definite except in degenerate cases. However, $\nabla_{xx}L(x^*, p^*)$ restricted to the null space of $B = \nabla g(x^*)$ is positive semidefinite when second order necessary conditions hold at x^*, and is positive definite when second order sufficiency conditions hold.
5. The matrix B in (6) is rank-deficient, unless boundary conditions are imposed on the pressure p .

Solvers for systems of the form (1) typically make assumptions that conflict with the last three characteristics, and in CFD applications in particular it is common for the physical model to be perturbed in order to achieve a system that can be solved.

3. Derivation of the Algorithm. First, the classical Uzawa system is derived for comparison purposes. Suppose A is nonsingular and B is full rank. Solving for u in $Au + Bp = f$ and substituting into $B^T u = g$ gives

$$B^T A^{-1} B p = B^T A^{-1} f - g \tag{7}$$

Solving this system for p and substituting into the first equation gives the expression

$$u = \left[A^{-1} - A^{-1}B(B^T A^{-1} B)^{-1} B^T A^{-1}\right] f + A^{-1}B(B^T A^{-1}B)^{-1} g \tag{8}$$

A classical Uzawa-type algorithm [1, 8] normally starts with an estimate for p , solves $Au = f - Bp$ for u, and then updates p by a Richardson iteration: $p \leftarrow p + \alpha_k(B^T u - g)$. To simplify the discussion assume without loss of generality that (as is the case for CFD applications) $g = 0$; a nonzero g changes the right hand side vector of the systems implicitly defined by (7) and (8) but not the coefficient matrices. The Uzawa algorithm is a steepest descent method for solving (7), and if A is symmetric and positive definite a more efficient preconditioned conjugate gradient (CG) algorithm can be used instead. Note that A^{-1} need not be explicitly formed, but the requisite matrix–vector products can be found by solving systems with A , possibly by another preconditioned CG algorithm. This yields an inner–outer iterative method: each outer iteration requires multiplying by $B^T A^{-1} B$, and each inner iteration requires multiplying by A. This approach has been successfully implemented in the industrial CFD code *N3S* by Electricité de France [3].

The algorithm analysed here can be similarly derived, but without assuming anything about the matrices A and B . Solve $Au + Bp = f$ for p in a least–squares sense: $p = B^\dagger(f - Au)$, where $B^\dagger$ is the Moore-Penrose pseudoinverse of B. Substitute this expression for p back into the *same* equation to get

$$PAu = Pf, \tag{9}$$

where $P = I - BB^\dagger$ is the orthogonal projector onto null(B^T). Furthermore, $B^T u = 0$ iff $u = Pu$, and substituting this into (9) gives

$$PAPu = Pf. \tag{10}$$

Conversely, suppose u satisfies equation (10), and set $\tilde{u} = Pu$. Then $\tilde{u} \in \mathrm{null}(B^T)$, i.e., $B^T\tilde{u} = 0$. Furthermore, setting $p = B^\dagger(f - A\tilde{u})$ implies

$$\begin{aligned} A\tilde{u} + Bp &= A\tilde{u} + BB^\dagger f - BB^\dagger A\tilde{u} \\ &= PA\tilde{u} + BB^\dagger f \\ &= Pf + BB^\dagger f. \end{aligned}$$

This proves:

THEOREM 3.1. *Let $A \in \Re^{n\times n}$ and $B \in \Re^{n\times q}$ be arbitrary, and $P = I - BB^\dagger$. Then (Pu, p) solves*

$$\begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$

if and only if $PAPu = Pf$ *and* $p = B^{\dagger}(f - APu) + v$ *for some* $v \in \mathrm{null}(B)$.

Properties and practical solution methods for solving (10) are presented in Section 5. First a survey of related approaches is given in Section 4.

4. History of Related Methods. As might be guessed from the ease of derivation, algorithms related to the projected system (10) have appeared frequently in the past, especially in the application targeted here. Some of this work has concentrated on applying the projected system to the continuous model, instead of as a strictly linear algebraic solution method. For the general nonlinear programming problem, iterates are restricted more or less stringently to the manifold $g^{-1}(0)$ [24, 10] with typical hypotheses including that the Jacobian matrix $B = \nabla g(x)$ is full rank for all x. Practical computational algorithms concentrate on two issues. The first is how to select a quasi-Newton update to keep the restriction of A to the subspace $\mathrm{null}(B^T)$ positive definite and how to update only an $(n-q) \times (n-q)$ matrix on each iteration. This issue is not further addressed in this paper, because we concentrate on the linear problem (1).

The second deals with how to computationally restrict each search direction u to the tangent space to $g(x)$, and is important here. All versions rely on finding a matrix $Z \in \Re^{n\times(n-q)}$ with $\mathrm{range}(Z) = \mathrm{null}(B^T)$; two numerical approaches are used, although both can be put into the same framework by considering an arbitrary right inverse for B [9]. The first is to partition $B^T = [B_1, B_2]$ with B_1 nonsingular (note that a permutation of the rows of B may be required), and then setting

$$Z = \begin{bmatrix} -B_1^{-1}B_2 \\ I_{n-q} \end{bmatrix}.$$

This approach can fail if B is not full rank, but has the advantage of being cheaply computed when $q \ll n$ and when the constraint function g is linear. The second approach performs a QR factorization $B = QR$ and takes $Z = Q_2$, where $Q = [Q_1, Q_2]$ has been partitioned so that Q_2 is an orthonormal basis matrix for $\mathrm{null}(B^T)$. The system solved is then

$$Z^T AZv = -Z^T \nabla f(x). \tag{11}$$

Note that storage can be prohibitive in this case because Q_2 is generally dense even when B is sparse. System (11) is closely related to (10), as can be seen by premultiplying by Z and using $Z^T Z = I$, $ZZ^T = P$ to get

$$ZZ^T AZ(Z^T Z)v = (PAP)Zv = -ZZ^T \nabla f(x) = -P\nabla f(x).$$

Taking $u = Zv$ and $f = -\nabla f(x)$ gives the projected system (10). The key advantage of using (10) instead of (11) is that computationally a null space basis matrix is more difficult and costly to find than a range space basis matrix. Reasons for avoiding the projected system are that it is singular (but consistent) whenever $\mathrm{null}(B^T)$ is nontrivial, and the computation of P and $B^{\dagger}$ in general has in the past been considered impractical. According to Tanabe in 1979 [24], his formulae using pseudoinverses " . . . are of little value in practical computation, their main use being to give a geometric insight ...." Fletcher [7, page 88] also noted the relationship between solving the reduced Hessian system (11) and the projected system (10).

In [22, 26] system (10) was referred to and numerically solved by a CG algorithm as the main component of an algorithm for solving generalized eigenvalue problems by a trace minimization method. Since dense routines from Linpack were used to factor the matrix B, the distinction between using Q_1 and Q_2 was not important, and in fact the projector P was explicitly formed. The matrix A was assumed to be symmetric and positive definite, and the authors noted that only one projection is needed per iteration when initializing with $u_0 \in \text{null}(B^T)$. Recently, applying CG to solve a symmetric indefinite system was proposed to solve domain–decomposed problems in finite elements [6]; here the Lagrange multipliers p correspond to the coupling between subregions in a finite element mesh. A preconditioner based on the block structure of the problem was also suggested.

Projection–motivated solution methods specifically for CFD problems date back at least to 1968 [4]. CFD researchers have identified the incompressibility condition $\nabla \cdot u = 0$ as the main source of numerical difficulty, and eliminate it by applying the divergence operator to the first equation in (5) to obtain a Poisson equation on the pressure. As Gresho and others have shown [15, 13, 14] such a procedure is equivalent to orthogonally projecting the velocity field onto the set of null divergence vectors, and great care must be taken in defining boundary conditions for the pressure. Solving the projected system for the linearized problem (1), however, does not require defining such boundary conditions.

The methods that Gresho deals with also address the time–stepping procedure used in the CFD solver, unlike the approach taken here. In the context of the simplified problem (5), the algorithms he examines generally solve $Au = f - Bp$ for u and then project u onto $\text{null}(B^T)$. Such a scheme approximately solves (1) by setting

$$u = PA^{-1}(f - Bp), \tag{12}$$

which is examined further in Section 5.

5. Properties of the Algorithm. The algorithm proposed applies CG to the system $PAPu = Pf$. This section first shows that with proper initialization only one projection per CG iteration need be computed, and the residual norm updated by CG is the residual for (1). A comparison is made with the system solved by CG–accelerated Uzawa algorithm, and an approximation method is described and bounds are given for it. We assume throughout this section that A is symmetric; for the quadratic programming problem (3) this follows if $f, g \in \mathcal{C}^2$. This Section refers to u as the velocity and p as pressure to emphasize the application to CFD problems, but the results are independent of the application.

5.1. Use of CG for Solving the Projected System. Formally the CG algorithm applied to $PAPu = Pf$ is:

1) Initialize:

$$
\begin{array}{rcl}
u_0 & = & \text{given} \\
\epsilon & > & 0 \\
d_0 & = & r_0 = P(f - APu_0) \\
\rho_0 & = & r_0^T r_0 \\
k & = & 0
\end{array}
\tag{13}
$$

2) Iterate:

$$
\begin{array}{rrcl}
 & w_k & = & PAPd_k \\
 & \alpha_k & = & \rho_k / d_k^T w_k \\
 & u_{k+1} & = & u_k + \alpha_k d_k \\
 & r_{k+1} & = & r_k - \alpha_k w_k \\
 & \rho_{k+1} & = & r_{k+1}^T r_{k+1} \\
\text{If} & \rho_{k+1} & < & \epsilon_\rho \ \text{ stop.} \\
 & \beta_{k+1} & = & \rho_{k+1}/\rho_k \\
 & d_{k+1} & = & r_{k+1} + \beta_{k+1} d_k
\end{array}
\tag{14}
$$

THEOREM 5.1. *Let $u_0 \in null(B^T)$, $A|_{null(B^T)}$ be positive definite, and $P = I - BB^\dagger$. Then:*

(a) the CG algorithm converges to the unique minimum norm solution to $PAPu = Pf$,
(b) the iterates u_k, residuals r_k, and search directions d_k remain in null(B^T) for all k,
(c) the matrix–vector product $w_k = PAPd_k$ can be replaced with $w_k = PAd_k$, and
(d) the vector $(r_k, \mathbf{0})$, $\mathbf{0} \in \Re^q$, is the residual for the original symmetric indefinite system (1).

Proof. Since $A \mid_{\mathrm{null}(B^T)}$ is positive definite, PAP is positive semidefinite and $\mathrm{null}(PAP) = \mathrm{null}(P) = \mathrm{range}(B)$. Hence the system is consistent, and convergence follows from classical work by Hestenes and Stiefel [18]. Note that $d_0 = r_0 = P(f - APu_0) \in \mathrm{null}(B^T)$, and suppose that $u_k, d_k, r_k \in \mathrm{null}(B^T)$. Then $w_k = PAPd_k \in \mathrm{null}(B^T)$, and so

$$
\begin{array}{rcl}
u_{k+1} & = & u_k + \alpha_k d_k \in \mathrm{null}(B^T), \\
r_{k+1} & = & r_k - \alpha_k w_k \in \mathrm{null}(B^T), \\
d_{k+1} & = & r_{k+1} + \beta_{k+1} d_k \in \mathrm{null}(B^T),
\end{array}
$$

proving (b) by induction. The unique minimum norm solution to $PAPu = Pf$ is the one with $u \in \mathrm{null}(B^T)$, so (a) also follows from this. Since $Pd_k = d_k$, the first projection operation can be omitted in computing w_k, verifying (c). Finally, note that from the derivation of system (10), the pressure p corresponding to a given velocity u is $p = B^\dagger(f - Au)$. Thus

$$
\begin{array}{rcl}
r_k & = & Pf - PAPu_k \\
 & = & P(f - Au_k)
\end{array}
$$

$$
\begin{aligned}
&= (I - BB^{\dagger})(f - Au_k) \\
&= f - Au_k - BB^{\dagger}(f - Au_k) \\
&= f - Au_k - Bp_k.
\end{aligned}
$$

Since $B^T u = 0$, (d) follows. □

Other good features follow from using CG on (10). As suggested by the proof, the pressure can also be updated at the cost of a SAXPY operation of length q on each iteration by initializing $p_0 = B^{\dagger}(f - Au_0)$, since

$$
\begin{aligned}
p_{k+1} &= B^{\dagger}(f - A(u_k + \alpha_k d_k)) \\
&= p_k - \alpha_k B^{\dagger} A d_k
\end{aligned}
$$

and the vector $B^{\dagger} A d_k$ is computed as an intermediate step in finding w_k.

Secondly, it may be inefficient to initialize with $u_0 \in \text{null}(B^T)$ (although $u_0 = 0$ satisfies the condition), since starting with a velocity vector from the previous time step may be a better approximation. In this case u_k has a component in range(B) which is unchanged during the iterations, and the final velocity can be obtained after the CG algorithm at the cost of another projection; another alternative is to initialize with Pu_0 instead of u_0.

Similarly, note that if the second block equation in (1) is inhomogeneous ($B^T u = g$, with $g \neq 0$), then the necessary initialization is $u_0 = v + (B^T)^{\dagger} g$, with $v \in \text{null}(B^T)$. In this case, the residual is initialized as $r_0 = P(f - Av)$. Again all the relevant vectors remain in $\text{null}(B^T)$, except for $u \in \text{null}(B^T) + (B^T)^{\dagger} g$. The extra cost entailed by this is less than that required to compute one projection with P, and can be overlapped with the computation of the initial residual by solving the triangular systems with two right hand side vectors, followed by multiplying the two vectors by B. When B is not full rank, it can be shown that this initialization has a residual in the normal equations for (1) of $APA(B^T)^{\dagger} g = AP(Au + Bp - f)$. Hence if $g = 0$, a least squares solution is found but otherwise this is not assured.

Finally, if A is not symmetric positive definite on $\text{null}(B^T)$ and the initializations described above are chosen, then applying a Krylov subspace iterative method such as GMRES or ORTHOMIN to $PAPu = Pf$ will also keep the relevant vectors in $\text{null}(B^T)$. Furthermore, if the symmetric part of A is positive definite then so is the symmetric part of PAP; this property is important to assure convergence of many CG-like methods [5].

5.2. Comparison with Uzawa–type Systems. A comparison is now made with solving Uzawa–type systems, the key computational component of which is the solution of

$$
B^T A^{-1} B p = B^T A^{-1} f \tag{15}
$$

where A is symmetric and positive definite on all of $\Re^{n \times n}$ and B is full rank. The relationship with (10) follows immediately from

$$
(PAP)^{\dagger} = A^{-1} - A^{-1} B (B^T A^{-1} B)^{-1} B^T A^{-1}, \tag{16}
$$

a relationship that can be verified by checking the Moore-Penrose conditions. In this case solving (10) corresponds to solving a dual problem to (15).

Since B is full rank, $P = I - B(B^TB)^{-1}B^T$. Often in CFD applications B is independent of time, so the Cholesky factor R of B^TB can be found once at the beginning of the CFD solver either by forming B^TB and using Cholesky factorization, or by using a more numerically stable QR factorization. To save on storage, only the triangular factor R need be saved, and the requisite matrix-vector products found as $Pv = v - BR^{-1}R^{-T}B^Tv$, requiring the solution of two triangular systems of order q.

Note that solving either (15) and (10) with a CG-like iterative method requires on each iteration products with either A^{-1} or $(B^TB)^{-1}$, respectively. However, the first is of order n and the second of order $q < n$. Furthermore, from the minimax characterization of singular values the smallest singular values of A and PAP satisfy $\sigma_{\min}(A) \leq \sigma_{\min}(PAP)$ while the largest satisfy $\sigma_{\max}(A) \geq \sigma_{\max}(PAP)$. Hence the condition number of PAP is no worse than that of A. Additionally, the possible number of distinct nonzero eigenvalues of PAP is $n-q$, compared to n for A. These results suggest that the number of CG iterations required to solve (10) is no larger than that required to solve a single system with coefficient matrix A. However, in practice the number of iterations the CG method requires depends on the total distribution of eigenvalues, and the simple bounds provided by the condition number or the number of distinct eigenvalues may be irrelevant.

Another advantage of the proposed algorithm is given by part (d) of Theorem 5.1: an iterative method applied to the Uzawa system does not give a direct measure of the residual norm of the original problem (1), while the projected method does. Furthermore, this permits using a stopping test that depends on certain components of the residual vector, allowing the user to require, e.g., more stringent accuracy near boundaries or other regions of interest, or in the case of adaptive finite element methods to give local error measures.

One weakness of the projected system is that it is difficult to use an incomplete factorization based preconditioner for the matrix A. However, the Uzawa method also suffers from this drawback and a commonly used preconditioner for $B^TA^{-1}B$ is $B^TD^{-1}B$, where D is the diagonal of A. The projected system also easily allows the use of diagonal preconditioning of A. However, for (6) the matrix A is usually well-conditioned, and difficulties of solution occur in the reduced system $B^TA^{-1}B$ instead, making the projected system a more attractive option.

A more precise comparison of the work in the two methods requires specifying a CG-accelerated Uzawa scheme, which is now done. Because the implementation of the projection method described above uses the full factorization of B^TB, the Uzawa preconditioner used here will also use the full factorization of $B^TD^{-1}B$. It should be noted that within the CFD code *N3S*, the most frequent source of failures in the Stokes solver is the nonconvergence of incomplete Cholesky preconditioned CG applied to systems with coefficient matrix $B^TD^{-1}B$, so using a full factorization not only makes the methods comparable in storage but also is a reasonable choice.

Given the factorization $R^TR = B^TD^{-1}B$, the Uzawa algorithm is then:

1) Initialize:

$$
\begin{array}{rcl}
p_0 & = & \text{given} \\
\epsilon & > & 0 \text{ given} \\
\text{Solve } Au_0 & = & f - Bp_0 \\
r_0 & = & -B^T u_0 \\
k & = & 0
\end{array} \tag{17}
$$

2) Iterate:

$$
\begin{array}{rcl}
\text{Solve } (R^T R) g_k & = & r_k \\
\rho_k & = & r_k^T g_k \\
\text{If } \rho_k & < & \epsilon \text{ stop.} \\
\text{If } k & = & 0 \\
 & & d_k = g_k \\
\text{Else} & & \\
 & & \beta_k = \rho_k / \rho_{k-1} \\
 & & d_k = g_k + \beta_k d_{k-1} \\
\text{EndIf} & & \\
\text{Solve } Az_k & = & Bd_k \\
w_k & = & B^T z_k \\
\alpha_k & = & \rho_k / g_k^T w_k \\
u_{k+1} & = & u_k + \alpha_k z_k \\
p_{k+1} & = & p_k - \alpha_k d_k \\
r_{k+1} & = & r_k - \alpha_k w_k \\
k & = & k+1
\end{array} \tag{18}
$$

Suppose an average of m iterations are needed to solve each system with coefficient matrix A. Let *saxpy(s)* denote a saxpy operation of length s, and similarly for *dotproduct(s)*. Then Table 1 compares the total work performed by the preconditioned Uzawa method above and the projection method. Let k_U be the number of Uzawa iterations performed and k_P the number of projection method iterations. If work is measured by multiplications by A, then the breakeven point is for $k_P = k_U(m+1) + 2m + 1$, and if the work is measured by triangular solves it is $k_P = k_U$. In practice the breakeven point will lie between those two extremes.

For a final comparison, note that the Uzawa method solves the first block equation in (1) for u and substitutes this into the second block equation. A simple analysis of the algorithm shows that if the systems with coefficient matrix A are solved exactly, then the Uzawa algorithm keeps the residual of $Au + Bp = f$ exactly equal to zero, while attempting to drive the residual of $B^T u = 0$ down to zero iteratively. On the other hand, if the projections are computed exactly the projection method keeps the residual of $B^T u = 0$ exactly equal to zero and attempts to drive the residual of $Au + Bp = f$ down to zero iteratively.

5.3. An Approximation Method. When considering $(PAP)^\dagger$ it is tempting but incorrect to use the formula $(GH)^\dagger = H^\dagger G^\dagger$. Since A is invertible and $P^\dagger = P$ for any

Operation	Uzawa Initial	Uzawa Per Iter	Projection Initial	Projection Per Iter
$B^TD^{-1}B$ or B^TB Solve		1	1	1
A-multiply	m	m	1	1
B-multiply	1	1	1	1
B^T-multiply	1	1	1	1
saxpy(n)	$m+1$	$3m+1$	2	4
saxpy(q)	1	3		
dotproduct(n)	$2m+1$	$2m+1$	1	2
dotproduct(q)		2		

TABLE 1
Comparison of Uzawa and Projection Methods

orthogonal projector, this gives $PA^{-1}P$ – compare to the CFD algorithms of Gresho as described in Section 4. The question arises: how close are $PA^{-1}P$ and $(PAP)^\dagger$? When can the first be substituted for the second? The next theorem answers those questions, but first two devices of linear algebra are defined.

If A is symmetric and positive definite, there is a symmetric and positive definite matrix $A^{\frac{1}{2}}$ such that $(A^{\frac{1}{2}})^2 = A$; this can be defined with the Schur decomposition $A = QDQ^T$, $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ by setting $A^{\frac{1}{2}} = QD^{\frac{1}{2}}Q^T$, where $D^{\frac{1}{2}} = \mathrm{diag}(\lambda_1^{\frac{1}{2}}, \ldots, \lambda_n^{\frac{1}{2}})$. Secondly, the smallest angle between two subspaces L and M is defined as the $\theta \in [0, \pi/2]$ such that

$$\cos\theta = \max\left\{\frac{u^Tv}{\| u \| \| v \|} : u \in L, v \in M, u \neq 0, v \neq 0\right\} \tag{19}$$

Finally, note that if U is a full–rank basis matrix for B, formula (16) holds with B replaced by U.

THEOREM 5.2. *Let A be symmetric and positive definite, and let $[U, V]$ be an orthogonal matrix such that range(U) = range(B) and range(V) = null(B^T). Let $G = PA^{-1}P - (PAP)^\dagger$. Then:*

(a) $G \cdot$ range$(B) = 0$,

(b) $G = PA^{-\frac{1}{2}}QA^{-\frac{1}{2}}P$, where Q is the orthogonal projector onto range$(A^{-\frac{1}{2}}U)$,

(c) $\| G \|_2 \leq \cos^2\theta \cdot \| \left(A \mid_{null(B^T)}\right)^{-1} \| \leq \cos^2\theta \cdot \lambda_{\min}^2(A)$ where θ is the smallest angle between null$(B^TA^{\frac{1}{2}})$ and range$(A^{-\frac{1}{2}}B)$, and

(d) $G = 0$ iff range(B) is an invariant subspace of A.

Proof. From basic properties of the pseudoinverse, $\mathrm{null}((PAP)^\dagger) = \mathrm{null}(PAP) = \mathrm{null}(P) = \mathrm{range}(B)$. Since $\mathrm{null}(PA^{-1}P) = \mathrm{range}(B)$ also, (a) follows. Note that this also shows that $G(I - P) = 0$. Now from the definition of $[U, V]$, $P = I - UU^T$. Using $A^{-1} = PA^{-1} + UU^TA^{-1}$ and the idempotency of P,

$$G = GP + G(I-P) = GP$$

$$
\begin{aligned}
&= [PA^{-1}P - (PAP)^{\dagger}]P \\
&= [PA^{-1} - A^{-1}(I - U(U^TA^{-1}U)^{-1}U^TA^{-1})]P \\
&= [PA^{-1} - PA^{-1}(I - U(U^TA^{-1}U)^{-1}U^TA^{-1}) - UU^TA^{-1}(I - U(U^TA^{-1}U)^{-1}U^TA^{-1})]P \\
&= [PA^{-1}U(U^TA^{-1}U)^{-1}U^TA^{-1} - UU^TA^{-1} + UU^TA^{-1}U(U^TA^{-1}U)^{-1}U^TA^{-1}]P \\
&= PA^{-1}U(U^TA^{-1}U)^{-1}U^TA^{-1}P \\
&= PA^{-\frac{1}{2}}S(S^TS)^{-1}S^TA^{-\frac{1}{2}}P,
\end{aligned}
$$

where $S = A^{-\frac{1}{2}}U$. Since $Q \equiv S(S^TS)^{-1}S^T$ is the orthogonal projector onto $\text{range}(S) = \text{range}(A^{-\frac{1}{2}}U)$, (b) is proven. Next, from (a) and (b)

$$G = PA^{-\frac{1}{2}}QA^{-\frac{1}{2}}P = (QA^{-\frac{1}{2}}P)^T(QA^{-\frac{1}{2}}P),$$

and so $\| G \| = \| QA^{-\frac{1}{2}}P \|^2$. Since the 2-norm of a matrix is equal to its largest singular value, the minimax characterization of singular values implies

$$\| QA^{-\frac{1}{2}}P \| = \max\left\{\frac{x^TQA^{-\frac{1}{2}}Py}{\| x \| \cdot \| y \|} : x, y \neq 0\right\}.$$

The maximum is clearly attained only if $x \in \text{range}(Q)$, i.e., $x = A^{-\frac{1}{2}}w$ for some $w \in \text{range}(B)$, and if $y \in \text{range}(P)$. So

$$
\begin{aligned}
\| QA^{-\frac{1}{2}}P \| &= \max\left\{\frac{w^TA^{-1}y}{\| A^{-\frac{1}{2}}x \| \cdot \| y \|} : y \in \text{null}(B^T); w \in \text{range}(B); y, w \neq 0\right\} \\
&= \max\left\{\frac{w^TA^{-1}y}{\| A^{-\frac{1}{2}}x \| \cdot \| A^{-\frac{1}{2}}y \|} \cdot \frac{\| A^{-\frac{1}{2}}y \|}{\| y \|} : y \in \text{null}(B^T); w \in \text{range}(B); y, w \neq 0\right\} \\
&\leq \cos\theta \cdot \max\left\{\frac{\| A^{-\frac{1}{2}}y \|}{\| y \|} : y \in \text{null}(B^T); y \neq 0\right\} \\
&= \cos\theta \cdot \| A^{-\frac{1}{2}}|_{\text{null}(B^T)} \|
\end{aligned}
$$

Then

$$\| G \| = \| QA^{-\frac{1}{2}}P \|^2 \leq \cos^2\theta \cdot \| A^{-1}|_{\text{null}(B^T)} \| \leq \cos^2\theta \cdot \lambda_{\min}^{-1}(A),$$

proving (c). Finally, from (b) $G = (MU^TA^{-1}P)^T(MU^TA^{-1}P)$, where $M = (U^TA^{-1}U)^{-\frac{1}{2}}$ is nonsingular. Then

$$
\begin{aligned}
G = 0 \quad &\text{iff} \quad PA^{-1}UM = 0 \\
&\text{iff} \quad PA^{-1}U = 0 \\
&\text{iff} \quad A^{-1} \text{ maps range}(B) \text{ into range}(B) \\
&\text{iff} \quad A \text{ maps range}(B) \text{ into range}(B),
\end{aligned}
$$

proving (d). □

Note that the angle θ is the angle between null(B^T) and range(B) in the A^{-1}-norm, and can be considered as a measure of how much A^{-1} rotates one subspace into the other. Furthermore, since $\cos\theta \leq 1$, an upper bound that depends *only* on A is given by the Theorem.

Some results from the last Theorem apply particularly to CFD problems. If we assume that A is a discrete version of $\left(\frac{R}{\delta t}I - \nabla^2\right)$ in (6), Theorem 5.2 has an immediate consequence: $\| PA^{-1}P - (PAP)^\dagger \| \leq \cos^2\theta \cdot \lambda_{\min}^{-1}(A) \leq R/\delta t$. For large Reynold's number and/or small time step δt, this implies that the approximation is good, providing an explanation for the effectivness of the projection–based algorithms used in CFD. For practical problems with boundary conditions and complicated geometries, the testing results of the next section show that this heuristic argument still holds. Furthermore, if $B \approx \nabla$, then the vector calculus rules $\nabla \cdot (\nabla \times v) = 0$ and $\triangle = \nabla\nabla\cdot - \nabla\times\nabla\times$ imply that $PAP = AP$ (see Semeraro [23] for a discussion of this). This in turn says that range(P) is an invariant subspace of A, which in turn implies that range(B) is also an invariant subspace. From (d) in Theorem 5.2, this says that $PA^{-1}P = (PAP)^\dagger$, regardless of the value of α. In practice the vector calculus rules do not apply to the discretized problem, but these heuristic arguments do suggest that for CFD problems if M is a good preconditioner for A, then PMP is a good preconditioner for PAP. The numerical tests verify this.

6. Testing Results. The numerical tests use version 4.0.BETA.2 of Matlab, run on a Sun 4/490 SparcServer with SunOS 4.1.1 and 32Mb of memory. Three algorithms are tested; Uzawa (UZA) is the CG method applied to (15), preconditioned with the complete Cholesky factorization $R^TR = B^TD^{-1}D$ where D is the diagonal of A. The projection method (PAP) applies CG to (10), and the preconditioned projection method (PC-PAP) applies CG to $PA^{-1}PAPu = PA^{-1}Pf$, where matrix–vector products with A are provided by an inner CG iteration as in UZA. Testing was performed to:

a. show that the projection method works, and in particular that it can be safely implemented using only one projection per iteration as indicated in Theorem 5.1,
b. compare the behaviour of the CG algorithm applied to systems with coefficient matrices of A, PAP, and $PA^{-1}PAP$,
c. compare UZA, PAP, and PC-PAP on problems extracted from *N3S* in terms of the amount of storage needed, the number and type of operations required, and the accuracy of solution attainable.
d. determine the effect of changing the inner tolerance used in UZA and PC-PAP for the CG algorithm applied to systems with coefficient matrix A.

Characteristics of the four test problems used are given in Table 2. For the symmetric matrices A and B^TB, the number of nonzeros is only for the lower triangular part. Three possible orderings - natural (NAT), column minimum degree (CMD), and reverse Cuthill-McKee (RCM) - of the matrix B^TB were examined for each problem, and the one giving the fewest nonzeros in the Cholesky factor R was chosen. For details about

Name	Problem Type	Dimension of B	Number of Nonzeros			
			A	B	B^TB	R
LAM338	2D Driven Cavity	338 × 49	2162	623	348	537 (RCM)
CONVEC8	2D Natural Convection	3362 × 484	253448	13997	5134	16230 (CMD)
STRAT8	2D Salinity Mixing	16640 × 2205	49424	66192	21701	95580 (NAT)
NIMBUS30	3D Hot Water Injection	23871 × 1325	206598	181972	33791	134900 (RCM)

TABLE 2
Test Problem Characteristics

the reordering methods used, see [11]. LAM338 is a 2D driven cavity problem on the unit square, provided by Ulrike Meier [19]. This small problem was considered only to provide an example for which complete spectral decompositions could be reasonably made. The remaining three problems are extracted from the code *N3S*. CONVEC8 and STRAT8 are taken from the eigth time step for a natural heat-driven convection problem and a saline mixing problem, respectively, both in 2D. NIMBUS30 is from the thirtieth time step and models the flow of hot water into a larger tube carrying cold water in 3D. Similar results hold for systems taken from other time steps.

Each algorithm is initialized with a vector of all zeros. Although in practice the solution from the previous time step is normally used, this was not found to significantly reduce the number of iterations required to reduce the relative residual by a fixed amount.

Each problem was preconditioned with the diagonal of A by explicitly scaling the matrices A and B. Each algorithm stops if either the maximum allowed number of iterations was reached, or if $\| r_k \| \leq \epsilon \| r_0 \|$, where r_k is the residual on the k^{th} iteration. The term "relative residual" is used here for $\| r_k \| / \| r_0 \|$, while "absolute residual" is used for $\| r_k \|$. Plots that start with the initial residual equal to 1 use the relative residual, while the others show the absolute residual. Note that each algorithm has a different "residual" that occurs naturally; for UZA, $r = (B^TB)^{-1}B^TA^{-1}(f - Bp)$, for PAP, $r = f - Au - Bp$, and for PC-PAP, $r = PA^{-1}P(f - Au - Bp)$. Plots showing the relative residual are based on the natural residual for the corresponding algorithm, while those showing absolute residuals are based on the residual of the original equation (1).

Unless otherwise noted, a maximum of 40 iterations were allowed, and $\epsilon = 10^{-6}$. For UZA and PC-PAP, inner stopping tests are needed for solving systems with the matrix A. A maximum of 80 iterations is allowed for UZA and $\epsilon_{in} = 10^{-7}$. The preconditioned projection method was allowed a maximum of 5 iterations for the inner CG iteration needed to solve systems with coefficient matrix A. This number was chosen because more iterations simply increase the number of flops per outer iteration without improving performance. Note that the accuracy of the inner CG solve in UZA is crucial to the accuracy achievable in the outer algorithm, while in PC-PAP it is used to define a preconditioner and so does not limit the accuracy of the outer iterations.

After reordering for sparsity, the matrix $B^T B$ was explicitly formed and then factored as $R^T R = B^T B$ with R upper triangular. The projections needed by PAP and PC-PAP are computed as $Pv = v - BR^{-1}R^{-T}B^T v$, that is, by means of the *seminormal* equations.

6.1. Spectra of A and PAP. Figure 1 shows the spectra of A, PAP, and the preconditioned projection system $PA^{-1}PAP$ for problem LAM338, with all three normalized to have the same largest eigenvalue. The $q = 49$ eigenvalues of PAP and $PA^{-1}PAP$ that are exactly equal to zero have been omitted. Note that the smallest eigenvalues of A and PAP are approximately equal, so $\kappa(A) \approx \kappa(PAP)$. However, PAP has its eigenvalue clusters smeared out slightly compared to A (for example, note the two eigenvalues of PAP that are approximately equal to 0.5, without corresponding ones for A). This is why even though $\kappa(PAP) \approx \kappa(A)$ and the maximal number of distinct eigenvalues of PAP is $n - q$ versus n for A, CG applied to $PAPu = Pf$ still requires a few more iterations than when applied to A. The preconditioned system has 83% of its eigenvalues equal to 1 ± 10^{-7}, showing that preconditioning with $PA^{-1}P$ does produces a better spectrum for CG.

A complete spectral analysis for the three problems from *N3S* is impractical because of the size of those problems. However, the spectra are of interest primarily because of the effect on the convergence of the CG algorithm. Figures 2–5 show the residual versus iteration number when CG is applied to systems $Au = f$ and $PAPu = Pf$ for the four test problems. The convergence behaviour for the two systems is identical for LAM338 (although the initial residuals differ), and similar for the other three problems. This shows that the projection method can solve the entire linear Stokes problem with only slightly more iterations than required by a single inner iteration of Uzawa.

6.2. Direct versus Iterative Solution of Uzawa Preconditioner. The Uzawa scheme compared here to the projection methods preconditions its system with $B^T B$ (since B has been scaled by the diagonal of A, this is the same as $B^T D^{-1} B$ where D is the diagonal of A). The code *N3S* solves the systems with coefficient matrix $B^T B$ by an incomplete Cholesky preconditioned CG algorithm (ICCG), while the tests here are performed with the complete Cholesky factorization of $B^T B$, that is, a direct solution. This comes at an additional cost in memory as shown in Table 2. The four test problems require from 1.5 to 4.4 times the memory that is needed to store the incomplete Cholesky factor of $B^T B$. Figures 6, 7, and 8 show the logarithm of the residual versus the number of floating point operations for the Uzawa algorithm; solid lines are with a direct solve for $B^T B$, while the dotted lines are for ICCG solves with tolerances set to 10^{-4}, 10^{-5}, and 10^{-6}. Each run kept the tolerance for the inner CG algorithm applied to systems with coefficient matrix A fixed at 10^{-7}. For a given accuracy, using a direct solve for $B^T B$ provides the solution in fewer flops for STRAT8 and NIMBUS30, respectively. However, for CONVEC8 this is only true for the first 6 iterations, after which the residual

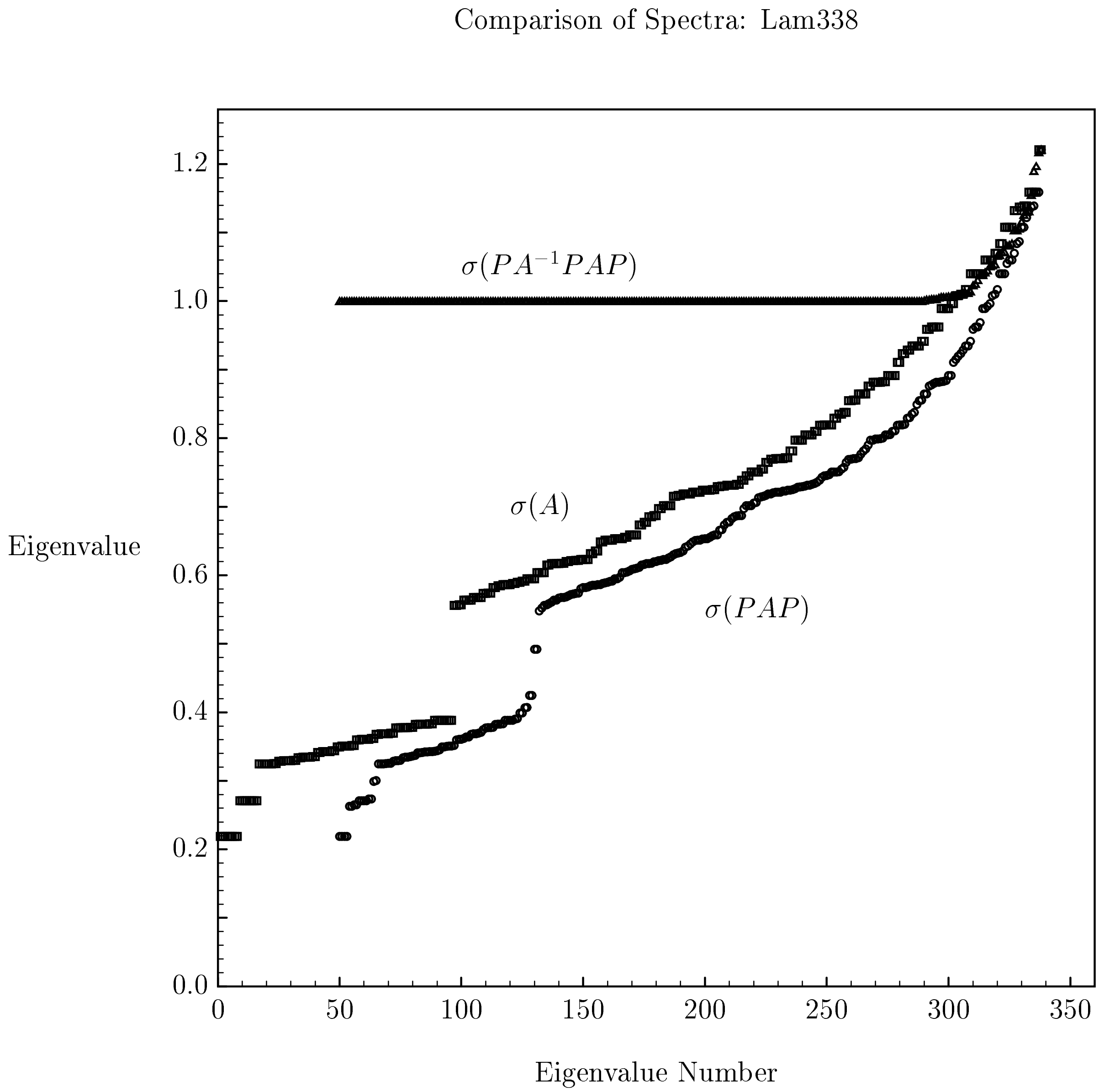

FIG. 1. *Comparison of Spectra for* LAM338*; circle: PAP, box: A, triangle: $PA^{-1}PAP$*

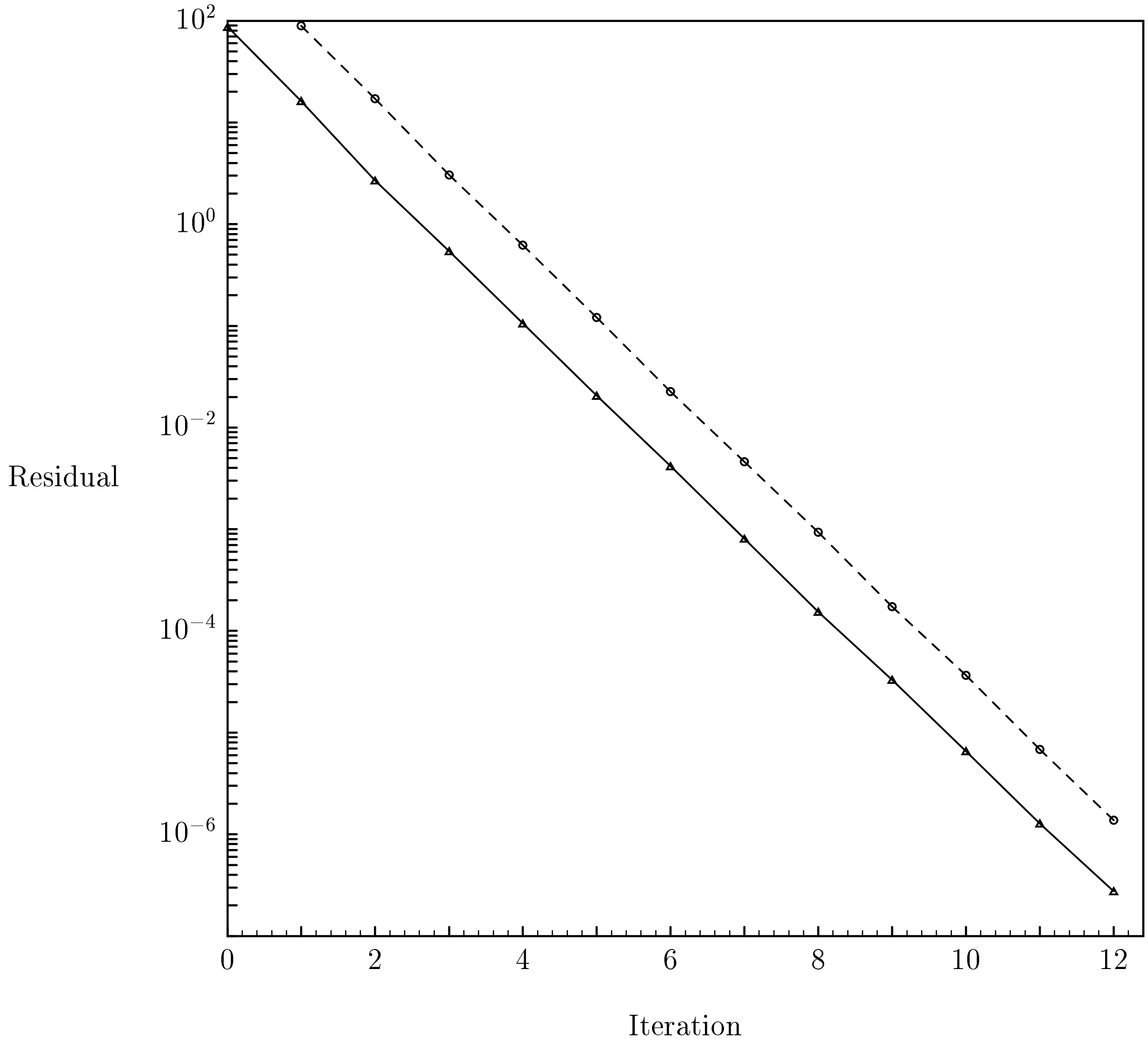

Fig. 2. *CG Applied to* Lam338*; circle:* $Au = f$*, triangle:* $PAPu = Pf$

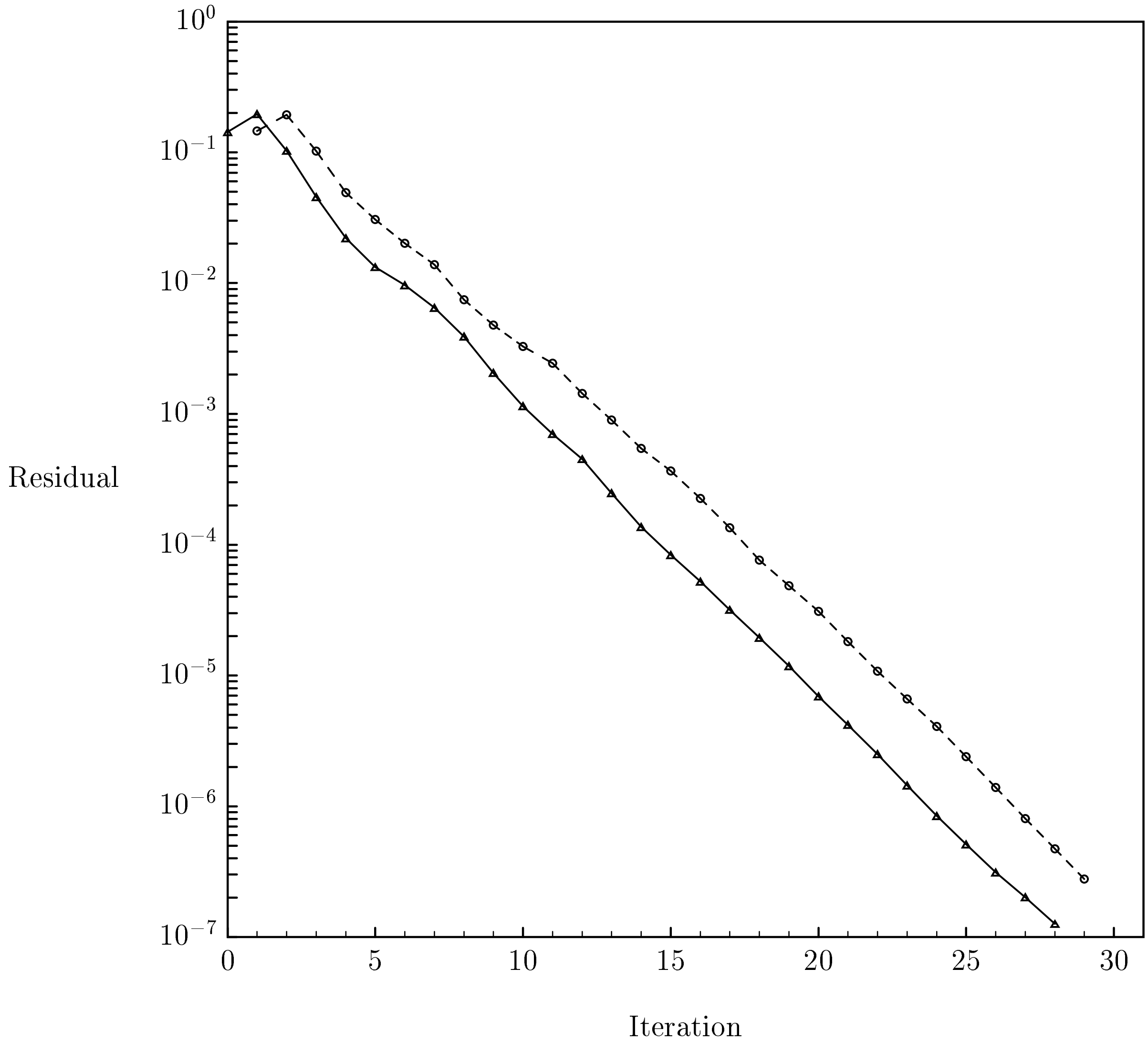

FIG. 3. *CG Applied to* CONVEC8; *circle:* $Au = f$, *triangle:* $PAPu = Pf$

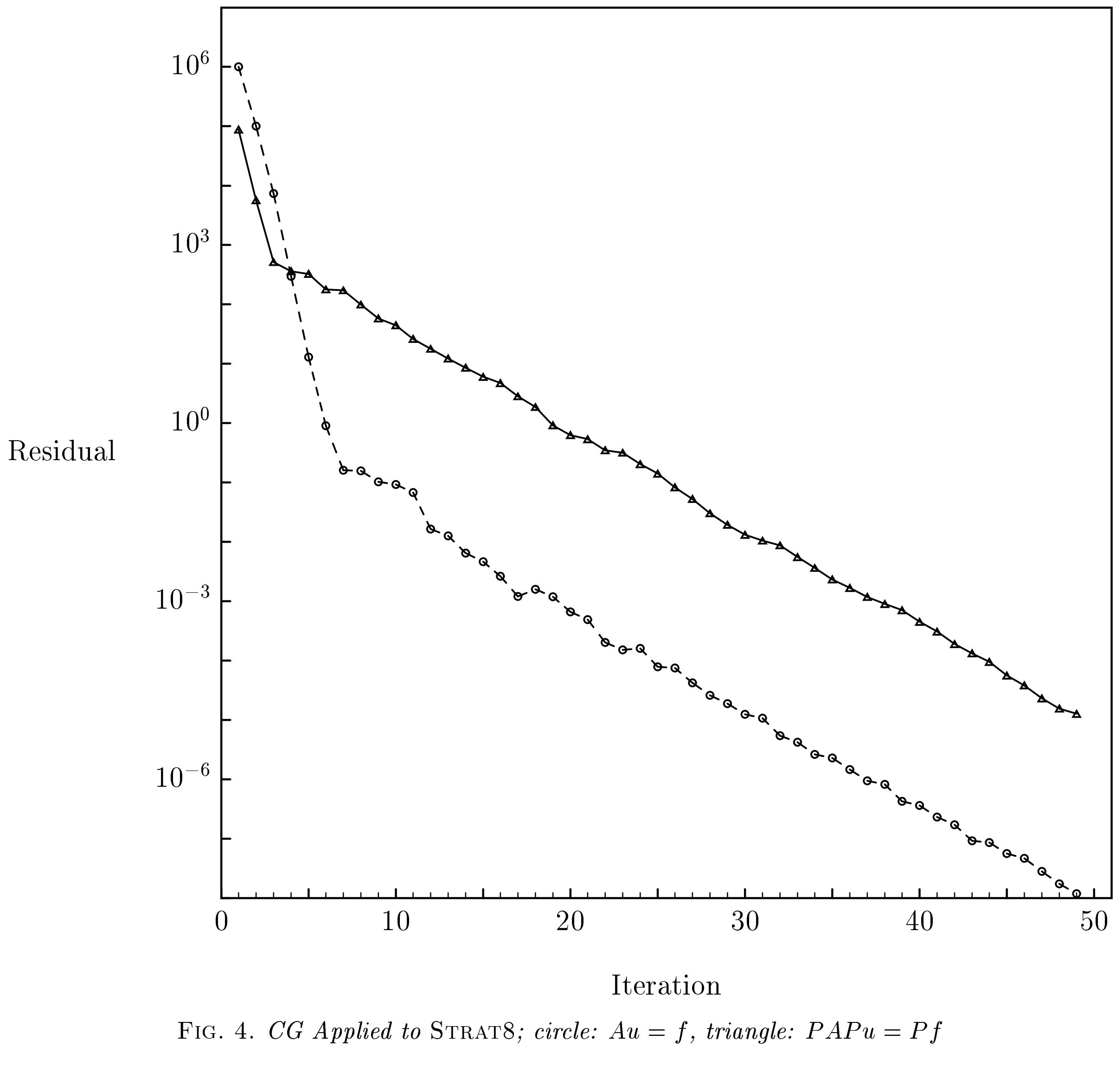

FIG. 4. *CG Applied to* STRAT8*; circle: $Au = f$, triangle: $PAPu = Pf$*

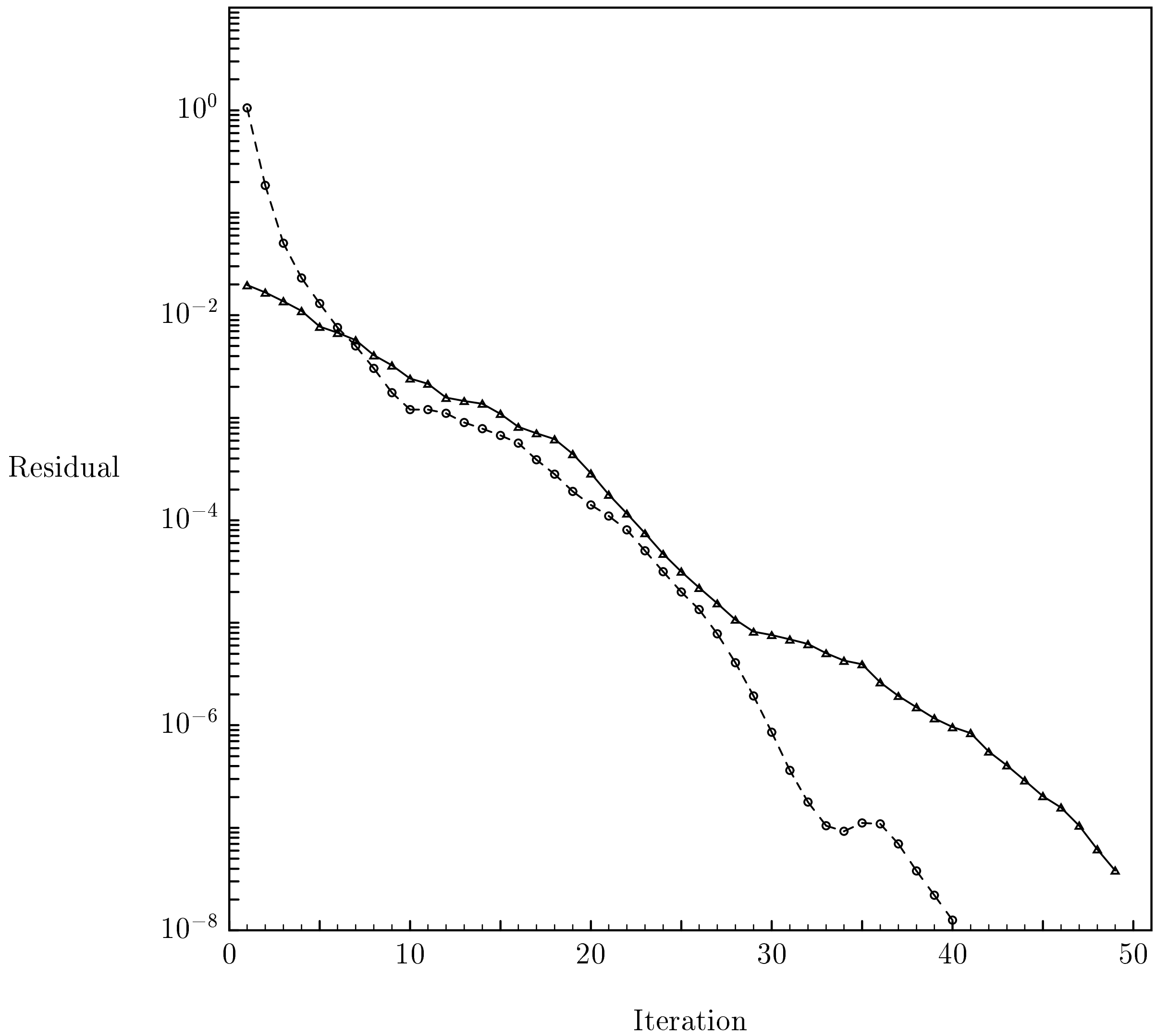

FIG. 5. *CG Applied to* NIMBUS30*; circle:* $Au = f$*, triangle:* $PAPu = Pf$

	Operation	Uzawa Per Iter	Uzawa Total	Projection Per Iter	Projection Total	PC-Projection Per Iter	PC-Projection Total
% Flops	R^TR-solve	2.4	2.2	18.2	19.3	8.5	9.0
	A-mult	63.9	63.9	14.2	14.3	52.4	45.5
	B-mult	0.7	0.6	5.5	5.8	2.5	2.7
	B^T-mult	0.7	0.7	5.2	5.5	2.4	2.5
	Other ops	32.3	32.6	57.0	55.1	34.2	40.3
Total Flops		4.07E6	4.48E7	5.68E5	1.63E7	1.07E6	1.28E7
% Time	R^TR-solve	2.0	1.7	14.0	14.9	6.6	6.5
	A-mult	77.4	74.7	18.5	18.6	63.8	51.4
	B-mult	0.5	0.4	7.1	7.5	1.7	1.9
	B^T-mult	0.9	0.8	4.9	5.3	3.1	3.0
	Other ops	19.3	22.4	55.6	53.7	24.7	37.3
Total Seconds		14.6	170.9	2.9	84.1	4.6	51.0
Resid Norm	$Au + Bp - f$	6.42E-10		4.89E-9		3.67E-9	
	B^Tu	2.00E-10		2.19E-18		1.81E-19	

TABLE 3

Comparison of Uzawa, Projection and Preconditioned Projection Methods on CONVEC8

levels off but the iterative method continues to decrease the residual norm. Similar plots hold for other tolerances on the ICCG solve, showing that if enough memory is available to store the Cholesky factor, then a direct solve can be a more efficient and accurate method.

It should be noted that since B^TB is independent of the time step in *N3S*, much effort can reasonably be spent in finding an ordering of the columns of B that will minimize fill-in for the Cholesky factorization of B^TB. The cost of this effort is amortized over many time steps, and is minor in comparison. Even for the single systems tested here, the reordering and factorization of B^TB never required more than 10% of the total program flops, although three different orderings are tested to find the one with smallest fill-in.

6.3. Comparison of Uzawa and Projection Method. Tables 3, 4, and 5 summarize the results for CONVEC8, STRAT8, and NIMBUS30, respectively. The relative proportion of time spent in different phases is also shown in the Tables. Several points stand out:

- The accuracy achieved for the first block equation $Au+Bp-f$ is comparable for all three methods. However, the projection methods give significantly better results for the divergence, that is, the norm of B^Tu. Those residual norms are from 7 to 10 orders of magnitude better for the projected method, and 9 to 12 orders better for the preconditioned projected method, than for the Uzawa algorithm.
- The improved accuracy comes at a reduced cost in flops. The projected method requires only 36, 21, and 54% as many flops as Uzawa, while the preconditioned scheme requires only 29, 21, and 43% as many.

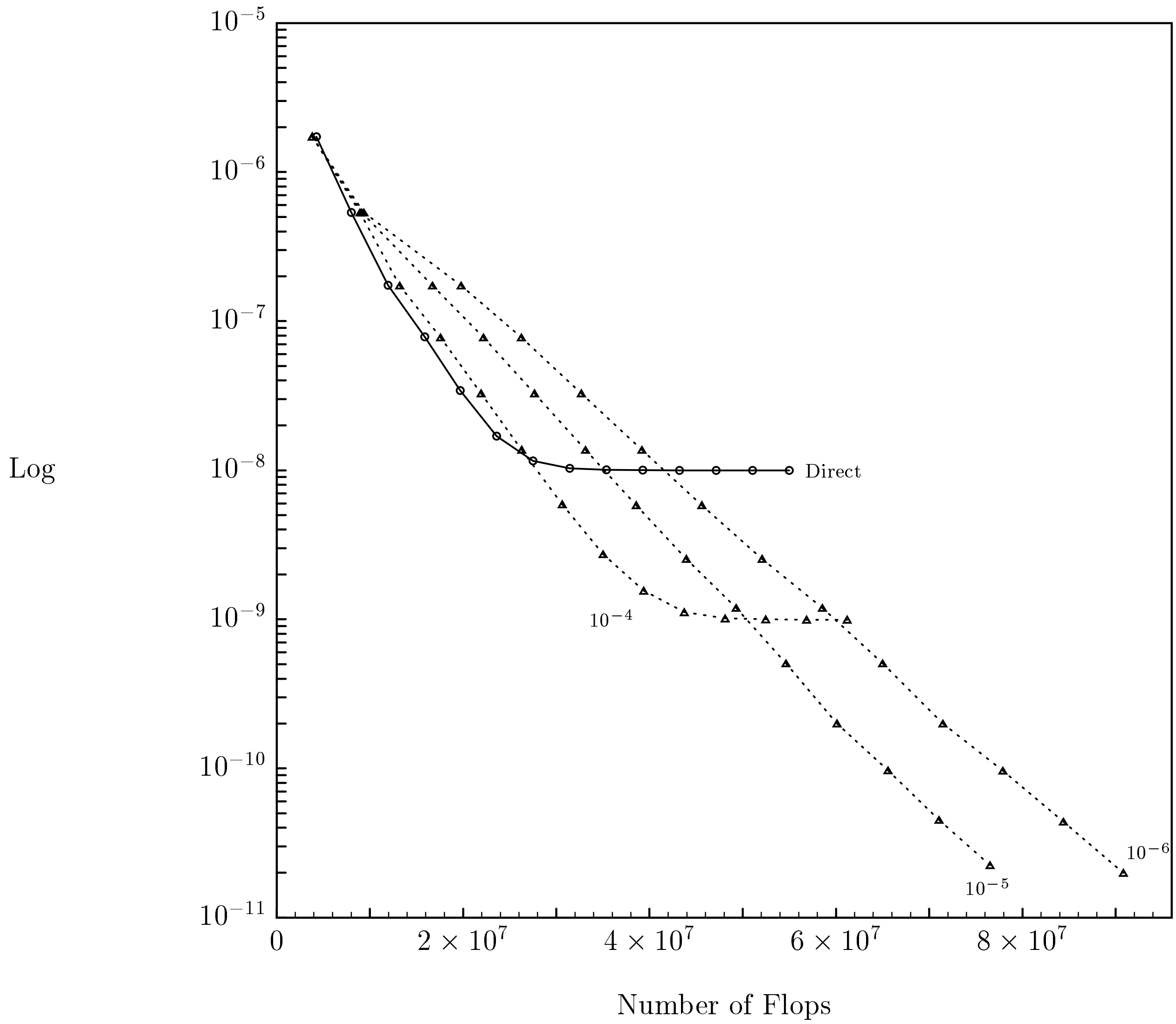

FIG. 6. *Comparison of Uzawa with Direct and Iterative Solution of* $B^T B$ *for* CONVEC8*; circle: direct, triangle: iterative with tolerances* 10^{-4}, 10^{-5}, *and* 10^{-6}

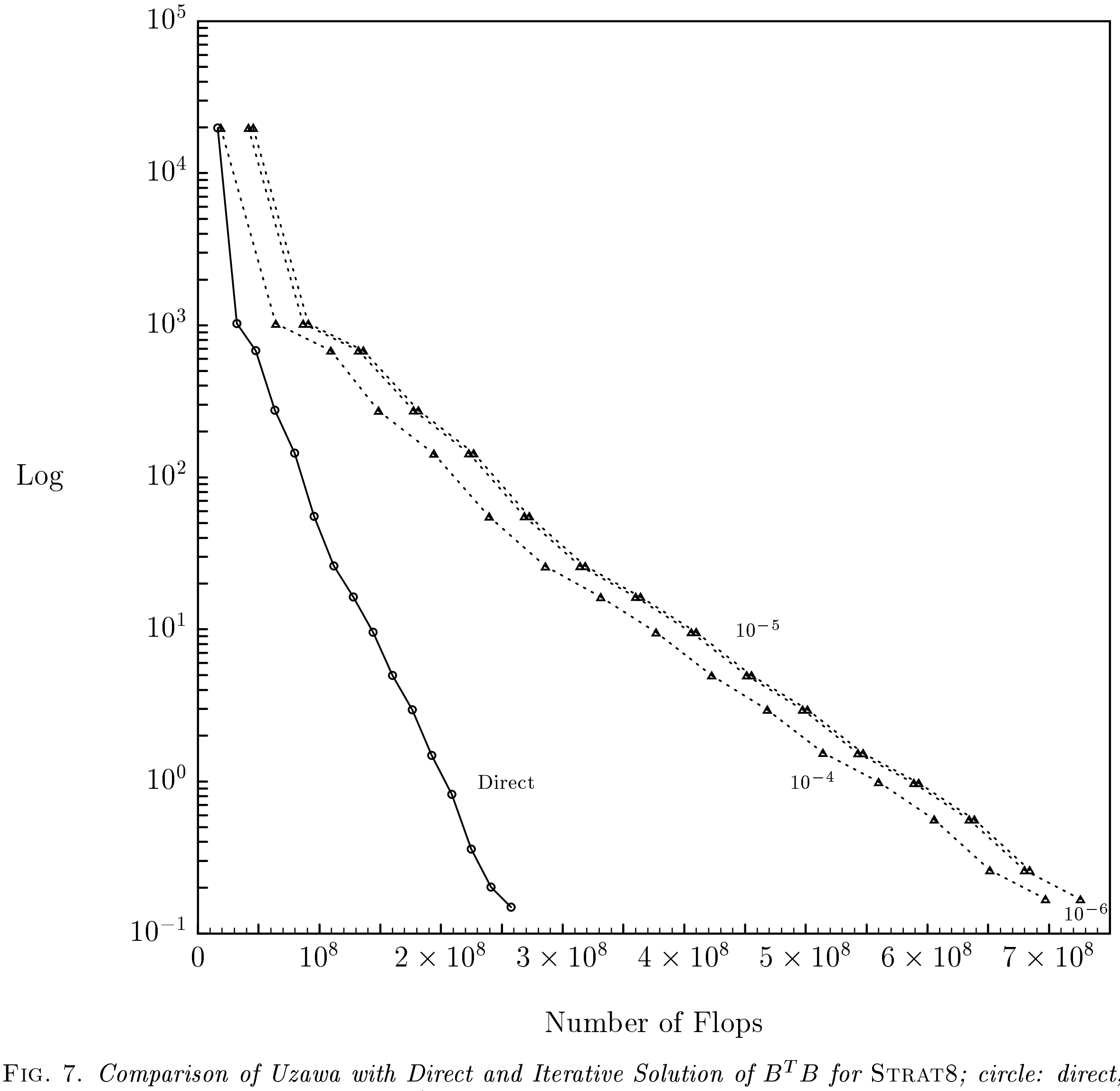

Fig. 7. *Comparison of Uzawa with Direct and Iterative Solution of* B^TB *for* Strat8*; circle: direct, triangle: iterative with tolerances* 10^{-4}, 10^{-5}, *and* 10^{-6}

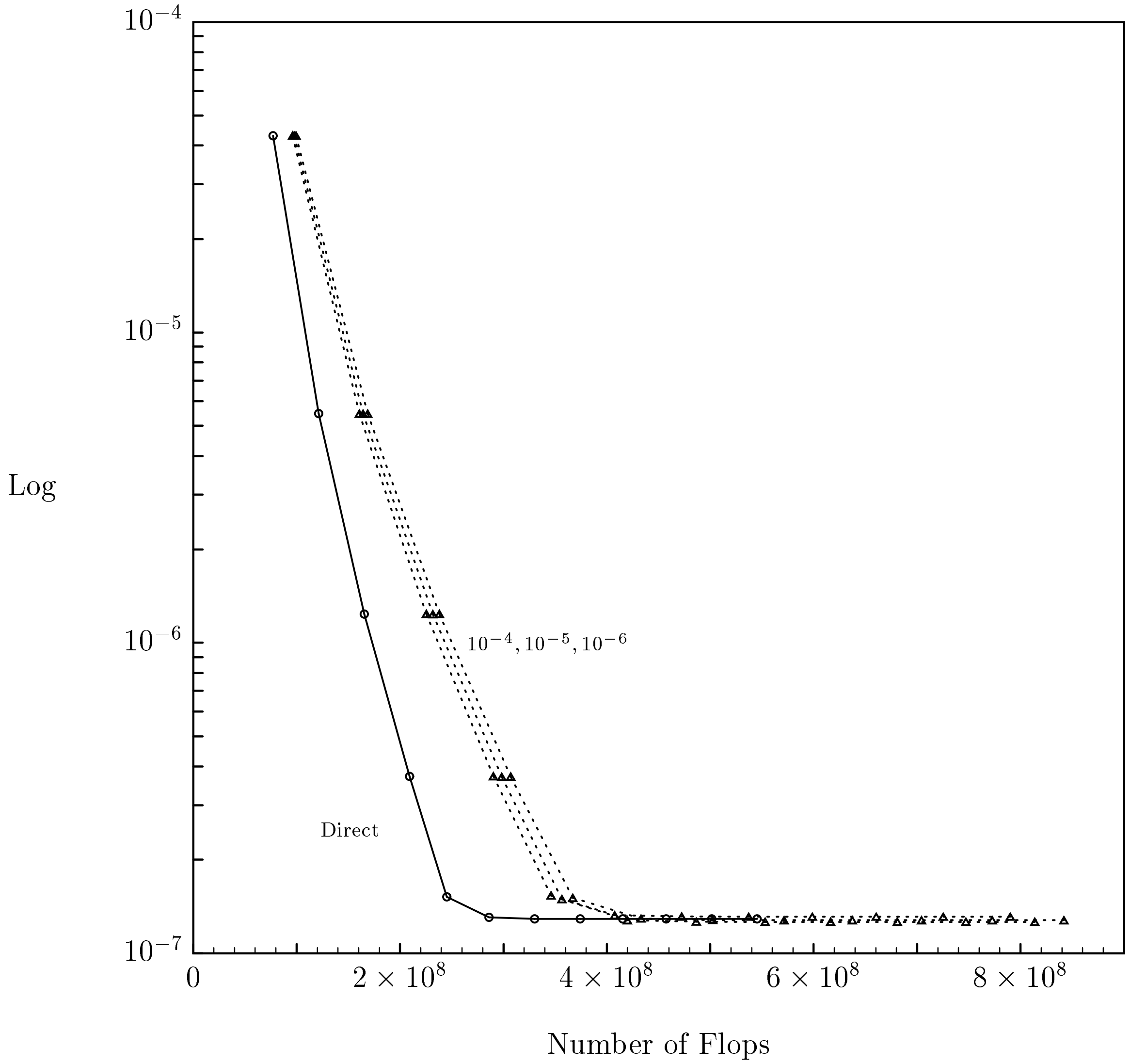

FIG. 8. *Comparison of Uzawa with Direct and Iterative Solution of* $B^T B$ *for* NIMBUS30*; circle: direct, triangle: iterative with tolerances* 10^{-4}, 10^{-5}, *and* 10^{-6}

	Operation	Uzawa		Projection		PC-Projection	
		Per Iter	Total	Per Iter	Total	Per Iter	Total
% Flops	$R^T R$-solve	2.4	2.4	19.6	21.7	11.1	11.7
	A-mult	48.0	47.9	9.1	9.2	35.2	30.0
	B-mult	0.8	0.8	6.7	7.5	3.8	4.1
	B^T-mult	0.8	0.9	6.6	7.3	3.7	3.9
	Other ops	48.0	48.1	58.1	54.3	46.2	50.3
Total Flops		1.66E7	1.99E8	2.19E6	4.34E7	4.28E6	4.26E7
% Time	$R^T R$-solve	1.8	1.8	21.4	23.1	8.9	8.6
	A-mult	69.5	68.9	17.1	17.3	52.3	38.3
	B-mult	0.6	0.5	8.1	9.1	2.7	2.9
	B^T-mult	1.0	1.0	5.3	5.7	4.5	4.2
	Other ops	27.1	27.7	48.2	44.8	31.6	46.0
Total Seconds		59.6	733.7	12.5	250.0	17.3	185.6
Resid Norm	$Au + Bp - f$	1.39E-2		6.05E-2		1.36E-2	
	$B^T u$	1.48		6.67E-10		7.90E-12	

TABLE 4
Comparison of Uzawa, Projection and Preconditioned Projection Methods on STRAT8

	Operation	Uzawa		Projection		PC-Projection	
		Per Iter	Total	Per Iter	Total	Per Iter	Total
% Flops	$R^T R$-solve	1.2	1.1	11.1	11.7	5.7	6.2
	A-mult	71.7	71.7	16.5	16.7	57.6	47.5
	B-mult	0.8	0.7	7.7	8.1	3.6	4.0
	B^T-mult	0.8	0.9	7.5	7.9	3.9	4.1
	Other ops	25.4	25.6	57.1	55.6	29.2	38.2
Total Flops		4.38E7	3.40E8	5.04E6	1.85E8	1.06E7	1.47E8
% Time	$R^T R$-solve	0.9	0.9	8.6	8.9	4.2	4.8
	A-mult	82.2	79.1	21.6	23.5	67.0	46.8
	B-mult	0.5	0.4	7.2	7.5	2.4	2.5
	B^T-mult	1.0	1.5	6.8	8.2	5.0	6.7
	Other ops	15.4	18.0	55.8	51.9	21.5	39.2
Total Seconds		159.3	1354	23	879.7	42.6	669.4
Resid Norm	$Au + Bp - f$	5.55E-9		1.90E-9		9.84E-10	
	$B^T u$	5.25E-9		7.69E-16		9.22E-21	

TABLE 5
Comparison of Uzawa, Projection and Preconditioned Projection Methods on NIMBUS30

- The relative amount of flops and time spent in sparse matrix–vector multiplications with A is significantly reduced for the projection method, with the emphasis shifting to triangular solves and other operations, primarily dense saxpy and inner product operations. However, the preconditioned method tends to shift the emphasis back towards multiplications with A.

Figures 9, 10 and 11 show the relative residual norm versus iteration number for the test problems. For these Figures the projection methods were allowed to continue iterating until the relative residual was less than 10^{-6}. The performance of the projected methods seems poor when the residual is plotted versus iteration number, but when the relative residual is plotted versus flops as in Figures 12, 13 and 14, it is clear that the projected methods can completely solve the Stokes problem with fewer flops than needed to initialize the Uzawa algorithm. Similarly, the preconditioned projection method appears to outperform the projection method when the relative residual is plotted versus iteration number, but in fact the two methods have almost identical convergence behaviour when plotted against flops.

Because an inner tolerance must be set for the Uzawa scheme, it is possible that the projection methods appear superior because it has been set either too small or large; Figure 15 plots the absolute residual $(\| Au + Bp - f \|^2 + \| B^T u \|^2)^{-1/2}$ versus flops for PAP and for UZA on problem CONVEC8, with the inner tolerance on the Uzawa algorithm varied from 10^{-2} to 10^{-12}. As the inner tolerance gets smaller, the initial absolute residual for UZA also decreases because the inner iteration implicitly keeps the equation $Au + Bp = f$ satisfied. A large error in the inner iteration causes a large residual for that equation. When the inner tolerance reaches 10^{-6}, however, the overall residual is then dominated by the residual for $B^T u = 0$, and further decreases in the inner tolerance do not improve the initial residual. This is why for a inner tolerance larger than 10^{-6}, the absolute residual is constant for all iterations in UZA. For any inner tolerance, however, the projection method provides as good or better of a solution with the same or fewer flops.

6.4. Comparison of the Projection Method With and Without Preconditioning. When drawn on the same scale of flops as UZA, PAP and PC-PAP appear to have the same performance (Figures 12 – 14). To better distinguish between these two approaches, Figures 16–18 plot the relative residual versus flops for the test problems as the inner tolerance for computing the preconditioner is varied. Except for NIMBUS30, PAP provides a better solution with fewer flops, and even for NIMBUS30 the performance is close to that of PC-PAP. However, as noted earlier PC-PAP spents proportionately more of its flops in sparse matrix–vector multplications and fewer in triangular solves with the Cholesky factors than PAP does. For a vector machine, PC-PAP may significantly outperform PAP and so should not be rejected on the basis of number of flops alone.

6.5. Error Growth in the Projection Method. Because PAP and PC-PAP apply CG to a singular system, one concern is that residual components in the null space of the coefficient matrix may grow rapidly, making the method numerically unstable

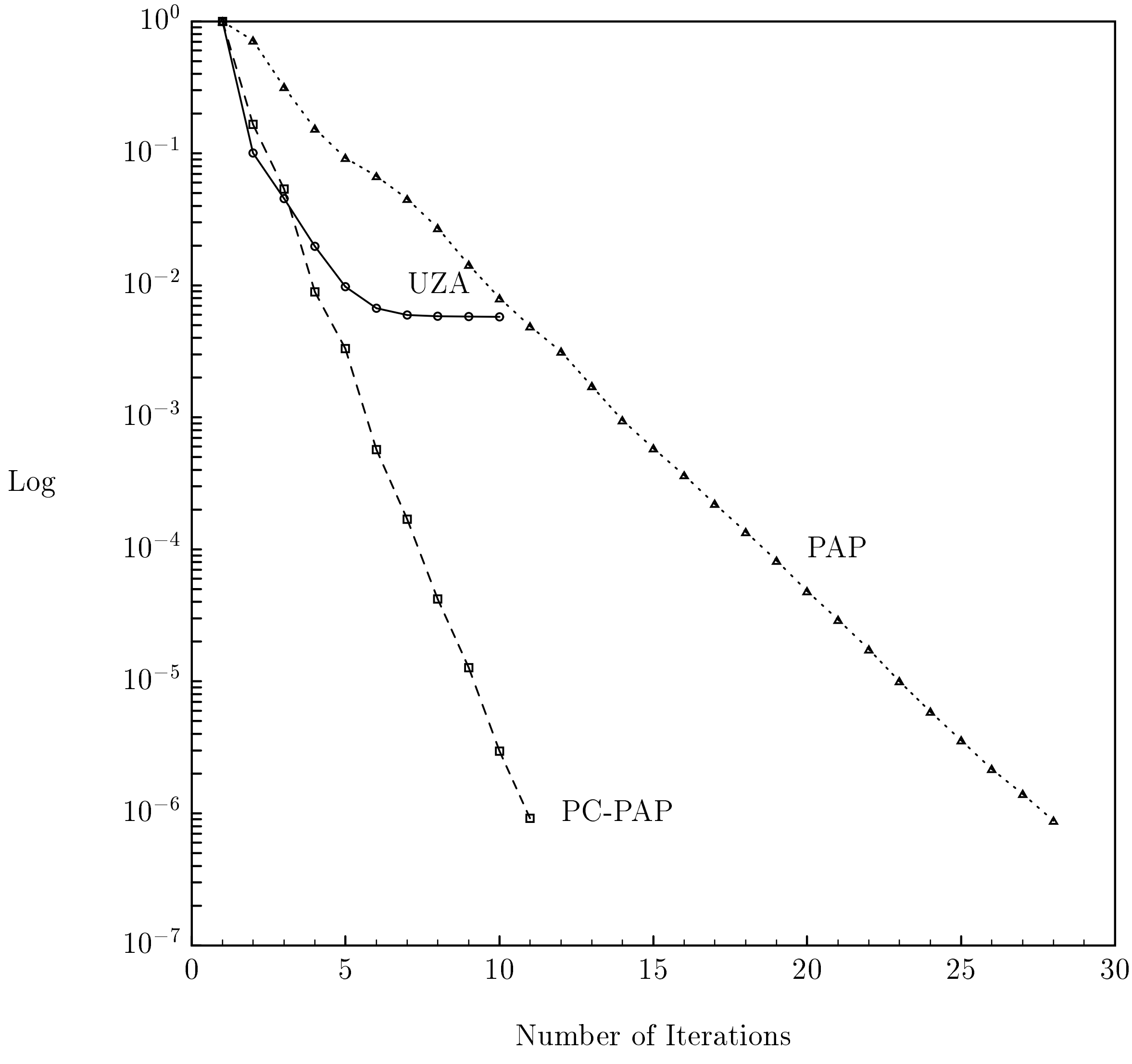

FIG. 9. *Relative Residual Norm vs. Iteration Number for* CONVEC8; *circle: Uzawa, triangle: PAP, box: PC-PAP*

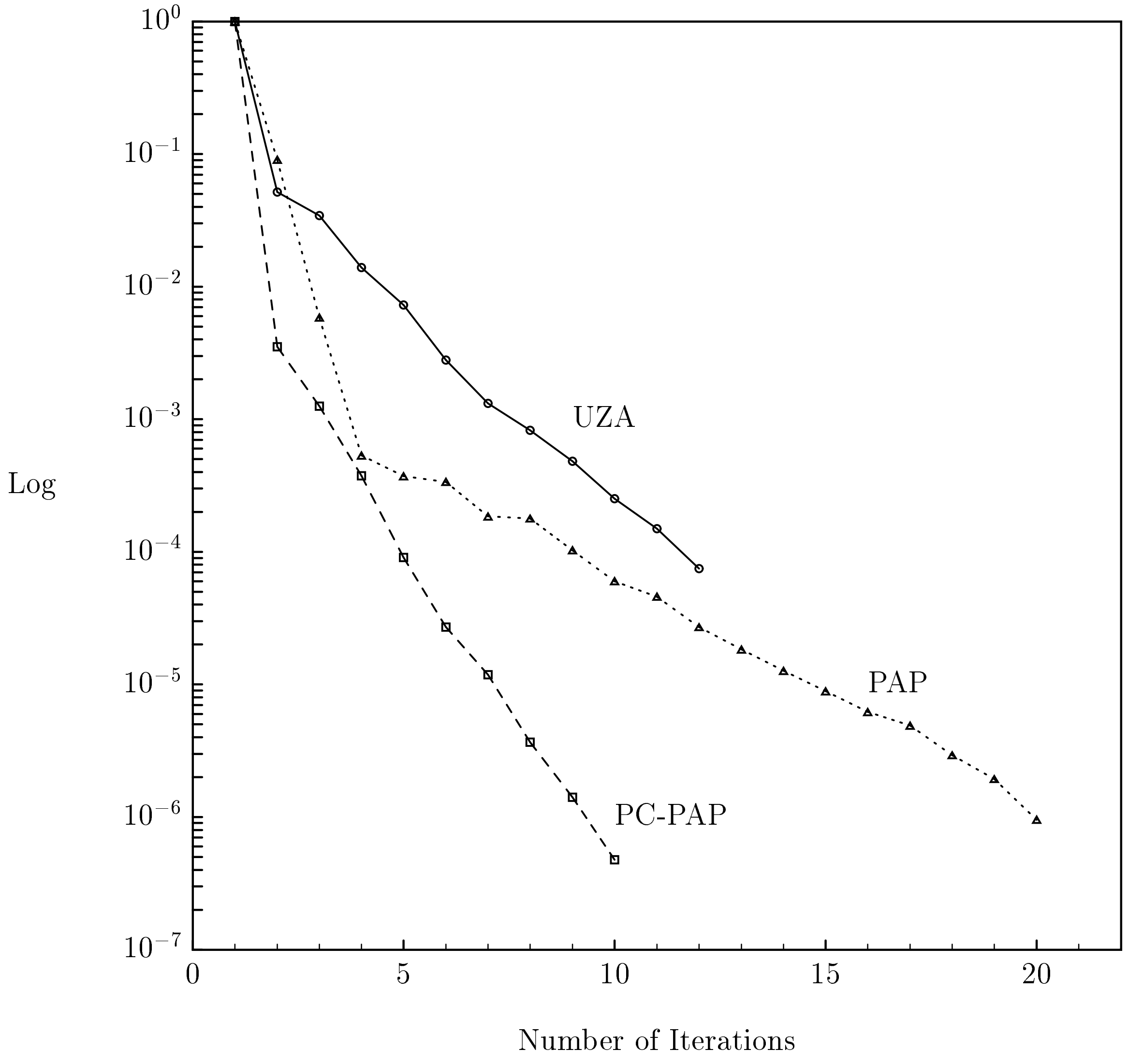

FIG. 10. *Relative Residual Norm vs. Iteration Number for* STRAT8*; circle: Uzawa, triangle: PAP, box: PC-PAP*

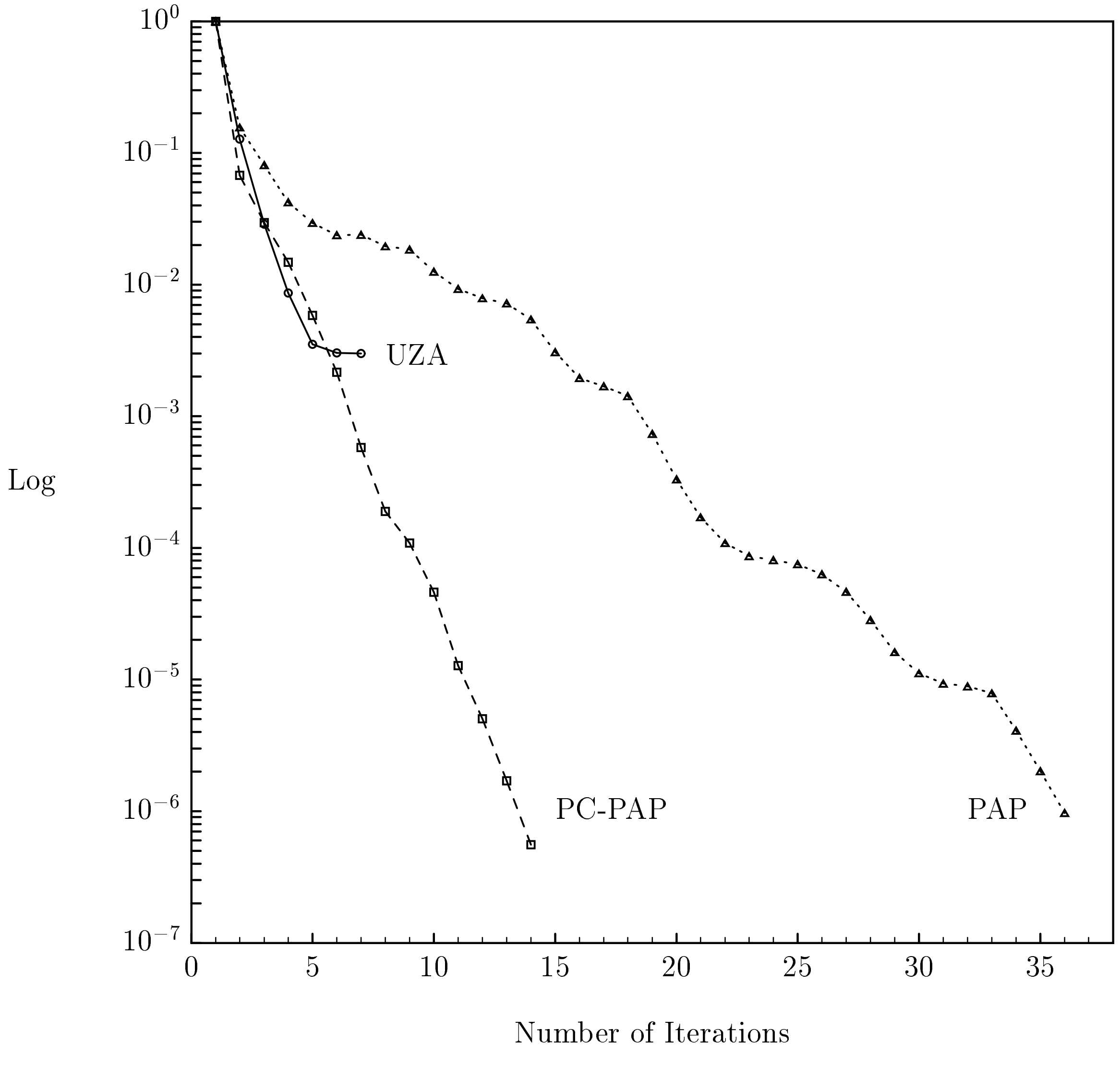

FIG. 11. *Relative Residual Norm vs. Iteration Number for* NIMBUS30*; circle: Uzawa, triangle: PAP, box: PC-PAP*

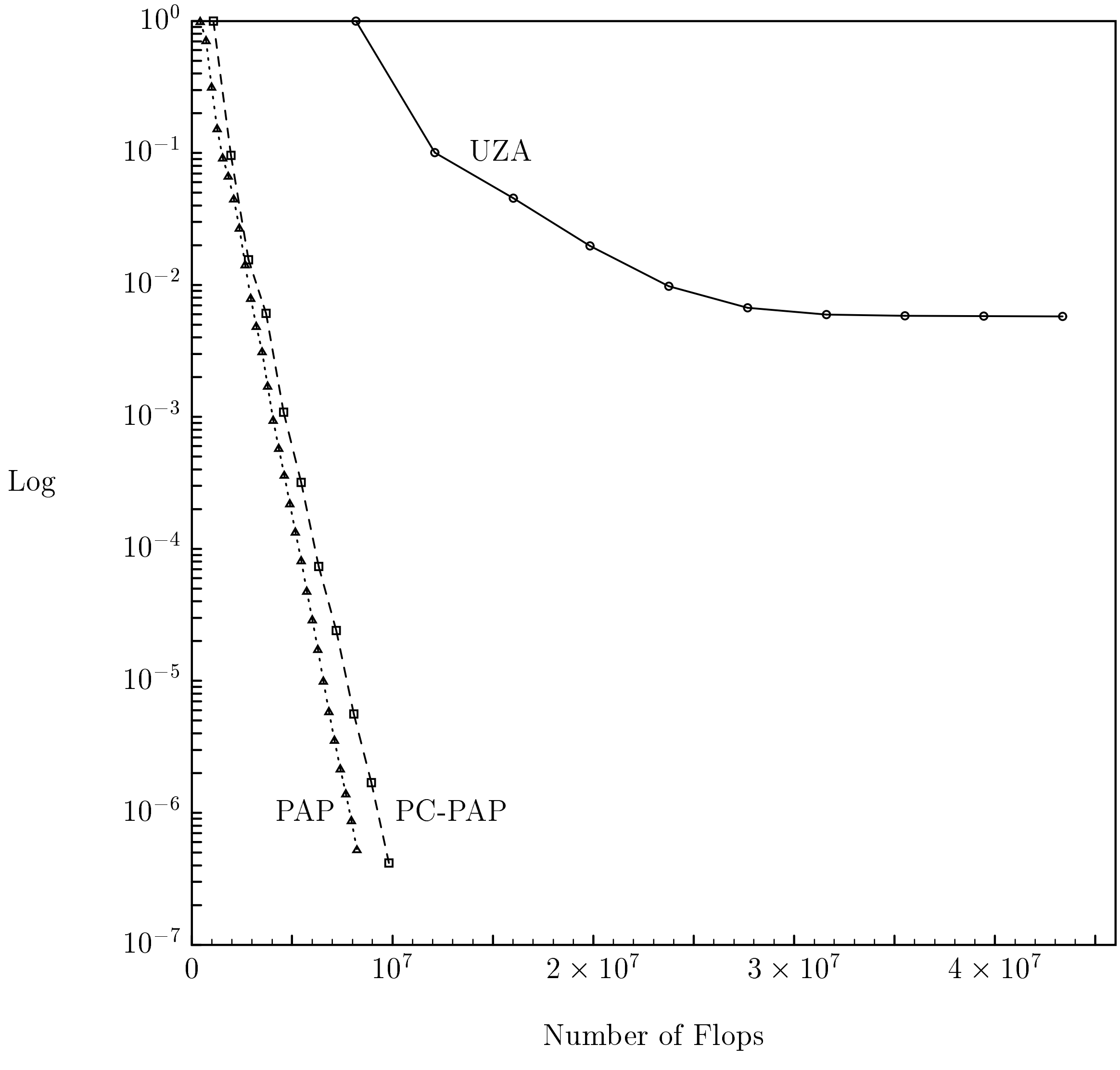

FIG. 12. *Relative Residual Norm vs. Flops for* CONVEC8*; circle: Uzawa, triangle: PAP, box: PC-PAP*

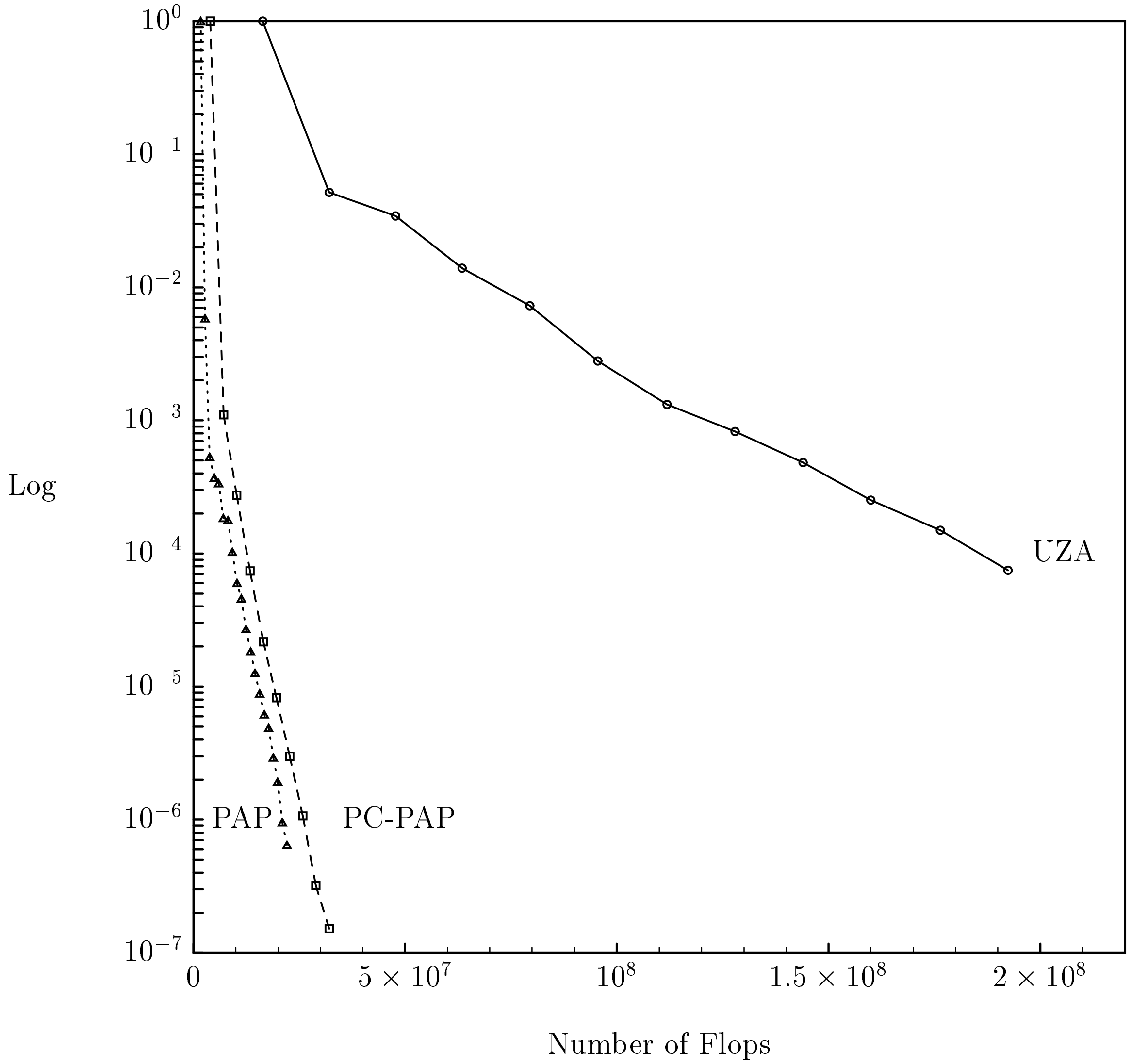

FIG. 13. *Relative Residual Norm vs. Flops for* STRAT8*; circle: Uzawa, triangle: PAP, box: PC-PAP*

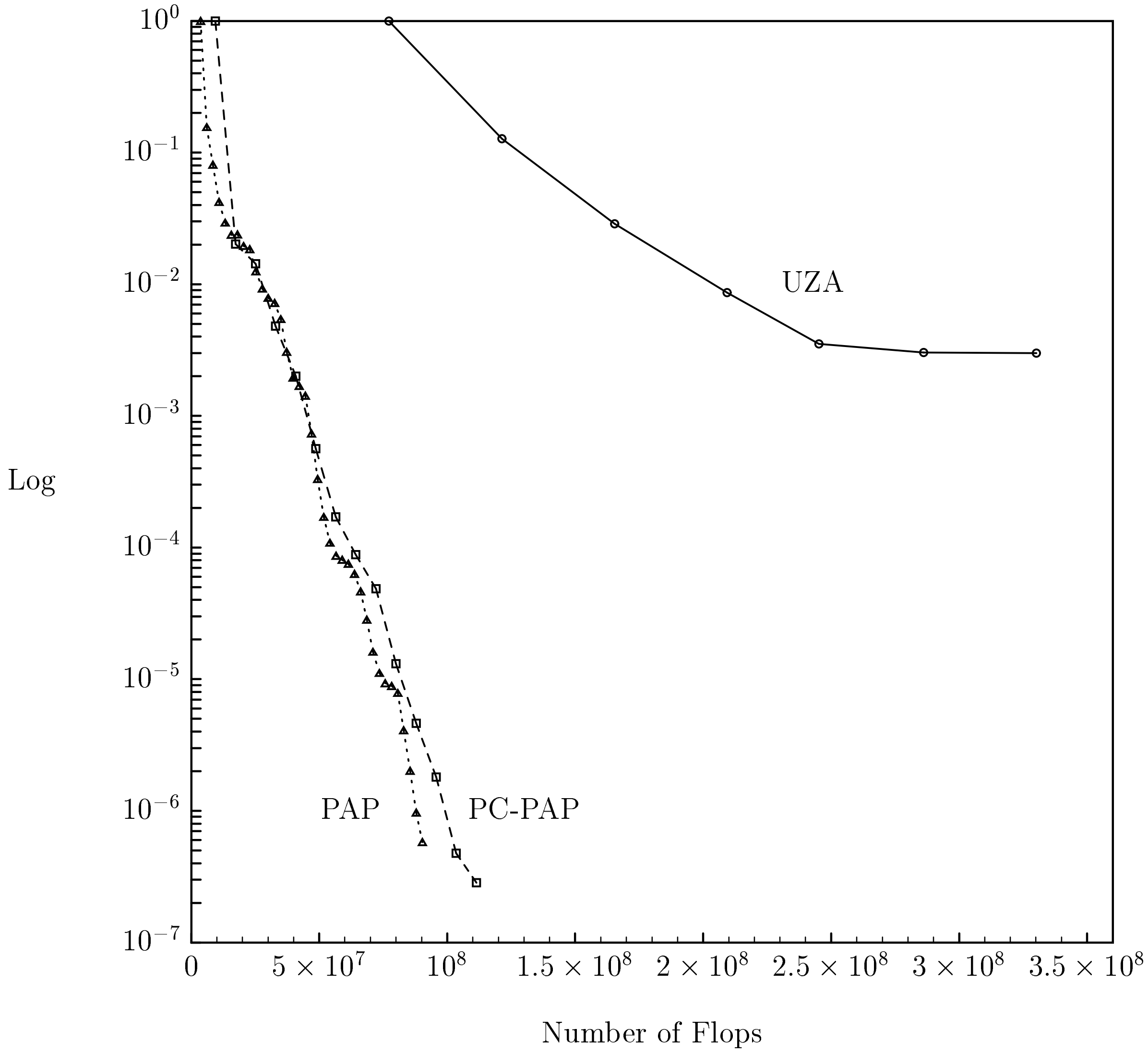

FIG. 14. *Relative Residual Norm vs. Flops for* NIMBUS30*; circle: Uzawa, triangle: PAP, box: PC-PAP*

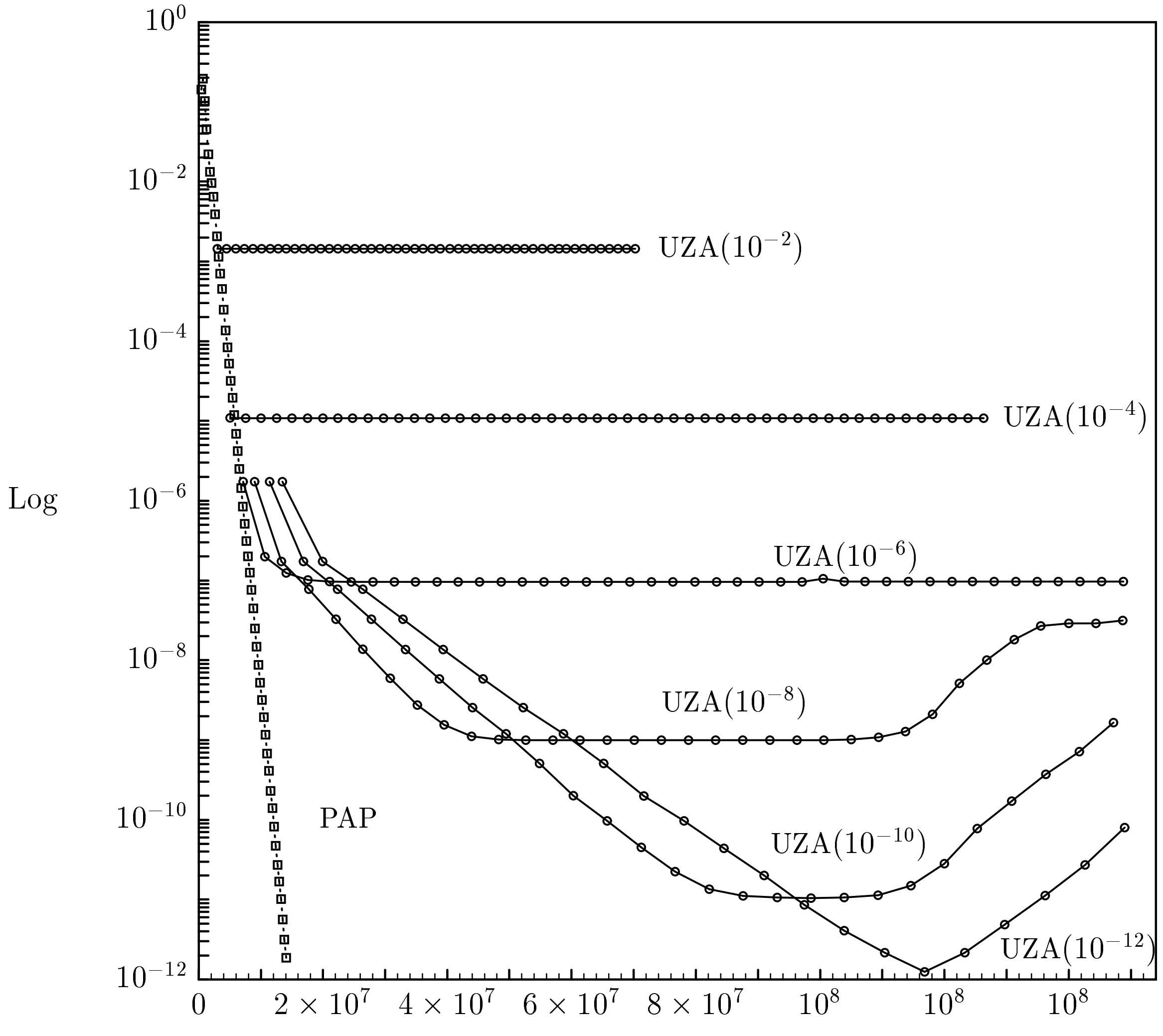

FIG. 15. *Absolute Residual Norm vs. Flops for* CONVEC8 *as Inner Tolerance on UZA Varies; circle: UZA, box: PAP*

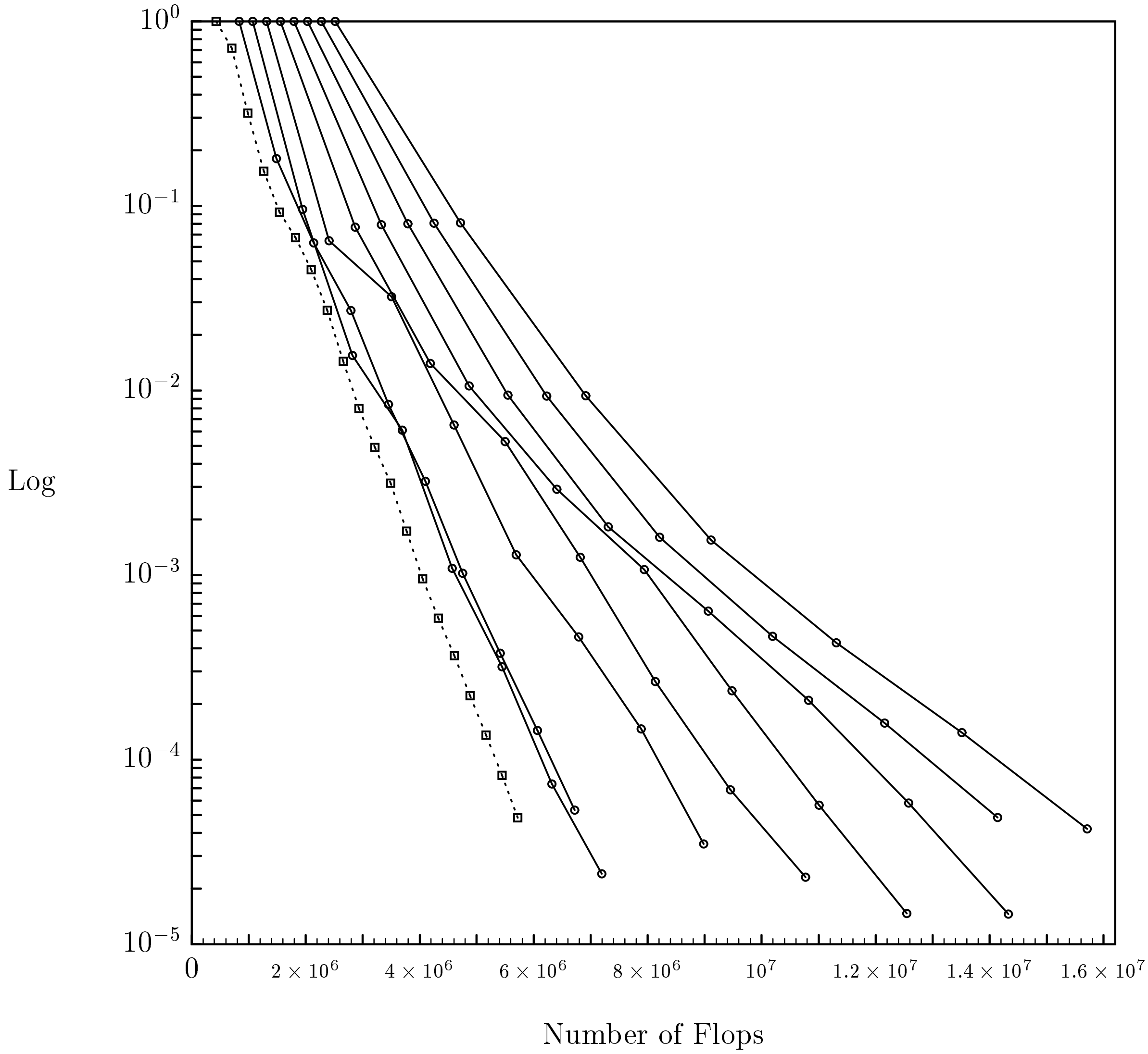

FIG. 16. *Relative Residual vs. Flops for* CONVEC8 *as Inner Tolerance on PC-PAP Varies; circle: PC-PAP, box: PAP*

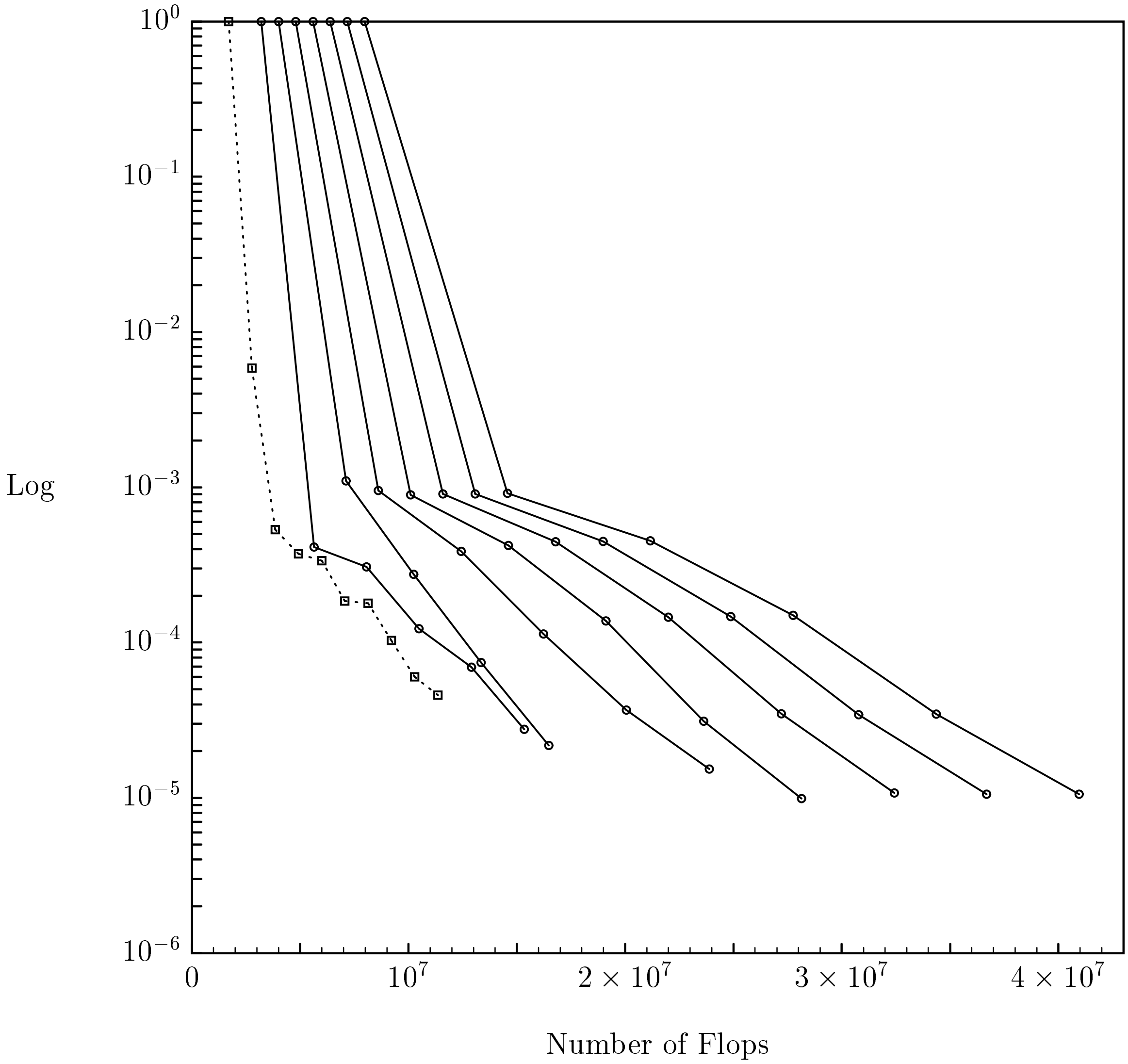

FIG. 17. *Relative Residual vs. Flops for* STRAT8 *as Inner Tolerance on PC-PAP Varies; circle: PC-PAP, box: PAP*

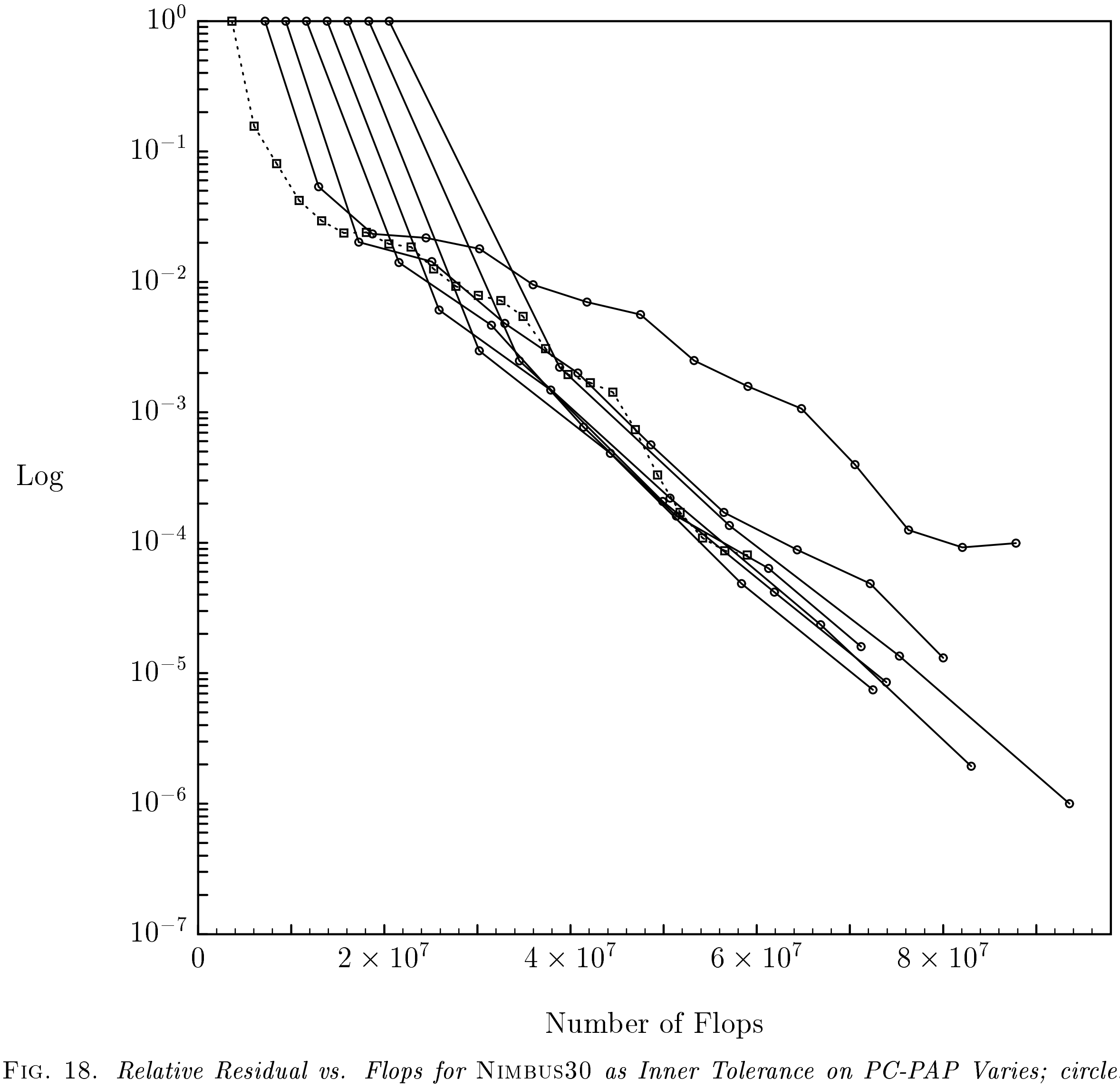

FIG. 18. *Relative Residual vs. Flops for* NIMBUS30 *as Inner Tolerance on PC-PAP Varies; circle: PC-PAP, box: PAP*

when many iterations are needed. Theorem 5.1 assures that such a growth cannot occur in exact arithmetic, but since the projections are computed with rounding error, numerically it is possible. Figure 19 shows the norm of $B^T u$ vs. iteration number for the three test problems. The growth is nearly linear and even for a large number of iterations usually does not exceed a one order of magnitude. For STRAT8, the initial $\| B^T u \|$ is of order 10^{-11} because the method for computing the projections forms the Cholesky factor R for the seminormal equations by forming $B^T B$. More accurate methods can be based on finding R as the triangular factor of the QR decomposition, and using a few steps of iterative refinement [20].

7. Summary and Conclusions. A projection method for iteratively solving linear Stokes problems has been proposed and analyzed. Previously the projected system has been used in theoretical analyses but not as a practical computational approach. The exceptions are in nonlinear programming where a null space basis matrix is explicitly formed, and in a trace minimization algorithm for generalized eigenvalue problems where the matrix B is small and dense, consisting of a few eigenvectors.

The properties of the projection algorithm have been examined and it has been compared with a commonly used iterative method for solving linear Stokes problems, the CG-accelerated Uzawa algorithm. The projection algorithm requires fewer floating point operations, provides an estimate of the true residual norm for the problem instead of the norm of a derived system, and does not require setting arbitrary parameters for inner tolerances. However, the scheme does require additional memory when the projections are computed directly as recommended here. Based on a projection method used in CFD, an approximation has been analyzed and bounds provided for its accuracy. This leads to a preconditioner for the method, which can have significant impact on a vector machine.

Numerical testing on realistic problems from a CFD code shows that the projection method requires fewer floating point operations and achieves smaller residual norms than the Uzawa method. The effectiveness of this algorithm relies on three important features of the application, however: first, the leading principal submatrix A is well-conditioned, so diagonal preconditioning of it is effective. Secondly, the number of pressure unknowns is significantly smaller than for the velocity unknowns. Since the matrix $B^T B$ roughly corresponds to a Laplace operator on the pressure nodes, this means direct methods can be efficiently applied to the sparse matrix $B^T B$. Finally, the matrix in (1) is symmetric. Replacing the matrix B^T in it with another matrix C leads to using oblique projectors, which in general tend to be numerically difficult to compute accurately. However, for CFD problems this occurs when $C = B^T M$ for a matrix M with well-understood properties, which may allow the practical computation of the requisite projections.

REFERENCES

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

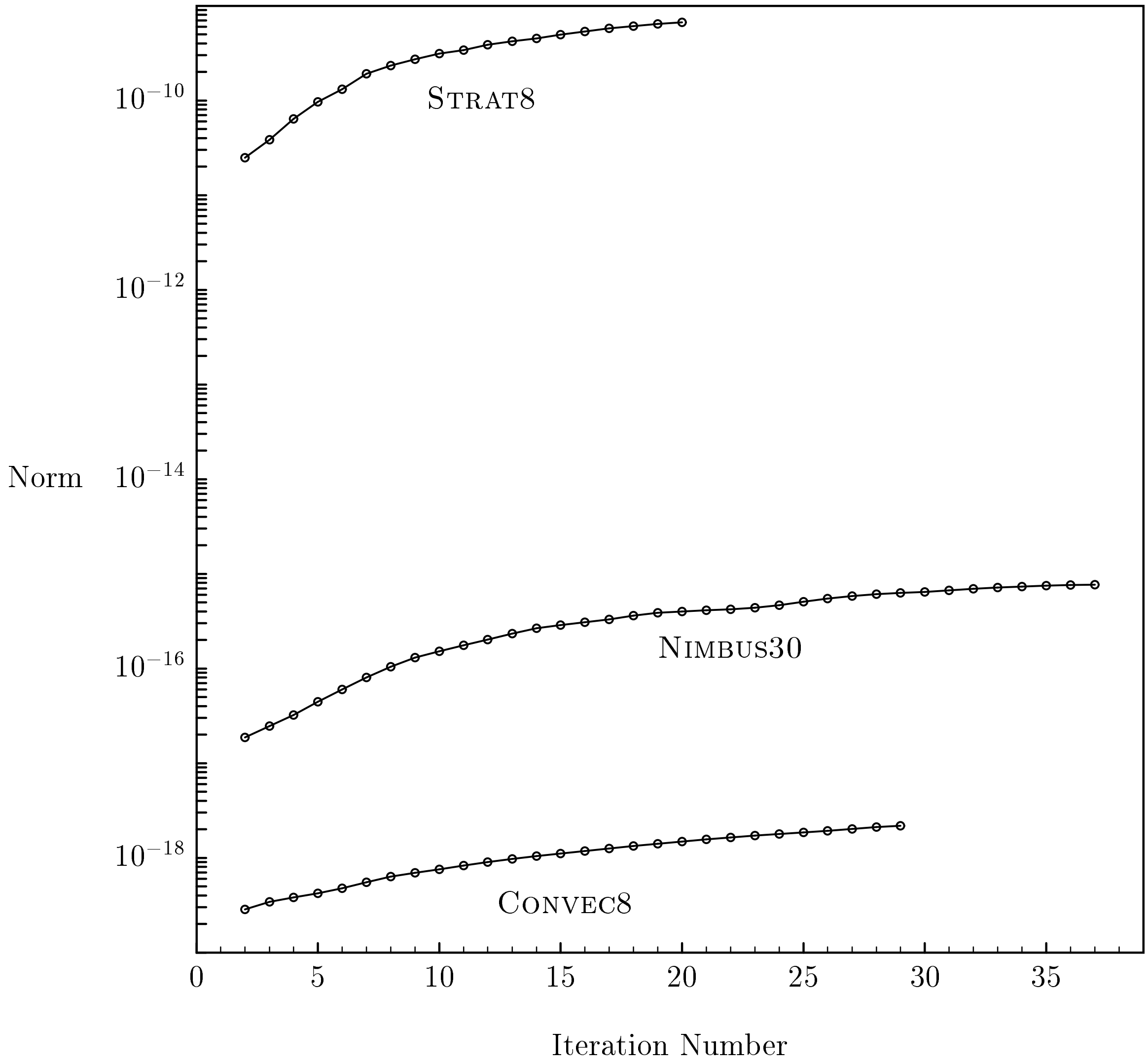

FIG. 19. *$B^T u$ Norm vs. Iteration Number for the Three Test Problems*

[2] D. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

[3] J. Chabard, *N3S Code for Fluid Mechanics, Theoretical Manual*, Electricité de France, Direction des Etudes et Recherches, 6 quai Watier, 78400 Chatou, version 2.0 ed., 1989. Report Number HE-41/89.14.

[4] A. Chorin, *Numerical solution of the navier–stokes equations*, Math. Comput., 22 (1968), pp. 745–762.

[5] H. Elman, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Tech. Rep. 229, Yale University, 1982.

[6] C. Farhat, *A Lagrange multiplier based divide and conquer finite element algorithm*, Computing Systems in Engineering, 2 (1991), pp. 149–156.

[7] R. Fletcher, *Practical Methods of Optimization*, vol. 2, John Wiley & Sons, New York, 1981.

[8] M. Fortin and R. Glowinski, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary–Value Problems*, North Holland, New York, 1983.

[9] D. Gabay, *Minimizing a differentiable function over a differentiable manifold*, Journal of Optimization Theory and Applications, 37 (1982), pp. 171–219.

[10] ———, *Reduced quasi-Newton methods with feasibility improvement for nonlinearly constrained optimization*, Mathematical Programming Study, 16 (1982), pp. 18–44.

[11] J. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: Design and implementation*, SIAM J. Matrix Analysis, (1992). To appear in 1992; also available as a technical report from the Mathworks.

[12] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, New York, 1981.

[13] P. Gresho, *On the theory of semi–implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 1: Theory*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 587–620.

[14] ———, *On the theory of semi–implicit projection methods for viscous incompressible flow and its implementation via a finite element method that also introduces a nearly consistent mass matrix. part 2: Implementation*, International Journal for Numerical Methods in Fluids, 11 (1990), pp. 621–659.

[15] P. Gresho and R. Sani, *On pressure boundary conditions for the incompressible Navier–Stokes equations*, International Journal for Numerical Methods in Fluids, 7 (1987), pp. 1111–1145.

[16] M. Gunzburger, *Finite Element Methods for Viscous Incompressible Flow*, Academic Press, New York, 1989.

[17] S.-P. Han, *Superlinearly convergent variable metric algorithms for general nonlinear programming problems*, Mathematical Programming, 11 (1976), pp. 263–282.

[18] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, Journal of Research National Bureau of Standards, 49 (1952), pp. 409–436.

[19] J. Laminie and U. Meier, *Solving Navier-Stokes equations on the Cedar multi-cluster system*, Tech. Rep. 978, Center for Supercomputing Research and Development, University of Illinois, Urbana, Illinois, March 1990.

[20] P. Matstoms, *The Multifrontal Solution of Sparse Linear Least Squares Problems*, PhD thesis, Linköping University, Department of Mathematics, Linköping University, S-581 83 Linköping, Sweden, 1991. Also available as Thesis No. 293, LIU-TEK-LIC-1991:33.

[21] M. Powell, *The convergence of variable metric algorithms for nonlinearly constrained optimization calculations*, in Nonlinear Programming 3, O. Mangasarian, R. Meyer, and S. Robinson, eds., Academic Press, New York, 1978, pp. 27–63.

[22] A. Sameh and J. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM Journal of Numerical Analysis, 19 (1982), pp. 1243–1259.

[23] D. Semeraro, *Operator Splitting Methods for Navier–Stokes Equations*, PhD thesis, University of Illinois Urbana-Champaign, 1992.

[24] K. Tanabe, *Differential geometric methods in nonlinear optimization*, in Applied Nonlinear Analysis, V. Lakshmikantham, ed., Academic Press, New York, 1979, pp. 707–719.

[25] R. TEMAM, *Navier-Stokes Equations*, North Holland, New York, 1984.

[26] J. WISNIEWSKI, *On Solving the Large Sparse Generalized Eigenvalue Problem*, PhD thesis, University of Illinois Urbana-Champaign, 1981. Also available as Technical Report UIUCDCS–R–81–1056, Department of Computer Science, University of Illinois – Urbana.