ORTHOGONALIZATION-BASED ITERATIVE SOLVERS FOR GENERALIZED STOKES PROBLEMS

R. BRAMLEY, X. WANG and D. PELLETIER

Two methods for solving the generalized Stokes problems that occur in viscous, incompressible flows are described and tested. Both are based on some type of linear algebraic orthogonalization process. The first, EMGS, is a preconditioner derived from an incomplete Gram–Schmidt factorization, and it is proven to exist whenever the matrix being preconditioned can be factored using Gaussian elimination. For M–matrices, EMGS is proven to give a better approximate factorization than ILU preconditioning, but this requires a large amount of intermediate storage for computing the preconditioner. The second orthogonalization approach gives a derived system which eliminates the indefiniteness of the generalized Stokes problem, and keeps the velocity field in the space of discrete null–divergence vectors. Both methods are tested on problems extracted from a finite element fluid flow solver.

1. INTRODUCTION

One of the most challenging numerical linear algebra problems that computational fluid dynamics (CFD) applications pose is the generalized Stokes problem

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

where $B \in \Re^{n\times q}$, $A \in \Re^{n\times n}$, and u and p are the velocity and pressure vectors, respectively. Even if A is symmetric and positive definite, (1) is indefinite which causes difficulties both for iterative solvers and commonly used preconditioners. This work examines approaches for solving generalized Stokes problems which share the characteristic of being based on some form of orthogonalization. Essentially there are two motivations for considering these approaches.

First, in spite of over 40 years of intensive research into iterative methods for solving linear systems, there is essentially only one method which is theoretically robust, that is, guaranteed to converge for any nonsingular linear system. That method is conjugate gradients (CG) applied to the normal equations $G^TGx = G^Tb$[1]; other robust methods such as row projection algorithms[2] and CGNE[3] can be viewed as normal equations for a derived system. In spite of this theoretical robustness, CG on the normal equations often has an unacceptably low rate of convergence. This leads to the idea of preconditioning the normal equations by finding a matrix M that approximates G in some sense, for which it is easy to solve systems of the form $Mz = r$, and then to apply the iterative method to the system $M^{-1}Gx = M^{-1}b$. Heuristically, if M is a good approximation to G then $M^{-1}G$ approximates the identity matrix, making the iterative method converge faster.

However, if CG on the normal equations is to be used, the goal can be relaxed to have $M^{-1}G \approx Q$, where Q is an orthogonal matrix. In that case the preconditioned normal equations approximate the identity matrix: $(M^{-1}G)^T(M^{-1}G) \approx Q^TQ = I$.

Note that since the set of orthogonal matrices is much larger than the set containing only the identity matrix, this provides a larger target for our preconditioning method to aim at. Section 3. of this chapter presents a new preconditioner which is based on ideas of *incomplete orthogonalization*, in the same way that many standard preconditioners are based on ideas of *incomplete LU factorization*. This preconditioner, called extended incomplete modified Gram–Schmidt (EMGS), is derived from a class of preconditioners which are more robust than standard methods, both in theory and numerically.

The second motivation for using orthogonalization approaches is based on an interpretation of the Stokes problem as being a constrained problem. In this view, we are solving for u subject to the constraint $B^T u = g$, and p is a Lagrange multiplier. The solution u can be limited to the affine set of vectors satisfying $B^T u = g$ by using orthogonal projections. This converts the system to a smaller one. When A is symmetric and positive definite the derived system is symmetric, positive semidefinite, and consistent, so the CG method can be applied to solve it. When A is nonsymmetric, other Krylov subspace methods can be applied to the derived system. Although orthogonal projectors have long been used as theoretical tools, this chapter shows that they can also be used to form a practical iterative solution strategy.

2. TARGET APPLICATION

This chapter focuses, for testing purposes, on systems of equations of the form (1) arising from the finite element discretization of the Navier–Stokes equations for an incompressible flow. However, the scope of the methods described is much broader, and they can be applied to systems arising from finite differences, finite volume, and spectral element methods.

Several approaches can be taken to solve the generalized Stokes problem. The first one solves (1) directly. A difficulty arises from the block of zeros on the diagonal, which makes the system indefinite. The constraint equation $B^T u = g$ and its associated Lagrange multiplier p can be handled by the Uzawa algorithm which solves for u in the first block of equations and substitutes it into the second block, giving the system

$$A_0 p = f_0, \tag{2}$$

where

$$\begin{aligned} A_0 &= B^T A^{-1} B \\ f_0 &= B^T A^{-1} f - g \end{aligned}$$

Problem (1) can then be solved in two steps, by first solving (2) and then retrieving u by solving $Au = f - Bp$. The computation of A^{-1} in the expression for A_0 can be eliminated by using an iterative method for solving (2). The Uzawa algorithm is obtained by applying a gradient (steepest descent) algorithm. For the composite coefficient matrix A_0, this becomes:

Algorithm Uzawa

Set $p = p_0$, where p_0 is the initial pressure.
Set $R_p = B^T u_0 - g$, where u_0 is the initial velocity
while ($\|R_p\| >$ tolerance)
  Solve $Au = f - Bp$
  Evaluate $R_p = B^T u - g$
  Solve $Aw = BR_p$
  Evaluate $t = R_p^T R_p / R_p^T B^T w$
  Set $p = p + tR_p$
endwhile
Solve $Au = f - Bp$

The convergence rate of the Uzawa algorithm can be greatly improved by applying it to the augmented (penalized) system

$$[A + rBB^T]u = f - Bp + rBg,$$

where r is the penalty parameter and is usually given a large value. This leads to the following algorithm

Augmented Lagrangian Algorithm

Set $p = p_0$, where p_0 is the initial pressure.
Set $R_p = B^T u_0 - g$, where u_0 is the initial velocity
Set $A_r = A + rBB^T$ and $f_r = f + rBg$
while ($\|R_p\| >$ tolerance)
  Solve $A_r u = f_r - Bp$
  Evaluate $R_p = B^T u - g$
  Solve $A_r w = BR_p$
  Evaluate $t = R_p^T R_p / R_p^T B^T w$
  Set $p = p + tR_p$
endwhile
Solve $A_r u = f - Bp$

For large values of r[4], the algorithm further simplifies to

Simplified Augmented Lagrangian Algorithm

Set $p = p_0$, where p_0 is the initial pressure.
Set $R_p = B^T u_0 - g$, where u_0 is the initial velocity
Set $A_r = A + rBB^T$ and $f_r = f + rBg$
while ($\|R_p\| >$ tolerance)
  Solve $A_r u = f_r - Bp$
  Evaluate $R_p = B^T u - g$
  Set $p = p + rR_p$
endwhile
Solve $A_r u = f - Bp$

The augmented Lagrangian approach offers the advantage of solving smaller systems of order n as opposed to systems of order $n + q$. However the price paid is that the matrix A_r has a poor distribution of eigenvalues for large values of r, as illustrated in Fig. 1. Direct solvers can handle this difficulty and are feasible and cost-effective for two–dimensional problems, but are impractical for three–dimensional problems because of memory requirements. For a uniform $s \times s$ mesh in two dimensions direct solvers require $\mathcal{O}(s^3)$ storage, while a $s \times s \times s$ mesh in three dimensions requires $\mathcal{O}(s^5)$ storage.

The task here is to iteratively solve penalty or augmented Lagrangian systems for useful 3-D problems. Because of the ill–conditioning of the coefficient matrix, this requires finding robust and effective preconditioning approaches, and we now present two such approaches.

3. INCOMPLETE ORTHOGONALIZATION PRECONDITIONERS

Performing a complete orthogonal factorization $G = QR$, with Q orthogonal and R upper triangular, suffers from the same drawback as applying Gaussian elimination. Both create new nonzeros (*fill–in*) during the factorization process, possibly exceeding the storage capabilities of the computer. In addition, an orthogonal factorization is computationally more expensive than Gaussian elimination, although the factorization itself is numerically more reliable to compute.

Incomplete orthogonal factorizations can be derived in the same way as incomplete LU factorization (ILU), by discarding certain fill–in entries during the factorization process. Early work[5,6,7,8] indicated relatively poor results, because of the additional storage needed to keep the incomplete Q factor, or because the factorization process could suffer numerical breakdown, or because the resultant preconditioner did not accelerate the iterative process sufficiently. Another approach[9,10] uses the idea of *compression*, by implicitly working on G^TG and storing only the incomplete factor R. This compressed incomplete modified Gram–Schmidt (CIMGS) factorization will not fail when G is full rank. In[9], the connection between CIMGS and incomplete Cholesky (IC) factorization on G^TG was established, showing that CIMGS is more robust and provides a better approximation to the complete Cholesky factor than IC. Here we extend CIMGS to nonsymmetric systems, giving a new preconditioner called extended incomplete modified Gram–Schmidt (EMGS). Because we will be applying EMGS to matrices of the same size as A, rather than directly on G, the algorithms are stated as operating on an $n \times n$ matrix.

3.1 Extended IMGS

Let $\mathcal{P}$ be a *drop set*, that is, positions (i, j) in the incomplete factor R which we require to be zero. In incomplete LU factorization, a common choice (ILU(0)) is to let $\mathcal{P}$ be the set of positions where A is zero. If the nonzeros of A are called *level 0* elements and the fill–in elements directly caused by level 0 entries are called level 1 fill–in, then other drop sets can be defined recursively. Keeping only entries that are of level s or lower defines the ILU(s) preconditioner.

Because it is needed for later proofs, we now state the incomplete modified Gram–Schmidt algorithm applied to a matrix W, which is assumed to have columns w_k, $k = 1, \ldots, n$. The columns of the matrix Q are similarly denoted q_k. The algorithm gives $W = QR$, where R is upper triangular but Q need not be orthgonal.

Algorithm IMGS

for $k = 1, 2, \ldots, n$

(I1) $r_{kk} = \| w_k \|_2$

(I2) $q_k = w_k / r_{kk}$

for $j = k+1, k+2, \ldots, n$

(I3) $$r_{kj} = \begin{cases} 0 & (k,j) \in \mathcal{P} \\ q_k^T w_j & (k,j) \notin \mathcal{P} \end{cases}$$

(I4) $w_j = w_j - q_k r_{kj}$

endfor

endfor

CIMGS[10] avoids storing Q by working instead on the normal equations $W^T W$, which is symmetric and positive definite. The following algorithm is stated for a general symmetric and positive definite matrix A:

Algorithm CIMGS

for $k = 1, 2, \ldots, n$,

(C1) $a_{kk} = \sqrt{a_{kk}}$

(C2) $r_{kk} = a_{kk}$

for $j = k+1, k+2, \ldots, n$

(C3) $a_{kj} = a_{kj} / \sqrt{a_{kk}}$

(C4) $$r_{kj} = \begin{cases} 0 & (k,j) \in \mathcal{P} \\ a_{kj} & (k,j) \notin \mathcal{P} \end{cases}$$

endfor

for $j = k+1, k+2, \ldots, n$

for $i = k+1, k+2, \ldots, n$

(C5) $a_{ij} = a_{ij} - a_{ki} a_{kj} \quad (k,j) \notin \mathcal{P}$ or $(k,i) \notin \mathcal{P}$

endfor

endfor

endfor

The EMGS algorithm is obtained by simply applying the operations of CIMGS to a possibly nonsymmetric matrix A. This requires keeping two factors $L = [l_{ij}]$ and $U = [u_{ij}]$, which are lower and upper triangular, respectively.

Algorithm EMGS

for $k = 1, 2, \ldots, n$,

$u_{kk} = a_{kk}$

$l_{kk} = 1$

(E1) $a_{ik} = a_{ik} / a_{kk}$, $i = k+1, \ldots, n$

for $j = k+1, k+2, \ldots, n$

(E2) $$u_{kj} = \begin{cases} 0 & (k,j) \in \mathcal{P} \\ a_{kj} & (k,j) \notin \mathcal{P} \end{cases}$$

(E3) $$l_{jk} = \begin{cases} 0 & (j,k) \in \mathcal{P} \\ a_{jk} & (j,k) \notin \mathcal{P} \end{cases}$$

endfor

for $j = k+1, k+2, \ldots, n$

for $i = k+1, k+2, \ldots, n$

(E4) $a_{ij} = a_{ij} - a_{ik} a_{kj} \quad (k,j) \notin \mathcal{P}$ or $(i,k) \notin \mathcal{P}$

endfor

endfor
endfor

If the condition in statement (E4) is changed to $(k,j) \notin \mathcal{P}$ and $(i,k) \notin \mathcal{P}$ then the algorithm computes the ILU preconditioner with the same drop set $\mathcal{P}$. The condition in (E4) is the crucial difference that lead to a more stable preconditioner.

Two important results will now be established, the first showing that if the LU factorization (without pivoting) of A exists, then EMGS will not break down with a division by zero, and the second showing that if A is an M–matrix, then EMGS produces a better approximation of the true LU factors than the ILU factorization does.

Theorem 3.1. *Let $\mathcal{P}$ be a given drop set not containing any diagonal positions. If $A \in R^{n\times n}$ has full rank, and the complete LU factorization of A without pivoting exists, then EMGS using drop set $\mathcal{P}$ completes.*

Proof. Let $A = Y^T W$ where W and Y^T are the complete LU factors of A. We prove the theorem by induction on n. It holds trivially for $n = 1$. Assume that it is true for matrices of order $n-1$. The EMGS algorithm computes the first row and column of the factorization, then updates and recursively applies the algorithm to the trailing $(n-1) \times (n-1)$ submatrix $\tilde{A}$. So it suffices to show that after one step of EMGS factorization, $\tilde{A}$ is nonsingular and can be factored with Gaussian elimination without pivoting. Let $\tilde{W}$ and $\tilde{Y}$ be the reduced matrices obtained from one step of IMGS on W and Y, respectively.

The key to showing this is to use the relationship between applying one step of IMGS to W and Y and applying one step of EMGS to A. Specifically, the first row of U and first column of L generated by one step of EMGS on A are equal to the first rows of the triangular factors generated by one step of IMGS on W and Y, respectively. Furthermore, $\tilde{A} = \tilde{Y}^T \tilde{W}$ which will complete the induction.

Let the columns of A be $a_1, a_2, \ldots, a_n$, and similarly for the other matrices. After one step of IMGS applied to W and Y,

$$W = (w_1, \ldots, w_n) = (q_1, \ldots, q_n) \begin{pmatrix} r_{11} & (r_{12}, \ldots, r_{1n}) \\ 0 & I \end{pmatrix}, \tag{3}$$

where $q_i = \begin{cases} w_i & (1,i) \in \mathcal{P} \\ w_i - r_{1i} q_1 & (1,i) \notin \mathcal{P} \end{cases}$, and $r_{1i} = \begin{cases} 0 & (1,i) \in \mathcal{P} \\ q_1^T w_i & (1,i) \notin \mathcal{P} \end{cases}, i \neq 1.$

Similarly

$$Y = (y_1, \ldots, y_n) = (p_1, \ldots, p_n) \begin{pmatrix} t_{11} & (t_{12}, ..., t_{1n}) \\ 0 & I \end{pmatrix} \tag{4}$$

where $p_i = \begin{cases} y_i & (i,1) \in \mathcal{P} \\ y_i - t_{1i} p_1 & (i,1) \notin \mathcal{P} \end{cases}$ and $t_{1i} = \begin{cases} 0 & (i,1) \in \mathcal{P} \\ p_1^T y_i & (i,1) \notin \mathcal{P} \end{cases}, i \neq 1.$

From the definition of W and Y, $y_1 = w_1/r_{11}$, and from the IMGS algorithm $q_1 = w_1/ \parallel w_1 \parallel_2 = w_1/r_{11}$, $p_1 = y_1/ \parallel y_1 \parallel_2 = y_1$. Therefore $r_{1i} = q_1^T w_i = y_1^T w_i =$

a_{1i} when $(1,i) \notin \mathcal{P}$ and $t_{1i} = p_1^T y_i = w_1^T y_i / r_{11} = a_{i1}/a_{11}$ when $(i,1) \notin \mathcal{P}$. Four cases occur for the elements $p_i^T q_j$ of the product $(p_2, p_2, \ldots, p_n)^T (q_2, q_3, \ldots, q_n)$:

Group 1: $(1,i) \in \mathcal{P}$ and $(j,1) \in \mathcal{P}$. Then $p_i^T q_j = y_i^T w_j = a_{ij}$.

Group 2: $(1,i) \in \mathcal{P}$ and $(j,1) \notin \mathcal{P}$. Then

$$p_i^T q_j = y_i^T (w_j - r_{1j} q_1) = a_{ij} - a_{1j} y_i^T w_1 / a_{11} = a_{ij} - a_{1j} a_{i1} / a_{11}.$$

Group 3: $(1,i) \notin \mathcal{P}$ and $(j,1) \in \mathcal{P}$. Then

$$p_i^T q_j = (y_i - t_{1i} p_1)^T w_j = a_{ij} - a_{i1} y_1^T w_j / a_{11} = a_{ij} - a_{i1} a_{1j} / a_{11}.$$

Group 4: $(1,i) \notin \mathcal{P}$ and $(j,1) \notin \mathcal{P}$. Then

$$\begin{aligned} p_i^T q_j &= (y_i - t_{1i} p_1)^T (w_j - r_{1j} q_1) \\ &= y_1^T w_j - r_{1j} y_i^T q_1 - t_{1i} p_1^T w_j + t_{1i} r_{1j} p_1^T q_1 \\ &= a_{ij} - a_{1j} a_{i1}/a_{11} - a_{1j} a_{i1}/a_{11} + a_{1j} a_{i1}/a_{11} \\ &= a_{ij} - a_{1j} a_{i1}/a_{11} \end{aligned}$$

So one step of EMGS gives $\tilde{A} = (p_2, p_2, \ldots, p_n)^T (q_2, q_3, \ldots, q_n)$ and hence $\tilde{A}$ is nonsingular. In addition, for each $i \geq 2$ the i-th leading principle submatrices of $\tilde{A}$ are $(p_2, p_2, \ldots, p_i)^T (q_2, q_3, \ldots, q_i)$ and so are also nonsingular. Therefore, the LU factorization of this reduced matrix also exists. Using the induction hypothesis, EMGS factorization can be completed on the reduced matrix. Letting the EMGS factors of $\tilde{A}$ be L_2 and U_2,

$$L^T = \begin{pmatrix} t_{11} & (t_{12}, t_{13}, \ldots, t_{1n}) \\ 0 & L_2^T \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} r_{11} & (r_{12}, r_{13}, \ldots, r_{1n}) \\ 0 & U_2 \end{pmatrix},$$

completing the proof. □

This theorem insures that EMGS is more stable than ordinary incomplete LU factorization. It does not establish the relationship between full LU, ILU, and EMGS. The next theorem shows that, under certain conditions, EMGS will generate a better approximation of the full LU factors than ILU, suggesting it is a better quality preconditioner. The price paid for improved robustness and approximating quality can be seen by noting that the only difference between ILU factorization for a given sparsity pattern and EMGS with the same sparsity pattern is in step (E4) of EMGS. ILU factorization replaces the condition "$(k,j) \notin \mathcal{P}$ **or** $(i,k) \notin \mathcal{P}$" with "$(k,j) \notin \mathcal{P}$ **and** $(i,k) \notin \mathcal{P}$". This means for the same drop set $\mathcal{P}$, EMGS requires more computation and intermediate storage than ILU.

Theorem 3.2. *Let $A \in R^{n \times n}$ be an M-matrix and $\mathcal{P}$ be a given drop set for the sparsity pattern in ILU and EMGS factorization. Let L_{lu}, L_{ilu} and L_{emgs} be the unit lower and U_{lu}, U_{ilu} and U_{emgs} be the upper triangular factors of LU, ILU and EMGS factorization of A, respectively. Let $E^L_{emgs} = L_{emgs} - L_{lu}$, $E^L_{ilu} = L_{ilu} - L_{lu}$, $E^U_{emgs} = U_{emgs} - U_{lu}$ and $E^U_{ilu} = U_{ilu} - U_{lu}$ be the error matrices from* *the approximation of the full LU factorization by EMGS and ILU respectively. Then the following elementwise inequalities hold:*

$$L_{lu} \le L_{emgs} \le L_{ilu}, \;\; U_{lu} \le U_{emgs} \le U_{ilu},$$

and

$$E^L_{emgs} \le E^L_{ilu}, \;\; E^U_{emgs} \le E^U_{ilu}$$

Some convenient notation and earlier results will help in the proof. The matrix $A(s:t, u:v)$ denotes the $(t-s+1)\times(v-u+1)$ submatrix of A consisting of entries a_{ij} with $s \le i \le t$ and $u \le j \le v$. Let p be an ordered set of indices drawn from the set $\{1,2,\ldots,n\}$. Then $A(p,j)$ is the column vector of length n whose i–th component is a_{ij} if $i \in p$ and is zero otherwise. A similar definition holds for $A(i,p)$. Also, let $A^{(i)}_{lu}$, $A^{(i)}_{ilu}$ and $A^{(i)}_{emgs}$ be the reduced matrices of order $(n-i)\times(n-i)$ obtained after i steps of LU, ILU and EMGS factorization, respectively.

Ky Fan[11] has shown that if $A=(a_{ij})$ is a nonsingular M-matrix, then $A^{(1)}_{lu}$ exists and is also an M-matrix. From this, induction shows that $A^{(i)}_{lu}$, $i=1,2,...,n-1$ are M-matrices and the LU factorization of A exists when A is M-matrix. In addition, the factors L_{lu} and U_{lu} have all positive diagonal and non–positive off diagonal elements, so L and U are also M-matrices.

For ILU factorization, J.A. Meijerink and H. A. Van Der Vorst[11] have shown that if A is M-matrix, then ILU factorization completes, $A^{(i)}_{ilu}$, $i=1,2,...,n-1$ are M-matrices, and the factors L_{ilu} and U_{ilu} have all positive diagonal and non–positive off diagonal elements, and so L_{ilu} and U_{ilu} are also M-matrices. The proof of Theorem 3.2 shows the same results for EMGS factorization, allowing comparison with the other factorizations.

Proof. First, we compare EMGS with ILU. From the algorithms, the first step computes the elements of $L_{emgs}(1:n,1)$, $L_{ilu}(1:n,1)$, $U_{emgs}(1,1:n)$ and $U_{ilu}(1,1:n)$. Since the drop set $\mathcal{P}$ is the same for both factorizations, those columns and rows are the same.

When comparing LU with EMGS, after one step the elements of $L_{emgs}(1:n,1)$ are greater than or equal to the corresponding elements of L_{lu}, and elementwise the first row of U_{emgs} is greater than or equal to the first row of $U^{(1)}_{lu}$, because M-matrix properties imply that the dropped elements are all non–positive. This establishes the relations

$$\begin{aligned} L_{lu}(1:n,1) &\le L_{emgs}(1:n,1) = L_{ilu}(1:n,1) \\ U_{lu}(1,1:n) &\le U_{emgs}(1,1:n) = U_{ilu}(1,1:n) \end{aligned}$$

Now consider the reduced matrices $A^{(1)}_{lu}$, $A^{(1)}_{ilu}$ and $A^{(1)}_{emgs}$. Let $p^l_i = \{j | j > i \text{ and } (j,i) \notin \mathcal{P}\}$ and $p^u_i = \{j | j > i \text{ and } (i,j) \notin \mathcal{P}\}$ be the set of allowed off–diagonal element positions for the i–th column of L and row of U, respectively. Let

$\bar{p}_i^l = \{j | j > i \text{ and } (j,i) \in \mathcal{P}\}$ and $\bar{p}_i^u = \{j | j > i \text{ and } (i,j) \in \mathcal{P}\}$ be the complement set of p_i^l and p_i^u respectively.

On step i, the three factorization algorithms give

$$\begin{aligned}
A_{lu}^{(i)} &= A_{lu}^{(i-1)}(i+1:n, i+1:n) - \frac{A_{lu}^{(i-1)}(i+1:n,i)A_{lu}^{(i-1)}(i,i+1:n)}{A_{lu}^{(i-1)}(i,i)}, \\
A_{emgs}^{(i)} &= A_{emgs}^{(i-1)}(i+1:n, i+1:n) - \frac{A_{emgs}^{(i-1)}(p_i^l,i)A_{emgs}^{(i-1)}(i,i+1:n)}{A_{emgs}^{(i-1)}(i,i)} \\
&\quad - \frac{A_{emgs}^{(i-1)}(\bar{p}_i^l,i)A_{emgs}^{(i-1)}(i,p_i^u)}{A_{emgs}^{(i-1)}(i,i)}, \text{ and} \\
A_{ilu}^{(i)} &= A_{ilu}^{(i-1)}(i+1:n, i+1:n) - \frac{A_{ilu}^{(i-1)}(p_i^l,i)A_{ilu}^{(i-1)}(i,p_i^u)}{A_{ilu}^{(i-1)}(i,i)}.
\end{aligned}$$

Since A is an M-matrix, all off–diagonal elements are non–positive and all diagonal elements are positive. It follows easily that $A_{lu}^{(1)} \le A_{emgs}^{(1)} \le A_{ilu}^{(1)}$. Continuing the process for steps $i = 2, 3, \ldots, n-1$ gives the relations:

$$\begin{aligned}
A_{lu}^{(i)} &\le A_{emgs}^{(i)} \le A_{ilu}^{(i)} \\
L_{lu}(1:n,i) &\le L_{emgs}(1:n,i) \le L_{ilu}(1:n,i), \\
U_{lu}(i,1:n) &\le U_{emgs}(i,1:n) \le U_{ilu}(i,1:n).
\end{aligned}$$

Therefore $L_{lu} \le L_{emgs} \le L_{ilu}$ and $U_{lu} \le U_{emgs} \le U_{ilu}$, and so $E_{emgs}^L \le E_{ilu}^L$ and $E_{emgs}^U \le E_{ilu}^U$. □

Summarizing, the theory indicates that EMGS factorization will not break down when applied to matrices for which LU factorization without pivoting succeeds, and when applied to an M-matrix, EMGS will better approximate the full LU factorization than ILU does.

As a final comment, the motivation for seeking a more robust preconditioner comes from penalized and augmented Lagrangian methods for solving the generalized Stokes problem (1). As Fig. 1 illustrates, penalized systems have extremely unfavorable eigenvalue distributions for the coefficient matrix of the system. These systems challenge iterative methods, and are generally solved only with direct methods.

4. PROJECTED STOKES SOLVER

For simplicity, this section considers the Stokes problem (1) with $g = 0$. This is not a restriction on the method, and the additional cost associated with $g \ne 0$ is less than a single iteration of the proposed algorithm. The second equation is then $B^T u = 0$. In the Stokes problem this is the (discrete) null–divergence condition. As was proposed in[12], let P be the orthogonal projector onto the null space null(B^T)

of B^T. Then (1) is equivalent to solving the system $PAPu = Pf$; $Pu = u$. Summarizing earlier results[12], we have

- For certain time accurate operator splitting based methods[13] the matrix A is symmetric positive definite and well conditioned, and the CG algorithm can be used to solve $PAP = Pf$.
- More generally, for Krylov subspace iterative methods applied to $PAPu = Pf$ if the initial velocity u_0 has null–divergence, then all succeeding iterates u_k will also have null–divergence. So $Pu_k = u_k$ and only one orthogonal projection need be performed per iteration. If the initial velocity does not have null–divergence, a single projection step is made before the iterations begin.
- The number of nonzero eigenvalues of PAP is $n - q$, so generally fewer iterations are needed to solve a system with that coefficient matrix than are required for solving a system with coefficient matrix A.
- If A has a positive definite symmetric part, then so does PAP.
- The condition number of PAP is no greater than that of A.
- The projected Stokes algorithm can greatly outperform the preconditioned Uzawa method both in overall computational cost and in accuracy, for large scale three–dimensional industrial problems.

The question now is whether or not the projected Stokes method is practical for steady state problems, where A is nonsymmetric and possibly ill–conditioned.

4.1 Computing Orthogonal Projections

For the iterative methods considered here, the matrix P need not be formed or computed. Instead only the action of P on a vector is needed ($y = Px$). The orthogonal projector can written as

$$P = I - B(B^T B)^{-1} B^T \tag{5}$$

except when $B^T B$ is singular. The matrix $B^T B$ corresponds to a pressure problem, which is well posed except when Dirichlet conditions are specified for all velocity components everywhere on the boundary. In this case the pressure is defined up to an additive constant, so there are an infinite number of solution for the pressure that differ by a constant. This translates into a rank deficiency of one for B.

This situation can be remedied in either of two ways. In some cases a suitable Neumann condition can be specified to set the required pressure level and impose unicity of the pressure solution. If a Dirichlet problem must absolutely be solved, unicity can be recovered by requiring that the pressure field have zero mean value or by providing one Dirichlet value for pressure somewhere in the domain. With the projected Stokes approach, the difficulty can be handled by simply discarding any column of B, giving a full rank matrix $\tilde{B} \in \Re^{n\times(q-1)}$. The rank deficiency is such that $Be = 0$, where e is a vector of all ones. This in turn means that $\mathrm{null}(\tilde{B}^T) = \mathrm{null}(B^T)$, and so P can be computed as in (5). This is one advantage of the projected Stokes method – it does not require imposing artificial constraints in the physical model.

In the numerical implementation used here, the matrix B^TB is formed and its Cholesky factor R is found using sparse matrix techniques[14]. Since generally B has many fewer columns than A ($n \gg q$), and B^TB is symmetric and positive definite, this factorization requires only storing a small triangular matrix R. Because B^TB corresponds to a discretization of a Laplace operator on the pressure nodes, it will be sparse. Also, the matrix B does not change from one nonlinear iteration to the next, so this factorization only need be computed once in a preprocessing step. Compare this to direct methods for solving G in (1) or the penalized system $A_r = A + rBM^{-1}B^T$ that occurs in penalty or augmented Lagrangian methods: those matrices are nonsymmetric, signficantly larger, and must be refactored at every nonlinear iteration.

4.2 Relationship to Penalty Systems

Augmented Lagrangian and penalty methods solve generalized Stokes problems by treating the null divergence condition as a constraint, and penalizing it. This gives linear systems of the form

$$A + rCB \tag{6}$$

with $C = B^T$ or $C = M^{-1}B^T$, where M is a matrix that scales the columns of B elementwise. The penalty parameter r is chosen on the basis of numerical testing, and $r = 10^5$ is used here. When A is symmetric and positive definite, so is A_r. However, the penalty parameter causes A_r to have two clumps of eigenvalues, one of order 10^5 and the other of order 1. Fig. 1 shows the eigenvalues for A, A_r, and CB for a backward facing step problem with 1458 velocity and 399 pressure degrees of freedom. Note that the eigenvalues of A_r are nearly the same as the sum of those of A and the corresponding ones of CB (the 1458 — 399 = 1059 eigenvalues of CB that are exactly equal to zero are not shown.) The eigenvalues show that the penalized system is badly conditioned. Furthermore, having eigenvalues fall into two large clumps at the extreme ends cause great difficulties for iterative solvers based on Krylov subspace methods, as we show later. The numerical testing will also show that using EMGS preconditioning will generally allow solution of penalized systems, but for meshes with widely varying sizes of elements even it can fail to make iterative methods converge. Even when direct factorization methods are used on A_r, the ill-conditioning can adversely affect the accuracy of the solution.

Since the addition of the term CB causes the second clump of eigenvalues to become large, one idea is to remove the effects of penalty terms by preconditioning the system with $I + rCB$, and solving instead the system

$$(I + rCB)^{-1}(A + rCB)u = (I + rCB)^{-1}h.$$

Although this does in practice remove the penalized term contribution to the eigenvalues, the matrix $I + rCB$ is as large as A_r, and solving systems with it is nearly as expensive (recall that a practical preconditioner must provide a favorable spectrum

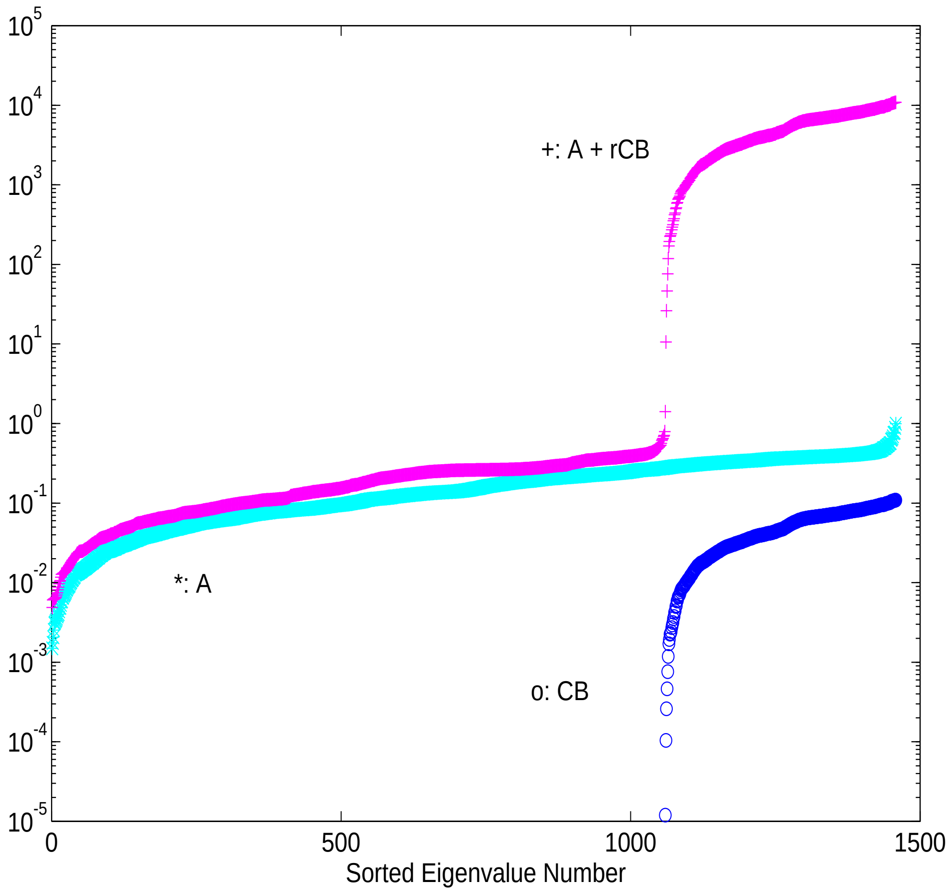

FIGURE 1: Eigenvalues for Backward Facing Step Matrices

and easily solved systems.) Applying the Sherman–Morrison–Woodbury formula[15] for the case $C = B^T$ shows that

$$(I + rB^TB)^{-1} = I - rB[I + rB^TB]^{-1}B^T = I - B[(1/r)I + B^TB]^{-1}B^T. \tag{7}$$

This form requires instead inverting the smaller matrix $(I + rB^TB)$. However, comparing the last member in (7) with the projector P reveals that for large r they become asymptotically close, and the computational costs of the two are virtually identical. Furthermore, it is easy to show that $PA_r = P(A + rCB^T) = PA$ for any choice of the penalty parameter r and any $C = M^{-1}B^T$. In other words, the projected Stokes method can be used to solve penalty systems, and it completely removes the deleterious effects of the penalty terms. However, if the projected Stokes formulation is used, there is no need to use a penalty parameter and the problem can be considered directly in the form (1).

4.3 Preconditioning Projected Systems

Although the projected Stokes system has condition number bounded by that of A, for problems with large convective terms or on meshes with elements with widely varying sizes it is necessary to use a high quality preconditioner on A. Preconditioning the generalized Stokes problem from either the left or right does not allow reducing the problem to one with coefficient matrix PAP, unless either P is redefined to be an oblique projector, or the matrix B is replaced with $S^{-1}B$, where S is related to the preconditioner. Oblique projectors are undesirable because they can cause the eigenvalues of A to spread out, and that growth is linked to the degree of obliquity between the underlying spaces. Replacing B with $S^{-1}B$ is also undesirable because it involves sparse matrix multiplication, and can result in a dense matrix.

The obvious way to precondition PAP is to form a preconditioner S for A alone, and then to use the projected system PSP as the preconditioner. Because orthogonal projectors are idempotent, this means solving the system

$$PS^{-1}PAPu = PS^{-1}Pf. \tag{8}$$

Again, the first projection can be eliminated, leaving the cost per iteration as two projection steps, one multiply by A, and one system solution with the preconditioner S. If V is a $n \times (n-q)$ matrix with columns forming an orthonormal basis for $\mathrm{null}(B^T)$, then $PSP - PAP = V^T(S-A)V$, and so a good preconditioner S to A will still be a good preconditioner to use in (8). However, the testing results will show that the projection operation can turn an inferior preconditioner into an effective one.

5. NUMERICAL TESTING

The methods were tested on a backward facing step problem in two dimensions, with varying Reynolds numbers and meshes. The characteristics of the problems,

TABLE 1: Test Problem Characteristics

Name	Velocities	Pressures	Mesh Type	Re
11992	11992	3098	Uniform	100
20284	20284	5192	Nonuniform	100
4844	4844	1270	Nonuniform	100
5936	5936	1554	Nonuniform	100
StepA3	1985	537	Adapted	100
StepA4	5019	1331	Adapted	100
StepA5	13797	3584	Adapted	100
Step4	4803	2469	Adapted	389
Step5	11099	5646	Adapted	389

TABLE 2: Storage Requirements in 10^5 Integer Words

Problem	A	B	R	Level s Precond $s=0$	$s=1$	$s=2$	EMGS Extra Space for $s=2$
11992	8.05	1.18	3.04	8.05	10.3	13.5	47.1
20284	13.8	2.01	7.52	13.8	17.9	23.7	144.7
4844	3.2	0.47	1.10	3.2	4.1	5.4	16.8
5936	3.9	0.56	1.2	3.9	5.1	6.6	16.6
StepA3	1.2	0.19	0.36	1.2	1.7	2.3	4.8
StepA4	3.2	0.48	1.04	3.2	4.9	7.0	24.2
StepA5	9.1	1.34	3.42	9.1	15.9	16.1	73.4
Step4	3.3	1.02	4.01	3.3	4.5	5.9	16.4
Step5	7.8	2.4	13.2	7.8	15.7	25.4	128.3

and the names used for them, are summarized in Tab. 1. When it is relevant, the nonlinear iteration number from which the system was extracted is indicated in parentheses after its name, e.g., StepA3(4). All problems used Crouzeix–Raviart P_2^+–P_1 elements, with the mid–element bubble node removed analytically, except for the Step problems which are unreduced. The mesh type is "Uniform", with roughly equal sized elements, "Nonuniform", with elements near one corner 100 times smaller than the largest elements, or "Adapted", being refined using the automatic adaptation methods in[16,17,18]. The last category tends to give problems that are more difficult to solve than nonuniform meshes of the same size, but can better resolve flow features using fewer elements.

The storage requirements are shown in Tab. 2. The storage required includes the data structures needed to represent the matrices as well as the nonzero values. The preconditioners were tested with up to $s = 2$ levels of fill–in allowed, and for a given level of fill both ILU and EMGS will have the same sparsity pattern $\mathcal{P}$ and so use the same amount of storage. However, EMGS requires intermediate storage not needed by ILU, and that is shown in the last column for $s = 2$ levels of fill–in. Note that in some cases the amount of additional storage needed by EMGS is impractical; these tests were run without reordering the matrices, and other experiments have shown that the cost of EMGS can be reduced by an order of magnitude by a suitable reordering. The storage needed for the Cholesky factor of $B^T B$ is shown in the column labeled R. Note that it generally requires about a third of the storage needed for A, with the exception of the STEP problems. The STEP problems do not reduce the centroid node, and so the number of pressure unknowns is roughly half the number of velocity unknowns, while for the other (reduced) problems that ratio is 25%. In practice the reduction would be applied, decreasing the number of unknowns in the Stokes problem, but here it is a useful challenge to the projected Stokes method.

The basic iterative method used was biconjugate gradients stabilized (CGSTAB)[19]. This method essentially requires two (preconditioned) matrix vector products per iteration, and was chosen because it is has been found to be reasonably robust. Because the purpose here is to examine a new preconditioner and a derived projected Stokes system, we used only this iterative method in the tests for uniformity.

5.1 Testing of Penalized Systems

Tab. 3 shows the number of iterations required to solve the penalized systems $A + rCB$ for $r = 10^5$. The unpreconditioned systems were allowed 1000 iterations, and the preconditioned ones 300 iterations. The two cases which failed with zero iterations were found to have an initial preconditioned residual norm 20 orders of magnitude larger than the unpreconditioned residual norm, indicating failure in the preconditioner. The stopping test was to reduce the initial residual norm by 6 orders of magnitude. The most striking feature is the large number of failures, which is not surprising given the effect of the penalty terms on the spectrum of the system. Without preconditioning, convergence occurred in only one case. With preconditioning, ILU converged in 8 out of 54 cases, and EMGS in 30 out of 54 cases. This does show EMGS is more effective than ILU for reducing the number of iterations needed. EMGS requires fewer iterations as the number of levels of fill–in increase, while ILU is nonmonotonic. Also note that ILU failed on every one of the STEPA problems, while EMGS succeeded except for the largest one, STEPA5. However, both methods had great difficulty with the unreduced STEP problems.

The price paid for the additional robustness of EMGS is primarily in additional temporary storage needed to compute the preconditioner, which is also reflected in the timings in Tab. 4. The one timing of 10278 seconds occurred because of underflow, which generated floating point numbers that are handled in software, not by hardware. In some cases where ILU(s) converges, it does so before EMGS(s) has completed computing its preconditioner! The conclusion is that if the systems are not too ill–conditioned and ILU works, then it is a more efficient method to use. However, if robustness is a problem, EMGS has a clear superiority.

TABLE 3: Iterations Needed For Penalized System (* indicates failure to converge)

Problem	Preconditioner						
	None	ILU(0)	ILU(1)	ILU(2)	EMGS(0)	EMGS(1)	EMGS(2)
11992(1)	1000*	300*	36	41	104	15	8
20284(1)	1000*	300*	300*	300*	300*	187	114
4844(1)	1000*	300*	300*	300*	211	111	62
5936(1)	822	300*	8	6	45	8	5
StepA3(1)	1000*	300*	300*	300*	300*	268	33
StepA4(1)	1000*	300*	300*	300*	300*	179	49
StepA5(1)	1000*	300*	300*	300*	300*	300*	300*
Step4(1)	1000*	300*	300*	300*	300*	300*	256
Step5(1)	1000*	300*	300*	0*	300*	300*	300*
11992(2)	1000*	300*	67	61	147	28	12
20284(2)	1000*	300*	300*	300*	300*	300*	280
4844(2)	1000*	300*	300*	300*	300*	144	90
5936(2)	1000*	300*	13	8	83	11	6
StepA3(4)	1000*	300*	300*	300*	300*	205	34
StepA4(4)	1000*	300*	300*	300*	300*	232	50
StepA5(4)	1000*	300*	300*	300*	300*	300*	300*
Step4(3)	1000*	300*	300*	300*	300*	300*	157
Step5(3)	1000*	300*	300*	0*	300*	300*	300*

TABLE 4: Total Time For Penalized Systems (* indicates failure to converge)

Problem	Preconditioner						
	None	ILU(0)	ILU(1)	ILU(2)	EMGS(0)	EMGS(1)	EMGS(2)
11992(1)	150.3*	105.4*	15.9	21.4	56.0	26.9	26.5
20284(1)	262.6*	184.6*	212.4*	251.8*	275.5*	237.3	210.5
4844(1)	50.1*	38.6*	45.8*	54.1*	34.3	23.8	18.8
5936(1)	56.9	48.3*	2.3	2.5	12.9	7.4	7.6
StepA3(1)	12.9*	9.5*	13.1*	17.6*	14.2*	15.8	4.1
StepA4(1)	51.3*	39.3*	51.5*	65.8*	53.6*	45.6	27.9
StepA5(1)	173.9*	119.6*	142.1*	168.7*	165.4*	190.6*	220.5*
Step4(1)	53.5*	39.0*	48.0*	56.7*	46.4*	53.9*	55.7
Step5(1)	150.0*	103.2*	158.3*	9.49*	251.9*	335.7*	418.1*
11992(2)	150.3*	104.7*	28.3	30.6	71.2	32.1	28.4
20284(2)	261.2*	183.0*	213.0*	251.9*	275.0*	316.5*	345.7
4844(2)	51.9*	38.8*	45.9*	54.0*	46.2*	28.7	23.8
5936(2)	69.1*	48.3*	3.3	3.0	18.9	8.0	7.9
StepA3(4)	12.9*	9.5*	13.7*	17.5*	14.2*	12.5	4.1
StepA4(4)	51.5*	38.8*	51.1*	65.3*	53.2*	54.2	28.1
StepA5(4)	172.0*	119.3*	142.0*	167.4*	165.5*	190.5*	220.8*
Step4(3)	51.6*	39.0*	48.0*	56.8*	45.9*	53.9*	37.4
Step5(3)	148.5*	103.4*	158.2*	9.5*	259.9*	10278.4*	433.6*

TABLE 5: Iterations Needed For Projected System (* indicates failure to converge)

Problem	Preconditioner						
	None	ILU(0)	ILU(1)	ILU(2)	EMGS(0)	EMGS(1)	EMGS(2)
11992(1)	26	33	37	40	35	37	44
20284(1)	45	35	33	38	40	39	49
4844(1)	28	20	32	27	26	29	33
5936(1)	19	27	31	34	28	32	40
StepA3(1)	325	52	43	45	54	46	43
StepA4(1)	400*	115	77	80	87	72	82
StepA5(1)	400*	132	139	146	139	145	155
Step4(1)	400*	300*	300*	297	300*	300	300*
Step5(1)	400*	300*	129	103	300*	115	115
11992(2)	159	80	69	65	84	69	60
20284(2)	192	130	100	85	140	107	87
4844(2)	113	51	45	38	51	41	48
5936(2)	105	53	47	41	52	45	40
StepA3(4)	308	51	42	41	51	44	41
StepA4(4)	400*	109	72	75	78	71	73
StepA5(4)	400*	126	134	138	130	131	122
Step4(3)	400*	300*	300*	300*	300*	300*	300*
Step5(3)	400*	300*	191	300	300*	300*	300

5.2 Testing of Projected Systems

Tab. 5 shows the number of iterations required to solve the projected system problems. The most striking feature compared to Tab. 3 is the improved success rate, with only 23 failures in 126 experiments. The failures occur only in the last two StepA problems and the Step problems, which are unlikely to be solved in practice without first reducing the mid–element node. All of the projected systems converge if $s = 3$ levels of fill–in are allowed, but those results are not shown in Tab. 5. The reason for increased robustness is obvious from the effect of the penalty term on the eigenvalues and the relation $P(A + rCB^T) = PA$, which implies that the projection removes all contributions from the penalty term. Note that this comes at the cost of storing a small matrix (the Cholesky factor of B^TB) requiring about one third the space of A itself. Again, the exception is for the Step problems, where the storage cost is up to 1.7 times the storage for A. Since the preconditioners ILU and EMGS use storage at least equal to that of A, the projection method is clearly a better use of available memory.

For problems from the first nonlinear iteration the number of iterations required can actually increase when a preconditioner is used. The adapted mesh problems start with an interpolated solution from a previous mesh (using a so–called mesh sequencing approach), and have characteristics closer to intermediate nonlinear iterations. This contrasts with the penalized formulation of the same problems, for which preconditioning reduces the number of iterations significantly. These results suggest that the projection step itself is having a favorable effect on the spectrum of the system, one which the preconditioner then partly obviates. However, for the adapted meshes further preconditioning is necessary to achieve convergence.

EMGS for many of the problems is less effective at reducing the number of iterations than ILU is. This generally holds regardless of the mesh. Because EMGS is generally better at reducing the number of iterations for penalized systems and systems with coefficient matrix A alone, this suggests that EMGS reduces eigencomponent contributions from the range of B more strongly than ILU does, while ILU has its preconditioning strength concentrated in the null space of B^T, which is what is needed for projected Stokes systems. Another possibility is that ILU introduces a few eigenvectors with large magnitudes, which the projection step eliminates.

For the highly adapted mesh in StepA5, increasing levels of fill–in produce no improvement in the quality of preconditioning. This could be expected in going from $s = 1$ to $s = 2$ levels since the increase in storage (and hence in elements retained) is less than 2%. However, going from $s = 0$ to $s = 1$ entails a 75% increase in storage which usually implies an improvement in preconditioning.

Tab. 6 shows the time required to solve the projected systems. As with the penalized systems, the EMGS preconditioned systems take longer, but the effect is less pronounced here because some of the work per iteration has shifted to computing the projections, which are needed for both the ILU and EMGS preconditioned systems.

In general, the times indicate that if the projected Stokes method converges without preconditioning on A, then it is better not to apply preconditioning – the one exception is StepA3 on the first time step. Furthermore, when compared to the times in Fig. 4, the projection system can be solved much faster in general. On only two out of the fourteen test problems did a preconditioned iterative method work faster on the penalized system than on the projected system, and in those two cases the times are comparable.

Tab. 7 shows the time required to factor B^TB. Note that it is insignificant compared to the overall solution time. Furthermore, as noted earlier this need only be done once for a given mesh — although A changes from one nonlinear iteration to the next, the matrix B is constant.

Finally, it should be noted that in all the problems, the divergence norm $\|B^Tu\|$ remained less than 10^{-12}. If many iterations are made, the divergence norm does slowly grow. However, following the iterative method with a single projection on the velocity vector u restores the divergence norm to less than 10^{-14} for all the test problems examined here.

6. FURTHER WORK

Although success rates of 56% for EMGS applied to penalized systems or 82% for the projected Stokes method may sound abysmal, this should be contrasted with the 18% success rate of ILU applied to penalized systems. Nevertheless, the most

TABLE 6: Total Time For Projected Systems (* indicates failure to converge)

Problem	Preconditioner						
	None	ILU(0)	ILU(1)	ILU(2)	EMGS(0)	EMGS(1)	EMGS(2)
11992(1)	7.7	21.0	26.2	31.5	40.6	45.5	54.7
20284(1)	24.9	42.6	44.9	57.3	136.5	155.9	181.9
4844(1)	3.0	5.0	8.5	8.5	11.8	14.2	16.9
5936(1)	2.6	7.8	10.0	12.3	12.8	15.3	19.6
StepA3(1)	8.9	3.6	3.7	4.5	5.1	5.4	5.8
StepA4(1)	39.6*	26.0	20.5	25.5	33.2	34.1	42.3
StepA5(1)	125.5*	92.1	106.5	125.9	142.5	161.0	187.9
Step4(1)	70.2*	109.6*	118.6*	126.1	115.3*	125.0	133.9*
Step5(1)	210.4*	326.8*	168.6	160.6	471.8*	326.9	373.8*
11992(2)	43.5	48.7	46.5	48.9	70.0	65.8	65.8
20284(2)	101.1	149.2	126.2	119.7	248.7	239.0	232.1
4844(2)	11.2	11.5	11.6	11.2	17.0	17.1	20.7
5936(2)	13.3	14.6	14.8	14.6	19.2	19.2	19.6
StepA3(4)	8.4	3.6	3.6	4.1	4.9	5.3	5.7
StepA4(4)	39.8*	24.8	19.9	24.1	31.8	34.2	39.3
StepA5(4)	124.1*	87.6	102.0	117.6	134.7	148.6	157.1
Step4(3)	70.2*	109.4*	118.6*	126.9*	115.0*	124.7*	133.4*
Step5(3)	211.1*	329.0*	247.8	442.3	473.9*	559.7*	638.5

TABLE 7: Time to Factor $B^T B$

Problem	11992	20284	4844	5936	StepA3	StepA4	StepA5
Time(sec)	0.35	1.21	0.11	0.11	0.02	0.11	0.43

important goal for a practical solver is robustness. The preconditioned iterative methods can succeed on all of these problems if enough levels of fill–in are allowed, because $s = n$ levels yields the direct factorization of the system. However, that eliminates their primary advantage – reduced storage costs.

One disappointment is that EMGS does no better at preconditioning the projected systems than ILU does. Forming a preconditioner S for A and then applying it as a projected operator $PS^{-1}P$ is a crude approach, since the preconditioner only need be effective on the null space of B^T. That suggests developing methods which target only the smaller space, and incomplete orthogonalization preconditioning seems more adaptable for this by only enforcing (partial) orthogonality against vectors in that space.

EMGS and the projection method work better on generalized Stokes systems from fluid flow problems than they do on more general systems of the form (1). Preliminary analyses suggest this is because of the matrix B^TB corresponds to a diffusion operator on the pressure unknowns, and A contains a smaller diffusion component in more unknowns. More analysis needs to be carried out to further delineate this relation, and this analysis will also help guide the development of a preconditioner more specifically targeted for the null space of B^TB.

The relatively poor performance of the projected method on the unreduced STEP problems shows, unsurprisingly, that the type of finite element used can greatly affect the performance of the iterative solver. Another future direction of research is an examination of the connection between the element used and the iterative solver applied.

7. CONCLUSIONS

A new preconditioner, derived from incomplete orthogonalization factorizations, has been proven to exist for any matrix for which Gaussian elimination without pivoting will work. In particular, this applies to systems which have been traditionally solved using skyline methods. Furthermore, for M–matrices, the preconditioner will provide a better estimate of the LU factors of the coefficient matrix. Numerical testing shows the method to be more robust and provide better preconditioning for a given amount of storage than standard incomplete LU factorization does. However, EMGS requires a large amount of intermediate storage and can take significantly longer to compute than a ILU preconditioner.

A generalized Stokes solver based on orthogonal projectors has also been examined, and has proven more robust than iterative methods applied to penalized systems. For fluid dynamicists with codes that use a penalty or augmented Lagrangian formulation the projection method can still be used and it completely eliminates contributions from penalty terms. Those terms are the source of eigenvalue distributions which prevent most iterative methods from being used. Furthermore, the projected system requires an amount of storage roughly equal to a third of that used for the coefficient matrix for reduced problems, and the computation of the auxillary data needed to implement the projections can be done once and then used for all nonlinear iterations. It can be viewed as a preconditioner for Stokes problems, one which maintains null divergence to within machine precision while simultaneously accelerating convergence.

The results presented here show that orthogonalization based methods for fluid flow problems are potentially the most effective iterative schemes to use, and provide a totally new way of deriving and analyzing algorithms for incompressible fluid flows.

REFERENCES

1. M. Hestenes and E. Stiefel (1952), *Methods of conjugate gradients for solving linear systems*, Journal of Research, National Bureau of Standards, **49**, 409–436.
2. R. Bramley and A. Sameh (1992), *Row projection methods for large nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., **13** (1), 168–193.
3. R. Barrett et al (1994), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, First edition.
4. M. Fortin and R. Glowinski (1983), *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary Value Problems*, North Holland, New York.
5. A. Jennings and M.A. Ajiz (1984), *Incomplete methods for solving* $A^TAx = b$, SIAM Journal of Scientific and Statistical Computing, **5** (4), 978–987.
6. Y. Saad (1988), *Preconditioning techniques for nonsymmetric and indefinite linear systems*, Journal of Computational and Applied Mathematics, **24**, 89–105.
7. Z. Zlatev and H.B.Nielson (1988), *Solving large and sparse linear least squares problems by conjugate gradient algorithm*, Computers and Mathematics with Applications, **15**, 185–202.
8. D. James (1990), *Conjugate Gradient Methods for Constrained Least-Squares Problems*, PhD thesis, North Carolina State University.
9. X. Wang (1993), *Incomplete Factorization Preconditioning for Linear Least Squares Problems*, PhD thesis, University of Illinois Urbana-Champaign. Also available as Tech. Rep. UIUCDCS-R-93-1834, Computer Science Department, University of Illinois – Urbana.
10. X. Wang, K.A. Gallivan and R. Bramley (1994), *CIMGS: A incomplete orthogonalization preconditioner*, Technical Report 393, Indiana University–Bloomington, Submitted for CCIM Special Issue of SIAM J. Sci. Comp.
11. J.A. Meijerink and H.A. van der Vorst (1977), *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., **31** (137), 148–162.
12. R. Bramley (1992), *An orthogonal projection algorithm for linear Stokes problems*, Technical Report 1190, Center for Supercomputing Research and Development, University of Illinois, Urbana, Illinois.
13. J.P. Chabard (1989), *N3S Code for Fluid Mechanics, Theoretical Manual, version 2.0*, Electricité de France, Report Number HE-41/89.14.
14. E. Chu, A. George, J. Liu and E. Ng (1984), *SPARSPAK: Waterloo sparse matrix package user's guide for SPARSPAK-A*, Technical Report CS-84-36, Department of Computer Science, University of Waterloo.
15. J.M. Ortega and W.C. Rheinboldt (1970), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.
16. D. Pelletier and J-F. Hétu (1992), *An adaptive finite element methodology for incompressible viscous flow*, In Advances in Finite Element Analysis in Fluid Dynamics II, ASME Winter Annual Meeting, ASME.
17. J-F. Hétu and D. Pelletier (1992), *Fast, adaptive finite element scheme for viscous incompressible flow*, AIAA Journal, **30** (11), 2677–2681.
18. J-F. Hétu and D. Pelletier (1992), *Adaptive remeshing for viscous incompressible flow*, AIAA Journal, **30** (8), 1986–1992.

19. H.A. van der Vorst (1992), *Bi–CGSTAB: A fast and smoothly converging variant of Bi–CG for the solution of nonsymmetric linear systems*, SIAM Journal of Scientific and Statistical Computing, **13**, 631–644.