Approximate Inverse Techniques for Block-Partitioned Matrices*

Edmond Chow and Yousef Saad

Department of Computer Science, and

Minnesota Supercomputer Institute

University of Minnesota

Minneapolis, MN 55455

January 28, 1995

Abstract

This paper proposes some preconditioning options when the system matrix is in block-partitioned form. This form may arise naturally, for example from the incompressible Navier-Stokes equations, or may be imposed after a domain decomposition reordering. Approximate inverse techniques are used to generate sparse approximate solutions whenever these are needed in forming the preconditioner. The storage requirements for these preconditioners may be much less than for ILU preconditioners for tough, large-scale CFD problems. The numerical experiments show that these preconditioners can help solve difficult linear systems whose coefficient matrices are highly indefinite.

Key words. preconditioning, sparse approximate inverse, block-partitioned matrix, Schur complement, Navier-Stokes

*Work supported in part by the National Science Foundation under grant NSF/CCR-9214116 and in part by NASA under grant NAG2-904.

1 Introduction

Consider the block partitioning of a matrix A, in the form

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \tag{1}$$

where the blocking naturally occurs due the ordering of the equations and the variables. Matrices of this form arise in many applications, such as in the incompressible Navier-Stokes equations, where the scalar momentum equations and the continuity condition form separate blocks of equations. In the 2-D case, this is a system of the form

$$A = \begin{pmatrix} B_{uu} & B_{uv} & F_{up} \\ B_{vu} & B_{vv} & F_{vp} \\ E_{pu} & E_{pv} & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ p \end{pmatrix} = \begin{pmatrix} f_u \\ f_v \\ f_p \end{pmatrix} \tag{2}$$

where u and v represent the velocity components, and p represents the pressure. Here, the B submatrix is a convection-diffusion operator, the F submatrices are pressure gradient operators, and the E submatrices are velocity divergence operators.

Traditional techniques such as the Uzawa algorithm have been used for these problems, often because the linear systems that must be solved are much smaller, or because there are zeros or small values on the diagonal of the fully-coupled system. These so-called segregated approaches, however, suffer from slow convergence rates when compared to aggregated, or fully-coupled solution techniques.

Another source of partitioned matrices of the form (1) is the class of domain decomposition methods. In these methods the interior nodes of a subdomain are ordered consecutively, subdomain after subdomain, followed by the interface nodes ordered at the end. This ordering of the unknowns gives rise to matrices which have the following structure:

$$\begin{pmatrix} B_1 & & & & F_1 \\ & B_2 & & & F_2 \\ & & \ddots & & \vdots \\ & & & B_n & F_n \\ E_1 & E_2 & \cdots & E_n & S \end{pmatrix}. \tag{3}$$

Typically, the linear systems associated with the B matrix produced by this reordering are easy to solve, being the result of restricting the original PDE problem into a set of independent and similar PDE problems on much smaller meshes. One of the motivations for this approach is parallelism. This approach ultimately requires solution methods for the Schur complement S. There is a danger, however, that for general matrices, B may be singular after the reordering.

Much work has been done on exploiting some form of blocking in conjunction with preconditioning. In one of the earlier papers on the subject, Concus, Golub, and Meurant

[7] introduce the idea of block preconditioning, designed for block-tridiagonal matrices whose diagonal blocks are tridiagonal. The inverses of tridiagonal matrices encountered in the approximations are themselves approximated by tridiagonal matrices, exploiting an exact formula for the inverse of a tridiagonal matrix. This was later extended to the more general case where the diagonal blocks are arbitrary [4, 17]. In many of these cases, the incomplete block factorizations are developed for matrices arising from the discretization of PDE's [2, 3, 7, 17, 19] and utilize approximate inverses when diagonal blocks need to be inverted. More recently, Elman and Silvester [13] proposed a few techniques for the specific case of the Stokes and Navier-Stokes problems. A number of variations of Block-Jacobi preconditioners have also been developed [1, 9]. In these techniques the off-block diagonal terms are either neglected or an attempt is made to approximate their effect.

This paper explores some preconditioning options when the matrix is expressed in block-partitioned form, either naturally or after some domain decomposition type reordering. The iterative method acts on the fully-coupled system, but the preconditioning has some similarity to segregated methods. This approach only requires preconditioning or approximate solves with submatrices, where the submatrices correspond to any combination of operators, such as reaction, diffusion, and convection. It is particularly advantageous to use the block-partitioned form if we know enough about the submatrices to apply specialized preconditioners, for example operator-splitting and semi-discretization, as well as lower-order discretizations.

Block-partitioned techniques also require the sparse approximate solution to sparse linear systems. These solutions need to be sparse because they form the rows or columns of the preconditioner, or are used in further computations. Dense solutions here will cause the construction or the application of the preconditioner to be too expensive. This problem is ideally suited for sparse approximate inverse techniques. The approximate solution to the sparse system $Ax = b$ is found by

$$\min_x \|b - Ax\|_2$$

using an iterative method implemented with sparse matrix–sparse vector and sparse vector–sparse vector operations. The intermediate and final solutions are forced to be sparse by numerically dropping elements in x with small magnitudes. If the right-hand-side b and the initial guess for x are sparse, this is a very economical method for computing a sparse approximate solution. We have used this technique to construct preconditioners based on approximating the inverse of A directly [6].

This paper is organized as follows. In Section 2 we describe the sparse approximate inverse algorithm and some techniques for finding sparse approximate solutions with the Schur complement. Section 3 describes how block-partitioned factorizations may be used as preconditioners. The most effective of these are the approximate block LU factorization and the approximate block Gauss-Seidel preconditioner. Section 4 reports the results of several numerical experiments, including the performance of the new preconditioners on problems arising from the incompressible Navier-Stokes equations.

2 Sparse approximate inverses and their use

It is common when developing preconditioners based on block techniques to face the need to compute an approximation to the inverse of a sparse matrix or an approximation to columns of the form $B^{-1}f$ in which both B and f are sparse. This is particularly the case for block preconditioners for block-tridiagonal matrices [7, 19]. For these algorithms to be practical, they must provide approximations that are sparse.

A number of techniques have recently been developed to construct a sparse approximate inverse of a matrix, to be used as a preconditioner [5, 6, 8, 10, 15, 17, 18]. Many of these techniques approximate each row or column independently, focusing on (in the column-oriented case) the individual minimizations

$$\min_x \|e_j - Ax\|_2, \qquad j = 1, 2, \ldots, n \tag{4}$$

where e_j is the j-th column of the identity matrix. Such a preconditioner is distinctly easier than most existing preconditioners to construct and apply on a massively parallel computer. Because they do not rely on matrix factorizations, these preconditioners often are complementary to ILU preconditioners [6, 22].

Previous approaches select a sparsity pattern for x and then minimize (4) in a least squares sense. In our approach, we minimize (4) with a method that reduces the residual norm at each step, such as Minimal Residual or FGMRES [20], beginning with a sparse initial guess. Sparsity is preserved by dropping elements in the search direction or current solution at each step based on their magnitude or criteria related to the residual norm reduction. The final number of nonzeros in each column is guaranteed to be not more than the parameter *lfil*. In the case of FGMRES, the Krylov basis is also kept sparse by dropping small elements. To keep the iterations economical, all computations are performed with sparse matrix–sparse vector or sparse vector–sparse vector operations.

For our application here, we point out that the approximate inverse technique for each column may be generalized to find a sparse approximate solution to the sparse linear problem $Ax = b$ by minimizing

$$\min_x \|b - Ax\|_2 \tag{5}$$

possibly with an existing preconditioner M for A.

2.1 Approximate inverse algorithm

We describe a modification of the technique reported in [6] that guarantees the reduction of the residual norm at each minimal residual step. Starting with a sparse initial guess, the fill-in is increased by one at each iteration. At the end of each iteration, it is possible to use a second stage that exchanges entries in the solution with new entries if this causes a reduction in the residual norm. Without the second stage, entries in the solution cannot be annihilated once they have been introduced. For the problems in this paper, however, this second stage has not been necessary.

In the first stage, the search direction d is derived by dropping entries from the residual direction r. So that the sparsity pattern of the solution x is controlled, d is chosen to have the same sparsity pattern as x, plus one new entry, the largest entry in absolute value. Minimization is performed by choosing the steplength

$$\alpha = \frac{(r, Ad)}{(Ad, Ad)}$$

and thus the residual norm for the new solution is guaranteed to be not more than the previous residual norm. The solution and the residual is updated at the end of this stage. If A is indefinite, the normal equations residual direction $A^T r$ may be used as the search direction, or simply to determine the location of the new fill-in. It is interesting to note that the largest entry in $A^T r$ gives the greatest residual norm reduction in a one-dimensional minimization. This explains why a transpose initial guess for the approximate inverse combined with self-preconditioning (preconditioning r with the current approximate inverse) is so effective for some problems [6].

There are many possibilities for the second stage. We choose to drop one entry in x and introduce one new entry in d if this causes a decrease in the residual norm. The candidate for dropping is the smallest absolute nonzero entry in x. The candidate to be added is the largest absolute entry in the previous search direction (at the beginning of stage 1) not already included in d. The previous direction is used so that the candidate may be determined in stage 1, and an additional search is not required. The steplength β is chosen by minimizing the new residual norm

$$\|b - A(x - x_s e_s + \beta e_l)\|_2$$

where e_i is the i-th coordinate vector, x_s is the entry in x to be dropped at position s (smallest), while β is the entry to be added at position l (largest), and we have generalized the notation so that b is the right-hand-side vector, previously denoted m_j. Let A_j denote the j-th column of A. Then the minimization gives

$$\beta = \frac{(b - Ax + x_s A_s, A_l)}{(A_l, A_l)}$$

which just involves one sparse SAXPY since $b - Ax$ is already available as r, and one sparse dot-product, since we may scale the columns of A to have unit 2-norm. It is guaranteed that $s \neq l$ since l is chosen from among the entries not including s.

The preconditioned version of the algorithm for minimizing $\|b - Ax\|_2$ with explicit preconditioner M may be summarized as follows. A is assumed to be scaled so that its columns all have unit 2-norm. The number of inner iterations is usually chosen to be *lfil* or somewhat larger.

ALGORITHM **2.1 Approximate inverse algorithm**

1. *Starting with some initial guess* x, $r := b - Ax$
2. *For* $inner = 1, 2, \ldots, n_i$ *do*

Stage 1

3. $t := Mr$
4. *Choose* d *to be* t *with the same pattern as* x*;*
   If $nnz(x) <$ lfil *then add one entry which is the*
   largest remaining entry in absolute value
5. $q := Ad$
6. $\alpha := \frac{(r,q)}{(q,q)}$
7. $r := r - \alpha q$
8. $x := x + \alpha d$

Stage 2

9. $s :=$ *index of smallest nonzero in abs*(x)
10. $l :=$ *index of largest nonzero in abs*$(t - d)$
11. $\beta := (r + x_s A_s, A_l)$
12. $\tilde{r} := r + x_s A_s - \beta A_l$
13. *If* $\|\tilde{r}\| < \|r\|$ *then*
14. *Set* $x_s := 0$ *and* $x_l := \beta$
15. $r := \tilde{r}$
16. *End if*
17. *End do*

2.2 Sparse solutions with the Schur complement

Sparse approximate solutions with the Schur complement $S = C - EB^{-1}F$ are often required in the preconditioning for block-partitioned matrices. We will briefly describe three approaches in this section: (1) approximating S, (2) approximating S^{-1}, and (3) exploiting a partial approximate inverse of A.

2.2.1 Approximating S

To approximate S with a sparse matrix, we can use

$$\tilde{S} = C - EY, \quad Y \approx B^{-1}F \ , \tag{6}$$

where Y is computed by the approximate inverse technique, possibly preconditioned with whatever we are using to solve with B. Since Y is sparse, $\tilde{S}$ computed this way is also sparse. Moreover, since S is usually relatively dense, solving with $\tilde{S}$ is an economical approach. Typically, a zero initial guess is used for Y. We remark that it is usually too expensive to form Y by solving $B^{-1}F$ approximately and then dropping small elements, since it is rather costly to search for elements to drop. We also note that we can generate $\tilde{S}$ column-by-column, and if necessary, compute a factorization of $\tilde{S}$ on a column-by-column basis as well. The linear systems with $\tilde{S}$ can be solved in any fashion, including with an iterative process with or without preconditioning.

2.2.2 Approximating S^{-1}

Another method is to compute an approximation to S^{-1} using the idea of induced preconditioning. Since S^{-1} is the (2,2) block of

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix}^{-1} = \begin{pmatrix} B^{-1} + B^{-1}FS^{-1}EB^{-1} & -B^{-1}FS^{-1} \\ -S^{-1}EB^{-1} & S^{-1} \end{pmatrix} \tag{7}$$

we can compute a sparse approximation to it by using the approximate inverse technique applied to the last block-column of A and then throwing away the upper block. In practice, the upper part of each column may be discarded before computing the next column. In our experiments, since the approximate inverse algorithm is applied to A, an indefinite matrix in most of the problems, the normal equations search direction $A^T r$ is used in the algorithm, with a scaled identity initial guess for the inverse.

2.2.3 Partial approximate inverse

A drawback of the above approach is that the top submatrix of the last block-column is discarded, and that the resulting approximation of S^{-1} may actually contain very few nonzeros. A related technique is to compute the partial approximate inverse of A in the last block-row. This technique does not give an approximation to S^{-1}, but defines a simple preconditioning method itself. Writing the inverse of A in the form,

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix}^{-1} \approx \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} \tag{8}$$

we can then get an approximate solution to $A\binom{x}{y} = \binom{f}{g}$ with

$$\begin{aligned} y &= M_2 \begin{pmatrix} f \\ g \end{pmatrix} \\ x &= B^{-1}(f - Fy). \end{aligned} \tag{9}$$

It is not necessary to solve accurately with B. Again, the normal equations search direction is used for the approximate inverse algorithm in the numerical experiments. Some results of this relatively inexpensive method will be given in Section 4.

3 Block-partitioned factorizations of A

We consider a sparse linear system

$$Au = b \tag{10}$$

which is put in the block form,

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{11}$$

For now the only condition we require on this partitioning is that B be nonsingular. We use extensively the following block LU factorization of A,

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} B & 0 \\ E & S \end{pmatrix} \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix} \tag{12}$$

in which S is the Schur complement,

$$S = C - EB^{-1}F. \tag{13}$$

As is well-known, we can solve (12) by solving the reduced system,

$$Sy = g' \quad \text{with} \quad g' = g - EB^{-1}f \tag{14}$$

to compute y, and then back-substitute in the first block-row of the system (11) to obtain x, i.e., compute x by

$$x = B^{-1}(f - Fy) .$$

The above block structure can be exploited in several different ways to define preconditioners for A. Thus, the block preconditioners to be defined in this section combine one of the preconditioners for S seen in Section 2.2 and a choice of a block factorization. Next, we describe a few such options.

3.1 Solving the preconditioned reduced system

A method that is often used is to solve the reduced system (14), possibly with the help of a certain preconditioner M_S for the Schur complement matrix S. Although this does not involve any of the block factorizations discussed above, it is indirectly related to it and to other well-known algorithms. For example, the Uzawa method which is typically formulated on the full system, can be viewed as a Richardson (or fixed point) iteration applied to the reduced system. The matrix S need not be computed explicitly; instead, one can perform the matrix-vector product $w = Sv$, with the matrix S, via the following sequence of operations:

1. Compute $t := Fv$;

2. Solve $Bu = t$;

3. Compute $w := Cv - Eu$.

If we wish to use a Krylov subspace technique such as GMRES on the preconditioned reduced system, we need to solve the systems in Step 2, exactly, i.e., by a direct solver or an iterative solver requiring a high accuracy. This is because the S matrix is the coefficient matrix of the system to be solved, and it must be constant throughout the GMRES iteration. We have experimented with this approach and found that this is a serious limitation. Convergence is reached in a number of steps which is typically comparable with that obtained with methods based on the full matrix. However, each step costs much more, unless a direct solution technique is used, in which case the initial LU factorization may be very expensive. Alternatively, a highly accurate ILU factorization can be employed for B, to reduce the cost of the many systems that must be solved with it in the successive outer steps.

3.2 Approximate block diagonal preconditioner

One of the simplest block preconditioners for a matrix A partitioned as in (1) is the block–diagonal matrix

$$M = \begin{pmatrix} B & 0 \\ 0 & M_C \end{pmatrix} \tag{15}$$

in which M_C is some preconditioning for the matrix C. If $C = 0$ as is the case for the incompressible Navier-Stokes equations, then we can define $M_C = I$ for example. An interesting particular case is when C is nonsingular and $M_C = C$. This corresponds to a block-Jacobi iteration. In this case, we have

$$I - M^{-1}A = \begin{pmatrix} 0 & B^{-1}F \\ C^{-1}E & 0 \end{pmatrix}$$

the eigenvalues of which are the square roots of the eigenvalues of the matrix $C^{-1}EB^{-1}F$. Convergence will be fast if all these eigenvalues are small.

3.3 Approximate block LU factorization

The block factorization (12) suggests using preconditioners based on the block LU factorization

$$M = LU$$

in which

$$L = \begin{pmatrix} B & 0 \\ E & M_S \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix}$$

to precondition A. Here M_S is some preconditioner to the Schur complement matrix S. If we had a sparse approximation $\tilde{S}$ to the Schur complement S we could compute a preconditioning matrix M_S to $\tilde{S}$, for example, in the form of an approximate LU factorization. We must point out here that any preconditioner for S will induce a preconditioner for A. As was discussed in Section 3.1 a notable disadvantage of an approach based on solving the reduced system (14) by an iterative process is that the action of S on a vector must be computed very accurately in the Krylov acceleration part. In an approach based on the larger system (11) this is not necessary. In fact any iterative process can be used for solving with M_S and B provided we use a flexible variant of GMRES such as FGMRES [20].

Systems involving B may be solved in many ways, depending on their difficulty and what we know about B. If B is known to be well-conditioned, then triangular solves with incomplete LU factors may be sufficient. For more difficult B matrices, the incomplete factors may be used as a preconditioner for an inner iterative process for B. Further, if the incomplete factors are unstable (see Section 4.2), an approximate inverse for B may be used, either directly or as a preconditioner. If B is an operator, an approximation to it may be used; its factors may again be used either directly or as a preconditioner. This kind of flexibility is typical of what is available for using iterative methods on block-partitioned matrices.

An important observation is that if we solve exactly with B then the error in this block ILU factorization lies entirely in the (2,2) block since,

$$A = LU + \begin{pmatrix} 0 & 0 \\ 0 & S - M_S \end{pmatrix} . \tag{16}$$

One can raise the question as to whether this approach is any better than one based on solving the reduced system (14) preconditioned with M_S. It is known that in fact the two approaches are mathematically equivalent if we start with the proper initial guesses. Specifically, the initial guess should make the x-part of the residual vector equal to 0 for the original system (11), i.e., the initial guess is

$$u_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \quad \text{with} \quad x_0 = B^{-1}(f - F y_0) \ .$$

This result, due to Eisenstat and reported in [16], immediately follows from (16) which shows that the preconditioned matrix has the particular form,

$$(LU)^{-1} A = \begin{pmatrix} I & 0 \\ 0 & M_S^{-1} S \end{pmatrix} . \tag{17}$$

Thus, if the initial residual has its x-component equal to zero then all iterates will be vectors with y components only, and a GMRES iteration on the system will reduce to a GMRES iteration with the matrix $M_S^{-1} S$ involving only the y variable.

There are many possible options for choosing the matrix M_S. Among these we consider the following ones.

- $M_S = I$ – no preconditioning on S.
- $M_S = C$ – precondition with the C matrix if it is nonsingular. Alternatively we can precondition with an ILU factorization of C.
- $M_S \approx S$ – construct a sparse approximation to S and use it as a preconditioner. In general, we only need to approximate the action of S on a vector, for example, with the methods described in Sections 2.2.1 and 2.2.2.

The following algorithm applies one preconditioning step to $\binom{f}{g}$ to get $\binom{x}{y}$.

Algorithm **3.1 Approximate block LU preconditioning**

1. $x := B^{-1}f$
2. $y := M_S^{-1}(g - Ex)$
3. $x := x - B^{-1}Fy$

We have experimented with a number of options for solving systems with M_S in step 2 of the algorithm above. For example, M_S may be approximated with $\tilde{S} = C - EY$, where $Y \approx B^{-1}F$ is computed by the approximate inverse technique. If this approximation is used, it is possible to also use Y in place of $B^{-1}F$ in step 3.

3.4 Approximate block Gauss-Seidel

By ignoring the U factor of the approximate block LU factorization, we are led to a form of block Gauss-Seidel preconditioning, defined by $M = L$, i.e.,

$$M = \begin{pmatrix} B & 0 \\ E & M_S \end{pmatrix}. \tag{18}$$

The same remarks on the ways to solve systems with B and ways to define the preconditioning matrix M_S apply here. The algorithm for this preconditioner is the same as Algorithm 3.1 without step 3.

To analyze the preconditioner, we start by observing that

$$M^{-1}A = \begin{pmatrix} I & B^{-1}F \\ 0 & M_S^{-1}S \end{pmatrix}, \tag{19}$$

showing that the only difference with the preconditioned matrix (17) is the additional block $B^{-1}F$ in the (1,2) position. The iterates associated with the block form and those of the associated Schur complement approach $M_S^{-1}Sy = g'$ are no longer simply related. However there are a few connections between (17) and (19). First, the spectra of the two matrices are identical. This does not mean, however, that the two matrices will require the same number of iterations to converge in general.

Consider a GMRES iteration to solve the preconditioned system $M^{-1}Au = M^{-1}b$. Here, we take an initial guess of the form

$$u_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{20}$$

in which x_0 is arbitrary. With this we denote the preconditioned initial residual by

$$r_0 = M^{-1}(b - Au_0) = \begin{pmatrix} z_0 \\ s_0 \end{pmatrix} .$$

Then GMRES will find a vector u of the form $u = u_0 + w$, with w belonging to the Krylov subspace

$$K_m(M^{-1}A, r_0) = \text{span}\{r_0, M^{-1}Ar_0, \dots, (M^{-1}A)^{m-1}r_0\} ,$$

which will minimize $\|M^{-1}(b - Au)\|_2$. For an arbitrary u in the affine space $u_0 + K_m(M^{-1}A, r_0)$, i.e.,

$$u = u_0 + w, \quad u_0 \equiv \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \quad w \equiv \begin{pmatrix} \delta \\ \eta \end{pmatrix}$$

the preconditioned residual is of the form

$$M^{-1}(b - Au) = M^{-1}(b - A(u_0 + w)) = M^{-1}(r_0 - Aw)$$

and by (19) this becomes,

$$M^{-1}(b - Au) = \begin{pmatrix} z_0 \\ s_0 \end{pmatrix} - \begin{pmatrix} \delta + B^{-1}F\eta \\ M_S^{-1}S\eta \end{pmatrix} .$$

As a result,

$$\|M^{-1}(b - Au)\|_2^2 = \|s_0 - M_S^{-1}S\eta\|_2^2 + \|z_0 - \delta - B^{-1}F\eta\|_2^2 . \tag{21}$$

Note that $\|s_0 - M_S^{-1}S\eta\|_2$ represents the preconditioned residual norm for the reduced system for the y obtained from the approximation of the large system. We have

$$\|M^{-1}(b - Au)\|_2 \geq \|s_0 - M_S^{-1}S\eta\|_2$$

which implies that if the residual for the bigger system is less than ϵ, then the residual obtained by using a full GMRES on the associated preconditioned reduced system $M_S^{-1}S\eta = M_S^{-1}g'$ will also be less than ϵ. We observe in passing that the second term in the right-hand-side of (21) can always be reduced to zero by a post-processing step which consists of forcing the first part of the residual to be zero by changing δ (only) into:

$$\delta = z_0 - B^{-1}F\eta .$$

Equivalently, once the current pair x, y is obtained, x can be recomputed by satisfying the first block equation, i.e.,

$$x = B^{-1}(f - Fy) \ .$$

This post-processing step requires only one additional B solve.

Assume now that we know something about the residual vector associated with m steps of GMRES applied to the preconditioned reduced system. Can we say something about the residual norm associated with the preconditioned unreduced system? We begin by establishing a simple lemma.

Lemma 3.1 *Let*

$$Z = \begin{pmatrix} I & Y \\ 0 & G \end{pmatrix} \ . \tag{22}$$

Then, the following equality holds

$$(I - Z)Z^k = \begin{pmatrix} 0 & -YG^k \\ 0 & (I-G)G^k \end{pmatrix} \tag{23}$$

Proof. First, it is easy to prove that

$$Z^k = \begin{pmatrix} I & Y_k \\ 0 & G^k \end{pmatrix}$$

in which $Y_k = Y[I + G + \cdots G^{k-1}]$. We now multiply both members of the above equality $I - Z$ to obtain,

$$(I - Z)Z^k = \begin{pmatrix} 0 & -Y \\ 0 & I - G \end{pmatrix} \begin{pmatrix} I & Y_k \\ 0 & G^k \end{pmatrix} = \begin{pmatrix} 0 & -YG^k \\ 0 & (I-G)G^k \end{pmatrix}$$

□

We now state the main result concerning the comparison between the two approaches.

Theorem 3.1 *Assume that the reduced system (14) is solved with GMRES using the preconditioner M_S starting with an arbitrary initial guess y_0 and let $s_m = M_S^{-1}(g' - Sy_m)$ the preconditioned residual obtained at the m-th step. Then the preconditioned residual vector r_{m+1} obtained at the $(m+1)$-st step of GMRES for solving the block system (11) preconditioned with the matrix M of (18) and with an initial guess $u_0 = \binom{x_0}{y_0}$ in which x_0 is arbitrary satisfies the inequality*

$$\|r_{m+1}\|_2 \leq \|I - M^{-1}A\|_2 \|s_m\|_2. \tag{24}$$

In particular if $s_m = 0$ then $r_{m+1} = 0$.

Proof. The preconditioned matrix for the unreduced system is of the form (22) with $Y \equiv B^{-1}F$ and $G \equiv M_S^{-1}S$. The residual vector s_m of the m-th GMRES approximation associated with the reduced system is of the form,

$$s_m = \rho_m(G)s_0$$

in which ρ_m is the m-th residual polynomial, which minimizes $\|p(G)s_0\|_2$ among all polynomials p of degree m satisfying the constraint: $p(0) = 1$. Let

$$\rho_m(t) \equiv \sum_{i=0}^{m} \alpha_i t^i \ .$$

Consider the polynomial of degree $m+1$ defined by

$$\beta_{m+1}(t) = (1-t)\rho_m(t). \tag{25}$$

It is clear that

$$\beta_{m+1}(0) = 1 \ .$$

The residual of u_{m+1}, the $m+1$-st approximate solution obtained by the GMRES algorithm for solving the preconditioned unreduced system minimizes $p(Z)r_0$ over all polynomials p of degree $m+1$ which are consistent, i.e., such that $p(0) = 1$. Therefore,

$$\|r_{m+1}\|_2 \leq \|\beta_{m+1}(Z)r_0\|_2 \ .$$

Using the equality established in the lemma, we now observe that

$$\begin{aligned}
\beta_{m+1}(Z) &= (I-Z)\sum_{i=0}^{m} \alpha_i Z^i \\
&= \sum_{i=0}^{m} \alpha_i (I-Z) Z^i \\
&= \sum_{i=0}^{m} \alpha_i \begin{pmatrix} 0 & -YG^i \\ 0 & (I-G)G^i \end{pmatrix} \\
&= \begin{pmatrix} 0 & -Y\rho_m(G) \\ 0 & (I-G)\rho_m(G) \end{pmatrix} \\
&= \begin{pmatrix} 0 & -Y \\ 0 & I-G \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & \rho_m(G) \end{pmatrix}.
\end{aligned}$$

The first matrix in the right-hand-side of the last equality is nothing but $I - Z$. Hence, the residual vector r_{m+1} is such that

$$\|r_{m+1}\|_2 \leq \|I-Z\|_2 \|\rho_m(G)s_0\|_2$$

which completes the proof. $\square$

It is also interesting to relate the convergence of this algorithm to that of the block-diagonal approach in the particular case when $M_S = C$. This case corresponds to a block Gauss-Seidel iteration. We can exploit Young and Frankel's theory for 2-cyclic matrices to compare the convergence rates of this and the block Jacobi approach. Indeed, in this case, we have from (19) that

$$I - M^{-1}A = \begin{pmatrix} 0 & -B^{-1}F \\ 0 & C^{-1}EB^{-1}F \end{pmatrix}.$$

Therefore, the eigenvalues of this matrix are the squares of those of matrix $I - M^{-1}A$ associated with the block-Jacobi preconditioner of Section 3.2.

4 Numerical Experiments

This section is organized as follows. In Section 4.1 we describe the test problems and list the methods that we use. In Section 4.2, we illustrate for comparison purposes the difficulty of incomplete LU factorizations for solving these problems in a fully-coupled manner. In Section 4.3, we make some comments in regard to domain decomposition types of reorderings. In Section 4.4 we show some results of the new preconditioners on a simple PDE problem. Finally, in Sections 4.5 and 4.6, we present the results of the new preconditioners on more realistic problems arising from the incompressible Navier-Stokes equations.

Linear systems were constructed so that the solution is a vector of all ones. A zero initial guess for right-preconditioned FGMRES [20] restarted every 20 iterations was used to solve the systems. The Tables show the number of iterations required to reduce the residual norm by 10^{-7}. The iterations were stopped when 300 matrix-vector multiplications were reached, indicated by a dagger (†). The codes were written in FORTRAN 77 using many routines from SPARSKIT [23], and run in single precision on a Cray C90 supercomputer.

4.1 Test problems and methods

The first set of test problems is a finite difference Laplace equation with Dirichlet boundary conditions. Three different sized grids were used. The matrices were reordered using a domain decomposition reordering with 4 subdomains. In the following tables, n is the order of the matrix, nnz is the number of nonzero entries, n_B is the order of the B submatrix, and n_C is the order of the C submatrix.

The second set of test matrices were extracted from the example incompressible Navier-Stokes problems in the FIDAP [14] package. All problems with zero C submatrix were tested. In the case of transient problems, the matrices are the Jacobians when the Newton iterations had converged. The matrices are reordered so that the continuity equations are

Grid	n	nnz	n_B	n_C
32 by 32	961	4681	900	61
48 by 48	2209	10857	2116	93
64 by 64	3969	19593	3844	125

Table 1: Laplacian test problems.

ordered last. The scaling of many of the matrices are poor, since each matrix contains different types of equations. Thus, we scale each row to have unit 2-norm, and then scale each column the same way. The problems are all originally nonsymmetric except 4, 12, 14 and 32.

Matrix	n	nnz	n_B	n_C	
EX04	1601	31850	1151	450	Hamel flow
EX06	1651	49063	1180	471	Die swell
EX12	3973	79078	2839	1134	Stokes flow
EX14	3251	65875	2351	920	Isothermal seepage
EX20	2203	67830	1603	600	Surface disturbance attenuation
EX23	1409	42761	1008	401	Fountain flow
EX24	2283	47901	1635	648	Forward roll coating
EX26	2163	74465	1706	457	Driven thermal convection
EX28	2603	77031	1853	750	Two merging liquids
EX31	3909	91223	3279	630	Dilute species deposition
EX32	1159	11047	863	296	Radiation heat transfer
EX36	3079	53099	2575	504	Chemical vapor deposition
EX40	7740	456189	5916	1824	3D Die swell

Table 2: FIDAP example matrices.

The third set of test problems is from a finite-element discretization of the square lid-driven cavity problem. Rectangular elements were used, with biquadratic basis functions for velocities, and linear discontinuous basis functions for pressure. We will show our results for problems with Reynolds number 0, 500, and 1000. All matrices arise from a mesh of 20 by 20 elements, leading to matrices of size $n = 4562$ and having nnz =138,187 nonzero entries. These matrices have 3363 velocity unknowns, and 1199 pressure unknowns. The matrices are scaled the same way as for the FIDAP matrices—the problems are otherwise very difficult to solve.

We will use the following names to denote the methods that we tested.

NOPRE No preconditioner.

ILUT(*nfil*) and **ILUTP**(*nfil*) Incomplete LU factorization with threshold of *nfil* nonzeros per row in each of the L and U factors. This preconditioner will be described in Section 4.2.

PAR(*lfil*) Partial approximate inverse preconditioner described in Section 2.2.3, using *lfil* nonzeros per row in M_2.

ABJ Approximate block-Jacobi preconditioner described in Section 3.2. This preconditioner only applies when $C \neq 0$.

ABLU(*lfil*) Approximate block LU factorization preconditioner described in Section 3.3. The approximation (6) to S with *lfil* nonzeros per column of Y was used.

ABLU_y(*lfil*) Same as above, but using Y whenever $B^{-1}F$ needs to be applied in step 3 of Algorithm 3.1.

ABLU_s(*lfil*) Approximate block LU factorization preconditioner, using (7) to approximate S^{-1} with *lfil* nonzeros per column when approximating the last block column of the inverse of A.

ABGS(*lfil*) Approximate block Gauss-Seidel preconditioner described in Section 3.4. The approximation (6) to S with *lfil* nonzeros per column of Y was used.

The storage requirements for each preconditioner are given in Table 3. The ILUT preconditioner to be described in the next subsection requires considerably more storage than the approximate block-partitioned factorizations, since its storage depends on n rather than n_C. Because the approximation to S^{-1} discards the upper block, the storage for it is less than *lfil* $\times n_C$. The storage required for $\tilde{S}$ is more difficult to estimate since it is at least the product of two sparse matrices. It is generally less than $2 \times$ *lfil* $\times n_C$; Table 11 in Section 4.5 gives the exact number of nonzeros in $\tilde{S}$ for the FIDAP problems.

4.2 ILU for the fully-coupled system

We wish to compare our new preconditioners with the most general, and in our experience, one of the most effective general-purpose preconditioners for solving the fully-coupled system. In particular, we show results for ILUT, a dual-threshold, incomplete LU factorization preconditioner based on a drop-tolerance and the maximum number of new fill-in elements allowed per row in each L and U factor. This latter threshold allows the storage for the preconditioner to be known beforehand. Drop-tolerance ILU rather than level-fill ILU is often more effective for indefinite problems where numerical values play a much more important role. A variant that performs column pivoting, called ILUTP, is even more suitable for highly indefinite problems.

	Matrices	Matrix locations
ILUT(*nfil*)	L, U	$2 \times$ *nfil* $\times n$
PAR(*lfil*)	M_2	*lfil* $\times n_C$
ABJ	none	none
ABLU(*lfil*)	$\tilde{S}$	less than $2 \times$ *lfil* $\times n_C$
ABLU_y(*lfil*)	$\tilde{S}, Y$	less than $3 \times$ *lfil* $\times n_C$
ABLU_s(*lfil*)	approx S^{-1}	less than *lfil* $\times n_C$
ABGS(*lfil*)	$\tilde{S}$	less than $2 \times$ *lfil* $\times n_C$

Table 3: Storage requirements for each preconditioner.

We use a small modification that we have found to often perform better and rarely worse on matrices that have a wide ranging number of elements per row or column. This arises for various reasons, including the fact that the matrix contains the discretization of different equations. Instead of counting the number of new fill-ins, we keep the nonzeros in each row of L and U fixed at *nfil*, regardless of the number of original nonzeros in that row. We also found better performance when keeping *nfil* constant rather than having it increase or decrease as the factorization progresses.

If A is highly indefinite or has large nonsymmetric parts, an ILU factorization often produces unstable L and U factors, i.e., $\|(LU)^{-1}\|$ can be extremely large, caused by the long recurrences in the forward and backward triangular solves [11]. To illustrate this point, we computed for a number of factorizations the rough lower bound

$$\|(LU)^{-1}\|_\infty \geq \|(LU)^{-1}e\|_\infty,$$

where e is a vector of all ones. For the FIDAP example matrix EX07 modeling natural convection with order 1633 and 46626 nonzeros, we see in Table 4 that the norm bound increases dramatically as *nfil* is decreased in the incomplete factorization. GMRES could not solve the linear systems with these factorizations as the preconditioner. This matrix we chose is a striking example because it can be solved without preconditioning.

nfil	10	20	30	40	50	60	70	80	90	100
$\log_{10} \|(LU)^{-1}e\|_\infty$	132	174	203	175	277	359	231	31	27	22

Table 4: Estimate of $\|(LU)^{-1}\|_\infty$ from ILUT factors for EX07.

To illustrate the difficulty of solving the FIDAP problems with ILUTP, we progressively allowed more fill-in until the problem could be solved, incrementing *nfil* in multiples of 10, with no drop tolerance. The results are shown in Table 5. For these types of problems, it is typical that very large amounts of fill-in must be used for the factorizations to be successful. An iterative solution was not attempted if the LU condition lower bound was greater than 10^{30}. If a zero pivot must be used, ILUT and ILUTP attempt to complete the factorization by using a small value proportional to the norm of the row. The matrices were taken in their original banded ordering, where the degrees of freedom of a node or element are numbered together. As discussed in the next subsection, this type of ordering having low bandwidth is often essential for an ILU-type preconditioning—many problems including these cannot be solved otherwise.

Matrix	*nfil*
EX04	20
EX06	50
EX12	70
EX14	>100
EX20	30
EX23	10
EX24	10
EX26	>100
EX28	20
EX31	10
EX32	>100
EX36	10
EX40	10

Table 5: *nfil* required to solve FIDAP problems with ILUTP.

We should note that ILUTP is occasionally worse than ILUT. This can be alleviated somewhat by using a low value of *mbloc*, a parameter in ILUTP that determines how far to search for a pivot. In summary, indefinite problems such as these arising from the incompressible Navier-Stokes equations may be very tough for ILU-type preconditioners.

4.3 Domain decomposition reordering considerations

Graph partitioners subdivide a domain into a number of pieces and can be used to give the domain decomposition reordering described in Section 1. This is a technique to impose a block-partitioned structure on the matrix, and adapts it for parallel processing, since B is now a block-diagonal matrix. This technique is also useful if B is highly indefinite and produces an unstable LU factorization; by limiting the size of the factorization, the instability cannot grow beyond a point for which the factorization is not useful. For general, nonsymmetric matrices, the partitioner may be applied to a symmetrized graph.

In Table 6 we show some results of ILUT(40) on the Driven cavity problem with different matrix reorderings. We used the original unblocked ordering where the degrees of freedom of the elements are ordered together, the blocked ordering where the continuity equations are ordered last, and a domain decomposition reordering found using a simple automatic recursive dissection procedure with four subdomains. This latter ordering found 3680 nodes internal to the subdomains, and 882 interface nodes.

Re.	Unblocked	Blocked	DD ordered
0	24	48	60
500	27	†	51
1000	78	†	51

Table 6: Effect of ordering on ILUT for Cavity problems.

The poorer quality of the incomplete factorization for the Driven cavity problems in block-partitioned form is due to the poor ordering rather than instability of the L and U factors; in fact, zero pivots are not encountered. For the problem with Reynolds number 0, the unblocked format produces 745,187 nonzeros in the strictly lower-triangular part during the incomplete factorization (which is then dropped down to less than $n\times$ *nfil* = 182,480 nonzeros) while the block-partitioned format produces 2,195,688 nonzeros, almost three times more.

The factorization for the domain decomposition reordered matrices encounters many zero pivots when it reaches the (2,2) block. These latter orderings do not necessarily cause ILUT to fill-in zeros on the diagonal. Nevertheless, the substitution of a small pivot described above seems to be effective here. The domain decomposition reordering also reduces the amount of fill-in because of the shape of the matrix (a downward pointing arrow). Combined with its tendency to limit the growth of instability, the results show this reordering is advantageous even on serial computers.

In Table 7 we compare the difficulty of solving the B and $\tilde{S}$ subsystems for the blocked and domain decomposition reorderings of the Driven cavity problems. $\tilde{S}$ was computed as $\tilde{S} = C - EY$, where Y was computed using the approximate inverse technique with *lfil* of 30. Here we used ILUT(30) and only solved the linear systems to a tolerance of 10^{-5}. Solves with these submatrices in the block-partitioned preconditioners usually need to be much less accurate. In most of the experiments that follow, we used unpreconditioned iterations to a tolerance of 10^{-1} or 100 matrix-vector multiplications to solve with B and $\tilde{S}$. Other methods would be necessary depending on the difficulty of the problems. The table gives an idea of how difficult it is to solve with B and $\tilde{S}$, and again shows the advantage of using domain decomposition reorderings for hard problems.

Re.	Blocked		DD ordered	
	B	$\tilde{S}$	B	$\tilde{S}$
0	6	3	9	†
500	180	4	7	26
1000	†	†	7	45

Table 7: Solving with B and $\tilde{S}$ for different orderings of A.

4.4 Test results for the Laplacian problem

In Tables 8 and 9 we present the results for the Laplacian problem with three different grid sizes, using no preconditioning, approximate block diagonal, partial approximate inverse, approximate block LU, and approximate block Gauss-Seidel preconditioners. Note that in Table 9, an *lfil* of zero for the approximate block LU and Gauss-Seidel preconditioners respectively indicate the preconditioners

$$M = \begin{pmatrix} B & 0 \\ E & C \end{pmatrix} \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix} \qquad \text{and} \qquad M = \begin{pmatrix} B & 0 \\ E & C \end{pmatrix}. \tag{26}$$

Grid	**NOPRE**	**ABJ**	**PAR**			
			5	10	15	20
32 by 32	135	33	21	18	16	15
48 by 48	367	50	29	21	19	17
64 by 64	532	57	36	33	25	20

Table 8: Test results for the Laplacian problem.

Grid	**ABLU**					**ABGS**				
	0	5	10	15	20	0	5	10	15	20
32 by 32	23	17	15	15	15	15	17	15	15	15
48 by 48	17	18	16	15	15	18	19	19	18	18
64 by 64	19	20	18	18	17	20	23	21	20	20

Table 9: Test results for the Laplacian problem.

4.5 Test results for the FIDAP problems

For the block-partitioned factorization preconditioners, unpreconditioned GMRES, restarted every 20 iterations, was used to approximately solve the inner systems involving B and $\tilde{S}$ by reducing the initial residual norm by a factor of 0.1, or using up to 100 matrix-vector multiplications. Solves with the matrix B are usually not too difficult because for most problems, it is positive definite. A zero initial guess for these solves was used. The results for a number of the preconditioners with various options are shown in Table 10. The best preconditioner appears to be ABLU_y; using Y for $B^{-1}F$ is better than solving a system with B very inaccurately. The number of nonzeros in $\tilde{S}$ is small, as illustrated by Table 11 for two values of *lfil*.

	PAR	**ABLU_s**	**ABLU**		**ABLU_y**		**ABGS**	
Matrix	20	5	20	40	20	40	20	40
EX04	77	78	†	91	227	100	†	126
EX06	†	†	†	†	†	†	†	†
EX12	61	72	53	36	48	34	61	41
EX14	†	†	75	35	83	40	72	48
EX20	141	†	†	60	246	90	†	178
EX23	†	90	†	†	†	269	†	†
EX24	†	†	†	93	†	91	207	136
EX26	†	117	†	91	104	63	209	163
EX28	289	†	214	67	105	67	144	104
EX31	101	49	†	†	122	74	120	162
EX32	136	†	38	21	39	29	63	37
EX36	54	70	80	52	47	36	69	54
EX40	205	†	†	96	84	75	119	102

Table 10: Test results for the FIDAP problems.

4.6 Test results for the Driven cavity problems

The driven cavity problems are much more challenging because the B block is no longer positive definite, and in fact, acquires larger and larger negative eigenvalues as the Reynolds number increases. For these problems, the unpreconditioned GMRES iterations with B were done to a tolerance of 10^{-3} or a maximum of 100 matrix-vector multiplications. Again, ABLU_y appears to be the best preconditioner. The results are shown in Table 12.

	lfil	
Matrix	20	40
EX04	13377	20796
EX06	15077	23533
EX12	30851	48940
EX14	33144	49932
EX20	21759	33119
EX23	12463	19084
EX24	18966	29010
EX26	13395	21468
EX28	25181	38716
EX31	16551	24452
EX32	6775	11390
EX36	13621	21063
EX40	49729	93330

Table 11: Number of nonzeros in $\tilde{S}$.

	ABLU		**ABLU_y**		**ABGS**	
Re.	20	40	20	40	20	40
0	62	42	59	42	84	58
500	†	†	182	92	130	103
1000	†	†	164	118	†	†

Table 12: Test results for the Driven cavity problems.

5 Conclusions

We have presented a few preconditioners which are defined by combining two ingredients: (1) a sparse approximate inverse technique for obtaining a preconditioner for the Schur complement or a part of the inverse of A, and (2) a block factorization for the full system. The Schur complement S which appears in the block factorization is approximated by its preconditioner. Approximate inverse techniques [6] are used in different ways to approximate either S directly or a part of A^{-1}.

As can be seen by comparing Tables 5 and 10, we can solve more problems with the block approach than with a standard ILU factorization. In addition, this is typically achieved with a far smaller memory requirement than ILUT or a direct solver. The better robustness of these methods is due to the fact that solves are only performed for small matrices. In effect, we are implicitly using the power of the divide-and-conquer strategy which is characteristic of domain decomposition methods. The smaller matrices obtained from the block partitioning can be preconditioned with a standard ILUT approach. The larger matrices use a block-ILU, and the glue between the two is the preconditioning of the Schur complement.

Acknowledgements The authors wish to acknowledge the support of the Minnesota Supercomputer Institute which provided the computer facilities and an excellent environment to conduct this research.

References

[1] F. Alvarado, H. Dağ and M. ten Bruggencate. Block-bordered diagonalization and parallel iterative solvers. Colorado Conference on Iterative Methods, April 5-9, 1994.

[2] O. Axelsson. Incomplete block matrix factorization preconditioning methods. The ultimate answer? *J. Comput. Appl. Math.,* 12 & 13 (1985), pp. 3-18.

[3] O. Axelsson. *Iterative Solution Methods.* Cambridge University Press, New York, 1994.

[4] O. Axelsson and B. Polman. On approximate factorization methods for block matrices suitable for vector and parallel processors. *Lin. Alg. Appl.*, 77 (1986), pp. 3-26.

[5] M. W. Benson and P. O. Frederickson. Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems. *Utilitas Math.,* 22 (1982), pp. 127-140.

[6] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices, Technical Report UMSI 94/101, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, Minnesota, 1994.

[7] P. Concus, G. H. Golub and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.,* 6 (1985), pp. 309-332.

[8] J. D. F. Cosgrove, J. C. Díaz and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *Intl. J. Comp. Math.*, 44 (1992), pp. 91-110.

[9] L. C. Dutto, W. G. Habashi and M. Fortin. Parallelizable block diagonal preconditioners for the compressible Navier-Stokes equations, *Comput. Methods Appl. Mech. Engrg.,* 117 (1994), pp. 15-47.

[10] T. Huckle and M. Grote. A new approach to parallel preconditioning with sparse approximate inverses. Manuscript SCCM-94-03, Scientific Computing and Computational Mathematics Program, Stanford University, Stanford, California, 1994.

[11] H. C. Elman. A stability analysis of incomplete LU factorizations. *Math. Comp.,* 47 (1986), pp. 191-217.

[12] H. C. Elman. Multigrid and Krylov subspace methods for the discrete Stokes equations. *Proc. Matrix Analysis and Parallel Computing,* Keio University, March 14–16, 1994, pp. 151-164.

[13] H. C. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations. UMIACS-TR-94-66, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, 1994.

[14] M. Engleman. *FIDAP: Examples Manual*, Revision 6.0. Fluid Dynamics International, Evanston, Illinois, 1991.

[15] M. Grote and H. D. Simon. Parallel preconditioning and approximate inverses on the Connection Machine. In R. F. Sincovec, D. E. Keyes, L. R. Petzold and D. A. Reed, eds., *Parallel Processing for Scientific Computing,* vol. 2, pp. 519-523. SIAM, Philadelphia, Pennsylvania, 1993.

[16] David E. Keyes and William D. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM J. Sci. Stat. Comput.*, 8 (1987) pp. s166-s202.

[17] L. Yu. Kolotilina and A. Yu. Yeremin. On a family of two-level preconditionings of the incomplete block factorization type, *Soviet J. Numer. Anal. Math. Model.,* 1 (1986), pp. 293-320.

[18] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM J. Matrix Anal. Appl.*, 14 (1993), pp. 45-58.

[19] L. Yu. Kolotilina and A. Yu. Yeremin. Incomplete block factorizations as preconditioners for sparse SPD matrices. Research Report EM-RR 6/92, Elegant Mathematics, Inc. Bothell, Washington, 1992.

[20] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.,* 14 (1993), pp. 461-469.

[21] Y. Saad. ILUT: A dual threshold incomplete LU factorization. *Num. Lin. Alg. Appl.,* 1 (1994), pp. 387-402.

[22] Y. Saad. Preconditioned Krylov subspace methods for CFD applications, *Proceedings of the International Workshop on Solution Techniques for Large-Scale CFD Problems,* Montréal, Sept. 26–28, 1994, pp. 179-195.

[23] Y. Saad. SPARSKIT: a basic tool kit for sparse matrix computations, Version 2. Preprint, University of Minnesota, Minneapolis, Minnesota, 1993.