Electronic Transactions on Numerical Analysis.
Volume 10, 2000, pp. 115-130.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

ON PRECONDITIONING SCHUR COMPLEMENT AND SCHUR COMPLEMENT PRECONDITIONING *

JUN ZHANG†

Abstract. We study two implementation strategies to utilize Schur complement technique in multilevel recursive incomplete LU preconditioning techniques (RILUM) for solving general sparse matrices. The first strategy constructs a RILUM to precondition the original matrix. The second strategy solves the first Schur complement matrix using the lower level parts of the RILUM as the preconditioner. We discuss computational and memory costs of both strategies and the potential effect on grid independent convergence rate of RILUM with different implementation strategies.

Key words. sparse matrices, Schur complement, RILUM, preconditioning techniques.

AMS subject classifications. 65F10, 65N06.

1. Introduction. In this paper, we discuss the issue of implementing a class of multilevel recursive incomplete LU (RILUM) preconditioners. These preconditioners were first reported and implemented in [40]. RILUM is a general framework for constructing robust multilevel preconditioning techniques based on block incomplete LU factorization of the coefficient matrix. The problems targeted by RILUM are general sparse linear systems of the form

$$Ax = b, \tag{1.1}$$

where A is an unstructured real matrix of order n.

It is common to solve general sparse linear systems by a preconditioned iterative method consisting of an accelerator and a preconditioner [24]. In most situations, a Krylov subspace method such as GMRES is used as the accelerator [25, 24]. A sparse matrix M based on an incomplete LU (ILU) factorization of the coefficient matrix A is constructed to serve as the preconditioner. Analytical studies [18] and experimental investigations [37] have shown that the convergence rate of a preconditioned Krylov subspace method is mainly determined by the quality of the preconditioner. The ILU type preconditioning techniques lie between direct and iterative methods and provide a balance between reliability and scalability.

Although preconditioners may be developed from many sources and with different strategies, most general purpose preconditioners seem to be derived from ILU factorizations of the coefficient matrices [17]. It has been noted by a few researchers [12, 23, 24, 35] that standard ILU preconditioners are not robust for solving realistic problems. High accuracy preconditioners that allow more fill-in in the ILU factorizations are needed in these situations. Recently, a class of preconditioners based on multilevel block ILU factorization (BILUM) have been introduced and shown to yield grid independent convergence rates for certain types of problems [23, 29]. BILUM is faster and more robust than standard ILU preconditioners. For certain difficult problems, these favorable attributes come with the added benefit of smaller memory usage. In addition, these preconditioners are highly parallel and their inherent parallelism can be exploited on parallel computers [26]. Other multilevel preconditioning methods have also been developed from various multilevel incomplete factorization of the coefficient matrices [1, 2, 13, 19, 30, 39] and with different construction techniques [3, 4, 5, 6, 7, 15, 34].

*Received May 25, 1999. Accepted for publication February 1, 2000. Recommended by S. Parter. This research was supported in part by the U.S. National Science Foundation under grant CCR-9902022 and in part by the University of Kentucky Center for Computational Sciences.

†Department of Computer Science, University of Kentucky, 773 Anderson Hall, Lexington, KY 40506-0046, USA (`E-mail:jzhang@cs.uky.edu.` `URL:http://www.cs.uky.edu/~jzhang`).

Multilevel and multigrid techniques build on the idea that different error components can be treated efficiently on different level scales. This is the fundamental philosophy behind many multiscale computation techniques. Multilevel ILU preconditioning techniques such as BILUM are similar in philosophy to classical algebraic multigrid methods [8, 10, 20]. The latter were designed to mimic geometric multigrid methods for solving sparse linear systems without a grid structure. In fact, it has been shown that there are links between multilevel preconditioning techniques and algebraic multigrid methods [28]. Our recent research work shows that comparable multilevel ILU preconditioners can also be constructed using algebraic multigrid approaches [39].

A common problem with most ILU preconditioners is that their accuracy is fixed once they are computed. If it is found that the accuracy of a computed preconditioner is insufficient and the preconditioned iterative method converges unacceptably slowly, a new preconditioner with a higher accuracy has to be recomputed. The previously computed preconditioner cannot be updated and the computation is wasted.

In a recent paper [40], we proposed a general framework for constructing robust multilevel recursive ILU preconditioning techniques (RILUM). We enhanced the robustness of BILUM preconditioner by adding interlevel acceleration in the form of solving interlevel Schur complement matrices approximately. The main quality of RILUM that differs it from other ILU preconditioners is that its accuracy is dynamically determined in the preconditioning process by specifying different stopping criteria to the coarse level iterations. These preconditioners, by their construction and by the philosophy on which they are based, may be highly robust and scalable for solving general sparse linear systems. Two implementations have been tested in [40], mainly to compare different treatments of successive Schur complement matrices. It has been found that forming the Schur complement matrices in the preconditioning process is more efficient than computing and storing the approximate Schur complement matrices in the construction phase. For certain types of problems, the former implementation may yield a convergence rate that is independent of the problem sizes; the latter may not.

This paper follows the idea of [40]. Instead of using the constructed multilevel recursive matrices to precondition the original coefficient matrix A, we propose to solve the first Schur complement matrix by an iterative process. Such a solution process is then preconditioned by the coarser level recursive Schur complement matrices. This new strategy is realized with the implementation of forming Schur complement matrices in the preconditioning process.

This paper includes four additional sections. The next section introduces a general framework for constructing multilevel recursive ILU preconditioning techniques. Section 3 discusses the advantages and disadvantages of different treatments of the Schur complement matrices. Section 4 reports numerical comparisons. Concluding remarks are given in Section 5.

2. RILUM Preconditioner. We use α for the level reference and assume $0 \le \alpha \le \mathcal{L}$ for a certain integer $\mathcal{L}$. For convenience we denote the original coefficient matrix as $A_0 = A$.

On a given level α, we first permute the matrix A_α into a two by two block form

$$P_\alpha A_\alpha P_\alpha^T = \begin{pmatrix} D_\alpha & F_\alpha \\ E_\alpha & C_\alpha \end{pmatrix}, \tag{2.1}$$

where P_α is a permutation matrix to be specified later. We denote the order of A_α by n_α and that of D_α by m_α. The order of the submatrix C_α is $n_{\alpha+1} = n_\alpha - m_\alpha$. The submatrix D_α is block diagonal, well conditioned, and not too small. Such a reordering of the matrix can be obtained by a block independent set ordering [29]. The conditioning of the submatrix D_α may be controlled by imposing a diagonal threshold strategy in the block independent set search process [31, 32]. For the ease of notation in our following discussions, we use A_α to denote both the permuted and the unpermuted matrices. The permutation matrix P_α will no longer appear explicitly.

In multilevel preconditioning techniques the block partitioning (2.1) induces a block LU factorization of the form

$$\begin{pmatrix} I_\alpha & 0_\alpha \\ E_\alpha D_\alpha^{-1} & I_\alpha \end{pmatrix} \begin{pmatrix} D_\alpha & F_\alpha \\ 0_\alpha & A_{\alpha+1} \end{pmatrix}, \tag{2.2}$$

where

$$A_{\alpha+1} = C_\alpha - E_\alpha D_\alpha^{-1} F_\alpha \tag{2.3}$$

is the Schur complement of A_α with respect to C_α. I_α and 0_α are the generic identity and zero matrices on level α. Suppose that the above block factorizations can be performed successfully for $0 \le \alpha \le \mathcal{L}$, we will have a sequence of matrices

$$\mathcal{A} = \{A_0, A_1, \ldots, A_{\mathcal{L}}, A_{\mathcal{L}+1}\}.$$

The relative density (the number of nonzero elements in each row) of the matrices is increasing as the matrix order decreases. To maintain sparsity during the multilevel recursive LU factorization we drop small size elements in the computation of $A_{\alpha+1}$ so that $A_{\alpha+1}$ (see $\bar{A}_{\alpha+1}$ below) is approximately as sparse as A_α. There are at least two types of dropping strategy that can be employed during the construction of A_α [40]. A *single dropping strategy* uses a threshold parameter $\tau > 0$. Any computed elements that are smaller than τ times the average of the nonzero elements of the current row are dropped. A *double dropping strategy* will use another parameter p, in addition to τ, to control the total number of nonzero elements kept in the L and U parts of each row. For detailed descriptions on various dropping strategies, see [22, 30, 40]. Hence, we actually compute an approximate Schur complement $\bar{A}_{\alpha+1}$ of A_α. Denote $\bar{A}_0 = A_0 = A$, we have a sequence of matrices

$$\bar{\mathcal{A}} = \{\bar{A}_0, \bar{A}_1, \ldots, \bar{A}_{\mathcal{L}}, \bar{A}_{\mathcal{L}+1}\},$$

which is an approximation of $\mathcal{A}$. ¿From the matrix sequence $\bar{\mathcal{A}}$ we can construct another sequence of matrices in a block factored form

$$M_\alpha = \begin{pmatrix} I_\alpha & 0_\alpha \\ E_\alpha D_\alpha^{-1} & I_\alpha \end{pmatrix} \begin{pmatrix} D_\alpha & F_\alpha \\ 0_\alpha & \bar{A}_{\alpha+1} \end{pmatrix}, \qquad \alpha = 0, 1, \ldots, \mathcal{L}. \tag{2.4}$$

The matrix M_α is an approximate factorization of $\bar{A}_\alpha$ and an approximate to A_α in the form of (2.2). It can be used as a preconditioner for the matrix A_α or $\bar{A}_\alpha$. For completeness, we assume that the matrix $M_{\mathcal{L}+1}$ is constructed differently as a fixed preconditioner, which means its action does not require any further subsidiary iteration.

A preconditioning process is to transform a linear system

$$\bar{A}_\alpha x_\alpha = b_\alpha \tag{2.5}$$

into an equivalent form

$$M_\alpha^{-1} \bar{A}_\alpha x_\alpha = M_\alpha^{-1} b_\alpha. \tag{2.6}$$

The purpose of performing such a transformation is the expectation that, with a "good" preconditioner M_α, an iterative method solving (2.6) will converge much faster than solving

(2.5). The reduction in iteration counts is so significant that the additional cost of implementing preconditioning is compensated.

The set of the matrices M_α can be considered as a class of multilevel preconditioner based on the recursive ILU factorization of the coefficient matrix. This class of preconditioner is referred to as multilevel recursive ILU preconditioners or RILUM [40].

The application of the preconditioner M_α requires the solution of a linear system (the approximate Schur complement)

$$\bar{A}_{\alpha+1} x_{\alpha+1} = b_{\alpha+1}. \tag{2.7}$$

The solution of the system (2.7) can be obtained by another iteration process preconditioned by the preconditioner $M_{\alpha+1}$. The application of $M_{\alpha+1}$ requires the solution of $\bar{A}_{\alpha+2} x_{\alpha+2} = b_{\alpha+2}$, and so on. This recursive preconditioning-solving process continues until the solution of the linear system

$$\bar{A}_{\mathcal{L}+1} x_{\mathcal{L}+1} = b_{\mathcal{L}+1}$$

can be obtained inexpensively either by a direct method [1] or by an iterative method preconditioned by a fixed preconditioner $M_{\mathcal{L}+1}$. Figure 2.1 shows the logical relations between $\bar{A}_\alpha$ and M_α on each level. Figure 2.2 is an illustration of the multilevel recursive ILU preconditioner in a simple form in which one iteration is performed on each level. Figure 2.2 is reminiscent of a multigrid V cycle algorithm [9]. Alternative cycling algorithms can be deduced analogously.

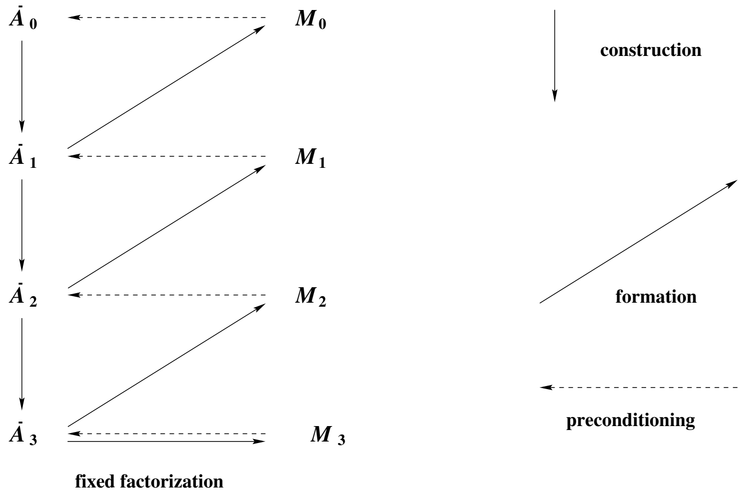

FIG. 2.1. *Logical relations between the Schur complement (coefficient) matrices and the preconditioners on each level in the general framework of multilevel recursive preconditioning techniques.*

The more accurately the system (2.7) is solved, the more accurate the preconditioner M_α is. It follows that a more accurate RILUM preconditioner results. Thus, the accuracy of RILUM can be controlled dynamically in the preconditioning process, after it has been computed. If the iteration process indicates that the accuracy of the current preconditioning process is insufficient, a more strict stopping criterion can be placed on the subiteration on solving the system (2.7) so that it yields better preconditioning effect on level α.

[1] In this case, $M_{\mathcal{L}+1}$ is not needed.

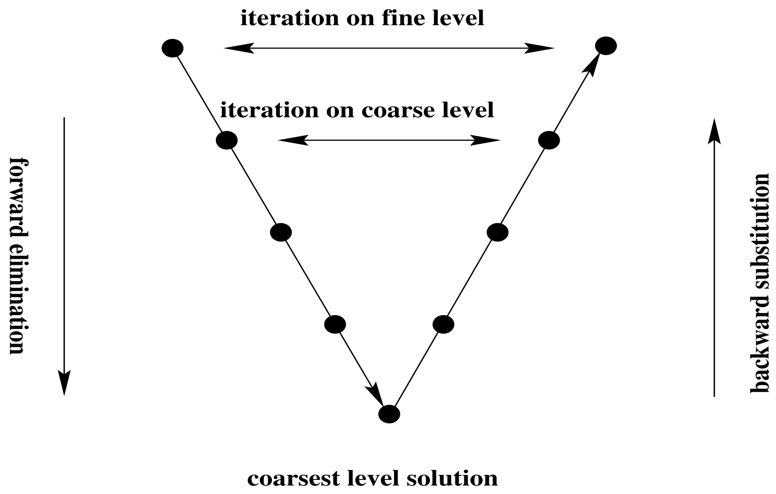

FIG. 2.2. *An illustration of a multilevel recursive ILU preconditioning process.*

Let the right-hand side vector b and the solution vector x be partitioned in accordance to the block partitioning (2.1). On each level α we have

$$x_\alpha = \begin{pmatrix} x_{\alpha,1} \\ x_{\alpha,2} \end{pmatrix} \qquad \text{and} \qquad b_\alpha = \begin{pmatrix} b_{\alpha,1} \\ b_{\alpha,2} \end{pmatrix}.$$

Within each iteration on the original linear system (1.1) with $\alpha = 0$ we need to solve the linear systems (2.5) for $1 \le \alpha \le \mathcal{L}+1$ to a certain accuracy. We may use a Krylov subspace method and use M_α as the preconditioner. The preconditioning action consists of a block forward elimination and a block backward substitution. On each level α, the forward elimination is performed by solving

$$\begin{pmatrix} I_\alpha & 0_\alpha \\ E_\alpha D_\alpha^{-1} & I_\alpha \end{pmatrix} \begin{pmatrix} y_{\alpha,1} \\ y_{\alpha,2} \end{pmatrix} = \begin{pmatrix} b_{\alpha,1} \\ b_{\alpha,2} \end{pmatrix} \qquad \text{with} \qquad \left\{ \begin{array}{lcl} y_{\alpha,1} & = & D_\alpha^{-1} b_{\alpha,1}, \\ y_{\alpha,2} & = & b_{\alpha,2} - E_\alpha y_{\alpha,1}, \end{array} \right.$$

where $y_\alpha = (y_{\alpha,1}, y_{\alpha,2})^T$ is a temporary vector. Note that we separate the applications of D_α^{-1} and E_α since we do not in general compute the explicit product matrix $E_\alpha D_\alpha^{-1}$, but keep them as two separate factors. The backward substitution is performed to obtain the preconditioning solution x_α by solving

$$\begin{pmatrix} D_\alpha & F_\alpha \\ 0_\alpha & \bar{A}_{\alpha+1} \end{pmatrix} \begin{pmatrix} x_{\alpha,1} \\ x_{\alpha,2} \end{pmatrix} = \begin{pmatrix} y_{\alpha,1} \\ y_{\alpha,2} \end{pmatrix} \quad \text{with} \quad \left\{ \begin{array}{lcl} x_{\alpha,2} & \approx & \bar{A}_{\alpha+1}^{-1} y_{\alpha,2}, \\ x_{\alpha,1} & = & D_\alpha^{-1}(y_{\alpha,1} - F_\alpha x_{\alpha,2}). \end{array} \right.$$

In the first step above, we solve the linear system

$$\bar{A}_{\alpha+1} x_{\alpha,2} = y_{\alpha,2} \tag{2.8}$$

approximately to a certain accuracy by a preconditioned Krylov subspace method. Note that if $x_{\alpha,2} = x_{\alpha+1}$ and $y_{\alpha,2} = b_{\alpha+1}$, the system (2.8) is just the system (2.7) and can therefore be preconditioned by $M_{\alpha+1}$.

Since the preconditioning procedure involves iteration processes, the preconditioner changes in different outer iterations. Such a feature requires that the Krylov subspace accelerator be able to accommodate a variable preconditioner. Krylov subspace methods such as FGMRES [21] and GMRESR [33] can accept a variable preconditioner and are ideal for such applications.

3. Block factorization and Schur complement. The block factorization (2.2) is performed implicitly. The submatrix D_α^{-1} instead of D_α is stored, along with the submatrices E_α and F_α. The product matrix $E_\alpha D_\alpha^{-1}$ is not formed explicitly. Its action will be recovered in the preconditioning process by applying D_α^{-1}, followed by E_α, to the vector in question. Structurally, we store the sparse matrices

$$\tilde{M}_\alpha = \begin{pmatrix} D_\alpha^{-1} & F_\alpha \\ E_\alpha & \bar{A}_{\alpha+1} \end{pmatrix} \qquad \text{or} \qquad \tilde{M}_\alpha = \begin{pmatrix} D_\alpha^{-1} & F_\alpha \\ E_\alpha & C_\alpha \end{pmatrix}$$

level by level in a single long vector, followed by $M_{\mathcal{L}+1} = L_{\mathcal{L}+1}U_{\mathcal{L}+1}$.

The Schur complement matrix $\bar{A}_{\alpha+1}$, similarly to $A_{\alpha+1}$ in (2.3), will be computed explicitly in order for the next level preconditioner $M_{\alpha+1}$ to be constructed. One of the two dropping strategies discussed previously is used to keep $\bar{A}_{\alpha+1}$ sparse. The computed $\bar{A}_{\alpha+1}$ is an approximate to $A_{\alpha+1}$, as in (2.3).

Since we have to solve a system with $A_{\alpha+1}$ or $\bar{A}_{\alpha+1}$ in the preconditioning iteration process, either one of these matrices has to be stored or one of their actions has to be recovered. In the first place, $\bar{A}_{\alpha+1}$ is computed explicitly for the construction of the coarser level preconditioner $M_{\alpha+1}$ and is stored after serving that purpose. The preconditioning process for M_α will solve the linear system (2.7). In this case, the submatrix C_α needs not to be stored. We emphasize that $\bar{A}_\alpha x_\alpha = b_\alpha$ is solved with M_α as preconditioner, which requires that $\bar{A}_{\alpha+1} x_{\alpha+1} = b_{\alpha+1}$ be solved,

Alternatively, we may discard $\bar{A}_{\alpha+1}$ after the construction of $M_{\alpha+1}$. Instead, we store the submatrix C_α and solve the linear system with $A_{\alpha+1}$ for the preconditioner M_α. The action of $A_{\alpha+1}$ is recovered with matrix by matrix multiplications and a matrix subtraction as in (2.3), since the matrices $C_\alpha, D_\alpha^{-1}, E_\alpha$, and F_α are readily available. In this case, we solve $A_{\alpha+1}x_{\alpha+1} = b_{\alpha+1}$ with $M_{\alpha+1}$ as preconditioner, which is actually the permuted $\bar{A}_{\alpha+1}$.

Suppose both linear systems with $A_{\alpha+1}$ and with $\bar{A}_{\alpha+1}$ are solved to the same accuracy. The solution from the system with $A_{\alpha+1}$ will have a better preconditioning effect than that from the system with $\bar{A}_{\alpha+1}$, since $A_{\alpha+1}$ is an exact Schur complement of A_α and $\bar{A}_{\alpha+1}$ is an approximate. Keeping $\bar{A}_{\alpha+1}$ is expensive in terms of storage cost while computing $A_{\alpha+1}$ is expensive in terms of computational cost. Numerical experiments in [40] show that the second strategy of assembling the exact Schur complement matrix in the preconditioning process may yield a preconditioner with a convergence rate independent of the problem size for certain types of test problems. In the sequel, we will discuss the RILUM implementation without storing the approximate Schur complement matrices $\bar{A}_\alpha$.

3.1. Preconditioning Schur complement. The two implementation strategies described above use the Schur complement matrix to precondition the matrix A. They can be viewed as *Schur complement preconditioning strategies*. Since we favor the strategy that does not store the approximate Schur complement matrices, we propose a new implementation for this strategy which forms the Schur complement action in the preconditioning process.

The first Schur complement matrix can be formed exactly, even it is not computed explicitly. There seems to be no need to use it as a preconditioner. We can solve it to the required accuracy and recover the solution of the original system by one step of backward substitution. In an alternative view, we can consider solving the first level system

$$(C_0 - E_0 D_0^{-1} F_0)x_1 = b_1 \tag{3.1}$$

to the accuracy that would be required by the outer iteration on the system (1.1), then the outer iteration process will converge in one iteration. This strategy will be referred to as *preconditioning Schur complement strategy*.

There are advantages to applying a Krylov subspace accelerator to the reduced system (3.1). A Krylov subspace accelerator like FGMRES has to be restarted after a few iterations and a restarted FGMRES is not as robust as the full FGMRES [24]. In order for a restarted FGMRES to be as robust as possible, the dimension of the Krylov subspace should be as large as the computer memory permits. Since the size of the reduced system (3.1), n_1, is substantially smaller than the size of A, n_0, we can allow a larger restart value for the FGMRES applied on the reduced system (3.1) than that we could allow for the FGMRES applied on the original system $A_0x_0 = b_0$. Furthermore, it is generally believed that, for certain classes of problems, the condition of a Schur complement matrix is better than that of the original matrix [14]. The preconditioned iterative solver may converge faster when applied to the reduced system (3.1).

Another saving of memory comes from the observation that, in the case of solving the Schur complement system (3.1) the original matrix is no longer needed since no matrix-vector product with A is required and since the stored submatrices D_0^{-1}, E_0, F_0, and C_0 and the permutation matrix P_0 are enough to compute an approximate solution to the original linear system.

Let $nz(A)$ be the number of nonzero elements in A and k_0 be the dimension of FGMRES of the outer iteration originally applied on A. The total saving in storage cost on the first level with preconditioning on the first Schur complement matrix A_1 is approximately

$$(2k_0 + 3)m_0 + nz(A),$$

where m_0 is the size of the block independent set on level 0. The factor $(2k_0 + 3)$ instead of $(k_0 + 3)$ is due to the fact that additional k_0 vectors are needed in FGMRES to store the preconditioned residuals [21]. This extra storage space can be used to increase the dimension of the Krylov subspace of the outer iteration from k_0 to

$$k_0 + \frac{(2k_0 + 3)m_0 + nz(A)}{n - m_0}.$$

The quality of a preconditioner is also related to a parameter commonly referred to as the *sparsity ratio*. A sparsity ratio is defined as the ratio of the nonzero elements of a preconditioner to that of the original matrix [28]. In the Schur complement preconditioning case, the sparsity ratio is

$$S_{sp} = \frac{\sum_{\alpha=0}^{\mathcal{L}+1} nz(\tilde{M})}{nz(A)}.$$

In the case of preconditioning the Schur complement matrix with a Krylov subspace accelerator of the same dimension, the sparsity ratio is reduced to

$$S_{ps} = \frac{\sum_{\alpha=1}^{\mathcal{L}+1} nz(\tilde{M}) + nz(D_0^{-1}) - nz(D_0)}{nz(A)} = S_{sp} - 1.$$

There will also be fewer floating point operations involved in the preconditioning Schur complement strategy. The number of floating point operations at each preconditioning step is proportional to the sparsity ratio of the preconditioner.

It is also possible to use the extra storage space associated with the preconditioning Schur complement strategy to construct a more accurate RILUM preconditioner.

3.2. Stopping criteria. As proposed in [40], the interlevel preconditioning processes can be stopped when the residual norm with respect to the linear system on level α satisfies a certain stopping criterion. In practice, we should also set an upper bound for the maximum number of iterations allowed in an iteration process. The coarse level computations can be very complicated if many iterations are allowed on each level, since each iteration needs preconditioning processes on all coarser levels. In fact, the total number of preconditioning iterations can be quite large. Suppose a maximum of q iterations are allowed on level α (except for the outer iterations on level 0), the total number of preconditioning iterations for one outer iteration is bounded by

$$\sum_{\alpha=1}^{\mathcal{L}} q^{\alpha} = \frac{q(q^{\mathcal{L}} - 1)}{q - 1}.$$

Of course, each iteration on a coarse level costs much less than the one on a finer level.

Numerical experiments in [40] show that too many coarse level iterations may result in an inefficient solver. There needs to be a mechanism to determine whether or not an iteration on a given level needs to be preconditioned by all coarser level iterations.

The preconditioning step associated with solving the preconditioned system (2.6) is usually realized in the form of solving

$$\bar{A}_{\alpha} \bar{w}_{\alpha} = r_{\alpha}, \tag{3.2}$$

where r_{α} is the current residual vector on level α. An exact preconditioning would be equivalent to solving the linear system

$$A_{\alpha} w_{\alpha} = r_{\alpha} \tag{3.3}$$

exactly.

PROPOSITION 3.1. *Let w_{α} be the solution of the exact preconditioning equation (3.3) and $\tilde{w}_{\alpha}$ be the approximate solution to the preconditioning equation (3.2) such that $\|\bar{w}_{\alpha} - \tilde{w}_{\alpha}\| < \epsilon_{\alpha}$ for a given stopping criterion ϵ_{α} and any consistent norm $\|\cdot\|$, then the preconditioning error on level α is bounded by*

$$\|w_{\alpha} - \tilde{w}_{\alpha}\| \le \|A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1}\| \, \|r_{\alpha}\| + \epsilon_{\alpha}. \tag{3.4}$$

Proof. ¿From Equations (3.2) and (3.3), we have

$$\bar{w}_{\alpha} = \bar{A}_{\alpha}^{-1} r_{\alpha} \qquad \text{and} \qquad w_{\alpha} = A_{\alpha}^{-1} r_{\alpha}.$$

It follows that

$$\|w_{\alpha} - \bar{w}_{\alpha}\| = \|(A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1}) r_{\alpha}\| \le \|(A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1})\| \, \|r_{\alpha}\|.$$

Hence,

$$\begin{aligned} \|w_{\alpha} - \tilde{w}_{\alpha}\| &\le \|w_{\alpha} - \bar{w}_{\alpha}\| + \|\bar{w}_{\alpha} - \tilde{w}_{\alpha}\| \\ &\le \|A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1}\| \, \|r_{\alpha}\| + \epsilon_{\alpha}. \end{aligned}$$

□

It can be seen from (3.4) that there is no need to have a small ϵ_{α} by solving (3.2) to a high accuracy if $\|A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1}\|$ is large. The quantity $\|A_{\alpha}^{-1} - \bar{A}_{\alpha}^{-1}\|$ is determined by the incomplete factorization process and can be referred to as the *factorization error*. The factorization error is a function of τ in the case of a single dropping strategy or a function of both τ and p in the case of a double dropping strategy. ϵ_α is controlled by the coarser level iteration process on level $\alpha + 1$ and can be viewed as the *iteration error*. Hence, the bound (3.4) implies that the accuracy of the preconditioner is controlled by both the factorization error and the iteration error. A high accuracy RILUM preconditioner can be obtained by achieving a high accuracy both in the factorization process and in the iteration process.

3.3. Grid independent convergence rate. In the case of the Schur complement preconditioning, the zeroth level exact preconditioner would be A (disregarding the permutation matrix P_0) and the exact preconditioning solution would be

$$w_0 = A^{-1} r_0.$$

It is obvious that the factorization error on the 0th level is zero, since we form the exact Schur complement matrix in the preconditioning process. If the iterative solution process on the first level is solved to a certain fixed accuracy at each preconditioning step such that $\|\bar{w}_0 - \tilde{w}_0\| \le \epsilon_0$ independently of the size of the matrix A, as it would be in an implementation with no limit on the number of coarse level iterations, the convergence rate of the outer Krylov subspace iteration is independent of the size of the matrix A. This is the nature of the grid independent convergence rate of RILUM with the Schur complement preconditioning strategy.

In the case of preconditioning Schur complement, we are actually solving A_1, which is then preconditioned by $\bar{A}_1$. Let the factorization error be $\|A_1^{-1} - \bar{A}_1^{-1}\| = \bar{\epsilon}_1(\tau)$ as a function of the dropping tolerance τ, assuming a single dropping strategy is employed. Suppose that the preconditioning system $\bar{A}_1 \bar{w}_1 = r_1$ is solved to a certain fixed accuracy such that $\|\bar{w}_1 - \tilde{w}_1\| \le \epsilon_1$. Then the overall preconditioning error on the first level is bounded by

$$\|w_1 - \tilde{w}_1\| \le \|b_1\| \, \bar{\epsilon}_1(\tau) + \epsilon_1. \tag{3.5}$$

Since the factorization error $\epsilon_1(\tau)$ is usually dependent on the size of the matrix A, the bound (3.5) implies that the first level preconditioner cannot yield a preconditioning accuracy that is independent of the size of the matrix A. In other words, the strategy of preconditioning Schur complement cannot yield a grid independent convergence rate in general. We summarize our findings in the following proposition.

PROPOSITION 3.2. *Suppose the first level preconditioning iteration process is solved to a fixed accuracy regardless of the number of iterations required on the subiteration processes. The Schur complement preconditioning strategy may yield a grid independent convergence rate while the preconditioning Schur complement strategy may not.*

Nevertheless, as indicated in our previous analyses, the preconditioning Schur complement strategy has the advantage of using less memory and performing fewer floating point operations in each iteration step. With the same factorization error and the same stopping criterion, the Schur complement preconditioning strategy tends to converge in a smaller number of iterations than the preconditioning Schur complement strategy does. On the other hand, the preconditioning Schur complement strategy will take less CPU time if its iteration number is not too large. As far as efficient computation is concerned, the preconditioning Schur complement strategy may be preferred under certain conditions.

In the worst case of the preconditioning Schur complement strategy, there is a potential problem of having too many iterations, since current iterative accelerators that allow variable preconditioners (like FGMRES) are usually implemented with a restarted version as stated previously. A restarted FGMRES may take even more iterations to converge or may stagnate. The potential gain in reducing floating point operations by solving a smaller system in the preconditioning Schur complement strategy may be lost if too many more iterations are needed. It is crucial to allocate the memory saved from using the preconditioning Schur complement strategy to allow a larger restart value for FGMRES to minimize the frequency of restart or to allow a more accurate preconditioner to be constructed.

4. Numerical Experiments. Various implementations of multilevel ILU preconditioning techniques (e.g., ILUM, BILUM, and BILUTM) have been described in detail in [23, 29, 30]. One significant difference between RILUM and BILUM (BILUTM) is that we had an outer iteration on the original system and several inner iterations on the coarse level systems. The preconditioner for the coarsest level system is the ILUT(p, τ) of Saad [23]. Unless otherwise explicitly indicated, we used the following default parameters for our preconditioned iterative solver: FGMRES(50) (FGMRES with a restart value of 50) was used as the outer iteration accelerator; the inner iterations on each coarse level used FGMRES(10); the maximum number of levels allowed was 10; i.e., $\mathcal{L} = 10$.

All matrices were considered general sparse and any available structures were not exploited. The right-hand side was generated by assuming that the solution is a vector of all ones and the initial guess was a vector of some random numbers. The computations were terminated when the 2-norm of the residual was reduced by a factor of 10^8. We also set an upper bound of 200 for the outer FGMRES iteration. The inner iterations on each coarse level satisfied convergence test if the 2-norm of the coarse level residual was reduced by a factor of 10; and a maximum of 10 iterations were allowed. The initial guesses of all inner iterations were reset to zero vectors before each round of iterations on a given level. The numerical experiments were conducted on a Silicon Graphics workstation using the Fortran 77 programming language.

In all tables with numerical results, "bsize" is the size of the uniform blocks, "iter" shows the number of outer FGMRES iteration, "prec" shows the CPU time in seconds for the preprocessing phase (preconditioner construction), "solu" shows the CPU time for the solution phase, "level" shows the number of actual levels of reduction, "spar" shows the sparsity ratio. The parameters τ and p were used in the single or double dropping strategy.

We implemented RILUM with the strategy of forming the exact Schur complement action on each level during the preconditioning process. The preconditioning Schur complement strategy will be abbreviated as PreSch and the Schur complement preconditioning strategy as SchPre.

4.1. Convection diffusion problem. We first consider a convection diffusion problem

$$u_{xx} + u_{yy} - \mathrm{Re}[x(x-1)(1-2y)u_x - y(y-1)(1-2x)u_y] = 0 \qquad (4.1)$$

defined on the unit square. Here Re is the Reynolds number. Dirichlet boundary condition was assumed, but the linear systems used the artificially generated right-hand side as stated previously. We used the standard 5-point central difference discretization scheme and a fourth order 9-point compact finite difference discretization scheme [16]. The percentage of the rows with a diagonal dominance becomes smaller as Re increases [38, 36].

We first tested several sparse matrices arising from the 9-point finite difference discretization scheme. We refined the mesh size to see how the performances of the preconditioners were affected when the problem size was enlarged. The other parameters were fixed as $\mathrm{Re} = 1000, \mathrm{bsize} = 30, p = 30, \tau = 0.05$. The single dropping strategy was used for tests in this subsection. The test results are listed in Table 4.1.

The results in Table 4.1 confirm our analyses in Section 3.3. It shows that SchPre strategy yields a convergence rate that is independent of the problem size. It also shows that PreSch

TABLE 4.1
Comparison of PreSch and SchPre for solving the convection diffusion problem with the 9-point finite difference discretization scheme and different mesh size.

		PreSch			SchPre		
h	prec	iter	solu	spar	iter	solu	spar
1/32	0.09	7	0.04	3.09	6	0.08	4.09
1/64	0.58	8	0.53	3.99	6	0.95	4.99
1/128	4.12	14	6.35	4.38	6	8.20	5.38
1/256	32.30	33	61.82	4.39	7	92.36	5.39

strategy does not usually converge independently of the problem size. However, in all cases listed in Table 4.1, PreSch took less CPU time to converge and used less memory space than SchPre did.

We did another test with a matrix arising from the 5-point central difference discretization of Equation (4.1) with $\mathrm{Re} = 1$ and $h = 1/201$. We varied the dropping parameter τ so that preconditioners with different accuracy were computed. The other parameters were fixed as $\mathrm{bsize} = 20, p = 30$. The results are shown in Figure 4.1. We note that when the accuracy of the incomplete factorization decreased, the iteration number of PreSch increased rapidly, while that of SchPre increased moderately. On the other hand, PreSch took less CPU time than SchPre did when FGMRES(50) did not restart. The situation was reversed when FGMRES(50) was restarted more than once for PreSch. When FGMRES(50) was restarted just once for PreSch and was not restarted for SchPre, both took about the same CPU time to converge. This test indicates that restarting FGMRES deteriorates the convergence severely. Frequently restarting FGMRES makes PreSch strategy unattractive. The test results agree with our analyses and comments made in previous sections.

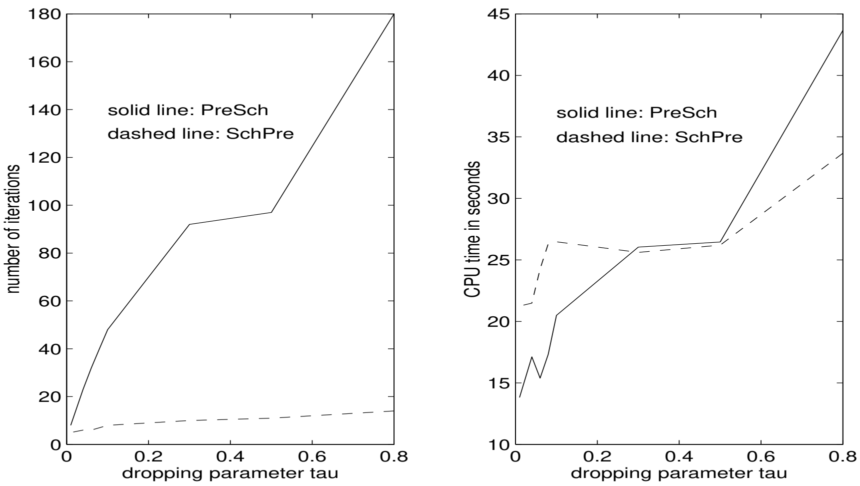

FIG. 4.1. *Comparison of PreSch and SchPre with respect to the dropping parameter τ for solving a matrix from the 5-point central difference discretization scheme.*

We repeated our test without restarting FGMRES and used the same parameters as before. The results are depicted in Figure 4.2. This time PreSch took less CPU time than SchPre did for all but $\tau = 0.8$. Our observation is that preconditioning Schur complement strategy is viable when the incomplete factorization is accurate enough and the Krylov subspace accel-

ETNA
Kent State University
etna@mcs.kent.edu

erator does not have to be restarted frequently.

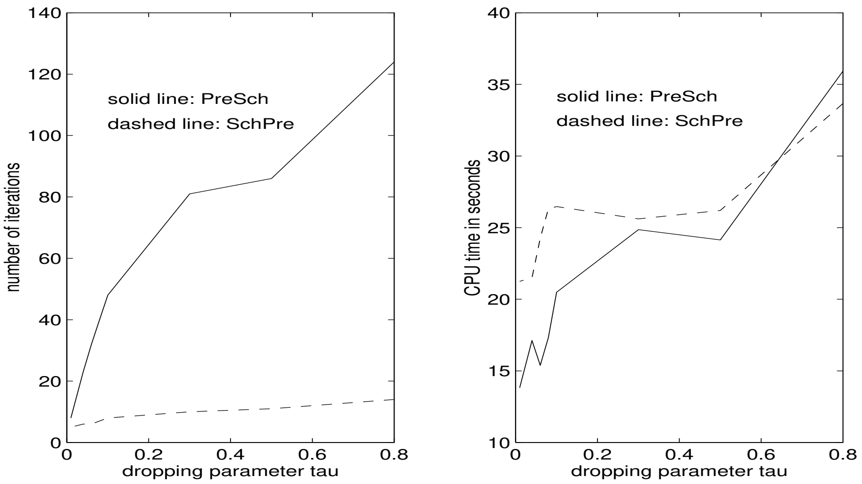

FIG. 4.2. *Comparison of PreSch and SchPre with respect to different dropping parameter τ for solving a matrix from the 5-point central difference discretization scheme. Full FGMRES was used in both cases.*

4.2. Double dropping strategy. For test results reported in this subsection, the double dropping strategy was used and a maximum number of 5 levels of reduction was allowed.

The parameter p was used to control the maximum number of nonzero elements in $\bar{A}_\alpha$ on all levels. Table 4.2 tabulates the comparison of PreSch and SchPre for solving the matrices arising from the 9-point discretization of Equation 4.1 with $h = 1/151$. The other parameters used were $\mathrm{bsize} = 40, \tau = 0.01$.

TABLE 4.2
Comparison of PreSch and SchPre for solving the convection diffusion problem with the 9-point finite difference discretization scheme and different Reynolds number.

		PreSch			SchPre		
Re	prec	iter	solu	spar	iter	solu	spar
0	6.26	40	10.16	3.97	7	12.85	4.97
1	6.24	40	9.99	3.96	7	12.94	4.96
10	6.19	39	9.80	3.95	7	12.76	4.95
100	6.23	49	12.31	3.98	9	17.69	4.98
1000	4.21	24	9.74	4.64	7	20.99	5.64
10000	6.32	9	7.40	4.21	5	9.19	5.21

The results in Table 4.2 show that the convergence rate of PreSch strategy was heavily affected by the Reynolds number. The convergence rate of SchPre strategy was only slightly affected by the variation of the Reynolds number. In all but one cases, PreSch used less CPU time than SchPre did to reach convergence. Note that for these tests, FGMRES was never restarted.

Next, we solved the UTM3060 matrix using PreSch and SchPre with different parameters. This matrix arises from a nuclear plasma modeling (tokamak) and is generated by P. Brown at the Lawrence Livermore National Laboratory. The matrix has 3060 unknown and 42211 nonzero elements.

TABLE 4.3
Comparison of PreSch and SchPre for solving the UTM3060 matrix with the double dropping strategy and 5 *levels of reduction.*

				PreSch			SchPre		
p	τ	bsize	prec	iter	solu	spar	iter	solu	spar
40	0.01	80	1.20	25	10.55	6.26	9	28.19	7.26
40	0.01	50	1.29	20	14.21	5.55	8	48.54	6.55
40	0.01	30	1.27	15	60.80	5.01	7	148.57	6.01
60	0.01	80	1.48	15	9.44	7.14	8	25.80	8.14
60	0.01	50	1.68	16	27.17	6.43	7	73.44	7.43
60	0.01	30	1.68	14	32.95	6.08	7	69.41	7.08
60	0.001	80	2.37	15	12.30	8.41	8	26.84	9.41
60	0.001	50	2.33	14	48.84	7.74	7	123.47	8.74
60	0.001	30	2.38	12	76.58	7.74	7	191.00	8.74

Based on the results presented in Table 4.3, we can see that the convergence rate of SchPre strategy is not much affected by the variation of the parameters, but the computational cost is. It seems that large size blocks help both PreSch and SchPre reduce computational cost (CPU time), although the memory cost (sparsity ratio) is increased slightly. It is very interesting to note that, for this test problem, smaller τ values and larger p values, which are supposed to yield a more accurate preconditioner, did not produce better results.

4.3. More tests. Further, we compared the two RILUM implementation strategies using a few sparse matrices downloaded online either from the Matrix Market of the National Institute of Standards and Technology [2] or from the University of Florida sparse matrix collection [11]. [3] Many of these matrices have been used in various tests of sparse iterative solvers [22, 24, 29].

For this set of tests, we used the same parameters for all matrices. The parameters were chosen as $\mathrm{bsize} = 50, \tau = 0.1, p = 30$. At most 5 levels of reduction were allowed and the single dropping strategy was used. The test results are listed in Table 4.4. The sparsity ratios reported in Table 4.4 are for the PreSch strategy. The corresponding sparsity ratios for the SchPre strategy would be "$\mathrm{spar} + 1$". We find that, except for solving the SAYLR4 matrix, PreSch strategy took less CPU time to converge than SchPre strategy did for solving all other matrices. Once again, the test results reinforce our observations made earlier that the PreSch strategy is computationally more efficient.

4.4. Coarse level iterations. Finally, we tested the two implementation strategies to solve the SAYLR4 and UTM1700b matrices with different number of coarse level iterations. In the experiments reported previously, the maximum number of coarse level iterations on all levels is 10. One iteration on a given level will be preconditioned by at most 10 iterations on a coarser level (or until the 2-norm residual on the level in question is reduced by 10 orders of magnitude) and so on. In the current test, we fixed all other parameters as $\mathrm{level} = 5, \mathrm{bsize} = 20, \tau = 0.1, p = 30$ and a single dropping strategy was used. The test results are given in Table 4.5. Based on our test results, we can say that too many iterations on coarse levels do not yield efficient computational schemes, although the numbers of outer iterations are usually smaller for larger number of inner iterations. On the other hand, too few inner iterations do

[2] `http://math.nist.gov/MatrixMarket`.
[3] `http://www.cise.ufl.edu/~davis/sparse`.

TABLE 4.4
Comparison of PreSch and SchPre for solving general sparse matrices.

					PreSch		SchPre	
Matrix	order	nonzero	prec	spar	iter	solu	iter	solu
ORSIRR11	1 030	6 858	0.09	4.77	25	0.17	7	0.24
RAEFSKY1	3 242	294 276	20.31	4.27	11	17.67	6	38.76
RAEFSKY2	3 242	294 276	27.89	4.27	10	37.45	7	76.42
RAEFSKY5	6 316	168 658	1.23	0.70	2	0.23	4	0.65
RAEFSKY6	3 402	137 845	0.83	0.47	3	0.19	4	0.42
SAYLR3	1 000	3 750	0.08	8.84	11	0.12	6	0.19
SAYLR4	3 564	22 316	0.29	5.16	90	2.02	8	1.50
SHERMAN1	1 000	3 750	0.08	8.84	11	0.12	6	0.21
SHERMAN3	5 005	20 033	0.69	8.61	21	3.23	7	6.42
SHERMAN4	1 104	3 786	0.08	6.83	9	0.05	6	0.01
UTM1700b	1 700	21 509	0.36	3.79	44	9.05	25	47.20
WATT2	1 856	11 550	0.34	8.49	30	0.60	7	0.94

not provide sufficient preconditioning effect and the outer iterations do not converge within the maximum number of iterations allowed. As far as computational efficiency is concerned, the number of inner iterations should be just slightly more than what is sufficient to guarantee convergence. Nevertheless, more inner iterations usually make the preconditioned iterative solvers more robust to reduce the possibility of no convergence.

TABLE 4.5
Comparison of PreSch and SchPre for solving the SAYLR4 and UTM1700b matrices with different number of coarse level iterations.

	SAYLR4				UTM1700b			
	PreSch		SchPre		PreSch		SchPre	
inner-iter	iter	solu	iter	solu	iter	solu	iter	solu
1	–	–	–	–	–	–	–	–
2	–	–	–	–	–	–	–	–
3	–	–	86	4.91	–	–	–	–
4	189	4.37	30	2.48	177	21.58	152	51.63
5	95	2.52	17	1.82	75	21.29	60	69.96
6	93	2.78	14	1.95	50	28.87	45	128.12
7	92	3.15	12	2.02	37	33.91	23	119.97
8	92	3.23	11	2.22	34	35.19	21	156.65
9	70	2.45	10	2.20	35	33.55	19	153.19
10	89	3.13	10	2.40	34	32.23	15	139.21
11	89	3.18	10	2.66	35	33.50	13	130.35
12	89	3.20	9	2.54	35	33.29	13	122.27

5. Concluding Remarks. We have proposed a new implementation strategy within the general framework of multilevel recursive ILU preconditioning techniques (RILUM) for solving general sparse matrices. This strategy is based on an implementation introduced in [40] to form the exact Schur complement action in the preconditioning process. The new strategy abandons the original outer Krylov subspace iteration and solves the first level Schur complement matrix. The Krylov subspace iteration on the first level Schur complement matrix is further preconditioned by the lower level RILUM.

Analyses on the new implementation strategy, which is referred to as the preconditioning Schur complement (PreSch) strategy, show that the PreSch strategy can save significant amount of memory space and can lower the sparsity ratio of RILUM by 1, provided that the preconditioner is accurate enough.

Numerical results show that PreSch strategy does not usually yield a grid independent convergence rate. It can nevertheless reduce the CPU timings significantly, comparing to the standard implementation with the Schur complement preconditioning strategy. However, the number of iterations of the PreSch strategy is usually large which entails more restarts of FGMRES if the incomplete factorization is not accurate enough. Frequent restart of FGMRES may render the PreSch strategy inefficient.

Another possible implementation strategy for RILUM is to use a domain based multilevel block ILU factorization (BILUTM) [30], in which the individual blocks are factored by an ILUT strategy. This implementation strategy has been preliminarily reported by Saad and Suchomel [27]. However, as we mentioned in [40], the RILUM implementation based on BILUTM strategy may not yield a grid independent convergence rate, since the first level factorization of BILUTM is not exact. It may also be impossible to implement a *preconditioning Schur complement strategy* as discussed in this paper with the BILUTM implementation due to the same reason (inexact factorization). On the other hand, RILUM based on BILUTM implementation may be more suitable for very large blocks. In this case, the approximate Schur complement matrices on all levels are stored and are used in the preconditioning process.

Maybe the most efficient implementations of RILUM are yet to be found. The relation among preconditioning error and factorization error and iteration error has been addressed qualitatively in Proposition 3.1. But we have not found a practical implementation that will automatically determine the iteration accuracy based on the factorization accuracy. This and other issues are to be investigated in our ongoing research on multilevel recursive ILU preconditioning techniques.

REFERENCES

[1] O. Axelsson and M. Neytcheva, *Algebraic multilevel iteration method for Stieltjes matrices*, Numer. Linear Algebra Appl., 1 (1994), pp. 216–236.

[2] O. Axelsson and P. S. Vassilevski, *Algebraic multilevel preconditioning methods. I*, Numer. Math., 56 (1989), pp. 157–177.

[3] R. E. Bank and R. K. Smith, *The incomplete factorization multigraph algorithm*, SIAM J. Sci. Comput., 20 (1999), pp. 1349–1364.

[4] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.

[5] E. F. F. Botta and F. W. Wubs, *Matrix renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.

[6] D. Braess, *Towards algebraic multigrid for elliptic problems of second order*, Computing, 55 (1995), pp. 379–393.

[7] C. I. W. Brand, *An incomplete-factorization preconditioning using red-black ordering*, Numer. Math., 61 (1992), pp. 433–454.

[8] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse equations*, in Sparsity and Its Applications (Loughborough 1983), D. J. Evans, ed., Cambridge, 1985, Cambridge Univ. Press, pp. 257–284.

[9] W. L. Briggs, *A Multigrid Tutorial*, SIAM, Philadelphia, PA, 1987.

[10] Q. S. Chang, Y. S. Wong, and H. Q. Fu, *On the algebraic multigrid method*, J. Comput. Phys., 125 (1996), pp. 279–292.

[11] T. Davis, *University of Florida sparse matrix collection*, NA Digest, 97 (1997).

[12] E. F. D'Azevedo, F. A. Forsyth, and W. P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 944–961.

[13] H. C. ELMAN, *Approximate Schur complement preconditioners on serial and parallel computers*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 581–605.
[14] H. C. ELMAN AND G. H. GOLUB, *Iterative methods for cyclically reduced nonselfadjoint linear systems*, Math. Comp., 54 (1990), pp. 671–700.
[15] M. GRIEBEL, *Parallel domain-oriented multilevel methods*, SIAM J. Sci. Comput., 16 (1995), pp. 1105–1125.
[16] M. M. GUPTA, R. P. MANOHAR, AND J. W. STEPHENSON, *A single cell high order scheme for the convection-diffusion equation with variable coefficients*, Internat. J. Numer. Methods Fluids, 4 (1984), pp. 641–651.
[17] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[18] N. M. NACHTIGAL, S. C. REDDY, AND L. N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM Matrix Anal. Appl., 13 (1992), pp. 778–795.
[19] A. A. REUSKEN, *Approximate cyclic reduction preconditioning*, Tech. Report RANA 97-02, Department of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands, 1997.
[20] J. W. RUGE AND K. STÜBEN, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., Frontiers in Appl. Math., SIAM, Philadelphia, PA, 1987, ch. 4, pp. 73–130.
[21] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 461–469.
[22] ———, *ILUT: a dual threshold incomplete LU preconditioner*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.
[23] Y. SAAD, *ILUM: a multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
[24] ———, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, NY, 1996.
[25] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual method for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[26] Y. SAAD, M. SOSONKINA, AND J. ZHANG, *Domain decomposition and multi-level type techniques for general sparse linear systems*, in Domain Decomposition Methods 10, J. Mandel, C. Farhat, and X. Cai, eds., no. 218 in Contemporary Mathematics, Providence, RI, 1998, AMS, pp. 174–190.
[27] Y. SAAD AND B. SUCHOMEL, *ARMS: an algebraic recursive multilevel solver for general sparse linear systems*, in Abstracts of the Ninth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1999, p. 37.
[28] Y. SAAD AND J. ZHANG, *Enhanced multilevel block ILU preconditioning strategies for general sparse linear systems*, J. Comput. Appl. Math. to appear.
[29] ———, *BILUM: block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (1999), pp. 2103–2121.
[30] ———, *BILUTM: a domain-based multilevel block ILUT preconditioner for general sparse matrices*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 279–299.
[31] ———, *Diagonal threshold techniques in robust multi-level ILU preconditioners for general sparse linear systems*, Numer. Linear Algebra Appl., 6 (1999), pp. 257–280.
[32] ———, *A multi-level preconditioner with applications to the numerical simulation of coating problems*, in Iterative Methods in Scientific Computing II, D. R. Kincaid and A. C. Elster, ed., New Brunswick, NJ, 1999, IMACS, pp. 437–449.
[33] H. A. VAN DER VORST AND C. VUIK, *GMRESR: a family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[34] C. WAGNER, W. KINZELBACH, AND G. WITTUM, *Schur-complement multigrid - a robust method for groundwater flow and transport problems*, Numer. Math., 75 (1997), pp. 523–545.
[35] D. M. YOUNG, R. G. MELVIN, F. T. JOHNSON, J. B. BUSSOLETTI, L. B. WIGTON, AND S. S. SAMANT, *Application of sparse matrix solvers as effective preconditioners*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 1186–1199.
[36] J. ZHANG, *Preconditioned iterative methods and finite difference schemes for convection-diffusion*, Appl. Math. Comput., 109 (2000), pp. 11–30.
[37] ———, *Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications*, Comput. Methods Appl. Mech. Engrg. to appear.
[38] ———, *On convergence and performance of iterative methods with fourth-order compact schemes*, Numer. Methods Partial Differential Equations, 14 (1998), pp. 262–283.
[39] ———, *A grid based multilevel incomplete LU factorization preconditioning technique for general sparse matrices*, Tech. Report No. 283-99, Department of Computer Science, University of Kentucky, Lexington, KY, 1999.
[40] ———, *RILUM: a general framework for robust multilevel recursive incomplete LU preconditioning techniques*, Tech. Report No. 284-99, Department of Computer Science, University of Kentucky, Lexington, KY, 1999.