Diagonal Threshold Techniques in Robust Multi-Level ILU Preconditioners for General Sparse Linear Systems [*]

Yousef Saad[†] and Jun Zhang[‡]

Department of Computer Science and Engineering, University of Minnesota,
4-192 EE/CS Building, 200 Union Street S.E., Minneapolis, MN 55455

January 22, 1998

Abstract

This paper introduces techniques based on diagonal threshold tolerance when developing multi-elimination and multi-level incomplete LU (ILUM) factorization preconditioners for solving general sparse linear systems. Existing heuristics solely based on the adjacency graph of the matrices have been used to find independent sets and are not robust for matrices arising from certain applications in which the matrices may have small or zero diagonals. New heuristic strategies based on the adjacency graph and the diagonal values of the matrices for finding independent sets are introduced. Analytical bounds for the factorization and preconditioned errors are obtained for the case of a two-level analysis. These bounds provide useful information in designing robust ILUM preconditioners. Extensive numerical experiments are conducted in order to compare robustness and efficiency of various heuristic strategies.

Key words: Incomplete LU factorization, reordering techniques, multi-level preconditioner, Krylov subspace methods, multi-elimination ILU factorization.
AMS subject classifications: 65F10, 65N06.

1 Introduction

We study reordering techniques in developing efficient multi-level preconditioner to solve general sparse linear system

$$Au = b, \tag{1}$$

where A is an unstructured matrix of order n. Such large linear systems are often solved by Krylov subspace methods coupled with a suitable preconditioner [30]. It is widely accepted that the convergence rate of a preconditioned Krylov subspace method is primarily determined by the quality of the preconditioner employed [20, 30]. As a result, recent focus has shifted from designing iterative accelerators to constructing efficient preconditioners.

[*]This work was supported in part by NSF under grant CCR-9618827, and in part by the Minnesota Supercomputer Institute.

[†]E-mail: saad@cs.umn.edu. URL: http://www.cs.umn.edu/~saad.

[‡]E-mail: jzhang@cs.umn.edu. URL: http://www.cs.umn.edu/~jzhang.

Many preconditioners have been developed with specific applications in mind and as such their efficiency is somewhat limited to those applications. The main trade-offs when comparing preconditioners are in their intrinsic efficiency, their parallelism, and their robustness. An experimental study on robustness of some of these general-purpose preconditioners has been conducted in [14]. For an introduction to Krylov subspace methods and various preconditioning techniques, see [30].

The 'multi-elimination ILU preconditioner' (ILUM), introduced in [29], is based on exploiting the idea of successive independent set orderings and several heuristic algorithms have been suggested to find the independent set. The ILUM preconditioner has a multi-level structure and offers a good degree of parallelism. Similar preconditioners have been designed and tested in [5, 32] to show near grid-independent convergence for certain problems. In a recent report, some of these multi-level preconditioners have been tested and compared favorably with other preconditioned iterative methods and direct methods at least for the Laplace equation [7].

Some of these approaches require grid information. Examples of such approaches include the nested recursive two-level factorization and the repeated red-black orderings [2, 6, 9, 22, 33] (see also the survey paper by Axelsson and Vassilevski [1]). Other methods require only the adjacency graph of the coefficient matrices [5, 29, 32]. For the repeated red-black ordering approach, a near-optimal bound $O(n^{0.153})$ for the condition number of the preconditioned matrix has been obtained [21]. There are other algebraic multigrid methods [11, 24, 25, 26] and multigrid methods with matrix dependent treatments [15, 23], as well as multi-level preconditioning techniques based on hierarchical basis or ILU decomposition associated with the finite difference or finite element analysis [3, 4, 10]. The ILUM preconditioning technique has been extended to block version (BILUM) in which the pivoting elements are not diagonals, but small block diagonals [8, 32]. For some hard-to-solve problems, the performance of ILUM may be enhanced by the block treatment.

The heuristic algorithms introduced in [29] to find independent sets ignore the values of the matrix. For matrices arising from certain applications such as finite element methods in computational fluid dynamics on unstructured meshes, some diagonal values may be very small or even zero. Heuristics that only consider the adjacency graph of the matrices will cause nodes with small or zero diagonal values to be part of the independent set. As a result, the inverse of the diagonal matrix with such diagonal entries will contain very large elements and this may cause the resulting preconditioner to be 'unstable' [14]. Although techniques may be used to invert a near singular diagonal matrix approximately, the resulting preconditioner is sometimes less efficient.

A common method for assessing the quality of a preconditioner is to estimate the condition number of the preconditioned system. These estimates are difficult to obtain when the matrix in question has no predetermined structure. Instead we choose to analyze the factorization and preconditioned errors to get an insight on how the parameters affect the performance of the ILUM preconditioning technique. It is possible to compare different preconditioners by comparing their preconditioned errors. In general, a preconditioner with a smaller preconditioned error tends to yield faster convergence.

In summary, this paper introduces a simple strategy to prevent a node with a small or zero diagonal from being included to the independent set. Algorithms for finding independent set based on the adjacency graph and diagonal values of the matrix are introduced and studied in Section 2. The factorization and preconditioned error bounds of ILUM are derived in Section 3 to guide the design of robust ILUM preconditioner. We test robustness and efficiency of several ILUM implementations and a single-level ILUT over a few sets of problems in Section 4. Concluding remarks are included in Section 5.

2 Independent Set Orderings

The ILUM preconditioning technique exploits a set of unknowns that are not coupled to each other by an equation. Such a set is called an 'independent set'. In independent set orderings, the unknowns are permuted such that those associated with the independent set are listed first, followed by the other unknowns. The permutation matrix P, associated with such an ordering, transforms the original matrix into a matrix which has the following block structure

$$A \sim PAP^T = \begin{pmatrix} D & F \\ E & C \end{pmatrix}, \tag{2}$$

where D is a diagonal matrix of dimension m, and C is a square matrix of dimension $l = n - m$. In the sequel, the notation is slightly abused by not distinguishing the original system from the permuted system, so both permuted and unpermuted matrices will be denoted by A. We use the notation:

$$D = \text{diag}[d_1, \dots, d_m], \quad F = (f_{i,j})_{m \times l}, \quad E = (e_{i,j})_{l \times m}, \quad \text{and} \quad C = (c_{i,j})_{l \times l}.$$

The right-hand side of (2) can be factored in block form as

$$\begin{pmatrix} D & F \\ E & C \end{pmatrix} = \begin{pmatrix} I & 0 \\ ED^{-1} & I \end{pmatrix} \times \begin{pmatrix} D & F \\ 0 & A_1 \end{pmatrix} = LU, \tag{3}$$

where the matrix $A_1 = (\tilde{a}_{j,k})$ is the Schur complement with respect to C

$$A_1 = C - ED^{-1}F. \tag{4}$$

In the simplest two-level method, the reduced system associated with the matrix A_1 is solved exactly and the solution of the original system is found by backward substitution.

Since D is diagonal and C, E, F are sparse, A_1 is also sparse in general. In many applications, A_1 is still too large to be solved inexpensively and the above reduction process can be repeated on A_1, until a reduced system that is small enough is reached. Standard threshold dropping strategies can also be used to maintain a certain level of sparsity in the L and U factors, and in the reduced systems. The solution is then backward substituted level by level until the solution of the original system is found on the finest level. However, the analysis proposed in the next section will be restricted to the two-level method.

Since we need the inverse of D for the Schur complement (4) and for the block (ED^{-1}) in the right-hand side of (3), a small or zero d_i in D may cause problems in the process of forming D^{-1} and may also give rise to instabilities in the LU-solves [14]. An obvious strategy to prevent these problems is to choose each d_i so that $|d_i| > \varepsilon$ for some tolerance ε. This can be combined with the process of finding the independent set.

We now consider finding restricted independent sets, i.e., independent sets with some threshold tolerance restriction on the diagonal entries. Let $G = (V, E)$ denote the adjacency graph of the matrix A, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of vertices and E is the set of edges. Let (v_j, v_k) denote an edge from vertex v_j to vertex v_k. Recall that an *independent set* S is a subset of the vertex set V such that no two elements of S are coupled by an equation, see for example, [29, 30, 32]. An independent set S is *maximal* if there is no independent set which strictly includes S, i.e., for any $v \in V$, $S \cup \{v\}$ is not independent.

The vertex cover S_{vc} associated with the independent set S is the subset S_{vc} of all vertices that are coupled with vertices in S, i.e.,

$$\text{if} \quad v_j \in S \quad \text{then} \quad \{(v_j, v_k) \in E \quad \text{or} \quad (v_k, v_j) \in E\} \to v_k \in S_{vc}. \tag{5}$$

Obviously, the relation $S \cup S_{vc} = V$ holds only when S is maximal. The complement of S with respect to V, denoted by S^c, clearly always satisfies $S \cup S^c = V$.

The term *independent set* is often used to mean a *Maximal Independent Set* (MIS), see for example [29, 32]. Clearly, because of the restrictions on the diagonal elements, the independent sets discussed in this paper are not necessarily maximal.

2.1 Algorithms for independent sets with diagonal threshold

In [30], several heuristic algorithms were discussed to find a maximal independent set. Some of these strategies use locally optimal heuristics and numerical experiments in [30] suggeste that they have similar performance. We first give the simplest greedy algorithm with diagonal threshold tolerance.

ALGORITHM **2.1 Greedy algorithm**

```
1. Set $S = \emptyset$ and select a threshold tolerance $\varepsilon > 0$.
2. for $j = 1, 2, \ldots, n$, do:
3.     if node $j$ is not marked and if $|a_{j,j}| > \varepsilon$, then
4.        $S = S \cup \{j\}$.
5.        mark node $j$, then mark all its nearest neighbors and add them to $S_{vc}$.
6.     else
7.        mark node $j$ if it is not marked and add it to $S^c$.
8.     endif.
9. enddo.
```

The greedy algorithm is one of the simplest and most efficient algorithms for finding independent sets. A variant to the diagonal threshold tolerance in Algorithm 2.1 would be to use relative instead of absolute tolerance. In other words, the acceptance test in line 3 is to be replaced by

$$|a_{j,j}| > \frac{\varepsilon}{|\mathrm{Nz}(j)|} \sum_{i \,\in\, \mathrm{Nz}(j)} |a_{j,i}|. \tag{6}$$

in which $\mathrm{Nz}(j)$ is the set of indices i for which $a_{j,i} \neq 0$, i.e., the nonzero row-pattern for row j. We will also consider another heuristic algorithm that searches an independent set with increasing degree traversal and diagonal threshold tolerance [29].

ALGORITHM **2.2 Increasing degree traversal algorithm**

```
1. find an ordering $i_1, \ldots, i_n$ of the nodes by increasing degree.
2. run a version of Algorithm 2.1, in which the do loop in Line 2 is replaced by the loop:
3.           for $j = i_1, i_2, \ldots, i_n$.
```

There are other locally optimal heuristics that could be used to find an independent set, see [29] for a discussion and comparisons. In this paper, we restrict our attention to the above two heuristics.

Note that the independent set found by Algorithms 2.1 and 2.2 is not necessarily maximal, but has the property that $|D| \geq \varepsilon$ which is defined (component-wise) as $|d_i| \geq \varepsilon$ for all i. For sufficiently large ε, this property will ensure that the size of the inverse of D is not too large. This property will be assumed in the rest of this paper.

The first question we raise is what happens to a node that is in S^c but not in S_{vc}. The following proposition states that the reduction process will not alter its corresponding row.

Proposition 2.1 *Assume that node $j \in S^c$ does not belong to S_{vc}, i.e., that it is a node that is neither in S nor in its vertex cover. Then the corresponding row of C associated with this node is not altered by the reduction process.*

Proof. This is an immediate consequence of the graph model of Gaussian elimination [19]. The reduction process corresponds to eliminating all nodes in S in Gaussian elimination. When a node s in S is eliminated, all its incident edges are removed and only the nodes associated with nodes adjacent to s are altered. These are nodes in S_{vc} and therefore no other row (rows in $S^c - S_{vc}$) is modified. $\square$

In other words, a node that is excluded from the independent set at some level due to diagonal thresholding will remain excluded in the following levels unless it becomes part of a vertex cover later in the process. With the acceptance test used in Algorithm 2.1 as well as the relative test (6), a small diagonal element will be rejected for as long as it is not a neighbor of an element in the independent set. It may be altered later in the elimination process so that its magnitude increases, and may then become eligible to be part of an independent set. Otherwise it will become part of the last reduced system to solve.

2.2 Relation to dropping tolerance

As mentioned before, the ILUM reduction process is usually implemented with some dropping strategies to keep the sparsity of the LU factors and the reduced systems. Standard threshold dropping strategies are suggested in [29, 32]. In the simplest case, all entries in some blocks whose absolute values are smaller than some threshold tolerance τ are dropped, i.e., $a_{j,k}$ is dropped whenever

$$|a_{j,k}| < \tau. \tag{7}$$

Other dropping strategies are based on the values of the row or column or both. For example, the dropping rule used in [32] is relative to the average absolute value of the current row, i.e., an entry $a_{j,k}$ in the U factor ED^{-1} and in the reduced system A_1 is dropped if

$$|a_{j,k}| < \frac{\tau}{|\mathrm{nz}(j)|} \sum_{i \,\in\, \mathrm{nz}(j)} |a_{j,i}|. \tag{8}$$

Diagonal entries in the reduced systems are not dropped regardless of their magnitude. There are other dropping strategies that control the total number of entries in the LU factors and in the reduced systems, see [28, 29, 31, 32].

For an M-matrix, it can be shown that the magnitude of a diagonal element is non-increasing during the reduction process, i.e., $\tilde{a}_{j,j} \leq c_{j,j}$. This seems to suggest that we need to reduce ε as the number of levels grows. It has also been suggested to decrease the dropping tolerance τ for the coarser levels [8]. For general matrices, the magnitudes of the diagonal entries may increase, decrease, or remain the same during the reduction process and so it is preferable to keep ε constant unless dropping tolerance τ changes. We found no obvious benefit in changing τ during the construction of ILUM or its block variant BILUM [29, 32]. Similarly, our tests showed no gains in reducing the diagonal threshold tolerance. Hence, based on our experience we recommend keeping both ε and τ constant and close to each other throughout the reduction process.

3 Factorization and Preconditioned Errors

The actual reduction process is accompanied by dropping strategies discussed in the last subsection. Hence, we have an approximate LU factorization

$$A = \begin{pmatrix} I & 0 \\ \widehat{ED^{-1}} & I \end{pmatrix} \begin{pmatrix} D & F \\ 0 & \hat{A}_1 \end{pmatrix} + R = LU + R, \tag{9}$$

where R is the error matrix resulting from the dropping rule employed. The matrices $\widehat{ED^{-1}}$ and $\hat{A}_1$ result from applying a dropping rule to ED^{-1} and A_1, respectively, see [29] for details. If we use the simplest dropping rule (7) and only drop small entries in ED^{-1} and A_1, then

$$R = (r_{j,k}) = \begin{pmatrix} 0 & 0 \\ R_0 & R_1 \end{pmatrix},$$

where $R_0 = (\tilde{r}_{j,k})_{l\times m}$ and $R_1 = (\bar{r}_{j,k})_{l\times l}$ are

$$\begin{aligned} R_0 &= E - \widehat{ED^{-1}}D, \\ R_1 &= C - (\hat{A}_1 + \widehat{ED^{-1}}F). \end{aligned}$$

It is difficult to give a precise representation of the terms $r_{j,k}$ that is satisfied by every dropping rule, as well as every implementation. However it is expected that $r_{i,j}$ will be of the order of τ. In fact for an implementation based on 'elimination', the terms dropped are exactly the error terms above and therefore

$$|r_{j,k}| < \tau, \quad 1 \le j,k \le n. \tag{10}$$

We now discuss this implementation in some detail since it has useful properties and allows us to give an algorithmic interpretation of the Schur complement system. We call this technique a restricted IKJ version of Gaussian elimination. The rows associated with the independent set, i.e., rows 1 to m are eliminated using a variant of the IKJ version of Gaussian elimination [30]. The IKJ version of Gaussian elimination determines the i-th row of L and the i-th row of U in the i-th step of Gaussian elimination as follows:

Algorithm **3.1 Gaussian elimination – IKJ variant.**

```
1. do $i = 2, n$
2.        do $k = 1, i-1$
3.            $a(i,k) := a(i,k)/a(k,k)$.
4.            do $j = k+1, n$
5.                $a(i,j) := a(i,j) - a(i,k) * a(k,j)$.
```

```
6.            enddo.
7.        enddo.
8.  enddo.
```

Assume that the first m equations are associated with the independent set. Then, in order to perform the block factorization (3) it is sufficient to eliminate rows 1 to m but restrict the triangulation to the columns 1 through m. In other words we would obtain the following algorithm:

ALGORITHM **3.2 Restricted IKJ version of Gaussian elimination.**

```
1.  do $i = m+1, n$
2.        do $k = 1, min(i-1, m)$
3.            $a(i,k) := a(i,k)/a(k,k)$.
4.            do $j = k+1, n$
5.                $a(i,j) := a(i,j) - a(i,k) * a(k,j)$.
6.            enddo.
7.        enddo.
8.  enddo.
```

It is easy to see that this does indeed perform the block factorization (3) – in other words, the $a(i,k)$'s for $k < i$ are the elements in ED^{-1} and the other elements are those in A_1. We now add dropping to Line 3 (L-part):

```
3a.       if $|a(i,k)| < \tau$ then $a(i,k) := 0$,
```

and to the final row obtained after line 8 (U-part):

```
8a.       do $k = m+1, n$
8b.           if $|a(i,k)| < \tau$ then $a(i,k) := 0$.
8c.       enddo.
```

Call w the working row represented by $a(i,k), k = 1, \dots, n$, in the above algorithm with dropping. Initially w is the original i-th row of A. Then each elimination represented by Lines 4-6 combines this row with a row in the U-part of the factorization (occupied by $a(k,:)$ for $k < m$). The dropping rule in Line 3a actually changes w prior to this combination:

$$w(k) := w(k) - r(i,k).$$

Then the elimination in Lines 4-6 does an operation of the form,

$$w := w - l(i,k) * u(k,:).$$

At the end of all these operations, the dropping rule in Lines 8a-8c is applied:

$$w(k) := w(k) - r(i,k), \quad k = m+1, \dots, n,$$

in which $r(i,k)$ is zero unless the corresponding element is dropped in which case it is $a(i,k)$. The U-part of w becomes the i-th row of A_1. In the end,

$$u(i,*) = a(i,*) - \sum_{k=1}^{m} l(i,k) * u(k,*) - r(i,*).$$

And therefore,

$$a(i,*) = \sum_{k=1}^{i} l(i,k) * u(k,*) + r(i,*),$$

with the convention that $l(i,i) = 1$ and $l(i,j) = 0$ for $m < j < i$. In other words, with this implementation the R matrix is *exactly the matrix of the successive elements dropped during the elimination process.* The magnitude of these elements is less than τ for the simple dropping strategy (7).

The notation is simplified by removing the hat signs from representation (9) in the following discussion. Furthermore, we denote by $\text{nz}(A)$ the number of nonzeros in A, for an arbitrary matrix A.

Proposition 3.1 *When the simple dropping strategy (7) is applied with a threshold tolerance τ, then the factorization error of (9) is bounded by*

$$\|R\|_F \le \tau\ \text{nz}(R) \le \tau\sqrt{l(m+l)}. \tag{11}$$

The proof of this result is straightforward.

The bound (11) shows that the norm of the factorization error depends on the dropping tolerance and the number of elements dropped. The actual convergence rate of the preconditioned iteration is controlled by the so-called preconditioned error. The preconditioned matrix is

$$L^{-1}AU^{-1} = I + L^{-1}RU^{-1}.$$

The matrix $L^{-1}RU^{-1}$ is the preconditioned error matrix [14], which actually governs the convergence of the preconditioned iteration. It is easy to verify that

$$L^{-1} = \begin{pmatrix} I & 0 \\ -ED^{-1} & I \end{pmatrix}, \qquad U^{-1} = \begin{pmatrix} D^{-1} & -D^{-1}FA_1^{-1} \\ 0 & A_1^{-1} \end{pmatrix}.$$

The next result will require the following inequality on matrix norms:

$$\|XY\|_F \le \|X\|_F\ \|Y\|_2, \tag{12}$$

where $\|\cdot\|_F$ and $\|\cdot\|_2$ are the matrix Frobenius norm and 2-norm, respectively. (12) follows from the fact that

$$\begin{aligned} \|XY\|_F^2 &= \sum_{j=1}^{m} \|XY_{:,j}\|_2^2 \\ &\le \|X\|_2^2 \sum_{j=1}^{m} \|Y_{:,j}\|_2^2 \le \|X\|_2^2 \|Y\|_F^2. \end{aligned}$$

The desired result is obtained by applying the above inequality to the matrix Y^TX^T.

Proposition 3.2 *Assume that the dropping rule (7) is applied and A_1 is non-singular, and $\varepsilon < 1$. Then, the preconditioned error of ILUM with diagonal threshold tolerance is bounded by*

$$\|L^{-1}RU^{-1}\|_F \le \frac{\tau}{\varepsilon} \times \text{nz}(R)(1 + \|F\|_2) \times \max\{1, \|A_1^{-1}\|_2\}. \tag{13}$$

Proof. We have

$$\begin{aligned} L^{-1}RU^{-1} &= \begin{pmatrix} I & 0 \\ -ED^{-1} & I \end{pmatrix} \begin{pmatrix} 0 & 0 \\ R_0 & R_1 \end{pmatrix} \begin{pmatrix} D^{-1} & -D^{-1}FA_1^{-1} \\ 0 & A_1^{-1} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ R_0 & R_1 \end{pmatrix} \begin{pmatrix} D^{-1} & -D^{-1}FA_1^{-1} \\ 0 & A_1^{-1} \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ R_0 & R_1 \end{pmatrix} \begin{pmatrix} D^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -F \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & A_1^{-1} \end{pmatrix}. \end{aligned} \tag{14}$$

Using inequality (12) it follows that

$$\|L^{-1}RU^{-1}\|_F \leq \|R\|_F \times \max\{1, \|D^{-1}\|_2\} \times \left\| \begin{pmatrix} I & -F \\ 0 & I \end{pmatrix} \right\|_2 \times \max\{1, \|A_1^{-1}\|_2\}.$$

It can be easily shown that
$$\left\| \begin{pmatrix} I & -F \\ 0 & I \end{pmatrix} \right\|_2 \leq 1 + \|F\|_2.$$

Hence the desired result:

$$\|L^{-1}RU^{-1}\|_F \leq \mathrm{nz}(R) \times \frac{\tau}{\varepsilon} \times (1 + \|F\|_2) \times \|A_1^{-1}\|_2.$$

□

Proposition 3.3 *The nonzero eigenvalues of the preconditioned error matrix are identical with the eigenvalues of the generalized problem*

$$(R_1 - R_0 D^{-1} F)z = \lambda\ A_1 z. \tag{15}$$

Assuming that there exists $\gamma > 0$ *such that,* $(A_1 x, x) \geq \gamma(x, x)$ *then*

$$|\lambda| \leq \frac{\tau}{\gamma} \left(\mathrm{nz}(R_1) + \frac{\mathrm{nz}(R_0)\|F\|_2}{\varepsilon} \right). \tag{16}$$

Proof. Starting from (14)

$$L^{-1}RU^{-1} = \begin{pmatrix} 0 & 0 \\ R_0 D^{-1} & (R_1 - R_0 D^{-1} F) A_1^{-1} \end{pmatrix}.$$

The eigenvalues of the above matrix consist of a multiple eigenvalue of zero and the eigenvalues of the matrix $(R_1 - R_0 D^{-1} F)A_1^{-1}$ which are the same as the eigenvalues of the generalized problem (15) The bound (16) follows from the fact that

$$|\lambda| = \left| \frac{((R_1 - R_0 D^{-1} F)z, z)}{(A_1 z, z)} \right| \leq \left| \frac{(R_1 z, z)}{(A_1 z, z)} \right| + \left| \frac{(R_0 D^{-1} F z, z)}{(A_1 z, z)} \right|.$$

The result follows by exploiting the inequality

$$|(Xz, z)| \leq \|Xz\|_2 \|z\|_2 \leq \|X\|_F \|z\|_2^2$$

for an arbitrary matrix X. □

It is interesting to note that the eigenvalues of the error matrix involve two similar Schur complements: A_1 and the Schur complement of

$$\begin{pmatrix} D & F \\ R_0 & R_1 \end{pmatrix}.$$

If a few assumptions were to be added on the nature of the matrix A (symmetric positive definiteness), it could be shown that the γ constant for exact Schur complement A_1 will remain bounded from below.

Since the preconditioned error determines the convergence rate of the preconditioned iteration, by examining the bound (13), the following observations on the convergence behavior of ILUM with diagonal threshold tolerance can help selecting parameters for constructing ILUM:

1. An accurate factorization with small τ yields fast convergence;

2. Given a dropping threshold τ, a small diagonal threshold tolerance will usually deteriorate convergence;

3. Iterative processes are likely to be slower for larger systems (with large n, m and l). However, this may be offset by a small reduced system (small l). The size of the reduced system relative to the size of the independent set seems to influence the convergence, provided the (last) reduced system can be solved accurately;

4. The quality of the solution of the reduced system has a strong influence on convergence. However, highly accurate solutions of the reduced system are expensive (assuming an iterative process is used);

5. Sparsity of the F block influences convergence. The sparser F, the faster the convergence.

The fact that the sparsity of F influences the convergence can also provide some justification to the increasing degree traversal algorithm. By visiting the nodes with small degree first, F is likely to be sparser, leading, in general, to a larger independent set. This observation has been confirmed by our numerical results in [29, 32].

Although several parameters have an effect on the convergence rate, it is the size of the reduced system and its minimization that seem to draw most of the recent attention. This leads to the development of various blocking strategies and dropping rules [31, 32].

4 Numerical Experiments

Standard implementations of ILUM and BILUM have been described in detail in [29, 32]. The iteration process consists of an outer iteration, which is the main preconditioned process to solve the underlying linear system, and an inner iteration, which is the secondary preconditioned process to solve the last reduced system. We used FGMRES(10) as the accelerator for both the inner and outer iterations [27]. The outer iteration process was preconditioned by ILUM with diagonal threshold tolerance as discussed in this paper, using 10 levels of reduction. The inner iteration process for solving the last reduced system approximately was preconditioned by a dual-threshold ILUT(τ, p) [28]. Unless otherwise stated explicitly, we chose $\tau = 10^{-4}, p = 20$ for ILUT. The same τ was also used as the dropping threshold tolerance in the construction of ILUM. In each table, the last column with title 'ILUT' shows the results when the (single-level) ILUT with the same parameters was used to solve the whole system. Although the single-level ILUT used less memory than ILUM in this case, the results do give us some indication of the robustness of the multi-level preconditioned method versus single-level method. Formula (6) was not used in our tests. The construction and application of ILUM preconditioner was described in [32], but here we applied the dropping rules (8) from the first level (the dropping rules were applied starting from the 2nd level only in [32]).

For all linear systems, the right-hand side was generated by assuming that the solution is a vector of all ones. The initial guess was a vector of some random numbers. The inner iteration was stopped when the (inner iteration) residual in 2-norm was reduced by a factor of 10^2 or the number of iterations exceeded 10, whichever was reached first. The outer iteration was stopped when the 2-norm of the residual was reduced by a factor of 10^7. We also set an upper bound of 100 for the outer FGMRES(10) iteration. (A symbol $\dagger$ in a table indicates that convergence was not reached in 100 outer iterations.)

The FIDAP matrices and the matrices from the driven cavity problem can be hard to solve. In some situations, it is beneficial to scale both columns and rows before constructing the preconditioner [13]. However, we did not resort to scaling or permuting for any matrix before computing the preconditioner. For ILUM without diagonal threshold tolerance ($\varepsilon = 0$), we set a safeguard to prevent the inversion of D from breaking down by putting $d_i = 10^{-16}$ whenever $|d_i| < 10^{-16}$.

The numerical experiments were conducted on a Power-Challenge XL Silicon Graphics workstation equipped with 512 MB of main memory, two 190 MHZ R10000 processors, and 1 MB secondary cache. We used Fortran 77 programming language in 64-bit precision.

4.1 FIDAP matrices

Our first set of test matrices were extracted from the test problems provided in the FIDAP package [18]. The examples tested model the incompressible Navier-Stokes equations. The right bottom sub-block of these matrices may be a zero block [13] and many of these matrices have small or zero diagonals and are very hard to solve with standard ILU preconditioners. Table 1 is a simple description of the FIDAP matrices [1] with n being the dimension of the matrix, nz its number of non-zero elements.

We varied the diagonal threshold tolerance ε and recorded the number of iterations in Table 2 for the greedy algorithm and in Table 3 for the increasing degree traversal algorithm. Without diagonal tolerance, ILUM could solve 4 out of 17 systems with the greedy algorithm and 5 out of 17 with the increasing degree traversal algorithm. With some diagonal tolerance, ILUM can solve 16 matrices with the greedy algorithm and all 17 matrices with the increasing degree traversal algorithm. However, it seems that ILUM with the greedy algorithm converged in more cases than with the increasing degree traversal algorithm. Also, the algorithm seems to be most robust when the diagonal threshold tolerance ε was chosen close to the dropping tolerance τ of ILUM. In both cases, the single-level ILUT performed similarly to ILUM without diagonal threshold tolerance.

[1]Matrices available online from the MatrixMarket (`http://math.nist.gov/MatrixMarket`). Some values for nz in Table 1 are larger than those listed in Table 4.2 of [13] and those of the same set in the MatrixMarket because all diagonal elements of the right bottom sub-blocks are kept even though their actual values may be numerically zero.

Matrix	n	nz	Matrix	n	nz
EX01	216	4 352	EX04	1 601	32 299
EX12	3 973	80 211	EX20	2 203	69 981
EX21	656	19 144	EX22	839	22 715
EX23	1 409	43 703	EX24	2 283	48 737
EX25	848	24 612	EX26	2 163	94 033
EX27	974	40 782	EX28	2 603	77 781
EX29	2 870	23 754	EX31	3 909	115 357
EX32	1 159	11 343	EX36	3 079	53 843
EX37	3 565	67 591			

Table 1: Size and number of nonzero elements for the FIDAP matrices.

ε	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-16}	0.0	ILUT
EX01	3	4	3	3	3	3	3	3	3	3	9
EX04	11	4	4	21	†	†	†	†	†	†	†
EX12	45	45	45	†	†	†	†	†	†	†	†
EX20	†	†	†	5	6	6	7	†	†	†	†
EX21	†	†	30	50	21	29	†	†	†	98	†
EX22	3	4	7	7	7	11	†	19	19	†	22
EX23	†	†	†	†	†	†	†	†	†	†	†
EX24	6	†	†	†	†	†	†	†	†	†	†
EX25	†	†	†	10	†	7	†	†	†	†	†
EX26	†	†	†	62	†	†	†	†	†	†	†
EX27	†	†	8	4	†	3	20	†	†	†	†
EX28	9	8	†	†	†	†	†	†	†	†	†
EX29	2	2	2	2	2	2	2	2	2	2	3
EX31	4	4	12	16	16	16	16	16	14	†	31
EX32	4	4	5	6	6	6	6	6	6	†	41
EX36	4	4	4	†	†	†	†	†	†	†	†
EX37	11	10	10	10	10	9	9	10	11	11	5

Table 2: Number of iterations for the greedy algorithm with different diagonal threshold tolerance ε for solving the FIDAP matrices.

ε	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-16}	0.0	ILUT
EX01	3	4	3	3	3	3	3	3	3	3	9
EX04	11	4	4	†	†	†	†	†	†	†	†
EX12	50	40	47	†	†	†	†	†	†	†	†
EX20	†	†	†	6	5	5	7	†	†	90	†
EX21	†	†	†	†	56	†	43	†	†	†	†
EX22	3	4	4	18	†	†	†	†	†	†	22
EX23	†	†	†	†	†	†	†	†	†	19	†
EX24	6	†	†	†	†	†	†	†	†	†	†
EX25	†	†	†	†	5	7	†	†	†	†	†
EX26	†	†	†	16	†	†	†	†	†	†	†
EX27	†	†	8	4	3	4	90	†	†	†	†
EX28	30	17	†	†	†	†	†	†	†	†	†
EX29	2	2	2	2	2	2	2	2	2	2	3
EX31	4	3	4	4	†	†	†	†	†	†	31
EX32	4	4	4	†	†	†	†	†	†	†	41
EX36	4	4	4	†	†	†	†	†	†	†	†
EX37	11	10	10	10	10	9	9	10	11	11	5

Table 3: Number of iterations for the increasing degree traversal algorithm with different diagonal threshold tolerance ε for solving the FIDAP matrices.

Results of Tables 2 and 3 show that ILUM with diagonal threshold tolerance is more robust than ILUM without diagonal tolerance.

As to the efficiency of forming independent set, the increasing degree traversal algorithm seems slightly more efficient than the greedy algorithm in the sense that the last reduced system which it yields has a smaller size. Figure 1 describes the relation between the size of the last reduced system and the value of the diagonal threshold tolerance for two FIDAP matrices. (For easy visualization, we actually plotted the iteration counts against the reciprocal of the diagonal threshold tolerance ($1/\varepsilon$).) For EX20, both algorithms produced a last reduced systems of comparable size except when ε is very small. The reduced systems are quite different for ILUM without diagonal threshold tolerance. For EX31, the increasing degree traversal algorithm produced an obviously smaller last reduced system. However, comparison of convergence results of Tables 2 and 3 indicates that the greedy algorithm is more robust than the increasing degree traversal algorithm, despite of its delivering slightly smaller independent sets. Hence, we prefer the greedy algorithm in general.

Figure 2 shows the convergence history (the residual in 2-norm against the number of iterations) of ILUM for solving the EX31 matrix with and without diagonal threshold tolerance. We see that ILUM with suitable diagonal threshold tolerance converged fast. The convergence of ILUM without diagonal threshold tolerance was fast for the first few iterations, but quickly slowed down and started stagnating.

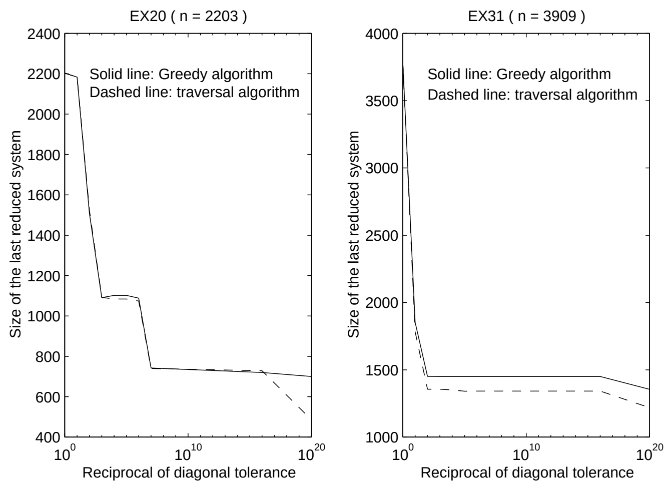

Figure 1: Size of the last reduced system as a function the reciprocal of the diagonal threshold tolerance $1/\varepsilon$ generated by the greedy algorithm and the increasing degree traversal algorithm for two FIDAP matrices.

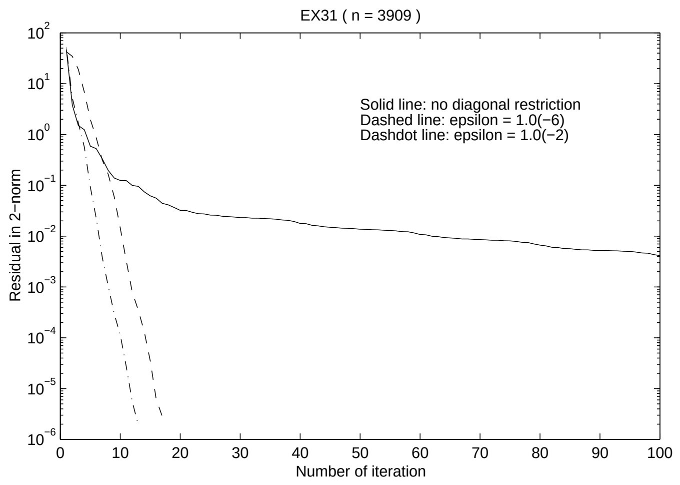

Figure 2: Convergence history of ILUM with and without diagonal threshold tolerance for solving the EX31 matrix.

Matrix	n	nz	Description
RAEFSKY1	3 242	294 276	Incompressible flow in pressure driven pipe, T=05
RAEFSKY2	3 242	294 276	Incompressible flow in pressure driven pipe, T=25
RAEFSKY3	21 200	1 488 768	Fluid structure interaction turbulence problem
RAEFSKY5	6 316	168 658	Landing hydrofoil airplane FSE model
RAEFSKY6	3 402	137 845	Slosh tank model
VENKAT01	62 424	1 717 792	Unstructured 2D Euler solver, time step = 0
VENKAT25	62 424	1 717 792	Unstructured 2D Euler solver, time step = 25
VENKAT50	62 424	1 717 792	Unstructured 2D Euler solver, time step = 50

Table 4: Description of the Simon matrices.

ε	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-16}	0.0	ILUT
RAEFSKY1	7	7	7	7	7	7	7	7	7	7	†
RAEFSKY2	10	10	10	10	10	10	10	10	10	10	†
RAEFSKY3*	†	4	4	†	5	†	†	6	5	5	19
RAEFSKY5	2	2	2	2	2	2	2	2	2	2	3
RAEFSKY6	2	2	2	3	3	3	3	2	3	2	3
VENKAT01	-	-	-	3	3	4	4	4	4	3	10
VENKAT25‡	-	-	-	12	16	17	16	16	16	16	93
VENKAT50‡	-	-	-	19	36	40	34	33	34	34	†

Table 5: Number of iterations for the greedy algorithm with different diagonal threshold tolerance ε for solving the Simon matrices. "‡" indicates $\tau = 10^{-2}$ was used in the reduction. "*" indicates $p = 30$ was used in ILUT. "-" indicates insufficient memory for ILUT.

4.2 Simon matrices

Our next set of test matrices are the so-called Simon matrices. These matrices were supplied by H. Simon and other researchers and are generally larger than other sets of matrices. They are all from applications in computational fluid dynamics and have been used as test matrices for high order ILU preconditioners in [12, 31]. Table 4 gives a brief description of these matrices. Table 5 lists the number of iterations. Since some of these matrices are very large, we had to adjust some parameters in ILUM and ILUT in order to solve them. These adjustments are also indicated in Table 5. For the VENKAT matrices with large ε, the last reduced system were too large with the parameters set in ILUT. We abandoned our attempt to solve them in these cases.

Since the Simon matrices in general do not have a large number of small or zero diagonals, ILUM with and without diagonal tolerance performed comparably. These results also indicate that ILUM may also be very efficient for solving large linear systems. It does not hurt to impose suitable diagonal threshold tolerance, but a very large diagonal threshold may result in a large reduced system that is hard to solve. For this set of test matrices, the single-level ILUT is not as good as ILUM with or without diagonal threshold tolerance.

Figure 3 depicts the CPU time in seconds for solving the RAEFSKY1 and RAEFSKY5

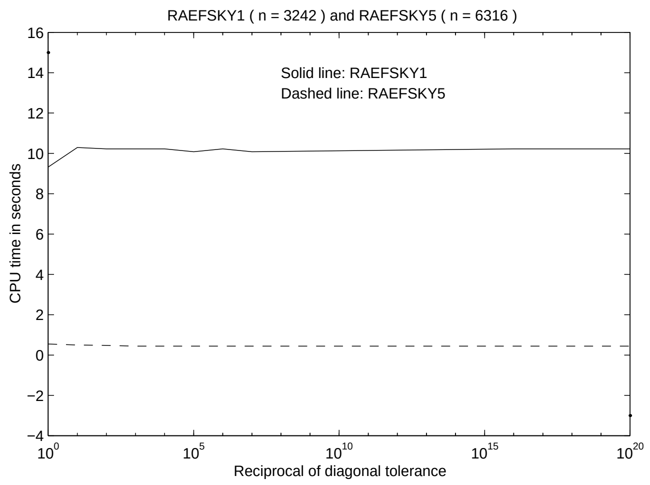

Figure 3: CPU time in seconds against the reciprocal of the diagonal threshold tolerance for solving the RAEFSYK1 and RAEFSKY5 matrices.

Matrix	n	nz	Matrix	n	nz
ADD20	2 395	17 319	FS7602	760	5 976
ORSIRR1	1 030	6 858	ORSIRR2	886	5 970
ORSREG1	2 205	14 133	PDE9511	961	4 681
PORES2	1 224	9 613	SAYLR4	3 564	22 316
WATT1	1 856	11 360	WATT2	1 856	11 550

Table 6: Description of the Harwell-Boeing matrices.

matrices using different diagonal threshold tolerance. We see that there is no much difference in the CPU time by using different diagonal tolerances *if the convergence rate is the same.*

4.3 Harwell-Boeing collection

The third set of test matrices was taken from the Harwell-Boeing collection [16, 17]. (The only exception is that the matrix ADD20 was taken from NIST's MatrixMarket.[2]) Many of these matrices have been used as test matrices for iterative sparse matrix solvers [29, 32]. The description of these Harwell-Boeing matrices is listed in Table 6 and the test results are listed in Table 7. Since most of these matrices have large absolute diagonal values, the iteration counts do not vary much. For FS7602 and WATT2, ILUM performed poorly without diagonal threshold tolerance. These test results also indicate that it normally does not hurt convergence to use diagonal thresholding in ILUM, even when the diagonal entries are large. Furthermore, ILUM outperformed the single-level ILUT for most problems in this set of test matrices.

[2]Available online at http://math.nist.gov/MatrixMarket.

ε	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-16}	0.0	ILUT
ADD20	4	3	3	3	3	3	3	3	3	3	4
FS7602	5	7	6	7	29	17	17	17	17	17	†
ORSIRR1	4	4	4	4	4	4	4	4	4	4	6
ORSIRR2	4	4	4	4	4	4	4	4	4	4	6
ORSREG1	3	3	3	3	3	3	3	3	3	3	4
PDE9511	3	3	3	3	3	3	3	3	3	3	5
PORES2	20	20	20	20	20	20	20	20	20	20	38
SAYLR4	8	8	8	8	8	8	8	8	8	8	†
WATT1	1	1	1	1	1	1	1	1	1	1	2
WATT2	1	1	1	1	1	1	1	1	42	42	25

Table 7: Number of iterations for the greedy algorithm with different diagonal threshold tolerance ε for solving the Harwell-Boeing matrices.

4.4 Driven cavity problem

Our last set of test problems was from a finite element discretization of the square driven cavity problem. Rectangular elements were used, with biquadratic basis functions for velocities, and linear discontinuous basis functions for pressure. All matrices arise from a mesh of 20 by 20 elements, leading to matrices of size $n = 4562$ and having $nz = 138,187$ nonzero entries. These matrices have 3363 velocity unknowns and 1199 pressure unknowns. The sub-matrices associated with the velocity unknowns have small or zero diagonals. We tested 11 matrices with Reynolds number (Re) from 0 to 1000.

To ensure convergence of the coarsest level solution for these matrices, we had to use more fill-ins for ILUT on the coarsest (10th) level. The parameter $\tau = 10^{-4}$ was kept as before for the dropping rule in the ILUM reduction and ILUT, but the value of p was increased until at least we had one convergence. The iteration counts as a function of the diagonal threshold tolerance, along with the p values used, are listed in Table 8.

Table 8 shows that without diagonal threshold tolerance, ILUM did not converge for any test problems. This was precisely caused by the small and zero diagonals of the matrices. There are so many such diagonals that simply replacing them by a small value in the construction of ILUM deteriorated the quality of the preconditioner. The resulting ILUM preconditioner had little preconditioning effect on the iterative solver. This is a case when ILUM with diagonal threshold tolerance strongly outperformed ILUM without diagonal threshold tolerance and the single-level ILUT.

We also tested how the diagonal threshold tolerance ε affects the construction of ILUM. We fixed the coarsest level ILUT as $\tau = 10^{-6}, p = 40$. We then used different τ as the dropping threshold tolerance to control the sparsity of ILUM. The problem we solved is the driven cavity problem with Re = 200. We took $\tau = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$ and recorded the number of iterations in Figure 4. Note again that the iteration numbers were actually plotted against the reciprocal of the diagonal threshold tolerance $(1/\varepsilon)$. It is seen that the value of ε did influence the quality of the ILUM preconditioner. It seems that the optimal value of ε is related to the value of τ used as dropping rule in constructing the ILUM factorization. Since our dropping rule is based on the average absolute value of the current

Re	p	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-16}	0.0	ILUT
0	30	20	20	20	21	21	†	†	†	†	†	†
100	30	20	20	19	20	81	†	†	†	†	†	†
200	40	25	25	26	26	49	†	†	†	†	†	†
300	60	50	58	50	39	61	40	39	40	38	†	†
400	70	35	35	35	38	51	†	†	†	†	†	†
500	80	20	21	20	21	27	22	22	20	21	†	†
600	80	31	29	27	25	61	45	50	54	49	†	†
700	80	91	90	91	98	†	†	†	†	†	†	†
800	90	20	20	20	19	67	37	47	50	36	†	†
900	90	51	58	55	58	†	†	†	†	†	†	†
1000	90	72	†	83	71	†	†	†	†	†	†	†

Table 8: Number of iterations for the greedy algorithm with different diagonal threshold tolerance ε for solving the driven cavity matrices. p was different for each matrix.

row (8), there is no exact match between τ and ε, but we again suggest to choose τ around ε. This also indicates that more accurate ILUM factorization (smaller τ) works better with smaller diagonal tolerances (smaller ε).

Figure 5 shows the CPU time in seconds of the same test as in Figure 4. An iteration number of 100 in Figure 4 means lack of convergence and this is reflected in Figure 5 by a large CPU time (100 seconds). Figure 5 looks similar to Figure 4, showing that there is no obvious penalty in computational efficiency by using a diagonal threshold tolerance.

Figure 6 shows the convergence history of the ILUM preconditioner for solving the driven cavity problem (Re = 800) with and without diagonal threshold tolerance. We see that ILUM with suitable diagonal tolerance converged fast. Unlike the convergence displayed in Figure 2 for the EX31 matrix, the convergence of ILUM for solving this driven cavity matrix without diagonal threshold tolerance just stagnated, with little reduction in residual norm in 100 iterations.

5 Conclusion

We have presented a number of techniques for incorporating diagonal threshold strategies into heuristic algorithms for finding independent sets in constructing ILUM preconditioners. The new algorithms work by essentially never accepting into the independent sets any diagonal elements that are too small. These elements are therefore moved down the matrix and they are either modified during the elimination to become acceptable later in the process, or they remain in the last reduced system which is solved with high accuracy with a standard ILUT preconditioned Krylov technique or a direct solution method.

These heuristics have been tested to show improved robustness for hard-to-solve systems arising from finite element discretization of flow problems in computational fluid dynamics. Many of these problems could not be solved by ILUM without diagonal threshold tolerance or by the single-level ILUT. The numerical tests also reinforce earlier conclusions made on this type of methods and show the advantages of the multi-level preconditioned methods over their single-level counterparts.

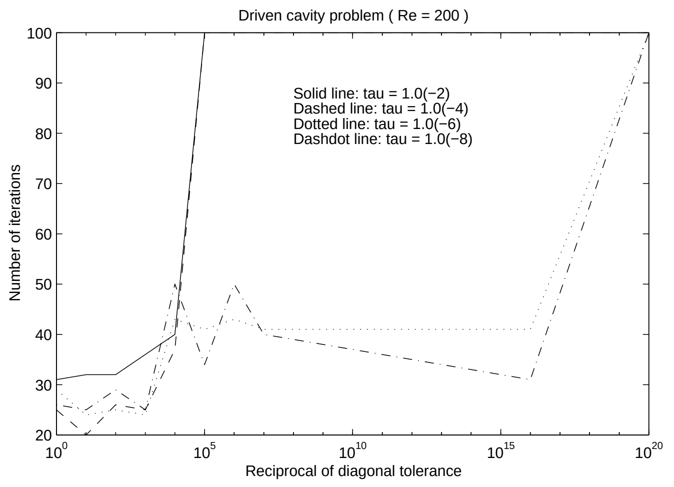

Figure 4: Comparison of iteration counts against the reciprocal of the diagonal threshold tolerance with different τ used in constructing ILUM. Driven cavity problem with Re = 200.

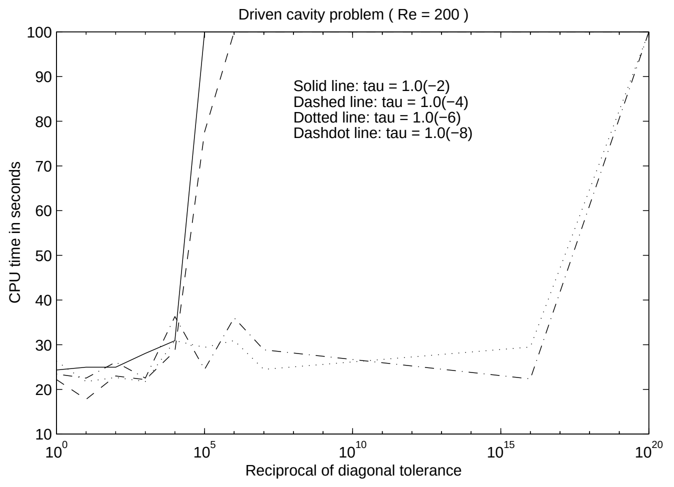

Figure 5: Comparison of CPU time in seconds against the reciprocal of the diagonal threshold tolerance with different τ used in constructing ILUM. Driven cavity problem with Re = 200.

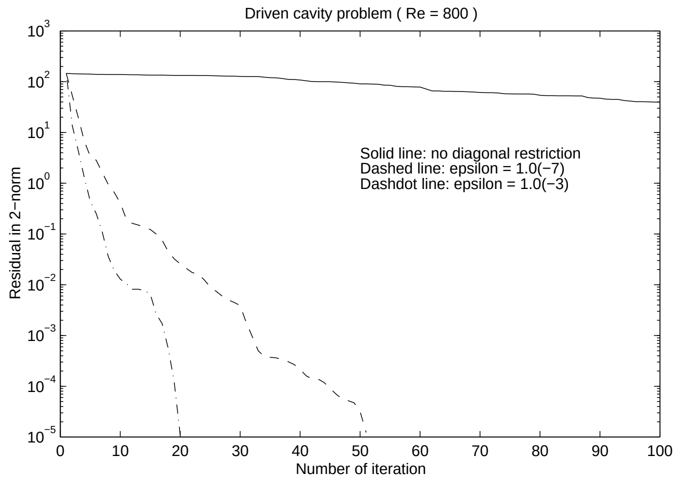

Figure 6: Convergence history of ILUM with and without diagonal threshold tolerance. Driven cavity problem with Re = 800.

The analytical bounds for the factorization and preconditioned errors proved in Section 3 are limited to a two-level factorization. Yet they provide some insight on how to select parameters in practical implementations of ILUM. The main insight provided by these analytical bounds is that the value of the diagonal threshold tolerance ε should be chosen to be around the value of the threshold tolerance τ used as for the dropping strategy in ILUM. Our numerical experiments confirmed that this is indeed an effective strategy.

Because of the restrictions imposed by thresholding, it may well happen that after a few levels of reduction, the resulting independent set is very small. In these cases, it is preferable to stop the reduction process and try to solve the reduced system on that level. Keeping a small independent set and continuing the reduction process can require too much memory.

An alternative to avoid small or zero diagonals is to perturb diagonal entries. These perturbations introduce errors into the upper parts of the error matrix. However, this approach does not exclude the nodes in question from the independent set and the resulting independent set may be larger. (This is not absolutely true for all cases, since a node with small diagonal value may have a neighbor with large diagonal value which is to be excluded from the independent set by this approach.) We intend to compare this alternative with our diagonal threshold tolerance strategy in the future.

References

[1] O. Axelsson and P. S. Vassilevski, A survey of multilevel preconditioned iterative methods, *BIT* **29** (4), 769–793 (1989).

[2] O. Axelsson and V. Eijkhout, The nested recursive two level factorization for nine-point difference matrices, *SIAM J. Sci. Stat. Comput.* **12**, 1373–1400 (1991).

[3] R. E. Bank and J. Xu, The hierarchical basis multigrid method and incomplete LU decomposition, in *Seventh International Symposium on Domain Decomposition Methods for Partial Differential Equations,* D. Keyes and J. Xu, eds., AMS, Providence, RI, pp. 163–173 (1994).

[4] R. E. Bank and C. Wagner, Multilevel ILU decomposition, Technical Report, Department of Mathematics, University of California at San Diego, La Jolla, CA, 1997.

[5] E. F. F. Botta, A. van der Ploeg and F. W. Wubs, A fast linear-system solver for large unstructured problems on a shared-memory computer, in *Proceedings of the Conference on Algebraic Multilevel Methods with Applications,* O. Axelsson and B. Polman, eds., pp. 105–116 (1996).

[6] E. F. F. Botta, A. van der Ploeg, and F. W. Wubs, Nested grids ILU-decomposition (NGILU), *J. Comput. Appl. Math.* **66**, 515–526 (1996).

[7] E. F. F. Botta, K. Dekker, Y. Notay, A. van der Ploeg, C. Vuik, F. W. Wubs, and P. M. de Zeeuw, How fast the Laplace equation was solved in 1995, *Appl. Numer. Math.* **24**, 439–455 (1997).

[8] E. F. F. Botta and F. W. Wubs, MRILU: it's the preconditioning that counts, Technical Report W-9703, Department of Mathematics, University of Groningen, The Netherlands, 1997.

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

[10] T. F. Chan, S. Go, and J. Zou, Multilevel domain decomposition and multigrid methods for unstructured meshes: algorithms and theory, Technical Report 95-24, Department of Mathematics, University of California at Los Angeles, Los Angeles, CA, 1995.

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

[12] A. Chapman, Y. Saad, and L. Wigton, High order ILU preconditioners for CFD problems, Technique Report UMSI 96/14, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1996.

[13] E. Chow and Y. Saad, Approximate inverse techniques for block-partitioned matrices, *SIAM J. Sci. Comput.* **18**, 1657–1675 (1997).

[14] E. Chow and Y. Saad, Experimental study of ILU preconditioners for indefinite matrices, *J. Comput. Appl. Math.* **86** (2), 387–414 (1997).

[15] P. M. de Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, *J. Comput. Appl. Math.* **33**, 1–27 (1990).

[16] I. S. Duff, R. G. Grimes and J. G. Lewis, Sparse matrix test problems, *ACM Trans. Math. Software* **15**, 1–14 (1989).

[17] I. S. Duff, R. G. Grimes, and J. G. Lewis, *User's Guide for the Harwell-Boeing Sparse Matrix Collection,* Technical Report TR/PA/92/86, CERFACS, Toulouse, France, 1992.

[18] M. Engelman, *FIDAP: Examples Manual, Revision 6.0,* Fluid Dynamics International, Evanston, IL, 1991.

[19] J. A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[20] G. H. Golub and H. A. van der Vorst, Closer to the solution: iterative linear solvers, in *The State of the Art in Numerical Analysis,* I. S. Duff and G. A. Watson, eds., Clarendon Press, Oxford, pp. 63–92, 1997.

[21] Y. Notay and Z. Ould Amar, A nearly optimal preconditioning based on recursive red-black orderings, *Numer. Linear Algebra Appl.* **4**, 369–391 (1997).

[22] Y. Notay and Z. Ould Amar, Incomplete factorization preconditioning may lead to multigrid like speed of convergence, in *Advanced Mathematics: Computation and Applications,* A. S. Alekseev and N. S. Bakhvalov, eds., NCC Publisher, Novosibirsk, Russia, pp. 435–446, 1996.

[23] A. A. Reusken, Multigrid with matrix-dependent transfer operators for convection-diffusion problems, in *Multigrid Methods IV,* Proceedings of 4th European Multigrid Conference, Amsterdam, P.W. Hemker and P. Wesseling, eds., Birkhauser Verlag, Basel, pp. 269–280, 1994.

[24] A. A. Reusken, Approximate cyclic reduction preconditioning, Technical Report RANA 97-02, Department of Mathematics and Computing Science, Eindhoven University of Technology, The Netherlands, 1997.

[25] J. W. Ruge and K. Stüben, Efficient solution of finite difference and finite element equations, in *Multigrid Methods for Integral and Differential Equations*, (D. J. Paddon and H. Holstein, eds.), Clarendon Press, Oxford, 169–212 (1985).

[26] J. W. Ruge and K. Stüben, Algebraic multigrid, in *Multigrid Methods, Vol. 4*, S. F. McCormick, ed., SIAM, Philadelphia, PA, pp. 73–130, 1987.

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

[28] Y. Saad, ILUT: a dual threshold incomplete ILU preconditioner, *Numer. Linear Algebra Appl.* **1**, 387–402 (1994).

[29] Y. Saad, ILUM: a multi-elimination ILU preconditioner for general sparse matrices, *SIAM J. Sci. Comput.* **17**, 830–847 (1996).

[30] Y. Saad, *Iterative Methods for Sparse Linear Systems,* PWS Publishing Co., Boston, 1996.

[31] Y. Saad, M. Sosonkina, and J. Zhang, Domain decomposition and multi-level type techniques for general sparse linear systems, Technical Report UMSI 97/244, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1997.

[32] Y. Saad and J. Zhang, BILUM: block versions of multi-elimination and multi-level ILU preconditioner for general sparse linear systems, Technical Report UMSI 97/126, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1997.

[33] A. van der Ploeg, *Preconditioning for Sparse Matrices with Applications,* Ph.D. thesis, University of Groningen, The Netherlands, 1993.