TOWARDS A COST-EFFECTIVE ILU PRECONDITIONER WITH HIGHER LEVEL FILLS [*]

E. F. D'AZEVEDO [†], P. A. FORSYTH[‡] AND WEI-PAI TANG [‡]

Abstract. A recently proposed Minimum Discarded Fill (*MDF*) ordering (or pivoting) technique is effective in finding high quality $ILU(\ell)$ preconditioners, especially for problems arising from unstructured finite element grids. This algorithm can identify anisotropy in complicated physical structures and orders the unknowns in a "preferred" direction. However, the *MDF* ordering is costly, when ℓ increases.

In this paper, several less expensive variants of the *MDF* technique are explored to produce cost-effective *ILU* preconditioners. The Incomplete *MDF* and Threshold *MDF* orderings combine *MDF* ideas with drop tolerance techniques to identify the sparsity pattern in the *ILU* preconditioners. These techniques produce orderings that encourage fast decay of the entries in the *ILU* factorization. The Minimum Update Matrix (*MUM*) ordering technique is a simplification of the *MDF* ordering and is an analogue of the minimum degree algorithm. The *MUM* ordering is especially effective for large matrices arising from Navier-Stokes problems.

Key Words. minimum discarded fill(*MDF*), incomplete *MDF*, threshold *MDF*, minimum updating matrix(*MUM*), incomplete factorization, matrix ordering, preconditioned conjugate gradient, high-order *ILU* factorization.

AMS(MOS) subject classification. 65F10, 76S05

1. Introduction.

The use of Preconditioned Conjugate Gradient (PCG) methods has proven to be a robust and competitive solution technique for large sparse matrix problems [1, 22, 28]. A vital step for the successful application of PCG methods is the computation of a high quality preconditioner. Many previous studies have explored this topic and their various approaches can be summarized as follows:

- When the incomplete LU (ILU) preconditioner was first proposed it was observed that as the fill level ℓ in an ILU decomposition increases, the quality of the $ILU(\ell)$ preconditioner improves [1, 19, 22, 25, 29]. Unfortunately, the resulting reduction in the number of iterations cannot compensate for the increased cost of the factorization, and forward and backward solve in each iteration. The most cost-effective fill level is $\ell = 1, 2$ in most cases. For some problems, the high-level fill entries in the ILU decomposition are numerically very small and contribute little to the quality of the preconditioner. This observation leads naturally to the improved technique described below.
- Based on this observation with the high-level-fill approach, a drop tolerance technique was investigated by Munksgaard [30] and Zlatev [37]. The strategy

[*] This work was supported by the Natural Sciences and Engineering Research Council of Canada, by the Information Technology Research Centre, which is funded by the Province of Ontario, and by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc., through an appointment to the U.S. Department of Energy Postgraduate Research Program administered by Oak Ridge Associated Universities.

[†] Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

[‡] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

for deciding the sparsity pattern of a preconditioner was to discard all "small" fill entries that are less than a given tolerance. This approach works well for the model Laplace problem. However, it was soon noticed that if the original ordering was "poor", then the decay of the fill entries in the decomposition was very slow, sometimes leading to an unacceptably dense decomposition. Consider the following problem,

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial P}{\partial y}\right) = -q \tag{1}$$

with a 5-point discretization on the regular grid, where $K_x,\ K_y > 0$. When $K_x \gg K_y$, the drop tolerance approach will produce a very dense preconditioner, if a natural row ordering is used. In contrast, a cost-effective preconditioner is yielded using the same approach, when $K_y \gg K_x$ [8]. Interesting results concerning the effects of ordering on the performance of *PCG* were reported recently in [8, 14, 12, 13, 15, 31].

- Greenbaum and Rodrigue [18] addressed the problem of computing an optimal preconditioner for a given sparsity pattern. The values of the nonzero elements in the preconditioner were determined by numerical optimization techniques for a given sparsity pattern. Kolotilina and Yeremin investigated some least squares approximations for block incomplete factorization [24]. Their results provide insights of a theoretical nature, but have limited practical application.
- Numerous special techniques that produce good preconditioners for elliptic problems and domain decomposition methods have also been reported [3, 4, 20, 23]. These approaches are very successful for applying *PCG* to the solutions of elliptic PDE's. However, they are difficult to generalize to Jacobians arising from systems of PDE's and to general sparse matrix problems.

In summary, the following factors are observed to have a significant impact on the quality of a preconditioner:

1. The ordering of the unknowns in the original matrix A.
2. The sparsity pattern of the preconditioner M.
3. How closely the spectrum of M resembles that of A.

Motivated by the significant effect of ordering on the preconditioner, we have proposed the Minimum Discarded Fill ($MDF(\ell)$) ordering (or pivoting strategy) [8] for general sparse matrices. This ordering technique is effective in finding high quality $ILU(\ell)$ preconditioners, especially for problems arising from unstructured finite element grids. This algorithm can identify anisotropy in complicated physical structures and orders the unknowns in a "preferred" direction. Numerical testing of the $MDF(\ell)$ method shows that it yields better convergence performance than some other orderings. The $MDF(\ell)$ ordering is successful because it takes into account the numerical values of matrix entries during the factorization process, and not just the topology of the mesh. However, the $MDF(\ell)$ ordering may be costly to produce. A rough estimate of the cost of $MDF(\ell)$ ordering is Nd^3, where d is the average number of nonzero elements in each row of the fill matrix, $L + L^t$, and N is the size of the original matrix [1]. Thus, if the average number of nonzero in each row is large, the *MDF*(ℓ) ordering is practical only when similar matrix problems need to be solved several times, such as solving the Jacobian matrices in a Newton iteration [5, 7].

In this paper, several variants of the *MDF* ordering algorithm for use with a drop tolerance incomplete factorization are investigated. Section 2 contains a detailed description of these ordering techniques. Test problems are described in Section 3 and the numerical results and discussion are provided in Section 4, with our concluding remarks in the last section.

Results from applying the *MDF* algorithm and its variants to unsymmetric matrices derived from a system of PDE's will be discussed in another paper [5].

2. Algorithms .

2.1. ***ILU*(ℓ) factorization.** Let us define the nonzero sparsity pattern of a matrix C as the the set

$$NZ\,[C] = \{(i,j) \mid c_{ij} \neq 0\}\,. \tag{2}$$

Given a sparsity pattern $\mathcal{P}$, denote $\tilde{C} := C[\mathcal{P}]$ to be the matrix extracted from C with sparsity pattern defined by $\mathcal{P}$,

$$\tilde{c}_{ij} = \begin{cases} c_{ij} & \text{if } (i,j) \in \mathcal{P}, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Apply an Incomplete LU (*ILU*) factorization to a sparse matrix problem $Ax = b$. After k steps of the factorization, we have the following (incomplete LDU) decomposition[2],

$$A \longrightarrow \begin{bmatrix} L_k & 0 \\ P_k & I_{n-k} \end{bmatrix} \begin{bmatrix} D_k & 0 \\ 0 & A_k \end{bmatrix} \begin{bmatrix} U_k & Q_k^t \\ 0 & I_{n-k} \end{bmatrix}, \tag{4}$$

where L_k (U_k) is $k \times k$ lower (upper) unit triangular, D_k is $k \times k$ diagonal matrix, P_k and Q_k are $(n-k) \times k$, I_{n-k} is the $(n-k) \times (n-k)$ identity, and A_k is the $(n-k) \times (n-k)$ submatrix remaining to be factored. Let $A = A_0$ and $G_k = (\mathcal{V}_k, \mathcal{E}_k)$, $k = 0, 1, \cdots, n-1$ be the graphs [32] of A_k, i.e.

$$\mathcal{V}_k = \{v_{k+1}, \ldots, v_n\}\,, \quad \mathcal{E}_k = \left\{(v_i, v_j) \mid a_{ij}^{(k)} \neq 0, \quad i,j > k, \right\}. \tag{5}$$

Some new fill entries in A_k may be created in the factorization process, yet many fills are also discarded. A common criterion for discarding a new fill is by its fill level. The notion of "fill-level" can be defined through reachable sets and the shortest path length in the graph G_0 [17]. Let $\mathcal{S}_k : \{v_1, \ldots, v_k\}$ be the set of the eliminated nodes at step k, $\mathcal{S}_k \subset \mathcal{V}_0$, and choose nodes $u, v \notin \mathcal{S}_k$. Node u is said to be reachable from vertex v through $\mathcal{S}_k$ if there exists a path $(v, u_{i_1}, \ldots, u_{i_m}, u)$ in graph G_0, such that each $u_{i_j} \in \mathcal{S}_k$,

[1] To simplify notation, a case of symmetric nonzero structure is presented here.
[2] we assume the final elimination sequence is $v_1, \ldots, v_n$.

$1 \leq j \leq m$. Note that m can be zero, so that any pair of adjacent nodes $u, v \notin \mathcal{S}_k$ is reachable through $\mathcal{S}_k$. The reachable set of v through $\mathcal{S}_k$ is denoted by

$$Reach(v, \mathcal{S}_k) := \{u \mid \ u \text{ is reachable from } v \text{ through } \mathcal{S}_k \ \}. \tag{6}$$

Let v_i, $v_j \notin \mathcal{S}_k$ and $v_j \in Reach(v_i, \mathcal{S}_k)$ with the shortest path $(v_i, u_{i_1}, \ldots, u_{i_m}, v_j)$. Then the shortest path length from v_i to v_j through $\mathcal{S}_k$ is defined m. For convenience, we define the path length between v_i and v_j through $\mathcal{S}_k$ to be ∞, if $v_j \notin Reach(v_i, \mathcal{S}_k)$. Then a fill level $\text{level}_{ij}^{(k)}$ for ordered node pair (v_i, v_j) in the incomplete eliminated graph G_k can be defined as [3]:

$$\text{level}_{ij}^{(k)} = \begin{cases} m, & \text{if } a_{ij} \text{ is an accepted fill,} \\ \infty, & \text{otherwise} \end{cases} . \tag{7}$$

where m is the shortest path length from v_i to v_j through $\mathcal{S}_k$ and $i,\ j > k$. It is clear that

$$\text{level}_{ij}^{(k)} = \begin{cases} 0, & \text{if } a_{ij} \neq 0, \\ \infty, & \text{otherwise} \end{cases} \tag{8}$$

since $\mathcal{S}_0 = \emptyset$.

As the elimination proceeds, there may be a shorter path between v_i, v_j through the new set of eliminated nodes. Hence the fill-levels can be updated :

$$\text{level}_{ij}^{(k)} = \min\left(Level_{ik}^{k-1} + Level_{kj}^{k-1} + 1,\ Level_{ij}^{k-1}.\right). \tag{9}$$

In an $ILU(\ell)$ factorization, only fill entries with fill-level less than or equal to ℓ are kept. More precisely, consider again the k-th step in an $ILU(\ell)$ factorization of A,

$$A_{k-1} = \begin{bmatrix} a_{kk}^{(k-1)} & \beta^t \\ \gamma & B_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \gamma / a_{kk}^{(k-1)} & I_{n-k} \end{bmatrix} \begin{bmatrix} a_{kk}^{(k-1)} & 0 \\ 0 & C_k \end{bmatrix} \begin{bmatrix} 1 & \beta^t / a_{kk}^{(k-1)} \\ 0 & I_{n-k} \end{bmatrix} \tag{10}$$

where

$$C_k = B_{k-1} - \gamma \beta^t / a_{kk}^{(k-1)}. \tag{11}$$

Fill-level $\text{level}_{ij}^{(k)}$ for the node pair (v_i, v_j), $i,\ j > k$ can be deduced from (9). Let $\mathcal{D}_k$ be the sparsity pattern that contains all nonzero elements in C_k having fill-level higher than ℓ, the given level for the ILU decomposition,

$$\mathcal{D}_k = \left\{(i, j) \mid \text{level}_{ij}^{(k)} > \ell, \quad i, j > k\right\}. \tag{12}$$

[3] If the order of the unknowns is predetermined before the incomplete factorization, a fill level which is independent to k can be introduced. However, in this particular study, the pivoting order is determined dynamically during the computations. Therefore, it is not advantageous to define such a level.

The nonzero elements corresponding to $\mathcal{D}_k$ will be discarded after this elimination step and

$$NZ\,[C_k] - \mathcal{D}_k = \Big\{(i,j) \mid \text{level}_{ij}^{(k)} \le \ell, \quad \text{where} i,j > k\Big\} \tag{13}$$

is the accepted sparsity pattern in the $ILU(\ell)$ decomposition. The $ILU(\ell)$ decomposition process at the next step will operate on a truncated matrix of

$$A_k = B_{k-1} - \gamma\beta^t/a_{kk}^{(k-1)} - F_k = C_k - F_k, \quad F_k = C_k[\mathcal{D}_k]\ , \tag{14}$$

where $F_k = C_k[\mathcal{D}_k]$ is the discarded fill matrix at step k.

The following theorem characterizes the elimination graph G_k produced by an $ILU(\ell)$ factorization.

Theorem 2.1.

$$\mathcal{E}_k = \Big\{(v_i, v_j) \mid v_j \in \mathit{Reach}(v_i, \{v_1, \dots, v_k\}) \ \mathit{and}\ \mathit{level}_{ij}^{(k)} \le \ell\Big\}$$

where $v_1, v_2, \dots, v_k$ *are previously eliminated nodes.*

Proof. If $\ell = 0$, no new fill is introduced in the $ILU(0)$ decomposition, hence all entries have fill-level zero. Therefore G_k is the subgraph of G_0 induced by $\{v_{k+1}, \dots, v_n\}$. The theorem holds trivially in this case.

For $\ell \ge 1$, an induction on k can be used to prove this theorem. □

In Section 1, we introduced a problem which demonstrates very different convergence behaviors if different anisotropy is used. Here we introduce two decay estimates for the entries of the ILU decompostion presented in [9]. These estimates explain the causes of the different convergence behaviors and demonstrate the need for developing some ordering (or pivoting) strategies which will be able to encourage a fast decay in the ILU decompostion.

First, a lower bound for the entries $a_{ij}^{(k)}$ of A_k in $ILU(\ell)$ decomposition of an M-matrix A exists.

Theorem 2.2. *Let A be an M-matrix, $a_{ij}^{(k)}$ are the fill entries in A_k and $\ell \ge$ $\mathit{level}_{ij}^{(k)} \ge 1$, and if*

$$(v_i, v_{i_1}, \dots, v_{i_m}, v_j), \quad v_{i_1}, \dots, v_{i_m} \in \mathcal{S}_k : \{v_1, \dots, v_k\} \tag{15}$$

is a path from v_i to v_j through $\mathcal{S}_k$, then

$$\Big|\ a_{ij}^{(k)}\ \Big| \ge \frac{|a_{ii_1} a_{i_1 i_2} \cdots a_{i_m j}|}{d_{i_1} d_{i_2} \dots d_{i_m}}, \quad \mathit{where}\ d_i = a_{ii}.$$

If $\ell = \infty$, then this theorem also provides a lower bound for LU decomposition.

For the problem (1), the resulting matrix is an M-matrix. If $K_x \gg K_y$, all new higher level fill entries of the factorization will have a path for which all edges lie in the x-direction but at most one lies in the y direction. From Theorem 2.2, and since the entries in $L = \{l_{ij}\}$ of (4) are given by $l_{ij} = a_{ij}^{(j-1)}/a_{jj}^{(j-1)}$, entries l_{ij} will have a very slow decay rate with the fill level $\mathit{Level}_{ij}^{i-1}$. Thus, nearly all fill entries in the decomposition

TABLE 1
Comparison of two problems

Problem	Nonzeros in L	Iterations	Fact. time	Sol. time
$K_x = 100$, $K_y = 1$	10330	17	0.15	0.20
$K_x = 1$, $K_y = 100$	2705	13	0.04	0.11

will be kept, resulting in an unacceptably dense factorization, when a drop tolerance technique is used.

In contrast, if $K_y \gg K_x$ in problem (1), we have the following estimate of the entries l_{ij} in the LU decomposition for that problem,

THEOREM 2.3.

$$(16) \qquad |l_{i,i-n+j}| = O\left(\frac{1}{(K_y/K_x)^j}\right), \qquad j = 0, 1, \cdots, n-1-(i-1) \bmod n; \quad \text{and} \quad i > n,$$

$$(17) \qquad |l_{i,i-j}| = O\left(\frac{1}{(K_y/K_x)^j}\right), \qquad j = 0, 1, \cdots, (i-1) \bmod n,$$

where l_{ij} are the entries of the LDL^T factorization of the matrix A in problem (1).

Notice that $j = Level^{i-1}_{i,i-n+j}$, in (16) and $j+1 = Level^{i-1}_{i,i-j}$ in (17), respectively. It is obvious that l_{ij} decay rapidly with the fill level. A much sparser incomplete factorization will be resulted, if the same drop tolerance is used. Our numerical results can provide a more clear picture of these estimate. We use a 30×30 regular grid in the computations. A new fill entry is dropped if

$$|a^{(k)}_{ij}| \leq 0.001 \min(|a_{ii}|, |a_{jj}|) \ .$$

Table 1 summarizes the effects of different anisotropy on the quality of the preconditioneres, where convergence tolerance reduces the initial l_2 residual by 10^{-6}. The second column shows the number of off-diagonal non-zeros in L. An interesting observation from this table is: many more extra fills in the first problem provides us a worse preconditioner!

2.2. *MDF*(ℓ) Algorithm .

The *MDF*(ℓ) ordering is motivated by the observation that a small discarded fill matrix F_k in (14) would produce a more "authentic" factorization for matrix A. We define the *discard value* for eliminating the k^{th} node as the Frobenius norm of the discarded fill matrix F_k,

$$(18) \qquad discard(v_k) = \|F_k\|_F = \|C_k[\mathcal{D}_k]\|_F = \left(\sum_{i \geq 1}\sum_{j \geq 1} {F_{ij}}^2\right)^{1/2}$$

Note the discard value for another node v_i can be similarly obtained by first performing a symmetric exchange of node v_k and v_i.

The basic idea of the minimum discarded fill algorithm (*MDF*(ℓ)) is to choose the next pivot node that has the minimum discard value. It is an attempt to obtain locally

```
Initialization:
            $A_0 := A$
            for each $a_{ij} \neq 0$
                $Level(a_{ij}) := 0$
            end
            for each node $v_i$
                Compute the discarded value $discard(v_i)$
            end
for k $= 1 \ldots n-1$
    Choose as the next pivot node $v_j$ in matrix $A_{k-1}$ which has
    minimum $discard(v_j)$ (break ties arbitrarily).
    Update the decomposition,

    $C_k := B_k - \gamma_k \beta_k^t / d_k$
    $A_k := C_k - F_k,$,   where $F_k = C_k[\mathcal{D}_k]$,  $P_k A_{k-1} P_k^t = \left[\begin{array}{cc} d_k & \beta_k^t \\ \gamma_k & B_k \end{array}\right]$.

    $P_k$ is permutation matrix to exchange $v_j$ to first position.
    Update the discard values of $v_j$'s neighbors.
    Update the fill-level of elements in $A_k$ by (9).
end
```

FIG. 1. *Description of MDF(ℓ) algorithm.*

one of the best approximations of the complete decomposition. Since the updating matrix, $\gamma\beta^t/a_{kk}^{(k-1)}$, affects only a few rows of B_k, only the discard values of the neighbors of v_k need be recomputed. A description of the *MDF*(ℓ) algorithm is given in Figure 1. Several tie breaking strategies are discussed in [8].

During the *MDF*(ℓ) ordering process, the nonzero pattern of L and the number of fill entries in an *ILU*(ℓ) decomposition depend on the ordering and are rather dynamic. Consequently, the complexity analysis of *MDF*(ℓ) is still an open problem. However, a rough estimate can help indicates the direction for possible improvements in efficiency.

Assume matrix A is $N \times N$, the average number of non-zero elements in each row is c, and the average number of non-zero elements in each row in $L + L^t$ is d. It is clear that $c \leq d$ and d depends on fill-level ℓ. The complexity of the initialization step is about $O(c^2 N)$. One elimination step costs $O(d^2)$ while updating the discard values of neighbor nodes costs $O(d^3)$. Therefore by a rough estimate, the complexity for *MDF*(ℓ) is dominated by the cost of updating the discard values, which is about $O(d^3 N)$. If we have a rather large node degree c in matrix A, d will grow quickly with level ℓ. For example, $d \sim O(\ell^{k-1})$ if matrix A is derived from a k-dimensional Laplace equation on a regular grid using central difference.

2.3. Threshold *MDF*(ℓ) Algorithm. As we just mentioned, the average number of nonzeros per row in L grows rather quickly, when ℓ increases. Fortunately, most of the high-level fill entries are "small". For example, the number of nonzeros in L in our test problem ANISO (*MDF*(4) ordering) is 7184 and the number of iterations needed for convergence is 21 (see Table 2 and 3). If we drop all fill entries which are smaller than 10^{-3}, the new ordering which we will discuss in this section has only 4872 nonzeros in L (many of them have an even fill level higher than 4) while the number of iterations is 20. Extra entries introduced in *MDF*(4) ordering did not bring any improvement in the quality of the preconditioner. This observation suggests that there are some higher level fill entries which do make an important contribution to the quality of the preconditioner and many of the higher level fill entries can be safely ignored. Based on these observations, a new heuristic threshold $MDF(\ell)$ is proposed. The discard sparsity pattern $\mathcal{D}_k$ in L from the $MDF(\ell)$ algorithm is replaced by

$$\mathcal{D}_k(\varepsilon) = \left\{(i,j) \mid Level(c_{ij}^{(k)}) > \ell \text{ or } |c_{ij}^{(k)}| < \varepsilon \ \min(R_i, R_j)\right\}, \tag{19}$$

where

$$R_i = \max_{m=1\ldots N}(|a_{im}|) = \|a_{i*}\|_\infty$$

and ε is a given relative threshold. Note that if A is diagonally dominant and scaled to have unit diagonal, then the absolute threshold criterion $|c_{ij}^{(k)}| \le \varepsilon$ is equivalent to using in (19) [4]

$$|c_{ij}^{(k)}| \le \varepsilon \ (|a_{ii}a_{jj}|)^{1/2} \ .$$

The description of this algorithm will be exactly the same as $MDF(\ell)$ except the discard matrix F_k is

$$F_k = C_k[\mathcal{D}_k(\varepsilon)] \quad \text{and } A_k = C_k - F_k = C_k - C_k[\mathcal{D}_k(\varepsilon)] \ . \tag{20}$$

A detailed discussion of our numerical results is given in the next section. One observation is worth noting here. Comparing the results using different combinations of ε and fill-level ℓ in threshold $MDF(\ell)$, we observe that the best choice for the combination is

$$\ell = \infty, \quad 10^{-3} \le \varepsilon \le 10^{-4} \ ,$$

in all of our test problems. As an added benefit, if we choose $\ell = \infty$, there is no overhead for recording the fill-level, which can bring us an extra 5–15% of saving in ordering time.

2.4. Minimum update matrix algorithm. The most costly part of an *MDF* type ordering is the traversal of the adjacency lists three levels deep while updating discard values. If the average node degree is large, the high cost of updating discard values makes the *MDF* ordering impractical. We propose a simpler scheme, the *minimum update matrix ordering* (*MUM* ordering), which is motivated by the Markowitz algorithm [27] in sparse Gaussian elimination. The computation of the discard value

$$\|C_k[\mathcal{D}_k]\|_F, \quad C_k = B_{k-1} - \gamma\gamma^t/a_{kk}^{(k)}$$

[4] Munksgaard [30] has used the drop criterion $|a_{ij}^{(k)}| \le \varepsilon \ (|a_{ii}^{(k-1)}a_{jj}^{(k-1)}|)^{1/2}$.

requires the determination of all new fill entries in C_k by nested traversal of adjacency lists. If we replace the discard value by the norm of the updating matrix $\|\gamma\gamma^t/a_{kk}^{(k-1)}\|_F$, the computation of the norm requires only the information about the current adjacency list and will be much cheaper to compute.

Note that $MDF(\ell)$ and threshold *MDF* orderings have discard value computations intimately coupled to the incomplete factorization strategy. The discard values are the norm of actual discarded fill matrices. The *MUM* ordering uses a simpler measure for the discard value and has less coupling to the factorization strategy.

The *MUM* ordering did not perform very well for anisotropic problems with small molecule. Five-point molecules are used in three out of the eight test problems. If the problem is very anisotropic, the decay rate of the *LU* decomposition is very sensitive to the direction of the ordering. This algorithm failed to identify a fast decay direction for two of our anisotropic test problems. When we carefully examined the process of this ordering, both orientations have updating matrices with the same norm. Without considering the locations of the actual fill entries, a fast decay direction cannot be identified. Numerical results confirm this analysis.

However, the *MUM* heuristic works quite well if the average node degree [5] is high (a large molecule), and the decay rate in the *LU* decomposition is less sensitive to the anisotropy in the problem. Two additional large unsymmetric sparse matrix problems are derived from incompressible Navier-Stokes equations in order to test the *MUM* ordering. Our numerical results indicates that an *MUM* ordering not only took less time, but surprisingly, also produced higher quality *ILU* preconditioners.

2.5. Incomplete *MDF* ordering. *MUM* ordering adopts the strategy of reducing the depth of list traversal by ignoring sparsity in the computation of the discard value. Another approach is to reduce the width of list traversal by pruning. This is an extension of the threshold drop tolerance heuristic to further reduce the amount of work used to compute and update discard values.

If node v_k is eliminated, let

$$\beta = \left[a_{k,k+1}^{(k-1)}, \ldots, a_{kn}^{(k-1)}\right], \quad \gamma^t = \left[a_{k+1,k}^{(k-1)}, \ldots, a_{nk}^{(k-1)}\right]$$

be the vectors in the rank-one update in (11). By the threshold *MDF* criterion, a new fill entry $a_{ij}^{(k)}$ is ignored if

$$\left|a_{ik}^{(k-1)} a_{kj}^{(k-1)} / a_{kk}^{(k-1)}\right| < \varepsilon \min(\|a_{i*}\|_\infty, \|a_{j*}\|_\infty).$$

Let $\mathcal{P}$ be the sparsity pattern of "significant" entries of β, with τ a given threshold value,

$$\mathcal{P} = \left\{ j \;\middle|\; |a_{kj}^{(k-1)}| \geq \tau \min(\|a_{i*}\|_\infty, \|a_{j*}\|_\infty) \right\}, \tag{21}$$

and let $\tilde{\beta} = \beta(\mathcal{P})$. If $|a_{kj}^{(k-1)}|$ is "small", we may consider the contribution of this element to the updating matrix, $\gamma\beta^t/a_{kk}^{(k-1)}$, to be insignificant. So instead of using $\gamma\beta^t/a_{kk}^{(k-1)}$

[5] *maybe this needs more explanation*

in computing the discard fill value, we use one of the following "incomplete" matrices,

$$\tilde{\gamma}\beta^t/a_{kk}^{(k-1)}, \quad \gamma\tilde{\beta}^t/a_{kk}^{(k-1)}, \quad \tilde{\gamma}\tilde{\beta}^t/a_{kk}^{(k-1)}, \tag{22}$$

where we can similarly define $\tilde{\gamma}$. A simple calculation shows that the choice $\tilde{\gamma}\tilde{\beta}^t/a_{kk}^{(k-1)}$ cannot identify the preferred orientation in our model anisotropic test problem (1). Hence, we have used the incomplete updating matrix, $\gamma\tilde{\beta}^t/a_{kk}^{(k-1)}$, in our experiments.

After node v_k is eliminated, instead of updating the discard values of all neighbors of v_k, we update only $discard(v_j)$ that satisfies

$$\left|a_{kj}^{(k-1)}\right| \geq \tau \min(\|a_{k*}\|_\infty, \|a_{j*}\|_\infty), \tag{23}$$

by the similar reasoning where, if $|a_{kj}^{(k-1)}|$ is small, the contribution from this entry to $discard(v_j)$ is insignificant.

3. Test Problems.

Numerical testing for all four ordering algorithms was carried out on a variety of problems.

3.1. Problem 1 (LAPD5).

The first problem is Laplace's equation on the unit square with Dirichlet boundary conditions, as used in [14]. The usual five-point finite difference discretization was used on a regular 30×30 grid.

3.2. Problem 2 (STONE).

This is Stone's third problem [35]. The equation

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial P}{\partial y}\right) = -q \tag{24}$$

was discretized on the unit square using a finite difference technique. The region is shown in Figure 2. The values of K_x, K_y and q are:

$$(K_x, K_y) = \begin{cases} (1,1) & \text{if } (x,y) \in A, \quad \text{elsewhere} \\ (1,100) & \text{if } (x,y) \in B, \quad 14 \leq x \leq 30,\ 0 \leq y \leq 16 \\ (100,1) & \text{if } (x,y) \in C, \quad 5 \leq x \leq 12,\ 5 \leq y \leq 12 \\ (0,0) & \text{if } (x,y) \in D, \quad 12 \leq x \leq 19,\ 21 \leq y \leq 28 \end{cases}, \tag{25}$$

$$\begin{aligned} &q_1(3,3) = 1.0, \quad q_2(3,27) = 0.5, \quad q_3(23,4) = 0.6, \\ &q_4(14,15) = -1.83, \quad q_5(27,27) = -0.27\ . \end{aligned} \tag{26}$$

A 33×33 grid was used, and an harmonic average was used for discontinuities in K_x and K_y [2].

3.3. Problem 3 (ANISO).

This problem has the same equation as in (24) except the value distributions of K_x and K_y are different:

$$(K_x, K_y) = \begin{cases} (100,1) & \text{if } 0 \leq x \leq 1/2, \quad 0 \leq y \leq 1/2 \\ (1,100) & \text{if } 1/2 \leq x \leq 1, \quad 0 \leq y \leq 1/2 \\ (100,1) & \text{if } 1/2 \leq x \leq 1, \quad 1/2 \leq y \leq 1 \\ (1,100) & \text{if } 0 \leq x \leq 1/2, \quad 1/2 \leq y \leq 1 \end{cases},$$

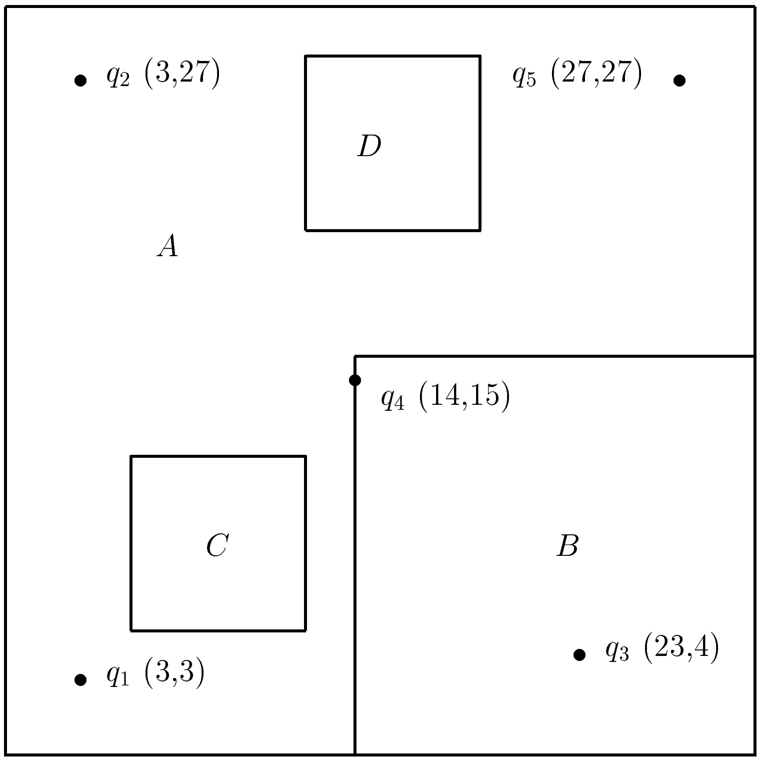


FIG. 2. *Stone's third problem.*

A 30×30 grid was used.

Test problems (4–6) are derived from two and three dimensional pressure equations arising in groundwater contamination simulations [16, 21]. The pressure equation is essentially equation (24). Since the actual values of K_x, K_y, q and the boundary conditions are quite complicated, only a brief description of these problems will be given.

3.4. Problem 4 (REFINE2D). A finite element method using linear triangular basis functions was used to discretize this problem. In this example, K_x and K_y are constant. The triangulation was such that the resulting equation was an M-matrix [16]. The grid was constructed by first defining a very coarse triangulation, and then repeatedly defining finer grids by subdividing a triangle into four smaller triangles, with new nodes determined by the nodes of the original triangle and the midpoints of the original triangle edges.

3.5. Problem 5 (FE2D). A finite element method using linear triangular basis functions was also used on this problem. However, in this example, K_x and K_y (equation (24)) varied by four orders of magnitude. The grid was defined by constructing a distorted quadrilateral grid, and then triangulating in the obvious manner. A Delaunay-type edge swap [16] was used to produce an M-matrix.

3.6. Problem 6 (FE3D). This problem is a three-dimensional version of equation (24). A finite element discretization was used, with linear basis functions defined on tetrahedra. The absolute permeabilities (K_x,K_y,K_z) varied by eight orders of magnitude (this model was derived from actual field data). The nodes were defined on a $25 \times 13 \times 10$ grid (3250 nodes) of distorted hexahedra, which were then divided into tetrahedra. The resulting matrix was *not* an M-matrix, and the average node connectivity was fifteen. In general, it is not possible for a given node placement to obtain an M-matrix in three dimensions if linear tetrahedral elements are used [26].

3.7. Problem 7 (NS2D). This problem is derived from a finite volume discretization of the incompressible Navier-Stokes equations [33]. A 40×40 grid was used to model the backward step problem, with a Reynold's number $R_e = 1000$. The matrix was generated from the Jacobian produced for the second Newton iteration. This problem was unsymmetric, so CGSTAB acceleration [11, 10] was used.

3.8. Problem 8 (NS3D). This problem was derived from a small three dimensional finite element discretization of the incompressible Navier-Stokes equations [34]. Use of a finite element method in three dimensions resulted in a very large computational molecule. The test matrix was generated from a Jacobian produced near the start of a pseudo-time solution of the steady-state problem. CGSTAB acceleration was used here as well.

4. Numerical Results. The computations to solve the test problems (1-6) were carried out on an IBM RISC/6000 using double precision. Because the matrices derived from the last 6 test problems were ill-conditioned, a stricter convergence criterion

$$\|\mathbf{r}^k\|_2 \leq 10^{-12}\|\mathbf{r}^0\|_2 \quad ,$$

was used, where $\mathbf{r}^k$ was the residual vector after the k^{th} iteration in the conjugate gradient acceleration. The right hand side vectors for the Navier-Stokes problems were the residual of the non-linear equations. The initial vector $\mathbf{x}^0$ was chosen to be the zero vector. Radom initial guesses are also tested, results are qualitatively similar.

Table 2 is a timing summary of $MDF(\ell)$ for the test problems. For a matrix that has a high connectivity, the cost of ordering grows rather quickly with ℓ (see column FE3D) and soon becomes impractical. We include the number of off-diagonal nonzeros of L to show this phenomenon.

Table 3 lists the total solution time (i.e., total time for symbolic incomplete factorization, numeric incomplete factorization and conjugate gradient acceleration) and the corresponding number of iterations. For contrast, we also include the timing from $ILU(0)$ and $ILU(1)$ with the original ordering. In particular, the improvement brought by $MDF(\ell)$ ordering was rather significant for problems where the initial "preferred" order was not easily identifiable (see column REFINE2D or FE2D). For most of our test problems, the solution time stops decreasing for $MDF(\ell)$, $\ell > 3$, though the number of iterations continues to be reduced.

Numerical testing of threshold MDF ordering is done for different levels ℓ and tolerances ε. Table 4 is a comparison between $MDF(\ell)$ and threshold MDF ordering.

The solution time for $ILU(1)$ with the original ordering is also included for comparison. As we can see, threshold *MDF* ordering brings further improvement in performance for all the test problems.

The detailed data for ordering time, total solution time, number of iterations and number of off-diagonal nonzeros in L for various values of ℓ and ε are reported in [6].

From these detailed data, we observe:

- Most of the high level fills are small and can be safely ignored.
- Some high-level fills are important for the quality of the ILU preconditioner.
- Holding the number of nonzeros in L roughly constant, the choice of a higher fill level and larger tolerance is more cost effective than that of a smaller tolerance and lower fill level.
- The optimal choice of ℓ and ε for all our test problems is:

$$\ell = \infty, \quad \varepsilon = 10^{-3} \text{ or } 10^{-4} .$$

It is also instructive to visualize the ordering produced by the different techniques. Using a visualization tool *MATVIEW* developed at Waterloo [36], we present ordering pictures for the first three test problem (see Figures 3, 4 and 5). For the Laplace problem, both $MDF(0)$ and $MDF(1)$ orderings are given. Here the nodes with lightest shading were ordered first and the darkest last. We can see that the $MDF(0)$ ordering for the Laplace problem is very similar to the spiral ordering, which was one of the best orderings for $ILU(0)$ [14]. $MDF(1)$ ordering gave a generalized red-black ordering, which is again known to be effective for $ILU(1)$ preconditioning. The actual timings supported the same conclusions.

For Stone's problem, $MDF(0)$ ordering did identify the physical structure (regions B, C, and D), but failed to detect the fast decay orientations in different physical regions. Numerical results confirm this observation. The reason for the failure is clear. No fill is allowed in this case. Identifying a fast decay direction requires at least level 1 fill. When a higher level *MDF* ordering was considered, both physical structure and fast decay orientation were successfully identified in the ordering.

We designed the anisotropic test problem to emphasize the importance of the orientation of the ordering to the convergence. Again, $MDF(0)$ recognized the physical regions but failed to determine the "preferred" orientation, while $MDF(1)$ effectively identified both.

The numerical results from Minimum Updating Matrix (*MUM*) ordering are mixed. The detailed results are listed in Table 5. For convenience, the timing results with Threshold *MDF* ordering (T-*MDF*)and the original $ILU(1)$ are also included. The parameter pair (ℓ, ε) in the table represents the fill level of the preconditioner and the tolerance used to control fill. The *MUM* ordering did demonstrate improvements over $ILU(1)$ in total solution timing. However, only two out of six test problems obtained any significant improvement. For several anisotropic problems, the improvement was rather limited if any. One particular interesting result is that *MUM* ordering needs only one tenth of the ordering time in problem FE3D and pays only a small penalty in solution time. This leads us to test a few more problems with large molecules.

Two large unsymmetric sparse matrices derived from the incompressible Navier-Stokes equations were chosen for comparing the effects of the different orderings. Both matrices had large molecules. The matrix (NS3D), derived from the incompressible Navier-Stokes equation in a three dimensional finite element grid, had an average of 100 nonzeros in each row. We used CGSTAB [10] as the acceleration scheme, with the same convergence criteria as for symmetric problems.

Tables 6 and 7 list the ordering time, nonzeros in the incomplete LU factorization, time for factorization, number of iterations and solution time in three different orderings. Different levels and thresholds were applied for all three orderings. Here ORG stands for the ordering provided by the original discretization code. We observe that the original orderings were poor, and sometimes did not converge. The MDF ordering provided better results. However, the ordering times were large. The MUM ordering gave the best results for both ordering time and solution time in most cases. In particular, for the NS3D problem (see Table 7) convergence could not be obtained with the original ordering for all types of ILU preconditioning. On the other hand, the MUM ordering produced a convergent method for all types of ILU preconditioning. The MDF ordering was beneficial but the ordering cost was very large. For this example, MUM ordering was the only practical way to obtain an iterative solution.

Several interesting but puzzling observations about the unsymmetric test problems are:

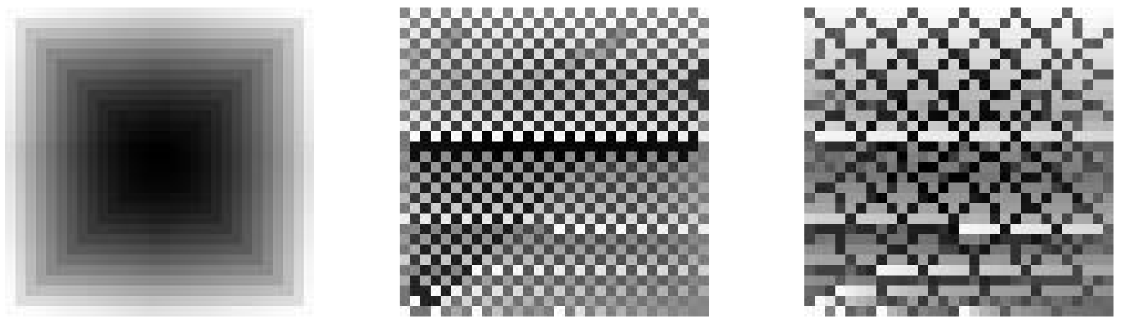
- The threshold ordering for both MDF and MUM converges faster than the no-threshold ordering in most of the cases.
- The lower level MUM ordering converges even faster than MDF ordering with only one exception.

Our numerical experiments show that the Incomplete MDF ($IMDF$) strategy is effective in reducing the ordering time, with only a minor penalty in the quality of the preconditioner. The detailed data is reported in [6]. Since the Threshold MDF only accepts nonzero fill based on the threshold ε, a value of τ less than ε has little effect on the ordering. The optimal choice of τ may be problem dependent, but $\tau \approx \sqrt{\varepsilon}$ seems to work well on most problems. All problems show a substantial reduction in ordering time.

5. Conclusion.

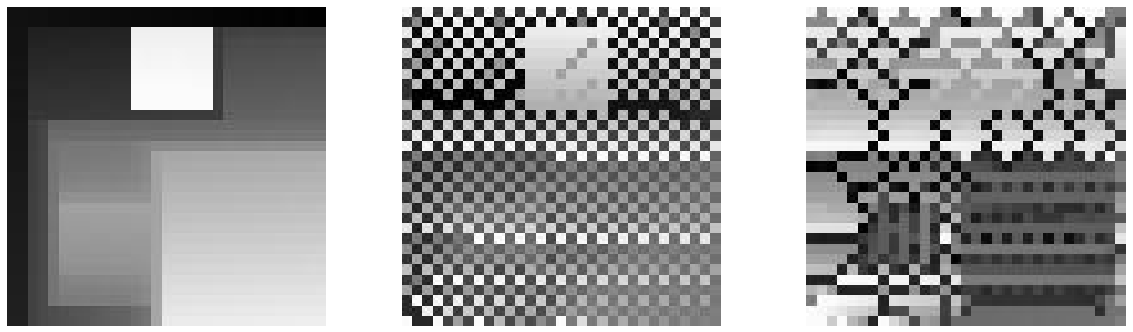
In [8], it has been demonstrated that $MDF(\ell)$ ordering is effective for matrix problems arising from complicated partial differential equation discretizations, and in particular, for problems with anisotropic and heterogeneous physical parameters. However, the cost of $MDF(\ell)$ ordering is high if the average degree in the original matrix is large. When solving partial differential equations, this situation occurs if the discretization uses a large molecule or stencil.

In order to reduce both ordering and iteration cost, the threshold MDF ordering has been developed. This combined ordering/factorization method drops any fill entries that are less than a prescribed tolerance. It is important to combine an ordering technique with drop tolerance ILU preconditioning. Without a good ordering strategy, an initial poor ordering results in slow decay of the fill entries, and produces an unacceptably dense ILU factorization.



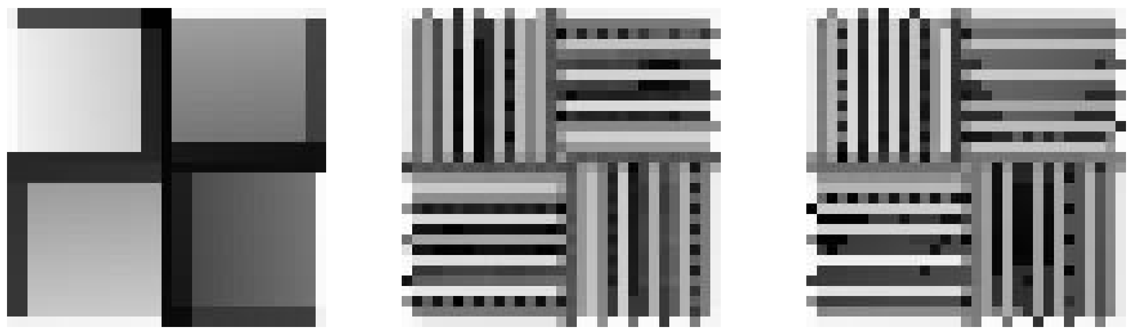
$MDF(0)$ ordering $MDF(1)$ ordering $MDF(\infty, 10^{-3})$ ordering

FIG. 3. *Ordering pictures for Laplace problem (LAPD5). Lightest, first; darkest, last*



$MDF(0)$ ordering $MDF(1)$ ordering $MDF(\infty, 10^{-3})$ ordering

FIG. 4. *Ordering pictures for Stone's problem (STONES). Lightest, first; darkest, last.*



$MDF(0)$ ordering $MDF(1)$ ordering $MDF(\infty, 10^{-3})$ ordering

FIG. 5. *Ordering pictures for Anisotropic problem (ANISO). Lightest, first; darkest, last.*

Numerical tests show that a high-level threshold $MDF(\ell)$ ordering combined with a drop tolerance produces excellent results for partial differential equation problems having a relatively small molecule. This is because most of the high level fill is small and can be ignored, but there are a few high-level fill entries that improve the quality of the preconditioner. The testing for anisotropic problems confirms this conclusion.

Discretization of systems of partial differential equations, for example the Navier-Stokes equations, typically gives rise to large molecules. For these problems, even threshold MDF ordering is too expensive. Consequently, we have developed an approximate MDF ordering based on the concept of the minimum updating matrix (MUM). The MUM ordering is much less expensive to compute. However, MUM ordering can fail to select a good ordering for anisotropic problems having small molecules. Of course, this is precisely the situation where threshold MDF works well. On the other hand, partial differential equation discretizations with large molecules are less affected by anisotropy. For large molecules, numerical tests indicate that MUM ordering is very effective. In particular, we have applied MUM ordering to Jacobians from Navier-Stokes problems. In some cases, the original ordering did not converge at all, while MUM ordering resulted in fast convergence.

The incomplete MDF variant is another attempt to reduce the ordering cost for matrices with large node degrees by ignoring small fill contributions in the updating of discard values. Our tests indicate this is effective in reducing the ordering time with a minor penalty in the quality of the preconditioner.

To summarize, it appears from our numerical tests that threshold MDF is an effective ordering for partial differential equation discretizations with small molecules, while threshold MUM ordering is a good choice for discretizations having large molecules. We have also demonstrated by a simple counter example, that an effective ordering strategy must consider the value of the matrix entries, not just the graph of a matrix. This is particularly important for incomplete factorization with drop tolerance.

TABLE 2
MDF(ℓ) ordering time and number of nonzeros in L.

Ordering Time (number of nonzeros)						
Ordering	LAPD5 (neq=900)	STONE (neq=961)	ANISO (neq=900)	REFINE2D (neq=1161)	FE2D (neq=1521)	FE3D (neq=3250)
$MDF(0)$	0.21(1740)	0.23(1860)	0.23(1860)	0.30(2480)	0.52(4373)	4.56(19725)
$MDF(1)$	0.47(3386)	0.49(3596)	0.51(3633)	0.67(4672)	1.35(8117)	43.65(49080)
$MDF(2)$	0.77(4414)	0.77(4664)	0.80(4651)	1.24(6368)	3.10(11480)	331.69(90609)
$MDF(3)$	1.44(5704)	1.44(6094)	1.54(6093)	2.18(7943)	6.16(14545)	1670.84(140979)
$MDF(4)$	2.12(6770)	2.13(7349)	2.11(7184)	3.58(9502)	10.74(17303)	N/A

TABLE 3
Total solution time and number of iterations.

Solution Time (number of iterations)						
	LAPD5	STONE	ANISO	REFINE2D	FE2D	FE3D
$ILU(0)$	0.38(43)	0.58(65)	0.62(70)	2.44(239)	3.37(230)	2.95(63)
$ILU(1)$	0.29(28)	0.39(38)	0.51(52)	1.52(132)	1.15(70)	2.51(38)
$MDF(0)$	0.37(42)	0.58(64)	0.63(69)	0.64(61)	0.76(39)	1.82(32)
$MDF(1)$	0.23(22)	0.30(27)	0.37(35)	0.56(40)	0.61(30)	2.50(26)
$MDF(2)$	0.24(19)	0.30(23)	0.36(27)	0.44(26)	0.56(22)	4.02(18)
$MDF(3)$	0.21(15)	0.27(17)	0.31(23)	0.38(22)	0.56(18)	7.88(15)
$MDF(4)$	0.24(13)	0.25(15)	0.30(21)	0.40(19)	0.58(15)	N/A

TABLE 4
Comparison of timings from ILU(1), MDF(ℓ) and threshold MDF.

Solution Time (number of iterations)						
	LAPD5	STONE	ANISO	REFINE2D	FE2D	FE3D
$ILU(1)$	0.27(28)	0.35(38)	0.50(52)	1.53(132)	1.20(70)	2.48(38)
$MDF(1)$	0.21(22)	0.30(27)	0.37(35)	0.55(40)	0.61(30)	2.47(26)
$MDF(\infty, 10^{-3})$	0.20(8)	0.20(11)	0.27(20)	0.28(16)	0.44(17)	1.63(13)
$MDF(\infty, 10^{-4})$	0.22(5)	0.21(7)	0.20(13)	0.24(10)	0.42(10)	2.41(9)

TABLE 5
Summary of the test results for Minimum Updating Matrix (MUM) ordering.

		LAPD5			**STONE**		
Algorithm	(ℓ,ε)	Ordering timing	Nonzeros in L	Solution timing	Ordering timing	Nonzeros in L	Solution timing
MUM	$(0, 0.0)$	0.14	1740	0.32(42)	0.15	1860	0.53(64)
MUM	$(1, 0.0)$	0.18	2581	0.23(27)	0.19	2784	0.37(43)
MUM	$(\infty, 10^{-3})$	0.46	7481	0.11(11)	0.53	8045	0.23(21)
T-*MDF*	$(\infty, 10^{-3})$	3.03	7616	0.08(8)	2.06	6666	0.11(11)
ILU(1)	$(1, 0.0)$	0	2581	0.23(28)	0	2760	0.35(38)

		ANISO			**REFINE2D**		
Algorithm	(ℓ,ε)	Ordering timing	Nonzeros in L	Solution timing	Ordering timing	Nonzeros in L	Solution timing
MUM	$(0, 0.0)$	0.16	1860	0.58(69)	0.19	2480	0.63(62)
MUM	$(1, 0.0)$	0.19	2762	0.58(65)	0.23	3607	0.56(53)
MUM	$(\infty, 10^{-3})$	0.40	6334	0.35(35)	0.75	11094	0.38(27)
T-*MDF*	$(\infty, 10^{-3})$	0.81	4872	0.22(20)	1.11	6572	0.19(16)
ILU(1)	$(1, 0.0)$	0	2760	0.45(52)	0	4626	1.40(132)

		FE2D			**FE3D**		
Algorithm	(ℓ,ε)	Ordering timing	Nonzeros in L	Solution timing	Ordering timing	Nonzeros in L	Solution timing
MUM	$(0, 0.0)$	0.27	4373	1.03(73)	1.16	19725	2.17(53)
MUM	$(1, 0.0)$	0.36	5986	0.89(59)	2.38	35963	1.43(30)
MUM	$(\infty, 10^{-3})$	0.91	14226	0.58(30)	3.30	42222	0.89(17)
T-*MDF*	$(\infty, 10^{-3})$	1.80	9339	0.29(17)	30.85	44793	0.70(13)
ILU(1)	$(1, 0.0)$	0	7965	1.09(70)	0	35613	1.88(38)

TABLE 6
Results for NS2D.

NS2D (neq=2369)							
Level	Threshold	Ordering	Ordering timing	Nonzeros in L	Nonzeros in U	Fact. timing	Solution timing
		ORG	0.43	9125	9125	0.21	4.98(130)
0	0.0	*MDF*	1.41	9125	9125	0.19	9.14(226)
		MUM	0.94	9125	9125	0.22	2.95(69)
		ORG	0.52	11613	11613	0.20	5.28(130)
1	0.0	*MDF*	7.62	19244	19244	0.37	4.49(83)
		MUM	2.13	20281	20281	0.40	2.45(48)
		ORG	0.52	11557	11579	0.20	6.86(163)
1	1.0^{-3}	*MDF*	6.39	18758	17932	0.29	1.53(34)
		MUM	1.58	16369	18141	0.36	2.07(41)
		ORG	0.72	14709	14709	0.21	3.94(87)
2	0.0	*MDF*	20.08	27211	27211	0.50	3.30(59)
		MUM	7.70	43630	43630	0.90	2.48(35)
		ORG	0.75	13838	14584	0.24	diverge
2	1.0^{-3}	*MDF*	11.42	23330	23471	0.40	1.43(26)
		MUM	4.04	28717	31130	0.54	1.69(29)

TABLE 7
Results for NS3D.

NS3D (neq=684)							
Level	Threshold	Ordering	Ordering timing	Nonzeros in L	Nonzeros in U	Fact. timing	Solution timing
		ORG		26123	26123	2.06	diverge
0	0.0	*MDF*	136.69	26123	26123	2.22	12.59(245)
		MUM	9.39	26123	26123	2.13	6.33(121)
		ORG		46610	46610	3.42	diverge
1	0.0	*MDF*	2406.56	61970	61970	5.43	12.93(157)
		MUM	114.74	74790	74790	8.14	11.32(121)
		ORG		45708	45726	3.38	diverge
1	1.0^{-3}	*MDF*	1903.47	57619	58529	5.07	9.52(121)
		MUM	60.37	53374	60699	5.04	11.15(141)
		ORG		60360	60360	4.78	diverge
2	0.0	*MDF*	6590.49	82100	82100	8.20	0.70(7)
		MUM	337.70	121392	121392	18.48	7.66(56)
		ORG		59248	59238	4.72	diverge
2	1.0^{-3}	*MDF*	5952.01	79343	80678	8.83	3.45(34)
		MUM	163.81	76666	92216	9.27	2.90(28)

REFERENCES

[1] O. Axelsson, *Solution of Linear Systems of Equations: Iterative Methods*, Springer-Verlag, Berlin, Heidelberg, New York, 1977, pp. 2–51.

[2] A. Behie and P. A. Forsyth, *Comparison of fast iterative methods for symmetric systems*, IMA J. Num. Anal., 3 (1983), pp. 41–63.

[3] T. F. Chan, *Analysis of preconditioners for domain decomposition*, SIAM J. Num. Anal., 24 (1987).

[4] T. F. Chan and D. C. Resasco, *Analysis of domain decomposition preconditioners on irregular regions*, in Advances in Computer Methods for Partial Differential Equations VI, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, 1987, pp. 317–322.

[5] P. Chin, E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Iterative methods for solution of full Newton Jacobians in incompressible viscous flow (in preparation)*, research report, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1990.

[6] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Towards a cost-effective high order ILU preconditioner*, Research Report CS-90, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1990.

[7] ——, *An automatic ordering method for incomplete factorization iterative solvers*, in Proceedings of the 1991 Reservoir Simulation Symposium, Anaheim, 1991. Paper SPE 21226.

[8] ——, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM Journal On Matrix Analysis and Applications, (1991). (to appear).

[9] ——, *Some decay estimate for the elements in LU factorization*, Research Report CS-91-, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1991.

[10] H. A. V. der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comp., (1991). (to appear).

[11] H. A. V. der Vorst and P. Sonneveld, *CGSTAB: A more smoothly converging variant of CG-S*, Tech. Report Report 90-50, Delft University of Technology, Delft, the Netherlands, May 21, 1990.

[12] S. Doi, *A graph-theory approach for analyzing the effects of ordering on ILU preconditioning*, SIAM J. Sci. Statist. Comp. (submitted).

[13] S. Doi and A. Lichnewsky, *An ordering theory for incomplete LU factorizations on regular grids*, SIAM J. Sci. Statist. Comp. (submitted).

[14] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, (1989), pp. 635–657.

[15] V. Eijkhout, *Analysis of parallel incomplete point factorizations*, Tech. Report CSRD Report No. 1045, Center for Supercomputing Research and Development, University of Illinois, Urbana, Illinois, 1990.

[16] P. A. Forsyth, *A control volume finite element approach to NAPL groundwater contamination*, SIAM J. Sci. Statist. Comp., (to appear) (1990).

[17] A. George and J. W. H. Liu, *Computer solution of large sparse positive-definite systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[18] A. Greenbaum and G. Rodrigue, *Optimal preconditioners of a given sparsity pattern*, BIT, 29 (1989), pp. 610–634.

[19] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[20] T. Hughes, I. Levit, and J. Winget, *An element by element solution algorithm for problems of structural and solid mechanics*, Computer Methods in applied Mechanics and Engineering, 36 (1983), pp. 241–254.

[21] P. S. Huyakorn and G. F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, New York, 1983.

[22] D. S. Kershaw, *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, Journal of Computational Physics, 26 (1978), pp. 43–65.

[23] D. Keyes and W. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations*, SIAM J. Sci. Statist. Comp., 8 (March (1987)), pp. 166–202.

[24] L. Y. Kolotilina and A. Y. Yeremin, *On a family of two-level preconditionings of the incom-*

plete block factorization type, Sov. Journal of Numerical analysis and Mathematical Modelings, 1 (1986), pp. 293–320.

[25] H. P. Langtangen, *Conjugate gradient methods and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns*, Int. J. Num. Meth. Fluids, 9 (1989), pp. 213–233.

[26] F. W. Letniowski, *Three dimensional Delauney triangulations for finite element approximations to a second order diffusion operator*, SIAM J. Sci. Statist. Comp., (accepted) (1989).

[27] H. M. Markowitz, *The elimination form of the inverse and its application to linear programming*, Management Science, 3 (1957), pp. 255–269.

[28] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a M-matrix*, Mathematics Of Computation, 31 (1977), pp. 148–162.

[29] ———, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, Journal of Computational Physics, 44 (1981), pp. 134–155.

[30] N. Munksgaard, *Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients*, ACM Trans. Math. Software, 6 (June 1980), pp. 206–219.

[31] J. M. Ortega, *Orderings for conjugate gradient preconditionings*, Tech. Report TR-90-24, Department of Computer Science, University of Virginia, Charlottesville, Virginia, August 14, 1990.

[32] S. V. Parter, *The use of linear graphs in Gaussian elimination*, SIAM Review, 3 (1961), pp. 364–369.

[33] S. V. Patankar, *Numerical heat tranfer and fluid flow*, Hemisphere, New York, 1980.

[34] O. Pironneau, *Finite element methods for fluids*, John Wiley & Sons, New York, 1989.

[35] H. L. Stone, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Num. Anal., 5 (1968), pp. 530–558.

[36] G. Townsend and W.-P. Tang, *Matview 1.1 — A two dimensional matrix visualization tool*, Tech. Report CS-89-36, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1989.

[37] Z. Zlatev, *Use of iterative refinement in the solution of sparse linear systems*, SIAM J. Num. Anal., 19 (1982), pp. 381–399.