SIAM J. MATRIX ANAL. APPL.
Vol. 12, No. 4, pp. 000–000, July 1992

ORDERING METHODS FOR PRECONDITIONED CONJUGATE GRADIENT METHODS APPLIED TO UNSTRUCTURED GRID PROBLEMS [*]

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

Abstract. It is well known that the ordering of the unknowns can have a significant effect on the convergence of Preconditioned Conjugate Gradient (PCG) methods. There has been considerable experimental work on the effects of ordering for finite difference problems. In many cases, good results have been obtained with preconditioners based on diagonal, spiral, red/black reduced system orderings or some others. The reduced system approach generally gives rapid convergence. There has been comparatively less work on the effect of ordering for finite element problems on unstructured meshes. In this paper, we develop an ordering technique for unstructured grid problems. At any stage of the partial elimination, the next pivot node is selected so as to minimize the norm of the discarded-fill matrix. Numerical results are given for model problems and for problems arising in groundwater contamination. Computations are reported for two-dimensional triangular grids, and for three-dimensional tetrahedral grids. The examples show that ordering is important even if a reduced system (based on a generalized red/black ordering) method is used.

Key Words. ordering method, preconditioned conjugate gradient method

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

1. Introduction.

It is well known that the ordering of the unknowns can affect the convergence behavior of preconditioned conjugate gradient methods. There have been many studies of the use of various ordering techniques coupled with incomplete LU (ILU) factorization preconditioners [3, 4, 5, 7, 11, 12, 13, 15, 16, 17, 25, 29, 30, 34, 35, 38, 39].

Most of these studies have been restricted to the analysis of partial differential equation problems arising from five or seven-point finite difference discretizations in two or three dimensions. For the most part, these ordering methods are based on the graph of the matrix, and do not use actual values of the matrix entries.

In general, the results can be summarized as follows:

1. Random orderings are poor.
2. "Natural" row orderings perform quite well.
3. Fill-reducing orderings, such as minimum degree and nested dissection are poor.
4. Reduced systems are very effective (red/black ordering of a bipartite graph, and exact elimination of the red nodes).

Incomplete factorizations can be classified by the allowed level of fill [17, 29, 38, 39]. A rigorous definition of the level is given in § 2.2 . It is generally agreed that level 1 or level 2 ILU factorization are usually the best in terms of total work required

[*] 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.

[†] Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. Present address, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831.

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

for convergence [3, 4, 17, 25, 29, 38]. Consequently, at least for five or seven-point molecules, a reduced system level 1 or level 2 ILU is a popular choice.

For finite element type discretizations on unstructured grids, it is possible to define a *generalized red/black ordering*, and use a reduced system preconditioner, even in the finite element case. Red nodes are defined as being connected only to black nodes, while black nodes have at least one red neighbor [4, 20]. The red nodes are eliminated exactly and the remaining matrix constitutes the reduced system. However, if the average node connectivity is large, then the number of red nodes can be small, and this approach may not be very advantageous. It is also not clear how to order the remaining black nodes in the reduced system.

An ordering based solely on the graph of the matrix cannot detect anisotropies. For example, consider the equation

$$(KU_x)_x + U_{yy} = f(x,y) \tag{1}$$

with $K \gg 1$. If this equation is discretized using the usual five-point molecule, then, as will be shown in the numerical results (see § 4), the effect of ordering on the convergence of PCG is very large.

Note that numerical anisotropy is very common in practical situations. Even if the equation coefficients are not anisotropic, it is often the case that the grids are very anisotropic. This is especially common in geophysical applications (reservoir simulation, groundwater contamination) where the vertical distance is often one or two orders of magnitude smaller than the horizontal distances.

The idea of developing an ILU factorization based on a drop tolerance has been suggested by Munksgaard [28], Zlatev [40], Tuff and Jennings [37]. In this case, the sparsity pattern of the ILU was determined by a drop tolerance. However, if the initial ordering is poor then the fill may not decay very rapidly, leading to a dense ILU factorization, and hence an inefficient PCG method.

The objective of this paper is to develop an *automatic* method for producing an ordering that reduces the discarded fill in an ILU factorization. We are particularly interested in solving time-dependent problems, which are typical of groundwater contamination modeling. In this case, the order of magnitude of the matrix coefficients is determined by time-invariant physical parameters. Consequently, an ordering can be determined at the start of a simulation, and used for many Newton iterations. The cost of the ordering can then be amortized over many solves [6, 10].

The ordering method used in this work assumes that the level of fill is given, and then the ordering is selected so as to minimize discarded fill. Comprehensively varying the level of fill as well as the ordering is also a possibility but this is beyond the scope of this work.

Note that if a very high level of fill is allowed in the ILU factorization, then a fill reducing ordering such as Reverse Cuthill-McKee (RCM) [8, 27] may become efficient. This is because a higher level of fill can be retained for a given number of nonzeros in the ILU factorization, compared to orderings that do not try to minimize the number of nonzeros in the incomplete factors. For a five-point molecule on a square grid, RCM has fewer nonzeros in the factors for level 3 factorizations (and higher) compared to natural row orderings [3]. Of course, in the extreme case that the allowed level of fill becomes infinite, then fill reducing orderings are clearly more efficient than other orderings (since the method is now a direct technique). Consequently, we shall concern ourselves with low levels of fill in the following, since this is usual in practice.

Natural or row orderings applied to structured finite difference grids have the property that nodes ordered consecutively are (graph) neighbors of previously ordered nodes. An obvious generalization of this idea to unstructured grids is an RCM ordering.

Test results will be presented for some matrices generated by two and three-dimensional groundwater contamination simulations (triangular and tetrahedral elements). These problems have large jump discontinuities in absolute permeability [24], and therefore constitute a severe test of the ordering algorithm. Some results are also given for some standard two-dimensional model problems [15, 36].

The results are compared using natural (row) ordering, RCM, and the Minimum Discarded Fill (MDF) technique developed in this work. Factorization levels are varied from level 0 to level 3, and both full and reduced system methods are used.

2. Minimum Discarded Fill Ordering.

2.1. Motivation by Matrix Formulation. The Cholesky factorization of an $n \times n$ symmetric positive definite matrix A can be described by the following equations:

$$A = A_0 = \begin{bmatrix} d_1 & {\gamma_1}^t \\ \gamma_1 & B_1 \end{bmatrix} = L_1 \begin{bmatrix} 1 & 0 \\ 0 & A_1 \end{bmatrix} {L_1}^t, \tag{2}$$

where

$$L_1 = \begin{bmatrix} \sqrt{d_1} & 0 \\ \gamma_1/\sqrt{d_1} & I_{n-1} \end{bmatrix}, \quad A_1 = B_1 - \gamma_1 {\gamma_1}^t / d_1. \tag{3}$$

At the k^{th} step,

$$A_{k-1} = \begin{bmatrix} d_k & {\gamma_k}^t \\ \gamma_k & B_k \end{bmatrix} = L_k \begin{bmatrix} 1 & 0 \\ 0 & A_k \end{bmatrix} {L_k}^t, \tag{4}$$

where

$$L_k = \begin{bmatrix} \sqrt{d_k} & 0 \\ \gamma_k/\sqrt{d_k} & I_{n-k} \end{bmatrix}, \quad A_k = B_k - \gamma_k {\gamma_k}^t / d_k. \tag{5}$$

Here I_k denotes a $k \times k$ identity matrix, d_k a scalar, γ_k is a column vector of length $n-k$. The matrix A_k is the $(n-k) \times (n-k)$ submatrix that remains to be factored after the first k steps of the factorization.

In the incomplete factorization of matrix A, some of the entries in the factor are discarded to prevent excessive fill and computation. Let matrix F_k contain the discarded values. Then the incomplete factorization proceeds with the perturbed matrix,

$$\tilde{A}_k = A_k - F_k = B_k - \gamma_k {\gamma_k}^t / d_k - F_k. \tag{6}$$

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

$$\|F_k\|_F = \left(\sum_{i \geq 1} \sum_{j \geq 1} |f_{ij}^{(k)}|^2 \right)^{1/2}. \tag{7}$$

The discarded fill at the k^{th} step for an arbitrary node is similarly defined by performing a symmetric permutation that exchanges this node with the k^{th} node. To determine the sparsity pattern for matrix F_k that will yield a high quality preconditioner is still a very interesting research subject. A current popular choice is to discard the fills that have a "higher fill level" during the incomplete factorization [23]. The simplest strategy is ILU(0) where all new fill is discarded and ILU(1) where only level 1 fills produced by eliminating original nonzeros are retained but higher level fill produced in the elimination of level 1 fill is discarded. The notion of "fill level" will be defined more precisely through the graph model presented in § 2.2

The basic idea of the minimum discarded fill ordering scheme is to eliminate the node with the minimum discarded fill at each stage of the incomplete factorization. This scheme can be considered as the numerical analogue of the minimum deficiency ordering strategy [14] for minimizing the amount of fill. The most computationally intensive calculations are in the updating of new discard values after each stage of the factorization process.

2.2. Graph Model. In this section we present a graph model [31, 33] for describing the factorization process as a series of node eliminations. The graph model is invaluable in providing an insight into the minimum discarded fill ordering.

To simplify notation, we present a symmetric case and assume the elimination sequence is $v_1, v_2, \ldots, v_n$. Let graph $G_k = (\mathcal{V}_k, \mathcal{E}_k)$, $k = 0, 1, \ldots, n-1$ be the graph corresponding to matrix $A_k = \left[a_{ij}^{(k)}\right]$ of (5). The vertex set and edge set are defined as

$$(8) \qquad \mathcal{V}_k = \{v_{k+1}, v_{k+2}, \ldots, v_n\}, \quad \mathcal{E}_k = \left\{(v_i, v_j) \mid a_{ij}^{(k)} \neq 0\right\} .$$

We assume each vertex has a self-loop edge (v_i, v_i) and each edge (v_i, v_j) has a value of $a_{ij}^{(k)}$.

The notion of "fill level" can be defined through reachable sets [22] in the graph G_0. Let $\mathcal{S}$ be a subset of the node set, $\mathcal{S} \subset \mathcal{V}_0$, and nodes $u, v \notin \mathcal{S}$. Node u is said to be reachable from a vertex v through $\mathcal{S}$ if there exists a path $(v, u_1, \ldots, u_m, u)$ in graph G_0, such that each $u_i \in \mathcal{S}$, $1 \leq i \leq m$. Note that m can be zero, so that any adjacent pair of nodes $u, v \notin \mathcal{S}$ is reachable through $\mathcal{S}$. The reachable set of v through $\mathcal{S}$ is denoted by

$$(9) \qquad \mathit{Reach}(v, \mathcal{S}) = \{u \mid u \text{ is reachable from } v \text{ through } \mathcal{S}\} .$$

Let $\mathcal{S}$ be the set of eliminated nodes so far, $\{v_1, \ldots, v_k\}$, and let $v_j \in \mathit{Reach}(v_i, \mathcal{S})$ with the *shortest* path $(v_i, u_1, \ldots, u_m, v_j)$, and nodes u's in $\mathcal{S}$ are eliminated nodes. We define the fill level for entry $a_{ij}^{(k)}$ to be the length of the shortest path from v_i to v_j minus one, i.e. $\mathit{Level}(a_{ij}^{(k)}) = m$. We initially set

$$(10) \qquad \mathit{Level}(a_{ij}^{(0)}) = \begin{cases} 0 & \text{if } a_{ij} \neq 0, \\ \infty & \text{otherwise.} \end{cases}$$

Since $\mathit{Level}(a_{ij}^{(k)})$ is defined by reachable sets through $\{v_1, \ldots, v_k\}$, as more nodes are eliminated, there may be a shorter path between v_i and v_j. Thus as the elimination proceeds, the fill levels are modified by

$$(11) \qquad \mathit{Level}(a_{ij}^{(k)}) := \min\left(\mathit{Level}(a_{ik}^{(k-1)}) + \mathit{Level}(a_{kj}^{(k-1)}) + 1,\ \mathit{Level}(a_{ij}^{(k-1)})\right) .$$

It is possible to define a fill level independent of k, if the order of the unknowns is predetermined. In this application, however, the order of the unknowns is dynamically changing during the incomplete factorization. A predetermined level, therefore, is not practical.

The elimination of v_k to form A_k can be modeled as a graph transformation [33],

$$a_{ij}^{(k)} = \begin{cases} a_{ij}^{(k-1)} - \dfrac{a_{ik}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}} & \text{if } (v_i, v_k) \text{ and } (v_k, v_j) \in \mathcal{E}_{k-1}, \\ a_{ij}^{(k-1)} & \text{otherwise.} \end{cases}$$

Note that if $a_{ij}^{(k-1)}$ is a zero entry, $(v_i, v_j) \notin \mathcal{E}_{k-1}$, then the elimination of their common neighbor v_k would create at position $a_{ij}^{(k)}$, a new fill of value

$$0 - \frac{a_{ik}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}.$$

With the minimum discarded fill reordering that corresponds to an ILU(0) factorization, only entries with fill level zero are kept, i.e. all new fill-in's must be discarded. If node v_m were eliminated after the k^{th} stage of the incomplete factorization (equation (6)), the *discarded fill* value for node v_m would be

$$\text{(12)} \qquad \mathit{discard}(v_m) = \left[\sum_{(v_i,v_j) \in \mathcal{F}} \left(\frac{a_{im}^{(k-1)} a_{mj}^{(k-1)}}{a_{mm}^{(k-1)}} \right)^2 \right]^{1/2} ,$$

where

$$\text{(13)} \qquad \mathcal{F} = \{(v_i, v_j) \mid (v_i, v_j) \notin \mathcal{E}_{k-1}, \quad (v_i, v_m) \in \mathcal{E}_{k-1}, \quad (v_m, v_j) \in \mathcal{E}_{k-1}\} \ ,$$

and $G_{k-1} = (\mathcal{V}_{k-1}, \mathcal{E}_{k-1})$ is the graph corresponding to matrix A_{k-1}. The minimum discarded fill strategy can be generalized to correspond to ILU(ℓ) factorization by accounting for only new fill-in's with fill level greater than ℓ in the computing of *discarded fill* value. Then set $\mathcal{F}$ in (12) is taken to be

$$\text{(14)} \qquad \begin{aligned} \mathcal{F} \;=\; \{(v_i, v_j) \mid (v_i, v_j) \notin \mathcal{E}_{k-1}, \quad (v_i, v_m) \in \mathcal{E}_{k-1}, \\ (v_m, v_j) \in \mathcal{E}_{k-1} \text{ and } \mathit{Level}(a_{ij}^{(k)}) > \ell\}. \end{aligned}$$

In the following discussion, we shall denote MDF(ℓ) as the minimum discarded fill ordering corresponding to an ILU(ℓ) factorization.

Observation 1:

For the MDF(ℓ) algorithm, discard values for all nodes can be initially precomputed. At each elimination step, if v_k is chosen to be eliminated, only discard values of the neighbors of v_k need to be updated.

Observation 2:

The MDF(0) algorithm overwrites the original matrix A with the corresponding ILU(0) incomplete factorization.

2.3. MDF(ℓ) Algorithm. The MDF(ℓ)ordering algorithm can be described as follows:

```
Initialization:
        $A = A_0$
        for each $a_{ij} \neq 0$
            $Level(a_{ij}) := 0$
        end
        for each node $v_i$
            Compute the discarded fill value $discard(v_i)$ from (12), and (14).
        end
for k = $1 \ldots n-1$
    Choose a node $v_m$ that has the minimum $discard(v_m)$ as the next pivot node
    (see tie-breaking section).
    Update the decomposition,
```

$$\tilde{A}_k = B_k - \gamma_k \gamma_k^t / d_k - F_k, \quad \text{where } P_k A_{k-1} P_k^t = \begin{bmatrix} d_k & \gamma_k \\ \gamma_k^t & B_k \end{bmatrix}.$$

```
    $P_k$ is permutation matrix to exchange $v_k$ with $v_m$ and $F_k$ is the matrix of dis-
    carded fill-in entries,
```

$$F_{ij}^{(k)} := \begin{cases} \dfrac{a_{im}^{(k-1)} a_{mj}^{(k-1)}}{a_{mm}^{(k-1)}} & \text{if } Level(a_{ij}^{(k)}) > \ell \text{ and } a_{ij}^{(k-1)} = 0, \\ 0 & \text{otherwise.} \end{cases}$$

```
    Update the discarded values of $v_m$'s neighbors.
    Update the fill level of entries in $\tilde{A}_k$ by (11).
end
```

2.4. Tie-breaking. There are often cases where many nodes will have the same (typically zero) discarded fill. Several possible tie breaking strategies are investigated in the following.

Ties can be broken by selecting the nodes that have the smallest degree in the incomplete factorization (smallest number of non-zeros in the row). Another possibility is to use the node with the smallest deficiency (smallest number of new non-zero fill elements introduced if this node is used as a pivot) [14]. If there are still ties remaining, then the node that has the smallest discarded fill from a previous stage of the incomplete factorization is selected first. If further ties exist, the unordered node with the smallest original number is selected. The minimum deficiency and minimum degree strategies attempt to minimize the number of fill elements in the case of ties.

Tests were run using minimum deficiency, minimum degree, and random tie breaking for all our test problems. On average, the minimum degree strategy required 2% more solution time than minimum deficiency, while random tie breaking required 13% more solution time than minimum deficiency. Consequently, all test results will be reported using minimum deficiency tie-breaking. Our tests also show that these tie breaking algorithms have little effect on the cost of the MDF ordering. Therefore, no timing comparison is given.

2.5. An Example of MDF(0) Ordering. In this section we consider an example of an MDF(0) ordering on the model Laplace's problem. The Laplace's problem with Dirichlet boundary conditions is discretized by the 5-point molecule on a regular

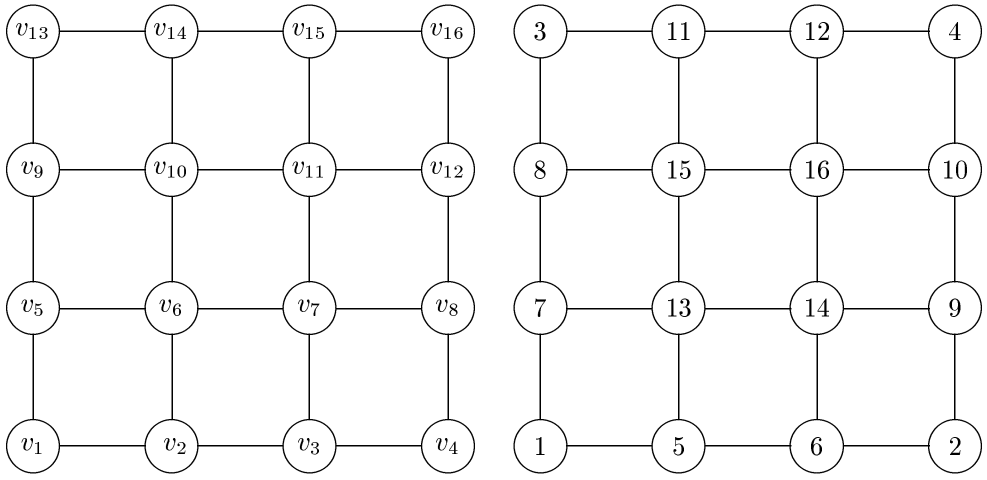

FIG. 1. *Natural row ordering and final MDF*(0) *ordering.*

4×4 grid. Figure 1 displays the grid with initial natural row ordering.

The initial discard values for the four corner nodes $\{v_1, v_4, v_{13}, v_{16}\}$ are equal to $discard(v_1) = \sqrt{0.125} \approx 0.354$. The discard values for the boundary nodes $\{v_2$, v_3, v_5, v_8, v_9, v_{12}, v_{14}, $v_{15}\}$ are equal to $discard(v_2) = \sqrt{0.375} \approx 0.612$. Similarly, discard values of interior nodes $\{v_6, v_7, v_{10}, v_{11}\}$ are equal to $discard(v_6) = \sqrt{0.75} \approx 0.866$. The corner nodes have the smallest discard values and should be eliminated first. Note that these corner nodes are not connected and their discard values are unaffected by the elimination of other corner nodes. After the corner node v_1 is eliminated, its boundary neighbor node v_2 has new discard value given as $discard(v_2) = \sqrt{32}/15 \approx 0.377$. After the four corner nodes are eliminated, the discard values for the interior nodes are unchanged. For example, $discard(v_2) = \sqrt{32}/15 \approx 0.377$, and discard values for interior nodes are unchanged at $discard(v_6) = \sqrt{0.75} \approx 0.866$. The boundary nodes have the smallest discard values. Boundary node v_2 is chosen by the tie-breaking strategy, and its neighbor v_3 with $discard(v_3) = 0$, could be eliminated next with no fill. Similarly, nodes v_5 and v_9, v_8 and v_{12}, v_{14} and v_{15} will be eliminated in sequence. Node v_{15} will be eliminated next since $discard(v_{15}) = 0$. Further computation would show than an MDF(0) ordering for this example is given by Figure 1.

2.6. MDF(1) **Ordering.** In § 2.5, we looked at an example of minimum discarded fill ordering that corresponds to an ILU(0) incomplete factorization. This ordering is abbreviated as the MDF(0) ordering. The MDF(1) ordering is the extension of this basic strategy to correspond to an ILU(1) incomplete factorization.

Although an MDF(0) ordering overwrites the original matrix A with its ILU(0) factorization, MDF(1) does not *exactly* reproduce the ILU(1) factorization. There are some subtleties in the computing of level 2 contributions that happen to fall upon non-zero entries. Consider the scenario in Figure 2, where v_1 and v_2 have been eliminated causing fill contribution to edges (v_3, v_4) and (v_3, v_5). Suppose we wish to eliminate v_3 next. This would cause a level 2 fill contribution to edge (v_4, v_5). The subtle problem is in deciding whether this (v_4, v_5) level 2 fill should be discarded. Note if we knew in advance that v_6 would be eliminated before v_4 and v_5, then this elimination of v_6 would cause a level 1 fill contribution to edge (v_4, v_5). Thus this level 2 contribution of (v_4, v_5) would fall on a non-zero entry and may be accepted. The MDF(1) algorithm

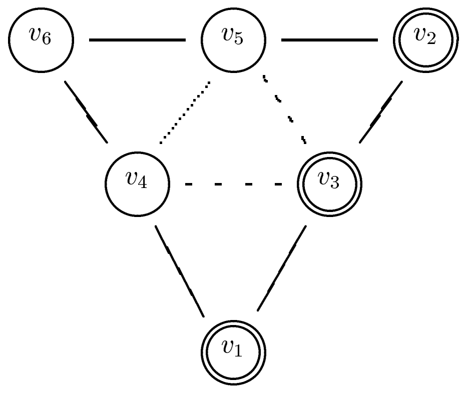

FIG. 2. *Level-2 fill at* (v_4, v_5).

always discards (pessimistically) the level 2 fill contribution (v_4, v_5).

Axelsson and Gustafsson [1] have observed that reduced system preconditioners are very effective (red/black partitioning of nodes and exact elimination of the red nodes). It is interesting to note that a generalized red/black partitioning of the nodes is an MDF(1) ordering of the red nodes. A generalized red/black partitioning of a graph has the property that each red node has *only* black neighbors and each black node has at least one red neighbor. In the special case where each black node has *only* red neighbors, the graph is bipartite, or the corresponding matrix is two-cyclic. Note in this special case, the red nodes form an independent set so that the discard value for each red node is unaffected by elimination of other red nodes; therefore, the elimination order of the red nodes is immaterial.

REMARK 2.1. *A generalized red/black partitioning of the nodes is an MDF*(1) *ordering of the red nodes.*

In an ILU(1) incomplete factorization, all level 1 fill is accepted. Hence if a vertex has no eliminated neighbors, its discard value is zero and would be a candidate for selection by the MDF(1) criterion. A generalized red/black partition of the nodes orders red nodes first, and these red nodes have (by definition) zero discarded fill.

REMARK 2.2. *If a matrix is symmetric and two-cyclic (its graph is bipartite), then an MDF*(0) *ordering on the reduced matrix formed with the bipartite red/black partition is an MDF*(1) *reordering on the original matrix.*

The reduced matrix is obtained by exact elimination of the red nodes. Since the graph is bipartite, there are no black to black connections in the original graph. Therefore all black to black connections in the reduced matrix are level 1 fill. Level 3 fill from the original graph is exactly the new level 1 fill generated from the reduced matrix. Thus an MDF(0) ordering on the reduced matrix is an MDF(1) reordering on the original matrix.

While a generalized red/black partition is an MDF(1) ordering of the red nodes, an MDF(1) ordering may or may not produce a generalized red/black partitioning. Consider a tridiagonal matrix. All nodes initially have no level 1 discarded fill. Consequently, the ordering depends crucially on the tie-breaking strategy. For example, either a red/black partition or the perfect elimination order (no fill) would be consistent with the MDF strategy, in this case.

Although the description of MDF(0) and MDF(1) orderings has been given for

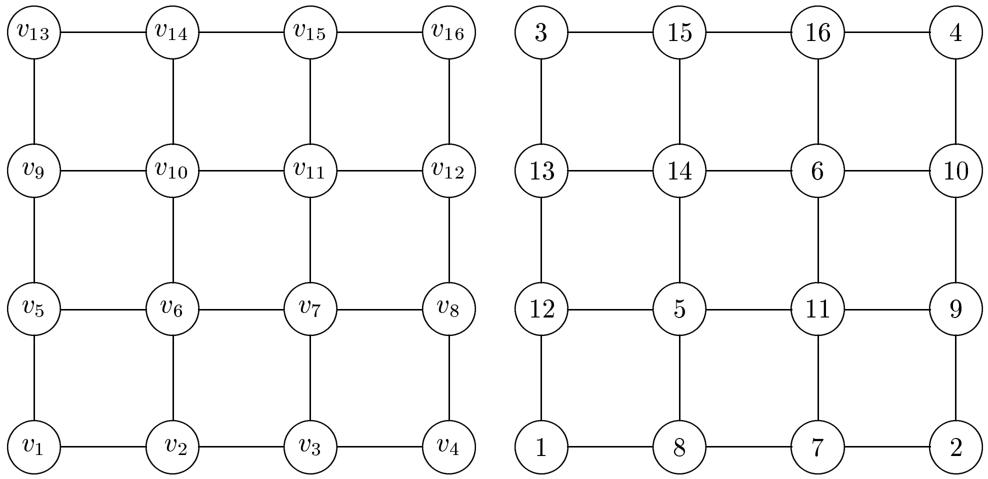

FIG. 3. *Natural row ordering and final MDF*(1) *ordering.*

symmetric matrices, it is clearly trivial to generalize to the case of a non-symmetric matrix having a symmetric incidence matrix. This is how we have in fact, implemented the MDF(ℓ) ordering algorithms. Of course, the minimum discarded fill algorithm can also be applied to matrices with non-symmetric non-zero structure, and our findings will be reported in a forthcoming paper [9]

2.7. An Example of MDF(1) Ordering. We consider an example of an MDF(1) ordering on the model Laplace's problem used in § 2.5. The minimum deficiency criterion is used for tie breaking.

Since level one fill entries are accepted, initially all nodes have zero discard values. The four corner nodes $\{v_1, v_4, v_{13}, v_{16}\}$ are chosen based on deficiency. Nodes v_6 and v_{11} are chosen next since these have zero discard values. Nodes $\{v_3, v_8, v_9, v_{14}\}$ have the same discard value $\mathit{discard}(v_3) \approx 0.094$, nodes $\{v_2, v_5, v_{12}, v_{15}\}$ have discard value, $\mathit{discard}(v_2) \approx 0.226$, nodes $\{v_7, v_{10}\}$ have discard value, $\mathit{discard}(v_7) \approx 0.381$. By the tie-breaking criterion, v_3 will the next eliminated node. Then v_2 followed by v_8 and v_{12} are eliminated with no new fill. Among the uneliminated nodes, v_7 has the smallest discard value of $\mathit{discard}(v_7) \approx 0.056$. After v_7 is eliminated, there is a perfect elimination sequence of v_5, v_9, v_{10}, v_{14} and v_{15}. The final ordering is shown in Figure 3.

3. Test Problems. The minimum discarded fill orderings were tested on a variety of problems. For Problems 1,2,4 below, the matrices are only positive semi-definite. The solution is determined only to within a constant. These matrices can be made definite by fixing the solution at a single node. However, the conjugate gradient method still converges even if this is not done. In fact, if the solution is fixed at a node, the algorithm actually converges more slowly [2, 21] . For Problems 1,2,4, the matrices are left as semi-definite.

3.1. Problem 1 (STRONGX). The first problem solves the equation

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial P}{\partial y}\right) = -q \tag{15}$$

on the region $x \in [0,1]$, $y \in [0,2]$, using a five-point cell centered finite difference discretization [32] with Neuman boundary conditions, where $K_x = 1000$ and $K_y = 1$

. Let $h = 1/30$ be the cell spacing; $n_x = 30$, $n_y = 60$ be the number of cells in the x and y direction respectively. The source term is:

$$q(x, y) = \begin{cases} -1/h^2 & \text{if } (x, y) = (h/2, h/2), \\ 1/h^2 & \text{if } (x, y) = (1 - h/2, 2 - h/2), \\ 0 & \text{elsewhere.} \end{cases}$$

3.2. Problem 2 (STRONGY). This problem is identical to Problem 1, except that the anisotropic property is reversed, $K_x = 1, \quad K_y = 1000$.

3.3. Problem 3 (LAPD5). This is Laplace's equation on the unit square with Dirichlet boundary conditions, as used in [15]. The usual five-point finite difference discretization was used on a regular 30×30 grid.

3.4. Problem 4 (STONE). This problem is Stone's third problem [36]. The equation

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial P}{\partial y}\right) = -q, \tag{16}$$

was discretized on the unit square using a vertex centered finite difference technique [32], with Neuman boundary conditions. If the node spacing is $h = 1/30$, then

$$x_i = ih, \quad y_j = jh, \quad 0 \le i, j \le 30.$$

We will refer to the location of the source and sink terms by

$$q(x_i, y_j) \quad = \quad q(i, j)$$

in the following and in Figure 4 which shows the problem domain.

The values of K_x, K_y and q were:

$$(K_x, K_y) = \begin{cases} (1, 100) & \text{if } (x_i, y_j) \in B, \quad 14 \le i \le 30,\ 0 \le j \le 16, \\ (100, 1) & \text{if } (x_i, y_j) \in C, \quad 5 \le i \le 12,\ 5 \le j \le 12, \\ (0, 0) & \text{if } (x_i, y_j) \in D, \quad 12 \le i \le 19,\ 21 \le j \le 28, \\ (1, 1) & \text{if } (x_i, y_j) \in A, \end{cases} \tag{17}$$

$$\begin{aligned} &q_1(3, 3) = 1.0, \quad q_2(3, 27) = 0.5, \quad q_3(23, 4) = 0.6, \\ &q_4(14, 15) = -1.83, \quad q_5(27, 27) = -0.27\ . \end{aligned} \tag{18}$$

A 31×31 grid was used, and an harmonic average was used for to define K_x and K_y [3] at cell boundaries.

Test problems (5–7) are derived from two and three-dimensional pressure equations arising in groundwater contamination simulations [18, 24]. The pressure equation is essentially equation (16). 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. The choice of boundary conditions (fixed pressure) resulted in a sparse right hand side vector.

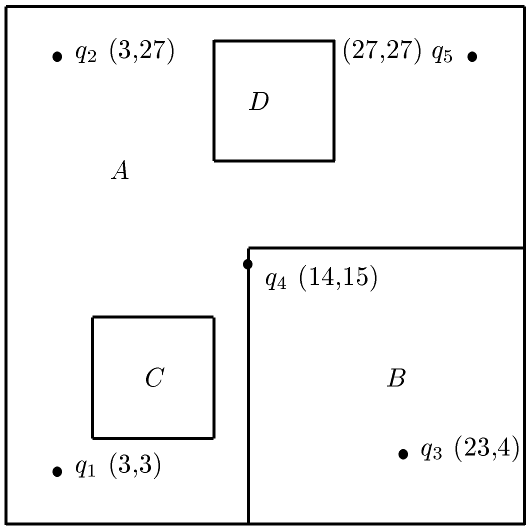

FIG. 4. *Stone's third problem.*

3.5. Problem 5 (REFINE2D). A finite element method using linear triangular basis functions was used to discretize this problem. In this example, K_x and K_y were constant. The triangulation is such that the resulting equation is an M-matrix [18]. 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. This problem had 1161 nodes, and is described in more detail in [18]. The nodes are originally ordered using an RCM ordering.

3.6. Problem 6 (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 (16)) varied by four orders of magnitude. The grid, which had 1521 nodes, was defined by constructing a distorted quadrilateral grid, and then triangulating in the obvious manner. A Delaunay-type edge swap was used to produce an M-matrix. The original ordering for this problem used a natural or lexicographic ordering based on the distorted quadrilaterals. This problem is described in more detail in [18].

3.7. Problem 7 (FE3D). This problem is a three-dimensional version of equation (16). 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]. The original ordering for this problem used a natural ordering based on distorted hexahedra. This problem is described in more detail in [19].

4. Results. The computations to solve the test problems (1-5) were done on a Sun SPARC SLC workstation in double precision and using

$$\|\mathbf{r}^k\|_2 \le \varepsilon \|\mathbf{r}^0\|_2, \quad \varepsilon = 10^{-6} \tag{19}$$

Table 1
Summary for test problem STRONGX

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
ORG(0)	3510	n/a	0.25	4.49(33)
ORG(1)	5221	n/a	0.27	4.64(32)
ORG(2)	6903	n/a	0.31	4.65(31)
ORG(3)	10238	n/a	0.42	5.05(30)
RCM(0)	3510	0.06	0.22	4.46(33)
RCM(1)	5221	0.07	0.27	4.65(32)
RCM(2)	6903	0.06	0.30	1.98(13)
RCM(3)	8527	0.06	0.36	2.11(13)
MDF(0)	3510	1.65	0.23	4.46(33)
MDF(1)	6867	3.38	0.31	2.01(13)
MDF(2)	7074	3.50	0.32	1.72(11)
MDF(3)	10210	7.83	0.46	1.56(9)
Reduced system (black nodes= 900, red nodes= 900)				
ORG(1)	3421	n/a	0.31	3.97(40)
ORG(3)	5060	n/a	0.36	4.10(38)
RCM(1)	3421	0.07	0.31	3.97(40)
RCM(3)	5060	0.07	0.36	1.99(18)
MDF(1)	3421	2.14	0.31	1.17(11)
MDF(3)	6584	6.54	0.44	1.00(8)

Table 2
Summary for test problem STRONGY

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
ORG(0)	3510	n/a	0.22	8.06(60)
ORG(1)	5221	n/a	0.26	2.92(20)
ORG(2)	6903	n/a	0.30	3.02(20)
ORG(3)	10238	n/a	0.41	1.72(10)
RCM(0)	3510	0.06	0.22	8.05(60)
RCM(1)	5221	0.06	0.27	2.93(20)
RCM(2)	6903	0.07	0.30	2.87(19)
RCM(3)	8527	0.06	0.35	1.63(10)
MDF(0)	3510	1.63	0.24	8.25(60)
MDF(1)	6873	3.35	0.31	2.16(14)
MDF(2)	7064	3.48	0.31	2.32(15)
MDF(3)	10150	7.69	0.45	1.24(7)
Reduced system (black nodes = 900, red nodes = 900)				
ORG(1)	3421	n/a	0.31	1.93(19)
ORG(3)	5060	n/a	0.36	1.36(12)
RCM(1)	3421	0.06	0.31	1.94(19)
RCM(3)	5060	0.07	0.37	1.14(10)
MDF(1)	3421	2.11	0.32	1.45(14)
MDF(3)	6572	6.53	0.45	0.77(6)

Table 3
Summary for test problem LAPD5

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
ORG(0)	1740	n/a	0.11	1.73(26)
ORG(1)	2581	n/a	0.13	1.24(17)
ORG(2)	3393	n/a	0.15	1.05(14)
ORG(3)	4988	n/a	0.20	0.92(11)
RCM(0)	1740	0.04	0.11	1.74(26)
RCM(1)	2581	0.03	0.13	1.24(17)
RCM(2)	3393	0.04	0.15	0.83(11)
RCM(3)	4177	0.03	0.17	0.80(10)
MDF(0)	1740	0.84	0.11	1.67(25)
MDF(1)	3391	1.67	0.16	0.90(12)
MDF(2)	3571	1.79	0.16	0.92(12)
MDF(3)	5041	3.80	0.22	0.60(7)
Reduced system (black nodes = 450, red nodes = 450)				
ORG(1)	1681	n/a	0.15	0.70(14)
ORG(3)	2465	n/a	0.18	0.56(10)
RCM(1)	1681	0.04	0.16	0.70(14)
RCM(3)	2465	0.03	0.18	0.50(9)
MDF(1)	1681	1.09	0.16	0.65(13)
MDF(3)	3261	3.29	0.22	0.48(8)

Table 4
Summary for test problem STONE

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
ORG(0)	1860	n/a	0.12	3.31(47)
ORG(1)	2760	n/a	0.14	2.08(27)
ORG(2)	3630	n/a	0.16	1.82(23)
ORG(3)	5340	n/a	0.22	1.43(16)
RCM(0)	1860	0.04	0.12	3.32(47)
RCM(1)	2760	0.03	0.14	2.07(27)
RCM(2)	3630	0.03	0.16	1.35(17)
RCM(3)	4471	0.04	0.19	1.18(14)
MDF(0)	1860	0.92	0.12	3.25(46)
MDF(1)	3658	1.81	0.17	1.36(17)
MDF(2)	3819	1.90	0.18	1.38(17)
MDF(3)	5361	3.98	0.24	0.99(11)
Reduced system (black nodes = 480, red nodes = 481)				
ORG(1)	1798	n/a	0.17	1.16(22)
ORG(3)	2638	n/a	0.20	0.87(15)
RCM(1)	1798	0.03	0.17	1.16(22)
RCM(3)	2638	0.03	0.20	0.76(13)
MDF(1)	1798	1.16	0.17	0.91(17)
MDF(3)	3487	3.42	0.24	0.69(11)

Table 5
Summary for test problem REFINE2D

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
RCM(0)	2480	0.05	0.15	4.38(49)
RCM(1)	3571	0.05	0.17	2.86(30)
RCM(2)	4758	0.05	0.21	2.59(26)
MDF(0)	2480	1.19	0.15	4.29(48)
MDF(1)	4645	2.35	0.21	2.10(21)
MDF(2)	5292	3.04	0.24	2.16(21)
Reduced system (black nodes= 648, red nodes= 513)				
RCM(1)	2629	0.04	0.21	1.96(28)
RCM(2)	2952	0.05	0.23	1.80(25)
MDF(1)	2753	1.72	0.23	1.51(21)
MDF(2)	3390	2.52	0.25	1.59(21)

Table 6
Summary for test problem FE2D

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
RCM(0)	4373	0.06	0.25	6.40(49)
RCM(1)	5846	0.07	0.29	4.12(30)
RCM(2)	8325	0.07	0.37	3.31(22)
MDF(0)	4373	2.09	0.25	3.99(30)
MDF(1)	7942	4.66	0.36	3.43(23)
MDF(2)	10620	8.39	0.49	2.64(16)
Reduced system (black nodes = 1091, red nodes = 430)				
RCM(1)	5286	0.08	0.38	4.00(34)
RCM(2)	6888	0.08	0.43	2.93(23)
MDF(1)	5782	3.86	0.41	2.84(23)
MDF(2)	8565	7.90	0.54	2.33(17)

Table 7
Summary for test problem FE3D

Ordering & level ℓ	Nonzeros in L	Ordering time	Fact. time	Solution time & Iterations
Full system				
RCM(0)	19725	0.23	1.02	16.13(41)
RCM(1)	39066	0.24	2.01	11.70(24)
RCM(2)	72038	0.23	5.27	10.29(16)
MDF(0)	19725	16.49	1.11	10.08(25)
MDF(1)	49599	150.2	3.45	9.88(18)
MDF(2)	92181	1002	10.6	10.69(14)
Reduced system (black nodes = 2593, red nodes = 657)				
RCM(1)	38822	0.23	3.00	10.66(23)
RCM(2)	66933	0.23	6.32	10.32(17)
MDF(1)	43141	143	3.36	8.96(18)
MDF(2)	87472	1067	9.34	9.97(14)

as the stopping criterion, where $\mathbf{r}^k$ is the residual vector after the k^{th} iteration in the conjugate gradient acceleration and the zero vector is the initial guess.

Some tests were carried out using a random initial guess (random numbers between (-1,+1)), and the results were qualitatively similar. The tests were also repeated using a stopping criteria of $\varepsilon = 10^{-12}$, and the trends were similar to the results obtained with $\varepsilon = 10^{-6}$, and hence will not be shown.

The reduced system factorizations were constructed by first using a generalized red/black partitioning of the nodes. The initial red node was selected as the initial node in the given ordering. The initial ordering (ORG) was x–y natural for Problems 1–4, and RCM ordering for Problems 5–7.

The levels of fill will be defined so that all original entries in the full system have level 0. This means that the lowest level reduced system factorization will be level 1. If the original matrix has a bipartite graph, then the next level of fill in the reduced system is level 3. Note that in the finite element case, the next level of fill in the reduced system is level 2. For this reason, our definition of levels for reduced systems differs from that used previously [38]. For all reduced system methods, the ordering was determined using the reduced system. For example, RCM on the reduced system refers to the following sequence of steps: the full system is red/black ordered, the red nodes are eliminated exactly, and the reduced system is reordered using an RCM algorithm.

Table 1 shows the results for Problem STRONGX. This problem has a strong coupling in the x-direction. As discussed in [9] , ORG ordering (x–y natural) is very poor for this example, since the entries in LU factorization decay very slowly with this ordering. This is reflected in the results for ORG ordering, full system, all levels of fill. The full system computations for RCM are poor for levels 0 and 1, but become competitive with MDF as the level of the ILU increases. MDF(0) is poor (level 0 factorizations cannot detect anisotropies [9]), but is much improved for level 1. Note that the ordering time for RCM varies slightly for different runs. This is due to the inaccuracy in the system timing calls.

For the reduced system, MDF(1) is much faster than either RCM(1) or ORG(1). In all cases, the amount of fill in the ILU(1) factorization is identical. This demonstrates that the orderings for reduced system factorizations can be very important. As the level of factorization on the reduced system is increased, the ORG ordering actually becomes slower, while RCM(3) shows a large improvement. However, MDF(1) is still superior to RCM(3) with less fills. If the levels are increased to very high levels, we would expect RCM to eventually become more efficient than MDF, due to the fill reducing property of RCM as discussed in the introduction.

Note also that the cost of the MDF ordering is quite high. However, as discussed previously, we expect to carry out the ordering only once for many matrix solves, for time-dependent, non-linear problems [6, 10].

Table 2 lists the results for Problem STRONGY. In this case, the ORG orderings result in rapid decay in the size of the fill entries, and hence the ORG orderings (for level > 0) are quite efficient. The reduced system factorizations are generally more efficient than the full system factorizations.

For the reduced system, MDF(1) is superior to RCM(1), and ORG(1). All methods have the same fill. MDF(3) is also faster than either ORG(3) or RCM(3), but at the cost of greater fill.

Problem LAPD5 (Table 3) has constant coefficients for all interior nodes, and as expected, all the orderings behave very similarly.

Table 4 shows the results for Problem STONE. The reduced system factorizations are the most efficient for this problem. Again, the reduced system MDF(1) is faster than RCM(1) or ORG(1). However, reduced system MDF(3) and RCM(3) are quite close, especially if the factorization time is included.

The first finite element problem REFINE2D tests are listed in Table 5. The reduced system factorization is effective for this problem since almost half the nodes are exactly eliminated. The reduced system MDF(1) has the smallest solution cost.

Table 6 shows the results for Problem FE2D. Even though the generalized red/black partitioning has only 430 (out of 1521) red nodes, the reduced system factorizations are superior to the full system factorizations. For a given level of fill, reduced system MDF has a smaller solution cost than reduced system RCM.

For the the three-dimensional problem FE3D, high levels of fill are not very effective because of the large amount of fill in the ILU factorization. If factorization cost is included, the best method is MDF(0) (Table 7).

To summarize, for Problems STRONGX, STRONGY, and STONE, the reduced system MDF(1) ordering outperforms RCM(1) and ORG(1). All orderings have the identical amount of fill for level 1 reduced systems. Reduced system MDF(3) is either the best or tied with RCM(3), although at the expense of greater fill. Note that for STRONGX, reduced system RCM(1) is almost four times slower than reduced system MDF(1).

For Problems REFINE2D and FE2D, reduced system MDF(1) again outperforms RCM(1). Reduced system RCM becomes more competitive with reduced system MDF as the level increases. It is interesting to note that for the three-dimensional problem FE3D, MDF(0) is superior to RCM(0).

5. Conclusions. In agreement with previous work, we have found that the ordering of the unknowns has a large effect on the convergence of the ILU preconditioned PCG iterative methods.

In some cases, it is possible to select an a priori ordering that results in rapid convergence. However, for partial differential equation problems that have rapidly varying coefficients, and are discretized on unstructured grids, a good ordering is far from obvious.

As demonstrated in the anisotropic examples, RCM orderings can be quite poor for level 1 fill, for both full system and reduced systems, compared to MDF orderings. Reduced system methods were superior to full system iteration for all the two-dimensional problems. Reduced system MDF orderings with lower fill level outperformed reduced system natural and RCM orderings.

Because of the large factorization cost, and the relatively small number of red nodes exactly eliminated, the reduced system approach was not very effective for the three-dimensional problem. MDF(0) was the best choice.

In all our tests, the MDF ordering method always resulted in good convergence behavior, even for anisotropic and inhomogeneous (rapidly varying equation coefficient) problems. Of course, the ability of MDF ordering to perform well for anisotropic, inhomogenous problems comes at a price. The time taken for determining the MDF ordering is much larger than the ordering cost for RCM. Consequently, we believe that the major application of MDF ordering will be in the solution of time-dependent or non-linear problems. In these situations, a sequence of matrix problems must be solved, where the matrix elements are only slightly changed from one time-step to the next. An ordering determined from one of these matrices can be used for the sequence. The ordering cost can then be amortized over the cost of many solves. Applications of this idea to reservoir simulation and Navier-Stokes equations are discussed in [6, 10]

If a single solution is required for a two-dimensional problem which is isotropic, then a reduced system RCM method would be a good choice. On the other hand, if several similar anisotropic problems are being solved, then it is worthwhile to use a reduced system MDF ordering. For three-dimensional problems, MDF(0) would appear to be a good choice.

We are currently developing approximate MDF ordering methods that are less expensive to compute, and hence can be applied to problems with a large node connectivity, which is typical of discretized systems of partial differential equations.

REFERENCES

[1] O. Axelsson and I. Gustafsson, *On the use of preconditioned conjugate gradient methods for red-black ordered five-point difference schemes*, J. Comp. Phys., 35 (1980), pp. 284–299.

[2] A. Behie, *Comparison of nested factorization, constrained pressure residual and incomplete factorization preconditionings*, in Proceedings of the 1985 Reservoir Simulation Symposium, Dallas, 1985, pp. 387–396. Paper SPE 13531.

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

[4] ———, *Incomplete factorization methods for fully implicit simulation of enhanced oil recovery*, SIAM J. Sci. Statist. Comp., 5 (1984), pp. 543–561.

[5] G. Brussion and V. Sonnad, *A comparison of direct and preconditioned iterative techniques for sparse, unsymmetric systems of linear equations*, Int. J. Num. Meth. Eng., 28 (1989), pp. 801–815.

[6] P. Chin, E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Preconditioned conjugate gradient methods for incompressible Navier-Stokes equations*, Tech. Report CS-91-04, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1991. submitted to International Journal on Numerical Methods in Fluids.

[7] P. Concus, G. H. Golub, and G. Meurant, *Block preconditioning for the conjugate gradient method*, SIAM J. Sci. Statist. Comp., 6 (1985), pp. 543–561.

[8] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings 24th National Conference of the Association for Computing Machinery, New Jersey, 1969, Brandon Press, pp. 157–172.

[9] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Towards a cost-effective high order ILU preconditioner*, Research Report CS-90-50, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 1990.

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

[11] S. Doi, *A Gustafsson-type modification for parallel ordered incomplete LU factorization*, in Advances in Numerical Methods for Large Sparse Sets of Linear Systems, T. Nodera, ed., Keio University, 1991, ch. 7, pp. 36–42.

[12] ———, *On parallelism and convergence of incomplete LU factorizations*, Appl. Numer. Math., 7 (1991), pp. 417–436.

[13] S. Doi and A. Lichnewsky, *A graph-theory approach for analyzing the effects of ordering on ILU preconditioning*, Tech. Report No. 1452, INRIA, June 1991.

[14] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*, Oxford University Press, London, 1986.

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

[16] V. Eijkhout, *Vectorizable and parallelizable preconditioners for the conjugate gradient method*, PhD thesis, University of Nijmegen, Nijmegen, the Netherlands, 1990.

[17] S. C. Eisenstat, H. C. Elman, and M. H. Schultz, *Block-preconditioned conjugate-gradient-like methods for numerical reservoir simulation*, Soc. Pet. Eng. J. Res. Eng., 3 (1988), pp. 307–312.

[18] P. A. Forsyth, *A control volume finite element approach to NAPL groundwater contamination*, SIAM J. Sci. Statist. Comp., 12 (1991), pp. 1029–1057.

[19] P. A. Forsyth and F. W. Letniowski, *A control volume finite element method for three dimensional NAPL groundwater contamination*, Int. J. Num. Meth. Fluids, 13 (1991), pp. 955–970.

[20] P. A. Forsyth and P. H. Sammon, *Local mesh refinement and modelling of faults and pinchouts*, SPE. J. Form. Eval., 1 (1986), pp. 275–285.

[21] G. Forsythe and W. Wasow, *Finite difference methods for partial differential equations*, Wiley, New York, 1960.

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

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

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

[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 Delaunay triangulations for finite element approximations to a second order diffusion operator*, SIAM J. Sci. Statist. Comp., to appear.

[27] J. W.-H. Liu and A. H. Sherman, *Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices*, SIAM J. Num. Anal., 13 (1975), pp. 198–213.

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

[29] E. J. Northrop and P. T. Woo, *Application of preconditioned conjugate gradient methods in reservoir simulation*, Soc. Pet. Eng. J. Res. Eng., 3 (1988), pp. 295–301.

[30] D. P. O'Leary, *Solving sparse matrix problems on parallel computers*, Tech. Report Report 1234, Computer Science Center, University of Maryland, Maryland, 1982.

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

[32] D. W. Peaceman, *Fundamentals of numerical reservoir simulation*, Elseview, New York, 1977.

[33] D. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, 1972, pp. 183–217.

[34] R. Schreiber and W.-P. Tang, *Vectorizing the conjugate gradient method*, in Proceedings of Symposium on CYBER 205 Applications, 1982.

[35] H. D. Simon, *Incomplete LU preconditioners for conjugate-gradient-type iterative methods*, Soc. Pet. Eng. J. Res. Eng., 3 (1988), pp. 302–306.

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

[37] A. D. Tuff and A. Jennings, *An iterative method for large systems of linear structural equations*, Int. J. Num. Meth. in Engng., 7 (1973), pp. 175–183.

[38] J. R. Wallis, *Incomplete Gaussian elimination as a preconditioning for generalized conjugate gradient acceleration*, in Proceedings of the 1983 SPE Symposium on Reservoir Simulation in San Francisco, 1983. paper SPE 12265.

[39] J. W. Watts, *A conjugate gradient-truncated direct method for the iterative solution of the reservoir pressure equation*, Soc. Pet. Eng. J., 21 (1981), pp. 345–353.

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