Weighted Graph Based Ordering Techniques for Preconditioned Conjugate Gradient Methods

Simon S. Clift[1] and Wei-Pai Tang[2]

The Research Institute of Advanced Computer Science is operated by Universities Space Research Association, The American City Building, Suite 212, Columbia, MD 21044, (410) 730-2656

Work reported herein was sponsored by NASA under contract NAS 2-13721 between NASA and the Universities Space Research Association (USRA) and by the Natural Sciences and Engineering Research Council of Canada, and by the Information Technology Research Center, which is funded by the Province of Ontario.

The first author's current address is Dept. of Mechanical Engineering, Queen's University, Kingston, Ontario. The second author is on sabbatical leave from University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

Abstract

We describe the basis of a matrix ordering heuristic for improving the incomplete factorization used in preconditioned conjugate gradient techniques applied to anisotropic PDE's. Several new matrix ordering techniques, derived from well-known algorithms in combinatorial graph theory, which attempt to implement this heuristic, are described. These ordering techniques are tested against a number of matrices arising from linear anisotropic PDE's, and compared with other matrix ordering techniques. A variation of RCM is shown to generally improve the quality of incomplete factorization preconditioners.

Keywords: Preconditioned conjugate gradient, preconditioner, matrix ordering, weighted graph

Running Title: Weighted Graph Ordering for PCG Methods.

AMS Subject Classification: 65F10

1 Introduction

Preconditioned conjugate gradient (PCG) methods have been proven to be robust and competitive techniques for the solution of matrices arising from PDE's in a number of applications [11, 4, 17, 2, 14, 5, 20, 33, 31]. The successful application of PCG methods depends to a great extent on the formation of a rapidly convergent preconditioner. A number of studies have examined the effect of matrix ordering on the quality of preconditioners based on incomplete factorization [7, 8, 9, 13, 14, 26, 12]. In [7, 8, 9] evidence was presented to demonstrate how matrix ordering can have a profound effect on the quality of preconditioners. A heuristic was described that was shown to produce a good matrix ordering. This study examines the use of efficient algorithms from combinatorial graph theory which implement that matrix ordering heuristic.

By way of background, we refer the reader to [34, 8, 21] for an outline of level based, incomplete, L/U factorization (denoted ILU(l), where l is the level of fill retained in the preconditioner). We will be referring to matrices as weighted graphs, where the matrix rows represent vertices, and the graph edges are encoded in the off-diagonals, the magnitude of the off-diagonal coefficients providing the "strength" of the connections. The reader may wish to review [27, 29, 18, 7] for relevant information on this view of matrices.

Duff and Meurant [13] studied a large number of preconditioner orderings for matrices arising from isotropic and anisotropic PDE's discretized on a regular grid. Their study considered orderings based solely on the sparsity pattern of the matrix, and concluded that Reverse Cuthill-McKee (RCM) ordering [18] was, in general, a good choice. This ordering reduces the bandwidth of the matrix, which tends to increase the overlap of fill and hence reduce the effect of dropped terms in ILU factorization. Dutto [14] also considered sparsity pattern based orderings, this time on Jacobian matrices arising from the discrete Navier-Stokes equations on irregular grids. Her results coincided with those of Duff and Meurant, indicating that RCM ordering, or the related Gibbs ordering [19] were good choices.

Recently, D'Azevedo, Forsyth, and Tang derived the Minimum Discarded Fill (MDF) and Minimum Update Matrix (MUM) orderings [7, 8, 9], which are sensitive to the matrix coefficients, as well as the sparsity pattern. The development of these orderings was prompted in part by the problem of highly

anisotropic PDE's, whose discretization can lead to matrices for which the wrong ordering will produce very unsatisfactory preconditioners. The analysis leading to these techniques revealed that the most effective ordering for an anisotropic matrix follows the direction of the weakest connections in the graph.

MDF ordering is capable of detecting anisotropy in a matrix graph, and exploiting it to produce exceptionally good orderings. Its one drawback is that it is expensive to compute; the algorithm has a time complexity of roughly $\mathcal{O}(Nd^3)$, where N is the number of matrix rows, and d is the average number of non-zeros in a matrix row. MUM ordering, which is an approximation of MDF, does not detect anisotropy when the fill level l is small, but has been shown to produce workable orderings even for difficult matrices. Unfortunately, it is also fairly expensive to compute. In [4, 3] MUM was tested against the Navier-Stokes equations, and in [8, 26] both MUM and MDF were tested against problems arising from linear PDE's with moderately extreme (1000:1) anisotropy.

The objective of this study is to implement and test the heuristic of following weak connections in the matrix graph to produce an ordering. This is done using algorithms which are considerably faster than MDF or MUM. The mathematical motivation for this heuristic will be outlined in more detail in Section 2.

Ordering techniques will be outlined which are sensitive to both the matrix sparsity patterns and matrix coefficients. These attempt to follow anisotropy and hence improve level-based ILU factorizations. A modification to the RCM algorithm will be considered. The orderings tested are based on the standard graph theoretical algorithms for a minimum spanning tree (MST), and the single-source problem (SSP), on the matrix graph. The main attraction of the MST and SSP algorithms as anisotropy detectors is their speed: they have a time complexity of $\mathcal{O}(M \log(N))$ or better, (where N is the number of matrix rows, and M the number of off-diagonal non-zeros in either the upper or lower half of the matrix). The new ordering techniques are described in detail in Section 3

A number of matrices taken from the discretization of linear PDE's will serve as test cases. These test cases are outlined in Section 4. The numerical results of matrix solver runs on these matrices using each ordering will be presented in Section 5, and summarized in the final section.

2 The Motivation for Weighted Graph Based Techniques

Consider the anisotropic PDE

$$\frac{\partial}{\partial x}\left(K\,\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\partial U}{\partial y}\right)=-q(x,y),\quad (x,y)\in(0,1)\times(0,1) \qquad (1)$$

with a Neumann boundary condition, $K = 1000$, and discretized on a 30×30 regular grid with a five-point molecule with $h = 1/30$ as the grid size. The right hand side $q(x,y)$ was defined as

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

The resulting linear system (which is similar to cases arising from highly anisotropic convection-diffusion problems) was solved with the preconditioned conjugate gradients method using an ILU(1) preconditioner. A zero initial guess was used, and the matrix was solved to a reduction of 10^{-12} in the l_2 norm of the residual. Table 1 shows the solution time when the matrix was ordered in two ways: natural x-y ordering, which numbered the nodes in the x direction first, and natural y-x ordering, which numbered the nodes in the y direction first. Theorem 1 will show why the incomplete factorization in the x-y direction was poorer, despite both preconditioners having the same level of fill, and number of fill entries.

If a given matrix A is symmetric, the fill entries in the factor L can be conveniently described through a graph model [27, 29]. Let the elimination sequence be $v_1,\ldots,v_n$ and $\mathcal{G}_k=(\mathcal{V}_k,\mathcal{E}_k)$ be the graph of $A_k=\left[a_{ij}^{(k)}\right]$

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

where k is the step of the elimination.

It can be shown [18] that there is a nonzero entry l_{ij} if and only if there exists a path $(v_i, v_{i_1},\ldots,v_{i_m},v_j)$ in the graph of A where

$$v_i,\ldots,v_{i_m}\in\{v_1,\ldots,v_{j-1}\}\ .$$

The size of l_{ij} is related to the size of entries on this path.

Theorem 1 *Let A be an M-matrix and let $(v_i, v_{i_1}, \ldots, v_{i_m}, v_j)$ be a path in the graph A where*

$$v_i, \ldots, v_{i_m} \in \{v_1, \ldots, v_{j-1}\} ,$$

then for $i > j$

$$l_{ij} \geq \frac{|a_{ii_1} a_{i_1 i_2} \cdots a_{i_m j}|}{d_{i_1} d_{i_2} \cdots d_{i_m} d_j}, \quad d_k = a_{kk} .$$

Proof: *See [25] and [8].*

For the anisotropic problem 1, the resulting matrix is a symmetric M-matrix. All edges aligned along the x-axis have values $\mathcal{O}(K/K+1)$, and edges aligned along the y-axis have values $\mathcal{O}(1/K)$. If the natural x-y row order is used, then all new fill entries will be oriented more in the x direction (see Figure 1). From the lower bound in Theorem 1, the fill entries in the matrix will have a slow decay rate. Conversely, if the natural y-x ordering is used, the fill entries will have a more rapid decay rate. Thus the value of the fill using the y-x ordering will have less of a bearing on the quality of the preconditioner as the level of fill increases than the fill using the x-y ordering.

In this study MDF ordering [7] and MUM ordering [8, 9] will be used as examples of effective, matrix coefficient sensitive orderings. Both orderings attempt to minimize the amount of fill discarded by the incomplete factorization process. MDF uses a more accurate, and more expensive measure for the discarded fill, and is, as noted in the introduction, capable of detecting anisotropy. The reader is referred to the papers cited for the details about MDF and MUM, and earlier comparative studies using these algorithms.

Reverse Cuthill-McKee ordering [18] will be used as a generally effective [13, 14] matrix coefficient insensitive ordering which is quick to compute. Because we will be making a modification to this algorithm to render it coefficient sensitive, we briefly outline the algorithm in Figure 2.

3 Weighted Graph Algorithms

As noted above, the following techniques are all matrix coefficient sensitive, and based on well established algorithms. The following algorithms seek out the weak connections in the matrix graph, and attempt to produce an ordering consistent with the heuristic that follows from Theorem 1.

All of the techniques described below operate on the graph $\mathcal{G}$ of the symmetric matrix A, where each node in the graph corresponds to a diagonal entry in the matrix. Weights are assigned to the edges between two nodes i and j by the absolute value of the matrix off-diagonal $\mid A_{ij} \mid$.

A detailed description of minimum spanning tree and single source problem algorithms may be found in [32], and many data structures textbooks. All of the ordering algorithms described below have an overall time complexity of $\mathcal{O}(M \log N)$ or better.

3.1 Modified RCM

RCM, as it stands in Figure 2, can miss the anisotropy of a problem and produce a poor ordering (see Section 5, also [4]). Step 3b of the algorithm as given was modified so that un-numbered neighbors of a node were sorted by the weight of their connection strengths whenever their degree was equal. Since the ordering is reversed in the final stage, it was not clear whether an ascending, or descending connection strength order was appropriate, so both were tried. We will denote the modified RCM with degree tie breaking in ascending order as $\mathrm{RCM_A}$, and in descending order as $\mathrm{RCM_D}$.

3.2 Minimum Spanning Tree with Three Versions

Three ordering methods based on the minimum spanning tree (MST) of the matrix graph were tested. The MST on the matrix graph creates an acyclic subgraph using the smallest possible edge weights, and hence will select edges that connect nodes weakly. Consistent with the heuristic outlined in Section 2, these weak connections will be followed to attempt to produce a good ordering.

Figure 3 outlines the MST-based algorithms. The first two steps are the same for all three variations. To construct the MST, we used Kruskal's algorithm [32]. It was selected for its simplicity; faster, $\mathcal{O}(M \log \log N)$ algorithms are available (where M is the number of graph edges).

In the first variation on the algorithm, a root node is selected on the tree, and a depth-first search performed. The nodes are numbered in the order in which they are encountered in the tree, always choosing the weakest connections first at branching points. The depth-first search will thus tend to follow lines of weakly connected nodes, producing an ordering that follows the anisotropy in the desired

direction, and which keeps nodes grouped locally in a reasonably natural fashion. This ordering will be denoted MST_D.

The second variation on the algorithm is a pre-processing step done before the MST is constructed to break ties in the depth-first search stage of MST_D. A small ($\ll$ off-diagonals) value is multiplied by the original node number, and (symmetrically) added to the row above (i.e. also to the column below) the diagonal entry corresponding to that node. This forces a slight bias in the matrix so that when the MST algorithm must decide between edges that were previously equally weighted, an edge will be selected that reflects the original ordering. A depth-first search is then performed on the tree. We will see that tie-breaking by natural ordering proves useful in situations of mixed anisotropy and isotropy. This variation of MST ordering will be denoted $\text{MST}_\text{D}^\text{T}$.

The final variation on MST-based ordering computes the distance from the given root node to every node on the MST. The nodes are then sorted in order of this distance. This effectively produces level sets, much as with RCM [18], only biased in the desired direction of the anisotropy by the removal of strong edge connections. The Cuthill-McKee (CM) ordering (which RCM reverses) produces level sets that might also be thought of as nodes grouped by contours on the graph. This variation distorts the contours, using the MST as a measure of distance in the graph. This variation will be denoted MST_C.

3.3 Single Source Problem with Contouring

MST_C ordering uses an inexact measure of distance from a given root node to the rest of the graph. The solution to the single source problem on a weighted graph produces the exact minimum weighted distance from a root node to all nodes in the graph. For anisotropic problems, nodes with weak connections to the root will appear "closer" to it than those with strong connections. With the single source problem solved, this ordering, as with MST_C, sorts the nodes in order of distance from the root, again producing distorted level sets, or contour sets which follow the weakly connected direction of the anisotropy. This ordering will be denoted SSP.

This ordering may also be seen as a Breadth-First Search traversal of the graph, using the weighted distance from the root node to determine the depth of each search level.

The reader will note that given a graph with all equal weights, this will pro-

duce roughly Cuthill-McKee ordering, the difference being that SSP ordering does not order based on node degree within a level set. SSP ordering is, by the contour set analogy, a "skewed" variation on CM ordering. This in turn suggests that, as with RCM, reversing the SSP ordering might be beneficial. Thus we will also consider Reversed SSP, or RSSP ordering.

4 Test Problems

All of the ordering methods mentioned were tested against fourteen problems, eleven on a regular grid and three on simplicial grids. Each test case is different in some aspect of dimension, isotropy or anisotropy, direction of anisotropy, varying coefficients, and grid type. The problems are all based on the PDE

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial U}{\partial y}\right) = -q(x,y), \quad (x,y) \in (0,1) \times (0,1) \tag{2}$$

in two dimensions, and

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial U}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial U}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial U}{\partial z}\right) = -q(x,y,z), \tag{3}$$

$$(x,y,z) \in (0,1) \times (0,1) \times (0,1)$$

in three dimensions. The usual five- or seven-point finite difference discretization was used on the regular grid problems, with an harmonic average used to deal with cases where the coefficients (K_x, K_y or K_z) were discontinuous [1]. Despite the fact that a number of the problems produced only positive semi-definite matrices, the PCG method still converged.

For the various regular grid problems the same number of points were used in each of the x, and y, (and z in three dimensions) directions. We define a discrete (x_i, y_j, z_k) coordinate system for a grid with node spacing $h = 1/(N_{\text{edge}} - 1)$, where N_{edge} is the number of nodes along the edge of the unit square or cube, so that

$$x_i = (i-1)h,\ y_j = (j-1)h,\ z_k = (k-1)h, \quad 1 \le i,j,k \le N_{\text{edge}}\ .$$

This will be used in the definition of the regular grid problems. The term $q_d(x_i, y_j, z_k)$ denotes the discretized right hand side, and is zero unless otherwise noted. The unit square or cube region will be denoted $\mathbf{R}$, with boundary $\delta\mathbf{R}$, as required.

The three problems on irregular grids are too complicated to fully describe here. The reader is referred to [16] for a more detailed definition of the REFINE2D and FE2D problems, and [22] for the FE3D problem.

We surmised, based on the results of MDF(0) ordering (see Figures 5 through 7), that grouping regions of similar permitivity on regular grid problems was the correct thing to do. The magnitude of matrix row off-diagonals, not merely their difference, might be useful to ensure that those differently weighted blocks were ordered together. For the irregular grid problems, where the weighting of the edges varies gradually, such a particular approach to grouping was not evident. Preliminary testing revealed that regular grid matrices were indeed more effectively ordered if they were not scaled, hence preserving the magnitude information for blocks on the grid. The matrices from irregular grid problems were more quickly solved if they were first symmetrically scaled so that their diagonal entries were equal to one. Thus all the following testing will be done on the unscaled regular grid, and scaled irregular grid matrices.

4.1 Problem 1. LAP5D

This first problem is the two dimensional Laplace's equation on the unit square with Neumann boundary conditions and five point sources, discretized on a regular 30×30 grid. It is similar to that used in [13, 8].

4.2 Problem 2. BIG1DIR

This problem is similar to that presented as Equation 1 earlier in the paper. A single, fairly strong anisotropy defines the problem.

Parameters: 2 Dimensions, Regular Grid, Neumann BC's

$$\begin{aligned} N_{\text{edge}} &= 30 \\ (K_x, K_y) &= (1000, 1) \end{aligned}$$

Source terms $q_d(x_i, y_j)$ defined as in LAP5D.

4.3 Problem 3. VDVORST

Tested in [26], this problem from [10] also exhibits fairly extreme anisotropy over the entire region, but with variations in the coefficient strengths.

Parameters: 2 Dimensions, Regular Grid, Dirichlet BC's

$$N_{\text{edge}} = 41$$

$$(K_x, K_y) = \begin{cases} (100, 0.01) & \text{in } (0.25, 0.75) \times (0.25, 0.75) \\ (1, 0.0001) & \text{elsewhere} \end{cases}$$

$$q(x, y) = \begin{cases} 100 & \text{in } (1/4, 3/4) \times (1/4, 3/4) \\ 0 & \text{elsewhere} \end{cases}$$

$$U = 0 \text{ on } \delta\mathbf{R}$$

4.4 Problem 4. STONE

Stone's third problem [30] presents a large isotropic region, with inset isotropic and anisotropic blocks of different orientations and magnitudes. See [30, 8] for the exact specification of this problem, which is two dimensional, and on a 31×31 regular grid.

4.5 Problem 5. ANISO

The region in this problem is completely anisotropic, but with four distinct regions in two directions. The ratio of K_x to K_y was 100:1 in each region. It is defined on a two dimensional, 30×30 regular grid. For exact specifications, see [8].

4.6 Problem 6. LAP7D

This problem is the three dimensional Laplace's equation on the unit cube with Neumann boundary conditions and five point sources, discretized on a $30 \times 30 \times 30$ regular grid.

4.7 Problem 7 & 8. BIG1DIR3E and BIG1DIR3F

These two problems have uniform coefficients that give them strong directions of anisotropy. Two directions were used hoping to differentiate between the methods that are incapable of detecting anisotropy.

Parameters: 3 Dimensions, Regular Grid, Neumann BC's

$$N_{\text{edge}} = 30$$

$$(K_x, K_y, K_z) = \begin{cases} (100, 1, 1000) & \text{for BIG1DIR3E} \\ (1000, 100, 1) & \text{for BIG1DIR3F} \end{cases}$$

Source terms $q_d(x_i, y_j, z_k)$ defined as in LAP7D.

4.8 Problem 9. STONE3D

This is a three dimensional version of the 2D STONE problem. It was formed by projecting the blocks defined by STONE into three dimensions using the 2D pattern for the z ranges.

Parameters: 3 Dimensions, Regular Grid, Neumann BC's

$$N_{\text{edge}} = 31$$

$$(K_x, K_y, K_z) = \begin{cases} (1, 100, 100) & \text{for } (x_i, y_j, z_k)\ 15 \le i \le 31,\ 1 \le j \le 17,\ 1 \le k \le 17 \\ (100, 1, 1) & \text{for } (x_i, y_j, z_k)\ 6 \le i \le 13,\ 6 \le j \le 13,\ 6 \le k \le 13 \\ (0, 0, 0) & \text{for } (x_i, y_j, z_k)\ 13 \le i \le 20,\ 22 \le j \le 29,\ 22 \le k \le 29 \\ (1, 1, 1) & (x_i, y_i, z_k) \text{ elsewhere} \end{cases}$$

$$q_d(4, 4, 4) = 1, \quad q_d(4, 28, 28) = 0.5, \quad q_d(24, 5, 5) = 0.6$$

$$q_d(15, 16, 16) = -1.83, \quad q_d(28, 28, 28) = -0.27$$

4.9 Problem 10 & 11. ANISO3E and ANISO3F

These are three dimensional versions of the ANISO problem. Six blocks are defined so that abutting regions have different strong directions of anisotropy. Again, two variations are defined, ANISO3E showing only one distinct direction in each block.

Parameters: 3 Dimensions, Regular Grid, Neumann BC's

$$N_{\text{edge}} = 30$$

$$(K_x, K_y, K_z) = \begin{cases} (100, 1, K_v) & \text{in } (0, 1/2) \times (0, 1/2) \times (0, 1/2) \\ & \text{and } (1/2, 1) \times (1/2, 1) \times (0, 1/2) \\ (K_v, 1, 100) & \text{in } (1/2, 1) \times (0, 1/2) \times (1/2, 1) \\ & \text{and } (0, 1/2) \times (1/2, 1) \times (1/2, 1) \\ (100, K_v, 1) & \text{elsewhere} \end{cases}$$

$$K_v = \begin{cases} 100 & \text{for ANISO3E} \\ 1000 & \text{for ANISO3F} \end{cases}$$

Source terms $q_d(x_i, y_j, z_k)$ defined as in LAP7D.

4.10 Problem 12. 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. Grid refinement was applied, and the final triangulation was such that the resulting matrix was an M-matrix.

4.11 Problem 13. FE2D

A finite element method using linear triangular basis functions was also used for this problem. However, K_x and K_y varied by four orders of magnitude. The grid was defined by constructing a distorted quadrilateral grid, and then triangulating in an obvious manner. A Delaunay-type edge swap was used to produce an M-matrix.

4.12 Problem 14. FE3D

This problem is a three-dimensional version of FE2D. A finite element discretization was used, with linear basis functions defined on tetrahedra. The coefficients (K_x, K_y, K_z) vary eight orders of magnitude (this model was derived from actual field data from a groundwater flow experiment). 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 [23].

5 Numerical Results

The numerical experiments were run on a Sun 4/670 server (nominally rated at 4 MFLOPS) using double precision arithmetic. The convergence criterion

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

was used, where r^k was the residual vector after the k^{th} iteration of conjugate gradient acceleration. In all cases the initial guess was chosen to be the zero vector.

A note concerning CPU times is in order. The mechanism provided in Sun FORTRAN for computing CPU times tends to be out as much as 10% between runs of the same test, and provides an accuracy of only 0.01 of a CPU second. The reader should keep this margin of error in mind while interpreting the following results. All results are for CPU time required only for the program execution, and are reported in seconds.

5.1 Ordering Time

Table 2 lists the time required to perform the various orderings for a few of the problems. The time to perform RCM ordering is given, and the other ordering times are scaled by that value for each problem. On average, performing RCM ordering accounted for between 2 and 4% of the overall solution time. The time required for MDF varies considerably depending on the fill level requested for the calculation, so MDF for level 1 fill is given.

Note from these results that the graph based orderings take roughly 1 to 4 times longer to produce than RCM, which is considerably less than the 10 to 21 times longer for MUM, and 28 to 191 times longer for MDF. Note that RCM_A and RCM_D take, on average, the same amount of time to compute, so the numbers for RCM_D are not shown.

5.2 Solution Time

One test run was made for each of the ordering methods, on each of the fourteen test matrices, for preconditioners using ILU(0), ILU(1) and ILU(2). Of particular interest in analyzing the results are the number of iterations required to solve the problem to the desired tolerance, the amount of fill produced in the ILU factorization, and the total time required for the iterative solve. Since the last measurement is a function of the first two, only the iterative solve times will be presented in detail.

Natural (default node order) ordering, was generally worse than RCM in 2D, and on irregular grids, producing marginally better solution times on 3D regular grids. Results with MUM were mixed, however, MUM generally does best on

problems with large computational molecules [8, 9], and the problems posed here have relatively small molecules. MDF proved again to be a good ordering in most cases.

The weighted graph based orderings performed best at ILU(1). Table 3 shows the time required for the all the iterations required to solve the linear system, using each of the ordering methods, at this level of fill. Because RCM is popular, and often viewed as the best matrix coefficient insensitive ordering [13, 14], we have scaled all the solution times by the value for RCM ordering for that matrix, giving only the solution time for RCM ordering in CPU seconds. At ILU(0), weighted graph methods generally performed as well as RCM on regular grid problems, and significantly worse on irregular grids. At ILU(2), the MST and SSP based orderings fared somewhat poorly.

At ILU(1), the MST based algorithms ran significantly faster than RCM in all two-dimensional, regular grid cases where the regions were completely anisotropic. Tie breaking (MST_D^T) was required to produce good results in the two 3D problems where one direction of anisotropy was dominant. Tie breaking also was required to produce adequate results for STONE, which had mixed anisotropic and isotropic regions, but the result was still slower than RCM.

Table 3 shows that SSP and RSSP ordering produced significant improvements over RCM only when the region was anisotropic in a single direction. Unlike the MST algorithms, the SSP orderings could not follow the sharp changes in anisotropic direction of the ANISO problem (see next subsection).

The RCM_A ordering showed favourable results. At worst it caused a 30% slower solution over RCM, and that in only one case at ILU(1). It generally produced solutions taking 47% to 108% the time of basic RCM, and did better on irregular grids, and regions that were all anisotropic in a single direction. Comparing the BIG1DIR3E and BIG1DIR3F problems, RCM_A produced a solution in the same time for both, whereas basic RCM failed to follow the direction of anisotropy of the latter case, and produced a poor ordering. At ILU(0), and ILU(2), RCM_A was never worse than 12% slower than RCM, and slightly faster in almost half the cases. RCM_D fared somewhat poorly, and was never significantly faster than RCM_A, hence its timing results are not shown.

We note in passing that, in the preliminary testing for this study, different root node placements were tried for the MST and SSP orderings. This was shown to have little effect on the solution time.

5.3 Drop Tolerance Preconditioning

Also noteworthy is the application of these algorithms to drop tolerance preconditioning [6], where matrix fill is discarded based on its magnitude. With drop tolerance, a good ordering will speed computation by causing the magnitude of the fill to decay more quickly, leading to fewer fill entries and faster preconditioner matrix multiplies.

Tests were performed dropping preconditioner fill entries $|a_{ij}| < \epsilon \left|\text{RowMax}^i\right|$ where $|\ \text{RowMax}^i\ |$ is the maximum magnitude entry in row i, and ϵ is the drop tolerance parameter. Solutions with $\epsilon = 0.01$ produced good results for well conditioned problems, and $\epsilon = 0.001$ was required for more difficults ones. MST_C ordering, and SSP ordering produced the best results at these levels, reducing solution time (compared to RCM) by an average of 26%. Weighted RCM orderings generally did not outperform plain RCM.

Drop tolerance methods can be defined in many different ways (our matrix solver package defines eight), thus it is a little more difficult to analyze the outcome of the tests. The exact number of fill entries is not determined by the graph of the problem, adding another level of complexity to the analysis. Previous experience [6] has show that the parameters tend to be problem dependant and the results hard to generalize. Thus we feel that a detailed presentation of these results is not justified.

5.4 Some Ordering Pictures

Using MATLAB [24], a number of pictures of graph orderings are given in Figures 5, 6 and 7. In this visualization technique, nodes numbered first appear as darker squares, and the nodes numbered last appear as the lightest squares.

Figure 5 shows the orderings produced by MDF(0), MST_D, and SSP on the VDVORST problem. Notice how MDF(0) picks up the interior region of differing coefficients. The other orderings pick up only on the single direction of weak connections, producing essentially the same pictures.

Figure 6 shows the MDF(0), MST_D, and SSP orderings on the ANISO problem. Note the similarity between the MDF(0) and MST orderings, both detected and followed the changes in the direction of anisotropy. Note the SSP ordering tended to order the middle and work outward, missing the basic anisotropy. (Note that the two anomalously ordered blocks were caused by an unavoidable

tie-breaking problem inherent in the binary heap used to compute the Single Source Problem.)

Figure 7 shows the MDF(0), MST_D, and MST_D^T orderings on the STONE problem. While MDF identifies and orders the various blocks in the test, the MST routines fail to identify them clearly. Thus we see a weakness in the MST routines when isotropic and anisotropic regions exist in the same problem. However, if the subregions of anisotropy are known in advance, MST ordering could potentially be applied to those individual subregions, then the results linked to produce a final ordering. MDF(0) is essentially doing this for the STONE problem.

6 Summary

We have presented and tested seven new matrix ordering techniques which are sensitive to the coefficients, as well as the sparsity pattern, of matrices. These techniques attempt to implement the heuristic, based on Theorem 1, that ordering along lines of weakly connected nodes results in an improved incomplete factorization for PCG methods.

MDF ordering again proved to be very good, however, it is expensive to compute. It would be the most useful if a large number of similar matrix problems were to be solved which could efficiently exploit a single MDF ordering computation.

Methods based on the minimum spanning tree and single source problem only showed significant advantage over RCM on two-dimensional regions that were entirely anisotropic, with one level of fill in the preconditioner. The single source problem based techniques were unable to follow changes in the direction of anisotropy, and hence were only advantageous when no changes in direction were present.

The modified RCM technique RCM_A proved to be generally better than RCM, and rarely significantly worse. RCM_A performed consistently well on irregular grids, and for all levels of fill. We conclude that RCM_A is, in general, a good choice over plain RCM.

A number of questions surrounding good matrix ordering for ILU preconditioners remain to be solved. Good results have been obtained for systems of equations using block ordering [15, 4], but more work needs to be done in this area. Also, investigations into ordering based on eigenvalue computations, called

spectral ordering (and closely related to [28]), are being undertaken.

References

[1] G. A. Behie and P. A. Forsyth. Comparison of fast iterative methods for symmetric systems. *IMA Journal of Numerical Analysis*, 3:41–63, 1983.

[2] M. Buffat. Simulation of two- and three-dimensional internal subsonic flows using a finite-element method. *International Journal for Numerical Methods in Fluids*, 12:683–704, 1991.

[3] P. Chin, E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang. Preconditioned conjugate gradient methods for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 15:273–295, 1992.

[4] Simon S. Clift and Peter A. Forsyth. Linear and non-linear methods for the incompressible Navier-Stokes equations. Technical Report CS-93-02, University of Waterloo, Waterloo, Ontario, January 1993. Submitted to Int. J. Num. Methods in Fluids.

[5] O. Dahl and S. O. Wille. An ILU preconditioner with coupled node fill-in for iterative solution of the mixed finite element formulation of the 2d and 3d Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 15:525–544, 1992.

[6] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang. Drop tolerance preconditioning for incompressible viscous flow. *Intern. J. Computer Math.*, 44:301–312, 1992.

[7] E. F. D'Azevedo, P. A. Forsyth, and W. P. Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM J. Matrix Anal. Appl.*, 13(3):944–961, July 1992.

[8] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang. Towards a cost-effective ILU preconditioner with high level fill. *BIT*, 32:442–463, 1992.

[9] E. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang. Two variants of minimum discarded fill ordering. In R. Beauwens and P. de Groen, editors, *Proc. IMACS Intern. Symp. on Iterative Methods in Linear Algebra*, pages 603–612. North-Holland, 1992.

[10] H. Van der Vorst. The convergence behaviour of preconditioned CG and CG-S. In O. Axelsson and L. Kolotilina, editors, *Preconditioned Conjugate Gradient Methods, Lecture Notes in Mathematics No. 1547*, pages 126–136. Springer Verlag, 1990.

[11] John K. Dickinson and Peter A. Forsyth. Preconditioned conjugate gradient methods for three dimensional linear elasticity. Technical Report CS-93-13, Department of Computer Science, University of Waterloo, February 1993.

[12] Shun Doi. A Gustafsson-type modification for parallel ordered incomplete factorizations. Technical report, C&C Information Technology Research Labs, NEC Corp.,, 1991.

[13] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT: Nordisk Tijdskrift for Informationbehandling*, 29:635–657, 1989.

[14] Laura C. Dutto. The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations. Research report, Université de Montréal, CRM, February 1992. to appear in International Journal for Numerical Methods in Engineering.

[15] Q. Fan, P. A. Forsyth, and W.-P. Tang. Performance issues for iterative solvers in semiconductor device simulation. Technical report, University of Waterloo, 1993. In preparation.

[16] P. A. Forsyth. A control volume finite element approach to NAPL groundwater contamination. *SIAM J. Sci. Stat. Comput.*, 12(5):1029–1057, 1991.

[17] P. A. Forsyth and B. Y. Shao. Numerical simulation of gas venting for NAPL site remediation. *Adv. Water Resources*, 14(6), 1991.

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

[19] N. E. Gibbs, W. G. Poole Jr., and P. K. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Analysis*, 13:236–250, 1976.

[20] D. Howard, W. M. Connolley, and J. S. Rollett. Unsymmetric conjugate gradient methods and sparse direct methods in finite element flow simulation. *International Journal for Numerical Methods in Fluids*, 10:925–945, 1990.

[21] H. P. Langtangen. Conjugate gradient methods and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns. *International Journal for Numerical Methods in Fluids*, 9:213–233, 1989.

[22] F. W. Letniowski and P. A. Forsyth. A control volume finite element method for three-dimensional NAPL groundwater contamination. *International Journal for Numerical Methods in Fluids*, 13:955–970, 1991.

[23] Frank W. Letniowski. Three-dimensional Delaunay triangulations for finite element approximations to a second-order diffusion operator. *SIAM J. Sci. Stat. Comput.*, 13(3):765–770, May 1992.

[24] The Math Works Inc., Cochituate Place, 24 Prime Park Way, Natick, Mass. *MATLAB 4.0 Users Manual*, 1992.

[25] 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*, 31:148–162, 1977.

[26] Yvan Notay. Ordering methods for approximate factorization preconditioning. Technical report, Service de Métrologie Nucléaire, Université Libre de Bruxelles, January 1993.

[27] S. V. Parter. The use of linear graphs in gaussian elimination. *SIAM Review*, 3:364–369, 1961.

[28] Alex Pothen, Horst D. Simon, and Lie Wang. Spectral nested dissection. Technical Report 92-01, Computer Science Dept., Pennsylvania State University, January 1992.

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

[30] H. L. Stone. Iterative solution of implicit approximations of multidimensional partial differential equations. *SIAM J. Numer. Analysis*, 5:530–558, 1968.

[31] J. Strigberger, G. Baruzzi, and W. Habashi. Some special purpose preconditioners for conjugate gradient- like methods applied to CFD. *International Journal for Numerical Methods in Fluids*, 16:581–596, 1993.

[32] Robert E. Tarjan. *Data structures and network algorithms.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[33] C. Vuik. Solution of the discretized incompressible Navier-Stokes equations with the GMRES method. *International Journal for Numerical Methods in Fluids*, 16:507–523, 1993.

[34] 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. SPE 12265.

Figure 1: Orientation of new fill in x-y and y-x natural orderings. New fill is indicated by dotted lines.

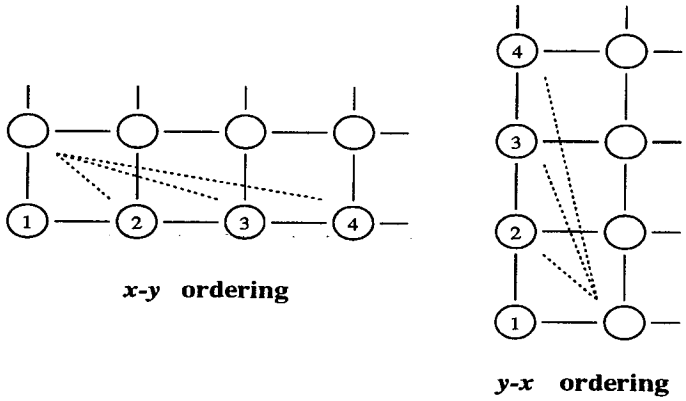

Figure 2: The Reverse Cuthill-McKee (RCM) ordering algorithm

1. DETERMINE A STARTING PSEUDO-PERIPHERAL NODE $\mathcal{R}$ OF GRAPH $\mathcal{G}$.
2. NUMBER $\mathcal{R}$ FIRST IN THE ORDERING.

3. FOR $i = 1 \ldots$ (NUMBER OF NODES) DO (FOLLOWING NODE ORDERING)
   3A. $\mathcal{U} = \{$ ALL UN-NUMBERED NEIGHBORS OF NODE $i\}$
   3B. NUMBER ELEMENTS OF $\mathcal{U}$ IN ORDER OF NODE DEGREE
   ENDDO

4. REVERSE THE ORDERING DETERMINED IN STAGE 3.

Table 1: Solution time for an anisotropic problem with two orderings.

Ordering	Solution Time (s)	Iterations
Natural x-y	1.13	35
Natural y-x	0.43	7

Figure 3: The Minimum Spanning Tree based ordering algorithms

1. CONSTRUCT THE MINIMUM WEIGHT SPANNING TREE $\mathcal{T}$ OF $\mathcal{G}$.
2. SELECT A ROOT NODE $\mathcal{R}$ OF GRAPH $\mathcal{G}$.

Variant 1: $\mathrm{MST_D}$
3. PERFORM A DEPTH-FIRST SEARCH (PRE-ORDER TRAVERSAL) OF $\mathcal{T}$ STARTING FROM $\mathcal{R}$. AT EACH BRANCH SELECT THE MINIMUM WEIGHT EDGE FIRST.
4. NUMBER THE NODES IN THE ORDER THEY WERE ENCOUNTERED IN STEP 3.

Variant 2: $\mathrm{MST_D^T}$
3. USING THE ORIGINAL ORDERING, ADD SOME FACTOR ϵ TIMES THE NODE NUMBER TO THE WEIGHTS OF EDGES CONNECTING NODE i WITH NODE j WHERE $i < j$ IN THE ORIGINAL ORDER. ($\epsilon \ll$ THE OFF DIAGONAL WEIGHTS).
4. PERFORM A DEPTH-FIRST SEARCH (PRE-ORDER TRAVERSAL) OF $\mathcal{T}$ STARTING FROM $\mathcal{R}$. AT EACH BRANCH SELECT THE MINIMUM WEIGHT EDGE FIRST.
5. NUMBER THE NODES IN THE ORDER THEY WERE ENCOUNTERED IN STEP 4.

Variant 3: $\mathrm{MST_C}$
3. COMPUTE THE (WEIGHTED) DISTANCE FROM $\mathcal{R}$ TO EACH NODE OF $\mathcal{T}$.
4. ORDER THE NODES STARTING WITH $\mathcal{R}$, THEN IN ORDER OF SHORTEST TO LONGEST DISTANCE FROM $\mathcal{R}$ ALONG $\mathcal{T}$.

Figure 4: The Single-Source Problem based ordering algorithm

1. SELECT A ROOT NODE $\mathcal{R}$ OF GRAPH $\mathcal{G}$.
2. COMPUTE THE SINGLE-SOURCE PROBLEM FROM $\mathcal{R}$ FOR THE ENTIRE GRAPH $\mathcal{G}$. THIS ASSOCIATES THE MINIMUM WEIGHTED DISTANCE FROM $\mathcal{R}$ TO EACH NODE IN THE GRAPH $\mathcal{G}$.
4. ORDER THE NODES STARTING WITH $\mathcal{R}$, THEN IN ORDER OF SHORTEST TO LONGEST DISTANCE FROM $\mathcal{R}$ ON $\mathcal{G}$.

Table 2: Time Required to Perform Ordering

	Ordering Method									
	(CPU s)	Scaled by RCM								
Problem	RCM	MST_D	MST_D^T	MST_C	SSP	RSSP	RCM_A	RCM_D	MDF	MUM
LAPD5	0.03	1.7	3.3	3.0	1.3	2.0	2.0	1.7	32.7	21.3
ANISO	0.03	2.0	4.0	3.7	2.0	2.0	2.3	2.7	31.0	15.7
LAPD7	1.85	2.2	2.6	2.7	1.5	1.6	1.5	1.4	70.4	10.5
STONE3D	2.13	2.2	3.3	3.1	1.5	1.6	1.4	1.4	69.5	9.8
FE2D	0.09	3.3	3.2	3.9	1.6	1.6	1.2	1.1	27.8	10.7
FE3D	0.53	2.8	3.1	2.9	1.0	1.0	1.0	1.0	191.3	12.1

Table 3: Time Required to Perform All PCG Solver Iterations (at ILU(1))

	Ordering Method									
	(CPU s)	Scaled by RCM								
Problem	RCM	MST_D	MST_D^T	MST_C	SSP	RSSP	RCM_A	Nat	MDF	MUM
LAPD5	0.71	1.01	1.01	1.00	1.08	1.06	0.99	1.06	0.92	1.07
BIG1DIR	0.73	0.92	0.45	0.51	0.47	0.44	0.47	1.07	0.41	0.97
VDVORST	0.76	0.54	0.57	0.62	0.53	0.55	0.59	1.01	0.33	1.01
STONE	0.77	1.86	1.17	2.23	1.29	1.26	1.03	1.04	0.84	1.21
ANISO	0.71	0.66	0.68	1.24	1.66	1.20	1.08	1.10	0.80	1.42
LAPD7	62.81	0.90	1.05	1.10	1.07	1.05	1.07	0.92	0.89	1.01
BIG1DIR3E	131.95	2.06	0.89	0.90	0.88	0.86	0.97	0.97	0.84	1.64
BIG1DIR3F	231.29	1.20	0.58	0.58	0.57	0.61	0.55	0.87	0.49	0.85
ANISO3E	90.40	0.97	1.03	1.15	1.16	1.09	0.98	0.93	0.80	0.91
ANISO3F	104.54	1.32	1.29	1.49	1.67	1.41	1.30	0.83	0.81	1.30
STONE3D	114.47	1.00	1.16	1.27	1.02	1.05	1.06	0.92	0.74	0.87
REFINE2D	1.54	2.39	2.64	2.58	2.01	1.45	0.97	3.42	1.04	1.29
FE2D	1.48	3.32	2.63	2.70	2.24	1.82	0.94	1.95	0.89	1.47
FE3D	7.63	1.15	1.23	1.01	0.92	0.72	0.77	1.09	0.88	0.73

Note: This table lists the time to complete all PCG iterations required to solve the problem to $\mathsf{tol} = 10^{-12}$, where tol is the l_2 norm of the linear solution residual. RCM times are given in CPU seconds. Other ordering times are scaled by the time required to solve the problem using RCM ordering.

Figure 5: The VDVORST problem ordered with MDF(0), MST_D and SSP. Node ordering is from darkest to lightest.

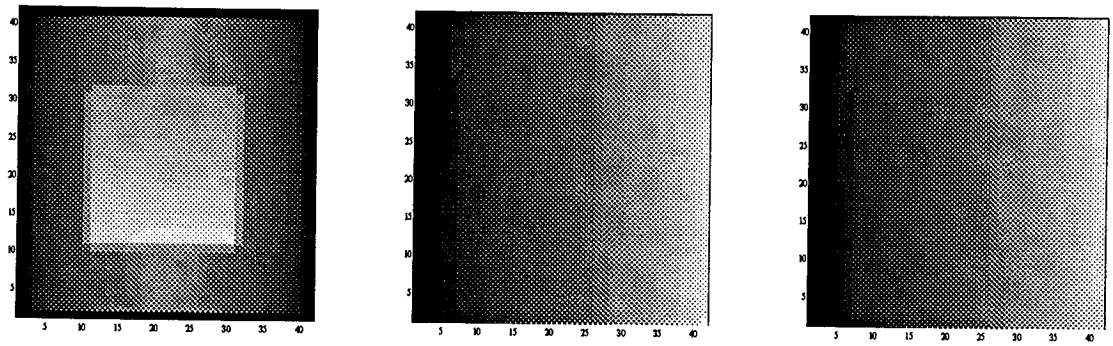

Figure 6: The ANISO problem ordered with MDF(0), MST_D and SSP. Node ordering is from darkest to lightest.

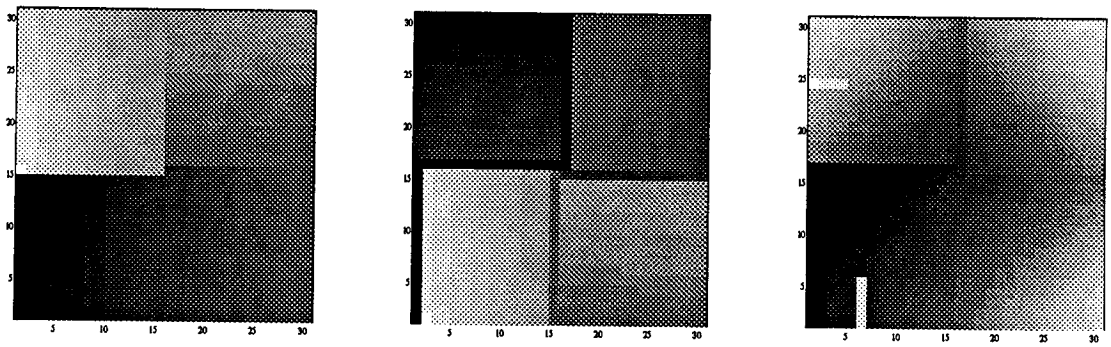

Figure 7: The STONE problem ordered with MDF(0), MST_D, and MST_D^T orderings. Node ordering is from darkest to lightest.