SIAM J. NUMER. ANAL.
Vol. 13, No. 2, April 1976

COMPARATIVE ANALYSIS OF THE CUTHILL–McKEE AND THE REVERSE CUTHILL–McKEE ORDERING ALGORITHMS FOR SPARSE MATRICES*

WAI-HUNG LIU† AND ANDREW H. SHERMAN‡

Abstract. In this paper we examine the Cuthill–McKee algorithm for ordering the unknowns and equations in systems of linear equations, $A\mathbf{x}=\mathbf{b}$, where A is sparse, symmetric, and positive definite. This algorithm is designed to produce a permutation matrix P such that PAP^T has a small bandwidth. If we wish to exploit zeros in the band of A which occur before the first nonzero in each row and column, it has been experimentally observed that reversing the ordering produced by the Cuthill–McKee algorithm is often very much better than the original ordering in terms of the amount of storage and work required to factor A. We prove that for band elimination methods, the two orderings are equivalent and that, surprisingly, the reverse ordering is always at least as good as the original one when envelope elimination techniques are used. We give a condition on the matrix A under which the reverse ordering is strictly better than the original one, and we include several numerical experiments and analyses of practical examples to illustrate our results.

1. Introduction.

Consider the linear algebraic system

$$A\mathbf{x}=\mathbf{b}, \tag{1.1}$$

where A is an $N\times N$ symmetric, sparse, positive definite matrix. To solve (1.1), we factor A into the product LDL^T, where L is unit lower triangular and D is a positive diagonal matrix. We then sequentially solve $L\mathbf{z}=\mathbf{b}$, $D\mathbf{y}=\mathbf{z}$ and $L^T\mathbf{x}=\mathbf{y}$. Direct methods such as this fall into two basic classes: the general sparse methods, which exploit all the zeros in L, and the band or envelope methods, which exploit only those zeros outside a particular set of matrix positions in L. In this paper we consider mainly methods in the latter group, particularly with respect to the effect on such methods of various ordering strategies for the variables and equations of the system (1.1).

Let P be any permutation matrix. In the permuted linear system

$$PAP^T(P\mathbf{x})=P\mathbf{b}, \tag{1.2}$$

PAP^T remains sparse, symmetric, and positive definite. It is well known that PAP^T possesses a triangular factorization LDL^T and that the factorization is numerically stable [10], so that we may solve (1.2) rather than (1.1). The possible advantage of solving the permuted system is that the solution might require a smaller number of arithmetic operations and/or storage locations. For the methods with which we deal in this paper, this means that the band or envelope of PAP^T may be smaller than the corresponding structure of A.

* Received by the editors May 28, 1974, and in revised form, July 8, 1975.

† Department of Applied Analysis and Computer Science, University of Waterloo, Waterloo, Ontario, Canada. This author's work supported in part by University of Waterloo Graduate Bursary funds.

‡ Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801. This author's work was performed while the author was a student in the Department of Computer Science at Yale University and was supported in part by the Office of Naval Research under Grant N0014-67-0097-0016.

A popular ordering strategy for band solution of (1.1) is the Cuthill–McKee algorithm [2]. It tries to find a permutation matrix P for which PAP^T has a small bandwidth. The reverse of the Cuthill–McKee ordering has been observed to produce a permutation matrix P for which PAP^T has a small envelope [3], [1]. In this paper we give a detailed analysis of these two ordering strategies for use with band and envelope methods. In particular, we will show that the reverse Cuthill–McKee algorithm (RCM) always produces an ordering at least as good as that produced with the Cuthill–McKee algorithm (CM), and we will give estimates of the improvement with RCM for several classes of problems arising in the application of the finite element method.

2. Definitions. Let M be an N by N symmetric or lower triangular matrix with elements M_{ij}. For the ith row of M, $i = 1, 2, \cdots, N$, we define

$$f_i(M) \equiv \min\{j : M_{ij} \neq 0\}, \tag{2.1}$$

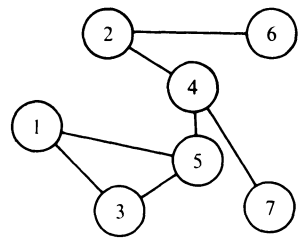
$$\beta_i(M) \equiv i - f_i(M), \tag{2.2}$$

$$\omega_i(M) \equiv |\{k : k > i \text{ and } \exists l \leqq i \text{ such that } M_{kl} \neq 0\}|;^1 \tag{2.3}$$

that is, $f_i(M)$ is the column subscript of the first nonzero element of the ith row of M, $\beta_i(M)$ is the "bandwidth" of the ith row of M, and $\omega_i(M)$ is the "frontwidth" of

$$A = \begin{bmatrix} X & 0 & X & 0 & X & 0 & 0 \\ 0 & X & 0 & X & 0 & X & 0 \\ X & 0 & X & 0 & X & 0 & 0 \\ 0 & X & 0 & X & X & 0 & X \\ X & 0 & X & X & X & 0 & 0 \\ 0 & X & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & X & 0 & 0 & X \end{bmatrix}$$

$G(A)$:



i	$f_i(A)$	$\beta_i(A)$	$\omega_i(A)$
1	1	0	2
2	2	0	4
3	1	2	3
4	2	2	3
5	1	4	2
6	2	4	1
7	4	3	0

$b(A) = 4$, $|\text{Env}(A)| = 22$, $|\text{Tenv}(A)| = 20$

FIG. 2.1

the ith row of M (see Fig. 2.1). Following Cuthill and McKee [2], we define the bandwidth of M by

$$b(M) \equiv \max\{|i - j| : M_{ij} \neq 0\} \tag{2.4}$$

[1] $|X|$ denotes the number of elements or size of X.

and note that

$$b(M) = \max \{\beta_i(M)\}. \tag{2.5}$$

The envelope and transpose envelope structures of M are now defined by

$$\text{Env}\,(M) \equiv \{(i, j) : f_i(M) \leqq j \leqq i\}, \tag{2.6}$$

$$\text{Tenv}\,(M) \equiv \{(i, j) : j \leqq i \text{ and } \exists k \geqq i \text{ such that } M_{kj} \neq 0\}, \tag{2.7}$$

From this it follows that

$$\text{Tenv}\,(M) = \text{Env}\,(M^{\#}), \tag{2.8}$$

where $M^{\#}$ is the transpose of M about its minor diagonal.

The size of the envelope of M may be expressed in terms of the $\beta_i(M)$ or the $\omega_i(M)$:

$$|\text{Env}\,(M)| = N + \sum_{i=1}^{N} \beta_i(M) = N + \sum_{i=1}^{N} \omega_i(M). \tag{2.9}$$

Now let A be a symmetric matrix having the symmetric factorization LDL^T. We define

$$\text{Fill}\,(A) \equiv \{(i, j) : A_{ij} = 0 \text{ and } L_{ij} \neq 0\}. \tag{2.10}$$

Since $\text{Fill}\,(A) \subseteq \text{Env}\,(A)$ (see [5]), the number of locations required to store all of the nonzero entries of L is always less than or equal to $|\text{Env}\,(A)|$. Envelope methods do not exploit zeros within the envelope of A, so that the storage for L will be exactly $|\text{Env}\,(A)|$.

The following lemma on *general* sparse symmetric decompositions which exploit all of the zeros in L is given by George [4] and follows directly from results of Rose [8].

LEMMA 2.1. *If the symmetric factorization of the matrix A into the product LDL^T requires $\theta(A)$ multiplicative operations,*[2]*, then*

$$\theta(A) \geqq \frac{1}{2} \sum_{i=1}^{N} \omega_i(A)[\omega_i(A) + 3], \tag{2.11}$$

with equality exactly when L has a full envelope.

In order to facilitate what follows, we introduce the undirected graph $G(M) = (X(M), E(M))$ associated with the symmetric or lower triangular matrix M. Here $X(M) = \{x_1, x_2, \cdots, x_N\}$ is the set of vertices corresponding to and labeled as the rows of M, and $E(M)$ is the set of edges, where $\{x_i, x_j\} \in E(M)$ if and only if $M_{ij} \neq 0$ or $M_{ji} \neq 0$. The adjacency and the degree of a vertex $x_i \in X(M)$ are then defined by

$$\deg(x_i) \equiv |\text{adj}\,(x_i)| \equiv |\{x_j : \{x_i, x_j\} \in E(M)\}|. \tag{2.12}$$

For any permutation matrix P, the graphs $G(M)$ and $G(PMP^T)$ are structurally identical, but the node labels in $G(PMP^T)$ have been permuted according to P.

[2] Throughout this paper, "multiplications" or "multiplicative operations" will include both multiplications and divisions.

We now restate several of our definitions in terms of the graph $G(M)$:

(2.13) $f_i(M) \equiv \min \{j : x_j \in \text{adj}\,(x_i)\}$;

(2.14) $\text{Env}\,(M) \equiv \{\{x_i, x_j\} : f_i(M) \leqq j \leqq i\}$;

(2.15) $\text{Tenv}\,(M) \equiv \{\{x_i, x_j\} : j \leqq i \text{ and } \exists k \geqq i \text{ such that } x_k \in \text{adj}\,(x_j)\}$.

Finally, if A is a symmetric matrix having the symmetric factorization LDL^T, then

(2.16) $\text{Fill}\,(A) \equiv \{\{x_i, x_j\} : \{x_i, x_j\} \notin E(A) \text{ and } \{x_i, x_j\} \in E(L)\}$.

3. The Cuthill–McKee algorithm. The ordering algorithm proposed by Cuthill and McKee [2] is most easily described in terms of labeling the graph structure $G(A)$ associated with the $N \times N$ symmetric matrix A. Although our results and the Cuthill–McKee algorithm apply to graphs with any number of connected components, we shall henceforth assume that $G(A)$ is connected, or equivalently, that the matrix A is irreducible. In the following algorithm, Q, R and S are first-in-first-out data structures known as queues. Q is the input queue which initially contains the vertices of $G(A)$ in an arbitrary order, and R and S are working queues. The starting vertex for the labeling is assumed to be the first vertex in Q. The output of the algorithm is a permutation of the vertices of $G(A)$ stored in the vector $x \equiv (x_1, x_2, \cdots, x_N)$. Such a permutation effectively labels vertex v with label i if $x_i = v$.

The operation of adding an element to a queue (denoted by $Q \leftarrow Q, x_i$) is always performed by placing the element at the end or back of the queue. The operation of removing the first element of the queue (denoted by $x_i \leftarrow \text{Top}\,(Q)$) assigns the first or oldest element of the queue to x_i and deletes that element from the queue. Whenever a set of vertices is assigned to a queue, the vertices are assumed to be ordered by increasing vertex degree in $G(A)$. The empty set is denoted by Λ.

Step 0. $i \leftarrow 1$; $S \leftarrow \Lambda$;

$x_1 \leftarrow \text{Top}\,(Q)$; $R \leftarrow \text{adj}\,(x_1) \cap Q$; $Q \leftarrow Q - R$;

Step 1. **while** $R \neq \Lambda$ **do** ($i \leftarrow i+1$; $x_i \leftarrow \text{Top}\,(R)$; $S \leftarrow S, x_i$);

if $Q = \Lambda$ **then stop**;

$z \leftarrow \text{Top}\,(S)$; $R \leftarrow \text{adj}\,(z) \cap Q$; $Q \leftarrow Q - R$; **go to** *Step* 1.

As noted in [2], the Cuthill–McKee (CM) ordering algorithm corresponds to the generation of a spanning tree for the graph $G(A)$ in a breadth-first or level-by-level fashion. Such a spanning tree is formed by adding each new vertex as far to the top and left as possible.

We denote by $G(A_c)$ the labeled graph obtained by applying the Cuthill–McKee algorithm to $G(A)$, and we use A_c to refer to the matrix obtained by reordering the rows and columns of A to correspond with $G(A_c)$.

LEMMA 3.1. *A_c satisfies the* monotone envelope property *that if $i \leqq j$, then $f_i \leqq f_j$. Furthermore, for $i > 1$, $f_i < i$.*

Proof. Let $X=\{x_1, x_2, \cdots, x_N\}$ be the ordered vertex set of $G(A_c)$. Then the first part of the lemma is equivalent to the condition that if x_i precedes x_j in X, then

$$I \equiv \min\{l : x_l \in \text{adj}(x_i)\} \leqq J \equiv \min\{l : x_l \in \text{adj}(x_j)\}.$$

But this follows directly from the breadth-first nature of the Cuthill–McKee algorithm, for if $J<I$, then the algorithm would have labeled x_j before x_i, contradicting the assumption that x_i precedes x_j.

The second part of the lemma follows immediately from the first, for if there were a row with $f_i = i$, then the monotone envelope property would imply that A_c is reducible, contradicting the assumption made earlier.

The following theorem is a restatement, in our notation, of a result of George and Liu [5]. By Lemma 3.1, both it and its corollary apply to A_c.

THEOREM 3.2. *Let $A = LDL^T$ with $f_i(A) < i$ for $i > 1$. Then the envelope of L is full (i.e., $(i, j) \in \text{Env}(L) \Rightarrow L_{ij} \neq 0$).*

COROLLARY 3.3. *Let $A = LDL^T$ with $f_i(A) < i$ for $i > 1$. Then the number of nonzero entries in L is exactly $|\text{Env}(A)|$, and the forward and backward solving (i.e., the solution of $LDL^T\mathbf{x} = \mathbf{b}$, given L and D) can be done in $2 \cdot |\text{Env}(A)| - N$ multiplications.*

We now turn to the problem of estimating the number of multiplications required for the symmetric factorization of an envelope oriented $N \times N$ matrix A. A is assumed to satisfy the monotone envelope property and the hypothesis that $f_i(A) < i$ for $i > 1$. Therefore, by Lemma 3.1, the results apply directly to the matrix A_c.

The factorization process can be defined by the following equations for $i = 1, 2, \cdots, N$.

$$\text{(3.1a)} \qquad a'_{ij} = a_{ij} - \sum_{k=\max(f_i, f_j)}^{j-1} a'_{ik} l_{jk}, \qquad j = f_i, f_i+1, \cdots, i-1,$$

$$\text{(3.1b)} \qquad l_{ij} = a'_{ij}/d_{jj}, \qquad j = f_i, f_i+1, \cdots, i-1,$$

$$\text{(3.1c)} \qquad d_{ii} = a_{ii} - \sum_{k=f_i}^{i-1} a'_{ik} l_{ik}.$$

Equations (3.1) are simply an envelope oriented variant form of the conventional defining equations for the Choleski decomposition for symmetric matrices [7] which exploits zeros to the left of the first nonzero in each row of A. Since A is assumed to be a monotone envelope matrix, (3.1a) can be rewritten in a simpler form as:

$$\text{(3.1a}') \qquad a'_{ij} = a_{ij} - \sum_{k=f_i}^{j-1} a'_{ik} l_{jk}, \qquad j = f_i, f_i+1, \cdots, i-1.$$

Recalling that $\beta_i = i - f_i$, we have the following theorem.

THEOREM 3.4. *Let A be a symmetric monotone envelope matrix satisfying $f_i(A) < i$ for $i > 1$. Then the symmetric factorization of A requires*

$$\theta(A) = \frac{1}{2} \sum_{i=2}^{N} \beta_i(\beta_i + 3)$$

multiplications.

Proof. From (3.1a′) and Theorem 3.2, we observe that the a'_{ij} are all nonzero at positions within the envelope of A. Thus the products $a'_{ik}l_{jk}$ and $a'_{ik}l_{ik}$ appearing in (3.1a′) and (3.1c), respectively, involve nonzero operands. In performing the ith step defined by (3.1a′), (3.1b) and (3.1c), it follows that:

(a) for $j = f_i, f_i+1, \cdots, i-1$, the computation of a'_{ij} requires $j-f_i$ multiplications. Thus the total number of multiplications for (3.1a′) is

$$\sum_{j=f_i}^{i-1} j-f_i = \tfrac{1}{2}(i-f_i)(i-f_i-1);$$

(b) to compute l_{ij} for $j = f_i, f_{i+1}, \cdots, i-1$ requires $i-f_i$ multiplications;

(c) to compute d_{ii} requires $i-f_i$ multiplications.

Hence for the ith step of the factorization, the total number of multiplications required is

$$\tfrac{1}{2}(i-f_i)(i-f_i+3) = \tfrac{1}{2}\beta_i(\beta_i+3).$$

Summing on i, then, the entire factorization process requires

$$\theta(A) = \frac{1}{2}\sum_{i=2}^{N} \beta_i(\beta_i+3)$$

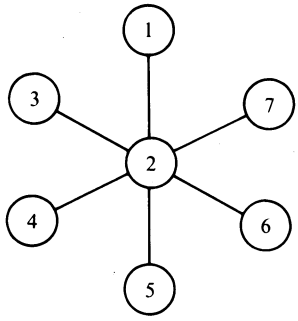
multiplications, since $\beta_1 = 0$.

It should be noted that the operation count given in Theorem 3.4 does not really depend on the particular variant of the envelope oriented Choleski decomposition described by (3.1). Also, for a general sparse matrix, the operation count is just an upper bound for its sparse symmetric factorization. Only for irreducible monotone envelope matrices whose triangular factors have a full envelope is the bound achieved exactly.

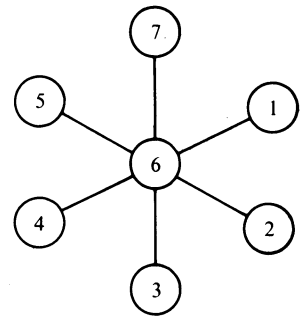
4. The reverse Cuthill–McKee ordering. In his study of envelope methods, George [3] considered the reverse Cuthill–McKee algorithm (RCM) which renumbers the CM ordering in the reverse way. Surprisingly, this simple modification often yields an ordering superior to the original ordering in terms of efficiency, although the bandwidth remains unchanged. Experimental evidence of the superior performance of the reversed ordering for matrices arising from the finite element method has been reported in the thesis of George [3] and in the survey paper by Cuthill [1]. In this section we shall prove that the reverse scheme is *always* at least as good, as far as storage and operation counts are concerned.

Much of what we will develop in the remainder of this paper may be most easily seen by examining several examples. Therefore, we now introduce three graphs, and their associated systems of equations (1.1), which will be used to illustrate our results. The simplest of these is the star graph with N vertices. For any ordering of the vertices of such a graph, we obtain an $N \times N$ system (1.1) in which $A_{ij} \neq 0$ if and only if the nodes x_i and x_j are adjacent in the graph. Figure 4.1 illustrates a star graph with seven nodes ordered by both the CM and RCM algorithms.

Our other two examples arise in the use of finite difference or finite element methods for the solution of partial differential equations. Consider the mesh graph obtained by subdividing the unit square into n^2 square elements of side $1/n$



CM *ordering*

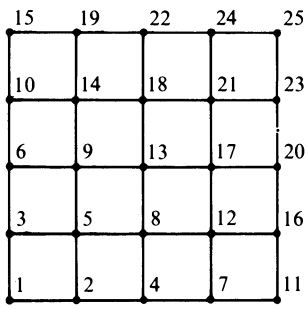


RCM *ordering*

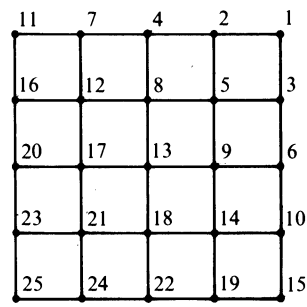
$$A_c = \begin{bmatrix} X & X & & & & & & \\ X & X & X & X & X & X & X & \\ & X & X & & & & & \\ & X & & X & & & & \\ & X & & & X & & & \\ & X & & & & X & & \\ & X & & & & & & X \end{bmatrix} \qquad A_r = \begin{bmatrix} X & & & & & X & & \\ & X & & & & X & & \\ & & X & & & X & & \\ & & & X & & X & & \\ & & & & X & X & & \\ X & X & X & X & X & X & X & \\ & & & & & X & X & \end{bmatrix}$$

FIG. 4.1. *Orderings for star graph with* $N = 7$ *nodes*

(see Fig. 4.2). Let $N = (n+1)^2$. We may derive an $N \times N$ system (1.1) from any ordering of the mesh points of the $n \times n$ square mesh by using one of several techniques. A five-point difference system is obtained by considering each mesh point to be adjacent to those vertices to which it is joined by a line in the mesh. Then the matrix A associated with such a system satisfies the condition that $A_{ij} \neq 0$ if and only if the nodes x_i and x_j are connected by a line in the mesh.



CM



RCM

FIG. 4.2. *Orderings for a five-point finite difference mesh*

Alternatively, we may derive a nine-point finite element system by considering any mesh point x_i to be adjacent to all those vertices which border on small square elements containing x_i. The matrix A associated with the system will then be such that $A_{ij} \neq 0$ if and only if the nodes x_i and x_j are both in the same small square element of the mesh. Figures 4.2 and 4.3, respectively, illustrate the CM and RCM orderings of the five- and nine-point square meshes.

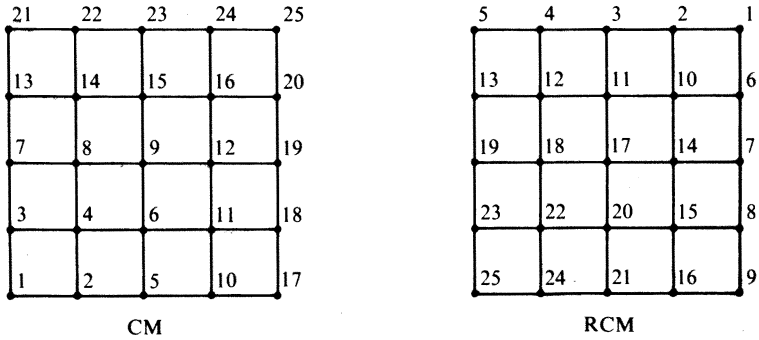


FIG. 4.3. *Orderings for a nine-point finite element mesh*

As before, we denote by A_c the matrix A reordered by the Cuthill–McKee algorithm. Let A_r be the matrix obtained by reversing the ordering of the rows and columns of A_c. Then

$$A_r = A_c^{\#} = \tilde{P} A_c \tilde{P}^T, \tag{4.1}$$

where

$$\tilde{P} = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \iddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix}.$$

The following lemma is immediate.

LEMMA 4.1. $\text{Env}(A_r) = \text{Tenv}(A_c)$.

It is helpful to consider the labeled undirected graphs $G(A_c) = (X(A_c), E(A_c))$ and $G(A_r) = (X(A_r), E(A_r))$ associated with the matrices A_c and A_r, respectively. Let $X(A_c) = \{x_1, x_2, \cdots, x_N\}$ and $X(A_r) = \{y_1, y_2, \cdots, y_N\}$, where x_i is the ith node in the Cuthill–McKee ordering and y_j is the jth node in the reverse ordering. From our definitions, it may be shown that $G(A_c)$ and $G(A_r)$ are identical structurally and that x_i and y_{N-i+1} represent the same node in the underlying unlabeled graph structure. We now have the following theorem and its corollaries.

THEOREM 4.2. $\text{Tenv}(A_c) \subseteq \text{Env}(A_c)$.

Proof. Assume that for some $i \geqq j$, $\{x_i, x_j\} \in \text{Tenv}(A_c)$ and $\{x_i, x_j\} \notin \text{Env}(A_c)$.

$$\{x_i, x_j\} \in \text{Tenv}(A_c) \Rightarrow \exists k \geqq i \quad \text{such that } x_k \in \text{adj}(x_j), \tag{4.2}$$

and

$$\{x_i, x_j\} \notin \text{Env}(A_c) \Rightarrow \forall l \leqq j, \qquad x_i \notin \text{adj}(x_l). \tag{4.3}$$

But (4.2) and (4.3) together imply that $f_k(A_c) < f_i(A_c)$, contradicting Lemma 3.1. This proves the theorem.

COROLLARY 4.3. *For all* i, $\omega_{N-i+1}(A_r) \leqq \beta_i(A_c)$.

Proof. If for some i, $\omega_{N-i+1}(A_r) > \beta_i(A_c)$, then there would be some $\{x_i, x_j\}$

such that $\{x_i, x_j\} \in \text{Tenv}(A_c) = \text{Env}(A_r)$ and $\{x_i, x_j\} \notin \text{Env}(A_c)$ both held. This would contradict Theorem 4.2.

COROLLARY 4.4. $|\text{Env}(A_r)| \leqq |\text{Env}(A_c)|$.

Proof. Theorem 4.2 and Lemma 4.1 together imply that $\text{Env}(A_r) \subseteq \text{Env}(A_c)$. The corollary follows immediately from this.

COROLLARY 4.5. $|\text{Fill}(A_r)| \leqq |\text{Fill}(A_c)|$.

Proof. Let $n(A)$ be the number of nonzeros in the lower triangle of the matrix A. From (2.10), Lemma 3.1 and Theorem 3.2, it follows that

$$|\text{Fill}(A_c)| = |\text{Env}(A_c)| - n(A).$$

On the other hand,

$$|\text{Fill}(A_r)| \leqq |\text{Env}(A_r)| - n(A),$$

so that applying Corollary 4.4 proves the result.

COROLLARY 4.6. $\theta(A_r) \leqq \theta(A_c)$.

Proof. Combining Lemmas 2.1 and 3.1, Theorem 3.4 and Corollary 4.3 yields

$$\begin{aligned}\theta(A_r) &\leqq \frac{1}{2}\sum_{k=1}^{N} \omega_k(A_r)[\omega_k(A_r)+3] \\ &\leqq \frac{1}{2}\sum_{k=1}^{N} \beta_{N-k+1}(A_c)[\beta_{N-k+1}(A_c)+3] \\ &= \frac{1}{2}\sum_{i=1}^{N} \beta_i(A_c)[\beta_i(A_c)+3] = \theta(A_c).\end{aligned}$$

5. Comparisons of CM and RCM. In the previous section we saw that the RCM ordering was always at least as good as the CM ordering for solving (1.1). In this section we give a theorem concerning the conditions under which the RCM ordering is actually strictly better than the CM ordering. We also present several examples and numerical experiments which illustrate the results of this paper. Finally, we give an analysis of certain special cases of the finite element method in order to demonstrate the savings which are actually achievable in practical problems.

For a graph $G(A_c)$ associated with the CM ordering, we will say that $G(A_c)$ has Property $\tilde{\text{P}}$ if there exist vertices x_i, x_j and x_k with $i<j<k$ such that $x_k \in \text{adj}(x_i)$ and for all $l \geqq k$, $x_l \notin \text{adj}(x_j)$. Then we have the following theorem.

THEOREM 5.1. $|\text{Env}(A_r)| < |\text{Env}(A_c)|$ *if and only if* $G(A_c)$ *has Property* $\tilde{\text{P}}$.

Proof. $G(A_c)$ has Property $\tilde{\text{P}}$ if and only if there exist $i<j<k$ such that $\{x_k, x_i\} \in \text{Env}(A_c)$, $\{x_k, x_j\} \in \text{Env}(A_c)$ and $\{x_k, x_j\} \notin \text{Tenv}(A_c)$. But this is equivalent to $|\text{Tenv}(A_c)| < |\text{Env}(A_c)|$, and the result follows by Lemma 4.1.

By examining Figure 4.2, we see that the five-point finite difference mesh cannot satisfy Property $\tilde{\text{P}}$. This suggests the following corollary.

COROLLARY 5.2. *For a rectangular five-point finite difference mesh,*

$$|\text{Env}(A_r)| = |\text{Env}(A_c)|.$$

In contrast, we can let $i=14$, $j=21$ and $k=23$ in the CM ordering of the nine-point finite element mesh shown in Figure 4.3. Then we see that the mesh graph has Property $\tilde{\text{P}}$. This may be generalized to prove the following result.

COROLLARY 5.3. *For a rectangular nine-point finite element mesh,*

$$|\text{Env}(A_r)| < |\text{Env}(A_c)|.$$

For both meshes, similar results hold for the storage requirements and operation counts for the solution of the associated system of equations.

Example 1. For our first example, we use the unlabeled star graph with N nodes. The best possible CM ordering is to start with one of the "planet" vertices, and the resulting CM and RCM orderings for $N=7$ are shown in Figure 4.1. It is clear that for the star graph with N nodes,

$$\frac{|\text{Env}(A_r)|}{|\text{Env}(A_c)|} = \frac{2N-1}{N(N-1)/2+2} \cong \frac{4}{N} \quad \text{and} \quad \frac{\theta(A_r)}{\theta(A_c)} \cong \frac{3}{N^2}$$

This example shows that the RCM ordering can be significantly better than the CM ordering.

Example 2. We now analyze some examples that arise in the application of the finite element method. As before, we consider the mesh obtained by subdividing a unit square into n^2 small square elements of side $1/n$ and the nine-point finite element system associated with it. We number the nodes starting at the lower left hand corner using the CM algorithm. The case for $n=4$ is given in Figure 4.3.

By inductive analysis for $n \geqq 2$,

(i) $$|\text{Env}(A_c)| = 1 + \sum_{k=2}^{n+1} (4k^2-3k-1) = \tfrac{4}{3}n^3 + \tfrac{9}{2}n^2 + \tfrac{19}{6}n + 1,$$

(ii) $$|\text{Env}(A_r)| = 2n^2+7n+1 + \sum_{k=2}^{n} (4k^2-3k-1) = \tfrac{4}{3}n^3 + \tfrac{5}{2}n^2 + \tfrac{31}{6}n + 1,$$

(iii) $$\theta(A_c) = \sum_{k=2}^{n+1} (4k^3-2k^2-5k+2) = n^4 + \tfrac{16}{3}n^3 + \tfrac{15}{2}n^2 + \tfrac{13}{6}n,$$

(iv) $$\theta(A_r) = \tfrac{4}{3}n^3 + 7n^2 + \tfrac{52}{6}n - 1 + \sum_{k=2}^{n} (4k^3-2k^2-5k+2)$$
$$= n^4 + \tfrac{8}{3}n^3 + \tfrac{9}{2}n^2 + \tfrac{47}{6}n.$$

The savings for different values of n are given in Table 5.1. It is of interest to note that for this example, the relative savings with RCM decrease with increasing n and that the well-known natural, or row-by-row, mesh ordering is better than either CM or RCM.

TABLE 5.1

Theoretical amount saved for the regular square mesh

Problem			Storage			Operation count		
n	$N=(n+1)^2$	Nonzeros	CM	RCM	RCM/CM	CM	RCM	RCM/CM
4	25	97	171	147	.860	726	530	.730
8	81	353	997	885	.888	7324	5812	.793
16	289	1345	6665	6185	.928	89336	77736	.870
32	1089	5249	48401	46417	.959	1231088	1140816	.927

Example 3. The savings in Example 2 are not particularly impressive. We now consider the same model problem with triangular elements (that is, each small square in Figure 4.3 is subdivided into two right triangles), and its corresponding linear system. The basic mesh is shown in Figure 5.1.

With quadratic elements, each triangle has three vertex unknowns and three edge unknowns. The corresponding CM ordering is shown in Fig. 5.2 for $n = 2$. The results given in Table 5.2 show that substantial savings in computation and storage can be achieved if the RCM rather than the CM ordering is used.

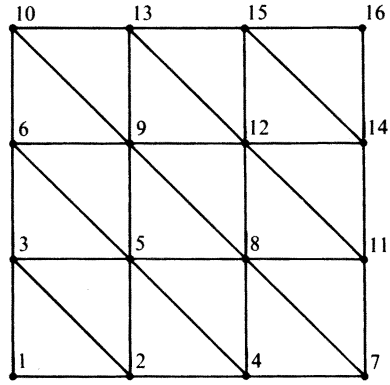


FIG. 5.1. 3×3 *regular right triangular mesh ordered by the* CM *algorithm*

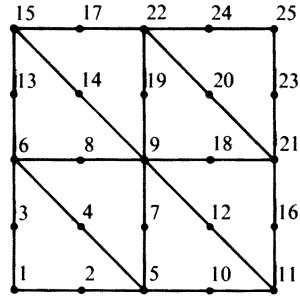


FIG. 5.2. 2×2 *regular right triangular mesh with element*[3] 2-6 *ordered by the* CM *algorithm*

TABLE 5.2
Experimental savings for the regular right triangular mesh with element 2-6

Problem			Storage			Operation count		
n	$N=(2n+1)^2$	Nonzeros	CM	RCM	RCM/CM	CM	RCM	RCM/CM
4	81	441	1078	755	.700	8758	4183	.478
5	121	676	1971	1310	.665	19214	8324	.433
6	169	961	3244	2077	.640	36808	14857	.404
7	225	1296	4961	3088	.622	64040	24506	.383
8	289	1681	7186	4375	.609	103914	38115	.367
9	361	2116	9983	5970	.598	159698	56600	.354

[3] We adopt the notation in [3], where the two parts of the hyphenated name refer respectively to the degree of the interpolating polynomial and to the number of nodes associated with the element.

In the case of cubic elements having ten nodal unknowns, the savings are even more dramatic. The CM ordering for $n = 2$ is shown in Fig. 5.3. The storage and work requirements for CM and RCM are given in Table 5.3.

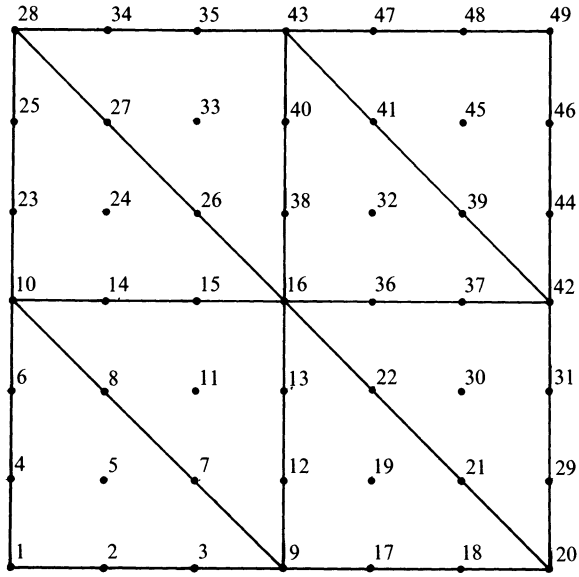


FIG. 5.3. 2×2 *regular right triangular mesh with element* 3-10 *ordered by* CM *algorithm*

TABLE 5.3

Experimental savings for the regular right triangular mesh with element 3-10

Problem			Storage			Operation count		
n	$N=(3n+1)^2$	Nonzeros	CM	RCM	RCM/CM	CM	RCM	RCM/CM
3	100	784	2002	1252	.625	25002	9429	.377
4	168	1368	4558	2518	.552	74528	22046	.296
5	256	2116	8626	4396	.510	172453	43624	.253
6	361	3025	14530	6994	.481	342564	77574	.226

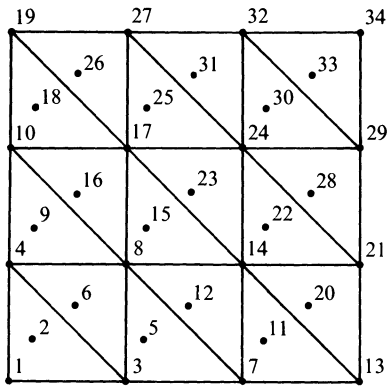


FIG. 5.4. 3×3 *regular right triangular mesh with* $p_0 = 1$ *interior nodes, ordered by* CM

Finally, we consider elements with three vertex unknowns and a variable number p_0 of interior unknowns. The case for $p_0 = 1$ is shown in Fig. 5.4. The results of experiments for $p_0 = 0$ and $p_0 = 1$ are shown in Tables 5.4 and 5.5.

TABLE 5.4
Experimental results for the regular right triangular mesh with element 1-3

Problem			Storage			Operation count		
n	$N=(n+1)^2$	Nonzeros	CM	RCM	RCM/CM	CM	RCM	RCM/CM
4	25	81	115	115	1.00	320	320	1.00
8	81	289	597	597	1.00	2616	2616	1.00
16	289	1089	3689	3689	1.00	27472	27472	1.00
32	1089	4225	25553	25553	1.00	344608	344608	1.00

TABLE 5.5
Experimental results for the regular right triangular mesh with $p_0 = 1$

Problem			Storage			Operation count		
n	$N=(n+1)^2+2n^2$	Nonzeros	CM	RCM	RCM/CM	CM	RCM	RCM/CM
4	57	209	529	323	.611	2975	1088	.366
8	209	801	3687	1781	.483	38037	8808	.232
16	801	3137	27139	11177	.412	527081	89200	.169
32	3137	12417	207099	77393	.374	7761201	1083232	.140

The examples in this section have shown that dramatic savings may be achieved by using RCM rather than CM. We will now give a more detailed analysis of the last example. The basic finite element mesh to be analyzed is shown in Fig. 5.1, and as above, we consider generalizations of the basic mesh with exactly p_0 interior nodes in each triangular element. These meshes are simplifications of the first two meshes presented in Example 3, so that the results for them are closely related to the experimental results given there. In fact, our investigations have led to the conclusion that the difference between the CM and RCM orderings is only increased by the addition of either interior or edge nodes to the regular right triangular mesh of Fig. 5.1. Hence the meshes that we will analyze give lower bounds on the difference between CM and RCM for meshes in actual use.

As mentioned in § 3, the CM algorithm generates a breadth-first spanning tree for the mesh graph. A sample spanning tree for the mesh in Fig. 5.4 is shown in Fig. 5.5. The analysis below will be based upon the characteristics of the generated spanning tree and the relationship between CM and RCM, which is formalized in the following lemma.

LEMMA 5.4. *Consider a spanning tree generated by the* CM *algorithm, numbered top-to-bottom, left-to-right (see Fig.* 5.5). *For any node* x_i *on level* l *of the tree,*

$$\beta_i(A_c) = i - f_i(A_c) = (\textit{the number of nodes to the left of } x_i \textit{ on level } l) + (\textit{the number of nodes to the right of the father of } x_i \textit{ on level } l-1) + 1.$$

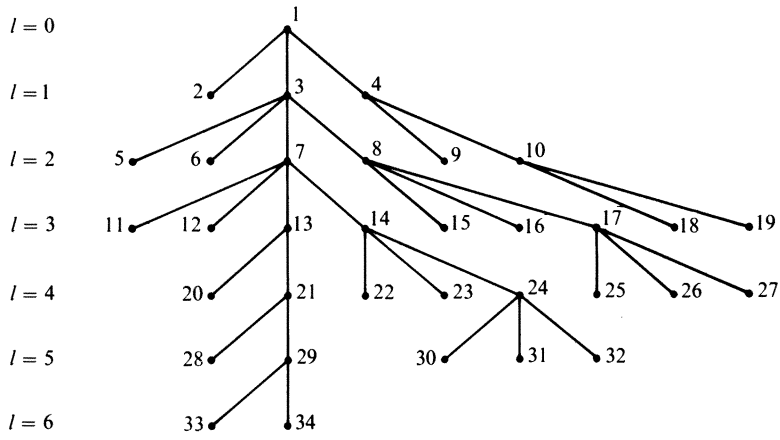


FIG. 5.5. *Spanning tree generated by the* CM *algorithm for a* 3×3 *right triangular mesh with* $p_0 = 1$

If there are N nodes in the tree, x_i *will be relabeled* x_{N-i+1} *in the* RCM *ordering. Then, if* x_i *has a son on level* $l+1$,

$$\beta_{N-i+1}(A_r) = N - i + 1 - f_{N-i+1}(A_r)$$

$$= (\textit{the number of nodes to the right of } x_i \textit{ on level } l) + (\textit{the number of nodes to the left of the rightmost son of } x_i \textit{ on level } l+1) + 1.$$

And if x_i *has no sons in the tree,*

$$\beta_{N-i+1}(A_r) = N - i + 1 - f_{N-i+1}(A_r)$$

$$= (\textit{the number of nodes strictly between } x_i \textit{ and the rightmost node on level } l \textit{ or } l+1 \textit{ which is a member of } \text{adj}\,(x_i)) + 1.$$

Proof. From Figures 5.5 and 5.6, it may be seen that for the CM ordering, $x_{f_i(A_c)}$ is the father of x_i in the spanning tree. Since there must be exactly $i - f_i(A_c) - 1$ nodes between $x_{f_i(A_c)}$ and x_i on levels $l-1$ and l, the result for CM is immediate. For the RCM ordering, $x_{f_{N-i+1}(A_r)}$ is the rightmost son of x_i, if one exists, so there are $N - i + 1 - f_{N-i+1}(A_r) - 1$ nodes between them on levels l and

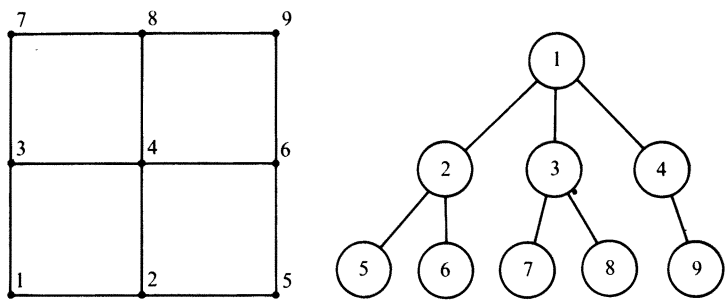


FIG. 5.6. *Spanning tree associated with the* CM *ordering for the* 2×2 *regular square mesh*

$l+1$. And if x_i has no sons, then $f_{N-i+1}(A_r)$ is the index in the RCM ordering of that node in adj (x_i) which is labeled last by CM. Since the CM spanning tree is labeled top-to-bottom and left-to-right, this node must be the rightmost member of adj (x_i) on either level l or level $l+1$.

Using the simple counting argument of Lemma 5.4, we can obtain the following results, which are stated without proof.

THEOREM 5.5. *For a regular right triangular mesh with* $(n+1)^2$ *vertex nodes and* p_0 *interior nodes in each triangular element,*

(i) $$|\text{Env}(A_c)| = \tfrac{2}{3}(2p_0+1)^2 n^3 + O(n^2);$$

(ii) $$|\text{Env}(A_r)| = \tfrac{2}{3}(2p_0+1) n^3 + O(n^2);$$

(iii) $$\frac{|\text{Env}(A_r)|}{|\text{Env}(A_c)|} \cong \frac{1}{2p_0+1}.$$

These results show that the use of RCM leads to very impressive storage reductions for even small values of p_0. For $p_0 = 1$, as in Example 3, the reduction would be nearly 70% for large values of n. Even larger savings may be expected in the operation counts for factorization. Analysis suggests that the additional savings due to the use of edge nodes in a triangular finite element mesh are not so dramatic as those for interior nodes, and the results given in Tables 5.3 and 5.5 bear this out.

6. Conclusion. In [3], George advocates the use of envelope methods rather than band methods, and this may lead to substantial savings in many cases of practical interest. In the implementation of envelope methods, the use of Jennings' envelope storage scheme [6] is attractive. The scheme stores the rows of the envelope of the lower triangle of the $N \times N$ coefficient matrix A in a linear array. An additional N address pointers are used to locate the positions of the diagonal elements in the main storage array.[4] From Lemma 5.4, we see that these pointers may be obtained as a direct by-product of the CM ordering process, for to compute them we need only the value of f_i for each i. In order to get the pointers for the RCM ordering, we merely note that for each i, $f_{N-i+1}(A_r) = \min(N-i+1, N-k+1)$, where k is the largest label assigned by CM to a member of adj (x_i). For noninterior nodes, x_k will usually be the rightmost son of x_i in the CM spanning tree.

In terms of both storage and computation, we have seen that the RCM ordering algorithm is superior to the original CM scheme for envelope methods. Intuitively, we can attribute this to the fact that CM attempts to minimize the ordering distance between a graph node and its unordered neighbors, while RCM attempts to minimize the distance between a vertex and its ordered neighbors. Hence it appears that CM tends to minimize $|\text{Tenv}(A)|$, while RCM tends to minimize $|\text{Env}(A)|$. For systems to be solved with envelope methods, then, RCM is a better ordering to use than CM. However, we should note that for some finite element structures (e.g., the nine-point finite element grid) neither RCM nor CM is the best ordering to use.

[4] Other envelope storage schemes are possible (e.g., see [9]).

The savings found in Example 3 and the following analysis are dramatic. This demonstrates that in practical applications where the CM ordering scheme is used, it is possible to substantially reduce the amount of storage and the number of multiplicative operations required for the direct solution of systems (1.1). And although all of our examples were based on symmetric systems, it is clear that envelope methods with the RCM ordering may be applied to systems (1.1) in which only the zero structure of A is symmetric (see [9]). Thus our results may have an important effect on the solution of linear systems of equations which arise in many physical problems to which envelope methods are not currently applied.

Acknowledgments. The authors would like to thank Professor J. A. George of the University of Waterloo and Professors M. H. Schultz and S. C. Eisenstat of Yale University for their helpful discussions and advice. We also acknowledge the many constructive comments of the referees.

REFERENCES

[1] E. CUTHILL, *Several strategies for reducing the bandwidth of matrices*, Sparse Matrices and their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 157–166.

[2] E. CUTHILL AND J. MCKEE, *Reducing the bandwidth of sparse symmetric matrices*, Proc. 24th Nat. Conf. of the ACM, ACM Publ P-69, Association for Computing Machinery, New York, 1969, pp. 157–172.

[3] J. A. GEORGE, *Computer implementation of the finite element method*, Tech. Rep. STAN-CS-71-208, Computer Sci. Dept., Stanford Univ., Stanford, Calif., 1971.

[4] ———, *A survey of sparse matrix methods in the direct solution of finite element equations*, Proc. Summer Computer Simulation Conf., Montreal, Canada, July 17–19, 1973, pp. 15–20.

[5] J. A. GEORGE AND W. H. LIU, *A note on fill for sparse matrices*, this Journal, 12 (1975), pp. 452–455.

[6] A. JENNINGS, *A compact storage scheme for the solution of symmetric simultaneous equations*, Comput. J., 9 (1966), pp. 281–285.

[7] R. S. MARTIN, G. PETERS AND J. H. WILKINSON, *Symmetric decomposition of a positive definite matrix*, Numer. Math., 7 (1965), pp. 362–383.

[8] D. J. ROSE, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1972, pp. 184–218.

[9] S. C. EISENSTAT AND A. H. SHERMAN, *Subroutines for envelope solution of sparse linear systems*, Tech. Rep. 35, Dept. of Computer Sci., Yale Univ., New Haven, Conn., 1974.

[10] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, London, 1965.