On the Cuthill-McKee Algorithm
for Ordering Sparse Symmetric Matrices*
by
Wai-Hung Liu

Research Report CS-73-26
October 1973

Department of Applied Analysis & Computer Science
University of Waterloo
Waterloo, Ontario, Canada

* Work supported in part by a University of Waterloo Graduate Bursary
In this paper we examine the popular Cuthill-McKee algorithm for ordering sparse positive definite matrices. For a given matrix $A$, this algorithm is designed to produce a permutation matrix $P$ such that $A_P = PAP^T$ has a small bandwidth. If we exploit zeros to the left of the first nonzero in each row of $A$, it has been observed that reversing the ordering produced by the Cuthill-McKee algorithm is often very much better than the original ordering. We prove that in general this reversed ordering is always at least as good as the original one in terms of storage and operation count. Some numerical experiments are included.
1. Introduction

Consider the \( N \) by \( N \) sparse linear algebraic system

\[
Ax = b
\]  
(1.1)

where \( A \) is symmetric and positive definite. To solve (1.1), we may factor

\( A \) into the product \( \tilde{L}\tilde{D}\tilde{L}^T \), where \( \tilde{L} \) is unit lower triangular and \( \tilde{D} \) is a positive
diagonal matrix, and we then solve \( \tilde{L}y = b \), \( \tilde{D}z = y \), and \( \tilde{L}^Tx = z \).

Let \( P \) be any permutation matrix. The permuted linear system

\[
PAP^T(Px) = Pb
\]  
(1.2)

remains sparse, symmetric and positive definite. It is well known that

\( PAP^T \) possesses a triangular factorization \( LDL^T \) and the factorization is

numerically stable [8]. Thus, we might solve (1.2) rather than (1.1).

In this case, the relevant factors are \( L \) and \( D \). It is well known that an

appropriate choice of \( P \) can result in a significant reduction on the amount

of storage and arithmetic operations required for its direct solution. This

remark assumes that some form of compact and efficient storage scheme is

used so that some of the zeros in \( A \) and \( L \) can be exploited.

A popular ordering strategy is the Cuthill-McKee algorithm [2].

Its primary aim is to produce a permutation matrix \( P \) for which

\[
A_P = PAP^T
\]

has a small bandwidth. Following [2], we define the bandwidth of a symmetric matrix \( M \) by

\[
m = \max\{|i-j|: M_{ij} \neq 0\}
\]  
(1.3)
Although orderings which minimize the bandwidth are often far from optimal in the least-computation and/or least-fill sense, they lend themselves well to convenient and efficient data management.

In this paper, we present a detailed analysis of this widely used Cuthill-McKee algorithm (CM) and its reversed scheme (RCM). We give, in Section 3, the exact storage requirement and arithmetic operations required to factor $A_C$ in terms of its profile structure. If we exploit zeros to the left of the first nonzero in each row of the coefficient matrix, it has been observed that reversing the ordering produced by the CM algorithm is, in certain applications, very much better than the original ordering. See George [3] and Cuthill [1]. In Section 4, we prove that in general this reversed ordering is always at least as good as the original one. For certain classes of problems arising in the application of the finite element method, we present bounds for the improvement. Some numerical experiments are included. We discuss aspects of implementation in Section 5. Section 6 contains our concluding remarks.
2. Some definitions

In this section, we introduce some definitions, notations and simple results which are useful in subsequent sections.

Let \( A \) be a given matrix. Let

\[
    f_i(A) = \min\{j | A_{ij} \neq 0\}, \quad i = 1, 2, \ldots, N, \tag{2.1}
\]

that is, the column subscript of the first nonzero component in the \( i \)-th row of \( A \). We then define the envelope\(^{\dagger} \) of \( A \), where \( A \) is symmetric or lower triangular, by

\[
    \text{Env}(A) = \{(i,j) | f_i(A) \leq j \leq i\} \tag{2.2}
\]

and also the quantities

\[
    \beta_i(A) = i - f_i(A), \quad i = 1, \ldots, N. \tag{2.3}
\]

It is useful to introduce the quantities

\[
    \mu_i(A) = |\{k | k > i \text{ and } A_{k\ell} \neq 0 \text{ for some } \ell \leq i\}|, \quad i = 1, \ldots, N. \tag{2.4}
\]

The number \( \mu_i(A) \) is simply the number of "active" rows at the \( i \)-th step of factorization; that is, the number of rows in the envelope of \( A \), which intersect column \( i \). The following observation is due to George [4].

\(^{\dagger} \) It is slightly different from the definition given in [5], where the envelope for a general unsymmetric matrix is defined to be \( \text{Env}(A) \cup \text{Env}(A^T) \).
Lemma 2.1

If the symmetric factorization of matrix $A$ requires $\theta(A)$ multiplicative operations, then

$$\theta(A) \leq \frac{1}{2} N - 1 \sum_{i=1}^{N-1} \mu_i(A)[\mu_i(A)+3].$$

Furthermore, it is an equality if $A$ is assumed to have a full envelope.

It is easy to express the size of the envelope in terms of $\beta_i$ and $\mu_i$. In fact, we have [1]:

$$|\text{Env}(A)| = \sum_{i=1}^{N} \beta_i + N = \sum_{i=1}^{N} \mu_i + N. \quad \cdots (2.5)$$

We remark here that the bandwidth defined in (1.3) is exactly the quantity $\max_i \beta_i$, while the number $\max_i \mu_i$ is usually referred to as the wavefront or frontwidth of matrix $A$ [4].

Assuming that $A$ has the symmetric factorization $LDL^T$, it is natural to define

$$\text{Fld}(A) = \{(i,j)|A_{ij} = 0, \ L_{ij} \neq 0\}. \quad (2.6)$$

Since $\text{Fld}(A) \subset \text{Env}(A)$ (see [5]), the main storage requirement to keep all the nonzero entries of $L$ is always less than or equal to $|\text{Env}(A)|$. In general, zeros within the envelope may also be exploited.

\[+\] When the matrix $A$ under consideration is clear from the context, we denote by $\beta_i$ and $\mu_i$ the quantities $\beta_i(A)$ and $\mu_i(A)$ respectively.
To facilitate the discussion of ordering algorithms, we define the symmetric graph \( G(A) = (X(A), E(A)) \) associated with the symmetric matrix \( A \). Here \( X(A) = \{X_1, \ldots, X_N\} \) is the set of vertices, labelled as implied by \( A \); and \( E(A) \) is the set of edges where \( \{X_i, X_j\} \in E(A) \) if and only if \( A_{ij} \neq 0 \), \( i > j \). For any permutation matrix \( P \), the graphs \( G(A) \) and \( G(PAP^T) \) are structurally identical, but the node labels in \( G(PAP^T) \) have been permuted according to \( P \).
3. The Cuthill-McKee algorithm

The algorithm proposed by Cuthill and McKee [2] involves a direct and fast method most easily described in terms of labelling the graph structure associated with the matrix $A$, (that is, the unlabelled graph of $G(A)$).

**Step 0** Choose a starting node and relabel it 1.

**Step 1** Let $Q = \{\text{nodes which have been numbered but are connected to unnumbered nodes}\}$

$$Q = \{x_{q_1}, x_{q_2}, \ldots, x_{q_{|Q|}}\}$$

where node $x_{q_1}$ is numbered $q_1$ in the CM ordering and $q_1 < q_2 < \ldots < q_{|Q|}$. Number the neighbours of $x_{q_1}$ in order of increasing degree, counting only those connections with unnumbered vertices, then followed by the neighbours of $x_{q_2}$ and so on.

Stop whenever all nodes have been numbered.

**Step 2** Go to Step 1.

As remarked in [2], the Cuthill-McKee numbering scheme corresponds to the generation of a spanning tree\(^+\) of the graph $G(A)$ in a level-by-level fashion. In the case when $G(A)$ has more than one connected component, the process can be continued by selecting a new and unlabelled node to start a new component. Henceforth, we assume that $G(A)$ is connected, or equivalently

\(^+\) A tree is a connected graph with $N$ nodes and $N-1$ edges. A spanning tree of the graph $G$ is a subgraph of $G$, which is a tree and contains all $N$ nodes.
that the matrix $A$ is irreducible. However, the results hold in the general case.

We begin by showing that $A_C$ satisfies the following monotone profile property which is essential in subsequent discussions:

**Lemma 3.1**

Let $A_C$ be the matrix ordered by the Cuthill-McKee algorithm. If $j \leq k$, then $f_j \leq f_k$.

**Proof**

It follows from the property of the Cuthill-McKee algorithm that if $f_k < f_j$, then the node with label $k$ should be numbered before the one with $j$.

Consider the storage requirement in factoring $A_C$. The following theorem is a restatement, in our notation, of results in [5].

**Theorem 3.2**

Let $A = LDL^T$ with $f_i(A) < i$. Then $Env(L)$ is full.

It is straightforward to see that $A_C$ satisfies the hypothesis of theorem 3.2, for if there is a row in $A_C$ with $f_i = i$, the monotone profile property of $A_C$ would imply that $A_C$ is reducible, a contradiction to our assumption. The following corollaries are immediate.

**Corollary 3.3**

Let $A_C$ be factored into $LDL^T$. The number of nonzero entries in $L$ is exactly $|Env(A_C)|$.

**Corollary 3.4**

Let $A_C$ be factored into $LDL^T$. The forward and backward solving can be done in $2 \cdot |Env(A_C)| - N$ arithmetic operations.
We now turn to the problem of estimating the amount of arithmetic required for the symmetric factorization of the profile-oriented matrix $A_C$. The process can be defined by the following equations, for $i = 1, \ldots, N$,

\begin{align}
  d_{ii} &= a_{ii} - \sum_{k \neq f_i}^{i-1} a'_{ik} \ell_{ik} \quad (3.1a) \\
  a'_{ij} &= a_{ij} - \sum_{k = \max\{f_i, f_j\}}^{j-1} a'_{ik} \ell_{jk}, \quad j = f_i, \ldots, i-1 \quad (3.1b) \\
  \ell_{ij} &= a'_{ij}/d_{jj}, \quad j = f_i, \ldots, i-1 \quad (3.1c)
\end{align}

It should be clear that (3.1) is simply a variant form of the conventional defining equations for the Cholesky decomposition of symmetric matrices (Martin, Peters, and Wilkinson [7]). Here in (3.1), zeros to the left of the first nonzero in each row are exploited. In view of the monotone profile property of $A_C$, (3.1b) can be rewritten in a more simplified form as:

\begin{align}
  a'_{ij} &= a_{ij} - \sum_{k = f_i}^{j-1} a'_{ik} \ell_{jk}, \quad j = f_i, \ldots, i-1. \quad (3.1b')
\end{align}

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

**Theorem 3.5**

The symmetric factorization of the matrix $A_C$ requires

\[\Theta(A_C) = \frac{1}{2} \sum_{i=2}^{N} \beta_i (\beta_i + 3)\] arithmetic operations.
Proof From (3.1a) and theorem 3.2, we observe that $a_{ij}^*$ are all nonzero at positions within the envelope of $A_C$. Thus the products $a_{ik}^*l_{ik}^*$ and $a_{ik}^*l_{jk}$ appeared in (3.1a) and (3.1b)' respectively involve nonzero operands. In performing the i-th step defined by (3.1a), (3.1b)' and (3.1a), it is clear that:

a) to compute $d_{ii}$, we need $i-f_i$ multiplicative operations;

b) for $j = f_i, \ldots, i-1$, $a_{ij}^*$ requires $j-l-f_i$ operations, making a total of

$$\sum_{j=f_i}^{i-1} (j-l-f_i) = \frac{1}{2}(i-f_i)(i-f_i-3);$$

c) to compute $l_{ij}$ ($j = f_i, \ldots, i-1$), we use $i-f_i$ operations.

Hence, for the i-th step, the number of arithmetic operations required is

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

Summing up, we can perform the decomposition process in $\frac{1}{2} \sum_{i=2}^{N} \beta_i (\beta_i+3)$ operations, since $\beta_1 = 0$.

It should be noted that for a general profile-oriented matrix,

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

is an upperbound on the total arithmetic required for its triangular factorization. Often, this turns out to be an over-estimate. Only for monotone profile matrices whose triangular factors have full envelopes is the operation count in theorem 3.5 achieved. As an immediate corollary of this observation, we have the well-known result for band matrices (Wilkinson [8]).
Corollary 3.6

The number of operations required for the symmetric factorization of a band matrix with bandwidth $m$ is bounded by $\frac{1}{2}m(m+3)N$.

Proof Set each $\beta_i = m$. 
4. The Reversed Cuthill-Mckee Ordering

In his study of profile methods, George [3] discovered the reversed Cuthill-Mckee algorithm which renumbers the CM ordering in the reversed way. Surprisingly, this simple modification often turns out to be superior to the original one in several aspects, although the bandwidth remains unchanged. The general superior performance of the reversed algorithm has been reported in the thesis of George [3] and in the survey paper by Cuthill [1]. In this section, we shall show that in general the reversed scheme is always at least as good, as far as storage and operation counts are concerned.

As before, we denote by $A_C$ the matrix ordered by the Cuthill-McKee algorithm. Let $A_R$ be the one obtained by reversing the ordering. It is easy to see that

$$A_R = \tilde{P}A_C\tilde{P}$$

where

$$\tilde{P} = \begin{pmatrix}
1 & 1 & & & \\
& 1 & & & \\
& & \ddots & & \\
& & & 1 & \\
& & & & 1
\end{pmatrix}.$$ 

It is helpful to consider the labelled symmetric graphs $G(A_C) = (X(A_C), E(A_C))$ and $G(A_R) = (X(A_R), E(A_R))$ associated with matrices $A_C$ and $A_R$ respectively. Let

$$X(A_C) = \{x_1, x_2, \ldots, x_N\}$$

and

$$X(A_R) = \{y_1, y_2, \ldots, y_N\}.$$
where $x_i$ is numbered $i$ in the Cuthill-McKee ordering and $y_j$ is the $j$-th node in the reversed algorithm. $G(A_C)$ and $G(A_R)$ are identical structurally, and $x_i$ and $y_{N-i+1}$ represent the same node in the underlying unlabelled graph.

**Lemma 4.1**

If $i < j < k$ and $(A_C)_{ki} \neq 0$, then there exists an $r \leq i$ such that $(A_C)_{jr} \neq 0$.

**Proof** Choose $r = f_j(A_C)$. Then nodes $x_j$ and $x_r$ are connected, and it follows from the monotone envelope property of $A_C$ that

$$r = f_j(A_C) \leq f_k(A_C) \leq i$$

The following result is central to much of the subsequent analysis in this section.

**Lemma 4.2**

Let $A_C$ and $A_R$ be defined as above. For $j = 1, \ldots, N$,

$$\beta_{N-j+1}(A_R) \leq \beta_j(A_C).$$

**Proof** From the definitions in (2.3) and (2.4), it is sufficient to show that if $(A_R)_{N-i+1,N-j+1}$ lies within the envelope of row $N-i+1$ in $A_R$ where $N-i+1 > N-j+1$ (that is, $N-i+1$ is an active row during the $(N-j+1)$-st step of factorization), then the entry $(A_C)_{j,i}$ satisfies the relation $f_j(A_C) \leq i < j$.

For this $(A_R)_{N-i+1,N-j+1}$, we can always find a variable $y_{N-k+1}$ such that
\[ N-k+1 \leq N-j+1 < N-i+1 \]

and

\[(A_R)_{N-i+1,N-k+1} \neq 0.\]

\[
A_R = \begin{array}{c}
N-j+1 \\
N-i+1 \\
N-k+1 \\
\end{array}
\]

This means that we have a \( k \) such that

\[ i < j \leq k \]

and

\[(A_C)_{k,i} = (A_R)_{N-i+1,N-k+1} \neq 0 \]

By lemma 4.2, there exists a node \( x_r \) in \( G(A_C) \) with the properties: \( r \leq i < j \) and \((A_C)_{j,r} \neq 0.\)

\[
A_C = \begin{array}{c}
i \\
j \\
r \\
\end{array}
\]

Hence the corresponding entry \((A_C)_{j,i}\) lies within the envelope of \( A_C \).

The result then follows.
We are now at a position to compare the general performance of the Cuthill-McKee and its reversed algorithms in theoretical terms.

**Theorem 4.3**

\(|\text{Env}(A_R)| \leq |\text{Env}(A_C)|.\)

**Proof**

Use \(|\text{Env}(A_R)| = \sum_{i=1}^{N} \mu_i(A_R) + N\)

and \(|\text{Env}(A_C)| = \sum_{i=1}^{N} \beta_i(A_C) + N\)

and apply lemma 4.2.

**Theorem 4.4**

\(|\text{Fill}(A_R)| \leq |\text{Fill}(A_C)|.\)

**Proof**

Let \(n(A)\) be the number of nonzeros in the lower triangle of the original matrix \(A\). The symmetric factorization of \(A_C\) yields a full envelope, so that \(|\text{Fill}(A_C)| = |\text{Env}(A_C)| - n(A)\). On the other hand, we have \(|\text{Fill}(A_R)| \leq |\text{Env}(A_R)| - n(A)\). Together with theorem 4.3, we obtain the desired result.

**Theorem 4.5**

\(\theta(A_R) \leq \theta(A_C).\)

**Proof**

On combining lemma 2.1, lemma 4.2, and theorem 3.5, we have

\[
\theta(A_R) \leq \frac{1}{2} \sum_{k=1}^{N-1} \mu_k(A_R)[\mu_k(A_R)+3]
\]

\[
\leq \frac{1}{2} \sum_{i=2}^{N} \beta_i(A_C)[\beta_i(A_C)+3]
\]

\[= \theta(A_C).\]

Here the transformation \(k = N-i+1\) is used.
Example 1

To illustrate the results in this section, we choose the unlabelled star graph with \( N \) nodes. The best possible CM ordering is to start with one of the "planet" vertices. Both orderings for \( N = 7 \) are shown in Figure 5.1 below.

\[
A_C = \begin{pmatrix}
    x & x \\
    x & x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
\end{pmatrix}
\]

\[
A_R = \begin{pmatrix}
    x & x \\
    x & x & x \\
    x & x & x & x \\
    x & x & x & x & x \\
    x & x & x & x & x & x \\
    x & x & x & x & x & x & x \\
\end{pmatrix}
\]

Figure 4.1 Orderings for star graph with 7 nodes

It is clear that for the star graph with \( N \) nodes,

\[
|\text{Env}(A_C)| - |\text{Env}(A_R)| = |\text{Fill}(A_C)| - |\text{Fill}(A_R)|
\]

\[
= \sum_{i=1}^{N-3} i = \frac{1}{2}(N-3)(N-2).
\]

and

\[
\theta(A_C) - \theta(A_R) = \frac{1}{6} N^3 + \Theta(N^2).
\]

This example shows that the RCM algorithm can be significantly better than the CM strategy.
Example 2

We now analyse some examples that arise in the application of finite element method. Consider the mesh obtained by subdividing a unit square into $n^2$ small square elements of side $\frac{1}{n}$, and the finite element system\( ^{\dagger} \) associated with it. We number the nodes starting at the lower left hand corner using the CM algorithm. The case with $n = 4$ is given below.

![Diagram of a 4 by 4 regular mesh ordered by CM algorithm](image)

Figure 4.2 4 by 4 regular mesh ordered by CM algorithm

It is not difficult to see that for $n \geq 2$,

(i) \[ |\text{Env}(A_C)| = 1 + \sum_{k=1}^{n} (4k^2 + 5k) \]

\[ = \frac{2}{3}n(n+1)(2n+1) + \frac{5}{2}n(n+1)+1 \]

(ii) \[ |\text{Env}(A_C)| - |\text{Env}(A_R)| = 2n(n-1) \]

\( ^{\dagger} \) A finite element system or mesh system of equations associated with a mesh $M$ is any $N$ by $N$ symmetric positive definite system $Ax = b$, with the property that entry $A_{ij}$ is nonzero only if unknowns $x_i$ and $x_j$ are associated with nodes of the same mesh element of $M$. 
(iii) \( \theta(A_C) = \sum_{k=1}^{n} (4k^3 + 10k^2 + 3k + 1) \)
\[ = n^2(n+1)^2 + \frac{5}{3}n(n+1)(2n+1) + \frac{1}{2}n(3n+1) \]

(iv) \( \theta(A_C) - \theta(A_R) \geq \frac{8}{3}n^3 + 3n^2 - \frac{17}{3}n. \)

In table 4.1 we have tabulated the amount saved for different values of \( n \).

<table>
<thead>
<tr>
<th>Problem</th>
<th>Storage</th>
<th>Operation Count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CM</td>
<td>RCM</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>36</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>171</td>
</tr>
<tr>
<td>8</td>
<td>81</td>
<td>997</td>
</tr>
<tr>
<td>16</td>
<td>269</td>
<td>6665</td>
</tr>
<tr>
<td>32</td>
<td>1089</td>
<td>48401</td>
</tr>
</tbody>
</table>

Table 4.1 Theoretical amount saved for the regular square mesh

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 4.2 is subdivided into two right triangles), and its corresponding mesh system.
Quadratic interpolation is first used, where each triangular element has three vertex unknowns and three edge unknowns. The corresponding CM ordering for $n = 2$ is given in figure 4.3.

```
   15 --- 17 --- 22 --- 24 --- 25
   |     |     |     |     |
  13   14 --- 19 --- 20 --- 23
   |     |     |     |     |
   6    8     9 --- 18 --- 21
   |     |     |     |     |
   3    4     7 --- 12 --- 16
   |     |     |     |     |
   1    2     5 --- 10 --- 11
```

Figure 4.3 2 by 2 regular right triangular mesh with element 2-6* ordered by CM algorithm

Experiments using different values of $n$ are tried. The result tabulated in table 4.2 shows that substantial savings in computation and storage can be achieved if RCM rather than CM algorithm is used.

---

* We adopt the notation in [3], where the two-part hyphenated name refers respectively to the degree of the polynomial and the number of nodes associated with the element.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Storage</th>
<th>Operation Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>N=(2n+1)^2</td>
<td>Nonzeros</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>49</td>
<td>207</td>
</tr>
<tr>
<td>4</td>
<td>81</td>
<td>360</td>
</tr>
<tr>
<td>5</td>
<td>121</td>
<td>555</td>
</tr>
<tr>
<td>6</td>
<td>169</td>
<td>792</td>
</tr>
<tr>
<td>7</td>
<td>225</td>
<td>1071</td>
</tr>
<tr>
<td>8</td>
<td>289</td>
<td>1392</td>
</tr>
<tr>
<td>9</td>
<td>361</td>
<td>1755</td>
</tr>
</tbody>
</table>

Table 4.2 Experimental savings on the regular right triangular mesh with element 2-6.

In case of cubic interpolation with each element having ten nodal unknowns, the savings are even more dramatic. The mesh with n = 2 would be numbered by the CM ordering as shown below.

![Figure 4.4 2 by 2 regular right triangular mesh with element 3-10 ordered by CM algorithm](image)
As before, we tabulate the storage and operation requirements for the two algorithms using different n.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Storage</th>
<th>Operation Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>N=(3n+1)^2</td>
<td>Nonzeros</td>
</tr>
<tr>
<td>2</td>
<td>49</td>
<td>312</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>684</td>
</tr>
<tr>
<td>4</td>
<td>169</td>
<td>1200</td>
</tr>
<tr>
<td>5</td>
<td>256</td>
<td>1860</td>
</tr>
<tr>
<td>6</td>
<td>361</td>
<td>2664</td>
</tr>
</tbody>
</table>

Table 4.3 Experimental savings on the regular right triangular mesh with element 3-10.
5. Implementation

In [3], George has advocated the use of profile methods instead of band methods. In fact, it is possible to save about one-third of the storage and to half the computation required to perform the factorization in many cases of practical interest.

Corollary 3.3 strongly suggests the use of Jennings' profile storage scheme [6] in connection with the CM ordering (see [5]). The scheme stores the rows of the envelope of the lower triangle of coefficient matrix A in a linear array. An additional N auxiliary address pointers are needed to locate the positions of the diagonal elements in the main storage array. Thus, the data structure of the scheme requires the computation of these N extra pointers. We point out, however, they can be obtained as an immediate byproduct of the CM ordering process.

This can be best explained in terms of the spanning tree associated with the CM algorithm. To set up the address pointers, it is sufficient to know the number of locations required for each row in the envelope. Note that the storage for row \( i \) is exactly \( i - f_i + 1 \). It is easy to see that

\[
f_1(A_C) = 1
\]

and for \( i \neq 1 \)

\[
f_i(A_C) = j, \text{ where } x_j \text{ is the direct ancestor of node } x_i \text{ in the spanning tree associated with the CM order. We use example 2 with } n = 2 \text{ to illustrate this observation.}
Figure 5.1 Spanning tree associated with the CM ordering for the 2 by 2 regular square mesh.

Hence, the address pointers can be readily obtained during the CM ordering procedure.

To implement the RCM algorithm, it is also attractive to use Jennings' scheme in view of theorem 4.3. Extra work is needed to set up our data structure for the coefficient matrix. If we define

$$\ell_i(A) = \max\{j | A_{ij} \neq 0\}, \quad i = 1, \ldots, N$$

that is, the column subscript of the last nonzero component in the i-th row of A, then

$$f_j(A_R) = N - \ell_{N-j+1}(A_C)+1, \quad j = 1, \ldots, N.$$  

We can establish the address pointers for the main storage of $A_R$ without much difficulty by keeping track of the quantities $\ell_i(A_C)$.

One final remark is that we should use equations (3.1a), (3.1b) (instead of (3.1b)') and (3.1c) for the symmetric factorization of the matrix $A_R$. In that case, the number of operations for the decomposition is exactly given by

$$\frac{1}{2} \sum_{k=1}^{N-1} \nu_k(A_R)[\nu_k(A_R)+3].$$
6. Conclusion

Evidently, in the context of profile or envelope methods, the RCM algorithm compares favourably with the original CM ordering. Thus far we have not brought into the discussion of the execution time in performing the two ordering algorithms. But, there is little doubt that the two strategies under consideration require approximately the same amount of work. In exchange for the extra but comparatively insignificant effort to reverse the ordering and to set up the address pointers for the data management, there is the possibility of substantial reduction in cost of storage and factorization. Moreover, we are guaranteed that the RCM ordering is at least as good, provided we exploit all the zeros to the left of the first nonzero in each row of the matrix.

The savings obtained in example 3 are dramatic. This demonstrates that in practical applications it is possible to reduce over half of the storage requirement and to save over three-quarters of the operations necessary for the symmetric decomposition. Our study provides strong arguments for using the RCM algorithm. Since the CM algorithm is a popular ordering scheme (particularly in structural analysis applications), our investigation in this paper is of practical interest.

Acknowledgement

The author would like to thank Professor Alan George for his many helpful discussions and criticisms.
References


