JOURNAL OF RESEARCH of the National Bureau of Standards
Volume 83, No. 5, September–October 1978

Some Comments on Shier's Paper for Inverting Sparse Matrices*

J. M. McNamee**

Department of Computer Science and Mathematics, York University, Downsview, Ontario M3J 2R7

May 9, 1978

A paper by Shier (J. Res NBS **80B**) shows how to partition the graph of a matrix into a tree so as to minimize the number of operations required to invert the matrix. The present paper shows how to economically solve a sparse system of linear equations after the application of Shier's method to the coefficient matrix.

Keywords: Sparse equations; tree partitions

1. Introduction

In [1][1] Shier points out that if: (a) the graph corresponding to a sparse matrix A is partitioned into subgraphs which themselves can be regarded as nodes of a tree, and (b) the nodes of this tree are suitably numbered; then A can be partitioned as (A_{ij}) where A_{ij} are submatrices and the ith row of A is

$$\underset{\sim}{A}_i = (A_{i1}, \cdots, A_{ii}, 0, 0, \cdots, 0, A_{i,r(i)}, 0, \cdots, 0), \qquad i = 1, \cdots, n \tag{1}$$

and where node $r(i)$ is the "father" of i in the tree. Also, $A_{ik} = 0$ $(k < i)$ unless $r(k) = i$ and A is block incidence-symmetric. He then describes a relatively efficient way of finding A^{-1}, involving the computation of A_{ii}^{-1} and similar sub-matrices by standard methods for dense matrices combined with recursive application of his algorithm. He also describes a method for carrying out the tree partitioning in (a) above.

Unfortunately he does not describe in detail how his method can be applied to the much more common problem of solving sparse equations, although he does mention (p. 252, lines 3–5) that it can be so applied. This will now be done.

2. Solution of Equations

We have to solve:

$$A\underset{\sim}{x} = \underset{\sim}{b} \tag{2}$$

where A is partitioned as in (1) and x and b are partitioned conformably into sub-vectors

$$(\underset{\sim}{x}_i) \quad \text{and} \quad (\underset{\sim}{b}_i) \tag{3}$$

We may very efficiently solve (2) by Block Gaussian Elimination as follows:

(A) (Elimination of sub-diagonal sub-matrices)

For $i = 1, \cdots, n-1$ do:

$$\text{(I) Form multipliers } m_{r(i),i} = A_{r(i),i}A_{ii}^{-1} \tag{4}$$

---

* An invited paper.

** Present address: Department of Computer Science and Mathematics, York University, Downsview, Ontario M3J 2R7.

[1] Figures in brackets indicate the literature references at the end of this paper.

Eliminate $A_{r(i),i}$ by subtracting $m_{r(i),i}$x (row i) from row $r(i)$, i.e.

$$\text{(II) Update } A_{r(i),r(i)} \leftarrow A_{r(i),r(i)} - A_{r(i),i}A_{ii}^{-1}A_{i,r(i)} \tag{5}$$

$$\text{(III) Update } \underset{\sim}{b}_{r(i)} \leftarrow \underset{\sim}{b}_{r(i)} - A_{r(i),i}A_{ii}^{-1}\underset{\sim}{b}_i \tag{6}$$

(B) (Back-Substitution)

$$\text{(IV) } \underset{\sim}{x}_n = A_{nn}^{-1}\underset{\sim}{b}_n \tag{7}$$

(V) For $i = n - 1, \cdots, 1$ do:

$$\underset{\sim}{x}_i = A_{ii}^{-1}[\underset{\sim}{b}_i - A_{i,r(i)}\underset{\sim}{x}_{r(i)}] \tag{8}$$

(The above simply constitutes Gaussian Elimination with coefficients consisting of submatrices instead of scalars.) The great advantage of this method is that there is no fill-in except within the blocks, i.e., a zero submatrix always remains zero.

3. A More Economical Method

Further economy can be obtained by omitting the explicit calculation of m in (4). Rather we can perform triangular decomposition

$$A_{ii} = L_{ii}U_{ii} \tag{9}$$

Then (6) can be replaced by:

$$\text{(I). Solve } L_{ii}\underset{\sim}{v}_i = \underset{\sim}{b}_i \tag{10}$$

$$\text{(II) Solve } U_{ii}\underset{\sim}{z}_i = \underset{\sim}{v}_i \tag{11}$$

$$\text{Then } \underset{\sim}{z}_i = A_{ii}^{-1}\underset{\sim}{b}_i \tag{12}$$

$$\text{(III) Form } \underset{\sim}{w}_{r(i)} = A_{r(i),i}\underset{\sim}{z}_i \tag{13}$$

$$\text{(IV) } \underset{\sim}{b}_{r(i)} \leftarrow \underset{\sim}{b}_{r(i)} - \underset{\sim}{w}_{r(i)} \tag{14}$$

(5) Can be replaced by similar calculations with each column of $A_{i,r(i)}$ taking the place, in turn, of $\underset{\sim}{b}_i$.

(7) Can be replaced by (9), (10), (11) with $i = n$.

(8) Can be replaced by:

$$\text{(I) } \underset{\sim}{b}_i \leftarrow \underset{\sim}{b}_i - A_{i,r(i)}\underset{\sim}{x}_{r(i)} \tag{15}$$

$$\text{(II) Solve } L_{ii}\underset{\sim}{v}_i = \underset{\sim}{b}_i \tag{16}$$

$$\text{(III) Solve } U_{ii}\underset{\sim}{x}_i = (\underset{\sim}{v})_i \tag{17}$$

4. Operation Count

(1) If the explicit inverse A_{ii}^{-1} and $m_{r(i),i}$ are employed as in §2, using dense matrix techniques, the operation count would be as follows, assuming A_{ii} is order p_ixp_i, and $p_i = p$ for all i: the formation of A_{ii}^{-1}, $m_{r(i),i}$ and equation (5) each require $0(p^3)$ multiplications, for a total of $0(3p^3)$; while eq (6) requires $0(p^2)$, (7) requires $0(p^3 + p^2)$ and (8) requires $0(2p^2)$. Thus, the total number of multiplications is approximately

$$(n-1)3p^3 + (n-1)3p^2 + p^3 + p^2 = (3n-2)(p^3 + p^2). \tag{18}$$

(II) If the method of §3 is used we have: equation (9) requires $0(p^3/3)$ multiplications; equations (10), (11) and (13) together need $0(2p^2)$. The solution of equations (10), (11) and (13) with any column of $A_{i,r(i)}$ in place of $\underset{\sim}{b}_i$ requires $0(2p^2)$ for each column, i.e. $0(2p^3)$ in all. Equations (15)–(17) require $0(2p^2)$. Equations (9), (10) and (11) for $i = n$ require $0\left(\frac{p^3}{3} + p^2\right)$. Thus, the total number of multiplications required for this method is approximately

$$\left(\frac{7}{2}n - 2\right)p^3 + (4n-3)p^2 \tag{19}$$

Thus the method of §3 is more efficient for large $p_i = p$, when we may ignore multiplicative and overhead factors.

(III) If the equations are solved directly without any partitioning, as if they were full, the number of multiplications required is $\approx \frac{1}{3}(np)^3 + (np)^2$, which for large n is much greater than $\frac{7}{3}np^3$.

5. Labelling of Tree Nodes

The nodes of the tree must be numbered in such a way that its incidence matrix has the form (1). This can be accomplished for example by a modification of the "Reverse Cuthill-McKee Algorithm" [2]. (This was originally devised as a band-width minimization technique, although that aspect has no relevance in the present context.) Simplified and re-worded for our purposes the algorithm may be described thus:

A. Suppose there are N nodes in the tree. Choose an arbitrary node and number it N (this is defined as the only member of "level" 1). Set $I = 1$ and $J = N - 1$.

B. Consider all nodes adjacent to nodes in level I but as yet unnumbered (they will be defined as members of level $I + 1$). Suppose there are K such nodes in all. If $K = 0$ terminate. Otherwise assign to them the numbers $J, J-1, \cdots, J-K+1$.
Set $J = J - K$ and $I = I + 1$.
Repeat step B until $K = 0$.

It is simple to prove that the incidence matrix of a tree thus numbered has the form (1), i.e. each node (numbered i, say) is adjacent to only one node having a higher number (say $r(i)$).

PROOF: Suppose if possible a node numbered i is adjacent to *two* nodes numbered r_1 and r_2 such that $r_1 > i$, $r_2 > i$. Then the nodes r_1 and r_2 belong to lower levels than node i. Hence they are both connected, via paths not including node i, to node N. Thus we have two separate paths connecting nodes i and N, i.e. we have a loop. But this contradicts the assumption that the graph is a tree. Hence there must be only one node adjacent to i with number $> i$. Q.E.D.

6. References

[1] Shier, D. R., Inverting sparse matrices by tree partitioning, J. Res. Nat. Bur. Stand (U.S.), **80B** (Math. Sci.), No. 2, pp. 245–257 (Apr.–June 1976).

[2] Cuthill, E., Several strategies for reducing the bandwidth of a matrix. In: Sparse Matrices and their applications, Ed. D. J. Rose and R. A. Willoughby, pp. 157–160 (Plenum Press, New York, N. Y., 1972).