REDUCING THE BANDWIDTH OF SPARSE SYMMETRIC MATRICES

E. Cuthill and J. McKee
Naval Ship Research and Development Center
Washington, D. C. 20007

Abstract

The finite element displacement method of analyzing structures involves the solution of large systems of linear algebraic equations with sparse, structured, symmetric coefficient matrices. There is a direct correspondence between the structure of the coefficient matrix, called the stiffness matrix in this case, and the structure of the spatial network delineating the element layout. For the efficient solution of these systems of equations, it is desirable to have an automatic nodal numbering (or renumbering) scheme to ensure that the corresponding coefficient matrix will have a narrow bandwidth. This is the problem considered by R. Rosen[1]. A direct method of obtaining such a numbering scheme is presented. In addition several methods are reviewed and compared.

1. Introduction

The finite element displacement method of analyzing structures involves the solution of large systems of linear algebraic equations with sparse, structured, symmetric coefficient matrices. This is also true of many other physical problems involving networks explicitly, or into which networks are introduced when approximate solutions of field problems are sought.

In our discussion we confine ourselves to systems of equations with sparse, symmetric, positive definite coefficient matrices. In order to solve such systems of equations efficiently we must conserve computer storage as well as calculation time.

Several approaches are available for the storing of these coefficient matrices; the choice of scheme being governed by the particular method of solution, whether iterative or direct. Iterative methods as described by Varga[2] can be very efficient. For the types of problems considered here the methods described by Carré[3] are especially pertinent. Direct methods based on Gaussian elimination may be preferable if the pattern of non-zero elements in the coefficient matrix is such that not too many non-zero elements have to be introduced during the elimination. Parter[4], Sato and Tinney[5], Tewarson[6], Nathan and Even[7], Steward[8] among others consider strategies that will introduce into the coefficient matrix as few non-zero elements as possible during the solution of the system of equations. The disadvantage of these methods is that when automated, unless the matrix is of rather special form, they tend to involve a considerable amount of "red-tape programming" along with the calculations.

Another approach to this problem of taking advantage of sparseness, which tends to involve fewer clerical instructions when automated, is to determine a permutation matrix P such that the symmetric matrix:

$$PAP^T$$

derived from the coefficient matrix A (by corresponding row and column permutations) will be such that the non-zero elements will cluster as much as possible, in some sense, about the main diagonal. For Jennings[12] storage scheme we would want to determine P to minimize for PAP^T a function of the form:

$$\frac{1}{N}\sum_{i=1}^{N}\theta_i \qquad (1)$$

where θ_i is the difference in column indices of the diagonal element for row i and the first non-zero element for row i, and N is the order of A. For the diagonal band matrix storage scheme, one would want bandwidth minimization, i.e., one would want to minimize:

$$\max_i \theta_i \ . \qquad (2)$$

In this paper we address ourselves to the problem of bandwidth minimization. For some discussion of the problem of minimizing (1) see Tewarson[9]. An iterative program for minimizing (1) is presented by Akyuz and Utku[10].

For automatic bandwidth minimization, Rosen[1] has published a program based on an iterative method. This method may converge to a local minimum as we shall see. Alway and Martin[11] do an educated search of permutations to arrive at one which can be used to permute rows and corresponding columns of the given matrix to minimize bandwidth. This can be very time consuming, especially for large matrices. Our strategy is to look at only a few such permutations suggested by the structure of a graph associated with the coefficient matrix. These in many cases of interest lead to minimum or near minimum bandwidth for the permuted matrix. Our program is significantly faster than the others.

The initial purpose of our effort was to arrive at a permutation to give a matrix with reasonably narrow bandwidth for use as a starting point so that iterative methods such as that of reference 1 would converge to a numbering scheme giving somewheres near the minimum bandwidth. However, for fairly regular networks such as those arising in our structures work, the permutations generated by our program have usually been such that Rosen's program when applied to the output has given very little improvement. For sparse, random matrices, we have gotten more significant improvements with Rosen's program.

2. Notation and Definitions

Before describing our algorithm for determining a permutation matrix which will give a narrow bandwidth, we will summarize some of the notation to be used plus definitions of needed terms from graph theory.

The systems of linear algebraic equations being considered are of the form:

$$\vec{Ax} = \vec{b}$$

where A is an N x N sparse symmetric, positive definite matrix. The elements of A will be designated a_{ij} where i is a row index and j a column index. We will use the maximum value of $|i-j|$ for non-zero a_{ij} as a measure of the bandwidth of A.

For the graph G(A) corresponding to the matrix A we will have N nodes labeled i = 1,2,...,N. For each non-zero element a_{ij}, i < j of A there will be an edge connecting nodes i and j. From the graph of A we can determine the position of all off-diagonal non-zero elements of A.

Two nodes of G(A) are said to be adjacent if they are connected by an edge. Two nodes of G(A) are said to be connected if there is a sequence of edges joining them such that consecutive edges have a common end point. A graph is said to be connected if every pair of nodes of the graph are connected. If G(A) is connected, the corresponding matrix[2] is irreducible. A component of a graph is a connected subgraph which is not contained in a larger connected subgraph.

The degree of a node i of G(A) is the number of edges meeting at i. For the corresponding matrix, this is the number of non-zero off-diagonal elements in row i.

A tree is a graph which has no isolated nodes and which has one more node than the number of edges. A tree contains no closed loops.

A spanning tree of G(A) is a subgraph of G(A) which is a tree containing all the nodes of G(A).

When we permute the rows and columns of A using the permutation matrix P generating PAP^T, the graph of PAP^T, namely $G(PAP^T)$, is structurally identical to A but the node labels have been permuted according to the permutation matrix P. P has one non-zero unit element in each row and column; in row i it will be p_{ij} where i is the new node label and j designates the original one.

3. A Nodal Numbering Scheme

Given the matrix A, from its graph G(A) as defined above, we can see immediately that a lower bound for our measure of the bandwidth of PAP^T for any permutation matrix P is given by the smallest integer greater than or equal to D/2 where D is the maximum degree of any node of G(A). For the matrix of Figure 1A, a lower bound for the bandwidth of PAP^T is thus 1, while for those of Figures 1B and 1C it is 2. Note that in Figure 1A the nodal numbering pattern is such that this lower bound has been achieved.

The first procedure presented here for determining P or equivalently a renumbering scheme for G(A) is given below:

a) Select a starting node which might be a node of minimum degree. Relabel it 1. See Figure 2. We consider two possibilities, labeling node A as 1 in Figure 2A and labeling node B as 1 in Figure 2B.

b) The nodes adjacent to this node are numbered in sequence beginning with 2 in the order of their increasing degree. These nodes are at distance 1 from node 2. They are said to be at the first level. The 3 nodes of Figures 2A and 2B at distance 1 from node 2 are labeled 2, 3, and 4.

c) This procedure is repeated for each node at the first level in sequence, i.e., first for node 2, then for node 3, etc. Thus, in Figure 2A, the node adjacent to node 2 which has not already been numbered will become node 5, adjacent to 3 we have additional nodes 6, 7, and 8, node 6 having a lower degree than node 7, which is of lower degree than node 8. Thus, nodes 5, 6, 7, and 8 are at the second level. These are the nodes at distance 2 from the starting node. They are said to be at the second level.

d) The above procedure is repeated for the nodes at each successive level.

This procedure will terminate when all the nodes of a component have been labeled. If there is more than one component, the procedure can be continued by selecting the unlabeled node to start a new component.

For the two sets of labels developed in Figure 2 for the same graph, we see that the one in Figure 2A leads to a matrix with a bandwidth of 5, while that in Figure 2B leads to a matrix with bandwidth 3. Since the nodes of maximum degree are I and J which are of degree 6, a lower bound for the bandwidth will be 3. This we have achieved. Figure 3 gives an "obvious" type of numbering scheme that one might guess would result in a minimum bandwidth. It does not.

When one starts with a node of minimum degree, our simple numbering scheme described does surprisingly well for many occurring patterns which are fairly regular. For a matrix which can be transformed to band diagonal form with no zero elements in the band, an optimum numbering scheme will automatically result. Some examples are given in Figure 4.

Other cases in which our scheme will automatically give either an optimum bandwidth or at most one greater than the optimum are for an n x m rectangular network as shown in Figure 5A and the corresponding triangular network in Figure 5B.

The generation of a numbering scheme by our procedure corresponds to the generation of a spanning tree for G(A). See the trees in part d) of Figures 2A and 2B indicated by the solid lines. Note that a node of minimum or near minimum degree is chosen as the root of the tree. The nodes adjacent to this node are at the next level in the tree. They are at a distance 1 from the root. The nodes adjacent to these nodes which have not yet been labeled become second level nodes. They are at a distance 2 from the root, etc. A numbering scheme which will minimize the maximum number of nodes per level of such a spanning tree will in general be a good one since nodes at any level are connected only to those of the same level, the preceding level, or the following level. Note in Figure 4A the number of nodes per level are 1, 3, 4, 2 respectively while in Figure 2B they are 1, 3,

3, 3 respectively, so that one might expect that the numbering scheme in Figure 2B would be the better one, which it is. In fact, it is optimum since there is a node of degree 6 and this scheme gives a bandwidth of 3.

Regardless of the details of the numbering scheme, when the maximum number of nodes per level is known for a spanning tree such as we are generating, an upper bound for the bandwidth of PAP^T is automatically available. It will be 2N-1, where N is the maximum number of nodes per level. The factor of two occurs since at worst the first node at one level can be connected to the last node of the next level. Note that for the trees of Figures 2A and 2B respectively these are 7 and 5 which are both conservative.

In addition, the minimum bandwidth that can be obtained using a numbering scheme based on a particular spanning tree is the maximum number of nodes per level. In the numbering schemes of Figures 2A and 2B this is 4 and 3, respectively.

In many cases, starting with a node of lowest degree is good strategy, but it is easy to give examples in which this will not lead to an optimum numbering. Figure 6 contains such an example in which the starting node for an optimum numbering scheme is not of lowest degree.

Our FORTRAN program includes as parameters two integers N and M which are used as follows: If for any node of G(A) D_{min} is the minimum degree, and D_{max} is the maximum degree, then for each component of G(A) spanning trees are generated starting at each node of degree D such that

$$D_{min} \leq D \leq D_{min} + \frac{N}{M} D_{max} . \qquad (3)$$

This allows one to control the degrees of the set of nodes tried. Then for each component of G(A), for those starting nodes leading to the minimum maximum number of nodes per level, relabeling schemes are generated following our prescription. Again for each component, that leading to a minimum bandwidth is adopted. A good strategy appears to be to choose M=2, N=1. The program is very fast even when the number of nodes is large. The running time increases nearly linearly with the product of the number of nodes, the number of nodes of degree D satisfying (3) and, the average degree per node. For every other program of which we are aware the running time increases much faster as the number of nodes increases.

Several byproducts result from the organization of our computer program. One can readily check whether a matrix is reducible or not. If it is reducible, any relabeling scheme generated by our program will lead to a matrix in completely reduced form[13]. As a byproduct of generating a spanning tree, we generate the distance of every node from the starting node. When this is done for every node of the graph, the maximum distance encountered is the diameter of the graph.

One should also consider sets of trees rooted at more than one node. These will often give improvements for more complex graph structures, and even some simple ones. See Figure 7. Work related to developing improved strategies for such schemes is in progress. Our present program has an input option permitting the specification of one or more nodes as starting points for the numbering scheme. Other improvements to the numbering procedure can be made when there is some exploration of the alternate paths possible under the rules for generating our scheme. See Figure 5, for example. After rooting the spanning tree at node 1, nodes 2 and 3 can be labeled in one of two ways. Once a decision is made here, there are no further alternate paths that our numbering scheme could take. In this case, one of the two choices leads to an optimum numbering scheme.

4. Comparison with Other Methods

Alway and Martin[11] present a method of surveying those permutations which could lead to bandwidth reduction. Whenever the matrix is such that the permuted form contains no zero elements within the band, they will find the appropriate permutation for minimum bandwidth very efficiently. Our scheme is also especially efficient for such a matrix; in fact, as already noted, our numbering scheme, starting with any node of lowest degree will lead to an optimum in this case. In the form described in reference 11, their algorithm is not usually practical for the very large order matrices (of the order of 1000 to 10,000) for which our program was developed. The amount of calculation time required goes up too rapidly.

Rosen[1] presents an iteration scheme for "matrix bandwidth minimization". This scheme is based on a search for interchanges of two rows and the corresponding columns which will reduce or at worst not increase the bandwidth. One can readily construct matrices for which such a procedure will give no reduction in bandwidth, but for which a permutation exists which would give a considerable reduction.

For example, consider a matrix with a graph represented by a rectangular network numbered in a regular pattern as shown in Figure 8. Figure 8A shows a possible numbering scheme for which our measure of the corresponding matrix bandwidth is M, while Figure 8B gives a numbering scheme for the same network for which our measure of the corresponding matrix bandwidth is N. Since we are assuming that N > M, the first scheme is better. Suppose we try to interchange two node labels, say i and j (this is equivalent to permuting corresponding rows and columns of the related matrix) in the scheme of Figure 8B. Figure 8C shows the case in which one of them (i) is not from the top or bottom row of the network; Figure 8D shows the case in which they are both in the top row; Figure 8E shows the case in which they are both in the bottom row; and Figure 8F shows the case in which one is in the top row and the other in the bottom row. These are all of the possible types of interchanges involving two nodes. In each case, we see that our measure of the bandwidth of the corresponding matrix will be greater than N. Thus Rosen's program will not improve upon this numbering, although Figure 8A shows that such an improvement is clearly possible.

Versions of Rosen's program[1] and ours have been compared on a number of sample problems. Figures 9 to 12 illustrate the starting numbering schemes used and the renumbering schemes generated by both Rosen's program and ours for 4 simple structures. In each case for our scheme we used as starting node, a node of minimum degree. Table 1 summarizes the results obtained for these problems. Here column (O) contains the original bandwidth. (R) contains the bandwidth obtained using Rosen's scheme and (C) contains that for our scheme. In none of these cases would Rosen's program improve upon the scheme generated by our program.

TABLE 1

Problem	No.Nodes	(O)	(R)	(C)
1. Fig. 9	16	15	3	2
2. Fig. 10	45	36	12	5
3. Fig. 11	19	18	4	4
4. Fig. 12	42	37	9	9

Nodal numbering schemes for a set of five larger structural problems were generated by an input processor using a convenient input description, but with no thought of optimal numbering for bandwidth minimization. The structures and their modeling are briefly described in Table 2.

TABLE 2

Problem	Description	Average degree/node
1A	Plate with elliptic hole modeled with triangular elements	6.
2A	Spherical sector with hole modeled with triangular elements	6.
3A	Cylinder with hole modeled with truss elements	4.
4A	Cylinder with hole modeled with triangular elements	6.
5A	Cylinder with hole modeled with quadrilateral elements	8.

The strategy used for our numbering scheme was that described in Section 3 with $\frac{N}{M}$ of Equation 3 equal to ½. The results are given in Table 3. Here columns (O), (R) and (C) have the same meaning as before. Columns (RT) and (CT) contain the running times in seconds taken on an IBM 360/91 for Rosen's program and for ours, respectively.

TABLE 3

Problem	No.Nodes	(O)	(R)	(C)	(RT)	(CT)
1A	111	36	18	12	5.3	.26
2A	82	73	18	9	3.4	.26
3A	114	113	20	13	8.9	.38
4A	114	113	26	15	15.1	.40
5A	114	113	29	18	18.9	.53

In none of these cases would Rosen's program improve upon our generated scheme.

Finally, a set of problems was run for which the positions of off-diagonal elements of sparse symmetric matrices were generated using a random number generator. The results are summarized in Table 4. The density refers to the ratio of the number of non-zero elements in the matrix to the total number of elements. To obtain the starred results, the first node encountered of lowest degree was used to start our numbering scheme. The unstarred ones were obtained using essentially the strategy used for the problems of Table 2, that is, the scheme of Section 3 with $\frac{N}{M}$ of Equation 3 equal to ½. Again (R) identifies Rosen's scheme, (C) identifies our scheme, but in addition, (CR) identifies our scheme followed by Rosen's.

TABLE 4

Order	Density	No.	(O)	FINAL (R)	FINAL (C)	FINAL (CR)
10	.4	6	7	3.8	3.3	3.3
	.5	16	7.9	4.4	4.5	4.4
50	.04	4	40.3	3.8	3.3	3.0
		7*	42.9*	3.7*	3.0*	2.6*
	.07	2*	44.5*	12.5*	13.5*	12.0*
	.10	6	47.2	19.3	18.5	17.0
		8*	46.8*	18.9*	22.5*	
	.14	5*	46.2*	23.6*	28.2*	22.3*

In contrast to the results presented earlier for the more regular networks, the results of these runs show that Rosen's scheme will frequently improve upon results generated by ours for random matrices, especially as the density of elements increases.

5. References

1. R. Rosen, "Matrix bandwidth minimization." Proceedings of 23rd National Conference, ACM, ACM Publication P-68, Brandon/Systems Press, Princeton, N. J. (1968), pp. 585-595.
2. R. S. Varga, "Matrix Iterative Analysis." Prentice-Hall, Inc., New York (1962).
3. B. A. Carré, "The partitioning of network problems for block iteration." Computer Journal 9, (1966), pp. 84-97.

4. S. Parter, "The use of linear graphs in Gauss elimination." SIAM Review 3, (1961), pp. 119-130.

5. N. Sato and W. F. Tinney, "Techniques for exploiting the sparsity of the network admittance matrix." IEEE Transactions on Power Apparatus and Systems, 82, (1963), pp. 944-950.

6. R. P. Tewarson, "Solution of a system of simultaneous linear equations with a sparse coefficient matrix by elimination methods." BIT 7, (1967), pp. 226-239.

7. A. Nathan and R. K. Even, "The inversion of sparse matrices by a strategy derived from their graphs." Computer Journal 9, (1966), pp. 190-194.

8. D. V. Steward, "On an approach to techniques for the analysis of the structure of large systems of equations." SIAM Review 4, (1962), pp. 321-342.

9. R. P. Tewarson, "Row-Column permutation of sparse matrices." Computer Journal 10, (1967), pp. 300-305.

10. F. A. Akyuz and S. Utku, "An automatic relabeling scheme for bandwidth minimization of stiffness matrices." AIAA Journal, 6, (1968), pp. 728-730.

11. G. G. Alway and D. W. Martin, "An algorithm for reducing the bandwidth of a matrix of symmetrical configuration." Computer Journal 8, (1965), pp. 264-272.

12. A. Jennings, "A compact storage scheme for the solution of symmetric linear simultaneous equations." Computer Journal 9, (1966),pp.281-285

13. F. Harary, "A graph theoretic approach to matrix inversion by partitioning." Numerische Mathematik, 4, (1962), pp. 128-135.

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{pmatrix} X & X & 0 & 0 \\ X & X & X & 0 \\ 0 & X & X & X \\ 0 & 0 & X & X \end{pmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

$G(A)$:

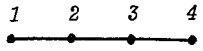


Figure 1A

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{pmatrix} X & X & 0 & X & 0 \\ X & X & X & 0 & X \\ 0 & X & X & 0 & 0 \\ X & 0 & 0 & X & X \\ 0 & X & 0 & X & X \end{pmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

$G(A)$:

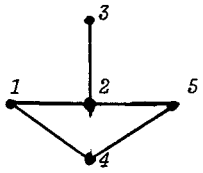


Figure 1B

$$A = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{pmatrix} X & X & X & X & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 & 0 & 0 \\ X & 0 & X & 0 & X & 0 & 0 \\ X & 0 & 0 & X & 0 & X & X \\ 0 & 0 & X & 0 & X & 0 & 0 \\ 0 & 0 & 0 & X & 0 & X & 0 \\ 0 & 0 & 0 & X & 0 & 0 & X \end{pmatrix} \end{array} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix}$$

$G(A)$:

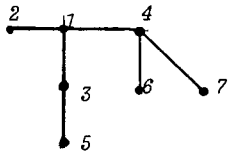
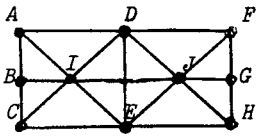


Figure 1C

FIGURE 1: Some Examples of Matrices and Their Graphs



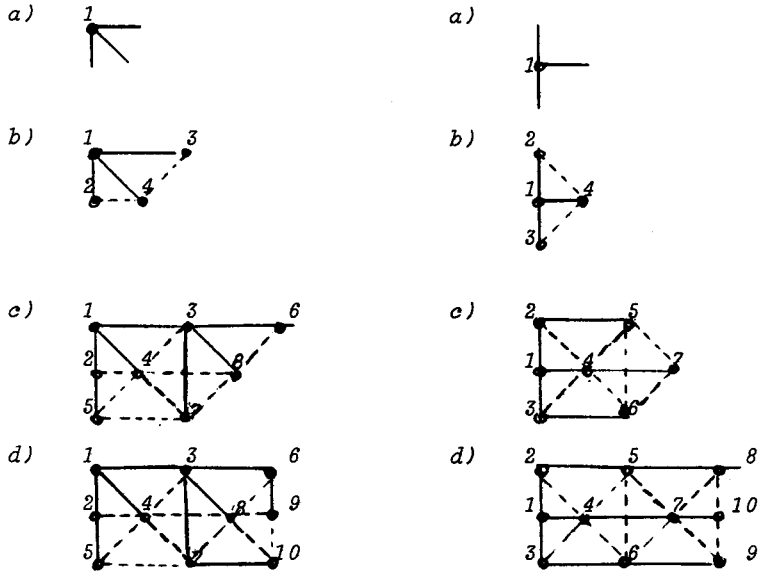


FIGURE 2A FIGURE 2B

FIGURE 2: Our Nodal Numbering Scheme

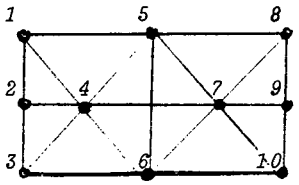
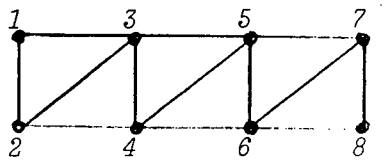


FIGURE 3

$PAP^T =$

	1	2	3	4	5	6	7	8	
	X	X	X	0	0	0	0	0	1
	X	X	X	X	0	0	0	0	2
	X	X	X	X	X	0	0	0	3
	0	X	X	X	X	X	0	0	4
	0	0	X	X	X	X	X	0	5
	0	0	0	X	X	X	X	X	6
	0	0	0	0	X	X	X	X	7
	0	0	0	0	0	X	X	X	8

$G(PAP^T)$:



$PAP^T =$

	1	2	3	4	5	6	7	8	9	
	X	X	X	X	0	0	0	0	0	1
	X	X	X	X	X	0	0	0	0	2
	X	X	X	X	X	X	0	0	0	3
	X	X	X	X	X	X	X	0	0	4
	0	X	X	X	X	X	X	X	0	5
	0	0	X	X	X	X	X	X	X	6
	0	0	0	X	X	X	X	X	X	7
	0	0	0	0	X	X	X	X	X	8
	0	0	0	0	0	X	X	X	X	9

$G(PAP^T)$:

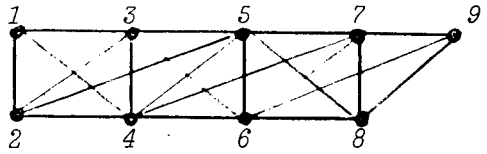
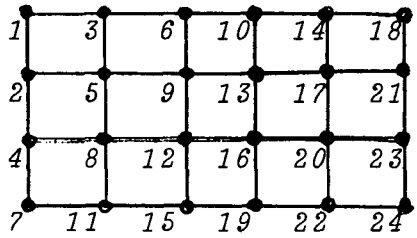
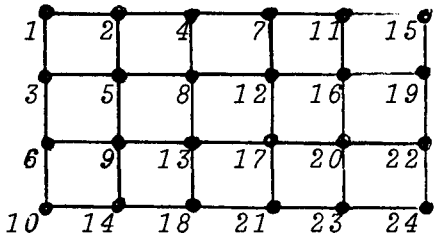


FIGURE 4: Matrices in Band Diagonal Form with no Zeros in the Bands

The two possible numberings generated for a 4 X 6 rectangular network.

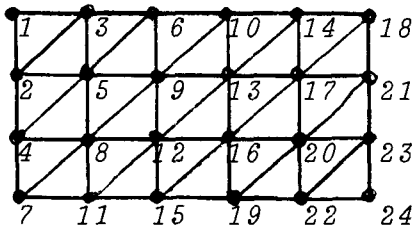


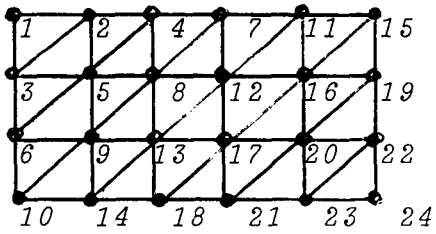


Bandwidth: 4 *Bandwidth: 5*

FIGURE 5A

For the corresponding triangular network:



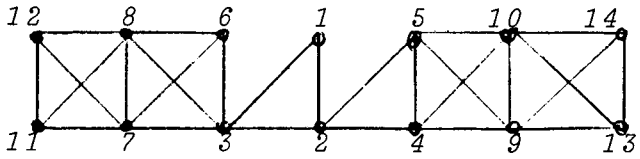


Bandwidth: 4 *Bandwidth: 5*

FIGURE 5B

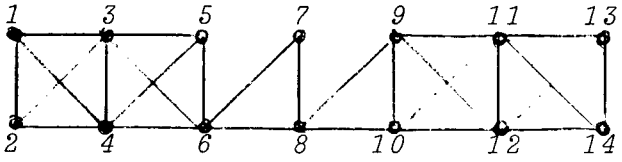
FIGURE 5: Our Numbering Scheme for Rectangular and Triangular Networks

Starting with node of lowest degree:



Bandwidth: 5

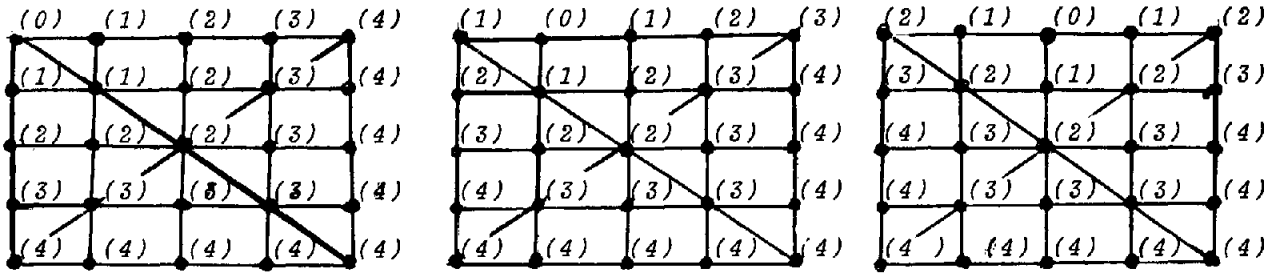
A better numbering scheme:



Bandwidth: 3

Figure 6: Example showing that to get optimum numbering scheme one should not always start at node of lowest degree.

Start with one node of lowest degree; all generated spanning trees given by our scheme have a maximum of 9 nodes per level. Minimum bandwidth with such a numbering scheme is 9. Numbers in parentheses indicate distances from node (0).



Start with five connected nodes of lowest degree; spanning tree generated by our scheme then has a maximum of 5 nodes per level. Numbering below then leads to a bandwidth of 6.

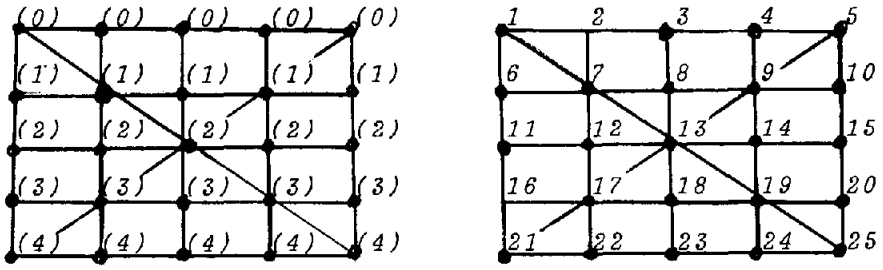


Figure 7 : Example showing that one should sometimes start with several connected nodes of lowest degree.

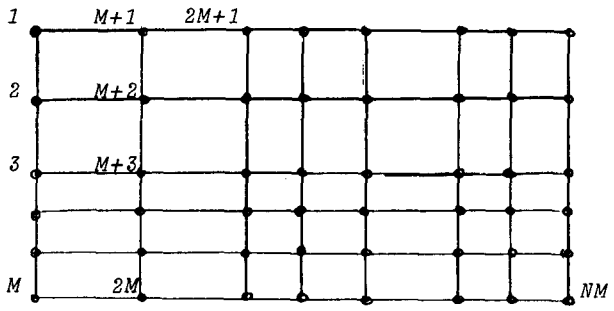
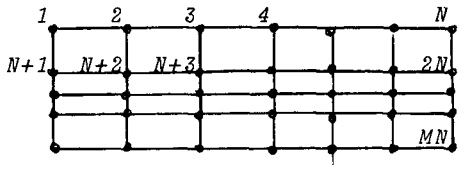
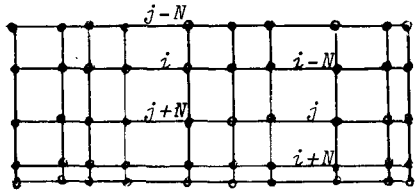


Figure 8A



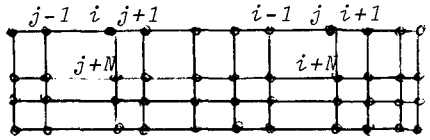
$N > M > 2$

FIGURE 8B



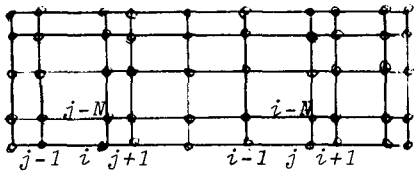
$j \neq i$ so that
$max((i+N)-j, j-(i-N)) > N$

FIGURE 8C



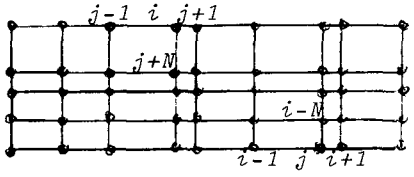
$i > j$ so that
$(i+N)-j > N$

FIGURE 8D



$i > j$ so that
$i-(j-N) > N$

FIGURE 8E



$i > j+N+1$ so that
$i-j \pm 1 > N$

FIGURE 8F

FIGURE 8 : Interchanging two node labels leads to increase in bandwidth.

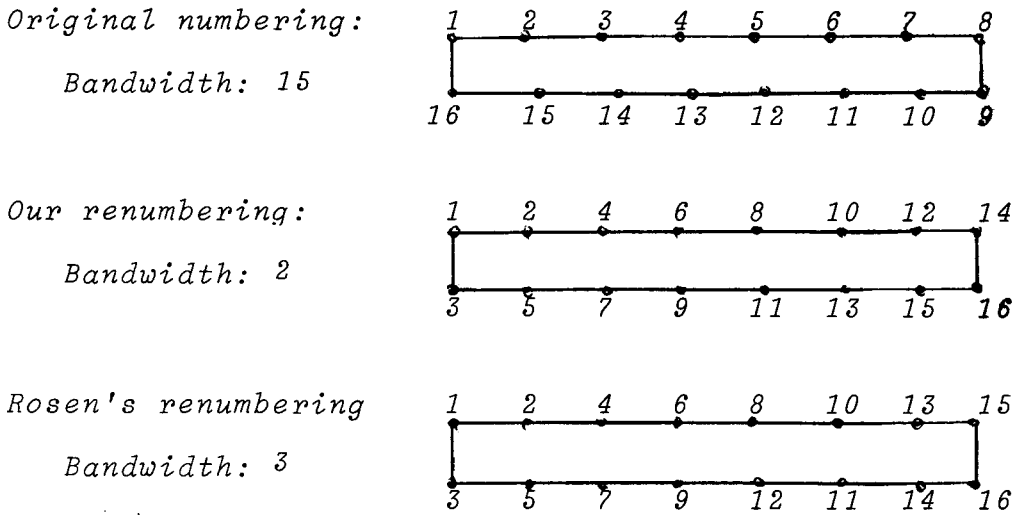


Figure 9: Problem 1. Circle

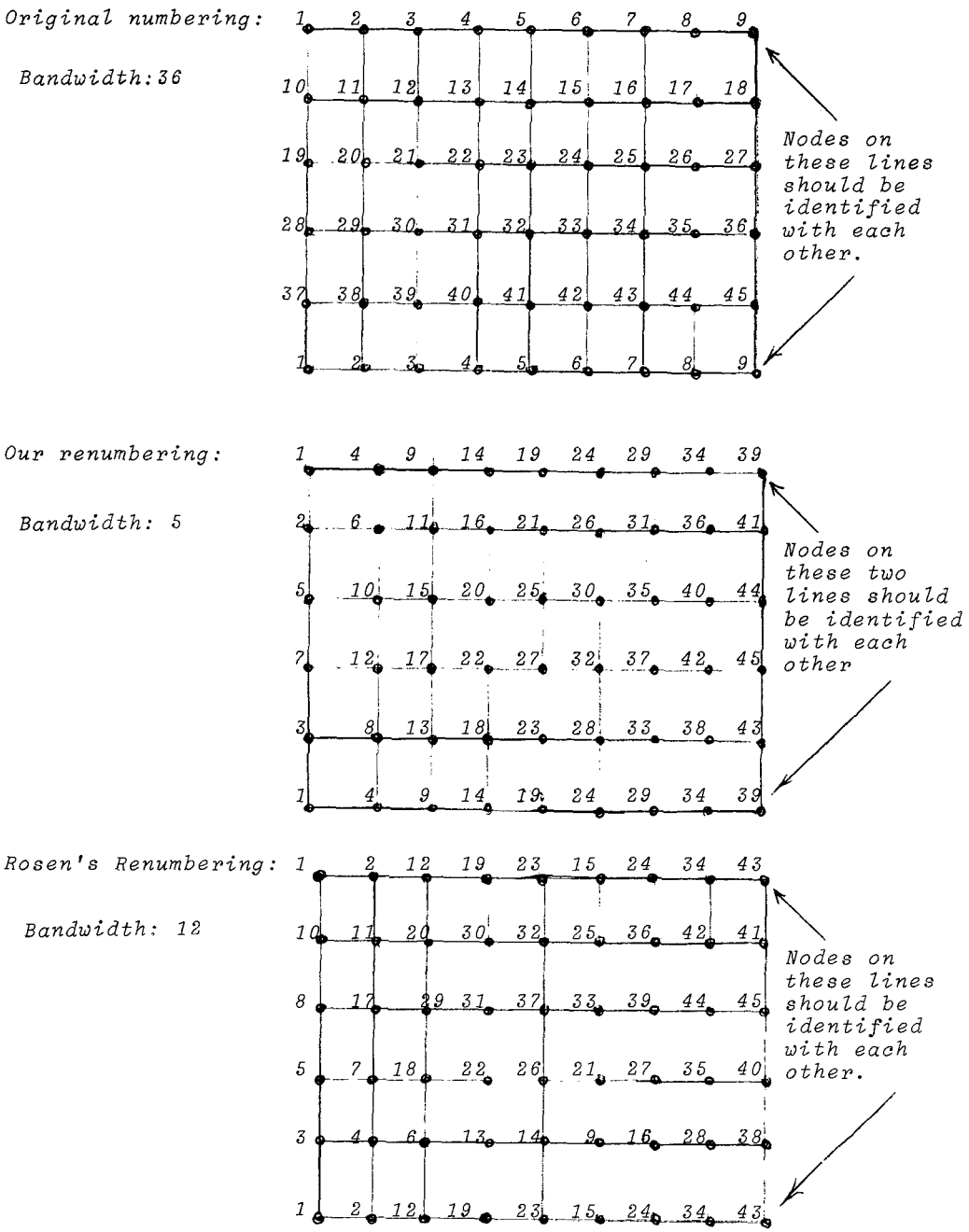


Figure 10:Problem 2. Cylinder

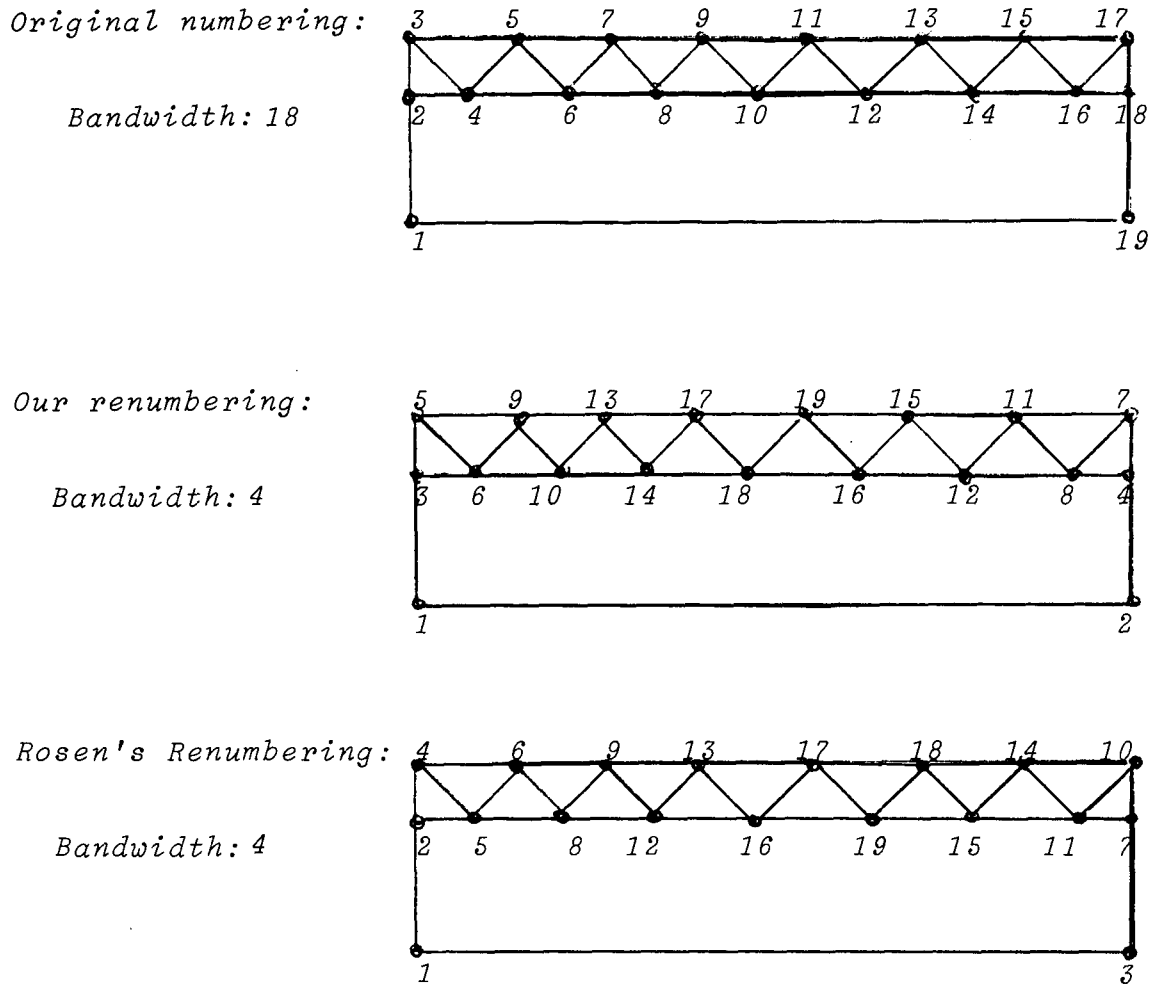
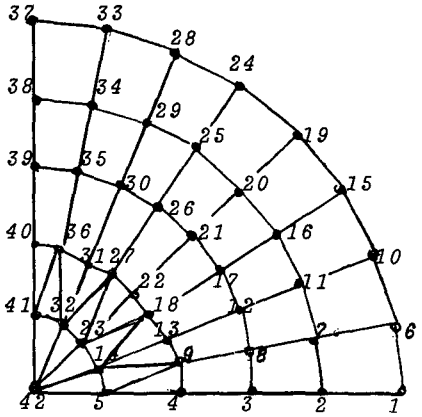


Figure 11. Problem 3. Truss with extras

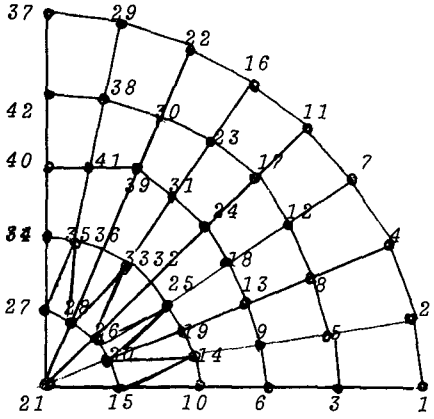
Original Numbering:

Bandwidth: 37



Our Numbering:

Bandwidth: 9



Rosen's Numbering:

Bandwidth: 9

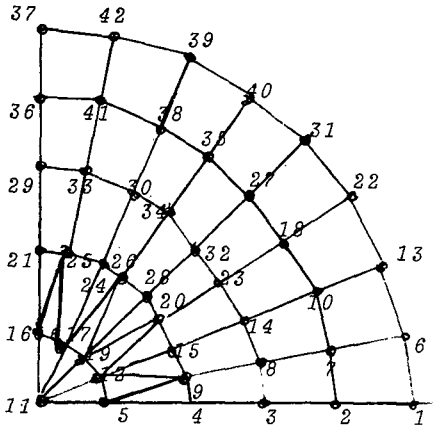


Figure 12: Problem 4. Quarter disc.