N72-32212

NASA TECHNICAL MEMORANDUM

NASA TM X-2660

NASA TM X-2660

RELABELING OF FINITE-ELEMENT MESHES USING A RANDOM PROCESS

by Ernest Roberts, Jr.

Lewis Research Center

Cleveland, Ohio 44135

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • OCTOBER 1972

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM X-2660		

4. Title and Subtitle	5. Report Date
RELABELING OF FINITE-ELEMENT MESHES USING A RANDOM PROCESS	October 1972
	6. Performing Organization Code
7. Author(s) Ernest Roberts, Jr.	8. Performing Organization Report No. E-7001
9. Performing Organization Name and Address Lewis Research Center National Aeronautics and Space Administration Cleveland, Ohio 44135	10. Work Unit No. 501-21
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	13. Type of Report and Period Covered Technical Memorandum
	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

An algorithm is presented to relabel automatically the nodes of an arbitrary finite-element mesh. The purpose of such relabeling is to reduce the bandwidth of the master stiffness matrix produced by the finite-element method. The algorithm uses a random process for the relabeling. Computing time is reduced substantially, compared to systematic methods.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Alogrithms Finite-element method Computer programs Random variables	Unclassified - unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	22	$3.00

* For sale by the National Technical Information Service, Springfield, Virginia 22151

RELABELING OF FINITE-ELEMENT MESHES USING A RANDOM PROCESS

by Ernest Roberts, Jr.

Lewis Research Center

SUMMARY

An algorithm is presented to relabel automatically the nodes of an arbitrary finite-element mesh. The purpose of such relabeling is to reduce the bandwidth of the master stiffness matrix produced by the finite-element method. The algorithm uses a random process for the relabeling. Computing time is reduced substantially, compared to systematic methods.

INTRODUCTION

The finite-element method has become a commonplace tool in structural analysis. A structure is merely modeled by a series of simple elements connected at a number of nodes. An example is a flat plate, represented by many small triangles connected at their corners. The advantage of the finite element method is that very complicated overall behavior may be approximated by much simpler local behavior. If the local regions are chosen small enough, and their properties are chosen wisely enough, in the aggregate they closely simulate the original. Mathematically speaking, a difficult problem in the calculus of variations is replaced by an equivalent system of algebraic equations. The latter are quite conveniently handled by computers.

Equations can be written in terms of either the displacements of the nodes representing a structure or the forces acting on those nodes. The displacement formulation is most frequently used. The displacement method offers a number of advantages for many problems. One of them is a relatively modest demand of computer resources. The master stiffness matrix produced possesses a number of desirable attributes: for example, positive definiteness, sparseness, and bandedness. These, in turn, reduce storage requirements, processing time, and roundoff errors. A further consequence is that an elaborate algorithm is not required for matrix decomposition. This implies that minimum programming effort is required for, at least, the matrix operations.

In spite of this, for many practical problems, the number of degrees of freedom necessary to simulate a structure taxes the resources of a computer system. Every effort must be made when developing a program to reduce storage requirements and processing time, not only for the sake of economy but also to reduce roundoff errors. It is known that the manner in which the various nodes of the model are labeled influences the matrix bandwidth. Hence, work is necessary to produce an algorithm for automatically labeling the nodes of a model to minimize bandwidth. Akyuz et al. (ref. 1) have reported a scheme which is at least partially effective. Barlow et al. (ref. 2) have discussed the limitations of the method and have suggested improvements. The question dealt with in this discussion concerns the possibility of eliminating some of the limitations with a minimum investment of programming effort.

Barlow's most serious criticism of Akyuz's algorithm is that there is no assurance of producing a minimum bandwidth. The method produces a true minimum only when the coupling between elements is simple. When the coupling is more complicated, the bandwidth may be reduced but not necessarily minimized. He states further that the interchange of single pairs of variables, which is the basis of Akyuz's algorithm, does not ensure monotonic reduction. In his reply to Barlow, Akyuz (ref. 3) suggests that a disturbance in the computational stream will restart the process should it stop before a minimum is attained. Such a disturbance may be introduced by arbitrarily interchanging a single pair of variables; this process causes no conflict with the philosophy of the algorithm.

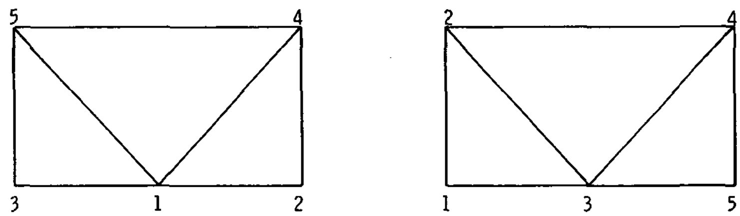
Akyuz's suggestion provides the motivation for this research. Namely, if Akyuz's basic algorithm is used, will a disturbance produce a minimum, and can the algorithm be increased in speed? It appears the answer to the first question is no, and the answer to the second is yes. Furthermore, one of Akyuz's assumptions is questionable because of the research results. (This point is covered more fully in the RESULTS section.) It must be emphasized, however, that his method is effective, in that it reduces the bandwidth of an arbitrary stiffness matrix.

After the conclusion of this work, another reference appeared. It is included in the list of references (ref. 6) without comment for the sake of completeness.

THE ALGORITHM

A characteristic of the algorithm is an artifice, called by Akyuz the connectivity matrix, which is directly related to the master stiffness matrix. The basic scheme is quite simple. One starts with a finite-element representation of a structure. All the nodes are labeled in some arbitrary manner. One merely interchanges pairs of labels in an effort to reduce the bandwidth of the master stiffness matrix. However, it is undesirable to interchange label pairs arbitrarily - bandwidth might increase instead of decreasing. One needs some intermediate criterion to decide if a given interchange should be made without actually deriving the master stiffness matrix. The connectivity matrix provides such a criterion, and it requires little storage space but, unfortunately, some elaborate coding.

Figure 1 illustrates the relation between a finite-element representation and its connectivity matrix. The elements of the connectivity matrix are either 0 or 1. Each row of the connectivity matrix represents one node of the finite-element model. Each element of the row (i.e., each column position) represents each of the other nodes of the model. If the node in question is connected to some other given node, a 1 is placed in that column position. Otherwise, a 0 is placed there. As shown in figure 1(a),



1	1	1	1	1
1	1	0	1	0
1	0	1	0	1
1	1	0	1	1
1	0	1	1	1

(a) Original.

1	1	1	0	0
1	1	1	1	0
1	1	1	1	1
0	1	1	1	1
0	0	1	1	1

(b) Relabeled.

Figure 1. - Example of connectivity matrix.

row 3 of the matrix represents node 3 of the model. Node 3 is connected to nodes 1 and 5 and, by definition, to itself. Therefore, column positions 1, 3, and 5 of row 3 contain 1's, and column positions 2 and 4 contain 0's.

The nonzero elements of the connectivity matrix occupy exactly the same position as the nonzero elements of the master stiffness matrix.[1] Furthermore, in figure 1(b) it can be seen that a simple relabeling of the finite-element model produces a banded connectivity matrix. The 0's in the upper right and lower left of the matrix need not be stored. Finally, the operation is commutative. If the rows and columns of the connec-

[1]However, it must be remembered that each node usually has two or more degrees of freedom associated with it (e.g., two orthogonal displacement components, certain derivatives, etc.). Therefore, there are at least twice as many rows and columns in the master stiffness matrix as in the connectivity matrix. If the equations are ordered properly, however, the ensuing comments are still valid.

tivity matrix are interchanged to produce a banded matrix, a new labeling scheme can be derived from it. For example, if rows 1 and 3 and columns 1 and 3 are interchanged, then nodal labels 1 and 3 on the model are interchanged.

We return to the previous comment that the connectivity matrix requires little storage space but some elaborate coding. Inasmuch as the elements of the connectivity matrix are always either 0 or 1, it is apparent that only a single binary digit is necessary to represent each element. Therefore, many elements may be stored in each computer word. However, normally only whole words (or at least bytes) of computer memory are accessable by the standard instructions. Hence, subprograms must be written to access and manipulate the individual bits of each word. And, of course, many machine cycles are required for each reference to a specific bit.

Nevertheless, the programming problems are minor. If the master stiffness matrix is too large to be contained entirely within the computer core memory and the connectivity matrix is not, it is evident that less elapsed time is required to manipulate the connectivity matrix than the stiffness matrix.[2] Once a connectivity matrix has been derived, and means are available for accessing its elements, we can begin interchanging the various rows and columns for decreasing the bandwidth. One obvious way is to try every possible combination and take the one producing the minimum. But a moment's thought reveals that the number of possible combinations is of the order of the factorial of the order of the matrix. Computing time becomes prohibitive. This being so, some judgment must be exercised. The matrix must be swept through, and a rational criterion must be applied to the decision of interchanging a given pair of rows and columns.

Inspection of figure 1(b) reveals two things. First, the rows have increasing numbers of 0's on the left moving down from the center, and increasing numbers of 0's on the right moving up from the center. Second, the rows have increasing numbers of 1's moving toward the center. It is immediately apparent that these observations may be used as criteria. A pair of rows and columns is examined. If either of the two criteria applies, they are interchanged. If not, another pair is examined. Akyuz merely sweeps through the connectivity matrix in a systematic manner examining each pair of rows and columns in sequence. If an interchange occurs, he starts the process over again. He stops when no more interchanges are possible.

This work deviates in only one major respect from Akyuz. The pairs of rows and columns are chosen randomly instead of systematically. Furthermore, Akyuz's contention (ref. 3), referred to in the INTRODUCTION, that a disturbance in the process will continue the path toward minimization is tested by randomly interchanging a pair of rows and columns after the process has halted. Figure 2 is a flow chart illustrating the algorithm.

---

[2]The repeated input/output operations necessary to process a data store too large for the core memory slow the processing more than the extra instructions to access bits.

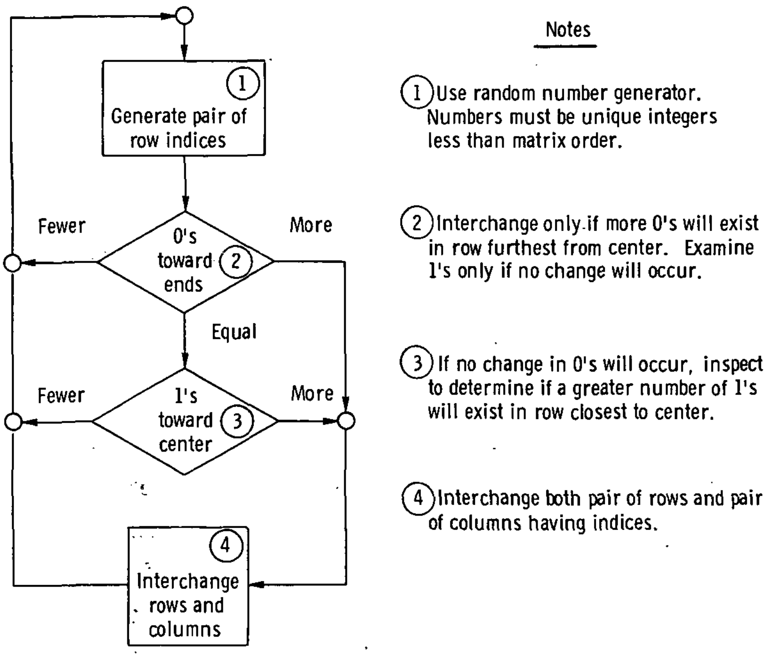


Figure 2. - Schematic of algorithim.

DEFINITIONS

The label difference of an element is the maximum of the absolute values of the differences between the labels of its three nodes. Examining the leftmost element of figure 3(a) shows that there are three possible differences in absolute value: $|1-2|$, $|1-8|$, and $|2-8|$. The greatest of these is 7. The label difference for an entire map is the maximum of the element label differences. In figure 3(a) it is also seen that the label differences for the remaining five elements are 6, 5, 4, 3, and 2, going from left to right. Therefore, the label difference for the map of figure 3(a) is defined to be 7. The element label differences for the mesh of figure 3(b) are 2, 2, 3, 3, 2, and 2, going from left to right, and 3 is the overall label difference.

For the purposes of this report, it is sufficient to consider only the bandwidth of the connectivity matrix. When that is at a minimum, the bandwidth of the master stiffness matrix is at a minimum. For the definition of matrix bandwidth, it is helpful to imagine every row of the matrix, one by one. Count the inclusive number of elements between the first and last nonzero elements of the row. The maximum of these counts, over all the rows of the matrix, is the bandwidth.

A moment's reflection reveals that there is a minimum value for the matrix bandwidth. This minimum occurs when there are no zero elements between the first and last nonzero elements in a certain row. That certain row is the one containing the

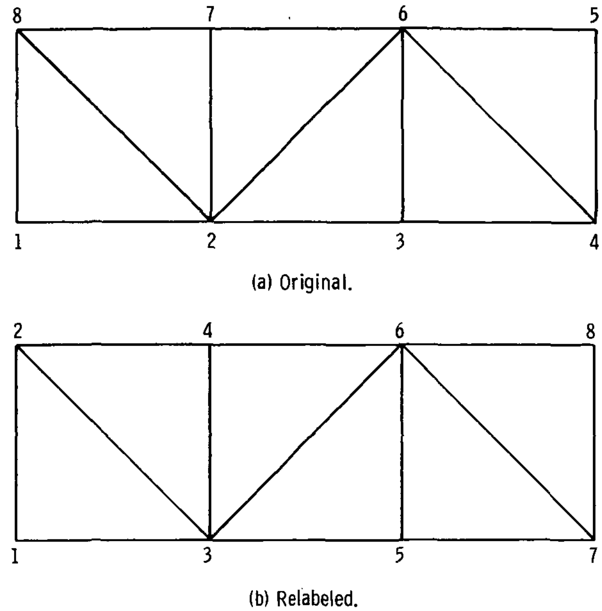


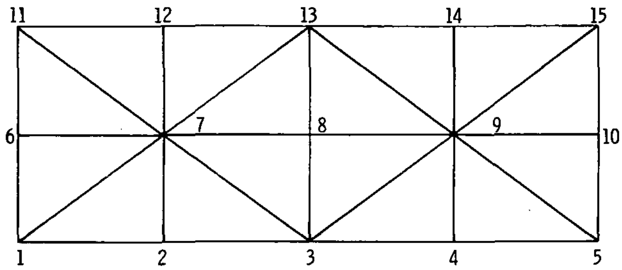
Figure 3. - Grid for first test.

greatest number of nonzero elements. From the definition of the connectivity matrix, that row is the one representing the node connected to the greatest number of other nodes. In figure 3(a) nodes 2 and 6 share that property. Each is connected to 6 nodes, including itself. That number is referred to as the maximum connectivity in this report.

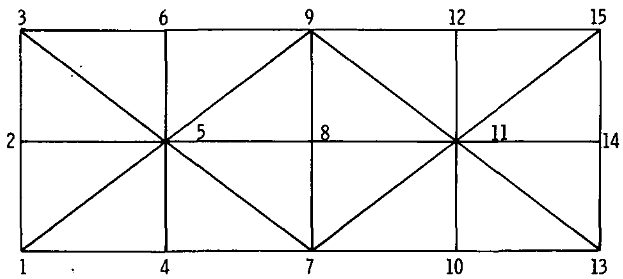
THE EXPERIMENT

The three finite-element models tested are shown in figures 3 to 5. Figure 5 does not show all the details of a very elaborate model. The model shown in figure 5 was used in an application of a finite-element program (ref. 4). There were 369 elements and 218 nodes. For each model an arbitrary labeling system was used as a starting point. In the case of figure 5, three initial labeling schemes were investigated. First, the nodes were labeled systematically in the order of increasing ordinates within increasing abscissas. Second, the initial labels were selected by a random number generator. Finally, the initial labels were determined manually in such a manner as to minimize the difference between labels for each element. Akyuz considers this a reasonable measure of matrix bandwidth, and on the surface it certainly appears to be so.

In each case the algorithm of figure 2 was used in an attempt to reduce the bandwidth of the connectivity matrix. For figure 5, a comparison was made with Akyuz's method. The method of selecting the pairs of nodes was modified to take them in order instead of randomly. The computer used was an IBM 360/67 operating under 360/TSS.

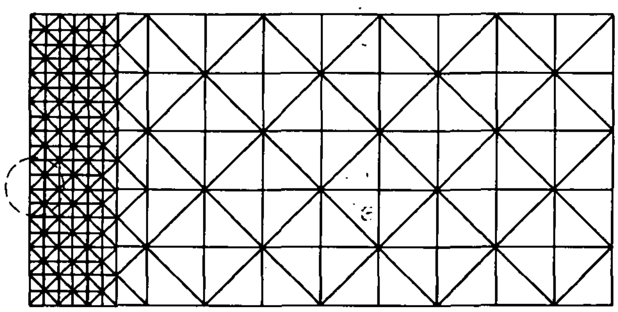


(a) Original.

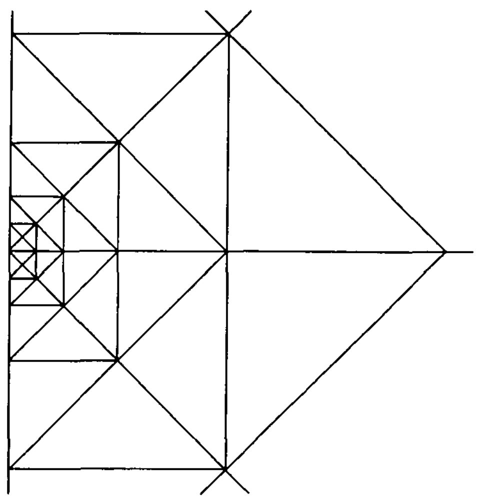


(b) Relabeled.

Figure 4. - Grid for second test.



(a) Overall grid.



(b) Detail near crack tip.

Figure 5. - Grid for third test.

In addition, the model of figure 5 was run on another computer using Akyuz's program as published in reference 1.

RESULTS

For figure 3 the minimum possible bandwidth was achieved. Before this occurred, 243 trials resulting in 12 interchanges were made, costing 0.6 second of processing time. Table I summarizes the process. An entry is made in the table each time one of the criteria is satisfied and an interchange results. It should be noted that a given interchange does not necessarily reduce either the maximum label difference or the bandwidth. Furthermore, the label difference may be reduced without reducing the bandwidth. Finally, even when the minima on both label difference and bandwidth are achieved after 139 tries, the criteria have not yet been satisfied. To satisfy the criteria 104 more tries are made, resulting in two more interchanges. This implies that a substantial reduction in processing time would be possible if a better criterion were available.

TABLE I. - SUMMARY OF COMPUTATIONAL PROCESS FOR FIRST TEST

[Initial label difference, 7; initial band width, 8; maximum connectivity, 6.]

Total tries	Current tries	Label difference	Band width
37	37	6	8
44	7	6	↓
58	14	6	↓
76	18	5	↓
81	5	↓	↓
94	13	↓	↓
95	1	↓	↓
96	1	4	6
132	36	4	↓
139	7	3	↓
242	103	3	↓
243	1	3	↓

For figure 4 the minimum was not attained. In fact, figure 4(b), which was derived manually, shows a better arrangement than figure 4(a). It was impossible to derive the arrangement in figure 4(b) using the algorithm. In fact, the algorithm was useless. The matrix was completely scanned, using both the random and systematic methods, and no interchange satisfied the various criteria. When a disturbance was introduced by arbitrarily interchanging two rows and columns, the process began. However, the best that could be attained was a labeling scheme which produced no narrower a connectivity matrix than the starting point. After a second disturbance, a poorer matrix was produced. Finally, after two more disturbances, no further progress could be made, and the computational stream halted with the poor bandwidth of interruption two. Table II summarizes the stream.

For figure 5 a positive result was produced. Less time was required for the random method than the systematic method for each of the three initial conditions. However, the results were disappointing.

Where the original labels were chosen systematically, the initial label difference was 49 and the initial bandwidth was 72. No improvement resulted from either the random program or the systematic program. The random program made 54 interchanges, using 208 seconds of processing time, before it could no longer satisfy the criteria for interchanging. The systematic program made 627 interchanges, using a total of 2463 seconds of processing time. At this point it was stopped arbitrarily, inasmuch as it had made 109 357 attempts at improvement.

Where the initial labels were chosen by a random number generator, the initial label difference was 209 and the initial bandwidth was 210. Admittedly, this is an unrealistic starting point, and it is certainly a difficult test for any such computer program. The random method reduced the label difference to 64 and the bandwidth to 88, using 3991 seconds of processing time, before it could no longer satisfy the criteria for interchanging. It made a total of 1114 interchanges. The systematic method could not handle the problem. It made 372 interchanges, using 2420 seconds of processing time, before failing to satisfy the criteria for interchange. At the end of that time, the label difference was 206 and the bandwidth 209.

Where the initial labels were chosen manually, the initial label difference was 25, and the initial bandwidth was 43. The random method took 138 seconds of processing time and made 30 interchanges before it could find no further pairs to satisfy all criteria and produce another interchange. The systematic method took 212 seconds and made 29 interchanges. In both cases the label difference was reduced from 25 to 24 but the bandwidth remained at 43. When disturbances were introduced to the computational scheme, the same result occurred as for figure 4. Interestingly enough, after the first disturbance, the label difference decreased to 23, but the bandwidth increased to 45. That is the question raised in the INTRODUCTION. Akyuz assumes that a decrease in label difference corresponds to a decrease in matrix bandwidth. This anomaly brings that assumption into question.

TABLE II. - SUMMARY OF COMPUTATIONAL PROCESS FOR SECOND TEST

[Initial label difference, 6; initial band width, 13; maximum connectivity, 9.]

Total tries	Current tries	Label difference	Band width
Arbitrary interchange			
419	420	11	13
436	17	11	↓
456	20	9	↓
468	12	8	↓
469	1	7	↓
472	3	7	↓
492	20	7	↓
530	38	6	↓
Arbitrary interchange			
949	420	14	15
962	13	10	↓
965	3	↓	↓
987	22	↓	↓
989	2	↓	↓
1078	89	↓	↓
1150	72	9	↓
1155	5	9	↓
Arbitrary interchange			
1574	420	10	15
1576	2	10	↓
1584	8	10	↓
1613	29	9	↓
1615	2	↓	↓
1648	33	↓	↓
1660	12	↓	↓
1669	9	↓	↓
1705	36	↓	↓
1767	62	↓	↓
1895	128	↓	↓
Arbitrary interchange			
2314	420	9	15
Process halted			

One additional item of information was produced. The program published by Akyuz was run and produced a label difference of 21. Yet the bandwidth remained 43. Unfortunately, two different computers were used so a time comparison is not possible.

DISCUSSION

One point to consider is the desirability of using a computer program to decrease matrix bandwidth. It has become apparent that this relabeling program is expensive. It is conceivable that the sum of the times to execute two programs would exceed the execution time required by a poorly labeled mesh. Furthermore, certain manual techniques in common use produce a reasonable labeling scheme at the outset.

Certainly, if a given mesh is going to be used only once, the decrease in execution time required by a well-labeled mesh does not justify the use of a relabeling program. However, there are other considerations:

(1) Many meshes are themselves generated by computer programs. Since there is no choice, these must be labeled automatically.

(2) Most meshes are used repeatedly, especially in solving elasto-plastic problems. These problems traditionally require incremental or iterative processes. In such cases, the sum of the processing times saved per increment can easily exceed the execution time of the relabeling program.

(3) There is also the philosophy of using computers to perform routine clerical tasks. Manual labeling techniques require much time of experienced technical personnel. More efficient use of their time results from using any device such as automatic labeling programs.

CONCLUSIONS

The following conclusions are inescapable based on the results obtained from the experiments:

1. The relabeling scheme developed by Akyuz and modified here is path dependent. The end point depends not only on the starting point but on the order in which interchanges are made. This conclusion is substantiated by the fact that the program written by Akyuz produces a different result from the program written by this author, following Akyuz's algorithm. The obvious reason is that the rows of the connectivity matrix were scanned in a different order.

2. Interrupting the computational stream by arbitrarily interchanging a pair of rows and columns did not produce an improvement in the experiments conducted.

3. Selecting the pairs of variables randomly instead of systematically reduces computation time.

These conclusions imply that a reasonable computer program exists to perform the task of node relabeling. The program simplifies the task of producing the necessary input data for any finite-element program. Although it does not ensure the absolute minimum matrix bandwidth, it does produce a narrow one. The task of producing a minimum width stiffness matrix is a formidable one and is essentially a research program in itself. The appendix contains a source program listing in FORTRAN IV for the IBM system TSS/360. The subroutine for generating random numbers is taken from reference 5.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, August 21, 1972,
501-21.

```
C     RELABELS AN ARBITRARILY LABELED FINITE ELEMENT MESH
C
C**************************************************************************
C*                                                                       *
C*    INPUT                                                              *
C*                                                                       *
C*    3 RECORDS REQUIRED                                                 *
C*                                                                       *
C*         A)  NUMBER OF NODES, NUMBER OF ELEMENTS (FORMAT 901)          *
C*         B)  TABLE OF NODE LABELS FOR EACH ELEMENT, COUNTER-CLOCKWISE  *
C*             IN ORDER OF ELEMENT LABELS (FORMAT 902)                   *
C*         C)  TABLE OF COORDINATES OF EACH NODE, X AND Y, IN ORDER      *
C*             OF NODE LABELS (FORMAT 911)                               *
C*                                                                       *
C**************************************************************************
C
      COMMON /  ONE/ ORDER,NELMNT,LAST,SCALE,WORD,LIMIT,HALF1,HALF2,MAX,      -
     1               BAND,WHICH(2),PLUS,UNUSED,DIFF                           -
     2       /  TWO/ RIGHT(1024),LEFT(1024),ONES(1024)                        -
     3       /THREE/ ELEMNT(3,682)                                            -
     4       / FOUR/ MATRIX(32,1024)                                          -
     5       / FIVE/ CRDNAT(2,512)
      INTEGER ORDER,WORD,HALF1,HALF2,BAND,WHICH,PLUS,UNUSED,RIGHT,ONES,       -
     1        ELEMNT,TOTAL,CURENT,TIME1,TIME2,DIFF,XY,CRDNAT
C
C      ***INITIALIZE CLOCK, READ INPUT***
C
      CALL CPUTIM(TIME1)
      READ (5,901) ORDER,NELMNT
      READ (5,902) ((ELEMNT(NODE,LEMNT),NODE=1,3),LEMNT=1,NELMNT)
      READ (5,911) ((CRDNAT(XY,NODE),XY=1,2),NODE=1,ORDER)
C
C     ***INITIALIZE VARIABLES FOR USE IN LATER SUBROUTINES***
C
      CALL NITILZ
      LIM12=3*LIMIT
C
C     ***FORM INITIAL CONNECTIVITY MATRIX***
C
      CALL CONECT
C
C     ***GENERATE VECTORS OF COUNT OF NULL ELEMENTS***
C     *********ON RIGHT AND LEFT OF EACH ROW**********
C
      CALL GNRATE
C
C     ***GENERATE VECTOR OF COUNT OF NON-ZERO ELEMENTS IN EACH ROW***
C
      CALL NUMBER
C
C     ***CALCULATE INITIAL BAND-WIDTH AND LABEL DIFFERENCE***
C
      CALL WIDTH
C
C     ***PRINT HEADINGS FOR TABLE***
C
      WRITE (6,905) DIFF,BAND,MAX
      WRITE (6,903)
      NBAND=0
      NDIFF=0
```

```
      TOTAL=0
      NTRCNG=0
 10   CURENT=0
 20   CURENT=CURENT+1
C
C     ***EXIT IF CURRENT NUMBER OF ATTEMPTS IS TOO LARGE***
C
      IF (CURENT.GE.LIMIT) GO TO 40
      TOTAL=TOTAL+1
C
C     ***EXIT IF TOTAL NUMBER OF ATTEMPTS IS TOO LARGE***
C
      IF (TOTAL.EQ.LIM12) GO TO 50
C
C     ***RANDOMLY SELECT TWO DIFFERENT ROWS AND COLUMNS FOR INTERCHANGE***
C
      CALL CHOOSE
C
C     ***WILL NUMBER OF ZEROS AWAY FROM CENTER INCREASE?***
C
      CALL CRTRN1(&20,&30)
C
C     ***WILL NUMBER OF ONES NEAR CENTER INCREASE?***
C
      CALL CRTRN2(&20)
C
C     ***IF SO, INTERCHANGE ROWS AND COLUMNS SELECTED BY 'CHOOSE'***
C
 30   CALL SWITCH
C
C     ***RECALCULATE BAND-WIDTH AND LABEL DIFFERENCE***
C
      CALL WIDTH
      NTRCNG=NTRCNG+1
C
C     ***IF BANDWIDTH OR CONNECTIVITY DECREASE,***
C     ******BREAKPOINT AND PRINT TABLE ENTRY******
C
      IF (NBAND.EQ.BAND.AND.NDIFF.EQ.DIFF) GO TO 35
      WRITE (6,904) NTRCNG,TOTAL,CURENT,DIFF,BAND
      NBAND=BAND
      NDIFF=DIFF
      WRITE (7,902) ((ELEMNT(NODE,LEMNT),NODE=1,3),LEMNT=1,NELMNT)
      WRITE (7,911) ((CRDNAT(XY,NODE),XY=1,2),NODE=1,ORDER)
      WRITE (7,901) WHICH
C
C     ***PROTECT AND REWIND BREAKPOINT FILE***
C
      CALL SYSOBF(16,' CLOSE ,,FT07F001')
C
C     ***EXIT IF BANDWIDTH EQUALS CONNECTIVITY***
C
 35   CALL DONE(&10)
      WRITE (6,906)
C
C     ***READ CLOCK, PRINT EXECUTION TIME***
C
 38   CALL CPUTIM(TIME2)
      TIME=(TIME2-TIME1)/1000.
      WRITE (6,909) TIME
      STOP
 40   WRITE (6,908) LIMIT
      GO TO 38
```

```
 50   WRITE (6,910) LIM12
      GO TO 38
C
C     ***FORMATS***
C
 901  FORMAT (2I5)
 902  FORMAT (8(3I3,1X))
 903  FORMAT ('0','INTERCHANGE',3X,'TOTAL',3X,'CURRENT',3X,'LABEL',4X,          -
     1         'BAND'/' ',14X,2('TRIES',4X),' DIFF',3X,'WIDTH'/' ')
 904  FORMAT (' ',I11,I8,I10,2I8)
 905  FORMAT ('1','INITIAL LABEL DIFFERENCE =',I4,5X,'INITAL BAND WIDTH'       -
     1         ,' =',I4,5X,'MAXIMUM CONNECTIVITY =' I3)
 906  FORMAT ('OMINIMUM BAND-WIDTH HAS BEEN ACHIEVED***PROGRAM HALTED')
 908  FORMAT ('0','PROGRAM HALTED AFTER',I7,1X,'UNSUCCESSFUL ATTEMPTS ',       -
     1         'AT IMPROVEMENT')
 909  FORMAT (' ','EXECUTION TIME =',F6.1,1X,'SECONDS')
 910  FORMAT ('0','PROGRAM HALTED AFTER',I9,1X,'TOTAL ATTEMPTS ',              -
     1         'AT IMPROVEMENT')
 911  FORMAT (20I4)
 912  FORMAT (F5.1,19F4.1)
      END
```

```
      SUBROUTINE NITILZ
      COMMON /ONE/ ORDER,DUMY1,LAST,SCALE,WORD,LIMIT,HALF1,HALF2,DUMY2(4),PLUS,  -
     1              UNUSED,DUMY3
      INTEGER ORDER,WORD,HALF1,HALF2,PLUS,UNUSED
      LAST=1
      SCALE=ORDER*0.4656613E-9
      WORD=(ORDER-1)/32+1
      PLUS=WORD+1
      UNUSED=WORD*32-ORDER+1
      LIMIT=2*ORDER*(ORDER-1)
      HALF1=ORDER/2
      HALF2=HALF1+1
      IF (MOD(ORDER,2).EQ.1) HALF1=HALF2
      RETURN
      END
```

```
      SUBROUTINE CONECT
      COMMON /  ONE/ ORDER,NELMNT,DUMY(13)                                     -
     1        /THREE/ ELEMNT(3,682)                                            -
     2        / FOUR/ MATRIX(32,1024)
      INTEGER ORDER,ELEMNT,COLUMN,BIT,ROW
      DO 50 ROW=1,ORDER
      COLUMN=(ROW-1)/32+1
      BIT=MOD(ROW,32)
      IF (BIT.EQ.0) BIT=32
      CALL PUT (MATRIX(COLUMN,ROW),BIT,1)
          DO 40 LEMNT=1,NELMNT
                DO 10 NODE=1,3
                IF (ROW.EQ.ELEMNT(NODE,LEMNT)) GO TO 20
 10             CONTINUE
          GO TO 40
```

```
20            DO 30 NODE=1,3
              IF (ROW.EQ.ELEMNT(NODE,LEMNT)) GO TO 30
              COLUMN=(ELEMNT(NODE,LEMNT)-1)/32+1
              BIT=MOD(ELEMNT(NODE,LEMNT),32)
              IF (BIT.EQ.0) BIT=32
              CALL PUT (MATRIX(COLUMN,ROW),BIT,1)
30            CONTINUE
40        CONTINUE
50    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GNRATE
      COMMON /  ONE/ ORDER,DUMY1(14)
     1       /  TWO/ RIGHT(1024),LEFT(1024),DUMY2(1024)
      INTEGER ORDER,RIGHT,COUNT
      DO 10  K=1,ORDER
      RIGHT(K)= COUNT(K)
      LEFT(K)=KOUNT(K)
10    CONTINUE
      RETURN
      END
```

```
      INTEGER FUNCTION COUNT(ROW)
      COMMON /  ONE/ ORDER,DUMY1(3),WORD,DUMY2(7),PLUS,UNUSED,DUMY3
     1       / FOUR/ MATRIX(32,1024)
      INTEGER ROW,ORDER,WORD,PLUS,UNUSED,COLUMN,BIT
      PLUS= WORD+1
      DO 10  J=1,WORD
      COLUMN= PLUS-J
      DO 10  I=1,32
      BIT= 33-I
      IF(LOOK(MATRIX(COLUMN,ROW),BIT).EQ.1)  GO TO 20
10    CONTINUE
20    COUNT= (J-1) * 32+I-UNUSED
      RETURN
      END
```

```
      INTEGER FUNCTION KOUNT(ROW)
      COMMON /  ONE/ ORDER,DUMY1(3),WORD,DUMY2(10)
     1       / FOUR/ MATRIX(32,1024)
      INTEGER ROW,ORDER,WORD,COLUMN,BIT
      DO 10  COLUMN=1,WORD
      DO 10  BIT=1,32
      IF(LOOK(MATRIX(COLUMN,ROW),BIT).EQ.1)  GO TO 20
10    CONTINUE
20    KOUNT=(COLUMN-1)*32+BIT-1
      RETURN
      END
```

```
      SUBROUTINE NUMBER
      COMMON / ONE/ ORDER,DUMY1(3),WORD,DUMY2(3),MAX,DUMY3(6)          -
     1       / TWO/ DUMY4(2048),ONES(1024)                             -
     2       /FOUR/ MATRIX(32,1024)
      INTEGER ORDER,WORD,ONES,ROW,COLUMN,BIT
      MAX=0
      DO 20 ROW=1,ORDER
      ONES(ROW)=0
      DO 10 COLUMN=1,WORD
      DO 10 BIT=1,32
      IF (LOOK(MATRIX(COLUMN,ROW),BIT).EQ.1) ONES(ROW)=ONES(ROW)+1
10    CONTINUE
      MAX=MAXO(MAX,ONES(ROW))
20    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE WIDTH
      COMMON /  ONE/ ORDER,NELMNT,DUMY1(7),BAND,WHICH(2),DUMY2(2),DIFF   -
     1       /  TWO/ RIGHT(1024),LEFT(1024),DUMY3(1024)                  -
     2       /THREE/ ELEMNT(3,682)
      INTEGER ORDER,BAND,DIFF,LEFT,RIGHT,ROW,ELEMNT
      DIFF=0
      BAND=0
      DO 100  LEMNT=1,NELMNT
      DIFF=MAXO(DIFF,MAXO(IABS(ELEMNT(1,LEMNT)-ELEMNT(2,LEMNT)),       -
     1                    IABS(ELEMNT(1,LEMNT)-ELEMNT(3,LEMNT)),       -
     2                    IABS(ELEMNT(2,LEMNT)-ELEMNT(3,LEMNT))))
100   CONTINUE
      DO 200 ROW=1,ORDER
      BAND=MAXO(BAND,(ORDER-(LEFT(ROW)+RIGHT(ROW))))
200   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHOOSE
      COMMON / ONE/ DUMY1(2),LAST,SCALE,DUMY2(6),WHICH1,WHICH2,DUMY3(3)
      INTEGER WHICH1,WHICH2,SAVE
      LAST= LAST*65539
      IF(LAST) 10,20,20
10    LAST= LAST+2147483647+1
20    WHICH1= LAST*SCALE+0.5
      IF (WHICH1.EQ.0) WHICH1=1
25    LAST= LAST*65539
      IF(LAST) 30,40,40
30    LAST= LAST+2147483647+1
40    WHICH2= LAST*SCALE+0.5
      IF (WHICH2.EQ.0) WHICH2=1
      IF (WHICH2-WHICH1) 50,25,60
50    SAVE=WHICH2
      WHICH2=WHICH1
      WHICH1=SAVE
60    RETURN
      END
```

```
SUBROUTINE CRTRN1(*,*)
COMMON /ONE/ DUMY1(10),WHICH1,WHICH2,DUMY2(3)                      -
1       /TWO/ RIGHT(1024),LEFT(1024),DUMY3(1024)
INTEGER WHICH1,WHICH2,RIGHT
LOGICAL TEST
TEST=.FALSE.
IF (RIGHT(WHICH1)-RIGHT(WHICH2)) 20,10,100
10   TEST=.TRUE.
20   IF (LEFT(WHICH2)-LEFT(WHICH1)) 40,30,100
30   IF (TEST) RETURN
40   RETURN 2
100  RETURN 1
END
```

```
SUBROUTINE CRTRN2(*)
COMMON /ONE/ DUMY1(6),HALF1,HALF2,DUMY2(2),WHICH1,WHICH2,DUMY3(3) -
1       /TWO/ DUMY4(2048),ONES(1024)
INTEGER HALF1,HALF2,WHICH1,WHICH2,ONES,CENTR1,CENTR2
LOGICAL TEST1,TEST2
TEST1=.FALSE.
TEST2=.FALSE.
IF (ONES(WHICH1)-ONES(WHICH2)) 20,50,10
10   TEST1=.TRUE.
20   IF (WHICH1.GE.HALF2) GO TO 40
IF (WHICH2.LE.HALF1) GO TO 30
CENTR1=IABS(HALF1-WHICH1)
CENTR2=IABS(HALF2-WHICH2)
IF (CENTR1-CENTR2) 40,50,30
30   TEST2=.TRUE.
40   IF (TEST1.AND.TEST2) RETURN
IF (.NOT.TEST1.AND..NOT.TEST2) RETURN
50   RETURN 1
END
```

```
SUBROUTINE SWITCH
COMMON /  ONE/ ORDER,NELMNT,DUMY1(2),WORD,DUMY2(5),WHICH1,WHICH2,DUMY3(3)  -
1       /  TWO/ RIGHT(1024),LEFT(1024),ONES(1024)                          -
2       /THREE/ ELEMNT(3,682)                                              -
3       / FOUR/ MATRIX(32,1024)                                            -
4       / FIVE/ CRDNAT(2,512)
INTEGER ORDER,WORD,WHICH1,WHICH2,RIGHT,ONES,ELEMNT,WORD1,WORD2,BIT1,BIT2,  -
1        SAVE
DO 10 I=1,WORD
SAVE=MATRIX(I,WHICH1)
MATRIX(I,WHICH1)=MATRIX(I,WHICH2)
MATRIX(I,WHICH2)=SAVE
10   CONTINUE
WORD1=(WHICH1-1)/32+1
WORD2=(WHICH2-1)/32+1
BIT1=MOD(WHICH1,32)
IF (BIT1.EQ.0) BIT1=32
BIT2=MOD(WHICH2,32)
IF (BIT2.EQ.0) BIT2=32
DO 20 I=1,ORDER
```

```
      SAVE=LOOK(MATRIX(WORD1,I),BIT1)
      CALL PUT(MATRIX(WORD1,I),BIT1,LOOK(MATRIX(WORD2,I),BIT2))
      CALL PUT(MATRIX(WORD2,I),BIT2,SAVE)
20    CONTINUE
      CALL GNRATE
      SAVE=ONES(WHICH1)
      ONES(WHICH1)=ONES(WHICH2)
      ONES(WHICH2)=SAVE
      DO 50 LEMNT=1,NELMNT
      DO 50 NODE=1,3
      IF (ELEMNT(NODE,LEMNT).EQ.WHICH1) GO TO 30
      IF (ELEMNT(NODE,LEMNT).EQ.WHICH2) GO TO 40
      GO TO 50
30    ELEMNT(NODE,LEMNT)=WHICH2
      GO TO 50
40    ELEMNT(NODE,LEMNT)=WHICH1
50    CONTINUE
      HOLDX=CRDNAT(1,WHICH1)
      HOLDY=CRDNAT(2,WHICH1)
      CRDNAT(1,WHICH1)=CRDNAT(1,WHICH2)
      CRDNAT(2,WHICH1)=CRDNAT(2,WHICH2)
      CRDNAT(1,WHICH2)=HOLDX
      CRDNAT(2,WHICH2)=HOLDY
      RETURN
      END
```

```
      SUBROUTINE DONE(*)
      COMMON /ONE/ ORDER,DUMY1(14)
     1       /TWO/ RIGHT(1024),LEFT(1024),DUMY2(1024)
      INTEGER ORDER,RIGHT,ROW
      DO 10 ROW=2,ORDER
      IF (LEFT(ROW).LT.LEFT(ROW-1)) RETURN 1
      IF (RIGHT(ROW).GT.RIGHT(ROW-1)) RETURN 1
10    CONTINUE
      RETURN
      END
```

```
LOOKP    PSECT
SAVE     DS    19F
         ENTRY LOOK
         ENTRY PUT
ADDR     DC    A(LOOK)
BIT      DC    F'0'
IW       DC    F'0'
MASK     DC    X'7FFFFFFF'
MASK1    DC    X'80000000'
LOOKC    CSECT
LOOK     STM   14,12,12(13)      LONG SAVE
         L     14,72(0,13)
         ST    14,8(0,13)       STORE FWD POINTER
         ST    13,4(0,14)       STORE BWD POINTER
         LR    13,14            COVER OUR PSECT
         USING LOOKP,13
         USING LOOKC,12
```

```
         L      12,ADDR          COVER OUR CSECT
*        FUNCTION LOOK  ( A, I)
*                                  REGISTER 1 POINTS TO PARAMETERS
         LM     2,3,0(1)           GET ADDRS OF BOTH ARGS
         L      2,0(0,2)              GET VALUE OF FIRST ARG
         L      3,0(0,3)         GET VALUE OF 2D ARG
         LA     4,1              SET GPR 4 TO A ONE
         SR     3,4              REDUCE BIT POSITION BY ONE
         ST     3,BIT            SAVE BIT POSITION
*         MASK AND TEST
SHIFT    SLL    2,0(3)             NO. OF BITS IS IN GPR 3
         L      6,MASK1
         NR     2,6
         LTR    2,2
         LA     5,0
         BZ     ZERO             BRANCH IF ZERO
         LA     5,1
*         PUT ANSWER IN GPR 0, THIS IS A FUNCTION ENTRY
ZERO     LR     0,5                  PUT RESULT INTO 0
         B      RTN
PUT      STM    14,12,12(13)       LONG SAVE
         L      14,72(0,13)
         ST     14,8(0,13)        FWD PTR
         ST     13,4(0,14)        BWD PTR
         LR     13,14
         L      12,ADDR
*         SUBROUTINE PUT   (A, I, N)
         LM     2,4,0(1)           PICK UP ADDRS OF ALL 3 ARGS
         L     5,0(0,2)            GET THE WORD OF BITS
         L     4,0(0,4)            GET VALUE OF BIT TO BE STORED
         L      3,0(0,3)             GET WHICH BIT
         LTR    4,4           IS IT A ZERO OR A ONE
         BZ     ZEROO
*     HE WANTS TO STORE A ONE BIT
         LA     6,32(0)       PUT A 32 IN REG 6
         SR     6,3           SUB BIT POSITION FROM 32
         SLL    4,0(6)         SHIFT THE ONE TO POSITION
         OR     5,4           LOGICAL OR BIT INTO POSITION
         B      MEET        FINISHED
*    HE WANTS TO ZERO THAT BIT
ZEROO    SR     6,6           ZERO REG 6
         L      7,MASK                =X'7FFFFFFF'
         LA     8,33(0)       PUT A 33 INTO REG 8
         SR     8,3           SUBTR BIT POSITION FROM 33
         SLDL   6,0(8)        SHIFT MISSING BIT TO POSITION
         OR     7,6           PUT THE MASK BACK TOGETHER
         NR     5,7           AND THE MISSING BIT INTO WORD A
MEET     ST     5,0(0,2)        PASS BACK TO CALLER
*        COMBINED RETURN
RTN       L      13,4(0,13)         RESTORE GPR 13
         LM     14,15,12(13)        RESTORE THESE TWO
         LM     1,12,24(13)         RESTORE THE REST (SAVE GPR 0)
         BCR    15,14                EXIT
         END
```

REFERENCES

1. Akyuz, Fevzican A.; and Utku, Senol: An Automatic Node-Relabeling Scheme for Bandwidth Minimization of Stiffness Matrices. AIAA J., vol. 6, no. 4, Apr. 1968, pp. 728-730.
2. Barlow, John; and Marples, Christopher G.: Comment on "An Automatic Node-Relabeling Scheme for Bandwidth Minimization of Stiffness Matrices." AIAA J. vol. 7, no. 2, Feb. 1969, pp. 380-381.
3. Akyuz, Fevzican A.: Reply by Author to J. Barlow and C. G. Marples. AIAA J., vol. 7, no. 2, Feb. 1969, pp. 381-382.
4. Srawley, J. E.; Swedlow, J. L.; and Roberts, E., Jr.: On the Sharpness of Cracks Compared With Wells's COD. Int. J. Fracture Mech., vol. 6, Dec. 1970, pp. 441-444.
5. Anon.: IBM System/360 Scientific Subroutine Package, (360A-CM-03X) Vers. III, H20-0205-3, Programmer's Manual.
6. Grooms, H. R.: Algorithm for Matrix Bandwidth Reduction. J. Structural Div. Am. Soc. Civil Eng., vol. 98, 1972, pp. 203-214.

NASA-Langley, 1972 —— 8 E-7001

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Scientific and technical information generated under a NASA contract or grant and considered an important contribution to existing knowledge.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

TECHNOLOGY UTILIZATION PUBLICATIONS: Information on technology used by NASA that may be of particular interest in commercial and other non-aerospace applications. Publications include Tech Briefs, Technology Utilization Reports and Technology Surveys.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION OFFICE

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546