NOAA Technical Memorandum NOS NGS-4

REDUCING THE PROFILE OF SPARSE SYMMETRIC MATRICES

Richard A. Snay

National Geodetic Survey
Rockville, Md.
June 1976

reprinted 1979

UNITED STATES
DEPARTMENT OF COMMERCE
Elliot L. Richardson, Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION
Robert M. White, Administrator

National Ocean
Survey
Allen L. Powell, Director

CONTENTS

REDUCING THE PROFILE OF SPARSE SYMMETRIC MATRICES*

Richard A. Snay
National Geodetic Survey
National Ocean Survey, NOAA, Rockville, MD

ABSTRACT. An algorithm for improving the profile of a sparse symmetric matrix is introduced. Tests on normal equation matrices encountered in adjustments of geodetic networks by least squares demonstrate that the algorithm produces significantly lower profiles than the widely used reverse Cuthill-McKee algorithm.

I. INTRODUCTION

Let $Ax = b$ be a system of linear equations where A is a sparse, symmetric, positive definite matrix. Such systems are generated, for example, by the least-squares method in geodesy and by the finite element displacement method in structural analysis. To expedite an automated solution of the system, a data structure is desired which will avoid the storage of a significant number of the zero matrix elements in A and which will allow trivial arithmetic operations to be circumvented by program logic. For many systems, this is effectively accomplished by the variable band data structure described by Jennings (1967).

The variable band data structure can be viewed as a modification of the band data structure. Given a symmetric matrix $A = (a_{ij})$ of order n, define the column height p_j of column j $(1 \le j \le n)$ to equal 0, if $a_{ij} = 0$ for $1 \le i \le j$; otherwise,

---

*A shortened version of this paper has been submitted for publication in Bulletin Géodésique.

p_j equals j-i, where i is the smallest integer such that $a_{ij} \neq 0$. Figure 1 depicts the value of p_j as the length of a spike. With the band data structure, one stores and manipulates only the elements a_{ij} of A, for which $0 \leq j-i \leq B$; here $B = \max_j p_j$ is called the bandwidth of A. In contrast, with the variable band data structure, one stores and manipulates only the elements a_{ij} of A for which $0 \leq j-i \leq p_j$, i.e., the diagonal and spikes of figure 1. The storage allocation of the variable band data structure is a function of the column profile P of the matrix A, where P equals $\Sigma_j p_j$.

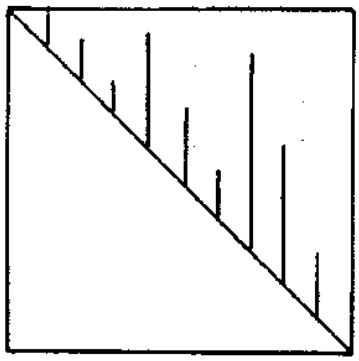

Figure 1.--Required storage of matrix.

To take better advantage of the band data structure, several algorithms (Akyuz and Utku 1968, Alway and Martin 1965, Arany *et al.* 1971, Cheng 1973, Collins 1973, Cuthill and McKee 1969, Gibbs *et al.* 1976, Grooms 1972, King 1970, Rosen 1968) are designed to rearrange the columns and rows of A to translate the original system of equations into an equivalent system A'x' = b', where A' is a symmetric matrix with a smaller bandwidth than that of A. Of these algorithms, the one by Cuthill and McKee (1969) is probably the most widely used. Rearranging the columns and rows of the matrix to reduce column profile

correspondingly allows one to take better advantage of the variable band data structure. Besides requiring less storage, the system with a lower profile requires fewer computations and less computer time, and decreases round-off error.

In this paper, the problem of deciding which rows and columns to permute for approximating minimum column profile, and the corresponding problem for bandwidth, are modeled as versions of a queuing problem. Heuristically, this model justifies the use of the Cuthill-McKee algorithm for reducing bandwidth. However, for reducing column profile, the model suggests a different algorithm. The description of this algorithm and the results of several experiments follow the construction of the model.

II. GRAPH ANALOGY

A symmetric matrix $A = (a_{ij})$ of order n defines an undirected graph G whose vertexes are numbered 1, 2, ..., n. (See figure 2.) A connection or edge exists between vertexes i and j ($i \neq j$) if, and only if, $a_{ij} \neq 0$. In this case, it is said that i sees j or j sees i. A renumbering f (a one-to-one function $f: \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$) of the vertexes of G defines a rearrangement of the rows and columns of A, producing a symmetric matrix $A' = (a'_{ij})$ satisfying the relation $a'_{ij} = a_{f(i)f(j)}$.

Given a numbering of the graph G, define the column height P_j at vertex j or $1 \leq j \leq n$ as follows: If vertex j does not see any vertex with a lower number, then $p_j = 0$; otherwise, $p_j = j-i$ where i is the lowest numbered vertex seen by j. The column profile P of G defined by this numbering is given by the equation $P = \sum_{j=1}^{n} p_j$. The bandwidth B of G is given by the equation $B = \max_j p_j$.

A renumbering f of the vertexes of G produces a column profile P_f and a bandwidth B_f. The problem of minimizing column profile, respectively bandwidth, of a system of linear equations is thus equivalent to finding a renumbering f such that P_f, respectively B_f, is minimal.

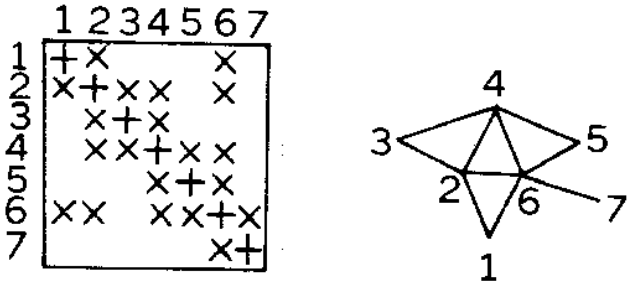

Figure 2.--Associated graph of matrix.

III. BANKER'S DILEMMA

Light is shed on the profile minimization problem if it is viewed as a queuing or waiting problem. Consider a multi-millionaire who wishes to witness the effect of giving away free land to each of the families in a small village. However, as the millionaire believes in the Shakespearian tenet, "neither a borrower nor a lender be," he adds the stipulation that before the villager can receive the land, the villager must settle his financial accounts with the other citizens of the village. Having divided the land into plots, the millionaire employs a banker to settle the accounts and distribute the plots accordingly. The banker calls in one of the families of the village together with all the families which are either indebted to this family or to which this family is indebted. The accounts are settled and this first family chooses the plot which they prefer. It takes a whole day to settle the papers. The next day, in addition to the family which the

banker has scheduled to process and the neighbors which have some bond of obligation with this family, there are at the bank the families who were at the bank on the first day and who had not received a plot of land. The banker serves the family he has scheduled.

Each day the banker can serve only one family, and each day at the bank there are the family which the banker scheduled, the families which have a bond of obligation with this scheduled family, and the families which have previously been to the bank without having received any land. This last group of families continuously returns to the bank hoping to influence the banker to serve them, so that they might be able to choose the next choicest plot of land. For this reason, the banker denotes this group as the hopeful families. Since the bank's image will suffer each day a family waits at the bank, the banker decides to schedule the families in an order which will minimize the total waiting time of all the families.

Let the families correspond to the vertexes of a graph G. An edge or connection exists between two vertexes if a bond of obligation exists between the families. The order in which the banker serves the families defines a numbering of the vertexes. Family j does not wait if they had no obligation with the families served before them. On the other hand, if family j had an obligation to a family served before them, then family j waits j-i days, where i is the first family served to have an obligation with family j. Hence, the column height p_j of the vertex j is the waiting time in days of family j. The column profile P of G associated with this numbering is the total waiting time of the families.

IV. THE PROFILE ALGORITHM

Suppose the banker has served the first k families, and suppose there are h hopeful families, i.e., families who had a bond of obligation to at least one of the first k families and who have not yet received a plot of land. At this time, the banker decides to serve a family i for which r_i is a minimum, where r_i is the number of hopeful families who would return to the bank tomorrow if family i is served today. One sees that $r_i = h + m_i - n_i$, where m_i is the number of nonhopeful families obligated to family i who have not yet been to the bank. The parameter $n_i = 1$ if family i is a hopeful family; otherwise $n_i = 0$. When the corresponding matrix is rearranged according to the order in which the families are served, then the minimized quantity r_i equals the number of off-diagonal elements of the k + 1 row in the new matrix which must be stored, i.e., r_i equals the change in column profile realized by assigning family i the number k + 1.

A path of length n in a graph G is a sequence of vertexes $[v_1, v_2, \ldots, v_{n+1}]$ of G such that v_i sees v_{i+1} for $1 \leq i \leq n$. G is said to be connected if each pair of vertexes of G is contained in some path. A maximal connected subgraph of G is called a component. The algorithm is presented below in terms of numbering the vertexes of a graph, when the graph has one component. The discussion for obtaining a starting vertex for the algorithm is postponed momentarily.

Algorithm 1: To number vertexes of a graph for near minimal column profile.

1. Pick a starting vertex v; number it 1.

2. Each unnumbered vertex seen by v is denoted as a hopeful vertex until it is numbered.

3. All vertexes seen by v or seen by vertexes which are seen by v are added to the list of candidates unless:

   a. they are already numbered, or

   b. they are presently on the list of candidates.

4. If the list of candidates is empty, then stop; all the vertexes in this component are numbered.

5. If the first k vertexes are numbered, then choose vertex k + 1 to be any vertex v from the list of candidates for which the value $m_v - n_v$ is a minimum. m_v equals the number of unnumbered vertexes seen by v which are not presently hopeful vertexes. n_v equals 1 if v is a hopeful vertex; n_v equals 0 otherwise.

6. Vertex v is subtracted from the list of candidates and is no longer denoted as a hopeful vertex.

7. Return to step 2.

If the graph has several components, then having sequenced the components arbitrarily, the vertexes of the first component are numbered as above. If the vertexes of the first s components are numbered, then the starting vertex v of component s + 1 is assigned the lowest unused number, and the algorithm proceeds as step 2.

Note that, in practice, if the first k vertexes of a component are numbered, then not every unnumbered vertex is a candidate for the number k + 1. For the efficiency of the algorithm, the list of candidates consists only of the hopeful vertexes and the unnumbered vertexes seen by hopeful vertexes. As the numbering progresses, these vertexes are easily accumulated concurrently with the process of updating the values for the m_v's.

V. FINDING STARTING VERTEXES

Since algorithm 1 is based on local properties of the graph the resulting column profile will be highly dependent on the choice of a starting vertex. Described here is an algorithm which produces a set of vertexes. The column profile is calculated using each vertex in this set as the starting vertex in algorithm 1, and the best result is retained.

The algorithm is described in terms of the distance between vertexes. If u and v are vertexes in a connected graph G, then the distance between u and v is the length of the shortest path from u to v.

Algorithm 2: To find candidates for starting vertexes.

1. Pick a vertex v.

2. Choose a set S of (five) vertexes whose distances from v are at least as great as that of any vertex not in S.

3. Let x be a vertex in S whose distance from v is maximal.

4. Choose a set T of (five) vertexes whose distances from x are at least as great as that of any vertex not in T.

5. The vertexes of S and T are the candidates for the starting vertex.

The above algorithm, which is similar to that proposed by Gibbs, Poole, and Stockmeyer (1976) is motivated by modeling the graph as a village in which all families live on one street, and by assuming that the bonds of obligation can only exist between families who live relatively close together. If this

model is accurate, then it is best to start at one of the ends of the street. Starting with a family not on the end would cause families on one side to wait while families on the other side are being served.

VI. BANDWIDTH ALGORITHM

The Cuthill-McKee algorithm (1969) is stated below for the case when the graph has only one component. The details pertaining to the selection of candidates for the starting vertex are omitted.

Algorithm 3: Cuthill-McKee

1. Pick a starting vertex, number it 1.

2. If the first k vertexes are numbered and vertex w is the lowest numbered of these k vertexes which sees unnumbered vertexes, then choose vertex k + 1 to be any of the unnumbered vertexes seen by w which has the lowest degree. The degree of a vertex v is the number of vertexes seen by v.

That the Cuthill-McKee algorithm produces a small bandwidth, while algorithm 1 produces a large one, can be seen by slightly adjusting the queuing model. If instead of total waiting time, the banker attempts to minimize the longest waiting time of any one family, then he is trying to minimize bandwidth. The Cuthill-McKee algorithm says that among the hopeful families who have waited for the longest time, schedule next any of these families which has the fewest bonds of obligation. In contrast, algorithm 1 completely neglects the length of time the families have already waited, and in doing so, it causes families with many bonds of obligation to wait longer to be served.

VII. EMPIRICAL RESULTS

When automated, algorithms of this paper are designed to be time efficient. As such, only near minimal profile or bandwidth is expected of them. That algorithm 1, hereafter referred to as the "banker's" algorithm, does not always produce the minimal column profile is demonstrated by the graphs of figures 3 and 4. The banker's algorithm was used to number the graph of figure 3, resulting in a column profile of 14. An alternative numbering of the graph, figure 4, yields a column profile of 13.

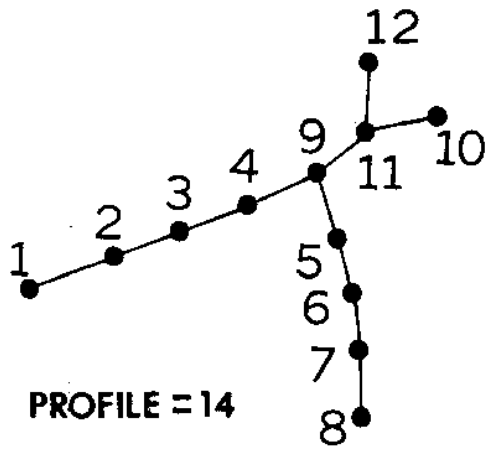

Figure 3.--Numbering by banker's algorithm.

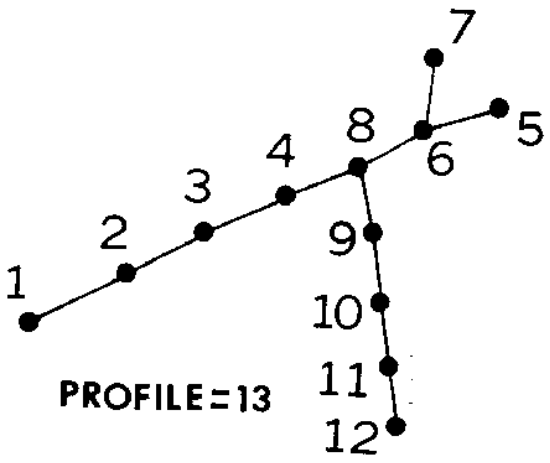

Figure 4.--Alternative numbering.

To measure the effectiveness of the banker's algorithm for reducing column profile, its results are compared with those produced by the reverse Cuthill-McKee algorithm on a collection of sixteen data sets. The reverse Cuthill-McKee algorithm, which simply inverts the numbering of the standard Cuthill-McKee algorithm, is recognized for reducing column profile by

several authors (Cuthill 1971, George 1971, Gibbs *et al.* 1976, Liu and Sherman 1976). Until mid-1975 it was the profile reducing algorithm used by the National Geodetic Survey (NGS) of the National Ocean Survey.

Part of the tested data consisted of five graphs which Dr. Cuthill (1971) used to compare her algorithm with several other algorithms. The remaining eleven data sets correspond to systems of linear equations which were generated in the process of performing least-squares adjustments of geodetic networks existing in the files of the National Geodetic Survey. Figure 5 is a sketch of the network identified as INDI--7-I, and figure 6 pictures the resulting matrices obtained for INDI--7-I after applying each algorithm, where in both cases all vertexes of degree five or less were considered for the starting vertex.

The empirical results, as displayed in table 1, favor the banker's algorithm except in the case when the graph is especially homogeneous, as the second, third, and fourth data sets. The results also demonstrate that the profile of the banker's algorithm can be slightly improved by considering all vertexes of degree five or less for the starting vertex, rather than considering the vertexes generated by algorithm 2. However, the work which is involved in testing many starting vertexes (see column M of table 1) would generally outweigh the benefits of the profile reduction.

Figure 7 is a graph of the information contained in the columns of table 1 identified as L, P_C, and P_B. The quantity L, the number of nonzero terms above the diagonal of the matrix, provides a lower limit for the column profile of the matrix. The true minimum profile is usually impractical to obtain (it can be obtained by considering the n factorial numberings of the associated graph). Although L is presented here as estimate of the true minimum profile, there can be a significant

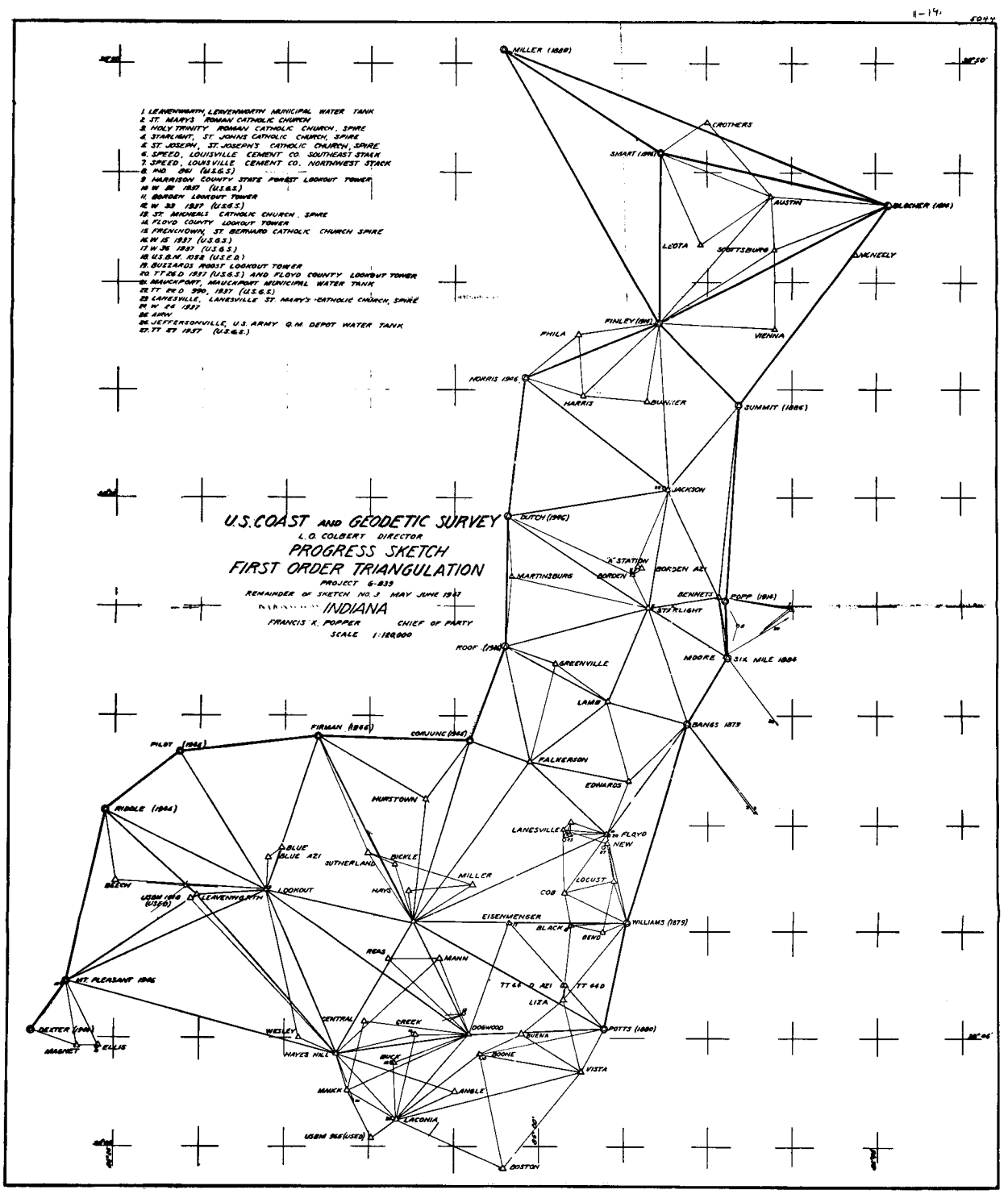

Figure 5.--The geodetic network, INDI--7-I.

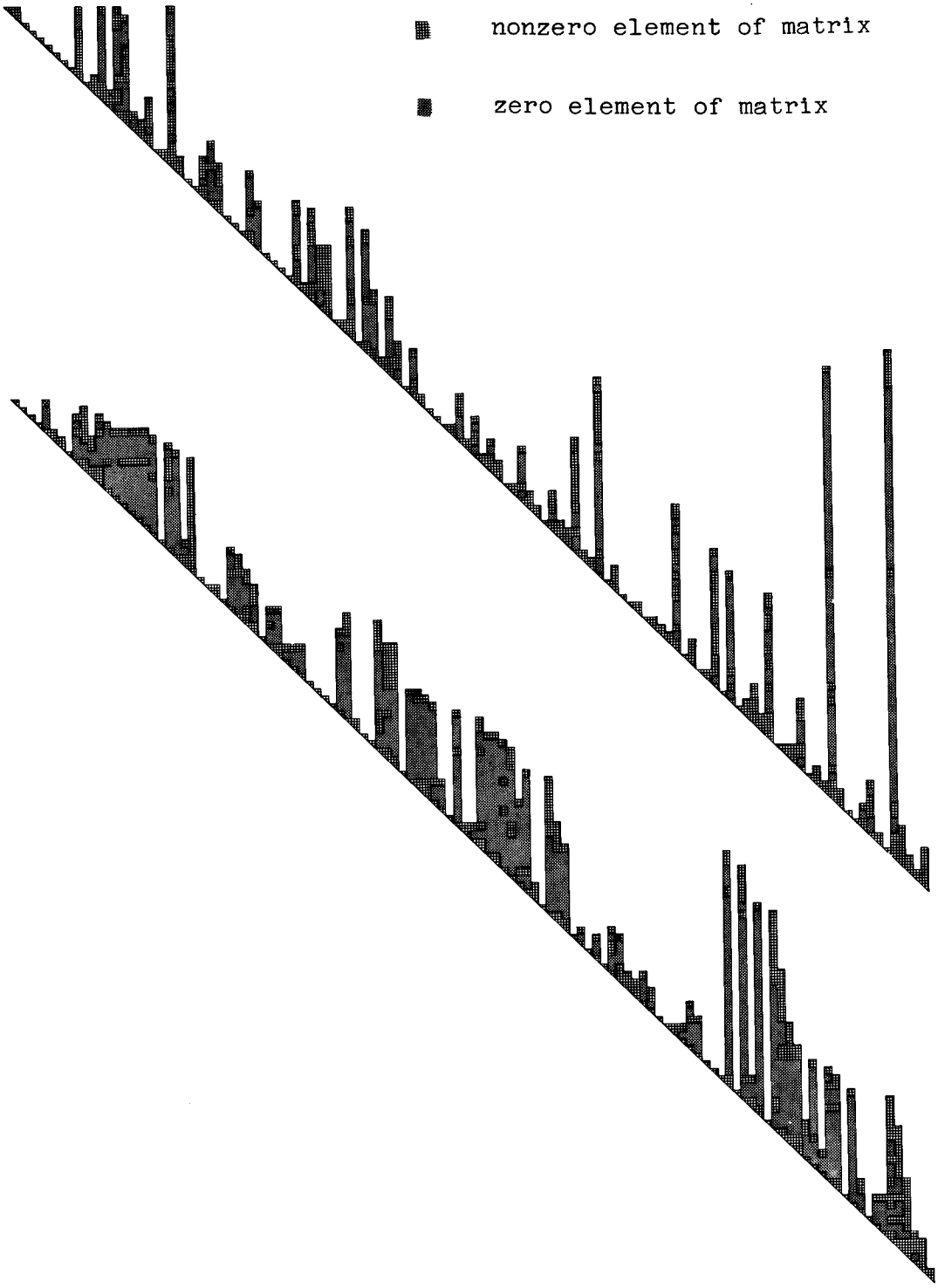

Figure 6.--Required storage for matrix of INDI--7-I, after applying banker's algorithm (top) and reverse Cuthill-McKee algorithm (bottom).

Table 1.--Empirical results.

Data set	N	M	L	P_C	P_B	P_S
Figure 4*	24	24	23	54	30	31
Figure 5A*	64	64	112	364	371	371
Figure 5B*	64	28	161	364	371	371
Figure 6A*	45	45	85	204	204	204
Figure 6B*	42	42	81	211	195	195
BOONFIELD	56	33	148	217	187	187
WESTERNNC	86	55	203	319	278	278
HUNTWILWV	86	61	215	558	430	430
MSHS 11782	92	54	259	411	329	329
INDI--7-I	119	85	281	860	569	591
OHIO2	141	102	393	2083	1012	1024
INDI-7-II	149	98	372	1321	952	971
MTCARMELI	157	96	499	1526	1222	1227
CETVERILL	301	192	916	4380	3042	3083
SALT-LAKE	278	179	844	6903	4173	4487
G15031	464	325	1204	11326	5908	6265
Totals	2168	1483	5796	31101	19273	20044

*E. Cuthill (1971). In this table,

N = order of matrix = number of vertexes in graph.

M = number of vertexes of degree 5 or less.

L = number of nonzero terms above the diagonal of the matrix.

P_C = column profile obtained with reverse Cuthill-McKee algorithm considering each vertex of degree 5 or less for the starting vertex.

P_B = column profile obtained with banker's algorithm considering each vertex of degree 5 or less for the starting vertex.

P_S = column profile obtained with banker's algorithm considering each vertex produced by algorithm 2 for the starting vertex.

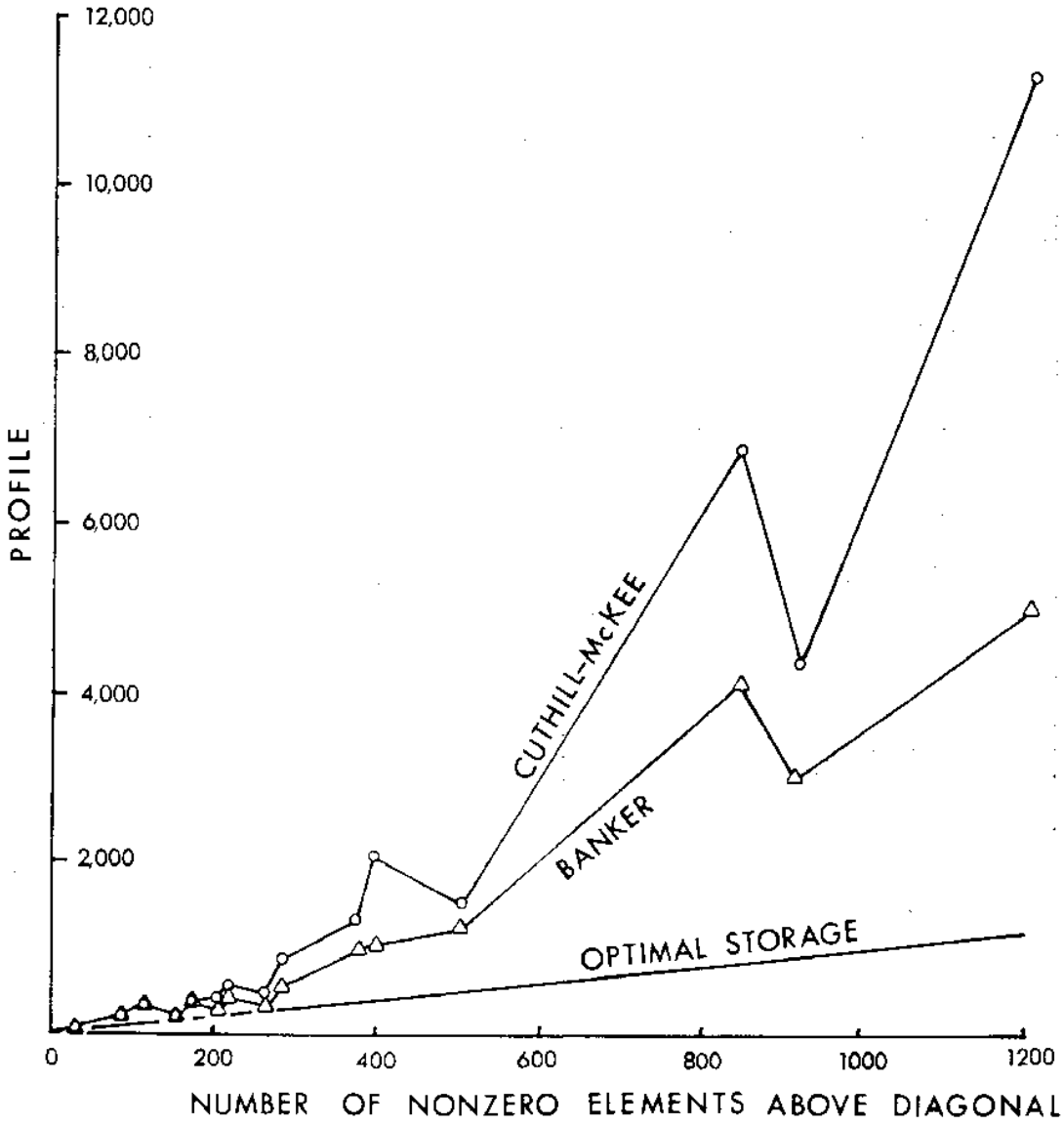

Figure 7.--Empirical results.

difference between these two numbers. For example, consider a graph G constructed by distinguishing n points on a circle as the vertexes. By induction on n, it can be shown that the minimum column profile of G equals 2n-3, while the value of L is n.

VIII. EPILOG

From the experimental results, it is concluded that the banker's algorithm effectively reduces column profile. That this lower profile not only saves computer storage, but also reduces execution time, is demonstrated by the result of the following experiment. An execution time of 71.47 seconds was required to solve the equations generated by the geodetic network, SALT-LAKE (278 stations, 556 unknowns), when the unknowns were ordered randomly. When the unknowns were ordered by the banker's algorithm, the equations solved in 6.02 seconds. The application of the banker's algorithm required 19.66 seconds.

ACKNOWLEDGMENTS

The author wishes to thank E. H. Cuthill and G. C. Everstine of the Naval Ship Research and Development Center for the use of their computer program, BANDIT. The author also wishes to acknowledge R. H. Hanson of NGS who modified BANDIT to accept standard geodetic data.

REFERENCES

Akyuz, F. A. and Utku, S., 1968: An automatic relabeling scheme for bandwidth minimization of stiffness matrices. Journal of the American Institute of Aeronautics and Astronautics, 6, 728-730.

Alway, G. G. and Martin, D. W., 1965: An algorithm for reducing the bandwidth of a matrix of symmetric configuration. The Computer Journal, 8, 264-272.

Arany, I., Smyth, W. F., and Szoda, L., 1971: An improved method for reducing the bandwidth of sparse symmetric matrices. Proceedings of International Federation for Information Processing, 1246-1250.

Cheng, K. Y., 1973: Minimizing the bandwidth of sparse symmetric matrices. Computing, 11, 103-110.

Collins, R. J., 1973: Bandwidth reduction by automatic renumbering. International Journal for Numerical Methods in Engineering, 6, No. 3, 345-356.

Cuthill, E. and McKee, J., 1969: Reducing the bandwidth of sparse symmetric matrices. Proceedings of Association for Computing Machinery, 24th National Conference, August 26-28, 157-172.

Cuthill, E., 1971: Several Strategies for Reducing the Bandwidth of Matrices. Naval Ship Research and Development Center, Bethesda, Maryland. Technical Note CMD-42-71, 19 pp.

George, A., 1971: Computer Implementation of the Finite Element Method. Computer Science Department, Stanford University, Stanford, California. Report No. STAN-CS-71-208.

Gibbs, N. E., Poole, W. G. Jr., and Stockmeyer, P. K., 1976: An algorithm for reducing the bandwidth and profile of a sparse matrix. SIAM Journal of Numerical Analysis, 13, No. 2, 236-250.

Grooms, H. R., 1972: Algorithm for matrix bandwidth reduction. American Society of Civil Engineers, Structural Division Journal, 98, 203-214.

Jennings, A., 1967: A compact storage scheme for the solution of symmetric linear simultaneous equations. The Computer Journal, 9, No. 3, 281-285.

King, I. P., 1970: An automatic reordering scheme for simultaneous equations derived from network systems. International Journal for Numerical Methods in Engineering, 2, No. 4, 523-533.

Liu, W. and Sherman, A. H., 1976: Comparative analysis of the Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices. SIAM Journal of Numerical Analysis, 13, No. 2, 198-213.

Rosen, R., 1968: Matrix bandwidth minimization. Proceedings of Association for Computing Machinery, 23rd National Conference, August 27-29, 585-595.

APPENDIX I. Implementation of the banker's algorithm.

This supplement contains a listing of the Fortran IV subroutine REORD which implements the banker's algorithm. The purpose of the subroutine is to number the vertexes of a graph G to approximate minimum column profile. If G has n vertexes, the subroutine assumes the vertexes are identified in the calling program by the integers 1, 2, ..., n. The subroutine parameters are defined first.

NUMSTA is a variable. On call, NUMSTA contains the number of vertexes of G.

NABOR is an array of dimension n by m, where m is the maximum number of vertexes seen by any one vertex. On call, NABOR contains the connectivity information of G. If vertex i sees 7 vertexes, then the identifiers of these vertexes will be stored in the locations NABOR(i,k) for $1 \le k \le 7$.

KDEG is a vector of length n. On call, KDEG(i) contains the number of vertexes seen by vertex i.

NORD is a vector of length n. On return, NORD(i) contains the new number assigned to vertex i.

IDEG is a vector of length n. Prior to statement number 320, IDEG(i) is used to contain the identifier of a vertex. After statement number 320, IDEG(i) contains the current number of nonhopeful vertexes seen by vertex i.

NAT is a vector of length n. NAT(i) is originally assigned the number 6 for all i. When the component of G containing vertex i is being numbered, then NAT(i) assumes the values 1 through 5.

$$\text{NAT(i)} = \begin{cases} 1, & \text{if vertex i is numbered} \\ 2, & \text{if vertex i is a hopeful vertex} \\ 3, & \text{if vertex i is a nonhopeful vertex on the list of candidates} \end{cases}$$

NWAIT is a vector of length n. NWAIT is used to contain the identifiers of vertexes on the list of candidates.

The program documentation is shown on figure 8, pages 20 through 23.

```
      SUBROUTINE REORD(NABOR,IDEG,KDEG,NORD,NAT,NUMSTA,NWAIT)
C
      DIMENSION NABOR(500,50),IDEG(1),KDEG(1),NORD(1),NAT(1),
     1      NWAIT(1),START(10)
C
C      THIS SUBROUTINE REORDERS THE UNKNOWNS TO ACHIEVE MINIMUM
C      COLUMN PROFILE
C
      INTEGER START
    1 FORMAT (* LOGICAL ERROR IN REORDER *)
      DO 90 I=1,NUMSTA
   90 NAT(I)=6
      KZ=0
      LXZ=0
   95 JPZ=1
      IZ=1
C
C FIND A VERTEX, LXZ, BELONGING TO AN
C UNNUMBERED COMPONENT
C
  100 LXZ=LXZ + 1
      JTRY=NAT(LXZ)
      IF (JTRY.NE.6)GO TO 100
C
C  FIND THE FIVE VERTEXES FURTHEST AWAY FROM THE
C  VERTEX LXZ
C  IZ COUNTS THE NUMBER OF VERTEXES IN THIS COMPONENT
C
      IDEG(1)=LXZ
      NAT(LXZ)=5
      IXZ=LXZ
  125 J6 = KDEG(IXZ)
      IF(J6.NE.0)GO TO 130
C
C THIS IS A COMPONENT WITH JUST ONE VERTEX
C
      KZ = KZ + 1
      NORD(IXZ) = KZ
      IF(KZ.EQ.NUMSTA)RETURN
      GO TO 95
  130 DO 200 K1=1,J6
      NSTA = NABOR(IXZ,K1)
      IF(NAT(NSTA).NE.6)GO TO 200
      IZ=IZ+1
      IDEG(IZ)=NSTA
      NAT(NSTA)=5
  200 CONTINUE
      IF(JPZ.EQ.IZ)GO TO 210
      JPZ=JPZ + 1
      IXZ=IDEG(JPZ)
      GO TO 125
C
C  TRANSFER THE LAST FIVE VERTEXES IN VECTOR IDEG
C  TO VECTOR START
C
```

Figure 8.--Fortran program source listing (page 1 of 4).

```
  210 KSTART = MIN0(5,IZ)
      I1=IZ+1
      DO 220  L=1,KSTART
      L1=I1-L
      START(L)= IDEG(L1)
  220 CONTINUE
C
C  FIND THE FIVE VERTEXES FURTHEST AWAY FROM THE
C  VERTEX START(1)
C
      JPZ=1
      IZ=1
      IDEG(IZ)=START(1)
      IXZ=START(1)
      NAT(IXZ)=4
  225 J6 = KDEG(IXZ)
      DO 300 K1=1,J6
      NSTA = NABOR(IXZ,K1)
      IF(NAT(NSTA).NE.5)GO TO 300
      IZ=IZ+1
      IDEG(IZ)=NSTA
      NAT(NSTA)=4
  300 CONTINUE
      IF(JPZ.EQ.IZ)GO TO 310
      JPZ=JPZ+1
      IXZ=IDEG(JPZ)
      GO TO 225
C
C  TRANSFER THE LAST FIVE VERTEXES OF IDEG TO THE
C  VECTOR START. THE VECTOR START THUS CONTAINS THE
C  TEN VERTEXES WHICH WILL BE TESTED AS THE STARTING
C  VERTEXES IN NUMBERING THE COMPONENT
C
  310 I1=IZ+1
      DO 320 L=1,KSTART
      L1=I1-L
      J=KSTART+L
      START(J)=IDEG(L1)
  320 CONTINUE
      KSTART=KSTART*2
C
C   TRY DIFFERENT STARTING POINTS FOR THIS  COMPONENT
C
      NSTOP=0
      ISTART=1
      NPROF=10000000
  488 DO 980 JQ = ISTART,KSTART
      DO 498 I=1,NUMSTA
      IDEG(I) = KDEG(I)
        NAT(I) = MAX0(4,NAT(I))
  498 CONTINUE
C
C  NCOUNT CONTAINS THE NUMBER OF VERTEXES IN THIS
C  COMPONENT WHICH ARE CURRENTLY NUMBERED.
C  NPRO CONTAINS THE CURRENT COLUMN PROFILE FOR THE
```

Figure 8.--Continued (page 2 of 4).

```
C  NUMBERING OF THIS COMPONENT FOR THE CURRENT
C  STARTING VERTEX
C
      NCOUNT=1
      NSTA = START(JQ)
      NPRO=0
      MWAIT=0
      NAT(NSTA) = 1
      NORD(NSTA) = KZ + 1
  570 JE = KDEG(NSTA)
      DO 760 J=1,JE
      KSTA = NABOR(NSTA,J)
      JTRY=NAT(KSTA)
      GO TO (760,755,690,685),JTRY
  685 MWAIT=MWAIT+1
      NWAIT(MWAIT)=KSTA
  690 NAT(KSTA)=2
      KE = KDEG(KSTA)
      DO 750 K1=1,KE
      MSTA = NABOR(KSTA,K1)
      JTRY=NAT(MSTA)
      GO TO (750,745,745,740),JTRY
  740 NAT(MSTA)=3
      MWAIT=MWAIT+1
      NWAIT(MWAIT) = MSTA
  745 IDEG(MSTA)=IDEG(MSTA)-1
  750 CONTINUE
  755 IDEG(KSTA)=IDEG(KSTA)-1
  760 CONTINUE
  770 IF (MWAIT.GT.1)GO TO 800
      NCOUNT = NCOUNT +1
C
C  THIS IS THE LAST VERTEX IN THIS COMPONENT
C
      NSTA = NWAIT(1)
      NORD(NSTA) = KZ + NCOUNT
      JE = KDEG(NSTA)
      NCX = 0
      DO 780 J=1,JE
      LSTA = NABOR(NSTA,J)
      NCX = MAX0(NCX,NORD(NSTA)-NORD(LSTA))
  780 CONTINUE
      NPRO = NPRO + NCX
      NAT(NSTA) = 1
      MWAIT=0
      GO TO 974
  800 NTEST=1
      NBR1=NWAIT(1)
      DO 835 L=2,MWAIT
      NBR2=NWAIT(L)
      IF(IDEG(NBR2)+NAT(NBR2)-IDEG(NBR1)-NAT(NBR1))825,835,835
  825 NTEST=1
      NBR1=NWAIT(L)
  835 CONTINUE
      NCOUNT=NCOUNT+1
```

Figure 8.--Continued (page 3 of 4).

```
      NSTA=NWAIT(NTEST)
      NORD(NSTA) = KZ + NCOUNT
      NWAIT(NTEST) = NWAIT(MWAIT)
      MWAIT = MWAIT - 1
      JTRY=NAT(NSTA)
      NAT(NSTA)=1
      GO TO (846,850,670,846),JTRY
  846 WRITE(6,1)
      RETURN
  850 NCX =0
      ME = KDEG(NSTA)
      DO 970 L=1,ME
      NSTAL = NABOR(NSTA,L)
      JTRY = NAT(NSTAL)
      GO TO (960,970,930,905),JTRY
  905 WRITE(6,1)
      RETURN
  930 NAT(NSTAL)=2
      JE = KDEG(NSTAL)
      DO 940 K1=1,JE
      MSTA=NABOR(NSTAL,K1)
      JTRY=NAT(MSTA)
      GO TO (940,932,932,931),JTRY
  931 MWAIT=MWAIT+1
      NWAIT(MWAIT)=MSTA
      NAT(MSTA)=3
  932 IDEG(MSTA)=IDEG(MSTA)-1
  940 CONTINUE
      GO TO 970
  960 NCX = MAX0(NCX,NORD(NSTA)-NORD(NSTAL))
  970 CONTINUE
      NPRO = NPRO + NCX
      GO TO 770
  974 IF(NSTOP.EQ.1)GO TO 986
      IF(NPROF.LT.NPRO)GO TO 988
      IKEEP=JQ
      NPROF=NPRO
  980 CONTINUE
      NSTOP=1
      ISTART=IKEEP
      GO TO 488
  986 KZ=KZ+IZ
      IF(KZ.NE.NUMSTA)GO TO 95
      RETURN
      END
```

Figure 8.--Continued (page 4 of 4).

Remarks

The data structure employed in REORD to store the connectivity information is inefficient in its use of central memory because many of the vertexes in a graph might see only a small fraction of the number of vertexes seen by the vertex which sees the maximum number. Although the wasted memory is contrary to the purpose of the algorithm, the algorithm has been illustrated with this data structure because of its simplicity. In practice a more efficient data structure is suggested. The data structure used by the National Geodetic Survey stores the connectivity information on a direct access disk. For each vertex v, the identifiers of the vertexes seen by vertex v are stored as one record. If m is the length of the longest record and n is the number of vertexes, then the central memory required by this technique consists of two record buffers of length m. On the CDC 6600 computer used by NGS, the direct access method available through the Fortran compiler also requires an index n words in length for the direct access file. With this technique, the need for the array NABOR is eliminated. The use of external memory proves to be practical, since in numbering the vertexes of the graph, the record of any vertex has to be read at most two times into central memory per starting vertex tested.