General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

AD-A009 431

MATRIX BANDWIDTH AND PROFILE REDUCTION

H. L. Crane, Jr., et al

College of William and Mary

Prepared for:

Office of Naval Research
National Aeronautics and Space Administration

May 1975

DISTRIBUTED BY:

NTIS
National Technical Information Service
U. S. DEPARTMENT OF COMMERCE

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D AD-A009 431

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) College of William and Mary Department of Mathematics Williamsburg, VA. 23185	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Matrix Bandwidth and Profile Reduction

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report No. 8, May 1975

5. AUTHOR(S) (First name, middle initial, last name)
Norman E. Gibbs H. L. Crane, Jr.
William G. Poole, Jr.
Paul K. Stockmeyer

6. REPORT DATE May 1975	7a. TOTAL NO. OF PAGES 23	7b. NO. OF REFS 6
8a. CONTRACT OR GRANT NO. N00014-73-A0374-0001 b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Technical Report 8 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT
Approved for public release;
distribution unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Mathematics Program Office of Naval Research Arlington, VA. 22217

13. ABSTRACT

This program, REDUCE, reduces the bandwidth and profile of sparse symmetric matrices, using row and corresponding column permutations. It is a realization of the algorithm described by the authors elsewhere. It was extensively tested and compared with several other programs and was found to be considerably faster than the others, superior for bandwidth reduction and as satisfactory as any other for profile reduction.

KEY WORDS AND PHRASES: sparse matrices, bandwidth reduction, profile reduction, diameter of a graph.

PRICES SUBJECT TO CHANGE

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6801

Unclassified
Security Classification

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
US Department of Commerce
Springfield, VA. 22151

0066

ADA009431

Matrix Bandwidth and Profile Reduction

H. L. Crane, Jr.
Norman E. Gibbs
William G. Poole, Jr.
Paul K. Stockmeyer

Technical Report No. 8
May 1975

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. Department of Commerce
Springfield, VA. 22151

MATRIX BANDWIDTH AND PROFILE REDUCTION

H. L. Crane, Jr.*

Norman E. Gibbs**

William G. Poole, Jr.**

Paul K. Stockmeyer**

Reproduced from best available copy.

KEY WORDS AND PHRASES: sparse matrices, bandwidth reduction, profile reduction, diameter of a graph.

CR CATEGORIES: 5.14, 5.32

LANGUAGE: FORTRAN

This research was supported by Office of Naval Research Contract N00014-73-A-0374-0001, NR044-459 and in part under NASA Grant NGR 47-102-001 while the third author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665.

*Comptek Research, Inc., Hyattsville, MD 20782.

**Department of Mathematics, College of William & Mary, Williamsburg, VA 23185.

DESCRIPTION

Introduction. This program, REDUCE, reduces the bandwidth and profile of sparse symmetric matrices, using row and corresponding column permutations. It is a realization of the algorithm described by the authors in [4]. It was extensively tested and compared with several other programs [5] and was found to be considerably faster than the others, superior for bandwidth reduction and as satisfactory as any other for profile reduction.

Outline of the Method. Only an outline of the algorithm is given here: a detailed description can be found in [4]. The algorithm can best be described in terms of the adjacency graph, G, which has the characteristic that there is an edge in G between vertices v_i and v_j if and only if $a_{ij} \neq 0$ and $i \neq j$.

Step 1. Find the endpoints of a pseudo-diameter; i.e., a pair of vertices that are at nearly maximal distance . This is done by a finite, iterative process of determining a vertex that is a maximum distance away from a given vertex.

Step 2. Given pseudo-diameter endpoints u and v of distance k apart, partition the set of vertices into levels $L_1, L_2, \ldots, L_k$ such that adjacent vertices in G are in the same or adjacent levels and such that $\max_i |L_i|$ is nearly minimized.

Step 3. Number the vertices of G, level by level, beginning at an endpoint of the pseudo-diameter.

Matrix Data Structure. Sparse matrices are typically stored in some compact form which takes advantage of the sparsity. The data structure assumed here is one which is commonly used in bandwidth and profile schemes;

e.g., [1], [2], [3] and [6]. Subroutine REDUCE accepts as input a <u>connection table</u>, C, representing the indices of the nonzero elements of the $n \times n$ matrix A. The connection table has n rows and m columns where m is the number of off-diagonal nonzero elements in the row which has a maximum number of off-diagonal nonzero elements (i.e., the maximum degree of the graph G). The entries in row i of C are the column indices of the nonzero elements in row i of the matrix A. For example, if

$$A = \begin{bmatrix} X & X & 0 & 0 & X \\ X & X & 0 & X & X \\ 0 & 0 & X & X & 0 \\ 0 & X & X & X & 0 \\ X & X & 0 & 0 & X \end{bmatrix},$$

where X represents a nonzero element, then

$$C = \begin{bmatrix} 2 & 5 & 0 \\ 1 & 4 & 5 \\ 4 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 2 & 0 \end{bmatrix}.$$

The order of indices in a row of C is immaterial. The nonzero elements of the matrix A are never needed, only their indices.

<u>Test Results</u>. REDUCE was tested on an IBM System/360 (model 50) computer using the FORTRAN IV G and H compilers and on a CDC 6600 computer using the FORTRAN (RUN) and FORTRAN extended (FTN) compilers.

In another paper [5], the authors compared the execution times, bandwidths and profiles produced by REDUCE with those of five other programs on a wide range of problems. REDUCE typically produced the smallest

bandwidths; it produced profiles which were on average as small as those for any other program; and REDUCE was faster than all the others by at least an order of magnitude.

REFERENCES

1. Collins, R. J., Bandwidth reduction by automatic renumbering. Int. J. Numer. Meth. Engrg. 6 (1973), 345-356.

2. Cuthill, E. H., and McKee, J., Reducing the bandwidth of sparse symmetric matrices. Proc. 1969 ACM Annual Conf., Assoc. for Computing Mach., New York, 1969, 157-172.

3. Everstine, G. C., The BANDIT computer program for the reduction of matrix bandwidth for NASTRAN. NSRDC Report 3827, 1972.

4. Gibbs, N. E., Poole, W. G., Jr., and Stockmeyer, P. K., An algorithm for reducing the bandwidth and profile of a sparse matrix. ICASE Report, Hampton, VA, July, 1974.

5. Gibbs, N. E., Poole, W. G., Jr., and Stockmeyer, P. K., A comparison of several bandwidth and profile reduction algorithms. ICASE Report, Hampton, VA, November, 1974.

6. Wang, P. T. R., Bandwidth minimization, reducibility, decomposition, and triangularization of sparse matrices. Ph.D. dissertation, Department of Computer and Information Science, Ohio State University, Columbus, Ohio, 1973.

```
      SUBROUTINE REDUCE(NDSTK,NR,IOLD,RENUM,NDEG,LVL,LVLS1,LVLS2,
     -                  CCSTOR,IBW2,IPF2)
C
C  SUBROUTINE REDUCE DETERMINES A ROW AND COLUMN PERMUTATION WHICH,
C  WHEN APPLIED TO A GIVEN SPARSE MATRIX, PRODUCES A PERMUTED
C  MATRIX WITH A SMALLER BANDWIDTH AND PROFILE.
C  THE INPUT ARRAY IS A CONNECTION TABLE WHICH REPRESENTS THE
C  INDICES OF THE NONZERO ELEMENTS OF THE MATRIX, A.  THE ALGO-
C  RITHM IS DESCRIBED IN TERMS OF THE ADJACENCY GRAPH WHICH
C  HAS THE CHARACTERISTIC THAT THERE IS AN EDGE (CONNECTION)
C  BETWEEN NODES I AND J IF A(I,J) .NE. 0 AND I .NE. J.
C
C  DIMENSIONING INFORMATION--THE FOLLOWING INTEGER ARRAYS MUST BE
C  DIMENSIONED IN THE CALLING ROUTINE.
C    NDSTK(NR,D1)          D1 IS .GE. MAXIMUM DEGREE OF ALL NODES.
C    IOLD(D2)              D2 AND NR ARE .GE. THE TOTAL NUMBER OF
C    RENUM(D2+1)           NODES IN THE GRAPH.
C    NDEG(D2)              STORAGE REQUIREMENTS CAN BE SIGNIFICANTLY
C    LVL(D2)               DECREASED FOR IBM 360 AND 370 COMPUTERS
C    LVLS1(D2)             BY REPLACING INTEGER NDSTK BY
C    LVLS2(D2)             INTEGER*2 NDSTK IN SUBROUTINES REDUCE,
C    CCSTOR(D2)            DGREE, FNDIAM, TREE AND NUMBER.
C
C  COMMON INFORMATION--THE FOLLOWING COMMON BLOCK MUST BE IN THE
C  CALLING ROUTINE.
C    COMMON/GRA/N,IDPTH,IDEG
C
C  EXPLANATION OF INPUT VARIABLES--
C    NDSTK-      CONNECTION TABLE REPRESENTING GRAPH.
C                NDSTK(I,J)=NODE NUMBER OF JTH CONNECTION TO NODE
C                NUMBER I.  A CONNECTION OF A NODE TO ITSELF IS NOT
C                LISTED.  EXTRA POSITIONS MUST HAVE ZERO FILL.
C    NR-         ROW DIMENSION ASSIGNED NDSTK IN CALLING PROGRAM.
C    IOLD(I)-    NUMBERING OF ITH NODE UPON INPUT.
C                IF NO NUMBERING EXISTS THEN IOLD(I)=I.
C    N-          NUMBER OF NODES IN GRAPH (EQUAL TO ORDER OF MATRIX).
C    IDEG-       MAXIMUM DEGREE OF ANY NODE IN THE GRAPH.
C
C  EXPLANATION OF OUTPUT VARIABLES--
C    RENUM(I)-   THE NEW NUMBER FOR THE ITH NODE.
C    NDEG(I)-    THE DEGREE OF THE ITH NODE.
C    IBW2-       THE BANDWIDTH AFTER RENUMBERING.
C    IPF2-       THE PROFILE AFTER RENUMBERING.
C    IDPTH-      NUMBER OF LEVELS IN REDUCE LEVEL STRUCTURE.
C  THE FOLLOWING ONLY HAVE MEANING IF THE GRAPH WAS CONNECTED--
C    LVL(I)-     INDEX INTO LVLS1 TO THE FIRST NODE IN LEVEL I.
C                LVL(I+1)-LVL(I)= NUMBER OF NODES IN ITH LEVEL
C    LVLS1-      NODE NUMBERS LISTED BY LEVEL.
```

```
C     LVLS2(I)-  THE LEVEL ASSIGNED TO NODE I BY REDUCE..
C
C   WORKING STORAGE VARIABLE--
C     CCSTOR
C
C   LOCAL STORAGE--
C     COMMON/CC/-SUBROUTINES REDUCE, SORT2 AND PIKLVL ASSUME THAT
C                THE GRAPH HAS AT MOST 50 CONNECTED COMPONENTS.
C                SUBROUTINE FNDIAM ASSUMES THAT THERE ARE AT MOST
C                100 NODES IN THE LAST LEVEL.
C     COMMON/LVLW/-SUBROUTINES SETUP AND PIKLVL ASSUME THAT THERE
C                ARE AT MOST 100 LEVELS.
C
C
C USE INTEGER*2 NDSTK  WITH AN IBM 360 OR 370.
C
      INTEGER NDSTK
      INTEGER STNODE,RVNODE,RENUM,XC,SORT2,STNUM,CCSTOR,SIZE,STPT,SBNUM
      COMMON /GRA/ N,IDPTH,IDEG
C
C   IT IS ASSUMED THAT THE GRAPH HAS AT MOST 50 CONNECTED COMPONENTS.
C
      COMMON /CC/ XC,SIZE(50),STPT(50)
      DIMENSION CCSTOR(1),IOLD(1)
      DIMENSION NDSTK(NR,1),LVL(1),LVLS1(1),LVLS2(1),RENUM(1),NDEG(1)
      IBW2=0
      IPF2=0
C   SET RENUM(I)=0 FOR ALL I TO INDICATE NODE I IS UNNUMBERED
      DO 10 I=1,N
        RENUM(I)=0
   10 CONTINUE
C
C   COMPUTE DEGREE OF EACH NODE AND ORIGINAL BANDWIDTH AND PROFILE
C
      CALL DGREE(NDSTK,NR,NDEG,IOLD,IBW1,IPF1)
C   SBNUM= LOW END OF AVAILABLE NUMBERS FOR RENUMBERING
C   STNUM= HIGH END OF AVAILABLE NUMBERS FOR RENUMBERING
      SBNUM=1
      STNUM=N
C   NUMBER THE NODES OF DEGREE ZERO
      DO 40 I=1,N
        IF(NDEG(I).GT.0) GO TO 40
        RENUM(I)=STNUM
        STNUM=STNUM-1
   40 CONTINUE
C   FIND AN UNNUMBERED NODE OF MIN DEGREE TO START ON
   50 LOWDG=IDEG+1
      NFLG=1
      ISDIR=1
      DO 70 I=1,N
        IF(NDEG(I).GE.LOWDG) GO TO 70
        IF(RENUM(I).GT.0) GO TO 70
        LOWDG=NDEG(I)
        STNODE=I
   70 CONTINUE
```

```
C   FIND PSEUDO-DIAMETER AND ASSOCIATED LEVEL STRUCTURES.
C   STNODE AND RVNODE ARE THE ENDS OF THE DIAM AND LVLS1 AND LVLS2
C   ARE THE RESPECTIVE LEVEL STRUCTURES.
      CALL FNDIAM(STNODE,RVNODE,NDSTK,NR,NDEG,LVL,LVLS1,LVLS2,CCSTOR,
     -IDFLT)
      IF(NDEG(STNODE).LE.NDEG(RVNODE)) GO TO 75
C   NFLG INDICATES THE END TO BEGIN NUMBERING ON
      NFLG=-1
      STNODE=RVNODE
   75 CALL SETUP(LVL,LVLS1,LVLS2)
C   FIND ALL THE CONNECTED COMPONENTS  (XC COUNTS THEM)
      XC=0
      LROOT=1
      LVLN=1
      DO  80 I=1,N
        IF(LVL(I).NE.0) GO TO 80
        XC=XC+1
        STPT(XC)=LROOT
        CALL TREE(I,NDSTK,NR,LVL,CCSTOR,NDEG,LVLWTH,LVLBOT,LVLN,MAXLW,N)
        SIZE(XC)=LVLBOT+LVLWTH-LROOT
        LROOT=LVLBOT+LVLWTH
        LVLN=LROOT
   80 CONTINUE
      IF(SORT2(DMY).EQ.0) GO TO 90
      CALL PIKLVL(LVLS1,LVLS2,CCSTOR,IDFLT,ISDIR)
C   ON RETURN FROM PIKLVL, ISDIR INDICATES THE DIRECTION THE LARGEST
C   COMPONENT FELL.  ISDIR IS MODIFIED NOW TO INDICATE THE NUMBERING
C   DIRECTION.  NUM IS SET TO THE PROPER VALUE FOR THIS DIRECTION.
   90 ISDIR=ISDIR*NFLG
      NUM=SBNUM
      IF(ISDIR.LT.0) NUM=STNUM
      CALL NUMBER(STNODE,NUM,NDSTK,LVLS2,NDEG,RENUM,LVLS1,LVL,NR,NFLG,
     -IBW2,IPF2,CCSTOR,ISDIR)
C   UPDATE STNUM OR SBNUM AFTER NUMBERING
      IF(ISDIR.LT.0) STNUM=NUM
      IF(ISDIR.GT.0) SBNUM=NUM
      IF(SBNUM.LE.STNUM) GO TO 50
      IF(IBW2.LE.IBW1) RETURN
C   IF ORIGINAL NUMBERING IS BETTER THAN NEW ONE, SET UP TO RETURN IT
      DO 100 I=1,N
        RENUM(I)=IOLD(I)
  100 CONTINUE
      IBW2=IBW1
      IPF2=IPF1
      RETURN
      END
```

```
      SUBROUTINE DGREE(NDSTK,NR,NDEG,IOLD,IBW1,IPF1)
C
C  DGREE COMPUTES THE DEGREE OF EACH NODE IN NDSTK AND STORES
C  IT IN THE ARRAY NDEG.  THE BANDWIDTH AND PROFILE FOR THE ORIGINAL
C  OR INPUT RENUMBERING OF THE GRAPH IS COMPUTED ALSO.
C
C USE INTEGER*2 NDSTK  WITH AN IBM 360 OR 370.
C
      INTEGER NDSTK
      COMMON /GRA/ N IDPTH,IDEG
      DIMENSION NDSTK(NR,1),NDEG(1),IOLD(1)
      IBW1=0
      IPF1=0
      DO 100 I=1,N
        NDEG(I)=0
        IRW=0
        DO 80 J=1,IDEG
          ITST=NDSTK(I,J)
          IF(ITST) 90,90,50
   50     NDEG(I)=NDEG(I)+1
          IDIF=IOLD(I)-IOLD(ITST)
          IF(IRW.LT.IDIF) IRW=IDIF
   80   CONTINUE
   90   IPF1=IPF1+IRW
        IF(IRW.GT.IBW1) IBW1=IRW
  100 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FNDIAM(SND1,SND2,NDSTK,NR,NDEG,LVL,LVLS1,LVLS2,
     -IWK,IDFLT)
C
C  FNDIAM IS THE CONTROL PROCEDURE FOR FINDING THE PSEUDO-DIAMETER OF
C  NDSTK AS WELL AS THE LEVEL STRUCTURE FROM EACH END
C
C  SND1-          ON INPUT THIS IS THE NODE NUMBER OF THE FIRST
C                 ATTEMPT AT FINDING A DIAMETER.  ON OUTPUT IT
C                 CONTAINS THE ACTUAL NUMBER USED.
C  SND2-          ON OUTPUT CONTAINS OTHER END OF DIAMETER
C  LVLS1-         ARRAY CONTAINING LEVEL STRUCTURE WITH SND1 AS ROOT
C  LVLS2-         ARRAY CONTAINING LEVEL STRUCTURE WITH SND2 AS ROOT
C  IDFLT-         FLAG USED IN PICKING FINAL LEVEL STRUCTURE. SET
C                 =1 IF WIDTH OF LVLS1 .LE. WIDTH OF LVLS2. OTHERWISE =2
C  LVL,IWK-       WORKING STORAGE
C
C USE INTEGER*2 NDSTK  WITH AN IBM 360 OR 370.
C
      INTEGER NDSTK
      INTEGER FLAG,SND,SND1,SND2
      COMMON /GRA/ N,IDPTH
C
C  IT IS ASSUMED THAT THE LAST LEVEL HAS AT MOST 100 NODES.
C
      COMMON /CC/  NDLST(100)
      DIMENSION NDSTK(NR,1),NDEG(1),LVL(1),LVLS1(1),LVLS2(1),IWK(1)
      FLAG=0
      MTW2=N
      SND=SND1
C  ZERO LVL TO INDICATE ALL NODES ARE AVAILABLE TO TREE
   20 DO 25 I=1,N
        LVL(I)=0
   25 CONTINUE
      LVLN=1
C  DROP A TREE FROM SND
      CALL TREE(SND,NDSTK,NR,LVL,IWK,NDEG,LVLWTH,LVLBOT,LVLN,MAXLW,MTW2)
      IF(FLAG.GE.1) GO TO 110
      FLAG=1
   70 IDPTH=LVLN-1
      MTW1=MAXLW
C  COPY LEVEL STRUCTURE INTO LVLS1
      DO 75 I=1,N
        LVLS1(I)=LVL(I)
   75 CONTINUE
      NDXN=1
      NDXL=0
      MTW2=N
```

```
C  SORT LAST LEVEL BY DEGREE  AND STORE IN NDLST
      CALL SORTDG(NDLST,IWK(LVLBOT),NDXL,LVLWTH,NDEG)
      SND=NDLST(1)
      GO TO 20
  110 IF(IDPTH.GE.LVLN-1) GO TO 120
C  START AGAIN WITH NEW STARTING NODE
      SND1=SND
      GO TO 70
  120 IF(MAXLW.GE.MTW2) GO TO 130
      MTW2=MAXLW
      SND2=SND
C  STORE NARROWEST REVERSE LEVEL STRUCTURE IN LVLS2
      DO 125 I=1,N
        LVLS2(I)=LVL(I)
  125 CONTINUE
  130 IF(NDXN.EQ.NDXL) GO TO 140
C  TRY NEXT NODE IN NDLST
      NDXN=NDXN+1
      SND=NDLST(NDXN)
      GO TO 20
  140 IDFLT=1
      IF(MTW2.LE.MTW1) IDFLT=2
      RETURN
      END
```

```
      SUBROUTINE TREE(IROOT,NDSTK,NR,LVL,IWK,NDEG,LVLWTH,LVLBOT,
     -LVLN,MAXLW,IBORT)
C
C  TREE DROPS A TREE IN NDSTK FROM IROOT
C
C  LVL-          ARRAY INDICATING AVAILABLE NODES IN NDSTK WITH ZERO
C                ENTRIES. TREE ENTERS LEVEL NUMBERS ASSIGNED
C                DURING EXECUTION OF THIS PROCEDURE
C  IWK-          ON OUTPUT CONTAINS NODE NUMBERS USED IN TREE
C                ARRANGED BY LEVELS (IWK(LVLN) CONTAINS IROOT
C                AND IWK(LVLBOT+LVLWTH-1) CONTAINS LAST NODE ENTERED)
C  LVLWTH-       ON OUTPUT CONTAINS WIDTH OF LAST LEVEL
C  LVLBOT-       ON OUTPUT CONTAINS INDEX INTO IWK OF FIRST
C                NODE IN LAST LEVEL
C  MAXLW-        ON OUTPUT CONTAINS THE MAXIMUM LEVEL WIDTH
C  LVLN-         ON INPUT THE FIRST AVAILABLE LOCATION IN IWK
C                USUALLY ONE BUT IF IWK IS USED TO STORE PREVIOUS
C                CONNECTED COMPONENTS. LVLN IS NEXT AVAILABLE LOCATION.
C                ON OUTPUT THE TOTAL NUMBER OF LEVELS + 1
C  IBORT-        INPUT PARAM WHICH TRIGGERS EARLY RETURN IF
C                MAXLW BECOMES .GE. IBORT
C
C USE INTEGER*2 NDSTK  WITH AN IBM 360 OR 370.
C
      INTEGER NDSTK
      DIMENSION NDSTK(NR,1),LVL(1),IWK(1),NDEG(1)
      MAXLW=0
      ITOP=LVLN
      INOW=LVLN
      LVLBOT=LVLN
      LVLTOP=LVLN+1
      LVLN=1
      LVL(IROOT)=1
      IWK(ITOP)=IROOT
   30 LVLN=LVLN+1
   35 IWKNOW=IWK(INOW)
      NDROW=NDEG(IWKNOW)
      DO 40 J=1,NDROW
         ITEST=NDSTK(IWKNOW,J)
         IF(LVL(ITEST).NE.0) GO TO 40
         LVL(ITEST)=LVLN
         ITOP=ITOP+1
         IWK(ITOP)=ITEST
   40 CONTINUE
      INOW=INOW+1
      IF(INOW.LT.LVLTOP) GO TO 35
      LVLWTH=LVLTOP-LVLBOT
      IF(MAXLW.LT.LVLWTH) MAXLW=LVLWTH
      IF(MAXLW.GE.IBORT) RETURN
      IF(ITOP.LT.LVLTOP) RETURN
      LVLBOT=INOW
      LVLTOP=ITOP+1
      GO TO 30
      END
```

```
      SUBROUTINE SORTDG(STK1,STK2,X1,X2,NDEG)
C
C  SORTDG SORTS STK2 BY DEGREE OF THE NODE AND ADDS IT TO THE END
C  OF STK1 IN ORDER OF LOWEST TO HIGHEST DEGREE.  X1 AND X2 ARE THE
C  NUMBER OF NODES IN STK1 AND STK2 RESPECTIVELY.
C
      INTEGER X1,X2,STK1,STK2,TEMP
      COMMON /GRA/ N,IDPTH
      DIMENSION NDEG(1),STK1(1),STK2(1)
      IND=X2
   10 ITEST=0
      IND=IND-1
      IF(IND.LT.1) GO TO 40
      DO 30 I=1,IND
         J=I+1
         ISTK2=STK2(I)
         JSTK2=STK2(J)
         IF(NDEG(ISTK2).LE.NDEG(JSTK2)) GO TO 30
         ITEST=1
         TEMP=STK2(I)
         STK2(I)=STK2(J)
         STK2(J)=TEMP
   30 CONTINUE
      IF(ITEST.EQ.1) GO TO 10
   40 DO 50 I=1,X2
         X1=X1+1
         STK1(X1)=STK2(I)
   50 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SETUP(LVL,LVLS1,LVLS2)
C
C   SETUP COMPUTES THE REVERSE LEVELING INFO FROM LVLS2 AND STORES
C   IT INTO LVLS2.  NACUM(I) IS INITIALIZED TO NODES/ITH LEVEL FOR NODES
C   ON THE PSEUDO-DIAMETER OF THE GRAPH.  LVL IS INITIALIZED TO NON-
C   ZERO FOR NODES ON THE PSEUDO-DIAM AND NOD    IN A DIFFERENT
C   COMPONENT OF THE GRAPH.
C
      COMMON /GRA/ N,IDPTH
C
C   IT IS ASSUMED THAT THERE ARE AT MOST 100 LEVELS.
C
      COMMON /LVLW/ NHIGH(100),NLOW(100),NACUM(100)
      DIMENSION LVL(1),LVLS1(1),LVLS2(1)
      DO 30 I=1,IDPTH
        NACUM(I)=0
   30 CONTINUE
      DO 140 I=1,N
        LVL(I)=1
        LVLS2(I)=IDPTH+1-LVLS2(I)
        ITEMP=LVLS2(I)
        IF(ITEMP.GT.IDPTH) GO TO 140
        IF(ITEMP.NE.LVLS1(I)) GO TO 100
        NACUM(ITEMP)=NACUM(ITEMP)+1
        GO TO 140
  100   LVL(I)=0
  140 CONTINUE
      RETURN
      END
```

```
      INTEGER FUNCTION SORT2(DMY)
C
C  SORT2 SORTS SIZE AND STPT INTO DESCENDING ORDER ACCORDING TO
C  VALUES OF SIZE. XC=NUMBER OF ENTRIES IN EACH ARRAY
C
      INTEGER TEMP,CCSTOR,SIZE,STPT,XC
C
C  IT IS ASSUMED THAT THE GRAPH HAS AT MOST 50 CONNECTED COMPONENTS.
C
      COMMON /CC/ XC,SIZE(50),STPT(50)
      SORT2=0
      IF(XC.EQ.0) RETURN
      SORT2=1
      IND=XC
   10 ITEST=0
      IND=IND-1
      IF(IND .LT.1) RETURN
      DO 17 I=1,IND
        J=I+1
        IF(SIZE(I).GE.SIZE(J)) GO TO 17
        ITEST=1
        TEMP=SIZE(I)
        SIZE(I)=SIZE(J)
        SIZE(J)=TEMP
        TEMP=STPT(I)
        STPT(I)=STPT(J)
        STPT(J)=TEMP
   17 CONTINUE
      IF(ITEST.EQ.1) GO TO 10
      RETURN
      END
```

```
      SUBROUTINE PIKLVL(LVLS1,LVLS2,CCSTOR,IDFLT,ISDIR)
C
C  PIKLVL CHOOSES THE LEVEL STRUCTURE  USED IN NUMBERING GRAPH
C
C  LVLS1-        ON INPUT CONTAINS FORWARD LEVELING INFO
C  LVLS2-        ON INPUT CONTAINS REVERSE LEVELING INFO
C                ON OUTPUT THE FINAL LEVEL STRUCTURE CHOSEN
C  CCSTOR-       ON INPUT CONTAINS CONNECTED COMPONENT INFO
C  IDFLT-        ON INPUT =1 IF WDTH LVLS1*WDTH LVLS2, =2 OTHERWISE
C  NHIGH         KEEPS TRACK OF LEVEL WIDTHS FOR HIGH NUMBERING
C  NLOW-         KEEPS TRACK OF LEVEL WIDTHS FOR LOW NUMBERING
C  NACUM-        KEEPS TRACK OF LEVEL WIDTHS FOR CHOSEN LEVEL STRUCTURE
C  XC-           NUMBER OF CONNECTED COMPONENTS
C  SIZE(I)-      SIZE OF ITH CONNECTED COMPONENT
C  STPT(I)-      INDEX INTO CCSTORE OF 1ST NODE IN ITH CON COMPT
C  ISDIR-        FLAG WHICH INDICATES WHICH WAY THE LARGEST CONNECTED
C                COMPONENT FELL.  =+1 IF LOW AND -1 IF HIGH
C
      INTEGER CCSTOR,SIZE,STPT,XC,END
      COMMON /GRA/ N,IDPTH
C
C  IT IS ASSUMED THAT THE GRAPH HAS AT MOST 50 COMPONENTS AND
C  THAT THERE ARE AT MOST 100 LEVELS.
C
      COMMON /LVLW/ NHIGH(100),NLOW(100),NACUM(100)
      COMMON /CC/ XC,SIZE(50),STPT(50)
      DIMENSION LVLS1(1),LVLS2(1),CCSTOR(1)
C  FOR EACH CONNECTED COMPONENT DO
      DO 270 I=1,XC
        J=STPT(I)
        END=SIZE(I)+J-1
C  SET NHIGH AND NLOW EQUAL TO NACUM
        DO 205 K=1,IDPTH
          NHIGH(K)=NACUM(K)
          NLOW(K)=NACUM(K)
  205   CONTINUE
C  UPDATE NHIGH AND NLOW FOR EACH NODE IN CONNECTED COMPONENT
        DO 210 K=J,END
          INODE=CCSTOR(K)
          LVLNH=LVLS1(INODE)
          NHIGH(LVLNH)=NHIGH(LVLNH)+1
          LVLNL=LVLS2(INODE)
          NLOW(LVLNL)=NLOW(LVLNL)+1
  210   CONTINUE
        MAX1=0
        MAX2=0
C  SET MAX1=LARGEST NEW NUMBER IN NHIGH
C  SET MAX2=LARGEST NEW NUMBER IN NLOW
        DO 240 K=1,IDPTH
          IF(2*NACUM(K).EQ.NLOW(K)+NHIGH(K)) GO TO 240
          IF(NHIGH(K).GT.MAX1) MAX1=NHIGH(K)
          IF(NLOW(K).GT.MAX2) MAX2=NLOW(K)
  240   CONTINUE
```

```
C   SET IT= NUMBER OF LEVEL STRUCTURE TO BE USED
          IT=1
          IF(MAX1.GT.MAX2) IT=2
          IF(MAX1.EQ.MAX2) IT=IDFLT
          IF(IT.EQ.2) GO TO 265
          IF(I.EQ.1) ISDIR=-1
C   COPY LVLS1 INTO LVLS2 FOR EACH NODE IN CONNECTED COMPONENT
          DO 260 K=J,END
            INODE=CCSTOR(K)
            LVLS2(INODE)=LVLS1(INODE)
  260     CONTINUE
C   UPDATE NACUM TO BE THE SAME AS NHIGH
          DO 262 K=1,IDPTH
            NACUM(K)=NHIGH(K)
  262     CONTINUE
          GO TO 270
C   UPDATE NACUM TO BE THE SAME AS NLOW
  265     DO 267 K=1,IDPTH
            NACUM(K)=NLOW(K)
  267     CONTINUE
  270   CONTINUE
        RETURN
        END
```

```
      SUBROUTINE NUMBER(SND,NUM,NDSTK,LVLS2,NDEG,RENUM,LVLST,LSTPT,NR,
     -NFLG,IBW2,IPF2,IPFA,ISDIR)
C
C   NUMBER PRODUCES THE NUMBERING OF THE GRAPH FOR MIN BANDWIDTH
C
C   SND-          ON INPUT THE NODE TO BEGIN NUMBERING ON
C   NUM-          ON INPUT AND OUTPUT, THE NEXT AVAILABLE NUMBER
C   LVLS2-        THE LEVEL STRUCTURE TO BE USED IN NUMBERING
C   RENUM-        THE ARRAY USED TO STORE THE NEW NUMBERING
C   LVLST-        ON OUTPUT CONTAINS LEVEL STRUCTURE
C   LSTPT(I)-     ON OUTPUT, INDEX INTO LVLST TO FIRST NODE IN ITH LVL
C                 LSTPT(I+1) - LSTPT(I) = NUMBER OF NODES IN ITH LVL
C   NFLG-         =+1 IF SND IS FORWARD END OF PSEUDO-DIAM
C                 =-1 IF SND IS REVERSE END OF PSEUDO-DIAM
C   IBW2-         BANDWIDTH OF NEW NUMBERING COMPUTED BY NUMBER
C   IPF2-         PROFILE OF NEW NUMBERING COMPUTED BY NUMBER
C   IPFA-         WORKING STORAGE USED TO COMPUTE PROFILE AND BANDWIDTH
C   ISDIR-        INDICATES STEP DIRECTION USED IN NUMBERING(+1 OR -1)
C
C USE INTEGER*2 NDSTK  WITH AN IBM 360 OR 370.
C
      INTEGER NDSTK
      INTEGER SND,STKA,STKB,STKC,STKD,XA,XB,XC,XD,CX,END,RENUM,TEST
      COMMON /GRA/ N,IDPTH,IDEG
C
C   THE STORAGE IN COMMON BLOCKS CC AND LVLW IS NOW FREE AND CAN
C   BE USED FOR STACKS.
C
      COMMON /LVLW/ STKA(100),STKB(100),STKC(100)
      COMMON /CC/ STKD(100)
      DIMENSION IPFA(1)
      DIMENSION NDSTK(NR,1),LVLS2(1),NDEG(1),RENUM(1),LVLST(1),LSTPT(1)
C   SET UP LVLST AND LSTPT FROM LVLS2
      DO 3 I=1,N
        IPFA(I)=0
    3 CONTINUE
      NSTPT=1
      DO 5 I=1,IDPTH
        LSTPT(I)=NSTPT
        DO 5 J=1,N
          IF(LVLS2(J).NE.I) GO TO 5
          LVLST(NSTPT)=J
          NSTPT=NSTPT+1
    5 CONTINUE
      LSTPT(IDPTH+1)=NSTPT
```

```
C   STKA, STKB, STKC AND STKD ARE STACKS WITH POINTERS
C   XA,XB,XC, AND XD.  CX IS A SPECIAL POINTER INTO STKC WHICH
C   INDICATES THE PARTICULAR NODE BEING PROCESSED.
C   LVLN KEEPS TRACK OF THE LEVEL WE ARE WORKING AT.
C   INITIALLY STKC CONTAINS ONLY THE INITIAL NODE, SND.
      LVLN=0
      IF(NFLG.LT.0) LVLN=IDPTH+1
      XC=1
      STKC(XC)=SND
   10 CX=1
      XD=0
      LVLN=LVLN+NFLG
      LST=LSTPT(LVLN)
      LND=LSTPT(LVLN+1)-1
C   BEGIN PROCESSING NODE STKC(CX)
   20 IPRO=STKC(CX)
      RENUM(IPRO)=NUM
      NUM=NUM+ISDIR
      END=NDEG(IPRO)
      XA=0
      XB=0
C   CHECK ALL ADJACENT NODES
      DO 50 I=1,END
        TEST=NDSTK(IPRO,I)
        INX=RENUM(TEST)
C   ONLY NODES NOT NUMBERED OR ALREADY ON A STACK ARE ADDED
        IF(INX.EQ.0) GO TO 30
        IF(INX.LT.0) GO TO 50
C   DO PRELIMINARY BANDWIDTH AND PROFILE CALCULATIONS
        NBW=(RENUM(IPRO)-INX)*ISDIR
        IF(ISDIR.GT.0) INX=RENUM(IPRO)
        IF(IPFA(INX).LT.NBW) IPFA(INX)=NBW
        GO TO 50
   30   RENUM(TEST)=-1
C   PUT NODES ON SAME LEVEL ON STKA, ALL OTHERS ON STKB
        IF(LVLS2(TEST).EQ.LVLS2(IPRO)) GO TO 40
        XB=XB+1
        STKB(XB)=TEST
        GO TO 50
   40   XA=XA+1
        STKA(XA)=TEST
   50 CONTINUE
C   SORT STKA AND STKB INTO INCREASING DEGREE AND ADD STKA TO STKC
C   AND STKB TO STKD
      IF(XA.EQ.0) GO TO 55
      IF(XA.EQ.1) GO TO 52
      CALL SORTDG(STKC,STKA,XC,XA,NDEG)
      GO TO 55
   52 XC=XC+1
      STKC(XC)=STKA(XA)
   55 IF(XB.EQ.0) GO TO 65
      IF(XB.EQ.1) GO TO 62
      CALL SORTDG(STKD,STKB,XD,XB,NDEG)
      GO TO 65
   62 XD=XD+1
      STKD(XD)=STKB(XB)
```

```
C   BE SURE TO PROCESS ALL NODES IN STKC
   65 CX=CX+1
      IF(XC.GE.CX) GO TO 20
C   WHEN STKC IS EXHAUSTED LOOK FOR MIN DEGREE NODE IN SAME LEVEL
C   WHICH HAS NOT BEEN PROCESSED
      MAX=IDEG+1
      SND=N+1
      DO 70 I=LST,LND
        TEST=LVLST(I)
        IF(RENUM(TEST).NE.0) GO TO 70
        IF(NDEG(TEST).GE.MAX) GO TO 70
        RENUM(SND)=0
        RENUM(TEST)=-1
        MAX=NDEG(TEST)
        SND=TEST
   70 CONTINUE
      IF(SND.EQ.N+1) GO TO 75
      XC=XC+1
      STKC(XC)=SND
      GO TO 20
C   IF STKD IS EMPTY WE ARE DONE, OTHERWISE COPY STKD ONTO STKC
C   AND BEGIN PROCESSING NEW STKC
   75 IF(XD.EQ.0) GO TO 100
      DO 80 I=1,XD
        STKC(I)=STKD(I)
   80 CONTINUE
      XC=XD
      GO TO 10
C   DO FINAL BANDWIDTH AND PROFILE CALCULATIONS
  100 DO 120 I=1,N
        IF(IPFA(I).GT.IBW2) IBW2=IPFA(I)
        IPF2=IPF2+IPFA(I)
  120 CONTINUE
      RETURN
      END
```