NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER
Bethesda, Maryland 20034

THE PERFORMANCE ON THE CDC 6400 OF A RHEINBOLDT-MESZTENYI PROGRAM FOR SOLVING LARGE SPARSE SYMMETRIC SYSTEMS OF LINEAR EQUATIONS

Donald A. Gignac

COMPUTATION AND MATHEMATICS DEPARTMENT
RESEARCH AND DEVELOPMENT REPORT

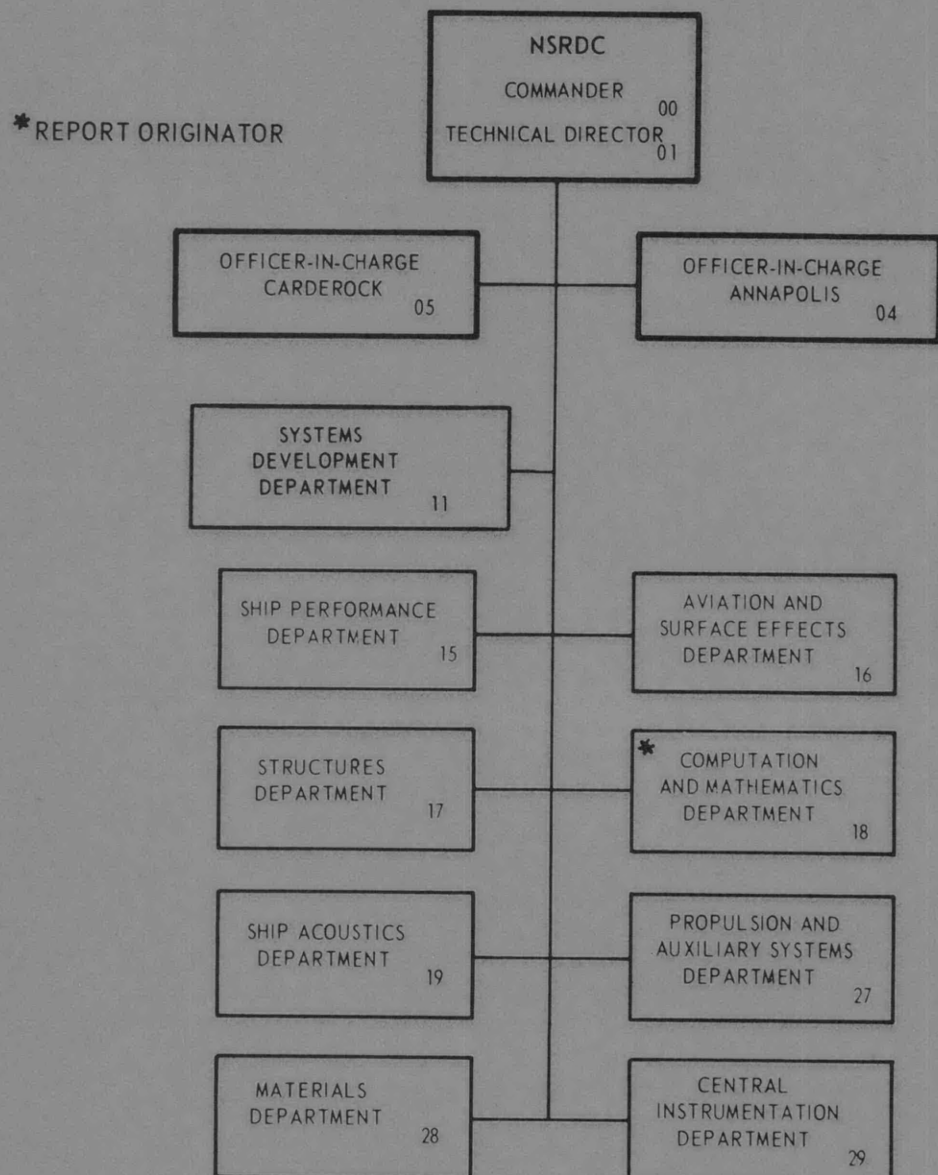
JULY 1974

Report 4496

The Naval Ship Research and Development Center is a U. S. Navy center for laboratory effort directed at achieving improved sea and air vehicles. It was formed in March 1967 by merging the David Taylor Model Basin at Carderock, Maryland with the Marine Engineering Laboratory at Annapolis, Maryland.

Naval Ship Research and Development Center
Bethesda, Md. 20034

MAJOR NSRDC ORGANIZATIONAL COMPONENTS



NDW-NSRDC 3960/44 (REV. 8/71)
GPO 917-872

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 4496	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Performance on the CDC 6400 of a Rheinboldt-Mesztenyi Program for Solving Large Sparse Symmetric Systems of Linear Equations		5. TYPE OF REPORT & PERIOD COVERED Research and Development 6. PERFORMING ORG. REPORT NUMBER 4496
7. AUTHOR(s) Donald A. Gignac		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Ship Research and Development Center Bethesda, Maryland 20034		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS SR 014 03 01, Task 15322 Work Unit 1-1844-044
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE July 1974 13. NUMBER OF PAGES 25
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Out-of-Core Solution
Symmetric System
Linear Equation Solver
Arc-graph theory

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

On the CDC 6400 computer this FORTRAN Extended version of the Rheinboldt-Mesztenyi computer program for solving sparse symmetric matrix equations was tested with respect to certain sample problems representative of structural analysis problems. This program does not appear to be competitive with CSKYDG, another linear equation solver. Lack of several special features in the CDC 6400 instruction set results in high overhead for manipulating the data structures used.

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

On the CDC 6400 computer this FORTRAN Extended version of the Rheinboldt-Mesztenyi computer program for solving sparse symmetric matrix equations was tested with respect to certain sample problems representative of structural analysis problems. This program does not appear to be competitive with CSKYDG, another linear equation solver. Lack of several special features in the CDC 6400 instruction set results in high overhead for manipulating the data structures used.

ADMINISTRATIVE INFORMATION

The Naval Ship Systems Command (0311) sponsored this study under Subproject SR 014 03 01, Task 15322. The work was performed under work unit number 1-1844-004.

INTRODUCTION

In the finite element approach to static structural analysis, the computation of the solution of the equation

$$KU = P$$

a positive definite system of simultaneous linear equations, is basic. However, the order of K is often so large that it does not suffice merely to take advantage of K's symmetry and banded structure. For such K it is necessary to consider an out-of-core solution or to devise some storage scheme which exploits sparsity more fully, or even to utilize both of these approaches together. The Rheinboldt-Mesztenyi program[1] was investigated as part of an effort to develop an improved solution capability for large sparse K. This program utilizes a data structure for storing sparse matrices based on arc-graph theory to facilitate an in-core triangular decomposition solution of KU = P for such a K. The best of such programs encountered will eventually be incorporated into large-scale structural programs such as NASTRAN.

The author was recently asked for an opinion on the usefulness of the program of Professors Rheinboldt and Mesztenyi for solving the sparse matrix equations arising in structural analysis calculations on the CDC 6000 series computers. A FORTRAN version of the Rheinboldt-Mesztenyi (R-M) program for symmetric matrices was obtained and modified for the CDC 6400. This report discusses a comparison of this FORTRAN implementation of the R-M program with another FORTRAN linear equation solver, CSKYDG,[2] for certain sample problems.

---

[1] Rheinboldt, W. and Mesztenyi, C.,"Problems for the Solution of Large Sparse Matrix Problems Based on the Arc-Graph Structure," University of Maryland Computer Science Center, Technical Report TR-262, September 1973.

[2] Gignac, D.A., "CSKYDG, An Out-of-Core Cholesky Algorithm Equation Solver for Large Positive Definite Systems of Linear Equations," Naval Ship Research and Development Center Report 4377, February 1974.

THEORY

The R-M program solves the system KU = P using that form of triangular decomposition which does not require square roots. As the decomposition proceeds the rows and corresponding columns of K are interchanged in accordance with the pivoting strategy of Curtis and Reid.[3] If Q represents the permutation matrix for the required row interchanges, then

$$QKQ^T = LDL^T$$

where L is a unit lower triangular matrix and D is a diagonal matrix. We then solve the triangular systems

$$LX_1 = QP$$

$$DX_2 = X_1$$

$$L^TX_3 = X_2$$

using forward or backward substitution as required, and finally obtain

$$U = Q^TX_3$$

In theory this procedure can solve KU = P for any non-singular symmetric K. However, the method of Cholesky (square root method) works better for positive definite K. The Cholesky algorithm factors K into the product of a lower triangular matrix S and its transpose, that is,

$$K = SS^T ,$$

then solves the triangular systems

$$SX = P$$

$$S^TU = X$$

using forward or backward substitution as required.

The Cholesky algorithm has two advantages over the LDL^T procedure. First it does not require pivoting to ensure stability, making the matrix

---

3 Curtis, A.R., and Reid, J.K., "FORTRAN subroutines for the solution of sparse sets of linear equations," United Kingdom Atomic Energy Research Establishment, Harwell, England, Tech. Report AERE-R6844, 1971.

decomposition and the forward and backward substitutions less involved. Secondly the forward substitution can be readily incorporated into the Cholesky decomposition with a significant saving of time. (This last advantage may be realized only for a single solution of KU = P for a given K.) The CSKYDG program takes advantage of these features. Wilkinson and Reinsch[4] give details of both procedures.

EXAMPLES

Professor Mesztenyi provided the author with a FORTRAN implementation of the R-M program for the UNIVAC 1108. This program consisted of three subroutines: READ, LU, and SOLVE. The SETUP subroutine was added to facilitate the input of the matrix element. Its two arguments are NS and W. The SETUP subroutine reads from tape 4 the non-zero elements in the lines of the upper triangular half of the coefficient matrix K in the form of triplets

I, J, K(K,J)

and writes these triplets in batches of NS on tape 5. If the number of triplets is not a multiple of NS, then the last batch of triplets is filled out to NS elements by adding the appropriate number of triplets

1,0,0.0 .

The righthand side of KU = P is read from tape 4 and passed through the argument W. The READ subroutine then reads the triplets in batches of NS from tape 5 and sets up arcs of non-zero elements. The LU subroutine then obtains the LDL^T decomposition of K in terms of these arcs. The SOLVE subroutine then obtains U from P by solving the intermediate systems of equations. The integer packing and unpacking subroutines IPACK and IUNPK were added later.

---

[4] Wilkinson, J.H. and Reinsch, C., editors, "Handbook for Automatic computation," vol. II, "Linear Algebra," Springer-Verlag, New York 1971, pp. 9-11.

After the CDC 6400 version of the R-M program had been checked out, it was compared with CSKYDG,[2] the author's own previously developed linear equation solver for KU = P. The following examples were chosen as the basis for this comparison because these are in some sense representative of systems which arise in structural analysis.

The matrix family of the first example in Table 1, A_N^1, is generated as follows: Let N be an integer ≥ 3. Let C_N be the tridiagonal of order N with 4's on the diagonal and a line of -1's above and below the diagonal. Let I_N be the identity matrix of order N. An (N+1)-banded matrix of order N^2, A_N^1 is constructed by

(1) stringing N C_N submatrices along the diagonal,

(2) inserting lines of N-1 $-I_N$ submatrices above and below the diagonal, and

(3) setting the remaining elements of A_N^1 equal to 0.

The right-hand side of the system $A_N^1 X = B$ is chosen such that the exact solution X has all components equal to 1.

The matrix family of the second example in Table 2, A_N^2, is similarly generated. This time let C_N have diagonal elements of 6. An (N^2+1)-banded matrix of order N^3 is constructed by

(1) stringing N^2 C_N submatrices along the diagonal,

(2) inserting lines of N^2-1 $-I_N$ submatrices above and below the diagonal,

(3) inserting lines of N^2-N $-I_N$ submatrices as the N^{th} lines above and below the diagonal, and

(4) setting the remaining elements of A_N^2 equal to 0.

The right-hand side of the system $A_N^2 X = B$ is chosen such that the exact solution is ℓ_1. These two matrix families have characteristics of matrices arising from finite difference approximations to the Dirichlet problem.

OBSERVATIONS AND CONCLUSIONS

The tables assembled in this section present the data concerning the performance of the present FORTRAN version of the R-M program on the CDC 6400. In these tables, N indicates the order of the matrix and M its bandwidth. The information in Tables 3 and 4 has been published previously.[2] Note that the SETUP subroutine and the NS parameter of Table 3 are different from those of Tables 1 and 2.

The CDC 6400 version of the R-M program was able to handle the order 225, bandwidth 16 case but not the order 400, bandwidth 21 case of the first matrix family using a field length of 153,400 (Table 1a). Making use of the integer packing and unpacking subroutines IPACK and IUNPK* to realize a fourfold compression of certain integer arrays, the 'packed' version of the R-M program handled the order 900, bandwidth 31 case but not the order 1225, bandwidth 36 case of the first matrix family using a field length of 145,500 (Table 1b). Similarly the 'unpacked' version of the R-M program handled the order 125, bandwidth 26 case but not the order 216, bandwidth 37 case of the second matrix family using a field length of 153,400 (Table 2a). The 'packed' version of the R-M program handled the order 343, bandwidth 50 case but not the order 512, bandwidth 65 case of the second matrix family using a field length of 145,500.

Neither the 'packed' or 'unpacked' CDC 6400 version of the R-M program seems competitive in solution times with existing FORTRAN linear equation solvers, in particular CSKYDG, for the examples investigated. Moreover, the CSKYDG program required a field length of only 70,000 to produce the results shown in Tables 3 and 4. However, the accuracy of the solutions of the two programs was comparable.

---

* Those subroutines were provided by Mr. Michael Golden of the Theory of Structures Branch (Code 1844). IPACK packs four integer words (each of which requires no more than fifteen bits) into one word. IUNPK reverses the packing operation. No documentation is available for these subroutines.

The R-M program was at a special disadvantage in this investigation. The "bookkeeping" procedures, so crucial to the R-M program, are most efficient when the program is coded in assembly language (COMPASS for the CDC 6400) rather than FORTRAN. Then the integer packing and unpacking procedures apparently required by the R-M program can be directly implemented in the program rather than using subroutines which are of necessity slower.

Moreover, in discussing the somewhat disappointing performance of the CDC 6400 version of the R-M program, Professor Mesztenyi pointed out that the "fetch" cycle of the CDC 6400 is more expensive than that of the UNIVAC 1108. He also noted that certain FORTRAN compilers appear to provide somewhat inefficient FORTRAN compilations, although the results published here were obtained using that compilation option of the present version of the FTN FORTRAN compiler which usually produces the most efficient code.

It is only fair to note that on the UNIVAC 1108 the performance of the assembly language version of the R-M program is quite satisfactory. For example, the LDL^T decomposition times listed in the first column of Table 5 for certain members of the first matrix family are UNIVAC 1108 assembly language times taken from Rheinboldt and Mesztenyi (page 48, Table 5). The second column of Table 5 shows the CDC 6400 FORTRAN LDL^T decomposition times from Table 1b of this report, and the third column of Table 5 gives the corresponding FORTRAN CDC 6400 CSKYDG total solution times from Table 3a of this report. Using the rule of thumb that the UNIVAC 1108 is about three times faster than the CDC 6400, these assembly language results indicate a creditable performance on the part of the R-M program. Even so, CSKYDG (for which, it must be remembered, total solution times are given) still appears to be significantly faster.

TABLE 1 - R-M SOLUTION TIMES FOR $A_L^1 X = B$

NS = 250

A - UNPACKED VERSION

N	M	SETUP (Secs.)	READ (Secs.)	SETUP+READ (Secs.)	LU (Secs.)	SOLVE (Secs.)	LU+SOLVE (Secs.)	TOTAL TIME (Secs.)
25	6	.08	.05	.12	.10	.02	.12	.24
100	11	.14	.14	.28	4.80	.29	5.09	5.37
225	16	.26	.17	.44	16.32	.97	17.29	17.73

B - PACKED VERSION

N	M	SETUP (Secs.)	READ (Secs.)	SETUP+READ (Secs.)	LU (Secs.)	SOLVE (Secs.)	LU+SOLVE (Secs.)	TOTAL TIME (Secs.)
25	6	.06	.08	.14	.31	.07	.39	.53
100	11	.13	.27	.40	4.43	.64	5.07	5.47
225	16	.25	.54	.80	23.22	2.32	25.53	26.33
400	21	.47	.98	1.44	83.04	6.25	89.29	90.74
625	26	.79	1.53	2.32	198.56	12.39	210.95	213.28
900	31	1.19	2.19	3.39	479.70	25.15	504.85	508.24

TABLE 2 - R-M SOLUTION TIMES FOR $A_L^2 X = B$

NS = 250

A - UNPACKED VERSION

N	M	SETUP (Secs.)	READ (Secs.)	SETUP+READ (Secs.)	LU (Secs.)	SOLVE (Secs.)	LU+SOLVE (Secs.)	TOTAL TIME (Secs.)
125	26	.18	.12	.30	25.23	1.05	26.28	26.58

B - PACKED VERSION

N	M	SETUP (Secs.)	READ (Secs.)	SETUP+READ (Secs.)	LU (Secs.)	SOLVE (Secs.)	LU+SOLVE (Secs.)	TOTAL TIME (Secs.)
125	26	.18	.38	.56	27.61	2.48	30.09	30.65
216	37	.36	.70	1.06	170.84	9.41	180.25	181.31
343	50	.62	1.10	1.72	511.02	20.92	531.94	533.66

TABLE 3 - CSKYDG SOLUTION TIMES FOR $A_L^1 X = B$

NS = 10

N	M	SETUP (Secs.)	SOLUTION (Secs.)	TOTAL TIME (Secs.)
25	6	.07	.15	.21
100	11	.16	.61	.76
225	16	.48	3.07	3.55
400	21	.96	5.69	6.65
625	26	1.92	15.32	17.25
900	31	3.06	22.59	25.64
1225	36	4.87	46.49	51.36
1600	41	6.83	62.39	69.22
2025	46	9.76	108.11	117.87
2500	51	12.69	136.23	148.92
3025	56	17.25	218.10	235.35

TABLE 4 - CSKYDG SOLUTION TIMES FOR $A_L^2X = B$

NS = 10

N	M	SETUP (Secs.)	SOLUTION (Secs.)	TOTAL TIME (Secs.)
125	26	.39	2.67	3.06
216	37	.80	7.25	8.05
343	50	1.66	17.18	18.84
512	65	3.21	44.19	47.40
729	82	5.67	96.24	101.91
1000	101	9.14	163.59	172.72
1331	122	14.69	347.39	362.08
1728	145	22.54	585.77	608.31

TABLE 5 - A FINAL COMPARISON

N	M	UNIVAC 1108 ASSEMBLY LDL^T DECOMPOSITION TIMES (Secs.)	CDC 6400 FORTRAN LDL^T DECOMPOSITION TIMES (Secs.)	CDC 6400 CSKYDG TOTAL SOLUTION TIMES
100	11	.343	4.43	.76
225	16	1.657	23.22	3.55
400	21	5.921	83.04	6.65

ACKNOWLEDGMENTS

The author wishes to thank the following people for their interest and assistance: Professor Charles K. Mesztenyi (University of Maryland Computer Science Center); Dr. Elizabeth H. Cuthill (NSRDC Code 1805); Mr. Michael E. Golden (NSRDC Code 1844); and Dr. Ed Cohen (Naval Ordnance Laboratory).

APPENDIX - PROGRAM LISTINGS

```
      SUBROUTINE SETUP(NS,W)
C
      DIMENSION X(250),II(250),JJ(250),XX(250)
      DIMENSION W(1)
C
      IO1=4
      IO2=5
      L=0
      READ(IO1)NN,MM
      REWIND IO2
    1 WRITE(IO2)NN
      DO 5  I=1,NN
      READ(IO1) (X(K),K=1,MM)
      DO 4  K=1,MM
      IF (X(K)) 2,4,2
    2 L=L+1
      II(L)=I
      JJ(L)=I+K-1
      XX(L)=X(K)
      IF (L-NS) 4,3,4
    3 WRITE(IO2) (II(KK),JJ(KK),XX(KK),KK=1,NS)
      L=0
    4 CONTINUE
    5 CONTINUE
      KLIM=L+1
      DO 6  K=KLIM,NS
      II(K)=1
      JJ(K)=1
    6 XX(K)=0.
      WRITE(IO2)(II(KK),JJ(KK),XX(KK),KK=1,NS)
      REWIND IO2
C
      READ(IO1)(W(K),K=1,NN)
C
      RETURN
      END
```

```
      SUBROUTINE READ(NS)
************************************************************************
*                                                                      *
*        THE READ SUBROUTINE READS THE NON-ZERO ELEMENTS OF A          *
*     SYMMETRIC MATRIX FROM UNIT IO, AND SETS UP THE CORRESPONDING     *
*     ARC-GRAPH. THE ELEMENTS ARE READ BY TRIPLETS,                    *
*                          I,J,B(I,J)                                  *
*     WITH THE 1,0,0. TRIPLET INDICATING THE END OF INPUT RECORD.      *
*     ONLY THE UPPER DIAGONALS SHOULD BE GIVEN TOGETHER WITH THE       *
*     DIAGONAL ELEMENTS.                                               *
*                                                                      *
************************************************************************
C  STORAGE ASSIGNMENT FOR SYMMETRIC DECOMPOSITION                      SB000200
C  MX - THE MAXIMUM ALLOWED NUMBER OF NON-ZERO COEFFICIENTS,           SB000300
C  NX - THE MAXIMUM ALLOWED SIZE OF THE MATRIX.                        SB000400
      COMMON /DIM/ MX, NX                                              SB000500
C  ARRAYS OF SIZE M -                                                  SB000700
C  R - INTEGER ARRAY USED FOR ROW LINKAGE                              SB000800
C  C - INTEGER ARRAY USED FOR COLUMN LINKAGE                           SB000900
C  L - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )                       SB001000
C  T - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )                       SB001100
C  B - FL. PT. ARRAY CONTAINING THE VALUES OF THE COEFFICIENTS         SB001200
C     THE ARRAYS R,C,L AND T CONTAIN PACKED INTEGERS.
      COMMON /ARRAYM/ R(5000),C(5000),L(5000),T(5000),B(20000)
      INTEGER R,C,T                                                    SB001500
C  ARRAYS OF SIZE N -                                                  SB001600
C  IP - INTEGER ARRAY CONTAINS THE SEQUENCE OF PIVOTS                  SB001700
C  ND - INTEGER ARRAY CONTAINS THE NUMBER OF ELEMENTS                  SB001800
C     IN A ROW OF THE UNDECOMPOSED PART OF THE MATRIX                  SB001900
C  IH1, IH2 - ARE TEMPORARY INTEGER ARRAYS, ONE OF THEM                SB002000
C     MAY BE EQUIVALENCED TO IP                                        SB002100
      COMMON /ARRAYN/ IP(500),ND(500),IH1(500),IH2(500)
C  INDIVIDUAL DATA -                                                   SB002300
C  M - NUMBER OF NONZERO ELEMENTS                                      SB002400
C  N - SIZE OF THE MATRIX                                              SB002500
C  UP - PIVOT SELECTION PARAMETER                                      SB002600
C  CT1 - MAXIMUM ELEMENT IN THE ORIGINAL MATRIX                        SB002700
C  CT2 - MAXIMUM ELEMENT ENCOUNTERED DURING DECOMPOSITION              SB002800
      COMMON /DATA/ M,N,CT1,CT2
C
      DIMENSION II(250),JJ(250),VV(250)
C
      MX=20000
      NX=1000
      IO=5
      CT1 = 0.0                                                        RS001700
C  LOOP TO ESTABLISH DIAGONALS                                         RS001800
      READ(IO) N
      DO 10 I=1,N                                                      RS001900
      R(I)=I
      C(I)=I
      L(I)=0
      T(I)=0
      ND(I)=0
10    B(I) = 0.                                                        RS002500
```

```
      M = N                                                    RS002600
C   LOOP TO READ ELEMENTS                                      RS002700
      LLIM=((N*(N+1))/2)/NS+2
      DO 25 LL=1,LLIM
      READ(IO) (II(I),JJ(I),VV(I),I=1,NS)
C    **** MODIFY THE FOLLOWING FORMAT AS NEEDED ****           RS002900
      DO 20 K=1,NS
      I=II(K)
      J=JJ(K)
      V=VV(K)
      IF (ABS(V).GT.CT1) CT1=ABS(V)                            RS003100
      IF (I-J) 45,50,60
C   ESTABLISH ARC FOR OFF-DIAGONAL ELEMENT                     RS003700
45    M = M+1                                                  RS003800
      IF (M.GT.MX) GO TO 70                                    RS003900
      B(M) = V                                                 RS004000
      R(M)=R(I)
      C(M)=C(J)
      R(I)=M
      C(J)=M
      L(M)=0
      T(M)=0
      ND(I)=ND(I+1)
      ND(J)=ND(J+1)
      GO TO 20                                                 RS004900
C   STORE DIAGONAL                                             RS005000
50    B(I) = V                                                 RS005100
      ND(I)=ND(I+1)
   20 CONTINUE
   25 CONTINUE
C   END OF MATRIX ELEMENTS
60    RETURN                                                   RS005400
C                                                              RS005500
C   ERROR-INSUFFICIENT STORAGE                                 RS005600
70    WRITE (6,80)
80    FORMAT (21H0INSUFFICIENT STORAGE)                        RS005800
      STOP                                                     RS005900
      END                                                      RS006000
```

```
      SUBROUTINE LU(IPONE,UP)
*********************************************************************
*                                                                   *
*        THE LU SUBROUTINE DECOMPOSES THE SYMMETRIC MATRIX.         *
*                                                                   *
*********************************************************************
C
C  THE ROUTINE DECOMPOSES THE SYMMETRIC
C  MATRIX WHICH HAD BEEN ESTABLISHED BY
C  THE READ ROUTINE. IF IPONE IS ZERO,
C  THEN THE ROUTINE SELECTS THE PIVOTS
C  ACCORDING TO THE PIVOTING STRATEGY,
C  OTHERWISE IT ASSUMES THAT THE ORDER
C  OF PIVOTS IS PROVIDED IN THE ARRAY IP.
C
C  THE ROUTINE USES TAG T FOR MARKING THE
C  THE ALREADY DECOMPOSED PART OF THE MATRIX.
C  IT USES TAG L SUCH THAT THE ROW OF A
C  PIVOT IS THE UNION OF THE SET OF LEFT
C  CONNECTED ARCS WITH TAG L = 1, AND THE
C  SET OF RIGHT CONNECTED ARCS WITH TAG L = 0.
C
C  STORAGE ASSIGNMENT FOR SYMMETRIC DECOMPOSITION
C  MX - THE MAXIMUM ALLOWED NUMBER OF NON-ZERO COEFFICIENTS,
C  NX - THE MAXIMUM ALLOWED SIZE OF THE MATRIX.
      COMMON /DIM/ MX, NX
C  ARRAYS OF SIZE M -
C  R - INTEGER ARRAY USED FOR ROW LINKAGE
C  C - INTEGER ARRAY USED FOR COLUMN LINKAGE
C  L - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )
C  T - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )
C  B - FL. PT. ARRAY CONTAINING THE VALUES OF THE COEFFICIENTS
      COMMON /ARRAYM/ R(5000),C(5000),L(5000),T(5000),B(20000)
      INTEGER R,C,T
C  ARRAYS OF SIZE N -
C  IP - INTEGER ARRAY CONTAINS THE SEQUENCE OF PIVOTS
C  ND - INTEGER ARRAY CONTAINS THE NUMBER OF ELEMENTS
C     IN A ROW OF THE UNDECOMPOSED PART OF THE MATRIX
C  IH1, IH2 - ARE TEMPORARY INTEGER ARRAYS, ONE OF THEM
C     MAY BE EQUIVALENCED TO IP
      COMMON /ARRAYN/ IP(500),ND(500),IH1(500),IH2(500)
C  INDIVIDUAL DATA -
C  M - NUMBER OF NONZERO ELEMENTS
C  N - SIZE OF THE MATRIX
C  UP - PIVOT SELECTION PARAMETER
C  CT1 - MAXIMUM ELEMENT IN THE ORIGINAL MATRIX
C  CT2 - MAXIMUM ELEMENT ENCOUNTERED DURING DECOMPOSITION
      COMMON /DATA/ M,N,CT1,CT2
C
      LOGICAL SWP
C
      CT2 = CT1
      SWP = .TRUE.
      IF ( IPONE.NE.0) SWP=.FALSE.
C  LOOP FOR THE N PIVOTS
```

```
      DO 300 I=1,N
      IF (SWP)  GO TO 30
      IF (I.NE.1)  GO TO 20
C  SAVE THE PIVOTS  IN ND IF  THEY WERE GIVEN
      DO 10  J=1,N
   10 ND(J)=IP(J)
C  GET  THE  NEXT GIVEN PIVOT
   20 IX=ND(I)
      IP(I)=IX
      GO TO 135
C  PIVOTS  WERE  NOT  GIVEN, THUS FIND ONE
   30  DMAX  = 0.
      IF (UP.LE.0.)  GO  TO 110
C  FIND MAXIMAL ELEMENT IN THE AVAILABLE DIAGONALS
      DO 100 J=1,N
      IF (T(J).EQ.1)  GO TO  100
      DMAX=AMAX1(DMAX,ABS(B(J)))
  100  CONTINUE
      DMAX  = UP*DMAX
C  NOW FIND AN  AVAILABLE DIAGONAL WHOSE VALUE IS
C  GREATER  THAN DMAX AND HAS MINUMUM NUMBER OF
C  ELEMENTS IN  ITS ROW
  110  IX =  0
      XM =  0.
      DO 130 J=1,N
      IF (T(J).EQ.1 .OR. B(J).EQ.0.)
     1     GO TO 130
      IF (ABS(B(J)).LT.DMAX)
     1  GO  TO 130
      IF (IX.EQ.0)  GO  TO 120
      IF (ND(J)-NY)  120,119,130
  119 IF(ABS(B(J)).LE.XM)GOTO 130
  120  IX =  J
      XM =  ABS(B(J))
      NY=ND(J)
  130  CONTINUE
      IF ((IX.EQ.0).OR.
     1  (XM.LT.1.E-20))  GO  TO 900
      IP(I)=IX
C  NOW THE  PIVOT IS IN  IX,  SET ITS  TAG T
  135  K = I
      T(IX)=1
      XMAX  = B(IX)
C  LOOP TO  COLLECT  THE  ELEMENTS  IN  THE  ROW  OF  THE  PIVOT
      IY =  IX
C  COLLECT  THE  LEFT CONNECTED ARCS
C  WITH THEIR RIGHT INDEX VALUE
  140 IY=R(IY)
      IF (IY.EQ.IX)  GO  TO 170
      IF (L(IY).EQ.1)  GO TO  140
      T(IY)=1
      IZ=IY
  160 IZ=C(IZ)
      IF (IZ.GT.N)  GO TO 160
      K = K+1
```

```
      IH1(K)=IY
      IH2(K)=IZ
      IF (SWP) ND(IZ)=ND(IZ-1)
      GO TO 140                                                          LUS08500
 170  IY = IX                                                            LUS08600
C  COLLECT THE RIGHT CONNECTED ARCS                                      LUS08700
C  WITH THEIR LEFT INDEX VALUES                                          LUS08800
  180 IY=C(IY)
      IF (IY.EQ.IX) GO TO 195                                            LUS09000
      IF (T(IY).EQ.1) GO TO 180
      IZ = IY                                                            LUS09200
  190 IZ=R(IZ)
      IF (IZ.GT.N) GO TO 190                                             LUS09400
      K = K+1                                                            LUS09500
      IH1(K)=IY
      IH2(K)=IZ
      IF (SWP) ND(IZ)=ND(IZ-1)
      L(IY)=1
      T(IY)=1
      GO TO 180                                                          LUS10100
 195  IF (K.EQ.I) GO TO 300                                              LUS10200
      K1 = I+1                                                           LUS10300
C  LOOP ON THE COLLECTED ARCS                                            LUS10400
      DO 250 J1=K1,K                                                     LUS10500
      IY=IH1(J1)
      IZ=IH2(J1)
      Y = B(IY)                                                          LUS10800
C  DIVIDE THE ELEMENT BY THE PIVOT AND                                   LUS10900
C  MODIFY ITS CORRESPONDING DIAGONAL                                     LUS11000
      B(IY) = B(IY)/XMAX                                                 LUS11100
      B(IZ) = B(IZ) - B(IY)*Y                                            LUS11200
      IF (ABS(B(IY)).GT.CT2)                                             LUS11300
     1   CT2 = ABS(B(IY))                                                LUS11400
      IF (ABS(B(IZ)).GT.CT2)                                             LUS11500
     1   CT2 = ABS(B(IZ))                                                LUS11600
      IF (J1.EQ.K) GO TO 250                                             LUS11700
      K2 = J1+1                                                          LUS11800
C  INSIDE LOOP FOR THE REST OF THE COLLECTED ARCS                        LUS11900
C  TO MODIFY THE INTERSECTING ARCS                                       LUS12000
      DO 240 J2=K2,K                                                     LUS12100
      IY1=IH1(J2)
      IZ1=IH2(J2)
C  FIND THE ARC W BETWEEN IZ-IZ1 DIRECTED FROM                           LUS12400
C  MIN(IZ,IZ1) TO MAX(IZ,IZ1) IF EXISTS                                  LUS12500
      I1 = MIN0(IZ,IZ1)                                                  LUS12600
      I2 = MAX0(IZ,IZ1)                                                  LUS12700
      L1=R(I1)
      L2=C(I2)
 200  IF (L1.EQ.I1.OR.L2.EQ.I2)                                          LUS13000
     <    GO TO 220                                                      LUS13100
      IF (L1.EQ.L2) GO TO 230                                            LUS13200
      IF (L1.GT.L2) GO TO 210                                            LUS13300
      L2=C(L2)
      GO TO 200                                                          LUS13500
  210 L1=R(L1)
```

```
      GO TO 230                                                         LUS13700
C  IT DOES NOT EXIST, THUS CREATE ONE
 220  IF (M.EQ.MX) GO TO 950                                            LUS13900
      M = M+1                                                           LUS14000
      IF (SWP) ND(I1)=ND(I1+1)
      IF (SWP) ND(I2)=ND(I2+1)
      L1 = M                                                            LUS14300
      IF (L1.GE.16000) WRITE(6,261)L1
  261 FORMAT(1X,3HAAA,I15)
      B(L1) = 0.                                                        LUS14400
      R(L1)=R(I1)
      C(L1)=C(I2)
      R(I1)=L1
      C(I2)=L1
      L(L1)=0
      T(L1)=0
C  MODIFY THE VALUE OF THE INTERSECTING ARC                             LUS15100
 230  B(L1) = B(L1)-B(IY)*B(IY1)                                        LUS15200
      IF (ABS(B(L1)).GT.CT2)                                            LUS15300
     1   CT2 = ABS(B(L1))                                               LUS15400
 240  CONTINUE                                                          LUS15500
 250  CONTINUE                                                          LUS15600
 300  CONTINUE                                                          LUS15700
      RETURN                                                            LUS15800
C                                                                       LUS15900
C  **** ERRORS ****                                                     LUS16000
C  NUMERICALLY SINGULAR                                                 LUS16100
 900  WRITE (6,910) I                                                   LUS16200
 910  FORMAT (24H0NUMERICALLY SINGULAR AT,I5)                           LUS16300
      STOP                                                              LUS16400
C  INSUFFICIENT STORAGE                                                 LUS16500
 950  WRITE (6,960) I                                                   LUS16600
 960  FORMAT (24H0INSUFFICIENT STORAGE AT,I5)                           LUS16700
      STOP                                                              LUS16800
      END                                                               LUS16900
```

```
      SUBROUTINE SOLVE(W)                                                  SS000100
***********************************************************************
*                                                                     *
*        THE SOLVE SUBROUTINE SOLVES THE THE SYMMETRIC LINEAR SYSTEM  *
*     FOR THE GIVEN RIGHT HAND SIDE W. THE SOLUTION IS RETURNED       *
*     THROUGH W. THE COEFFICIENT MATRIX MUST HAVE BEEN DECOMOSED PRIOR *
*     TO CALLING THIS SUBROUTINE. THE CORRESPONDING SEQUENCE OF PIVOTS *
*     IS ASSUMED TO BE IN THE IP ARRAY.                               *
*                                                                     *
***********************************************************************
C  LOOP TO SOLVE LOWER TRIANGULAR SYSTEM                               SS001400
C
C  STORAGE ASSIGNMENT FOR SYMMETRIC DECOMPOSITION                      SB000200
C  MX - THE MAXIMUM ALLOWED NUMBER OF NON-ZERO COEFFICIENTS,           SB000300
C  NX - THE MAXIMUM ALLOWED SIZE OF THE MATRIX.                        SB000400
      COMMON /DIM/ MX, NX                                              SB000500
C  ARRAYS OF SIZE M -                                                  SB000700
C  R - INTEGER ARRAY USED FOR ROW LINKAGE                              SB000800
C  C - INTEGER ARRAY USED FOR COLUMN LINKAGE                           SB000900
C  L - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )                       SB001000
C  T - INTEGER ARRAY USED FOR TAGGING ( 0 OR 1 )                       SB001100
C  B - FL. PT. ARRAY CONTAINING THE VALUES OF THE COEFFICIENTS         SB001200
C        THE ARRAYS R,C,L AND T CONTAIN PACKED INTEGERS.
      COMMON /ARRAYM/ R(5000),C(5000),L(5000),T(5000),B(20000)
      INTEGER R,C,T                                                    SB001500
C  ARRAYS OF SIZE N -                                                  SB001600
C  IP - INTEGER ARRAY CONTAINS THE SEQUENCE OF PIVOTS                  SB001700
C  ND - INTEGER ARRAY CONTAINS THE NUMBER OF ELEMENTS                  SB001800
C     IN A ROW OF THE UNDECOMPOSED PART OF THE MATRIX                  SB001900
C  IH1, IH2 - ARE TEMPORARY INTEGER ARRAYS, ONE OF THEM                SB002000
C     MAY BE EQUIVALENCED TO IP                                        SB002100
      COMMON /ARRAYN/ IP(500),ND(500),IH1(500),IH2(500)
C  INDIVIDUAL DATA -                                                   SB002300
C  M - NUMBER OF NONZERO ELEMENTS                                      SB002400
C  N - SIZE OF THE MATRIX                                              SB002500
C  UP - PIVOT SELECTION PARAMETER                                      SB002600
C  CT1 - MAXIMUM ELEMENT IN THE ORIGINAL MATRIX                        SB002700
C  CT2 - MAXIMUM ELEMENT ENCOUNTERED DURING DECOMPOSITION              SB002800
      COMMON /DATA/ M,N,CT1,CT2
C
      DIMENSION W(1)                                                   SS000200
C
      DO 100 I=2,N                                                     SS001500
      IX=IP(I)
      IY=IX
   20 IY=R(IY)
      IF (IY.EQ.IX) GO TO 40                                           SS001900
      IF (L(IY).EQ.0) GO TO 20
      IZ = IY                                                          SS002100
   30 IZ=C(IZ)
      IF (IZ.GT.N) GO TO 30                                            SS002300
      W(IX) = W(IX)-W(IZ)*B(IY)                                        SS002400
      GO TO 20                                                         SS002500
  40  IY = IX                                                          SS002600
   50 IY=C(IY)
```

```
      IF (IY.EQ.IX) GO TO 100                                   SS002800
      IF (L(IY).EQ.1) GO TO 50
      IZ = IY                                                   SS003000
   60 IZ=R(IZ)
      IF (IZ.GT.N) GO TO 60                                     SS003200
      W(IX) = W(IX)-W(IZ)*B(IY)                                 SS003300
      GO TO 50                                                  SS003400
  100 CONTINUE                                                  SS003500
C  LOOP TO SOLVE DIAGONAL SYSTEM                                SS003600
      DO 110 I=1,N                                              SS003700
  110 W(I) = W(I)/B(I)                                          SS003800
C  LOOP TO SOLVE TRANSPOSED UPPER TRIANGULAR SYSTEM             SS003900
      DO 200 I=2,N                                              SS004000
      K = N+1-I                                                 SS004100
      IX=IP(K)
      IY = IX                                                   SS004300
  120 IY=R(IY)
      IF (IY.EQ.IX) GO TO 140                                   SS004500
      IF (L(IY).EQ.1) GO TO 120
      IZ = IY                                                   SS004700
  130 IZ=C(IZ)
      IF (IZ.GT.N) GO TO 130                                    SS004900
      W(IX) = W(IX)-W(IZ)*B(IY)                                 SS005000
      GO TO 120                                                 SS005100
  140 IY = IX                                                   SS005200
  150 IY=C(IY)
      IF (IY.EQ.IX) GO TO 200                                   SS005400
      IF (L(IY).EQ.0) GO TO 150
      IZ = IY                                                   SS005600
  160 IZ=R(IZ)
      IF (IZ.GT.N) GO TO 160                                    SS005800
      W(IX) = W(IX)-W(IZ)*B(IY)                                 SS005900
      GO TO 150                                                 SS006000
  200 CONTINUE                                                  SS006100
      RETURN                                                    SS006200
      END                                                       SS006300
```

INITIAL DISTRIBUTION

Copies:		Copies:	
1	ONR 430/R. LUNDEGARD	1	NAVSHIPYD PTSMH/LIB
1	ONR 432/L. BRAM	1	NAVSHIPYD BSN/LIB
1	NRL/8441/J. HANSON	1	NAVSHIPYD PHILA/LIB
1	USNA DEPT MATH	1	NAVSHIPYD NORVA/LIB
1	USNA LIB	1	NAVSHIPYD CHASN/LIB
1	NAVPGSCOL/59Ci/G. CANTIN	1	NAVSHIPYD MARE/LIB
1	NAVPGSCOL/MATH DEPT	1	NAVSHIPYD BREM/LIB
1	NAVPGSCOL LIB	1	NAVSHIPYD PEARL/LIB
1	NROTC & NAVADMINU, MIT	1	AIR FORCE AERO RES LABS/P. NIKOLAI
1	NAVWARCOL	12	DDC
1	NOL 331/E. COHEN		

CENTER DISTRIBUTION

Copies:		Copies:	
1	0000 NELSON PERRY W	1	1843 SCHOT JOANNA WOOD
1	0100 POWELL ALAN	1	1844 DHIR SURENDRA K
1	1725 GIFFORD LEROY N JR	1	1844 EVERSTINE GORDON C
1	1725 JONES REMBERT F JR	10	1844 GIGNAC DONALD A
1	1725 RODERICK JOAN E	1	1844 GOLDEN MICHAEL M
1	1725 ROTH PETER N	1	1844 HENDERSON FRANCIS M
1	1745 NG CHRISTOPHER	1	1844 MATULA PETRO
1	1800 GLEISSNER GENE H	1	1850 CORIN THOMAS
1	1802 SHANKS DANIEL	1	1860 SULIT ROBERT A
1	1802 FRENKIEL FRANCOIS N	1	1880 CAMARA ABEL W
1	1802 LUGT HANS J	1	1890 GRAY GILBERT R
1	1802 THEILHEIMER FEODOR	1	1890 TAYLOR NORA M
1	1805 CUTHILL ELIZABETH H	1	1892 GOOD SHARON E
1	1830 ERNST HERBERT M	1	1892 RUMSEY JUDITH J
1	1830 CULPEPPER LINWOOD M	1	1966 LIU YUAN-NING
1	1830 WALTON THOMAS S	30	5614 REPORTS DISTRIBUTION
1	1840	1	5641 LIBRARY
1	1842 EDDY ROBERT P	1	5642 LIBRARY
1	1842 MEALS L KENTON		