TR/04/87 March 1987

A Program To Reorder And Solve Sparse Unsymmetric Linear Systems Using the Envelope Method

J. J. Judice

G. Mitra

M. Tamiz

A PROGRAM TO RECRDER AND SOLVE SPARSE UNSYMMETRIC LINEAR SYSTEMS USING THE ENVELOPE METHOD

J.J. JUDICE[+]

G. MITRA[*]

M. TAMIZ[*]

[+] Departamento de Matematica, Universidade de Coimbra, Portugal.

[*] Department of Mathematics and Statistics, Brunel University, England.

ABSTRACT

The envelope data structure and the Choleski based (bordering) method for the solution of symmetric sparse systems of linear equations have been extended by the authors to solve unsymmetric systems of linear equations. The data structures used in this general linear equation Solver and a set of FORTRAN 77 subroutines are described. Some test data (extracted from LP problems as basis matrices) together with experimental results are presented.

A PROGRAM TO REORDER AND SOLVE SPARSE UNSYMMETRIC LINEAR SYSTEMS USING THE ENVELOPE METHOD

J.J. JUDICE
Departamento de Matematica , Universidade de Coimbra , Portugal.

G . MITRA and M . TAMIZ
Department of Mathematics and Statistics, Brunel University, England.

1. Introduction

Direct methods for solving sparse linear systems use Gaussian Elimination method in a combination with reordering of the coefficient matrix to preserve sparsity. When the matrix is symmetric positive definite then there are a number of algorithms to reorder the rows and columns of the matrix (for a description of the main algorithms see [6]). After the ordering has been found the so called ANALYSE PHASE terminates and the data structure for FACTOR PHASE is set up. In this phase the LL^T or LDL^T decomposition of the matrix is obtained. At this stage solving the system amounts to solving two triangular systems (this is called the SOLVE PHASE). The process of obtaining this decomposition is "static", that is, the data structure remains unaltered after being set up at the end of the ANALYSE PHASE .

If the matrix is unsymmetric then a "dynamic" process has to be used to factorize the matrix A. The permutations of the rows and columns of the matrix are dictated by sparsity and stability requirements during the factorization [2]. It is, in general, not possible to predict where fill-in occurs and the initial data structure is modified during the process in order to allocate storage for this fill-in as the factorization proceeds.

The advantage of the static processes over the dynamic schemes and of the separation of the phases ANALYSE and FACTOR is nowadays well accepted (see for instance [2,5]). One of the main static schemes for symmetric positive definite systems is the so-called ENVELOPE METHOD [6, chapter 4]. In [8] we have developed a generalization of this method to unsymmetric matrices. As in the symmetric case the method uses a preassigned sequence of diagonal pivots and exploits static data structures. The occurence of a zero diagonal pivot is overcome by a novel method based on the Schur Complement update. In this paper our main interest is to describe a program which carries out the general solution process.

The contents of the paper are organized in the following way. In Section 2 we provide a summary description of the different algorithmic phases of the procedure and in section 3 the function and use of the important subroutines of the program are described. The data structures are considered in section 4 and finally, in section 5, we present the experimental results together with the test data.

2. The Main Algorithmic Phases

In this section we briefly describe the three phases of the whole procedure. The ANALYSE PHASE is carried out by a method which is an extension of the envelope method for unsymmetric matrices. This procedure reorders the matrix A by a symmetric permutation P so that all the nonzero elements of the permuted matrix $B=P^TAP$ are brought nearest to the diagonal. For a symmetric matrix A this consists of two combinatorial algorithms (GPS and RCM) which operate on the undirected graph associated with A. These algorithms employ the degree of a node. For unsymmetric matrices this measure is replaced by the "directed degree" which we define as

$$\deg (V_k) = 100*(\text{outdeg} * \text{indeg}) + (\text{outdeg} + \text{indeg}) ,$$

where outdeg and indeg are the number of arcs of the directed graph leaving and entering the node V_k. This is a nominal extension of the celebrated Markowitz criterion and is designed to break ties which occur quite often if the latter is adopted in its original form. This measure is used to extend the GPS and RCM algorithms which produce the desired symmetric permutation of the matrix A. In the last step of the ANALYSE PHASE the static envelope data structure is constructed for the permuted matrix.

In the FACTOR PHASE we apply the bordering method [6, page 89] and try to obtain the LU decomposition of the permuted matrix. In each iteration a row of the matrix L and a column of the matrix U are computed. The procedure may break down if the leading diagonal element takes the value zero (in the program an absolute value less than the chosen pivot tolerance XTOL) is found. In this situation we add +1 (unity) to the leading (zero) diagonal element and continue with the factorization. Let p denote the number of such occurences (in the program the value of p is stored in the variable ADCL). At the end of the factorization phase we obtain the LU decomposition of the matrix B+D where D is a diagonal matrix with unit diagonal elements in those positions which required addition of unit coefficients. The solution of the system

$$B x = b \qquad (1)$$

is equivalent to solving the augmented system

$$\begin{aligned} (B+D)\,x - E\,y &= b \\ -E^T x + I\,y &= 0 \end{aligned} \qquad (2)$$

where I is the identity matrix of order p, and $E \in R^{nxp}$ is a rectangular matrix with unit columns which match the unit entries of D in the row positions.

The solution of (2) is obtained by solving two systems with the matrix (B+D) and one system with the Schur Complement matrix of order p given by

$$S_c = I - E^T \{B + D) ^{-1} E. \qquad (3)$$

The system set out in (2) is only considered implicitly. The value p is usually quite small and the Schur Complement matrix S_c is computed explicitly. To obtain S_c the already computed LU decomposition of B+D is used together with the integer array INDMAT of dimension p which compactly represents the matrix E. The LU decomposition of S_c is obtained by partial pivoting [4] and this completes the FACTOR PHASE.

The SOLVE PHASE consists of solving the system (1) and two cases may occur as presented below.

(i) If p=0 then system (1) is solved by using the computed LU decomposition of B.

(ii) If p>0 then system (2) is solved implicitly as explained before by using the computed LU decompositions of the matrices B+D and S_c.

3. Description of the Subroutines

The program assumes that the matrix has a zero-free diagonal. This is a reasonable assumption since well known graph theoretic algorithms exist that perform row and column permutations to put the matrix in this form [1,3]. The program starts by calling the subroutine INPUT, which reads the nonzero matrix elements and constructs the column-wise representation of the matrix.

The next subroutine to be called is named ROWISE and obtains the data structure for the row-wise nonzero representation of the matrix [7]. Using the column-wise and row-wise representations of the matrix we can find the adjacency lists of the innergraph and outergraph associated with the matrix [8]. This is performed by the subroutines FDINGR and FDOUGR respectively.

The ANALYSE PHASE is carried out next and consists of finding an ordering for the columns and rows of the matrix. We do this by modifying the process described in [6, Chapter 4] and our method is an extension to this procedure. This algorithm is fully explained in [8] and is performed by the subroutine GENRCM, which in turn calls the four subroutines FDDEG, FDROOT, ROOTLS, RCMS. The calling sequence and dependencies are shown in Display 1.

GENRC

- - -

FDDEG

- - -

FDROOT- - - →ROOTLS

- - -

RCMS

DISPLAY 1

The subroutine FDDEG, finds the "directed degree" of the nodes of the directed graph associated with the matrix. These quantities are stored in the real vector DEG and we have used this measure to extend the Cuthill Mckee (CM) and Reverse Cuthill McKee (RCM) algorithms to directed graphs [8], These extended algorithms are presented in the subroutine RCMS which needs a starting node ROOT. This node is obtained by the extension of GPS algorithm [6, Chapter 4] to directed graphs. This is achieved by the subroutines FDROOT and ROOTLS which are minor extensions of similar routines presented in [6 , Chapter 4].

The ordering process is made for each connected component of the directed graph associated with A, that is, for each diagonal block of the matrix. These connected components are specified by an integer vector MASK in the same way as explained in [6, Chapter 4]. The final ordering is given by an integer array PERM, where

$$\mathrm{PERM}(i) = j \qquad (4)$$

means that the jth initial row and column of the matrix is the ith row and column of the permuted matrix.

The ANALYSE PHASE is completed by constructing the envelope data structure of the permuted matrix. To achieve this the subroutines FENVRW and FENVCL are first called, which yield the (pointer) vectors XENVRW and XENVCL respectively. These subroutines use the adjacency lists of the inner and outer graphs and the integer array INVP. INVP represents the inverse of the permutation defined by PERM, whereby,

$$\mathrm{INVP}(\mathrm{PERM}(i)) = i, \text{ for all } i \qquad (5)$$

The subroutine INVRSE constructs INVP. Subsequently, the remaining vectors EVRW, EVCL, and DIAG are constructed by the subroutine ENVMAT.

The subroutine FACTOR carries out the LU decomposition of the FACTOR PHASE. When necessary the Schur Complement matrix is computed by the subroutine SCHCOM. These two subroutines call LOWSOL since each needs to solve lower triangular systems. The subroutine DECOMP computes the LU decomposition of the Schur Complement matrix S_c.

In order to establish the correctness and accuracy of the decompositions a VERIFICATION PHASE is incorporated. This phase consists of solving the system

$$Bx = b \qquad (6)$$

where

$$b = Ae \qquad (7)$$

and e is a vector with unit components.
If $\bar{x}$ is the computed solution then the accuracy of the decomposition is measured by the quantity

$$\text{ERROR} = ||\bar{x} - e||_\infty = \max_i |\bar{x}_i - 1| \qquad (8)$$

and smaller value of ERROR implies better accuracy. For this purpose a subroutine INIVER is first called in which the vector b = Ae is calculated by using the data structure of the initial matrix and the array INVP. The subroutine GETRHS solves the system (6) by the method outlined in Section 2 and calls the subroutines LOWSOL, UPSOL and SOLVE. The first two subroutines carry out solution of lower and upper triangular systems using the envelope data structure. The subroutine SOLVE processes the two triangular systems which are given by the dense LU decomposition of S_c.

It is quite straightforward to adopt this suite of subroutines to solve a linear system Ax=b, where b is any right hand side vector. It is sufficient to modify the subroutine INIVER so that it reads the vector b and constructs the vector RHS in the order induced by the array PERM defined earlier in this section.

4. Description of the Data Structures

In this section we describe the main data structures referred to in section 2 and section 3. These data structures include the column-wise and row-wise representations of the original matrix, the adjacency lists of the inner and outer graphs associated with the original matrix, the level tree which is used by the GPS algorithm and finally the envelope representation of the permuted matrix. A number of one dimensional arrays of integer (INTEGER*2) and real (REAL*4) words are used. These arrays are dimensioned by global variables which are defined below.

MROW = number of rows (and columns) of the matrix.
NONZER = number of nonzero elements of the original matrix.
NZNDG = NONZER – MROW = number of non-zero off - diagonal elements of the original matrix.
ENVRW(ENVCL) = number of elements which are stored in strictly lower (upper) part of the envelope of the permuted matrix.

The column-wise representation of the original matrix is given by two integer arrays PTCL and ELCL of dimensions (MROW+1) and NONZER respectively and a real array VMATCL of dimension NONZER. The arrays ELCL and VMATCL contain the row positions and the numerical values of the nonzero elements of the original matrix. The array PTCL is such that PTCL(k) points to the location of the first nonzero element of column k represented in the arrays ELCL and VMATCL.

The row-wise representation of the matrix structure is given by two integer arrays PTRW and ELRW which are comparable to PTCL and ELCL respectively. The actual coefficient values are not given in this representation as this would lead to unnecessary duplication. For the matrix shown in Display 2 the data structures are illustrated by the contents of these arrays set out in Display 3.

The inner graph and the outer graph of a matrix are represented by the adjacency lists stored in arrays ADJNCL, ADJNRW. These arrays locate the row and column positions of the off-diagonal elements. Two arrays of pointers XADJCL, XADJRW which are comparable to PTCL and PTRW are also required. The contents of these arrays for the example are shown in Display 4.

The level tree for the GPS algorithm is given by the two integer arrays XLS and LS, which are explained in [6, Chapter 4], The envelope representation of the permuted matrix consists of five different arrays. DIAG is a real array of dimension MROW, and contains all the diagonal elements of the permuted matrix in the order induced by the array PERM. The arrays EVRW and EVCL are real arrays of dimensions ENVRW and ENVCL respectively. ENVRW, ENVCL contain the number of words reserved to store the rows of the strictly lower triangular part and the columns of the strictly upper triangular part of the permuted matrix. XENVRW and XENVCL are integer arrays of dimension (MROW+1) and their contents point to the first nonzero position of each row and column as contained in ENVRW and ENVCL respectively. If we assume that the matrix in Display 2 is already permuted then its envelope data structure is given by the arrays shown in Display 5.

The program has been designed in such a way that all the arrays are created in contiguous work space provided by the user and consists of an integer (INTEGER*2) array ISTOR and a real (REAL*4) array RSTOR. The dimension of these two arrays has been set to 10,000 but obviously can be modified if required.

Since the three phases ANALYSE, FACTOR and VERIFY are processed sequentially, some of the arrays required in one phase may not be used subsequently. This permits overlaying of storage and reduces the total amount of storage needed. This is easily achieved by the use of suitable start pointers and this strategy is followed in different parts of the program. The integer and real storage areas, together with their overlays, are shown in Display 6.

The matrix data is input by following the coordinate scheme for specifying the nonzero element values. The output is designed to provide a number of useful statistics. These include ERROR, MROW, NONZER, ENVSZE, INTSPA, RELSPA, and also the growth factor GROWTH [8]. The number of multiplications/divisions required to perform the LU decompositions is also computed and is given by the double precision variable OPSF.

$$A = \begin{bmatrix} 2.0 & & 1.0 & & & \\ & 5.0 & 3.0 & 2.0 & & 2.0 \\ & 3.0 & 4.0 & 1.0 & & 6.0 \\ 3.0 & & & 6.0 & & \\ & & & & 1.0 & 1.0 \\ & 5.0 & 2.0 & 3.0 & & 3.0 \end{bmatrix}$$

DISPLAY 2

MROW=6 NONZER=18 NZNDG=12

PTCL = 1 3 6 10 14 15 19

ELCL = 1 4 2 3 6 1 2 3 6 2 3 4 6 5 2 3 5 6

VMATCL= 2.0 3.0 5.0 3.0 5.0 1.0 3.0 4.0 2.0 2.0 1.0 6.0 3.0 1.0 2.0 6.0 1.03.0

PTRW = 1 3 7 11 13 15 19

ELRW = 1 3 2 3 4 6 2 3 4 6 1 4 5 6 2 3 4 6

DISPLAY 3

ENVRW=8 ENVCL=8

ADJNCL = 4 3 6 1 2 6 2 3 6 2 3 5

XADJCL = 1 2 4 7 10 10 13

ADJNRW = 3 3 4 6 2 4 6 1 6 2 3 4

XADJRW = 1 2 5 8 9 10 13

DISPLAY 4

DIAG = 2.0 5.0 4.0 6.0 1.0 3.0

XENVRW= 1 1 1 2 5 5 9

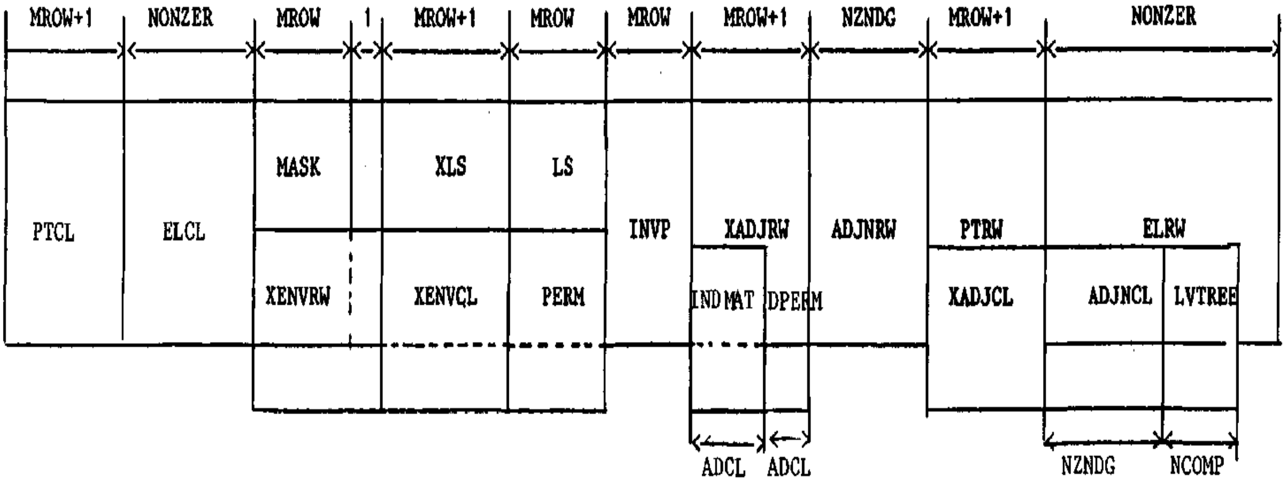
XENVCL = 1 1 1 3 5 5 9

EVRW = 3.0 3.0 0.0 0.0 5.0 2.0 3.0 0.0

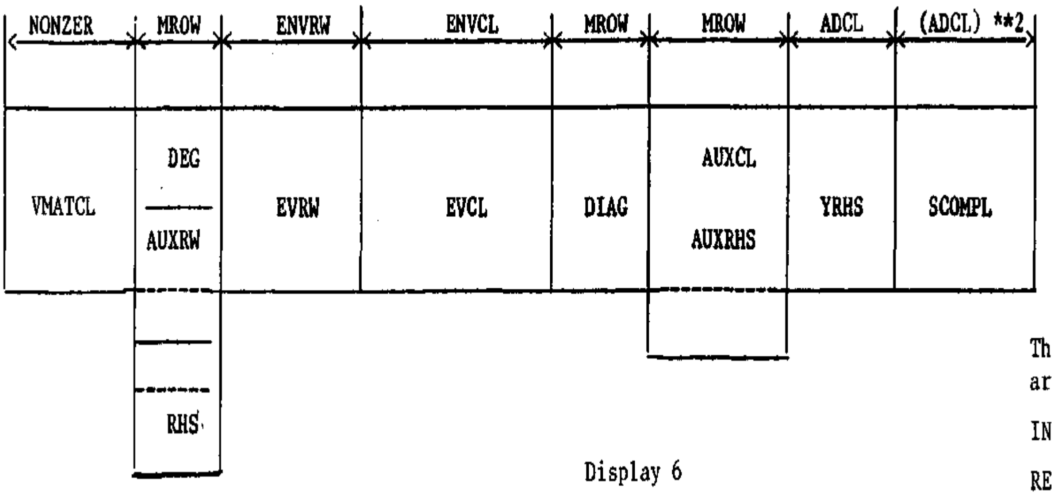
EVCL = 1.0 3.0 2.0 1.0 2.0 6.0 0.0 1.0

DISPLAY 5

INTEGER ARRAY AREA(ARRAY ISTOR)



REAL ARRAY AREA(ARRAY RSTOR)



Display 6

The size of the integer and real storage areas are given by the expressions

$$\text{INTSPA} = 3 * \text{NONZER} + 6 * \text{MROW} + 5$$

$$\text{RELSPA} = \begin{cases} \text{NONZER} + \text{ENVSZE} + \text{MROW} \text{ if } p = 0 \\ \text{NONZER} + \text{ENVSZE} + 3*\text{MROW} + \\ \text{ADCL} * (\text{ADCL}+1) \text{if } p > 0 \end{cases}$$

where $\text{ENVSZE} = \text{MROW} + \text{ENVRW} + \text{ENVCL}$

6. Test Data and Experimental Results

The investigation reported in this section was carried out with test matrices taken from a real life linear programming model. The model represents an oil company refinery planning operation and consists of 315 rows and 458 columns. In course of solving this problem by The FORTLP system [10] a set of seven basis matrices at the time of reinversion were written out to a data file. These basis matrices were then restructured to the Lower Block Triangular form with a nonsingular bump matrix having a zero free diagonal [1]. The present set of experiments were carried out for these bump matrices.

An IBM PC/AT working at 8 MHz and with an 80287 floating point processor was used for our experiments. The programs were compiled and linked using the Professional Fortran compiler and linker.

A number of important statistics were compiled: These are set out in Table 1. The columns of Table 1 are labelled by variables which are already defined in section 4. The test runs were carried out following three alternative strategies, namely,

Strategy 1: RCM ordering and XTOL = 0.1
Strategy 2: RCM ordering and XTOL = 0.001
Strategy 3: CM ordering and XTOL = 0.001

The main purpose of introducing the high tolerance value XTOL=0.1 was to force pivot rejection in the LU decomposition phase. In this way the use of Schur Complement update to deal with zeros in the leading pivot positions could be fully tested.

MATRIX	MROW	NONZER	STRATEGY	ENVSZE	ADCL	INTSPA	RELSPA	OPSF	GROWTH	ERROR
M1	24	60	1	108	1	329	218	121	1.23	1x10^{-5}
			2	108	0	329	192	75	1.23	1x10^{-5}
			3	103	0	329	187	101	0.78	1x10^{-5}
M2	49	174	1	512	3	821	796	3199	1.06	1x10^{-4}
			2	512	0	821	735	1249	6.71	2x10^{-4}
			3	788	0	821	1011	3744	23.42	8x10^{-4}
M3	61	235	1	633	8	1076	1062	12654	1.00	7x10^{-4}
			2	633	1	1076	992	1675	75.30	2x10^{-4}
			3	811	1	1076	1170	3543	67.70	1x10^{-4}
M4	92	326	1	1163	5	1535	1703	7972	6.77	2x10^{-5}
			2	1163	0	1535	1581	3708	19.15	1x10^{-4}
			3	1690	0	1535	2108	9522	24.09	4x10^{-5}
M5	117	400	1	1696	7	1907	2386	38133	30.39	3x10^{-4}
			2	1696	0	1907	2213	6178	25.55	6x10^{-4}
			3	2406	0	1907	2923	14216	12.32	9x10^{-4}
M6	130	461	1	2035	3	2168	2768	9836	26.74	3x10^{-5}
			2	2035	0	2168	2626	7283	18.21	4x10^{-4}
			3	2788	1	2168	3511	17621	57.36	1x10^{-3}
M7	141	504	1	2469	3	2363	3267	13095	20.55	2x10^{-5}
			2	2469	0	2363	3114	10542	13.99	3x10^{-4}
			3	3519	1	2363	4307	26510	44.07	1x10^{-3}

TABLE 1

6. Acknowledgements

Dr. M. Tamiz is supported by an SERC grant and Dr, J. Judice's visit to Brunel University was made possible by a fellowship also offered by the SERC. We are grateful to Dr. I. Duff and Dr. N. Gould of AERE, Harwell, who discussed with us some aspects of this research work.

REFERENCES

[1] K. DARBY-DOWMAN and G. MITRA, An investigation of algorithms used in restructuring of linear programming basis matrices prior to inversion, Studies of Graphs and Discrete Programming, Ed. P. Hanson, North Holland, (1981) 69-93.

[2] I.S. DUFF, Direct methods for solving sparse systems of linear equations, SIAM Journal of Scientific and Statistical Computing 5(1984) 605-619.

[3] I.S. DUFF, On algorithms for obtaining a maximum transversal, ACM Transactions on Mathematical Software 7 (1981) 315-330 and 387-390.

[4] G.E. FORSYTHE and C.B. MOLER, Computer solution of linear algebraic systems, Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[5] W.M. GENTLEMEN and A. GEORGE, Sparse matrix software, in "sparse matrix computations", edited by J.R. Bunch and D.J. Rose, Academic Press, New York, 1976, pp 243-261.

[6] A. GEORGE and J.W.H. LIU, Computer solution of large sparse positive definite systems, Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[7] F.G. GUSTAVSON, Two fast algorithms for sparse matrices: multiplication and permuted transposition, ACM Transactions on Mathematical Software 4(1978) 250-269.

[8] J.J. JUDICE, and G. MITRA, Extension of envelope method for the solution of unsymmetric systems, Technical Report, Department of Mathematics and Statistics, Brunel University (in preparation), 1987.

[9] J.J. JUDICE, G. MITRA and M. TAMIZ, Application of envelope method to LP reinversion, Technical Report, Department of Mathematics and Statistics, Brunel University (in preparation), 1987.

[10] G. MITRA, and M. TAMIZ, FORTLP user manual, Department of Mathematics and Statistics, Brunel University, 1985 (Revised 1987).

```
C
C PROGRAM TO PERFORM REINVERSIONS OF L.P. BASES BY USING THE ENVELOPE
C METHOD. THE RE ARE TWO PHASES, NAMELY
C                       1) SYMBOLIC PHASE
C                       2) FACTORIZATION PHASE
C  IN THE SYMBOLIC PHASE THE ROWS AND COLUMNS OF THE MATRIX ARE ORDERED
C  BY USING THE CUTHILL-MCKEE METHOD (RCM=0 ) OR THE REVERSE CUTHILL-MCKEE
C  METHOD (RCM=1).THEN THE ENVELOPE DATA STRUCTURE IS CONSTRUCTED FOR THE
C  FACTORIZATION PHASE.
C  IN THE FACTORIZATION PHASE THE LU DECOMPOSITION OF THE PERMUTED MATRIX
C IS OBTAINED BY USING THE BORDERED METHOD.
C
                  INTEGER*2 ISTCR (10000), NI, NO, MROW, NONZER, NZNDG, INTSPA, RELSPA,
         1             BANDRW, BANDCL, ENVRW, ENVCL, ENVSZE, RCM, NSINGL, NCOFP,
         2             ADCL, L, M, HOUR, MIN, SEC, HSEC, IDIAG, NZRENV
                  REAL RSTOR(10000),ERROR, MAXINP, MAXVAL, GROWTH, XTOL
                  REAL*8 OPS,CPSF
C
C       MEANINGS OF VARIABLES:
C             NI - INPUT CHANNEL
C             NO -OUTPUT CHANNEL
C             MROW - NUMBER OF ROWS AND COLUMNS OF THE MATRIX
C             NONZER - NUMBER OF NONZEROS OF THE MATRIX
C             NZNDG - NUMBER OF NONZEROS OFF DIAGONAL ELEMENTS
C             INTSPA - INTEGER STORAGE REQUIRED
C             RELSPA - REAL STORAGE REQUIRED
C             BANDRW - LOWER BANDWIDTH
C             BANDCL - UPPER BANDWIDTH
C             ENVRW - LOWER ENVELOPE SIZE
C             ENVCL - UPPER ENVELOPE SIZE
C             ENVSZE - ENVELOPE SIZE (=MROW+ENVRW+ENVCL)
C             NCOMP - NUMBER OF CONNECTED COMPONENTS OF MATRIX GRAPH =
C                        NUMBER OF NONSINGLETON DIAGONAL BLOCKS
C             NSINGL - NUMBER OF SINGLETONS ,
C             ADCL - NUMBER OF COLUMNS TO BE ADDED FOR FACTORIZATION TO BE
C                     POSSIBLE
C             ERROR - IT MEASURES THE ACCURACY OF THE DECOMPOSITION AND IS
C                    EQUAL TO MAX (ABS(X(I)-1.)),WHERE X(I) ARE COMPONENTS
C                    OF THE COMPUTED SOLUTION OF SYSTEM LU*X=B WHERE B=A*1
C                          WITH 1 A VECTOR OF ONES
C             OPSF - NUMBER OF OPERATIONS(MULTIPLICATI0NS + DIVISIONS) IN
C                    FACTORIZATION
C             OPS - TOTAL NUMBER OF OPERATIONS OF FACTOR AND VERIFY
C             MAXINP -  MAXIMUM ABSOLUTE VALUE OF ORIGINAL MATRIX ELEMENTS
C             MAXVAL - MAXIMUM ABSOLUTE VALUE OF L AND U MATRICES ELEMENTS
C             GROWTH - GROWTH FACTOR = MAXVAL / MAXINP
C             NZRENV  - NUMBER OF NONZEROS INSIDE ENVELOPE
C             XTOL  -  TOLERANCE FOR ZERO
C
C         POINTERS ...
C
                      INTEGER*2  PTVMCL, PTVMRW, PTPTCL, PTELCL, PTPTRW, PTELRW, PTXVRW,
           1                 PTDEG,  PTMASK, PTXVCL, PTPERM, PTINVP, PTXARW, PTADRW,
           2                 PTXACL, PTADCL, PTDIAG, PTEVRW, PTEVCL, PTXLS, PTNCP,
           3                 PTRHS,PTAURW,PTAUCL,PTYRHS,PTSCPL,PTAUX,PTAUY,
           4                 PTIMAT, PTDPER

C                     COMMON /ISNI/ NI
                      COMMON /ISNO/ NO
                      CCMMON /ISROM/ ROM
                      COMMON /RSOPS/ OPS
                      COMMON /RSXTOL/ XTCL
                      COM'MON /RSOPSF/ CPSF
C
                      NI = 11
                      NO = 12
                      NE = 13
                      OPEN (NE, FILE='INPT')
                      OPEN (NI, FILE='INP')
                      OPEN (NO, FILE = 'OUT ')
C
                      REWIND (NI)
                      REWIND (NE)
                      REWIND (NO)
                      READ (NE, 57) RCM, XTOL
     50               FORMAT (15,F10.5)
    100               READ (NI, 200) MROW
    200               FORMAT (I 5)
                      IF (MROW.E, Q) STOP
C
```

```
C   INPUT THE MATRIX COLUMNWISE
C
        PTVMCL=1
        PTPTCL=1
        PTELCL=PTPTCL +MROW+1
        CALL INPUT (MROW, NONZER, ISTOR (PTPTCL), ISTOR (PTELCL),
     1             RSTOR (PTVMCL) ,MAX INP)
        NZNDG  = NONZER-MROW
C
C   INITIALIZE POINTERS FOR OTHER SUBROUTINES
C
        PTDEG = PTVMCL + NONZER
        PTRHS=PTDEG
        PTXVRW=PTELCL+NONZER
        PTMASK=PTXVRW
        PTXLS=PTMASK +MROW
       PTXVCL=PTXLS+1
        PTPERM=PTXVCL+MROW+1
        PT INVP = PTPERM+MROW
        PTXARW=PTINVP+MROW
        PTADRW = PTXARW+MROW+1
        PTXACL=PTADRW+NZNDG
        PTADCL = PTXACL+MROW+1
        PTPTRW=PTXACL
        PTELRW=PTADCL
        PTNCP=PTADCL+NZNDG
        PTIFAT = PTXARW
C
C   DETERMINE THE ROWISE R EPRE SENTATION OF THE MATRIX
C
        CALL ROWISE (MROW, NONZER, ISTOR (PTPTCL), ISTOR (PTELCL),
     1             ISTCR (PTPTRW), ISTOR (PTELRW))
C
C   FIND THE OUTERGRAPH OF THE MATRIX
        CALL FDOUGR (MROW, ISTOR (PTPTRW), ISTOR (PTELRW),
     1             ISTOR (PTXARW), ISTOR (PTADRW))
C
C   FIND THE INNERGRAPH OF THE MATRIX
C
        CALL FDINGR (MROW, ISTOR (PTPTCL), ISTOR (PTELCL),
     1             ISTOR (FTXACL), ISTOR (PTADCL))
C
C   SWITCH ON TIME
C
        CALL GETTIM (HOUR, MIN, SEC, HSEC)
        WRITE (NO, 300) HCLR, MN, SEC, HSEC
  300   FORMAT (TX, I2,' : ',I2,' : ', I2)
C
C   DETERMINE ORDERING FOR THE MATRIX
C
        CALL  GENRCM  (MROW,  ISTOR   (PTPERM),   ISTOR  (PTXARW),  ISTOR   (PTADRW)
     1             ISTOR (PTXACL), ISTOR (PTADCL), RSTOR (PTDEG),
     2             I S TOR (PTMAS K), I STOR (P TX L S), NCOMP, ISTOR (PTNCP),
     3             NSINGL)
C
C   DETERMINE INVERSE OF PERMUTATION
C
        CALL INVRSE (MROW, ISTOR (PTPERM), ISTOR (PTINVP))
C
C DETERMINE THE ENVELOPE STRUCTURE OF THE LOWER PART OF THE MATRIX
C
        CALL  FENVRW(MROW,ISTOR(PTXARW),ISTOR(PTADRW),ISTOR(PTPERM),
     1                     ISTOR(PTINVP), ISTOR (PTXVRW), ENVRW, BANDRW)
C
C DETERMINE THE ENVELOPE STRLCTURE OF THE UPPER PART OF THE MATRIX
C
        CALL FENVCL(MROW,ISTOR(PTXACL),ISTOR(PTADCL),ISTOR(PTPERM),
     1          ISTOR(PTINVP),ISTOR(PTXVCL),ENVOL,BANDCL)
C
C DETERMINE THE NUMBER OF ELEMENTS STORED BY THE ENVELOPE METHOD
C AND TOTAL STORAGE FOR INTEGER AND REAL ARRAYS
C
        ENVSZE=MROW+ENVRL+ENVCL
        RELSPA = NONZER+ENVSZE+MROW
        INTSPA=PTELRW+NCNZER-1

C DETERMINE THE ENVELOPE REPREESNTATION OF THE  MATRIX

         PTEVRW=PTDEC+MROW
```

```
      PTEVCL=PTEVRW+ENVRW
      PTDIAG=PTEVOL+ENVOL
      PTYRHS=PTDIAG+MROW
      CALL ENVMAT(MROW, ISTOR (PTPTCL), ISTOR (PTELCL), RSTOR (PTVMCL),
     1            ISTOR (PTINVP),ENVRW,ENVCL,ISTOR (PTXVRW),
     2            ISTOR(PTXVCL), RSTOR(PTEVRW), RSTOR(PTEVCL),
     3            RSTOR (PTDIAG))
C
C  FACTORIZE MATRIX INTO L*U
C
       OPSF=0.D0
       OPS=0.D0
       CALL FACTOR (MROW, ISTOR (PRTXVRW),RSTOR (PTEVRW), ISTOR (PTXVCL),
     1            RSTOR (PTEVCL),RSTOR (PTDIAG), NSINGL, ADCL,
     2            ISTOR (PTIMAT), MAXVAL, NZRENV)

C
       IF (ADCL.EQ.O) GO TO 400
C
C  CALCULATE SCHUR COMPLEMENT MATRIX
C
         RELSPA=RELSPA+MROW+ADCL*(ADCL+1)
         PTAURW=PTDEG
         PTAUCL=PTYRHS
         PTAUX=PTAUCL
         PTYRHS=PTAUCL+MROW
         PTSCPL=PTYRHS+ADCL
         PTDPER=PTIMAT+ADCL
         CALL SCHCCM (MROW, ISTOR (PTXVRW), ISTOR (PTXVCL), RSTOR (PTEVRW),
     1            RSTOR (PTEVCL), RSTOR (PTDIAG), ADCL, ISTOR (PTIMAT),
     2            RSTOR (PTAUCL), RSTOR (PTAURW), RSTOR (PTSCPL), MAXVAL,
     3            ISTOR (PTDPER))
C
C  CALCULATE GROWTH FACTOR AND SWITCH OFF TIME
C
 400  GROWTH = MAXVAL/MAXINP
     CALL GETTITM (HOUR, MIN, SEC, HSEC)
      WRITE (N0, 300) HOUR, MIN, SEC, HSEC
      OPSP=OPS
C
C  VERIFY ACCURACY OF DECOMPOSITION
C
      CALL INIVER (MROW, ISTOR (PTPTCL), ISTOR (PTELCL), RSTOR (PTVMCL),
     1            ISTOR (FTINVP),RSTOR (PTRHS))
C
      IDIAG=1
      CALL LOWSOL (MROW, ISTOR (PTXVRW), RSTOR (PTEVRW), RSTOR (PTDIAG)
     1            RSTOR  (PTRHS), IDIAG)
C
      IF (ADCL.EQ.O)GO TO 500
      CALL    GETRHS (MROW, ISTOR (PTXVRW), ISTOR (PTXVCL), RSTOR (PTEVRW),
     1            RSTOR (PTEVCL), RSTOR (PTDIAG), RSTOP (PTRHS),
     2            RSTOR (PTAUX), RSTOR (PTYRHS), ISTCR (FTIMAT),
     3            RSTOR (PTSCPL), ISTOR (PTDPER), ADCL)
C
 500  CALL UPSOL (MROW, ISTOR (PTXVCL), RSTOR (PTEVCL),RSTOR (PTDIAG),
     1            RSTOR (PTRHS))
C
      CALL GETERR (MROW,RSTOR(PTRHS),RSTOR(PTYRHS),ADCL,ERROR)
C
C  OUTPUT AND FINISH
C
      CALL OUTPUT (MROW, NONZER, ENVSZE, BANDRW, BANDCL, INTSPA,
     1      RELSPA, NCCMP, ISTCR (PTNCP), NSINGL, ACCL, ERRCR,
     2      GROWTH, NZRENV)
      GO TO 100
      END
```

```
C-----------------------------------------------------------------------------------------------------
C
            SUBROUTINE INIVER (NEQNS, PTCL, ELCL,VMATCL, INVP, RHS)
C-----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE CALCULATES THE VECTOR RHS=A*1,WHERE 1 IS A VECTOR OF ONES
C
C   MEANING OF VARIABLE :
C       RHS(NEGNS) - THE DESIRED VECTOR
C
          INTEGER*2 PTCL (1),ELCL(1),INVP(1),NEQNS, I, ISUB, JSTRT,JSTOP
      REAL VMATCL (1), RHS(1)
C
            DO 100  I=1, NEGNS
               RHS (I)=0.
  100       CONTINUE
C
            DO 300 I=1, NEGNS
               JSTRT=PTCL (I)
               JSTOP=PTCL (I+1)-1
               DO 200 J=JSTRT, JSTOP,
                  ISUB=ELCL (J)
                  ISUB=INVP (ISUB)
                  RHS (ISUB)=RHS (ISUB)+VMATCL (J)
   200         CONTINUE
   300      CONTINUE
            RETURN
            END
C
C
C-----------------------------------------------------------------------------------------------------
C
            SUBROUTINE GETRHS (NEQNS, XENVRW,XENVCL,EVCW,EVCL,DIAG,RHS,
      1           AUXRHS, YRHS, INDMAT, SCOMPL, DPERM, ADCL)
C
C-----------------------------------------------------------------------------------------------------
C
C   THIS ROUTINE GETS THE RHS TO SOLVE THE SYSTEM U*X=RHS WHEN AT LEAST
C   ACOLUMN HAD TO BE ADDED TO GET THE FACTORIZATION
C
C       MEANINGS OF VARIABLES:
C          YRHS (ADCL) - VECTOR OF THE VARIABLES COPRESPONDING TO ADDED
C                      COLUMNS
C          AUXRHS (NEQNS) - AUXILIAR VECTOR
C
            INTEGER*2 XENVRW (1), XENVCL (1), INDMAT (1), DPERM (1), NEQNS, ADCL,
      1           IDIAG, NEG, I, IFIRST,IPERM
            REAL EVRW(1) EVCL(1),RHS(1),AUXRHS(1),YRHS(1),DIAG(1),
      1     SCOMPL (ADCL,ADCL)
C
C  SOLVE   U*X=AUXRHS
C
            DO 100 I=1, NEQNS
               AUXRHS (I)= RHS(I)
 100        CONTINUE
            CALL UPSCL (NEQNS, XENVCL, EVCL, DIAG, AUXRHS)
C
C  CALCULATE AUYRHS=F*AUXRHS, WHERE F IS THE MATRIX OF THE ADDED ROWS
C
            DO 200 I = 1, ADCL
                L= INDMAT (I)
                YRHS (I)= AUXRHS (L)
  200       CONTINUE
C
C  SOLVE SCOMPL*Y=YRHS
C
            DO 300 I=1,  ADCL
               DPERM (I)=1
 300        CONTINUE
            CALL SOLVE (ADCL, DPERM, YRHS, SCOMPL)
C
C CALCULATE AUXRHS=E*YRHS, WHERE E IS THE MATRIX OF THE ADDED COLUMNS
C
  225       DO 600 1=1, NEQNS
               AUXRHS (I)=0.

  600       CONTINUE
            IFIRST =0
            DO 700 I=1, ADCL
              L= INDMAT (I)
```

```
            IFERM=DPERM (I)
            AUXRHS (L)=-YRHS (IPERM)
          IF (IFIRST.EQ.O) IF IRST = L
  700     CONTINUE
C
C  SOLVE  L*X=AUXRHS
C
          IDIAG=1
          NEQ=NEQNS-IFIRST + 1
          CALL   LOWSOL (NEQ, XENVRW (IFIRST), EVRW, DIAG (IFIRST),
     1               AUXRHS (IFIRST), IDIAG)
C
C  CALCULATE    RHS
C
          DO 800 I =1, NEQNS
             RHS (I) = RHS (I)-AUXRHS (I)
  800     CONTINUE
          RETURN
          END
C
C
C----------------------------------------------------------------------------------------------------
C
          SUBROUTINE GETERR (NEQNS, RHS, YRHS, ADCL, ERROR)
C
C----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE CALCULATES THE ERROR OF THE COMPUTED SOLUTION
C
          INTEGER*2 NEQNS, I, ADCL
          REAL RHS (1), YRHS (1 ), ERROR, S
C
          ERROR = C.
          DO 100 I=1, NEQNS
             S=RHS (I)-1.
             S=ABS (S)
             IF (S.GT.ERROR) ERROR = S
  100     CONTINUE
C
          IF (ADCL.EQ.O) RETURN
          DO 200 I=1, ADCL
                S = YRHS (I)-1.
                S=ABS (S)
                IF (S.GT.ERROR) ERROR=S
  200       CONTINUE
          RETURN
          END
```

```
C------------------------------------------------------------------------------------------------------
C
          SUBROUTINE UPSOL (NEQNS, XENVCL, EVCL, DIAG, RHS)
C
C------------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE SOLVES AN UPPER TRIANGULAR SYSTEM U*X=RHS, WHERE U IS
C  STORED IN ENVELOPE FORMAT REPRESENTATION
C
                INTEGER*2 XENVCL (1 ),NEQNS, I, IBAND, JSTRT, JSTOP,J, L
                REAL EVCL (1 ), DIAG (1), RHS (1),S
                REAL* 3 COUNT, CPS
C
                COMMON /RSOPS/ OPS
C
                I = NEQNS + 1
 100            I=I-1
                IF (I .E Q. 0) RETERN
                     IF (RHS (I) .E Q.O.) GO TO 100
                          S = RHS (I) /DIAG (I)
                          RHS (I) = S
                          OPS = OPS+1 .D C
                          IBAND=XCNVCL (I+1)-XENV0L (I)
                          IF (IBAND. EQ. 0) GO TO 100
                              IF (IBAND. GE.I) IBAND=I-1
                              L = XENVCL (I + 1)- IBAND
                              JSTRT=I-IBAND
                              JSTOP=I-1
                              DO 200 J=JSTRT, JSTOP
                                    RHS (J ) = RHS (J)-S*EVOL (L)
                                    L = L+1
        200               CONTINUE
                          COUNTINUE=IBAND
                          OPS=OPS+COUNT
                          GO TO 100
                END
```

```
C-----------------------------------------------------------------------------------------------
C
                SUBROUTINE OUTPUT (NEQNS, NONZER, ENVSZE, BANDRW, BANDOL, lNTSPA,
        1                          RELSPA, NCOMP, LVTREE, NSINGL, ADCL, ERROR,
        2                          GROWTH, NZRENV)
C
C-----------------------------------------------------------------------------------------------
C
C  THIS ROUTINE PROVIDES THE OVERALL RESULTS
C
                INTEGER*2 LVTREE (1), NEQNS, NONZER, ENVSZE, BANDRW, BANDOL, BANDW,
        1               INTSPA, RELSPA, NO, ADCL, RCM, NCOMP, NSINGL, NZRENV

                REAL*8 OPS, OPSF
                REAL ERROR, GROWTH, XTOL
C
                COMMON /RSOPS/ OPS
                COMMON /RSOPSF/ OPSF
                COMMON /ISRCM/ RCM
                COMMON /ISNC/ NO
                COMMON /RSXTOL/ XTCL
C
                WRITE (NO, 100) NEQNS
  100           FORMAT (1X,'MATRIX ORDER ',16)
C
                WRITE (NO, 200) RCM
  200           FORMAT (1X,'REVERSE CUTHILL-MCKEE =',I2)
C
                WRITE (NO,300) XTOL
  300           FORMAT (1X,' TOLERANCE FOR ZERO ',F10.5)
C
                WRITE (NO,400) NCOMP
  400       FORMAT (1X,'NUMBER OF DIAGONAL BLOCKS ',14)
                WRITE (NO, 500)
  500           FORMAT (1X,'LEVEL TREES LENGTHS: ')
                WRITE (NO, 600) (LVTREE(I), I=1, NCOMP)
  600           FORMAT (2014)
                WTITE (NO, 700) NSINGL
  700           FORMAT (1X,'NUMBER OF SINGLETONS',14 )
C
                WRITE (NO,800) NONZER
  800           FORRMAT (1X,'NUMBER OF NONZEROS IN ORIGINAL MATRIX ',I6)
C
                WRITE (NO,900) ENVSZE
    900         FORMAT( 1X,'NUMBER OF STORED ELEMENTS IN ENVELOPE METHOD ',16)
                NZRENV=NZRENV + NEQNS
                WRITE (NO,950) NZRENV
   950          FORMAT(1X,'NUMBER OF NONZEROS INSIDE ENVELOPE ',16)
C
                WRITE (NO,1000) BANDRW, BANDCL
  1000          FORMAT (1X,'LOWER BANDWIDTH ',I6,' UPPER BANDWIDTH ',I6)
C
                WRITE (NO,1200) INTSPA
  1200          FORMAT (1X,'NUMBER OF STORED ELEMENTS OF INTEGER ARRAYS ',I6)
C
               WRITE (NO/1300) RELSPA
  1300          FORMAT (1X,'NUMBER OF STORED ELEMENTS OF REAL ARRAYS ',I6)
C
              WRITE (NO,1400) OPSF
  1400        FORMAT (1X, 'NUMBER OF OPERATIONS IN FACTOR , D20.10)
                WRITE (NO,1450) OPS
  1450        FORMAT (1 X, ' TOTAL NUMBER OF OPERATIONS ',D20. 10)
C
                WRITE (NO, 1500) GROWTH
  1500          FORMAT (1X, 'GROWTH FACTOR ',F15.5)
C
                WRITE (NO, 1600) ERROR
  1600          FORMAT (1X,' ERROR OF COMPUTED SOLUTION ',F15.12)
C
                IF (ADCL.GT.0) WRITE (NO, 1700) ADCL
  1700          FORMAT ( 1X, I4, ' COLUMNS TO ADD TO GET FACTORIZATION')
                RETURN
                END
C
C
C--------------------------------------------------------------------------------------------
C
                SUBROUTINE DGNINT (ARRAY,NDIM, ITOP, IBOT, A8)
C
C--------------------------------------------------------------------------------------------
C
C  THIS ROUTINE PRINTS THE INTEGER CONTFWTS OF AN INTEGER ARRAY
```

```
C
C      MEANIGS OF VARIABLES:
C         ARRAY - ARRAY TO BE PRINTED
C         NDIM - NUMBER .OF ITEMS TO BE PRINTED
C         ITOP, IBOT - FIRST AND LAST ELEMENTS OF ARRAY TO BE PRINTED
C         A8 - NAME OF APRAY CONTAINING AT MOST 6 LETTERS
C
           INTEGER*2 ARRAY(1),NO,NDIM,ITOP,IBOT,I
           CHARACTER*8 A8
C
C          COMMON/ ISNO/ NO
C
           WRITE (NO,100) A8, ITOP, IBOT
  100     FORMAT (1X,'ELEMENTS OF ',A8,'ARRAY FROM ',16,' TO ',16 )
C
           WRITE(NO,200)(ARRAY(I),I=ITOP,IBOT)
  200      FORMAT (2014)
C
           WRITE(.NO,300)A8,NDIM
  300      FORMAT (IX,'DIMENSION OF', A8,'ARRAY : ',16)
C
           RETURN
           END
C
C-----------------------------------------------------------------------------------------------
C
           SUBROUTINE DGNRELCARRAY,NDIM,ITOP,IBOT,A8)
C
C-----------------------------------------------------------------------------------------------
C
C  THIS ROUTINE PRINTS THE REAL CONTENTS OF A REAL ARRAY
C
           INTEGER*2 NO, NDIM, ITOP, IBOT, I
           REAL ARRAY (1)
           CHARACTER*8 A8
C
           COMMON/ISNO/ NO
C
           WRITE (,NO,100)A8, ITOP, IBOT
  100      FORMAT (1X,' ELEMENTS OF ', A8, 'ARRAY FROM', 16,' TO ',16)
C
           WRITE (NO,200)(ARRAY(I),I=ITOP, IBOT)
  200      FORMAT (8F10.5)
C
           WRITE (NO, 300) A8, NDIM
  300      FORMAT (1X,'DIMENSION OF,A8,'ARRAY: ',16)
C
           RETURN
           END
```

```
C-----------------------------------------------------------------------------------------------------
C
            SUBROUTINE LOWSOL (NEQNS, XENV, ENV, DIAG, RHS, IDIAG)
C
C-----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE SOLVES A LOWER TRIANGULAR SYSTEM L*X= RHS, WHERE L IS
C  STORED IN ENVELOPE FORMAT REPRESENTATION. IT IS ASSUMED THAT THE
C   FIRST RHS ELEMENT IS NONZERO
C
C        MEANINGS OF VARIABLES:
C             NEQNS - NUMBER OF SYSTEM EQUATIONS
C             XENV, ENV, DIAG - ARRAYS OF ENVELOP MATRIX REPRESENTATION
C             RHS - SYSTEM RIGHT-HAND SIDE VECTOR.IN FACTORIZATION IT IS
C                   A ROW OR COLUMN OF THE MATRIX TO BE FACTORIZED
C             IDIAG - INTEGER VARIABLE WHICH TAKES VALUE 1 IF ALL DIAGONAL
C                   ELEMENTS OF THE LOWER TRIANGULAR MATRIX ARE E QUAL TO
C                   ONE AND ZERO OTHERWISE
C
            INTEGER*2 XENV (1), NEQNS, IDIAG, IFIRST, LAST, IBAID, JSTRT,
      1               JSTOP, I, J, L
            REAL ENV(1)DIAG(1),RHS(1),S
            REAL*3 OPS, COUNT
C
            COMMON /RSOPS/ CFS
C
            IFIRST=1
            LAST=0
C
C  LAST CONTAINS THE POSITION OF THE MOST RECENTLY COMPUTED NONZERO
C  COMPONENT OF THE SOLUTION
C
            DO 300 I = IFIRST,MEQNS
               IBAND=XENV (I+1)-XENV (I)
               IF (IBAND.GE.I) IBAND=I-1
               S=RHS (I)
               L = I – IBAND
               RHS (I)=0.
C
C   IF ENVELOPE ROW IS EMPTY OR CORRESPONDING COMPONENTS OF SOLUTION
C  ARE ALL ZEROS THEN ONLY DIVISION BY DIAGONAL ELEMENT IS DONE
C
            IF (BAND. EQ.O OR. LAST.LT.L) GO TO 200
                  JSTRT = XENV (1+1)-IBAND
                  JSTOP=XENV (I+1)-1
                  DO 100 J=JSTRT, JSTOP
                     S=S-ENV (J)*RHS (L)
                     L = L+1
  100             CONTINUE
                  COUNT= IBAND
                  OPS=OPS+COUNT
C
  200       IF (S. EQ .C.)GO TO 300
                  LAST=I
                  RHS (I)=S
                  IF (IDIAG. EQ.1) GO TO 300
                     RHS (I)=S /DIAG(I)
                     OPS=OPS+1.DO
  300       CONTINUE
            RETURN
            END
```

```
C-----------------------------------------------------------------------------------------------
C
        SUBROUTINE DECOMP (N, DPERM,A, MAXVAL)
C
C--------------------------------------------------------------------------------------------
C THIS ROUTINE FINDS THE LU DECOMPOSITION OF A DENSE MATRIX A BY USING
C  PARTIAL PIVOTING.THE LU DECOMPOSITION OVERWRITES THE MATRIX A.
C
C       MEANINGS OF VARIABLES:
C            N - ORDER OF THE MATRIX
C            A (N, N) - MATRIX TO BE FACTORIZED
C            DPERM (N) - INTEGER VECTOR WHICH GIVES THE ROW PERMUTATION
C            TOL - TOLERANCE FOR ZERO FIVOT
C
               INTEGER* 2 DPERM(1) ,N,-NM1,K,KPERM,KP1 ,I,IPERM,J ,IND, ITEHP,NO
               REAL A(N,N),VAL,MAXVAL,PIVOL,TOL,,S,MAX
               REAL*8 OPS,COUNT
C
               COMMON /RSOPS/ OPS
               COMMON /ISNO/ NO
               TOL=1.E-4
C
               IF (N.GT.1) GO TO 100
               VAL = ABS (A(1,1 ) )
               IF (VAL.LT.TOL) GO TO 700
               RETURN
  100          NM1 = N-1
               DO 600 K=1, NM1

C PARTIAL PIVOTING IN OPERATION...
C
C
                    KPERM = DPERM (K)
                    PIVOT=A (KPEM, K)
                    MAX=ABS (PIVOT)
                    IND = K
                    KP1= K + 1
                    DO 200 I=KP1,N
                         IPERM = DPERM (I)
                         VAL=ABS (A(IPERM,K))
                      IF (VAL, LE, MAX) GO TO 200
                           MAX = VAL
                           IND = I
 200                CONTINUE
                    IF (IND.EQ.K) GO TO 300
                      ITEMP=DPERM (IND)
                      DPERM (IND) = DPERM (K)
                      DPERM (K) = ITEMP
                      KPERM= DPERM (K)
                      PIVCT=A (KPERM, K)
C
C EFECTUE DECOMPOSITION STEP...
C
  300              IF(MAX.LT.TOL)GO TO 700
                   DO 500 I=KP1,N
                        IPERM=DPERM (I)
                        VAL = A(IPERM,K)/PIVOT
                        A (IPERM, J)=VAL
                        DO 400 J=KP1, N
                              S=A (IPERM,J) - VAL*A(KPERM, J)
                          A (IPERM, J)=S
                            S=ABS (S)
                            IF (S. GT. MAXVAL) MAXVAL=S
 400                    CONTINUE
                        COUNT=N-K+1
                        OPS=OPS+COUNT
 500               CONTINUE
 600           CONTINUE
               RETURN
C
C  MATRIX IS SINGULAR
C
 700           WRITE (NO, 300)
 800           FORMAT (1X, MATRIX IS NONSINGULAR')
               RETURN
               END
C
C
C-----------------------------------------------------------------------------------------------
               SUBROUTINE SOLVE (N, DRERM, A, B)
```

```
C
C--------------------------------------------------------------------------------------------------------------------
C
C     MEANINGS OF VARIABLES:
C           N - ORDER OF SYSTEM
C           A (N, N) - LU DECOMPOSITION OF MATRIX A
C           DPERM (N) - INTEGER VECTOR THAT GIVES THE ROW PERMUTATION
C            E (N) - R.H.S. VECTOR
C
                 INTEGER*2 DPERM (1),N, NM1, NPERM, K, KM1, KPERM, J, JPERM
                 REAL A (N, N),B(1 ), SUM
                 REAL*8 COUNT, OPS
C
                 COMMON /RSOPS/ OPS
C
                 IF (N. EQ. 1) GO  TO 300
C
C     SOLVE THE SYSTEM L*Y=B
C
                 DO 200 K=2, N
                      KPERM=DPERM (J)
                      KM1=K-1
                      SUM=B (KPERM)
                      DO 100 J=1, KM1
                           JPERM= DPERM (J)
                           SUM = SUM-A (KPERM, J )* B (JPERM)
   100                CONTINUE
                      B (KPERM)= SUM
                      COUNT=KM1
                      OPS=OPS+COUNT
    200          CONTINUE
C
C    SOLVE THE SYSTEM U*X=Y
C
   300           NPERM=DPFRM (N)
                 B (NPERM)= B(NPERM)/A(NPERM,N)
                 OPS = OPS + 1.DO
                 IF (N.EQ.1) RETURN
                 NM1=N-1
                 DO   500 K=NM1, 1,-1
                        KPERM=DPERM (K)
                        KP1=K+1
                        SUM=B (KPERM)
                        DO 400 J=KP1, N
                             JPERM=DPERM (J)
                             SUM = SUM-A(KPERM, J)* B ( JPERM)
   400                  CONTINUE
                        B (KPERM)=SUM/A(KFERM,K)
                        COUNT=N-K+1
                        OPS=OPS+COUNT
  500            CONTINUE
                 RETURN
                 END
```

```
C------------------------------------------------------------------------------------------
C
            SUBROUTINE SCHCOM (NEQNS, XENVRW, XENVCL, EVRW, EVCL, DIAG, ADCL,
     1                 INDMAT, AUXCL, AUXRW, SCOMPL, MAXVAL, DPERM)
C
C------------------------------------------------------------------------------------------
C
C  THIS ROUTINE CALCULATES THE SCHUR COMPLEMENT MATRIX FOR THE CASE
C   IN WHICH AT LEAST A COLUMN HAS TO BE ADDED TO GET THE FACTORIZATION
C
C     MEANINGS OF VARIABLES:
C         NEQCL - NUMBER OF ELEMENTS OF COLUMN ADDED WHICH ARE NECESSARY
C                   TO CALCULATE A COLUMN OF SCOMPL MATRIX (NEQCL <= NEQNS)
C         NEQRW - NUMBER OF ELEMENTS OF ROW ADDED WHICH ARE NECESSARY TO
C                    CALCULATE A ROW OF SOOMPL MATRIX (NEQRW <= NEQNS)
C         SCOMPL (ADCL, ADCL) - SCHUR COMPLEMENT MATRIX
C         AUXCL (NEQCL) - AUXILIAR VECTOR FOR ADDED COLUMNS
C         AUXRW (NEQRW) - AUXILIAR VECTOR FOR ADDED ROWS

C
                INTEGER*2 XNVRW(1), XENVCL (1),INDMAT(1),DPFRM(1),NEQNS,I,J,L,K,
     1                    JFIRST, IFIRST, IDIAG, NEQRW, NEQCL, ADCL
                REAL SCOMPL (ADCL, ADCL), EVRW(1), EVCL(1), DIG(1), AUXCL(1),
     1                    AUXRW (1), S, MAXVAL
                REAL*8 COUNT, OPS
C
                COMMON /RSOPS/ OPS
C
C  INITIALISE SCOMPL
C
          DO   200 I=1, ADCL
          DO   100 J=1, ADCL
               SCOMPL (I, J)=0.
  100          CONTINUE
  200     CONTINUE
C
C  CALCULATE SCOMPL MATRIX COLUMN BY COLUMN
C
               DO 1000   J=1, ADCL
C
C  COLUMN J . . .
C
                JFIRST=INDMAT (J)
                NEQCL=NEQNS-JFIRST+1
                AUXCL (1) =-1.
                DO 300 K = 2, NEQCL
                     AUXCL (K)=0.
  300           CONTINUE
                IDIAG=1
                CALL LOWSOL (NEQCL, XENVRW (JFIRST), FVRW, DIAG (JFIRST),
     1                    AUXCL, IDIAG)
                DO 900 I=1, ADCLC
C
C  ROW  I …
C
                   IFIRST=INDMAT (I)
                   NEQRW=NEQNS-IFIRST+1
                   AUXRW (1)=-1.
                   DO 400 K=2, NEQRW
                        AUXRW (K)=0.
  400              CONTINUE
                IDIAG=0
                   CALL LOWSOL (NEQRW, XENVCL (IFIRST), EVCL, DIAG (IFIRST),
     1                    AUXRW, IDIAG)
C
C  CALCULATE ELEMENT IN (I, J) POSITION
C
                   S=0.
                   IF (IFIRST.GE.JFIRST) GO TO 600
                        L= JFIRST- IFIRST+1
                        DO 500 K=1, NEQCL
                           S=S+ALXCL (K) * AUXRW (L)
                           L=L + 1
  500                   CONTINUE
                        COUNT = NEQCL
                           OPS=OPS+COUNT
                           GO  TO  800
C
  600             L= IFIRST-JFIRST+1
                  DO 700 K =1,   NEQRW
                       S=S+ ALXCL (L) * AUXRW (K)
```

```
                    L=L+1
 700            COUNTINUE
                COUNT=NEQRW
                OPS=OPS+COUNT
C
 800            IF (I. EQ. J) S=S-1.
                SCOMPL (I, J)=-S
                S=ABS (S)
                IF (S.GT.MAXVAL) MAXVAL=S
 900          CONTINUE
1000        CONTINUE
C
C  FIND LU DECOMPOSITION OF SCHUR COMPLEMENT MATRIX USING PRATIAL
C   PIVCTING
C
C
C           CALL DECOMP (ADCL, DPERM, SCOMPL, MAXVAL)
            RETURN
            END
```

```
C-----------------------------------------------------------------------------------------------
C
                    SUBROUTINE INVRSE (NEQNS, PER, INVP)
C
C-----------------------------------------------------------------------------------------------
C
C   THIS ROUTINE FINDS THE INCERSE PERMUTATION OF PREM
C       MEANING OF VARIABLE:
C         INVP (NEQNS) - ARRAY OF THE IN VERSE PERMUTATION OF PERM
C
              INTEGER*2 PERM (1), INVP (1),I, IPERM, NEQNS
C
              DO 100 I = 1, NEQNS
                  IPERI=PE RM ( I )
                  INVP (1PERM) = I
  100         CONTINUE
              RETURN
              END
```

```
C-----------------------------------------------------------------------
C
      SUBROUTINE INPUT (NEQNS,NONZER,PTCL,ELCL,VMATCL, MAXINP)
C
C-----------------------------------------------------------------------
C
C THIS ROUTINE READS THE NONZERO ELEMENTS OF THE MATRIX AND GENERATES
C THE COLUMNWISE REPRESENTATION OF THE MATRIX
C
C     MEANINGS OF VARIABLES:
C        NEQNS - NUMBER OF ROWS AND COLUMNS OF THE MATRIX
C        PTCL (NEQNS+1) - ARRAY OF POINTERS OF THE DATA STRUCTURE
C        ELCL (NONZER) - ARRAY OF ROW INDICES OF NONZERO ELEMENTS
C        VMATCL (NONZER) -ARRAY OF THE VALUES OF THE NONZERO ELEMENTS
C                          WHOSE INDICES ARE ELCL (NCNZER)
C
         INTEGER*2 PTCL (1),ELCL (1),NEQNS,NONZER,NI
         INTEGER*2 NODE, ISUB, JSUB, K, M
         REAL VMATCL (1),VALUE,MAXINF
C
         COMMON /ISNI/ NI
C
         MAXINP=0.
         NONZER=0
         NODE=0
 100     READ (NI, 150) JSUB,I SUB,VALUE
 150     FORMAT (2I5, F10.5)
C
C  GET ELCL AND VMATCL ARRAYS
C
         IF (JSUB.EQ.O)GO TO 300
            NONZER=NONZER+1
            ELCL(NONZER)=ISUE
            VMATCL(NONZER)=VALUE
            VALUE = ABS (VALUE)
            IF (VALUE.GT.MAXINP)MAXINP = VALUE
C
C  GET PTCL ARRAY
C
          IF (J SUB.EQ.NODE)GO TO 100
            NODE=NODE +1
            DO 200 K=KODE,JSUB
               PTCL (K) = NONZER
 200        CONTINUE
            NODE = JSUB
            GO TO 100
C
C  LAST ELEMENT OF PTCL ARRAY
C
 300     NODE=NODE+1
         M=NEQNS+1
         DO 400 K=NODE, M
            PTCL (K) = NONZER+1
 400     CONTINUE
       RETURN
         END
C
C
C-----------------------------------------------------------------------
       SUBROUTINE ROWISE (NEQNS,NONZER,PTCL,ELCL,PTRW,ELRW)
C
C-----------------------------------------------------------------------
C
C THIS ROUTINE INPUTS THE MATRIX IN COLUMNWISE FORMAT AND FINDS ITS
C ROWISE FORMAT REPRESENTATION
C
C    MEANINGS OF VARIABLES:
C        PTRW (NEQNS+1) - ARRAY OF POINTERS OF ROW DATA STRUCTURE
C        ELRW (NONZER) - ARRAY OF COLUMN INDICES OF NONZERO ELEMENTS
C
         INTEGER*2 PTCL (1)ELCL(1),PTRW(1),ELRW(1),NEQNS,NONZER,
     1            I,J,K,M,JP, FIRST, ILAST
C
C  INITIALIZE POINTERS FOR ROWISE FORMAT
C
         M=NEQNS+1
         DO 100 I=1, M
            PTHW (I) = 0
 100     CONTINUE
```

```
C
C   DETERMINE POINTERS FOR ROWISE FORMAT
 C
                   DO 200 I=1, NONZER
                       J = ELCL (I) + 2
                       IF (J.LE.M) PTRW (J)=PTRW(J)+1
   200             CONTINUE
C
                    PTRW (1) =1
                    PTRW (2)=1
                    IF (NEQNS.EQ.1) GO TO 400
                       DO 300 I=3, M
                            PTRW (I)=PTRW (I)+PTRW (I-1)
   300                 CONTINUE
C
C  DETERMINE THE COLUMN INDICES AND THE MATRIX VALUES OF ROWISE FORMAT
C
  400              DO 600 I=1, NEQNS
                       IFIRST=PTCL (I)
                       ILAST = PTCL (I+1)-1
                       IF (ILAST.LT.IFIRST) GO TO 600
                           DO 500 JP= IFIRST, ILAST
                                 J = ELCL (JF)+1
                                 K=PTRW (J)
                                 ELRW (K)=I
                                 PTRW (J) = K+1
  500                      CONTINUE
  600             CONTINUE
                  RETURN
                  END
```

```
C---------------------------------------------------------------------------------------------------
C
         SUBROUTINE ENVMAT (NERNS,PTCL,ELCL,VMATCL,INVF,ENVRW,ENVCL,
     1                            ENVRW,XENVCL,EVRW,EVCL,DIAG)
C
C---------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE GETS THE ENCELOPE REPRESENTATION OF THE MATRIX FROM
C  ITS ENVELOPE STRUCTURE AND COLUMNWISE REPRESENTATION
C
C     MEANINGS OF VARIABLES :
C        EVRW (ENVRW)  - ARRAY WITH THE ENVELOPE ELEMENTS OF THE MATRIX
C                        LOWER TRMNGULAR PART
C        EVCL (ENVCL) -  ARRAY WITH THE ENVELOPE ELEMENTS OF THE MATRIX
C                        UPPER TRIANGULR PART
C        DIAG (NEQNS) -  ARRAY WITH THE MATRIX DIAGONAL ELEMENTS
C
             INTEGER*2 PTCL (1),ELCL(1)INVP(1),XENVRW(1),XENVCL(1),
     1                   NEQNS, ENVRW, ENVCL
             INTEGER*2 ISUE, JSUE, JSTRT, JSTOP, I,J,K
             REAL VMATCL (1),EVRW (1),EVCL(1),DIAG(1)
C
C   INITIALIZATION
C
             DO 100 I=1, ENVRW
                EVRW (I)=C.
   100        CONTINUE
              DO 200 I=1, ENVCL
                EVCL (I)=0.
   200        CONTINUE
C
C   INTRODUCE MATRIX ELEMENTS COLUMN BY COLUMN INTO THE ENVELOPE
C   FORMAT REPRESENTATION OF THE MATRIX
C
              DO 600 J=1, NEGNS
                JSUB=INVP (J)
                JSTRT=PTCL (J)
                JSTOP=PTCL (J+1)-1
                DO 500 I=JSTRT, JSTOP
                    ISUB=ELCL (I)
                    ISUB=INVP (ISUB)
                    IF ( ISUB. EQ.JSUB) GO TO 400
                        IF (ISUB.LT.JSUE) GO TO 300
C
C  ELEMENT OF THE MATRIX LOWER TRIANGULAR PART
C
                           K=XENVCL (JSUE+1)-JSUE+ISUB
                           EVRW (K)=VMATCL(I)
                           GO TO 500
C  ELEMENT OF THE MATRIX UPPER TRIANGULAR PART
C
  300                   K=XENVCL (JSUE+1)-JSUE+ISUB
                        EVCL (K)=VMATCL(I)
                        GO TO 500
C
C  ELEMENT OF THE MATIX DIGONAL
C
  400               DIAS (ISUE)=VMATCL(I)
  500           CONTINUE
  600        CONTINUE
             RETURN
             END
```

```
C---------------------------------------------------------------------------------------------------
C
            SUBROUTINE FACTOR(NEQNS,XENVRW,EVRW,XENVCL,EVCL,DIAG,NSINGL,
     1                           ADCL,INDMT,MXVAL,NZRNV)
C
C---------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE FACTORS A MATRIX OF ORDER GREATER THAN ONE INTC L* U  .
C  THE MATRIX IS STORED IN THE NONSYMMETRIC ENVELOPE FORMAT AND THE
   METHOD USED IS THE BORDERING METHOD.
C
          INTEGER*2 XENVRW(1),XENVCL(1)INDMAT(1),NEONS,ADCL,IXENRW,IBANRW,
     1               IXENCL,IBNCL,IFIRST,MINBAN,I,J,L,JSTRT,JSTOP,ISTRT,
     2                     NSINGL, IDIAG, NZRENV
          REAL EVRW(1),EVCL(1),DIAG(1),TEMP,XTOL,S,MAXVAL
          REAL*8 OPS,COUNT
C
          COMMON /RSOPS/ OPS
          COMMON /RSXTOL/ XTOL
C
          MAXVAL=0.
          NZRENV=0
          ADCL=0
          ISTRT=NSINGL+1
          DO 400 I=ISTRT, NEQNS
C
C   COMPUTE I-TH ROW OF LOWER TRIANGULAR FACTOR
C
             IXENRW=XENVRW(I)
             IBANRW=XENVRW(I+1)-IXENRW
             IF (BANRW.EQ.C) GO TO 100
                IFIRST=I-IBANRW
                IDIAG=0
                CALL LOWSOL (IBNRW,XENVCL(IFIRST),EVCL,DIAG(IFIRST),
     1             EVRW (IXENRW),IDIAG)
C
C   CALCULATE NUMBER OF NONZEROS IN I-TH ROW AND UPDATE MAXIMUM
C  ABSOLUTE VALUE OF FACTORS IF NECESSARY
C
                L=IXENRW+IBANRW-1
                DO 50 J=IXENRW,L
                      S = EVRW (J)
                     IF (S.EQ.O.)GO TO 50
                        NZRENV=NZRENV+1
                        S=ABS (S)
                        IF (S.GT.MAXVAL) MAXVAL=S
  50            CONTINUE
C
C   COMPUTE I-TH COLUMN OF UPPER TRIANGULAR FACTOR
C
 100         IXENCL=XENVCL (I )
             IBANCL=XENVCL (I+1)-IXENCL
             IF (IBNCL. EQ .0)GO TO 400
                IFIRST=I-IBNCL
                IDIAG=1
                CALL LOWSOL (IBANCL,XENVRW(IFIRST),EVRW,DIAG(IFIRST),
     1                  EVCL (IXENCL),IDIAG)
C
C  CALCULATE NUMBER OF NONZEROS IN I-TH COLUMN AND UPDME MAXIMUM
C  ABSOLUTE VALUE IF NECESSARY
C
                L=IXENCL+IBANCL-1
                DO 150 J=IXENCL, L
                   S=EVCL(J)
                   IF(S.EQ.C.)GO TO 150
                      NZRENV=NZRENV+1
                      S=ABS(S)
                      IF (S.GT.MAXVAL)MAXVAL= S
 150            CONTINUE
C
C   COMPUTE I-TH DIAGONAL ELEMENT OF MATRIX U
C
             IF  (IBANRW.EQ.O)GO TO 400
                MINBAN = IBANRW
                IF (IBNCL.LT. IBANRW)MINBAN=IBANCL
                TEMP = DIAG( I)
                L=XENVCL (I+1)-MINBAN
                JSTRT=XEMVRW (1+1)-MINBAN
                JSTOP=XENVRW (I+1)-1
                DO 200 J=JSTRT, JSTOP
```

```
                              TEMP=TEMP-EVRW (J)* EVCL (L)
                              L=L+1
  200                    CONTINUE
C
C   CHECK IF DIAGONAL ELEMENT OF U IS NONZERO
C
                      DIAG (I)=TEMP
                      COUNT=MINBAN
                      OPS=OPS+COUNT
                         S=ABS (TEMP)
                         IF (S.GT.MAXVAL) MAXVAL=S
                         IF (S.GE.XTCL) GO TO 400
                            DIAG (I)=TEMP+1.
                            ADCL=ADCL+1
                            INDMAT (ADCL)=I
  400       CONTINUE
            RETURN
            END
```

```
C-----------------------------------------------------------------------------------------------
C
           SUBROUTINE FENVRW (NEQNS,XADJRW,ADJNRW,PERM,INVP,XEVRW,
     1                        ENVRW, BANDRW)
C
C-----------------------------------------------------------------------------------------------
C
C  THIS ROUTINE FINDS THE ENVELOPE STRUCTURE OF THE LOWER PART OF THE
C  PERMUTED MATRIX
C
C     MEANINGS OF VARIABLES :
C         XENVRW (NEQNS-1) - ARRAY OF POINTERS OF ENVELOPE DATA STRUCTURE
C
          INTEGER*2 XADJRW(1),ADJNRW(1) ,PERM(1),INVP(1) ,XENVRW(1) ,
     1                NEQNS, BANDRW, ENVRW
          INTEGER*2 NABOR,I,J,BAND,IFBST,IPERM,JSTRT,JSTOP
C
          BANDRW=0
          ENVRW=1
          DO 200 I=1, NEQNS
             XENVRW (I)=ENVRW
             IPERM=PERM ( I)
             JSTRT=XADJRW (IPERM)
             JSTOP=XADJRW (IPERM+1)-1
             IF (JSTOP .LT. JSTRT) GO TO 200
C
C   FIND THE FIRST NONZERO IN ROW I, CALCULATE THE I-TH LOWER
C   BANDWIDTH AND UPDATE THE LOWER BANDWIDTH IF NECESSARY
C
               IFIRST=I
               DO 100 J=JSTRT, JSTOP
                  NFABOR= ADJNRW(J)
                  NABOR=INVP (NABOR)
                  IF (NABCR. LT. IFIRST)IFIRST=NABOR
  100          CONTINUE
               IBAND=I-IFIRST
               ENVRW=ENVRW+IBAND
               IF (BANDRW.LT.IBAND)BANDRW=IBAND
  200     CONTINUE
C
C  FIND THE LAST ELEMENT OF THE VECTOR XENVRW OF THE DATA STRUCTURE
C
          XENVRW (NEQNS-1) =ENVRW
          ENVRW=ENVRW-1
          RETURN
          END
C
C
C-----------------------------------------------------------------------------------------------
C
           SUBROUTINE FENVCL (NEQNS,XADJCL,ADJNCL, PERM,INVP,XENVCL,
     1                          ENVCL,BANDCL)
C
C-----------------------------------------------------------------------------------------------
C
C   THIS ROUTINE FINDS THE ENVELOPE STRUCTURE OF THE UPPER PART OF
C   THE PERMUTED MATRIX
C
C     MEANIGS OF VARIABLES:
C        XENVCL (NEQNS+1) - ARRAYOF POINTERS OF ENVELOPE DATA STRUCTURE
C
             INTEGER*2  XADJCL (1),ADJNCL(1),PERM(1) IMVP(1) XENVCL(1)
     1                  NEQNS, BANDCL, ENVCL
             INTEGER* 2 NABOR,I,J,IBAND,IFIRST,IPERM,JSTRT,JSTOP
C
             BANDCL=0
             ENVCL=1
             DO 200 I=1, NEQNS
                XENVCL (I)=ENVCL
                IPERM=PERM (I)
                JSTRT=XADJCL (IPERM)
                JSTOP=XADJCL (IPERM+1)-1
                IF (JSTOP.LT.JSTRT) GO TO 200
C
C  FIND THE FIRST NONZERO IN COLUMN I, CALCULATE THE I-TH UPPER
   BANDWIDTH AND UPDATE THE UPPER BANDWIDTH IF NECESSAPY
C
                  IFIRST = I
                  DO 100 J=JSTRT, JSTOP
                     NABOR=ADJNCL (J)
```

```
                        NABCR= INVF (NABOR )
                        IF (NABOR..LT.IFIRST) IFIRST=NABOR
  100               CONTINUE
                    IBAND=I-IFIRST
                    ENVCL=ENVCL+IBAND
                    IF(BANDCL.LT.IBAND)=IBAND OL= IBAND
  200       CONTINUE
C
C   FIND THE LAST ELEMENT OF THE VECTOR XENVCL OF DATA STRUCTURE
C
            XENVCL(NEQNS+1)=ENVOL
            ENVCL=ENVCL-1
            RETURN
            END
```

```
C-----------------------------------------------------------------------------------------------------
C
                SUBROUTINE FDOUGR(NFQNS,PTRW,ELRW,XADJRW,ADJNRW)
C
C----------------------------------------------------------------------------------------------------
C
C   THIS ROUTINE FINDS THE OUTER ADJACENCY LIST OF THE MATRIX GRAPH
C
C      MEANINGS OF VARIABLES:
C         XADJRW (NEQNS+1) - ARRAY OF POINTERS OF OUTER ADJACENCY LIST
C         ADJNRW (NZNDG) - ARRAY OF NODES OF OUTER ADJACENCY LIST
C
                  INTEGER*2 PTRW(1),ELRW(1),XADJRW(1)ADJNRW(1),NEGNS,I,
         1                      J,L,LROW,JSTRT,JSTOP
 C
                  LROW=1
                  XADJRW (1) =1
                  DO 300 I=1, NEGNS
                        JSTRT=PTRW (I)
                        JSTOP=PTRW (1+1)-1
                        IF (JSTOP.LT.JSTRT)GO TO 200
                         DO 100 J = JSTRT, JSTOP
                                L = ELRW (J)
                                 IF (L.EQ.I)GO TO 100
                                  ADJNRW (LROW)=L
                                  LROW=LROW+l
  100             CONTINUE
  200            XADJRW (I+1 )=LROW
  300        CONTINUE
         RETURN
            END
C
C
C-----------------------------------------------------------------------------------------------------
C
                SUBROUTINE FDINGR (NEQNS,PTCL,ELCL,XADJCL,ADJNCL)
C
C-----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE FINDS THE INNER ADJACENCY LIST OF THE MATRIX GRAPH
C
C     MEANINGS OF VARIABLES:
C         XADJCL (NEQNS+1) - ARRAY OF POINTERS OF INNER ADJACENCY LIST
C         ADJNCL (NZNDG) - ARRAY OF NODES OF INNER ADJACENCY LIST
C
               INTEGER*2 PTCL(1),ELCL(1),XADJCL(1),ADJNCL(1),NEQNS,
        1                 I,J,L,JSTRT,JSTOP,LCOL
C
                LCOL=1
                XADJCL (1)=1
                DO 300 I=1, NEGNS
                     JSTRT=PTOL (I)
                     JSTOP = PTOL (1+1)-1
                     IF (JSTOP.LT. JSTRT) GO TO 200
                         DO 100 J=JSTRT, JSTOP
                            L=ELCL (J )
                            IF (L.EQ.I)GO TO 100
                            ADJNCL (LCOL)=L
                                LCOL=LCOL+1
  100                    CONTINUE
  200             XADJOL (I+1)=LCOL
  300           CONTINUE
              RETURN
              END
```

```
C-----------------------------------------------------------------------------------------------------
C
          SUBROUTINE GENRCM (NEQNS,PERM, XADJRW, ADJNRW,XADJCL,ADJNCL,
     1                    DEG,MASK,XLS,NCOMP, LVTREE,NSINGL)
C
C-----------------------------------------------------------------------------------------------------
C
C   THIS ROUTINE FINDS THE CUTHILL-MCKEE (RCM=0) OR THE REVERSE CUTHILL
C -MCKEE (RCM=1) ORDERING FOR GENERAL GRAPH.FOR EACH CONNECTED
C  COMPONENT IN THE GRAPH GENRC7 OBTAINS THE ORDERING BY CALLING
C  SUBROUTINE RCMS
C
C    MEANINGS OF VARIABLES:
C        PERM (NEQNS) - ARRAY REPRESENTING THE ORDER OF THE ROWS AND
C                       COLUMNS OF THE PERMUTED MATRIX
C        DEG (NEQNS) - ARRAY CONTAINING THE DEGREES OF NODES OF MATRIX
C                      GRAPH
C        MASK (NEQNS) - ARRAY FOR MARKING COLUMNS AND ROWS.IF MASK (I)=0
C                       THEN ROW AND COLUMN I ARE NOT TO BE CONSIDERED
C        XLS (NLVL) - ARRAY OF POINTERS OF THE STARTING NODES OF EACH
C                     LEVEL OF THE LEVEL TREE
C        LVTREE (NCOMP) - ARRAY OF LEVEL TREES LENGTHS OF EACH CONNECTED
C                         COMPONENT
C
          INTEGER*2 XADJRW (1),XADJCL(1),ADJNRW(1),ADJNCL(1),PERM(1),
     1              MASK (1),XLS(1),LVTREE(1),NEQNS,NCOMP,NSINGL
          INTEGER*2 NUM,ROOT,COSIZE,NLVL,I
          REAL DEG (1)
C
C    MEANINGS OF REMAINING VARIABLES:
C        NUM - POINTS TO THE FIRST NODE OF CONNECTED COMPONENT
C        ROOT - FIRST NODE OF THE CONNECTED COMPONENT
C        COSIZE - NUMBER OF NODES OF CONNECTED COMPONENT
C        NLVL - LENGTH OF THE LAST GENERATED LEVEL TREE
C
C  FIND THE DEGREES DF NODES OF THE GRAPH AND GET SINGLETONS
C
          DO 100 I = 1, NEQNS
             MASK (I) = 1
  100     CONTINUE
          CALL FDDEG(NEQNS,XADJRW,XADJCL,MASK,PERM,DEG,NSINGL)
C
          NUM = NSINGL+1
          NCOMP=0
C
C  FOR EACH CONNECTED COMPONENT...
C
          DO 200 I=1, NEQNS
            IF (MASK (I) .EQ. 0)GO TO 200
               ROOT=I
C
C  FIND A PSEUDO-PERIPHERAL NODE ROOT. THE LEVEL STRUCTURE FOUND
C   BY FDROOT IS STORED AT PERM (NUM)
C
            CALL FDRCOT (ROOT, XADJRW, ADJNRW, XADJCL, ADJNCL,MASK,
     1                   NLVL, XLS, PERM (NUM), DEG)
               NCOMP=NCOMP+1
             LVTREE (NCOMP)=NLVL
C
C  ORDER THE CONNECTED COMPONENT WITH ROOT AT THE STARTING NODE
C
            PERM (NUM) =ROOT
            CALL ROMS (ROOT, XADJRW, ADJNRW, XADJCL, ADJNCL, MASK,
     1                     PERM(NUM)DEG,COSIZE )
C
               NUM=NUM+COSIZE
               IF (NUM.GT.NEQNS)RETURN
  200     CONTINUE
          END
C
C
C-----------------------------------------------------------------------------------------------------
C
          SUBROUTINE FDDEG(NEQNS,XADJRW,XADJCL,MASK,PERM,DEG,NS1NGL)
C
C-----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE FINDS THE DEGREES OF NODES OF THE MATRIX GRAPH AND
C  GETS THE SINGLETONS
```

```
               INTEGER*2 XADJRW (1),XADJCLW(1),MASK(1),PERM(1),NEQNS, I, NSlNGL
             REAL DEG (1),SUMDEG, OUTDEG, INDEG
C
C        MEANINGS OF VARIABLES:
C             OUTDEG – OUTER-DEGREE OF A NODE
C             INDEG - OUTER-DEGREE OF A NODE

               NSINGL =0
               DO 300 I=1, NEQNS
                   OUTDEG = XADJRW (I+1)-XADJRW (I)
                    INDEG = XADJCL (I+1)-XADJCL (I)
                   DEG (I)=OUTDEG*INDEG
                   SUMDEG=OUTDEG*INDEG
                   DEG (I)=100.*DEG(I)+SUMDEG
                   IF (SUMDEG.GT.O) GO TO 300
                       NSINGL=NSING=1
                       PERM (NSINGL)=l
                       MASK (I)=0
  300          CONTINUE
               RETURN
               END
C
C
C-----------------------------------------------------------------------------------------------
C
               SUBROUTINE FDROOT (ROOT, XADJW, ADJNRW,XADJCL,ADJNCL,MASK,
      1                                NLVL, XLS, LS, DEG)
C
C-----------------------------------------------------------------------------------------------
C
C  THIS ROUTINE IMPLEMENTS A MODIFIED VERSION OF THE GPS SCHEME TO
C   FIND PSEUDO-PERIPHERAL NODES
C
               INTEGER*2 XADJRW (1),ADJNRW(1),XADJCL (1),ADJNCL(1),LS(1),
      1                        MASK (1),XLS(1),ROOT,NLVL
             INTEGER *2 COSIZE, JSTRT,NODE, NUNLVL,J
               REAL DEG (1) MINDEG
C
C   DETERMINE THE LEVEL STRUCTURE ROOTED AT ROOT
C
               CALL ROOTLS (ROOT, XADJRW, ADJRW, XADJCL, ADJNCL, MASK, NLVL,
      1                                 XLS, LS)
C
               COSIZE=XLS(NLVL+1)-1
               IF (NLVLEQ.1 .OR. NLVL.EQ.COSIZE) RETURN
C
C   PICK A NODE WITH MINIMUM DEGREE OF THE LAST LEVEL
C
  100          JSTRT=XLS (NLVL)
               ROOT=LS (JSTRT)
               IF (CCSIZE.EQ.JSTRT) GO TO 300
                    MINDEG=DEG (ROOT)
                    JSTRT=JSTRT+1
                    DO 200 J=JSTRT, CCSIZE
                       NODE=LS (J)
                       IF (DEG(NOOE).GE.MINDEG) GO TO 200
                               ROOT=NODE
                               MINDEG=DEG (NODE)
  200               CONTINUE
C
C  AND GENERATE ITS LEVEL STRUCTURE
C
  300          CALL ROOTLS (ROOT,XADJRW,ADJNRW,XADJCL,ADJNCL MASK,NUNLVL
       1                       XLS, LS)
               IF (NUNLVL.LE.NLVL) RETURN
                  NLVL = NUNLVL
                  IF (NLVL.LT.CCSIZE) GO TO 100
                      RETURN
               END
C
C
C-----------------------------------------------------------------------------------------------
C
               SUBROUTINE ROOTLS (ROOT,XADJRW,ADJNRW,XADJCL,,ADJNCL, MASK,
      1                               NLVL, XLS, LS)
C
C-----------------------------------------------------------------------------------------------
C   THIS ROUTINE GENERATES THE LEVEL STRUCTURE ROOTED AT THE NODE
C    ROOT FOR THE CONNECTED COMPONENT SPECIFIED BY MASK
```

```
C
            INTEGER*2 XADJRW(1),ADJNRW(1),XADJCL(1)ADJNCL(1),MASK(1),
     1               XLS(1) ,LS (1),ROOT,NLVL
            INTEGER*2 JSTRT, JSTOP,I,J,LBEGIN,LVLEND,CCSIZE,NBR,NODE
C
            MASK (ROOT)=0
            LS (1)=ROOT
            NLVL=0
            COSINZE=1
C
C  LBEGIN AND LVLEND POINT TO THE BEGINNING AND END OF THE CURRENT
C   LEVEL
C
  100       LBEGIN=LVLEND+1
            LVLEND=CCSIZE
            NLVL=NLVL+1
            XLS (NLVL)=LBEGIN
C
C  GENERATE THE NEXT LEVEL BY FINDING ALL THE MASKED NEIGHBOURS OF
C  NODES IN CURRENT LEVEL
C
            DO 500 I=LBEGIN,LVLEND
               NODE=LS (I)
               JSTRT=XADJRW (NODE)
               JSTOP = XADJRW (NODE-1 )-1
               IF (JSTOP.LT.JSTRT)GO TO 300
                  DO 200 J=JSTRT, JSTOP
                     MBR=ADJNRW (J)
                     IF (MASK (NBR).EQ.0)GO TO 200
                        CCSIZE=CCSIZE+1
                        MASK (NBR)=0
                        LS (CCSIZE)=NBR
  200             CONTINUE

C
  300          JSTRT=XADJCL (NODE)
               JSTOP=XADJCL (NODE+1)-1
               IF (JSTOP.LT.JSTRT) GO TO 500
                  DO 400 J=JSTRT, JSTOP
                        NBR=ADJNCL (J)
                        IF (MASK (NBR). EQ.0) GO TO 400
                           CCSIZE = CCSIZE+1
                           LS (CCSIZE)=NBR
                           MASK (NER)=0
  400             CONTINUE
  500       CONTINUE
C
C IF THE LEVEL WIDTH IS NONZERO GENERATE NEXT LEVEL
C
            IF (CCSIZE.GT.LVLEND) GO TO 100
C
C  RESET MASK TO ONE FOR THE NODES IN LEVEL STRUCTURE
C
               XLS (NLVL+1) =LVLEND+1
               DO 600 I=1 CCSIZE
                  NODE=LS(I)
                  MASK (NODE)=1
  600          CONTINUE
            RETURN
            END
C
C
C-----------------------------------------------------------------------------------------------------
C
            SUBROUTINE ROMS (ROOT, XADJRW, ADJNRW, XADJCL, ADJNCL, MASK, PERM,
     1                     DEG,LABR)
C
C-----------------------------------------------------------------------------------------------------
C
C  THIS ROUTINE NUMBERS A CONNECTED COMPONENT SPECIFIED BY MASK AND
C  ROOT USING THE CUTHILL-MCKEE ALGORITHM (RCM=0) OP THE REVERSE
C  CUTHILL-MCKEE ALGORITHM (RCM=1). THE NUMERING IS TO BE STARTED
C  AT THE NODE ROOT
C
            INTEGER*2 XADJRW (1),ADJNRW(1),XADJOL(1),ADJNCL(1),MASK(1),
     1                     PERM (1),ROOT,RCM
            INTEGER*2 LEEGIN, LEND, NER, LNER, FNBR, JSTRT, JSTOP, I, J, K, L, LFER
            REAL DEG (1)
C
            COMMON/ISRCM/ RCM
```

```
C
            MASK (ROOT)=0
            LEND=0
            LNBR=1
C
C   LBEGIN AND LEND POINT TO THE BEGINNING AND THF END OF THE CURRENT
C   LEVEL OF THE CONNECTED COMPONENT
C
  100       LBEGIN=LEND+1
            LEND=LNBR
            DO 900 I = LBEGIN, LEND
C   FOR EACH NODE OF THE CURRENT LEVEL OF THE CONNECTED COMPONENT
C
                  NODE=PERM (I)
                  JSTRT=XADJRW (NODE)
                  JSTOP=XADJRW (NODE+1)-1
C
C  FIND THE UNNUMBERED NEIGHBOURS OF NODE.FNBR AND LNBR POINT TO THE
C  FIRST AND LAST UNNUMBERED NEIGHBOURS RESPECTIVELY OF THE CURRENT
C  NODE (NAMED NODE) IN PERM
C
                  FNBR=LNBR+1
                  IF (JSTOP.LT.JSTRT) GO TO 300
                     DO 200 J = JSTRT, JSTOP
                        NBR=ADJNRW (J)
                        IF (MASK (NBR).EQ.0) GO TO 200
                           LNBR=LNGR+1
                           MASK (NER)=0
                           PERM (LNBR)=NBR
  200                CONTINUE
C
  300             JSTRT=XADJCL (NODE)
                  JSTOP=XADJCL (NODE+1)-1
                  IF (JSTDP.LT.JSTRT) GO TO 500
                     DO 400 J=JSTRT, JSTOP
                        NBR= ADJNCL (J)
                        IF (MASK (NBR).EQ.0) GO TO 400
                          LNBR=LNBR+1
                           MASK (NBR)=0
                           PERM (LNBR) =NBR
  400                CONTINUE
C
  500             IF (FNBR.GE.LNBR) GO TO 900
C
C  SORT THE NEIGHBOURS OF NODE IN INCREASING ORDER BY DEGREE.LINEAR
C  INSERTATION IS USED
C
                  K=FNBR
 600              L=K
                        K = K+1
                        NBR=PERM (K)
  700                   IF (L.LT.FNBR) GO TO 300
                          LPERM=PERM (L)
                           IF (DEG (LPERM).LE.DEG(NBR))GO TO 800
                             PERM (L+1) =LPERM
                              L = L-1
                               GO TO 700
  800                   PERM (L+1) =NBR
                        IF (K.LT.LNBR) GO TO 600
  900       CONTINUE
C
            IF (LNBR.GT.LEND) GO TO 100
              IF (LNBR.LE.1) RETURN
C
C  CUTHILL-MCKEE ALGORITHM
C
                  IF (RCM EQ.0) RETURN
C
C  REVERSE CUTHILL-MCKEE ALGORITHM : REVERSE ORDERING
C
                     K = LNBR/2
                     L=LNBR
                     DO 1000 I=1, K
                         LPERM=PERM(L)
                         PERM (L)=PERM(I)
                         PERM (I) =LPERM
                         L = L-1
  1000               CONT I NUE
                     RETURN
            END
```