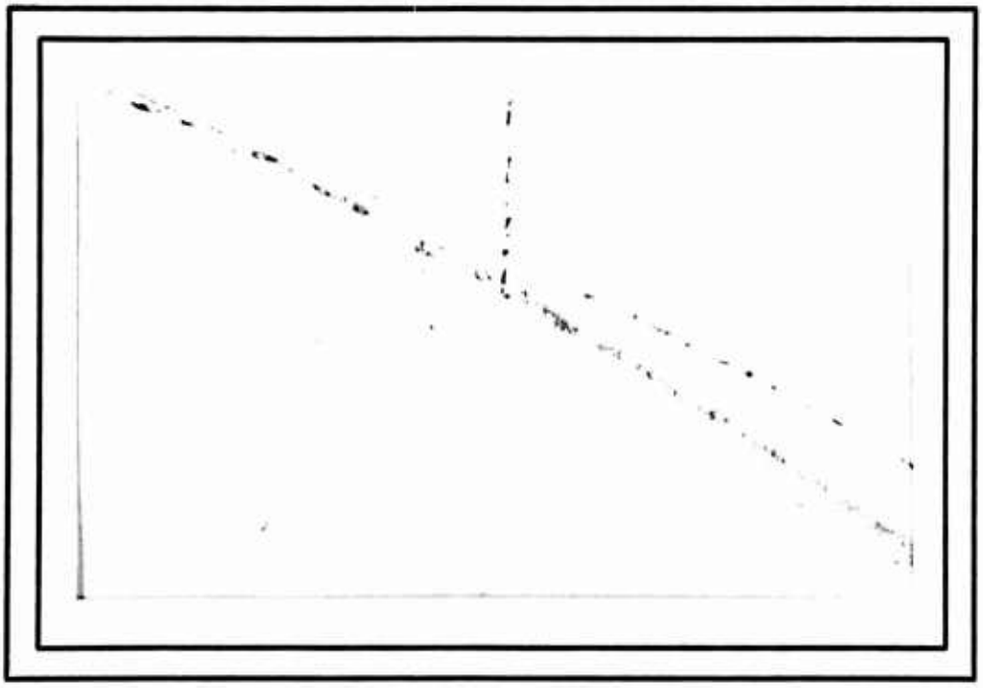

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

(12)

(9) (14)

Technical Report TR-394
NSF-OCA-GJ-35568X-394
N00014-67-A-0239-0021
NSF GJ-35568

(11) August 1975

(16) NR-044-431

Further Programs for the Solution of Large Sparse Systems of Linear Equations.

by Werner C. K./Mesztenyi and John C./Rheinboldt

(12) 51 p.

Abstract

A package of FORTRAN subroutines is presented for the solution of nonsymmetric or symmetric sparse linear systems by triangular decomposition. Two principal aims are (1) to handle matrices which originally fit into primary core storage but do so no longer after decomposition, and (2) to solve a sequence of linear systems all of which have the same sparsity structure by generating--in secondary storage--a record of the decomposition process in the form of an integer array. Some experimental results using the package are included.

FURTHER PROGRAMS FOR THE SOLUTION OF LARGE SPARSE SYSTEMS OF LINEAR EQUATIONS[1)]

by

Charles K. Mesztenyi[2)] and Werner C. Rheinboldt[2)]

1. Introduction

In a previous report [1] a package of FORTRAN subroutines was presented for the solution of a linear system

$$Ax = b \tag{1}$$

based on triangular decomposition of the (symmetric or nonsymmetric) matrix A. The underlying data structure was motivated by a more general arc-graph structure discussed in [2].

The programs given here have the same purpose but pursue the following two different aims:

a. To handle matrices which originally fit into primary core storage but do so no longer after decomposition.

b. To solve a sequence of systems (1) all of which have the same sparsity structure. This case arises, for example, in the solution of nonlinear systems by Newton's method.

The first aim is accomplished during decomposition by writing the decomposed part of the matrix into secondary storage and using its place for newly introduced nonzero elements. In order to meet the second aim

[1)] This work was supported in part by the National Science Foundation under Grant GJ-35568X and the Office of Naval Research under Grant N00014-67-A-0239-0021 (NR-044-431).

[2)] Computer Science Center, University of Maryland, College Park, Md. 20742.

we follow an idea in [3] and generate--in secondary storage--a record of the decomposition process in the form of an integer array. This record can be used to decompose any matrix with the same sparsity structure provided there are no round-off problems.

2. Some Background

The desired triangular decomposition of the $n \times n$, nonsingular matrix A has the form

$$PAQ = LU, \quad L = I + L^0 , \tag{2}$$

where L^0 is strictly lower triangular, U upper triangular, and the permutation matrices P,Q define the pivoting sequence. The decomposition is accomplished in n steps, such that

$$P_iAQ_i = (I+L_i^0)U_i + A_i, \quad i = 0,1,\ldots,n \tag{3}$$

where the first i rows and i columns of A_i, the last $n-i$ columns of L_i^0, and the last $n-i$ rows of U_i are zero, respectively. Moreover, $PP_i^TL_i^0$ has the same first i columns as L^0 and $U_iQ_i^TQ$ the same first i rows as U. This latter fact allows us to keep L_i^0 and U_i in secondary storage.

Let $\eta(B)$ denote the number of nonzero elements of any matrix B. Then the storage required before and after decomposition is of the order of $m_0 = \eta(A)$ and $m_2 = \eta(U) + \eta(L^0)$, respectively. Furthermore, $m_1 = \max_i \eta(A_i)$ is the maximal storage needed for the matrices A_i. Clearly, we have

$m_0 \le m_1 \le m_2$ and, in practice, it turns out that m_2-m_0 is very much larger than m_1-m_0. In fact, sometimes we found m_1 to be equal to m_0 (see Section 5). Hence by retaining only the A_i in primary storage we require, in general, only little more storage than for A itself.

The basic storage structure allows for easy modification of the pivoting strategy. In fact, in the nonsymmetric case the pivot selection is handled by an easily replacable subroutine. We use here the well-known minimal degree algorithm. If S_i is the set of nonzero elements of A_i, then for any $x \in S_i$ we denote by $R_i(x)$ and $C_i(x)$ the subsets of S_i consisting of the elements in the same row and column as x, respectively. Now, with $E_i(x) = R_i(x)$ if $|R_i(x)| \le |C_i(x)|$ and otherwise $E_i(x) = C_i(x)$, the set of potential pivots is given by

$$S_i^0 = \{x \in S_i;\ |a(x)| \ge \mu \max_{z \in E_i(x)} |a(z)|\}. \tag{4}$$

Here $a(z)$ is the value of the matrix element corresponding to z and $\mu \in [0,1]$, a user-defined parameter. The ith pivot is then the element $x \in S_i^0$ for which $(|R_i(x)|-1)(|C_i(x)|-1)$ is minimal. Generally, with decreasing μ the fill-in decreases while the round-off influence increases.

For symmetric A it is assumed that the pivots remain on the main diagonal and hence that $Q = P^T$. In that case each matrix A_i is again symmetric. If D_i is the set of nonzero diagonal elements of A_i, then the ith pivot is the element x of the set

$$D_i^0 = \{z \in D_i;\ |a(z)| \ge \mu \max_{y \in D_i} |a(y)|\} \tag{5}$$

for which the number of nonzero elements in its row is minimal.

It is theoretically possible to use the value $\mu = 0$. In that case, (4) and (5) show that any nonzero element of S_i or D_i, respectively, is a potential pivot. Then the pivot selection depends only on the sparsity structure and not on the elements of the matrix--but, of course, the round-off influence may be considerable.

3. Basic Storage Arrangements

3.1 Matrices in Primary Storage: As mentioned before, the basic storage structure used here is the same as that in [1]. We summarize it briefly.

Set $N = \{1,2,\ldots,n\}$ and let $S \subset N \times N$ be the set of locations corresponding to the nonzero elements of a given $n \times n$ matrix A. We number the elements of S from $n+1$ to $n+m$, $m = |S|$, that is, we introduce a bijective mapping

$$\nu: S \to \{n+1,\ldots,n+m\} . \tag{6}$$

Now define two integer arrays RY and CY each of length $n+m$ in which the relative addresses $n+1,\ldots,n+m$ correspond to the elements of S in the order provided by ν. The images $\nu(R_i)$ of the row sets

$$R_i = \{(i,k) \in S;\ \text{some } k \in N\},\ i \in N$$

form a partition of $\{n+1,\ldots,n+m\}$. For any set $\nu(R_i) = \{i_1,\ldots,i_k\}$ we link the locations $i,i_1,i_2,\ldots,i_k$ into a circular list

$$i_1 = RY(i),\ i_{j+1} = RY(i_j),\ j = 1,\ldots,k-1,\ i = R(i_k) \tag{7}$$

where for practical reasons

(8) $$i_1 > i_2 > \ldots > i_k > i \ .$$

Analogously, we proceed with the images $\nu(C_i)$ of the column sets

$$C_j = \{(k,j) \in S;\ \text{some } k \in N\},\ j \in N$$

in the array CY.

In order to store the associated matrix elements a third array A is, of course, needed. Thus, for example, the matrix

$$\begin{pmatrix} 0 & 1 & 0 & 3 \\ 0 & 0 & 5 & 0 \\ -1 & 0 & 2 & 0 \\ 0 & -2 & 0 & 0 \end{pmatrix}$$

may be stored as follows:

loc	1	2	3	4	5	6	7	8	9	10
RY	10	8	9	7	3	1	4	2	5	6
CY	5	7	9	10	1	2	6	3	8	4
A	*	*	*	*	-1	1	-2	5	2	3

We shall refer to RY and CY as the sparsity structure arrays and to A as the coefficient array.

For symmetric A the set S only needs to be the set of locations of the nonzero elements in the upper (or lower) triangle (including the diagonal) of A. Moreover, we assume always that in the symmetric case all diagonal elements are nonzero. Then the first n cells of the sparsity structure arrays RY and CY are no longer needed if (6) is changed to

$$\nu\colon S \to \{1,2,\ldots,m\},\ \nu(i,i) = i, i=1,\ldots,n\ .$$

There is no need to repeat the details; the resulting data arrangement should be self-evident from the following example:

$$\begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 2 & 0 & -2 \\ -1 & 0 & 3 & 0 \\ 0 & -2 & 0 & 4 \end{pmatrix}$$

loc	1	2	3	4	5	6
RY	5	6	3	4	1	2
CY	1	2	5	6	3	4
A	1	2	3	4	-1	-2

During decomposition, this storage arrangement is used for the matrices A_i, $i = 0,\ldots,n$. When a nonzero element of A_i remains in A_{i+1} its position in the RY,CY,AY arrays is maintained. After each pivoting step the pivot row and column are written on secondary storage and the corresponding cells in the storage arrays are freed, that is, the circular linkages containing these elements are modified appropriately. The resulting free locations are reassigned when fill-in occurs.

3.2 Triangular Matrices in Secondary Storage: The programs here are written for use with a random-access secondary storage device. Some information about the necessary I/O routines is provided in Section 4.1 below.

In the nonsymmetric case the triangular matrices L and U are written as two arrays of pairs of numbers. The L-array contains the columns of L in consecutive order and each column has the form

$$\begin{matrix} -i_c, -i_r \\ j_1, \ell_{j_1, i_c} \\ \vdots \\ j_k, \ell_{j_k, i_c} \end{matrix}$$

where i_c and i_r are the column- and row-index, respectively, of the pivot (stored negatively) and j_i represents the row index and ℓ_{j_i,i_c} the value of each nonzero element in the particular column. Similarly the U-array contains the rows of U in consecutive order, each of them in the form

$$\begin{matrix} j_1,u_{i_r,j_1} \\ \vdots \\ j_k,u_{i_r,j_k} \\ -i_c,p_{i_c} \end{matrix}$$

Here i_c is the column index of the pivot and p_i its value, while j_i,u_{i_r,j_i} denote the column index and value, respectively, of each nonzero element in the row. The entire U-array is initialized by a dummy pair -1,-1.

For the backsubstitution programs it is assumed that the L-array is read forward and the U-array backward.

In the symmetric case, there is, of course, no need for both the L- and U-array. Accordingly, only the L-array is set up containing the columns of L in consecutive order, each one in the form

$$\begin{matrix} -i_d,d_{i_d} \\ j_1,\ell_{j_1,i_d} \\ \vdots \\ j_k,\ell_{j_k,i_d} \\ -i_d,d_{i_d} \end{matrix}$$

Here i_d is the index of the pivot (on the diagonal) and d_i its value and j_i, ℓ_{j_i}, i_d have the previous meaning. The header at the beginning and end of each column is needed, since during backsubstitution the array is read once forward and once backward.

3.3 Representation of the Decomposition Record: As mentioned in the introduction, the programs can generate a record of the decomposition process for later use with any other matrix of the same sparsity type. This record is in the form of an array of positive integers in secondary storage. For each pivoting step the following information is recorded:

Nonsymmetric Case:

$$i_x,\ i_c,\ i_r,\ k_c,\ x_1,\ j_1,\ \cdots,\ x_{k_c},\ j_{k_c}$$
$$k_r,\ y_1,\ m_1,\ \cdots,\ y_{k_r},\ m_{k_r}$$
$$\ell_1,\ \ell_2,\ \cdots,\ \ell_t$$

i_x relative location of the pivot in the RY,CY arrays

i_c, i_r the column- and row-index of the pivot, respectively

k_c, k_r the number of nonzero elements in the pivot-column and pivot-row, respectively

x_i, j_i relative location (in CY) and row-index, respectively, of the nonzero elements in the pivot column

y_i, m_i relative location (in RY) and column-index, respectively, of the nonzero elements in the pivot row

ℓ_i relative locations of the elements in A_i which must be modified at the step, $t = k_c \cdot k_r$

Symmetric Case:

$i, k, y_1, m_1, \ldots, y_k, m_k$

$\ell_1, \ldots, \ell_t$

i index of the pivot and hence also its relative location in the RY,CY arrays

k the number of nonzero elements in the pivot row. Each of these elements is identified by a pair (y_j, m_j) as in the nonsymmetric case

ℓ_j the relative locations of the elements in A_i to be modified, $t = k(k-1)$

4. Description of the Programs

The package consists of four groups of subroutines with names INT, BLD, DEC, and SLV; in addition, there is a pivot selection routine PVT01 for the nonsymmetric case and a set of I/O routines for interface with the random storage device.

The INT programs initialize the storage area and have to be called first. The BLD routines establish the data structure described in Section 3.1 for the given matrix A. Then the DEC routines are called to perform the decomposition of the matrix and/or to generate a record of the decomposition process. Finally, if applicable, the SLV routines are used to obtain the solution of the given system (1) by backsubstitution.

In general, any efficient routine for building up the basic data structure from given data about the matrix depends strongly on the details

of the files used. The BLD programs presented here avoid all assumptions about file formats, etc., by establishing the data structure one matrix element at a time. In other words, the chosen BLD routine has to be called once for each nonzero matrix element. For many practical purposes this may be inefficient. The routines were included principally for the sake of completeness; it should be easy to rewrite them for any specific application.

The names of all subroutines in the four principal groups are preceded by the letters S or N for the case of symmetric or nonsymmetric matrices, respectively. The names of the subroutines in the INT, BLD, DEC group are ending with one of the numerals 0, 1 or 01. This indicates the following alternatives:

0 - Initialize or build only the sparsity structure arrays of the matrix or generate a record of the decomposition based solely on the sparsity structure. These routines are only available for symmetric matrices; for nonsymmetric matrices it is not an advisable approach since the resulting round-off error could be severe.

1 - Initialize or build only the coefficient array for the matrix elements, or decompose the matrix using a previously generated record of a decomposition for matrices with the same sparsity structure.

01 - Initialize or build both the sparsity structure arrays and the coefficient array, or decompose the given matrix and, optionally, generate a record of the decomposition.

The pivot selection for the nonsymmetric case is performed by the routine PVT01. For the symmetric case, pivot selection is incorporated within the routines SDEC0 and SDEC01.

4.1 Catalog of Subroutines: In this subsection we list the various subroutines of the package together with their calling sequences and brief descriptions of their purposes. The arguments in the calling sequences are discussed in the next subsection.

INT - Routines

SINT0(MD,RY,CY,ND)

Initialize the sparsity structure arrays of a symmetric matrix.

SINT01(MD,FD,RY,CY,A,AN,ND)

Initialize the sparsity structure arrays and the coefficient array of a symmetric matrix.

SINT1(MD,FD,AN)

Initialize the coefficient array of a symmetric matrix.

NINT01(MD,FD,RY,CY,AN,NDR,NDC)

Initialize the sparsity structure arrays and the coefficient array of a nonsymmetric matrix.

NINT1(MD,FD,AN)

Initialize the coefficient array of a nonsymmetric matrix.

BLD - Routines

All routines add a matrix element with value V, row index I, and column index J to the structure. Note that in the symmetric case only the nonzero elements in the upper (or lower) triangle and the diagonal should be given.

SBLD0(I,J,MD,RY,CY,ND)

Insert element (I,J) into the sparsity structure arrays of a symmetric matrix.

SBLD01(I,J,V,MD,FD,RY,CY,A,AN,ND)

Insert element (I,J) into the sparsity structure arrays of a symmetric matrix and a corresponding value V into the coefficient array.

SBLD1(I,J,V,MD,FD,A,AN)

Associate a value V to element (I,J) of a symmetric matrix. The V-values must be in the same order in which the (I,J)-values were read-in during prior construction of the corresponding sparsity structure arrays by SBLD0 or SBLD01.

NBLD01(I,J,V,MD,FD,RY,CY,A,AN,NDR,NDC)

Insert element (I,J) into the sparsity structure arrays of a nonsymmetric matrix and a corresponding value V into the coefficient array.

NBLD1(I,J,V,MD,FD,A,AN)

Associate a value V to element (I,J) of a nonsymmetric matrix. The V-values must be in the same order in which the (I,J)-values were read-in during prior construction of the corresponding sparsity structure arrays by NBLD01.

DEC - Routines

SDEC0(MD,RY,CY,ND,IP,IE,IH)

Generate a record of the decomposition of a symmetric matrix on the basis of the given sparsity structure.

SDEC01(MD,FD,RY,CY,A,AN,ND,IP,IE,IH)

Decompose a given symmetric matrix and optionally (MD(3)≠0) generate a record of the decomposition.

SDEC1(MD,FD,A,AN,IE)

Decompose a given symmetric matrix using a previously generated decomposition record.

NDEC01(MD,FD,RY,CY,A,AN,NDR,NDC,IPR,IPC,IE,IH,NG1,NG2)

Decompose a given nonsymmetric matrix and optionally (MD(3)≠0) generate a decomposition record.

NDEC1(MD,FD,A,AN,IE,IH)

Decompose a given nonsymmetric matrix using a previously generated decomposition record.

Pivot Routine

PVT01(I,N,IX,KR,KC,F,RY,CY,A,IPR,IPC,NDR,NDC,IE,IH)

Select the next pivot by the minimal degree algorithm during the decomposition of a nonsymmetric matrix. The routine is used by NDEC01.

SLV - Routines

These routines use the decomposed matrix in secondary storage. The right side of the system is given in the form of the input array X which in turn may be the same as the output array Y of the solution. The routines may be called repeatedly for different right sides.

SSLV(MD,X,Y)

Return the solution Y of the symmetric system with right side in X.

NSLV(MD,X,Y,AN)

Return the solution Y of the nonsymmetric system with right side in X.

I/O - Routines

The I/O routines for communication with the random access storage device are not in basic FORTRAN. They should be modified to suit a user's machine configuration. The routines have the following entries:

For I/O of decomposition record (array of positive integers)

DWI - Initialize for writing.

DW(K) - Write K as next entry of the array.

DWE - Terminate writing.

DRI - Initialize for reading.

DR(K) - Return the next entry of the array in K.

For I/O of symmetric decomposed matrix (array of pairs)

SVWI - Initialize for writing.

SVW(ℓ,s) - Write (ℓ,s) as next entry of the array. ℓ-signed integer, s-real.

SVWE - Terminate writing.

SVRI - Initialize for reading (forward).

SVRF(ℓ,s) - Return the next entry of the array in ℓ and s.

SVRB(ℓ,s) - Return the previous entry of the array in ℓ and s.

For I/O of a nonsymmetric decomposed matrix (two arrays of pairs)

NVWI - Initialize both files for writing.

NVWF(ℓ,s) - Write (ℓ,s) as next entry of the L-array (ℓ-signed integer, s-real).

NVWB(ℓ,s) - Write (ℓ,s) as next entry of the U-array.

NVWE - Terminate writing of both files.

NVRI - Initialize for reading, file L forward, file U backward.

NVRF(ℓ,s) - Return next entry of L-array in (ℓ,s).

NVRB(ℓ,s) - Return previous entry of U-array in (ℓ,s).

The following Table 1 shows the usage of the I/O routines by the various main routines:

Table 1

I/O Routine Usage

Program	DMR Decomposition Record File 10		SVMR Symmetric Dec. Matrix File 11		NVMR Nonsymmetric Dec. Matrix Files 12, 13	
	Write	Read	Write	Read	Write	Read
SDEC0	X					
SDEC01	∅		X			
SDEC1		X	X			
SSLV				X		
NDEC01	∅				X	
NDEC1		X			X	
NSLV						X

X - routine used

∅ - use is optional, depending on user's request

4.2 Arguments: The arguments in the various calling sequences either reference single values or data arrays. For simplicity the single variables are collected in two arrays, an integer array

$$MD(I),\ I = 1,2,\ldots,8$$

and a real array

$$FD(I),\ I = 1,2,\ldots,7\ .$$

The first three values of MD and the first two of FD are to be supplied by the user; the others represent output of various other routines. Care should be taken that these latter values are not modified whenever they are still to be used as input by other routines.

The use of the various arguments by the routines in the package is summarized in Tables 2 and 3 below.

MD - Array

MD(1) = N The dimension of the matrix; to be supplied by the user.

MD(2) = MX The lengths of the arrays RY, CY and A. If the decomposition of the matrix requires more internal storage, that is, if $m_1 >$ MX, then the error indicator MD(4) is set to one and the process is terminated with a return to the user's main program.

MD(3) If this indicator is zero, the DEC01 program does not produce a decomposition record; for any nonzero value of MD(3) such a record is generated.

MD(4) Error indicator set as follows:

= 0 : no error.

= 1 : storage overflow; MX is too small for decomposition.

= 2 : on the basis of the sparsity structure (independent of the values of the elements) the matrix is singular.

= 3 : the matrix is declared numerically singular.

MD(5) In the nonsymmetric case equal to M0 + N where M0 is the number of nonzero elements in the matrix; in the symmetric case equal to M0, the number of nonzero elements on the diagonal and in the upper (or lower) triangle of the matrix.

MD(6) The length of the actually utilized portion of the arrays RY, CY or A, equal to M1 + N or M1 in the nonsymmetric or symmetric case, respectively.

MD(7) In the nonsymmetric case equal to M2 + N where M2 is the number of nonzero elements in the decomposed matrix; in the symmetric case equal to the nonzero elements on the diagonal and in the lower triangle after the decomposition.

MD(8) Length of the decomposition record.

FD - Array

FD(1) A tolerance EPS supplied by the user. If a pivot value in magnitude is less than EPS the matrix is considered to be numerically singular and the decomposition is terminated.

FD(2) Initial input by the user containing the pivot selection parameter μ.

FD(3) Largest coefficient value in magnitude in the original matrix. Initialized by the INT routines and updated by the BLD routines.

FD(4) Largest coefficient value in magnitude encountered during decomposition, calculated by the DEC routines.

FD(5) Natural logarithm of the absolute value of the determinant calculated by the DEC routines.

FD(6) The sign of the determinant as +1.0 or -1.0 calculated by the DEC routines.

FD(7) The natural logarithm of the product of the L_2 norms of the row vectors of the original matrix calculated by the DEC routines.

Data Arrays

RY(MX),CY(MX) Integer arrays for the sparsity structure. Their length MX is specified by MD(2).

A(MX) Real array for the values of the nonzero matrix elements.

AN(N) Real array of length N (see MD(1)) used to collect row-vector norms of the matrix.

ND(N) For symmetric A.

NDR(N),NDC(N) For nonsymmetric A. Integer arrays of length N containing the number of nonzero elements by row (or column).

IE(N),IH(N) For any matrix.

IP(N) For symmetric matrices.

IPR(N),IPC(N)
NG1(N),NG2(N) For nonsymmetric matrices. Temporary arrays of length N.

Except for the last seven temporary arrays all data arrays are initialized in the appropriate INT routines. All integer arrays are used only for storing nonnegative integers. Thus for particular computers these arrays could be packed together.

Name / Type / Length / Program	I	J	V	MD	FD	RY	CY	A	AN	ND	IP	IE	IH	X	Y
	I	I	R	I	R	I	I	R	R	I	I	I	I	R	R
	1	1	1	8	7	M_1	M_1	M_1	N	N	N	N	N	N	N
SINT0	-	-	-	US	-	S	S	-	-	S	-	-	-	-	-
SINT01	-	-	-	US	US	S	S	S	S	S	-	-	-	-	-
SINT1	-	-	-	UPS	US	-	-	-	S	-	-	-	-	-	-
SBLD0	U	U	-	PS	-	PS	PS	-	-	PS	-	-	-	-	-
SBLD01	U	U	U	PS	PS	PS	PS	PS	PS	PS	-	-	-	-	-
SBLD1	U	U	U	PS	PS	-	-	PS	PS	-	-	-	-	-	-
SDEC0	-	-	-	PS	-	PT	PT	-	-	PT	T	T	T	-	-
SDEC01	-	-	-	PS	PS	PT	PT	PT	PT	PT	T	T	T	-	-
SDEC1	-	-	-	PS	PS	-	-	PT	PT	-	-	T	-	-	-
SSLV	-	-	-	PS	-	-	-	-	-	-	-	-	-	U	Ø

Table 2

Usage of Arguments in the Symmetric Case

(for legend see Table 3)

Table 3

Usage of Arguments in the Nonsymmetric Case

Name / Type / Length / Program	I	J	V	MD	FD	RY	CY	A	AN	NDR	NDC	IPR	IPC	IE	IH	NG1	NG2	X	Y
	I	I	R	I	R	I	I	R	R	I	I	I	I	I	I	I	I	R	R
	1	1	1	8	7	M_1	M_1	M_1	N	N	N	N	N	N	N	N	N	N	N
NINT01	-	-	-	US	US	S	S	-	S	S	S	-	-	-	-	-	-	-	-
NINT1	-	-	-	UPS	US	-	-	-	S	-	-	-	-	-	-	-	-	-	-
NBLD01	U	U	U	PS	PS	PS	PS	PS	PS	PS	PS	-	-	-	-	-	-	-	-
NBLD1	U	U	U	PS	PS	-	-	PS	PS	-	-	-	-	-	-	-	-	-	-
NDEC01 (PVT01)	-	-	-	PS	PS	PT	PT	PT	PT	PT	PT	T	T	T	T	T	T	-	-
NDEC1	-	-	-	PS	PS	-	-	PT	PT	-	-	-	-	T	T	-	-	-	-
NSLV	-	-	-	PS	-	-	-	-	T	-	-	-	-	-	-	-	-	U	Ø

Legend: Argument type: I - integer
R - real

Entries: U - user-supplied data
S - upon exit, the argument contains data to be saved for other reasons
P - contains data generated by previously called program
T - temporary storage
Ø - output result

5. Some Experimental Results

Two groups of computational experiments were conducted on the Univac 1108 of the University of Maryland, Computer Science Center. They correspond to the computational experiments reported in [1]. Since the basic decomposition procedure is the same as in [1], the overall elapsed time for execution of the decomposition programs (T_{LU}) and the number of elements after decomposition (M_2) are essentially the same as reported there.

The new results presented below concern the maximal in-core storage requirement (M_1), the elapsed time for execution of the decomposition routines (T_1) using a previously generated decomposition record, and the elapsed time for execution of the backsubstitution routines (T_{SO}) when the decomposed matrices are residing on auxiliary storage. These times are given below relative to the elapsed time (T_{LU}) for the execution of the decomposition programs. It should be pointed out that for N larger than 100 there is less than a three percent difference between the elapsed time for the generation of a decomposition record (SDEC0) and that for a regular decomposition with or without retaining the record (SDEC01, NDEC01).

The first group of experiments involved nonsymmetric matrix decompositions. As in [1], a special program was used to generate permuted diagonally dominant random sparse matrices $B = (b_{ij})$. Given the dimension N of B and a number M_0 of nonzero elements, the program randomly generates $M_0 - N$ distinct index pairs (i,j), $i \neq j$, $1 \leq i,j \leq N$, and the corresponding matrix elements b_{ij}. Then each diagonal element is obtained by adding a random positive number to the sum of the moduli of the off-diagonal elements in its row. Finally, the rows and columns are independently and randomly

permuted. Table 4 contains the results obtained when the decomposition programs are applied to these random matrices. The pivot selection parameter $\mu = 0.125$ was used.

Table 4

Results for Nonsymmetric Random Matrices

N	M_0	M_1	M_2	T_1/T_{LU}	T_{SO}/T_{LU}
50	200	200	375	.255	.068
50	300	300	632	.218	.041
100	400	400	811	.198	.044
100	600	1,114	2,026	.140	.016
200	800	913	2,312	.138	.020
200	1,200	3,991	6,439	.074	.005
300	1,200	1,924	4,224	.121	.012

The second group of experiments involved the decomposition of symmetric matrices obtained by the discretizing of Dirichlet's problem for Laplace's equation on the unit square with the five-point and nine-point formulas. In all cases a uniform mesh was used. The coefficients of the resulting matrices are well known and are not repeated here. Table 5 contains the results obtained with the symmetric decomposition programs.

Table 5

Symmetric Matrix Decomposition

5-point Formula	M_0	M_1	M_2	T_1/T_{LU}	T_{SO}/T_{LU}
N = 81	225	225	469	.47	.18
N = 289	833	883	2,413	.22	.06
N = 484	1,408	1,699	4,733	.17	.04
N = 625	1,825	2,328	6,566	.16	.03
9-point Formula					
N = 81	353	353	715	.35	.10
N = 289	1,345	1,673	4,296	.15	.03
N = 484	2,290	2,928	8,054	.13	.02
N = 625	2,977	4,047	11,800	.10	.01

6. Program Listings

6.1 Main Package:

```
001           SUBROUTINE SINT0(MD,RY,CY,ND)
002           DIMENSION MD(1),RY(1),CY(1),ND(1)
003           INTEGER RY,CY
004     C ****************************************
005     C * INITIALIZE SYMMETRIC STRUCTURE ARRAYS *
006     C ****************************************
007     C
008           N = MD(1)
009           MD(5) = N
010           DO 10 I=1,N
011           RY(I) = I
012           CY(I) = I
013     10    ND(I) = 1
014           RETURN
015           END
```

```
001           SUBROUTINE SINT01(MD,FD,RY,CY,A,AN,ND)
002           DIMENSION MD(1),FD(1),RY(1),CY(1),A(1),AN(1),ND(1)
003           INTEGER RY,CY
004     C ************************************************************
005     C * INITIALIZE SYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
006     C ************************************************************
007     C
008           N = MD(1)
009           MD(5) = N
010           FD(3) = 0.
011           DO 10 I=1,N
012           AN(I) = 0.
013           RY(I) = I
014           CY(I) = I
015     10    ND(I) = 1
016           RETURN
017           END
```

```
001           SUBROUTINE SINT1(MD,FD,AN)
002           DIMENSION MD(1),FD(1),AN(1)
003     C ****************************************
004     C * INITIALIZE SYMMETRIC COEFFICIENT ARRAY *
005     C ****************************************
006     C
007           N = MD(1)
008           MD(5) = N
009           FD(3) = 0.
010           MU = MD(5)
011           DO 10 I=1,N
012     10    AN(I) = 0.
013           RETURN
014           END
015     C
```

```
001           SUBROUTINE NINTU1(MD,FD,RY,CY,AN,NDR,NDC)
002           DIMENSION MD(1),FD(1),RY(1),CY(1),NDR(1),NDC(1),AN(1)
003           INTEGER RY,CY
004     C ***************************************************************
005     C * INITIALIZE NONSYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
006     C ***************************************************************
007     C
008           N = MD(1)
009           MD(5) = N
010           FD(3) = 0.
011           DO 10 I=1,N
012           AN(I) = 0.
013           RY(I) = I
014           CY(I) = I
015           NDR(I) = 0
016     10    NDC(I) = 0
017           RETURN
018           END
```

```
001           SUBROUTINE NINT1(MD,FD,AN)
002           DIMENSION MD(1),FD(1),AN(1)
003     C ********************************************
004     C * INITIALIZE NONSYMMETRIC COEFFICIENT ARRAY *
005     C ********************************************
006     C
007           N = MD(1)
008           MD(5) = N
009           FD(3) = 0.
010           DO 10 I=1,N
011     10    AN(I) = 0.
012           RETURN
013           END
```

```
001           SUBROUTINE SBLD0(I,J,MD,RY,CY,ND)
002           DIMENSION MD(1),RY(1),CY(1),ND(1)
003           INTEGER RY,CY
004     C ************************************
005     C * BUILD SYMMETRIC STRUCTURE ARRAY *
006     C ************************************
007     C
008           IF (I.EQ.J) RETURN
009           MD(5) = MD(5)+1
010           MU = MD(5)
011           ND(I) = ND(I)+1
012           ND(J) = ND(J)+1
013           IF (I.GT.J) GO TO 10
014           RY(MU) = RY(I)
015           CY(MU) = CY(J)
016           RY(I) = MU
017           CY(J) = MU
018           RETURN
019     10    RY(MU) = RY(J)
020           CY(MU) = CY(I)
021           RY(J) = MU
022           CY(I) = MU
023           RETURN
024           END
```

```
001           SUBROUTINE SBLD01(I,J,V,MD,FD,RY,CY,A,AN,ND)
002           DIMENSION MD(1),FD(1),RY(1),CY(1),A(1),AN(1),ND(1)
003           INTEGER RY,CY
004     C *****************************************************
005     C * BUILD SYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
006     C *****************************************************
007     C
008           S = V**2
009           FD(3) = AMAX1(FD(3),ABS(V))
010           AN(I) = AN(I)+S
011           IF (I.EQ.J) GO TO 20
012           MD(5) = MD(5)+1
013           M0 = MD(5)
014           A(M0) = V
015           AN(J) = AN(J)+S
016           ND(I) = ND(I)+1
017           ND(J) = ND(J)+1
018           IF (I.GT.J) GO TO 10
019           RY(M0) = RY(I)
020           CY(M0) = CY(J)
021           RY(I) = M0
022           CY(J) = M0
023           RETURN
024     10    RY(M0) = RY(J)
025           CY(M0) = CY(I)
026           RY(J) = M0
027           CY(I) = M0
028           RETURN
029     20    A(I) = V
030           RETURN
031           END
```

```
001           SUBROUTINE SBLD1(I,J,V,MD,FD,A,AN)
002           DIMENSION MD(1),FD(1),A(1),AN(1)
003     C *************************************
004     C * BUILD SYMMETRIC COEFFICIENT ARRAY *
005     C *************************************
006     C
007           S = V**2
008           FD(3) = AMAX1(FD(3),ABS(V))
009           AN(I) = AN(I)+S
010           IF (I.EQ.J) GO TO 10
011           MD(5) = MD(5)+1
012           M0 = MD(5)
013           A(M0) = V
014           AN(J) = AN(J)+S
015           RETURN
016     10    A(I) = V
017           RETURN
018           END
```

```
001           SUBROUTINE NBLD01(I,J,V,MD,FD,RY,CY,A,AN,NDR,NDC)
002           DIMENSION MD(1),FD(1),RY(1),CY(1),A(1),NDR(1),NDC(1),AN(1)
003           INTEGER RY,CY
004     C ************************************************************
005     C * BUILD NONSYMMETRIC STRUCTURE AND COEFFICIENT ARRAYS *
006     C ************************************************************
007     C
008           MD(5) = MD(5)+1
009           MU = MD(5)
010           RY(MU) = RY(I)
011           CY(MU) = CY(J)
012           RY(I) = MU
013           CY(J) = MU
014           NDR(I) = NDR(I)+1
015           NDC(J) = NDC(J)+1
016           A(MU) = V
017           AN(I) = AN(I)+V**2
018           FD(3) = AMAX1(FD(3),ABS(V))
019           RETURN
020           END
```

```
001           SUBROUTINE NBLD1(I,J,V,MD,FD,A,AN)
002           DIMENSION MD(1),FD(1),A(1),AN(1)
003     C ************************************************************
004     C * BUILD NONSYMMETRIC COEFFICIENT ARRAY *
005     C ************************************************************
006     C
007           AN(I) = AN(I)+V**2
008           MD(5) = MD(5)+1
009           MU = MD(5)
010           A(MU) = V
011           FD(3) = AMAX1(FD(3),ABS(V))
012           RETURN
013           END
```

```
001           SUBROUTINE SDEC0(MD,RY,CY,ND,IP,IE,IH)
002           DIMENSION MD(1),RY(1),CY(1),ND(1),IP(1),IE(1),IH(1)
003           INTEGER RY,CY
004     C ********************************************
005     C * GENERATE SYMMETRIC DECOMPOSITION RECORD *
006     C ********************************************
007     C
008           MM = 0
009           N = MD(1)
010           MD(4) = 0
011           MD(6) = MD(5)
012           MD(7) = MD(5)
013           MD(8) = 0
014     C INITIALIZE AUXILIARY FILE FOR WRITING
015           CALL DWI
016           DO 10 I=1,N
017     10    IP(I) = I
```

```
C
C  LOOP ON PIVOTING
C
      DO 250 I=1,N
      K = 0
      IF (I.EQ.N) GO TO 30
C
C  SELECT PIVOT BY MINIMAL DEGREE
C
      NDX = N+1
      DO 20 J=I,N
      IX = IP(J)
      IF (ND(IX).GE.NDX) GO TO 20
      NDX = ND(IX)
      IY = J
20    CONTINUE
      IF (I.EQ.IY) GO TO 30
      J = IP(IY)
      IP(IY) = IP(I)
      IP(I) = J
C
C  COLLECT THE ROW AND COLUMN OF THE PIVOT
C    ALSO DELETE THEM FROM THE STORAGE
C
30    IX = IP(I)
      IF (I.EQ.N) GO TO 110
      IY1 = 0
      IY = IX
40    IY = RY(IY)
      IF (IY.EQ.IX) GO TO 70
      IZ = IY
50    IZ = CY(IZ)
      IF (IZ.GT.N) GO TO 60
      K = K+1
      IE(K) = IY
      IH(K) = IZ
      ND(IZ) = ND(IZ)-1
60    IF (CY(IZ).NE.IY) GO TO 50
      CY(IZ) = CY(IY)
      CY(IY) = MM
      MM = IY
      GO TO 40
70    IY = CY(IY)
      IF (IY.EQ.IX) GO TO 100
      IZ = IY
      IY1 = IY
80    IZ = RY(IZ)
      IF (IZ.GT.N) GO TO 90
      K = K+1
      IE(K) = IY
      IH(K) = IZ
      ND(IZ) = ND(IZ)-1
90    IF (RY(IZ).NE.IY) GO TO 80
      RY(IZ) = RY(IY)
      GO TO 70
100   IF (IY1.EQ.0) GO TO 110
      CY(IY1) = MM
      MM = CY(IX)
C
C   MODIFICATION OF THE ROW ELEMENTS
C
C       WRITE OUT PIVOT
110   CALL DW(IX)
      CALL DW(K)
      MD(8) = MD(8)+1+K**2
      IF (K.EQ.0) GO TO 250
      DO 115 J=1,K
      CALL DW(IE(J))
115   CALL DW(IH(J))
      IF (K.EQ.1) GO TO 250
      K1 = K-1
```

```
C
C   LOOP FOR THE CROSS-POINT ELEMENTS
C
      DO 240 J=1,K1
      J1 = J+1
      IZ = IH(J)
      DO 230 JJ=J1,K
      JZ = IH(JJ)
      I1 = MINO(IZ,JZ)
      I2 = MAXO(IZ,JZ)
      L1 = RY(I1)
      L2 = CY(I2)
120   IF ((L1.EQ.I1).OR.(L2.EQ.I2)) GO TO 140
      IF (L1.EQ.L2) GO TO 220
      IF (L1.GT.L2) GO TO 130
      L2 = CY(L2)
      GO TO 120
130   L1 = RY(L1)
      GO TO 120
C   INSERTION OF A NEW NON-ZERO ELEMENT
140   ND(I1) = ND(I1)+1
      ND(I2) = ND(I2)+1
      MD(7) = MD(7)+1
      IF (MM.NE.0) GO TO 170
C   USE NEW STORAGE
      MD(6) = MD(6)+1
      IF (MD(6).LE.MD(2)) GO TO 160
      MD(4) = 0
      RETURN
160   L1 = MD(6)
      RY(L1) = RY(I1)
      CY(L1) = CY(I2)
      RY(I1) = L1
      CY(I2) = L1
      GO TO 220
C   USE AVAILABLE STORAGE
170   L1 = MM
      MM = CY(MM)
      L3 = I1
      L2 = I2
180   IF (RY(L3).LT.L1) GO TO 190
      L3 = RY(L3)
      GO TO 180
190   RY(L1) = RY(L3)
      RY(L3) = L1
200   IF (CY(L2).LT.L1) GO TO 210
      L2 = CY(L2)
      GO TO 200
210   CY(L1) = CY(L2)
      CY(L2) = L1
C   WRITE OUT CROSS-POINT ELEMENT
220   CALL DW(L1)
230   CONTINUE
240   CONTINUE
C   END OF MODIFICATION LOOP
250   CONTINUE
C
C   END OF PIVOTING LOOP
C
      CALL DWE
      RETURN
C
      END
```

```
001           SUBROUTINE SDECU1(MD,FD,RY,CY,A,AN,ND,IP,IE,IH)
002           DIMENSION MD(1),FD(1),RY(1),CY(1),ND(1),IP(1),IE(1),IH(1)
003           DIMENSION A(1),AN(1)
004           INTEGER RY,CY
005     C ***********************************************
006     C * DECOMPOSE SYMMETRIC MATRIX AND OPTIONALLY   *
007     C *      GENERATE DECOMPOSITION RECORD          *
008     C ***********************************************
009     C
010           N = MD(1)
011           MD(4) = 0
012           FD(5) = 0.
013           FD(6) = 1.
014           FD(7) = 0.
015           FD(4) = 0.
016           MM = 0
017           MD(6) = MD(5)
018           MD(7) = MD(5)
019           MD(8) = 0
020     C INITIALIZE AUXILIARY FILE FOR WRITING
021           IF (MD(3).NE.0) CALL DWI
022           DO 10 I=1,N
023           FD(7) = FD(7)+ALOG(AN(I))
024     10    IP(I) = I
025           FD(7) = 0.5*FD(7)
026           CALL SVWI
027     C
028     C LOOP ON PIVOTING
029     C
030           DO 250 I=1,N
031           K = 0
032           IF (I.EQ.N) GO TO 30
033     C
034     C SELECT PIVOT BY MINIMAL DEGREE
035     C
036           NDX = N+1
037           AX = 0.
038           DO 15 J=I,N
039           IX = IP(J)
040     15    AX = AMAX1(AX,ABS(A(IX)))
041           AX = AX*FD(2)
042           DO 20 J=I,N
043           IX = IP(J)
044           IF (ABS(A(IX)).LT.AX) GO TO 20
045           IF (ND(IX).GE.NDX) GO TO 20
046           NDX = ND(IX)
047           IY = J
048     20    CONTINUE
049           IF (I.EQ.IY) GO TO 30
050           J = IP(IY)
051           IP(IY) = IP(I)
052           IP(I) = J
053     C
054     C COLLECT THE ROW AND COLUMN OF THE PIVOT
055     C   ALSO DELETE THEM FROM THE STORAGE
056     C
057     30    IX = IP(I)
058           S = A(IX)
059           IF (ABS(S).LT.FD(1)) GO TO 300
060           IF (I.EQ.N) GO TO 110
061           IY1 = 0
062           IY = IX
```

```
063    40      IY = RY(IY)
064            IF (IY.EQ.IX) GO TO 70
065            IZ = IY
066    50      IZ = CY(IZ)
067            IF (IZ.GT.N) GO TO 60
068            K = K+1
069            IE(K) = IY
070            IH(K) = IZ
071            AN(K) = A(IY)
072            A(IY) =0.
073            ND(IZ) = ND(IZ)-1
074    60      IF (CY(IZ).NE.IY) GO TO 50
075            CY(IZ) = CY(IY)
076            CY(IY) = MM
077            MM = IY
078            GO TO 40
079    70      IY = CY(IY)
080            IF (IY.EQ.IX) GO TO 100
081            IZ = IY
082            IY1 = IY
083    80      IZ = RY(IZ)
084            IF (IZ.GT.N) GO TO 90
085            K = K+1
086            IE(K) = IY
087            IH(K) = IZ
088            AN(K) = A(IY)
089            A(IY) = 0.
090            ND(IZ) = ND(IZ)-1
091    90      IF (RY(IZ).NE.IY) GO TO 80
092            RY(IZ) = RY(IY)
093            GO TO 70
094    100     IF (IY1.EQ.0) GO TO 110
095            CY(IY1) = MM
096            MM = CY(IX)
097    C
098    C   MODIFICATION OF THE ROw ELEMENTS
099    C
100    C       WRITE OUT PIVOT
101    110     IF (MD(3).NE.0) CALL DW(IX)
102            IF (MD(3).NE.0) CALL DW(K)
103            FD(4) = AMAX1(FD(4),ABS(S))
104            FD(5) = FD(5)+ALOG(ABS(S))
105            IF (S.LT.0.) FD(6) = -FD(6)
106            CALL SVW(-IX,S)
107            MD(8) = MD(8)+1+K**2
108            IF (K.EQ.0) GO TO 250
109            DO 115 J=1,K
110            AN(J) = AN(J)/S
111            CALL SVW(IH(J),AN(J))
112            FD(4) = AMAX1(FD(4),ABS(AN(J)))
113            IF (MD(3).NE.0) CALL DW(IE(J))
114    115     IF (MD(3).NE.0) CALL DW(IH(J))
115            K1 = K-1
116    C
117    C   LOOP FOR THE CROSS-POINT ELEMENTS
118    C
119            DO 240 J=1,K
120            J1 = J+1
121            IZ = IH(J)
122            Z = AN(J)
123            A(IZ) = A(IZ)-S*Z**2
124            FD(4) = AMAX1(FD(4),ABS(A(IZ)))
125            IF (J.EQ.K) GO TO 240
```

```
126           DO 230 JJ=J1,K
127           JZ = IH(JJ)
128           I1 = MIN0(IZ,JZ)
129           I2 = MAX0(IZ,JZ)
130           L1 = RY(I1)
131           L2 = CY(I2)
132     120   IF ((L1.EQ.I1).OR.(L2.EQ.I2)) GO TO 140
133           IF (L1.EQ.L2) GO TO 220
134           IF (L1.GT.L2) GO TO 130
135           L2 = CY(L2)
136           GO TO 120
137     130   L1 = RY(L1)
138           GO TO 120
139     C  INSERTION OF A NEW NON-ZERO ELEMENT
140     140   ND(I1) = ND(I1)+1
141           ND(I2) = ND(I2)+1
142           MD(7) = MD(7)+1
143           IF (MM.NE.0) GO TO 170
144     C  USE NEW STORAGE
145           MD(6) = MD(6)+1
146           IF (MD(6).GT.MD(2)) GO TO 310
147     160   L1 = MD(6)
148           A(L1) = 0.
149           RY(L1) = RY(I1)
150           CY(L1) = CY(I2)
151           RY(I1) = L1
152           CY(I2) = L1
153           GO TO 220
154     C  USE AVAILABLE STORAGE
155     170   L1 = MM
156           MM = CY(MM)
157           L3 = I1
158           L2 = I2
159     180   IF (RY(L3).LT.L1) GO TO 190
160           L3 = RY(L3)
161           GO TO 180
162     190   RY(L1) = RY(L3)
163           RY(L3) = L1
164     200   IF (CY(L2).LT.L1) GO TO 210
165           L2 = CY(L2)
166           GO TO 200
167     210   CY(L1) = CY(L2)
168           CY(L2) = L1
169     C  WRITE OUT CROSS-POINT ELEMENT
170     220   IF (MD(3).NE.0) CALL DW(L1)
171           A(L1) = A(L1)-AN(J)*AN(JJ)*S
172     230   FD(4) = AMAX1(FD(4),ABS(A(L1)))
173     240   CONTINUE
174     C  END OF MODIFICATION LOOP
175     250   CALL SVW(-IX,S)
176     C
177     C  END OF PIVOTING LOOP
178     C
179           CALL SVWE
180           IF (MD(3).NE.0) CALL DWE
181           RETURN
182     C
183     C  SINGULAR MATRIX
184     C
185     300   MD(4) = 1
186           RETURN
187     310   MD(4) = 3
188           RETURN
189     C
190           END
```

```
      SUBROUTINE SDEC1(MD,FD,A,AN,IE)
      DIMENSION MD(1),FD(1),A(1),AN(1),IE(1)
C ******************************************************
C * DECOMPOSE SYMMETRIC MATRIX USING GENERATED RECORD *
C ******************************************************
C
      N = MD(1)
      MD(4) = 0
      FD(5) = 0.
      FD(6) = 1.
      FD(7) = 0.
      FD(4) = 0.
      DO 10 I=1,N
10    FD(7) = FD(7)+ALOG(AN(I))
      FD(7) = 0.5*FD(7)
C CLEAR STORAGE TO BE FILLED
      IF (MD(6).LE.MD(5)) GO TO 30
      MM = MD(5)+1
      M1 = MD(6)
      DO 20 I=MM,M1
20    A(I) = 0.
C INITIALIZE FOR READ-IN AND WRITE-OUT
30    CALL DRI
      CALL SVWI
C
C LOOP ON THE PIVOTS
C
      DO 90 I=1,N
      CALL DR(IX)
      CALL DR(K)
      S = A(IX)
      IF (ABS(S).LT.FD(1)) GO TO 100
      FD(4) = AMAX1(FD(4),ABS(S))
      FD(5) = FD(5)+ALOG(ABS(S))
      IF (S.LT.0.) FD(6) = -FD(6)
      IF (K.EQ.0) GO TO 70
C
C COLLECT THE ROW OF THE PIVOT
C
      DO 40 J=1,K
      CALL DR(L1)
      CALL DR(IE(J))
      AN(J) = A(L1)
      A(L1) = 0.
40    FD(4) = AMAX1(FD(4),ABS(AN(J)))
C
C    MODIFICATION OF THE ROW ELEMENTS
C
      DO 60 J=1,K
      IZ = IE(J)
      Z = AN(J)
      AN(J) = AN(J)/S
      A(IZ) = A(IZ)-Z*AN(J)
      FD(4) = AMAX1(FD(4),ABS(AN(J)))
      FD(4) = AMAX1(FD(4),ABS(A(IZ)))
      IF (J.EQ.K) GO TO 60
      J1 = J+1
C
C    MODIFICATION OF THE CROSS-POINT ELEMENTS
C
      DO 50 JJ=J1,K
      CALL DR(L1)
      A(L1) = A(L1)-AN(J)*AN(JJ)
50    FD(4) = AMAX1(FD(4),ABS(A(L1)))
60    CONTINUE
```

```
      C
      C WRITE OUT PIVOT AND ITS ROW
      C
   70     CALL SVW(-IX,S)
          IF (K.EQ.0) GO TO 90
          DO 80 J=1,K
   80     CALL SVW(IE(J),AN(J))
   90     CALL SVW(-IX,S)
      C
      C END OF PIVOTING LOOP
      C
          CALL SVWE
          RETURN
      C
      C SINGULAR MATRIX
      C
  100     MD(4) = 3
          RETURN
      C
          END
```

```
          SUBROUTINE NDEC01(MD,FD,RY,CY,A,AN,NDR,NDC,IPR,IPC,IE,IH,NG1,NG2)
          DIMENSION MD(1),FD(1),RY(1),CY(1),A(1),AN(1),NDR(1),NDC(1)
          DIMENSION IPR(1),IPC(1),IE(1),IH(1),NG1(1),NG2(1)
          INTEGER RY,CY
      C ***************************************************
      C * DECOMPOSE NONSYMMETRIC MATRIX AND OPTIONALLY    *
      C *       GENERATE DECOMPOSITION RECORD             *
      C ***************************************************
      C
          MM = 0
          N = MD(1)
          MD(4) = 0
          MD(6) = MD(5)
          MD(7) = MD(5)
          MD(8) = 0
          IF (MD(3).NE.0) CALL DWI
          FD(5) = 0.
          FD(6) = 1.
          FD(7) = 0.
          FD(4) = 0.
          DO 10 I=1,N
          FD(7) = FD(7)+ALOG(AN(I))
          IPR(I) = I
   10     IPC(I) = I
          FD(7) = 0.5*FD(7)
          CALL NVWI
          CALL NVWB(-1,-1)
      C
      C LOOP ON PIVOTING
      C
          DO 290 I=1,N
      C
      C  PIVOT SELECTION BY SEPARATE PIVOTING ROUTINE
      C
          CALL PVT01(I,N,IX,KR,KC,FD(2),RY,CY,A,IPR,IPC,NDR,NDC,IE,IH)
          IF (KR.EQ.0) GO TO 310
      C  CHECK PIVOT VALUE AND UPDATE DETERMINANT
          S = A(IX)
          IF (ABS(S).LE.FD(1)) GO TO 300
          FD(5) = FD(5)+ALOG(ABS(S))
          K = (KR+KC)/2
          K = KR+KC-2*K
          IF (K.NE.0) FD(6) = -FD(6)
          FD(4) = AMAX1(FD(4),ABS(S))
```

```
045    C
046    C  COLLECT THE ROW AND COLUMN OF THE PIVOT
047    C  ALSO FREE THEIR STORAGE LOCATIONS
048    C
049    C  THE ROW
050           K1 = 0
051           J = KR
052    20     J = RY(J)
053           IF (J.LE.N) GO TO 40
054           IF (J.EQ.IX) GO TO 20
055           K1 = K1+1
056           IE(K1) = J
057           AN(K1) = A(J)
058           J1 = J
059    30     J1 = CY(J1)
060           IF (J1.LE.N) NG1(K1) = J1
061           IF (J1.LE.N) NDC(J1) = NDC(J1)-1
062           IF (CY(J1).NE.J) GO TO 30
063           CY(J1) = CY(J)
064           CY(J) = MM
065           MM = J
066           GO TO 20
067    C  THE COLUMN
068    40     K2 = 0
069           J = KC
070           K3 = 0
071    50     J = CY(J)
072           IF (J.LE.N) GO TO 70
073           K3 = J
074           IF (J.EQ.IX) GO TO 50
075           K2 = K2+1
076           IH(K2) = J
077           A(K2) = A(J)/S
078           FD(4) = AMAX1(FD(4),ABS(A(K2)))
079           J1 = J
080    60     J1 = RY(J1)
081           IF (J1.LE.N) NG2(K2) = J1
082           IF (J1.LE.N) NDR(J1) = NDR(J1)-1
083           IF (RY(J1).NE.J) GO TO 60
084           RY(J1) = RY(J)
085           GO TO 50
086    70     CY(K3) = MM
087           MM = CY(KC)
088    C
089    C  WRITE OUT THE PIVOT, ITS ROW AND COLUMN
090    C  AS PART OF THE DECOMPOSITION RECORD
091    C
092           IF (MD(3).EQ.0) GO TO 105
093           CALL DW(IX)
094           CALL DW(KC)
095           CALL DW(KR)
096           CALL DW(K2)
097           MD(8) = MD(8)+(K1+1)*(K2+1)
098           IF (K2.EQ.0) GO TO 90
099           DO 80 J=1,K2
100           CALL DW(IH(J))
101    80     CALL DW(NG2(J))
102    90     CALL DW(K1)
103           IF (K1.EQ.0) GO TO 105
104           DO 100 J=1,K1
105           CALL DW(IE(J))
106    100    CALL DW(NG1(J))
107    105    IF ((K1.EQ.0).OR.(K2.EQ.0)) GO TO 230
108    C
109    C  LOOP TO MODIFY THE INTERSECTING ELEMENTS BETWEEN THE ROW
110    C           AND COLUMN
111    C
```

```
112           DO 220 J=1,K2
113           IY = NG2(J)
114           DO 220 K=1,K1
115           K3 = RY(IY)
116           JY = NG1(K)
117           K4 = CY(JY)
118     C  SEARCH FOR ELEMENT IY,JY
119     110   IF (K3.EQ.K4) GO TO 210
120           IF (K3.LT.K4) GO TO 120
121           K3 = RY(K3)
122           IF (K3.LE.N) GO TO 130
123           GO TO 110
124     120   K4 = CY(K4)
125           IF (K4.GT.N) GO TO 110
126     C  IT DOES NOT EXIST
127     130   NDR(IY) = NDR(IY)+1
128           NDC(JY) = NDC(JY)+1
129           MD(7) = MD(7)+1
130           IF (MM.NE.0) GO TO 160
131     C    GET NEW LOCATION FOR THE NEW ELEMENT
132           MD(6) = MD(6)+1
133           IF (MD(6).LE.MD(2)) GO TO 150
134           MD(4) = 1
135           RETURN
136     150   K3 = MD(6)
137           RY(K3) = RY(IY)
138           CY(K3) = CY(JY)
139           RY(IY) = K3
140           CY(JY) = K3
141           A(K3) = 0.
142           GO TO 210
143     C    OLD LOCATION AVAILABLE FOR THE NEW ELEMENT
144     160   K3 = MM
145           A(K3) = 0.
146           MM = CY(MM)
147           K4 = IY
148     170   IF (RY(K4).LT.K3) GO TO 180
149           K4 = RY(K4)
150           GO TO 170
151     180   RY(K3) = RY(K4)
152           RY(K4) = K3
153     190   IF (CY(JY).LT.K3) GO TO 200
154           JY = CY(JY)
155           GO TO 190
156     200   CY(K3) = CY(JY)
157           CY(JY) = K3
158     C    MODIFY ELEMENT
159     210   IF (MD(3).NE.0) CALL DW(K3)
160           A(K3) = A(K3)-AN(K)*A(J)
161           FD(4) = AMAX1(FD(4),ABS(A(K3)))
162     220   CONTINUE
163     C  END OF MODIFICATION LOOP
164     C
165     C WRITE OUT PIVOT ROW AND COLUMN
166     C
167     230   CALL NVWF(-KC,-KR)
168           IF (K2.EQ.0) GO TO 250
169           DO 240 J=1,K2
170     240   CALL NVWF(NG2(J),A(J))
171     250   IF (K1.EQ.0) GO TO 270
172           DO 260 J=1,K1
173     260   CALL NVWB(NG1(J),AN(J))
174     270   CALL NVWB(-KC,S)
175     290   CONTINUE
176     C  END OF PIVOTING LOOP
177     C
178     C END FILES
```

```
C
      IF (MD(3).NE.0) CALL DWE
      CALL NVWE
      RETURN
C
C SINGULAR MATRIX
C
300   MD(4) = 3
      RETURN
310   MD(4) = 2
      RETURN
C
      END
```

```
      SUBROUTINE PVT01(I,N,IX,KR,KC,F,RY,CY,A,IPR,IPC,NDR,NDC,IE,IH)
      DIMENSION RY(1),CY(1),IPR(1),IPC(1),NDR(1),NDC(1),IE(1),IH(1)
      DIMENSION A(1)
      INTEGER RY,CY
C ******************************************************
C * MINIMAL DEGREE PIVOTING FOR NON-SYMMETRIC MATRIX *
C ******************************************************
C    THE ROUTINE SELECTS PIVOT BY MINIMAL DEGREE, IT IS
C USED BY THE ROUTINE DEC01.
C
C
      IF (I.NE.N) GO TO 30
      KR = IPR(N)
      KC = IPC(N)
      IX = RY(KR)
      IF (IX.NE.KR) RETURN
10    KR = 0
      RETURN
30    NI = N+1-I
C
C SORT AVAILABLE ROWS BY DEGREE
C
      DO 40 J=1,NI
40    IE(J) = 0
      DO 50 J=I,N
      K1 = IPR(J)
      IH(J) = K1
      K2 = NDR(K1)
      IF (K2.LE.0) GO TO 10
50    IE(K2) = IE(K2)+1
      DO 60 J=2,NI
60    IE(J) = IE(J)+IE(J-1)
      DO 70 J=I,N
      K1 = IH(J)
      K2 = NDR(K1)
      K3 = IE(K2)+I-1
      IE(K2) = IE(K2)-1
70    IPR(K3) = K1
C
C SORT AVAILABLE COLUMNS BY DEGREE
C
      DO 80 J=1,NI
80    IE(J) = 0
      DO 90 J=I,N
      K1 = IPC(J)
      IH(J) = K1
      K2 = NDC(K1)
      IF (K2.LE.0) GO TO 10
90    IE(K2) = IE(K2)+1
```

```
      DO 100 J=2,NI
100   IE(J) = IE(J)+IE(J-1)
      DO 110 J=1,N
      K1 = IH(J)
      K2 = NDC(K1)
      K3 = IE(K2)+I-1
      IE(K2) = IE(K2)-1
110   IPC(K3) = K1
C
C INITIALIZE FOR MINIMAL DEGREE SEARCH
C
      IX = 0
      IDX = N**2
      JR = I
      JC = I
      JRP = IPR(JR)
      JCP = IPC(JC)
C
C TEST FOR TERMINATION OF SEARCH
C
120   NDX = (NDR(JRP)-1)*(NDC(JCP)-1)
      IF (NDX.GE.IDX) GO TO 240
      IF (NDC(JCP).GT.NDR(JRP)) GO TO 180
C
C SEARCH IN THE COLUMN
C
      J = JCP
      AM = 0.
130   J = CY(J)
      IF (J.EQ.JCP) GO TO 140
      AM = AMAX1(AM,ABS(A(J)))
      GO TO 130
140   AM = F*AM
150   J = CY(J)
      IF (J.EQ.JCP) GO TO 170
      IF (ABS(A(J)).LT.AM) GO TO 150
      K1 = J
160   K1 = RY(K1)
      IF (K1.GT.N) GO TO 160
      K2 = (NDR(K1)-1)*(NDC(JCP)-1)
      IF (K2.GE.IDX) GO TO 150
      IX = J
      KR = K1
      KC = JCP
      IDX = K2
      GO TO 150
170   JC = JC+1
      IF (JC.GT.N) GO TO 240
      JCP = IPC(JC)
      GO TO 120
C
C SEARCH IN THE ROW
C
180   J = JRP
      AM = 0.
190   J = RY(J)
      IF (J.EQ.JRP) GO TO 200
      AM = AMAX1(AM,ABS(A(J)))
      GO TO 190
200   AM = F*AM
210   J = RY(J)
      IF (J.EQ.JRP) GO TO 230
      IF (ABS(A(J)).LT.AM) GO TO 210
      K1 = J
220   K1 = CY(K1)
      IF (K1.GT.N) GO TO 220
```

```
116           K2 = (NDR(JRP)-1)*(NDC(K1)-1)
117           IF (K2.GE.IDX) GO TO 210
118           IX = J
119           KR = JRP
120           KC = K1
121           IDX = K2
122           GO TO 210
123     230   JR = JR+1
124           IF (JR.GT.N) GO TO 240
125           JRP = IPR(JR)
126           GO TO 120
127     C
128     C SEARCH FINISHED, REMOVE KR,KC FROM AVAILABLE
129     C PIVOT ROWS AND COLUMNS
130     240   IF (IX.EQ.0) GO TO 10
131           DO 250 J=I,N
132           IF (IPR(J).NE.KR) GO TO 250
133           IF (J.EQ.I) GO TO 260
134           IPR(J) = IPR(I)
135           IPR(I) = KR
136           GO TO 260
137     250   CONTINUE
138     260   DO 270 J=I,N
139           IF (IPC(J).NE.KC) GO TO 270
140           IPC(J) = IPC(I)
141           IPC(I) = KC
142           GO TO 280
143     270   CONTINUE
144     C
145     280   RETURN
146           END
```

```
001           SUBROUTINE NDEC1(MD,FD,A,AN,IE,IH)
002           DIMENSION MD(1),FD(1),A(1),AN(1),IE(1),IH(1)
003     C *************************************************************
004     C * DECOMPOSE NONSYMMETRIC MATRIX USING GENERATED RECORD *
005     C *************************************************************
006     C
007           N = MD(1)
008           MD(4) = 0
009           FD(5) = 0.
010           FD(6) = 1.
011           FD(7) = 0.
012           FD(4) = 0.
013           DO 10 I=1,N
014     10    FD(7) = FD(7)+ALOG(AN(I))
015           FD(7) = 0.5*FD(7)
016     C CLEAR EXTRA STORAGE
017           IF (MD(6).LE.MD(5)) GO TO 30
018           MM = MD(5)+1
019           M1 = MD(6)
020           DO 20 I=MM,M1
021     20    A(I) = 0.
```

```
022     C INITIALIZE FILES
023     30      CALL DRI
024             CALL NVWI
025             CALL NVWB(-1,-1)
026     C
027     C LOOP ON THE PIVOTS
028     C
029             DO 130 I=1,N
030     C GET PIVOT ADDRESS AND CHECK PIVOT MAGNITUDE
031             CALL DR(IX)
032             S = A(IX)
033             IF (ABS(S).LE.FD(1)) GO TO 150
034             FD(5) = FD(5)+ALOG(ABS(S))
035             IF (S.LT.0.) FD(6) = -FD(6)
036             FD(4) = AMAX1(FD(4),ABS(S))
037             A(IX) = 0.
038             CALL DR(KC)
039             CALL DR(KR)
040     C GET THE COLUMN ELEMENTS
041             CALL DR(K2)
042             IF (K2.LE.0) GO TO 50
043             DO 40 J=1,K2
044             CALL DR(K3)
045             CALL DR(IE(J))
046             A(J) = A(K3)/S
047             FD(4) = AMAX1(FD(4),ABS(A(J)))
048     40      A(K3) = 0.
049     C GET THE ROW ELEMENTS
050     50      CALL DR(K1)
051             IF (K1.LE.0) GO TO 80
052             DO 60 J=1,K1
053             CALL DR(K3)
054             CALL DR(IH(J))
055             AN(J) = A(K3)
056             FD(4) = AMAX1(FD(4),ABS(AN(J)))
057     60      A(K3) = 0.
058     C MODIFY CROSS-POINT ELEMENTS
059             IF (K2.LE.0) GO TO 80
060             DO 70 J=1,K2
061             DO 70 JJ=1,K1
062             CALL DR(K3)
063             A(K3) = A(K3)-A(J)*AN(JJ)
064     70      FD(4) = AMAX1(FD(4),ABS(A(K3)))
065     C WRITE OUT PIVOT ROW AND COLUMN
066     80      CALL NVWF(-KC,-KR)
067             IF (K2.EQ.0) GO TO 100
068             DO 90 J=1,K2
069     90      CALL NVWF(IE(J),A(J))
070     100     IF (K1.EQ.0) GO TO 120
071             DO 110 J=1,K1
072     110     CALL NVWB(IH(J),AN(J))
073     120     CALL NVWB(-KC,S)
074     130     CONTINUE
075     C
076             CALL NVWE
077             RETURN
078     C
079     150     MD(4) = 3
080             RETURN
081     C
082             END
```

```
001           SUBROUTINE SSLV(MD,X,Y)
002           DIMENSION MD(1),X(1),Y(1)
003     C *****************************************************
004     C * BACKSUBSTITUTION FOR SYMMETRIC DECOMPOSED MATRIX *
005     C *****************************************************
006     C
007           N = MD(1)
008           DO 10 I=1,N
009     10    Y(I) = X(I)
010           CALL SVRI
011           I = 0
012     C FORWARD BACKSUBSTITUTION
013     20    CALL SVRF(L2,Z)
014           I = I+1
015           L2 = -L2
016           S = Y(L2)
017     30    CALL SVRF(L2,Z)
018           IF (L2.LT.0) GO TO 40
019           Y(L2) = Y(L2)-Z*S
020           GO TO 30
021     40    IF (I.LT.N) GO TO 20
022     C BACKWARD BACKSUBSTITUTION
023     50    CALL SVRB(L2,Z)
024           I = I-1
025           J = -L2
026           Y(J) = Y(J)/Z
027     60    CALL SVRB(L2,Z)
028           IF (L2.LT.0) GO TO 70
029           Y(J) = Y(J)-Z*Y(L2)
030           GO TO 60
031     70    IF (I.GT.0) GO TO 50
032           RETURN
033     C
034           END
```

```
001           SUBROUTINE NSLV(MD,X,Y,AN)
002           DIMENSION MD(1),X(1),Y(1),AN(1)
003     C *********************************************************
004     C * BACKSUBSTITUTION FOR NONSYMMETRIC DECOMPOSED MATRIX *
005     C *********************************************************
006     C
007           EQUIVALENCE (KS,S)
008           N = MD(1)
009     C
010     C     SAVE RIGHT SIDE
011     C
012           DO 10 I=1,N
013     10    AN(I) = X(I)
014     C
015     C     INITIALIZE FILES
016     C
017           CALL NVRI
018           I = 0
019           J = 0
020     C
021     C     SOLVE LOWER TRIANGULAR SYSTEM
```

```
022     C
023     20    CALL NVRF(K4,S)
024           IF (K4.GT.0) GO TO 30
025           I = I+1
026           K4 = -K4
027           K3 = -KS
028           Y(K4) = AN(K3)
029           IF (I.GE.N) GO TO 40
030           D1 = AN(K3)
031           GO TO 20
032     30    AN(K4) = AN(K4)-D1*S
033           GO TO 20
034     C
035     C   SOLVE UPPER TRIANGULAR SYSTEM
036     C
037     40    CALL NVRB(K4,S)
038           IF (K4.GT.0) GO TO 50
039           J = J+1
040           IF (J.NE.1) Y(IX) = Y(IX)/D1
041           IF (J.GT.N) GO TO 60
042           IX = -K4
043           D1 = S
044           GO TO 40
045     50    Y(IX) = Y(IX)-S*Y(K4)
046           GO TO 40
047     C
048     60    RETURN
049     C
050           END
```

6.2 I/O Programs:

```
001    C ********************************************
002    C * I/O ROUTINE FOR DECOMPOSITION PROCESSES *
003    C ********************************************
004    C
005          SUBROUTINE DWI
006    C
007    C THE DECOMPOSITION PROCESS GENERATES AN ARRAY OF
008    C POSITIVE INTEGERS. THIS PROGRAM PROVIDES A BUFFERED
009    C INPUT/OUTPUT USING FILE 10. THE ENTRIES ARE AS FOLLOWS:
010    C
011    C    DWI - INITIALIZE FOR WRITE
012    C    DW(K) - WRITE K AS NEXT ENTRY
013    C    DWE - TERMINATE WRITING
014    C    DRI - INITIALIZE FOR READ
015    C    DR(K) - READ NEXT ENTRY K
016    C
017    C IX IS THE BUFFER SIZE
018          PARAMETER IX = 250
019          DIMENSION IB(IX)
020    C
021    C
022    C ***************************
023    C * INITIALIZE FOR WRITING *
024    C ***************************
025    C
026          J = 1
027          REWIND 10
028          RETURN
029    C
030          ENTRY DW(K)
031    C ************************
032    C * WRITE NEXT ENTRY K *
033    C ************************
034    C
035          IB(J) = K
036          J = J+1
037          IF (J.LE.IX) RETURN
038          WRITE (10) IB
039          J = 1
040          RETURN
041    C
042          ENTRY DWE
043    C ***********************
044    C * TERMINATE WRITING *
045    C ***********************
046    C
047          IF (J.NE.1) WRITE (10) IB
048          RETURN
049    C
050          ENTRY DRI
051    C ************************
052    C * INITIALIZE READ-IN *
053    C ************************
054    C
055          REWIND 10
056          J = IX
057          RETURN
058    C
059          ENTRY DR(K)
060    C ***********************
061    C * READ NEXT ENTRY K *
062    C ***********************
063    C
064          J = J+1
065          IF (J.LE.IX) GO TO 10
066          READ (10) IB
067          J = 1
068    10    K = IB(J)
069          RETURN
070    C
071          END
```

```
001     C ***************************************************
002     C * I/O ROUTINE FOR DECOMPOSED SYMMETRIC MATRIX *
003     C ***************************************************
004     C
005           SUBROUTINE SVWI
006     C THE DECOMPOSED MATRIX IS PLACED IN FILE 11 AS A RANDOM
007     C ACCESS FILE. IT CONSISTS OF A DOUBLE ARRAY WHICH IS
008     C BUFFERED, ALTHOUGH THE FIRST PART OF THE ARRAY
009     C IS AN INTEGER (SIGNED) ARRAY, THIS ROUTINE DOES NOT
010     C PACK IT. THE ROUTINE HAS THE FOLLOWING ENTRIES:
011     C
012     C     SVWI - INITIALIZE FOR WRITE
013     C     SVW(A1,A2) - WRITE A1,A2 AS NEXT ENTRY
014     C     SVWE - TERMINATE WRITE
015     C     SVRI - INITIALIZE FOR READ
016     C     SVRF(A1,A2) - READ NEXT ENTRY A1,A2
017     C     SVRB(A1,A2) - READ PREVIOUS ENTRY A1,A2
018     C
019     C THE ROUTINE ASSUMES THAT THE WRITTEN ARRAY IS READ ONCE
020     C FORWARD THEN READ BACKWARD.
021     C
022           PARAMETER NX = 100
023           PARAMETER MX = 280
024           PARAMETER MXX = 2*MX
025     C NX IS THE MAXIMUM NUMBER OF RECORDS
026     C MXX IS THE LENGTH OF THE RECORDS
027           DIMENSION B(2,MX)
028     C
029     C ***********************
030     C * INITIALIZE WRITING *
031     C ***********************
032     C
033           N = 0
034           J = 1
035           DEFINE FILE 11(NX,MXX,U,IX)
036           IX = IX
037           RETURN
038     C
039           ENTRY SVW(A1,A2)
040     C ****************************
041     C * WRITE NEXT ENTRY A1,A2 *
042     C ****************************
043     C
044           B(1,J) = A1
045           B(2,J) = A2
046           J = J+1
047           IF (J.LE.MX) RETURN
048           N = N+1
049           WRITE (11'N) B
050           J = 1
051           RETURN
052     C
053           ENTRY SVWE
054     C ***********************
055     C * TERMINATE WRITING *
056     C ***********************
057     C
058           IF (J.EQ.1) RETURN
059           N = N+1
060           WRITE (11'N) B
061           RETURN
062     C
063           ENTRY SVRI
064     C ***********************
065     C * INITIALIZE READ-IN *
066     C ***********************
067     C
068           M = 0
069           J = MX
070           RETURN
```

```
071     C
072           ENTRY SVRF(A1,A2)
073     C ************************
074     C * READ NEXT ENTRY A1,A2 *
075     C ************************
076     C
077           J = J+1
078           IF (J.LE.MX) GO TO 10
079           M = M+1
080           READ (11'M) B
081           J = 1
082     10    A1 = B(1,J)
083           A2 = B(2,J)
084           RETURN
085     C
086           ENTRY SVRB(A1,A2)
087     C ****************************
088     C * READ PREVIOUS ENTRY A1,A2 *
089     C ****************************
090     C
091           IF (J.GT.0) GO TO 20
092           M = M-1
093           READ (11'M) B
094           J = MX
095     20    A1 = B(1,J)
096           A2 = B(2,J)
097           J = J-1
098           RETURN
099     C
100           END
```

```
001    C ******************************************************
002    C * I/O ROUTINE FOR NONSYMMETRIC DECOMPOSED MATRIX *
003    C ******************************************************
004          SUBROUTINE NVWI
005    C THE LOWER AND UPPER TRIANGULAR MATRICES OF THE
006    C DECOMPOSED NONSYMMETRIC MATRIX ARE CONTAINED IN
007    C FILE 12 AND FILE 13, RESPECTIVELY, AS RANDOM
008    C ACCESS FILES. THEY ARE IN THE FORM OF BUFFERED
009    C DOUBLE ARRAYS. THE ENTRIES ARE AS FOLLOWS,
010    C
011    C     NVWI - INITIALIZE FOR WRITE
012    C     NVWF(A1,A2) - WRITE A1,A2 AS NEXT ENTRY ON FILE 12
013    C     NVWB(A1,A2) - WRITE A1,A2 AS NEXT ENTRY ON FILE 13
014    C     NVWE - TERMINATE WRITING
015    C     NVRI - INITIALIZE FOR READ
016    C     NVRF(A1,A2) - READ NEXT ENTRY FROM FILE 12
017    C     NVRB(A1,A2) - READ PREVIOUS ENTRY FROM FILE 13
018    C
019    C FILE 12 IS READ FORWARD, FILE 13 BACKWARD,
020    C
021          PARAMETER NX = 100
022          PARAMETER MX = 280
023          PARAMETER MXX = 2*MX
024    C NX IS THE MAXIMUM NUMBER OF RECORDS,
025    C MXX IS THE RECORD SIZE
026          DIMENSION B12(2,MX),B13(2,MX)
027    C
028    C ***********************
029    C * INITIALIZE WRITING *
030    C ***********************
031    C
032          N12 = 0
033          N13 = 0
034          J12 = 1
035          J13 = 1
036          DEFINE FILE 12(NX,MXX,U,IX12)
037          IX12 = IX12
038          DEFINE FILE 13(NX,MXX,U,IX13)
039          IX13 = IX13
040          RETURN
041    C
042          ENTRY NVWF(A1,A2)
043    C ****************************
044    C * WRITE A1,A2 ON FILE 12 *
045    C ****************************
046    C
047          B12(1,J12) = A1
048          B12(2,J12) = A2
049          J12 = J12+1
050          IF (J12.LE.MX) RETURN
051          N12 = N12+1
052          J12 = 1
053          WRITE (12'N12) B12
054          RETURN
055    C
056          ENTRY NVWB(A1,A2)
057    C ****************************
058    C * WRITE A1,A2 ON FILE 13 *
059    C ****************************
060    C
061          B13(1,J13) = A1
062          B13(2,J13) = A2
063          J13 = J13+1
064          IF (J13.LE.MX) RETURN
065          N13 = N13+1
066          J13 = 1
067          WRITE (13'N13) B13
068          RETURN
069    C
```

```
070           ENTRY NVWE
071     C *********************
072     C * TERMINATE WRITING *
073     C *********************
074     C
075           IF (J12.EQ.1) GO TO 10
076           N12 = N12+1
077           WRITE (12'N12) B12
078     10    JJ = MX
079           IF (J13.EQ.1) RETURN
080           N13 = N13+1
081           WRITE (13'N13) B13
082           JJ = J13-1
083           RETURN
084     C
085           ENTRY NVRI
086     C ************************
087     C * INITIALIZE READ-IN *
088     C ************************
089     C
090           N2 = 0
091           N3 = N13+1
092           J2 = MX
093           J3 = 0
094           RETURN
095     C
096           ENTRY NVRF(A1,A2)
097     C ******************************
098     C * READ A1,A2 FROM FILE 12 *
099     C ******************************
100     C
101           J2 = J2+1
102           IF (J2.LE.MX) GO TO 20
103           N2 = N2+1
104           READ (12'N2) B12
105           J2 = 1
106     20    A1 = B12(1,J2)
107           A2 = B12(2,J2)
108           RETURN
109     C
110           ENTRY NVRB(A1,A2)
111     C ********************************************
112     C * READ PREVIOUS ENTRY A1,A2 FROM FILE 13 *
113     C ********************************************
114     C
115           IF (J3.GT.0) GO TO 30
116           N3 = N3-1
117           READ (13'N3) B13
118           J3 = MX
119           IF (N3.EQ.N13) J3 = JJ
120     30    A1 = B13(1,J3)
121           A2 = B13(2,J3)
122           J3 = J3-1
123           RETURN
124     C
125           END
```

References

[1] Rheinboldt, W.C., and Mesztenyi, C.K., "Programs for the solution of large sparse matrix problems based on the arc-graph structure", University of Maryland, Computer Science Technical Report TR-262, 1973.

[2] Rheinboldt, W.C., and Mesztenyi, C.K., "Arc graphs and their possible application to sparse matrix problems", BIT 14, 1974, 227-239.

[3] Gustavson, F.G., Liniger, W.M., and Willoughby, R.A., "Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations", in Sparse Matrix Proceedings, (R.A. Willoughby, Ed.), IBM Research, Yorktown Heights, N.Y., 1968, 1-9.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: Technical Report TR-394	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Further Programs for the Solution of Large Sparse Systems of Linear Equations	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): C. K. Mesztenyi, W. C. Rheinboldt	8. CONTRACT OR GRANT NUMBER(s): N00014-67-A-0239-0021, GJ-35568X
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science Center, University of Maryland, College Park, Maryland 20742	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Mathematics Branch, Office of Naval Research, Arlington, VA 22217	12. REPORT DATE: August 1975 13. NUMBER OF PAGES: 50
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)

sparse linear systems
triangular systems
FORTRAN programs
fill-in
decomposition records

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

A package of FORTRAN subroutines is presented for the solution of non-symmetric or symmetric sparse linear systems by triangular decomposition. Two principal aims are (1) to handle matrices which originally fit into primary core storage but do so no longer after decomposition, and (2) to solve a sequence of linear systems all of which have the same sparsity structure by generating--in secondary storage--a record of the decomposition process in the form of an integer array. Some experimental results using the package are included.