

AD-A125 766

A VECTORIZED GENERAL SPARSITY SOLVER(U) MICHIGAN UNIV
ANN ARBOR SYSTEMS ENGINEERING LAB D A CALAHAN
01 OCT 82 SEL-168 AFOSR-TR-83-0076 AFOSR-80-0158

1/1

UNCLASSIFIED

F/G 12/1

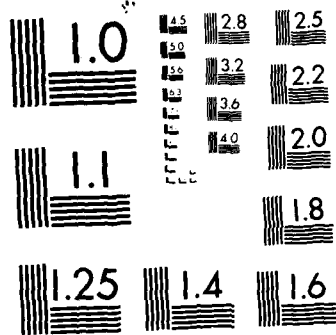
NL

END

FILED

101

010



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

3

AD A 125766

A Vectorized General Sparsity Solver

D.A. CALAHAN

October 1, 1982

Sponsored by
Directorate of Mathematical and Information Sciences,
Air Force Office of Scientific Research,
under Grant No: ~~80-0158~~ 80-0158

Approved for public release;
distribution unlimited.



Systems Engineering Laboratory
The University of Michigan

DTIC FILE COPY

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0076	2. GOVT ACCESSION NO. AD-412576	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A VECTORIZED GENERAL SPARSITY SOLVER		5. TYPE OF REPORT & PERIOD COVERED Interim
7. AUTHOR(s) D. A. Calahan		6. PERFORMING ORG. REPORT NUMBER 168
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Michigan Dept. of Elec. & Computer Engring. Ann Arbor, MI, 48109		8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0158
11. CONTROLLING OFFICE NAME AND ADDRESS NM Air Force Office of Scientific Research Bolling AFB, Washington DC 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2344/A3
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE October 1, 1982
		13. NUMBER OF PAGES 25
		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 (for 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 matrices Parallel processing Vector processing Linear algebra		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Description and use of a Fortran general sparse solver, modified to run efficiently on a vector processor, is given. CRAY-1 performance in the analysis of 2D grids is presented.		

A Vectorized
General Sparsity
Solver

D. A. Calahan

Systems Engineering Laboratory

University of Michigan

Ann Arbor, Michigan 48109

October 1, 1982

SEL Report #168

Sponsored by

Directorate of Mathematical and Information Sciences

Air Force Office of Scientific Research,

under Grant 80-0158

Division

ABSTRACT

Description and use of a Fortran general sparse solver, modified to operate efficiently on a vector processor, is given. CRAY-1 performance of the solver in analysis of 2-D grids is presented.



A

I. INTRODUCTION

A. General Sparse Solvers

The performance of sparse equation solvers on vector processors is highly dependent on the machine instruction set and associated timings. The availability of indirect addressing instructions (gather/scatter) is a particularly important issue. When the processor has a hierarchical or other flexible memory organization, such as the CRAY-1, control of the data flow by assembly language coding may also become important.

Algorithmically, another set of considerations is introduced by the possibility of achieving high execution rates via equation re-ordering and recognition of favorable matrix structures. For example, blocking of matrices can reduce data traffic [7]; recognition of global sparsity patterns can also yield a vector solution [9].

B. Generic Vectorized Solver

In view of these opportunities for specialized algorithms, the value of a generic sparsity algorithm in a common high-level language for vector processors is in some question. For example, it is known that traditional general sparsity codes [2] execute poorly from a high level language on current vector processors, at rates less than 1/200 of maximum processor performance [10].

It is the viewpoint of this report that a generic vectorized solver has value in establishing a baseline performance against which the performance of more specialized solvers can be judged.

The following properties are proposed for such a solver.

- (a) Consistent with general solvers for scalar processors, the solver should accept column- or row-ordered symbolic matrix descriptions, plus permutation vectors describing row and column re-orderings.
- (b) The inner loop of the numeric solution should recognize opportunities to

exploit vector hardware.

(c) The solution may be of a two-phase symbolic/numeric nature, where a symbolic phase preprocesses the matrix structure and provides descriptors to a solution phase, for possible repeated numerical solutions with the same matrix structure [2].

II. The LU Map Approach

A. Introduction

The algorithm to be discussed is a variation of the "looped index" or "LU map" method of Chuang [1] and Gustavson [2] to vector processors. It was first discussed in detail in 1977 in [3] and its application to electrical circuit analysis given in [4]. The following discussion is taken from [3].

B. Symbolic Vectorization

1. Introduction

Given a matrix A and right hand side B, it is proposed to perform a triangular factorization in the form

$$A=LU \quad (1)$$

where L and U are lower and upper triangular factors with elements l_{ij} and u_{ij} , respectively. Column ordered reduction is performed, with $l_{ii} = 1$. The forward and backward substitution has the form

$$LY=B \quad (2)$$

$$UX=Y \quad (3)$$

where X is the solution vector. (For descriptive clarity, pivoting is assumed down the main diagonal).

2. Scalar Model

In the work of Gustavson, the purpose of the symbolic phase was to determine the fill characteristics of A, i.e., the exact structure of L and U. This is a costly process that need be performed only once for a given matrix structure. To acquaint the reader with this approach, an example using Gustavson's "scalar" map is shown in Table 1. Special note should be taken of

- (1) the fill positions detected by the symbolic phase in the generation of the LU map;
- (2) the use of map indices in the numeric solution to extract information from the numeric arrays A, L, and U;
- (3) the use of an expanded current column (X array), requiring zeroing, expansion, and contraction in the loading and storing process;
- (4) the opportunities for the use of vector operations in the numeric solution, as evidenced by the indexed array operations marked "vector".

The following two sections are intended to give insight into the symbolic map generation by discussion of a proposed vectorized data structure and symbolic operations on it during the factorization process.

3. Vectorized List Data Structures

Consider a column of a sparse matrix having the non-zero row positions shown in Figure 1 (before fill). This structure would be described in a conventional ordered list as

$$31, 32, \dots, 36, 39, 42, 43, \dots, 47 \quad (4)$$

Such a list enumerating all row positions will be termed scalar storage. Clearly, the list can be shortened by identifying sets of contiguous positions (vectors) and retaining only the first and last row numbers, viz.,

$$\begin{array}{rcl}
 \begin{array}{ccccc}
 3 & 0 & 0 & 0 & 2 \\
 0 & 4 & 2 & 1 & 0 \\
 0 & 2 & 6 & 0 & 5 \\
 0 & 1 & 0 & 3 & 1 \\
 2 & 0 & 3 & 1 & 5
 \end{array}
 &
 \text{LU} = &
 \begin{array}{ccccc}
 3 & 0 & 0 & | & 0 & | & 2 \\
 0 & 4 & 2 & | & 1 & | & 0 \\
 0 & 1/2 & 5 & | & -1/2 & | & 3 \\
 0 & 1/4 & -1/10 & | & 27/10 & | & 13/10 \\
 2/3 & 0 & 3/5 & | & 13/27 & | & 67/54
 \end{array}
 \end{array}$$

current
column

matrix

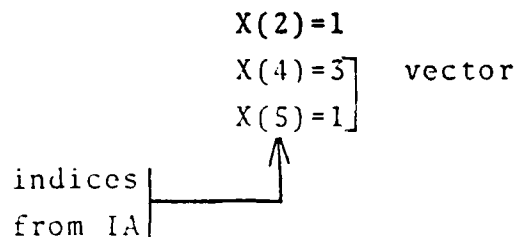
completely-factored matrix

from user	{	A	(column-ordered numeric values of A matrix) 3,2,4,2,1,2,6,3,1,3,1,2,3,1,5
		JA	(JA(j) points to beginning of jth column of A in IA) 1,3,6,9,12,16
		IA	(column-ordered list of row numbers of A) 1,5,2,3,4,2,3,5,2,4,5,1,3,4,5
gene- rated by sym- bolic	{	JL	(JL(j) points to beginning of jth column of L in IL) 1,2,4,6,7
		IL	(column-ordered list of row numbers of L) 5,3,4,4,5,5 fill
		JU	(JU(j) points to beginning of jth column of U in IU) 1,1,1,2,4,7
		IU	(column-ordered list of row numbers of U) 2,2,3,1,3,4 fill
gene- rated by nume- ric facto- riza- tion	{	L	(column-ordered numeric values of L) 2/3,1/2,1/4,-1/10,3/5,
		U	(column-ordered numeric values of U) 2,.,.,.,.,.
		DI	(ordered numeric values of diagonal) 3,4,5,.,.

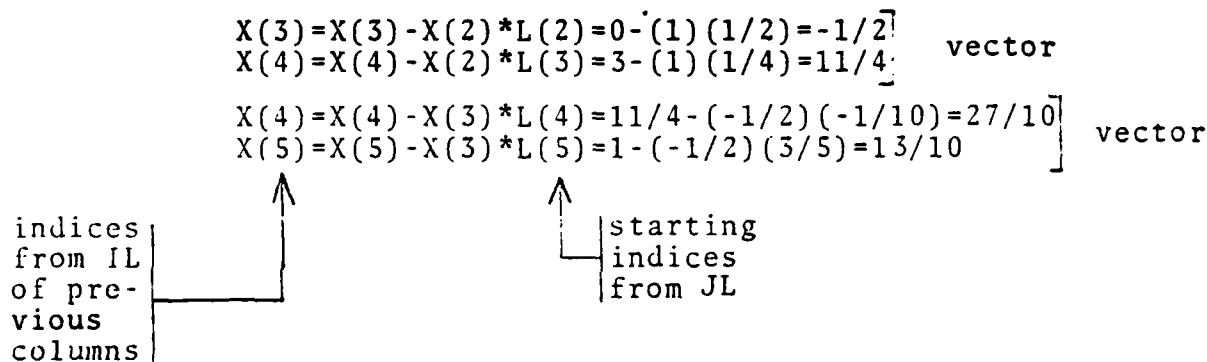
(a) Example up to factorization of fourth column

Table 1. Example of use of LU map in factorization

1. Zero expanded current column (X array)
2. Load current column with fourth column of A

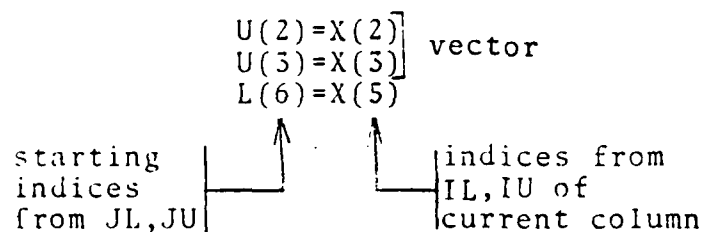


3. Factorize fourth column



$$\begin{array}{l} DI(4)=1/X(4)=10/27 \\ X(5)=X(5)*DI(4)=13/27 \end{array}$$

4. Store current column



(b) Steps in Factorization of Fourth Column

Table 1. Example of use of LU map in factorization

$$31,36,39,39,42,47 \quad (5)$$

This form is natural to looping operations for a scalar processor, where pairs of numbers are directly usable as upper and lower loop indices. Alternatively, the initial row position and the vector length could be stored as

$$31,6,39,1,42,6 \quad (6)$$

This form is favored by vector processors with hardware that counts down vector arithmetic operations to terminate a vector operation.

Another choice, preferred when a significant number of singleton (scalar) positions are present, represents a vector of length one with a minus sign prefixing the row number as

$$31,36,-39,42,47 \quad (7)$$

This latter structure has been adopted in this report.

4. Vector Fills

The multiply-subtract inner loop associated with factorization can result in production of fills that must be detected in the symbolic phase. In Figure 1, the process of multiplying the k^{th} column of L (termed a *preceeding* or *recalled* column) by $u_{k,r}$ and subtracting from the r^{th} column of L (termed the *current* column) is depicted. The zero-valued positions 37,38,40,41, which initially separate two vectors and a scalar, are filled by the dense vector (36,43) in the k^{th} column.

The symbolic phase produces the LU map by scanning the numbered pairs representing the vector structure of all the preceeding columns and the current column to determine zero-valued regions of the latter covered by at least one of the former. These are the fill positions.

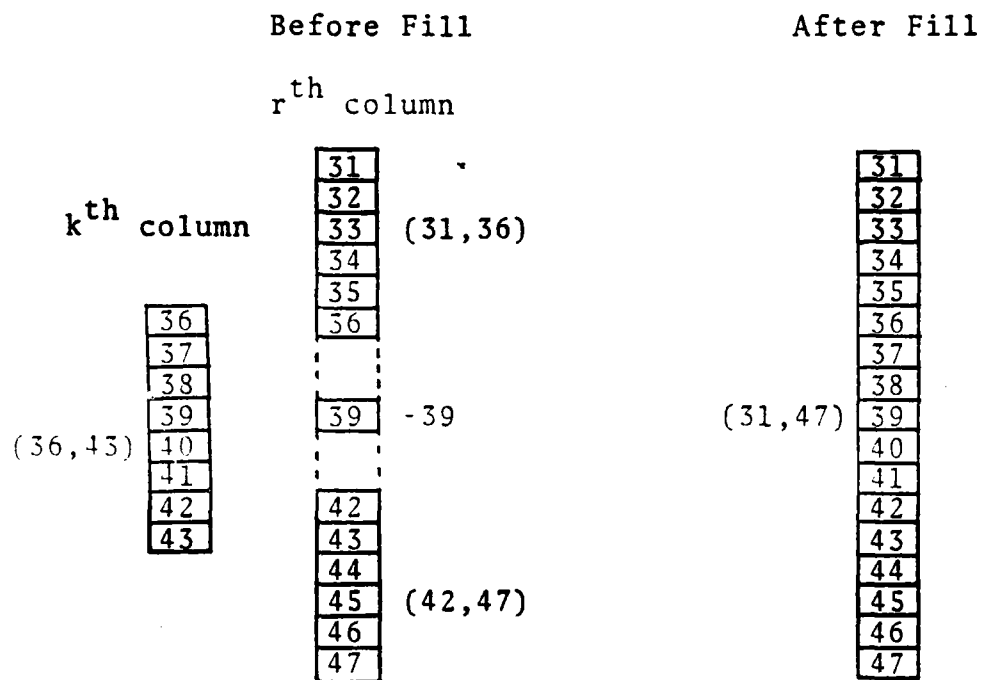


Figure 1. Example of vector fill, with data structure description (scalar indicated by - sign)

III. SOFTWARE DESCRIPTION

A. Symbolic preprocessing: Generation of compressed LU maps

CALL NEWFOR(N,IA,JA,IPC,IPR,IPRI,JL,JU,J,IL,IU,JVA,JVL,JVU,IX,IPOS,LENSC)

N^* is number of equations.

$IA(J)^*$ on entry, $IA(J)$ contains row number of j th
column-ordered matrix element;
on exit, IA contains compressed vector-scalar
format of same row number information.

$JA(J)^*$ points to first elements in IA of j th column;
 $JA(J)$ is changed by NEWFOR, as IA changes; $JA(N+1)$

*User supplied input data to subroutine.

IPC* points to one beyond last element of IA.
 is column permutation vector.
 IPR* is row permutation vector.
 IPRI inverse row permutation vector.
 JL(J) points to first element in IL and
 JU(J) IU of jth column; dimensioned at least N+1.
 IL(J) contains compressed vector-scalar row
 IU(J) map of L and U
 JVA(J) points to first elements in numeric
 JVu(J) arrays A, U, and L of jth column;
 JVL(J) dimensioned at least N.
 IX scratch arrays of dimension N.
 IPOS
 LENS C is the maximum number (≥ 1) of contiguous non-zeros
 in a column that are processed in scalar mode.

B. Numeric solution

CALL VMNP (N,JA,IA,JVA,A,IU,IL,JU,JL,DI,U,L,X,JVu,JVL,IPC,IPRI)

(see above argument list for NEWFOR)

A* array of numerical values of column-ordered matrix
 DI array of re-ordered diagonal elements of U.
 U arrays of numerical values of
 L column-ordered upper (U) and lower (L) triangular
 matrices; diagonal not included.
 X scratch array of dimension N.

C. Forward and back substitution.

CALL VMBP(N,IU,IL,JU,JL,JVu,JVL,DI,U,L,B,X,IPC,IPR)

*User supplied input data to subroutine

(see above argument lists for INEWFOR, VMNP)

B* array of numeric values of right hand side of entry,
and of solution on exit.

IV. PERFORMANCE

Three finite difference grids [5] illustrated in Figure 2 were solved using this code. The equations were ordered by alternate diagonals [6], which yields triangular LU factors of the form

$$\begin{array}{cc} D_1 & U_{12} \\ L_{21} & U_{22} \end{array}$$

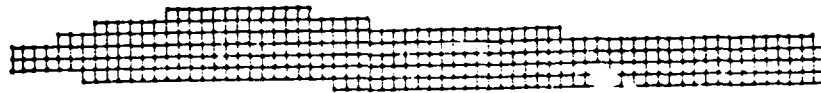
where D_1 is a diagonal matrix and U_{22} is profile matrix. Although U_{22} can be solved more efficiently [7][8], this code has the advantage of being in Fortran and simpler to use.

The performance on the CRAY-1 of the matrix factorizations step is depicted in Table 2. It should be noted that the execution rate is approximately proportional to the average vector length during solution. Of course, this is not true asymptotically, since the rate has a limiting value.

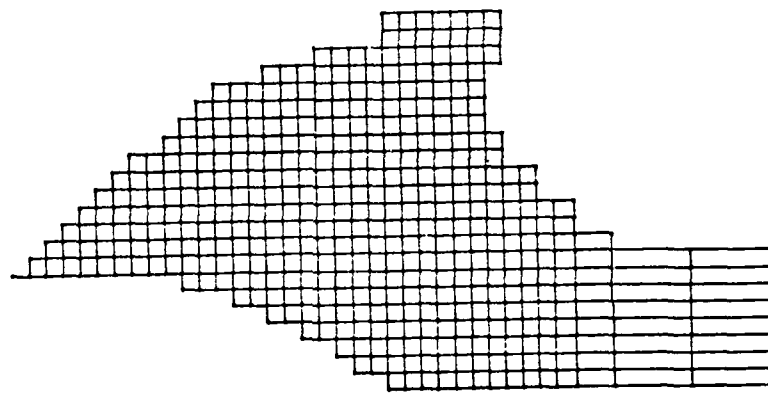
Problem	Equations	Time(msec)	MFLOPS	L
#1	391	20	1.59	4.2
#2	507	59	5.39	12.2
#3	2323	652	11.0	27.0

Table 2. CRAY-1 factorization performance summary for three grids of Figure 2;

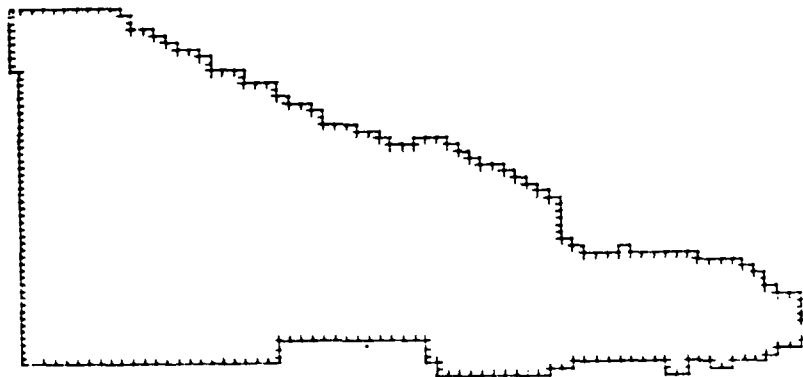
L is average vector length.



(a) Problem #1, 8x69, 391 equations



(b) Problem #2, 23x37, 507 equations



(c) Problem #3, 55x72, 2323 equations

Figure 2. Irregular grids

References

- [1] Chang, A., "Application of Sparse Matrix Methods in Electric Power System Analysis," *Sparse Matrix Proceedings*, Willoughby, Editor, pp. 113-122.
- [2] Gustavson, F.G., "Some Basic Techniques for Solving Sparse Systems of Linear Equations," *Sparse Matrices and Their Applications*, Rose & Willoughby Editors, Plenum Press, 1972, pp. 41-52.
- [3] Calahan, D.A., P.G. Buning, and W.N. Joy, "Vectorized General Sparsity Algorithms with Backing Store," Report #96, Systems Engineering Laboratory, University of Michigan, January, 1977.
- [4] Calahan, D.A., and W.G. Ames, "Vector Processors: Models and Applications," *IEEE Trans.*, vol. CAS-26, no. 9, September, 1979, pp. 715-726.
- [5] Woo, P.T., et al, "Application of Sparse Matrix Techniques to Reservoir Simulation," in *Sparse Matrix Computations*, ed. by J. R. Bunch and D. J. Rose, Academic Press, 1976, pp. 427-438.
- [6] Price, H.S., and K.H. Coates, "Direct Methods in Reservoir Simulation," *Soc. Pet. Engrs. Jour.*, vol. 14, 1974, pp. 295-308.
- [7] Calahan, D.A., "High Performance Banded and Profile Solvers for the CRAY-1: The Unsymmetric Case," Report #160, Systems Engineering Laboratory, University of Michigan, February, 1982.
- [8] Calahan, D.A., "High Performance Banded and Profile Solvers for the CRAY-1: The Symmetric Case," Report #166, Systems Engineering Laboratory, University of Michigan, October, 1982.
- [9] Calahan, D. A., "Multi-level Vectorized Sparse Solution of LSI Circuits," *Proc. IEEE Intl. Conf. on Circuits and Computers* (1980), Rye, NY., pp. 976-979.
- [10] Calahan, D. A., "Vectorized Direct Solvers for 2- D Grids," *Proc. 6th Symposium on Reservoir Simulation*, New Orleans, Feb. 1-2, 1982, pp. 489-506.

Appendix A
Program Listing

```

1  C**** THIS IS A DRIVER PROGRAM FOR TESTING A PROGRAM NEWFOR
2  C THAT COMPRESSES A COLUMN-ORDERED SPARSE MATRIX DESCRIPTION,
3  C AND PROGRAMS VMNP AND VMBP THAT FACTOR AND SOLVE THE
4  C MATRIX (RESPECTIVELY). NEWFOR NEED BE INVOKED ONLY ONCE
5  C FOR MULTIPLE SOLUTIONS WITH VMNP AND VMBP
6  C**** SCALAR/VECTOR GENERAL SPARSE SOLVER
7  C** DOCUMENTATION IN "VECTORIZED GENERAL SPARSITY ALGORITHMS
8  C** WITH BACKING STORE," D. A. CALAHAN, P. G. BUNING AND W. N. JOY
9  C** REPORT #96, U. OF MICHIGAN, JANUARY 1977; AND IN "USERS
10 C** MANUAL FOR VECTORIZED GENERAL SPARSE SOLVER," BY D. A. CALAHAN
11 C** OCTOBER 1982
12 C
13 C
14 C**** THE FOLLOWING ARRAYS HAVE A DIMENSION .GE. N
15 REAL A,B,DI,L,U,X,SUMR,SUMC
16 DIMENSION IPR(2325),IPC(2325),IPRI(2325),IX(2325)
17 &,JVA(2325),JVL(2325),JVU(2325),SUMR(2325),SUMC(2325),B(2325)
18 &,X(2325),DI(2325)
19 C**** THE FOLLOWING ARRAYS HAVE A DIMENSION .GE. N+1
20 DIMENSION IPOS(2326),JA(2326),JU(2326),JL(2326)
21 C**** THE FOLLOWING ARRAYS HAVE DIMENSIONS .GE. TO THE NUMBER
22 OF NON-ZEROS IN THE MATRIX
23 DIMENSION A(12000),IA(12000)
24 C**** THE FOLLOWING ARRAYS HAVE DIMENSIONS .GE. TO THE NUMBER
25 OF NON-ZEROS OF L AND U; IU AND IL ARE COMPRESSED AND
26 MAY REQUIRE MUCH LESS THAN THIS PESSIMISTIC ESTIMATE
27 DIMENSION IU(6700),IL(6700)
28 COMMON /EX2/U(86000)
29 COMMON /EX1/L(86000)
30 READ(5,88)N
31 NP1=N+1
32 READ(5,88)(JA(J),J=1,NP1)
33 NA=JA(NP1)-1
34 READ(5,88)(IA(J),J=1,NA)
35 FORMAT(16I5)
36 DO 89 J=1,N
37 IPR(J)=J
38 IPC(J)=J
39 CALL RANGEN(JA,IA,N,IPR,IPC)
40 NP1=N+1
41 WRITE(6,17)(JA(J),J=1,NP1)
42 NA=JA(NP1)-1
43 WRITE(6,17)(IA(J),J=1,NA)
44 FORMAT(20I3)
45 LENS=2
46 CALL FORM(A,B,IA,JA,IPR,IPC,SUMR,SUMC,N,NA)
47 CALL NEWFOR(IA,JA,IPC,IPR,IPRI,JL,JU,IL,IU,JVA,JVL,
48 1 JVU,IX,IPOS,N,LENSC)
49 CALL VMNP(N,JA,IA,JVA,A,IU,IL,JU,JL,DI,U,L,X,JVU,JVL,IPC,IPRI)
50 CALL VMBP(N,IU,IL,JU,JL,JVU,JVL,DI,U,L,B,X,IPC,IPR)
51 WRITE(7,77)(B(J),J=1,N)
52 FORMAT(5E12.4)
53 GO TO 18
54 END
77

```

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

```

55 C**** THIS SUBROUTINE COMPRESSES AN ARRAY OF O'S AND 1'S (IPOS)
56 REPRESENTING A COLUMN OF A,L,OR U INTO AN ORDERED LIST
57 IN VECTOR-SCALAR FORM
58 SUBROUTINE ILU(IU,JU,IPOS,IV,NU,NUV,ISTRT,IEND,J,LENSC)
59 DIMENSION IV(1),IU(1),JU(1),IPOS(1)
60 IV(J)=NUV+1
61 JU(J)=NU+1
62 IF(IEND.EQ.O)GO TO 7
63 I=ISTRT
64 IPL=O
65 IPC=IPOS(I)
66 IPOS(I)=O
67 I=I+1
68 IPN=IPOS(I)
69 IF(IPC.NE.O)GO TO 2
70 IF(I.GT.IEND)GO TO 7
71 IPL=IPC
72 IPC=IPN
73 I=I+1
74 IPN=IPOS(I)
75 GO TO 3
76 IPOS(I-1)=O
77 IF(IPL.NE.O)GO TO 4
78 IF(IPN.NE.O)GO TO 6
79 C**** SCALAR
80 NU=NU+1
81 NUV=NUV+1
82 IU(NU)=-(I-1)
83 GO TO 5
84 IF(IPN.NE.O)GO TO 5
85 C**** START OR END OF VECTOR
86 LEN=I-IU(NU)
87 NUV=NUV+LEN
88 IF(LEN.GT.LENSC)GO TO 6
89 II=-(I-1)+LEN
90 NU=NU-1
91 DO 8 L=1,LEN
92 NU=NU+1
93 IU(NU)=II-L
94 GO TO 5
95 NU=NU+1
96 IU(NU)=(I-1)
97 GO TO 5
98 RETURN
99 END

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

ISN

100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

```

C*** SUBROUTINE RANGEN(JA,IA,N,IPR,IPC)
      GENERATES RANDOMLY-POSITIONED TEST MATRICES
C*** DIMENSION IPR(1),IPC(1),IA(1),JA(1)
C*** AROW IS AVERAGE NUMBER OF NON-ZEROS PER ROW
C*** N IS THE NUMBER OF EQUATIONS
      READ(5,3)AROW,N
      FORMAT(F10.0,I6)
      NA=0
      NNN=999
      AN=N
      TOL=(AROW-1)/AN
      JA(1)=1
      DO 33 J=1,N
        DO 1 K=1,N
          IF(J.EQ.K)GO TO 5
          IF(RANF(NNN).GT.TOL)GO TO 1
5        NA=NA+1
1        IA(NA)=K
33       JA(J+1)=NA+1
        DO 37 J=1,N
          IPR(J)=J
          IPC(J)=J
37       RETURN
      END

```

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

1

SUBROUTINE VMNP(N,JA,IA,JVA,A,IU,IL,JU,IL,DI,U,L,X,JVU,JVL,

C IPC,IPRI)

C*****

C* V M N P - SPARSE LU FACTORIZATION (NUMERIC) *

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

C* *****

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

2 3

4

5

6

7

8

DATE:09-28-82,11:00 OWNER:SMXA FILE:SPARF

ISN

```

183 C STOP 27201 - NUMERICAL VALUE OF PIVOT IS APPROXIMATELY O.
184 C
185 C *****
186 C
187 C INITIALIZE POINTERS:
188 C JUJ IS INDEX INTO L MATRIX VALUE VECTOR.
189 C (USED TO STORE VALUES IN L AND U CALCULATION LOOPS.)
190 C JAA IS INDEX INTO A MATRIX VALUE VECTOR.
191 C (USED TO RETRIEVE VALUES FROM A MATRIX FOR EACH COLUMN.)
192 C
193 C
194 C JUU= O
195 C JLL= O
196 C ILBOT= O
197 C IABOT= O
198 C IUBOT= O
199 C DO 17 I=1,N
200 C 17 X(I)=O.
201 C *****
202 C
203 C LOOP TO CALCULATE NUMERICAL VALUES FOR COLUMN I OF L AND L
204 C *****
205 C DO 80 I= 1,N
206 C
207 C GET POINTERS FOR THIS COLUMN IN L AND L:
208 C IUTOP IS INDEX TO TOP OF ROW DESCRIPTORS FOR THIS COLUMN IN U.
209 C IUBOT IS INDEX TO BOTTOM OF ROW DESCRIPTORS FOR THIS COLUMN IN U.
210 C ILTOP IS INDEX TO TOP OF ROW DESCRIPTORS FOR THIS COLUMN IN L.
211 C ILBOT IS INDEX TO BOTTOM OF ROW DESCRIPTORS FOR THIS COLUMN IN L.
212 C IUTOP= IUBOT+1
213 C IUBOT= JU(I+1)-1
214 C ILTOP= ILBOT+1
215 C ILBOT= JL(I+1)-1
216 C
217 C GET POINTER TO DATA VALUE DESCRIPTORS FOR A:
218 C IATOP IS INDEX TO TOP OF ROW INDICIES FOR THIS COLUMN.
219 C IABOT IS INDEX TO BOTTOM OF ROW INDICIES FOR THIS COLUMN.
220 C
221 C *** NOTE *** WE ARE PROCESSING COLUMN IPC(I) OF A ***
222 C
223 C IATOP= JA(IPC(I))
224 C IABOT= JA(IPC(I)+1)-1
225 C
226 C INITIALIZE WORKSPACE X WITH ZEROS AT RESULTANT POSITIONS
227 C
228 C ***** COULD APPROXIMATE - ZERO LOWEST TO HIGHEST POSITION
229 C
230 C IF THERE ARE NO NONZERO POSITIONS IN L THEN THE ONLY RESULTANT
231 C POSITIONS WILL BE THE ELEMENTS IN L. SINCE THESE WILL BE GIVEN
232 C INITIAL VALUES IN THE LOOP BEGINNING AT #31 BELOW, WE DONT NEED
233 C TO ZERO ANY POSITIONS IN THIS CASE.
234 C IF (IUTOP.GT.IUBOT) GOTO 30
235 C ZEROS IN X AT POSITIONS OF COLUMNS OF JU
236 C K= IUTOP
237 C JUI= IU(K)
238 C IF(JUI.LT.O) GOTO 12
239 C K= K+1
240 C JUE= IU(K)
241 C DO 11 J= JUI,JUE
242 C X(J)= O.

```

DATE: 09-28-82, 11:00 OWNER: SMYA FILE: SPARF

ISN

```

241 C          GOTO 13
242 C          X(-JUI)= O.
243 C          K= K+1
244 C          IF (K.LE.IUBOT) GOTO 10
245 C          C ZEROS IN X AT POSITIONS OF COLUMNS OF JL
246 C          IF (ILTOP.GT.ILBOT) GOTO 30
247 C          K= ILTOP
248 C          JLI= IL(K)
249 C          IF (JLI.LT.O) GOTO 22
250 C          K= K+1
251 C          JLE= IL(K)
252 C          DO 21 J= JLI,JLE
253 C          X(J)= O.
254 C          GOTO 23
255 C          X(-JLI)= O.
256 C          K= K+1
257 C          IF (K.LE.ILBOT) GOTO 20
258 C          C COPY COLUMN IPC(I) OF A INTO X
259 C          CJNTINUE
260 C          JAA= JVA(IPC(I))-1
261 C          K= IATOP
262 C          JAP= IA(K)
263 C          IF (JAP.LT.O) GOTO 33
264 C          K= K+1
265 C          JAL=IA(K)-JAP+1
266 C          DO 32 J= 1,JAL
267 C          X(IPRI(JAP+J-1))= A(JAA+J)
268 C          JAA= JAA+JAL
269 C          GOTO 34
270 C          JAA= JAA+1
271 C          X(IPRI(-JAP))= A(JAA)
272 C          K= K+1
273 C          IF (K.LE.IABOT) GOTO 31
274 C
275 C          C*****
276 C          CALCULATE ENTRIES IN COLUMN I OF UPPER TRIANGULAR MATRIX U
277 C          C*****
278 C
279 C          C NOTHING TO DO IF COLUMN I OF U CONTAINS ONLY THE DIAGONAL ELEMENT
280 C          IF (IUTOP.GT.IUBOT) GOTO 60
281 C          C LOOP ON EACH DESCRIPTOR IN JU MATRIX FOR THIS COLUMN IN U.
282 C          J= IUTOP
283 C          JUJ= IU(J)
284 C          C GET NEXT ELEMENT FROM COLUMN I
285 C          IF (JUJ.GT.O) GOTO 41
286 C          JUJ= -JUJ
287 C          JUJL= JUJ
288 C          GOTO 42
289 C          J= J+1
290 C          JUJL= IU(J)
291 C          C LOOP ON EACH POSITION IN THE ELEMENT
292 C          DO 54 LL= JUJ,JUJL
293 C          XJI= X(LL)
294 C          X(LL)=O.
295 C          U(JUU+LL-JUJ+1)= XJI
296 C          C LOCATE ROW DESCRIPTOR INDICIES FOR COLUMN LL IN U
297 C          JTOP= JL(LL)
298 C          JBOT= JL(LL+1)-1

```

23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

```

299 IF (JTOP.GT.JBOT) GOTO 54
300 C COLUMN HAS SYMBOLIC POSITIONS SO LOOP THROUGH THEM
301 JLL2M= JVL(LL)-1
302 K= JTOP
303 JLK=IL(K)
304 C OPERATE ON AN ELEMENT FROM COLUMN LL
305 IF (JLK.LT.O) GOTO 52
306 C VECTOR IN COLUMN LL
307 K= K+1
308 JLKL= IL(K)
309 JDIFF=JLL2M-JLK+1
310 DO 51 JJ= JLK,JLKL
311 X(JJ)=X(JJ)-XJI*L(JDIFF+JJ)
312 C UPDATE VALUE INDEX TO POINT TO ABOVE NEXT VALUE IN L.
313 JLL2M= JLL2M+JLKL-JLK+1
314 GOTO 53
315 C SCALAR IN COLUMN LL
316 JLL2M= JLL2M+1
317 X(-JLK)= X(-JLK)-XJI*L(JLL2M)
318 K= K+1
319 IF (K.LE.JBOT) GOTO 50
320 CONTINUE
321 C DONE WITH THIS ELEMENT/VECTOR IN THE A MATRIX.
322 C UPDATE JUU TO POINT TO NEXT VALUE TO BE STORED.
323 JUU= JUU+JUUL-JUU+1
324 J= J+1
325 IF (J.LE.IUBOT) GOTO 40
326
327 C ***** END OF COMPUTATION OF COLUMN I OF L *****
328 C
329 C SET AND STORE RECIPROCAL OF DIAGONAL ELEMENT L(II)
330
331 60 XD= X(I)
332 X(I)=O.
333 IF (ABS(XD).EQ.O.DO) GOTO 90
334 XD= 1.DO/XD
335 DI(I)= XD
336 C *****
337 C CALCULATE ENTRIES IN COLUMN I OF UPPER TRIANGULAR MATRIX L
338 C *****
339
340 C BRANCH IF COLUMN I OF L CONTAINS ONLY THE DIAGONAL ELEMENT L(I)=1
341 IF (I.TOP.GT.IBOT) GOTO 80
342 C LOOP ON EACH DESCRIPTOR IN JU MATRIX FOR THIS COLUMN IN L.
343 J= I.TOP
344 JLJ= IL(J)
345 C GET ELEMENT FROM COLUMN I
346 IF (JLJ.LT.O) GOTO 72
347 J= J+1
348 JLJL= IL(J)
349 C OPERATE ON A VECTOR
350 DO 71 JJ=JLJ,JLJL
351 L(JLL+JJ-JLJ+1)= XD*X(JJ)
352 71 X(JJ)=O.
353 JLL= JLL+JLJL-JLJ+1
354 GOTO 73
355 C OPERATE ON A SCALAR
356 72 JLL= JLL+1

```

ISN

357
358
359
360
361
362
363
364
365

L(JLL)= XD*X(-JLJ)
X(-JLJ)=0.
J= J+1
IF (J.LE.1LBOT) GOTO 70
C***** END OF COMPUTATION OF COLUMN I OF L *****
80 CONTINUE
RETURN
90 STOP 27201
END

73

C*****

90

L(JLL)= XD*X(-JLJ)
X(-JLJ)=0.
J= J+1
IF (J.LE.1LBOT) GOTO 70
C***** END OF COMPUTATION OF COLUMN I OF L *****

IF (J.LE.1LBOT) GOTO 70

C***** END OF COMPUTATION OF COLUMN I OF L *****

80 CONTINUE

RETURN

90 STOP 27201

END

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

```

366 SUBROUTINE VMBP(N,IU,IL,JU,JL,JVU,JVL,DI,U,L,B,X,IPC,IPR)
367 .....
368 *
369 * V M B P - SPARSE FORWARD AND BACK SUBSTITUTION *
370 *
371 .....
372 *
373 * THIS SUBROUTINE SOLVES THE MATRIX EQUATION A*X=B GIVEN *
374 * THE LU FACTORED A MATRIX (FROM VMNSP) AND THE VECTOR B *
375 *
376 * ***** ARBITRARY PIVOTING ORDER VERSION *****
377 *
378 .....
379 * INTEGER JU,JL,IPC,IPR
380 * REAL L
381 *
382 * N - NUMBER OF COLUMNS IN THE MATRIX.
383 *
384 * DIMENSION JU(1),JL(1),IU(1),IL(1),JVU(1),JVL(1)
385 *
386 * JU(I) - START OF ROW POSITION DESCRIPTORS IN JU FOR COL I
387 * JL(I) - START OF ROW POSITION DESCRIPTORS IN JL FOR COL I
388 * JU - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
389 * - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
390 * NEXT ELEMENT OF JU IS ENDING ROW FOR THIS VECTOR.
391 * - IF NEGATIVE, IABS IS ROW INDEX OF SINGLE ELEMENT.
392 * - IF POSITIVE, STARTING ROW FOR A VECTOR, AND THE
393 * NEXT ELEMENT OF JL IS ENDING ROW FOR THIS VECTOR.
394 * JVU - INDEX OF FIRST COLUMN I J MATRIX VALUE.
395 * JVL - INDEX OF FIRST COLUMN I L MATRIX VALUE.
396 *
397 * DIMENSION U(1),L(1),B(1),X(1),DI(1)
398 *
399 * U - VECTOR OF UPPER TRIANGULAR NUMERICAL VALUES.
400 * L - VECTOR OF LOWER TRIANGULAR NUMERICAL VALUES.
401 * X - SCRATCH VECTOR OF LENGTH N.
402 * B - VECTOR OF RIGHT HAND SIDE VALUES.
403 * DI - VECTOR OF INVERSE DIAGONAL ELEMENT VALUES
404 * RETURNED SOLUTION VECTOR FOR EQUATION A*X=B
405 *
406 * DIMENSION IPC(1),IPR(1)
407 *
408 * IPC - COLUMN PIVOT PERMUTATION VECTOR.
409 * IPR - ROW PIVOT PERMUTATION VECTOR.
410 *
411 * NM1= N-1
412 *
413 *
414 * DO 5 J= 1,N
415 * IPRX= IPR(J)
416 * X(J)= B(IPRX)
417 *
418 * C FORWARD SUBSTITUTION
419 *
420 * DO 30 I=1,NM1
421 * ITOP= JL(I)
422 * IBOT= JL(I+1)-1
423 * IVTI= JVL(I)-1

```

ISN

```

15 15
16 16
17 17
18 18
19 19
20 20
21 21
22 22
23 23
24 24
25 25
26 26
27 27
28 28
29 29
30 30
31 31
32 32
33 33
34 34
35 35
36 36
37 37
38 38
39 39
40 40
41 41
42 42
43 43
44 44
45 45
46 46
47 47
48 48
49 49
50 50
51 51
52 52
53 53
54 54
55 55
56 56
57 57
58 58
59 59

10 10
15 15
20 20
25 25
30 30
35 35
40 40
45 45
50 50
55 55
60 60

AUX= X(I)
IF (ITOP.GT.IBOT) GOTO 30
J= ITOP
JLJL= IL(J)
IF (JLJL.LT.O) GOTO 20
JLJL= JLJL-1
J= J+1
JLJLEN= IL(J)-JLJL
DO 15 K= 1,JLJLEN
X(JLJL+K)= X(JLJL+K)-AUX*L(IVTI+K)
IVTI= IVTI+JLJLEN
GOTO 25
IVTI= IVTI+1
X(-JLJL)= X(-JLJL)-AUX*L(IVTI)
J= J+1
IF (J.LE.IBOT) GOTO 10
CONTINUE
X(N)= X(N)*DI(N)

C BACK SUBSTITUTION
C
I= N
32 ALPHA= X(I)
ITOP= JU(I)
IBOT= JU(I+1)-1
JVUI= JVUI(I)-1
IF (ITOP.GT.IBOT) GOTO 55
J= ITOP
JUJL= IU(J)
IF (JUJL.LT.O) GOTO 45
JUJL= JUJL-1
J= J+1
JUJLEN= IU(J)-JUJL
DO 40 K= 1,JUJLEN
X(JUJL+K)= X(JUJL+K)-ALPHA*U(JVUI+K)
JVUI= JVUI+JUJLEN
GOTO 50
JVUI= JVUI+1
X(-JUJL)= X(-JUJL)-ALPHA*U(JVUI)
J= J+1
IF (J.LE.IBOT) GOTO 35
I= I-1
X(I)= X(I)*DI(I)
IF (I.GT.1) GOTO 32
DO 60 I= 1,N
B(IPC(I))= X(I)
RETURN
END
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471

```

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

```

472 SUBROUTINE VSORT(A,N)
473
474 C PARTITION SORTING ALGORITHM
475 C REFERENCE COLLECTED ALGORITHMS OF THE ACM - 63,64,65
476 C
477 C INITIALIZE
478   INTEGER A(1)
479   DIMENSION IHIGH(32), ILOW(32)
480   NSEGS= 1
481   IL= 1
482   IH= N
483
484 C IF NO ELEMENTS IN THIS SEGMENT DO NOTHING
485   10 IF (IL.GE.IH) GOTO 70
486 C CHOOSE ISEP (SEPARATION ENTRY):
487 C MAKE A(IL) <= A(IL+IH)/2 <= A(IH) BY INTERCHANGE
488   20 SET ISEP= A((IL+IH)/2)
489   ISEP= A(ISEP)
490 C IXL IS LOWER SEGMENT INDEX (CURRENT)
491   IXL= IL
492 C MAKE A(IL) <= A(ISEP)
493   IF (A(IL).LE.ISEP) GOTO 30
494   A(ISEP)= A(IL)
495   A(IL)= ISEP
496   ISEP= A(ISEP)
497 C IXL IS HIGHEST SEGMENT INDEX (CURRENT)
498   30 IXL= IH
499 C MAKE A(IH) >= A(ISEP)
500   IF (A(IH).GE.ISEP) GOTO 50
501   A(ISEP)= A(IH)
502   A(IH)= ISEP
503   ISEP= A(ISEP)
504 C MAKE A(IL) <= A(ISEP)
505   IF (A(IL).LE.ISEP) GOTO 50
506   A(ISEP)= A(IL)
507   A(IL)= ISEP
508   ISEP= A(ISEP)
509   GOTO 50
510 C EXCHANGE LOW PART ENTRY WHICH IS GREATER THAN SEPARATOR WITH HIGH
511 C PART ENTRY WHICH IS LESS THAN OR EQUAL TO THE SEPARATOR VALUE.
512   40 ITT= A(IXH)
513   A(IXH)= A(IXL)
514   A(IXL)= ITT
515 C MOVE DOWN UPPER SEGMENT AS FAR AS WE CAN
516   50 IXL= IXL-1
517   IF (A(IXL).GT.ISEP) GOTO 50
518 C MOVE UP LOWER SEGMENT AS FAR AS WE CAN
519   51 IXL= IXL+1
520   IF (A(IXL).LT.ISEP) GOTO 51
521 C NOTHING TO DO IF BOTH SEGMENTS HAVE AT MOST ONE ENTRY IN COMMON
522   IF (IXL.LE.IXH) GOTO 40
523 C IF BOTH SEGMENTS OVERLAP THEN THEY ARE SEPARATED
524 C IN THIS CASE CONTINUE WITH SHORTER SEGMENT, STORING THE LONGER
525   IF (IXH-IL.LE.IH-IXL) GOTO 60
526 C LOWER SEGMENT LONGER, CONTIN WITH UPPER AFTER SAVING LOWER
527   ILOW(NSEGS)= IL
528   IHIGH(NSEGS)= IXH
529   IL= IXL

```

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558

```

NSEGS= NSEGS+1
GOTO 80
C UPPER SEGMENT LONGER, CONTIN WITH LOWER AFTER SAVING UPPER
60 ILOW(NSEGS)= IXL
   IH= IXL
   IHIGH(NSEGS)= IH
NSEGS= NSEGS+1
GOTO 80
C GET ANOTHER SEGMENT FOR PROCESSING IF THERE ARE ANY MORE
70 NSEGS= NSEGS-1
   IF (NSEGS.EQ.0) RETURN
   IL= ILOW(NSEGS)
   IH= IHIGH(NSEGS)
C CONTINUE TO SEGMENT AS LONG AS LENGTH IS GREATER THAN 11
80 IF (IH-IL GE.11) GOTO 20
   IF (IL.EQ.1) GOTO 10
GOTO 91
C SORT ELEMENTS WITHIN SEGMENT BY INTERCHANGE OF ADJACENT PAIRS
90 IL= IL+1
91 IF (IL.EQ.IH) GOTO 70
   ISEP= A(IL+1)
   IF (A(IL).LE.ISEP) GOTO 90
   IXL= IL
100 A(IXL+1)= A(IXL)
   IXL= IXL-1
   IF (ISEP.LT.A(IXL)) GOTO 100
   A(IXL+1)= ISEP
GOTO 90
END

```

37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61

DATE: 09-28-82, 11:00 OWNER: SMXA FILE: SPARF

ISN

```

559 SUBROUTINE FORM(A,B,IA,JA,IPR,IPC,SUMR,SUMC,N,NA)
560 DIMENSION A(1),B(1),IA(1),JA(1),IPR(1),IPC(1),SUMR(1),SUMC(1)
561 DO 12 I=1,NA
562   A(I)=0.
563   UNIFORMLY-DISTRIBUTED NEGATIVE OFF-DIAGONAL VALUES
564   NNN=999
565   DO 29 J=1,NA
566     A(J)=-RANF(NNN)
567   DO 103 J=1,N
568     SUMR(J)=0.
569     SUMC(J)=0.
570     B(J)=0
571   C*** FORMULATE EQUATIONS SO SOLUTION IS X(I) = I + 1
572   DO 37 I=1,N
573     I1=JA(I)
574     I2=JA(I+1)-1
575     DO 4 J=I1,I2
576       ICOL=IA(J)
577       SUMC(I)=SUMC(I)-A(J)
578       SUMR(ICOL)=SUMR(ICOL)-A(J)
579       B(ICOL)=B(ICOL)+A(J)*(I+1)
580   C*** CONTINUE
581   FIND PIVOTS AND FORCE DOMINANCE
582   DO 33 I1=1,N
583     I=IPC(I1)
584     I1=JA(I)
585     I2=JA(I+1)-1
586     DO 34 J=I1,I2
587       ICOL=IA(J)
588       IF(ICOL.NE.IPR(I1))GO TO 34
589       B(ICOL)=B(ICOL)-A(J)*(I+1)
590       A(J)=.01+1.1DO*AMAX1(SUMC(I)+A(J),SUMR(ICOL)+A(J))
591       B(ICOL)=B(ICOL)+A(J)*(I+1)
592   GO TO 33
593   C*** CONTINUE
594   CONTINUE
595   RETURN
596   END

```

```

597 C**** THIS SUBROUTINE GENERATES COMPRESSED MAPS OF A,U, AND L
598 SUBROUTINE NEWFOR(IA,JA,IPC,IPR,IPRI,JL,JU,IL,IU,JVA,JVL,
599 JVV,IX,IPOS,N,LWNSC)
600 DIMENSION IA(1),JA(1),IPC(1),IPRI(1),JL(1),JU(1),IL(1),IU(1),
601 &JVA(1),JVL(1),JVV(1),IX(1),IPOS(1),IPRI(1)
602 C**** DETERMINE INVERSE ROW PERMUTATION VECTOR
603 DO 104 J=1,N
604 IPRI(IPR(J))=J
605 NP1=N+1
606 DO 130 J=1,NP1
607 IPOS(J)=0
608 C**** DETERMINE FILL OF REORDERED MATRIX AND GENERATE COLUMN-ORDERED
609 C LIST OF L AND U STRUCTURE IN VECTOR-SCALAR FORM
610 NU=0
611 NL=0
612 NUV=0
613 NLV=0
614 DO 1 J=1,N
615 IP=IPC(J)
616 I1=JA(IP)
617 I2=JA(IP+1)-1
618 I3=0
619 DO 2 I=I1,I2
620 I3=I3+1
621 IX(I3)=IPRI(IA(I))
622 CALL VSORT(IX,I3)
623 IMIN=IX(1)
624 DO 10 I=1,I3
625 IPOS(IX(I))=1
626 IMAX=IX(I3)
627 L1=0
628 L2=0
629 JM1=J-1
630 IF(IMIN.GT.JM1)GO TO 11
631 DO 3 I=IMIN,JM1
632 IF(IPOS(I).EQ.O)GO TO 3
633 K1=JL(I)
634 K2=JL(I+1)-1
635 IF(K1.GT.K2)GO TO 3
636 K=K1-1
637 K=K+1
638 L1=IL(K)
639 IF(L1.LT.O)GO TO 7
640 K=K+1
641 L2=IL(K)
642 DO 6 LL=L1,L2
643 IPOS(LL)=1
644 GO TO 8
645 IPOS(-L1)=1
646 IF(K.LT.K2)GO TO 5
647 IMAX=MAXO(IMAX,L2)
648 IMAX=MAXO(IMAX,-L1)
649 CONTINUE
650 3 CONTINUE
651 IPOS(J)=0
652 CALL ILU(IU,JU,IPOS,JV,NU,NLV,IMIN,J-1,J,LENSC)
653 JU(J+1)=NU+1
654 CALL ILU(IL,JL,IPOS,JVL,NL,NLV,J+1,IMAX,J,LENSC)

```

DATE:09-28-82,11:00 OWNER:SMXA FILE:SPARF

ISN

```

655 JL(J+1)=NL+1
656 CONTINUE
657 C**** COMPRESS MATRIX (A) STRUCTURE INTO VECTOR-SCALR FORM
658 NA=O
659 NAV=O
660 DO 109 J=1,N
661 I1=JA(J)
662 I2=JA(J+1)-1
663 IMAX=O
664 IMIN=10000
665 DO 108 I=I1,I2
666 IAX=IA(I)
667 IPOS(IAI)=1
668 IMIN=MINO(IAI,IMIN)
669 IMAX=MAXO(IAI,IMAX)
670 CALL ILU(IA,JA,IPOS,JVA,NA,NAV,IMIN,IMAX,J.LENSC)
671 JA(NP1)=NA+1
672 FORMAT(20I3)
673 RETURN
674 END
```

3-8

DT