TWO SUBROUTINE PACKAGES FOR THE EFFICIENT UPDATING OF MATRIX FACTORIZATIONS

by

A. K. Cline

March, 1977

TR-68
CNA-120

This report is being released jointly with the Center for Numerical Analysis at The University of Texas at Austin under the number CNA-120.

Department of Computer Sciences
The University of Texas at Austin

INTRODUCTION:

The algorithms for the solution of many mathematical programming problems require solving a sequence of linear systems in which the successive matrices differ by just one row or column. Perhaps best known of these is the simplex algorithm for linear programs. Typically, this algorithm is described concurrently with its numerical implementation (instead of as a purely linear algebraic process), and this implementation involves the updating of a sequence of matrix inverses (see [8], for example). This inverse matrix updating, although efficient, has poor numerical properties. A detailed discussion of this may be found in [1] or [4]. Briefly, the difficulty lies in the fact that if, due to ill-conditioning, one matrix inverse is poorly determined, all succeeding inverses are poorly determined, be they ill-conditioned or not.

As is shown in [1], [2], and [4], the simplex algorithm can be separated from updated inverses, and these replaced by a special matrix factorization which is both numerically stable and efficient. This algorithm, suggested by Golub and Stoer and exploited by Bartels in [1], employs a factorization of the form $U = LA$ where A is the square, nonsingular matrix being factored, U is upper triangular and L is square. From the factorization of A, a second matrix A' differing from A in only one column, can be similarly factored in a number of additions and multiplications proportional to n^2. Details of this updating process can be found in [1], [3] or [4], and an analogous process for matrices differing in one row is described in [6]. The row updating algorithm employs an $L = AU$ factorization, where L is lower triangular and U is square.

Sequences of linear systems with matrices differing in only one row or column occur in problems other than linear programs. Algorithms for the L_1 and L_∞ solutions to over-determined linear systems by ascent and descent methods contain such sequences (see [5], [6] and [8]). Even the solution to a nonlinear system of equations could involve such a sequence if a Newton-like method were employed in which, at each iteration, only one or several rows or columns of the Jacobian matrix were re-evaluated. There are real time computing problems in which, at each step, a nonlinear equation differing slightly from the previous equation needs to be solved. If a new Jacobian cannot be totally evaluated in any time step, then perhaps one or several rows or columns could be evaluated, and the factorization method could be employed to solve for the Newton iterates.

DESCRIPTION:

There are two packages for the factorization updating: one for columns (COLUP) and one for rows (ROWUP). Each package contains five subroutines written in very portable FØRTRAN. Reference [4] contains an ALGOL procedure for column updating, and although the present codes employ the same general method, they are not translations of the ALGOL and they have additional capabilities. Whereas the ALGOL procedure begins with a factorization of the identity matrix and then allows this to be updated, the present packages have subroutines which perform initial U = LA (or L = AU) factorizations of arbitrary matrices A. A special capability for the initial matrix being the identity is also included here because many applications are characterized by that situation (or one in which the initial matrix differs from the identity in only one or several rows or columns).

Specifically, the column oriented package, COLUP contains

UELAD — which performs the initial factorization of U = LA,

UELAID — which performs the initial factorization of U = LI (i.e. a more efficient version of UELAD for A = I),

UELASO — which solves linear systems Ax = b using the U = LA factorization,

UELAST — which solves linear systems $A^T x = b$ using the U = LA factorization, (where A^T denotes A transposed),

UELAUP — which updates the U and L factors of U = LA, for a new matrix A' which differs from A in only one column. It is assumed that the one column to be omitted from A is removed, all columns to the right are shifted left one column, and the new column is appended as the rightmost one.

The analogous row oriented package, ROWUP contains

LEAUD — which performs the initial factorization of L = AU,

LEAUID — which performs the initial factorization of L = IU (i.e. a more efficient version of LEAUD for A = I),

LEAUSO — which solves linear systems Ax = b using the L = AU factorization,

LEAUST — which solves linear systems $A^T x = b$ using the U = LA factorization, (where A^T denotes A transposed),

LEAUUP — which updates the L and U factors of L = AU, for a new matrix A', which differs from A in only one row. It is assumed that the one row to be omitted is removed, all rows below it are moved up one row, and the new row is appended as the last one.

The package COLUP uses a row-wise packed form for the upper triangular factor to save storage. Similarly ROWUP uses a column-wise packed form for the lower triangular factor.

TESTING:

The packages have received extensive testing in applications programs for ascent and descent algorithms for the L_∞ solution to over-determined systems and for a linear complementary algorithm.

In order to display the stability properties, we have measured the performance of the packages in the following test situation:

1. Construct a 10x10 original matrix A with elements uniformly randomly distributed on [0,1]. Factor A with UELAD (LEAUD) and compute the "relative residual norm" as $||U - LA||/||L||\cdot||A||$ ($||L - AU||/||A||\cdot||L||$) using L_∞ matrix norms.
2. Generate a right-hand side vector b with similarly distributed random elements. Solve $Ax = b$ with UELASO (LEAUSO) and compute the "relative residual norm" as $||Ax - b||/||A||\cdot||x||$. Also solve $A^Tx = b$ with UELAST (LEAUST) and compute "relative residual norm" as $||A^Tx - b||/||A^T||\cdot||x||$.

For i = 1,...,n

3. Replace column (row) i of A by a new column (row) of similarly distributed random elements. Call the new matrix A and obtain its factorization from the previous one using UELAUP (LEAUUP). Determine "relative residual norms" as before.
4. Generate right-hand side vectors b as in 2 and solve $Ax = b$ with UELASO (LEAUSO) and solve $A^Tx = b$ with UELAST (LEAUST).

Compute "relative residual norms" for both.

These tests were performed on the CDC 6400 at the University of Texas at Austin: a machine with a relative precision of approximately $3.6 \cdot 10^{-15}$. The matrix decomposition relative residual norms were uniformly less than $2.9 \cdot 10^{-15}$ and showed no sign of deterioration. The relative residual norms for the solution of the linear systems were bounded by $6.1 \cdot 10^{-15}$.

ACKNOWLEDGMENTS:

This work was performed partially with the support of National Science Foundation Grant MCS76-07072. The author would like to recognize Mr. Robert Renka for his assistance in preparing the code for publication.

column/row replaced	COLUP relative residual norm	ROWUP relative residual norm
(original)	$1.7 \cdot 10^{-15}$	$1.4 \cdot 10^{-15}$
1	2.1	1.7
2	2.9	2.0
3	2.8	2.0
4	2.6	1.7
5	2.3	1.9
6	2.7	1.6
7	2.6	1.7
8	2.6	1.8
9	2.7	1.7
10	2.5	1.6

TABLE 1: RELATIVE RESIDUAL NORMS FOR ORIGINAL AND UPDATED MATRIX FACTORIZATIONS

(MACHINE PRECISION = $3.6 \cdot 10^{-15}$)

REFERENCES

1. Bartels, R. H., A numerical investigation of the simplex method. Technical Report CS 104, 1968. Computer Science Department, Stanford University.

2. Bartels, R. H. and Golub, G. H., The simplex method of linear programming using LU decomposition. Comm. ACM. 12, 266-268 (1969).

3. Bartels, R. H., Golub, G. H., and Saunders, M. A., Numerical techniques in mathematical programming. Technical Report CS 162, 1970. Computer Science Department, Stanford University.

4. Bartels, R. H., Stoer, J., and Zenger, Ch., A realization of the simplex method base on a triangular decomposition, Handbook for Automatic Computation, Vol. II. Linear Algebra, J. H. Wilkinson and C. Reinsch, eds. Springer-Verlag, New York, 1971, pp. 152-190.

5. Cheney, E. W., Introduction to Approximation Theory, McGraw-Hill, New York, 1966.

6. Cline, A. K., A descent method for the uniform solution to over-determined systems of linear equations, SIAM J. on Num. Anal. 13, 293-309 (1976).

7. Dantzig, G. B., Linear programming and extensions. Princeton University Press, Princeton, 1963.

8. Stiefel, E. L., Über Diskrete und Lineare Tschebyscheff-Approximationen, Numer. Math. 1 1-28 (1959).

```
      SUBROUTINE UELAD (N, A, NA, U, L, NL, IERR)
C
C                                 ALAN KAYLOR CLINE
C                                 DEPARTMENT OF COMPUTER SCIENCES
C                                 UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NA, NL, IERR
      REAL    A(NA,N), U(1), L(NL,N)
C
C THIS SUBROUTINE PERFORMS A FACTORIZATION OF THE N * N
C MATRIX A SO THAT U = L * A, WHERE U IS UPPER TRIANGULAR
C AND L IS PERMUTED LOWER TRIANGULAR. WITH THIS FACTORIZA-
C TION IT IS POSSIBLE TO SOLVE LINEAR SYSTEMS OF THE FORM
C A * X = B (USING UELASO) OR A**T * X = B (USING UELAST),
C WHERE A**T DENOTES THE TRANSPOSE OF A. ALSO BY USING
C UELAUP, IT IS POSSIBLE TO OBTAIN A SIMILAR FACTORIZATION
C OF A MATRIX WHICH DIFFERS FROM A IN ONLY ONE COLUMN, THUS
C A SEQUENCE OF LINEAR SYSTEMS IN WHICH THE MATRICES CHANGE
C BY ONE COLUMN AT A TIME CAN BE EFFICIENTLY SOLVED. THE
C METHOD USED FOR THE FACTORIZATION IS SIMILAR TO GAUSSIAN
C ELIMINATION BY COLUMNS.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX, (N .GE. 1),
C
C   A  CONTAINS THE MATRIX TO BE FACTORED,
C
C   NA  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF A IN THE CALLING SUBPROGRAM (NA .GE. N),
C
C   U  IS AN ARRAY OF LENGTH (N * (N+1))/2 LOCATIONS TO HOLD
C      A PACKED FORM OF THE UPPER TRIANGULAR FACTOR,
C
C   L  IS AN ARRAY TO HOLD THE N BY N PERMUTED LOWER TRI-
C      ANGULAR FACTOR (THE ACTUAL PARAMETERS FOR A AND L
C      MAY COINCIDE IN WHICH CASE A WILL BE DESTROYED),
C
C AND
C
C   NL  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF L IN THE CALLING SUBPROGRAM (NL .GE. N).
C
C ON OUTPUT--
C
C   U  CONTAINS THE UPPER TRIANGULAR FACTOR,
C
C   L CONTAINS THE N BY N PERMUTED LOWER TRIANGULAR FACTOR,
C
C   IERR  CONTAINS 0 IF A IS NONSINGULAR,
C                  1 IF A IS SINGULAR,
C                  2 IF N .LE. 0 OR NA .LT. N OR NL .LT. N,
C
C AND N, A, NA, AND NL ARE UNALTERED (EXCEPT A IS DESTROYED
C IF THE ACTUAL PARAMETERS FOR A AND L COINCIDE, AS MENTION-
C ED ABOVE).
C
      NN = N
      IF (NN .LE. 0 .OR. NA .LT. NN .OR. NL .LT. NN)
     1   GO TO 13
      IERR = 0
```

```
C
C COPY A INTO L
C
      DO 1 I = 1,NN
        DO 1 J = 1,NN
    1     L(I,J) = A(I,J)
      NM1 = NN-1
      IND = 0
      IF (NN .EQ. 1) GO TO 9
C
C DECOMPOSE BY COLUMNS
C
      DO 8 K=1,NM1
        AMAXK = 0.
        DO 2 I=K,NN
          IF (ABS(L(I,K)) .LE. AMAXK) GO TO 2
          AMAXK = ABS(L(I,K))
          IMAX = I
    2     CONTINUE
        IF (AMAXK .EQ. 0.) GO TO 12
        IF (IMAX .EQ. K) GO TO 4
        DO 3 J=1,NN
          SAV = L(K,J)
          L(K,J) = L(IMAX,J)
    3     L(IMAX,J) = SAV
    4   DIAGIN = 1./L(K,K)
        IND = IND+1
        U(IND) = L(K,K)
        L(K,K) = 1.
        L(1,K) = FLOAT(IMAX)
        KM1 = K-1
        KP1 = K+1
        DO 7 I=KP1,NN
          FAC = -L(I,K)*DIAGIN
C
C ZERO OUT ELEMENTS K+1,....,N OF COLUMN K OF L
C
          IF (K.EQ. 1) GO TO 6
          DO 5 J=1,KM1
    5       L(I,J) = L(I,J)+FAC*L(K,J)
    6     L(I,K) = FAC
          DO 7 J=KP1,NN
    7       L(I,J) = L(I,J)+FAC*L(K,J)
        DO 8 J=KP1,NN
          IND = IND+1
          U(IND) = L(K,J)
    8     L(K,J) = 0.
    9 IF (L(NN,NN) .EQ. 0.) GO TO 12
      U(IND+1) = L(NN,NN)
      L(NN,NN) = 1.
      IF (NN .EQ. 1) RETURN
C
C PERMUTE THE COLUMNS OF FINAL L
C
      IMAX = IFIX(L(1,1)+.5)
      DO 11 JBK = 1,NM1
        J = NN - JBK
        K = IFIX(L(1,J)+.5)
        IF (K .EQ. J) GO TO 11
        DO 10 I=2,NN
          SAV = L(I,J)
          L(I,J) = L(I,K)
```

```
   10       L(I,K) = SAV
   11    L(1,J) = 0.
      L(1,IMAX) = 1.
      RETURN
C
C SINGULAR MATRIX
C
   12 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N, NA, OR NL
C
   13 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE UELAID (N, U, L, NL)
C
C                                     ALAN KAYLOR CLINE
C                                     DEPARTMENT OF COMPUTER SCIENCES
C                                     UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NL
      REAL    U(1), L(NL,N)
C
C THIS SUBROUTINE STORES THE U = L * A FACTORIZATION
C FOR A = IDENTITY MATRIX. THIS PROVIDES A MORE EFFICIENT
C INITIALIZATION SUBROUTINE (THAN UELAD) IN THIS SPECIAL
C CASE.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX, (N .GE. 1),
C
C   U  IS AN ARRAY OF LENGTH (N*(N+1))/2 LOCATIONS TO
C      HOLD A PACKED FORM OF THE UPPER TRIANGULAR FACTOR,
C
C   L  IS AN ARRAY TO HOLD THE N BY N FACTOR,
C
C AND
C
C   NL  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE
C       STORAGE OF L IN THE CALLING SUBPROGRAM (NL
C       .GE. N).
C
C ON OUTPUT--
C
C   U  CONTAINS THE UPPER TRIANGULAR IDENTITY FACTOR,
C
C   L  CONTAINS THE IDENTITY MATRIX,
C
C AND N AND NL ARE UNALTERED.
C
      IND = 0
      DO 3 I=1,N
        IF (I .EQ. 1) GO TO 2
        IM1 = I-1
        DO 1 J=1,IM1
    1     L(I,J) = 0.
    2   L(I,I) = 1.
        IND = IND+1
        U(IND) = 1.
        IF (I .EQ. N) RETURN
        IP1 = I+1
        DO 3 J=IP1,N
          L(I,J) = 0.
          IND = IND+1
    3     U(IND) = 0.
      RETURN
      END
```

```
      SUBROUTINE UELASO (N, U, L, NL, X, B, IERR)
C
C                                  ALAN KAYLOR CLINE
C                                  DEPARTMENT OF COMPUTER SCIENCES
C                                  UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NL, IERR
      REAL    U(1), L(NL,N), X(N), B(N)
C
C THIS SUBROUTINE USES THE U = L * A FACTORIZATION PRODUCED
C BY SUBROUTINES UELAD OR UELAUP TO SOLVE LINEAR SYSTEMS
C OF THE FORM A * X = B.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE SYSTEM,
C
C   U  CONTAINS THE PACKED UPPER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER UELAD OR UELAUP. IT HAS LENGTH
C      (N * (N+1))/2 LOCATIONS,
C
C   L  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER UELAD OR UELAUP,
C
C   NL  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF L IN THE CALLING SUBPROGRAM, (NL .GE. N),
C
C   X  IS AN ARRAY OF LENGTH N TO HOLD THE SOLUTION,
C
C AND
C
C   B  IS AN ARRAY OF LENGTH N CONTAINING THE RIGHT HAND
C      SIDE OF THE SYSTEM.
C
C ON OUTPUT--
C
C   X  CONTAINS THE SOLUTION TO THE LINEAR SYSTEM,
C
C   IERR  CONTAINS 0 IF U IS NONSINGULAR,
C                  1 IF U IS SINGULAR,
C                  2 IF N .LE. 0 .OR. NL .LT. N,
C
C AND N, U, L, NL, AND B ARE UNALTERED.
C
      NN = N
      IF (NN .LE. 0 .OR. NL .LT. NN) GO TO 6
      IERR = 0
      NP1 = NN+1
C
C LET Y = L * B
C
      DO 2 I=1,NN
        SUM = 0.
        DO 1 J=1,NN
    1     SUM = SUM+L(I,J)*B(J)
    2   X(I) = SUM
C
C SOLVE U * X = Y
C
      IND = (NN*NP1)/2
      IF (U(IND) .EQ. 0.) GO TO 5
```

```
      X(NN) = X(NN)/U(IND)
      IF (NN .EQ. 1) RETURN
      DO 4 J=2,NN
        IND = IND-J
        JBK = NP1-J
        JBK1 = JBK+1
        SUM = 0.
        DO 3 I=JBK1,NN
          IND = IND+1
    3     SUM = SUM+U(IND)*X(I)
        IND = IND-J+1
        IF (U(IND) .EQ. 0.) GO TO 5
    4   X(JBK) = (X(JBK)-SUM)/U(IND)
      RETURN
C
C SINGULAR MATRIX
C
    5 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N OR NL
C
    6 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE UELAST (N, U, L, NL, X, B, W, IERR)
C
C                                 ALAN KAYLOR CLINE
C                                 DEPARTMENT OF COMPUTER SCIENCES
C                                 UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NL, IERR
      REAL    U(1), L(NL,N), X(N), B(N), W(N)
C
C THIS SUBROUTINE USES THE U = L * A FACTORIZATION PRODUCED
C BY SUBROUTINES UELAD OR UELAUP, TO SOLVE LINEAR SYSTEMS
C OF THE FORM  A**T * X = B, WHERE A**T DENOTES THE TRANS-
C POSE OF A.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE SYSTEM,
C
C   U  CONTAINS THE PACKED UPPER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER UELAD OR UELAUP. IT HAS LENGTH
C      (N * (N+1))/2 LOCATIONS,
C
C   L  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER UELAD OR UELAUP,
C
C   NL  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF L IN THE CALLING SUBPROGRAM, (NL .GE. N),
C
C
C   X  IS AN ARRAY OF LENGTH N TO HOLD THE SOLUTION,
C
C   B  IS AN ARRAY OF LENGTH N CONTAINING THE RIGHT HAND
C      SIDE OF THE SYSTEM,
C
C AND
C
C   W  IS A WORKSPACE OF LENGTH N (THE ACTUAL PARAMETERS FOR
C      B AND W MAY COINCIDE IN WHICH CASE B WILL BE DESTROY-
C      ED).
C
C ON OUTPUT--
C
C   X  CONTAINS THE SOLUTION TO THE LINEAR SYSTEM,
C
C   IERR  CONTAINS 0 IF U IS NONSINGULAR,
C                  1 IF U IS SINGULAR,
C                  2 IF N .LE. 0 .OR. NL .LT. N,
C
C AND N, U, L, NL, AND B ARE UNALTERED (EXCEPT B IS DESTROY-
C ED IF THE ACTUAL PARAMETERS FOR B AND W COINCIDE, AS MEN-
C TIONED ABOVE).
C
      NN = N
      IF (NN .LE. 0 .OR. NL .LT. NN) GO TO 7
      IERR = 0
C
C SOLVE LOWER TRIANGULAR SYSTEM WITH U TRANSPOSED
C
      DO 1 J=1,NN
    1   W(J) = B(J)
      IND = 0
```

```
      DO 2 J=1,NN
         IND = IND+1
         IF (U(IND) .EQ. 0.) GO TO 6
         FAC = W(J)/U(IND)
         W(J) = FAC
         IF (J .EQ. NN) GO TO 3
         JP1 = J+1
         DO 2 I=JP1,NN
           IND = IND+1
    2      W(I) = W(I)-FAC*U(IND)
C
C MULTIPLY BY L TRANSPOSED
C
    3 DO 5 J=1,NN
         SUM = 0.
         DO 4 I=1,NN
    4      SUM = SUM+L(I,J)*W(I)
    5    X(J) = SUM
      RETURN
C
C SINGULAR MATRIX
C
    6 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N OR NL
C
    7 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE UELAUP (N, U, L, NL, R, K, IERR)
C
C                                  ALAN KAYLOR CLINE
C                                  DEPARTMENT OF COMPUTER SCIENCES
C                                  UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NL, K, IERR
      REAL    U(1), L(NL,N), R(N)
C
C THIS SUBROUTINE PERFORMS AN UPDATING OF THE U = L * A
C FACTORIZATION (PRODUCED BY SUBROUTINE UELAD OR BY A PRE-
C VIOUS USE OF UELAUP) CORRESPONDING TO A CHANGE OF ONE
C COLUMN IN THE MATRIX A. THE NEW MATRIX A DIFFERS FROM THE
C PREVIOUS MATRIX A IN THAT COLUMN K HAS BEEN DELETED,
C COLUMNS K+1 THROUGH N SHIFTED UP TO COLUMNS K THROUGH
C N-1, AND A NEW N-TH COLUMN ADDED. THE NEW FACTORIZATION
C IS OF THE SAME U = L * A FORM AND THUS CAN BE USED TO
C SOLVE LINEAR SYSTEMS OF THE FORM A * X = B (USING UELASO)
C OR A**T * X = B (USING UELAST), WHERE A**T DENOTES THE
C TRANSPOSE OF A.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX A, (N .GE. 1),
C
C   U  CONTAINS THE PACKED UPPER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER UELAD OR UELAUP. IT HAS
C      LENGTH (N * (N+1))/2 LOCATIONS,
C
C   L  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER UELAD OR UELAUP,
C
C   NL  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF L IN THE CALLING SUBPROGRAM, (NL .GE. N),
C
C   R  CONTAINS THE NEW COLUMN TO BE INSERTED AS THE LAST
C      COLUMN OF A,
C
C AND
C
C   K  CONTAINS THE INDEX OF THE COLUMN OF A TO BE DELETED,
C      (1 .LE. K .LE. N).
C
C ON OUTPUT--
C
C   U  CONTAINS THE UPDATED TRIANGULAR FACTOR,
C
C   L  CONTAINS THE UPDATED MATRIX FACTOR,
C
C   IERR  CONTAINS 0 IF THE NEW MATRIX A IS NONSINGULAR,
C                  1 IF THE NEW MATRIX A IS SINGULAR,
C                  2 IF N .LE. 0 OR NL .LT. N OR K .LE. 0
C                    OR K .GT. N,
C
C AND N, NL, R, AND K ARE UNALTERED. NOTICE THAT THIS
C SUBROUTINE DESTROYS THE INPUT PARAMETERS U AND L. IF THEY
C ARE TO BE SAVED, THEY MUST BE COPIED IN THE CALLING SUB-
C PROGRAM.
C
      NN = N
      IF (K .LT. 1 .OR. K .GT. NN .OR. NN .LT. 0
```

```
     1    .OR. NL .LT. NN) GO TO 14
      IERR = 0
      K1 = K
      NM1 = NN-1
      K1P1 = K1+1
      K1P2 = K1+2
      NM1MK = NM1-K1
      IND = 0
C
C SHIFT ROWS UP IN COLUMNS 1,...,K1 AND INTRODUCE NEW
C ELEMENTS IN LAST COLUMN
C
      DO 4 KK=1,K1
        IND1 = IND+K1P2-KK
        IND = IND1+NM1MK
        IF (K1 .EQ. NN) GO TO 2
        DO 1 J=IND1,IND
    1     U(J-1) = U(J)
    2   SUM  = 0.
        DO 3 I=1,NN
    3     SUM = SUM+L(KK,I)*R(I)
    4   U(IND) = SUM
C
C CONTINUE DECOMPOSITION WITH ROW K1+1, SHIFT, AND ADD
C NEW ELEMENTS IN LAST COLUMN
C
      IF (K1 .EQ. NN) GO TO 12
      INDM1 = IND-1
      DO 11 KK=K1P1,NN
        KM1 = KK-1
        INDP1 = IND+1
        IND = INDM1-(N-KK)
        JDELT = INDP1-IND
        SUM = 0.
        DO 5 I=1,NN
    5     SUM = SUM+L(KK,I)*R(I)
        IF (ABS(U(IND)) .GE. ABS(U(INDP1))) GO TO 8
C
C PERMUTE COLUMNS KK-1 AND KK OF U AND L
C
        DO 6 J=IND,INDM1
          JPJDEL = J+JDELT
          SAV = U(JPJDEL)
          U(JPJDEL) = U(J)
    6     U(J) = SAV
        SAV = U(INDM1+1)
        U(INDM1+1) = SUM
        SUM = SAV
        DO 7 J=1,NN
          SAV = L(KK,J)
          L(KK,J) = L(KM1,J)
    7     L(KM1,J) = SAV
    8   IF (U(IND) .EQ. 0.) GO TO 13
        FAC = -U(INDP1)/U(IND)
        IND = INDM1+JDELT
        INDM1 = IND-1
        IF (KK .EQ. NN) GO TO 10
        DO 9 J=INDP1,INDM1
          JMJDEL = J-JDELT
    9     U(J) = U(J+1)+FAC*U(JMJDEL+1)
   10   INDMJD = IND-JDELT
        U(IND) = SUM+FAC*U(INDMJD+1)
```

```
        DO 11 J=1,NN
   11      L(KK,J) = L(KK,J)+FAC*L(KM1,J)
   12 IF (U(IND) .EQ. 0.) GO TO 13
      RETURN
C
C SINGULAR MATRIX
C
   13 IERR = 1
      RETURN
C
C IMPROPER INPUT OF K, N, OR NL
C
   14 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE LEAUD (N, A, NA, L, U, NU, IERR)
C
C                                  ALAN KAYLOR CLINE
C                                  DEPARTMENT OF COMPUTER SCIENCES
C                                  UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NA, NU, IERR
      REAL    A(NA,N), L(1), U(NU,N)
C
C THIS SUBROUTINE PERFORMS A FACTORIZATION OF THE N * N
C MATRIX A SO THAT L = A * U, WHERE L IS LOWER TRIANGULAR
C AND U IS PERMUTED UPPER TRIANGULAR. WITH THIS FACTORIZA-
C TION IT IS POSSIBLE TO SOLVE LINEAR SYSTEMS OF THE FORM
C A * X = B (USING LEAUSO) OR A**T * X = B (USING LEAUST),
C WHERE A**T DENOTES THE TRANSPOSE OF A. ALSO BY USING
C LEAUUP, IT IS POSSIBLE TO OBTAIN A SIMILAR FACTORIZATION
C OF A MATRIX WHICH DIFFERS FROM A IN ONLY ONE ROW, THUS
C A SEQUENCE OF LINEAR SYSTEMS IN WHICH THE MATRICES CHANGE
C BY ONE ROW AT A TIME CAN BE EFFICIENTLY SOLVED. THE METHOD
C USED FOR THE FACTORIZATION IS SIMILAR TO GAUSSIAN ELIM-
C INATION BY COLUMNS.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX, (N .GE. 1),
C
C   A  CONTAINS THE MATRIX TO BE FACTORED,
C
C   NA  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF A IN THE CALLING SUBPROGRAM (NA .GE. N),
C
C   L  IS AN ARRAY OF LENGTH (N * (N+1))/2 LOCATIONS TO HOLD
C      A PACKED FORM OF THE LOWER TRIANGULAR FACTOR,
C
C   U  IS AN ARRAY TO HOLD THE N BY N PERMUTED UPPER TRI-
C      ANGULAR FACTOR (THE ACTUAL PARAMETERS FOR A AND U
C      MAY COINCIDE IN WHICH CASE A WILL BE DESTROYED),
C
C AND
C
C   NU  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF U IN THE CALLING SUBPROGRAM (NU .GE. N).
C
C ON OUTPUT--
C
C   L  CONTAINS THE LOWER TRIANGULAR FACTOR,
C
C   U CONTAINS THE N BY N PERMUTED UPPER TRIANGULAR FACTOR,
C
C   IERR  CONTAINS 0 IF A IS NONSINGULAR,
C                  1 IF A IS SINGULAR,
C                  2 IF N .LE. 0 OR NA .LT. N OR NU .LT. N,
C
C AND N, A, NA, AND NU ARE UNALTERED (EXCEPT A IS DESTROYED
C IF THE ACTUAL PARAMETERS FOR A AND U COINCIDE, AS MENTION-
C ED ABOVE).
C
      NN = N
      IF (NN .LE. 0 .OR. NA .LT. NN .OR. NU .LT. NN)
     1   GO TO 13
      IERR = 0
```

```
C
C COPY A INTO U
C
      DO 1 I = 1,NN
        DO 1 J = 1,NN
    1     U(I,J) = A(I,J)
      NM1 = NN-1
      IND = 0
      IF (NN .EQ. 1) GO TO 9
C
C DECOMPOSE BY ROWS
C
      DO 8 K=1,NM1
        AMAXK = 0.
        DO 2 J=K,NN
          IF (ABS(U(K,J)) .LE. AMAXK) GO TO 2
          AMAXK = ABS(U(K,J))
          JMAX = J
    2     CONTINUE
        IF (AMAXK .EQ. 0.) GO TO 12
        IF (JMAX .EQ. K) GO TO 4
        DO 3 I=1,NN
          SAV = U(I,K)
          U(I,K) = U(I,JMAX)
    3     U(I,JMAX) = SAV
    4   DIAGIN = 1./U(K,K)
        IND = IND+1
        L(IND) = U(K,K)
        U(K,K) = 1.
        U(K,1) = FLOAT(JMAX)
        KM1 = K-1
        KP1 = K+1
        DO 7 J=KP1,NN
          FAC = -U(K,J)*DIAGIN
C
C ZERO OUT ELEMENTS K+1,...,N OF ROW K OF U
C
          IF (K.EQ. 1) GO TO 6
          DO 5 I=1,KM1
    5       U(I,J) = U(I,J)+FAC*U(I,K)
    6     U(K,J) = FAC
          DO 7 I=KP1,NN
    7       U(I,J) = U(I,J)+FAC*U(I,K)
        DO 8 I=KP1,NN
          IND = IND+1
          L(IND) = U(I,K)
    8     U(I,K) = 0.
    9 IF (U(NN,NN) .EQ. 0.) GO TO 12
      L(IND+1) = U(NN,NN)
      U(NN,NN) = 1.
      IF (NN .EQ. 1) RETURN
C
C PERMUTE THE ROWS OF FINAL U
C
      JMAX = IFIX(U(1,1)+.5)
      DO 11 IBK = 1,NM1
        I = NN - IBK
        K = IFIX(U(I,1)+.5)
        IF (K .EQ. I) GO TO 11
        DO 10 J=2,NN
          SAV = U(I,J)
          U(I,J) = U(K,J)
```

```
   10      U(K,J) = SAV
   11    U(I,1) = 0.
      U(JMAX,1) = 1.
      RETURN
C
C SINGULAR MATRIX
C
   12 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N, NA, OR NU
C
   13 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE LEAUID (N, L, U, NU)
C
C                                   ALAN KAYLOR CLINE
C                                   DEPARTMENT OF COMPUTER SCIENCES
C                                   UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NU
      REAL    L(1), U(NU,N)
C
C THIS SUBROUTINE STORES THE L = A * U FACTORIZATION
C FOR A = IDENTITY MATRIX. THIS PROVIDES A MORE EFFICIENT
C INITIALIZATION SUBROUTINE (THAN LEAUD) IN THIS SPECIAL
C CASE.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX, (N .GE. 1),
C
C   L  IS AN ARRAY OF LENGTH (N*(N+1))/2 LOCATIONS TO
C      HOLD A PACKED FORM OF THE LOWER TRIANGULAR FACTOR.
C
C   U  IS AN ARRAY TO HOLD THE N BY N FACTOR,
C
C AND
C
C   NU  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE
C       STORAGE OF U IN THE CALLING SUBPROGRAM (NU
C       .GE. N).
C
C ON OUTPUT--
C
C   L  CONTAINS THE LOWER TRIANGULAR IDENTITY FACTOR,
C
C   U  CONTAINS THE IDENTITY MATRIX,
C
C AND N AND NU ARE UNALTERED.
C
      IND = 0
      DO 3 I=1,N
        IF (I .EQ. 1) GO TO 2
        IM1 = I-1
        DO 1 J=1,IM1
    1     U(I,J) = 0.
    2   U(I,I) = 1.
        IND = IND+1
        L(IND) = 1.
        IF (I .EQ. N) RETURN
        IP1 = I+1
        DO 3 J=IP1,N
          U(I,J) = 0.
          IND = IND+1
    3     L(IND) = 0.
      RETURN
      END
```

```
      SUBROUTINE LEAUSO (N, L, U, NU, X, B, W, IERR)
C
C                                 ALAN KAYLOR CLINE
C                                 DEPARTMENT OF COMPUTER SCIENCES
C                                 UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NU, IERR
      REAL    L(1), U(NU,N), X(N), B(N), W(N)
C
C THIS SUBROUTINE USES THE L = A * U FACTORIZATION PRODUCED
C BY SUBROUTINES LEAUD OR LEAUUP, TO SOLVE LINEAR SYSTEMS
C OF THE FORM  A * X = B.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE SYSTEM,
C
C   L  CONTAINS THE PACKED LOWER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER LEAUD OR LEAUUP. IT HAS LENGTH
C      (N * (N+1))/2 LOCATIONS,
C
C   U  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER LEAUD OR LEAUUP,
C
C   NU  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF U IN THE CALLING SUBPROGRAM, (NU .GE. N),
C
C
C   X  IS AN ARRAY OF LENGTH N TO HOLD THE SOLUTION,
C
C   B  IS AN ARRAY OF LENGTH N CONTAINING THE RIGHT HAND
C      SIDE OF THE SYSTEM,
C
C AND
C
C   W  IS A WORKSPACE OF LENGTH N (THE ACTUAL PARAMETERS FOR
C      B AND W MAY COINCIDE IN WHICH CASE B WILL BE DESTROY-
C      ED).
C
C ON OUTPUT--
C
C   X  CONTAINS THE SOLUTION TO THE LINEAR SYSTEM,
C
C   IERR  CONTAINS 0 IF L IS NONSINGULAR,
C                  1 IF L IS SINGULAR,
C                  2 IF N .LE. 0 .OR. NU .LT. N,
C
C AND N, L, U, NU, AND B ARE UNALTERED (EXCEPT B IS DESTROY-
C ED IF THE ACTUAL PARAMETERS FOR B AND W COINCIDE, AS MEN-
C TIONED ABOVE).
C
      NN = N
      IF (NN .LE. 0 .OR. NU .LT. NN) GO TO 7
      IERR = 0
C
C SOLVE L * W = B
C
      DO 1 I=1,NN
    1   W(I) = B(I)
      IND = 0
      DO 2 I=1,NN
```

```
      IND = IND+1
      IF (L(IND) .EQ. 0.) GO TO 6
      FAC = W(I)/L(IND)
      W(I) = FAC
      IF (I .EQ. NN) GO TO 3
      IP1 = I+1
      DO 2 J=IP1,NN
        IND = IND+1
    2   W(J) = W(J)-FAC*L(IND)
C
C LET X = U * W
C
    3 DO 5 I=1,NN
        SUM = 0.
        DO 4 J=1,NN
    4     SUM = SUM+U(I,J)*W(J)
    5   X(I) = SUM
      RETURN
C
C SINGULAR MATRIX
C
    6 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N OR NU
C
    7 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE LEAUST (N, L, U, NU, X, B, IERR)
C
C                               ALAN KAYLOR CLINE
C                               DEPARTMENT OF COMPUTER SCIENCES
C                               UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NU, IERR
      REAL    L(1), U(NU,N), X(N), B(N)
C
C THIS SUBROUTINE USES THE L = A * U FACTORIZATION PRODUCED
C BY SUBROUTINES LEAUD OR LEAUUP TO SOLVE LINEAR SYSTEMS
C OF THE FORM A**T * X = B, WHERE A**T DENOTES THE TRANSPOSE
C OF A.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE SYSTEM,
C
C   L  CONTAINS THE PACKED LOWER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER LEAUD OR LEAUUP. IT HAS LENGTH
C      (N * (N+1))/2 LOCATIONS,
C
C   U  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER LEAUD OR LEAUUP,
C
C   NU  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF U IN THE CALLING SUBPROGRAM, (NU .GE. N),
C
C   X  IS AN ARRAY OF LENGTH N TO HOLD THE SOLUTION,
C
C AND
C
C   B  IS AN ARRAY OF LENGTH N CONTAINING THE RIGHT HAND
C      SIDE OF THE SYSTEM.
C
C ON OUTPUT--
C
C   X  CONTAINS THE SOLUTION TO THE LINEAR SYSTEM,
C
C   IERR  CONTAINS 0 IF L IS NONSINGULAR,
C                  1 IF L IS SINGULAR,
C                  2 IF N .LE. 0 .OR. NU .LT. N,
C
C AND N, L, U, NU, AND B ARE UNALTERED.
C
      NN = N
      IF (NN .LE. 0 .OR. NU .LT. NN) GO TO 6
      IERR = 0
      NP1 = NN+1
C
C MULTIPLY B BY U TRANSPOSED
C
      DO 2 I=1,NN
        SUM = 0.
        DO 1 J=1,NN
    1     SUM = SUM+U(J,I)*B(J)
    2   X(I) = SUM
C
C SOLVE UPPER TRIANGULAR SYSTEM WITH L TRANSPOSED
C
      IND = (NN*NP1)/2
```

```
      IF (L(IND) .EQ. 0.) GO TO 5
      X(NN) = X(NN)/L(IND)
      IF (NN .EQ. 1) RETURN
      DO 4 I=2,NN
        IND = IND-I
        IBK = NP1-I
        IBK1 = IBK+1
        SUM = 0.
        DO 3 J=IBK1,NN
          IND = IND+1
    3     SUM = SUM+L(IND)*X(J)
        IND = IND-I+1
        IF (L(IND) .EQ. 0.) GO TO 5
    4   X(IBK) = (X(IBK)-SUM)/L(IND)
      RETURN
C
C SINGULAR MATRIX
C
    5 IERR = 1
      RETURN
C
C IMPROPER INPUT OF N OR NU
C
    6 IERR = 2
      RETURN
      END
```

```
      SUBROUTINE LEAUUP (N, L, U, NU, R, K, IERR)
C
C                                    ALAN KAYLOR CLINE
C                                    DEPARTMENT OF COMPUTER SCIENCES
C                                    UNIVERSITY OF TEXAS AT AUSTIN
C
      INTEGER N, NU, K, IERR
      REAL    L(1), U(NU,N), R(N)
C
C THIS SUBROUTINE PERFORMS AN UPDATING OF THE L = A * U
C FACTORIZATION (PRODUCED BY SUBROUTINE LEAUD OR BY A PRE-
C VIOUS USE OF LEAUUP) CORRESPONDING TO A CHANGE OF ONE ROW
C IN THE MATRIX A. THE NEW MATRIX A DIFFERS FROM THE PREV-
C IOUS MATRIX A IN THAT ROW K HAS BEEN DELETED, ROWS K+1
C THROUGH N SHIFTED UP TO ROWS K THROUGH N-1, AND A NEW
C N-TH ROW ADDED. THE NEW FACTORIZATION IS OF THE SAME
C L = A * U FORM AND THUS CAN BE USED TO SOLVE LINEAR SYS-
C TEMS OF THE FORM A * X = B (USING LEAUSO) OR A**T * X = B
C (USING LEAUST), WHERE A**T DENOTES THE TRANSPOSE OF A.
C
C ON INPUT--
C
C   N  CONTAINS THE ORDER OF THE MATRIX A, (N .GE. 1),
C
C   L  CONTAINS THE PACKED LOWER TRIANGULAR FACTOR AS
C      PRODUCED BY EITHER LEAUD OR LEAUUP. IT HAS
C      LENGTH (N * (N+1))/2 LOCATIONS,
C
C   U  CONTAINS THE N BY N MATRIX FACTOR AS PRODUCED BY
C      EITHER LEAUD OR LEAUUP,
C
C   NU  CONTAINS THE ROW DIMENSION SPECIFIED FOR THE STORAGE
C       OF U IN THE CALLING SUBPROGRAM. (NU .GE. N),
C
C   R  CONTAINS THE NEW ROW TO BE INSERTED AS THE LAST ROW
C      OF A,
C
C AND
C
C   K  CONTAINS THE INDEX OF THE ROW OF A TO BE DELETED,
C      (1 .LE. K .LE. N).
C
C ON OUTPUT--
C
C   L  CONTAINS THE UPDATED TRIANGULAR FACTOR,
C
C   U  CONTAINS THE UPDATED MATRIX FACTOR,
C
C   IERR  CONTAINS 0 IF THE NEW MATRIX A IS NONSINGULAR,
C                  1 IF THE NEW MATRIX A IS SINGULAR.
C                  2 IF N .LE. 0 OR NU .LT. N OR K .LE. 0
C                    OR K .GT. N,
C
C AND N, NU, R, AND K ARE UNALTERED. NOTICE THAT THIS
C SUBROUTINE DESTROYS THE INPUT PARAMETERS L AND U. IF THEY
C ARE TO BE SAVED, THEY MUST BE COPIED IN THE CALLING SUB-
C PROGRAM.
C
      NN = N
      IF (K .LT. 1 .OR. K .GT. NN .OR. NN .LT. 0
     1    .OR. NU .LT. NN) GO TO 14
```

```
      IERR = 0
      K1 = K
      NM1 = NN-1
      K1P1 = K1+1
      K1P2 = K1+2
      NM1MK = NM1-K1
      IND = 0
C
C SHIFT ROWS UP IN COLUMNS 1,...,K1 AND INTRODUCE NEW
C ELEMENTS IN LAST ROW
C
      DO 4 KK=1,K1
        IND1 = IND+K1P2-KK
        IND = IND1+NM1MK
        IF (K1 .EQ. NN) GO TO 2
        DO 1 I=IND1,IND
    1     L(I-1) = L(I)
    2   SUM  = 0.
        DO 3 J=1,NN
    3     SUM = SUM+U(J,KK)*R(J)
    4   L(IND) = SUM
C
C CONTINUE DECOMPOSITION WITH COLUMN K1+1, SHIFT, AND ADD
C NEW ELEMENTS IN LAST ROW
C
      IF (K1 .EQ. NN) GO TO 12
      INDM1 = IND-1
      DO 11 KK=K1P1,NN
        KM1 = KK-1
        INDP1 = IND+1
        IND = INDM1-(N-KK)
        IDELT = INDP1-IND
        SUM = 0.
        DO 5 J=1,NN
    5     SUM = SUM+U(J,KK)*R(J)
        IF (ABS(L(IND)) .GE. ABS(L(INDP1))) GO TO 8
C
C PERMUTE COLUMNS KK-1 AND KK OF L AND U
C
        DO 6 I=IND,INDM1
          IPIDEL = I+IDELT
          SAV = L(IPIDEL)
          L(IPIDEL) = L(I)
    6     L(I) = SAV
        SAV = L(INDM1+1)
        L(INDM1+1) = SUM
        SUM = SAV
        DO 7 I=1,NN
          SAV = U(I,KK)
          U(I,KK) = U(I,KM1)
    7     U(I,KM1) = SAV
    8   IF (L(IND) .EQ. 0.) GO TO 13
        FAC = -L(INDP1)/L(IND)
        IND = INDM1+IDELT
        INDM1 = IND-1
        IF (KK .EQ. NN) GO TO 10
        DO 9 I=INDP1,INDM1
          IMIDEL = I-IDELT
    9     L(I) = L(I+1)+FAC*L(IMIDEL+1)
   10   INDMID = IND-IDELT
        L(IND) = SUM+FAC*L(INDMID+1)
        DO 11 I=1,NN
```

```
   11     U(I,KK) = U(I,KK)+FAC*U(I,KM1)
   12 IF (L(IND) .EQ. 0.) GO TO 13
      RETURN
C
C SINGULAR MATRIX
C
   13 IERR = 1
      RETURN
C
C IMPROPER INPUT OF K, N, OR NU
C
   14 IERR = 2
      RETURN
      END
```