NASA CR-
147821

(NASA-CR-147821) ON THE COMPUTATION AND UPDATING OF THE MODIFIED CHOLESKY DECOMPOSITION OF A COVARIANCE MATRIX (Rice Univ.) 27 p HC $4.00 CSCL 09B

N76-27901

Unclas
G3/61 44563

ICSA

INSTITUTE FOR COMPUTER SERVICES AND APPLICATIONS

RICE UNIVERSITY

275-025-024

On the Computation and Updating of the Modified Cholesky Decomposition of a Covariance Matrix

by

D. L. Van Rooy
I C S A
Rice University

ABSTRACT:

In this paper we discuss three known methods for obtaining and updating the modified Cholesky decomposition (MCD) for the particular case of a covariance matrix when one is given only the original data. These methods are the standard method of forming the covariance matrix K then solving for the MCD, L & D (where $K = LDL^T$); a method based on Householder reflections; and lastly, a method employing the composite-t algorithm developed by Fletcher and Powell (Math Comp., 28, 1974, pp. 1067-1087). For many cases in the analysis of remotely sensed data, the composite-t method is the superior method despite the fact that it is the slowest one, since (1) the relative amount of time computing MCD's is often quite small, (2) the stability properties of it are the best of the three, and (3) it affords an efficient and numerically stable procedure for updating the MCD. The properties of these methods are discussed and FORTRAN programs implementing these algorithms are listed in an appendix.

Institute for Computer Services and Applications
Rice University
Houston, TX 77001
May, 1976

This work was supported under NASA contract NAS 9-12776

I. Introduction

In digital processing of remotely sensed data, as well as many other areas employing multivariate analysis, solutions to many of the problems are formulated in terms of covariance matrices. Often these solutions are expressed in terms of linear transformations involving a covariance matrix or its inverse. In these and other cases, it is often more sound computationally for one to employ the Cholesky or modified Cholesky decomposition of the covariance matrix rather than the original matrix itself(1). Even in cases where the original covariance matrix is to be modified by the addition or deletion of data, it still may be computationally prudent to utilize these decompositions.

The purpose of this paper is to discuss methods for computing and updati the modified Cholesky decomposition (MCD) of a covariance matrix. These methods will be examined from the point of view of their ability to update the MCD when data is to be added or deleted, as well as computational efficiency and numerical stability. In particular, three methods for accomplishing the above will be discussed:

1) Standard--one computes the covariance matrix from the defining equations and then calculates the MCD of it.

2) Householder--here one directly computes the MCD of the covariance matrix from the data using Householder reflections(2).

3) Composite-t--this method was developed by Fletcher and powell(3) from work previously done by Bennett(4) and Gentleman(5). The method essentially uses Givens rotations(2) of

the data one point at a time to directly compute the MCD of the covariance matrix. Updating is straightforward and efficient.

Table 1 summarizes the properties of each of these methods. Here n is the dimension of the data and m is the total number of data vectors. Though numerical stability may not play much of a role in most cases, the times when it does, may occur without the user being aware of any difficulties. Thus this situation may lead to erroneous interpretations of the results. A method for computing the MCD should be chosen with this in mind. Also, in many areas of digital processing of remotely sensed data, the actual computation time for computing the MCD is inconsequential, so that the composite-t method with its superior stability, may be optimal despite its relatively slow performance. The added benefit of an efficient and stable updating capability may also be of value.

	Number of Multiplies* to Compute the MCD	Data Storage Requirements* (Words)	Stability	Remarks on Updating
Standard Method	$\frac{mn^2}{2} + \frac{n^3}{3}$	n	Poor	Require $\sim \frac{2n^3}{3}$ multiplies and is unstable
Householder	$mn^2 - \frac{n^3}{6}$	mn	Good	Updating is stable but slower than Composite-t
Composite-t	$mn(n+7)$	n	Excellent	Requires $\sim \frac{3n^2}{2}$ to n^2 multiplies and is stable

Table 1

Comparison of the Three Methods for Computing and Modifying the MCD

*These values are approximate; we have assumed $m >> n > 1$

II. Methods for Computing and Updating the MCD of the Covariance Matrix.

Given X the n x m data matrix containing m multivariate n-dimensional vectors, the mean vector u is defined by

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{ji} \qquad j=1,2,\ldots,n \quad (1)$$

and the covariance matrix K, by

$$K = \frac{1}{m-1} \sum_{i=1}^{m} (x_{*i}-\mu)(x_{*i}-\mu)^T \quad (2)$$

where x_{*i} is the i^{th} data vector and ^{T} denotes transpose. K is symmetric and positive semi-definite. (It should also be noted that K is singular if $m < n+1$). The MCD of K s given by $K = LDL^T$ where D is diagonal with positive diagonal entries and L is unit lower triangular.

A. Standard Method:

The usual method for computing K comes from rewriting (2) using (1) to yield

$$k_{ij} = \frac{1}{m-1} \left[\sum_{\ell=1}^{m} x_{i\ell} x_{j\ell} - \frac{1}{m} \left(\sum_{\ell=1}^{m} x_{i\ell}\right)\left(\sum_{\ell=1}^{m} x_{j\ell}\right) \right]$$

The MCD of this is then computed (see e.g. ref. 6). This method requires (we consider cases where $m >> n > 1$) approximately $\frac{m n^2}{2}$ multiplies to compute K and another $n^3/6$ multiplies to compute the elements of L and D. Though K itself may be computed with acceptable precision, functions involving K amy be evaluated quite inaccurately since matrix products of the form YY^T may be quite ill-conditioned[5]. Updating L and D in this manner is time consuming since one must first update K and then recompute L and D. Another method for updating L and D directly will be discussed in section C.

B. Householder:

One way to avoid problem of the possible ill conditioning of K is to compute L and D directly from the data. This may be done by using Householder reflections on the data matrix as follows.

Let M be the $n \times m$ matrix

$$M = \begin{pmatrix} u_1 & u_1 & u_1 & \cdots, & u_1 \\ u_2 & u_2 & & & \\ \vdots & & & & \\ u_n & \cdots & \cdots & \cdots & u_n \end{pmatrix}$$

Then eq. (1) can be written

$$K = \frac{1}{m-1}(X-M)(X-M)^T$$

If we then let

$$X - M = R^T Q \tag{3}$$

where R is $m \times n$ upper triangular and Q is $m \times m$ and orthogonal, we may write

$$K = \frac{1}{m-1} R^T Q Q^T R = \frac{1}{m-1} R^T R$$
$$= L D L^T$$

where L and D are the MCD of K as before with

$$L D^{\frac{1}{2}} = \frac{1}{\sqrt{m-1}} R^T$$

Rewriting (3), we have

$$Q^T (X - M)^T = R$$

We may then write Q^T as a product of n Householder reflections (see e.g. ref. 2)

$$Q^T = P_n \; P_{n-1} \; \cdots \; P_1$$

where P_i annihilates all elements from $i+1$ to n of the i^{th} column, changes the ii^{th} element and does not change elements 1 to i-1.

The algorithm, then for computing L and D in this fashion is:

<u>Householder MCD Algorithm</u>

1. Compute $u_i = \frac{1}{m} \sum_{\ell=1}^{m} x_{i\ell}$ $\qquad i = 1, 2, \ldots, n$

2. Form $t_{ji} = x_{ij} - u_i$ $\qquad i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$

3. For $i = 1, 2, \ldots, n$, compute

   a) $\alpha = \text{sgn}\,(t_{ii}) * \left(\sum_{j=i}^{m} t_{ji}^2 \right)^{\frac{1}{2}}$

   b) $u = (0, 0, \ldots, t_{ii} + \alpha_i, t_{i+1,i}, \ldots, t_{m,i})$

   c) $\beta = \alpha u$

d) $t_{j\ell} \leftarrow t_{j\ell} - \frac{1}{\beta} u_j \sum_{k=i}^{m} u_k t_{k\ell}$

for $j=i, \quad i+1, \ldots, m$

and for $\ell = i+1, \quad i+2, \ldots, n$

(N.B. "$\leftarrow$" denotes the replacement operation)

e) $t_{ii} \leftarrow -\alpha$

4. For $i=1, 2, \ldots, n$, compute

a) $d_i = t_{ii}^2 \;/\; (m-1)$

b) For $j=1, 2, \ldots, i-1$, compute

$\ell_{ij} = t_{ji} \;/\; t_{jj}$

It should be noted that the elements of T, X, L, and D can occupy the same storage locations (though X will then be lost) and that steps 3 and 4 can be combined.

This algorithm requires approximately $mn^2 - \frac{n^3}{3}$ multiplications, which may be (for m much larger than n) up to a factor of two times slower than the standard method. However, here the stability problems have been alleviated due to the use of orthogonal transformations. Storage considerations may be a problem with this algorithm, since it functions most efficiently only if the entire data matrix is in core. A sequential version of this algorithm may be used (see Chapter 27 of ref. 7) which alleviates the storage requirements and provides for an updating capability, but at a cost of increased computation time. The next method (composite-t), however, yields a more efficient and stable algorithm.

C. Composite-t:

This method is based on an algorithm developed by Fletcher and Powell[3] as a more numerically accurate extension of algorithms developed independently by Bennett[4] and Gentleman[5]. Essentially, Givens rotations[2] are used to directly compute the MCD from the data as in the previous method. Instead of working with all of the original data at once as in the Householder method, this algorithm updates the MCD as each data vector is processed. In this section, we will present two algorithms which employ the composite-t method to calculate the MCD and update it.

The generalized composite-t algorithm for a rank one update of L and D of the form

$$LDL^T \leftarrow LDL^T + \frac{1}{t_1} zz^T$$

for a positive semi-definite matrix where we assume $t_1 \neq 0$, that the rank of the matrix never decreases, and that $t_1 < 0$ only if D has full rank (i.e. X has full rank), is:

Rank-one Composite-t Algorithm

1. If $t_1 > 0$ go to 5

2. Solve $Lv = z$ for v
(i.e. $v_1 = z_1$, $v_i = z_i - \sum_{j=1}^{i-1} \ell_{ij} v_j$,

$i = 2, 3, \ldots, n$)

3. For $i = 1, 2, \ldots, n$, compute
$t_{i+1} = t_i + v_i^2 / d_i$

4. If any $t_{i+1} \geq 0$, then

   a) set $t_{n+1} = \epsilon t_1$, where ϵ is the machine precision

   b) for $i=n, n-1, \ldots, 1$

   $$t_i = t_{i+1} - v_i^2 / d_i$$

5. For $i=1, 2, \ldots, n$

   a) $v_i = z_i$

   b) If $d_i \neq 0$ go to substep c)

   (1) if $v_i \neq 0$ to to sub-substep (4)

   (2) $t_{i+1} = t_i$

   (3) go to substep k)

   (4) $d_i = v_i^2 / t_i$, $\ell_{*i} = z_* / v_i$

   (5) calculation complete

   c) If $t_1 > 0$ then $t_{i+1} = t_i + v_i^2 / d_i$

   d) $\alpha_i = t_{i+1} / t_i$

   e) $d_i \leftarrow d_i \alpha_i$

   f) If $i=n$ then calculation is complete

   g) $\beta_i = (\alpha_i v_i / d_i) / t_{i+1}$

   h) If $\alpha_i > 4$ then

   1) $\gamma_i = t_i / t_{i+1}$

   2) for $j = i+1, i+2, \ldots, n$

   $xx = \gamma_i \ell_{ji} + \beta_i z_j$

   $z_j \leftarrow z_j - v_i \ell_{ji}$

   $\ell_{ji} \leftarrow xx$

   3) go to substep k)

i) $z_* \leftarrow z_* - v_i \, \ell_{*i}$

j) $\ell_{*i} \leftarrow \ell_{*i} + \beta_i \, z_*$

k) Return to substep a)

Note that only the last $n - i$ components of ℓ_{*i} ($\ell_{ii} = 1$) and $n - i + 1$ components of z_* need be involved in these computations. Note also that the number of multiplications performed in this algorithm is data dependent. A detailed error analysis of this algorithm is given in ref. 3 showing this algorithm to be quite stable.

To employ this algorithm, we must first rewrite eq. 2 expressing the covariance matrix, $K^{(r+1)}$ associated with the first $r + 1$ data vectors to that, $K^{(r)}$, associated with the first r data vectors:

$$\tilde{K}^{(r+1)} = \tilde{K}^{(r)} + \frac{1}{r+1} \, z^{(r)} \, z^{(r)^T}$$

where

$$K^{(r+1)} = \frac{1}{r} \, \tilde{K}^{(r+1)}$$

and

$$z_*^{(r)} = \frac{1}{\sqrt{r}} \sum_{j=1}^{r} x_{*j} - \sqrt{r} \, x_{*\, r+1}$$

$$\equiv \frac{1}{\sqrt{r}} \, s^{(r)} - \sqrt{r} \, x_{*\, r+1}$$

Given m data points, then the algorithm for computing the MCD of $K^{(r+1)}$ is:

Composite-t MCD Algorithm

1. Set $L = I_n$, $D = 0$, and $s_* = x_{*1}$

2. For $r = 2, 3, \ldots, m$
   a) set $t_1 = r$
   b) for $i = 1, 2, \ldots, n$, compute
   $$z_i = s_i / \sqrt{r-1} - \sqrt{r-1}\, x_{ir}$$
   $$s_i \leftarrow s_i + x_{ir}$$
   (N.B. For $r = m$ $s_i = m\, u_i$)
   c) use Rank-one Composite-t Algorithm to update L and D (note that $t_1 > 0$)

3. For $i = 1, 2, \ldots, n$
   $$d_i \leftarrow d_i / (m-1)$$

The number of multiplications involved in this algorithm when $m >> n$ is approximately $mn^2 + 7mn$. m square roots are also necessary.

After L and D have been computed, data vectors may be added or deleted yielding a modified L and D by using the following algorithm

Composite-t Update Algorithm

1. Compute $d_* \leftarrow d_* * (\tilde{m} - 1)$ where $\tilde{m}$ is the net number of points used to compute L and D. If s is unavailable, it may be computed from $s_* = \tilde{m}\, u_*$

2. If a point is being added, set $t_1 = \tilde{m}+1$, $y = \sqrt{\tilde{m}}$, and $q = \tilde{m}+1$

3. If a point is being deleted,

   a) set $t_1 = -\tilde{m}$, $y = \sqrt{\tilde{m}-1}$, and $q = \tilde{m}-1$

   b) $s_* \leftarrow s_* - \alpha_*$ where α_* is the data vector to be added or deleted

4. Compute $z_* = s_* / y - y\,\alpha_*$ If a point is to be added, set $s_* \leftarrow s_* + \alpha_*$

5. Use Rank-one Composite-t Algorithm

6. Compute $d_* \leftarrow d_* / (q-1)$ and set $\tilde{m} = q$

This algorithm requires approximately $3n^2/2$ multiplications to delete a data point and n^2 to add a data point.

III. Numerical Examples

In this section, we present some numerical examples illustrating the properties of the algorithms discussed in the previous section. Listings of the programs used to implement these algorithms are given in the appendix.

Using some 12 dimensional pseudo-random data of 100 points, we tested all three methods on an IBM 370/155. All algorithms yielded the same results to ~ 5 decimal places. The times for computing the MCD's by each method are: standard method—

.59 sec., Householder—.78 sec., and composite-t—1.02 sec. Note that the order of these timings is as predicted, but due to differences in bookkeeping and other operations involved in each method, these timings do not follow the ratios of the number of multiplications in Table I.

To test the stability of the three methods for computing the MCD, we used the data matrix

$$X = \begin{pmatrix} 1 & -.999 & -.001 & 0. \\ 1 & -.99 & -.01 & 0. \\ 1 & -1. & .001 & -.001 \end{pmatrix}$$

which generates an ill-conditioned covariance matrix. The resultant L's and D's for each method and the exact L and D are given below (to six digits): (We use the subscripts E, S, HR, and CT for exact, standard, Householder, and composite-t methods, respectively. Also only the below diagonal elements of L are given, in the order ℓ_{21}, ℓ_{31}, ℓ_{32})

$$L_E = (.995505,\ 1.00050,\ -.185148)$$
$$L_S = (.995504,\ 1.00050,\ -.180695)$$
$$L_{HR} = (.995505,\ 1.00050,\ -.185112)$$
$$L_{CT} = (.995505,\ 1.00050,\ -.185147)$$
$$D_E = \text{diag.}\ (.666001,\ .405405 \times 10^{-4},\ .444444 \times 10^{-6})$$
$$D_S = \text{diag.}\ (.666001,\ .405539 \times 10^{-4},\ .823690 \times 10^{-6})$$
$$D_{HR} = \text{diag.}\ (.666000,\ .405427 \times 10^{-4},\ .444646 \times 10^{-6})$$
$$D_{CT} = \text{diag.}\ (.666000,\ .405405 \times 10^{-4},\ .444447 \times 10^{-6})$$

To illustrate the effect of these rounding errors, we then solved the system

$$Kb = LDL^T b = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

The computed b's (accurate to six digits), are given below

$$\begin{aligned} b_E &= (3182965.,\ \ -518130.,\ \ -2665833.)^T \\ b_S &= (1715970.,\ \ -283491.,\ \ -1433039.)^T \\ b_{HR} &= (3181339.,\ \ -517794.,\ \ -2664543.)^T \\ b_{CT} &= (3182936.,\ \ -518124.,\ \ -2665813.)^T \end{aligned}$$

Note that b_S is off by a factor of ~ 2 (mostly attributable to the computed value of d_3), whereas b_{HR} is accurate to 3 digits and b_{CT} to 5 digits.

We next tested the updating capability of the composite-t update algorithm. When data points are added, the algorithm yields the same results as if one started with all of the data points since, except for a few multiplications, the computations are equivalent. When data is deleted, however, the answers may differ, since the process of data deletion is intrinsically less stable[5]. The following example illustrates this:

Let

$$X = \begin{pmatrix} 1 & -.999 & -.001 & 0. & 1 \\ 1 & -.99 & -.01 & 0. & 2 \\ 1 & -1. & .001 & .001 & 1 \end{pmatrix}$$

(This is the same data as used above with the addition of the data vector $(1, 2, 1)^T$.) Using $\alpha = (1, 2, 1)^T$ and specifying deletion to the composite-t update algorithm yielded

$$L = (.995504,\quad 1.00050,\quad -.186673)$$
$$D = \text{diag.}\ (.665999,\ .402583 \times 10^{-4},\ .431288 \times 10^{-6})$$

which is to be compared to L_{CT} and D_{CT} as computed above:

$$L_{CT} = (.995505,\quad 1.00050,\quad -.185147)$$
$$D_{CT} = \text{diag.}\ (.666000,\ .405405 \times 10^{-4},\ .444447 \times 10^{-6})$$

The differences are in the second and third digits of some of the computed quantities. It should be pointed out, however, that this is a particularly ill-conditioned example, and other examples yielded satisfactory results.

IV. Conclusions

Though efficient, the standard method suffers from an inability to update accurately and efficiently the MCD, as well as stability problems associated with having to work with matrices of the form YY^T. The Householder method obviates these problems at the cost of storage requirements and efficiency. Though slower still, the composite-t method drastically reduces the storage requirements, readily provides for updating of the MCD and improves computational performance from a stability standpoint. Which method one should use depends on the problem at hand and the weights one assigns to the various trade-offs between speed, stability, and updating capability.

In many of the computations for the analysis of remotely sensed data, the actual calculation of the covariance matrices and their MCD's takes relatively little time, so speed may not be an important factor. In this case, the optimal choice would appear to be the composite-t method, due to its superior numerical stability, relatively small storage requirements, and its updating capability. In areas such as signature extension, the updating capability of this method could be especially valuable.

APPENDIX

Listings of the program used to test the three methods are given below. KBYSM computes the MCD by the standard method (subroutine MCHLSK is used to actually compute the MCD from the computed covariance matrix). KBYHR computes the MCD by the Householder method. KBYCT computes the MCD by the Composite-t method, using subroutine COMPT which computes a rank one update of the MCD (Note that all of the data is in KBYCT, though the algorithm only requires that one data vector at a time be available. This was done for timing purposes only.) CTUPDT updates the MCD using subroutine COMPT.

```
      SUBROUTINE KBYSM(X,M,N,MXN,LD)
C
C     THIS ROUTINE COMPUTES THE MCD OF A COVARIANCE MATRIX BY THE
C     STANDARD METHOD
C
      REAL*4 X(MXN,1),LD(1)
      REAL*8 S1(12),S2(78)
C
C     X - THE N BY  M DATA MATRIX WHOSE FIRST DIMENSION IN THE CALLING
C         PROGRAM IS MXN
C     M - THE NUMBER OF DATA VECTORS
C     N - THE DIMENSION OF THE DATA
C     LD - THE RESULTING MCD CONTAINGG THE ELEMENTS OF L & D9  THIS
C         MATRIX IS STORED IN SYMMETRIC STORAGE MODE (I.E. LOWER
C         TRIANGULAR PORTION STORED BY ROWS) WITH THE ELEMENTS OF D
C          OCCUPYING THE DIAGONAL ENTRIES.
C
C     INITIALIZE
C
      DO 20 I=1,12
   20 S1(I)=0.D0
      DO 30 I=1,78
   30 S2(I)=0.D0
C
C     COMPUTE THE SUM OF THE DATA VECTORS AND THEIR CROSS-PRODUCTS.
C
      DO 10 L=1,M
      K=0
      DO 10 I=1,N
      S1(I)=S1(I)+X(I,L)
      DO 10 J=1,I
      K=K+1
   10 S2(K)=S2(K)+X(I,L)*X(J,L)
C
C     COMPUTE THE COVARIANCE MATRIX & STORE IT IN LD.
C
      K=0
      DO 40 I=1,N
      DO 40 J=1,I
      K=K+1
      LD(K)=(S2(K)-S1(I)*S1(J)/M)/(M-1)
   40 CONTINUE
C
C     MCHLSK COMPUTES THE MCD OF THE COVARIANCE MATRIX & OVERWRITES
C     THE RESULT ON IT.
C
      CALL MCHLSK(LD,N,S1,S2)
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE MCHLSK(KK,NV,DUM,DET)
C
C     ******************************************************************
C           THIS ROUTINE COMPUTES THE MODIFIED CHOLESKY DECOMPOSITION OF
C           THE COVARIANCE MATRIX.  THE DECOMPOSITIONS OVERLAY THE ELEMENTS
C           OF THE COVARIANCE MATRIX.
C           KK=L D L*
C     ******************************************************************
C
C           KK - THE COVARIANCE MATRIX STORED IN SYMMETRIC STORAGE MODE.
C           NV - THE NUMBER OF CHANNELS USED
C           DUM - A DOUBLE PRECISION WORK AREA OF SIZE NV-1
C           DET - THE DETERMINANT OF THE COVARIANCE MATRIX.
C
      REAL KK(1)
      REAL*8 DUM(1)
      REAL*8 R,R1,T1,TF
      LOGICAL*1 JE1
      JE1=.TRUE.
      J1=0
      JD=0
      DET=1.
C
C           LOOP OVER ALL CHANNELS
C
      DO 10 J=1,NV
      KL=J-1
      L=J+1
      JD=J1
      J1=J1+J
      TF=0.D0
      IF (JE1) GO TO 12
      K1=0
C
C           COMPUTE THE DIAGONAL ELEMENTS OF D AND STORE IN KK
C           TEMPORARILY STORE THE PRODUCT KK(I,I)*KK(J,I) IN DUM(I)
C
      DO 15 I=1,KL
      R=KK(JD+I)
      K1=K1+I
      R1=KK(K1)*R
      TF=TF-R1*R
      DUM(I)=R1
   15 CONTINUE
   12 CONTINUE
      TF=TF+KK(J1)
      KK(J1      )=TF
      DET=DET*TF
      IF (L.GT.NV) GO TO 10
      IRD=J1-L+1
C
C           COMPUTE THE R,J-TH ELEMENT OF L USING T1
C
      DO 20 IR=L,NV
      IRD=IRD+IR-1
      T1=0.D0
      IF (JE1) GO TO 16
      DO 25 I=1,KL
      T1=T1-DUM(I)*KK(IRD+I)
   25 CONTINUE
   16 KK(IRD+J)=(T1+KK(IRD+J))/TF
   20 CONTINUE
      JE1=.FALSE.
   10 CONTINUE
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE KBYHR(X,MU,N,M,U,MXN,LD)
      REAL*4 LD(1)
      REAL*4 X(MXN,M),MU(N),U(M)
      REAL*8 S
C
C     GIVEN THE MATRIX OF OBSERVATIONS X, COMPUTE THE MEAN AND MCD OF THE
C     RESULTING COVARIANCE MATRIX & STORE IT IN THE LOWER TRIANGULAR
C     PART OF X.  HOUSEHOLDER REFLECTIONS ARE USE* TO COMPUTE THE MCD.
C
C     X - DATA MATRIX WHICH IS DESTROYED
C     MU - ON OUTPUT , THE MEAN VECTOR
C     N - THE DIMENSION
C     M - THE NUMBER OF OBSERVATIONS TO BE USED
C     MXN - THE DIMENSION  (# OF ROWS ) OF X IN THE CALLING PROG.
C     U - WORKING STORAGE OF DIMENSION AT LEAST M
C     LD - DESIRED MCD STORED IN SYM STORAGE MODE
C
      NP1=M-1
C
C     SSET UP MATRIX TO BE TRIANGULARIZED
C
      DO 10 I=1,N
      S=0.D0
C
C     COMPUTE MEANS
C
      DO 20 J=1,M
   20 S=S+X(I,J)
      MU(I)=S/M
C
C     COMPUT THE MATRIX X-MEANS
C
      DO 30 J=1,M
   30 X(I,J)=X(I,J)-MU(I)
   10 CONTINUE
C
C     PERFORM HOUSEHOLDER TRANSFORMATIONS
      KK=0
C
      DO 40 I=1,N
C
C     COMPUTE NECESSARY QUANTITIES TO ANNIHILATE BOTTOM PART OF I-TH COL
C
      S=0.D0
C
C     ONLY LAST M-I+1 ELEMENTS OF U ARE USED
C
      DO 45 J=I,M
      XX=X(I,J)
      S=S+XX*XX
   45 U(J)=XX
      ALP=SIGN(SNGL(DSQRT(S)),U(I))
      IF (I.EQ.N) GO TO 44
      U(I)=U(I)+ALP
      BETA=ALP*U(I)
      II=I+1
C
C     APPLY TRANSFORMATION TO ROWS I+1 TO M & COLS I+1 TO N & SET
C     I,I-TH ELEMENT TO -ALP
C
      DO 50  L=II,N
      S=0.D0
      DO 55 K=I,M
   55 S=S+U(K)*X(L,K)
      XX=S/BETA
      DO 50 J=I,M
      X(L,J)=X(L,J)-U(J)*XX
   50 CONTINUE
   44 X(I,I)=-ALP
      IF (I.EQ.1) GO TO 42
C
C     COMPUTE L & D FROM L*SQRT(D)=R-TRANS/SQRT(M-1).  STORE L IN
C     LOWRR TRIANFULAR PART OF X & D ALONG DIAG.
C
      I1=I-1
      DO 60 J=1,I1
      KK=KK+1
   60 LD(KK)=X(I,J)/X(J,J)
   42 KK=KK+1
      LD(KK)=ALP*ALP/NP1
   40 CONTINUE
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE COMPT (LD,T,Z,N,  V,TMP)
      REAL*4 LD(1)
      REAL*8 T(1),Z(N),V(N)
      REAL*8 TMP(N)
      REAL*8 S
      LOGICAL*1 TPCS,RNDERR,LALP
C
C     THIS ROUTINE IS AN IMPLEMENTATION OF THE COMPOSITE - T ALGORITHM
C     TO PRRFORM A RANK 1 UPDATE OF THE MCD STORED IN ARRAY LD(I.E.
C     K=L*D*L-TRANS & WE WISH TO COMPUTE L' &D' S.T.K'=K+Z*Z-TRANS/T(1)
C     & K'=L'*D'*L'-TRANS).
C     LD - ARRAY CONTAINING L & D STORED IN SYM.STORAGE MODE
C     T - AN N+1 VECTOR WHOSE FIRST ELEMENT IS AS ABOVE
C     Z - VECTOR OF THE UPDAEE AS ABOVE
C     N - THE DIMENSION
C     V - WORKING STORAGE OF LENGTH .GE. N
C     TMP - DOUBLE PRECISION WORKING STORAGE OF LENGTH .GE. N
C
      TPCS=T(1).GT.0.
      IF (TPCS) GO TO 35
C
C     A POINT IS TO BE DELETED
C
      EPS=5.97E-8
C
C     SOLVE L*V=Z FOR V
C
      K=1
      V(1)=Z(1)
      DO 10 I=2,N
      IJ=I-1
      S=0.D0
      DO 15 J=1,IJ
      K=K+1
   15 S=S+LD(K)*V(J)
      K=K+1
      V(I)=Z(I)-S
   10 CONTINUE
C
C     COMPUTE THE T(I'S)
C
      K=0
      RNDERR=.FALSE.
      DO 20 I=1,N
      K=K+I
      TMP(I)=V(I)*V(I)/LD(K)
      T(I+1)=T(I)+TMP(I)
      IF (T(I+1).GE.0.) RNDERR=.TRUE.
   20 CONTINUE
      IF (.NOT.RNDERR) GO TO 35
C
C     ROUNDING ERROR HAS MADE A T(I+1).GE.0. SO CORRECT FOR THIS
C
      T(N+1)=EPS*T(1)
      DO 30 J=1,N
      I=N-J+1
      T(I)=T(I+1)-TMP(I)
   30 CONTINUE
   35 CONTINUE
      IJ=0
      DO 40 I=1,N
      I1=I+1
      IJ=IJ+I
      V(I)=Z(I)
      DI=LD(IJ)
      IF (DI.GT.0.) GO TO 44
C
C     D(I) =0. SO RANK OF D WILL EITHER INCREASE OR REMAIN UNCHANGED.
C
      IF (DABS(V(I)).GT.1.E-30) GO TO 42
C
C     RANK OF D WILL REMAIN UNCHANGED
C
      T(I+1)=T(I)
      GO TO 40
```

```
C
C     RANK OF D WILL INCREASE BY 1
C
   42 LD(IJ)=V(I)*V(I)/T(I)
      IF (I.EQ.N) RETURN
      K=IJ
      DO 45 J=I1,N
      K=K+J-1
      LD(K)=Z(J)/V(I)
   45 CONTINUE
      RETURN
   44 CONTINUE
C
C     UPDATE D
C
      IF (TPOS) T(I+1)=T(I)+V(I)*V(I)/DI
      ALP=T(I+1)/T(I)
      LD(IJ)=DI*ALP
      IF (I.EQ.N) RETURN
C
C     UPDATE L & MODIFY Z ACCORDINGLY
C
      BETA=(V(I)/DI)/T(I+1)
      LALP=.FALSE.
      IF (ALP.LE.4.) GO TO 52
C
C     THIS METHOD USED TO INSURE STABILITY IF ALPHA GT. 4
      LALP=.TRUE.
      GAM=T(I)/T(I+1)
      K=IJ
      DO 50 J=I1,N
      K=K+J-1
      XX=GAM*LD(K)+BETA*Z(J)
      Z(J)=Z(J)-V(I)*LD(K)
      LD(K)=XX
   50 CONTINUE
      GO TO 40
   52 K=IJ
      DO 60 J=I1,N
      K=K+J-1
      Z(J)=Z(J)-V(I)*LD(K)
      LD(K)=LD(K)+BETA*Z(J)
   60 CONTINUE
   40 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE KBYCT (LD,N,MXN,S,X,M,T)
      REAL*4 LD(1),S(N),X(MXN,M)
      REAL*8 T(1)
      INTEGER*4 R
C
C     THIS ROUTINE COMPUTES THE MCD OF THE COVARIANCE MATRIX & MEANS
C     ,GIVEN THE N-DIMENSIONAL M DATA VECTORS STORED IN X.  L&D ARE
C     STORED IN LD & THE MEANS IN VECTOR S
C
C     LD - ON OUTPUT, THE MCD OF THE COVAR. MARRIX STORED IN SYM.
C         STORAGE MODE WITH D ALONG THE DIAGONAL.
C     N - THE DIMENSION
C     MXN - THE NUMBER OF ROWS OF X AS DIMENSIONED IN THE CALLING PROG.
C     S - ON OUTPUT, THE VECTOR OF MEANS
C     X - THE DATA MATRIX
C     M - THE NUMBER OF DATA CECTORS TO BE USED.
C     T - WORKING STORAGE OF DIMENSION .GE. 4*N+1
C
C
C     INITIALIZE L & D MATRICES & VECTOR S
C
      IJ=0
      DO 10 I=1,N
      IJ=IJ+I
      LD(IJ)=0
      S(I)=X(I,1)
      IF (I.EQ.1) GO TO 10
      I1=I-1
      DO 15 J=1,I1
   15 LD(IJ-I+J)=0.
   10 CONTINUE
C
C     LOOP OVER ALL POINTS TO COMPUTE L & D FOR (M-1)*K
C
      DO 20 R=2,M
      T(1)=R
      SR1=SQRT(FLOAT(R-1))
C
C     COMPUTE Z FOR THIS X & UPDATE S
      DO 25 I=1,N
      T(N+I+1)=S(I)/SR1-SR1*X(I,R)
   25 S(I)=S(I)+X(I,R)
C
C     UPDATE L & D
C
      CALL COMPT(LD,T,T(N+2),N,  T(2*N+2),T(3*N+2))
   20 CONTINUE
C
C     MODIFY D S.T. K=L*D*L-TRANS & STORE MEAN IN S
C
      SR1=M-1
      IJ=0
      DO 30 I=1,N
      IJ=IJ+I
      LD(IJ)=LD(IJ)/SR1
      S(I)=S(I)/M
   30 CONTINUE
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
      SUBROUTINE CTUPDT (LD,N,M,ADD,S,ALP,T)
      REAL*4 LD(1),ALP(N),      S(N)
      REAL*8 T(1)
      LOGICAL*1 ADD
C
C     THIS ROUTINE UPDATES THE MCD STORED IN LD BY EITHER ADDING
C     OR DELETING A DATA VECTOR AS SPECIFIED BY THE LOGICAL VARIABLE ADD
C
C     LD - THE MCD STORED IN SYM.STORAGE MODE WITH D ALONG THE DIAGONAL
C     N - THE DIMENSION OF THE DATA VECTOR
C     M - THE NET NUMBER OF DATA CECTORS USED TO COMPUTE LD.  M WILL BE
C         UPDATED ON OUTPUT
C     ADD = =T IF DATA TO BE ADDED, =F IF DATA TO BE DELETED.
C     S - VECTOR CONTAINING M*MEANS
CC        S WILL BE UPDATED UPON RETURN
C     ALP - DATA VECTOR TO BE ADDED/DELETED
C     T - WORKING SYORAGE OF DIMENSION .GE.4*N+1
C
C     MODIFY D S.T. (M-1)*K=L*D*L-TRANS
C
      XM=M
      IJ=0
      DO 10 I=1,N
      IJ=IJ+I
      LD(IJ)=LD(IJ)*(M-1)
   10 CONTINUE
      IF (.NOT.ADD) GO TO 12
C
C     ADD A POINT
C
      T(1)=M+1
      Y=SQRT(XM)
      Q=M+1
      GO TO 14
C
C     DELETE A POINT
C
   12 T(1)=-M
      Y=SQRT(XM-1)
      Q=M-1
C
C     COMPUTE Z AND UPDATE S
C
   14 DO 20 I=1,N
      IF (.NOT.ADD) S(I)=S(I)-ALP(I)
      T(I+N+1)=S(I)/Y-Y*ALP(I)
      IF (ADD) S(I)=S(I)+ALP(I)
   20 CONTINUE
C
C     UPDATE L & D
C
      CALL CCMFT(LD,T,T(N+2),N,  T(2*N+2),T(3*N+2))
C
C     MODIFY D S.T.K=L*D*L-TRANS
C
      IJ=0
      DO 30 I=1,N
      IJ=IJ+I
      LD(IJ)=LD(IJ)/(Q-1.)
   30 CONTINUE
C
C     RESET M
C
      M=Q
      RETURN
      END
```

ORIGINAL PAGE IS
OF POOR QUALITY

REFERENCES

[1] D. L. Van Rooy, M.S. Lynn, and C. H. Snyder, "The Use of the Modified Cholesky Decomposition in Divergence and Classification Algorithms," Conf. Proc. Machine Processing of Remotely Sensed Data, LARS, Purdue, Oct. 16-18, 1973, pp. 3B-35 - 3B-47.

[2] G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.

[3] R. Fletcher and J. D. Powell, "On the Modification of LDL^T Factorizations," Math. Comp., 28, No. 128, 1974, pp. 1067-1087.

[4] J. M. Bennett, "Triangular Factors of Modified Matrices," Numer. Math., 7, 1965, pp. 217-221.

[5] W. M. Gentleman, "Least Squares Computations by Givens Transformations Without Square Roots," J. Inst. Maths. Applics, 12, 1973, pp. 329-336.

[6] J. H. Wilkinson, " The Solution of Ill-Conditioned Linear Equations," in Mathematical Methods for Digital Computers, Vol. 2, A. Ralston and H. Wilf, Eds., John Wiley and Sons, Inc., New York, 1967.

[7] C. L. Lawson and R. J. Hanson, Solving Least Squares Problems, Prentice Hall, New Jersey, 1974.