N73-31550

CR-133990

CASE FILE COPY

ICSA

INSTITUTE FOR COMPUTER SERVICES AND APPLICATIONS

RICE UNIVERSITY

ICSA -275-025-008

THE USE OF THE MODIFIED CHOLESKY DECOMPOSITION IN DIVERGENCE AND CLASSIFICATION CALCULATIONS

BY

D.L. VAN ROOY, M.S. LYNN AND
C.H. SNYDER
RICE UNIVERSITY
HOUSTON, TEXAS

ABSTRACT

This report analyzes the use of the Cholesky decomposition technique as applied to the feature selection and classification algorithms used in the analysis of remote sensing data (e.g. as in LARSYS). This technique is approximately 30% faster in classification and a factor of 2-3 faster in divergence, as compared with LARSYS. Also numerical stability and accuracy are slightly improved. Other methods necessary to deal with numerical stability problems are briefly discussed.

It is, in our view, extremely important that the best numerical techniques be used in production calculations. The argument that sub-optimal techniques have sufficed in the past is not valid if one considers that unexpected failures in the future may be extremely costly to rectify; since the use of the validated techniques discussed in this report are more reliable and efficient, it would seem wiser to proceed into further production calculations with the assurance that the systems and methods used rest on a more secure algorithmic foundation.

Institute for Computer Services & Applications
Rice University
Houston, Texas 77001
May, 1973
Research supported under NASA contract NAS9-12776

<u>CONTENTS</u>

THE USE OF THE MODIFIED CHOLESKY DECOMPOSITION IN DIVERGENCE AND CLASSIFICATION CALCULATIONS

1. Introduction:

This report analyzes the use of the Cholesky decomposition[(1)*] technique in the analysis of remotely sensed data, specifically in divergence calculations and in the evaluation of the maximum likelihood function; the latter occur in, respectively, the feature selection and classification techniques used, for example, in the LARSYS[(2)] system developed by the Laboratory for the Applications of Remote Sensing of Purdue University.

Although LARSYS was primarily developed for research purposes, increasing use of the system and of derivative systems such as ERIPS[(3)] for production processing emphasizes the need for efficient, accurate and stable algorithms as the basis for design objectives of computational analysis. The organization of computation in certain segments of LARSYS and the use of subroutines such as MINV from the IBM Scientific Subroutine Package (SSP)[(4)] do not lend credence that such design objectives have been met. The purpose of this report is to describe how one possible re-organization of the computation and the use of preferred techniques can improve the efficiency and accuracy of the system.

*Numbers in superscripts refer to references

The focus of this report is on improved efficiency in terms of computation time. Thus (Appendices A and B) the arithmetic precision used is identical with that used in LARSYS, so that a meaningful comparison of efficiency can be obtained. It will be shown that the algorithms proposed yield improvements in computational speed with no loss in accuracy or stability (in fact, slight improvements can be obtained in the latter).

Improvements in accuracy and stability can be achieved by further refinements in the techniques used. This will be the subject of a later report; however, in Section 6, we discuss where such improvements can be expected by the use of higher precision and/or the use of such techniques as iterative refinement, scaling and equilibration.

It is, in our view, extremely important that the best numerical techniques be used in production calculations. The argument that sub-optimal techniques have sufficed in the past is not valid if one considers that unexpected failures in the future may be extremely costly to rectify; since the use of the validated techniques discussed in this report are both more reliable and efficient, it would seem wiser to proceed into future production calculations with the assurance that the systems and methods used rest on a more secure algorithmic foundation.

2. Cholesky Decomposition:

Let K be real, nxn, symmetric positive-definite matrix. In the applications under consideration, K would be a covariance matrix. Then there is a unique, nxn, real, lower-triangular matrix, L, such that (Cholesky decomposition)

$$K = LL^* \tag{2.1}$$

where L^* denotes the (conjugate) transpose of L. There is also a unique, real, lower triangular matrix, $\tilde{L}$, and a real, positive diagonal matrix, $\tilde{D}$, such that (<u>modified Cholesky decomposition</u>)

$$K = \tilde{L}\tilde{D}\tilde{L}^* \tag{2.2}$$

where $\tilde{L}$ has diagonal elements equal to unity. From (2.1) and (2.2) it can be seen that

$$L = \tilde{L}\tilde{D}^{\frac{1}{2}} \tag{2.3}$$

where $\tilde{D}^{1/2}$ is the diagonal matrix whose entries are the square roots of the corresponding elements of $\tilde{D}$.

Either the Cholesky or modified Cholesky decompositions can be readily obtained from the following recurrence relationships[1],[5] (we use the notation $K = (k_{ij})$, $L = (\ell_{ij})$, $\tilde{L} = (\tilde{\ell}_{ij})$, $D = \text{diag}\left\{d_i\right\}$, $\tilde{D} = \text{diag}\left\{\tilde{d}_i\right\}$):

<u>Cholesky</u>

$$\left.\begin{aligned}
\ell_{11} &= k_{11}^{\frac{1}{2}} \\
\ell_{jj} &= \left(k_{jj} - \sum_{s=1}^{j-1} \ell_{js}^2\right)^{\frac{1}{2}} \\
\ell_{ij} &= \left(k_{ij} - \sum_{s=1}^{j-1} \ell_{is}\,\ell_{js}\right)\Big/\ell_{jj}
\end{aligned}\right\} \quad j = 1,\ldots,n \tag{2.4}$$

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

and, of course, $\ell_{ij} = 0$ for $j > i$.

Modified Cholesky

$$\tilde{d}_1 = k_{11}$$

$$\left.\begin{aligned} \tilde{d}_j &= \left(k_{jj} - \sum_{s=1}^{j-1} \tilde{d}_s \, \tilde{\ell}_{js}^2\right) \\ \tilde{\ell}_{ij} &= \left(k_{ij} - \sum_{s=1}^{j-1} \tilde{d}_s \, \tilde{\ell}_{is} \, \tilde{\ell}_{js}\right) \Big/ \tilde{d}_j \\ & \qquad i=j+1,\dots,n \end{aligned}\right\} \quad j = 1,\dots,n \tag{2.5}$$

where $\tilde{\ell}_{ii} = 1 \ (i=1,\dots,n)$ and $\tilde{\ell}_{ij} = 0$ for $j > i$.

For the applications under consideration, the modified Cholesky decomposition is more useful since it avoids the computation of square roots inherent in (2.4). It can easily be shown that, under the assumption that K is positive-definite, $\tilde{d}_j > 0 \ (j=1,\dots,n)$.

Once either decomposition is obtained, solutions of equations of the form

$$Kx = b \tag{2.6}$$

may readily be obatined from the back and forward substitutions (we henceforth only consider the modified Cholesky decomposition):

$$y_1 = \tilde{L}^{-1} b \tag{2.7}$$

$$y_2 = \tilde{D}^{-1} y_1 \tag{2.8}$$

$$x = \tilde{L}^{*-1} y_2 \tag{2.9}$$

since

$$
\begin{aligned}
Kx &= \tilde{L}\tilde{D}\tilde{L}^*x \\
&= \tilde{L}\tilde{D}y_2 \\
&= \tilde{L}y_1 \\
&= b
\end{aligned}
$$

as desired. (2.7) - (2.9) may alternatively be written (using $\leftarrow$ to denote replacement as opposed to equality) to economize on storage:

$$
\begin{aligned}
&x_1 = b_1/\tilde{d}_1 \\
&x_i \leftarrow \Big(b_i - \sum_{j=1}^{i-1} \tilde{\ell}_{ij}\, \tilde{d}_j\, x_j\Big)/\tilde{d}_i \\
&\qquad\qquad i=2,\ldots,n \\
&x_i \leftarrow \Big(x_i - \sum_{j=i+1}^{n} \tilde{\ell}_{ji}\, x_j\Big) \\
&\qquad\qquad i=n-1,n-2,\ldots,1\quad .
\end{aligned}
\tag{2.10}
$$

Note that in order to solve such systems there is no requirement to calculate K^{-1}, only $\tilde{L}$ and $\tilde{D}$ which requires approximately 1/3 the amount of computation.

This saving in itself is significant if one considers that the amount of time devoted to computing matrix inverses in connection with feature extraction in LARSYS varies roughly as mn^3, where m is the number of classes and n is the number of features under consideration - the corresponding amount of time devoted to the actual divergence calculation varies as $\frac{1}{2}m^2n^2$, which is of the same order of magnitude for most problems considered. Thus reducing the first factor by a third can significantly effect the overall computation time of itself.

In the applications under consideration, we thus have m covariance matrices $K_s\,(s=1,\ldots,m)$ corresponding to the number of classes. The dimensionality, n, of each K_s corresponds to the number of channels. With obvious notation, we write

$$K_s = \tilde{L}_s\tilde{D}_s\tilde{L}_s^* \qquad s=1,\ldots,m$$

where

$$K_s = (k_{ij}^{(s)}) \;,\; \tilde{L}_s = (\tilde{\ell}_{ij}^{(s)}) \;,\; \tilde{D}_s = \text{diag}\{\tilde{d}_i^{(s)}\}$$

and $\{\tilde{\ell}_{ij}^{(s)}\}$, $\{\tilde{d}_i^{(s)}\}$ are calculated as in (2.5)

3. Feature Selection:

Feature selection, as implemented in LARSYS, depends upon calculating a measure of inter-class divergence for multiple classes, requiring calculations of the form

$$D = D_1 + D_2 \tag{3.1}$$

where

$$D_1 = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \mathrm{tr}\left[(K_i - K_j)(K_j^{-1} - K_i^{-1}) \right] \tag{3.2}$$

$$D_2 = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (M_i - M_j)^* (K_i^{-1} + K_j^{-1})(M_i - M_j) \tag{3.3}$$

where tr A denotes the trace of A (sum of its diagonal elements) and M_s $(s=1,\ldots,m)$ is the mean vector for the s^{th} class. We first simplify (3.2) and (3.3).

We note that we can write

$$D_1 = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (\mathrm{tr} K_i K_j^{-1}) + \sum_{j=1}^{m-1} \sum_{i=j+1}^{m} (\mathrm{tr} K_i K_j^{-1}) - nm(m-1)$$

$$= \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (\mathrm{tr} K_i K_j^{-1}) + \sum_{i=2}^{m} \sum_{j=1}^{i-1} (\mathrm{tr} K_i K_j^{-1}) - nm(m-1)$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} \mathrm{tr}(K_i K_j^{-1}) - nm^2$$

$$= \sum_{j=1}^{m} \sum_{i=1}^{m} \mathrm{tr}(K_j^{-1} K_i) - nm^2$$

(since $\mathrm{tr}(AB) = \mathrm{tr}(BA)$)

$$= \sum_{j=1}^{m} \mathrm{tr}(K_j^{-1} K) - nm^2 \tag{3.4}$$

where

$$K = \sum_{i=1}^{m} K_i$$

$$= \tilde{L}\tilde{D}\tilde{L}^* \qquad \text{(say)}$$

Now

$$\mathrm{tr} K_j^{-1} K = \mathrm{tr}(\tilde{L}_j^{*-1} \tilde{D}_j^{-1} \tilde{L}_j^{-1} \tilde{L}\tilde{D}\tilde{L}^*)$$
$$= \mathrm{tr}(\tilde{D}_j^{-1} T_j \tilde{D} \tilde{T}_j^*)$$

where

$$T_j = \tilde{L}_j^{-1}\tilde{L} = (t_{ij}) \qquad \text{(say)} \tag{3.5}$$

Thus

$$\mathrm{tr}(K_j^{-1}K) = \sum_{p=1}^{n} \sum_{q=1}^{p} (t_{pq}^{(j)2}/\tilde{d}_p^{(j)})\tilde{d}_q \tag{3.6}$$

$$j=1,\ldots,m$$

Hence D_1 may economically be calculated from (3.4), (3.5) and (3.6). It should be noted that the calculation of the $\{T_j\}$ in (3.5) each require n calculations of the form (2.10); however, since $T_j, \tilde{L}_j$ and $\tilde{L}$ are all lower triangular, it is important to remark that much of the computation may be reduced by observing that, in calculating the q^{th} column of T_j, the index n in (2.10) is actually replaced by n-q+1 (q=1,...,n).

The calculation of D_2 may be similarily simplified. For, from (3.3), we may write

$$D_2 = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (M_i - M_j)^* (K_i^{-1} + K_j^{-1}) (M_i - M_j)$$

$$= \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} (M_i - M_j)^* K_i^{-1} (M_i - M_j)$$

$$+ \sum_{j=1}^{m-1} \sum_{i=j+1}^{m} (M_i - M_j)^* K_j^{-1} (M_i - M_j) \qquad (3.7)$$

(interchanging i and j in the second sum). Interchanging the order of subscripts gives

$$D_2 = \sum_{i=1}^{m} \sum_{j=1}^{m} (M_i - M_j)^* K_i^{-1} (M_i - M_j)$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{m} \eta^{ij*} \tilde{D}_i^{-1} \eta^{ij} \qquad (3.8)$$

where

$$\eta^{ij} = \tilde{L}_i^{-1} (M_i - M_j)$$

$$= \delta^{ii} - \delta^{ij} \qquad i,j=1,\ldots,m$$

where the computation of

$$\delta^{ij} = \tilde{L}_i^{-1} M_j \tag{3.9}$$

involves a forward substitution, that is, δ^{ij} is obtained from

$$\delta_1^{(ij)} = m_1^{(j)}$$

$$\delta_p^{(ij)} = m_p^{(j)} - \sum_{q=1}^{p-1} \tilde{\ell}_{pq}^{(i)} \delta_q^{(j)} \quad , \quad p=2,\ldots,n \ . \tag{3.10}$$

We thus have, from (3.8) that

$$D_2 = \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{p=1}^{n} (\eta_p^{(ij)})^2 / \tilde{d}_p^{(i)} \tag{3.11}$$

where

$$\eta_p^{(ij)} = \delta_p^{(ii)} - \delta_p^{(ij)}$$

and the $\left\{\delta_p^{(ij)}\right\}$ are calculated from (3.10).

The use of the above formulae should probably not be compared with the approach used in LARSYS itself, but with the improvements proposed by G. Austin[8] which take full advantage of the symmetry of the $\left\{K_i\right\}$ and of the symmetric structure of the summands in (3.7). It can be shown that the amount of work involved in calculating D_2 in (3.11) is comparable with that involved with the corresponding terms in Ref (8). However, the amount of work involved in evaluating (3.4) is actually considerably less than the method proposed in Ref (8) on account of the asymptotic linear dependence on m, as opposed to the quadratic dependence of Ref (8). It should nevertheless be pointed out that, from (3.4)

$$D_1 = \mathrm{tr}(\hat{K}K) - nm^2 \tag{3.12}$$

where

$$\hat{K} = \sum_{j=1}^{m} K_j^{-1} \tag{3.13}$$

Thus, if the K_j^{-1} have been precomputed, the amount of work involved in evaluating D_1 may become negligible compared with the evaluation of D_2 by using (3.12) and the fact that, for symmetric matrices A, B:

$$\mathrm{tr}(AB) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}b_{ji}$$

$$= 2\sum_{i=1}^{n} \sum_{j=1}^{i-1} a_{ij}b_{ji} + \sum_{i=1}^{n} a_{ii}b_{ii} \quad .$$

However, this approach does not obviate the overall savings in feature selection of using the Cholesky decomposition instead of computing matrix inverses.

4. Classification:

Classification involves the calculation of the maximum likelihood functions

$$f_j(x) = \sigma_n \alpha_j \exp[-\tfrac{1}{2}(x-M_j)^* K_j^{-1}(x-M_j)] \qquad (4.1)$$

$$j=1,\ldots,m$$

where x is the observation vector, $\sigma_n = 1/(2\pi)^{m/2}$, and

$$\alpha_j = (\det K_j^{-1})^{\frac{1}{2}}$$

$$= \left(\prod_{p=1}^{n} \tilde{d}_p^{(j)}\right)^{-\frac{1}{2}} \qquad (4.2)$$

Actually, since $\exp(x)$ is a monotonic increasing function, only $\log f_j(x)$ needs to be computed in determining the maximum of $f_j(x)$ over all m classes.

However, (4.1) is again simplified by noting that

$$(x-M_j)^* K_j^{-1}(x-M_j) = y_j^* \tilde{D}_j^{-1} y_j$$

where

$$y_j = \tilde{L}_j^{-1}(x-M_j)$$

is calculated in a manner analogous to (3.9).

5. Results:

The above techniques have been tested by appropriately modifying the OS version of LARSYS[6] supplied by NASA-JSC. These modifications are listed in Appendices A and B. In actuality the modification to the divergence calculations in feature selection use the Cholesky decomposition as opposed to the modified Cholesky decomposition as discussed in Section 3 - further savings of time, obtained by not having to calculate square roots, could be realized by using the modified Cholesky decompsosition.

The modifications were written in single-precision FORTRAN and compared with the original single-precision versions in LARSYS. In the case of classification, the results were also compared with a single-precision version of the corresponding calculations in LARSYS written in assembly code.

The precision of these timing results is very open to question due to the difficulty of obtaining accurate and reproducible timing information under the OS Operating System of the IBM 370/155. Timings are heavily dependent on general system activity; furthermore the considerable subroutine overhead inherent to the computation tends to mask much of the potential arithmetic economies of efficiency.

The results are summarized in Figures 1, 2 and 3 on test data supplied by Purdue University with LARSYS. Figure 1, depicts the ratio of the time taken by the original LARSYS version (DIVERG) to that taken by the proposed algorithm (CHOLESKY) in a divergence calculation for feature selection using six channels; this ratio is plotted for a varying number of target classes. It can be seen that CHOLESKY is approximately twice as fast as DIVERG.

Theoretical analysis shows that this ratio should be greater than three for all values of m, and asymptotically should approach four for large values of m . This discrepancy underscores the high degree of imprecision associated with the timing results.

In Figure 2, the same ratio is plotted for a fixed number of classes (11) and where a varying number of features is selected from twelve channels. Except for a very small number of features, where the order of the K_i is so small that the time of calculation is dominated by computational overhead, it can again be seen that CHOLESKY is between two and three times faster than DIVERG. Again, theoretically, this ratio should be between three and four for all values of the number of features.

In Figure 3, the time taken for classification using the three methods is compared for a number of points varying from 50,000 to 100,000. The Cholesky method is significantly faster (about 30%). It should be pointed out that, as has been noted elsewhere(7), equivalent savings can be obtained by using a variant of the LARSYS calculations which does not employ the modified Cholesky decomposition; however, this variant does not have the accuracy potential of the Cholesky approach(1).

6. <u>Improvements in Accuracy</u>:

The modifications described were executed in single-precision so as to provide a basis for comparison with the LARSYS calculations. Without further refinement, it should not be surprising that the accuracy will be correspondingly limited, since(1) accuracy in such computations is essentially a function of three principal components:

- the method employed
- the arithmetic significance
- the conditioning of the various matrices

For ill-conditioned systems (in the applications under consideration, these may arise, for example, from working with highly-correlated channels), more precise methods have to be employed and/or the arithmetic significance increased. Directions which need to be examined with higher accuracy objectives in mind include, not only that of using higher significance arithmetic in sensitive portions of the computation, but also those of employing iterative refinement, scaling or equilibration. These will, however, be studied in a later report.

References

(1) G. Forsythe & C. B. Moler, Computer Solution of Linear Algebraic Systems, Prentice Hall, Englewood Cliffs, N. J. 1967.

(2) Supplied by LARS, Purdue Univ., West Lafayette, Indiana.

(3) Earth Resources Interactive Processing System (ERIPS) prepared by IBM Federal Systems Division for NASA - JSC

(4) System 360 Scientific Subroutine Package, Version III, IBM Application Program.

(5) A. Ralston and H. S. Wilf, Mathematical Methods for Digital Computers, Vol. II, John Wiley and Sons, New York, 1967.

(6) The OS version of LARSYS was written by IBM - Federal Systems Division for NASA - JSC

(7) G. Austin, "Analysis of LARS Subroutine CLASS and Recommended Coding Improvements to Reduce Its Execution Time," NASA - JSC MPAD Memorandum, May 16, 1972.

(8) G. Austin, "Modifications to ERIPS Requirements to Reduce Computation Time and Storage Requirements," MSC Internal Note No. 72-FM-210. Jan. 16, 1973. NASA

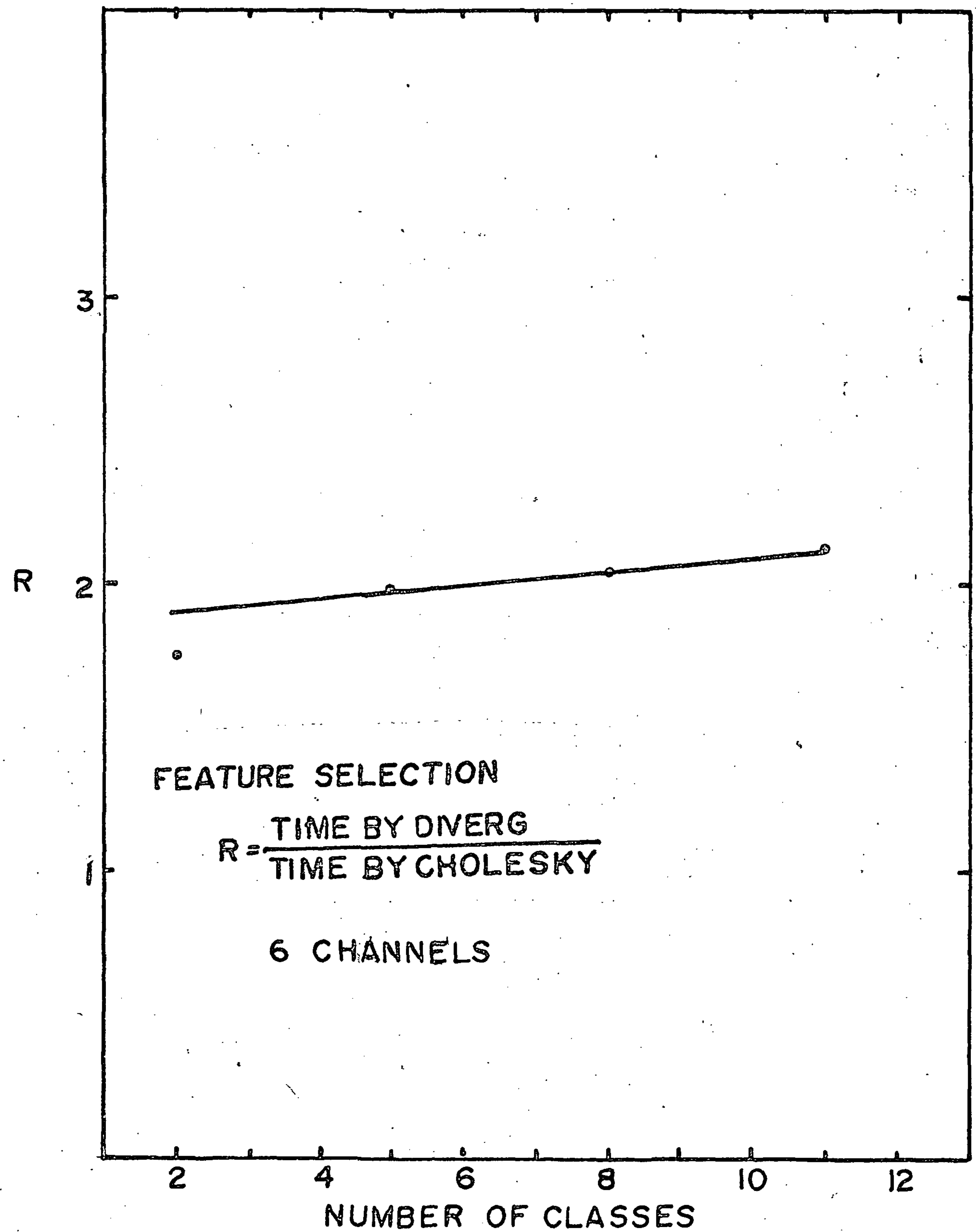


Figure 1

Timing comparison as a function of number of classes used for the divergence calculation.

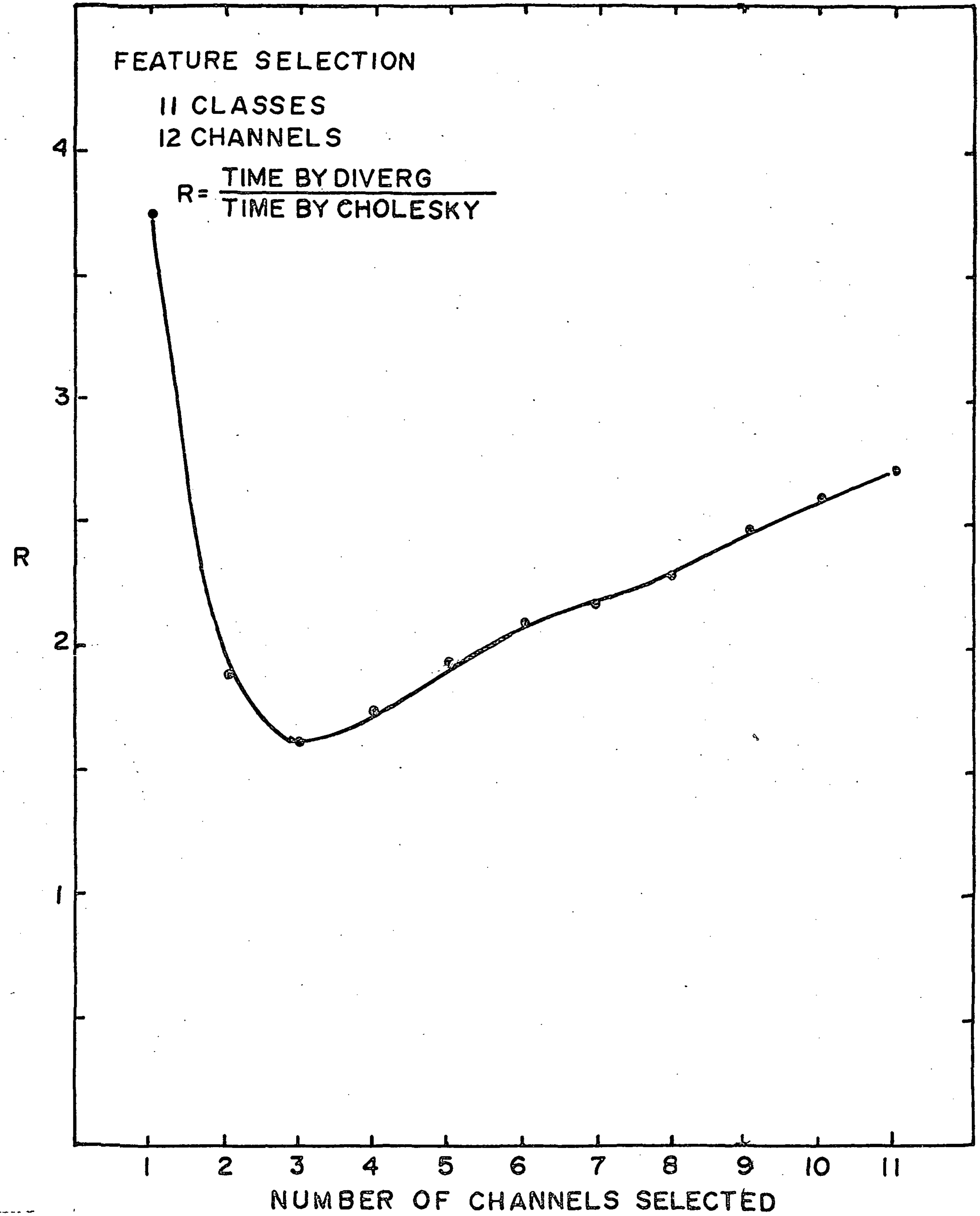


Figure 2
Timing comparison as a function of number of channels selected for the divergence calculation.

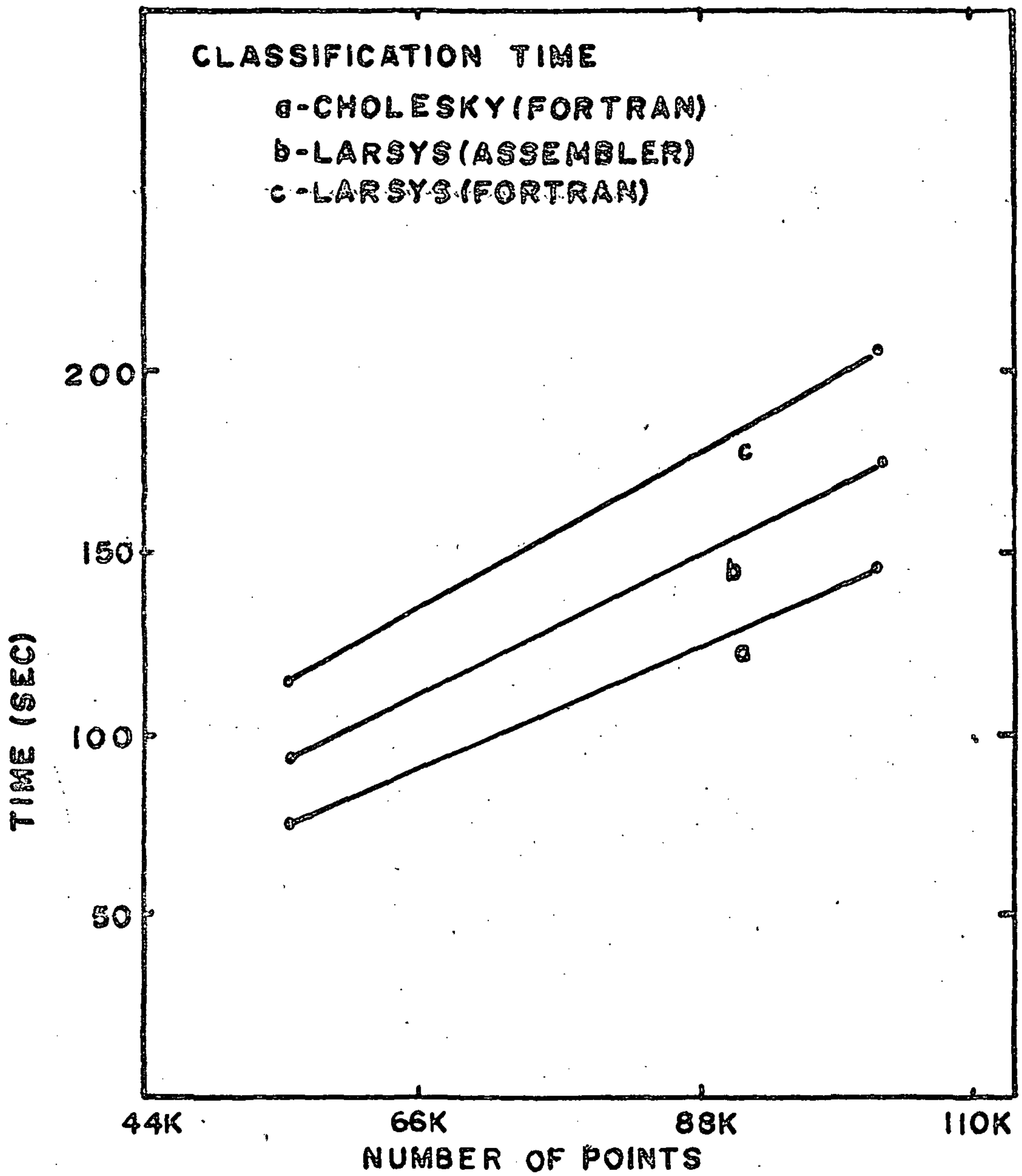


Figure 3

Timing comparison for classification programs.

APPENDIX A

Flowcharts and Listings of Subroutines Used in Performing Feature Selection Using the Divergence Criterion Employing the Cholesky decomposition

DIVCAL

Subroutine DIVCAL, which performs the divergence calculation, and stores the channels corresponding to the maximum separation, is called by the main program, which reads the aircraft data tape and calculates the mean vectors and covariance matrices of the various classes from specified training fields.

On the initial call to DIVCAL, the mean vectors and covariance matrices of each class corresponding to channels 1 to NF, where NF is the number of features desired, are placed into vectors in DIVCAL. The computation is done in vector rather than matrix form, since the computer will handle calculations with single subscripts faster than those involving multiple subscripts. However, the expressions which follow are given in matrix form for the sake of clarity.

The search for the set of channels which exhibits the maximum separation is done by the exhaustive technique in which every possible combination of NF channels is examined. Because of the method used to step through all the combinations, often the first (NF-1) channels selected remain unchanged, with only the highest numbered channel changing between divergence calculations. Thus, in subsequent calls, only those elements of the mean vector and covariance matrices corresponding to the changing channels are redefined, with the other elements being retained in storage.

DIVCAL then performs the Cholesky decomposition of the covariance matrix for each class, and simultaneously accumulates the sum of the matrices. The Cholesky decomposition of the sum of the covariance matrices is then calculated.

The total interclass separation is computed from

$$\text{TOTSEP} = D1 + D2$$

where D 1 and D2 are function subprograms. D 1 is defined as

$$D1 = \sum_j \| L^{-1}_j L \|^2$$

where the sum over j is over all the classes, L^{-1}_j is the inverse of the Cholesky decomposition of the NF $\times$ NF covariance matrix of the j^{th} class whose elements correspond to the set of channels under consideration. L is the Cholesky decomposition of the NF $\times$ NF sum of covariance matrices likewise corresponding to the set of channels under consideration and $\| A \|^2$ is defined as

$$\| A \|^2 = \sum_{i,j} a^2_{i,j}$$

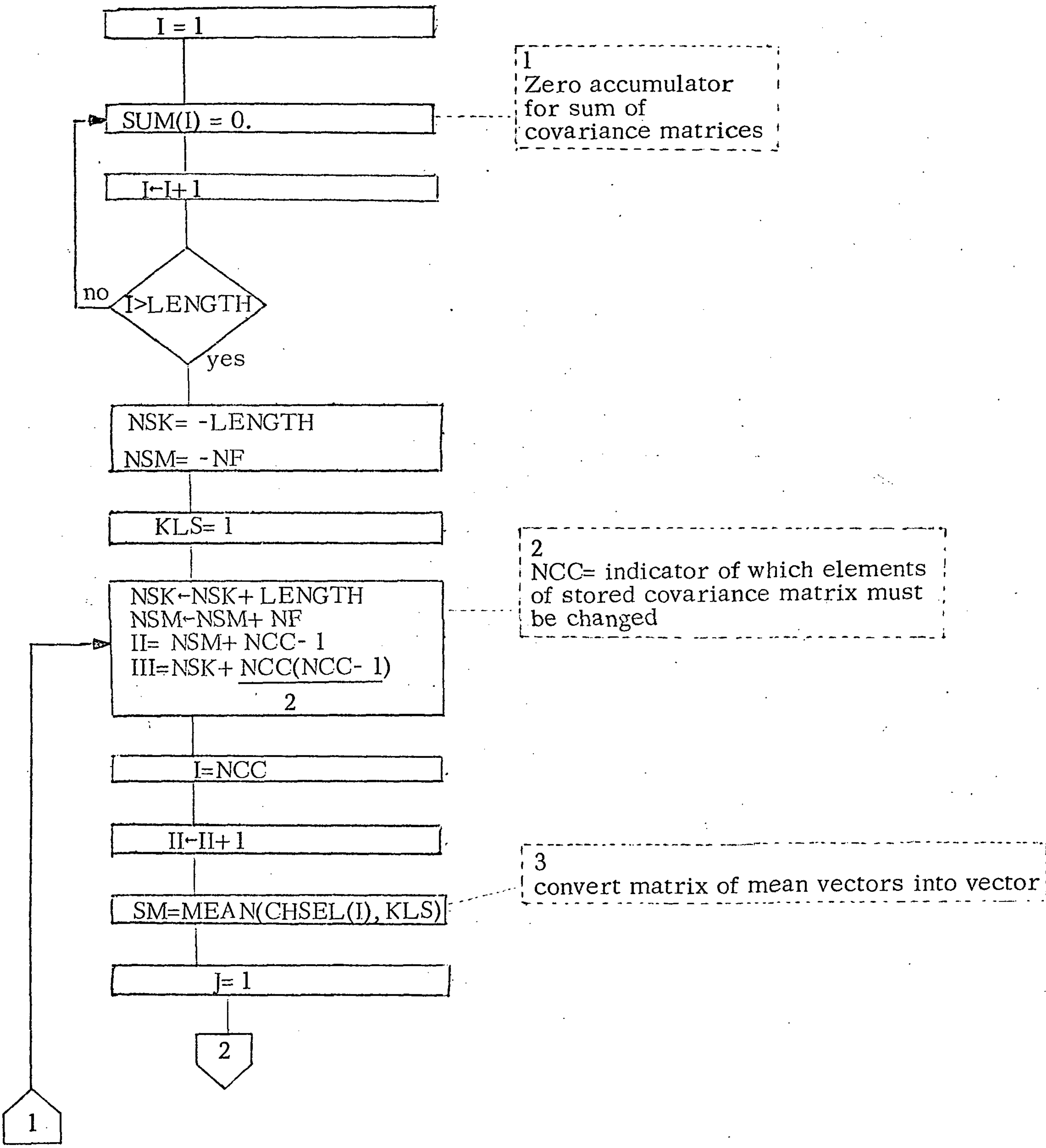
D 1 performs $\| A \|^2$ and the sum over classes after calling subroutine CK to perform $L^{-1}_j L$ by back substitution

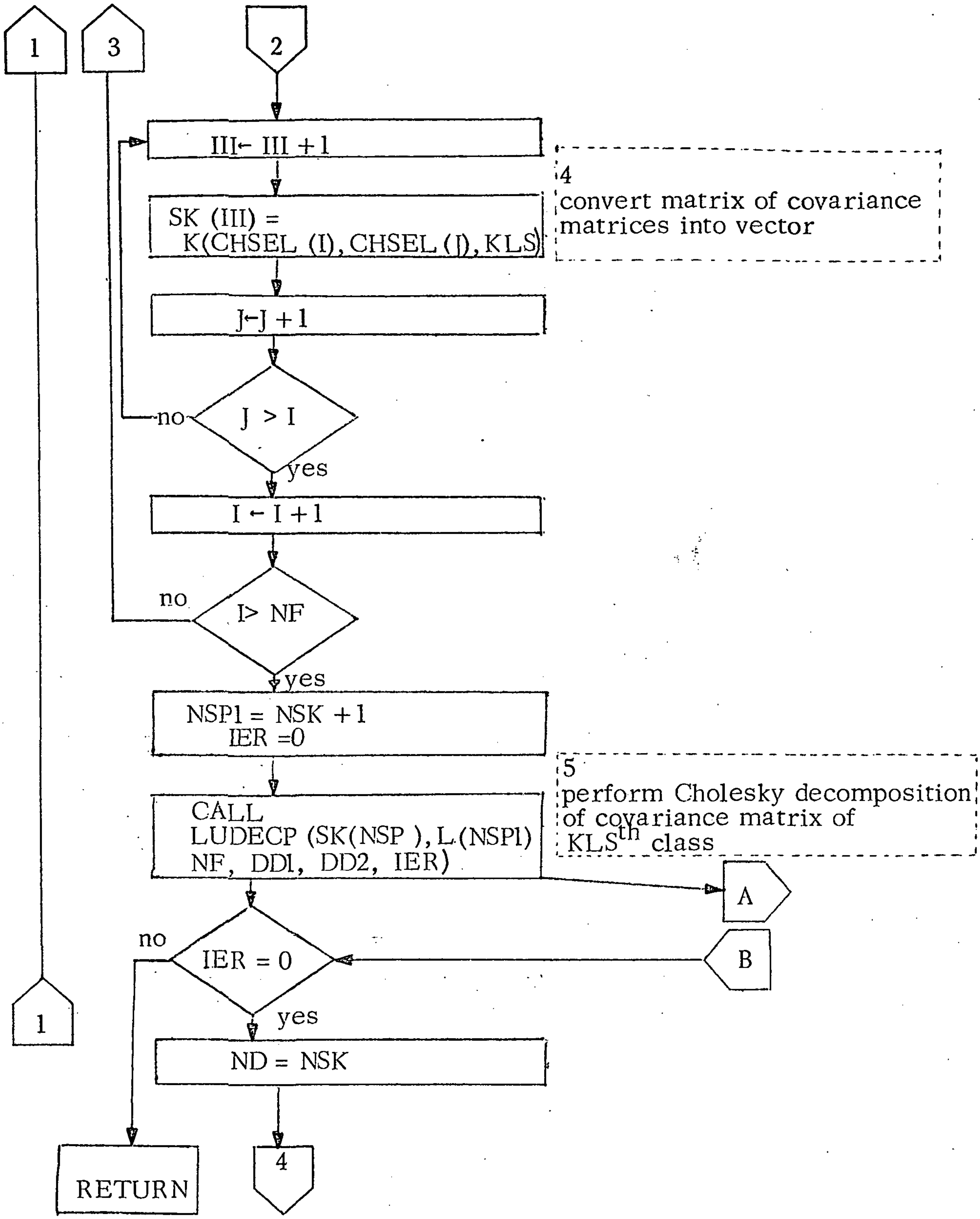
D2 is defined as

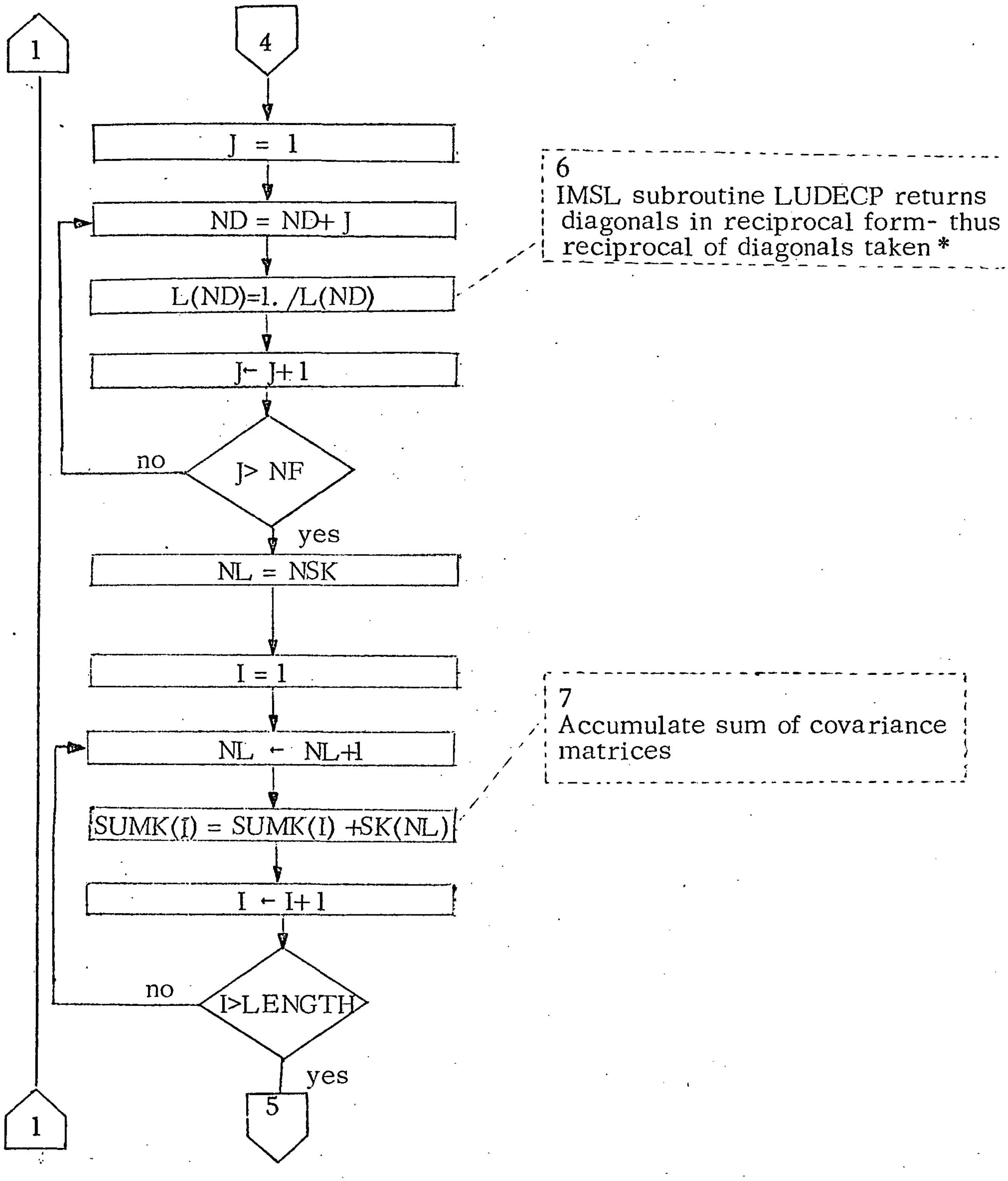
$$D2 = \sum_{i,j} \eta^*_{ij} \eta_{ij}$$

where $\eta_{ij} = \delta_{ii} - \delta_{ij}$, $\delta_{ij} = L^{-1}_i M_j$, and M_j is the mean vector formed of the selected channels of the j^{th} class. D2 performs the multiplication and sum over classes after calling subroutine DJM to calculate $L_i^{-1} M_j$ by back substitution.

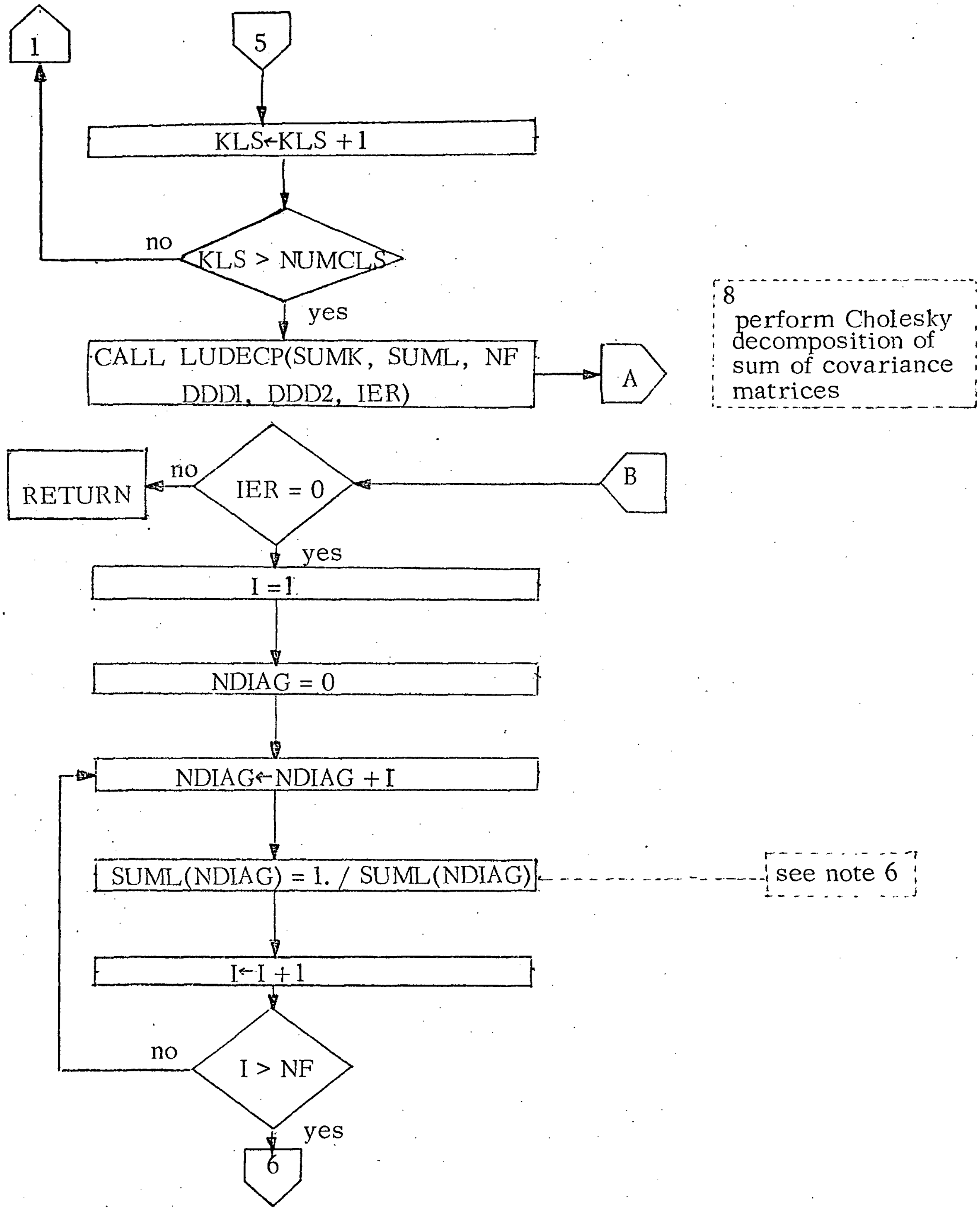
SUBROUTINE DIVCAL

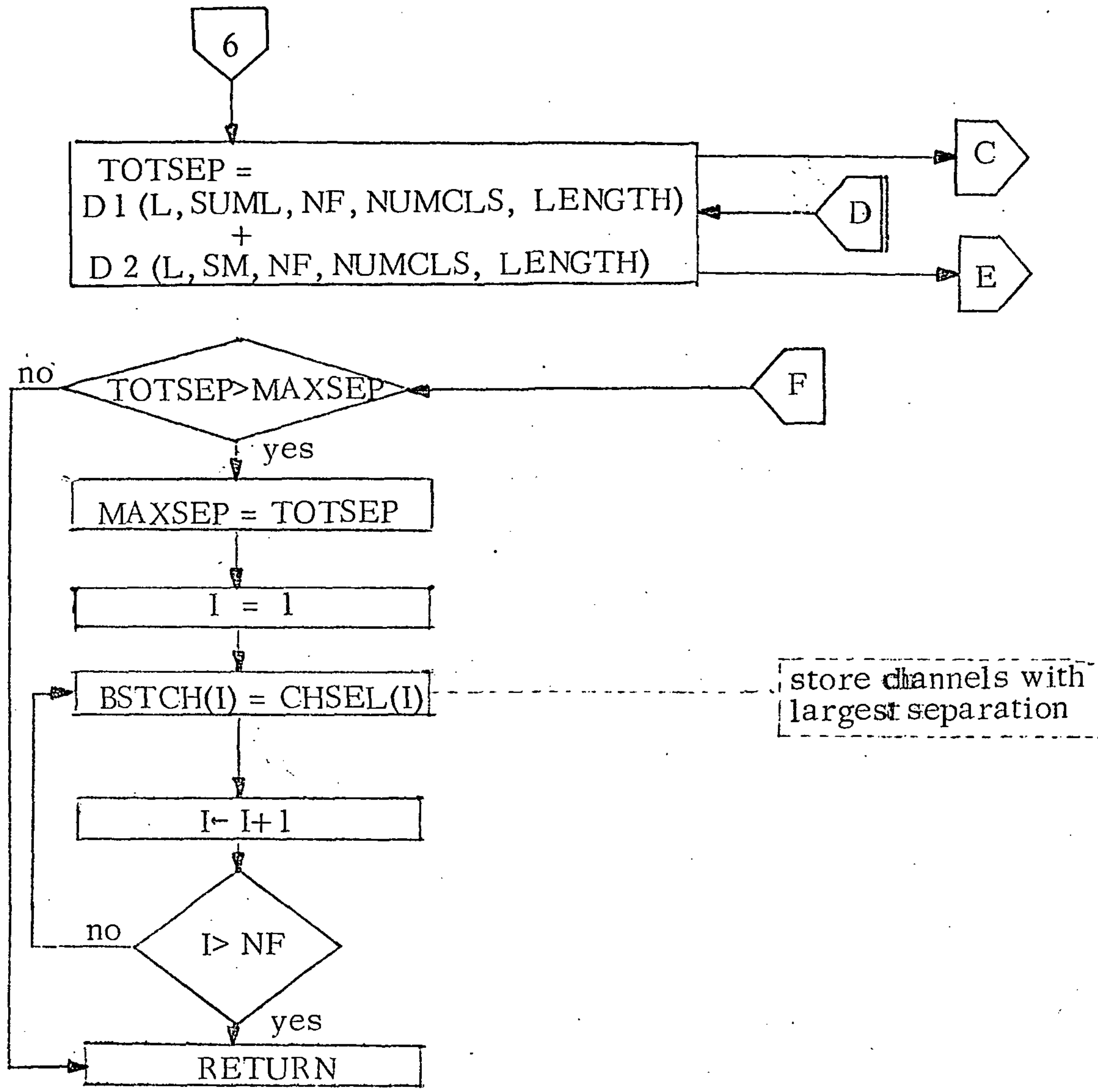



1
3
2
III← III +1
SK (III) =
K(CHSEL (I), CHSEL (J), KLS)
4
convert matrix of covariance matrices into vector
J←J +1
no
J > I
yes
I ← I +1
no
I> NF
yes
NSP1 = NSK +1
IER =0
CALL
LUDECP (SK(NSP), L (NSP1)
NF, DD1, DD2, IER)
5
perform Cholesky decomposition of covariance matrix of KLS^th class
A
no
IER = 0
B
1
yes
ND = NSK
RETURN
4

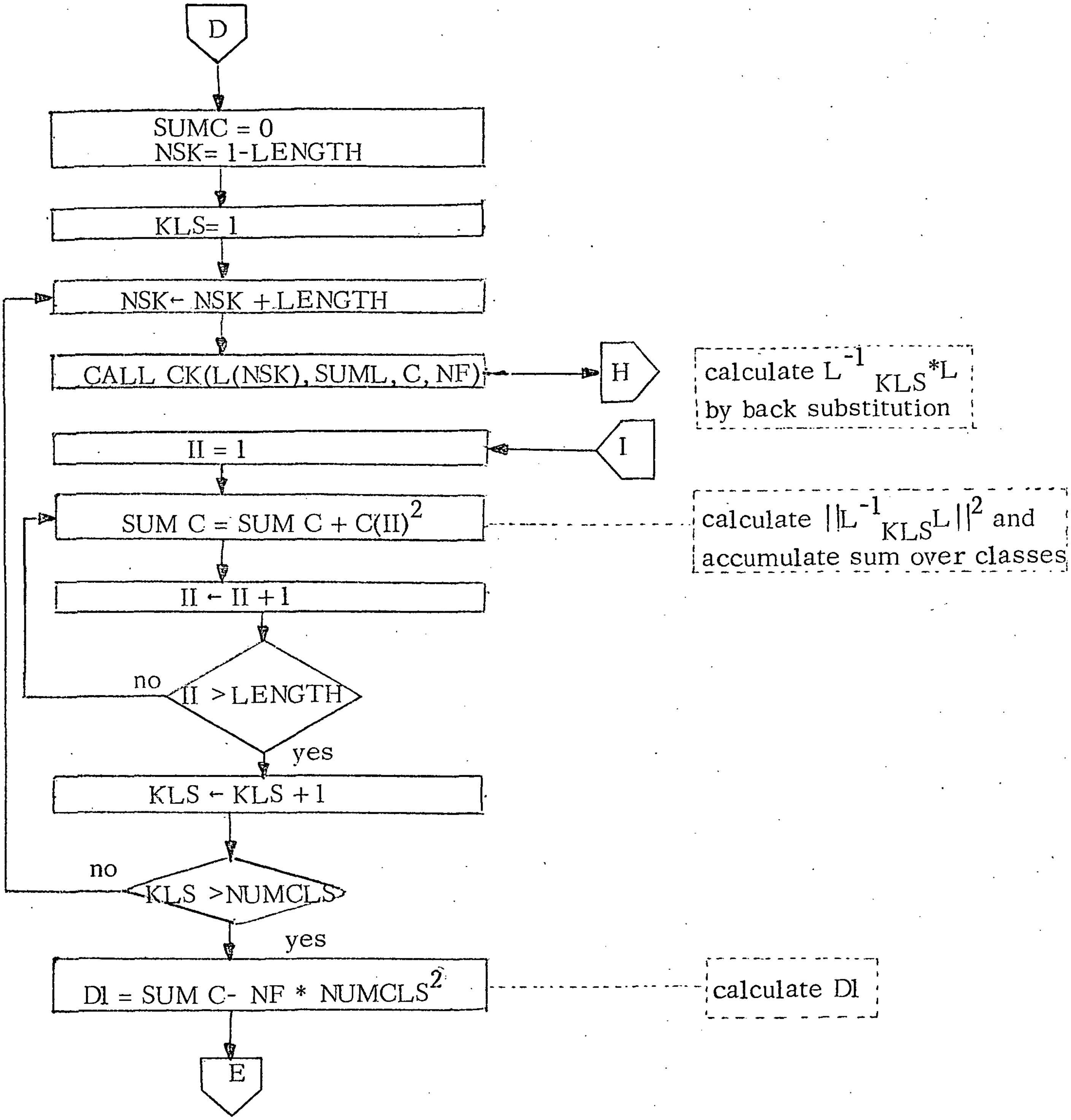


* IMSL, 6200 Hillcroft, Suite 510, Houston, Texas 77036

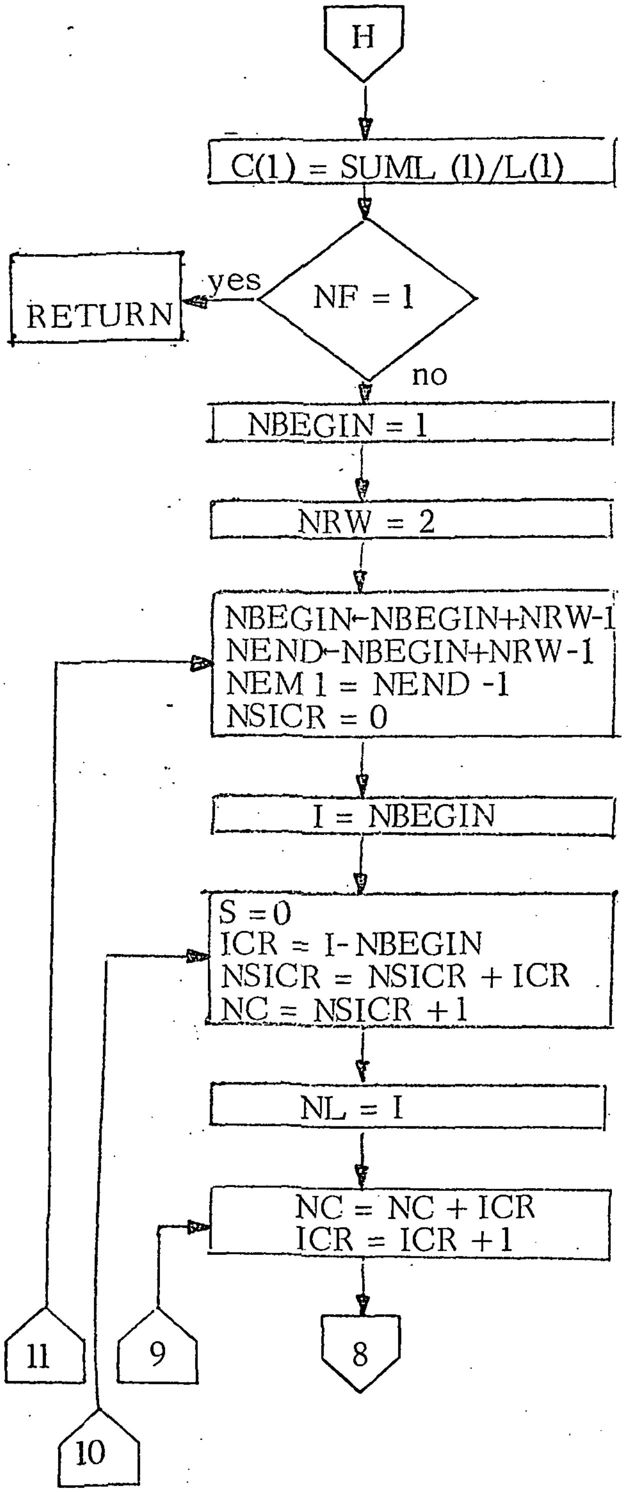

1
5
KLS←KLS +1
no
KLS > NUMCLS
yes
CALL LUDECP(SUMK, SUML, NF
DDD1, DDD2, IER)
A
8
perform Cholesky decomposition of sum of covariance matrices
RETURN
no
IER = 0
B
yes
I = 1
NDIAG = 0
NDIAG←NDIAG + I
SUML(NDIAG) = 1. / SUML(NDIAG)
see note 6
I←I +1
no
I > NF
yes
6


6
TOTSEP =
D 1 (L, SUML, NF, NUMCLS, LENGTH)
+
D 2 (L, SM, NF, NUMCLS, LENGTH)
C
D
E
TOTSEP>MAXSEP
F
no
yes
MAXSEP = TOTSEP
I = 1
BSTCH(I) = CHSEL(I)
store channels with largest separation
I← I+1
I> NF
no
yes
RETURN

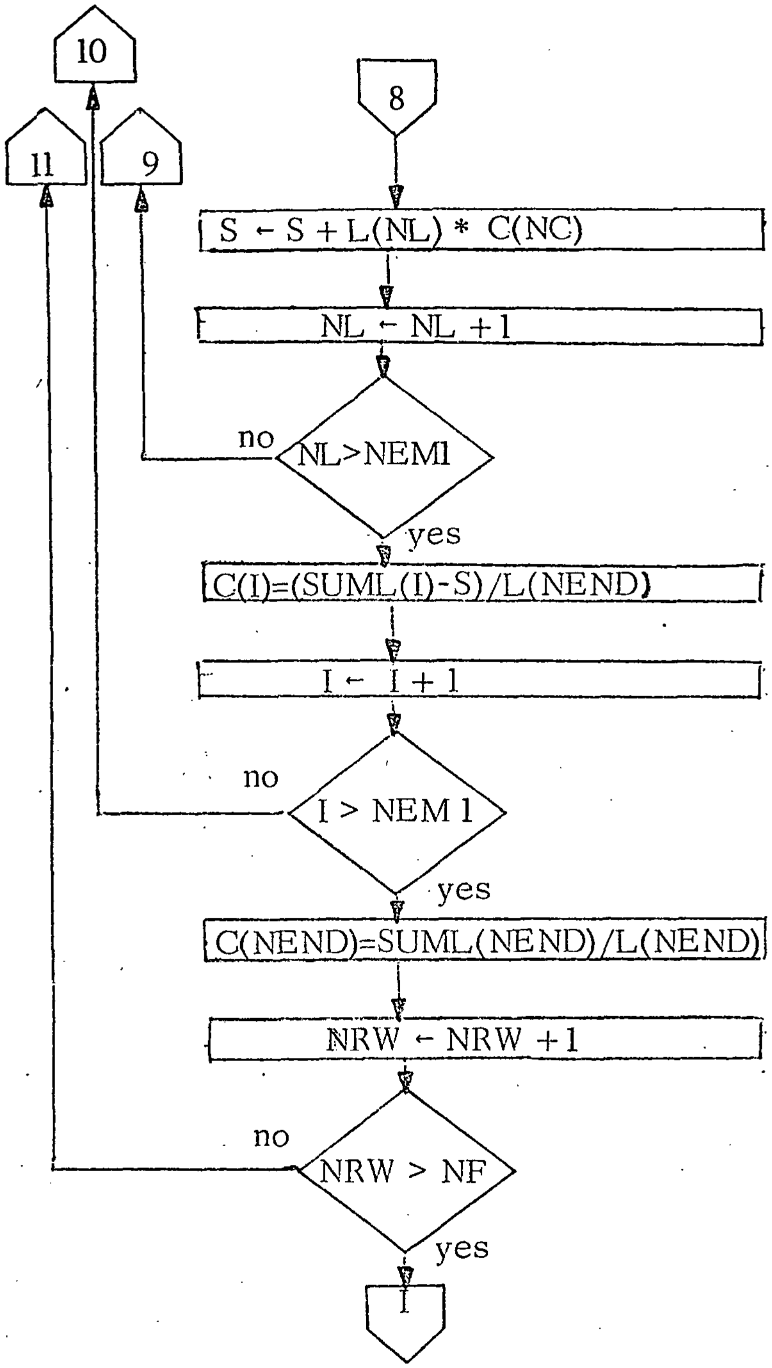
D_1 (L, SUML, NF, NUMCLS, LENGTH)



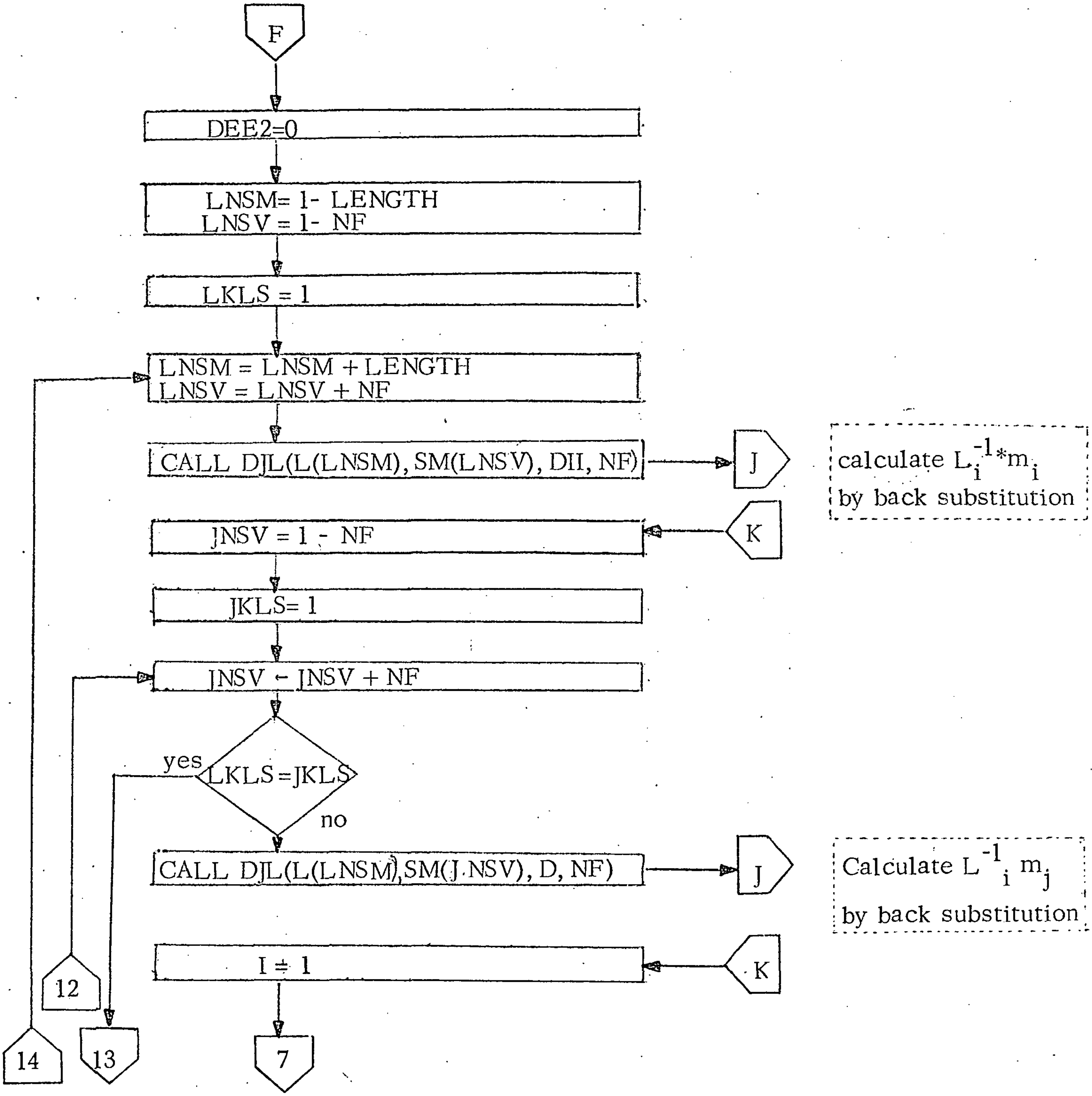
CK(L, SUML, C, NF)

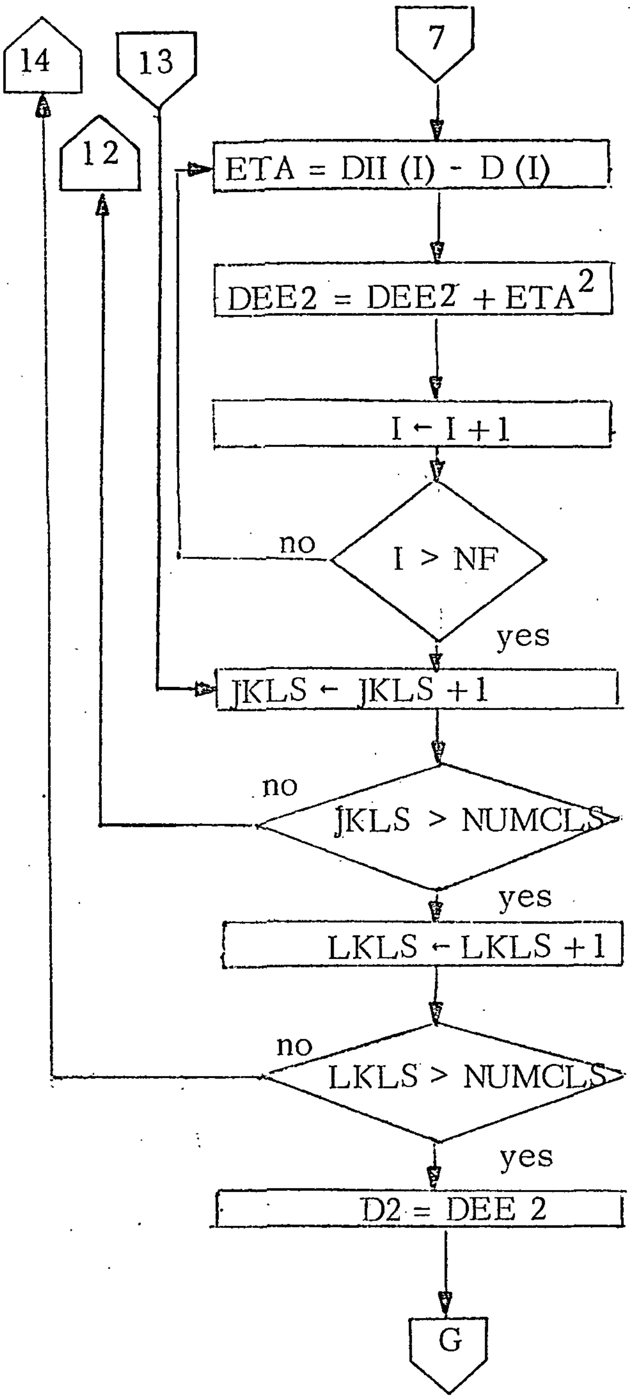


H
C(1) = SUML (1)/L(1)
NF = 1
yes
RETURN
no
NBEGIN = 1
NRW = 2
NBEGIN←NBEGIN+NRW-1
NEND←NBEGIN+NRW-1
NEM 1 = NEND -1
NSICR = 0
I = NBEGIN
S = 0
ICR = I-NBEGIN
NSICR = NSICR + ICR
NC = NSICR +1
NL = I
NC = NC + ICR
ICR = ICR +1
8
11
9
10

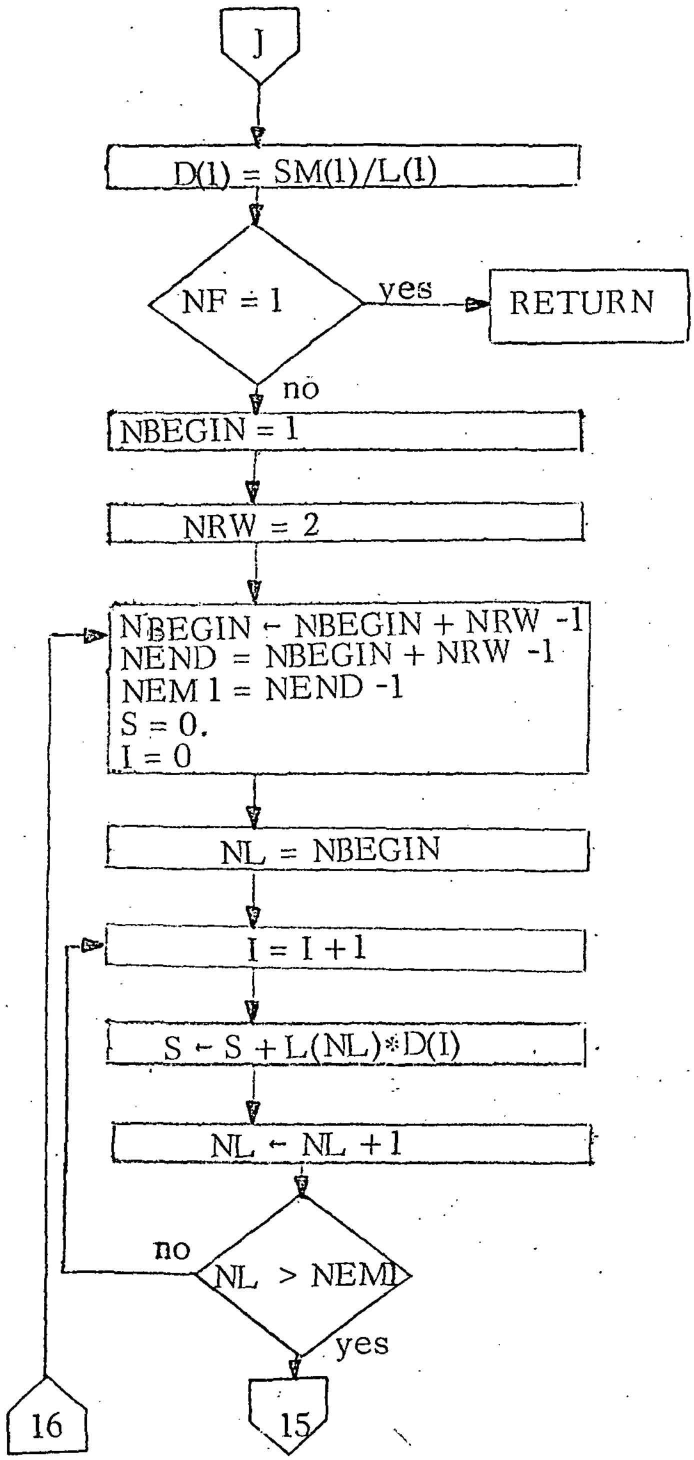

10
8
11
9
S ← S + L(NL) * C(NC)
NL ← NL + 1
no
NL>NEM1
yes
C(I)=(SUML(I)-S)/L(NEND)
I ← I + 1
no
I > NEM 1
yes
C(NEND)=SUML(NEND)/L(NEND)
NRW ← NRW + 1
no
NRW > NF
yes
I

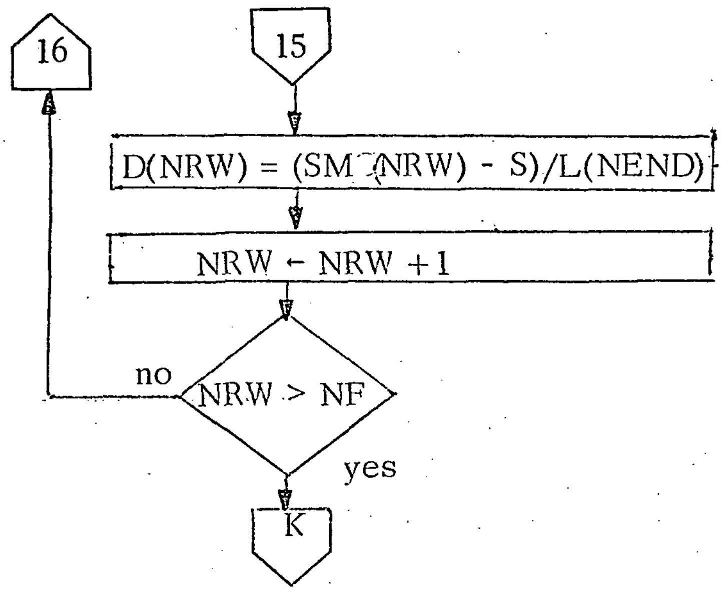
D2 (L, SM, NF, NUMCLS, LENGTH)




14
13
7
12
ETA = DII (I) - D (I)
DEE2 = DEE2 + ETA^2
I ← I + 1
no
I > NF
yes
JKLS ← JKLS + 1
no
JKLS > NUMCLS
yes
LKLS ← LKLS + 1
no
LKLS > NUMCLS
yes
D2 = DEE 2
G

DJL (L, SM, D, NF)


J
D(1) = SM(1)/L(1)
NF = 1
yes
RETURN
no
NBEGIN = 1
NRW = 2
NBEGIN ← NBEGIN + NRW -1
NEND = NBEGIN + NRW -1
NEM 1 = NEND -1
S = 0.
I = 0
NL = NBEGIN
I = I + 1
S ← S + L(NL)*D(I)
NL ← NL + 1
no
NL > NEM1
yes
16
15


16
15
D(NRW) = (SM (NRW) - S)/L(NEND)
NRW ← NRW +1
no
NRW > NF
yes
K

```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                  SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
      INTEGER*2 BLOCK(4),CSEL(12)/12*1/,BDATA(1500)
      INTEGER*4 ID(200),FR(5,12),ERROR,CSET(3,12),NN1(6,30,10),NN2(30),
     1CHSEL(12),BSTCH(12)
      REAL*4 RDATA(3000),SDATA(12,1000),DATTOT(12),T(12,12),MEAN(12,30),
     1K(12,12,30),SIGMA(12,30),RHO(12,12,30),TOTSMP(12),MAXSEP
      REAL*8 CLS(30)
      EQUIVALENCE (ID(51),FR(1,1))
      COMMON CLS,MEAN,SIGMA,K,RHO,FR,NN1,NN2,TOTSMP
      ID(1)=0
      ID(2)=0
      NCD=12
      CALL GADRUN(66000600,3,ID,ERROR)
      IF(ERROR.GT.0) GO TO 85
      DO 10 I=1,3
      DO 10 J=1,12
      CSET(I,J)=FR(I+2,J)
   10 CONTINUE
C
C     READ NUMBER OF SETS OF TRAINING FIELDS
C
      READ (5,15) NUMSET
      DO 150 NUMRUN=1,NUMSET
C
C     READ TOTAL NUMBER OF CLASSES
C
      READ (5,15) NUMCLS
   15 FORMAT(I5,1X,A8)
      DO 80 KLS=1,NUMCLS
      NSMP=0
C
C     READ NUMBER OF TRAINING FIELDS (NN2(KLS)) IN CLASS CLS(KLS)
C
      READ (5,15) NN2(KLS),CLS(KLS)
C
C     INITIALIZE ACCUMULATORS FOR UNNORMALIZED MEAN VECTOR
C     AND COVARIANCE MATRIX
C
      DO 20 I=1,12
      DATTOT(I)=0.
      DO 20 J=1,12
      T(I,J)=0.
   20 CONTINUE
      NN2KLS=NN2(KLS)
      DO 65 I1=1,NN2KLS
C
C     READ BOUNDARIES OF I1(TH) TRAINING FIELD OF THE KLS(TH) CLASS
C            NN1(1) = FIRST LINE              NN1(4) = COLUMN INCREMENT
C            NN1(2) = FIRST COLUMN            NN1(5) = LAST LINE
C            NN1(3) = LAST COLUMN             NN1(6) = LINE INCREMENT
C
      READ (5,25)(NN1(I,KLS,I1),I=1,6)
   25 FORMAT(6I5)
C
C     COMPUTE NUMBER OF SAMPLES PER CHANNEL PER LINE
C
      N1=NN1(1,KLS,I1)
```

```
      N2=NN1(2,KLS,I1)
      N3=NN1(3,KLS,I1)
      N4=NN1(4,KLS,I1)
      N5=NN1(5,KLS,I1)
      N6=NN1(6,KLS,I1)
      NSD=((N3-N2)/N4)+1
      NRQ=NSD+6
C
C     PREPARE BLCCK(I) FOR GADLIN
C
      DO 30 I=2,4
      BLOCK(I)=NN1(I,KLS,I1)
   30 CONTINUE
C
C     READ IN DATA FROM I2(TH) LINE OF I1(TH) TRAINING FIELD
C     OF THE KLS(TH) CLASS
C
      DO 60 I2=N1,N5,N6
      BLOCK(1)=I2
      CALL GADLIN(BLOCK,CSEL,CSET,ID,3,NCD,NRQ,BDATA,RDATA,ROLL,ERROR)
      IF(ERROR.GT.0)GO TO 90
C
C     SAVE THE DATA
C
      DO 45 I=1,12
      KK2=(I-1)*NRQ
      DO 45 J=1,NSD
      SDATA(I,J)=RDATA(J+KK2)
   45 CONTINUE
C
C     ACCUMULATE MEAN VECTOR
C
      DO 50 I=1,12
      DO 50 J=1,NSD
      DATTOT(I)=DATTOT(I)+SDATA(I,J)
   50 CONTINUE
C
C     ACCUMULATE COVARIANCE MATRIX
C
      DO 55 I=1,12
      DO 55 J=1,12
      DO 55 L=1,NSD
      T(I,J)=T(I,J)+SDATA(I,L)*SDATA(J,L)
   55 CONTINUE
      NSMP=NSMP+NSD
   60 CONTINUE
   55 CONTINUE
      TOTSMP(KLS)=NSMP
C
C     DIVIDE ACCUMULATORS BY TOTAL NUMBER OF SAMPLES TO OBTAIN
C     MEAN VECTOR AND COVARIANCE MATRIX OF KLS(TH) CLASS
C
      DO 70 I=1,12
      MEAN(I,KLS)=DATTOT(I)/TOTSMP(KLS)
   70 CONTINUE
      DO 71 I=1,12
      DO 71 J=I,12
      K(I,J,KLS)=T(I,J)/TOTSMP(KLS)-MEAN(I,KLS)*MEAN(J,KLS)
```

```
      K(J,I,KLS)=K(I,J,KLS)
   71 CONTINUE
      DO 75 I=1,12
      SIGMA(I,KLS)=SQRT(K(I,I,KLS))
      DO 75 J=1,I
      RHO(I,J,KLS)=K(I,J,KLS)/SQRT(K(I,I,KLS)*K(J,J,KLS))
      RHO(J,I,KLS)=RHO(I,J,KLS)
   75 CONTINUE
   80 CONTINUE
      CALL STATPT(NUMCLS)
      GO TO 100
   85 CALL ERGDRN(ERROR)
      GO TO 955
   90 CALL ERGDLN(ERROR)
      GO TO 955
  100 CONTINUE
C
C     READ NUMBER OF FEATURES TO BE USED IN SELECTION
C
  105 READ (5,25) NF
      IF(NF.EQ.0)GO TO 95
      WRITE (6,1000) NF
 1000 FORMAT('1FEATURE SELECTION FOR',I5,'     CHANNELS')
C
C     INITIALIZE CHANNEL SELECTORS
C
      DO 110 I=1,NF
      CHSEL(I)=I
  110 CONTINUE
      LC=NF
      KOUNTR=0
      II=1
      CD=NCD
      FEAT=NF
      MAXSEP=0.
      NCC=1
      LENGTH=NF*(NF+1)/2
C
C     CALCULATE NUMBER OF DIFFERENT COMBINATIONS TO BE TESTED
C
      NUMCHK=GAMMA(CD+1.)/(GAMMA(FEAT+1.)*GAMMA(CD-FEAT+1.))+.1
C
C     PERFORM DIVERGENCE AND SEPARABILITY CALCULATIONS
C
      CALL RTIME(JBEGIN)
  115 CALL DIVCAL(K,MEAN,NUMCLS,NF,LENGTH,CHSEL,BSTCH,MAXSEP,NCC)
      KOUNTR=KOUNTR+1
C
      NCC=NF
C     CHECK WHETHER ALL POSSIBLE COMBINATIONS HAVE BEEN EXAMINED
C
      IF(KOUNTR.EQ.NUMCHK)GO TO 135
      CHSEL(LC)=CHSEL(LC)+1
      IF(CHSEL(LC).LE.NCD)GO TO 115
  120 NN=LC-II
      CHSEL(NN)=CHSEL(NN)+1
      NCC=NN
      IF(CHSEL(NN).GT.(NCD-II))GO TO 130
```

```
      NR=NN+1
      DO 125 JJ=NR,NF
      CHSEL(JJ)=CHSEL(JJ-1)+1
  125 CONTINUE
      II=1
      GO TO 115
  130 II=II+1
      GO TO 120
  135 CALL RTIME(JEND)
      WRITE(6,9595)(CLS(KLS),KLS=1,NUMCLS)
 9595 FORMAT('1CLASSES:'/5X,10(A8,3X)/5X,10(A8,3X)/5X,10(A8,3X))
      WRITE (6,9000) NF
 9000 FORMAT('0FEATURE SELECTION FOR',I5,'     CHANNELS')
      WRITE (6,140) NF,(BSTCH(I),I=1,NF)
  140 FORMAT('0THE BEST',I5,'  CHANNEL(S) ARE',12I5)
      ELAPTM=(JBEGIN-JEND)/100.
      WRITE (6,200) NUMCLS,NF,ELAPTM
  200 FORMAT(//' ELAPSED TIME FOR FEATURE SELECTION FOR',I5,'   CLASSES,'
     1,I5,'   CHANNEL(S), IS',F8.2,' SECONDS')
      WRITE (6,141)
  141 FORMAT(1H1)
      GO TO 105
   95 CONTINUE
  150 CONTINUE
  955 STOP
      END
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
          SUBROUTINE STATPT(NUMCLS)
          INTEGER*4 FR(5,12),NN1(6,30,10),NN2(30)
          REAL*4 MEAN(12,30),SIGMA(12,30),K(12,12,30),RHO(12,12,30),
         1TOTSMP(12)
          REAL*8 CLS(30)
          COMMON CLS,MEAN,SIGMA,K,RHO,FR,NN1,NN2,TOTSMP
          DO 10 KLS=1,NUMCLS
          WRITE (6,100) KLS, CLS(KLS)
      100 FORMAT('1MEAN VECTOR, COVARIANCE MATRIX, AND CORRELATION MATRIX FO
         1R CLASS NUMBER',I3,' (',A8,')'/)
          NTF=NN2(KLS)
          DO 35 I1=1,NTF
          WRITE (6,95) I1,NN1(1,KLS,I1),NN1(5,KLS,I1),NN1(6,KLS,I1),NN1(2,KL
         1S,I1),NN1(3,KLS,I1),NN1(4,KLS,I1)
       95 FORMAT(5X,'TRAINING FIELD NUMBER',I4,': LINES',I6,' TO',I6,' (EVER
         1Y',I3,' LINE(S)), SAMPLES',I6,' TO',I6,' (EVERY',I3,' SAMPLE(S))')
       35 CONTINUE
          WRITE (6,96) TOTSMP(KLS)
       96 FORMAT(10X,'TOTAL NUMBER OF DATA POINTS = ',F7.0//)
          WRITE (6,110) (FR(1,I),I=1,12),(FR(2,I),I=1,12)
      110 FORMAT(1X,'SPECTRAL ',12(F5.2,'-    ')/3X,'BAND',4X,12(F5.2,4X)/)
          WRITE (6,120) (MEAN(I,KLS),I=1,12),(SIGMA(I,KLS),I=1,12)
      120 FORMAT(' MEAN',5X,12(F7.2,2X)//' ST DEV',3X,12(F7.2,2X))
          WRITE (6,125)
      125 FORMAT(///' COVARIANCE MATRIX'/)
          WRITE (6,110) (FR(1,I),I=1,12),(FR(2,I),I=1,12)
          DO 20 I=1,12
          WRITE (6,130) FR(1,I),(K(I,J,KLS),J=1,I)
      130 FORMAT(1XF5.2,'-',2X,12(F7.2,2X))
          WRITE (6,140) FR(2,I)
      140 FORMAT(2X,F5.2)
       20 CONTINUE
          WRITE (6,126)
      126 FORMAT(///' CORRELATION MATRIX'/)
          WRITE (6,110) (FR(1,I),I=1,12),(FR(2,I),I=1,12)
          DO 25 I=1,12
          WRITE (6,130) FR(1,I),(RHO(I,J,KLS),J=1,I)
          WRITE (6,140) FR(2,I)
       25 CONTINUE
       10 CONTINUE
          RETURN
          END
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
      SUBROUTINE ERGDRN(ERROR)
      INTEGER*4 ERROR
      WRITE (6,10) ERROR
      GO TO (101,102,103,104,105,106,107,108),ERROR
  101 WRITE (6,1)
      GO TO 109
  102 WRITE (6,2)
      GO TO 109
  103 WRITE (6,3)
      GO TO 109
  104 WRITE (6,4)
      GO TO 109
  105 WRITE (6,5)
      GO TO 109
  106 WRITE (6,6)
      GO TO 109
  107 WRITE (6,7)
      GO TO 109
  108 WRITE (6,8)
  109 WRITE (6,9)
    1 FORMAT(5X,'EXPECTED ID RECORD READ AS END OF FILE RECORD')
    2 FORMAT(5X,'EXPECTED ID RECORD HAD WRONG READ COUNT')
    3 FORMAT(5X,'EXPECTED ID RECORD READ WITH A TAPE PARITY CHECK')
    4 FORMAT(5X,'EXPECTED ID RECORD READ WITH A HARDWARE PARITY CHECK')
    5 FORMAT(5X,'EXPECTED ID RECORD HAD WRONG READ COUNT AND PARITY CHEC
     1K')
    6 FORMAT(5X,'TAPE UNIT NOT ASSIGNED')
    7 FORMAT(5X,'EXPECTED ID RECORD READ FROM CORRECT TAPE AND FILE, BUT
     1CONTAINED WRONG RUN NUMBER')
    8 FORMAT(5X,'RUN WAS NOT FOUND IN ADCRT')
    9 FORMAT(/10X,'EXECUTION TERMINATED')
   10 FORMAT('1ERROR NUMBER',I3,' IN SUBROUTINE GADRUN'/)
      RETURN
      ENTRY ERGDLN(ERROR)
      WRITE (6,28) ERROR
      GO TO (111,112,113,114,115,116,117,118,119,120,121,122,123,124,125
     1,126,127),ERROR
  111 WRITE (6,11)
      GO TO 128
  112 WRITE (6,12)
      GO TO 128
  113 WRITE (6,13)
      GO TO 128
  114 WRITE (6,14)
      GO TO 128
  115 WRITE (6,15)
      GO TO 128
  116 WRITE (6,16)
      GO TO 128
  117 WRITE (6,17)
      GO TO 128
  118 WRITE (6,18)
      GO TO 128
  119 WRITE (6,19)
      GO TO 128
  120 WRITE (6,20)
```

```
      GO TO 128
  121 WRITE (6,21)
      GO TO 128
  122 WRITE (6,22)
      GO TO 128
  123 WRITE (6,23)
      GO TO 128
  124 WRITE (6,24)
      GO TO 128
  125 WRITE (6,25)
      GO TO 128
  126 WRITE (6,26)
      GO TO 128
  127 WRITE (6,27)
  128 WRITE (6,9)
   11 FORMAT(5X,'DATA LINE REQUESTED DOES NOT EXIST ON TAPE')
   12 FORMAT(5X,'BYTE COUNT IN REQUESTED DATA RECORD IS INCORRECT')
   13 FORMAT(5X,'PARITY CHECK OCCURED IN READING REQUESTED DATA RECORD')
   14 FORMAT(5X,'HARDWARE ERROR CCCURED IN READING REQUESTED DATA RECORD
     1')
   15 FORMAT(5X,'IN READING REQUESTED DATA RECORD COMBINATION ERROR OF '
     1/5X'INCORRECT BYTE COUNT AND PARITY CHECK OR'/5X'INCORRECT BYTE CO
     2UNT AND HARDWARE ERROR')
   16 FORMAT(5X,'TAPE UNIT NOT ASSIGNED')
   17 FORMAT(5X,'LINE WAS DEFINED LESS THAN OR EQUAL TO ZERO')
   18 FORMAT(5X,'ISAM WAS DEFINED LESS THAN OR EQUAL TO ZERO')
   19 FORMAT(5X,'SINT WAS DEFINED LESS THAN OR EQUAL TO ZERO')
   20 FORMAT(5X,'ISAM IS GREATER THAN LSAM')
   21 FORMAT(5X,'CSEL FLAGS ARE LESS THAN 0 OR GREATER THAN 7')
   22 FORMAT(5X,'NO CHANNELS SELECTED OR SELECTED CHANNELS NOT IN RUN')
   23 FORMAT(5X,'NCR IS GREATER THAN NCD')
   24 FORMAT(5X,'NSD IS LESS THAN NSR + 6')
   25 FORMAT(5X,'DATA IN REQUESTED LINE DOES NOT EXIST')
   26 FORMAT(5X,'DATA CANNOT BE CALIBRATED AS REQUESTED')
   27 FORMAT(5X,'REQUESTED LINE CANNOT BE LOCATED ON TAPE')
   28 FORMAT('1ERROR NUMBER',I3,' IN SUBROUTINE GADLIN'/)
      RETURN
      END
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
      SUBROUTINE DIVCAL(K,MEAN,NUMCLS,NF,LENGTH,CHSEL,BSTCH,MAXSEP,NCC)
      REAL*4 K(12,12,1),MEAN(12,1),L(2340),SM(360),SK(2340),SUMK(78)
      REAL*4 MAXSEP,SUML(78)
      INTEGER*4 CHSEL(12),BSTCH(12)
      DO 5 I=1,LENGTH
    5 SUMK(I)=0.
      NSK=-LENGTH
      NSM=-NF
      DO 20 KLS=1,NUMCLS
      NSK=NSK+LENGTH
      NSM=NSM+NF
      II=NSM+NCC-1
      III=NSK+NCC*(NCC-1)/2
      DO 10 I=NCC,NF
      II=II+1
C
C     CONVERT MATRIX OF MEANS INTO VECTOR
C
      SM(II)=MEAN(CHSEL(I),KLS)
      DO 10 J=1,I
      III=III+1
C
C     CONVERT MATRIX OF COVARIANCE MATRICES INTO VECTOR
C
   10 SK(III)=K(CHSEL(I),CHSEL(J), KLS)
      NSP1=NSK+1
      IER=0
C
C     PERFORM CHOLESKY DECOMPOSITION OF MATRIX
C
      CALL LUDECF(SK(NSP1),L(NSP1),NF,DD1,DD2,IER)
      IF(IER.NE.0)RETURN
C
C     DIAGONALS RETURNED BY LUDECP ARE RECIPROCAL
C
      ND=NSK
      DO 15 J=1,NF
      ND=ND+J
   15 L(ND)=1./L(ND)
C
C     ACCUMULATE SUM OF K MATRICES
C
      NL=NSK
      DO 17 I=1,LENGTH
      NL=NL+1
   17 SUMK(I)=SUMK(I)+SK(NL)
   20 CONTINUE
C
C     COMPUTE CHOLESKY DECOMPOSITION OF SUM OF COVARIANCE MATRICES
C
      CALL LUDECP(SUMK,SUML,NF,DDD1,DDD2,IER)
      IF(IER.NE.0) RETURN
C
C     DIAGONALS RETURNED BY LUDECP ARE RECIPROCAL
C
      NDIAG=0
```

```
      DO 25 I=1,NF
      NDIAG=NDIAG+I
   25 SUML(NDIAG)=1./SUML(NDIAG)
C
C     COMPUTE TOTAL INTERCLASS SEPARATION AND COMPARE WITH MAXIMUM
C     SEPARATION OBSERVED SO FAR
C
      TOTSEP=D1(L,SUML,NF,NUMCLS,LENGTH)+D2(L,SM,NF,NUMCLS,LENGTH)
      IF(TOTSEP.LE.MAXSEP) RETURN
      MAXSEP=TOTSEP
      WRITE (6,151) MAXSEP,(CHSEL(I),I=1,NF)
  151 FORMAT('0****TOTAL SEPARATION = ',1PE14.8,'       CHANNELS:',12I5)
      DO 50 I=1,NF
   50 BSTCH(I)=CHSEL(I)
      RETURN
      END
```

```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
      REAL FUNCTION D1*4(L,SUML,NF,NUMCLS,LENGTH)
C
C     CALCULATION OF SUM OVER J OF ||L(J) INVERSE * SUM L||**2
C     BY BACK SUBSTITUTION
C
      REAL*4 L(1),SUML(1),C(78),SM(1),DII(12),D(12)
      SUMC=0.
      NSK=1-LENGTH
      DO 20 KLS=1,NUMCLS
      NSK=NSK+LENGTH
      CALL CK(L(NSK),SUML,C,NF)
      DO 10 II=1,LENGTH
      CII=C(II)
   10 SUMC=SUMC+CII*CII
   20 CONTINUE
      D1=SUMC-NF*NUMCLS*NUMCLS
      RETURN
      ENTRY D2(L,SM,NF,NUMCLS,LENGTH)
C
C     CALCULATION OF SUM OVER I AND J OF ETA(I,J)(*)*ETA(I,J)
C     WHERE ETA(I,J)=DELTA(I,I)-DELTA(I,J), AND
C     DELTA(I,J)=L(I) INVERSE*M(J),
C     BY BACK SUBSTITUTION
C
      DEE2=0.
      LNSM=1-LENGTH
      LNSV=1-NF
      DO 100 LKLS=1,NUMCLS
      LNSM=LNSM+LENGTH
      LNSV=LNSV+NF
      CALL DJL(L(LNSM),SM(LNSV),DII,NF)
      JNSV=1-NF
      DO 100 JKLS=1,NUMCLS
      JNSV=JNSV+NF
      IF(LKLS.EQ.JKLS) GO TO 100
      CALL DJL(L(LNSM),SM(JNSV),D,NF)
      DO 120 I=1,NF
      ETA=DII(I)-D(I)
  120 DEE2=DEE2+ETA*ETA
  100 CONTINUE
      D2=DEE2
      RETURN
      END
```

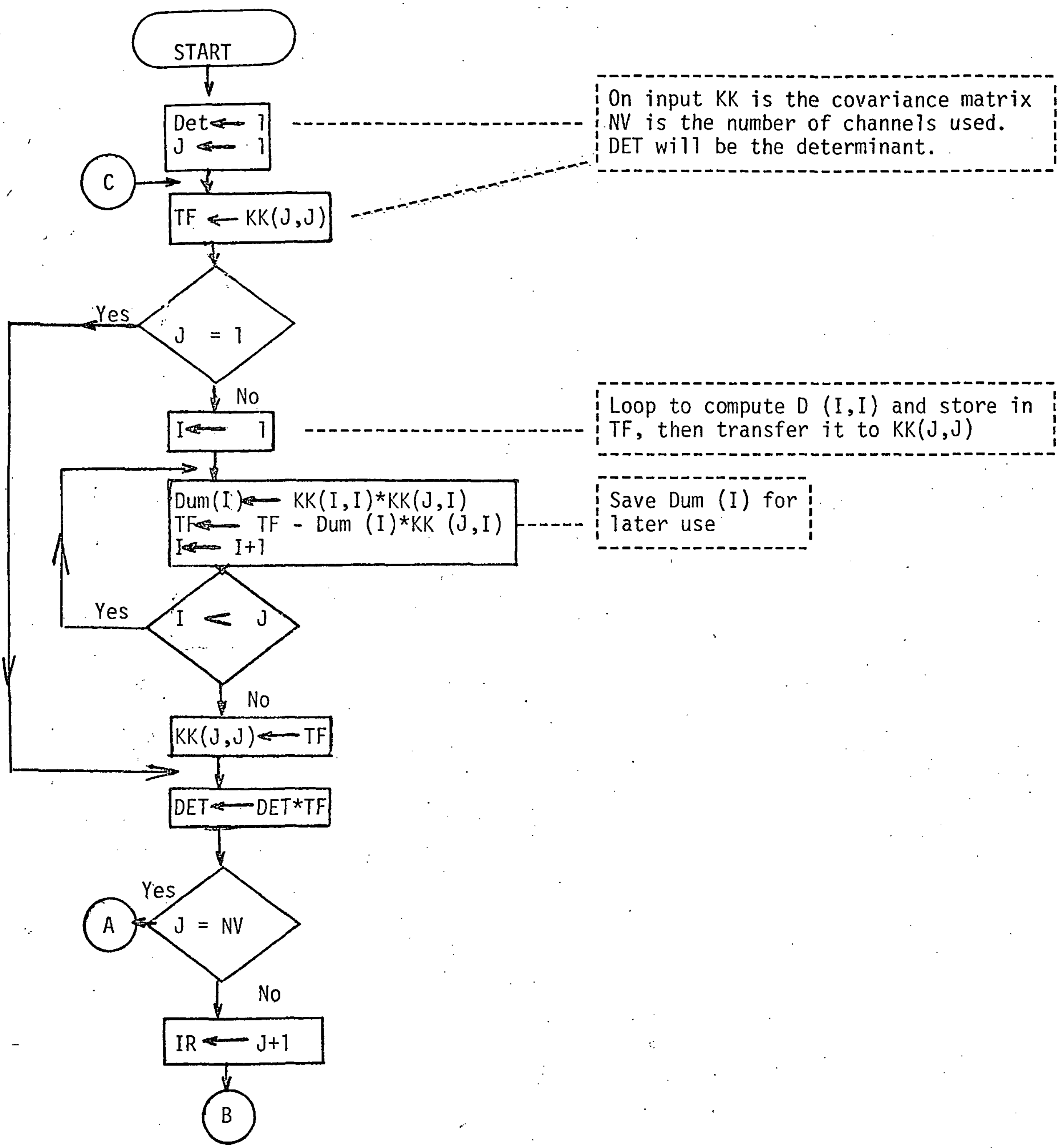
```
COMPILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=60,SIZE=0000K,
                   SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,ID,XREF
      SUBROUTINE CK(L,SUML,C,NF)
C
C     SUBROUTINE TO CALCULATE L(J) INVERSE * SUM L BY BACK SUBSTITUTION
C
      REAL*4 L(1),SUML(1),C(1),SM(1),D(1)
      C(1)=SUML(1)/L(1)
      IF(NF.EQ.1) RETURN
      NBEGIN=1
      DO 100 NRW=2,NF
      NBEGIN=NBEGIN+NRW-1
      NEND=NBEGIN+NRW-1
      NEM1=NEND-1
      NSICR=0
      DO 50 I=NBEGIN,NEM1
      S=0.
      ICR=I-NBEGIN
      NSICR=NSICR+ICR
      NC=NSICR+1
      DO 20 NL=I,NEM1
      NC=NC+ICR
      ICR=ICR+1
   20 S=S+L(NL)*C(NC)
   50 C(I)=(SUML(I)-S)/L(NEND)
  100 C(NEND)=SUML(NEND)/L(NEND)
      RETURN
      ENTRY DJL(L,SM,D,NF)
C
C     SUBROUTINE TO CALCULATE L(I) INVERSE * M(J) BY BACK SUBSTITUTION
C
      D(1)=SM(1)/L(1)
      IF(NF.EQ.1)RETURN
      NBEGIN=1
      DO 200 NRW=2,NF
      NBEGIN=NBEGIN+NRW-1
      NEND=NBEGIN+NRW-1
      NEM1=NEND-1
      S=0.
      I=0
      DO 210 NL=NBEGIN,NEM1
      I=I+1
  210 S=S+L(NL)*D(I)
  200 D(NRW)=(SM(NRW)-S)/L(NEND)
      RETURN
      END
```

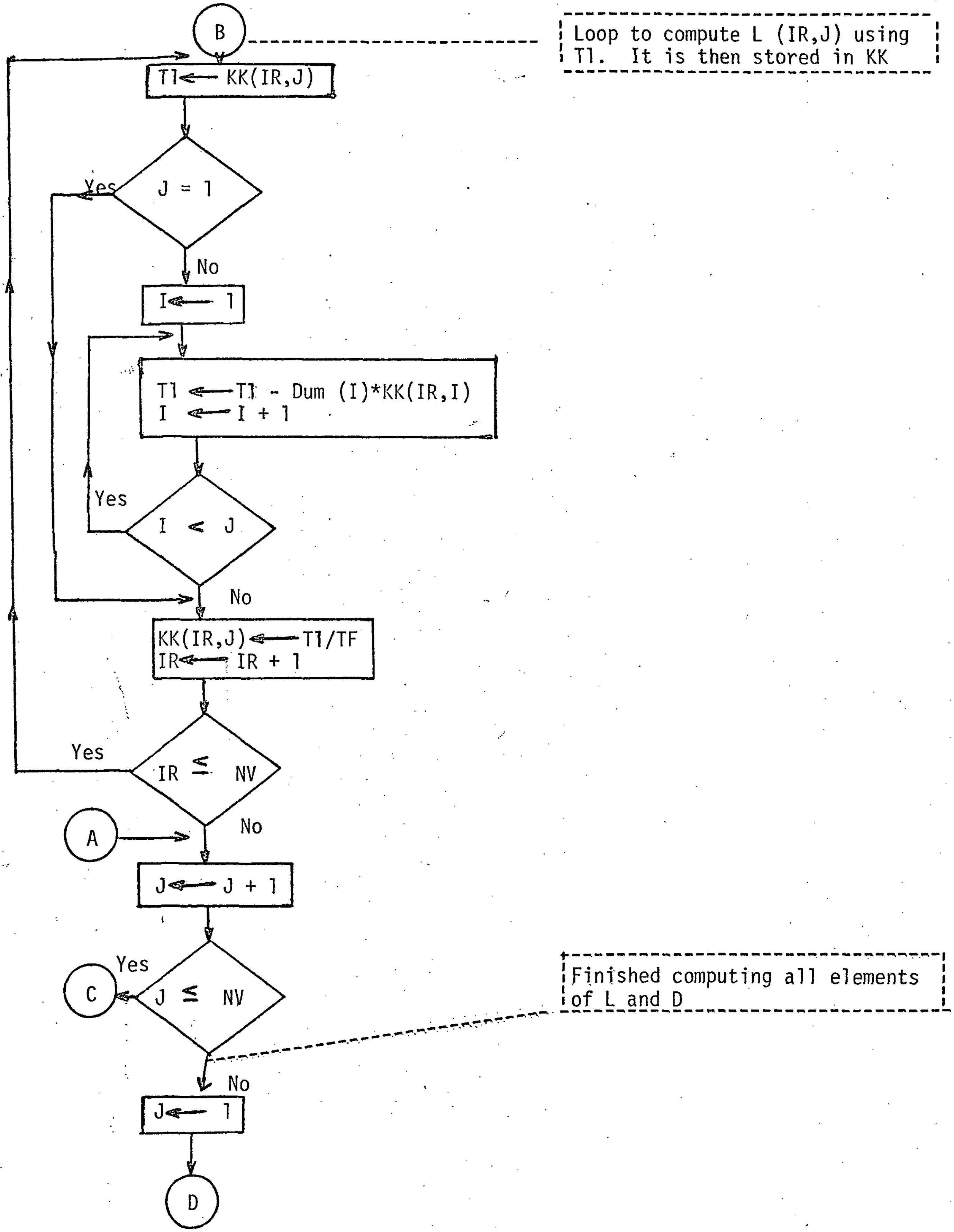
APPENDIX B

Flowcharts and Listing of Subroutines Used in the Classification Processor of LARSYS Employing the Modified Cholesky Decomposition

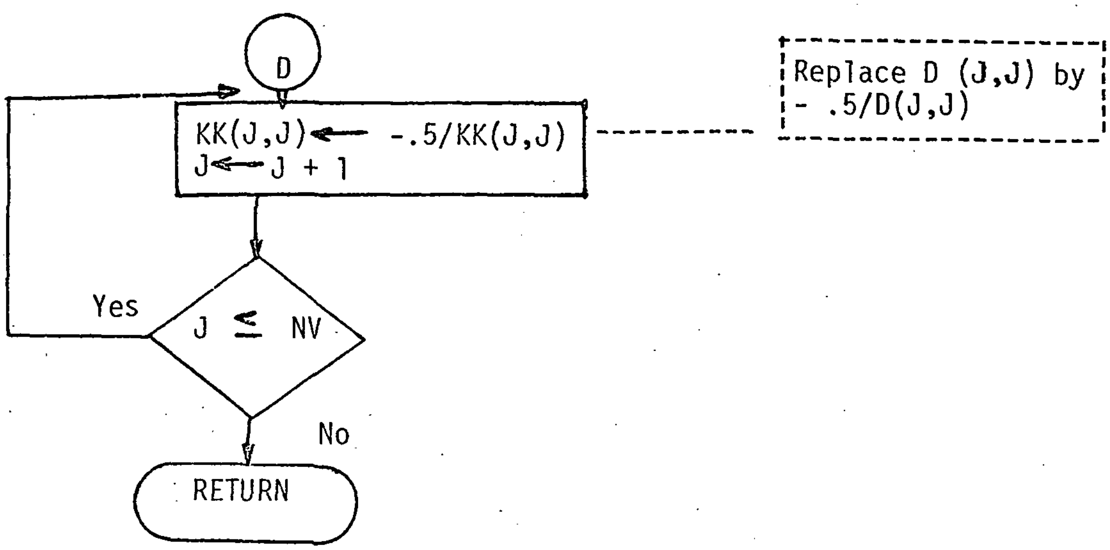
Subroutine MCHLSK

This subroutine computes the modified Cholesky decomposition of the covariance matrix as described earlier. The decomposition overwrites the covariance matrix. The determinant of the covariance matrix is also calculated. The covariance matrix is stored in symmetric storage mode (i.e. - the upper triangular part is stored by columns). The diagonal elements of the diagonal matrix D are stored as $-\frac{1}{2D_{ii}}$ in the diagonal positions of the covariance matrix. The off-diagonal elements of the lower triangular matrix are stored in the corresponding positions in the covariance matrix (the diagonal elements of L are all equal to unity). To simplify the understanding of the flowchart, full matrix notation has been used for matrix elements.


START
Det ← 1
J ← 1
On input KK is the covariance matrix
NV is the number of channels used.
DET will be the determinant.
C
TF ← KK(J,J)
Yes
J = 1
No
I ← 1
Loop to compute D (I,I) and store in
TF, then transfer it to KK(J,J)
Dum(I) ← KK(I,I)*KK(J,I)
TF ← TF - Dum (I)*KK (J,I)
I ← I+1
Save Dum (I) for
later use
Yes
I < J
No
KK(J,J) ← TF
DET ← DET*TF
Yes
A
J = NV
No
IR ← J+1
B



B
Loop to compute L (IR,J) using T1. It is then stored in KK
T1 ← KK(IR,J)
J = 1
Yes
No
I ← 1
T1 ← T1 - Dum (I)*KK(IR,I)
I ← I + 1
Yes
I < J
No
KK(IR,J) ← T1/TF
IR ← IR + 1
Yes
IR ≤ NV
No
A
J ← J + 1
Yes
C
J ≤ NV
Finished computing all elements of L and D
No
J ← 1
D

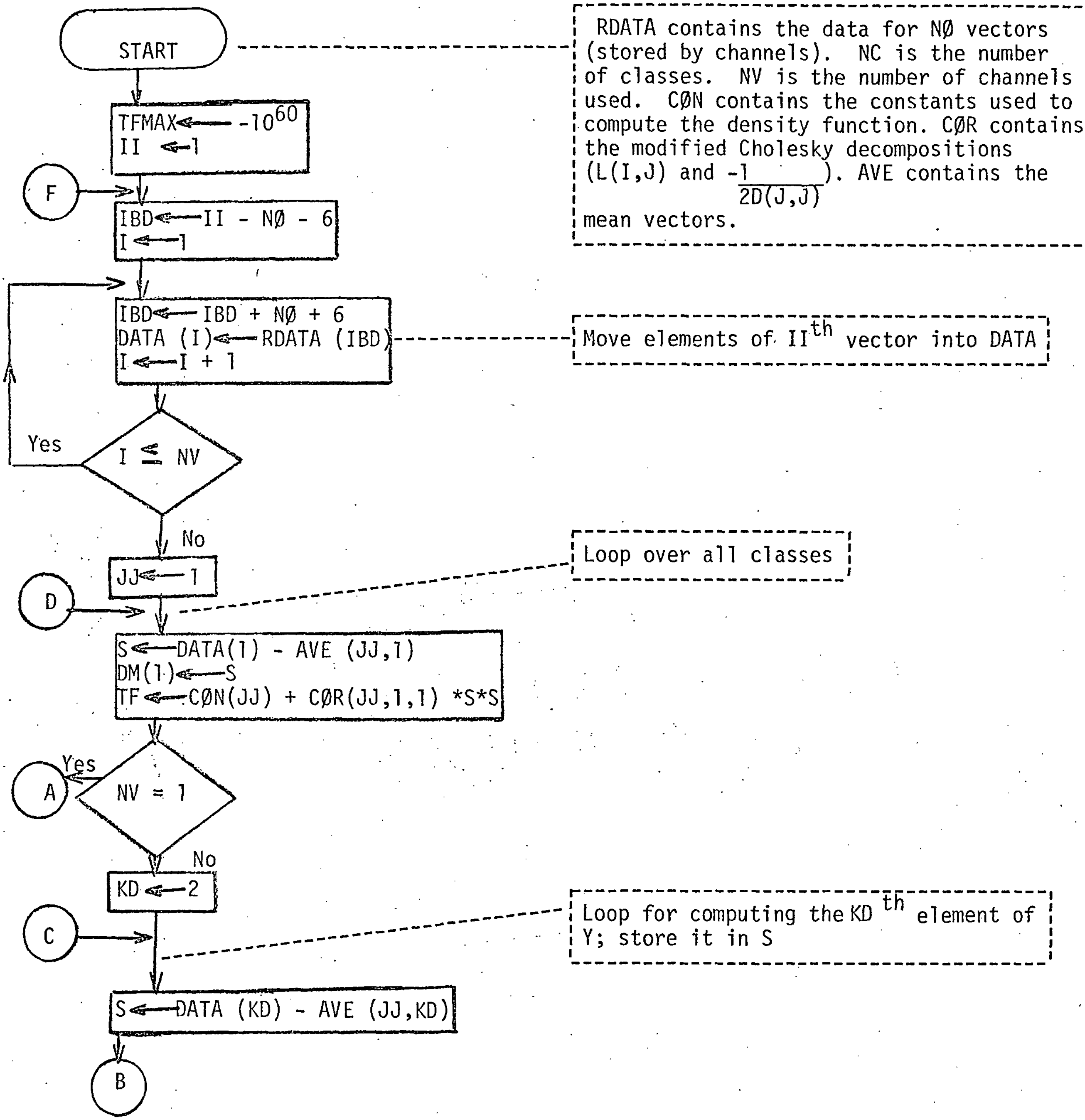

D
KK(J,J) ← -.5/KK(J,J)
J ← J + 1
Replace D (J,J) by - .5/D(J,J)
Yes
J ≤ NV
No
RETURN

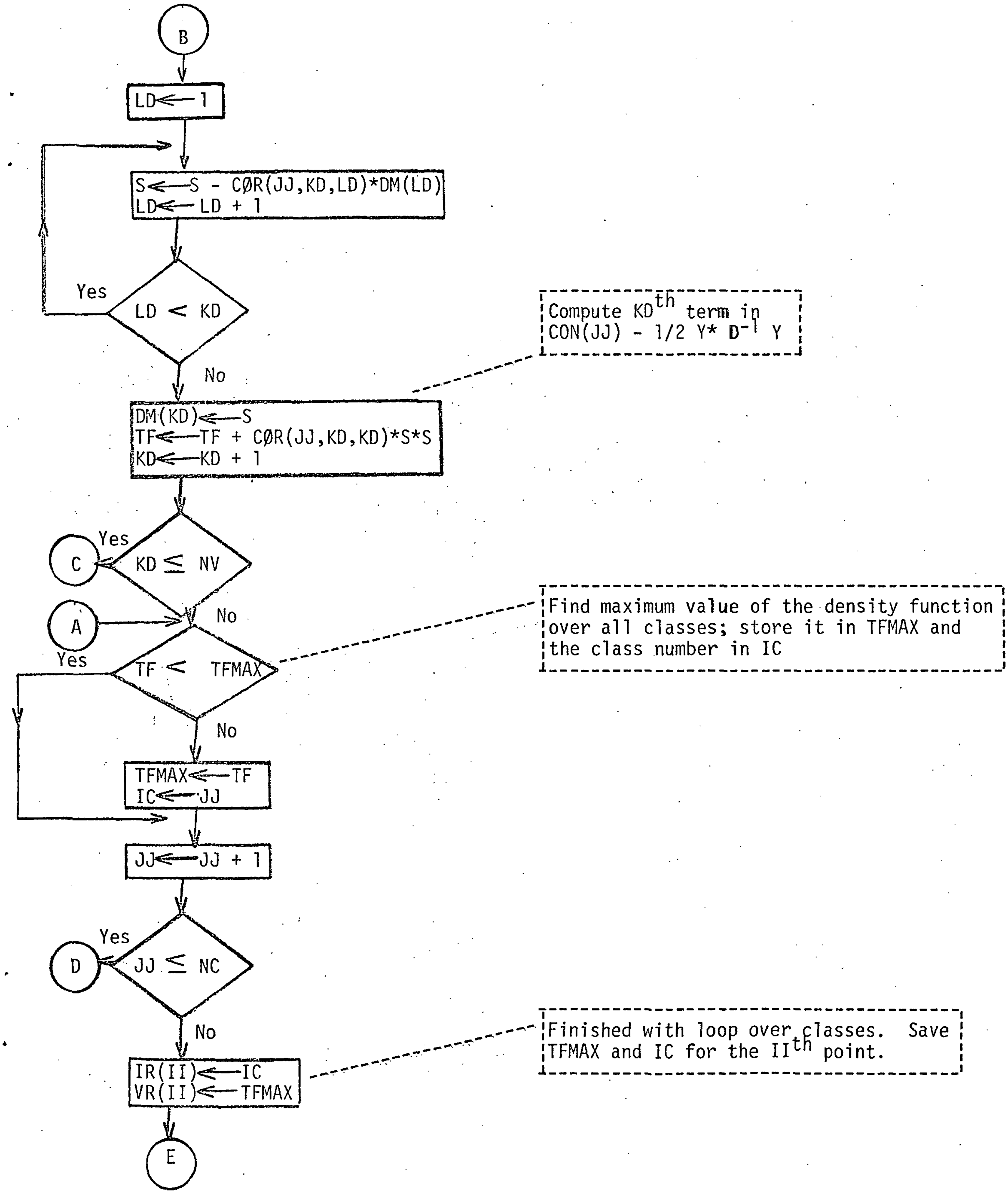
```
        COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=44,SIZE=0000K,
                          SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,XREF
ISN 0002            SUBROUTINE MCHLSK(KK,NV,DUM,DET)
              C
              C  **********************************************************************
              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 WORK AREA OF SIZE NV-1
              C         DET - THE DETERMINANT OF THE COVARIANCE MATRIX.
              C
ISN 0003            REAL KK(1),DUM(1)
ISN 0004            LOGICAL*1 JE1
ISN 0005            JE1=.TRUE.
ISN 0006            J1=0
ISN 0007            JD=0
ISN 0008            DET=1.
              C
              C         LOOP OVER ALL CHANNELS
              C
ISN 0009            DO 10 J=1,NV
ISN 0010            KL=J-1
ISN 0011            L=J+1
ISN 0012            JD=J1
ISN 0013            J1=J1+J
ISN 0014            TF=KK(J1)
ISN 0015            IF (JE1) GO TO 12
ISN 0017            K1=0
              C
              C         COMPUTE THE DIAGONAL ELEMENTS OF D AND STORE IN KK
              C         TEMPORARILY STORE THE PEODUCT KK(I,I)*KK(J,I) IN DUM(I)
              C
ISN 0018            DO 15 I=1,KL
ISN 0019            R=KK(JD+I)
ISN 0020            K1=K1+I
ISN 0021            R1=KK(K1)*R
ISN 0022            TF=TF-R1*R
```

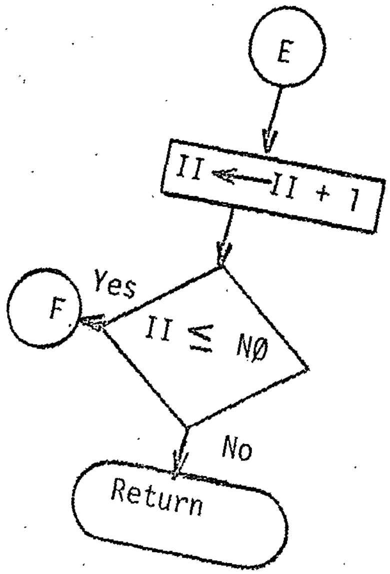
```
ISN 0023            DUM(I)=R1
ISN 0024         15 CONTINUE
ISN 0025            KK(J1      )=TF
ISN 0026         12 CONTINUE
ISN 0027            CET=DET*TF
ISN 0028            IF (L.GT.NV) GO TO 10
ISN 0030            IRD=J1-L+1
                 C
                 C     COMPUTE THE R,J-TH ELEMENT OF L USING T1
                 C
ISN 0031            DO 20 IR=L,NV
ISN 0032            IRD=IRD+IR-1
ISN 0033            T1=KK(IRD+J)
ISN 0034            IF (JE1) GO TO 16
ISN 0036            DO 25 I=1,KL
ISN 0037            T1=T1-DUM(I)*KK(IRD+I)
ISN 0038         25 CONTINUE
ISN 0039         16 KK(IRD+J)=T1/TF
ISN 0040         20 CONTINUE
ISN 0041            JE1=.FALSE.
ISN 0042         10 CONTINUE
ISN 0043            J1=C
                 C
                 C     STORE THE ELEMENTS OF D IN THIS FORM FOR USE IN SUBROUTINE
                 C       CLASS
                 C
ISN 0044            DO 30 J=1,NV
ISN 0045            J1=J1+J
ISN 0046         30 KK(J1)=-.5/KK(J1)
ISN 0047            RETURN
ISN 0048            END
```

Subroutine CLASS

This subroutine classifies NØ data points using the modified Cholesky decomposition. The decomposition is generated by subroutine MCHLSK. The class number selected and the corresponding value of the density function are saved in arrays IR and VR, respectively. Arrays COR and AWE are stored and indexed as vectors, but, for simplicity, we have used matrix notation in the flowchart.


START
RDATA contains the data for NØ vectors (stored by channels). NC is the number of classes. NV is the number of channels used. CØN contains the constants used to compute the density function. CØR contains the modified Cholesky decompositions (L(I,J) and -1/2D(J,J)). AVE contains the mean vectors.
TFMAX ← -10^60
II ← 1
F
IBD ← II - NØ - 6
I ← 1
IBD ← IBD + NØ + 6
DATA (I) ← RDATA (IBD)
I ← I + 1
Move elements of II^th vector into DATA
I ≤ NV
Yes
No
JJ ← 1
Loop over all classes
D
S ← DATA(1) - AVE (JJ,1)
DM(1) ← S
TF ← CØN(JJ) + CØR(JJ,1,1) *S*S
NV = 1
Yes
A
No
KD ← 2
Loop for computing the KD th element of Y; store it in S
C
S ← DATA (KD) - AVE (JJ,KD)
B


B
LD ← 1
S ← S - CØR(JJ,KD,LD)*DM(LD)
LD ← LD + 1
LD < KD
Yes
No
Compute KD^th term in
CON(JJ) - 1/2 Y* D^-1 Y
DM(KD) ← S
TF ← TF + CØR(JJ,KD,KD)*S*S
KD ← KD + 1
KD ≤ NV
Yes
C
No
A
TF < TFMAX
Yes
No
Find maximum value of the density function
over all classes; store it in TFMAX and
the class number in IC
TFMAX ← TF
IC ← JJ
JJ ← JJ + 1
JJ ≤ NC
Yes
D
No
Finished with loop over classes. Save
TFMAX and IC for the II^th point.
IR(II) ← IC
VR(II) ← TFMAX
E


E
II ← II + 1
Yes
F
II ≤ NØ
No
Return

```
      COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=44,SIZE=0000K,
                         SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,XREF
ISN 0002              SUBROUTINE CLASS (RDATA,NV,NO,NC,AVE,COR,IR,VR,CON)
                 C
                 C  *******************************************************************
                 C        MODIFIED CHOLESKY VERSION OF THE CLASSIFICATION ROUTINE
                 C  *******************************************************************
                 C
                 C        RDATA CONTAINS THE  DATA POINTS FOR NV CHANNELS (STORED BY
                 C          CHANNELS)
                 C        NV IS THE NUMBER OF CHANNELS USED
                 C        NO IS THE NUMBER OF POINTS TO BE CLASSIFIED
                 C        NC IS THE NUMBER OF CLASSES
                 C        AVE CONTAINS THE MEAN VECTORS FOR THE CLASSES
                 C        COR CONTAINS THE MODIFIED CHOLESKY DECOMPOSITION
                 C        IR ON OUTPUT WILL CONTAIN THE CLASS NUMBER SELECTED FOR EACH
                 C          OF THE DATA POINTS
                 C        VR ON OUTPUT WILL CONTAIN THE CORRESPONDING VALUE OF THE
                 C          DENSITY FUNCTION FOR USE IN THRESHOLDING.
                 C        CON CONTAINS THE CONSTANTS USED FOR COMPUTING THE DENSITY FUNC.
                 C
ISN 0003              REAL RDATA(1),DATA(30),AVE(1),COR(1),VR(1),CON(1)
ISN 0004              REAL DM(30)
ISN 0005              INTEGER*2 IR(1)
ISN 0006              LOGICAL*1 KD1
ISN 0007              KD1=NV.EQ.1
ISN 0008              DO 200 II=1,NO
ISN 0009              IBD=II-NC-6
ISN 0010              TFMAX=-1.E60
                 C
                 C        MOVE ELEMENTS OF II-TH VECTOR INTO DATA
                 C
ISN 0011              DO 10 I=1,NV
ISN 0012              IBD=IBD+NO+6
ISN 0013           10 DATA(I)=RDATA(IBD)
ISN 0014              IBM=-NV
ISN 0015              LC=0
                 C
                 C        LOOP OVER ALL CLASSES
                 C
ISN 0016              DO 150 JJ=1,NC
ISN 0017              IBM=IBM+NV
```

```
ISN 0018          KM=IBM+1
ISN 0019          S=DATA(1)-AVE(KM)
ISN 0020          DM(1)=S
ISN 0021          LC=LC+1
ISN 0022          TF=CON(JJ)+COR(LC)*S*S
ISN 0023          IF (KD1) GO TO 142
            C
            C        LOOP FOR COMPUTING THE KD-TH ELEMENT OF Y WHICH IS STORED IN S
            C
ISN 0025          DO 145 KD=2,NV
ISN 0026          KM=IBM+KD
ISN 0027          S=DATA(KD)-AVE(KM)
ISN 0028          J1=KD-1
ISN 0029          DO 140 LD=1,J1
ISN 0030          LC=LC+1
ISN 0031      140 S=S-COR(LC)*DM(LD)
ISN 0032          DM(KD)=S
ISN 0033          LC=LC+1
            C
            C        COMPUTE KD-TH TERM IN -.5Y* D**(-1) Y
            C
ISN 0034      145 TF=TF+COR(LC)*S*S
ISN 0035      142 CONTINUE
            C
            C        FIND MAX VALUE OF THE DENSITY FUNCTION OVER ALL CLASSES
            C
ISN 0036          IF (TF.LE.TFMAX) GO TO 150
ISN 0038          TFMAX=TF
ISN 0039          IC=JJ
ISN 0040      150 CONTINUE
ISN 0041          IR(II)=IC
ISN 0042      200 VR(II)=TFMAX
ISN 0043          RETURN
ISN 0044          END
```

Subroutines COVIN AND STATS

These LARSAA routines were slightly modified to employ the modified Cholesky decompositions. COVIN now calls MCHLSK and does not compute K^{-1}. STATS now employs MCHLSK to calculate the determinants.

```
       COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=44,SIZE=0000K,
                          SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,XREF
  ISN 0002            SUBROUTINE COVIN (NC,NV,IVAR,COR,R,DET,CON)
                C
                C        COVIN FOR MODIFIED CHOLESKY DECOMPOSITION.
                C
  ISN 0003            REAL CON(1),COR(IVAR,NC),DET(1),R(NV)
  ISN 0004            AL6=NV*ALOG(6.283185)
  ISN 0005            DO 229 I=1,NC
  ISN 0006            CALL MCHLSK(COR(1,I),NV,R,DET(I))
  ISN 0007            IF (DET(I).GT.0.) GO TO 150
  ISN 0009            WRITE (6,1000) I
  ISN 0010       1000 FORMAT (/' THE COVARIANCE MATRIX FOR CLASS ',I2,' IS SINGULAR. PRO
                     1GRAM CANNOT PROCEED')
  ISN 0011            DET(1)=-1.
  ISN 0012            RETURN
  ISN 0013        150 CON(I)=-.5*(ALOG(DET(I))+AL6)
  ISN 0014        229 CONTINUE
  ISN 0015            RETURN
  ISN 0016            END
```

```
          COMPILER OPTIONS - NAME=  MAIN,OPT=00,LINECNT=44,SIZE=0000K,
                             SOURCE,EBCDIC,NOLIST,NODECK,LOAD,MAP,NOEDIT,NOID,XREF
               C*****//STATS**************************************************************
               C                                                                          *
               C STATS       REVISED 06/10/71 PWS                                         *
               C                                                                          *
               C             READ STATISTICS FROM MAPTAP AND PRINT IF REQUESTED           *
               C                                                                          *
               C**************************************************************************
ISN 0002             SUBROUTINE  STATS(COVMTX,AVEMTX,VARSIZ,CON,R,W1,W2,COR)
               C**************************************************************************
               C                                                                          *
               C DEFINE PROGRAM VARIABLES                                                 *
               C                                                                          *
               C**************************************************************************
ISN 0003             COMMON/GLOCOM/  CONPUT,DATAPE,FILESV,HEAD,ID,MAPSAV,MAPTAP,
                    *                MAXCHA,MAXCLS,PAGSIZ,READIN,SAVTAP
               C
ISN 0004             INTEGER*4       CONPUT,DATAPE,FILESV,HEAD(79),ID(200),MAPSAV,
                    *                MAPTAP,MAXCHA,MAXCLS,PAGSIZ,READIN,SAVTAP
               C
ISN 0005             REAL*4          FRQCAL(5,30)
ISN 0006             INTEGER*4       COMENT(16),CURRUN,DATE(5),DATSAV,HED1(16),
                    *                HED2(16),TIME(5)
               C
ISN 0007             EQUIVALENCE     (DATSAV,ID(1)),(CURRUN,ID(3))
ISN 0008             EQUIVALENCE     (ID(51),FRQCAL(1,1))
ISN 0009             EQUIVALENCE     (HED1(1),HEAD(8)),(DATE(1),HEAD(26))
ISN 0010             EQUIVALENCE     (HED2(1),HEAD(37)),(TIME(1),HEAD(55))
ISN 0011             EQUIVALENCE     (COMENT(1),HEAD(63))
               C**************************************************************************
               C                                                                          *
               C                    GLOBAL COMMON VARIABLES USED IN AIRANAL               *
               C                                                                          *
               C COMENT = ARRAY FOR STORING A 64 CHARACTER COMMENT                        *
               C CONPUT = COMPUTED-GO-TO VARIABLE (CONPUT,READIN = 1,5 OR 2,15)           *
               C CURRUN = CURRENT RUN NUMBER ON DATA TAPE                                 *
               C DATAPE = FORTRAN UNIT NUMBER FOR AIRCRAFT DATA STORAGE TAPE              *
               C DATE   = ARRAY FOR STORING DATE OF PROGRAM EXECUTION                     *
               C DATSAV = CURRENT MOUNTED DATA TAPE                                       *
               C FILESV = FILE NUMBER LOCATION FOR MAP TAPE                               *
               C FRQCAL = EQUIVALENCED TO ID(51,200)                                      *
```

```
      C FRQCAL(1,N) = LOWER LIMITS OF SPECTRAL BANDS ON TAPE                  *00041000
      C FRQCAL(2,N) = UPPER LIMITS OF SPECTRAL BANDS ON TAPE                  *00042000
      C FRQCAL(3,N) = VALUE OF C0 ON TAPE                                     *00043000
      C FRQCAL(4,N) = VALUE OF C1 ON TAPE                                     *00044000
      C FRQCAL(5,N) = VALUE OF C2 ON TAPE                                     *00045000
      C FRQCAL = WAVELENGTH,C0,C1,C2 FOR THE CHANNELS IN THE CURRENT RUN      *00046000
      C HEAD   = VARIABLE FORMAT FOR PRINTING PAGE HEADER                     *00047000
      C HED1   = ARRAY FOR STORING A 64 CHARACTER FIRST LINE HEADER           *00048000
      C HED2   = ARRAY FOR STORING A 64 CHARACTER SECOND LINE HEADER          *00049000
      C ID     = ARRAY FOR STORING IDENTIFICATION OF AIRCRAFT DATA SET        *00050000
      C ID(1)        = LARS TAPE NUMBER                                       *00051000
      C ID(2)        = FILE NUMBER ON THIS TAPE                               *00052000
      C ID(3-4)      = RUN NUMBER                                             *00053000
      C ID(5)        = CONTINUATION CODE                                      *00054000
      C ID(6)        = NUMBER OF DATA CHANNELS ON ADST (30 MAXIMUM)           *00055000
      C ID(7)        = NUMBER OF DATA SAMPLES PER CHANNEL PER LINE ON ADST    *00056000
      C ID(8-15)     = FLIGHT LINE IDENTIFICATION (16 CHARACTERS)             *00057000
      C ID(16)       = MONTH DATA WAS TAKEN                                   *00058000
      C ID(17)       = DAY DATA WAS TAKEN                                     *00059000
      C ID(18)       = YEAR DATA WAS TAKEN                                    *00060000
      C ID(19-20)    = TIME DATA WAS TAKEN                                    *00061000
      C ID(21)       = ALTITUDE OF AIRCRAFT ABOVE TERRAIN WHEN DATA WAS TAKEN *00062000
      C ID(22)       = GROUND HEADING OF AIRCRAFT WHEN DATA WAS TAKEN         *00063000
      C ID(23-28)    = DATE OF REFORMATING                                    *00064000
      C ID(29-50)    = ALL ZERO (TO BE DEFINED LATER)                         *00065000
      C ID(51)       = LOWER LIMIT OF SPECTRAL BAND ONE ON TAPE               *00066000
      C ID(52)       = UPPER LIMIT OF SPECTRAL BAND ONE ON TAPE               *00067000
      C ID(53)       = VALUE OF C0 BASED ON PREPROCESSING CALCULATIONS(CHAN 1)*00068000
      C ID(54)       = VALUE OF C1 BASED ON PREPROCESSING CALCULATIONS(CHAN 1)*00069000
      C ID(55)       = VALUE OF C2 BASED ON PREPROCESSING CALCULATIONS(CHAN 1)*00070000
      C ID(56-200)   = REPEAT OF ID(51-55) FOR UP TO ID(5) CHANNELS           *00071000
      C ID(56-200)   = 0.0 FOR NONEXISTING DATA CHANNELS                      *00072000
      C MAPSAV = CURRENT MOUNTED MAP TAPE                                     *00073000
      C MAPTAP = FORTRAN UNIT NUMBER FOR MAP TAPE                             *00074000
      C MAXCHA = MAXIMUM NUMBER OF CHANNELS ALLOWED                           *00075000
      C MAXCLS = MAXIMUM NUMBER OF CLASSES ALLOWED                            *00076000
      C PAGSIZ = MAXIMUM NUMBER OF LINES PER PRINTER PAGE                     *00077000
      C READIN = FORTRAN UNIT FOR READING MONITOR + SUPERVISOR CONTROL CARDS  *00078000
      C SAVTAP = FORTRAN UNIT NUMBER FOR SAVE DISK                            *00079000
      C TIME   = ARRAY FOR STORING THE TIME OF PROGRAM EXECUTION              *00080000
      C***********************************************************************00081000
ISN 0012    COMMON/DISCOM/  GRPNAM,CLSNAM,LWORK,THRES,THSCNT,RUNNUM,CALC,      00082000
```

```
                  *          SERIAL,NOMAPS,STATKY,OTRKY,OTSKY,TRCLS,TRFLC,
                  *          TSFLD,TSCLS,PCT,NOFET5,NOCLS5,NOGRPS,NOCLS,COPIES,
                  *          SYMCNT,NOFLD,NOTST,LISTKY,LINE,RDCNT,WRCNT,STOPCD,
                  *          INFO,CLSDAT,SAVMAP,
                  *          SAVLIN,
                  *          FETVEC,CSEL,GRPSTK,BLOCK,
                  *          SYMMTX
              C
ISN 0013           REAL*8    GRPNAM(61),DWORK1(14),DWORK2(15),CLSNAM(60)
ISN 0014           REAL*4    THRES(2,60),RWORK(30)
ISN 0015           INTEGER*4 THSCNT,RUNNUM,CALC,SERIAL,NOMAPS,STATKY,OTRKY,
                  *          OTSKY,TRCLS,TRFLD,TSFLD,TSCLS,PCT,NOFET5,NOCLS5,
                  *          NOGRPS,NOCLS,COPIES,SYMCNT,NOFLD,NOTST,LISTKY,
                  *          CILIN,CLLIN,CLINT,CISAM,CLSAM,CSINT,CRUN,LINE,
                  *          RDCNT,WRCNT,STOPCD,SAVMAP,INFO(17),IWORK(30),
                  *          CLSDAT(5)
ISN 0016           LOGICAL*4 SAVLIN
ISN 0017           INTEGER*2 FETVEC(30),CSEL(30),GRPSTK(60),BLOCK(28),
                  *          HWORK(60),DILIN,DLLIN,DLINT,DISAM,DLSAM,DSINT,IPT,
                  *          LPT,PINT,PISAM,PLSAM,PPINT,PSINT,LEAD,LAG,NPD,NDW,
                  *          PTS,RILIN,RLLIN,RLINT,RISAM,RLSAM,RSINT,CALTRN,
                  *          CALTST
ISN 0018           LOGICAL*1 SYMMTX(64),LWORK(128),WE(2),WE2
              C
ISN 0019           EQUIVALENCE (INFO(1),CRUN),(INFO(4),CILIN),(INFO(5),CLLIN)
ISN 0020           EQUIVALENCE (INFO(6),CLINT),(INFO(7),CISAM),(INFO(8),CLSAM)
ISN 0021           EQUIVALENCE (INFO(9),CSINT)
ISN 0022           EQUIVALENCE (BLOCK(1),RILIN),(BLOCK(2),RLLIN),(BLOCK(3),RLINT)
ISN 0023           EQUIVALENCE (BLOCK(4),RISAM),(BLOCK(5),RLSAM),(BLOCK(6),RSINT)
ISN 0024           EQUIVALENCE (BLOCK(7),IPT),(BLOCK(8),LPT),(BLOCK(9),PINT)
ISN 0025           EQUIVALENCE (BLOCK(10),PISAM),(BLOCK(11),PLSAM)
ISN 0026           EQUIVALENCE (BLOCK(12),PSINT),(BLOCK(13),LEAD)
ISN 0027           EQUIVALENCE (BLOCK(14),LAG),(BLOCK(15),NPD)
ISN 0028           EQUIVALENCE (BLOCK(16),NDW),(BLOCK(17),PPINT)
ISN 0029           EQUIVALENCE (BLOCK(18),PTS)
ISN 0030           EQUIVALENCE (BLOCK(19),DILIN),(BLOCK(20),DLLIN)
ISN 0031           EQUIVALENCE (BLOCK(21),DLINT),(BLOCK(22),DISAM)
ISN 0032           EQUIVALENCE (BLOCK(23),DLSAM),(BLOCK(24),DSINT)
ISN 0033           EQUIVALENCE (BLOCK(25),CALTRN),(BLOCK(26),CALTST)
ISN 0034           EQUIVALENCE (DWORK1(1),DWORK2(1)),(DWORK1(1),RWORK(1))
ISN 0035           EQUIVALENCE (DWORK1(1),HWORK(1)),(DWORK1(1),LWORK(1))
ISN 0036           EQUIVALENCE (LWORK(120),WE(1)),(WE(2),WE2)
```

```
ISN 0037        EQUIVALENCE     (DWORK1(1),IWORK(1))                                    00124100
          C*********************************************************************00125000
          C                                                                    *00126000
          C         DISPLAY COMMON VARIABLES USED IN AIRANAL                   *00127000
          C                                                                    *00128000
          C BLOCK  = AREA TO BE DISPLAYED                                      *00129000
          C CALC   = FLAG TO CALCULATE ONLY EXACT AREA DISPLAYED               *00130000
          C CILIN  = FIRST LINE CLASSIFIED                                     *00131000
          C CISAM  = FIRST SAMPLE CLASSIFIED                                   *00132000
          C CLINT  = LINE INCREMENT CLASSIFIED                                 *00133000
          C CLLIN  = LAST LINE CLASSIFIED                                      *00134000
          C CLSAM  = LAST SAMPLE CLASSIFIED                                    *00135000
          C CLSMTX = REDUCED CLASS NAMES                                       *00136000
          C CLSNAM = NAMES OF CLASSES                                          *00137000
          C CLSPTR = POINTER FOR CLSSTK ARRAY                                  *00138000
          C CLSSTK = STACKED CLASSES                                           *00139000
          C COPIES = NUMBER OF COPIES OF ALL TABLES                            *00140000
          C CRUN   = RUN CLASSIFIED                                            *00141000
          C CSEL   = GROUP POINTER                                             *00142000
          C CSET   = CALIBRATION VALUES                                        *00143000
          C CSINT  = SAMPLE INCREMENT CLASSIFIED                               *00144000
          C DILIN  = DISPLAY FIRST LINE                                        *00145000
          C DISAM  = DISPLAY FIRST SAMPLE                                      *00146000
          C DLINT  = DISPLAY LINE INCREMENT                                    *00147000
          C DLLIN  = DISPLAY LAST LINE                                         *00148000
          C DLSAM  = DISPLAY LAST SAMPLE                                       *00149000
          C DSINT  = DISPLAY SAMPLE INCREMENT                                  *00150000
          C FETVEC = FEATURES USED                                             *00151000
          C GRPNAM = NAMES OF GROUPED CLASSES                                  *00152000
          C GRPSTK = STACKED GROUPS                                            *00153000
          C INFO   = 17 WORD ARRAY FOR STORING LARS12 CARD INFORMATION         *00154000
          C IPT    = DISPLAY FIRST POINT CLASSIFIED                            *00155000
          C LINE   = LINE NUMBER                                               *00156000
          C LPT    = DISPLAY LAST POINT CLASSIFIED                             *00157000
          C NOCLS  = NUMBER OF CLASSES CLASSIFIED                              *00158000
          C NOCLS5 = NUMBER OF CLASSES IN DISPLAY                              *00159000
          C NOFET5 = NUMBER OF FEATURES USED                                   *00160000
          C NOGRPS = NUMBER OF GROUPS IN DISPLAY                               *00161000
          C NOMAPS = NUMBER OF MAPS TO BE PRINTED                              *00162000
          C NOFLD  = NUMBER OF TRAING FIELDS                                   *00163000
          C NOTST  = NUMBER OF TEST FIELDS                                     *00164000
          C OTRKY  = FLAG TO OUTLINE TRAINING FIELDS                           *00165000
```

```
          C OTSKY  = FLAG TO OUTLINE TEST FIELDS                               *00166000
          C PCT    = FLAG TO PRINT TEST FIELD PERCENTAGES                      *00167000
          C PINT   = DISPLAY POINT INTERVAL                                    *00168000
          C PISAM  = FIRST SAMPLE FOR PERCENT CALCULATIONS                     *00169000
          C PLSAM  = LAST SAMPLE FOR PERCENT CALCULATIONS                      *00170000
          C PPINT  = POINT INTERVAL FOR PERCENT CALCULATIONS                   *00171000
          C PSINT  = SAMPLE INTERVAL FOR PERCENT CALCULATIONS                  *00172000
          C PTS    = NUMBER OF SAMPLES CLASSIFIED                              *00173000
          C RDCNT  = SAVTAP READ COUNT                                         *00174000
          C RUNNUM = RUN NUMBER TO DEFINE AREA DISPLAYED                       *00175000
          C SAVMAP = FORTRAN UNIT NUMBER FOR SCRATCH DISK                      *00176000
          C SERIAL = SERIAL NUMBER OF FILE TO BE DISPLAYED                     *00177000
          C STATKY = FLAG TO PRINT REDUCED STATISTICS                          *00178000
          C SYMCNT = NUMBER OF SYMBOLS IN ARRAY SYMMTX                         *00179000
          C SYMMTX = SYMBOLES TO BE USED ON MAP                                *00180000
          C THRES  = THRESHOLD VALUES                                          *00181000
          C THSCNT = NUMBER OF THRESHOLDS INPUT                                *00182000
          C TRCLS  = FLAG TO PRINT TRAINING CLASS PERFORMANCE                  *00183000
          C TRFLD  = FLAG TO PRINT TRAINING FIELD PERFORMANCE                  *00184000
          C TSCLS  = FLAG TO PRINT TEST CLASS PERFORMANCE                      *00185000
          C TSFLD  = FLAG TO PRINT TEST FIELD PERFORMANCE                      *00186000
          C WRCNT  = SAVTAP WRITE COUNT                                        *00187000
          C*********************************************************************00188000
ISN 0038        INTEGER*4       VARSIZ,CNT,INC,BCDTWO/'2'/,LOC,STOP             00189000
ISN 0039        REAL*4          COVMTX(VARSIZ,NOCLS),AVEMTX(NOFETS,NOCLS)       00190000
ISN 0040        REAL*4          R(NOFETS,NOFETS),CON(NOCLS ),W1(1),W2(1),DET,A  00191000
ISN 0041        REAL    *4      COR(1),DEV(30)                                  00192000
ISN 0042        EQUIVALENCE     (CNT,DET),(INC,A)                               00193000
          C*********************************************************************00194000
          C                                                                    *00195000
          C READ STATISTICS FROM MAPTAP                                        *00196000
          C                                                                    *00197000
          C*********************************************************************00198000
ISN 0043    100 READ(MAPTAP,END=109) ((COVMTX(I,J),I=1,VARSIZ),J=1,NOCLS),      00199000
               *                     ((AVEMTX(I,J),I=1,NOFETS),J=1,NOCLS)       00200000
ISN 0044        GO TO 150                                                       00201000
          C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++     00202000
          C ERROR CONDITION                                                +    00203000
          C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++     00204000
ISN 0045    109 WRITE(6,1090)                                                   00205000
ISN 0046   1090 FORMAT (' ****** ERRONEOUS FILE MARK READ WHILE READING STATISTICS00206000
               * FROM MAPTAP--JOB TERMINATED')                                  00207000
```

```
ISN 0047          STOP 109                                                         00208000
            C***********************************************************************00209000
            C                                                                      *00210000
            C PRINT STATISTICS                                                     *00211000
            C                                                                      *00212000
            C***********************************************************************00213000
ISN 0048      150 IF (STATKY .LE. 0) GO TO 181                                     00214000
ISN 0050          CNT = 6+(5+3+2+3*NOFET5)*((NOFET5+11)/12)                        00215000
ISN 0051          CNT = PAGSIZ/CNT                                                 00216000
ISN 0052          INC = CNT                                                        00217000
ISN 0053          DO 180 ICLAS=1,NOCLS                                             00218000
ISN 0054          IF (INC.LT.CNT) GO TO 160                                        00219000
ISN 0056          WRITE (6,HEAD)                                                   00221000
ISN 0057          WRITE(6,1500) SERIAL,CLSDAT                                      00222000
ISN 0058     1500 FORMAT (T16,'SERIAL NUMBER--------'I11,15X'CLASSIFIED '5A4,/'0'/) 00223000
ISN 0059          INC = 0                                                          00224000
ISN 0060      160 WRITE (6,1600) CLSNAM(ICLAS)                                     00225000
ISN 0061     1600 FORMAT ('0'/'0 CLASS  ' A8)                                      00226000
ISN 0062          DO 170 LOC = 1,NOFET5,12                                         00227000
ISN 0063          STOP = LOC +11                                                   00228000
ISN 0064          IF (STOP.GT.NOFET5) STOP = NOFET5                                00229000
ISN 0066          WRITE (6,1601) (FRQCAL(1,FETVEC(I)),I=LOC,STOP)                  00230000
ISN 0067          WRITE (6,1602) (FRQCAL(2,FETVEC(I)),I=LOC,STOP)                  00231000
ISN 0068     1601 FORMAT ('0',7X,12(F6.2,' - '))                                   00232000
ISN 0069     1602 FORMAT (10X,12(F6.2,3X))                                         00233000
ISN 0070      170 WRITE (6,1700) (AVEMTX(I,ICLAS),I=LOC,STOP)                      00234000
ISN 0071     1700 FORMAT ('0MEAN'3X,12F9.2)                                        00235000
ISN 0072          WRITE (6,1701)                                                   00236000
ISN 0073     1701 FORMAT ('0 CORRELATION MATRIX')                                  00237000
ISN 0074          JK = 0                                                           00238000
ISN 0075          DO 175 J=1,NOFET5                                                00239000
ISN 0076          DO 173 K=1,J                                                     00240000
ISN 0077          JK = JK+1                                                        00241000
ISN 0078      173 CONTINUE                                                         00242000
ISN 0079          DEV(J) = SQRT(ABS(COVMTX(JK,ICLAS)))                             00243000
ISN 0080      175 CONTINUE                                                         00244000
ISN 0081          JK  =0                                                           00245000
ISN 0082          DO 179 J=1,NOFET5                                                00246000
ISN 0083          DO 179 K=1,J                                                     00247000
ISN 0084          JK = JK+1                                                        00248000
ISN 0085          COR(JK) = 0.0                                                    00249000
ISN 0086          IF (DEV(K)+DEV(J).LT.1.0E-25) GO TO 179                          00250000
```

```
ISN 0088          COR(JK) = COVMTX(JK,ICLAS)/(DEV(J)*DEV(K))                         00251000
ISN 0089      179 CONTINUE                                                           00252000
ISN 0090          CALL WRTMTX(COR,NOFETS,FRQCAL,BCDTWO,FETVEC)                       00253000
ISN 0091          INC = INC+1                                                        00254000
ISN 0092      180  CONTINUE                                                          00255000
           C************************************************************************00256000
           C                                                                        *00257000
           C CALCULATE CONSTANT TERMS                                               *00258000
           C                                                                        *00259000
           C************************************************************************00260000
ISN 0093      181 IF (THSCNT .LE. 0) GO TO 190                                       00261000
ISN 0095          A=NCFETS*ALOG(6.283185)
ISN 0096          DO 186 I=1,NCCLS
ISN 0097          CALL MCHLSK(COVMTX(1,I),NOFETS,R,DET)
ISN 0098      186 CON(I)=-.5*(A+ALOG(DET))
ISN 0099      190 RETURN                                                             00272000
ISN 0100          END                                                                00273000
```

APPENDIX C

Flowcharts of Routines Necessary to Employ the Modified Cholesky Decomposition in Divergence Calculations

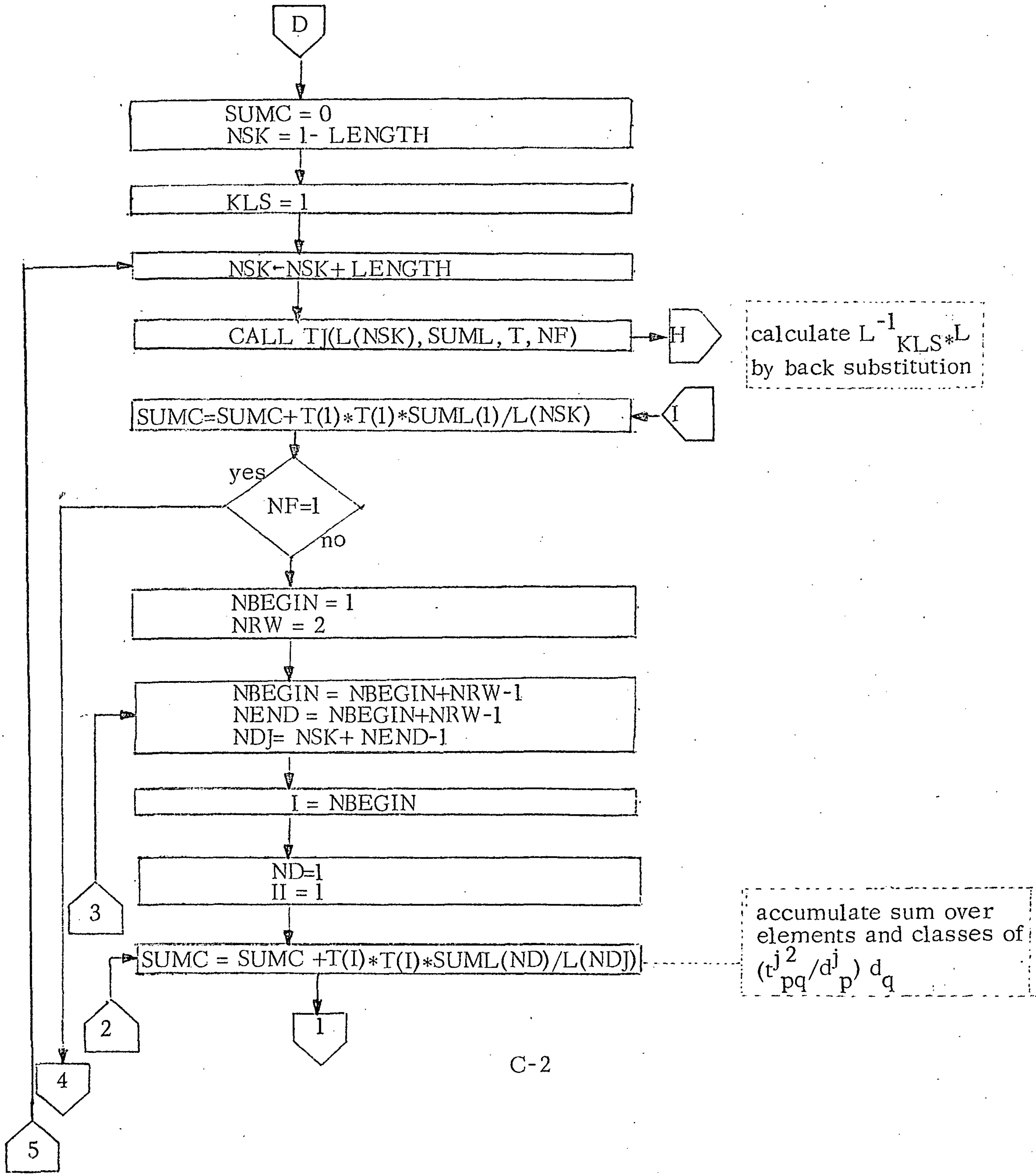
Modified CHOLESKY Feature Selection Method.

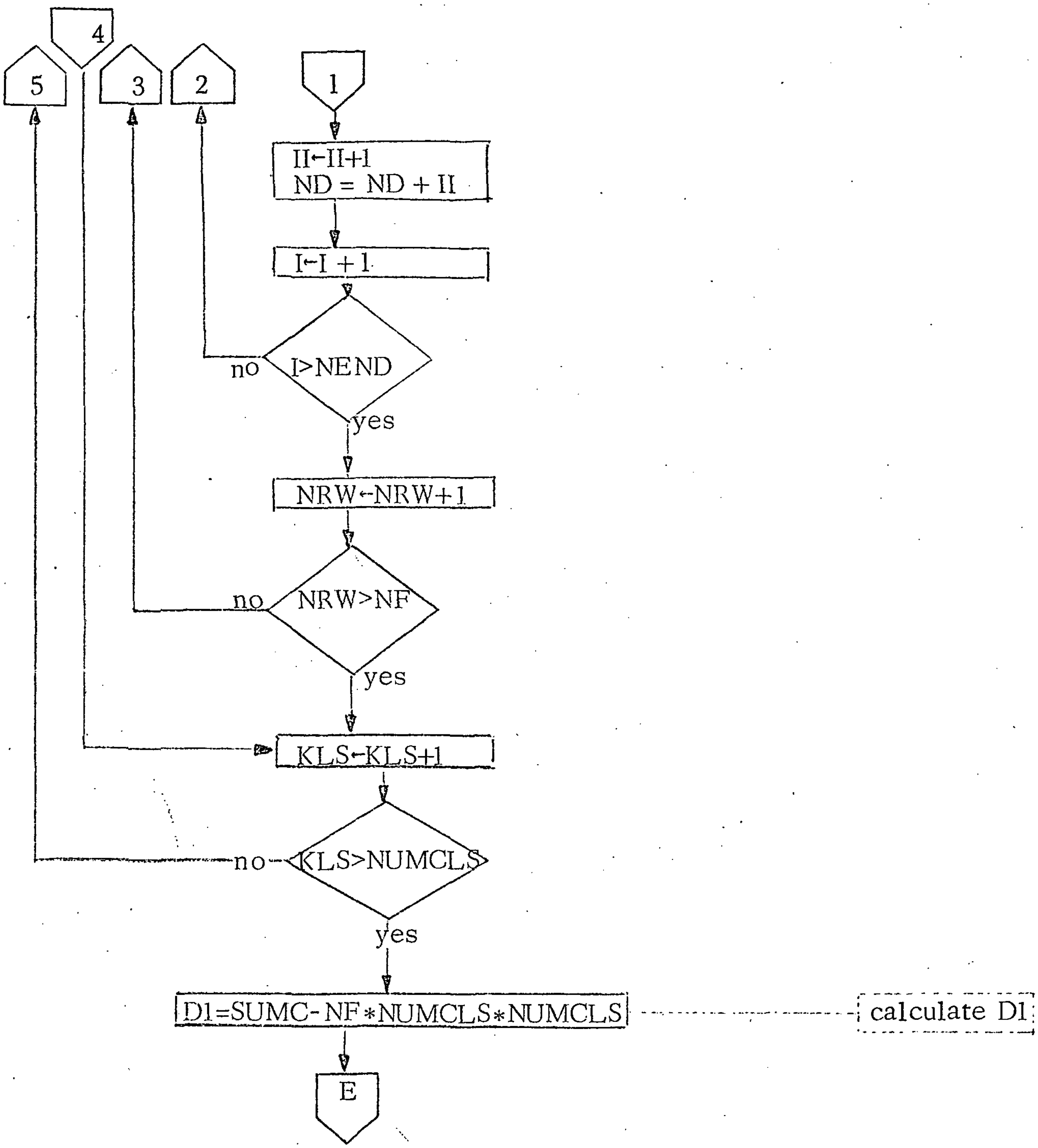
The accompanying flow charts detail the method of feature selection using the modified Cholesky decomposition. DIVCAL (Appendix A, pp. A-1 to A-7) will be modified in that the call to LUDECP will be replaced with a call to a routine which will decompose K into LDL* instead of LL*. This may be accomplished by a subroutine such as MCHLSK (see Appendix B), returning from the routine prior to the modification of the diagonal elements (i.e., before the step preceding off-page connector D on page B-3). Also, the steps in DIVCAL obtaining the reciprocal of the diagonal elements of L, returned by LUDECP, will be removed. The form of D1 and D2 will be changed, although the structure will remain quite similar.

Although the summations are given in matrix form for reasons of clarity, the calculations should be performed using singly-subscripted variables.

This modification has not yet been tested but the logic, as evident from the flowcharts, almost parallels that of the unmodified Cholesky method. For simplicity, and to save storage, the elements of D_j and D are stored as the diagonal elements of L_j and L respectively.

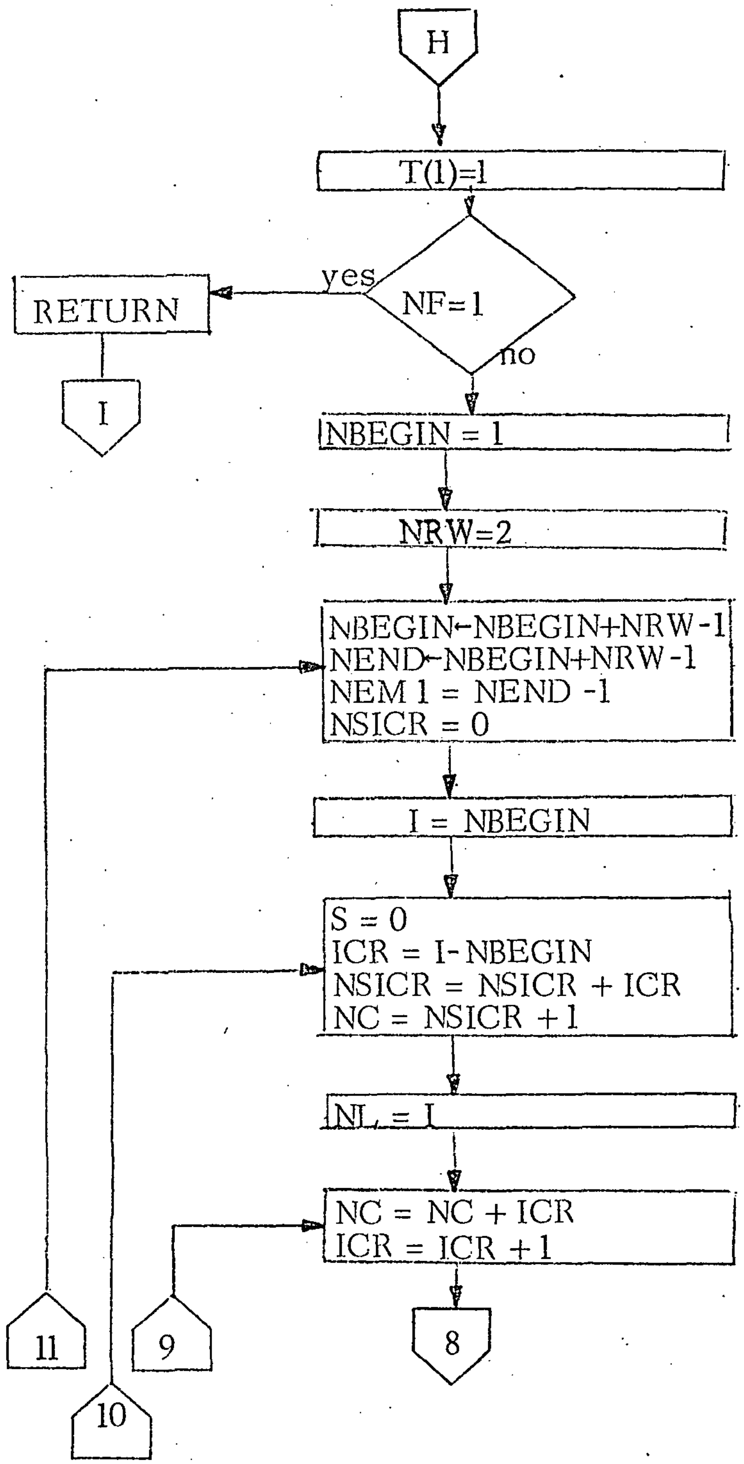
DI(L, SUML, NF, NUMCLS, LENGTH)

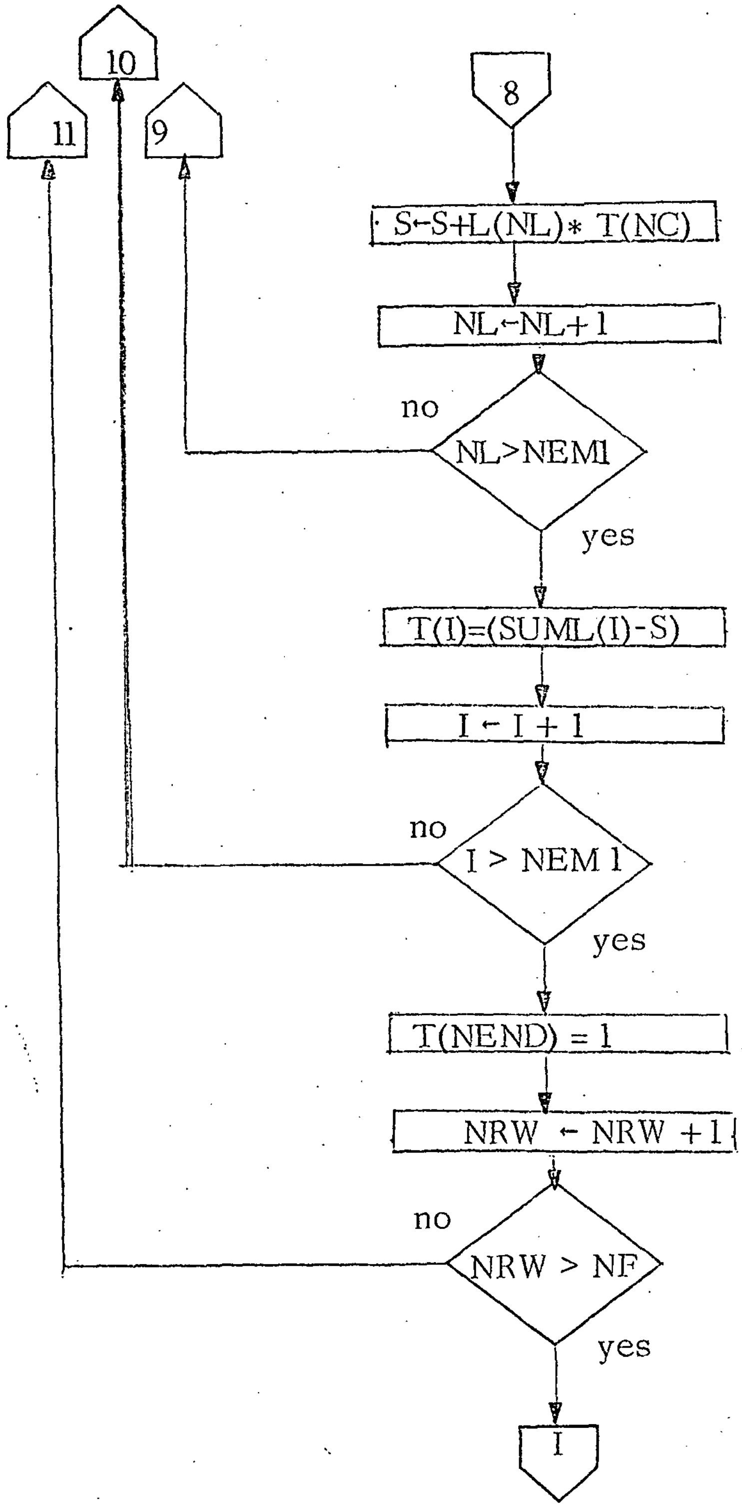




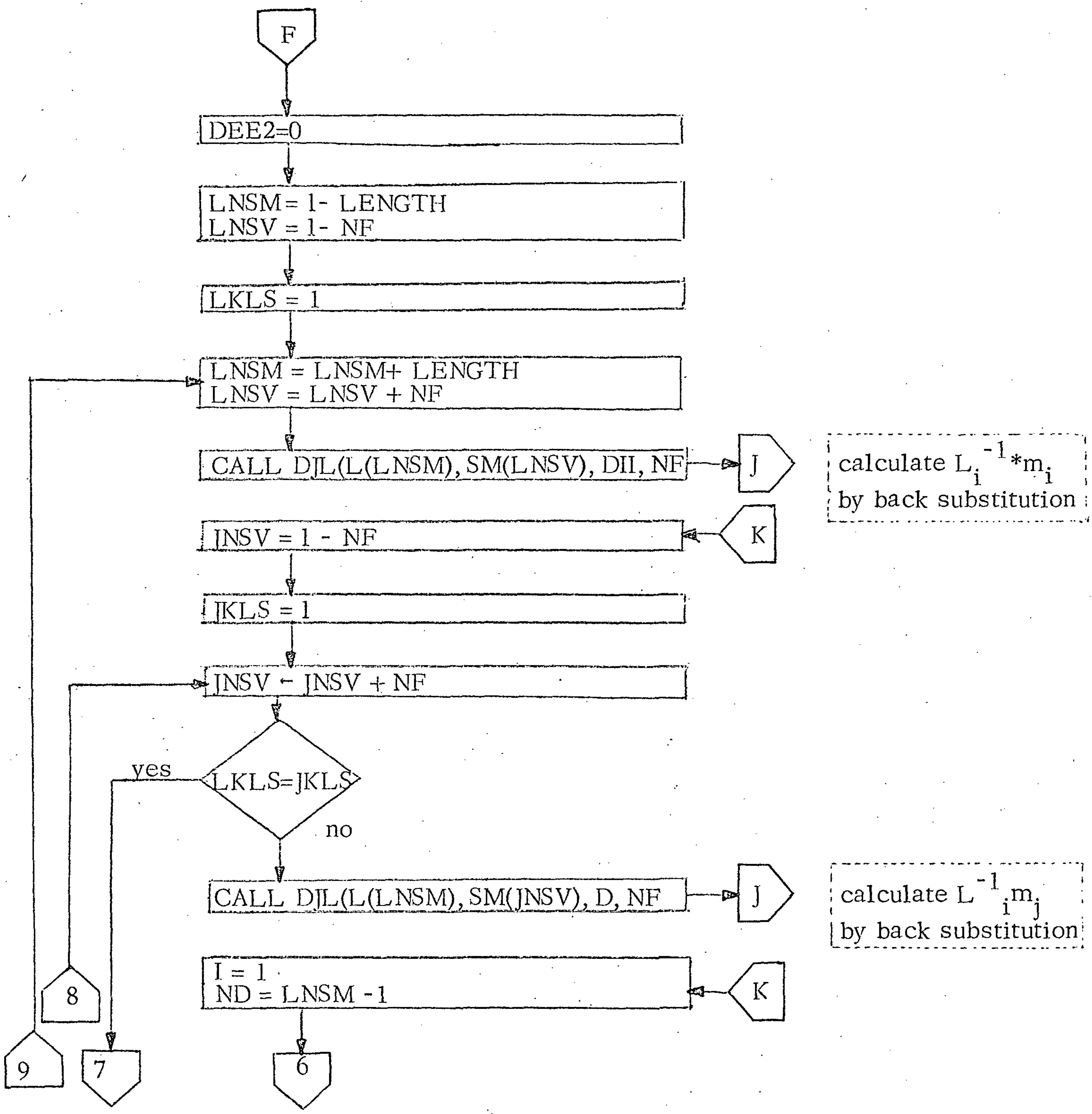
4
5
3
2
1
II←II+1
ND = ND + II
I←I + 1
no
I>NEND
yes
NRW←NRW+1
no
NRW>NF
yes
KLS←KLS+1
no
KLS>NUMCLS
yes
D1=SUMC-NF*NUMCLS*NUMCLS
calculate D1
E

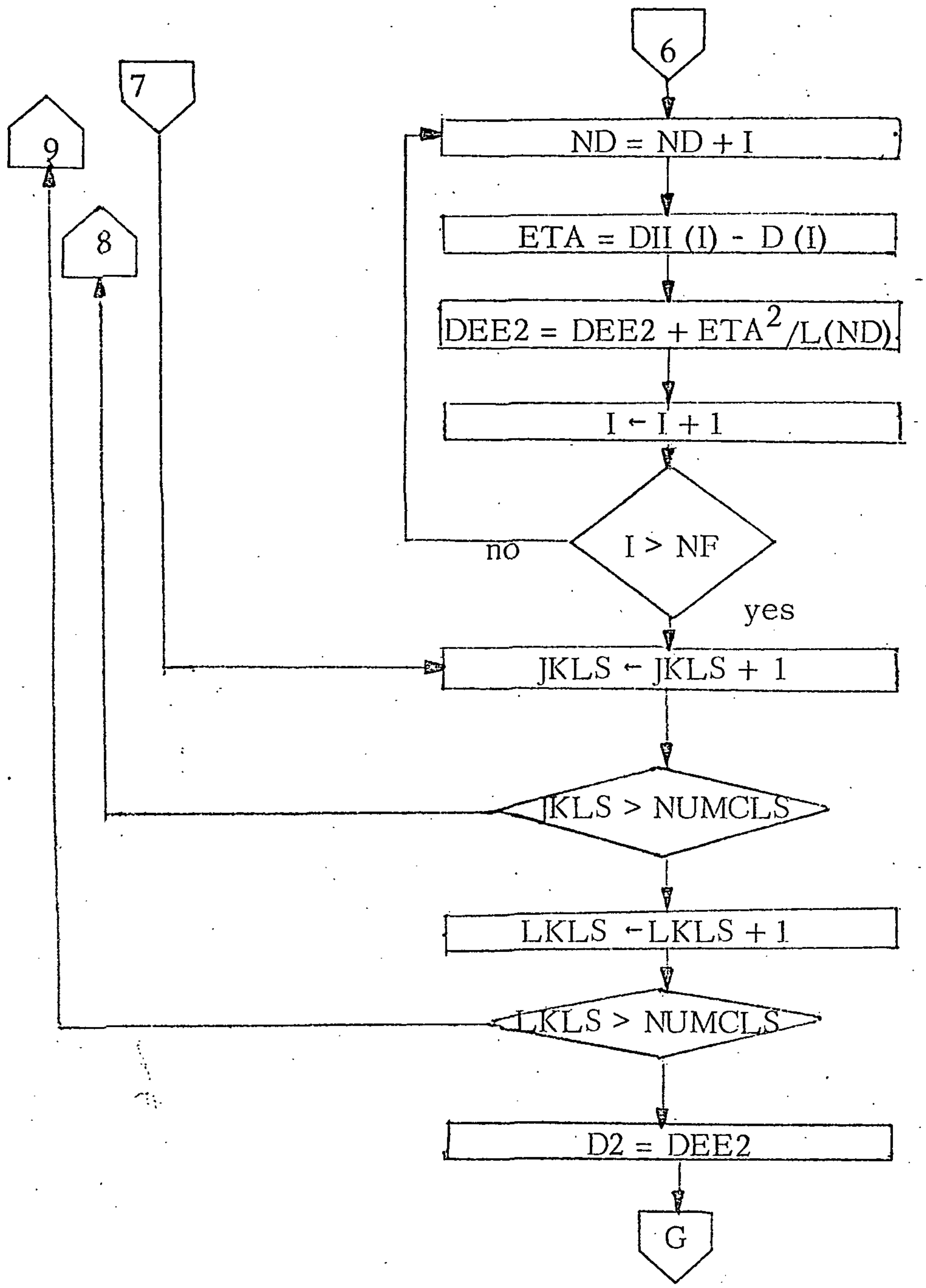
TJ(L, SUML, T NF)


H
T(1)=1
NF=1
yes
no
RETURN
I
NBEGIN = 1
NRW=2
NBEGIN←NBEGIN+NRW-1
NEND←NBEGIN+NRW-1
NEM 1 = NEND -1
NSICR = 0
I = NBEGIN
S = 0
ICR = I-NBEGIN
NSICR = NSICR + ICR
NC = NSICR +1
NL = I
NC = NC + ICR
ICR = ICR +1
8
11
9
10

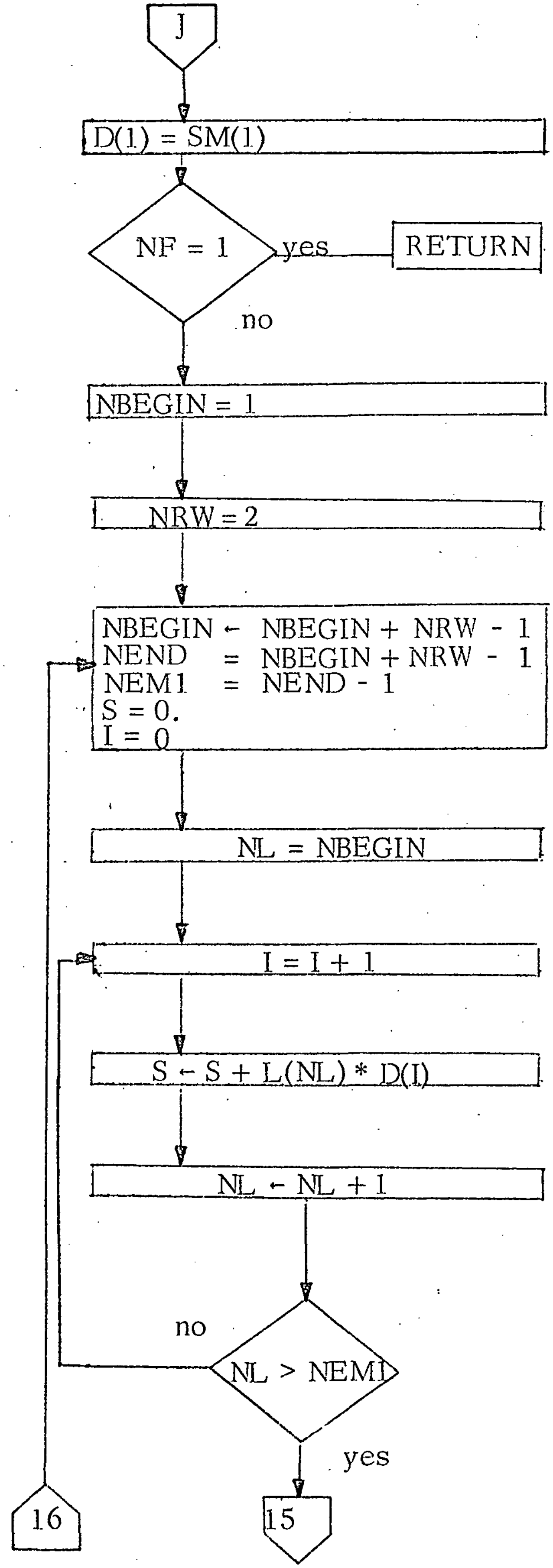

10
11
9
8
S←S+L(NL)* T(NC)
NL←NL+1
no
NL>NEM1
yes
T(I)=(SUML(I)-S)
I ← I + 1
no
I > NEM 1
yes
T(NEND) = 1
NRW ← NRW +1
no
NRW > NF
yes
I

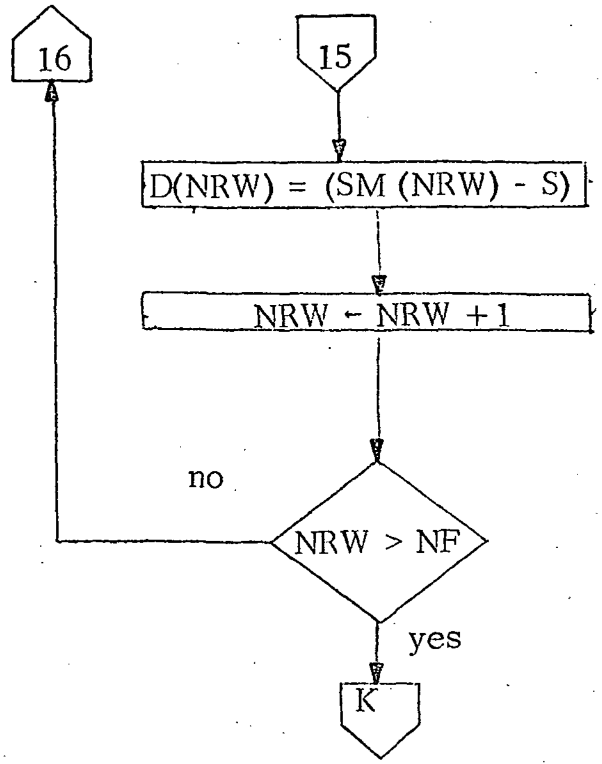
D2 (L, SM, NF, NUMCLS, LENGTH)




6
7
9
8
ND = ND + I
ETA = DII (I) - D (I)
DEE2 = DEE2 + ETA^2/L(ND)
I ← I + 1
I > NF
no
yes
JKLS ← JKLS + 1
JKLS > NUMCLS
LKLS ← LKLS + 1
LKLS > NUMCLS
D2 = DEE2
G

DJL (L, SM, D, NF)


J
D(1) = SM(1)
NF = 1
yes
RETURN
no
NBEGIN = 1
NRW = 2
NBEGIN ← NBEGIN + NRW - 1
NEND = NBEGIN + NRW - 1
NEM1 = NEND - 1
S = 0.
I = 0
NL = NBEGIN
I = I + 1
S ← S + L(NL) * D(I)
NL ← NL + 1
no
NL > NEM1
yes
16
15


16
15
D(NRW) = (SM (NRW) - S)
NRW ← NRW + 1
no
NRW > NF
yes
K