CAC Technical Memorandum No. 17

An ILLIAC IV Algorithm

for

Cluster Analysis of ERTS-1 Data

by

John Thomas

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

May 1974

Reported research was conducted in collaboration with staff of the Laboratory for Applications of Remote Sensing, Purdue University, Lafayette, Indiana 47907.

This work was supported in part by the National Aeronautics and Space Administration through Grant NGR 14-005-202 and in part by the Advanced Research Projects Agency of the Department of Defense through Contract DAHC04-72-C-0001. Collaborative support was also provided by the EROS Program of the Department of Interior.

1.0 Introduction

This is a verbal description of an ILLIAC IV 32-bit mode program, written in ILLIAC assembly language (ASK), that performs cluster analysis of ERTS (Earth Resources Technology Satellite) data into spectrally distinct categories. By multivariate cluster analysis, a sample of image resolution elements from an ERTS multispectral scene are analyzed with respect to spectral properties, and assigned to data clusters such that image elements within each cluster have similar spectral properties while elements of different clusters have dissimilar characteristics. Since the multivariate methodology, in general, presupposes the existence of distinct spectral profiles for all terrain categories of interest, there should exist some simple correspondence between resulting clusters and nominal terrain types.

2.0 The Algorithm

This clustering algorithm was developed at the Laboratory for Application of Remote Sensing (LARS) at Purdue University[5] and implemented on ILLIAC IV at the Center for Advanced Computation at the University of Illinois. This description is taken from the paper "Boundaries in Multispectral Imagery by Clustering", by Arthur G. Wacker and David A. Landgrebe, as it was presented at the 1970 IEEE Symposium on Adaptive Processes (9th) December, 1970.[6]

A clustering cell is a rectangular area or window. The vectors associated with each geographical point (in the case of ERTS1 MSS data, these are four dimensional--one for each spectral band) in the cell are clustered in observation space into a desired number of modes M_m. Thus, a "natural" grouping for these vectors is found in observation space.

The manner in which a set of vectors is clustered is outlined in detail below. The procedure is essentially the ISODATA of Ball and Hall[1] (1965, pp. 1-61) with modifications similar to those suggested by Swain and Fu (1968, pp. 14-19).[2]

Step 1 - Initialization

Let $X_1, X_2, \ldots, X_N$ be N 4-dimensional vectors from the clustering cell. If the number of modes desired is M_m, the M_m initial mode centers are generated as follows.

The sample mean of the N vectors is computed according to:

$$M_j = \frac{1}{N} \sum_{i=1}^{N} X_i \qquad j = 1,2,3,4$$

and the sample variance for each dimension

$$\sigma_j^2 = \frac{1}{N} \sum_{i=1} (X_{ij} - M_j)^2 \quad j = 1,2,3,4$$

Let $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4,)$ and $M = (M_1, M_2, M_3, M_4)$. Consider the real line intervals $\sigma_i = [M_i - \sigma_i, M_i + \sigma_i]$ $i = 1,2,3,4$. The cartesian product $X\sigma_1 . X\sigma_2 . X\sigma_3 . X\sigma_4$ defines a rectangular parallelepiped in the observation space which should contain most of the vectors from the clustering cell. The M_m initial mode centers are chosen to be uniformly spaced along a diagonal of this rectangular parallelpiped. Accordingly, the mode center for the k^{th} mode is:

$$M_k = M + \sigma[2 \frac{k-1}{M_m - 1} - 1] \quad k = 1,2,\ldots,M_m$$

Initially none of the vectors is assigned to any mode.

Step 2 - Mode Assignment

Determine the Euclidean distance from each vector to each mode center. Assign each vector to the mode with the nearest mode center.

Step 3 - Mode Migration

If Step 2 did not change the assignment of any of the N vectors, go to Step 4. Otherwise, replace the old mode centers by the means of the vector clusters resulting from Step 2, then return to Mode Assignment.

Step 4 - Variance - Covariance Calculation

Upon completion of the above iterative process, clustering of all vectors into M_m categories is complete. To complete the statistical description of a category, its variance and covariance are calculated. Variance is calculated via

$$\sigma_j^2 = \frac{1}{P} \sum_{i=1}^{P} (X_{ij} - M_j)^2 \quad j = 1,2,3,4$$

and covariances via

$$C^2_{jk} = \frac{1}{P} \sum_{i=1}^{P} (X_{ij} - M_j)(X_{ik} - M_k) \quad j = 1,2,3,4$$
$$k = j,4$$

where the summation is taken over the P vectors assigned to this category by clustering.

3.0 Data Structures

Due to the parallel architecture of ILLIAC IV, data structures are of primary importance for efficient implementation of any algorithm on this machine. ERTS data file formats were designed with this architecture in mind and PDP-10 (Tenex) data-management software written such that a user may interactively create disk files of ERTS data, from a magnetic tape file, in a format efficient for ILLIAC IV cluster analysis. The specific format of these data files is discussed in "ERTS-ILLIAC Data File Formats."[4] In the present paper we address ourselves to the structure of data within ILLIAC memory and how this bears on the ILLIAC clustering program.

3.1 Input

The ILLIAC clustering program is run on a core contained data set which is input from ILLIAC disk at the outset of execution. This data consists of a user specified geographical area of raw ERTS MSS data which may contain up to 32K points. After initial input, and before execution of the clustering program, the data resides in ILLIAC as follows:

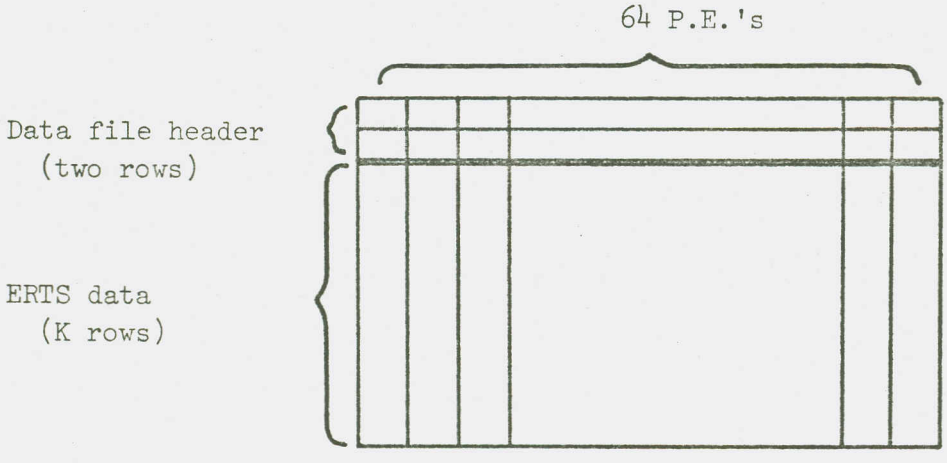

Fig. 1

Each ERTS data point consists of 4 8-bit intensity values for each of 4 spectral bands. Therefore, in the above diagram each P.E. memory location contains two ERTS points as follows:

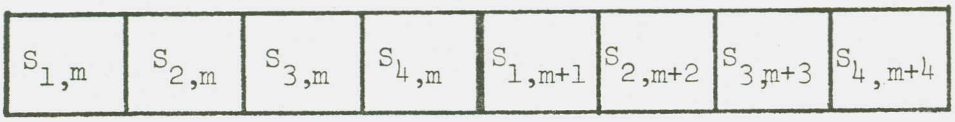

Fig. 2

where:

$S_{i,m}$ = 8-bit intensity value for the m^{th} point in the i^{th} spectral band.

For example, if an ERTS data file contains 32K points, each 64-bit P.E. contains 32,768/64 = 512 points in its local memory after initial input.

The data file header requires two memory locations in each P.E. (i.e. consists of 2 ILLIAC "long rows") and contains program control information, such as user-requested number of clusters M_m, and the number of long-rows of data following this header.[4] Initial execution steps consist of fetching this information to the ILLIAC Control Unit (CU) and initializing certain program control variables. Actually, two ILLIAC Disk I/O requests are made to read all data into PE-memory. The first reads the header which contains the number of long rows of data in the file and a second I/O request is made to read the specified quantity of data.

3.2 Output

The clustering program produces two distinct results. The first is an ERTS data file of classified points; the second, a file containing statistical descriptions (spectral signatures) of the M_m clusters.

3.2.1 Classified ERTS Points

At the completion of clustering, each ERTS point in the initial input buffer, has been assigned to one of M_m categories.
The classified ERTS data file is reduced to one half its original input size where each point required 32 bits. This compression preserves the original order of the input raw-data file. A typical word in the output buffer memory area contains category numbers of 4 ERTS points as follows:

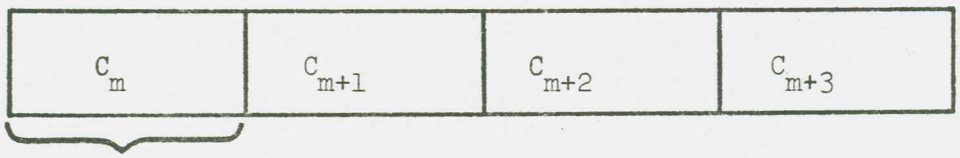

Fig. 3

where:

C_m = 8-bit category number for the m^{th} point

(the high-order 8 bits are left blank)

The two long-row data file header for this file of classified ERTS points is altered to indicate the file as containing classified data (i.e. the file-type field is set to 2) and the entire file is written to ILLIAC disk in one output request.

3.2.2 Spectral Signatures

Each cluster found by the clustering program is uniquely characterized by fourteen values, a four component mean value vector, and the ten values of the variance-covariance matrix--diagonal and lower half (this is a symmetric matrix). These values are output in ILLIAC 32-bit floating point format and are packed such that an 8-word block across P.E. memory contains all 14 values as follows:

word 1		word 2		word 3		word 4		word 5		word 6		word 7		word 8
M_1	M_2	M_3	M_4	C_{11}	C_{22}	C_{33}	C_{44}	C_{21}	C_{31}	C_{41}	C_{32}	C_{42}	C_{43}	

fig. 4

Therefore, one long-row of 64 words, can contain statistics for up to eight categories.

The type-field of the header for this statistics file is set to 4 and the entire file is output to ILLIAC disk.

4.0 Program Execution

4.1 Initialization of CU Variables

After initial input of the file header, the first eight words of this header are fetched from P.E. memory to the CU. The data file field is checked for type 1. If it is an improper ERTS data file, an error exit is performed. Next, two important loop control variables are constructed. The first controls the category count loop and the second the data count loop. They are constructed from M_m, the number of desired categories, and N the number of long-rows of data, both of which are contained in the first eight words of this header. Then, an I/O request is issued to read the remainder of the data file from disk.

4.2 Conversion of Input to Floating Point

Since the input data has 32 bits per point and higher precision calculation is unnecessary, this ILLIAC IV ASK program is written to operate on 32-bit floating point quantities. Operating ILLIAC IV in 32-bit mode essentially expands it to a 128 P.E. machine which in certain instruction-data environments almost doubles the speed of the 64 P.E. ILLIAC.[3] For all data in the input file, each of the four 8-bit bytes of intensity is converted to a 32-bit ILLIAC internal floating point number and stored. This operation is performed in parallel where all 128 P.E.'s convert the data local to their memories. The program is executed using these floating point quantities and the original input buffer area is now available as temporary storage of classified points and eventually as the output buffer.

4.3 Sample Mean and Variance

The calculation of the sample mean requires each P.E. to take a four component partial sum of all data points in its local memory. These partial sums are then routed and summed across P.E.'s until all P.E.'s contain the total sum. All P.E.'s divide these total component sums by the total number of vectors and store the resultant four component sample mean in their local memories. Sample variance is calculated similarly and each P.E. stores this four component vector in its local memory. This strategy of storing common values across all P.E.'s requires more storage, yet eliminates the necessity of routing values numerous times between P.E.'s.

4.4 Initial Mode Center Assignment

As per the theoretical description, each P.E. performs the identical calculation of the M_m initial four component mode centers and stores them locally.

4.5 Euclidean Distance and Mode Assignment

Since each P.E. holds the M_m four component mode centers in its local memory, it is able to calculate locally the Euclidean distance of each point to each of these mode centers and take the minimum value independent of all other P.E.'s. Here ILLIAC IV is operating completely in parallel with each P.E. performing identical calculations on different data. In this step each point is assigned to its nearest category. This category number is entered in the point's original input memory location.

4.6 Mode Migration

A CU variable flags if any point changed category in the previous Euclidean distance assignment. If this flag is set, new mode centers are calculated. The problem is similar to sample mean calculation except that here, we want partial sums of those points assigned to specific categories. For each category, the instruction stream to the P.E.'s is similar to that for the sample mean calculation with the addition of a screening procedure to exclude points not assigned to the particular category. For points not in the category for which the partial sum is being taken, P.E.'s are independently disabled during the addition operation. In this way, each P.E. in parallel computes a partial sum of all points in its local memory that have been assigned to a particular category. For each category these partial sums are routed and summed until all P.E.'s contain the total sum which is divided by the total number of points assigned that category. This yields the new mode center for that category for the next iteration through Euclidean distance assignment procedure.

4.7 Pack Classified Point and Output

At the completion of the iterative mode assignment and migration section, each location in the original input buffer area in P.E. memory contains an 8-bit category number in place of the 32-bit raw data. These 8-bit quantities are packed four per word such that the original order of points, from left to right across P.E.'s and down P.E. memory, is preserved.

The file type in the header is set to 2 and the classified data file plus header is written to ILLIAC Disk.

4.8 Variance and Covariance Calculation

Using the equation in the theoretical description and the per-category summing procedure described in sec. 4.6, all P.E.'s calculate variance and covariance for all M_m categories.

4.9 Pack Statistics and Output

The means, variance, and covariance for each of the M_m categories are packed into consecutive eight-word blocks across P.E. memory as shown in Fig. 4. The type field in the file header is set to 3 and this statistics file plus header is written to ILLIAC Disk.

References

1. Ball, G. H. and D. J. Hall (1965), ISODATA, a novel method of data analysis and pattern classification, Stanford Research Institute, Menlo Park, California. 1-16.

2. Swain, P. H. and K. S. Fu (1968), on the application of nonparametric techniques to crop classification problems, National Electronics Conference Proceedings 24, 14-19.

3. Thomas, J. (1974), Some Notes on ILLIAC IV 32 bit Mode Operation, CAC Technical Memorandum #20, May 1974.

4. Thomas, J. (1974), ERTS-ILLIAC Data File Formats, CAC Technical Memorandum #19, May 1974.

5. Swain, P. H., "Pattern Recognition: A Basis for Remote Sensing Data Analysis," LARS Information Note 11572 (Purdue University, West Lafayette, Indiana: The Laboratory for Applications of Remote Sensing, 1972).

6. Wacker, A. G. and Landgrebe, D. A., "Boundaries in Multispectral Imagery by Clustering," IEEE Symposium on Adaptive Processes (9th), 1970.

APPENDIX

ASK Listing of the ILLIAC IV

Cluster Analysis Algorithm

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   THIS IS THE LARS CLUSTERING ALGORITHM FOR ILLIAC IV WRITTEN
   IN ILLIAC ASSEMBLY LANGUAGE, ASK.

   THE CLUSTERING ALGORITHM IS FROM: "BOUNDARIES IN MULTISPECTRAL
   IMAGERY" BY ARTHUR G. WACKER AND DAVID A. LANDGREBE.

   DOCUMENTATION:  "AN ILLIAC IV ALGORITHM FOR CLUSTER
                    ANALYSIS OF ERTS-1 DATA",
                    CAC TECHNICAL MEMORANDUM #17, MAY 1974.

   THIS PROGRAM WAS WRITTEN AND DEBUGGED BY JOHN THOMAS AT THE
   CENTER FOR ADVANCED COMPUTATION, UNIVERSITY OF ILLINOIS.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


BEGIN
          BLK  O;
         %ACATEG AND INMEM DETERMINE MEMORY ALLOCATION.
         %TO INCREASE THE NUMBER OF
         %CATEGORIES THE PROGRAM CAN FIND,  THE AMOUNT OF DATA
         %MUST BE REDUCED AND VICE VERSA.
         % THE EQUATION RELATING ACATEG AND INMEM IS:
         %      (16*ACATEG+5*INMEM) LEQ 1980
         %      WHERE INMEM IS A MULTIPLE OF 16 (AN I/O PAGE).

         DEFINE ACATEG =  32##;   %MAX NUMBER OF CATS ALLOWED.
         DEFINE INMEM  =  288##;        %INPUT AREA IN MEM.
% THE FOLLOWING IS A DVRN DEFINE THAT
% CIRCUMVENTS THE FACT THAT 32-BIT DVRN
% WORKS CORRECTLY IN ONLY THE INNER OPERAND.
DVX:   SYN DVRN;
DVROW:  BLK 1;
DEFINE DVRN &ARG;=
     STA DVROW;
     SETF  E.OR.Z;
     SETF1 E1.OR.Z;
     SETE  E.AND.Z;
     DVX   &ARG;
     SETE1 E.AND.Z;
     SETE  E.OR.-E;
     LDA   DVROW;
     SETE1 E.OR.-E;
     SWAPA;
     STA   DVROW;
     LDB   $A;
```

```
     LDA   &ARG;
     SWAPA;
     LDR   $A;
     LDA   $B;
     SETE  E.AND.Z;
     SETE1 F.OR.Z;
     LDB   =0;
     DVX   $R;
     SETE1 E.AND.Z;
     SETE  E.OR.-E;
     LDA   DVROW;
     SETE1 E.OR.-E;
     SWAPA;
     SETE  F.OR.Z;
     SETE1 F1.OR.Z;
##;
WIN:        AREA "WINDOW",0,1,0;   %ERTS DATA DISK AREA.
COF:        AREA "COEFFS",0,1,0;   %STATISTICS DISK AREA.
COMPWORD: DATA 0;
HEADER:   BLK  2;     %FIRST TWO ROWS OF INPUT AREA
                      %CONTAIN THE STANDARD ERTS-ILLIAC
                      %SYSTEM HEADER.
X:          BLK   14;          %START OF DATA AREA.
PAGE2:      BLK  INMEM-16;     %SECOND I/O PAGE BOUNDARY.
PACKER:
          DATA
          (0,2,4*ACATEG,4*ACATEG+2,8*ACATEG,8*ACATEG+2,8*ACATEG+4,0)8;
PACKEROFFSET:
          DATA
           (0)8,(4)4,(6)4,(8)4,(12)4,(12)4,(18)4,(16)4,(24)4,
           (20)4,(30)4,(24)4,(36)4,(28)4,(42)4;
PACKERLOOP:
           DATA
           (32)4,(48)4,(32)4,(48)4,(32)4,(48)4,(32)4,(48)4,
           (32)4,(48)4,(32)4,(48)4,(32)4,(48)4,(32)4,(48)4;
MODCOV:     EQU  X;       %OUTPUT AREA FOR PACKED COVARIANCE AND MEANS
MDCNTR:     BLK 4*ACATEG;    %MODE CENTERS FOR CATEGORIES
VARIANCE:   BLK 4*ACATEG;
COVAR:      BLK    6*ACATEG;
COVART:     BLK    6;
XCONTEM:    BLK  4;
INCORE:     BLK   1;
DELTA:      BLK   1;
RTEM:       BLK  4;
AVERAGE:    BLK 4;
AVSQ:       BLK   4;
CLASSCNTR:  BLK  ACATEG+1;         %HOLDS TOTAL NUMBER OF PATTERNS IN CATE
TOTCLASS:   BLK   ACATEG;
CLASS:      EQU X;           %CATEGORY FOR A PATTERN
DISTEMP:    BLK  1;
```

```
CLASSTEMP: BLK  1;
CATEG:     BLK   1;    %NO. OF CATS IN 32-BIT FIXED PT.
MINDIST:   BLK 1;
PRESENTCAT: BLK 1;
NOCLASSMINUS1:  EQU  MINDIST;
INCREMENT: BLK 1;
CATEGFLT:  BLK 1;    %32-BIT FLT        PT OF NO. OF CATEGORIES
ONE:      BLK  1;
NOFVECTORSINMEM:  BLK  1;
SAMPLEVARDIVISOR:  BLK  1;
TTCLASS:   BLK 1;
FLTONE:    BLK  1;
DISTEM1:  EQU  AVERAGE;
DISTEM2:  EQU  XCONTEM;
%STANDARD DATA FILE HEADER CU VARIABLES.
.HEADER:          EQU    $D0;
.FILETYPE:        EQU    $D0;    %DATA FILE TYPE.
.CATEG:           EQU    $D1;    %CATEGORIES IN.
.DELTA:           EQU    $D3;    %MODE DISTINCTION THRESHOLD.
.VECTORS:         EQU    $D4;    %NO. OF LONG ROWS OF INPUT VECTORS.
%OTHER CU VARIABLES.
.CATCNTR: EQU  $D8;      %LOOP CONTROL ON CATEGORIES
.VECNTR:  EQU  $D9;     %LOOP CONTROL ON        VECTORS.
.ONLYONECATLEFT:   EQU    $D10;
.SOMENOTDISTINCT: EQU  $D11;
.FIRSTMODE:  EQU          $D12;
.SECONDMODE: EQU  $D13;
.ADDRESSMASK: EQU  $D14;
.PE0:    EQU  $D15;
.PE2:    EQU  $D16;
.LASTROWOUTER:   EQU    $D17;
.LASTROWINNER:   EQU    $D18;
.TEMP:           EQU    $D23;
.MASK:   EQU      $D24;
% SQUARE        ROOT VARIABLES.
SQRTEXP: BLK  1;
SAVMX:   BLK  1;
.SAVE3:  EQU  $D19;
.SAVE2:  FQU  $D20;
.SAVE1:  EQU  $D21;
.EXP:    EQU  $D31;
.ITERCNT: EQU  $D35;
.SAVECT:  EQU $D31;
.SAVEVC:  EQU  $D30;
.SAVE0:   EQU  $D22;
.SV3:      EQU  $D25;
.MD1:      EQU  $D26;
.MD2:      EQU  $D27;
.MD3:      FQU  $D28;
.MD4:       EQU  $D29;
```

```
.MD5:       EQU  $D32;
.MD6:       EQU  $D33;
.M1:        EQU  $D34;
.ALL:       EQU  $D36;
.MODE:      EQU  $D37;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %   SQUARE ROOT SUBROUTINE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

SQRT::    FILL;
          STL(3)        .SAVE3;       %SAVE ACARS.
          STL(2)        .SAVE2;
          STL(1)        .SAVE1;
          STL(0)        .SAVE0;
          LDS           $A;
          IB            7;            %EXPONENT EVEN OR ODD?
          SETE          I.AND.E;
          SETE1         G.AND.E1;
          LIT(3)        =40501000000000000000:8;
          ADEX          $C3;          %ADD ONE TO EXPONENT.
          SHAMR         1;            %DIVIDE MANT. BY TWO.
          LDS           $A;
          SETE          E.OR.-E;
          SETE1         E.OR.E;
          IB            1;
          SHAR          1;
          SETE          I.AND.E;
          SETE1         G.AND.E1;
          SAB           1;
          RAB           2;
          SETE          -E.OR.-E;
          SETE1         -E1.OR.-E1;
          SAB           2;
          SETE          E.OR.-E;
          SETE1         E.OR.E;
          STA           SQRTEXP;
          %FIND SQRT OF MANTISSA.
          LDA           $S;
          LIT(3)     =40100:8,0,0;
          LEX           $C3;
          NORM;
          SAN;
          STA         SAVMX;
          %FIND INITIAL VALUE FOR ITERATION.
          LIT(0)        =PAIR(2.22152,2.22152);
          LIT(1)        =PAIR(2.03146,2.03146);
          LIT(2)        =PAIR(.8125,.8125);
          SETE          -I.AND.E;
          SETE1         -G.AND.E1;
          MLRN          $C2;                    %BXT.
```

```
        ADRN        $C1;
        MLRN        SAVMX;
        ADRN        $C0;
        LIT(0)      =PAIR(3.16214,3.16214);
        LIT(1)      =PAIR(5.86118,5.86118);
        LIT(2)      =PAIR(4.75,4.75);
        SETE        -E.OR.-E;
        SETE1       -E1.OR.-E1;
        MLRN        $C2;
        ADRN        $C1;
        MLRN        SAVMX;
        ADRN        $C0;
        SETE        E.OR.-E;
        SETE1       E.OR.E;
        LIT(3)      =PAIR(3.0,3.0);
        %ITERATE.
        LDS         $A;
        MLRN        $A;
        MLRN        SAVMX;
        ADRN        $C3;
        MLRN        $S;
        SHAMR       1;
        NORM;
        %ONE ITERATION.
        LDS         $A;
        MLRN        $A;
        MLRN        SAVMX;
        ADRN        $C3;
        MLRN        $S;
        SHAMR       1;
        NORM;
        %TWO ITERATIONS.
        LDS         $A;
        MLRN        $A;
        MLRN        SAVMX;
        ADRN        $C3;
        MLRN        $S;
        SHAMR       1;
        NORM;
        %THREE.
        LDS         $A;
        MLRN        $A;
        MLRN        SAVMX;
        ADRN        $C3;
        MLRN        $S;
        SHAMR       1;
        NORM;
        LDS         $A;
        MLRN        $A;
        MLRN        SAVMX;
```

```
        ADRN        $C3;
        MLRN        $S;
        SHAMR       1;
        NORM;
        %END OF ITERATIONS.
        SAN;
        MLRN        SAVMX;
        ADEX         SQRTEXP;
        NORM;
        LDL(3)      .SAVE3;        %RESTORE ACARS.
        LDL(2)      .SAVE2;
        LDL(1)      .SAVE1;
        LDL(0)      .SAVE0;
        EXCHL(3)    $ICR;          %RETURN.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                %   END OF SQUARE ROOT SUBROUTINE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
START::
        CHWS        32;
        SETE        E.OR.-E;
        SETE1       E.OR.E;
        %INITIALIZE SOME VARIABLES.
        LIT(0)       =0,0,07F:16;
        STL(0)       .MASK;
        LIT(1)      =77777777:8;
        STL(1)      .ADDRESSMASK;
        LIT(1)     =PAIR(1.0,1.0);
        LDA         $C1;
        STA         FLTONE;
        LIT(1)     =PAIR(1,1);
        LDA         $C1;
        STA         ONE;
        LIT(0)      =8000:16,0,0;
        STL(0)      .PE0;
        LIT(0)      =2000:16,0,0;
        %MODE VARIABLES FOR PACKING OUTUT.
        LIT(0)    =0FFFFFFF00000000:16;
        STL(0)    .MODE;
        CLC(0);
        COMPC(0);
        STL(0)    .ALL;
        LIT(0)    =5555555555555555:16;
        STL(0)    .M1;
        LIT(0)    =8000000000000000:16;
        STL(0)    .MD1;
        LIT(0)    =4444444444444444:16;
        STL(0)    .MD2;
        LIT(0)    =3030303030303030:16;
        STL(0)    .MD3;
        LIT(0)    =0F000F000F000F00:16;
        STL(0)    .MD4;
        LIT(0)    =00FF000000FF0000:16;
        STL(0)    .MD5;
        LIT(0)    =0000FFFF00000000:16;
        STL(0)    .MD6;
        STL(0)      .PE2;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE FOLLOWING READS THE DATA FILE CONTAINING THE ERTS
%  WINDOW ON WHICH THE CLUSTERING WILL BE PERFORMED.
%  THIS IS A STANDARD ERTS-ILLIAC DATA FILE OF TYPE 1,
%  IT CONTAINS:
%             1.NUMBER OF CATEGORIES TO BE FOUND IN THE WINDOW.
%             2.DELTA, THE MODE DISTINCTION THRESHOLD.
%             3.THE NUMBER OF ILLIAC LONG ROWS OF DATA IN THE
%               WINDOW.
%             4.THE ERTS DATA FOR THE WINDOW.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         %FETCH DATA FROM DISK.
         OPEN WIN;
         GET WIN,HEADER,COMPWORD;
         PAUSE COMPWORD;
         JUMP ERROREXIT;
         CLOSE WIN;
         % PULL CONTROL INFO FROM FIRST 6 WORDS OF HEADER.
         LIT(2)       =HEADER;
         BIN(2)       .HEADER;      %FETCH TO CU
         %CHECK FILE TYPE, SHOULD BE TYPE 1
         LDL(2)       .FILETYPE;    %D0.
         LIT(3)       =0,1,0;
         TXET(2)      $C3,FILETYPEOK;
         JUMP          ERROREXIT;
FILETYPEOK:
        % ANY MORE PAGES OUT THERE.
        LDL(2)  .VECTORS;
        ALIT(2)  =1;
        CSHR(2)  4;
        % CLEAR THE MORE DATA FLAG.
        CLC(1);
        ZERT(2)  ,NOMORE;
        CSHL(2)  24;
        % ENTRY OF THE ADDRESS SETS THE MORE DATA FLAG.
        LIT(1)  =WIN;
        COR(2)  $C1;
        SETCOUNT  ($C2);
        GET WIN,PAGE2,COMPWORD;
NOMORE:
         DUPO(2)       .CATEG;       %D1.
         % BROADCAST NUMBER OF CATEGORIES TO BE FOUND.
         LDA           $C2;
         LDB           $A;
         SWAPX;
         LDA           $B;
         STA           CATEG;
         LIT(3)        =54130:8,0,0;
```

```
        STL(3)    .EXP;
        LEX           $C3;
        NORM;
        STA           CATEGFLT;
        %CREATE CATEGORY LOOP CONTROL COUNTER.
        LDL(2)        .CATEG;
        ALIT(2)       =-1;
        CSHL(2)       26;
        CSB(2)        13;
        STL(2)        .CATCNTR;
        %BROADCAST MODE DISTINCTION THRESHOLD DELTA
        LDL(2)        .DELTA;       %$D3
        LDA           $C2;
        STA           DELTA;
        %CREATE VECTOR LOOP CONTROL COUNTER.
        LDL(2)        .VECTORS;     %$D4.
        ALIT(2)       =-1;
        CSHL(2)       26;
        CSB(2)        13;
        STL(2)        .VECNTR;
        %BROADCAST NUMBER OF LONG ROWS OF VECTORS.
        DUPO(0)       .VECTORS;
        LDA           $C0;
        STA           INCORE;
        ALIT(0)      =-1;
        %WAIT FOR THE DATA.
        % DON'T WAIT IF THE MORE DATA FLAG IS ZERO.
        ZERT(1)  ,AROUND;
         PAUSE COMPWORD;
         JUMP  ERROREXIT;
         CLOSE  WIN;
AROUND:
        %COMPUTE NUMBER OF VECTORS TO BE PROCESSED.
        %LAST LONG ROW OF INPUT FILLED OUT WITH DUMMY
        %ENTRIES WITH ALL ONES.
        LDA           X(0);     %LAST ROW OF VECTORS.
        ILO;                    %PE'S WITH DUMMY ENTRIES SET I.
        SETC(2)       I;
        STL(2)        .LASTROWOUTER;
        LEADO(2);
        CTSBT(2)      55,FOUND;
        SLIT(2)       =64;          % 64 GOOD VECTORS IF NO LEAD 1.
FOUND:
        SETC(1)       G;
        STL(1)        .LASTROWINNER;
        LEADO(1);
        CTSBT(1)      55,FOUND1;
        SLIT(1)       =64;
```

```
FOUND1:
        CAND(2)         .MASK;
        CAND(1)         .MASK;
        CADD(2)         $C1;
        % $C2 CONTAINS NUMBER OF VECTORS IN LAST ROW
        % THAT ARE NOT DUMMY ENTRIES.
        STL(2)          .TEMP;
        DUPO(2)         .TEMP;
        LDA             INCORE;
        LDL(3)    .EXP;
        LEX         $C3;
        NORM;
        SBRN            FLTONE;
        LIT(1)        =PAIR(128.0,128.0);
        MLRN            $C1;        %$A CONTAINS TOTAL NO. OF VECTORS
                                    %NOT COUNTING LAST LONG ROW.
        LDS             $A;
        LDA             $C2;
        LEX             $C3;
        NORM;
        ADRN            $S;
        STA             NOFVECTORSINMEM;
        SBRN            FLTONE;
        STA             SAMPLEVARDIVISOR;
        % ZERO OUT DUMMY ENTRIES IN LAST ROW.
        LDL(2)          .LASTROWOUTER;
        LDE             $C2;
        STL(2)          .LASTROWOUTER;
        LDL(2)          .LASTROWINNER;
        LDE1            $C2;
        STL(2)          .LASTROWINNER;
        CLRA;
        STA             X(0);
        SETE          E.OR.-E;
        SETE1         E.OR.E;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  THE FOLLOWING CONVERTS THE PATTERN VECTORS INPUT AS 8-BIT BYTES
%  INTO 32-BIT ILLIAC FLOATING POINT INTERNAL FORMAT.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
         LDL(0)    .VECNTR;
        CLC(3);
GETIT:
          CHWS     64;
          LDA      X(3);     %CONVERT PATTERN VECTORS TO 32-BIT FLT PT
          RTAR     =8;
          CHWS     32;
          SETE1    E.AND.-E;
          RTAL     8;
          SETE1    E.OR.-E;
          LIT(2)    =1,3,0;
          LDX      =0;
```

```
D:
          LDS    $A;
          SHAR   24;
          LDL(1)     .EXP;
          LEX    $C1;
          NORM;
          STA    *XCONV(0);
          STA    XCONTEM(2);
          XI     =1;
          LDA    $S;
          SHAL   =8;
          TXLTM(2)    ,D;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   COMPUTE SAMPLE MEAN OF ALL VECTORS IN THE WINDOW.
%   THE METHOD FOR OBTAINING THE TOTAL SUM OF ALL VECTORS
%   ACROSS ALL P.E.'S IS AS FOLLOWS:
%      1.EACH PE ADDS THE FIRST COMPONENTS OF ALL VECTORS
 %        STORED IN ITS MEMORY AND STORES THIS IN RTEM.
%      2.EACH PE ADDS THE SECOND COMPONENTS AND STORES IN RTEM+1.
%      3.THE INNER AND OUTER PORTION OF RTEM IS ADDED IN EACH PE.
%      4.THE INNER AND OUTER PORTION OF RTEM+1 IS ADDED IN PE'S.
%      5.THE TWO ABOVE SUMS ARE PLACED IN THE INNER AND OUTER
%        PORTION OF A WORD WHICH IS ROUTED AND ADDED TO UNTIL
%        ALL PE'S CONTAIN THE TOTAL SUM OF THE FIRST COMPONENTS
%        OF ALL VECTORS IN THE INNER PORTION OF A WORD AND THE
%        TOTAL OF ALL SECOND COMPONENTS IN THE OUTER PORTION OF
%        THE SAME WORD.
%
%   THIS METHOD FOR SUMMING 32-BIT QUANTITIES ACROSS PE'S IS
%   USED NUMEROUS TIMES WITHIN THIS PROGRAM.
%
%   DIVIDE THE SUMS OBTAINED ABOVE BY THE TOTAL NUMBER OF VECTORS
%   IN THIS WINDOW TO OBTAIN A 4 COMPONENT VECTOR FOR THE SAMPLE
%   MEAN OF ALL VECTORS IN THE WINDOW.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          LIT(1)      =1,3,0;

ADDEM:

          LDA    XCONTEM(1);     %SUM CONVERTED NUMBERS
          ADRN   RTEM(1);
          STA    RTEM(1);
          TXLTM(1)   ,ADDEM;
          ALIT(3)   =1;
          TXLTM(0)   ,GETIT;
          LIT(0)     =2,2,0;
          LDX        =0;
PACKANDROUTE:
```

```
        LDA   RTEM+1(0);      %PACK TWO COMPONENT SUMS PER WORD
        LDB   $A;
        SWAPX;
        ADRN  $B;
        LDS   $A;
        LDA   RTEM(0);
        LDB   $A;
        SWAPX;
        ADRN   $B;
        LDB   $S;
        SWAPX;
                %SUM OF ITH AND I+1ST COMPONENT
                %LIE IN INNER AND OUTER PART OF $A.
        RTL   $A,-1;
        ADRN  $R;
        RTL   $A,-2;
        ADRN  $R;
        RTL   $A,-4;
        ADRN  $R;
        RTL   $A,-8;
        ADRN  $R;
        RTL   $A,-16;
        ADRN  $R;
        RTL   $A,-32;
        ADRN  $R;
        LDB   =0;
        DVRN  NOFVECTORSINMEM;
        %UNPACK AND STORE.
        LDB         $A;
        SWAPX;
        LDS         $B;
        STA         *AVERAGE;
        MLRN          $A;
        STA           *AVSQ;
        LDA         $S;
        STA         *AVERAGE+1;
        MLRN          $A;
        STA           *AVSQ+1;
        XI          =2;
        TXLTM(0)     ,PACKANDROUTE;
        CLRA;
        STA         RTEM;
        STA         RTEM+1;
        STA         RTEM+2;
        STA         RTEM+3;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%   COMPUTE SAMPLE VARIANCE OF ALL VECTORS IN THE WINDOW.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        LDL(0)     .VECNTR;
```

```
SM1:
          LDX       =0;
          LIT(1)    =1,3,0;
SUBMEAN:
          LDA       *XCONV(0);
          SBRN      AVERAGE(1);  %SUB SAMPLE MEAN.
          MLRN      $A;          %SQUARE IT.
          ADRN      RTEM(1);
          STA       RTEM(1);
          XI        =1;
          TXLTM(1)  ,SUBMEAN;
          TXLTM(0)  ,SM1;
          % SUB AVSQ FROM RTEM IN ALL PE'S CONTAINING
          % DUMMY POINTS IN LAST LONG ROW.
          LIT(0)    =1,3,0;
          LDL(2)    .LASTROWOUTER;
          LDE       $C2;
          LDL(2)    .LASTROWINNER;
          LDE1      $C2;
FIXUP:
          LDA       RTEM(0);
          SBRN      AVSQ(0);
          STA       RTEM(0);
          TXLTM(0)  ,FIXUP;
          SETE         E.OR.-E;
          SETE1        E.OR.E;
          LDX      =0;

          LIT(0)    =2,2,0;
PACKANDROUTE1:

          LDA   RTEM+1(0);     %PACK TWO COMPONENT SUMS PER WORD
          LDB   $A;
          SWAPX;
          ADRN  $B;
          LDS   $A;
          LDA   RTEM(0);
          LDB   $A;
          SWAPX;
          ADRN   $B;
          LDB   $S;
          SWAPX;
                %SUM OF ITH AND I+1ST COMPONENT
                %LIE IN INNER AND OUTER PART OF $A.
          RTL   $A,-1;
          ADRN  $R;
          RTL   $A,-2;
          ADRN  $R;
          RTL   $A,-4;
          ADRN  $R;
```

```
        RTL   $A,-8;
        ADRN  $R;
        RTL   $A,-16;
        ADRN  $R;
        RTL   $A,-32;
        ADRN  $R;
        LDB   =0;
        DVRN    NOFVECTORSINMEM;
        JMZ;                          %NON-ZERO CALL SQRT.
        SETC(1)   J;
        ONEST(1)  ,ZER;
        CALL    SQRT;
ZER:
        %UNPACK AND STORE.
        LDB       $A;
        SWAPX;
        STA       *VARIANCE;
        LDA       $B;
        STA       *VARIANCE+1;
        XI        =2;
        TXLTM(0)    ,PACKANDROUTE1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  USING THE SAMPLE MEAN OF ALL VECTORS IN THE SAMPLE SPACE
%  AND THEIR SAMPLE VARIANCE, COMPUTE N (WHERE N IS THE NO.
%  OF CATEGORIES TO BE FOUND) INITIAL MODE CENTERS THAT ARE
%  EQUALLY SPACED ALONG THE DIAGONAL OF THE HYPER-CUBE CENTERED
%  AT THE SAMPLE MEAN WITH SIDE LENGTH IN THE MTH DIMENSION
%  EQUAL TO THE MTH COMPONENT OF THE SAMPLE VARIANCE.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        LDA       CATEGFLT;
        SBRN      FLTONE;
      STA     DISTEMP;
        LDA       FLTONE;
      ADRN      $A;
        LDB       =0;
        DVRN      DISTEMP;
        STA       INCREMENT;      %INCREMENT ALONG THE DIAGONAL
                                  %OF THE HYPERCUBE CENTERED AT
                                  %THE SAMPLE MEAN.
        LDL(0)    .CATCNTR;
        CLRA;
ANOTHERCAT:
        STA         RTEM;
        LDX       =0;
        SBRN      FLTONE          ;
        STA       DISTEMP;
        LIT(1)    =1,3,0;         %COMPONENT COUNTER.
```

```
ANOTHERCOMP:
        LDA     VARIANCE(1);
        MLRN     DISTEMP;
        ADRN    AVERAGE(1);
        STA     *MDCNTR(0);
        XI      =1   ;
        TXLTM(1) ,ANOTHERCOMP;
        LDA     RTEM;
        ADRN    INCREMENT;
        TXLTM(0) ,ANOTHERCAT;
        LIT(3)    =1;     % SET $C3 NOT ALL ZEROS TO OVERIDE
                          % THE FLAG HELD THERE.  THIS FLAG HAS
                          % USE LATER WHEN ITERATING THRU THE
                          % MODE MIGRATION TO CHECK CONVERGENCE.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%  COMPUTE THE DISTANCE OF ALL POINTS TO ALL MODE CENTERS
%  AND ENTER ALL POINTS INTO THE CATEGORY THEY ARE CLOSEST TO.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CLC(1);
STL(1)  .ITERCNT;
DISTAN::
        LDL(0)    .VECNTR;
        CLRA;
        LDX =0;
        LDL(1)    .CATCNTR;
INI:
        STA       *CLASSCNTR;
        XI  =1;
        TXLTM(1)  ,INI;
        CLC(1)  ;
        STL(1)   .SAVEVC;
NEWPT:
        LDL(1)    .CATCNTR;
        CLRA;
        STA       PRESENTCAT;
        LDA       NOFVECTORSINMEM;  %ANY BIG NO. WILL DO.
        MLRN      $A;
        STA       DISTEMP;  %STARTING VALUE FOR MIN. DISTANCE.
        LDA          CATEG;
        STA          CLASSTEMP;
NEWCAT:
        LDX       =0;
        CLRA;
        STA       RTEM;
        LIT(2)   =1,3,0;
```

```
EUCLID:
        LDA     *XCONV(0);
        SBRN    *MDCNTR(1);
        MLRN    $A;
        ADRN    RTEM;
        STA     RTEM;
        XI      =1;
        TXLTM(2) ,EUCLID;
        LDA     RTEM;
        IAG     DISTEMP;
        SETE   -I.AND.E;
        SETE1  -G.AND.E1;
        STA     DISTEMP;
        LDA     PRESENTCAT;
        STA     CLASSTEMP;
        SETE    E.OR.-E;
        SETE1   E.OR.E;
        LDA     PRESENTCAT;
        ADM     ONE;
        STA     PRESENTCAT;
        TXLTM(1)  ,NEWCAT;
        LDA     CLASSTEMP;

        % IF THIS IS THE LAST ROW, LOAD  THE NUMBER OF CATS
        % IN A REG. OF ALL PE'S CONTAINING DUMMY VECTORS.
        TXLT(0)      $C0,NOTLASTROW;
        LDL(2)        .LASTROWOUTER;
        LDE          $C2;
        LDL(2)       .LASTROWINNER;
        LDE1         $C2;
        LDA          CATEG;
        SETE         E.OR.-E;
        SETE1        E.OR.E;
NOTLASTROW:
        LDL(1)  .SAVEVC;
        ZERF(3)  ,CHANGE;   % CHECK FOR ANY PTS. THAT HAVE
                            % CHANGED CATEGORY IF NO PTS.
                            % HAVE ALREADY CHANGED.
                            % EXIT FROM MODE MIGRATION OCCURS
                            % ONLY IF 2 PASSES THRU THIS MODE
                            % MIGRATION RESULTS IN NO CHANGE
                            % OF CLASSIFICATION FOR ANY PT.
        ILE     CLASS(1);
        SETC(2)  I;
        COMPC(2);
        COR(3)   $C2;
        SETC(2)  G;
        COMPC(2);
        COR(3)    $C2;      % $C3 IS NOW NON-ZERO IF ANY PTS.
                            % CHANGED CATEGORY.
```

```
CHANGE:
        % ENTER THE NEAREST CATEGORY INTO THE POINT'S CLASS
        % ARRAY ENTRY.
        STA       CLASS(1);
        LDS       $A;
        SETE1     E.AND.-E;
        LDA       #CLASSCNTR;
        ADRN      FLTONE;
        STA       #CLASSCNTR;
        SETE1     E.OR.E;
        LDA       $S;
        LDB       $A;
        SWAPX;
        LDS       $A;
        SETE      E.AND.-E;
        LDA       #CLASSCNTR;
        ADRN      FLTONE;
        STA       #CLASSCNTR;
        SETE      E.OR.-E;
        ALIT(1)  =1;
        STL(1)   .SAVEVC;
        TXLTM(0)  ,NEWPT;
        ZERT(3)  ,DISTINCT;   % IF $C3=0 NO POINTS CHANGED CLASS
                              % AND MODE MIGRATION IS COMPLETE.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                %MIGRATE THE MODES...FIND MODE CENTERS THAT
                %ARE THE AVERAGES OF ALL POINTS THAT HAVE
                %BEEN PLACED IN THE CATEGORY.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        LDL(1)    .CATCNTR;          %CATEGORY COUNTER.
        LDS       =0;                %PRESENT CLASS POINTER.
NEWCLASS::
        LDL(0)     .VECNTR;
      CLC(3);
        CLRA;
        STA       RTEM;
        STA       RTEM+1;
        STA       RTEM+2;
        STA       RTEM+3;
NEWPT1:
        LDA       CLASS(3);          %IS THE POINT IN THE CLASS
                                     %UNDER CONSIDERATION.
        LDX        =0;
        IME       $S;
        SETE      I.AND.E;           %TURN ON PE'S WHOSE POINTS ARE
                                     %IN THE CLASS.
        SETE1     G.AND.E1;
        LIT(2)    =1,3,0;
```

```
COMPNEW:
        LDA     *XCONV(0);          %SUM THE COMPONENTS OF ALL
                                    %VECTORS IN THIS CAT.
        ADRN    RTEM(2);
        STA     RTEM(2);
        SETE     E.OR.-E;
        XI      =1;
        SETE     I.AND.E;
        TXLTM(2) ,COMPNEW;
        SETE    E.OR.-E;
        SETE1   E.OR.E;
        ALIT(3)  =1;
        TXLTM(0) ,NEWPT1;
                %GET A SUM ACROSS PE'S OF THE NUMBER OF
                %POINTS IN THIS CATEGORY.
        LDA     #CLASSCNTR;
        RTL   $A,-1;
        ADRN  $R;
        RTL   $A,-2;
        ADRN  $R;
        RTL   $A,-4;
        ADRN  $R;
        RTL   $A,-8;
        ADRN  $R;
        RTL   $A,-16;
        ADRN  $R;
        RTL   $A,-32;
        ADRN  $R;
        LDB     $A;
        SWAPX;
        ADRN    $B;
        IMZ;                % CHECK FOR NULL CATEGORY.
        STA     #TOTCLASS;
        STA TTCLASS;
                %PACK AND ROUTE PARTIAL SUMS.
        LIT(2)  =2,2,0;
        LDX     =0;
PACKANDROUTE2:
        LDA     RTEM+1(2);
        LDB     $A;
        SWAPX;
        ADRN    $B;
        LDR     $A;
        LDA     RTEM(2);
        LDB     $A;
        SWAPX;
        ADRN    $B;
        LDB     $R;
        SWAPX;
        RTL   $A,-1;
```

```
        ADRN   $R;
        RTL    $A,-2;
        ADRN   $R;
        RTL    $A,-4;
        ADRN   $R;
        RTL    $A,-8;
        ADRN   $R;
        RTL    $A,-16;
        ADRN   $R;
        RTL    $A,-32;
        ADRN   $R;
        LDB      =0;
        SETE      -I.AND.E;          % PREVENT DIVIDE BY ZERO.
        SETE1     -G.AND.E1;
        DVRN      TTCLASS;
        SETE      E.OR.-E;           % RE-ENABLE ALL PE'S.
        SETE1     E.OR.E;
        LDB        $A;
        SWAPX;                    %UNPACK.
        STA        *MDCNTR(1);
        LDA        $B;
        XI         =1;
        STA        *MDCNTR(1);
        XI         =1;
        TXLTM(2)   ,PACKANDROUTE2;
        LDA       $S;
        ADM       ONE;              %INCREMENT CAT COUNTER.
        LDS        $A;
        TXETM(1) ,PAST55;
        JUMP  NEWCLASS;
PAST55:
        CLC(3);                  % CLEAR $C3...THIS IS THE FLAG
                                 % THAT IF NON-ZERO AFTER EUCLIDIAN
                                 % MODE ASSIGNMENT INDICATES THAT
                                 % SOME POINTS CHANGED CLASS DURING
                                 % THAT ASSIGNMENT.
        LDL(1)   .ITERCNT;
        ALIT(1)   =1;
        STL(1)   .ITERCNT;
        JUMP      DISTAN;        % GO COMPUTE DISTANCES OF ALL PTS.
                                 % TO THESE NEW MODE CENTERS.
DISTINCT::
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
                %YOU HAVE A GOOD INITIAL SET OF MODES...
                %COMPUTE SAMPLE VARIANCES TO PREPARE FOR
                %DISTINCTNESS TEST.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          CLRA;
          LDS    $A;
          STA       PRESENTCAT;
          LDL(1)    .CATCNTR;
NEWCLASS2:
          LDL(0)     .VECNTR;
          CLRA;
          STA       RTEM;
          STA       RTEM+1;
          STA       RTEM+2;
          STA       RTEM+3;
          CLC(3);
NEWPT2:
          LDA       CLASS(3);
          IME       PRESENTCAT;      %TURN ON PE'S WHOSE VECTOR
                                     %IS IN THIS CATEGORY.
          LDX       =0;
          SETE      I.AND.E;
          SETE1     G.AND.E1;
          LIT(2)    =1,3,0;
COMPNEW2:
          LDA       *XCONV(0);
          SBRN      *MDCNTR(1);
          MLRN      $A;
          ADRN      RTEM(2);
          STA       RTEM(2);
          SETE      E.OR.-E;
          XI        =1;
          SETE      I.AND.E;
          TXLTM(2)  ,COMPNEW2;
          SETE      E.OR.-E;
          SETE1     E.OR.E;
          ALIT(3)    =1;
          TXLTM(0)   ,NEWPT2;
          LDA       #TOTCLASS;
          SBRN       FLTONE;
          IMZ;      % SET I IN ALL PES IF THIS CATEGORY HAS ONLY
                    % ONE POINT.  THIS IS USED TO FLAG DIV ZERO
                    % WHEN DIVIDING BY #PTS. IN CAT. MINUS 1.
          STA       NOCLASSMINUS1;
          LIT(0)    =2,2,0;
          LDX        =0;
```

```
PACKANDROUTE3:
        LDA    RTEM+1(0);      %PACK TWO COMPONENT SUMS PER WORD
        LDB    $A;
        SWAPX;
        ADRN   $B;
        LDS    $A;
        LDA    RTEM(0);
        LDB    $A;
        SWAPX;
        ADRN    $B;
        LDB    $S;
        SWAPX;
                  %SUM OF ITH AND I+1ST COMPONENT
                  %LIE IN INNER AND OUTER PART OF $A.
        RTL    $A,-1;
        ADRN   $R;
        RTL    $A,-2;
        ADRN   $R;
        RTL    $A,-4;
        ADRN   $R;
        RTL    $A,-8;
        ADRN   $R;
        RTL    $A,-16;
        ADRN   $R;
        RTL    $A,-32;
        ADRN   $R;
        LDB       =0;
        SETE       -I.AND.E;    %PREVENT DIV ZERO.
        SETE1      -G.AND.E1;
        DVRN   NOCLASSMINUS1;
        SETE       E.OR.-E;
        SETE1      E.OR.E;
        LDB         $A;         %UNPACK.
        SWAPX;
        STA         *VARIANCE(1);
        LDA         $B;
        XI          =1;
        STA         *VARIANCE(1);
        XI          =1;
        TXLTM(0)    ,PACKANDROUTE3;
        LDA         ONE;
        ADM         PRESENTCAT;
        STA         PRESENTCAT;
     LDS    $A;
        TXLTM(1)    ,NEWCLASS2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  CLASSIFICATION OF ALL POINTS IS COMPLETE.  COMPUTE COVARIANCES
%  OF FINAL CATEGORIES.  THESE COVARIANCES ARE USED IN THE
%  DISRIMINANT FUNCTION ALGORITHM TO RECOGNIZE THESE CATEGORIES
```

```
%  IN OTHER WINDOWS.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ALLDISTINCT::
          CLRA;
          STA         PRESENTCAT;
          LDL(1)      .CATCNTR;
          CLC(3);
NEWCLASS3:
          STL(3)   .SAVECT;
          LDL(0)      .VECNTR;
          CLRA;
          STA         COVART;
          STA         COVART+1;
          STA         COVART+2;
          STA         COVART+3;
          STA         COVART+4;
          STA         COVART+5;
          CLC(3);
NEWPT3:
          STL(3)   .SAVEVC;
          LDA         CLASS(3);
          IME         PRESENTCAT;
          LDX         =0;
          SETE        I.AND.E;
          SETE1       G.AND.E1;
          LIT(2)      =1,3,0;
COMPNEW3:
          LDA         *XCONV(0);
          SBRN        *MDCNTR(1);
          STA         RTEM(2);
          SETE        E.OR.-E;
          XI          =1;
          SETE        I.AND.E;
          TXLTM(2)    ,COMPNEW3;
          SETE         E.OR.-E;
          LDX         =0;
          SETE         I.AND.E;
          LIT(2)      =1,3,0;
ADDC:
          LDL(3)      $C2;
          ALIT(3)     =1;
```

```
COVARC:
        LDA       RTEM(2);
        MLRN      RTEM(3);
        ADRN      *COVART;
        STA       *COVART;
        SETE      E.OR.-E;
        XI        =1;
        SETE      I.AND.E;
        TXLTM(3)  ,COVARC;
        ALIT(2)   =1;
        TXLT(2)   $C2,ADDC;
        SETE      E.OR.-E;
        SETE1     E.OR.E;
        LDL(3)    .SAVEVC;
        ALIT(3)   =1;
        TXLTM(0)  ,NEWPT3;
        LDS   PRESENTCAT;
        LDA       #TOTCLASS;
        SBRN      FLTONE;
        IMZ;    %SETE UP FOR DIV ZERO CHECK.
        STA       NOCLASSMINUS1;
        LDL(3)    .SAVECT;
        LIT(0)    =2,4,0;
        LDX       =0;
PACKANDROUTE4:
        LDA   COVART+1(0);   %PACK TWO COMPONENT SUMS PER WORD
        LDB   $A;
        SWAPX;
        ADRN  $B;
        LDS   $A;
        LDA   COVART(0);
        LDB   $A;
        SWAPX;
        ADRN   $B;
        LDB   $S;
        SWAPX;
              %SUM OF ITH AND I+1ST COMPONENT
              %LIE IN INNER AND OUTER PART OF $A.
        RTL   $A,-1;
        ADRN  $R;
        RTL   $A,-2;
        ADRN  $R;
        RTL   $A,-4;
        ADRN  $R;
        RTL   $A,-8;
        ADRN  $R;
        RTL   $A,-16;
        ADRN  $R;
        RTL   $A,-32;
        ADRN  $R;
```

```
        LDB      =0;
        SETE      -I.AND.E;
        SETE1     -G.AND.E;
        DVRN  NOCLASSMINUS1;
        SETE      E.OR.-E;
        SETE1     E.OR.E;
        LDB       $A;          %UNPACK.
        SWAPX;
        STA       *COVAR(3);
        LDA       $B;
        XI        =1;
        STA       *COVAR(3);
        XI        =1;
        TXLTM(0)  ,PACKANDROUTE4;
        LDA       ONE;
        ADM       PRESENTCAT;
        STA       PRESENTCAT;
        ALIT(3)   =6;
        TXLTM(1)  ,NEWCLASS3;
        LDL(0)    .VECNTR;
        LDX  =0;
        % ADD ONE TO ALL CAT NUMBERS.
FIX:
        LDA       *CLASS;
        ADM       ONE;
        SETE      E.AND.-E;
        SHAL      8;
        SETE      E.OR.-E;
        STA       *CLASS;
        XI  =1;
        TXLTM(0)  ,FIX;
        %%%%%%%%%%%%%%%%%%%%%
        %  PACK CLASSIFIED POINTS
        %%%%%%%%%%%%%%%%%%%%%
        CLC(2);
        STL(2)  .SV3;
        LDL(0)  .VECNTR;
BB:
        LIT(1)  =1,1,0;
ROW2:
        LDA  CLASS(0);
        CHWS  32;
        SETE1   E.AND.-E;
        SHAL  8;
        CHWS  64;
        SETE1  E.OR.-E;
        LDL(3)  .M1;
        LDEE1  $C3;
        SHAR  8;
        LDL(3)  .ALL;
```

```
        LDEE1   $C3;
        RTL   $A,-1;
        SHAL   24;
        OR   $R;
        LDL(3)   .MD1;
        LDEE1   $C3;
        LDS   $A;
        LDL(3)   .ALL;
        LDEE1   $C3;
        RTL   $A,-1;
        LDL(3)   .MD2;
        LDEE1   $C3;
        LDA   $R;
        LDS   $R;
        LDL(3)   .ALL;
        LDEE1   $C3;
        RTL   $A,-2;
        LDL(3)   .MD3;
        LDEE1   $C3;
        LDA   $R;
        LDS   $R;
        LDL(3)   .ALL;
        LDEE1   $C3;
        RTL   $A,-4;
        LDL(3)   .MD4;
        LDEE1   $C3;
        LDA   $R;
        LDS   $R;
        LDL(3)   .ALL;
        LDEE1   $C3;
        RTL   $A,-8;
        LDL(3)   .MD5;
        LDEE1   $C3;
        LDA   $R;
        LDS   $R;
        LDL(3)   .ALL;
        LDEE1   $C3;
        RTL   $A,-16;
        LDL(3)   .MD6;
        LDEE1   $C3;
        LDS   $R;
        LDL(3)   .ALL;
        LDEE1   $C3;
        LDL(3)   .SV3;
        RTL   $S,0(3);
        ALIT(3)   =32;
        STL(3)   .SV3;
        LDA   $R;
        LDL(3)   .MODE;
        LDEE1   $C3;
```

```
        CROTR(3)  32;
        STL(3)  .MODE;
        STA   CLASS(2);
        LDL(3)  .ALL;
        LDEE1  $C3;
        TXETM(0)  ,OUT1;
        TXLTM(1)  ,ROW2;
        ALIT(2)  =1;
        CLC(3);
        STL(3)  .SV3;
        SKIP  ,BB;
OUT1:
        % SET TYPE IN OUTPUT FILE HEADER TO TYPE 2.
        CHWS          64;
        LDL(2)      .PE0;
        LDEE1         $C2;
        LDA           =2;
        STA           HEADER;
        % SET CATEGORIES OUT IN HEADER.
        LDL(2)        .PE2;
        LDEE1         $C2;
        LDL(2)        .CATCNTR;
        CSHR(2)       26;
        ALIT(2)       =1;
        LIT(3)      =0,0,3F:16;
        CAND(2)     $C3;
        LDA           $C2;
        STA           HEADER;
        SETE          E.OR.-E;
        SETE1         E.OR.E;
        CHWS          32;
        % SET UP I/O REQUEST
        SETPAGE WIN,0;
        % SET THE PAGE COUNT FOR OUTPUT.
        LDL(2)  .VECTORS;
        ALIT(2) =1;
        CSHR(2) 4;
        ALIT(2) =1;
        CSHL(2) 24;
        LIT(1) =WIN;
        COR(2) $C1;
        SETCOUNT ($C2);
        PUT WIN,HEADER,COMPWORD;
        PAUSE COMPWORD;
        JUMP ERROREXIT;
        CLOSE WIN;

        % PACK MODE CENTERS, COVARIANCES INTO DISCRMINATE
        % FUNCTION ALGORITHM INPUT FORMAT.
```

```
        LDL(1)  .CATEG;
        ALIT(1)  =-1;
        CSHL(1) 24;
        CSB(1) 12;
        LDS  =0;
        LDA   PACKER;
        ADM  PACKEROFFSET;
        LDB    $A;
        LDX  $B;
PACKEM:
        LDA         *MDCNTR;
        LDB         *MDCNTR+1;
        SWAPX;
        STA         #MODCOV;
        LDA  $S;
        ADM  ONE;
        LDS  $A;
        LDA  $X;
        ADM  PACKERLOOP;
        LDB  $A;
        LDX  $B;
        TXLTM(1)    ,PACKEM;
        % SET TYPE TO 3...THIS IS A COVARIANCE MATRIX FILE.
        LDL(2)        .PE0;
        LDEE1         $C2;
        LDA           =3;
        STA           HEADER;
        SETE   E.OR.-E;
        SETE1  E.OR.E;
        % OUTPUT STATISTICS TO DISK.
        PUT COF,HEADER,COMPWORD;
        PAUSE  COMPWORD;
        JUMP ERROREXIT;
        CLOSE COF;
        SKIP          ,PAST1;
ERROREXIT::
        % SET CATEGORIES OUT TO ALL ONES TO FLAG ERROR.
        CLC(2);
        COMPC(2);
        LDL(3)        .PE2;
        LDEE1         $C3;
        LDA           $C2;
        STA           HEADER;
PAST1:
         HALT;
XCONV:  BLK  INMEM*4;        % CONVERTED INPUT STORAGE.
END START.
```