LARS Information Note 121569
Purdue University

THE IMPLEMENTATION OF THE MAXIMUM LIKELIHOOD CLASSIFICATION RULE ASSUMING A GAUSSIAN DENSITY FUNCTION

by

Terry L. Phillips

Several questions have been asked about the implementation of the classification algorithm used in LARSYSAA. The purpose of this Information Note is to briefly describe this implementation. It has been shown that the classification rule can be reduced to the following:[1,2]

Classify X as belonging to class ω_i if

$$\log |K_i| + (X - M_i)^T K_i^{-1} (X - M_i) \leq \log |K_j| + (X - M_j)^T K_j^{-1} (X - M_j) \quad \text{for all } j \neq i \tag{1}$$

For purposes of implementation of this decision rule, the programmer has chosen to follow the decision rule:

Classify X as belonging to class ω_i if

$$-\tfrac{1}{2} \log |K_i| - \tfrac{1}{2} (X-M_i)^T K_i^{-1} (X - M_i) \geq -\tfrac{1}{2} \log |K_j| - \tfrac{1}{2}(X - M_j)^T K_j^{-1} (X - M_j) \quad \text{for all } j \neq i \tag{2}$$

At classification time the classes have been defined and the statistics for each class have been calculated.

---

1. K. S. Fu, D. A. Landgrebe, T. L. Phillips, "Information Processing of Remotely Sensed Agricultural Data," Proceedings of the IEEE, Vol. 57, No. 4, April 1969, pp 639-653.

2. N. J. Nilsson, "Learning Machines," New York: McGraw-Hill, 1965.

Therefore the inputs to the classification program are:

- The number of features used in the classification process.
- The number of classes defined by the statistics set.
- $-\frac{1}{2} \log |K_i|$ for each class i.
- The elements of the inverse covariance matrix (K_i^{-1}) for each class i. Since this matrix is symmetric, only the lower triangular portion is stored and used in the calculation of the discriminant for each class i.
- The statistical mean vectors (one component for each feature) for each class i.
- A data vector with one component for each feature.

The outputs of the classification program are:

- A number indicating the class decision.
- The value of the discriminant function for the class decision which can be used for thresholding the decision when displaying the results.

The classification algorithm shown in Figure 1 calculates:

$$-\tfrac{1}{2} \log \left|K_i\right| \; -\tfrac{1}{2} (X - M_i)^T K_i^{-1} (X - M_i) \tag{3}$$

for each class i in the following manner. let $Y = (X - M_i)$ and K_{jk} equal a component of the matrix K_i^{-1}. Then equation (3) becomes:

$$-\tfrac{1}{2} \log \left|K_i\right| \; -\tfrac{1}{2} \begin{bmatrix} Y_1 K_{11} Y_1 + Y_1 K_{12} Y_2 + Y_1 K_{13} Y_3 + - - - - \\ + Y_2 K_{21} Y_1 + Y_2 K_{22} Y_2 + Y_2 K_{23} Y_3 + - - - - \\ + Y_3 K_{31} Y_1 + Y_3 K_{32} Y_2 + Y_3 K_{33} Y_3 + - - - - \\ \vdots \end{bmatrix} \tag{4}$$

And since $K_{21} = K_{12}$ equation 4 becomes:

$$-\tfrac{1}{2} \log \left|K_i\right| + \begin{bmatrix} -\tfrac{1}{2} Y_1 K_{11} Y_1 & & \\ - Y_2 K_{21} Y_1 & -\tfrac{1}{2} Y_2 K_{22} Y_2 & \\ - Y_3 K_{31} Y_1 & - Y_3 K_{32} Y2 & -\tfrac{1}{2} Y_3 K_{33} Y_3 \\ & \vdots & \end{bmatrix} \tag{5}$$

Equation (5) is implemented in the program shown in Figure 1. This discriminant is calculated for each class i and compared to the other discriminant calculations. The largest discriminant value indicates the classified class for the input data vector.

```
 MODEL 44  PS        VERSION 3,  LEVEL 3   DATE  69351
C*****************************************************************************
C                                                                              *
C  MLGC       REVISED 11/30/69 TLP                                             *
C                                                                              *
C             SUBROUTINE TO CLASSIFY A DATA POINT USING MAXIMUM                *
C                  LIKELIHCOD ASSUMING GAUSSIAN DISTRIBUTIONS.                 *
C                                                                              *
C*****************************************************************************
      SUBROUTINE MLGC(NF,NC,CON,COR,AVE,DATA,IC,TMAX)
C*****************************************************************************
C                                                                              *
C  DEFINE PROGRAM VARIABLES                                                    *
C                                                                              *
C*****************************************************************************
      INTEGER NF,NC,IC
      REAL CON(1),COR(1),AVE(1),DATA(1),TMAX
      REAL Y,X,FAC
      INTEGER NCF,JK,I,J,JI,K,KI
C*****************************************************************************
C                                                                              *
C NF          = THE NUMBER OF FEATURES USED IN THE CLASSIFICATION.            *
C NC          = THE NUMBER OF CLASSES DEFINED.                                *
C IC          = THE CLASS THE DATA VECTOR IS CLASSIFIED INTO.                 *
C CON(I)      = -1/2 LOG K(I), WHERE K(I) IS THE DETERMINANT OF THE           *
C                COVARIANCE MATRIX FOR EACH CLASS I.                           *
C COR(JK,I)   = THE ELEMENTS OF THE INVERSE COVARIANCE MATRIX FOR EACH        *
C                CLASS I. SINCE THE MATRIX IS SYMMETRIC, ONLY THE              *
C                LOWER TRIANGULAR PORTIONS ARE STORED. THUS THE                *
C                ELEMENTS ARE STORED IN THE FOLLOWING ORDER K11,K21,           *
C                K22,K31,..... FOR JK=1,2,3,4,.....                            *
C AVE(J,I)    = THE STATISTICAL MEAN VECTORS (J=1,NF COMPONENTS) FOR EACH      *
C                DEFINED CLASS (I=1,NC).                                       *
C DATA(J)     = THE DATA VECTOR TO BE CLASSIFIED (J=1,NF).                    *
C TMAX        = THE VALUE OF THE DISCRIMINANT FUNCTION FOR THE CLASSIFIED      *
C                DATA VECTOR FOR CLASS IC.                                     *
C Y           = THE CALCULATION OF THE DISCRIMINANT FUNCTION FOR EACH         *
C                CLASS.                                                        *
C X           = THE CALCULATION OF THE DATA MEAN - THE CLASS MEAN.            *
C FAC         = THE CALCULATION OF A FACTOR OF THE DISCRIMINANT FUNCTION.      *
C NCF         = A COUNTER FOR KEEPING TRACK OF CLASS AND FEATURE               *
C                COMPONENTS OF AVE.                                            *
C JK          = A COUNTER FOR KEEPING TRACK OF COMPONENTS OF COR.              *
C I           = THE CLASS COUNTER.                                            *
C J           = A FEATURE COUNTER.                                            *
C JI          = A COUNTER FOR KEEPING TRACK OF FEATURE AND CLASS (J AND        *
C                I) COMPONENTS OF AVE.                                         *
C K           = A FEATURE COUNTER.                                            *
C KI          = A COUNTER FOR KEEPING TRACK OF FEATURE AND CLASS (K AND        *
C                I) COMPONENTS OF AVE.                                         *
C                                                                              *
C*****************************************************************************
      NCF=-NF
      JK=0
      TMAX=-1.0E60
      DO 50 I=1,NC
      NCF=NCF+NF
      Y=CON(I)
      DO 30 J=1,NF
      JI=NCF+J
      X=DATA(J)-AVE(JI)
      DO 20 K=1,J
      JK=JK+1
      KI=NCF+K
      FAC=X*COR(JK)*(DATA(K)-AVE(KI))
   20 Y=Y-FAC
   30 Y=Y+0.5*FAC
      IF (Y.LE.TMAX) GO TO 50
      TMAX=Y
      IC=I
   50 CONTINUE
      RETURN
      END
```

Figure 1 Implementation of Classification Algorithm

Addendum
LARS Information Note 121569
Purdue University

Comparison of a FORTRAN-Encoded Gaussian Maximum Likelihood Classifier and an Assembly Language Equivalent

by

Philip H. Swain

In the LARS Information Note to which this is an addendum, the method of implementation of the Gaussian maximum likelihood classifier used in LARSYSAA is discussed in some detail, and a subroutine is given (MLGC) which performs the classification on a single data point in floating point form (see Figure 1 of the information note).

Figure A-2 shows the actual subroutine used by LARSYSAA. This routine (CONTEX) differs from MGLC only in that (1) it classifies an entire line of data points on each call, and (2) the data are input as halfword integers and are converted to floating point form by CONTEX before being classified.

Figure A-3 shows an assembly language equivalent (NEWCONTX) of CONTEX. The algorithm is unchanged, but advantage is taken of a priori knowledge about the algorithm which is unavilable to the FORTRAN compiler in generating machine language code. In particular, the relatively few DO variables and intermediate results can be maintained in registers rather than in core memory. The increase in efficiency of the assembly language program is substantial (see Figure A-1). NEWCONTX requires only 52% of the time required by CONTEX to classify 100 samples, assuming 10 pattern classes (the percentage is constant for one through twelve features and two through ten classes).

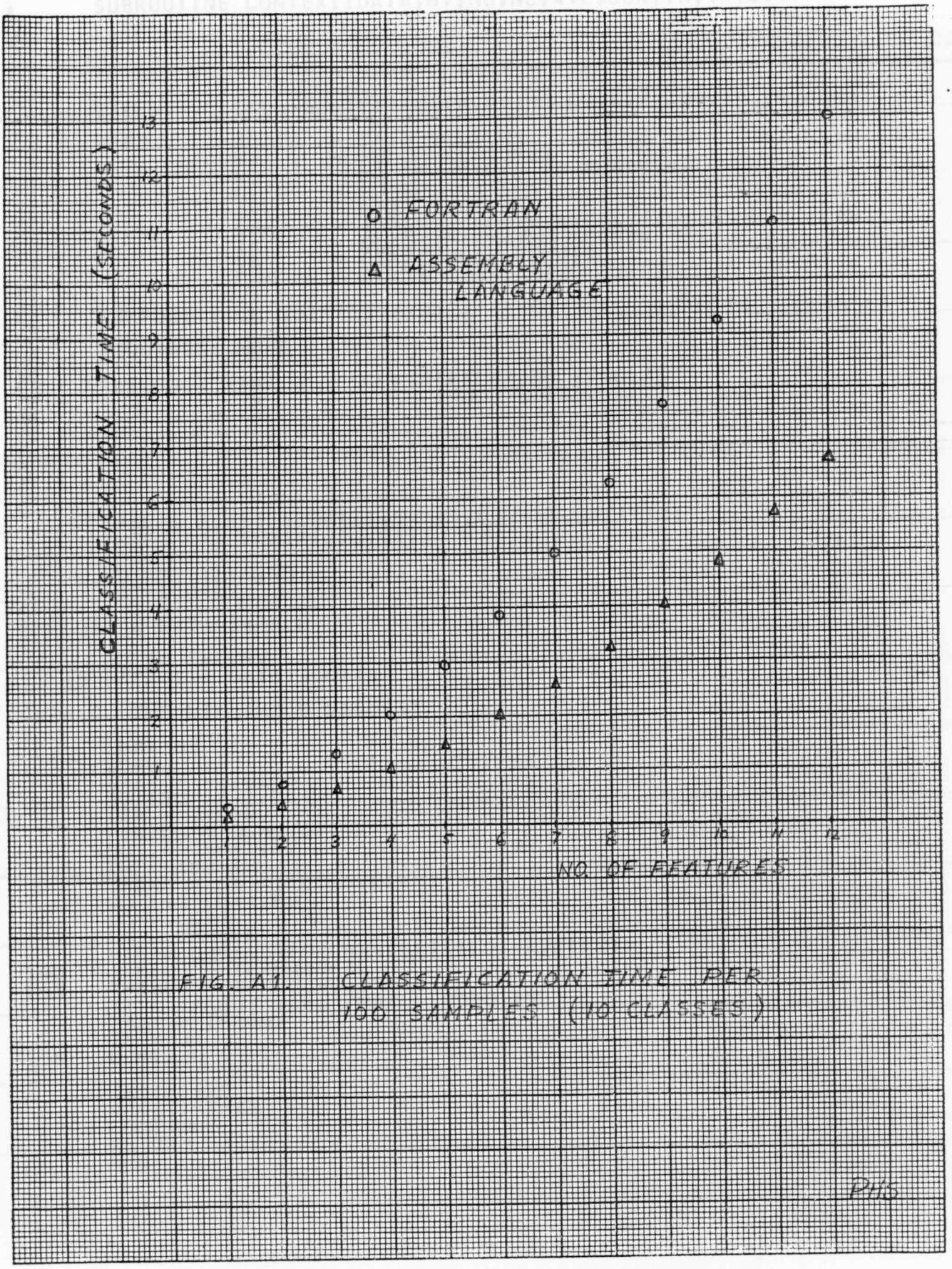

FIG. A1. CLASSIFICATION TIME PER 100 SAMPLES (10 CLASSES)

PHS

```
C     //CONTEX
      SUBROUTINE CONTEX(IDATA,NV,NO,NC,AVE,COR,IR,VR,CON)
C
C
CI----------------------------------------------------------------I
CI----------------------------------------------------------------I
CI                                                                 I
CI     CALL..     CALL CONTEX(IDATA,NV,NO,NC,AVE,COR,IR,VR,CON)   I
CI                                                                 I
CI     ARGS..     IDATA  - DATA VECTORS                           I
CI                NV     - NO OF FEATURES                         I
CI                NO     - NO OF POINTS TO CLASSIFY               I
CI                NC     - NO OF CLASSES                          I
CI                AVE    - MEAN VECTORS                           I
CI                COR    - INVERSE COVARIANCES (TRIANGULAR FORM)  I
CI                IR     - CLASSIFICATION BUFFER                  I
CI                VR     - LIKELIHOOD BUFFER                      I
CI                CON    - CLASS CONSTANTS                        I
CI                                                                 I
CI     PURPOSE.. CLASSIFIES 'IDATA' ('NO' PTS) INTO 'NC' CLASSES   I
CI                                                                 I
CI     REQUIRES. NONE                                              I
CI                                                                 I
CI     RETURNS.  IR     - CLASSIFICATION RESULTS                   I
CI               VR     - LIKELIHOOD BUFFER                       I
CI                                                                 I
CI----------------------------------------------------------------I
CI----------------------------------------------------------------I
C
C
C
      REAL*4 AVE(1),COR(1),VR(1),CON(1),DATA(12)
      INTEGER*2 IDATA(1),IR(1)
      IBD=-NV
      DO 200 II=1,NO
      IBD=IBD+NV
C
C FLOAT A DATA SAMPLE
C
      DO 10 I=1,NV
   10 DATA(I)=IDATA(IBD+I)
C
C CALC. THE LIKLIHOOD VALUES
C
      IBM=-NV
      LC=0
      TMAX = -1.0E60
      DO 150 JJ=1,NC
      IBM = IBM+NV
      TFREQJ = CON(JJ)
      DO 145 KD=1,NV
      KM = IBM+KD
      DKDKM = DATA(KD)-AVE(KM)
      DO 140 LD=1,KD
      LC = LC+1
      LM = IBM+LD
      FAC = COR(LC)*DKDKM*(DATA(LD)-AVE(LM))
  140 TFREQJ = TFREQJ-FAC
  145 TFREQJ = TFREQJ+0.5*FAC
      IF ( TFREQJ .LE. TMAX ) GOTO 150
      TMAX = TFREQJ
      IC = JJ
  150 CONTINUE
C
      IR(II)=IC
      VR(II)=TMAX
   200 CONTINUE
      RETURN
      END
```

Figure A-2. FORTRAN Classifier

```
NEWCONTX START 0
*I
*I-----------------------------------------------------------------------
*I-----------------------------------------------------------------------
*I
*I     CALL..     CALL CONTEX(IDATA,NV,NO,NC,AVE,COR,IR,VR,CON)
*I
*I     ARGS..     IDATA  - DATA VECTORS
*I                NV     - NO OF FEATURES
*I                NO     - NO OF POINTS TO CLASSIFY
*I                NC     - NO OF CLASSES
*I                AVE    - MEAN VECTORS
*I                COR    - INVERSE COVARIANCES (TRIANGULAR FORM)
*I                IR     - CLASSIFICATION BUFFER
*I                VR     - LIKELIHOOD BUFFER
*I                CON    - CLASS CONSTANTS
*I
*I     PURPOSE.. CLASSIFIES 'IDATA' ('NO' PTS) INTO 'NC' CLASSES
*I
*I     REQUIRES. NONE
*I
*I     RETURNS.  IR     - CLASSIFICATION RESULTS
*I               VR     - LIKELIHOOD BUFFER
*I
*I-----------------------------------------------------------------------
*I-----------------------------------------------------------------------
*I
          ENTRY CONTEX
          DC    CL7'CONTEX'
          DC    X'8'
*
*
SUB       EQU   14                  GENERAL REGISTER EQUATES
IBD       EQU   1
WRK       EQU   2
REG2      EQU   2
I         EQU   3
IBM       EQU   3
LC        EQU   4
JJ        EQU   5
KD        EQU   6
KM        EQU   7
LD        EQU   8
LM        EQU   9
IC        EQU   10
CORBASE   EQU   11
AVEBASE   EQU   12
*
FREGO     EQU   0                   FLOAT. PT. REG. EQUATES
DKDKM     EQU   0
TFREQJ    EQU   2
FAC       EQU   4
TMAX      EQU   6
*
*
          USING *,15
*
* SAVE REGISTERS
*
CONTEX    ST    0,20(13)
          ST    1,24(13)
          ST    2,28(13)
          ST    3,32(13)
          ST    4,36(13)
          ST    5,40(13)
          ST    6,44(13)
          ST    7,48(13)
          ST    8,52(13)
          ST    9,56(13)
          ST    10,60(13)
```

Figure A-3. Assembly Language Classifier

```
         ST     11,64(13)
         ST     12,68(13)
         ST     14,12(13)
*
* FETCH ARGS., INITIALIZE ARRAY REGS. & BASES
*
         L      REG2,4(1)       NV  NO. OF CHANNELS
         L      REG2,0(REG2)
         ST     REG2,NV
         L      REG2,8(1)       NO  NO. OF PTS. TO CLASSIFY
         L      REG2,0(REG2)
         ST     REG2,NO
         L      REG2,12(1)      NC  NO. OF CLASSES
         L      REG2,0(REG2)
         ST     REG2,NC
         L      REG2,0(1)       IDATA  INPUT DATA
         ST     REG2,IDBASE
         L      AVEBASE,16(1) AVE  MEAN VECTOR
         S      AVEBASE,FOUR    (REAL*4)
         L      CORBASE,20(1) COR  INVERSE COVARIANCE
         S      CORBASE,FOUR    (REAL*4)
         L      REG2,24(1)      IR  CLASSIFICATION CLASS
         S      REG2,TWO          (INTEGER*2)
         ST     REG2,IRBASE
         L      REG2,28(1)      VR  CLASSIFICATION LIKELIHOOD
         S      REG2,FOUR         (REAL*4)
         ST     REG2,VRBASE
         L      REG2,32(1)      CON  ADDITIVE CONSTANTS
         S      REG2,FOUR         (REAL*4)
         ST     REG2,CONBASE
*
*
* THE ASSEMBLY LANGUAGE IS ROUGHLY EQUIVALENT TO THE FORTRAN
*      STATEMENT SHOWN (NOTE *ROUGHLY*).
*
*
* BEGIN ALGORITHM
*
*  FLOAT A SAMPLE
*
         L      IBD,IDBASE
*
*  DO 200 II=1,NO
*
         L      REG2,ONE
DO200    ST     REG2,II
*
*  DO 10 I=1,NV
*
         L      I,ONE
*
*  DATA(I)=IDATA(IBD)          CONVERT TO FLOATING ROUTINE
*
DO10     LH     SUB,0(IBD)
         LPR    WRK,SUB
         ST     WRK,TEMP
         LD     FREG0,TEMPM3
         AD     FREG0,MASK
         LTR    SUB,SUB
         BNM    SKIP
         LCDR   FREG0,FREG0
SKIP     LR     SUB,I
         SLA    SUB,2
         STE    FREG0,DATA-4(SUB)
         LA     I,1(I)
         LA     IBD,2(IBD)
         C      I,NV
         BNH    DO10
*
* MAIN LOOP
```

Figure A-3. Continued

```
*
*   IBM=-NV
*
            L       IBM,NV
            LCR     IBM,IBM
*
*   LC=0
*
            SR      LC,LC
*
*   TMAX=-1.0E60
*
            LE      TMAX,=E'-1.0E60'
*
*   DO 150 JJ=1,NC
*
            L       JJ,ONE
*
*   IBM=IBM+NV
*
DO150       A       IBM,NV
*
*   TFREQJ=CON(JJ)
*
            LR      SUB,JJ
            SLA     SUB,2
            L       REG2,CONBASE
            AR      REG2,SUB
            LE      TFREQJ,0(REG2)
*
*   DO 145 KD=1,NV
*
            L       KD,ONE
*
*   KM=IBM+KD
*
DO145       LR      KM,IBM
            AR      KM,KD
*
*   DKDKM=DATA(KD)-AVE(KM)
*
            LR      SUB,KD
            SLA     SUB,2
            LE      DKDKM,DATA-4(SUB)
            LR      SUB,KM
            SLA     SUB,2
            SE      DKDKM,0(SUB,AVEBASE)
*
*   DO 140 LD=1,KD
*
            L       LD,ONE
*
*   LC=LC+1
*
DO140       LA      LC,1(LC)
*
*   LM=IBM+LD
*
            LR      LM,IBM
            AR      LM,LD
*
*   FAC=COR(LC)*DKDKM*(DATA(LD)-AVE(LM))
*
            LR      SUB,LD
            SLA     SUB,2
            LE      FAC,DATA-4(SUB)
            LR      SUB,LM
            SLA     SUB,2
            SE      FAC,0(SUB,AVEBASE)
            MER     FAC,DKDKM
```

Figure A-3. Continued

```
         LR     SUB,LC
         SLA    SUB,2
         ME     FAC,0(SUB,CORBASE)
*
*  TFREQJ=TFREQJ-FAC
*
         SER    TFREQJ,FAC
         LA     LD,1(LD)
         CR     LD,KD
         BNH    DO140
*
*  TFREQJ=TFREQJ+0.5*FAC
*
         ME     FAC,=E'0.5'
         AER    TFREQJ,FAC
         LA     KD,1(KD)
         C      KD,NV
         BNH    DO145
*
*  IF(TFREQJ.LE.TMAX) GO TO 150
*
         CER    TFREQJ,TMAX
         BNH    ST150
*
*  TMAX=TFREQJ
*
         LER    TMAX,TFREQJ
*
*  IC=JJ
*
         LR     IC,JJ
ST150    LA     JJ,1(JJ)
         C      JJ,NC
         BNH    DO150
*
*  IR(II)=IC
*
         L      SUB,II
         SLA    SUB,1
         L      REG2,IRBASE
         AR     REG2,SUB
         STH    IC,0(REG2)
*
*  VR(II)=TMAX
*
         SLA    SUB,1
         L      REG2,VRBASE
         AR     REG2,SUB
         STE    TMAX,0(REG2)
         L      REG2,II
         LA     REG2,1(REG2)
         C      REG2,NO
         BNH    DO200
*
* RETURN
*
         L      0,20(13)
         L      1,24(13)
         L      2,28(13)
         L      3,32(13)
         L      4,36(13)
         L      5,40(13)
         L      6,44(13)
         L      7,48(13)
         L      8,52(13)
         L      9,56(13)
         L      10,60(13)
         L      11,64(13)
         L      12,68(13)
         L      14,12(13)
```

Figure A-3. Continued

```
         MVI   12(13),X'FF'
         BR    14
*
* STORAGE
*
         DS    0D
TEMPM3   DC    X'4E000000'
TEMP     DS    1F
MASK     DC    D'0'
*
ONE      DC    F'1'
TWO      DC    F'2'
FOUR     DC    F'4'
*
II       DS    1F
NV       DS    1F
NO       DS    1F
NC       DS    1F
IDBASE   DS    1F
IRBASE   DS    1F
VRBASE   DS    1F
CONBASE  DS    1F
DATA     DS    12F
*
         END
/*
/&
```

Figure A-3. Concluded