AgRISTARS

SR-P0-00469
NAS9-15466

A Joint Program for Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing

Supporting Research

July 1980

Technical Report

A Multispectral Data Simulation Technique

by Marwan J. Muasher and Philip H. Swain

Purdue University
Laboratory for Applications of Remote Sensing
West Lafayette, Indiana 47907

NASA

AgRISTARS

SR-P0-00469
NAS9-15466

A Joint Program for Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing

July 1980

Supporting Research

Technical Report

A Multispectral Data Simulation Technique

by Marwan J. Muasher and Philip H. Swain

Purdue University
Laboratory for Applications of Remote Sensing
West Lafayette, Indiana 47907

NASA

SR-PO-00469
NAS9-15466

TECHNICAL REPORT

A MULTISPECTRAL DATA SIMULATION TECHNIQUE

By

M. J. Muasher

and

P. H. Swain

This report describes activity carried out
in the Supporting Research Project.

PURDUE UNIVERSITY
LABORATORY FOR APPLICATIONS OF REMOTE SENSING
1220 Potter Drive
WEST LAFAYETTE, INDIANA 47906, U.S.A.

JULY 1980

Star Information Form

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
SR-PO-00469		

4. Title and Subtitle	5. Report Date
A MULTISPECTRAL DATA SIMULATION TECHNIQUE	July 1980
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Marwan J. Muasher and Philip H. Swain	070980

9. Performing Organization Name and Address	10. Work Unit No.
Laboratory for Applications of Remote Sensing Purdue University West Lafayette, IN 47906	11. Contract or Grant No. NAS9-15466

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics and Space Administration Johnson Space Center Houston, TX 77058 Tech. Monitor: Richard Heydorn	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

In remote sensing data analysis, several assumptions are made that are not always precisely met. These assumptions include that the classes in the data are normally distributed, that training data are representative of the area of interest, that the number of classes is known, and that all pixels are pure.

In testing new algorithms, deviations from the assumptions may obscure the action of the new process. One way to clarify the situation is to apply the algorithm first to a data set satisfying the assumptions.

A method is presented to obtain an artificial data set through simulation. While retaining the natural spatial and spectral information in the scene by basing the simulation on a classification, the data set provides the analyst with an exact number of classes in the scene, true distributions of these classes, independent measurements and "pure" pixels.

Program listings in both Fortran and C-Language are provided in the appendices.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Data Simulation Multispectral Data	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified		

*For sale by the National Technical Information Service, Springfield, Virginia 22161

NASA - JSC

A MULTISPECTRAL DATA SIMULATION TECHNIQUE*

Marwan J. Muasher and Philip H. Swain

For remote sensing data analysis, several assumptions are commonly made. These assumptions are usually that the data are class-conditionally distributed multivariate normal and that the data used to train the classifier are representative of the area of interest. This second assumption actually has several parts. The assumption is made that in the process of training, all classes present in the scene are found, and all spectral subclasses of each class are also represented in the training data. Furthermore, the parameters of the distribution of each subclass are also assumed to be known from the training data. Each pixel is assumed to come from one of the training classes, and also is assumed to be entirely of one cover type.

In actual practice, these assumptions are not met. The number of spectral classes in the area is not known and clustering or some other method is used to determine the number of subclasses, in addition to estimating the statistics of those subclasses. Some of these methods also lead to non-normal subclasses. In particular, the clustering algorithm available through LARSYS truncates the tails of the subclass distributions and so leads to non-normal distributions.

There are also questions relating to a single picture element. A single pixel in Landsat data covers an area approximately 80 meters by 50 meters. More than one cover type may be present in this area and result in a "mixture pixel" observation. It is not clear how the distribution of the spectral response of mixture pixels can be related to the distribution of the spectral response of "pure pixels."

---

*This work was sponsored by NASA Contract NAS9-15466.

There has been much speculation in the remote sensing community as to the effect of the non-satisfaction of the basic assumptions. Whenever new algorithms are brought forth, the old questions are raised again, indicating that there is insufficient understanding of the interaction of the real attributes of the data and the theory of the algorithms. At times it is not clear whether a particular result is due to aspects of the algorithm or to the extent the data set deviates from the assumptions.

In testing new algorithms, deviations from the assumptions may obscure the action of the new process. One way to clarify the situation is to apply the algorithm first to a data set satisfying the assumptions.

Such a data set could be obtained artificially, through simulation. The analyst could then know: how many classes exist in the data; the true distributions of the classes, including normality if desired; the observations could really be independent; and no pixel would be a "mixture pixel." New algorithms could be studied on such a data set with the knowledge that any "strange" effects are indeed algorithm rather than data problems.

In many cases where simulated data have been used in the past, the data were too artificial, in the sense that all aspects of the image were controlled, removing the natural variation in object size, position, and relationship which occur in real data. This limited the use of the simulated data sets in testing new algorithms.

The natural spatial information occurring in multispectral data could be retained in a simulated image by spatially basing the simulation on a classification. It would be even better to base the simulated data on a digitized "ground truth" map if the spectral characteristics of the cover types were known. By basing the simulation on a classification, the number of classes, their exact distributions, and the class of each pixel in the

area are known. If the classification was sufficiently accurate, then the spatial information held in the classification map will be close to the actual cover type map and actual spatial content of the original data. For each pixel in the area, a random vector distributed according to the pixel's class statistics could be generated. This becomes the simulated data vector.

Statistical Background

From the classification chosen as a basis for the simulation, the following are known: the number of classes K, the set of classes $\{\omega_i, i=1,\ldots,K\}$, the class distributions $\{f(\omega_i), i=1,\ldots,K\}$, their means and covariances $\{\mu_i$ and $\Sigma_i, i=1,\ldots,K\}$, the number of channels p, and the class of every pixel in the scene.

From classifical statistics:

(1) Let X:pxl, A:pxp and b:pxl.
If $X \sim N(0, I_p)$, then $Y = AX + b \sim N(b, AI_pA^T = AA^T)$
(where I_p is the identity matrix having dimensionality p).

(2) Let Σ be a symmetric, positive definite matrix. Then there exists A, such that

$$AA^T = \Sigma \qquad (A \text{ is denoted } \Sigma^{\frac{1}{2}})$$

To simulate a pixel which was a member of class i in the base classification, $N(0, I_p)$ (the random vector for each pixel is indepedent of other vectors) is generated. (See Appendix I.) Next $Y = \Sigma_i^{\frac{1}{2}} X + \mu_i$ is calculated; it is then a random vector from the population $N(\mu_i, \Sigma_i)$. This process is repeated for each pixel of the base classification and the random vectors thus generated are stored appropriately, i.e., so as to correspond to their simulated spatial location.

The program requires as an input a classification map stored on a results tape. The results tape has the class statistics

for p-dimensions also stored on it. The program, then, uses the results map and the stored statistics to generate a p-dimensional data set, which is stored on a user specified output tape in LARSYS format.

Appendix I provides a mathematical derivation related to the generation of normally distributed samples. Appendix II provides the Fortran program listing for the simulation program. Appendix III provides the C program listing for the same program.

APPENDIX I

APPENDIX I

Let U_1 and U_2 be two random variables independent and identically distributed Uniform (0,1).

Then let $Z_1 = (-2 \ln U_1)^{\frac{1}{2}} \cos 2\pi U_2$

and $Z_2 = (-2 \ln U_1)^{\frac{1}{2}} \sin 2\pi U_2$

then Z_1 and Z_2 are independent and identically distributed normal (0,1).

Proof:

$$f(U_1,U_2) = \begin{cases} 1 & 0<U_1<1,\ 0<U_2<1 \\ 0 & \text{otherwise} \end{cases}$$

is the probability density function of two independent uniforms.

$$U_1 = \exp\left[-\tfrac{1}{2}(Z_1^2 + Z_2^2)\right]$$

$$U_2 = \frac{1}{2\pi} \arctan\left(\frac{Z_2}{Z_1}\right)$$

The Jacobian of the transformation is:

$$J = -\frac{1}{2\pi} \exp\left[-\tfrac{1}{2}(Z_1^2 + Z_2^2)\right]$$

$$f(Z_1,Z_2) = f(U_1,U_2) \cdot |J|$$

$$= \frac{1}{2\pi} \exp\left[-\tfrac{1}{2}(Z_1^2 + Z_2^2)\right] \qquad 0 < \left[\exp -\tfrac{1}{2}(Z_1^2 + Z_2^2)\right] < 1$$

$$0 < \frac{1}{2\pi} \arctan\left(\frac{Z_2}{Z_1}\right) < 1$$

$$= 0 \quad \text{otherwise}$$

$$\therefore f(Z_1) \sim N(0,1) \qquad f(Z_2) \sim N(0,1)$$

The side conditions give $-\infty < Z_1 < \infty$, $-\infty < Z_2 < \infty$. Strictly speaking, Z_1 cannot equal zero; however, prob $(Z_1 = 0) = 0$ as we are working with continuous densities.

To test the effectiveness of the pseudo random vectors in the multivariate case, random vectors distributed $N(0,I_p)$ were generated and then tested with a Kolmogorov-Smirnov test. Since the multivariate normal cdf is difficult to evaluate, the sum of squares was calculated and compared to the χ_p^2 distribution.

For sample sizes greater than 100, the pseudo random vectors were distributed properly. For sample sizes less than 100, the K-S test is not valid. Since we would generally (over an entire area) be working with more than 100 points per class, this was not pursued further.

In addition, the sample covariance matrices were tested for homogeneity against the true class statistics. For sample runs of up to 2000 points, there were not significant differences at the $\alpha = 0.10$ level.

APPENDIX II

FILE: SWRITE FORTRAN A PURDUE / LARS 3031

```
      DO 100 IP=1,NOCLAS                                                    SWRC3170
      DO 100 IO=1,NOCHAN                                                    SWRC3180
      IF(CLAPNT(IP).LE.0) GO TO 98                                          SWRC3190
      RMEAN(IP,IO)=FLOAT(IMEAN(IP,IO))/FLOAT(CLAPNT(IP))                    SWRC3200
   98 DO 100 IT=IO,NOCHAN                                                   SWRC3210
      IF(CLAPNT(IP).LE.1) GO TO 100                                         SWRC3220
      REPNT=FLOAT(CLAPNT(IP))                                               SWRC3230
      REVAR=FLOAT(IVAR(IP,IO,IT))                                           SWRC3240
      REMEAN=FLOAT(IMEAN(IP,IO))                                            SWRC3250
      SEMEAN=FLOAT(IMEAN(IP,IT))                                            SWRC3260
      RVAR(IP,IO,IT)=(1./(REPNT-1.))*(REVAR-REMEAN*SEMEAN/REPNT)            SWRC3270
      RVAR(IP,IT,IO)=RVAR(IP,IO,IT)                                         SWRC3280
  100 CONTINUE                                                              SWRC3290
      DO 645 IP=1,NOCLAS                                                    SWRC3300
      WRITE(6,605)IP,CLAPNT(IP)                                             SWRC3310
  605 FORMAT(1H1/5X,'CLASS NUMBER',I3,5X,I8,' POINTS'///)                   SWRC3320
      WRITE(6,610)                                                          SWRC3330
  610 FORMAT(37X,'ACTUAL',4X,'SIMULATED')                                   SWRC3340
      WRITE(6,615)                                                          SWRC3350
  615 FORMAT(38X,'MEAN',7X,'MEAN'/)                                         SWRC3360
      DO 622 IX=1,NOCHAN                                                    SWRC3370
      NINC=NOCOMP*NOCLAS+(IP-1)*NOCHAN                                      SWRC3380
      WRITE(6,620)FETVC3(IX),FRQCAL(1,FETVC3(IX)),FRQCAL(2,FETVC3(IX)),ZSWRC3390
     $2(NINC+IX),RMEAN(IP,IX)                                               SWRC3400
  620 FORMAT(5X,'CHANNEL',I3,2X,'(',F5.2,'-',F5.2,')',5X,F8.3,3X,F8.3)      SWRC3410
  622 CONTINUE                                                              SWRC3420
      WRITE(6,625)                                                          SWRC3430
  625 FORMAT(/////5X,'ACTUAL COVARIANCE MATRIX')                            SWRC3440
      DO 630 NO=1,NOCOMP                                                    SWRC3450
      NINC=(IP-1)*NOCOMP                                                    SWRC3460
  630 A(NO)=Z2(NINC+NO)                                                     SWRC3470
      CALL WRTMTX(A,NOCHAN,FRQCAL,THREE,FETVC3)                             SWRC3480
      WRITE(6,635)                                                          SWRC3490
  635 FORMAT(////5X,'SIMULATED COVARIANCE MATRIX')                          SWRC3500
      NO=0                                                                  SWRC3510
      DO 640 IO=1,NOCHAN                                                    SWRC3520
      DO 640 IN=1,IO                                                        SWRC3530
      NO=NO+1                                                               SWRC3540
  640 A(NO)=RVAR(IP,IO,IN)                                                  SWRC3550
      CALL WRTMTX(A,NOCHAN,FRQCAL,THREE,FETVC3)                             SWRC3560
  645 CONTINUE                                                              SWRC3570
      CALL TOPEF(12,IER)                                                    SWRC3580
      DO 650 IX=3,200                                                       SWRC3590
  650 IDREC(IX)=0                                                           SWRC3600
      CALL TOPWR(12,80C,IER,IDREC)                                          SWRC3610
      IF(IER.NE.0) WRITE(6,234)IER                                          SWRC3620
      IF(IER.GT.0) GO TO 310                                                SWRC3630
      GO TO 320                                                             SWRC3640
  234 FORMAT(5X,'ERROR IS',I5)                                              SWRC3650
C                                                                           SWRC3660
C*************************************************************************SWRC3670
C ERROR MESSAGES                                                            SWRC3680
C*************************************************************************SWRC3690
C                                                                           SWRC3700
  300 WRITE(6,305)                                                          SWRC3710
  305 FORMAT(5X,'ERROR -1')                                                 SWRC3720
  310 WRITE(6,315)                                                          SWRC3730
  315 FORMAT(5X,'ERROR GT 1')                                               SWRC3740
  320 STOP                                                                  SWRC3750
      END                                                                   SWRC3760
```

FILE: SWRITE FORTRAN A PURDUE / LARS 3031

```
C 150 CONTINUE                                                                  SWRC2380
C                                                                               SWRC2390
      CALL TOPWR(12,800,IER,IDREC)                                              SWRC2400
      IF(IER.NE.0) WRITE(16,234)IER                                             SWRC2410
      IF(IER.GT.0) GO TO 310                                                    SWRC2420
      DO 50 MA=1,NOCLAS                                                         SWRC2430
      CLAPNT(MA)=0                                                              SWRC2440
      DO 50 MB=1,NOCHAN                                                         SWRC2450
      IMEAN(MA,MB)=0                                                            SWRC2460
      RMEAN(MA,MB)=0.0                                                          SWRC2470
      DO 50 MC=1,NOCHAN                                                         SWRC2480
      IVAR(MA,MB,MC)=0                                                          SWRC2490
   50 RVAR(MA,MB,MC)=0.0                                                        SWRC2500
      LNWRT = 0                                                                 SWRC2510
   55 READ(11)J,K,LINENO,(PNTCLS(IX),IX=1,NOPNTS)                               SWRC2520
      IF(J.GT.6) GO TO 95                                                       SWRC2530
      LNWRT=LNWRT+1                                                             SWRC2540
      IF (MOD(LNWRT,25).EQ.0) WRITE(16,57)LNWRT,NOLINE                          SWRC2550
   57 FORMAT(5X,I4,' LINES OUT OF ',I4,' ARE COMPLETED')                        SWRC2560
C                                                                               SWRC2570
C*****************************************************************************SWRC2580
C  GENERATE AND WRITE DATA POINTS                                               SWRC2590
C*****************************************************************************SWRC2600
C                                                                               SWRC2610
   60 I2=ILIN(2)                                                                SWRC2620
      DATOUT(1)=L1(1)                                                           SWRC2630
      DATOUT(2)=L1(2)                                                           SWRC2640
      I2=32767                                                                  SWRC2650
      DATOUT(3)=L1(1)                                                           SWRC2660
      DATOUT(4)=L1(2)                                                           SWRC2670
      I2=0                                                                      SWRC2680
      ICOUNT=4                                                                  SWRC2690
      DO 90 IX=1,NOPNTS                                                         SWRC2700
      ICOUNT=ICOUNT+1                                                           SWRC2710
      I2=PNTCLS(IX)                                                             SWRC2720
      L1(1)=.FALSE.                                                             SWRC2730
      IPOL=(I2-1)*NOCHAN                                                        SWRC2740
      IBEG=(I2-1)*NOCOMP                                                        SWRC2750
      K=IBEG                                                                    SWRC2760
      DO 65 IY=1,NOCHAN                                                         SWRC2770
      DO 65 IZ=1,IY                                                             SWRC2780
      K=K+1                                                                     SWRC2790
      B(IY,IZ)=Z(K)                                                             SWRC2800
      IF(IY.EQ.IZ) GO TO 65                                                     SWRC2810
      B(IZ,IY)=0.0                                                              SWRC2820
   65 CONTINUE                                                                  SWRC2830
      DO 70 IY=1,NOCH                                                           SWRC2840
      CALL RANDU(ISTART(IY),NXINP,A2(IY))                                       SWRC2850
      ISTART(IY)=NXINP                                                          SWRC2860
      CALL RANDU(ISTART(IY),NXINP,A(IY))                                        SWRC2870
      ISTART(IY) = NXINP                                                        SWRC2880
      A(IY)=SQRT(-2.*ALOG(A2(IY)))*COS(6.28318*A(IY))                           SWRC2890
   70 CONTINUE                                                                  SWRC2900
      CLAPNT(I2)=CLAPNT(I2)+1                                                   SWRC2910
      DO 80 IY=1,NOCHAN                                                         SWRC2920
      DATA(IY)=0.0                                                              SWRC2930
      IQ=NOPOOL*NOCOMP+IPOL+IY                                                  SWRC2940
      DO 75 IZ=1,NOCHAN                                                         SWRC2950
   75 DATA(IY)=DATA(IY)+B(IY,IZ)*A(IZ)                                          SWRC2960
      DATA(IY)=DATA(IY)+Z(IQ)                                                   SWRC2970
      INTDAT=DATA(IY)+.5                                                        SWRC2980
      IF(INTDAT.LT.0) INTDAT=0                                                  SWRC2990
      IF(INTDAT.GT.255) INTDAT=255                                              SWRC3000
      ISTAT(IY)=INTDAT                                                          SWRC3010
      DATOUT((IY-1)*NOSAM+ICOUNT)=LOGDAT(2)                                     SWRC3020
      DO 92 IZ=1,6                                                              SWRC3030
   92 DATOUT((IY-1)*NOSAM+ICOUNT+IZ)=.FALSE.                                    SWRC3040
   80 CONTINUE                                                                  SWRC3050
      DO 90 II=1,NOCHAN                                                         SWRC3060
      IMEAN(I2,II)=IMEAN(I2,II)+ISTAT(II)                                       SWRC3070
      DO 90 JJ=II,NOCHAN                                                        SWRC3080
      IVAR(I2,II,JJ)=IVAR(I2,II,JJ)+ISTAT(II)*ISTAT(JJ)                        SWRC3090
   90 CONTINUE                                                                  SWRC3100
      NOBYTE=4+NOCHAN*NOSAM                                                     SWRC3110
      CALL TOPWR(12,NOBYTE,IER,DATOUT)                                          SWRC3120
      IF(IER.NE.0) WRITE(16,234)IER                                             SWRC3130
      IF(IER.GT.0) GO TO 310                                                    SWRC3140
      GO TO 55                                                                  SWRC3150
   95 CONTINUE                                                                  SWRC3160
```

FILE: SWRITE FORTRAN A PURDUE / LARS 3031

```
      IBEG=1                                                                    SWRC1590
C                                                                               SWRC1600
C*****************************************************************************SWRC1610
C  FACTOR COVARIANCE MATRICES                                                   SWRC1620
C*****************************************************************************SWRC1630
C                                                                               SWRC1640
      DO 30 IX=1,NOPOOL                                                         SWRC1650
      IDONE=IBEG+NOCOMP-1                                                       SWRC1660
      K=0                                                                       SWRC1670
      DO 20 IY=IBEG,IDONE                                                       SWRC1680
      K=K+1                                                                     SWRC1690
   20 A(K)=Z(IY)                                                                SWRC1700
      CALL MFSD(A,NOCHAN,EPS,IER)                                               SWRC1710
      IF(IER.EQ.-1) GO TO 300                                                   SWRC1720
      IF(IER.GE.1) GO TO 310                                                    SWRC1730
      K=0                                                                       SWRC1740
      DO 25 IY=IBEG,IDONE                                                       SWRC1750
      K=K+1                                                                     SWRC1760
   25 Z(IY)=A(K)                                                                SWRC1770
   30 IBEG=IBEG+NOCOMP                                                          SWRC1780
C                                                                               SWRC1790
C*****************************************************************************SWRC1800
C  GENERATE STARTING POINTS                                                     SWRC1810
C*****************************************************************************SWRC1820
C                                                                               SWRC1830
   29 WRITE(16,31)                                                              SWRC1840
   31 FORMAT(5X,'DO YOU WANT TO SPECIFY THE STARTING POINTS FOR THE'/5XSWRC1850
     $,'RANDOM NUMBER GENERATOR? (TYPE YES OR NO)')                             SWRC1860
      READ(16,32)INPUT                                                          SWRC1870
   32 FORMAT(A4)                                                                SWRC1880
      IF(INPUT.EQ.NO) GO TO 36                                                  SWRC1890
      IF(INPUT.EQ.YES) GO TO 33                                                 SWRC1900
      GO TO 29                                                                  SWRC1910
   33 DO 39 IX=1,NOCHAN                                                         SWRC1920
      WRITE(16,41)IX                                                            SWRC1930
   41 FORMAT(5X,'SPECIFY STARTING POINT FOR CHANNEL',I3/5X,'(TYPE A NINESWRC1940
     $ DIGIT ODD NUMBER)')                                                      SWRC1950
      READ(16,42)ISTART(IX)                                                     SWRC1960
   42 FORMAT(I9)                                                                SWRC1970
   39 CONTINUE                                                                  SWRC1980
      GO TO 43                                                                  SWRC1990
   36 CALL GTSERL(ISERL)                                                        SWRC2000
      ISERL=(ISERL/10)*8+1                                                      SWRC2010
      DO 40 I=1,NOCH                                                            SWRC2020
      ISERL=ISERL+10000000                                                      SWRC2030
      ISTART(I)=ISERL                                                           SWRC2040
   40 CONTINUE                                                                  SWRC2050
   43 WRITE(6,34)                                                               SWRC2060
   34 FORMAT('1'//5X,'STARTING POINTS FOR RANDOM NUMBER GENERATOR'//)           SWRC2070
      DO 44 I=1,NOCHAN                                                          SWRC2080
      WRITE(6,35)I,ISTART(I)                                                    SWRC2090
   35 FORMAT(5X,'STARTING POINT FOR CHANNEL ',I2,' IS ',I9)                     SWRC2100
   44 CONTINUE                                                                  SWRC2110
C                                                                               SWRC2120
C*****************************************************************************SWRC2130
C READ CLASSIFICATIONS                                                          SWRC2140
C*****************************************************************************SWRC2150
C                                                                               SWRC2160
      IDREC(1)=TAPEND                                                           SWRC2170
      IDREC(2)=JFILE                                                            SWRC2180
      IDREC(3) = RUNNO                                                          SWRC2190
      NOLD = IDREC(5)                                                           SWRC2200
      IDREC(5) = NOCHAN                                                         SWRC2210
      IDREC(6) = 4*(NOPNTS + 9)/4)                                              SWRC2220
      NOSAM = IDREC(6)                                                          SWRC2230
      IDREC(7) = FLGT                                                           SWRC2240
      DO 141 II=1,3                                                             SWRC2250
      IDREC(II+16) = DATE(II)                                                   SWRC2260
  141 CONTINUE                                                                  SWRC2270
      IDREC(20) = NOLINE                                                        SWRC2280
C     DO 145 II= 1, NOCHAN                                                      SWRC2290
C     INEW = FETVC3(II)                                                         SWRC2300
C     DO 145 II2 = 1,5                                                          SWRC2310
C     FRQCAL(II2,II) = FRQCAL(II2,INEW)                                         SWRC2320
C 145 CONTINUE                                                                  SWRC2330
C     LIP = NOCHAN + 1                                                          SWRC2340
C     DO 150 II = LIP,NOLD                                                      SWRC2350
C     DO 150 II2 = 1,5                                                          SWRC2360
C     FRQCAL(II2,II) = 0.0                                                      SWRC2370
```

```
FILE: SWRITE   FORTRAN  A   PURDUE / LARS 3031

      DATA  FLGT /'SIM '/                                                    SWR00800
      EPS=1.E-5                                                              SWR00810
C                                                                            SWR00820
C*************************************************************************  SWR00830
C LOAD TAPES AND READ PARAMETERS                                             SWR00840
C*************************************************************************  SWR00850
C                                                                            SWR00860
      WRITE(16,500)                                                          SWR00870
  500 FORMAT(///5X,'SPECIFY TAPE NUMBER ON WHICH RESULTS FILE IS LOCATED'    SWR00880
     $/5X,'(TYPE EIGHT DIGIT TAPE NUMBER)')                                  SWR00890
      READ(16,505)INTAP                                                      SWR00900
  505 FORMAT(I8)                                                             SWR00910
      WRITE(16,510)                                                          SWR00920
  510 FORMAT(5X,'SPECIFY FILE NUMBER AT WHICH RESULTS FILE IS LOCATED'/5X    SWR00930
     $,'(TYPE THREE DIGIT FILE NUMBER)')                                     SWR00940
      READ(16,515)IFILE                                                      SWR00950
  515 FORMAT(I3)                                                             SWR00960
      CALL MMTAPE(INTAP,IFILE,C)                                             SWR00970
      WRITE(16,570)                                                          SWR00980
  570 FORMAT(///5X,'SPECIFY THE TAPE NUMBER ONTO WHICH SIMULATED DATA IS     SWR00990
     $TO BE WRITTEN'/5X,'(TYPE EIGHT DIGIT TAPE NUMBER)')                    SWR01000
      READ(16,575)TAPEND                                                     SWR01010
  575 FORMAT(I8)                                                             SWR01020
      WRITE(16,580)                                                          SWR01030
  580 FORMAT(5X,'SPECIFY FILE NUMBER AT WHICH SIMULATED DATA IS TO BE WR     SWR01040
     $ITTEN'/5X,'(TYPE THREE DIGIT FILE NUMBER)')                            SWR01050
      READ(16,585)JFILE                                                      SWR01060
  585 FORMAT(I3)                                                             SWR01070
      WRITE(16,590)                                                          SWR01080
  590 FORMAT(///5X,'SPECIFY THE RUN NUMBER FOR THE SIMULATED DATA RUN'/      SWR01090
     1      5X,'(TYPE EIGHT DIGIT RUN NUMBER)')                             SWR01100
      READ(16,575) RUNNO                                                     SWR01110
      CALL MOUNT(TAPEND,12,'RI')                                             SWR01120
      MARG=JFILE-1                                                           SWR01130
      IF(MARG.LE.0) GO TO 3                                                  SWR01140
      DO 3 LIP=1,MARG                                                        SWR01150
      CALL TOPFF(12)                                                         SWR01160
    3 CONTINUE                                                               SWR01170
    5 READ(11)I                                                              SWR01180
      IF(I.NE.1) GO TO 310                                                   SWR01190
      READ(11)I,J,NOCLAS,NOCHAN,NOFLDS,NOPOOL,(FETVC3(IX),IX=1,NOCHAN)       SWR01200
      NOCH=((NOCHAN+1)/2)*2                                                  SWR01210
      NOCOMP=NOCHAN*(NOCHAN+1)/2                                             SWR01220
      ISTOP=NOCOMP*NOPOOL                                                    SWR01230
      IEND=ISTOP+NOCHAN*NOPOOL                                               SWR01240
   15 READ(11)I,J,K                                                          SWR01250
      IF(I.LT.3) GO TO 15                                                    SWR01260
      IF(K.NE.EOS) GO TO 15                                                  SWR01270
      READ(11)I,J,(Z(IX),IX=1,IEND)                                          SWR01280
      DO 17 IX=1,IEND                                                        SWR01290
      Z2(IX)=Z(IX)                                                           SWR01300
   17 CONTINUE                                                               SWR01310
   45 READ(11)I,AREANO,NOPNTS,NOLINE,INFO,IDREC                              SWR01320
      NOFET3=NOCHAN                                                          SWR01330
      IF(I.NE.5) GO TO 45                                                    SWR01340
      WRITE(6,520)                                                           SWR01350
  520 FORMAT(1H1////5X,'+++++++++++++++++++++++++++++++++++++++')            SWR01360
      WRITE(6,525)                                                           SWR01370
  525 FORMAT(5X,'+DATA SIMULATION USING MCCABES EQUATION+')                  SWR01380
      WRITE(6,530)                                                           SWR01390
  530 FORMAT(5X,'+++++++++++++++++++++++++++++++++++++++')                   SWR01400
      WRITE(6,535) RUNNO, IDREC(3)                                           SWR01410
  535 FORMAT(////5X,'SIMULATED DATA RUN IS',I9,' FROM RUN',I9)               SWR01420
      WRITE(6,537)INFO(4),INFO(5),INFO(7),INFO(8)                            SWR01430
  537 FORMAT(/5X,'LINE',I5,' TO LINE',I5,' AND COLUMN',I5,' TO COLUMN',I5    SWR01440
     $)                                                                      SWR01450
      WRITE(6,540)INTAP,IFILE                                                SWR01460
  540 FORMAT(/5X,'INPUT RESULTS FILE IS ON TAPE',I9,' FILE',I4)              SWR01470
      WRITE(6,545)TAPEND,JFILE                                               SWR01480
  545 FORMAT(/5X,'SIMULATED DATA IS ON TAPE',I9,' FILE',I4)                  SWR01490
      WRITE(6,550)                                                           SWR01500
  550 FORMAT(/5X,'CHANNELS USED')                                            SWR01510
      DO 560 IX=1,NOCHAN                                                     SWR01520
      WRITE(6,555)FETVC3(IX),FRQCAL(1,IX),FRQCAL(2,IX)                       SWR01530
  555 FORMAT(5X,I2,2X,F5.2,'-',F5.2)                                         SWR01540
  560 CONTINUE                                                               SWR01550
      CALL GTDATE(DATE)                                                      SWR01560
      WRITE(6,565)DATE                                                       SWR01570
  565 FORMAT(/5X,'DATE OF SIMULATION IS ',3A4)                               SWR01580
```

FILE: SWRITE FORTRAN A PURDUE / LARS 3031

```
C     ******************************************************************
C     WRITTEN BY: BILL PFAFF
C     EDITED BY: MARWAN MUASHER  JUNE 14,1980
C     ******************************************************************
C
C
C     ******************************************************************
C          THIS PROGRAM GENERATES SIMULATED DATA BASED ON A
C     CLASSIFICATION MAP OR A GROUND TRUTH MAP.  EACH PIXEL
C     GENERATED THUS COMES FROM A KNOWN CLASS DISTRIBUTION.  THE
C     METHOD USED IS AS FOLLOWS:
C     1. A GOOD CLASSIFICATION IS CHOSEN AS A BASE FOR
C        SIMULATED DATA
C     2. FROM THIS CLASSIFICATION WE KNOW THE NUMBER OF CLASSES, THE
C        CLASS STATISTICS, AND THE CLASS OF EACH PIXEL IN THE
C        AREA CLASSIFIED.
C     3. A STREAM OF UNIFORM RANDOM NUMBERS IS GENERATED FOR
C        EACH CHANNEL.  THEY ARE CHANGED TO NORMAL (0,1) DEVIATES.
C     4. FOR EACH PIXEL, A RANDOM N(0,I)VECTOR IS TRANSFORMED TO
C        BE DISTRIBUTED ACCORDING TO THE CLASS STATISTICS OF THAT
C        PIXEL.  THIS IS THE SIMULATED DATA VECTOR.
C     5. AS EACH LINE IS COMPLETED, IT IS WRITTEN TO AN OUTPUT TAPE.
C          TO RUN THE PROGRAM, YOU NEED TO HAVE THE FOLLOWING
C     EXEC FILE ON YOUR DISK:
C
C     GETDISK LARSYS
C     GETDISK DVSYS
C     GLOBAL TXTLIB CMSLIB FORTRAN SSP370
C     FILEDEF 6 PRINTER
C     FILEDEF 16 TERMINAL
C     FILEDEF 12 TAP2
C     FILEDEF 11 TAP1 (RECFM VS LRECL 1500 BLKSIZE 1500)
C     LOAD SWRITE GLOCOM MMTAPE TAPOP 3CDVAL GTSERL GTDATE MFSD
C     RANDU WRTMTX
C     START SWRITE
C
C          THE PROGRAM WILL ASK FOR INFORMATION SUCH AS
C     TAPE NUMBERS, FILE NUMBERS,  ..ETC.  FROM HERE ON, IT
C     SHOULD BE EASY TO FOLLOW.
C     ******************************************************************
C
C
C     ******************************************************************
C    VARIABLES USED IN TPRINT
C
C    A      =COVARIANCE STORAGE FOR FACTORING
C    AREANO=AREA NUMBER OF CLASSIFICATION
C    B      =COVARIANCE STORAGE FOR MULTIPLICATION
C    DATA   =DATA POINT STORAGE
C    DATVAL=LINE NUMBER AND ROLL PARAMETER
C    ICAL   =CALIBRATION INFORMATION
C    IDREC  =IDENTIFICATION RECORD STORAGE
C    ISTART=STARTING POINTS FOR GAUSS
C    LOGDAT=DATA POINTS IN LOGICAL FORMAT
C    NOCHAN=NUMBER OF CHANNELS IN CLASSIFICATION
C    NOCLAS=NUMBER OF CLASSES IN ORIGINAL STATISTICS
C    NOFLDS=NUMBER OF TEST FIELDS
C    NOPOOL=NUMBER OF POOLED CLASSES
C    PNTCLS=CLASSIFICATIONS ARRAY
C    Z      =STATISTICS STORAGE
C     ******************************************************************
C
C
C*********************************************************************
C  INITIALIZATION
C*********************************************************************
C
      INTEGER*2 I2,INTDAT,ICAL(3),ILIN(2),PNTCLS(1000),ISTAT(4),
     $          FETVC3(30)
      LOGICAL*1 L1(2),LOGDAT(2),LCAL(6),DATOUT(12000)
      REAL*4 A(78),A2(12),Z(2700),B(12,12),DATA(12),
     $       RMEAN(30,12),RVAR(30,12,12),Z2(2700),FRQCAL(5,30)
      INTEGER*4 ISTART(12),EOS,INFO(17),AREANO,IDREC(200),TAPENO,THREE,
     $          CLAPNT(30),IMEAN(30,12),IVAR(30,12,12),YES,NO,DATE(3)
      INTEGER*4  RUNNO,FLG
      EQUIVALENCE (I2,L1),(INTDAT,LOGDAT),(ICAL,LCAL),(LNWRT,ILIN)
      EQUIVALENCE(FRQCAL(1,1),IDREC(51))
      DATA EOS,S,AM /'EOS ',1.0,0.0 /
      DATA YES,NO,THREE /'YES ','NO  ','3'/
```

APPENDIX III

```
/*************************************************
*
*  Read data from LARS Results Tape
*  and simulate data from it
*  using the Box Muller relationship
*
*************************************************
*
*  Swrite.c has been translated from
*  the Lars Fortran Version of Swrite into
*  the language 'c' for the Unix O.S.
*  run on the DEC PDP-11/45
*
*************************************************
*
*  Variables used in Swrite:
*
*     a         == Covariance storage for factoring
*     b         == Covariance storage for multiplication
*     data      == Data point storage
*     nochan    == Number of channels in Classification
*     noclas    == Number of classes in original statistics
*     nopool    == Number of pooled classes
*     pntcls    == Classifications array
*     z         == Statistics storage
*
*************************************************
*
*  compile with      cc swrite.c -lp -lm  /usr/lib/pdslib
*
*************************************************
*
*  Initialize all variables used in swrite
*  External variables are available to all functions
*  within which they are declared
*/
main(){
   extern int noclas, nochan, nopool, fd1, nopnts, noline, recnum;
   extern int ier, fd2;
   extern float eps, upper[5], lower[5];
   int j, k, fetvc3[5], debug, info[17], pntcls[1000], jj, ipol;
   int ii, idone, iy, ma, mb, mc;
   int noch, nocomp, istop, iend, ix, ibeg;
   int lnwrt, i2, icount, iz, ia, intdat, istat[6], ip, io, it, ninc, no, in;
   int nosam, pos, pfd1;
   int fd3, fd4, err5;
   char buf[1500], *delim, *c1, *c2, *c3, datout[2100], obuf[2100];
   char pname[30], tt1[41];
   char mvfile[30], cmfile[30], opname[30];
   float z[610], repnt, r;
   float b[6][6];
   double rmean[31][6];
   float data[13], z2[610];
   float clapnt[35];
   float mobuf[30];
   double ivar[31][6][6], revar, imean[31][6], t1float, t2float, remean, semean;
   double temp1, temp2, temp0, a[20], a2[6];
   double sqrt(), log(), cos(), fmod(), s, t, f;
 /* begin main program
 *  if debug is set to minus one the following is
```

```
*  printed out for verification on usr terminal:
*     nochan, nopool, fetvc3, info, pntcls, nopnts, noline, z, upper, lower;
*/
  debug = 1;
  eps = .00001;
/*  read records 2,4,5,&6 off results tape*/
/*  skip records 1,3   */
  read1to5(buf, fetvc3, z, info);
  noch = (((nochan+1)/2)*2);
  nocomp = (nochan*(nochan+1)/2);
  istop = nocomp * nopool;
  iend = nochan * nopool + istop;
  nosam = 4 * ((nopnts + 9)/4);
  if(nochan>5) {
    printf("Number of channels is %d but internal", nochan);
    printf(" storage only allows 5\n");
    printf("Execution terminated abnormally \n");
    exit();
    }
  if(nopool>29) {
    printf("Number of pooled classes is %d but internal", nopool);
    printf(" storage only allows 29\n");
    printf("Execution terminated abnormally \n");
    exit();
    }
  if(nopnts>=1000) {
    printf("Number of points per line is %d but internal", nopnts);
    printf(" storage only allows 1000\n");
    printf("Execution terminated abnormally \n");
    exit();
    }
  printf("nochan=%d, nopool=%d\n", nochan, nopool);
  if(debug == -1) {
    for(k=0; k<nochan; ++k)
       printf("fetvc3[%d] = %d \n", k, fetvc3[k]);
    for(k=0; k<iend; k = k+10) {
       for(j=k; j<=k+9; ++j)
          printf("  %f", z[j]);
       printf("\n");
       }
    printf("\nField size:");
    printf("\n          line %d to %d with interval %d",
            info[4], info[5], info[6]);
    printf("\n          cols %d to %d with interval %d\n",
            info[7], info[8], info[9]);
    printf("Number of lines classified is %d\n", noline);
    printf("Number of points classified per line is %d\n", nopnts);
    for(j=0; j<nochan; ++j) {
       printf("upper[%d] = %f ", j, upper[j]);
       printf("   lower[%d] = %f \n", j, lower[j]);
       }
    }
  read6rec(buf, pntcls);
  if(debug == -1) {
    printf("\nFirst line of record %d follows\n", recnum);
    for(j=0; j<=nopnts; j=j+10) {
       for(k=j; k<=j+9; ++k)
          printf("   %d", pntcls[k]);
       printf("\n");
       }
```

```
    }
/* make a copy of statistics array z */
  for(ix=1;ix<=iend;++ix)
     z2[ix] = z[ix-1];
/*  Create output file  */
  gets("\nSpecify linerprinter output file pathname: ",orname);
  fd2 = creat(orname,0777);
  if(fd2<0)
    printf("Cannot creat lr output file %s\n",orname);
  pfd1 = 1;
  gets("Specify simulated data PDS file pathname: ",pname);
  gets("enter a 40 char title:\n",ttl);
  gets("\nSpecify mean vector file pathname: ",mufile);
  gets("\nSpecify covariance matrix file pathname: ",cmfile);
  pdsopen(pfd1,pname);
  putfmt(pfd1,1,nosam,noline,8,nochan);
  putttl(pfd1,ttl);
  fd3 = creat(mufile,0777);
  if(fd3<0)
    printf("Cannot creat mufile %s  (MAIN)\n",mufile);
  fd4 = creat(cmfile,0777);
  if(fd4<0)
    printf("Cannot creat cmfile %s  (MAIN)\n",cmfile);
/* write output page one */
  delim = "      ++++++++++++++++++++++++++++++++++++++++++++++++++";
  c1 =    "      +Data Simulation Using Box-Muller Relationship+";
  printf(fd2,"\n\n\n\n%s\n%s\n%s\n",delim,c1,delim);
  printf(fd2,"\n   Line %d to line %d with interval %d \n",
         info[4],info[5],info[6]);
  printf(fd2,"\n   Column %d to column %d with interval %d \n",
         info[7],info[8],info[9]);
  printf(fd2,"\n      channels used \n");
  for(ix=1;ix<=nochan;++ix)
     printf(fd2,"      %d   %f - %f\n",fetvc3[ix-1],lower[ix-1],
            urrer[ix-1]);
/* output new page character */
  printf(fd2,"\014");
/*  Factor covariance matrices  */
  ibeg = 1;
  for(ix=1;ix<=norool;++ix) {
     idone = ibeg + nocomp - 1;
     k = 0;
     for(iy=ibeg;iy<=idone;++iy) {
        k = k + 1;
        a[k] = z[iy-1];
        }
 /* call function to perform factoring  */
     mfsd(a);
     if(ier == -1)
       printf(fd2,"Error -1\n");
     if(ier>=1)
       printf(fd2,"Error st 1\n");
     if((ier == -1) || (ier >= 1))
       exit(1);
     k = 0;
     for(iy=ibeg;iy<=idone;++iy) {
        k = k + 1;
        z[iy-1] = a[k];
        }
     ibeg = ibeg + nocomp;
```

```
    }
/*  initialize random number generator */
  srand(1);
/*  initialize arrays   */
  for(ma=1;ma<=noclas;++ma) {
     clprnt[ma] = 0.0;
     for(mb=1;mb<=nochan;++mb) {
        imean[ma][mb] = 0.0;
        rmean[ma][mb] = 0.0;
        for(mc=1;mc<=nochan;++mc) {
           ivar[ma][mb][mc] = 0.0;
           }
        }
     }
  t = 25.0;
  lnwrt = 0;
/*  this while loop is repeated for each line in the classification  */
  while(recnum == 6) {
    lnwrt = lnwrt + 1;
    s = lnwrt;
    f = fmod(s,t);
    if(f == 0.0)
      printf("%d  Lines out of %d are completed\n",
             lnwrt,noline);
    i2 = 0;
    icount = 4;
    for(ix=1;ix<=nprnts;++ix) {
       icount = icount + 1;
       i2 = prtcls[ix-1];
       ipol = (i2-1) * nochan;
       ibeg = (i2-1) * nocomp;
       k = ibeg;
       for(iy=1;iy<=nochan;++iy) {
          for(iz=1;iz<=iy;++iz) {
             k = k + 1;
             b[iy][iz] = z[k-1];
             if(iy != iz)
               b[iz][iy] = 0.0;
             }
          }
       for(iy=1;iy<=noch;++iy) {
          a2[iy] = rand();
          a[iy] = rand();
          a2[iy] = a2[iy] / 32767.;
          a[iy] = a[iy] / 32767.;
          a[iy] = sqrt( -2.0 * log(a2[iy])) * cos(6.28318 * a[iy]);
          }
       clprnt[i2] = clprnt[i2] + 1.0;
       for(iy=1;iy<=nochan;++iy) {
          data[iy] = 0.0;
          iq = nopool * nocomp + ipol + iy;
          for(iz=1;iz<=nochan;++iz)
             data[iy] = data[iy] + b[iy][iz] * a[iz];
          data[iy] = data[iy] + z[iq-1];
          intdat = data[iy] + .5;
          if(intdat<0)  intdat = 0;
          if(intdat>255)  intdat = 255;
          istat[iy] = intdat;
          pos = (iy-1)*nosam + icount - 5;
          if(pos>2100) printf("datout internal buffer full\n");
```

```
            datout[pos] = intdat & 0377;
            for(iz=1;iz<=6;++iz)
               datout[pos+iz] = 0;
            }
          for(ii=1;ii<=nochan;++ii) {
            imean[i2][ii] = imean[i2][ii] + istat[ii];
            for(jj=ii;jj<=nochan;++jj) {
              temp0 = istat[ii];
              temp1 = istat[jj];
              ivar[i2][ii][jj] = ivar[i2][ii][jj] + temp0 * temp1;
               }
            }
          }
        ii = 0;
        for(iy=0;iy<nosam;++iy) {
           for(iz=0;iz<nochan;++iz)
             obuf[ii++] = datout[nosam*iz + iy];
          }
        pdslpos(pfd1,lnwrt-1);
        if(putline(pfd1,lnwrt-1,obuf,nosam*nochan) != 0)
          printf("Putline to PDS output failed\n");
        read6rec(buf,pntcls);
        }
/*   end of while loop  */
/*
   for(i2=1;i2<=18;++i2) {
     printf(fd2,"\n %d \n",i2);
     for(ii=1;ii<=nochan;++ii) {
       for(jj=ii;jj<=nochan;++jj) {
          printf(fd2," %f  ",ivar[i2][ii][jj]);
          }
       printf(fd2,"\n");
       }
    }
   for(i2=1;i2<=18;++i2) {
      printf(fd2,"\n");
      for(ii=1;ii<=nochan;++ii) {
         printf(fd2," %f \n",imean[i2][ii]);
         }
      }
*/
   for(ip=1;ip<=noclas;++ip) {
      for(io=1;io<=nochan;++io) {
         if(clapnt[ip]>0.0) {
           t1float = imean[ip][io];
           t2float = clapnt[ip];
           rmean[ip][io] = t1float/t2float;
           }
         for(it=io;it<=nochan;++it) {
            if(clapnt[ip]>1.0) {
              repnt = clapnt[ip];
              revar = ivar[ip][io][it];
              remean = imean[ip][io];
              semean = imean[ip][it];
              temp0 = revar/(repnt-1.0);
              temp1 = remean/repnt;
              temp2 = semean/(repnt-1.0);
              ivar[ip][io][it] = temp0 - (temp1*temp2);
              ivar[ip][it][io] = ivar[ip][io][it];
              }
```

```
            }
          }
        }
/*  output results  */
  for(ip=1;ip<=noclas;++ip) {
      printf(fd2,"      Class number %d      %f    points\n\n\n\n",
             ip,clapnt[ip]);
      printf(fd2,"                                                          Actual     Simulated\n")
      printf(fd2,"                                                           Mean        mean\n");
      printf(fd2,"
      for(ix=1;ix<=nochan;++ix) {
         ninc = nocomp * noclas + (ip - 1) * nochan;
         printf(fd2,"      Channel %d  ( %6.3f - %6.3f )      %6.3f    %6.3f\n",
                fetvc3[ix-1],lower[ix-1],upper[ix-1],z2[ninc+ix],
                rmean[ip][ix]);
         mobuf[ix-1] = rmean[ip][ix];
         }
      err5 = write(fd3,mobuf,4*nochan);
      if(err5<0)
        printf("Error occurred writing Mean Vector file (MAIN) \n");
      printf(fd2,"\n\n\n\n\n\n      Actual Covariance Matrix\n");
      for(no=1;no<=nocomp;++no) {
         ninc = (ip-1) * nocomp;
         a[no] = z2[ninc+no];
         }
      wrtmtx(a,fetvc3);
      printf(fd2,"\n\n\n\n\n      Simulated Covariance Matrix\n");
      no = 0;
      for(io=1;io<=nochan;++io) {
         for(in=1;in<=io;++in) {
            no = no + 1;
            a[no] = ivar[ip][io][in];
            mobuf[no-1] = ivar[ip][io][in];
            }
         }
      wrtmtx(a,fetvc3);
      err5 = write(fd4,mobuf,4*nocomp);
      if(err5<0)
        printf("Error occurred writing cm file  (MAIN) \n");
      printf(fd2,"\014");
      }
 /* cexit terminates activities on open files and flushes
 *  the output buffer.  cexit is part of the c library  */
   printf("swrite.c is finished\n");
   printf("\n The simulated data (in PDS format) is at %s \n",pname);
   printf("The lineprinter output file is at %s \n",opname);
   printf("The Mean vector file is at %s \n",mvfile);
   printf("The Covariance Matrix file is at %s \n",cmfile);
   pdsclose(pfd1);
   cexit();
 }
 /*  end of main program  */
 /*******************************************************/
 /*   function to read records 1 thru 5 follows        */
 read1to5(buf,fetvc3,word,info)
   int *fetvc3,*info;
   char *buf;
   float *word;
 {
   extern int noclas,nochan,nopool,fd1,nopnts,noline;
   extern float  upper[5],lower[5];
```

```
  int err1, j, k, recnum, statsize;
  char tbuf[4];
  float ibmdec();
  fd1 = open("/dev/rmt0", 0);
  if(fd1<0)
    printf("Cannot open 9 trk tape file   (READ1TO5) \n");
/* skip record 1 */
  err1 = read(fd1, buf, 1500);
  if(err1 == -1)
    printf("Error occurred in reading record 1 (READ1TO5) \n");
/* read record 2 */
  err1 = read(fd1, buf, 1500);
  if(err1 == -1)
    printf("Error occurred reading record 2  (READ1TO5) \n");
  noclas = buf[19];
  nochan = buf[23];
  nopool = buf[31];
  for(j=0, k=33; j<nochan; ++j, k=k+4)
     fetvc3[j] = buf[k];
  for(k=32+4*nochan, j=0; j<2*nochan; ++j, k=k+4) {
     tbuf[0] = buf[k];
     tbuf[1] = buf[k+1];
     tbuf[2] = buf[k+2];
     tbuf[3] = buf[k+3];
     if(j<nochan)
       lower[j] = ibmdec(tbuf);
     else upper[j-nochan] = ibmdec(tbuf);
     }
/* skip record 3 */
  err1 = read(fd1, buf, 1500);
  if(err1 == -1)
    printf("Error occurred reading record 3 (READ1TO5) \n");
  recnum = buf[11];
  while(recnum == 3) {
     err1 = read(fd1, buf, 1500);
     if(err1 == -1)
       printf("Error occurred reading record three (READ1TO5)\n");
     recnum = buf[11];
     }
/* Handle record 4 */
  statsize = nochan*nopool + (nochan*(nochan+1)/2)*nopool;
  for(k=0, j=16; k<statsize; ++k, j=j+4) {
     tbuf[0] = buf[j];
     tbuf[1] = buf[j+1];
     tbuf[2] = buf[j+2];
     tbuf[3] = buf[j+3];
     word[k] = ibmdec(tbuf);
     }
  err1 = read(fd1, buf, 1500);
  if(err1 == -1)
    printf("Error occurred reading record 5 (READ1TO5) \n");
  nopnts = (buf[18]<<8 | (buf[19] & 0377));
  noline = (buf[22]<<8 | (buf[23] & 0377));
  info[4] = (buf[38]<<8 | (buf[39] & 0377));
  info[5] = (buf[42]<<8 | (buf[43] & 0377));
  info[6] = (buf[46]<<8 | (buf[47] & 0377));
  info[7] = (buf[50]<<8 | (buf[51] & 0377));
  info[8] = (buf[54]<<8 | (buf[55] & 0377));
  info[9] = (buf[58]<<8 | (buf[59] & 0377));
}
```

```
int noclas, nopool, nochan, fd1, nornts, noline;
float upper[5], lower[5];
float ibmdec(buf)
   char *buf;
{
  int k, l, m, tint;
  char nbuf[2], tchar, temp3, byte[4], sign;
  float word;
  m = 0;
  sign = 0000;
  if(buf[0] < 0)
    sign =0200;
  tint=(((buf[0] & 0177) - 64) * 4);
  while(buf[1] > 0) {
        buf[1]=((buf[1]<<1) & 0376);
        ++m;
        }
  k = 8-m;
  nbuf[0]=buf[2];
  nbuf[1]=buf[3];
  for(l=1; l<=k; ++l) {
     nbuf[0]=((nbuf[0]>>1) & 0177);
     nbuf[1]=((nbuf[1]>>1) & 0177);
     }
  for(l=1; l<=m; ++l) {
     buf[2]=((buf[2]<<1) & 0376);
     buf[3]=((buf[3]<<1) & 0376);
     }
  buf[1]=(buf[1] | nbuf[0]);
  buf[2]=(buf[2] | nbuf[1]);
  tint = tint-m+128;
  if(tint < 0) tchar = 0000;
  else if(tint > 255) tchar = 0377;
  else tchar = tint;
  buf[1]=(buf[1] & 0177);
  temp3 = tchar<<7;
  temp3 = temp3 & 0200;
  buf[1] = temp3 | buf[1];
  buf[0] = tchar>>1;
  buf[0] = buf[0] & 0177;
  buf[0] = buf[0] | sign;
  byte[0] = buf[1] & 0377;
  byte[1] = buf[0] & 0377;
  byte[2] = buf[3] & 0377;
  byte[3] = buf[2] & 0377;
  pack(byte, &word);
  return(word);
}
/* the function pack takes the 4 8-bit character bytes
*  that have been rearranged by ibmdec and packs
*  them into a floating point word  */
 pack(byte, cword)
    char *byte, *cword; {
  int j;
  for(j=0; j<4; ++j)
     cword[j] = byte[j];
 }
/*********************************************************/
/*  function to read record 6 follows:  */
read6rec(buf, rntcls)
```

```
   int *pntcls;
   char *buf; {
  extern int nopnts,recnum,fd1;
  int j,n,err1,k;
  n = 0;
  err1=read(fd1,buf,1500);
  if(err1 == -1)
    printf("Error occurred reading record 6\n");
  recnum=buf[11];
  for(j=20,k=0; k<nopnts; k++,j=j+2) {
     pntcls[n]=(buf[j+1] & 0377);
     ++n;
     }
  }
 int recnum;
/*******************************************************/
/* Mfsd factors the matrices passed by the main program.
*  Mfsd is a direct translation from the Lars
*  fortran version of the same name  */
mfsd(a)
   double *a; {
  extern int nochan,ier;
  extern float eps;
  int kpiv,k,ind,lend,i,lanf,l,lind;
  double tol,dsum,dpiv,y,fabs();
  double done,sqrt(),x;
  done = 1.0;
  ier = 0;
  kpiv = 0;
  for(k=1;k<=nochan;++k) {
     kpiv = kpiv+k;
     ind = kpiv;
     lend = k-1;
     y = eps*a[kpiv];
     tol = fabs(y);
     for(i=k;i<=nochan;++i){
        dsum = 0.0;
        if(lend != 0) {
          for(l=1;l<=lend;++l) {
             lanf = kpiv-l;
             lind = ind-l;
             dsum = (dsum+(a[lanf]*a[lind]));
             }
          }
        dsum = a[ind]-dsum;
        if((i-k) == 0) {
          if(((dsum-tol) <= 0.0) && (dsum > 0.0) && (ier<=0))
            ier = k-1;
          if(((dsum-tol) <= 0.0) && (dsum <= 0.0)) {
            ier = -1;
            printf("Error -1\n");
            exit(1);
            }
          x = dsum;
          dpiv = sqrt(x);
          a[kpiv] = dpiv;
          dpiv = done/dpiv;
          }
        if((i-k) != 0)
          a[ind] = dsum*dpiv;
```

```
        ind = ind+i;
        }
      }
  }
 int ier;
 float eps;
/*********************************************************/
/* the write matrix function writes to the outrut
*  file the lower half of the nochan*nochan covariance matrix
*  which is passed throush the parameter a.    */
wrtmtx(a,fetvc3)
   int *fetvc3;
   double *a;    {
  extern int fd2,nochan;
  extern float upper[5],lower[5];
  int j,k,m;
  m = 0;
  printf(fd2,"\n      Spectral          ");
  for(j=0;j<nochan;++j)
     printf(fd2,"%7.5f -      ",lower[j]);
  printf(fd2,"\n      Band              ");
  for(j=0;j<nochan;++j)
     printf(fd2,"%7.5f        ",upper[j]);
  for(j=0;  j<nochan;  ++j) {
     printf(fd2,"\n\n      %7.5f -\n",lower[j]);
     printf(fd2,"       %7.5f          ",upper[j]);
     for(k=0;  k<=j;  ++k) {
        printf(fd2,"%7.3f        ",a[++m]);
        }
     }
  }
 int fd2;
/*********************************************************/
gets(p,s)
char *p,*s;  {
  char c,*p1;
  printf(p);
  p1 = s;
  while((c = getchar()) != '\n')
       *p1++ = c;
  *p1 = 0;
}
```