FATIGUE STRENGTH REDUCTION MODEL:
RANDOM3 and RANDOM4 USER MANUAL

Prepared by:
Lola Boyce, Ph.D., P.E.
Thomas B. Lovelace

APPENDIX 2
of Annual Report
of Project Entitled
Development of Advanced Methodologies
for Probabilistic Constitutive Relationships
of Material Strength Models

NASA Grant No. NAG 3-867

Prepared for:
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Lewis Research Center
Cleveland, OH 44135

The Division of Engineering
The University of Texas at San Antonio
San Antonio, TX 78285
January, 1989
FATIGUE STRENGTH REDUCTION MODEL:
RANDOM3 and RANDOM4 USER MANUAL

Prepared by:
Lola Boyce, Ph.D., P.E.
Thomas B. Lovelace

APPENDIX 2
of Annual Report
of Project Entitled
Development of Advanced Methodologies
for Probabilistic Constitutive Relationships
of Material Strength Models

NASA Grant No. NAG 3-867

Prepared for:
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
Lewis Research Center
Cleveland, OH 44135

The Division of Engineering
The University of Texas at San Antonio
San Antonio, TX 78285
January, 1989
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>SECTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2.0 Theoretical Background</td>
<td>2</td>
</tr>
<tr>
<td>3.0 Input Data</td>
<td>4</td>
</tr>
<tr>
<td>4.0 Sample Problem</td>
<td>7</td>
</tr>
<tr>
<td>5.0 References</td>
<td>11</td>
</tr>
<tr>
<td>6.0 Appendix A: Physical Quantities, Symbols, and Units</td>
<td>12</td>
</tr>
<tr>
<td>7.0 Appendix B: RANDOM3 Sample Problem: Source, Input and Output Files</td>
<td>13</td>
</tr>
<tr>
<td>8.0 Appendix C: RANDOM4 Sample Problem Source, Input and Output Files</td>
<td>37</td>
</tr>
<tr>
<td>9.0 Appendix D: IMSL Subroutine Calls from RANDOM3 and RANDOM4</td>
<td>76</td>
</tr>
<tr>
<td>10.0 Appendix E: SAS/GRAPH Program for RANDOM3 and RANDOM4</td>
<td>77</td>
</tr>
</tbody>
</table>
1.0 INTRODUCTION

This User Manual documents the FORTRAN programs RANDOM3 and RANDOM4. They are based on fatigue strength reduction, using a probabilistic constitutive model. They predict the random lifetime of an engine component to reach a given fatigue strength (see Section 2.0, Theoretical Background).

Included in this Manual are details regarding the theoretical backgrounds of RANDOM3 and RANDOM4, input data instructions and sample problems illustrating the use of RANDOM3 and RANDOM4. Appendix A gives information on the physical quantities, their symbols, FORTRAN names and both SI and U.S. Customary units. Appendix B and C include photocopies of the actual computer printout corresponding to the sample problems. Appendices D and E detail the IMSL, Version 10, subroutines and functions called by RANDOM3 and RANDOM4 and SAS/GRAPH programs that can be used to plot both the probability density functions (p.d.f.) and the cumulative distribution functions (c.d.f.).
2.0 THEORETICAL BACKGROUND

Fatigue strength data are usually presented as cycles to failure for each of several stress amplitudes, the familiar S-N diagram. Results indicate that for lower stress amplitudes the cycles (or time) to failure increases. Thus, a power curve fit through the data yields a monotonically decreasing curve. In general, this curve is represented as

\[ S = \left[ \frac{N}{C'} \right]^{-1/m'} \]  

(6)

where the primitive variables in this equation are as follows: \( S \) is the applied constant amplitude alternating stress at failure or fatigue strength, \( N \) is number of cycles, \( C' \) is a material parameter that varies from specimen to specimen and \( m' \) is a material constant.3 Equation (6) can be written in terms of "cycles to reach a given fatigue strength" as

\[ N = C'S^{-m'} \]  

(7)

Recently another fatigue strength reduction model has been proposed that takes into account the effect of temperature as well as other parameters that affect strength.4 The general form of the constitutive relationships for this model is applied to the constituents of high temperature composite materials. Specifically, it is applied herein for the case of a single material constituent. The mechanical property of interest is fatigue strength which is expressed in terms of primitive variables, including the general categories of temperature, mechanical cycles and mean stress. For these categories, the relationship becomes

\[ \frac{S}{S_0} = \left[ \frac{T_F - T}{T_F - T_0} \right]^n \left[ \frac{S_F - \sigma}{S_F - \sigma_0} \right]^m \left[ \frac{\log N_{MF} - \log N_M}{\log N_{MF} - \log N_{MO}} \right]^q \]  

(8)

where \( S \) is the applied constant amplitude alternating stress at failure (fatigue strength) at current (or operating) temperature, \( T \), mean stress, \( \sigma \), and mechanical cycle, \( N_M \). \( S_0 \) is fatigue strength at reference temperature, \( T_0 \) (usually room temperature), reference mean stress (or residual stress), \( \sigma_0 \), and reference mechanical cycle, \( N_{MO} \). Also, \( T_F \) is the final or melting temperature of the material, \( S_F \) is the final or tensile strength of the material, and \( N_{MF} \) is the final mechanical cycle or lifetime. Empirical parameters, \( n \), \( m \), and \( q \), are determined from available experimental data or estimated from anticipated behavior of the particular product term.5 Note that the term containing mechanical cycles is expressed in terms of the log of cycles rather than cycles. This formulation is attractive when \( N_M \) and \( N_{MO} \) are small compared to \( N_{MF} \). The equation may be solved for \( N_M \), or the "cycles to reach a given fatigue strength." The expression is

\[ N = 10 \exp \left[ \log N_{MF} - \left( \log N_{MF} - \log N_{MO} \right) \left[ \frac{S}{S_0} \left[ \frac{T_F - T}{T_F - T_0} \right]^n \left[ \frac{S_F - \sigma}{S_F - \sigma_0} \right]^m \left[ \frac{\log N_{MF} - \log N_M}{\log N_{MF} - \log N_{MO}} \right]^q \right]^{1/q} \right] \]  

(9)
For values typical of a cast nickel base-superalloy subjected to typical loads and temperatures, equation (9) indicates increasing life for decreasing temperature, decreasing tensile mean stress, and decreasing applied alternating stress. It indicates decreasing life for increasing temperature, decreasing compressive mean stress, and increasing applied alternating stress. Therefore, equation (9) predicts observed trends in general.

Probabilistic analysis, via simulation, yields the distribution of the dependent random variable, cycles, \( N \). A probability density function (p.d.f.) of cycles is generated using the maximum penalized likelihood method for RANDOM3. For RANDOM4, a p.d.f. of cycles is generated using the maximum entropy method. Maximum entropy uses Jaynes' principle which says that "the minimally prejudiced distribution is that which maximizes the entropy subjected to the constraints supplied by the given information."\(^6\)
3.0 INPUT DATA

Data input for RANDOM3 and RANDOM4 is user friendly and easy to manipulate (see, for example, the file entitled NORMAL.INP, in Section 4.0). The first twelve lines of input have the same format, 2E12.4 and the last two lines differ. The last two lines of input have the formats I3,2X,I3,2X,2E12.4,2X,I3 and I3, respectively. A brief, line by line description is given along with an example for each line (NOTE: the ruler is to aid the user in formatting and is not a part of the input). A table listing the physical quantities, their units and symbols is given in Appendix A.

1. Random Number Generator Seed, ISEED, and Sample Size, NTOT

EXAMPLE:

```
123456789012345678901234567890
1  40
```

2. Ultimate Tensile Strength, SF

EXAMPLE:

```
123456789012345678901234567890
900.0000 45.0000
```

3. Log of Final Cycle, NMF

EXAMPLE:

```
123456789012345678901234567890
8.0000 0.8000
```

4. Reference Fatigue Strength, SO

EXAMPLE:

```
123456789012345678901234567890
500.0000 25.0000
```

5. Log of Reference Cycle, NMO

EXAMPLE:

```
123456789012345678901234567890
7.0000 0.7000
```
6. Current Fatigue Strength, $S$
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 250.0000 & 12.0000 \\
\end{array}
\]

7. Residual Compressive Stress, SIGO
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 20.0000 & 1.0000 \\
\end{array}
\]

8. Current Mean Stress, SIG
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 150.0000 & 7.5000 \\
\end{array}
\]

9. Temperature Exponent, XXN, Stress Exponent, XXM, and Cycle Exponent, XXQ
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 0.5000 & 0.0150 \\
\end{array}
\]

10. Melting Temperature, TF
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 1500.0000 & 75.0000 \\
\end{array}
\]

11. Reference Temperature, TO
EXAMPLE:

\[
\begin{array}{c|c|c}
123456789012345678901234567890 & 20.0000 & 0.6000 \\
\end{array}
\]
12. Current Temperature, T

EXAMPLE:

\[
\begin{array}{c|c}
123456789012345678901234567890 & 850.0000 \ 25.0000 \\
\end{array}
\]

13. The DESPL\textsuperscript{1} parameters are NODE, INIT, ALPHA, EPS, and MAXIT and are entered in that order as follows:

EXAMPLE:

\[
\begin{array}{c|c|c|c|c}
1234567890123456789012345678901234567890 & 21 & 0 & 20.0000 & 1.0E-05 & 30 \\
\end{array}
\]

14. The DESPL parameter, IOPT, is entered as follows:

EXAMPLE:

\[
\begin{array}{c|c}
1234567890 & 2 \\
\end{array}
\]
4.0 SAMPLE PROBLEMS FOR RANDOM3 AND RANDOM4

The objective of these programs is to predict the random lifetime to reach a given fatigue strength for an engine component. The theory is based on fatigue strength reduction, using a probabilistic constitutive model. The only difference between RANDOM3 and RANDOM4 is the method used to generate p.d.f. estimates. RANDOM3 uses maximum penalized likelihood, while RANDOM4 uses maximum entropy (see Section 2.0, Theoretical Background). RANDOM3 and RANDOM4 input parameters are given in Table A2.1.

TABLE A2.1 RANDOM3 and RANDOM4 input (SI units)

<table>
<thead>
<tr>
<th>FORTRAN Name</th>
<th>Distribution Type</th>
<th>Mean</th>
<th>Standard Deviation (Value)</th>
<th>Deviation (% of Mean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SF</td>
<td>normal</td>
<td>900.0</td>
<td>45.0</td>
<td>(3%)</td>
</tr>
<tr>
<td>NMF</td>
<td>lognormal</td>
<td>8.0</td>
<td>0.8</td>
<td>(10%)</td>
</tr>
<tr>
<td>SO</td>
<td>lognormal</td>
<td>500.0</td>
<td>25.0</td>
<td>(5%)</td>
</tr>
<tr>
<td>NMO</td>
<td>lognormal</td>
<td>7.0</td>
<td>0.7</td>
<td>(10%)</td>
</tr>
<tr>
<td>S</td>
<td>lognormal</td>
<td>250.0</td>
<td>12.5</td>
<td>(5%)</td>
</tr>
<tr>
<td>SIGO</td>
<td>lognormal</td>
<td>-20.0</td>
<td>-1.0</td>
<td>(1%)</td>
</tr>
<tr>
<td>SIG</td>
<td>lognormal</td>
<td>150.0</td>
<td>7.5</td>
<td>(5%)</td>
</tr>
<tr>
<td>XXN</td>
<td>normal</td>
<td>0.5</td>
<td>0.015</td>
<td>(0.3%)</td>
</tr>
<tr>
<td>XXM</td>
<td>normal</td>
<td>0.5</td>
<td>0.015</td>
<td>(0.3%)</td>
</tr>
<tr>
<td>XXQ</td>
<td>normal</td>
<td>0.5</td>
<td>0.015</td>
<td>(0.3%)</td>
</tr>
<tr>
<td>TF</td>
<td>normal</td>
<td>1500.0</td>
<td>45.0</td>
<td>(3%)</td>
</tr>
<tr>
<td>TO</td>
<td>normal</td>
<td>20.0</td>
<td>0.6</td>
<td>(3%)</td>
</tr>
<tr>
<td>T</td>
<td>normal</td>
<td>850.0</td>
<td>25.5</td>
<td>(3%)</td>
</tr>
</tbody>
</table>
The input is entered in the following format in a file entitled NORMAL.INP.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>900.000</td>
<td>45.000</td>
</tr>
<tr>
<td>8.0000</td>
<td>0.8000</td>
</tr>
<tr>
<td>500.000</td>
<td>25.0000</td>
</tr>
<tr>
<td>7.0000</td>
<td>0.7000</td>
</tr>
<tr>
<td>250.000</td>
<td>12.5000</td>
</tr>
<tr>
<td>20.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>150.000</td>
<td>7.5000</td>
</tr>
<tr>
<td>0.5000</td>
<td>0.0150</td>
</tr>
<tr>
<td>1500.000</td>
<td>75.0000</td>
</tr>
<tr>
<td>20.0000</td>
<td>0.6000</td>
</tr>
<tr>
<td>850.0000</td>
<td>25.5000</td>
</tr>
<tr>
<td>21</td>
<td>0.00</td>
</tr>
<tr>
<td>20.00</td>
<td>1.0E-05</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Execution of RANDOM3 and RANDOM4 (source code entitled NR3.FOR and NR4.FOR, respectively) produces files entitled RNAM33 and RANMM44. These give intermediate results (see Appendices B and C). Execution also produces plotfiles entitled PLOT1 and PLOT2 (see Appendices B and C). These files are used to plot the X and Y axes of the probability density function (p.d.f.) and the cumulative distribution function (c.d.f.), respectively, generated by RANDOM3 and RANDOM4. The plots are drawn from the plotfiles by the SAS/GRAPH graphing program (see Appendix D). These plots for the sample problem are shown Figures 1, 2, 3, and 4. This same sample problem has been reported in Boyce and Chamis.7 There, however, it utilized U.S. Customary units and older versions of RANDOM3 and RANDOM4 (using IMSL Version 9.2 subroutines).
Fig. A2.1  p.d.f. of log of mechanical cycles for fatigue strength reduction model, using maximum penalized likelihood method of p.d.f. generation.

Fig. A2.2  c.d.f. of log of mechanical cycles for fatigue strength reduction model, using maximum penalized likelihood method of p.d.f. generation.
Fig. A2.3 p.d.f. of log of mechanical cycles for fatigue strength reduction model, using maximum entropy method of p.d.f. generation.

Fig. A2.4 c.d.f. of log of mechanical cycles for fatigue strength reduction model, using maximum entropy method of p.d.f. generation.
5.0 REFERENCES

1 IMSL, "STAT/LIBRARY, FORTRAN Subroutines for Statistical Analysis", Houston, Texas


6.0 APPENDIX A

PHYSICAL QUANTITIES, SYMBOLS, AND UNITS

The physical quantities, their symbols and units for the fatigue crack growth model are given in the following table.

Table A2.2 Physical quantities, symbols, and units for fatigue crack growth model for RANDOM3 and RANDOM4.

<table>
<thead>
<tr>
<th>Physical Quantity</th>
<th>Theory Symbol</th>
<th>FORTRAN Name</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ultimate Tensile Strength</td>
<td>SF</td>
<td>SF</td>
<td>MPa, ksi</td>
</tr>
<tr>
<td>Final Cycle (lifetime)</td>
<td>N_{MF}</td>
<td>NMF</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Reference Fatigue Strength</td>
<td>S_0</td>
<td>SO</td>
<td>MPa, ksi</td>
</tr>
<tr>
<td>Reference Cycles</td>
<td>N_{MO}</td>
<td>NMO</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Current Fatigue Strengths</td>
<td>S</td>
<td>S</td>
<td>MPa, ksi</td>
</tr>
<tr>
<td>Residual Compressive Stress</td>
<td>\sigma_0</td>
<td>SIGO</td>
<td>MPa, ksi</td>
</tr>
<tr>
<td>Current Mean Stress</td>
<td>\sigma</td>
<td>SIG</td>
<td>MPa, ksi</td>
</tr>
<tr>
<td>Empirical Material Parameters</td>
<td>n</td>
<td>XXN</td>
<td>dimensionless</td>
</tr>
<tr>
<td></td>
<td>m</td>
<td>XXM</td>
<td>dimensionless</td>
</tr>
<tr>
<td></td>
<td>q</td>
<td>XXQ</td>
<td>dimensionless</td>
</tr>
<tr>
<td>Melting Temperature</td>
<td>TF</td>
<td>TF</td>
<td>°C, °F</td>
</tr>
<tr>
<td>Reference Temperature</td>
<td>TO</td>
<td>TO</td>
<td>°C, °F</td>
</tr>
<tr>
<td>Current Temperature</td>
<td>T</td>
<td>T</td>
<td>°C, °F</td>
</tr>
</tbody>
</table>
7.0 APPENDIX B

RANDOM3 SAMPLE PROBLEM: SOURCE, INPUT AND OUTPUT FILES
CALL RNLM(1NTOT,%INTOT,30).
WRITE(*,2022).
2022 FORMAT(1X,'LOGNORMAL 3D')
WRITE(6,1001)(900(1),I=1,NTOT)
C LOGNORMAL LOG OF REFERENCE CYCLES, XLMO
WRITE(6,1002)ISEED,NTOT,
READ(3,1011)XN,XS
WRITE(6,1011)XM, XS
YS = 30RT(LOG(1.0+(XS/XM)**2))
YM = LOG(XM) - 0.5*YS**2
CALL RNSET(ISEED)
CALL RNLM(NTOT,IM,YS,XLMO)
WRITE(6,2023)
2023 FORMAT(1X,'LOGNORMAL XLMO')
WRITE(6,1001)(XLMO(I),I=1,NTOT)
C LOGNORMAL FATIGUE STRENGTH AT CURRENT CONDITIONS, S
WRITE(6,1002)ISEED,NTOT.
READ(3,1011)XM, XS
WRITE(6,1011)XM, XS
YS = 30RT(LOG(1.0+(XS/XM)**2))
YM = LOG(XM) - 0.5*YS**2.
CALL RNSET(ISEED)
CALL RNLM(NTOT,IM,YS,S).
WRITE(6,2024)
2024 FORMAT(1X,'LOGNORMAL S')
WRITE(6,1001)(S(I),I=1,NTOT)
C DEFINE RANDOM STRESSES
C LOGNORMAL REFERENCE STRESS, SIGO
WRITE(6,1002)ISEED,NTOT.
READ(3,1011)XM, XS
WRITE(6,1011)XM, XS
YS = 30RT(LOG(1.0+(XS/XM)**2))
YM = LOG(XM) - 0.5*YS**2.
CALL RNSET(ISEED)
CALL RNLM(NTOT,IM,YS,SIGO).
C CHANGE SIGO TO NEGATIVE VALUES FOR COMPRESSIVE
C RESIDUAL STRESSES
DO 201 I = 1,NTOT
SIGO(I) = -SIGO(I)
201 CONTINUE
WRITE(6,2036)
2036 FORMAT(1X,'LOGNORMAL SIGO')
WRITE(6,1001)(SIGO(I),I=1,NTOT)
C LOGNORMAL CURRENT STRESS, SIG
WRITE(6,1002)ISEED,NTOT.
READ(3,1011)YM,YS
WRITE(6,1011)YM,YS
YS = 30RT(LOG(1.0+(XS/XM)**2))
YM = LOG(XM) - 0.5*YS**2.
CALL RNSET(ISEED)
CALL RNLM(NTOT,IM,YS,SIG)
WRITE(6,2037)
2037 FORMAT(1X,'LOGNORMAL SIG')
WRITE(6,1002)ISEED,NTOT.
READ(3,1011)YM,YS
WRITE(6,1011)YM,YS
CALL RNSET(ISEED)
CALL RNNM(NTOT,XXS)
DO 202 I = 1,NTOT
XXS(I) = SIG(I)**3+YM
202 CONTINUE
WRITE(6,2025)
**CONTINUE**

**WRITE**(6,2028)

**FORMAT**(' LOG OF CYCLES TO REACH MEAN FATIGUE STR * * **

'**WRITE**(10,1001)(XNM(I),I=1,NTOT)

**C SORT LOG OF CYCLES**

**CALL** SORT(XNM,NTOT)

**WRITE**(6,2029)

**FORMAT**(' SORTED LOG OF CYCLES **

**WRITE**(6,1001)(XNM(I),I=1,NTOT)

**C CALCULATE PDF OF LOG OF CURRENT CYCLES LOG XNM**

**READ**(3,1009)(NODE,INIT,ALPHA,EPS,MAL)(

**WRITE**(6,985)

**FORMAT**(' DEPL PARAMETERS **

**WRITE**(6,1009)(NODE,INIT,ALPHA,EPS,MAL)(

**CALL** DEPL(NODE,NTOT)

**WRITE**(6,986)

**FORMAT**(' BDNS(1),BDNS(2) E12.4,1X,E12.4(2)

**CALL** DEPL(NTOT,XNM,NODE,BDNS,INIT,ALPHA,EPS,MAL)(

**WRITE**(6,990)

**FORMAT**(' PDF OF LOG OF CURRENT CYCLES LOG XNM, Y AXIS OF PDF PLOT **

**WRITE**(6,1001)(DEN(I),I=1,NODE)

**WRITE**(6,981)

**FORMAT**(' OUTPUT STATISTICS **

**WRITE**(6,1001)(STAT(I),I=1,4)

**WRITE**(6,992)

**FORMAT**(' NUMBER OF MISSING VALUES **

**WRITE**(A,1010)NMISS

**C CALCULATE WINDOW WIDTH, HH**

HH = (BDNS(2)-BDNS(1))/(NODE-1)

**C CALCULATE VALUES OF LOG OF CURRENT CYCLES AT WHICH PDF IS ESTIMATED ALSO CALLED 'NODE' VALUES**

**DO** 6001: I=1,NODE-2

BDNS(I+2)=BDNS(I)+ (IMHH)

**CONTINUE**

**WRITE**(6,983)

**FORMAT**(' LOG OF CURRENT CYCLES LOG XNM **

**WRITE**(6,1001)(BDNS(I),I=1,NODE)

**C REORDER BDNS FOR PLOTTING**

SAVE1 = BDNS(2)

SAVE2 = BDNS(NODE)

BDNS(NODE)=BDNS(1)

**DO** 6002: I=1,NODE-2

BDNS(I+1)=BDNS(I+2)

**CONTINUE**

**BDNS(NODE)=SAVE2**

**WRITE**(6,984)

**FORMAT**(' ORDERED LOG OF CURRENT CYCLES LOG XNM

**WRITE**(6,1001)(BDNS(I),I=1,NODE)

**C WRITE LOG OF CURRENT CYCLES AND PDF OF LOG OF CURRENT CYCLES**

**C LOG XNM TO PLOT FILES**

**WRITE**(A,990)

**FORMAT**(' (E12.4,1X,E12.4')
**Calculating CDF of Log of Current Cycles**

```fortran
C
C CALL DISPL (NOBS, X, NODE, BNDS, L, ALPHA, MAXIT, EPS,
DENS, STAT, HESS, LHESS, ILOHI, DENSEST, B,
IPVT, WKZ)
C
C Arguments:
C
NOBS - Number of observations, (Input)
```

- **Purpose:** Nonparametric probability density function estimation by the penalized likelihood method.
- **Usage:** CALL DISPL (NOBS, X, NODE, BNDS, L, ALPHA, MAXIT, EPS,
DENS, STAT, HESS, LHESS, ILOHI, DENSEST, B,
IPVT, WKZ)

**Computer:** IBM/SINGLE

**Revised:** November 1, 1990

**Language:** FORTRAN

**Library:** DISPL/DDISPL (Single/Double precision version)
SUBROUTINE DJSPL (NOBS, N, NODE, BNDS, INIT, ALPHA, MAXIT, EPS, DENS, STAT, HESS, LDHESS, ILOHI, DESEN, B, IPVT, WK2)

INTEGER NOBS, NODE, INIT, ALPHA, MAXIT, EPS, DENS, STAT, HESS, LDHESS, ILOHI, DESEN, B, IPVT, WK2

REAL ALPH, EPS, X(N), BNDS(2), DENS(4), STAT(4),
      HESS(LDHESS,4), DESEN(4), B(N), WK2(N)

INTEGER I, INTR, IPTR, ITER, K, K1, K2, KPI, KP2, M, NDL, NER, NOBL,
      BK, BK1, BSMALL, CK, CK1, CKM, CKMC1, CKF1, CKF2,
      CONS, EPS1, FACTOR, FK, FKM1, FKN2, FKP1, H, H2, H3,
      TN, TP, WK(4)

DOUBLE PRECISION SUM1, SUM2, SUM3

INTEGER MINCR(8)
SAVE MINCR

SPECIFICATIONS FOR ARGUMENTS

SPECIFICATIONS FOR LOCAL VARIABLES

SPECIFICATIONS FOR LOCAL VARIABLES

SPECIFICATIONS FOR SAVE VARIABLES

SPECIFICATIONS FOR INTRINSICS

intrinsics

SPECIFICATIONS FOR INTRINSICS

Copyright: 1985 by IML, Inc. All Rights Reserved.

Warranty: IML warrants only that IML testing has been applied to this code. No other warranty, expressed or implied, is applicable.
EXTERNAL EIMES, EIPOP, EIIPH, EISTI, EISTR, SADD, SAXPY, SCOPY, SIMDPROD, SSCALE, D33PT, LTRB, L3PRB

EXTERNAL ISMIN, MRCD, SDQT, SNRM2, SSUM

DATA MINCA/5, 9, 17, 33, 65, 129, 233, 100001/

CALL EIIPH ('D33PL') Error: checks

IF (MD1S .LT. 1) THEN
  CALL EIMES (5, 1) 'After removing all missing (NaN) not g
     'number) values from X there are no valid /4
     'observations. At least one valid observation
     'is necessary.'
END IF

IF (NODE .LE. 4) THEN
  CALL EISTI (1, NODE)
  CALL EIMES (5, 2) 'NODE = %I(I). The number of mesh /4
     'nodes, NODE, must be an odd integer greater /5
     'than 3.
ELSE IF (MOD(NODE-2) EQ. 0) THEN
  CALL EISTI (1, NODE)
  CALL EIMES (5, 3) 'NODE = %I(I) must be an odd integer /4
     'greater than 3.
END IF

IF (ALPHA .LE. 0.0) THEN
  CALL EISTR (1, ALPHA)
  CALL EIMES (5, 3) 'ALPHA = %R(I). The penalty weighting /4
     'factor which controls smoothness, ALPHA, must /5
     'be greater than 0.'
END IF

IF (MAXIT .LE. 0.0) THEN
  CALL EISTI (1, MAXIT)
  CALL EIMES (5, 3) 'MAXIT = %I(I). The maximum number /4
     'of iterations, MAXIT, must be greater than 0.'
END IF

IF (BANDS(1) .GT. BANDS(2)) THEN
  CALL EISTR (3, BANDS(1))
  CALL EISTR (2, BANDS(2))
  CALL EIMES (5, 6) 'BANDS(1) = %R(I) and BANDS(2) = /4
     'X(2). The minimum value for X, BANDS(1), must /5
     'be less than or equal to the maximum value for /4
     'X, BANDS(2).'
END IF

IF (INIT .NE. 0) THEN
  IF (BANDS(NODE) .OR. BANDS(NODE), NE. 0) THEN
    CALL EISTR (1, BANDS(NODE))
    CALL EISTR (2, BANDS(NODE))
    CALL EISTI (1, NODE)
    CALL EIMES (3, 5) 'BANDS(1) = %R(I) 'because /5
      'the initial estimates of the 'density, BANDS, must be greater than or /5
      'equal to 0.'
END IF
40 L01 = 0
00 10 
IF (X(I).LT.BNDS(1) .OR. X(I).GT.BNDS(2)) THEN
L01 = L01 + 1
END IF
10 CONTINUE
IF (L01.EQ. NOBS) THEN
CALL EIMES (S, 9, 'All elements in X lie outside the 
interval BNDS(1)-to-BNDS(2) At least one 
element of X must lie in this interval.' )
END IF
ELSE
EPSL = EPS
END IF
C IF (MNCD(0).NE. 0) GO TO 2000
C IMPTR = 0
C IF (INIT .EQ. 0) THEN
DENS(1) = 0.0
DENS(2) = 2.0/(BNDS(2)-BNDS(1))
M = 3
ELSE
M = NODE
END IF
20 IF (INIT .EQ. 0) THEN
MOLD = M
IMPTR = IMPTR + 1
M = MIN(MODE, MINCR(IMPTR))
END IF
C H2 = (BNDS(2)-BNDS(1))/(M-1)
C H3 = H2*M
C IF (INIT .NE. 0) THEN
CALL SCALE (M, 1.0/(MSSUM(MODE,DENS(1))), DENS(1))
END IF
C B(1) = BNDS(1)
DO 30 L=2, M
B(L) = B(L-1) + H
30 CONTINUE
C IMPTR = IMPTR + 1
DO 40 I=1, K
40 IF (X(I).LT. BNDS(1)) GO TO 40
DO 46 K=1, M - 1
ILOM(K+1) = IMPTR
ILOM(K+2) = IMPTR + 1
IF (IPTN .LE. NOBS) THEN
IF (X(I).LT. B(K+1)) THEN
IF (IPTN .LE. NOBS) GO TO 50
IPTN = IPTN + 1
END IF
END IF
END IF
50 CONTINUE
DO 90 I=2, M - 1
  DENS(1) = AMAX1(TEMP, SQRT(10000(1+1)))
  CONTINUE
END IF

DO 140, ITER=1, MAXIT
  MAXIMIZE
  HESS(1,1) = 0.0
  HESS(1,2) = 0.0
  HESS(2,1) = 0.0
  BSMALL = 0.0
  SUM = 0.0
  CK** are true estimates = FK**2

DO 120 K=2, M - 1
  KM1 = K - 1
  K1 = KM1 + 1
  FN = DENS(K1)
  FM1 = DENS(KM1)
  FM2 = DENS(KM2)
  CKM1 = FK**2
  CK = FK**2
  CKP1 = DENS(KP1)**2
  CKP2 = DENS(KP2)**2
  BK = B(K)
  BKM1 = B(KM1)
  SUM = SUM + CK
  IF (M <= 4) HESS(1, KM1) = 4.0*FWM2*FACTOR
  SUM1 = 0.0D0
  SUM2 = 0.0D0
  SUM3 = 0.0D0
  DO 100 I=ILOMI(K-1), ILOHI(K-2)
    TEMP = ((K-1)-BK)/H
    CONS = (1.0-TEMP)/(CK+CKP1-CK)*TEMP
    SUM1 = SUM1 + CONS
    SUM2 = SUM2 + CONS*CONS
    SUM3 = SUM3 + CONS*(1.0-CONS)/TEMP
  CONTINUE
  DO 110 I=ILOMI(KM1, KMP1)
    TEMP = CONS*CONS
    SUM1 = SUM1 + CONS
    SUM2 = SUM2 + CONS*CONS
    SUM3 = SUM3 + CONS*(1.0-CONS)/TEMP
  CONTINUE
  TEMP = FACTORS*(CKM2+CKP2-4.0*(CKM1+CKP1)+6.0*CK)+SUM1
  TEMP = 2.0*TEMP
  BSMALL = BSMALL + 2.0*CK*TEMP


```
HES(KM,1) = TEMP + 0.0*FK*KMKH + FACT*SUM2
IF (KX .NE. 1) HES(KM,1) = 0.0*FK*KMKH + FACT*SUM2
DEJEST(KM,1) = FK*TEMP
DEJEST(KM,2) = -2.0*FK
120 CONTINUE
BSMALL = 1.0/H - SUM + BSMALL
CALL SCOPY (M-2, DEJEST(1:2), 1, DEJEST(1:2), 1)
CALL SADD (M-2, -BSMALL/(2.0*SUM), HESS(3:1), LDHESs)
CALL SCOPY (M-3, HESS(1:3), LDHESs, HESS(3:1), LDHESs)
HESS(3:M-3) = 0.0
HESS(3:M-2) = 0.0
HESS(4:M-2) = 0.0
CALL LTTRI (M-2, HESS, LDHESs, 2, 2, HESS, LHESS, IPUT, WC)
CALL LISSP (M-2, HESS, LDHESs, 2, 2, HESS, LDHESs, IPUT, DEJEST, 1, DEJEST)
8 IF (MISS(1:10) .NE. 0) GO TO 9000
CONS = SDOT(M-2, DEJEST(1:2), 1, DEJEST(1:2), 1)/CONS
CONS = (1.0/H*SUM - SDOT(M-2, DEJEST(1:2), 1, DEJEST(1:2), 1))/CONS
CALL SASSP (M-2, 1.0, 1.0, DEJEST(1:2), 1)
CALL SASSP (M-2, -1.0, 1.0, DEJEST(1:2), 1, DENS(2), 1)
TEST = SNRM2(M-2, DENS(2), 1)
IF (SNRM2(M-2, DEJEST(1:2), 1, EPSI*TEST) .GT. 150)
DO 50 J = 1, M - 1, 1
50 DENS(J) = AMAX1(TEMP, DENS(1))
130 CONTINUE
140 CONTINUE
CALL ELATE (1, MAXIT)
CALL EMES (3, 1, 'The maximum number of iterations')
IF ('MAXIT' .LT. MAXIT) THEN
'('MAXIT' .LT. MAXIT) was exceeded.'
50 Parameter updates
SUM2 = 0.0
IF (M .NE. NOE) GO TO 20
SUM1 = 0.0
DO 160 K = 1, M
160 KMKH = MAX0(K-1, 1)
KM1 = MIN0(K+1, M)
SUM1 = SUM1 + (DENS(KM1) - 2.0*DENS(K1)+DENS(K1))**2
160 CONTINUE
STAT(2) = -0.5*FACTORSUM1
DO 170 I = 1, NOE
170 HES(I,1) = HES(I,1) + SUM2
DO 200 J = 1, NOE
200 HES(I,1) = SUM2 + ALOG(DEJEST(I))
END IF
170 CONTINUE
STAT(1) = SUM2
```
SUM1 = 0.0
SUM2 = 0.0
DO 130 K=1, M - 1
   FK = DENS(K)
   FKP1 = DENS(K+1)
   BK = B(K)
   CONS = FK + FKP1
   TEMP = CONS + FKP1
   SUM1 = SUM1 + M2*TEMP/6.0 + 0.5*H*BK*CONS
   SUM2 = SUM2 + M2*(TEMP*FKP1)/12.0 + M2*BK*CONS/3.0
   & - 0.5*H*BK*CONS
130 CONTINUE
   STAT(3) = SUM1
   STAT(4) = SUM2 - SUM1*SUM1
C 3000 CALL E1PO1 ('DISPL')
   Exit section
C 3000 CALL E1PO1 ('DISPL')
   Exit section
RETURN
END
<table>
<thead>
<tr>
<th>Cycles to Reach Mean Fatigue Str.</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.45E+01</td>
</tr>
</tbody>
</table>

**Sorted Log of Cycles**

<table>
<thead>
<tr>
<th>Sorted Log of Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.02E+01</td>
</tr>
</tbody>
</table>

**Despl Parameters**

- PDF Loop
- PDF of Log of Current Cycles, 59.95E+00
- X Axis of PDF Plot
- Log of Current Cycles, 1.30E+02
- 0.670E+01
- 0.595E+00

**Output Statistics**

- 0.45E+02
- Number of Missing Values
- 0.13E+02
- 0.760E+01
- 0.595E+00
File _DBAO: C:\PLOTZ.CFR;1 (323.178.0), last revised on 23-NOV-1986 11:26, is a 2 block sequential file owned by UIC [11.111]. The records are variable length with FORTRAN (FTN) carriage control. The longest record is 35 bytes.

8.0 APPENDIX C

RANDOM4 SAMPLE PROBLEM: SOURCE, INPUT AND OUTPUTFILES
C NORMAL REFERENCE TEMPERATURE TO
WRITE(*,1005)T,SEED,NTOT
READ(*,1006)YM,TS
YT=2.0
CALL RNOR(NTOT)
DO 406 I=1,NTOT
406 CONTINUE
WRITE(*,2047)
C NORMAL CURRENT TEMPERATURE TO
WRITE(*,1001)I,NTOT
C CURRENT LOG.OF CYCLES, LOG.XNM.
DO 102 I=1,NTOT
RS=(SF(I)-SIG(I))/(SF(I)-SIOO(I)))**XXM(I)
WRITE(*,6876)RS
C TEMP=((TF(I)-TFQ))**XXN(I)
TEMP=((TF(I)-TFQ))**XXN(I)
WRITE(*,7876)TEMP
C FORMAT(I,TEMP=,'E12.4)
S=SI(I)
SS=SO(I)
XXQ=XQ(I)
WRITE(*,1001)SS
WRITE(*,1001)XXQ
XNM1=(6(I))/((SO(I)*XXM))**((1./XXQ(I))
WRITE(*,8876)XNM1
C FORMAT(XNM1=,'E12.4)
XNM2=(XNM1)/((XNM1-XNM2(I))**XNM1)
WRITE(*,8875)XNM2
C FORMAT(XNM2=,'E12.4)
IF(XNM2.LT.0.0)XNM2=0.0
XNM(I)=XNM2
XNM(I)=10.**XNM2
102 CONTINUE
WRITE(*,2028)(XNM(I)),I=1,NTOT
WRITE(*,2028)
C Sort LOG.OF CYCLES
CALL SORT(XNM,NTOT)
WRITE(*,2029)XNM(I),I=1,NTOT
WRITE(*,2029)
C CALCULATE PDF OF LOG OF CURRENT CYCLES, LOG XNM.
D CALL SMOM(XNM,HMM,NTOT,SM)
D WRITE(39,1001)(SM(I),I=1,HMM)
D WRITE(6,1038)

C CALL SAMPLE MOMENTS
D WRITE(6,1001)(SM(I),I=1,HMM)
C OBTAIN MAXIMUM ENTROPY DISTRIBUTION
D START=1
D KDATA=1
C CALCULATE MAX AND MIN ORDINATES FOR PDF (AND CDF)
D BNDI(I)=XNM(I) - 0.05*XNM(I)
D BNSD(2)=XNM(NTOT) + 0.05*XNM(NTOT)
D WRITE(6,9977) BNDI(1),BNDI(2)
D WRITE(6,9977) BNSD(1),BNSD(2)
D CALL MEP1(HMM+SM,BNDI(1),BNDI(2),0,XP*,START,KDATA=AL-CUM)
D WRITE(31,1001)(AL(I),I=1,HMM+1)
D WRITE(6,1039)

C CALCULATE LAGRANGIAN MULTIPLIERS
D WRITE(6,1001)(AL(I),I=1,HMM+1)
C CALCULATE VALUES OF ORDINATES FOR PDF (AND CDF)
D NODE=2
D HNM=BNSD(2)-BNDI(1)/(NDOE-1)
C CALCULATE VALUES OF LOG OF CURRENT CYCLES AT WHICH PDF IS ESTIMATED.
D ALSO CALLED 'NODE' VALUES
D DO 6001 I=1,NODE-2
D BNSD(I+2)=BNDI(1)+(I*HNM)
D 6001 CONTINUE
D WRITE(6,9983)

C REORDER BND FOR PLOTTING
C
D SAVE1 = BNSD(2)
D SAVE2 = BNSD(NODE)
D BNSD(NODE)=BNSD(2)
D 6002 CONTINUE
D BNDI(I)=BNSD(I+2)
D BNDI(I)=SAVE1
D BNSD(NODE)=SAVE2
D NDOE=984
D CALL ORDERD LOG OF CURRENT CYCLES, LOG XNM
D WRITE(6,9984)(BNSD(I),I=1,NODE)

C CALCULATE VALUES OF THE PDF AT EACH ORDINATE
C FOR 4 MOMENTS. THERE ARE 3 LAGRANGIAN MULTIPLIERS
D BNDM(1)=AL(1)+AL(2)*BNDI(I)+AL(3)*BNDI(I)**2
D BNDM(1)+BNDI(I)**3+AL(5)*BNDI(I)**4
C WRITE LOG OF CURRENT CYCLES AND PDF OF LOG OF CURRENT CYCLES;
C - LOG-XMN TO PLOT FILES
WRITE(*,990)
990 FORMAT(1X,E12.4,1X,E12.4)
WRITE(*,991)(BNDS(J),DEN'S(J),J=1,NODE)
991 FORMAT(E12.4,1X,E12.4)
C CALCULATE CDF OF LOG OF CURRENT CYCLES
IOPT=2
C READ(3,1004)IOPT
WRITE(*,992)
992 FORMAT('3CDF PARAMETERS')
WRITE(*,1004)IOPT
X0=BNDS(1)
DO 993 IOPT=1,NODE
P=CDFX(X0,IOPT,NODE,BNDS,DENS)
BNDSX(I)=X0
X0=X0+MM
DISTX(I)=P
993 CONTINUE
WRITE(*,994)
994 FORMAT('CDF OF LOG OF CURRENT CYCLES, LOG XMN, 1Y AXIS OF PDF, CDF PLOT')
WRITE(*,1001)(DISTX(I),I=1,NODE)
C WRITE(4,993)
993 FORMAT('ORDERED LOG-OF-CURRENT-CYCLES, LOG XMN, 1X AXIS OF PDF, CDF PLOT')
WRITE(4,1001)(BNDS(I),I=1,NODE)
WRITE(4,1001)(BNDSX(I),I=1,NODE)
C WRITE LOG OF CURRENT CYCLES AND CDF OF LOG OF CURRENT
TO THE PLOT FILES
C WRITE(35,990)
WRITE(35,991)(BNDS(J),DISTX(J),J=1,NODE)
STOP
C SUBROUTINE SORT(Y,N)
DIMENSION Y(10000)
C Y IS THE ARRAY TO BE SORTED
C AT COMPLETION Y(1) IS SMALLEST VALUE
C AT COMPLETION Y(N) IS LARGEST VALUE
M1=N-1
DO 1,J=2,N
1 I = J-1
DO 2 K=J,N
IF (Y(K).LT.Y(I)) GO TO 2
2 CONTINUE
CONTINUE
RETURN
C SUBROUTINE SMOM(X,H,MSAMP,SM)
C CALCULATES SAMPLE CENTRAL MOMENTS
C X(I) = SAMPLE VALUES, DIMENSION HSAMP
C H = NUMBER OF MOMENTS DESIRED
C MSAMP = SAMPLE SIZE
C SM = VALUE OF MOMENTS, DIMENSION M
DIMENSION X(10000),SM(10)
C  CALCULATE MEAN
   SUM=0.0
   DO 1 I=1,NSAMP
     1 SUM=SUM+X(I)
     SM(I)=SUM/NSAMP
   IF(MLT.3)RETURN
   C  CALCULATE VARIANCE
   SUM=0.0
   DO 2 I=1,NSAMP
     SUM=SUM+(X(I)-SM(I))**2
   2 IF(MLT.3)RETURN
   C  CALCULATE HIGHER MOMENTS
   DO 3 J=1,NSAMP
     SUM=SUM/(J**(NSAMP-1))
     SM(I)=SUM/NSAMP
   3 CONTINUE
   RETURN
END

SUBROUTINE MEPI(N;CM;XMIN;XMAX;NXP;XP,KSTART,KDATA;AL;CUM)
IMPLICIT REAL*8(A-H,O-Z)

.... EXECUTIVE PROGRAM FOR USING MAXIMUM ENTROPY METHOD CONstrained BY
MOMENTS TO GENERATE A DENSITY FUNCTION

DIMENSION AL(N),CM(N),ETA(4),XP(N),CUM(N),C(3),APC(10)
COMMON /FAIL/,NFAIL
COMMON/HHELP/;X(16),C(8),M

... Line different from text
COMMON/MEPI,KPRINT,TO,MAXF}
DATA KPRINT,TO,MAXF/0,0,0.0,0.0,1.0/1,E-0.001
1.0=10

WRITE THE INPUT DATA
IF(KDATA.EQ.0)GO TO 1
WRITE(6,25)KDATA
WRITE(6,26)KPRINT
WRITE(6,27)M
WRITE(6,30)CM
WRITE(6,31)CM(I),I=1,N
IF(6.661)WRITE(6,32)(CM(I),I=5,N)
IF(MLT.3)WRITE(6,33)NXP
CONTINUE
1

MFAIL=0
M=11
X2MIN=0.0
X2MAX=1.0
SAUC=1.0
DO 100 I=1,N
CC(I)=CM(I)
100 CONTINUE

C  CALCULATE THE MOMENTS AT THE MODIFIED LIMITS
CALL TRNI(XMAX,XMIN,CC,X2MAX,X2MIN,N)

C  CALCULATE THE MOMENTS ABOUT THE ORIGIN FOR THE MODIFIED LIMITS

C  STORE THEM IN COMMON IN C
CALL CONVER(CC,N)

GENERATE THE SIMPSON MULTIPLIERS AND STORE THEM IN HELP COMMON

CALL SIMSON

GENERATE THE X,S POWER FOR SUBROUTINE FC, STORE THEM IN HELP COMMON ARRAY

CALL MULTI-(X2MAX+X2MIN+N)

DEFINE THE INPUT DATA FOR SUBROUTINE MPOPT

ETA(1)=1.0-12
ETA(2)=0
ETA(3)=1.0-24
ETA(4)=1.0-24
MODE=1
UMIN=0.0

WRITE THE INTERMEDIATE RESULTS YOU HAVE OBTAINED SO FAR

IF (KPRINT,EO.0) GO TO 2
WRITE (6,34)
WRITE (6,36) X2MAX+X2MIN
WRITE (6,37) (CC(I),I=1,4)
IF (N.GT.4) WRITE (6,22) (CC(I),I=5,N)
WRITE (6,38) (CC(I),I=1,4)
IF (N.GT.4) WRITE (6,22) (CC(I),I=5,N)
WRITE (6,39) (ETA(I),I=1,4)
CONTINUE

FIND A STARTING POINT FOR SUBROUTINE MPOPT TO START THE OPTIMIZATION ALGORITHM

IF (KSTART,EO.0) GO TO 16
IF (KSTART,EO.4) WRITE (6,44).
CALL START (X2MAX+X2MIN,AL,KSTART,CC,N,KPRINT,UMIN,MODE,MAXF,ETA)
IF (NFLA,EO.1) GO TO 9

PRINT THE STARTING VALUES

IF (KPRINT,EO.0) GO TO 7
GO TO (3,4,5,6), KSTART
WRITE (6,40)
WRITE (6,41) (AL(I),I=1,4)
IF (N.GT.4) WRITE (6,22) (AL(I),I=5,N)
WRITE (6,42)
WRITE (6,41) (AL(I),I=1,4)
IF (N.GT.4) WRITE (6,22) (AL(I),I=5,N)
GO TO 7
WRITE (6,43)
WRITE (6,41) (AL(I),I=1,4)
IF (N.GT.4) WRITE (6,22) (AL(I),I=5,N)
WRITE (6,44)
CONTINUE

RANGE=XMAX-XMIN

CHANGE STARTING VALUES TO 0-1 DOMAIN FOR KSTART=0
C. THE ALGORITHM IS SIMILAR TO TAYLOR'S APPROACH TO GIVE BETTER
NUMERICAL RESULTS
NPL = N+1
IF (ABS(XMIN) .LT. 1.E-10) GO TO 10
DO 19 J=2,NPL
ALS(I) = 0.0
DO 18 J=I,N
ALS(I) = ALS(I) + FACTO(J) * XMIN** (J-I1) * RANGE** I1 * AL(J1) / FACTO(I1)
/ FACTO(J11)
18 CONTINUE
17 CONTINUE
GO TO 50
19 DO 20 I = 2, NPL
20 ALS(I) = RANGE** (I-1) * AL(I)
C... PUT AL(I) IN PROPER LOCATIONS
50 DO 51 I = 1, N
51 AL(I) = ALS(I+1)
7 CONTINUE
8 IF (KPRINT .EQ. 0) GO TO 9
WRITE (6,45)
CONTINUE
AL(N+1) = 2.
AL(N+2) = 0.0
CALL MOPRI (AL, N, ETA, XMIN, MAXFN, MODE, KPRINT)
2 IF (NFAIL .EQ. 0) GO TO 10
IF (KSTART .EQ. 1.0) GO TO 9
THE PROGRAM HAS FAILED SO FAR. TRY ANOTHER STARTING POINT AND TRY
AGAIN
KSTART = KSTART + 1
IF (KSTART .EQ. 1. AND N .LE. 2) GO TO 9
GO TO 2
CONTINUE
WRITE (6,46)
CALL EXIT
CONTINUE
10 CALCULATE THE ZEROTH LAGRANGIAN MULTIPLIER
SUM = 0.0
11 DO 12 I = 1, N
SUM = SUM + AL(K) * XX(K, I)
12 CONTINUE
SUM = SUM + 9(I) * EXP (SZ)
13 NPL = N+1
DO 14 I = 1, N
14 SUM = SUM + AL(K-I)
13 CONTINUE
DELTA = (XMAX - XMIN) / FLOAT(N-1)
AL(K) = ALOG(SUM/DELTA/3.0)
WRITE (6,101) SUM
101 FORMAT (24H SUM OF RESIDUALS SQUARED , E12.5)
IF (KPRINT .EQ. 0) GO TO 14
WRITE (6,47) (AL(I), I = 1, NPL)
CONTINUE
C... RESET KSTART TO ZERO
CALCULATE THE LAGRANGIAN MULTIPLIERS FOR THE ORIGINAL LIMITS
CALL TRN2 (XMAX,XMIN,AL,X2MAX,X2MIN,N)

CALCULATE THE CUMULATIVE DISTRIBUTION FUNCTION VALUE AT THE GIVEN
POINT
IF(NXP.EQ.0)RETURN
DO 1=1,NXP
CUMUALI CDF(XMIN,XMAX,XP(I)-AL,NPL)
1 CONTINUE
RETURN

FORMAT (5X,4E18.9,/)  
FORMAT (5X,4E18.9,/)  
FORMAT (1X,/,20X,INPUT DATA FOR SUBROUTINE MEG/+/20X,33(/')  
25 FORMAT (/'INPUT DATA IS PRINTED OUT FOR KDATA =1 ONLY')  
26 FORMAT (/'INTERMEDIATE OUTPUT EVERY KPRINT(TH) CYCLE')  
28 FORMAT (/'NUMBER OF KNOWN FIRST MOMENTS')  
29 FORMAT (/'HIGHER LIMIT')  
30 FORMAT (/'LOWER LIMIT')  
31 FORMAT (/'FIRST MOMENTS')  
32 FORMAT (/'THE ALLOWED TOLERANCE IN LAGRANGIAN EQUATIONS')  
33 FORMAT (/'THE CUMULATIVE DISTRIBUTION REQUIRED AT NXP POINTS, NXP =')  
34 FORMAT (1X,/,20X,'INTERMEDIATE RESULTS FOR SUBROUTINE MEG/+/2  
35 FORMAT (1X,/,20X,'NUMBER OF INTEGRATION STATION')  
36 FORMAT (1X,/,20X,'MODIFIED MAXIMUM AND MINIMUM LIMITS')  
37 FORMAT (1X,/,20X,'MODIFIED MOMENTS ABOUT THE EXPECTED VALUE')  
38 FORMAT (1X,/,20X,'MODIFIED MOMENTS ABOUT THE ORIGIN')  
39 FORMAT (1X,/,20X,'SUBROUTINE MPOMT TOLERANCES')  
40 FORMAT (1X,/,20X,'NORMAL ASSUMPTION STARTING METHOD')  
41 FORMAT (1X,/,20X,'STARTING VALUES')  
42 FORMAT (1X,/,20X,'UNIFORM ASSUMPTION STARTING METHOD')  
43 FORMAT (1X,/,20X,'M POINTS STARTING METHOD')  
44 FORMAT (1X,/,20X,'STEP BY STEP STARTING METHOD')  
45 FORMAT (1X,/,20X,'CYC NUMF NORMGRAD TOTAL',24X,'VARIABLES',40  
46 FORMAT (1X,/,20X,'RESIDUALS',NO.,10X,'RESIDUALS',TOTAL,24X,'VARIABLES',40  
47 FORMAT (1X,/,20X,'THE PROGRAM HAS FAILED')  
48 WRITE(6,44)
49 FORMAT(1X,THE MODIFIED LAGRANGIAN MULTIPLIERS ARE)  
50 FORMAT(1X,THE MEAN IS NEARLY ZERO AND MEG1 WILL NOT WORK/12  
51 IN TRANSFORM X)
SUBROUTINE MPOPT (X,NDIM,ETA,EST,MAX,MODE,IPRINT)

IMPLICIT REAL*8 (A-H,O-Z)
REAL*8 KTB,IPRINT
COMMON /FAIL, NFAIL
DIMENSION X(*), GI(10), G1(10), G2(10), ALFA(10), H(10),
Y(10), P(10), PE(10), ETA(*), B1BO(10), RR(8)
EXTERNAL FUNCT
KRFST=0
IFLAG=0
M=0
N=NDIM+1
N1=NDIM+2
NUMF=0
IER=0
DO 1 I=1,N1
X(I)=X(I)
1 CONTINUE
CALL FUNCT (NDIM,X1,F1,G1,RR)
NUMF=NUMF+1
DO 2 I=1,NDIM
X2(I)=X(I)
2 CONTINUE
G2(I)=G1(I)
M(I)=G1(I)
3 CONTINUE
F2=F1
X2(M2)=X1(M2)
X2(N1)=X1(N1)
4 CALL EPS=ETA(4)
5 CALL LINES (FUNCT,X2,H,RO,NDIM,F2,G2,NUMF,IER,EPS,EST,RR)
IF (NFAIL,EQ.1) RETURN
IF (IER).NE.01 GO TO 30
6 DO 4 I=1,N1
BIGV(I)=X2(I)
7 CONTINUE
ALFA(I)=X2(I)
8 CONTINUE
9 DO 6 I=1,NDIM
G0=0.
10 CONTINUE
11 IF (PRINT.EQ.0) GO TO 7
12 IF (MOD(KTB,10).NE.01) GO TO 6
13 CALL OUTP (X2,F2,M,NDIM,G0,NUMF,RR)
14 KTB=KTB+1
15 CONTINUE
16 DO 8 J=1,N1
P(I,J)=0.
17 CONTINUE
18 CONTINUE
19 CONTINUE
20 PRINTS=KOUNT
21 CONTINUE
22 CONTINUE
23 CONTINUE
24 CONTINUE
25 CONTINUE
26 CONTINUE
27 CONTINUE
28 DO 12 I=1,NDIM
Y(I)=G2(I)
29 CONTINUE
30 END
Y(1) = ETA(1)
Y(0) = 0
DO 13 I = 1, NDIM
Y(I) = X(I) + ETA(I)
PRINT*, 'GOT BY A2'
13 CONTINUE
?A = 0
DO 14 I = 1, N1
YA(I) = Y(I) + ALFA(I)
PRINT*, 'GOT BY A3'
14 CONTINUE
UYA(U) = YA
BIG = KOUNT
DO 15 I = 1, N1
PY(I) = 0
PE(I) = P(I) * KOUNT
DO 15 J = 1, N1
Y(I) = PY(I) * P(J) * Y(J)
PRINT*, 'GOT BY A4'
15 EYP = PY(KOUNT)
IF (ABS(EYP).LT.ETA(3)) GO TO 31
PY(KOUNT) = PY(KOUNT) - 1
DO 16 J = 1, N1
DO 16 I = 1, N1
P(I, J) = P(I, J) - PE(I) * PY(J) / EYP
PRINT*, 'GOT BY A5'
16 CONTINUE
ALFA(1) = 0
DO 17 J = 1, N1
ALFA(I) = ALFA(I) * P(I, J) + BIG + (J)
PRINT*, 'GOT BY A6'
17 CONTINUE
DEL = 0
DO 18 I = 1, NDIM
X2(I) = X(I) + ETA(I)
PRINT*, 'GOT BY A7'
18 CONTINUE
IF (ABS(DEL).GT.ETA(4)) GO TO 19
IF (IFLAG.EQ.1) RETURN
IFLAG = 1
GO TO 31
19 CONTINUE
IFLAG = 0
DO 20 I = 1, N1
H(I) = X2(I) - ALFA(I)
IF (DEL.GT.0) H(I) = + H(I)
PRINT*, 'GOT BY A8'
20 CONTINUE
DO 21 I = 1, NDIM
X1(I) = X2(I)
Q1(I) = Q2(I)
PRINT*, 'GOT BY A9'
21 CONTINUE
FL = F2
X1(N2) = X2(N2)
X1(N3) = X2(N1)
X2(N1) = ALFA(N1)
PRINT*, 'GOT BY A10'
CALL LINES (FUNCT, X2, H, RO, NDIM, F2, G2, NUMF, IER, EPS, EST, RR)
PRINT*, 'GOT BY A11'
IF (IFAIL.EQ.1) RETURN
PRINT*, 'GOT BY A12'
IF (IER.NE.0) GO TO 30
PRINT*, 'GOT BY A13'
IF (DEL.GT.0) RO = RO
PRINT, GOT BY 411
GO TO 411
DO 12 I=1,N
GO=GO+G(I)*G(I)
PRINT*, 'GOT BY A15'
CONTINUE
GO=SORT(GG)
KOUNT=KOUNT+1
M=M+1
IF (IPRINT.EQ.0) GO TO 33
IF (MOD+NTB .EQ. 1) GO TO 33
PRINT*, 'GOT BY G'
CALL OUTP(X2,F2,M,NDIM,GO,NUMF,RR)
PRINT*, 'GOT BY H'
CONTINUE
!
KTB=KTB+1
IF (MODE.EQ.2) GO TO 25
PRINT*, 'GOT BY HA'
IF (M.GT.MAX) GO TO 30
!
PRINT*, 'GOT BY HB'
NSOL=0
DO 24 I=1,N
IF (ABS(RR(I)).GT.E7A(2)) NSOL=I
PRINT*, 'GOT BY HC'
CONTINUE
!
PRINT*, 'GOT BY HD'
IF (NSOL.EQ.0) GO TO 26
PRINT*, 'GOT BY HE'
GO TO 29
PRINT*, 'GOT BY HF'
IF ((GO(I).GT.MAX).OR.(M.GT.MAX)) GO TO 26
PRINT*, 'GOT BY HG'
GO TO 29
PRINT*, 'GOT BY HH'
CONTINUE
!
PRINT*, 'GOT BY HI'
IF (IPRINT.EQ.0) GO TO 27
PRINT*, 'GOT BY HJ'
WRITE(A433)
!
PRINT*, 'GOT BY I'
CALL OUTP(X2,F2,M,NDIM,GO,NUMF,RR)
PRINT*, 'GOT BY J'
DO 28 I=1,N
!
CONTINUE
NPAIL=0
RETURN
CONTINUE
PRINT*, KOUNT
PRINT*, 'GOT BY JA'
IF (KOUNT.LE.M1) GO TO 11
PRINT*, 'GOT BY JB'
GO TO 10
PRINT*, 'GOT BY JC'
!
PRINT 34, IER
NPAIL=1
RETURN
!
IF (KRST.GT.10) NPAIL=1
IF (NPAIL.EQ.1) RETURN
DO 32 I=1,N
X(I)=N
G(I)=G(I)
RETURN
IF (ALFA) 6+0+4
    PRINT*, 'GOT BY BS'
IF (ALFA)-AMBDA 5+0+6
    PRINT*, 'GOT BY B6'
AMBDA=ALFA
ALFA=0
DO 8 I=1:N+AMBDA#1
    PRINT*, 'GOT BY B7'
8 CONTINUE

IF (LY$,FX) RETURN
    PRINT*, 'GOT BY B11'

DX=0
    PRINT*, 'GOT BY BS'
    CALL FUNCT (N, X, G+AR)
    PRINT*, 'GOT BY BS'
    IF (MFAIL.EQ.1) RETURN
    PRINT*, 'GOT BY B20
    NUM=NURM+1
    IF (F.LT.FX) RETURN
    PRINT*, 'GOT BY B11'

DY=0
    PRINT*, 'GOT BY B12'
    CONTINUE

IF (X By 313)
    PRINT*, 'GOT By B14'
    CONTINUE

IF (F<FX) 11, 13, 17
    PRINT*, 'GOT BY B13
    AMBD=AMBDA
    ALFA=AMBDA
    IE (AMBDA, AMBD-1, E101, 7, 7, 12)
    PRINT*, 'GOT BY B16
    IER=2
    GO TO 31

    PRINT*, 'GOT BY B17'
    T=0
    IF (AMBD) 15, 30, 15
    PRINT*, 'GOT BY B18
    Z$=(ALFA-FX) AMBD+DX+DY
    ALFA=MAX1 (ABS(Z)+ABS(DX)+ABS(DY))
    ALFA=DALFA/DALFA-DX/DALFA#DY/DALFA
    IF (DALFA) 31, 16, 16

    PRINT*, 'GOT BY B19
    W=ALFA#SORT (DALFA)
    ALFA=DX+DY#W
    IE (ALFA) 17, 19, 17

    PRINT*, 'GOT BY B20
    ALFA=DY+W)/ALFA
    GO TO 19

    PRINT*, 'GOT BY B21
    ALFA=(Z+DY-W)/(Z+DY+Z+DY)
    IF (DFAIL.EQ.1) RETURN
    PRINT*, 'GOT BY B23
    NUM=NUM+1
    CONTINUE
    CALL FUNCT (N, X, G+RR)
    IF (MFAIL.EQ.1) RETURN
    CONTINUE
    PRINT*, 'GOT BY B23
    IF (F.LT.FX) GO TO 30
SUBROUTINE FUNCT (N, AL, U, GRAD, RR)
IMPLICIT REAL*8 (A-M, O-Z)

THIS SUBROUTINE IS USED TO CALCULATE THE OPTIMIZATION AND THE
GRADIENT AT ANY GIVEN POINT FOR SUBROUTINE POPT.

DIMENSION AL(N), GRAD(N), SUM(17), RR(N)
COMMON /FAIL/, NFAIL
COMMON /HELP/, S(101), XX(16,101), C(8), M

C.... ABOVE LINE CHANGED FROM TEXT

N21=2*N+1
ZERO=0.0
DO 1=1,N21
SUM(i)=0.0
1 CONTINUE

PRINT*, 'GOT BY C1'
DO 2=1,N
SUM(i)=SUM(i)+AL(i)*XX(K,i)
2 CONTINUE
PRINT*, 'GOT BY C2'

S=SUM(*)
C=’C’
IF (S. LT. 74.) GO TO 9
PRINT*, 'GOT BY C3'
C=’C’

IF (IDF-DALFA. LT. -1.D-10) IDA=-1
IF (IDF-DALFA. GT. 1.D-10) IDA=1
IF (SUM(i). LT. EPS) IDA=-1
IF (SUM(i). GT. EPS) IDA=1

RETURN
END
DO 4 J=1,N1
    SUM(J)=SUM(J)+XX(J-1,1)*S3
    PRINT*,'GOT BY C4'
    CONTINUE
DO 5 I=1,N2
    SUM(I)=SUM(I)/SUM(I)
    PRINT*,'GOT BY C5'
    CONTINUE
    U=0.0
    DO 5 I=1,N
    RRI=SUM(I-1)+C(I)+/C(I)
    U=U+RRI(I)*RR(I)
    PRINT*,'GOT BY C6'
    CONTINUE
DO 3 K=1,N
    GRAD(K)=0.0
    DO 7 J=1,N
    GRAD(K)=GRAD(K)+(SUM(J+K)-SUM(J+1)+S3*SUM(K+1))*.RR(J)/C(J)
    PRINT*,'GOT BY C7'
    CONTINUE
    GRAD(K)=GRAD(K)/S3
    PRINT*,'GOT BY C8'
    CONTINUE
    PRINT*,'GOT BY C9'
    PRINT*,'GOT BY C10'
    CONTINUE
    AA=32-J2
    ZERO=ZERO+AA
    GO TO 2
PRINT*,'GOT BY C11'
END
SUBROUTINE START (XMAX,XMIN,ALAMDA,NSTART,CC,ML,IPRINT,UHMIN,MODE,M)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION R(N1)
DIMENSION CC(*)
DIMENSION ALAMDA(*)
DIMENSION X(10)
COMMON /FAIL/, NFAIL
GO TO (3,15,16), NFAIL
1 CONTINUE
   RETURN
2 CONTINUE
   NFAIL=0
   ALAMDA(1)=CC(1)/CC(2)
   ALAMDA(1)=ALAMDA(1)+CC(2)
   ALAMDA(1)=0.0
   CONTINUE
   RETURN
3 CONTINUE
   NFAIL=0
   ALAMDA(1)=CC(1)/CC(2)
   ALAMDA(1)=ALAMDA(1)+CC(2)
   ALAMDA(1)=0.0
   CONTINUE
   RETURN
5 CONTINUE

CONTINUE
DO 23 I=1,NP1
W(I,1)=R(I)
CONTINUE
DO 24 J=1,NP1
M(I,J)=M(I-1,J)*X(J)
Y(I,1)=Y(I,1)/DELT
CONTINUE
Y(I,1)=Y(I,1)*Y(I,1)
CONTINUE
CALL SOLVE (W,Y;XID;NP1,10)
GO TO 12
CONTINUE
N=2
ALAMDA(2)=-0.5/CC(2)
ALAMDA(1)=CC(1)/CC(2)
IF (N.EQ.NL) RETURN
ALAMDA(N+1)=0.0
N=N+1
GO TO 27
END

SUBROUTINE SOLVE (AX;XID;N;NA)
IMPLICIT REAL*8 (AX=H0-2)
DIMENSION A(NA,N), X(N)
B=0.
DATA DIV/6931471817/
DO 7 J=1,N
AA=0.
DO 1 J=1,N
AB=ABS(A(J,1))
IF (AB.LE.AA) GO TO 1
K=J
AA=AB
1 CONTINUE
B=B+ALOG(AA).
IF (I.EQ.N) GO TO 7
IF (K.EQ.I) GO TO 3
DO 2 J=1,N
AB=ABS(A(J,1))
2 CONTINUE
AB*XI
X(I)=X(K)
X(K)=AB
DO 3 I=1,N
AA=AA+A(J,1)/A(I,1)
A(J,1)=0.
DO 4 K=1,N
A(J,K)=A(K,K)+AA*K(I,K)
4 CONTINUE
SUBROUTINE SIMSON

SUBROUTINE MULTI (XMAX, XMIN, N)

SUBROUTINE CONVEX (CM, ML)

DIMENSION CM(*)
COMMON HELP :: (1) (+X(+1)+10) -> C10
C.... ABOVE LINE CHANGED FROM TEXT
C(1)=CH(1)
IF (NL.EQ.1) RETURN
DO 2 J=3, NL
C(J)=C(J)-C(1)**J*(-1)**J
M=M-1
DO 1 K=1, M
C(J)=C(J)*(-1)**K*FACTO(J)/(FACTO(K)*FACTO(J-K))**C(1)**K*C(J-K)
1 CONTINUE
RETURN
END

SUBROUTINE TRM1 (XMAX,XMIN,C,X2MAX,X2MIN,NL)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION C(1)
SCL=(XMAX-XMIN)/(X2MAX-X2MIN)
C(1)=C(1)/SCL-XMIN/SCL+X2MIN
IF (NL.EQ.1) RETURN
DO 1 I=2, NL
C(I)=C(I)/SCL**(FLOAT(I))
1 CONTINUE
RETURN
END

SUBROUTINE TRN2(XMAX,XMIN,X,X2MAX,X2MIN,N)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION SMAX,DX101,FAC,DX1MAX,DXMIN,DX2MAX,DX2MIN
NPI=NP1
DO 10 I=1, NP1
10 S=(DX1MAX-DXMIN)/(DX2MAX-DX2MIN)
A=DX2MIN-DXMIN/S
DX(X)=DX(X)-ALOG(S)
DO 3 I=1,N
3 DX(I)=DX(I)+DX(I)*I**2
CONTINUE
IF (N.EQ.1) GO TO 6
DO 2 I=J+1, N
2 FACT=1
DO 2 K=K+1
2 FACT=FACT**DBLE(FLOAT(K))
CONTINUE
DX(J)=DX(J)+FACT**DBLE(FLOAT(J-I))*DX(I)**I
6 CONTINUE
C4 DX(J)=DX(J)/S**(J-1)
SUBROUTINE TRN2 (X1MAX, X1MIN, X2MAX, X2MIN, N)
IMPLICIT REAL*8 (A-H, O-Z)

THIS SUBROUTINE IS USED TO CALCULATE THE LAGRANGIAN MULTIPLIERS
AT THE ORIGINAL LIMITS

DIMENSION X(1)
S = (X1MAX - X1MIN)/(X2MAX - X2MIN)
A = (X1MIN - X1MIN)/S
X(1) = X(1) - ALOG(S)

DO I = 1, N
X(I) = X(I) + (X(I) + 1)**S
CONTINUE
IF (N.EQ.1) GO TO 5
DO 3 I = J - 1
FAC = 1.
K = I + 1
DO 2 K = J, I
FAC = FAC*FLOAT(K)
2 CONTINUE
X(J) = X(J) + FAC/FAC(I+1)**S
CONTINUE
X(N+1) = X(N+1)/S
RETURN
END

FUNCTION CDF (XMIN, XMAX, XP, AL, N)
IMPLICIT REAL*8 (A-H, O-Z)

THIS FUNCTION SUBROUTINE IS TO CALCULATE THE CUMULATIVE DISTRIBUTION FUNCTION AT A GIVEN POINT

INPUT
XMIN = LOWER BOUND
XMAX = UPPER BOUND
XP = SPECIFIED POINT
AL(I) = ARRAY OF PARAMETERS, DIMENSION N
N = NUMBER OF PARAMETERS

DIMENSION AL(N)

IF (XP.LE.XMIN) GO TO 3
IF (XP.GE.XMAX) GO TO 4
RANGE = XMAX - XMIN
RANGE = XP - XMIN
SS = RANGE/RANGE*51.
JSS = SS
JSS = (JSS/2)*2+5
AREA = 0.0
JSM1 = JSS - 1
DELTA = RANGE/FLOAT(JSM1)
DO 1 I = 2, JSM1+2
AREA = AREA + .5*TRNPF(AL, N, X)
**INPUT DATA FOR SUBROUTINE MEP:**

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Data is printed out for KDATA = 1 only</td>
<td>KDATA = 1</td>
</tr>
<tr>
<td>Intermediate output every KPRINT(TH) CYCLE</td>
<td>KPRINT = 1</td>
</tr>
<tr>
<td>Number of known first moments</td>
<td>N = 4</td>
</tr>
<tr>
<td>Higher limit</td>
<td>XMAX = 0.963779301E+01</td>
</tr>
<tr>
<td>Lower limit</td>
<td>XMIN = 0.37234981E+01</td>
</tr>
<tr>
<td>First moments</td>
<td>GC(i) = 0.735481e2SE+01</td>
</tr>
<tr>
<td>The allowed tolerance in Lagrangian equations</td>
<td>TOL = 0.100000000E-05</td>
</tr>
<tr>
<td>The cumulative distribution required at NXP points</td>
<td>NXP = 0</td>
</tr>
</tbody>
</table>
9.0 APPENDIX D

IMSL SUBROUTINE CALLS FROM RANDOM3 AND RANDOM4

RANDOM3

1. RNSET - Initializes a random seed for use in the IMSL random number generators.

2. RNNOR - Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.

3. RNLNL - Generates pseudorandom numbers from a lognormal distribution.

4. DESPL - Performs nonparametric probability density function estimation by the penalized likelihood method.

5. GCDF - Evaluates a general continuous cumulative distribution function given the ordinates of the density.

RANDOM4

1. RNSET - Initializes a random seed for use in the IMSL random number generators.

2. RNNOR - Generates pseudorandom numbers from a standard normal distribution using an inverse CDF method.

3. RNLNL - Generates pseudorandom numbers from a lognormal distribution.
10.0 APPENDIX E

SAMPLE SAS/GRAPH PROGRAM FOR RANDOM3 AND RANDOM4

data a;
INFILE 'PLOT1.CPR' FIRSTOBS=2; input x y;
GOPTIONS DEVICE=HP7470;
proc gplot;
  axis1 label=(h=l f=simplex 'LOG OF CYCLES')
    value=(h=l f=simplex);
  axis2 value=(h=l f=simplex) label=none;
  plot y*x / haxis=axis1 vaxis=axis2;
  TITLE H=1 A=90 F=SIMPLEX 'PROBABILITY DENSITY FUNCTION';
  symbol i=spline v=square;
data B;
INFILE 'PLOT2.CPR' FIRSTOBS=2; input x y;
proc gplot;
  axis1 label=(h=l f=simplex 'LOG OF CYCLES')
    value=(h=l f=simplex);
  axis2 value=(h=l f=simplex) label=none;
  plot y*x / haxis=axis1 vaxis=axis2;
  TITLE H=1 A=90 F=SIMPLEX 'CUMULATIVE DISTRIBUTION FUNCTION';
  symbol i=spline v=square;