THE PURGE3, MIX1 AND
MIX2 SUBROUTINES

by

Franklin McKie

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
in
Statistics

Approved:

Richard G. Krutchkoff, Ph.D.
Chairman

Boyd Harshbarger, Ph.D.

Klaus Hinkelmann, Ph.D.

May 1970

Blacksburg, Virginia
LD
5655
V855
1970
M35
Spec
ACKNOWLEDGEMENTS

The author wishes to express his grateful appreciation for the assistance rendered by the following persons cooperating in this study:

Dr. Richard G. Krutchkoff, whose supervision and generous assistance played a vital role in the completion of this study.

Dr. Klaus Hinkelmann for his reading and corrections of the manuscript.

Dr. Boyd Harshbarger for his concern and encouragement in the work.

Mrs. Marion Price for her diligence in typing the thesis.

This investigation was supported in part by NIH Training Grant No. 5T1GM2-10 from the National Institutes of Health, Research Training Grants Branch, National Institutes of General Medical Sciences, and by a Department of the Army Government Purchase Order issued from the Ammunition Procurement and Supply Agency, Joliet, Illinois.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II.</td>
<td>METHODS OF GENERATING PSEUDO-RANDOM NUMBERS AND PEARSON FREQUENCY CURVES</td>
<td>4</td>
</tr>
<tr>
<td>2.1</td>
<td>Description of Pseudo-Random Numbers</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>Common Methods of Generating Pseudo-Random Numbers</td>
<td>5</td>
</tr>
<tr>
<td>2.21</td>
<td>The Mid-Square Methods</td>
<td>5</td>
</tr>
<tr>
<td>2.22</td>
<td>Multiplicative Congruential Methods</td>
<td>5</td>
</tr>
<tr>
<td>2.23</td>
<td>Mixed Congruential Methods</td>
<td>6</td>
</tr>
<tr>
<td>2.24</td>
<td>Multiplicative Vs. Mixed Generators</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>The System of Pearson's Frequency Distributions</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Use of Pearson Curves with PURGE3</td>
<td>12</td>
</tr>
<tr>
<td>III.</td>
<td>PURGE3 SUBROUTINE</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>Description</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Method of Operation</td>
<td>17</td>
</tr>
<tr>
<td>3.4</td>
<td>Tests for Randomness</td>
<td>19</td>
</tr>
<tr>
<td>3.5</td>
<td>Operating Information</td>
<td>20</td>
</tr>
<tr>
<td>3.51</td>
<td>Data Input Options</td>
<td>20</td>
</tr>
<tr>
<td>3.52</td>
<td>Parameter Options</td>
<td>22</td>
</tr>
<tr>
<td>3.53</td>
<td>Regeneration</td>
<td>25</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS (Cont.)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.54 Variables Available Through COMMON</td>
<td>27</td>
</tr>
<tr>
<td>3.55 Error Conditions</td>
<td>27</td>
</tr>
<tr>
<td>3.6 Sample Computer Programs</td>
<td>30</td>
</tr>
<tr>
<td>IV. MIX1 SUBROUTINE</td>
<td>38</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>38</td>
</tr>
<tr>
<td>4.2 Method of Calculation</td>
<td>39</td>
</tr>
<tr>
<td>4.3 Graphing a MIX1 Density</td>
<td>41</td>
</tr>
<tr>
<td>4.4 Operating Instructions</td>
<td>42</td>
</tr>
<tr>
<td>4.41 Data Input Option</td>
<td>42</td>
</tr>
<tr>
<td>4.42 Regeneration</td>
<td>42</td>
</tr>
<tr>
<td>4.43 Variables in COMMON</td>
<td>46</td>
</tr>
<tr>
<td>4.44 Error Conditions</td>
<td>46</td>
</tr>
<tr>
<td>4.5 Sample Computer Programs</td>
<td>48</td>
</tr>
<tr>
<td>V. MIX2 SUBROUTINE</td>
<td>56</td>
</tr>
<tr>
<td>5.1 Introduction</td>
<td>56</td>
</tr>
<tr>
<td>5.2 Method and Operating Instructions</td>
<td>56</td>
</tr>
<tr>
<td>5.21 Regeneration</td>
<td>57</td>
</tr>
<tr>
<td>5.22 Variables in COMMON</td>
<td>57</td>
</tr>
<tr>
<td>5.23 Error Conditions</td>
<td>57</td>
</tr>
<tr>
<td>5.3 Sample Computer Program</td>
<td>60</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>62</td>
</tr>
</tbody>
</table>

iv
| APPENDIX A | Listing of Uniform and Normal Random Number Generators Used | 64 |
| APPENDIX B | Flow Chart and Listings for PURGE3 | 66 |
| APPENDIX C | Flow Chart and Listings for MIX1 | 95 |
| APPENDIX D | Flow Chart and Listings for MIX2 | 108 |
| VITA | | 114 |
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>16</td>
</tr>
<tr>
<td>II.</td>
<td>21</td>
</tr>
<tr>
<td>III.</td>
<td>23</td>
</tr>
<tr>
<td>IV.</td>
<td>28</td>
</tr>
<tr>
<td>V.</td>
<td>43</td>
</tr>
<tr>
<td>VI.</td>
<td>44</td>
</tr>
<tr>
<td>VII.</td>
<td>47</td>
</tr>
<tr>
<td>VIII.</td>
<td>58</td>
</tr>
<tr>
<td>IX.</td>
<td>59</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>Figure No.</strong></td>
<td><strong>Page</strong></td>
</tr>
<tr>
<td>1. Data Point Card Input</td>
<td>32</td>
</tr>
<tr>
<td>2. Moment Card Input</td>
<td>33</td>
</tr>
<tr>
<td>3. PURGE3 Output</td>
<td>34</td>
</tr>
<tr>
<td>4. PURGE3 Output</td>
<td>35</td>
</tr>
<tr>
<td>5. PURGE3 Output</td>
<td>36</td>
</tr>
<tr>
<td>6. PURGE3 Output</td>
<td>37</td>
</tr>
<tr>
<td>7. MIX1 Output with Smoothing</td>
<td>49</td>
</tr>
<tr>
<td>8. MIX1 Output Without Smoothing</td>
<td>50</td>
</tr>
<tr>
<td>9. MIX1 Output with Smoothing</td>
<td>52</td>
</tr>
<tr>
<td>10. MIX1 Output with Smoothing</td>
<td>53</td>
</tr>
<tr>
<td>11. MIX1 Output with Smoothing</td>
<td>55</td>
</tr>
<tr>
<td>12. Input for MIX2</td>
<td>61</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

Many applications of electronic computers require the efficient generation of large numbers of pseudo-random deviates. Tables of pseudo-random normal deviates are available, but they are not sufficiently extensive for many purposes and an outside source of this kind cannot usually be effectively used by the computer. What is required is some method of generation which can be rapidly carried out by the machine itself.

MacLean and Marsaglia (1965) define a pseudo-random number generator as any procedure for producing within a computer program a sequence of numbers \( u_1, u_2, \ldots \) which is supposed to represent a sequence of independent uniform random variables. Although the \( u_i \) look random, they are usually generated by a strictly deterministic procedure, thus the name pseudo-random numbers.

With the advent of high speed computers, a number of techniques in physics, operations research, applied mathematics, statistics, and many other disciplines now involve stochastic simulation or Monte Carlo methods, the branch of mathematics concerned with synthetic experiments involving random numbers. Monte Carlo calculations sometimes require large sequences of random numbers from various probability distributions.

This thesis was written for the purpose of modifying an existing PURGE (Pearson Universal Random Distribution Generator) subroutine
which was used to generate random numbers from Pearson distributions. Here, the subroutine is revised and made more flexible to increase efficiency in using the subroutine in conjunction with another subroutine, MIX. The MIX subroutine was previously written to be run on the IBM 7040 computer system and read in binary data cards containing the cumulative distribution function and other parameters, in a different computer run, that were generated in the PURGE subroutine.

Here the PURGE subroutine, called PURGE3, can store up to two Pearson cumulative distribution functions at one time and can pass these functions to a new MIX subroutine, called MIX1, VIA COMMON blocks in a single computer run. The Pearson distribution may be specified by supplying the program with either the first four moments of the desired distribution or sample data from that distribution. Random numbers may then be generated from the Pearson distribution curve fitted.

The MIX1 subroutine removes the restriction of generating pseudo-random numbers from only unimodal distributions as is the case in the PURGE3 subroutine. Any proportion of two Pearson distributions, for example, a normal distribution with arbitrary mean and variance and a uniform distribution on any finite interval may be used in creating a mixture of distributions from which random numbers are generated.

Another subroutine called MIX2 is developed to read data cards produced in PURGE3 containing the cumulative distribution function,
the number of intervals of the cumulative distribution function, and centile index, the first four moments of the Pearson distribution function and parameters computed from the Pearson equation which are used in the random number calculation. MIX2 is solely a random number generator without the output options available in the PURGE3 and MIX1 subroutines.

Flow charts and listings of each subroutine appear in the Appendix along with the random number generators used.
CHAPTER II

METHODS OF GENERATING PSEUDO-RANDOM NUMBERS
AND PEARSON'S FREQUENCY CURVES

2.1. Description of Pseudo-Random Numbers

A number of methods have been proposed for the generation of pseudo-random numbers hereafter referred to simply as random numbers. Generally, these methods follow the same pattern.

A starting number is selected from which the first random number is determined. Once started, the \((N+1)\)-th number is determined from the \(N\)-th number. It follows then that if the same starting number is used for two sequences of random numbers, generated by the same method, the sequences would be identical. The starting number of many random number generators is the same for different computer runs and care should be taken in subsequent runs to insure that the same batch of random numbers is not used repeatedly, thus invalidating the experimental results obtained.

This property of repeatability is extremely desirable when testing or debugging a computer program in which the order of executing the instructions may depend upon the value of a generated random number. This characteristic is also useful if it is desired to make conditional use of a sequence of random numbers depending, for example, on whether the mean, variance, or some other statistic lies within acceptable limits. These statistics may be calculated for a large sequence without storing any of the numbers in the sequence.
Another characteristic of all pseudo-random number generators is that the sequence repeats itself. The total number, \( X \), of numbers generated is usually called the period. In most cases the period has been determined for the generators most commonly used and can be made quite large by careful selection of parameters used in the generator.

2.2. Common Methods of Generating Pseudo-Random Numbers

2.21. The Mid-square Methods

This method can be described as follows. Beginning with an \( n \) (usually even) digit number, say \( x_0 \), the first random number \( x_1 \) is obtained by calculating \( x_0^2 \) and extracting the middle \( n \) digits from it. \( x_2 \) is obtained from the middle \( n \) digits of \( x_1^2 \), etc.

This method has been generally discarded because it does not have a long period. The period must be found by trial for each number and the sequence generally does not return to the starting data to repeat, i.e., after the first \( n+m \) numbers have been generated, the sequence may repeat beginning with the \( m \)th number.

2.22. Multiplicative Congruential Methods

This class of generation methods can be defined by the congruence relation

\[ x_{n+1} \equiv ax_n \pmod{m} \]

introduced by Lehmer (1951), where \( m \) is generally taken as \( 2^n \) for an \( n \)-bit binary machine and \( 10^n \) for an \( n \) digit decimal machine. The
sequence of integers \( x_0, x_1, x_2, \ldots \) is determined by the choice of \( x_0, a, \) and \( m \), where the three parameters are non-negative integers, \( m \) being the largest, and \( x_0 \) is the starting value. This number is multiplied by the integer \( a \) and the product is divided by \( m \). The resulting remainder is \( x_1 \), the first random number generated. The next random number is obtained by replacing \( x_0 \) with \( x_1 \) and repeating the same calculations. Then the hope is that the sequence \( x_0/m, x_1/m, x_2/m, \ldots \) will appear to be drawn at random from the uniform distribution on \([0, 1] \).

Clearly, this sequence must eventually repeat itself since it can contain at most \( m \) different numbers with each succeeding number in a sequence determined by its predecessor. Because of its speed and excellent statistical properties which may be obtained by proper choice of \( x_0, a \) and \( m \), this method of generating uniform random numbers is widely used.

2.23. **Mixed Congruential Methods**

The mixed congruential method is similar to the multiplicative methods which are a special case of them and are defined by the congruence relation

\[
x_{n+1} \equiv a \cdot x_n + c \pmod{m}.
\]

In this case \( c \) is a non-negative integer which is added to the product \( a \cdot x_n \) before division by \( m \). Otherwise the computations are identical to those in the multiplicative case.
2.24. **Multiplicative Vs. Mixed Generator**

In tests performed by Hull and Dobell (1962), the results indicate that although in general their statistical behavior is quite good, in some respects the mixed methods show poorer statistical behavior than multiplicative methods. In tests of 600 different multipliers using the mixed method, about one per cent of these were unacceptable in the sense that they lead to ridiculously large values of $\chi^2$, some as large as 900. They also report that in tests made with 513 different multipliers using the multiplicative congruential method "the results were entirely consistent with the hypothesis that the sequence was drawn at random from the uniform distribution."

2.3. **The System of Pearson's Frequency Distributions**

The methods of generating random numbers that were described in the previous section had the inherent limitation of being designed to generate a uniform distribution and were generally not easily modified to generate other distributions. Obviously, a method of generating random variables from a wide variety of continuous distributions would be highly desirable.

The generator discussed here generates pseudo-random variates from any member of the Pearson family of frequency distribution functions. The family of unimodal frequency functions obtained as solutions to the differential equation

$$\frac{df}{dx} = \frac{(x-a)f}{b_0 + b_1 x + b_2 x^2} \quad [1.1]$$
are known as Pearson distributions. The form of the solution depends on the values of the constants \(a, b_0, b_1,\) and \(b_2\) which may be shown to relate to the moments of the curve. See Kendall and Stuart (1963).

In equation [1.1] defining the Pearson distributions, it is evident that the mode is at the point \(x = a\). Further, if the origin is taken at the mode so that \(a = 0\) the equation

\[
\frac{d^2f}{dx^2} = \frac{d}{dx} \frac{xf}{b_0 + b_1 x + b_2 x^2} = \frac{f}{(b_0 + b_1 x + b_2 x^2)^2} (b_0 - b_2 x^2)
\]

results. Thus any points of inflection in the frequency curve are given by

\[
x^2 = \frac{b_0}{b_2}.
\]

Hence there cannot be more than two points of inflection, and if two exist, they are equidistant from the mode.

If we translate the origin to the mode, [1.1] may be written as

\[
\frac{d}{dx} (\log f) = \frac{x-a}{b_0 + b_1(x-a) + b_2(x-a)^2}
\]

or

\[
\frac{d}{dX} (\log f) = \frac{X}{b_0 + b_1 X + b_2 X^2}
\]

where \(X = x-a\).

The explicit expression of the frequency function \(f\) is a matter of integrating the right hand side of [1.4]. We may now distinguish
two main types of Pearson distributions depending on whether the
denominator on the right has real or imaginary roots.

TYPE I (BETA-DISTRIBUTION: REAL ROOTS)

Let \( B_0 + B_1 x + B_2 x^2 = B_2 (x + c_1)(x - c_2), \ c_1, c_2 > 0. \)

Then

\[
\frac{d}{dx} \log f = \frac{x}{B_2(x + c_1)(x - c_2)}
\]

\[
= \frac{c_1}{B_2(c_1 + c_2)} \cdot \frac{1}{(x + c_1)} + \frac{c_2}{B_2(c_1 + c_2)} \cdot \frac{1}{(x - c_2)}
\]

whereupon

\[
f = K(x + c_1) \frac{c_1}{B_2(c_1 + c_2)} \cdot \frac{c_2}{B_2(c_1 + c_2)} (x - c_2)
\]

This may be written in the form

\[
f = K(1 + \frac{x}{a_1})^{M_1} \cdot (1 - \frac{x}{a_2})^{M_2}
\]

where

\[
\frac{M_1}{a_1} = \frac{M_2}{a_2}
\]

If the origin is at the mean

\[
f = y_0 (1 + \frac{x}{A_1})^{M_1} \cdot (1 - \frac{x}{A_2})^{M_2}
\]
where

\[ A_1 + A_2 = a_1 + a_2 \]

and

\[ \frac{M_1 + 1}{A_1} = \frac{M_2 + 1}{A_2} . \]

The constants \( Y_0 \), \( M_1 \) and \( M_2 \) are determined by integrating the curve over its range from \(-a_1\) to \( a_2\). This curve varies in shape from bell to \( u \), and twisted \( J \).

**TYPE IV (IMAGINARY ROOTS):**

If the roots of \( B_0 + B_1X + B_2X^2 \) are imaginary, we have

\[
\frac{d}{dx} (\log f) = \frac{X}{B_2 [(X + \frac{B_1}{2B_2})^2 + \frac{B_0}{B_2} - \frac{B_1^2}{4B_2^2}]}\]

\[ = \frac{X}{B^2 [(X + g)^2 + h^2]} \]

giving

\[ \log f = \log K + \frac{1}{2B_2} \log [(X + g)^2 + h^2] - \frac{g}{B_2 h} \tan^{-1} \frac{X + g}{h} , \]

\[ f = K [(X + g)^2 + h^2]^{\frac{1}{2B_2}} \exp \left[ - \frac{g}{B_2 h} \tan \frac{X + g}{h} \right] , \]

usually written
\[ f = K (1 + \frac{x^2}{a^2})^{-\frac{M}{2}} \exp \left( -V \tan \frac{x}{a} \right) \]

where the origin is \( \frac{VA}{(2M-2)} \) after the mean. The distribution has unlimited range in both directions and is unimodal. For determination of the constants as well as the derivation of other Pearson curves see Elderton and Johnson (1969).

Pearson gives a criterion to distinguish the main types of Pearson curves. He defines

\[ K = \frac{\beta_1 (\beta_2 + 3)^2}{4 (4\beta_2 - 3\beta_1) (2\beta_2 - 3\beta_1 - 6)} \]

with \( \beta_1 = \frac{\mu_3}{\mu_2^3} \) and \( \beta_2 = \frac{\mu_4}{\mu_2^2} \) being coefficients of skewness and kurtosis respectively. For Type I curves, \( K < 0 \) and for Type IV curves \( 0 < K < 1 \).

In summary, all the Pearson distributions are determined by the first four moments, \( \mu_1 \) to \( \mu_4 \) inclusive, except some of the degenerate types which are determined by fewer than four moments. Pearson's method of fitting consists of

1. determining the numerical values of the first four moments from the observed distribution;
2. calculating the numerical values of \( \beta_1, \beta_2 \) and \( K \) and then determining the type to which the distribution belongs;
3. equating the observed moments to the moments of the appropriate distribution expressed in terms of its parameters.
solving the resulting equations for those parameters, whereupon the distribution is determined.

2.4. Use of Pearson Curves with PURGE3

In calling PURGE3 one may specify AVE, CM2, CM3, and CM4, variables of dimension two which represent the mean, variance, \( \mu(3) \), and \( \mu(4) \) respectively for a particular Pearson curve. It is possible to obtain the same shape curve or its mirror image without changing \( \beta_1 \) or \( \beta_2 \) in the following way (see Krutchkoff (1966)).

To Change Scale:

To change the scale to \( S \) times as large, one merely reads in \( S^2 \) times the indicated variance, \( S^3 \) times the indicated \( \mu(3) \) and \( S^4 \) times the indicated \( \mu(4) \). \( \beta_1 \) and \( \beta_2 \) and the shape of the curve will remain unchanged.

To Change Variance:

In order to change the variance to \( \sigma^2 \) one reads in \( CM2 = \sigma^2 \), \( CM3 = \sigma^3 \) times the indicated \( \mu(3) \) and \( CM4 = \sigma^4 \) times the indicated \( \mu(4) \). Again, \( \beta_1 \) and \( \beta_2 \) and the shape of the curve will remain unchanged.

To Change \( \mu(3) \):

In the rare event that one desires a particular shape curve or particular values of \( \beta_1 \) and \( \beta_2 \) with a definite particular \( \mu(3) \), this can be obtained. Find the proper curve disregarding the desired new
value of $\mu(3)$. Then set $t^3 = (\text{new } \mu(3)) \div (\text{the indicated } \mu(3))$.

Then set $CM2 = t^2$, $CM3 = \text{new } \mu(3)$, and $CM4 = t^4$ times the indicated $\mu(4)$.

To Change $\mu(4)$:

In a situation similar to the above one may keep the shape of a curve and also $\beta_1$ and $\beta_2$ constant while obtaining any $\mu(4)$ desired. This is done by setting $\sigma^4 = (\text{desired new } \mu(4)) \div (\text{indicated } \mu(4))$ and setting $CM2 = \sigma^2$, $CM3 = \sigma^3$ times the indicated $\mu(3)$ and $CM4 = \text{new } \mu(4)$.

Change in Position of Tail:

One may obtain the mirror image of any curve simply by defining $CM3 = -\mu(3)$. This will not change any of the previously obtained values. This can be done after the above adjustments.

To Change the Mean:

The mean should be the last adjustment considered. After any other adjustment is made the mean can now be set to any desired value $m$ by merely defining $AVE = m$. 
CHAPTER III

PURGE3 SUBROUTINE

3.1. Introduction

With the development of modern high speed computers, a versatile method of generating random numbers, using Pearson curves, for a family of unimodal distributions was developed by three members of the IBM Scientific Computation Department. By supplying the first four moments a wide variety of Pearson frequency functions can be defined. Cooper, David, and Lono (1963) wrote a computer program, "Pearson Universal Random Distribution Generator" (called PURGE) to generate random numbers from such distributions. This program was originally written to be run on the IBM 7090 system in FORTRAN II and was revised by Thomas (1966) to be used on the IBM 7040-1401 system in FORTRAN IV as a subroutine.

The work reported here revises and improves the program for use as a subroutine, called PURGE3, on the IBM 360 system in FORTRAN IV. The primary improvement enables the subroutine to store up to two fitted distribution curves for input to the MIX1 subroutine in a single computer run.

3.2. Description

The subroutine is entered by using the FORTRAN call statement

CALL PURGE3 (N1, N2)
where \( N_1 \) and \( N_2 \) are arguments which specify the options desired. Descriptions of these arguments appear in Table I. Initially, depending on the option selected, the subroutine reads data cards which describe the Pearson distribution to be fitted, or allows for the internal description of the distribution. The distribution may be described either by its first four moments or by sample data from the distribution which may be specified in various formats and fitted with several options. The distribution is fitted and parameters for the Pearson curve with origin at the mean are printed.

At this point, up to one hundred random numbers are generated, if desired, and returned to the calling program as implied arguments by placing the FORTRAN instruction

\[
\text{COMMON/MCK2/RDS(100), LIMIT}
\]

in the calling program. After each call to PURGE3, a new sequence of random numbers will be stored in the array RDS and may be referenced in that array. If desired, moments from the generated data are calculated.

If more random numbers are needed from the same distribution, the subroutine may be recalled, for each sequence of up to one hundred numbers required, by an argument option which does not necessitate refitting the curve. This procedure produces about 13,000 random numbers per minute on the IBM 360.

Available output options will print the moments used to fit the curve type as well as \( \beta_1, \beta_2 \), and \( K \) along with the values of these
Table I. Values of the Arguments for PURGE3

<table>
<thead>
<tr>
<th>Value of N1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read OPTION CARD and take action prescribed by it, i.e., read data cards, fit Pearson distribution, and generate 0 to 100 random numbers.</td>
</tr>
<tr>
<td>2</td>
<td>Generate up to 100 random numbers from a distribution previously described in the same program under the N1=1 or N1=3 option.</td>
</tr>
<tr>
<td>3</td>
<td>Fit a new Pearson curve from moments supplied by the calling program VIA the COMMON statement and generate 100 random numbers from this distribution.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value of N2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Summations of first four powers of all numbers generated from this distribution are stored for later calculation of sample moments.</td>
</tr>
<tr>
<td>2</td>
<td>Summations of first four powers of all numbers generated from this distribution are used to calculate sample moments which are then printed.</td>
</tr>
<tr>
<td>3</td>
<td>The c.d.f. is punched on cards in BCD format along with necessary parameters for generating random numbers from this distribution. (These data cards will be read as input to the MIX2 subroutine in a different computer run.)</td>
</tr>
<tr>
<td>4</td>
<td>Same action is taken as for N2=2, then a graph of the distribution is printed. (Must be last option taken for a given distribution.)</td>
</tr>
<tr>
<td>5</td>
<td>Random numbers are generated without calculating summations for computations of moments. Moments should not be printed for a given distribution after this option is used.</td>
</tr>
<tr>
<td>6</td>
<td>The c.d.f. along with the necessary parameters for generating random numbers from the first distribution are stored in arrays which are to be passed to the MIX1 subroutine within the same program VIA of COMMON blocks. Then the same action is taken as for N2=2.</td>
</tr>
<tr>
<td>7</td>
<td>Same action is taken as for N2=6 except that a second Pearson distribution has been fitted.</td>
</tr>
</tbody>
</table>

Under the N2=3 option, two job identification cards are in front of the first Pearson deck punched and an end-of-file card follows each deck. These cards must be removed before the deck is read in the MIX2 subroutine.
statistics calculated from the random numbers generated. After the last sequence of random numbers has been generated, a graph of the distribution function may be obtained. One should not graph the distribution and then attempt to generate random numbers from it because the graph routine shares part of its storage with the cumulative distribution function. As a result, part of the c.d.f. array is destroyed and further generation requires refitting the distribution.

Increased speed in generation may be achieved by selecting an option which does not calculate the moments of the generated numbers. The speed of generation is about 125,000 random numbers per minute when this option is used.

An option, particularly useful when dealing with empirical Bayes statistics, is available which allows moments for a new distribution to be supplied by the calling program without being read in on cards.

3.3. Method of Operation

If moments are given, the program calculates \( \beta_1, \beta_2, \) and \( K \) from which it determines the appropriate Pearson type. If a sample of the distribution is read as data input, the program computes the moments from this data and then determines the appropriate Pearson type.

After the Pearson type has been determined, the parameters for the curve with origin at the mean are calculated and printed along with the type of curve fitted. This printout occurs when a new curve is fitted. Pearson Types I, IV, VI and the normal curve may be fitted
as well as subtypes of them. An additional option allows all but TYPE III to be "force fitted" to a given set of moments or data, i.e., the usual Pearson criterion for determining types is overridden and parameters are calculated for the specific type from the given data. This procedure, intended for use with similar distributions, will not always yield a valid distribution. For example, if a u-shaped distribution is forced into a normal distribution, FORTRAN arithmetic errors such as attempting to take the square root of a negative number are encountered. Error messages are automatically printed when these conditions occur and the program continues although the results are incorrect.

If it is desired to generate random numbers from the fitted distribution, the distribution function is numerically integrated by Simpson's Rule as given in Scarborough (1958). The interval of integration is divided into 5120 equal parts (except 512 for the normal*) which divides the area under the curve into 2560 pieces. The number 2560 was chosen because it is large enough to give good results and yet not so large that it requires too much machine storage. The area of each piece is then accumulated in 2560 cells to obtain the cumulative distribution function (c.d.f.).

For the actual random number generation, a multiplicative congruential uniform random number generator is used to randomly select

---

*PURGE3 is not recommended to be used to generate normal random numbers since the MIX1 and MIX2 subroutine will generate normal random numbers several times faster.
points on the cumulative distribution function from which the random
numbers with the required distribution are determined. The starting
value of the uniform generator, which is preset in a Block Data
statement, is the integer 21319113 which is multiplied by the constant
65539. This generator and the normal random number generator, which
is used in the MIX1 and MIX2 subroutines, are coded as FUNCTION sub-
routines in the program.

To obtain random numbers from the desired Pearson distribution,
the program first determines the centiles of the c.d.f. and immediately
jumps to the appropriate centile to begin its search for the
ordinate of the c.d.f. If equality of the c.d.f. with the uniform
random number is not reached, linear interpolation is used between
intervals of the c.d.f. for greater accuracy and the abscissa, which
is the required random number, is calculated.

3.4. Tests for Randomness

Kaercher (1962), as referenced by Thomas (1966), reports satis-
factory statistical results were obtained from $\chi^2$ tests, tests on the
sample variances and means as well as checks of serial correlation.
However, since there is, theoretically, a one-to-one mapping of uni-
form random numbers into the desired distribution, many properties
of the generated sequence are determined by those of the uniform
random number generator used. For example, the Chi-square test, the
Kolmogorov-Smirnov test, and tests based on runs are all completely
determined by the corresponding tests on the uniform generator used.
3.5. Operating Information

The options available for various values of the arguments $N_1$ and $N_2$ used in the call state of PURGE3 are described in Table I.

3.5.1. Data Input Options

For each fit under the $N_1 = 1$ option, the format shown in Table II applies. The first data card to be used is the OPTION CARD. If the optional format control is desired for this fit, the next card should be the INPUT FORMAT CARD. Otherwise this card is omitted.

**Input Format Card:**

The optional format, when it is needed as indicated in columns 4-6 of the OPTION CARD, must be described on this one card. It must take the same form as a source program FORMAT statement except that the word format is omitted, and it may be punched in columns 1-72. Regardless of the format used, two characters (columns) of the record (card) must be reserved since the format must also read the END OF DATA CARD. Therefore, all INPUT FORMAT CARDS must end with I2 conversions. For example: to read input data four to the record, the following format, representing the built-in format that will be used if the optional format is not elected, might be used: $(4X,4F17.8,6X,I2)$. It reserves columns 79-80 for the two characters specifying the number of entries on the last data card. Use only F or E conversions.

The next card or cards should be the MOMENT CARD (or record) or DATA POINT CARDS (or records) depending on which option is chosen.
<table>
<thead>
<tr>
<th>Cols.</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*(Asterisk)</td>
<td>Blank</td>
</tr>
<tr>
<td>2-3</td>
<td>Blank</td>
<td>There are two types of input: four moments describing the distribution or an actual sampling of the distribution. To specify that the input is the four moments on the next card (see formats) punch -99 in this field. To specify sampling data in the built-in format, this field is left blank. To specify user's format, to be given on the next data card, a positive integer is placed in this field representing the number of entries per card or record of input; up to 30 per card or record. The computer run may be terminated by placing 999 in this field.</td>
</tr>
<tr>
<td>4-6</td>
<td>INPUT FORMAT</td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
<td>There are two ways to specify the number of data entries of the sampling type: (1) Enter the number of entries into this field. The number of entries on the last card or record will be computed automatically. (2) Place on the end of the data, a record in which two characters (columns) indicate the number of entries in the last entry card or record. (The remainder of that card or record should be blank.) To indicate this method leave this field blank.</td>
</tr>
<tr>
<td>7-9</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>10-15</td>
<td>INPUT NUMBER</td>
<td>The number of random numbers to be generated per call (1-100) is specified in this field. The optimum efficiency is obtained at 100. If blank, only the fit will be obtained.</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
<td>Blank</td>
</tr>
<tr>
<td>16-18</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>19-24</td>
<td>OUTPUT GEN</td>
<td>This field enables the user to override the usual Pearson criterion and force the fit to a specified type. If this field contains a one, four, six, or the integer 999, the Pearson Type One, Four, Six or Normal, respectively, will be used to fit the data. If blank, the usual Pearson criterion is used.</td>
</tr>
<tr>
<td></td>
<td>ERATION SPEC</td>
<td>Blank</td>
</tr>
<tr>
<td>25-27</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>28-30</td>
<td>FORCE FIT</td>
<td>Blank</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
<td>Blank</td>
</tr>
<tr>
<td>31-33</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>34-35</td>
<td>INPUT TAPE</td>
<td>Symbolic tape unit other than the normal card input (unit 5).</td>
</tr>
<tr>
<td></td>
<td>OPTION</td>
<td>Blank</td>
</tr>
<tr>
<td>36-38</td>
<td>Blank</td>
<td>Blank</td>
</tr>
<tr>
<td>39-40</td>
<td>Blank</td>
<td>Symbolic output tape unit other than standard output file (unit 6).</td>
</tr>
<tr>
<td>41-80</td>
<td>Blank</td>
<td>Blank</td>
</tr>
</tbody>
</table>
Moment Card:

The first four moments are specified on the MOMENT CARD as outlined in Table III.

Data Point Cards:

The DATA POINT CARDS (or records) used when the input is not moments, have the same format as the MOMENT CARD when using the built-in format. As explained earlier, the two columns to the right of the entries (columns 79 and 80 in the built-in format) must not be used since the same format reads the END OF DATA CARD.

If the sampling data input is used and the INPUT NUMBER OPTION is not used, i.e., columns 10-15 on the OPTION CARD are blank, the next card must be the END OF DATA CARD described below. Otherwise, this card is omitted.

The END OF DATA CARD is used to indicate the end of the sampling data and specifies how many entries are on the last DATA POINT CARD (or record). It is a blank card (or record) except for the two column integer field. These two columns are specified on the INPUT FORMAT CARD. Note that the built-in format uses columns 79-80. See section 3.5 for examples showing the use of the aforementioned data cards.

3.52. Parameter Options

Special attention here is focused on the use of the arguments in the call to PURGE3 to provide other options in using this subroutine.
Table III. Moment Card Format

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Cols.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-4</td>
<td>Blank</td>
<td></td>
</tr>
<tr>
<td>AVE</td>
<td>5-21</td>
<td>Mean or first moment about zero</td>
</tr>
<tr>
<td>CM2</td>
<td>22-38</td>
<td>Variance or second moment about zero</td>
</tr>
<tr>
<td>CM3</td>
<td>39-55</td>
<td>Third central movement about zero</td>
</tr>
<tr>
<td>CM4</td>
<td>56-72</td>
<td>Fourth central movement about zero</td>
</tr>
<tr>
<td>73-80</td>
<td>Blank</td>
<td></td>
</tr>
</tbody>
</table>

The format for each moment is F17.8.
This sequence of instructions in a control program will cause PURGE3 to read data cards, fit a distribution of up to 100 random numbers in the array TOM and exist, printing the type of distribution fitted, along with its parameters, and the moments calculated for the input data and the moments of the random numbers generated. The number of random numbers generated will be placed in the variable LIMIT.

```
COMMON/MCK2/TOM(100), LIMIT
.
.
.
CALL PURGE3(1,2)
```

If the next call to PURGE3 is

```
CALL PURGE3(2,2)
```

another sequence of up to 100 random numbers is generated for the same distribution, computing the summations of the first four powers of all numbers generated from this distribution and printing the same moments for the input data and the new moments of the output data.

If it is desired to generate a c.d.f. and related parameter data deck for later input to the MIX2 subroutine based on the same distribution as was fitted with the N1=1 option, the next call to PURGE3 would be

```
CALL PURGE(2,3).
```

To fit a different Pearson curve in the same program, the additional instructions
COMMON/MCK1/AVE(1),CM2(1),CM3(1),CM4(1),BETA1,BETA2,SKAPPA

AVE(1) = 0.
CM2(1) = 1.36
CM3(1) = 0.86
CM4(1) = 8.92
CALL PURGE3(3,4)

will fit a curve from the moments defined and generate another sequence of 100 random numbers and place them in array TOM. $\beta_1$, $\beta_2$, and K parameters will be placed in variables BETA1, BETA2, and SKAPPA, respectively. Then a graph of the 100 numbers generated is printed. This should be the last call to PURGE3 for this distribution since the graph routine shares storage with the c.d.f., hence destroying it. See section 3.6 for illustrations of sample programs.

3.53. Regeneration

PURGE3 has the capability of regenerating a sequence of random numbers or continuing from the same sequence in a different computer run.

This can be accomplished by modifying the uniform random number generator that appears in the Appendix. The following instructions in the control program will store the starting value of the uniform random number generator and reset it to its original value when it is desired to regenerate the numbers.

COMMON/VPI001/IX
IUNIF = IX (store starting value of uniform generator)
CALL PURGE3(1,1) (generate numbers)
IX = IUNIF  (reset starting value of uniform generator to its original value)

CALL PURGE3(2,1)  (generate the same numbers)

WRITE (5,15) IX  (punch BCD card with starting number for uniform random number generator)

15 FORMAT (I12)

The last value of IX is punched on a data card in integer BCD format and may be used as input in a later computer run to continue the sequence of random number generation at the point it was terminated in this run. The COMMON statement must appear in the control program whenever the uniform generator is modified. The instructions

COMMON/VPI001/IX

READ(5,15)IX

CALL PURGE3(1,1)

in the control program will continue the uniform random number sequence in a different computer run when a data card output from a prior run is used to redefine the starting value of the uniform random number generator. It should be noted that the sequence of uniform numbers generated will always be the same in different computer runs since the initial value of IX is always 21319113 unless set equal to some different arbitrary value at the beginning of each computer run.

The COMMON statement must be used whenever this is done. All sequences of uniform random numbers which begin with the same value of IX will be identical.
3.54. Variables Available Through Common

Table IV gives a description of the variables used in PURGE3 which may be placed in COMMON with the calling program.

As an example of how the variables in COMMON may be used, suppose it was desired to fit a Pearson distribution with PURGE3 and then make a test on the coefficients of skewness and kurtosis, i.e., $\beta_1$ and $\beta_2$, respectively, in the control program. The instructions

\begin{verbatim}
COMMON/MCK1/AVE(2),CM2(2),CM3(2),CM4(2),BETA1,BETA2,SKAPPA
CALL PURGE3(1,1)
X = BETA2*.333
IF(X.GT.BETAL)A = SQRT(BETA2)
\end{verbatim}

would set A equal to the square root of $\beta_2$ if $\beta_2$ was more than three times larger than $\beta_1$.

Note that the complete COMMON block had to be written even though only two variables were of interest to the calling program.

3.55. Error Conditions

In the event that an error condition develops when fitting a curve, a diagnostic message is printed on the system output, LIMIT is set to zero and the subroutine exits normally. Such error conditions arise when illegal values are obtained for $\beta_1$ and $\beta_2$ and for the Pearson equation parameters $M_1$ and $M_2$, and $A_1$ and $A_2$. The values of $\beta_1$ and $\beta_2$ are restricted to a certain domain for distribution functions. The
<table>
<thead>
<tr>
<th>Block</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VP1001</td>
<td>IX</td>
<td>Starting value of uniform generator</td>
</tr>
<tr>
<td>MCK1</td>
<td>AVE(2)</td>
<td>Mean of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>CM2(2)</td>
<td>Variance of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>CM3(2)</td>
<td>Third moment of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>CM4(2)</td>
<td>Fourth moment of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>BETA1</td>
<td>$\beta_1$ of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>BETA2</td>
<td>$\beta_2$ of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>SKAPPA</td>
<td>$\kappa$ of Pearson distribution</td>
</tr>
<tr>
<td>MCK2</td>
<td>TOM(100)</td>
<td>Array used to store generated random numbers</td>
</tr>
<tr>
<td></td>
<td>LIMIT</td>
<td>Number of random numbers to be generated per call (0-100)</td>
</tr>
<tr>
<td>MCK4</td>
<td>N(2)</td>
<td>Number of intervals for c.d.f.</td>
</tr>
<tr>
<td></td>
<td>EL(2)</td>
<td>Pearson variable used in random generation</td>
</tr>
<tr>
<td></td>
<td>H(2)</td>
<td>Pearson variable used in random generation</td>
</tr>
<tr>
<td></td>
<td>K2U(2,101)</td>
<td>Centile index for c.d.f.</td>
</tr>
<tr>
<td></td>
<td>SS(2,2641)</td>
<td>c.d.f.</td>
</tr>
<tr>
<td>Z2</td>
<td>FA(4)</td>
<td>Non-central moments calculated from generated data; FA(1) represents the mean of the generated data.</td>
</tr>
<tr>
<td>Z3</td>
<td>FAN(4)</td>
<td>Sums of first four powers of generated numbers</td>
</tr>
<tr>
<td>Z4</td>
<td>FCM2</td>
<td>Variance of generated numbers</td>
</tr>
<tr>
<td></td>
<td>FCM3</td>
<td>Third moment of generated numbers</td>
</tr>
<tr>
<td></td>
<td>FCM4</td>
<td>Fourth moment of generated numbers</td>
</tr>
<tr>
<td>Z5</td>
<td>FBETA1</td>
<td>$\beta_1$ of generated numbers</td>
</tr>
<tr>
<td></td>
<td>FBETA2</td>
<td>$\beta_2$ of generated numbers</td>
</tr>
<tr>
<td></td>
<td>FKAPPA</td>
<td>$\kappa$ of generated numbers</td>
</tr>
</tbody>
</table>
region being $\beta_1, \beta_2$ between the lines

$$8\beta_2 - 15\beta_1 - 36 = 0$$

and

$$\beta_2 - \beta_1 - 1 = 0 \text{ where } \beta_1 \geq 0.$$ 

Since the run will terminate normally under these error conditions, it is the responsibility of the programmer to test for an error condition after the first call to PURGE3 for a given distribution. The following sequence of instructions will perform such a test:

```fortran
COMMON/MCK2/TOM(100),LIMIT
.
.
CALL PURGE3(N1,N2)
IF (LIMIT.EQ.0) (error occurred, skip to next fit or stop)
.
```

This test should only be made when attempting to generate and use random numbers and not when LIMIT should be zero as is the case when only a fit is required.

The program tests for overflow conditions when computing moments from sampling data and when computing moments from the generated data. Should an overflow condition occur when computing moments from sampling data, LIMIT is set to zero and the diagnostic

"TO MUCH AND/OR TOO LARGE DATUM"

is printed. Setting LIMIT to zero is a warning to the calling program
program that PURGE3 was unable to fit a curve.

Should an overflow occur during the computation of moments from generated data, "OVERFL" is printed before printing moments as a warning that the moments from the generated data are incorrect. However, those error conditions arising when computing output moments have no effect on the graph of the generated data.

The IBM 360 system will automatically print out overflow or underflow diagnostics, but execution continues with no limit on these conditions.

3.6. Sample Computer Programs

(A) The following computer program when used with the data cards given in Figure 1 and the PURGE3 subroutine will generate Figures 3 and 4.

```plaintext
COMMON/MCK1/AVE(2) ,CM2(2) ,CM3(2) ,CM4(2) , BETA1,BETA2,SKAPPA
COMMON /MCK2/TOM(100) ,LIMIT
CALL PURGE3 (1,4) 
AVE(1) = 0.
CM2(1) = 1.0
CM3(1) = 1.7320509
CM4(1) = 9.6999999
CALL PURGE3 (3,4)
STOP
END
(DATA CARDS)
```

(B) This sample program reads the moment cards shown in Figure 2 and produces Figures 5 and 6.

```plaintext
COMMON/MCK2/TOM(100) ,LIMIT
CALL PURGE3 (1,4)
CALL PURGE3 (1,4)
STOP
END
(DATA CARDS)
```
This program shows how the PURGE3 subroutine may be used to generate and pass VIA COMMON up to two Pearson c.d.f. functions and related parameters to the MIX1 subroutine for the purpose of generating random numbers. The MIX1 subroutine is discussed in Chapter IV. See section 4.5, examples (A) and (D) for the application of the Pearson data generated.

```
COMMON/MCK1/AVE(2),CM2(2),CM3(2),CM4(2),BETA1,BETA2,SKAPPA
COMMON/MCK2/TOM(100),LIMIT
COMMON/MCK4/N(2),EL(2),H(2),K2U(2,101),SS(2,2641)
AVE(1) = 2.9090
CM2(1) = 6.26999992
CM3(1) = 10.9900
CM4(1) = 102.5000
AVE(2) = 0.5700
CM2(2) = 8.3739992
CM3(2) = 0.0260
CM4(2) = 124.4600
CALL PURGE3(3,6)
CALL PURGE3(3,7)
CALL MIX1(N1,N2) (see Chapter IV)
STOP
END
```
Figure 1. Data Point Cards.
Figure 2. Moment Card Input.
Figure 4. PEARSON CURVE TYPE FOUR

Output.
PEARSON CURVE TYPE ONE DISTRIBUTION GENERATOR

Mi=-9.82279E-01
M2=-8.78155E-91
A1= 4.60712756E-01
A2= 2.79918E-60
YE= 4.414104E-32

MOMENTS FROM ORIGINAL DATA FROM GENERATED DATA N= 109

MEAN $29 2245741463
VARIANCE -1.00000000 0.85980701

MU(3) 2223605919 -1494578557
MU(4) 6219999981 5 26396179
BETA 1 4.99999995 5295644093
BETA 2 6.19999981 7212050438
KAPPA -1.25533199 -1.49264145

Figure 5. PURGE Output.
Figure 6. PURGE3 Output.

PEARSON CURVE TYPE ONE DISTRIBUTION GENERATOR

Mi = 7.2393E-01 M2 = 2.502276 GO A1 = 1.75038E GG A2 = 3.556836 00 YE = 3.45716E-01

Moments from Original Data from Generated Data

<table>
<thead>
<tr>
<th>Mean</th>
<th>Variance</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>0.000000</td>
<td>0.330925</td>
<td>1.930715</td>
</tr>
<tr>
<td>0.25</td>
<td>0.433092</td>
<td>0.364241</td>
<td>3.200517</td>
</tr>
<tr>
<td>0.4</td>
<td>0.933092</td>
<td>0.364241</td>
<td>6.400517</td>
</tr>
<tr>
<td>0.5</td>
<td>1.433092</td>
<td>0.364241</td>
<td>9.600517</td>
</tr>
<tr>
<td>0.6</td>
<td>1.933092</td>
<td>0.364241</td>
<td>12.800517</td>
</tr>
</tbody>
</table>

Note: The table shows the moments for different values, with the mean and variance being the primary focus. The skewness and kurtosis values also increase with each subsequent moment.
CHAPTER IV

MIX1 SUBROUTINE

4.1. Introduction

The MIX1 subroutine was written to overcome the limitation, inherent in PURGE3, of being restricted to generating only unimodal Pearson distributions. This subroutine, called MIX, was previously written in FORTRAN IV to be used on the IBM 7040-1401 system, see Thomas (1966). Here, the MIX subroutine is considerably revised and improved for use as a subroutine, MIX1, on the IBM 360 System computer in FORTRAN IV. MIX1 is capable of obtaining random variables from a mixture (weighted average) of any combination of two different Pearson distributions, a uniform and a normal distribution. Under certain conditions this subroutine will generate random numbers from continuous distributions of almost any shape at the rate of about 143,000 random numbers per minute on the IBM 360. This great increase in speed of random number generation is accomplished by using large proportions of uniform and normal random variates in the mixtures since these may be generated much faster than the Pearson types. The speed of generation increases as the proportion of uniform and normal distributions in the mixture increases.

The desired Pearson distributions are generated by PURGE3 and the c.d.f. along with the necessary parameters for generating random numbers are stored in COMMON so that MIX1 may be called in the same program.
A control card specifying the proportions of the various distributions, the mean and variance of the normal distribution, and the interval of the uniform distribution may be read as data input by the control program or assigned values in the control program. The control program may later modify the proportion of distributions mixed. The MIX1 subroutine, like PURGE3, may be used to generate a sequence of random numbers from a mixture previously described in the same program.

Output options include the printing of theoretical moments along with the moments computed from the random numbers that are generated, and also a graph of the distribution which can be smoothed if desired.

4.2. Method of Calculation

The density function \( f \) of a random variable \( x \), which we wish to generate, is represented by

\[
f(x) = \sum_{i=1}^{4} a_i f_i(x)
\]

where each \( f_i \) is a density function and \( a_i \) is the probability of obtaining \( x \) from \( f_i \) where

\[
\sum_{i=1}^{4} a_i = 1, \ a_i \geq 0 \ \text{all } i.
\]

The moments of \( f \) are computed as

\[
E x^r = \int_{R} x^r f(x) \, dx = \sum_{i=1}^{4} a_i \int_{R_i} x^r f_i(x) \, dx
\]

or
The MIX1 subroutine first computes the moments about zero for each distribution, multiplies them by the corresponding probability, and sums them to obtain the non-central moments for the mix. The function statements used in computing these non-central moments appear as function subroutines. Then the second, third, and the fourth non-central moments are converted to central moments.

To generate random numbers, the MIX1 subroutine first generates a uniform \((0, 1)\) random number which determines the distribution, based on the probabilities specified in the calling program, from which to generate \(x\) with distribution \(f\). To illustrate the procedure suppose we have

\[
f(x) = .25 f_1(x) + .15 f_2(x) + .50 f_3(x) + .10 f_4(x)
\]

A uniform random variable \(u\) is generated. If \(u < .25\), \(x\) is generated from distribution \(f_1\); if \(.25 \leq u < .40\), \(x\) is generated from \(f_2\); if \(.40 \leq u < .90\), \(x\) is generated from \(f_3\); and if \(.90 \leq u < 1\), \(x\) is generated from \(f_4\). After \(x\) has been generated the above procedure is repeated 100 times to MIX1 and if required, moments are calculated from the generated numbers for comparison with their theoretical values.
4.3. Graphing a MIX1 Density

If a graph of the density function is desired, 10,000 random numbers are first generated from the required distribution to determine the range of the generated numbers. The range is then divided into 100 equally spaced intervals. Then the same sequence of random numbers are regenerated to compute the frequency in each interval.

An option is available that allows the program to smooth the data before printing the graph. This smoothing is performed by using moving averages of order five for the third through the ninety-eighth frequency intervals. These points are determined from

\[ X_i = \frac{x_i + x_{i+1} + x_{i+2}}{5} \]

where \( X_i \) represents the \( i \)th frequency interval, \( i \) equals three to ninety-eight.

To avoid dropping the first two and last two points of the curve, using this method, the first and last points remain unchanged whereas the second and 99th points are determined from

\[ X_2 = \frac{x_1 + x_2 + x_3}{3}, \quad x_{99} = \frac{x_{98} + x_{99} + x_{100}}{3}. \]

This procedure preserves the end points of J and U-shaped distributions and appears to give good results.

Using the range and summing the frequencies in each interval, the ordinate and abscissa scales are determined whence the curve is plotted. The scale is more accurately determined if smoothing is used.
4.4. Operating Instructions

The MIX1 subroutine is entered by the FORTRAN STATEMENT

CALL MIX1 (N1, N2)

where N1 and N2 are arguments which describe the options desired.
These options and the related values are given in Table V.

4.41. Data Input Option

For each fit under the N1=1 option a data CONTROL CARD is read.
The c.d.f. and other parameters for up to two Pearson distributions,
produced by PURGE3 in the same computer run, have been passed to the
MIX1 subroutine VIA the COMMON statement.

If it is desired to form a new mixture of distributions using the
same Pearson curve(s), then the next card would be a new CONTROL CARD.
If, however, one desires to fit a new mixture using different Pearson
distribution(s), then the control program would have to call PURGE3
with N2=6 and/or N2=7 before a new mixture of distributions can be
formed.

Control Card:

The CONTROL CARD format is described in Table VI. The format for
probabilities is F7.4, and is F10.5 for the other variables.

4.42. Regeneration

The MIX1 subroutine has the capability of regenerating a sequence
of random numbers or continuing from the same sequence in a different
<table>
<thead>
<tr>
<th>Value of N1</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read <strong>CONTROL CARD</strong> specifying probabilities of each distribution and generate 100 random numbers from the mixture.</td>
</tr>
<tr>
<td>2</td>
<td>Generate 100 random numbers from a mixture of distributions previously described under the N1=1 option.</td>
</tr>
<tr>
<td>3</td>
<td>Generate 100 random numbers from a mixture whose probabilities are supplied by the calling program VIA the COMMON statement.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value of N2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Summations of first four powers of all numbers generated from this mixture are kept for later calculation of sample moments.</td>
</tr>
<tr>
<td>2</td>
<td>Summations of first four powers of all numbers generated from this mixture of distributions are used to calculate sample moments which are then printed.</td>
</tr>
<tr>
<td>3</td>
<td>Same action is taken as for N2=2. Then a graph of the mixture of distributions without smoothing is printed. (Must be last option taken for a given distribution since the graph routine shares storage with the c.d.f., thus destroying the latter.)</td>
</tr>
<tr>
<td>4</td>
<td>Same as for N2=3 except that a graph with smoothing is printed.</td>
</tr>
<tr>
<td>5</td>
<td>Random numbers are generated without calculating summations for computation of moments. (Moments should not be printed for a given mixture of distributions after this option is used.)</td>
</tr>
</tbody>
</table>
Table VI. Control Card Format

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Cols.</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(1)</td>
<td>1-7</td>
<td>Probability of selection of first Pearson distribution. (The computer run may be terminated by placing a floating point number greater than one in this field.)</td>
</tr>
<tr>
<td>P(2)</td>
<td>8-14</td>
<td>Probability of selection of second Pearson distribution.</td>
</tr>
<tr>
<td>G</td>
<td>15-21</td>
<td>Probability of selection of normal distribution.</td>
</tr>
<tr>
<td>GM</td>
<td>22-31</td>
<td>Mean of normal distribution.</td>
</tr>
<tr>
<td>GV</td>
<td>32-41</td>
<td>Variance of normal distribution.</td>
</tr>
<tr>
<td>U</td>
<td>42-48</td>
<td>Probability of selection of uniform distribution.</td>
</tr>
<tr>
<td>UL</td>
<td>49-58</td>
<td>Lower endpoint of uniform interval.</td>
</tr>
<tr>
<td>UP</td>
<td>59-68</td>
<td>Upper endpoint of uniform interval.</td>
</tr>
</tbody>
</table>

Note: By using the N1=3 option the reading of the CONTROL CARD may be eliminated by specifying the data thereon directly in the calling program.
computer run. This necessitates modifying the uniform random number generator and the normal distribution generator if random numbers from the normal distribution are to be generated.

Both generators may be modified by placing the following FORTRAN statements in the calling program where IX and KX are the starting values of the uniform and normal random numbers generators, respectively, and NN represents an arbitrary format statement number:

```
COMMON/VPI001/IX/VPI002/KX
IUNIF = IX  (store starting value of uniform generator)
KNORM = KX  (store starting value of normal generator)
CALL MIX1(N1,N2)  (generate random numbers)
IX = IUNIF  (reset uniform generator to its original starting value)
KX = KNORM  (reset normal generator to its original starting value)
CALL MIX1(2,N2)  (regenerate prior sequence of numbers)
WRITE(7,NN)IX,KX  (PUNCH CARD CONTAINING STARTING VALUES OF UNIFORM AND NORMAL GENERATORS WHICH WILL CONTINUE SEQUENCE FROM THIS POINT IN A DIFFERENT COMPUTER RUN)
```

When it is desired to continue generating the same sequence in a succeeding computer run, the following FORTRAN instructions placed in the calling program will read the data card punched in a prior run and generate a sequence of random numbers, continuing from the point at which it was terminated.
COMMON/VP1001/IX/VP1002/KX

READ (5,NN)IX,KX (read data card punched in previous run which defines starting values of uniform and normal generators needed to continue sequence)

CALL MIX1(N1,N2) (continue generation)

4.43. Variables in COMMON

Through use of the FORTRAN COMMON statement, the calling program may modify the CONTROL CARD variables, obtain the random numbers generated, and improve the overall flexibility in methods of defining variables to be used in MIX1. See Table VII for the block names and variables available through COMMON. The arrays that appear in COMMON in the control program must have the exact dimensions as shown in Table VII and the order of the variables in a COMMON block may not be altered. However, the dimension of an array may be specified in either a DIMENSION statement or in the COMMON block, but not in both.

4.44. Error Conditions

Subroutine MIX1 tests the probability multipliers of the distributions to make sure they sum to unity. If the absolute difference of their sum from one is 0.001 or greater, the run is terminated after printing the diagnostic message.

"PROBABILITIES DO NOT SUM TO 1.
CORRECT AND RESUBMIT."

If overflow or underflow conditions occur when computing moments the IBM 360 System will print messages on the job standard output.
Table VII. Variables in MIX1 Available Through COMMON

<table>
<thead>
<tr>
<th>Block</th>
<th>Variable (Dimension)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPIO01</td>
<td>IX</td>
<td>Starting value of uniform generator</td>
</tr>
<tr>
<td>VPIO02</td>
<td>KX</td>
<td>Starting value of normal generator</td>
</tr>
<tr>
<td>MCK1</td>
<td>C1(2)</td>
<td>Mean of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>C2(2)</td>
<td>Variance of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>C3(2)</td>
<td>Third moment of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>C4(2)</td>
<td>Fourth moment of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>BETA1</td>
<td>$\beta_1$ of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>BETA2</td>
<td>$\beta_2^*$ of Pearson distribution</td>
</tr>
<tr>
<td></td>
<td>SKAPPA</td>
<td>$\kappa$ of Pearson distribution</td>
</tr>
<tr>
<td>MCK2</td>
<td>TOM(100)</td>
<td>Storage array for the generated random numbers</td>
</tr>
<tr>
<td>LIMIT</td>
<td></td>
<td>Number of random numbers to be generated per call (0-100)</td>
</tr>
<tr>
<td>MCK3</td>
<td>P(2), G, GM, GV, U, UL, UP</td>
<td>Control Card variables in the order in which they appear on it</td>
</tr>
<tr>
<td>MCK4</td>
<td>N(2) EL(2) H(2) KU(2,101) S(2,2641)</td>
<td>Number of intervals of the Pearson c.d.f. Centile index for c.d.f. c.d.f.</td>
</tr>
<tr>
<td>Z1</td>
<td>TM1</td>
<td>Theoretical mean of mixture</td>
</tr>
<tr>
<td></td>
<td>CM2</td>
<td>Theoretical variance of mixture</td>
</tr>
<tr>
<td></td>
<td>CM3</td>
<td>Theoretical third moment of mixture</td>
</tr>
<tr>
<td></td>
<td>CM4</td>
<td>Theoretical fourth moment of mixture</td>
</tr>
<tr>
<td>Z2</td>
<td>FA(4)*</td>
<td>Non-central moments calculated from generated data; FA(1) is the mean of the generated data</td>
</tr>
<tr>
<td>Z3</td>
<td>FAN(4)*</td>
<td>Sums of first four powers of generated numbers</td>
</tr>
<tr>
<td>Z4</td>
<td>FCM2*</td>
<td>Variance of generated distribution</td>
</tr>
<tr>
<td></td>
<td>FCM3*</td>
<td>Third moment of generated distribution</td>
</tr>
<tr>
<td></td>
<td>FCM4*</td>
<td>Fourth moment of generated distribution</td>
</tr>
</tbody>
</table>

*Not calculated for N2=5 option.
indicating that these conditions occurred. The moments generated from the data are incorrect. However, the random numbers generated are correct.

4.5. Sample Computer Programs

(A) The following computer program is an example of the MIX1 subroutine generating a mixture from two (2) Pearson distributions, a $N(0,2)$, and a uniform distribution on the interval $(-1,0)$. Each distribution occurs in equal proportion, i.e., a mixture of 25% for each distribution. The random numbers and the number of random numbers generated prior to graphing the distribution are returned to the control program. The Pearson curves were fitted in section 3.6, example (C).

```fortran
COMMON/MCK1/AVE(2),CM2(2),CM3(2),CM4(2),BETA1,BETA2,SKAPPA
COMMON/MCK2/TOM(100),LIMIT
COMMON/MCK4/N(2),EL(2),H(2),K2U(2,101),SS(2,2641)
CALL PURGE3(3,6)
CALL PURGE3(3,7)
CALL MIX1(1,4)
STOP
END
```

Figure 7 is a graph of 10,000 random numbers generated from this mixture of distributions with smoothing. Figure 8 is a graph of the same data without smoothing.

(B) This sample program shows how the control program may be used to assign values to CONTROL CARD variables in MIX1.* The graph

*Note that all CONTROL CARD variables are defined.
Figure 7. MIXTURES WITH SMOOTHING.

[Data Table]

<table>
<thead>
<tr>
<th>135.979</th>
<th>52.88453017</th>
<th>135.979</th>
<th>52.88453017</th>
</tr>
</thead>
<tbody>
<tr>
<td>135.979</td>
<td>52.88453017</td>
<td>135.979</td>
<td>52.88453017</td>
</tr>
<tr>
<td>0.35F</td>
<td>0.37F</td>
<td>0.35F</td>
<td>0.37F</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>0.728F</td>
<td>0.728F</td>
<td>0.728F</td>
<td>0.728F</td>
</tr>
</tbody>
</table>

DISTRIBUTION MIXTURES
of this mixture of 35% $N(0,1)$ and 65% uniform on $(-1,2)$ appears as Figure 9 where smoothing was desired.

\[
\begin{align*}
\text{COMMON/MCK3/P}(2),G,GM,GV,U,UL,UP \\
\text{COMMON/MCK2/TOM}(100),\text{LIMIT}
\end{align*}
\]

\[
\begin{align*}
P(1) &= 0. \\
P(2) &= 0. \\
G &= 0.35 \\
GM &= 0. \\
GV &= 1.0 \\
U &= 0.65 \\
UL &= -1.0 \\
UP &= 2.0 \\
\text{CALL MIX1}(3,4) \\
\text{STOP} \\
\text{END}
\end{align*}
\]

This sample program shows a mixture being generated from a single Pearson distribution with probability 0.5, a $N(0,2)$ and a uniform distribution on the interval $(0,3.5)$ with probabilities 0.3 and 0.2, respectively. The graph of the mixture appears as Figure 10 from the 10,000 random numbers generated. The smoothing option was selected.

\[
\begin{align*}
\text{COMMON/MCK1/AVE}(2),CM2(2),CM3(2),CM4(2),\text{BETA1,BETA2,SKAPPA} \\
\text{COMMON/MCK2/TOM}(100),\text{LIMIT} \\
\text{COMMON/MCK4/N}(2),\text{EL}(2),H(2),\text{K2U}(2,101),\text{SS}(2,2641) \\
\text{COMMON/VPI001/I}X/\text{VPI002/K}X \\
\text{AVE}(1) &= 0. \\
CM2(1) &= 1.0 \\
CM3(1) &= 1.7320509 \\
CM4(1) &= 7.800 \\
I\text{UNIF} &= IX \\
K\text{NORM} &= KX \\
\text{CALL PURGE3}(3,6) \\
\text{CALL MIX1}(1,4) \\
IX &= I\text{UNIF} \\
KX &= K\text{NORM} \\
\text{CALL MIX1}(1,4) \\
\text{STOP} \\
\text{END}
\end{align*}
\]
Figure 9. MIX1 Output With Smoothing.

**CONTE**

**CAL**

0.23E

0.2155

**CARH**

**P**=

tell

D**=**

dag

**UNTE**=

S500

FROM

~1.000

MOMENTS

MEAN

VA2TANCE

MU(3)

MU(4)

wo

~

we

eat

oC

Figure

9.

MIX1

Output

With

Smoothing.

**NORM**=

ia

3

TO

2 D008

QRIGINAL

DATA

9232499995 DB943749T —9209384376

2e8

2402878

where

+O1-O.4199F ©1-0.134F

+OL-9.691LF Co

+04422

NISTRISUTION

MIXTURE

mor Loe Vy

hi

GENERATED

DATA

h

231461799 Je

89124943

-2.99129476

12986543644

W

fie

RK oie ok ok

**N**

RK

BR

RR

EK

A

9.606E

60

0.125E

C1

0.190E

O1

0.255E

01

9.606E

20278

9.799976

-0.9960

0.9960

VA2TANCE

MEAN

Degrees

FROM

OBSERVED DATA

N = 10000

**UN** = 0.5500 FROM -1.0000 TO 2.0000

Original GCD

PI = 0.02

Q = 0.0

NORM = 0.5500 MEAN = 0.0

A = 0.0

VA2TANCE

DISTRIBUTION

**M**

**R**

**E**

**A**

**N**

**U**

**N**

**I**

**M**
Figure 10. MIX Output with Smoothing.
The second call to MIX1 creates a new mixture representing a $N(0,1)$ and a uniform distribution on the interval $(-1,1)$. The data CONTROL CARDS were read within the MIX1 subroutine for the first and second calls to MIX1. The output graph of the last distribution is shown in Figure 11.

(D) Using the Pearson distribution described in section 4.4, example (A), this example shows the use of MIX1 to generate random numbers only. Ten sequences of random numbers are generated. The control program reads the CONTROL DATA card.

```
COMMON/MCK4/N(2),EL(2),H(2),K2U(2,101),SS(2,2641)
COMMON/MCK1/AVE(2),CM2(2),CM3(2),CM4(2),BETA1,BETA2,SKAPPA
COMMON/MCK2/TOM(100),LIMIT
COMMON/MCK3/(2),G,GM,GV,U,UL,UP
CALL PURGE3(3,6)
CALL PURGE3(3,7)
READ(5,10)P(1),P(2),G,GM,GV,U,UL,UP
10 FORMAT(3F7.4,2F10.5,F7.4,2F10.5)
CALL MIX1(3,5)
DO 20 I=1,9
   CALL MIX1(2,5)
20 CONTINUE
STOP
END
```

The N2=5 in MIX1 option generates random numbers faster than any other option because no moments are computed.
CHAPTER V

MIX2 SUBROUTINE

5.1. Introduction

The MIX2 subroutine is a restricted version of the MIX1 subroutine. MIX2 was written to generate random numbers as its sole function, eliminating the options to compute and print moments and to graph the distribution of the random numbers generated. Limited to this exclusive function, MIX2 can generate from 113,000 to 143,000, depending on the mix, random numbers per minute on the IBM 360 system computer.

When the mixture to be generated is to consist of Pearson distributions, MIX2 differs from MIX1 in the method of input of the Pearson distribution data. The desired Pearson distributions are generated by PURGE3, with N2 equal 6 or 7, and cards containing the c.d.f. along with the necessary parameters for generating random numbers are punched in BCD for later input to MIX2 in a different computer run. Whereas MIX1 was passed the Pearson distribution data VIA COMMON blocks in the same program that PURGE3 generated the distribution, MIX2 must read the Pearson data as standard input, preceded by a CONTROL CARD specifying the properties of the various distributions to be mixed. This CONTROL CARD was described in the MIX1 subroutine. See Table VI.

5.2. Method and Operating Instructions

The method of generating random numbers is identical to that given in section 4.2 for the MIX1 subroutine.
The MIX2 subroutine is entered by the FORTRAN statement

\texttt{CALL MIX2(N1)}

where \texttt{N1} is an argument described in Table VIII. Note that no output option is required since MIX2 generates random numbers without any other computations. This is equivalent to setting \texttt{N2=5} in the MIX1 subroutine.

Under the \texttt{N1=1} option, if there is a number greater than zero but less than one in \texttt{P(1)} or \texttt{P(2)}, then a Pearson deck produced by PURGE3 must follow the \texttt{CONTROL CARD}. If \texttt{P(1)} and \texttt{P(2)} are non-zero then two such decks must follow the \texttt{CONTROL CARD}. If more than one fit per run is desired, the next card read will be a new \texttt{CONTROL CARD} followed by the necessary Pearson decks, decks obtained previously by a call to PURGE3 with \texttt{N2=6} or 7.

5.21. \textbf{Regeneration}

MIX2 may regenerate a sequence of random numbers or continue from the same sequence in a different computer run by using the technique described for the MIX1 subroutine in section 4.5.

5.22. \textbf{Variables in Common}

The variables in MIX2 that are available in COMMON are shown in Table IX.

5.23. \textbf{Error Condition}

The MIX2 subroutine tests the probability multipliers of the
Table VIII. Values of the Argument for MIX2

<table>
<thead>
<tr>
<th>Value of NL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Read CONTROL CARD and if required data cards for Pearson distribution(s). Generate 100 random numbers from the mixture.</td>
</tr>
<tr>
<td>2</td>
<td>Generate 100 random numbers from a mixture of distributions previously described under the N1=1 option.</td>
</tr>
</tbody>
</table>
Table IX. Variables in MIX2 Available Through Common

<table>
<thead>
<tr>
<th>Block</th>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPI001</td>
<td>IX</td>
<td>Starting value of uniform generator</td>
</tr>
<tr>
<td>VPI002</td>
<td>KX</td>
<td>Starting value of normal generator</td>
</tr>
<tr>
<td>MCK2</td>
<td>TOM(100)</td>
<td>Generated random numbers</td>
</tr>
<tr>
<td>LIMIT</td>
<td></td>
<td>Number of random numbers to be generated per call (0-100)</td>
</tr>
<tr>
<td>MCK3</td>
<td>P(2),G,GM, GV,U,UL,UP</td>
<td>CONTROL CARD variables in the order in which they appear on it</td>
</tr>
<tr>
<td>MCK4</td>
<td>N(2)</td>
<td>Number of intervals of the Pearson c.d.f.</td>
</tr>
</tbody>
</table>
| EL(2)  |          | Pearson variable used in random number genera-
|        |          | tions                                           |
| H(2)   |          | Pearson variable used in random                  |
| KU(2,101) |        | Centile index for c.d.f.                         |
| S(2,2641) |        | c.d.f.                                           |
distributions to make sure they sum to unity. The run is terminated if their sum differs from one by as much as 0.001 in absolute value. If this condition exists, LIMIT is set to zero and may be tested in the control program to detect that no random numbers were generated. The subroutine exits after printing the diagnostic message.

"PROBABILITIES DO NOT SUM TO 1. CORRECT AND RESUBMIT."

5.3. Sample Computer Program

The following computer program when used with the data shown in Figure 12 will generate 100 random numbers from a mixture of two Pearson distributions and a uniform distribution on the interval (-2,1). Another sequence of 100 random numbers is generated from the same mixture.

COMMON/MCK2/X(100),LIMIT

CALL MIX2(1)
CALL MIX2(2)
STOP
END

Note that the control program will have the random numbers returned in array X.
Figure 12. Input for MIX2.


APPENDIX A

Listing of Uniform and Normal Random Number Generators Used
UNIFORM RANDOM NUMBER GENERATOR

FUNCTION RDM(X)
COMMON /VPI001/ IX
IY = IX*65539
IF ( IY ) 1,2,2
  1 IY = IY+ 2147483647 +1
  2 YFL = IY
        RDM = YFL * .4656613E-9
        IX = IY
        RETURN
        END

NORMAL RANDOM NUMBER GENERATOR

FUNCTION RNOR (X)
COMMON /VPI002/ IX
A = 0.0
DO 3 I=1,12
  IY = IX*65539
  IF(IY) 1,2,2
  1 IY = IY+ 2147483647 +1
  2 YFL = IY
        IX = IY
  3 A = A + YFL * .4656613 E-9
      RNOR = A-6.0
    RETURN
    END
APPENDIX B

Flowchart and Listings for PURGE3
Flow Chart for PURGE3.
Flow Chart for PURGE3 (Cont.)
Flow Chart for PURGE3 (Cont.)
Flow Chart for PURGE3 (Cont.)
Flow Chart for PURGE3 (Cont.)
Flow Chart for PURGE3 (Cont.)
Store Data ifor First Curve for MIX1

Yes

Store Data for Second Curve for MIX1

Yes

\[ \Sigma (\text{Rand No})^2 = 0 \]

No

Write Moments for Pearson Distribution

Yes

Write Both Sets of Moments

No

Calculate Sample Moments

Yes

N2 = 4

No

N2 = 6

Yes

Store Data for First Curve for MIX1

Yes

Flow Chart for PURGE3 (Cont.)
Flow Chart for PURGE3 (Cont.)
SUBROUTINE PURGER (MKX, MKZ)
C
UNIVERSAL RANDOM DISTRIBUTION GENERATOR
DOUBLE PRECISION AN, X, Z, W
COMMON/VPI001/IY
DIMENSION SKUNK(5), FRM(12)
DIMENSION FM(18), FMH(5), S(2641), KU(101), FA(4), FAH(4)
COMMON/MCK1/AVE(2), CM2(2), CM3(2), CM4(2), BETA1, BETA2, SKAPPA
COMMON/MCK2/TOM(100), LIMIT
COMMON/MCK4/H(2), EL(2), H(2), K2U(2,101), SS(2,2641)
COMMON/Z2/FBETA1, FBETA2, FKAPPA
DIMENSION CN(8), TN(8), A(4), AN(4), X(30), O(100), VAL(100), NA(100)
DIMENSION BOT(100), SID(5)
EQUIVALENCE (S,0), (S(101), VAL), (S(201), NA), (S(301), BOT), (S(401), S1)
EQUIVALENCE (A,KU(61)), (AN,KU(69)), (CN,KU(77)), (TN,KU(85)), (FRM, IKU(93))
REAL LOGF, LGAME, NORMAL
DATA FMH(1)/4H(4X, /)
DATA FMH(2)/4H4F17 /
DATA FMH(3)/4H4H, 8, 6 /
DATA FMH(4)/4H2X, I2 /
DATA FMH(5)/4H2 /
DATA BLANK/4H /
DATA ASTRIC/4H* /

C
FUNCTIONS DESCRIBING EACH EQ
ATANF(P) = ATAN(P)
COSF(P) = COS(P)
EXPF(P) = EXP(P)
LOGF(P) = ALOG(P)
LGAME(P) = ALGAMA(P)
SORTF(P) = SORT(P)
ABS(P) = ABS(P)
FLDATE(L) = FLDATE(L)
SIGNF(P,Y) = SIGN(P,Y)
PAIRF(Y0, B1, B2, D1, D2, P) = Y0*(1.0+B1)**D1*(1.0-P/B2)**D2

PURGE001
PURGE002
PURGE003
PURGE004
PURGE005
PURGE006
PURGE007
PURGE008
PURGE009
PURGE010
PURGE011
PURGE012
PURGE013
PURGE014
PURGE015
PURGE016
PURGE017
PURGE018
PURGE019
PURGE020
PURGE021
PURGE022
PURGE023
PURGE024
PURGE025
PURGE026
PURGE027
PURGE028
PURGE029
PURGE030
PURGE031
PURGE032
PURGE033
PURGE034
PURGE035
PURGE036
C

\[ \text{PERIV} \left( Y_0, V, R, D, P \right) = \exp \left( Y_0 - D \times \log f \left( 1. + \left( \frac{P}{B-V/R} \right)^2 \right) \right) - V \times \tan \left( \frac{P}{B-R} \right) \]

\[ \text{PERIV} \left( Y_0, V, R, D, P \right) = Y_0 \times \exp \left( \frac{-P^2}{C} \right) \]

\[ \text{PERIV} \left( Y_0, B_1, B_2, D_1, D_2, P \right) = \exp \left( Y_0 + D_2 \times \log f \left( 1. + \frac{P}{B_2} \right) - D_1 \times \log f \left( 1. + \frac{P}{B_1} \right) \right) \]

\[ \text{PERIV} \left( Y_0, G, P, A, U \right) = Y_0 \times \exp \left( P \times \log f \left( 1. + \frac{U/A}{G} \right) - G \times U \right) \]

\[ \text{NORM} = 0.0 \]

\[ \text{N1} = \text{MKX} \]

\[ \text{N2} = \text{WKZ} \]

\[ \text{N3} = 1 \]

\[ \text{IF} (\text{N2} = 0.7) \text{ N3} = 2 \]

\[ \text{IF} (\text{N1} = 0.2) \text{ GO TO 696} \]

\[ \text{L} = 0 \]

\[ \text{LIMIT} = 100 \]

\[ \text{NTL} = 0 \]

\[ \text{NDOTP} = 6 \]

\[ \text{NINTP} = 5 \]

\[ \text{L2} = \text{NINTP} \]

\[ \text{L3} = \text{NDOTP} \]

\[ \text{CALL OVLWP(I)} \]

\[ \text{IF} (\text{N1} = 0.3) \text{ GO TO 664} \]

\[ \text{READ} \times \text{CONTROL CARD} \]

\[ \text{READ} \left( \text{L2}, \text{I} \right) \text{IDENT, NUMB, LIMIT, NORM, ITT, NDOT} \]

\[ \text{IF} (\text{IDENT} = 999) \text{ 604, 605, 604} \]

\[ \text{L2} = \text{NINTP} \]

\[ \text{L3} = \text{NDOTP} \]

\[ \text{PURGE}037 \]

\[ \text{PURGE}039 \]

\[ \text{PURGE}042 \]

\[ \text{PURGE}044 \]

\[ \text{PURGE}046 \]

\[ \text{PURGE}050 \]

\[ \text{PURGE}051 \]

\[ \text{PURGE}052 \]

\[ \text{PURGE}053 \]

\[ \text{PURGE}054 \]

\[ \text{PURGE}055 \]

\[ \text{PURGE}056 \]

\[ \text{PURGE}057 \]

\[ \text{PURGE}058 \]

\[ \text{PURGE}059 \]

\[ \text{PURGE}060 \]

\[ \text{PURGE}061 \]

\[ \text{PURGE}062 \]

\[ \text{PURGE}063 \]

\[ \text{PURGE}064 \]

\[ \text{PURGE}065 \]

\[ \text{PURGE}066 \]

\[ \text{PURGE}067 \]

\[ \text{PURGE}068 \]

\[ \text{PURGE}069 \]

\[ \text{PURGE}070 \]

\[ \text{PURGE}071 \]

\[ \text{PURGE}072 \]
WRITE (L3,9022)
9022 FORMAT (1H1, 92X,22H DISTRIBUTION GENERATOR)
      DO 600 I=1,2641
  600   S(I)=0.
      IF(N1.NE.3) GO TO 2080
      N1=1
      GO TO 2081
C
C
2080   IF (ITT) 2,2,3
      2   ITT=L2
      3   IF (NOUT) 4,4,5
      4   NOUT=L3
C
   5   IF(IDENT) 40,8,6
C
   6   READ (L2,7)(FM(I),I=1,18)
    7   FORMAT(18A4)
      GO TO 10
C
  8   CONTINUE
      FM(1) = FMH(1)
      FM(2) = FMH(2)
      FM(3) = FMH(3)
      FM(4) = FMH(4)
      FM(5) = FMH(5)
      IDENT=4
C
C
10   IF (NUMB) 11,11,12
    11   NUMB=32000
     12   M=3
C
         AN(1)=0.
      AN(2)=0.
      AN(3)=0.

DETERMINE OPTIONS
TAPES

TYPE OF INPUT
DATA POINT

BUILT IN FORMAT

NUMBER OF DP CARDS GIVEN
OR CALCULATED

INITIALIZE SUMS

PURGE073
PURGE074
PURGE075
PURGE076
PURGE077
PURGE078
PURGE079
PURGE080
PURGE081
PURGE082
PURGE083
PURGE084
PURGE085
PURGE086
PURGE087
PURGE088
PURGE089
PURGE090
PURGE091
PURGE092
PURGE093
PURGE094
PURGE095
PURGE096
PURGE097
PURGE098
PURGE099
PURGE100
PURGE101
PURGE102
PURGE103
PURGE104
PURGE105
PURGE106
PURGE107
PURGE108
AN(4)=0.
READ DATA POINT CARDS
READ (ITT,FM)(X(I),I=1,IDENT),LST
IF (LST) 14,14,20
M=M+IDENT
CALCULATE SUMS
DO 17 I=1,IDENT
AN(1)=AN(1)+X(I)
Z=X(I)*X(I)
AN(2)=AN(2)+Z
L=Z*X(I)
AN(3)=AN(3)+L
AN(4)=AN(4)+X(I)*L
CALL OVERFL(J3K)
GO TO (780,16,16),J3K
LIMIT=0
NUMB=NUMB-IDENT
IF (NUMB) 25,25,13
M=M-IDENT+LST
GO TO 27
M=M+NUMB
Determine of DP Cards
GO TO 30
CN=M
IF (LIMIT.EQ.0) GO TO 910
HOW MANY DP
DO 30 K=1,4
A(K)=AN(K)/M
Z=A(1)*A(1)
CM2(N3)=A(2)-Z
W=A(1)*A(2)
Z=Z*A(1)
CM2(N3)=A(3)-3.*W+2.*Z
CM4(N3)=A(4)-4.*A(1)*A(3)+6.*A(1)*W-3.*A(1)*Z
AVF(N3)=A(1)
SCM2 = SORT(CM2(N3))
Form Moments
GO TO 30
14
20
25
27
30
CALCULATE COEFF FOR PEARSON

35 TP = CM2(N3)*CM2(N3) 
BETA1 = CM3(N3) ##2/(CM2(N3) * TP) 
BETA2 = CM4(N3)/TP 

C DETERMINE IF BETA1 AND BETA2 ADMISSIBLE 
IF(((BETA1>BETA2+1.)/(L.141213562.*LT.*.0.)) OR (BETA1>LT.0.)) OR 
(15.*BETA1-3.*BETA2+36.)/(-17.)*GE.0.) GO TO 750 
DOP=0. 
SK1=BETA1*(BETA2+3.)##2 
SK2=4.*((BETA2-3.*BETA1)*(2.*BETA2-3.*BETA1-6.)) 
IF (SK1) 351, 621, 351 
351 IF (SK2) 352, 353, 352 
352 SKAPPA=SK1/SK2 
GO TO 45 
621 SKAPPA=C. 
GO TO 45 
353 SKAPPA=(10,##10)##10 
IF (BETA1-4.) 8008, 8007, 8008 
8007 WRITE (L3, 8006) 
8006 FORMAT (IH, 35X, 17HSUBCLASS TYPE TEN) 
DOP=1. 
GO TO 45 
8008 WRITE (L3, 8009) 
8009 FORMAT (IH, 35X, 10HSUBCLASS TYPE THREE) 
DOP=1. 
GO TO 45 

C READ MOMENTS CARD IF NECESSARY

40 READ (ITT, 3023) AVE(N3), CM2(N3), CM3(N3), CM4(N3) 
3023 FORMAT (4X, 4F17.8) 
2031 CM2 = SORT(CM2(N3)) 
GO TO 35 

C DETERMINE MAIN TYPE 
OR FORCE FIT

45 IF (NORM-1) 451, 50, 452
451 IF (DOP) 453,453,9400
453 IF (SKAPPA) 50,8001, 8002
8001 B83=BETA2-3.0
   IF (B83.LT. .3001 .AND. B83 .GT. -.3001) GO TO 90
   IF (BETA2-3.) .9000,90,3003
8002 IF (SKAPPA-1.) 71,8005,111
8004 WRITE (L3,8010)
8010 FORMAT (1H ,35X,17HSUBCLASS TYPE TWO)
   GO TO 60
8003 WRITE (L3,8011)
8011 FORMAT (1H ,35X,19HSUBCLASS TYPE SEVEN)
   GO TO 71
8005 WRITE (L3,8012)
8012 FORMAT (1H ,35X,18HSUBCLASS TYPE FIVE)
   GO TO 111
C
C TYPE ONE CONSTANTS
C
50 WRITE (L3,51)
51 FORMAT (1H ,30X,2?HPEARSON CURVE TYPE ONE)
511 R=.6,*(BETA2-BETA1-1)/(6.+3.*BETA1-2.*BETA2)
   TP= R*(R+2.)*SORTF(BETA1/(BETA1*(R+2.))**2+16.*R+1.))
   RP=.5*(R-2.+TP)
   RM=.5*(R-2.-TP)
53 IF (CM3=-1))57,57,56
56 D2=RP
   D1=RM
   GO TO 58
57 D1=RP
   D2=RM
58 SCOTT=,8*PROMP2=SQRTE(BETA1*(R+2.))**2+16.*R+1.))
   R2=SCOTT*(D2+1.)/(D2+D1+2.)
   R1=SCOTT-R2
   IF ((D1+1.)#(D2+1.)) 922,923,582
582 IF (R1+R2) 583,936,583
PURGE181
PURGE182
PURGE183
PURGE184
PURGE185
PURGE186
PURGE187
PURGE188
PURGE189
PURGE190
PURGE191
PURGE192
PURGE193
PURGE194
PURGE195
PURGE196
PURGE197
PURGE198
PURGE199
PURGE200
PURGE201
PURGE202
PURGE203
PURGE204
PURGE205
PURGE206
PURGE207
PURGE208
PURGE209
PURGE210
PURGE211
PURGE212
PURGE213
PURGE214
PURGE215
PURGE216
\[ Y_0 = \operatorname{SIGN}(1.0, B1+B2) \times \exp((D1 \times \log(1+0.1) - \log(\text{ABSF}(B1+B2)) - \log(\text{GAM}(D1 \times \text{PURGE}(217) + 0.1)) + D2 \times \log(0.1) - \log(\text{GAM}(D2+0.1)) + \log(\text{GAM}(D1+D2+2) - (D1+D2) \times \log(\text{GAM}(D1+D2+2))\} \]

585 WRITE (L3, 581) D1, D2, B1, B2, Y0

581 FORMAT (4H4M) = 1PE12.5, 4H M2 = 1PE12.5, 4H A1 = 1PE12.5, 4H A2 = 1PE12.5

61 H(N3) = (-B1-B2) / 5120.

UPPER = -B1

EL(N3) = B1

GO TO 145

63 IF (B1) 906, 64, 64

64 H(N3) = (B2+B1) / 5120.

UPPER = B2

EL(N3) = -B1

GO TO 145

C

JAB = 1

IF (B2) 60, 906, 63

C

RANGES

TYPE FOUR CONSTANTS

C

WRITE (L3, 72)

FORMAT(1H, 38X, 23HP Earson Curve Type Four)

R = 6.0 * (BETA2-BETA1-1.0) / (2.0 * BETA2-3.0*BETA1-6.0)

D = 0.5 = (R+2.0)

V = P * (R-1.0) / (SORTF(BETA1/SORTF(16.0*(R-1.0)-BETA1*(R-2.0))**2)

R = SORTF((CM2(N3)/16.0) * SORTF(16.0*(R-1.0)-BETA1*(R-2.0))**2)

IF (CM2(N3)) 77, 78, 793

V = -V

PHI = ATAN(V/P)
78  TOP=  ((COSF(PHI))**2/(3.*R)-1.)/(12.*R)-PHI*V)                PURGE253
    Y0=LOGF( .3989423)-LOGF(B)+.5*LOGF(R)+TOP-(R+1.)*LOGF(COSF(PHI))  PURGE254
    IF (Y0<=98.9) 782, 782, 784
782  Y1=EXPF(Y0)                                                  PURGE255
    GO TO 735                                                      PURGE256
784  Y1=0.0000000000 34                                              PURGE257
785  WRITE                                      (L3,781)R,D,V,B,Y1   PURGE258
781  FORMAT (3HID=',1PE12.5,4H M=',1PE12.5,4H V=',1PE12.5,3H A=',1PE12.5) PURGE259
      24H Y0=',1PE12.5)                                      PURGE260
      JAB=2                                                   PURGE261
      H(N3)=RCM2 /512.                                        PURGE262
      UPERP=5.0*RCM2                                          PURGE263
      EL (N3)=-UPPERP                                         PURGE264
      Y1=R                                                   PURGE265
      V2=V                                                   PURGE266
      V3=R                                                   PURGE267
      V4=0                                                   PURGE268
      GO TO 145                                             PURGE269
      PURGE270

C
      NORMAL CURVE CONSTANTS                                    PURGE271
C
92  WRITE                                      (L3,91)                       PURGE272
91  FORMAT(1H ,10X,29H THIS IS A NORMAL DISTRIBUTION)     PURGE273
92  C = CM2(N3)*2.                                           PURGE274
      Y0=1.0/SORTF(6.283185 *CM2(N3))                       PURGE275
      WRITE                                      (L3,1101)C,Y0         PURGE276
1101 FORMAT (3HOC=',1PE12.5,4H Y0=',1PE12.5)               PURGE277
      JAB=3                                                  PURGE278
      H(N3)=RCM2 /512.                                        PURGE279
      UPERP=5.0*RCM2                                          PURGE280
      EL (N3)=-UPPERP                                         PURGE281
      Y1=C                                                   PURGE282
      GO TO 145                                             PURGE283
C
      TYPE SIX CONSTANTS                                        PURGE284
C
WRITE  (L3,112)  
112 FORMAT('LH ,30X,22HPEARSON CURVE TYPE SIX',L3,113)  
113 *:=(R2-BETA2-BETA1-1.)/(R2+3.*BETA1-2.*BETA2)  
CAT=*RCM2  
*SORTF(BETA1*(R+2.)**2+16.*(R+1.))  
IF(CM3(N3)) 9836, 9837, 9838  
9836 CAT=-CAT  
9837 TERM=R**((R+2.)/2.)*SORTF(BETA1/(BETA1*(R+2.))**2+16.*(R+1.))  
D2=((R2-1.)/2.+TERM  
D1=-((R2-1.)/2.*TERM  
TP=  
B1=CAT*(D1-1.)/TP  
B2=CAT*(D2+1.)/TP  
C2=1.  
Y0=(D2*LOGF(D2+1.)-LGAMF(D2+1.)-LOGF(CAT)+(D1-2.)*LOGF(D1-D2-2.))-DPURGE302  
21*LOGF(D1-1.)+LGAMF(D1)-LGAMF(D1-D2-1.)+LOGF(CA1)  
Y1=EXPF(Y0)  
WRITE  (L3,1134)D2,D1,B1,B2,Y1  
1134 FORMAT('4H002=,1PE12.5,4H 01=,1PE12.5,4H A1=,1PE12.5,4H A2=,1PE12.5')  
25,4H 1F=,1PE12.5')  
JA3=4  
V1=B1  
V2=B2  
V3=01  
V4=02  
IF (B2 ) 115,806,120  
115 H(N3)=(-32+5.*RCM2 )/5120.  
UPPER=-B2  
ELSE (N3) =-5.*RCM2  
GO TO 145  
120 UPPER=5.*RCM2  
H(N3)=(B2+UPPER )/5120.  
ELSE (N3) =-B2  
GO TO 145  

c  

c  

c  

c  

TYPE THREE CONSTANTS  

PURGE289  
PURGE290  
PURGE291  
PURGE292  
PURGE293  
PURGE294  
PURGE295  
PURGE296  
PURGE297  
PURGE298  
PURGE299  
PURGE300  
PURGE301  
PURGE302  
PURGE303  
PURGE304  
PURGE305  
PURGE306  
PURGE307  
PURGE308  
PURGE309  
PURGE310  
PURGE311  
PURGE312  
PURGE313  
PURGE314  
PURGE315  
PURGE316  
PURGE317  
PURGE318  
PURGE319  
PURGE320  
PURGE321  
PURGE322  
PURGE323  
PURGE324
G=2.*CM2(N3) /CM3(N3)
P=4./BFA1-1.
A1=(P+1.)/6
Y0=G*(P+1.)*P/EXPF(P+1.)*LGAMF(P+1.)
WRITE (L3,9401)G,P,A1,Y0

9401 FORMAT (3HDG=,1PE12.5,3H P=,1PE12.5,3H A=,1PE12.5,4H YE=,1PE12.5)
JAB=5
V1=G
V2=P
V3=A1
IF (A1) 9402,9403,9403
9402 H(N3) =(-A1+5.*CM2 )/5120.
UPPER=-A1
EL(N3)=-5.*CM2
GO TO 145
9403 UPPER=5.*CM2
H(N3)=(A1+UPPER )/5120.
EL(N3)=-A1
C
C
C

950 FORMAT(1H)
145 V0=Y0
V6=EL(N3)
V7=UPPER
951 DO 146 KK=1,4
146 FAN(KK)=0.
 IF (LIMIT) 148,149,147
147 S(1)=0.
153 GO TO (200,250,300,350,400),JAB
C
C
C

TYPE ONE INTEGRATION

200 IF (D1) 2054,2055,2055
2055 IF (D2) 2056,2057,2057
2057  EXVAL=EL(N3)
      ADD1=EL(N3)+H(N3)
      TP = H(N3)+H(N3)
      XVAL = EL(N3)+TP
      I=2
2055  SKUNK(1)=PEAR IF(Y0,B1,B2,D1,D2,EXVAL)
2051  SKUNK(2)=PEAR IF(Y0,B1,B2,D1,D2,ADD1)
2052  SKUNK(3)=PEAR IF(Y0,B1,B2,D1,D2,EXVAL)
2053  S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)
      IF (XVAL+TP-UPPER)210,210,385
210   EXVAL=XVAL
      ADD1 = XVAL + H(N3)
      XVAL=XVAL+TP
      I=I+1
      GO TO 205

C    TYPE ONE L SHAPE
C
2054  IF (D2) 7001,3001,3001
3001  H(N3)=(UPPER-EL(N3))/5121.
      EL(N3)=EL(N3)+H(N3)
      EXVAL=EL(N3)
      ADD1=EL(N3)+H(N3)
      TP = H(N3)+H(N3)
      XVAL = EL(N3)+TP
      S(?)=0.
      I=3
3002  SKUNK(1)=PEAR IF(Y0,B1,B2,D1,D2,EXVAL)
      SKUNK(2)=PEAR IF(Y0,B1,B2,D1,D2,ADD1)
      SKUNK(3)=PEAR IF(Y0,B1,B2,D1,D2,EXVAL)
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)
      IF (XVAL+TP-UPPER)3003,3003,3004
3003  EXVAL=XVAL
      ADD1=XVAL+H(N3)
      XVAL=XVAL+TP
      I=I+1
      GO TO 3002
3004  \( S(2) = 3 \times H(N3) - S(1) \)
4004  \( N(N3) = I \)
\[ K = N(N3) \]
\[ \text{DO 4006 I=3,KI} \]
4006  \( S(I) = S(I) + S(2) \)
\[ I = N(N3) \]
\[ \text{GO TO 385} \]

C

C

7001  \( H(N3) = (\text{UPPER} - \text{EL}(N3)) / 5122 \)
\[ \text{EL}(N3) = \text{EL}(N3) + H(N3) \]
\[ \text{UPPER} = \text{UPPER} - H(N3) \]
\[ \text{EXVAL} = \text{EL}(N3) \]
\[ \text{ADD} = \text{EL}(N3) + H(N3) \]
\[ \text{TP} = 2 \times H(N3) \]
\[ \text{EXVAL} = \text{EL}(N3) + \text{TP} \]
\[ I = 2 \]

7002  \( \text{SKUNK}(1) = 5 \times \text{PEAK}(Y0, B1, B2, D1, D2, \text{EXVAL}) \)
\( \text{SKUNK}(2) = 5 \times \text{PEAK}(Y0, B1, B2, D1, D2, \text{ADD}) \)
\( \text{SKUNK}(3) = 5 \times \text{PEAK}(Y0, B1, B2, D1, D2, \text{EXVAL}) \)
\( S(I) = S(I-1) + \text{SKUNK}(1) + 4 \times \text{SKUNK}(2) + \text{SKUNK}(3) \)
\[ \text{IF (XVAL+TP-UPPER)} \text{7003, 7003, 385} \]

7003  \( \text{EXVAL} = \text{XVAL} \)
\[ \text{ADD} = \text{XVAL} + H(N3) \]
\[ \text{XVAL} = \text{EXVAL} + \text{TP} \]
\[ I = I + 1 \]
\[ \text{GO TO 7002} \]

C

C

2056  \( H(N3) = (\text{UPPER} - \text{EL}(N3)) / 5121 \)
\[ \text{UPPER} = \text{UPPER} - H(N3) \]
\[ \text{EXVAL} = \text{EL}(N3) \]
\[ \text{ADD} = \text{EL}(N3) + H(N3) \]
\[ \text{TP} = 2 \times H(N3) \]
\[ \text{EXVAL} = \text{EL}(N3) + \text{TP} \]
\[ I = 2 \]
4002 SKUNK(1)=PEARIF(Y0,B1,B2,D1,D2,EXVAL)
   SKUNK(2)=PEARIF(Y0,B1,B2,D1,D2,ADD1)
   SKUNK(3)=PEARIF(Y0,B1,B2,D1,D2,XVAL)
   S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)
   IF (XVAL+TP-UPPER)4003,4003,385
4003 EXVAL=XVAL
   ADD1=EXVAL+H(N3)
   XVAL=EXVAL+2.*H(N3)
   I=I+1
   GO TO 4002

C
250 EXVAL=EL(N3)
   ADD1=EL(N3)+H(N3)
   TP=2*H(N3)
   XVAL=EL(N3)+TP
   I=2
255 SKUNK(1)=PEARIV(Y0,B,V,R,D,EXVAL)
   SKUNK(2)=PEARIV(Y0,B,V,R,D,ADD1)
   SKUNK(3)=PEARIV(Y0,B,V,R,D,XVAL)
   S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)
   IF (XVAL-UPPER)260,385,385
260 EXVAL=XVAL
   ADD1=EXVAL+H(N3)
   XVAL=EXVAL+TP
   I=I+1
   GO TO 255
C
C
200 TP=2*H(N3)
   EXVAL=EL(NP)-TP
   ADD1=EL(N3)-H(N3)
   XVAL=EL(N3)
   I=2
305 SKUNK(1)=NORMAL(Y0,C,EXVAL)
   SKUNK(2)=NORMAL(Y0,C,ADD1)
   TYPE FOUR INTEG
   NORMAL INTEG

PURGE433
PURGE434
PURGE435
PURGE436
PURGE437
PURGE438
PURGE439
PURGE440
PURGE441
PURGE442
PURGE443
PURGE444
PURGE445
PURGE446
PURGE447
PURGE448
PURGE449
PURGE450
PURGE451
PURGE452
PURGE453
PURGE454
PURGE455
PURGE456
PURGE457
PURGE458
PURGE459
PURGE460
PURGE461
PURGE462
PURGE463
PURGE464
PURGE465
PURGE466
PURGE467
PURGE468
SKUNK(3) = NORMAL (Y3, C, XVAL)
S(1) = S(1-1) + SKUNK(1) + 4 * SKUNK(2) + SKUNK(3)
IF (XVAL > UPPER) 310, 385, 385
310  EXVAL = XVAL
ADD1 = XVAL + H(N3)
XVAL = XVAL + TP
I = I + 1
GO TO 305
C
350  EXVAL = EL(N3)
ADD1 = EL(N3) + H(N3)
TP = 2 * H(N3)
XVAL = EL(N3) + TP
I = 2
351  SKUNK(1) = 0
GO TO 3553
352  SKUNK(1) = PEARVI (Y1, B1, B2, D1, D2, EXVAL)
353  SKUNK(2) = PEARVI (Y1, B1, B2, D1, D2, ADD1)
SKUNK(3) = PEARVI (Y1, B1, B2, D1, D2, XVAL)
359  S(1) = S(1-1) + SKUNK(1) + 4 * SKUNK(2) + SKUNK(3)
IF (XVAL > UPPER) 360, 385, 385
360  EXVAL = XVAL
ADD1 = XVAL + H(N3)
XVAL = XVAL + TP
I = I + 1
GO TO 3552
C
460  EXVAL = EL(N3)
ADD1 = EL(N3) + H(N3)
TP = 2 * H(N3)
XVAL = EL(N3) + TP
I = 2
SKUNK(1) = 0
GO TO 452
461  SKUNK(1)=PERIIT (YO,G,P,A1,EXVAL)
462  SKUNK(2)=PERIIT (YO,G,P,A1,ADD1)
463  SKUNK(3)=PERIIT (YO,G,P,A1,EXVAL)
        S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)
        IF (XVAL-UPPER) 463,385,385
463  XVAL=XVAL
        ADD1=XVAL+H(N3)
        XVAL=XVAL+TP
        I=I+1
        GO TO 461
385  A28=H(N3)/3.
386  N(N3)=I+1
387  N33=N(N3)
388  DO 3857 I=1,N33
389  S(I)=S(I)*A28
        LL=N(N3)
        S(LL+1)=1.
C
C
0301  KU(1)=1
C
        A1=0.
        I=1
        DO 675 J=2,100
        A1=A1+.01
        M=I
        KKI=N(N3)
        DO 676 I=M,KKI
        IF (S(I)-A1)676,673,675
676  CONTINUE
677  KU(J)=I
678  KU(101)=N(N3)
C
        CALL DPERFL(J3K)
694  DO 393 K=1,LIMIT
        END=804(1,3)
90 \text{PR} = (100 \cdot \text{RND}) + 1.
\text{NR} = \text{RR}

671 \text{M} = \text{KU}(\text{NP})
\text{KKI} = \text{N}(\text{N3})
\text{DO 387 I} = \text{M}, \text{KKI}
\text{IF(S(I) - RND) 387, 390, 388}

387 \text{CONTINUE}

388 \text{BUT} = \text{FLOAT(I+I-2)} - 2 \cdot (S(I) - \text{RND})/(S(I) - S(I-1))
\text{C}

\text{GO TO 391}

391 \text{BUT} = I + I - 2

392 \text{TOM(K)} = \text{EL}(\text{N3}) + \text{BUT}^* \text{H(N3)} + \text{AVE(N3)}
\text{IF(N2 \cdot 50, 5) GO TO 393}

683 \text{DO 685 KK=1, 4}

685 \text{FAN(KK) = FAN(KK) + TOM(K) ** KK}

393 \text{CONTINUE}
\text{C}

\text{LT} = \text{LT} + \text{LIMIT}
\text{GO TO (952, 400, 953, 400, 952, 401, 401), N2}

952 \text{RETURN}

401 \text{KKI} = \text{N}(\text{N3})
\text{DO 720 MM=1, KKI}

720 \text{SS(N3, MM) = S(MM)}
\text{DO 730 MM=1, 101}

730 \text{K2U(N3, MM) = KU(MM)}
\text{GO TO 400}
\text{C}

\text{CALC OUTPUT MOMENTS}

953 \text{WRITE}(7, 954) \text{N(N3)}, \text{EL(N3)}, \text{H(N3)}, \text{AVE(N3)}, \text{CM2(N3)}, \text{CM3(N3)}, \text{CM4(N3)}

954 \text{FORMAT}(14, 4F18.8, 2F18.8)
\text{WRITE}(7, 955) \text{(KU(I), I=1, 101)}

955 \text{FORMAT}(20I4)
\text{KN=K(N3)}
\text{WRITE}(7, 956) \text{(S(I), I=1, KN)}

956 \text{FORMAT}(20F10.8)
400 \text{IF(FAN(2) 200, 9) GO TO 148}
\text{TD=FLOAT(LT)}

\text{PURGE541, PURGE542, PURGE543, PURGE544, PURGE545, PURGE546, PURGE547, PURGE548, PURGE549, PURGE550, PURGE551, PURGE552, PURGE553, PURGE554, PURGE555, PURGE556, PURGE557, PURGE558, PURGE559, PURGE560, PURGE561, PURGE562, PURGE563, PURGE564, PURGE565, PURGE566, PURGE567, PURGE568, PURGE569, PURGE570, PURGE571, PURGE572, PURGE573, PURGE574, PURGE575, PURGE576}
DO 405 KK=1,4
405 FA(KK)=FAM(KK)/TP
   FCM2 =FA(2)-FA(1)**2
   FCM3 =FA(3)-3.*FA(1)*FA(2) +2.*FA(1)**3
   FCM4 =FA(4)-4.*FA(1)*FA(3) +6.*FA(1)**2*FA(2) -3.*FA(1)**4
   FBETA1=FCM3**2/FCM2**3
   FBETA2=FCM4/FCM2**2
   FKAPPA=FBETA1*(FBETA2+3.)**2/(4.*FBETA2-3.*FBETA1)
2   CALL OVERFL(J3K)
   IF(J3K .EQ. 1) WRITE(6,922)
922 FORMAT(6H OVERFL)
WRITE (L3,410) LT
410 FORMAT(1HC,27X,32HMOMENTS FROM ORIGINAL DATA,6X,
       219HEROM GENERATED DATA,6H N=,I6)
407 WRITE(L3,411) AVE(N3),FA(1),CM2(N3),FCM2,CM3(N3),FCM3,CM4(N3),
   1 FCM4,
2   BETAI,BETA2,BETA2,SKAPPA,FKAPPA
411 FORMAT (35X,4HEAN,3X,F18.8,4X,F18.8 /31X,8HVARANCE,3X,F18.8,4X,)
2   F18.8 /34X,5HMU(3),3X,F18.8,4X,F18.8 /34X,5HMU(4),
3   3X,F18.8,4X,F18.8 /33X,6HBETA 1,3X,F18.8,4X,F18.8/
4   33X,6HBETA 2,3X,F18.8,4X,F18.8 /34X,5HKAPPA,3X,
5   F18.8,4X,F18.8)
C
C
C
631 IF(N2.EQ.4) GO TO 680
412 RETURN
C
C
C
NO GENERATION WRITE OUT
149 WRITE (L3,149)
149 FORMAT(1HC,51X,17HMOMENTS FROM DATA )
   WRITE(L3,150) AVE(N3),CM2(N3),CM3(N3),CM4(N3),BETAI,BETA2,
   1 SKAPPA
   680 FORMAT(1HC,51X,17HMOMENTS FROM DATA )
   WRITE(L3,150) AVE(N3),CM2(N3),CM3(N3),CM4(N3),BETAI,BETA2,
   1 SKAPPA
150  FORMAT (5*E12.8, 4*H16.8, 4*E12.8, 5*E12.8, 5*H16.8)
2   , 4*E12.8, 5*H16.8, 4*E12.8, 5*E12.8, 5*H16.8)   PURGE614
BETA 2, 4*E12.8 / 5*H16.8 / 6*H16.8 / 6*H16.8)
IF(N2 .EQ. 4) GO TO 680
RETURN
C   GRAPH ROUTINE
680  WRITE(L3,950)
    DO 80 I=1,100
    Q(I) = BLANK
80  CONTINUE
    DO 81 I=1,5
81  SLD(I)=0.
    BOX=(V7-V6)/99.
    Q=V6+BOX
    PIP=0.
    SMAX=0.
    SMIN=-6.
    DO 82 I=2,99
    GO TO (83,84,85,86,87),JAB
83  VAL(I)=PEARIF(V0,V1,V2,V3,V4,Q)
    GO TO 88
84  VAL(I)=PEARIV(V0,V1,V2,V3,V4,Q)
    GO TO 88
85  VAL(I)=NORMAL(V0,V1,Q)
    GO TO 88
86  VAL(I)=PEARVI(V0,V1,V2,V3,V4,Q)
    GO TO 88
87  VAL(I)=PERIIV(V0,V1,V2,V3,Q)
    GO TO 88
88  DGT(I)=Q+AVF(N3)
    Q=Q+BOX
    IF(VAL(I)-SMAX)93,93,89
    SMAX=VAL(I)
93  IF(VAL(I)-SMIN)94,94,94
    SMIN=VAL(I)
94  CONTINUE
    N(N3)=0
DO 95 K=1,5
F=8*(6-K)
SID(K)=SMIN+E/39.*(SMAX-SMIN)
DO 95 J=1,R
N(N3)=N(N3)+1
NA(1)=1
NA(2)=1
M1=0
DO 96 I=2,99
TMP=(VAL(I)-SMIN)/(SMAX-SMIN)*39.+1.
L=TMP
L=41-L
IF(N(N3).NE.L) GO TO 96
O(I)=ASTRIC
M1=M1+1
NA(M1)=I
96 CONTINUE
GO TO (97,98,98,98,98,98,98,98),J
97 WRITE (L3,99) SID(K),(O(I),I=1,100)
98 FORMAT(2X,E9.2,2H+,100A1)
GO TO 65
99 WRITE(L3,66) (O(I),I=1,100)
65 FORMAT(13X,100A1)
IF(M1.LE.0) M1 = 1
DO 95 I=1,M1
L=NA(I)
O(L)=BLANK
95 CONTINUE
C
C WRITE (L3,67) (80(T(I)),I=10,90,10)
67 FORMAT (/L3X,9L9X,1H+)/L3X,96E10.3)
RETURN
C ERROR NOTIFICATION
750 WRITE(L3,751) BETA1,BETA2
751 FORMAT(3X,2F18.8,29H ILLEGAL VALUES FOR B1 AND B2)
GO TO 983
903 WRITE (L3,904)01,02
904 FORMAT (3X,2F18.8,29H ILLEGAL VALUES FOR M1 AND M2)
   GO TO 983
906 WRITE (L3,907)81,82
907 FORMAT (3X,2F18.8,29H ILLEGAL VALUES FOR A1 AND A2)
   GO TO 983
910 WRITE (L3,911)
911 FORMAT (3X,33H TOO MUCH AND / OR TOO LARGE DATUM)
   SKIP TO NXT PASS
C 983 LIMIT=0
RETURN
605 STOP
END
APPENDIX C

Flow Chart and Listings for MIX1
Entry

N1 = 1

Yes

Read Option Card

Write Identifying Header

Probabilities Sum to 1

Yes

Compute Moments of Mixture

No

E

N1 = 3

Yes

A

No

A

Flow Chart for MIX1
Flow Chart for MIX1 (Cont.)
Flow Chart for MIX1 (Cont.)
Generate 10,000 Random Numbers and Determine Max and Min

Write Both Sets of Moments

Determine Interval Width For 100 Intervals From Max and Min

Regenerate Same 10,000 Nos. to Obtain Frequency in Intervals

N2=4

Yes

Smooth Curve Using Moving Averages of Order Five

No

Determine Ordinate Scale

Flow Chart for MIXI (Cont.)
Determine Abscissa Scale

Write Graph of Curve

Return

E

Write Prob.'s Do Not Sum to 1. Correct and Resubmit

Limit=0

Return

Flow Chart for MIX1 (Cont.)
SUBROUTINE MIX1(MKX, MKZ)
FUNCTIONS TO COMPUTE NONCENTRAL MOMENTS
FOR MIX OF DIST
RM2(U1, U2) = U2 + U1**2
RM3(U1, U2, U3) = U3 + 3.*U1*U2 + U1**3
RM4(U1, U2, U3, U4) = U4 + 4.*U1*U3 + 6.*U2*U1**2 + U1**4
N1 = MKX
N2 = MKZ
DATA ASTRSK/4H*
DATA BLANK/4H
DIMENSION D(100), NA(100), BOT(9), VAL(101), SID(5), NAL(100)
COMMON/VPI001/IX/VPI002/KX
COMMON/S2641, N2, EL2, H2, C12, C22, C32, C42,
1KU2, 101, F4, 4
COMMON/MCK1/ C1, C2, C3, C4, BETA1, BETA2, SKAPPA
COMMON/MCK2/TOM(100), LIMIT
COMMON/MCK3/ P(2), G, GM, GV, U, UL, UP
COMMON/MCK4/N, EL, H, KU, S
COMMON/Z1/ TM1, CM2, CM3, CM4
COMMON/FA/FZ/FAN/ FCM2, FCM3, FCM4
EQUIVALENCE(S, 0), (S(1, 100), NA), (S(1, 200), NAL)
AU = 0.
BU = 0.
DU = 0.
DG = 0.
LIMIT = 100
M = 1
IF(P(1) .GT. 1.) STOP
GO TO (15, 19, 5), N1
15 READ(5, 101) P(1), P(2), G, GM, GV, U, UL, UP
101 FORMAT(3F7.4, 2F10.5, F7.4, 2F10.5)
5 N1 = 1
IF(P(1) .EQ. 0.) GO TO 6
IF(P(2) .EQ. 0.) GO TO 8
GO TO 4
6 DO 9 J = 1, 2
9 CONTINUE
C1(J)=0.
C2(J)=0.
C3(J)=0.
9 C4(J)=0.
GO TO 4.
8 C1(2)=0.
C2(2)=0.
C3(2)=0.
C4(2)=0.
4 LT=0
DO 126 L=1,4
126 FAN(L)=0.
ASSIGN 60 TO KY
IF((P(2) .EQ. 0.) .AND. (G .EQ. 0.) .AND. (U .EQ. 0.)) ASSIGN 51 TO KY
110 WRITE(6,125) P(1),P(2),G,GM,GV,U,UL,UP
125 FORMAT(1H1,92X,21HDISTRIBUTION MIXTURES/13H0CONTROL CARD,6H P1=,)
1 F10.4,6H P2=,F10.4,6H NORM=,F10.4,6H MEAN=,F12.5,6H VAR=,F12.5
2/1H0,12X,6H UNIF=,F10.4,6H FROM,F10.4,6H TO,F10.4)
IF(ABS(1.-P(1)-P(2)-G-U)*GE. .001) GO TO 100
IF(G .EQ. 0.) GO TO 40
C FOR NORMAL
160 BG=SORT(GV)
DG=3.*GV**2
40 IF(U .EQ. 0.) GO TO 41
C COMPUTE CENTRAL MOMENTS FOR UNIFORM
AU=(UL+UP)/2.
BU=(UP-UL)**2/12.
DU=(UP-UL)**4/80.
C COMPUTE NONCENTRAL MOMENTS FOR MIX
41 TM1=P(1)*C1(1) + P(2)*C1(2) + G*GM + U*AU
TM2=P(1)*RM2(C1(1),C2(1)) + P(2)*RM2(C1(2),C2(2)) + G*RM2(GM,GV)
1+ U*RM2(AU,BU)
TM3=P(1)*RM3(C1(1),C2(1),C3(1)) + P(2)*RM3(C1(2),C2(2),C3(2)) +
1G*RM3(GM,GV,0.) + U*RM3(AU,BU,0.)
TM4=P(1)*RM4(C1(1),C2(1),C3(1),C4(1)) + P(2)*RM4(C1(2),C2(2),C3(2))
C  
COMPUTE CENTRAL MOMENTS  
CM2=TM2-TM1**2  
CM3=TM3-3.*TM1*TM2+2.*TM1**3  
CM4=TM4-4.*TM1*TM3+6.*TM2*TM1**2-3.*TM1**4  
C  
GENERATE RANDOM NOS.  
P(2)=P(2)+P(1)  
G=G+P(2)  
UR=UP-UL  
CALL OVERFL(J3K)  
19 GO TO (20,20,503,504,16),N2  
16 N1=3  
20 DO 393 K=1, LIMIT  
GO TO KY,(60,51)  
52 IF(RN1 .GE. G) GO TO 53  
TOM(K)=(RNOR(0.)*BG)+GM  
GO TO 675  
53 TOM(K)=RDM(1.0)*UR+UL  
GO TO 675  
60 RN1=RDM(1.0)  
IF(RN1 .GE. P(1)) GO TO 50  
M=1  
GO TO 51  
50 IF(RN1 .GE. P(2)) GO TO 52  
M=2  
51 RND=RDM(1.0)  
NR=INT(100.*RND)+1  
L=KU(M,NR)  
671 J=N(M)  
DO 387 I=L,J  
IF(S(M,I)-RND) 387,390,388  
387 CONTINUE  
388 BUT=FLOAT(I+I-2)-2.*(S(M,I)-RND)/(S(M,I)-S(M,I-1))  
C  
GO TO 391  
390 BUT=I+I-2
391  TOM(K)=EL(M)+BUT*H(M)+C1(M)
675  GO TO (683,683,393),N1
683  DO 685 KK=1,4
685  FAN(KK)=FAN(KK)+TOM(K)**KK
393  CONTINUE
       LT=LT+LIMIT
       GO TO (952,400,501,502,952),N2
952  RETURN

C  CAL OUTPUT MOMENTS
400  TP=LT
       DO 405 KK=1,4
405  FAM(KK)=FAN(KK)/TP
       FCM2=FA(2)=FA(1)**2
       FCM3=FA(3)=3.*FA(1)*FA(2)+2.*FA(1)**3
       FCM4=FA(4)=4.*FA(1)*FA(3)+6.*FA(1)**2*FA(2)-3.*FA(1)**4
       CALL OVERFL(J3K)
       IF(J3K.EQ.1) WRITE(6,922)
692  FORMAT(6H DAVRF)
       IF(FAN(2).EQ.0) GO TO 148
       WRITE(6,410) LT
       WRITE(6,411) TM1,FA(1),CM2,CM3,CM4,CM5,CM6
       FORMAT(1H0,27X,32HMOMENTS FROM ORIGINAL DATA,6X,19HFROM GENEM
142  1RATED DATA,6H N=,I6)
647  WRITE(6,412) TM1,FA(1),CM2,CM3,CM4
641  FORMAT(35X,4HMEAN,3X,F18.8,4X,F18.8/31X,8HVARIAANCE,3X,F18.8,4X,
       2F18.8/34X,5HMU(3),3X,F18.8,4X,F18.8/34X,5HMU(4),3X,F18.8,4X,F18.8/MIX10134
3)
       GO TO 412
148  WRITE (6,149)
149  FORMAT(1H0,51X,17HMOMENTS FROM DATA )
       WRITE (6,150) TM1,CM2,CM3,CM4
150  FORMAT (54X,4HMEAN,4X,F18.8/50X,8HVARIAANCE,4X,F18.8/53X,5HMU(3)/MIX10140
2,4X,F18.8/53X,5HMU(4),4X,F18.8)
412  IF(N2.EQ.4) GO TO 75
       RETURN

C  GRAPH ROUTINE
503 ASSIGN 904 TO KS
  GO TO 500
504 ASSIGN 903 TO KS
  N2=3
500 DO 1 I=1,100
  1 VAL(I)=0.
      VAL(101)=0.
      SMIN=TM1
      SMAX=TM1
      IXSAVE=IX
      KXSAVE=KX
      LT=0
  DO 39 I=1,4
39 FAN(I)=0.
    NN=0
70 NN=NN+1
   IF(NN .EQ. 101) GO TO 71
   GO TO 20
501 DO 3 NR=1,100
   IF(TOM(NR) .LT. SMIN) SMIN=TOM(NR)
   IF(TOM(NR) .GT. SMAX) SMAX=TOM(NR)
   3 CONTINUE
   GO TO 70
71 FIL=(SMAX-SMIN)/100.
   IX=IXSAVE
   KX=KXSAVE
   N1=3
   N2=4
   NN=0
   GO TO 400
75 NN=NN+1
   IF(NN .EQ. 101) GO TO 76
   GO TO 20
502 DO 7 NR=1,100
   J=(TOM(NR)-SMIN)/FIL + 1.
   7 VAL(J)=VAL(J)+1.
NA(1)=1
M1=0
DC 96 I=2,99
IF(M .NE. NAL(I)) GO TO 96
O(I)=ASTRSK
M1=M1+1
NA(M1)=I
96 CONTINUE
   GO TO (97,98,98,98,98,98,98,98,98,98),J
97 WRITE(6,99) SID(K),(O(I),I=1,100)
99 FORMAT(2X,E9.2,2H ++,100A1)
   GO TO 65
98 WRITE(6,66) (O(I),I=1,100)
66 FORMAT (13X,100A1)
65 DO 95 I=1,M1
   L=NA(I)
95 O(L)=BLANK
   WRITE(6,67) (BOT(I),I=1,9)
67 FORMAT (/13X,9(9X,1H+)/20 X,9E10.3)
RETURN
C ERROR NOTIFICATION
100 WRITE(6,120)
120 FORMAT(1HO,30X,51HPROBABILITIES DO NOT SUM TO 1. CORRECT AND RESUBMIX10239
1MIT)
   LIMIT=0
RETURN
END
APPENDIX D

Flow Chart and Listing for MIX2
Flow Chart for MIX2

Entry

N1=1

Yes

No → A

Read Option Card

Pearson Distribution Req.

Yes → Read Pearson Cards

No → Write Identifying Header

Probabilities Sum to 1

Yes → A

No → B
Flow Chart for MIX2 (Cont.)
SUBROUTINE MIX2(MKX)
N1=MKX
COMMON/VI001/IX/VI002/KX
DIMENSION S(2,2641),N(2),EL(2),H(2),C1(2),C2(2),C3(2),C4(2),
1 KU(2,101)
COMMON/MCK2/TOM(100),LIMIT
COMMON/MCK3/P(2),G,GM,GV,U,UL,UP
COMMON/MCK4/N,EL,H,KU,S
BG=0.
LIMIT=100
IF(N1.EQ.2) GO TO 19
C1(1)=0.
C1(2)=0.
C READ OPTION CARD
15 READ(5,101) P(1),P(2),G,GM,GV,U,UL,UP
101 FORMAT(3F7.4,2F10.5,F7.4,2F10.5)
IF(P(1).GT.1.) STOP
LT=6
ASSIGN 60 TO KY
IF((P(2) .EQ. 0.) .AND. (G.EQ. 0.) .AND. (U .EQ. 0.)) ASSIGN 51
1 TO KY
M=1
IF(P(1) .EQ. 0.) GO TO 110
L=1
IF(P(2) .NE. 0.) L=2
DO 102 I=1,L
READ(5,9999) N(I),EL(I),H(I),C1(I),C2(I),C3(I),C4(I)
9999 FORMAT(I4,4F18.8/2F18.8)
READ(5,9991) (KU(I,J),J=1,101)
9991 FORMAT(20I4)
K=N(I)
102 READ (5,9992) (S(I,J),J=1,K)
9992 FORMAT(8F16.8)
113 WRITE(6,125) P(1),P(2),G,GM,GV,U,UL,UP
125 FORMAT(1H1,92X,21HDISTRIBUTION MIXTURES/13HC=CONTROL CARD,6H P1=, MIX2001
1 F10.4,6H P2=F10.4,6H NORM=F10.4,6H MEAN=F12.5,6H VAR=F12.5 MIX2002
21H0, 12X, 6H UNIF=,F10.4,6H FROM,F10.4,6H TO,F10.4)
IF(ABS(1.-P(1)-P(2)-G-U) .GE. .001) GO TO 100
IF(G .EQ. 0.0) GO TO 40
C FOR NORMAL
160 BG=SQRT(GV)
C GENERATE RANDOM NOS.
40 P(2)=P(2)+P(1)
   G=G+P(2)
   UR=UP-UL
   CALL OVERFL(J3K)
19 CONTINUE
20 DO 393 K=1,LIMIT
   GO TO KY,(60,51)
52 IF(RN1 .GE. G) GO TO 53
   TOM(K)=(RNOR(0.)*BG)+GM
   GO TO 393
53 TOM(K)=RDM(1.0)*UR+UL
   GO TO 393
60 RN1=RDM(1.0)
   IF(RN1 .GE. P(1)) GO TO 50
   M=1
   GO TO 51
50 IF(RN1 .GE. P(2)) GO TO 52
   M=2
51 RND=RDM(1.0)
   NR=INT(100. *RND)+1
   L=KU(M,NR)
671 J=N(M)
   DO 387 I=L,J
       IF(S(M,I)-RND) 387,390,388
387 CONTINUE
388 BUT=FLOAT(I+I-2)-2.*(S(M,I)-RND)/(S(M,I)-S(M,I-1))
C GO TO 391
390 BUT=I+I-2
391 TOM(K)=EL(M)+BUT*H(M)+C1(M)
393 CONTINUE
   LT=LT+LIMIT
952 RETURN
100 WRITE(6,120)
120 FORMAT(1HO,30X,51HPROBABILITIES DO NOT SUM TO 1. CORRECT AND RESUBMIT)
      LIMIT=0
      RETURN
      END
VITA

The writer was born in Augusta, Georgia, on June 18, 1944. He received his elementary and secondary education in Augusta and was graduated from Lucy C. Laney High School in June, 1962.

He entered Paine College in Augusta in September, 1962, and received the Bachelor of Science degree in Mathematics in June, 1966.

The writer enrolled at the Virginia Polytechnic Institute in July, 1966, to begin his graduate work in Statistics. In February, 1967, he began his employment with the Department of the Army in Radford, Virginia, at the Deputy Chief of Staff for Logistics (DCSLOG) Data Processing Center as a Mathematician. He continued his graduate work as a part-time student.

[Signature]

114
THE PURGE3, MIX1, and MIX2 SUBROUTINES

by

Franklin McKie

ABSTRACT

The main purpose of this study was to review and modify some of the work that has been done concerning the generation of pseudo-random numbers. The PURGE3, MIX1, and MIX2 subroutines, written in FORTRAN IV to be run on the IBM 360 computer system, are used for the generation of pseudo-random numbers from a family of Pearson distributions as well as from any combination of mixtures of two Pearson distributions, a normal distribution with arbitrary mean and variance and a uniform distribution on any finite interval.

The primary improvement undertaken here was to increase the efficiency of operation by enabling the PURGE3 and MIX1 subroutines to be called in a single computer program.

The Pearson distribution may be specified either by the first four moments or from sample data. The parameters of the fitted distribution are printed and, if desired, the moments and a graph of the distribution of the 100 pseudo-random numbers generated are printed. A graph of the mixture of distributions generated in the MIX1 subroutine may be obtained from 10,000 random numbers generated from the mixture. The MIX2 subroutine functions to generate random numbers only.
The speed of generation varies from about 13,000 random numbers per minute for a Pearson distribution which computes and prints the moments from the generated numbers, to approximately 143,000 numbers per minute if mixtures are used without computing the moments from the generated numbers.