COMPUTER METHODS
FOR GENERATING PSEUDO-RANDOM NUMBERS
FROM PEARSON DISTRIBUTIONS AND
MIXTURES OF PEARSON AND UNIFORM DISTRIBUTIONS

by

Donald Gale Thomas

Thesis submitted to the Graduate Faculty of the
Virginia Polytechnic Institute
in candidacy for the degree of

MASTER OF SCIENCE
in
Statistics

APPROVED: _____
Chairman, Richard G. Krutchkoff, PhD.

_____ _____
Boyd Harshbarger, PhD. Whitney L. Johnson

_____
Raymond H. Myers, PhD.

May 1966

Blacksburg, Virginia

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The advent of the modern high speed computer has brought many changes in our society. Complex problems, in almost every discipline, heretofore unsolvable in a lifetime because of the massive computations involved, may now be solved in minutes with the aid of these high speed computers.

One technique, made particularly attractive by the computer era is that of Monte Carlo methods which comprise that branch of mathematics concerned with synthetic experiments involving random numbers. These methods are often useful on a wide variety of problems beyond the available resources of theoretical mathematics.

Since large sequences of random numbers are desired, techniques have been devised for generating them in a completely deterministic manner such that the resulting sequence of numbers, often called pseudo-random numbers, appears random.

In this thesis, a brief survey of existing methods for generation and testing of pseudo-random numbers is presented. One serious limitation inherent in these generation methods is that they are not easily modified to produce random numbers from distributions other than those for which they were written.

To overcome this limitation Cooper, Davis and Dono (1963) developed a computer program in FORTRAN II which generates random numbers from Pearson distributions. The distribution may be specified by supplying the program with either the first four moments of the desired distribution or sample data from that distribution. The program then

computes and prints out the parameters for the Pearson distribution fitted and if desired, generates random numbers from said distribution and/or produces a graph of the distribution.

The program has been considerably revised and improved as well as converted to be used as a FORTRAN IV subroutine. This subroutine is entered from the mainline program by using the FORTRAN call statement

CALL PURGE2(N1,N2)

where N1 and N2 are arguments which describe the option desired. Initially, depending on the option selected, the subroutine reads data cards (provision is also made for internal specification of moments) which describe the Pearson distribution to be fitted. The distribution is fitted and parameters for the Pearson curve with origin at the mean are printed. At this point, one hundred random numbers are generated (if the generation option was selected) and returned to the calling program as implied arguments via the COMMON statement. If desired, sample moments from the generated data are calculated.

If more random numbers are desired from the same distribution, the subroutine may be recalled, for each batch of one hundred numbers required, by an argument option which does not necessitate refitting the curve. This procedure produces ten to twenty thousand random numbers per minute.

A graph of the distribution may be obtained on the <u>final call</u> (the graph option destroys the facility for further generation without refitting). The first four moments of the Pearson distribution along with β_1, β_2 and K may be printed as well as comparable values

calculated from the generated numbers.

In order to remove the restriction of generating pseudo-random numbers from only unimodal distributions as done in the original version of PURGE, another subroutine is developed which allows random numbers to be generated from a mixture of distributions. Any proportion of two Pearson distributions, a normal distribution with arbitrary mean and variance and a uniform distribution on any finite interval may be used in the mixture. The theoretical moments for the mixture are calculated and, if desired, sample moments calculated from the generated random numbers.

When Pearson distributions are to be included in the mixture, this subroutine uses as input binary card output from the PURGE2 subroutine consisting of the cumulative distribution function of the Pearson distribution(s) along with necessary parameters for the generation of pseudo-random numbers from the distribution(s). With this data, pseudo-random numbers may be generated from Pearson distribution(s) by the same technique used in PURGE2.

The subroutine is entered by the FORTRAN statement

CALL MIX(N1,N2)

where N1 and N2 are arguments, similar to those of PURGE2, which describe the option desired. Initially the subroutine reads a data card specifying the proportion of distributions to be mixed then the Pearson data decks if necessary. For each pseudo-random number generated, a uniform random number is generated which determines the distribution from which to generate the pseudo-random number. As in PURGE2, batches

of one hundred random numbers are generated per call.

If desired, a graph of the distribution may be obtained on the final call. Unlike PURGE2 which obtains its graph from the distribution function, the MIX subroutine obtains its graph from ten thousand random numbers generated from the mixture. All other aspects of the MIX subroutine are similar to PURGE2.

Provision is made in both subroutines for regenerating the same sequence of numbers in the same or different computer runs as well as avoiding regeneration in different computer runs.

A small Monte Carlo investigation of the robustness of three methods of confidence interval estimation to departures from normality is given as an application of the aforementioned computer programs.

Flowcharts along with FORTRAN listings of both subroutines are given in the Appendix.

II. A BRIEF SURVEY OF METHODS OF GENERATING PSEUDO-RANDOM NUMBERS

2.1 Definition and Use of 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 what are called Monte Carlo calculations. Such calculations sometimes require large sequences of random numbers from various probability distributions.

An early approach to the problem of obtaining sufficient random numbers to handle problems involving Monte Carlo methods was to generate them by some mechanical means such as the electronic roulette wheel described by Brown (1951), record them in machine readable form and read them into the computer as needed. Since almost all computers are input-output limited, this method is entirely too slow.

One of the first papers on improved techniques for generating random numbers was presented at Harvard in September, 1949, by Lehmer (1951). He defined a pseudo-random sequence to be "a vague notion embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests, traditional with statisticians and depending somewhat on the uses to which the sequence is to be put." Here "uninitiated" refers to a person without a priori knowledge of how the numbers are obtained.

2.2 Common Methods of Generating Pseudo-Random Numbers

A number of methods have been proposed for the generation of pseudo-random numbers hereafter refered to simply as random numbers.

Generally these methods follow the following pattern.

A starting number is selected from which the first random number is determined. Once started, the N+1th number is determined from the Nth 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 are 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 pseudo-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 even 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 or variance lies within acceptable limits. These statistics may be calculated, for a large sequence, without storing any of the numbers in the sequence. The sequence can then be regenerated for further calculations, if desired, without requiring a large amount of computer storage to contain the generated numbers.

Another characteristic of all pseudo-random number generators is that the sequence repeats itself. The total number, N, of numbers generated is usually refered to as the period. Fortunately, in most cases the period has been determined mathematically for the generators

commonly used and can be made quite large by careful selection of parameters used in generation.

2.21 Mid-square Methods

Metropolis (1956) gives an interesting account of the mid-square method of random number generation. This was one of the first methods of generating random numbers, adapted for use on high speed computers, but has generally been discarded since it does not have a very long period. The period must be found by trial for each starting 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 mth number.

It 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.

The largest period given by any of the starting numbers used in the Metropolis paper is about 10^6 with the average being about 10^4.

2.22 Multiplicative Congruential Methods

Another class of generation methods is called multiplicative congruential or power residue methods. They can be defined by the congruence relation

(2.1) $$x_{n+1} \equiv ax_n \pmod{m}$$

introduced by Lehmer (1951). To begin the sequence $\{x_i\}$ of non-negative integers, a positive integer x_0 is chosen as the starting value. This

number is multiplied by another integer a called the multiplier whereupon the product is divided by an integer m, called the modulus, which is positive and greater in magnitude than the other two. The resulting remainder is x_1, the first pseudo-random number generated. The next pseudo-random number is obtained by replacing x_0 with x_1 and repeating the same calculations. In order to obtain numbers in the interval $[0,1)$, the sequence $\{x_i/m\}$ is formed.

Clearly, this sequence must eventually repeat itself since it can contain at most m different numbers with each number in the given sequence determined by its predecessor. Methods for insuring that the sequence will have maximum period m are given in an IBM (1959) report on this method. This report gives the mathematical development and includes an appendix providing programming illustrations for binary and decimal computers.

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 the one most widely used at this time.

2.23 Mixed Congruential Methods

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

$$x_{n+1} \equiv ax_n + c \pmod{m} \tag{2.2}$$

introduced by Coveyou (1960) and Rotenberg (1960).

In this case c is an integer which is added to the product ax_n before division by m. Otherwise the computations are the same as for the multiplicative case. Hull and Dobell (1962) give conditions on x_0, a, c and m which insure maximum period m.

2.24 Multiplicative vs. Mixed Generators

The underlying theory for mixed congruential generators is simpler than that of multiplicative methods and the periods are longer although both can be extremely long. The only disadvantage of the mixed methods seems to be their statistical properties. Although in general their statistical behavior is quite good, in some respects it is completely unacceptable in a small proportion of cases. Hull and Dobell (1962) report tests of 600 different multipliers about one percent of which they believe to be unacceptable in the sense that they lead to rediculously large values of X^2, some as large as 900. They also report tests made with 513 different multipliers using the multiplicative congruential method and "the results were entirely consistent with the hypothesis that the sequence was drawn at random from the uniform distribution".

MacLaren and Marsaglia (1965) report the results of extensive testing of both methods of generating random numbers including two new methods:

The first new method uses a table of random numbers from which a random number U is obtained. Once the original table is exhausted, new numbers are produced by a transformation of the form

(2.3) $$U' = aU + c \pmod{2^{27}}$$

where a and c are chosen such that the transformation is one-to-one and onto, hence the transformed table will be random if the original table was.

The second new method involves two different generators

(2.4) $$U_{k+1} = (2^{17} + 3)U_k \pmod{2^{35}}$$

(2.5) $$V_{k+1} = (2^{7} + 1)V_k + 1 \pmod{2^{35}}$$

with one used to shuffle the sequence produced by the other. Initially, $U_0 = 1$ and $V_0 = 0$. A table of 128 locations in core is filled with the numbers $U_1, \ldots, U_{128}$. To generate X_k, the kth random number to be used, the first 7 bits of V_k is used as an index to get X_k from the table. The location of X_k in the table is refilled with the next number from the generator (2.4).

Both new methods passed the tests whereas the conventional multiplicative and mixed congruential methods did not. The tests used here were somewhat more stringent than those usually applied although not at all unreasonable.

2.25 Other Methods

Most of the effort, to date, has been in the area of generating continuous uniformly distributed random variables. The uniform distribution is equated such a high level of importance because any continuous distribution can be transformed into the uniform distribution implying that the uniform distribution may be transformed into any continuous distribution, viz. the reverse of the transformation that

transforms the desired distribution into the uniform. See for example Kendall and Stuart (1963).

Marsaglia (1961) gives methods for expressing a random variable in terms of a uniform random variable and also a method for generating exponential random variables. A summary of methods for generating normal random variables is given by Muller (1959). A discussion of various methods for generating random variables is given in the Handbook of Mathematical Functions published by the United States Department of Commerce (1964). Additional references may be found in a comprehensive bibliography given in Hull and Dobell (1962).

III. A BRIEF SURVEY OF METHODS OF TESTING RANDOM NUMBERS

3.1 Necessity for Testing

A note of warning should be taken from Lehmer's definition of pseudo-random numbers. One should not proceed blindly using random numbers for a problem without first making sure that they satisfy the requirements of the particular problem at hand.

If the outcome of a Monte Carlo simulation is critically dependent on, say whether or not the random numbers used have a high serial correlation, it would behoove the user to test them for such, or at least use a method of generation in which the order or magnitude of the serial correlation is known to be small. More than one Monte Carlo simulation has given erroneous results simply because the user failed to check certain properties of the pseudo-random number generator used, which turned out to be critical in his problem. See Hull and Dobell (1962) and Coveyou (1960) for example.

MacLaren and Marsaglia (1965) report that "random numbers generated by mixed congruential methods gave poor results in a number of Monte Carlo calculations, notably those involving order statistics. . . . The principal difficulty seems to be that certain simple functions of n-tuples of uniform random numbers do not have the distribution that probability theory predicts." They discuss the performance of two new generators which seem to overcome the difficulties inherent in the other methods.

Hull and Dobell (1962) mention a private communication with

R. R. Coveyou in which he "refers to experiences in which special correlation within a sequence has caused erroneous results in Monte Carlo calculations, in spite of the fact that routine statistical tests did not reveal the existence of such correlation" and "draws attention to the need for more quantitative information about the reliability of tests."

To illustrate the point that tests should be made to insure that the random numbers satisfy the requirements of the problem at hand, consider the case of numerical integration. Here, the order of the generated numbers is not important. It is not hard however, to think of cases where the order of the generated numbers would be extremely important. For example, suppose we have a population of 10,000 items and we wish to select a simple random sample of 100 items from this population in order to observe some characteristic of the sample and make inferences about the population. If we assume the population is numerable, we may set up a random number generator which generates uniform random numbers on the interval [1,10000] and take the first 100 numbers generated as the item numbers for our sample. If the order of the first 100 pseudo-random numbers generated is not sufficiently random and the population is not homogeneous, our sample results will be seriously biased.

A simple technique for testing random number generators would be to try them on a similar problem for which the answer is known. See Chapter VII for an example using the N(0,1) distribution, i.e. a normal distribution with mean zero and variance one.

3.2 Tests of Fit

Often it is desired to test a set of numbers to determine if they could have come from a particular distribution. This is a standard question for statistical techniques and many tests have been devised and studied.

3.21 X^2 Goodness of Fit

Suppose the data is divided into k classes with $X_1, X_2, \ldots, X_k$ observations in each class. Let us define the expected values in the corresponding classes a priori to be $m_1, m_2, \ldots, m_k$ such that $\sum_{i=1}^{k} X_i = \sum_{i=1}^{k} m_i = n$. Thus $p_i = m_i/n$ is the probability of an observation falling in the ith class. We wish to test the hypothesis that the sample distribution came from a population with the given set of m_i.

An approximate test of this hypothesis is given by obtaining the probability of getting a X^2 with (k-1) degrees of freedom greater than the computed

$$X_0^2 = \sum_{i=1}^{k} \frac{(x_i - m_i)^2}{m_i} .$$

We reject the hypothesis if this probability is unusually small, say 0.05 or less. Such a test is known as the X^2 goodness of fit test and its theoretical aspects are discussed by Kendall and Stuart (1961). Examples of its use in testing random numbers are given in Taussky and Todd (1956), Hull and Dobell (1962 and 1964), and MacLaren and Marsaglia (1965). Particular attention should be given to the power of the test which may be calculated with the aid of exact tables, for $\alpha = 0.05$,

given by Patnaik (1949).

3.22 Kolmogorov Statistic

An alternative to the X^2 goodness of fit test is given by the Kolmogorov statistic which is based on deviations of the sample distribution function $S_n(x)$ from the completely specified hypothetical distribution function $F_o(x)$. The measure of deviation is the maximum absolute difference between $S_n(x)$ and $F_o(x)$ given by

$$D_n = \sup_x |S_n(x) - F_o(x)|.$$

The asymptotic distribution of D_n was obtained by Kolmogorov in 1933 and is completely distribution-free under H_o: $F(x) = F_o(x)$.

The advantage of this method over all other methods is that D_n may be used to set confidence limits for a continuous distribution function as a whole. See Kendall and Stuart (1961) for theory and examples.

3.23 Moments

Another procedure that is easily applied is to calculate the moments of the random sequence and compare them visually with their theoretical values. Usually only the first four mements are used due to the large sampling variability of higher moments. As the sample size increases, the sample moments should converge to their true values which are, if all is well, the theoretical values under consideration.

3.3 Tests of Performance

The aforementioned tests of fit do not give any indication of

the order in which the random numbers are drawn. Once it is known that the proper distribution is being generated, tests on the order of the random variables should be made, if the order is important for the particular problem being investigated.

3.31 Runs Up and Down

An example of one such test of order is given by Taussky and Todd (1956). The number of runs of length 1 through 4 and ≥ 5 were determined from 16 sets of 1024 numbers using a multiplicative congruential generator. The observed average number of runs was then compared visually with the expected average as well as the observed variance with the expected variance and the results were considered satisfactory. A disadvantage of this method is that these expected values may be somewhat difficult to calculate for non-uniform distributions.

3.32 Runs Above and Below Mean

Taussky and Todd (1956) also include tests of runs above and below the mean based on 16 sets of 1024 numbers from the uniform distribution giving tabulated values for run lengths from 1 to 10, comparing visually the observed average number of runs with the expected number. Here again the same difficulties may be encountered for non-uniform distributions.

3.33 Serial Correlation

Serial correlation in a sequence of random numbers refers to the

correlation between a random number and its ith successor. Coveyou (1960) gives a method for estimating the serial correlation and determining the parameters in mixed congruential methods such that it will be small.

3.34 Other Tests of Performance

Hunter (1960) gives a test for repeating cycles. Additional tests which are more stringent than those usually applied, although not at all unreasonable, are given by MacLaren and Marsaglia (1965). They include tests on pairs, triples, maximum and minimum of a set of n uniform variates and the sum of n uniform variates.

An extensive discussion of testing random numbers including examples and computer programs for performing the tests is given in an IBM Technical Report by Gorenstein (1966).

IV. PEARSON'S SYSTEM OF FREQUENCY CURVES

All of the previously mentioned methods for generating random numbers have a serious limitation in that they are designed to generate a specific distribution and are generally not easily modified to generate other distributions. Obviously, a method for generating random variables from a wide variety of continuous distributions would be highly desirable.

In order to accomplish this, a new approach was devised by Cooper, Davis and Dono (1963) who wrote a versatile computer program, PURGE, to generate random numbers from the Pearson system of frequency curves.

4.1 Definition and Theory

The family of unimodal frequency functions obtained as solutions to the differential equation

$$\frac{dy}{dx} = \frac{y(x - a)}{b_0 + b_1 x + b_2 x^2} \tag{4.1}$$

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 be related to the moments of the curve. A characteristic of curves of the family (4.1) is that they are completely determined by their first four moments, μ_1', μ_2, μ_3 and μ_4.

In equation (4.1), it is evident that the mode is at the point $x = a$. If we translate the origin to the mode, this equation may be written in the form

$$\frac{d}{dx}(\log y) = \frac{x - a}{B_0 + B_1(x - a) + B_2(x - a)^2}$$

or

$$\frac{d}{dx}(\log y) = \frac{X}{B_0 + B_1X + B_2X^2} \quad . \tag{4.2}$$

We may now obtain two main Pearson types depending on whether the denominator on the right has real or imaginary roots.

4.2 Types

Type I (Real roots)

We may write

$$B_0 + B_1X + B_2X^2 = B_2(X + a_1)(X - a_2) \qquad a_1, a_2 > 0$$

then

$$\frac{d}{dx}(\log y) = \frac{X}{B_2(X + a_1)(X - a_2)}$$

$$= \frac{a_1}{B_2(a_1 + a_2)} \cdot \frac{1}{(X + a_1)} + \frac{a_2}{B_2(a_1 + a_2)} \cdot \frac{1}{(X - a_2)}$$

whereupon

$$y = Y_0(X + a_1)^{\frac{a_1}{B_2(a_1 + a_2)}}(X - a_2)^{\frac{a_2}{B_2(a_1 + a_2)}}$$

which may be written in the form

$$y = Y_0 \left\{1 + \frac{x}{a_1}\right\}^{M_1} \left\{1 - \frac{x}{a_2}\right\}^{M_2}$$

where

$$\frac{M_1}{a_1} = \frac{M_2}{a_2} \quad .$$

If the origin is at the mean

$$(4.3) \qquad y = Y_e\left(1 + \frac{x}{A_1}\right)^{M_1}\left(1 - \frac{x}{A_2}\right)^{M_2}$$

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_e, M_1 and M_2 are determined by integrating the curve over its range from $-a_1$ to a_2, (see Elderton 1953). This curve varies in shape from bell to U, and, twisted J.

Type IV (Imaginary roots):

In this case we may write

$$\frac{d}{dx}(\log y) = \frac{x}{B_2\left[\left(\frac{X + \frac{B_1}{2B_2}}{}\right)^2 + \left(\frac{B_0}{B_2} - \frac{B_1}{4B_2^2}\right)^2\right]}$$

or $$= \frac{X}{B_2\left[(X - g)^2 - d^2\right]}$$

thus

$$\log y = \log Y_0 + \frac{1}{2B_2}\log\left[(X + g)^2 + d^2\right] - \frac{g}{B_2 d}\tan^{-1}\frac{X + g}{d}$$

$$y = Y_0\left[(X + g)^2 + d^2\right]^{\frac{1}{2}B_2}\exp\left[\frac{-g}{B_2 d}\tan^{-1}\frac{X + g}{d}\right]$$

usually written

$$y = Y_0\left(1 + \frac{x}{A^2}\right)^{-M}\exp(-V\tan^{-1}x/A)$$

where the origin is $VA/(2M - 2)$ after the mean.

The curve may also be written as

$$y = Y_0\left[1 + \left(\frac{x}{A} - \frac{V}{R}\right)^2\right]^{-M} \exp\left[- V\tan^{-1}(x/A - V/R)\right]$$

where

$R = 2M - 2$, with origin at the mean.

The curve is of unlimited range in both directions and is bell shaped. For determination of the constants as well as the derivation of other Pearson curves see Elderton (1953).

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

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

with $\beta_1 = \mu_3^2/\mu_2^3$ and $\beta_2 = \mu_4/\mu_2^2$ being coefficients of kurtosis and skewness respectively. For Type I curves, K is < 0 and for Type IV curves $0 < K < 1$. Table 1 gives equations of the Pearson curves with origin at the mean along with criterion for K and certain remarks about the curves.

Pearson and Hartley (1962) give a diagram (Table 2) plotting β_1, β_2 to rectangular axes, from which, one may determine the appropriate type of Pearson curve for given β_1 and β_2. This chart covers Types I-VII.

4.3 Uses

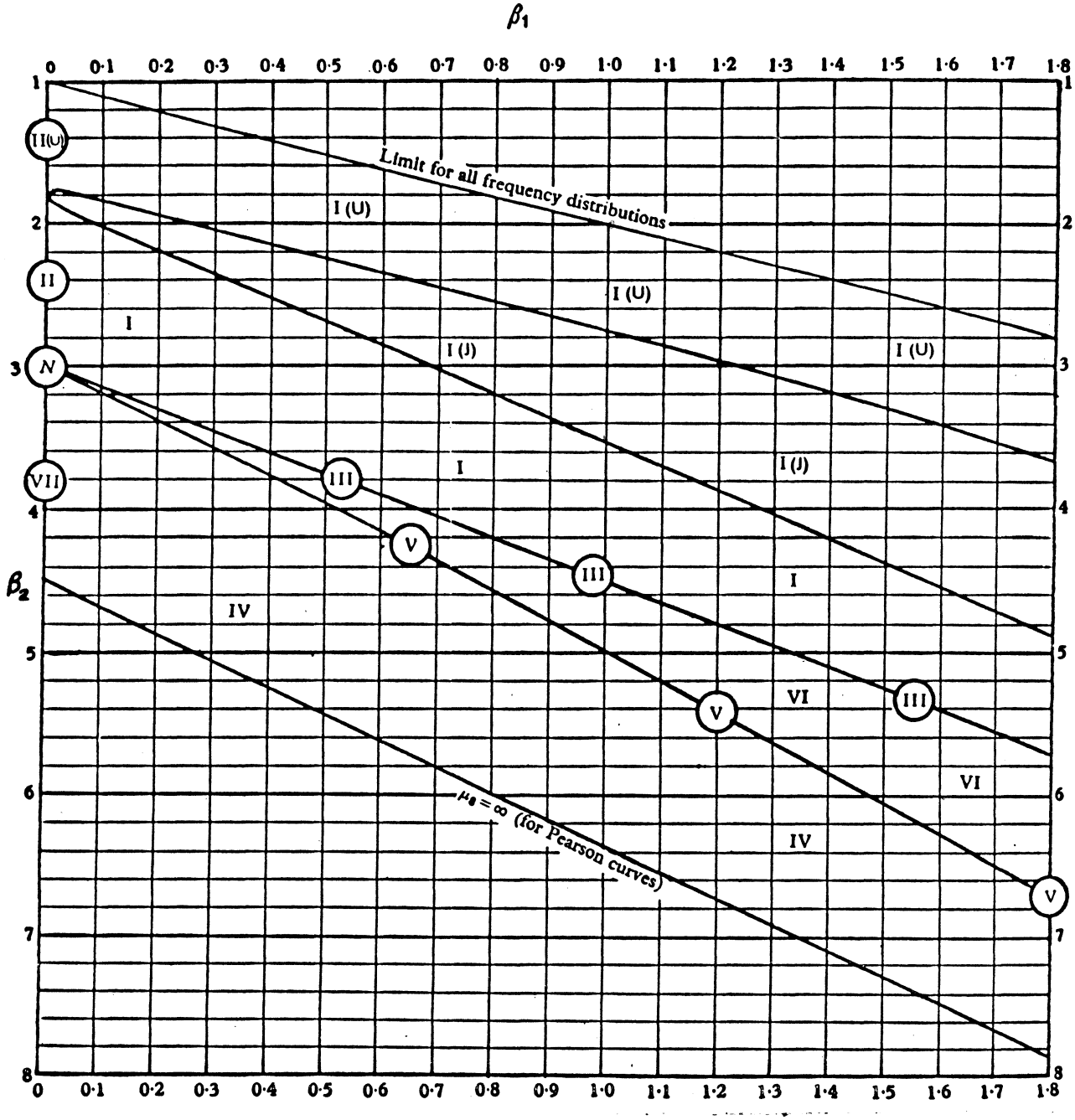
Pearson curves were originially used in actuarial statistics by calculating moments from a large number of observations and fitting

Table 1. Types of Pearson Curves Used*

Pearson Main Types	Equation with origin at the mean	Criterion	Remarks
I	$Y = Y_e(1+X/A_1)^{M1}(1-X/A_2)^{M2}$	κ negative	Limited range; skew; usually bell-shaped, but may be U-shaped, J-shaped or L-shaped.
IV	$Y = Y_0[1+(X/A-V/R)^2]^{-M}e^{-V\tan^{-1}(X/A-V/R)}$	$0<\kappa<1$	Unlimited range; skew; bell-shaped.
VI	$Y = Y_e(1+X/A_1)^{-Q1}(1+X/A_2)^{Q2}$	$\kappa>1$	Unlimited range in one direction; skew; bell-shaped, but may be J-shaped.
Subtypes			
Normal	$Y = Y_0e^{-X^2/C}$	$\kappa=0$, $\beta_1=0$, $\beta_2=3$	Unlimited range; symmetrical; bell-shaped.
II	$Y = Y_0(1-X^2/A^2)^M$	$\kappa=0$, $\beta_1=0$, $\beta_2<3$	Approximated by Type I. Limited range; usually bell-shaped,but U-shaped when $\beta_2<1.8$; symmetrical.
VII	$Y = Y_0(1+X^2/A^2)^{-M}$	$\kappa=0$, $\beta_1=0$, $\beta_2>3$	Type IV with V=0. Unlimited range; symmetrical; bell-shaped.
III	$Y = Y_e(1+X/A)^Pe^{-GX}$	$2\beta_2=6+3\beta_1$	Unlimited range in one direction; usually bell-shaped, but may be J-shaped.
V	$Y = Y_e(1+X/A)^{-P}e^{(P-2)/(1+X/A)}$	$\kappa=1$	Approximated by Type VI. Unlimited range in one direction; bell-shaped.
VIII	$Y = Y_e(1+X/A)^{-M}$	$\kappa<0$, $\lambda=0$, $5\beta_2-6\beta_1-9<0$	Type I with $M_2=0$.
IX	$Y = Y_e(1+X/A)^M$	$\kappa<0$; $\lambda=0$; $5\beta_2-6\beta_1-9>0$, $2\beta_2-3\beta_1-6<0$	Type I with $M_2=0$.
X	$Y = Y_ee^{-GX}$	$\beta_1=4$, $\beta_2=9$	Type III with P=0. Exponential form from finite ordinate at $-G^{-1}$ to infinitesimal ordinate at ∞; J-shaped.
XI	$Y = Y_e(1+X/A)^{-M}$	$\kappa>1$, $\lambda=0$, $2\beta_2-3\beta_1-6>0$	Type I with $M_2=0$. J-shaped.
XII	$Y = Y_e(1+X/A_1)^{M1}(1-X/A_2)^{M2}$	$5\beta_2-6\beta_1-9=0$	Special case of Type I, twisted J-shape.

*This table is a modified version of Table VI. given by Elderton (1953). His book should be consulted for calculation of constants.

Table 2.* Chart relating the type of Pearson frequency curve to the values of β_1, β_2.



*Reproduced from Pearson and Hartley (1962), Table 43, p. 210.

a Pearson curve which would satisfactorily represent the observations in order to allow accurate graduation and interpolation. For this purpose they are quite satisfactory.

The method has been criticized when the observed data are samples from a population and it is desired to find a mathematical equation satisfactorily representing that population. "In such cases the moments calculated from observation are only estimates of population-moments and they do not, in general, lead to the most efficient estimates of the population parameters", Kendall and Stuart (1963).

"In many instances experience has shown the value of the Pearson curves in approximating, on the basis of known moments, to the distributions of frequency functions which are either undetermined or are not readily expressed in simple form. In addition, these curves, of course, represent exactly the distributions of a number of statistics in common use, e.g. X^2, t and F", Pearson and Hartley (1962).

V. PURGE2 SUBROUTINE

The advent of modern high speed computers makes possible another use of Pearson curves . . . that of a highly versatile method of generating random numbers for a whole family of unimodal distributions, by simply supplying the first four moments of the distribution.

5.1 Description and Purpose

As a means to this end, three members of the IBM Scientific Computation Department, Cooper, Davis and Dono, wrote a computer program, "Pearson Universal Random Distribution Generator" (PURGE) to do just that. This program was originally written for the IBM 7090 system in FORTRAN II and has been considerably revised and improved, by this author, to be used as a subroutine (PURGE2) on the IBM 7040-1401 system in FORTRAN IV.

The subroutine is entered by use of the FORTRAN call statement

CALL PURGE2(N1,N2)

where N1 and N2 are arguments which specify the option desired, see Table 3. Initially, depending on the option selected, the subroutine reads data cards which describe the Pearson distribution to be fitted. The distribution may be described either by its first four moments (provision is also made for internal specification of 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 the generation option was selected) and returned to the calling program as implied arguments by placing the FORTRAN instruction

```
COMMON/Z2/RDS(100)
```

in the calling program. After each call to PURGE2, the new batch 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 batch of one hundred numbers required, by an argument option which does not necessitate refitting the curve. This procedure produces about ten thousand random numbers per minute.

Available output options will print the moments used to determine the curve type as well as β_1, β_2 and K along with sample values of these statistics calculated from the generated random variables. After the last batch of random variables has been generated, a graph of the distribution function may be obtained. One should not graph the distribution then attempt to generate additional random numbers from it, since the graph routine shares part of its storage with the cumulative distribution function and thus destroys the facility for further generation without refitting the distribution.

Additional speed in generation may be obtained by electing an option which does not calculate the moments of the generated numbers. The speed of generation is doubled when this option is used, producing

about twenty thousand random numbers per minute on the IBM 7040.

A special option is also available which allows moments for a new distribution to be supplied by the calling program without being read in on cards. This option may be particularly useful when dealing with emperical Bayesian statistics.

5.2 Method of Operation

If moments are given, the program calculates β_1, β_2 and K from which it determines the appropriate Pearson type. If raw point data are given, the program determines the appropriate Pearson type using moments calculated from the data.

Once 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. Pearson Types I, IV, III 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 errors such as attempting to take the square root of a negative number are encountered. Error messages are automatically printed under these conditions 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.) which takes on values from 0 to 1. The computer time, on the IBM 7040-1401 system, required for all operations up to this point is about two-thirds of a minute.

For the actual random number generation, a mixed congruential uniform random number generator is used to randomly select points on the c.d.f. from which the random numbers with the required distribution are determined. This generator is a library function at the Virginia Polytechnic Institute Computing Center and is not coded in the program. It was developed and tested by Kaercher (1962), using the method of Rotenberg (1960) and employes the congruence relation

$$x_{n+1} \equiv (2^7 + 1)x_n + 311715164025_{(8)} \pmod{2^{35}}$$

with $x_0 = 0$. The subscript on the constant denotes it to be in octal notation. The constant $311715164025_{(8)}$ was chosen since it is odd and and approximately equal to $[(.5 + \sqrt{3}/6)2^{35}]$ which was shown by Coveyou (1960) to cause the least serial correlation.

---

* It is not recommended that this program be used to generate normal random numbers since methods are available which are several times faster.

Kaercher reports satisfactory statistical results were obtained from X^2 tests, tests on the sample variances and means as well as checks of serial correlation.

Since there is, theoretically, a one to one mapping of uniform random numbers into the desired distribution, the properties of the generated sequence should be determined by those of the uniform random number generator used.

To obtain random numbers with the desired Pearson distribution, the original version of the program generates a uniform random number and begins at the zero end of the c.d.f. testing each value of it until finding the closest one to the given uniform random number. This value is the ordinate of the Pearson c.d.f. and the corresponding abscissa, being the required random number, is calculated.

This method requires, on the average, a very large number of tests especially for a J-shaped distribution and produces only about four hundred random numbers per minute. To increase the speed of generation, the program was modified so that it 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. The centiles are stored without requiring additional storage by sharing their locations with unused variables during this operation.

In the event 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. This procedure produces more

than ten thousand random numbers per minute.

5.3 Operating Information

The options obtained for various values of the arguments N1 and N2 used in the call statement of PURGE2 are given in Table 3.

Table 3. Values of the Arguments for PURGE2

Value of N1	Action Taken
1	Read *OPTION CARD and take action prescribed by it, i.e. read data cards, fit Pearson distribution, and generate 0 to 100 random numbers.
2	Generate up to 100 random numbers from a distribution previously described in the same program under the N1 = 1 or N1 = 3 option.
3	Fit a new Pearson curve from moments supplied by calling program via the common statement and generate 100 random numbers from this distribution.

Value of N2	Action Taken
1	Summations of first four powers of all numbers generated from this distribution are kept for later calculation of sample moments.
2	Summations of first four powers of all numbers generated from this distribution are used to calculate sample moments which are then printed.
3	The c.d.f. is punched on cards in binary along with necessary parameters for generating random numbers from this distribution then the same action is taken as for N2 = 2.
4	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.)
5	Random numbers are generated without calculating summations for computation of moments. (Moments should not be printed for a given distribution after this option is used.)

5.31* Data Input Options

For each fit under the N1 = 1 option, the following format applies:

The first data card to be read is the *OPTION CARD.

Table 4. Option Card Format

Cols.	Option	Description
1		*ASTERISK
2-3		Blank
4-6	Input Format Option	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 (which is to be given on the next data card) a positive integer is placed in this field. (This integer is equal to 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.
7-9		Blank
10-15	Input Number Option	There are two ways to specify the number of data entries of the sampling type. One is to enter the number of entries into this field. (The number of entries on the last card or record will be computed automatically.) The other is to place, on the end of the data, a record in which some two characters (columns) indicate the number of entries in the last entry card. (The remainder of that card should be blank.) This last method is indicated by leaving this field blank. For further clarification refer to the INPUT FORMAT CARD.

* This section is a modified version of the writeup of PURGE given by Cooper, Davis and Dono (1963).

Table 4, ctd.

Cols.	Option	Description
16-18		Blank
19-24	Output Generation Specifier	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.
25-27		Blank
28-30	Force Fit Option	This field enables the user to override the usual Pearson criterion and force the fit to a specific 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.
31-33		Blank
34-35	Input Tape Option*	If the user wishes to designate an input unit other than the normal card input, he may enter the symbolic tape unit in this field. If not, leave blank. (For moments and data points only.)
36-80		Blank

If the optional format control is desired for this fit, the next card should be the INPUT FORMAT CARD. This card is omitted if the optional format control is not used.

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

---

* Requires a IBM 7040 with 32K core.

the word format is omitted. It may be punched in columns 1-72. Regardless of the format used, two characters (columns) of the record (card) - characters to the right of data - must be reserved since this format also reads in the END OF DATA CARD. Therefore, all INPUT FORMAT CARDS will end with I2 conversion. For example: to read in input data four to the record, the following format might be used; (4X,4F17.8,6X, I2) - this particular format represents the built-in format used if the optional format is not elected. It reserves columns 79 and 80 for the two characters of the END OF DATA CARD. (Note: 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.

MOMENT CARD:

The first four moments are specified on the MOMENT CARD in four fields as outlined in Table 5.

Table 5. Moment Card Format

Variable Name	Cols.	Description
	1-4	- Blank
AVE	5-21	- Mean or first moment about zero
CM2	22-38	- Variance or second central moment
CM3	39-55	- Skewness or third central moment
CM4	56-72	- Kurtosis or fourth central moment
	73-80	- Blank

The format for each moment is F17.8, a 17 digit floating point number.

DATA POINT CARDS:

The DATA POINT CARDS (or records), used if the input is not moments, have the same format as the MOMENT CARD above when using the built-in format. 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 in the END OF DATA CARD. (Refer to INPUT FORMAT 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. This card is omitted when the Input Number Option is used or if sampling data input is not used.

END OF DATA CARD:

The END OF DATA CARD is used to indicate the end of the sampling data input entries when the Input Number Option is not used. It is a blank card except for the two columns which contain an integer specifying the number of entries on the last DATA POINT CARD. These two columns are specified on the INPUT FORMAT CARD. (Built-in format uses 79-80.)

5.32 Parameter Options

There are several other options available through the arguments N1 and N2 which deserve special attention.

The random numbers are returned to the calling program as implied arguments by the use of the FORTRAN common statement. The sequence of instructions

```
COMMON/Z2/RDS(100)
.
.
.
CALL PURGE2(1,1)
```

would cause PURGE2 to read data cards, fit a distribution with generation of up to 100 random numbers in the array RDS and exit, printing only the type of distribution fitted along with its parameters. The array in block Z2 must always be dimensioned at 100 even though fewer numbers may be generated per call as specified in columns 19-24 of the *OPTION CARD.

If the next call to PURGE2 is

```
CALL PURGE2(2,1)
```

another batch of up to 100 random numbers, from the same distribution, is stored in RDS without printing anything. This batch of random numbers represents a continuation of the sequence which began with the call with N1 = 1. If the graph of the distribution is desired along with the moments

```
CALL PURGE2(2,4)
```

should be the last call to PURGE2 for this distribution. This call will cause PURGE2 to generate another batch of random numbers placing them in RDS and to print the moments followed by the graph of the distribution. Since the graph routine shares storage with the c.d.f., hence destroying it, further generation can not be made from this distribution without refitting it.

In some cases, it is desirable to fit a distribution without reading input from cards. The following sequence of instructions will

cause a Pearson distribution to be fitted and 100 random numbers to be generated in RDS, printing only the type of curve fitted along with its parameters.

```
COMMON/Z1/AVE/Z2/RDS(100)/Z9/CM2,CM3,CM4,BETA1,BETA2,SKAPPA
.
.
.
AVE = 0.    (mean)
CM2 = 1.    (variance)
CM3 = 0.    (third central moment)
CM4 = 3.    (fourth central moment)
CALL PURGE2(3,1)
```

This particular case represents a N(0,1) distribution. Other distributions may be generated by inserting the proper moments.

If the next call to PURGE2 is

```
CALL PURGE2(2,3)
```

another batch of 100 random numbers, with the same distribution, will be placed in RDS, a binary deck of about 100 cards will be punched and the moments printed. This binary deck may be used as input to the MIX subroutine which generates mixtures of distributions as is discussed in Chapter VI.

If PURGE2 is called with N2 = 5, random numbers are generated about twice as fast by not calculating moments from them. If the moments calculated from the numbers generated up to this point are desired, they must be printed before this option is taken as they will be incorrect in any subsequent printout after this option is used.

5.33 Regeneration

Occasionally, it may be desired to regenerate a sequence of

random numbers or continue from the same sequence in a different computer run.

This can be accomplished by modifying the uniform random number generator used.* The instructions

```
COMMON/VPI001/NUMBR
NEXT = NUMBR          (store starting value of uniform
                       random generator)
CALL PURGE2(1,1)      (generate numbers)
NUMBR = NEXT          (set starting value of uniform
                       generator to its initial value)
CALL PURGE2(2,1)      (generate same numbers)
WRITE(7) NUMBR        (punch binary card with starting
                       number for uniform random number
                       generator)
```

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. Here the last value of NUMBR is punched on a card in binary and may be used as input in a later computer run to continue the sequence at the point it was terminated. The common statement must be used whenever NUMBR is modified.

The instructions

```
COMMON/VPI001/NUMBR
READ(5) NUMBR
CALL PURGE2(1,1)      (generate numbers)
```

---

* These modifications apply only to the random number generator used at the VPI Computing Center.

in the control program will continue the uniform random number sequence in a different computer run when used with the binary card output from a preceding run.

Note that the sequence of uniform random numbers generated will always be the same in different computer runs since the initial value of NUMBR is always zero. If desired, this difficulty may be overcome by setting NUMBR equal to some different arbitrary value at the beginning of each computer run. The COMMON statement must be used when this is done. All sequences of uniform random numbers which begin with the same value of NUMBR will be identical.

5.34 Variables Available thru COMMON

Table 6 gives the names of variables used in PURGE2 which may be placed in COMMON with the calling program along with their block name and use in PURGE2.

As an example of how these variables may be used, suppose the programmer wished to fit a Pearson distribution with PURGE2 and then make a test on Pearson's K in the calling program. The instructions

```
COMMON/Z9/CM2,CM3,CM4,BETA1,BETA2,SKAPPA
CALL PURGE2(1,1)
IF(SKAPPA .EQ. 0.)  XYZ = 0.
```

would set XYZ equal to zero if $K = 0$.

Table 6. Variables in PURGE2 Available thru COMMON

Block	Variable	Use
Z1	AVE	Mean of Pearson distribution
Z2	TOM(100)	Array used to store generated random numbers
Z3	LIMIT	Number of random numbers to be generated per call (0-100)
Z4	FA(4)	Noncentral moments calculated from generated data*
Z5	FAN(4)	Sums of first four powers of generated numbers*
Z6	N	Number of intervals of the c.d.f.**
Z7	KU(101)	Centile index for c.d.f.**
Z8	S(2641)	c.d.f.**
Z9	CM2	Variance of Pearson distribution
	CM3	Third moment of Pearson distribution
	CM4	Fourth moment of Pearson distribution
	BETA1	β_1 of Pearson distribution
	BETA2	β_2 of Pearson distribution
	SKAPPA	κ of Pearson distribution
Z10	FCM2	Variance of generated numbers*
	FCM3	Third moment of generated numbers*
	FCM4	Fourth moment of generated numbers*
	FBETA1	β_1 of generated numbers*
	FBETA2	β_2 of generated numbers*
	FKAPPA	κ of generated numbers*

* Not calculated for fast generation option.
** Not calculated unless random number generation is required.

5.35 Error Conditions

In the event that an error condition develops when fitting a curve, a diagnostic message is printed, LIMIT is set to zero and the subroutine exits normally. One such error condition arises when illegal values are obtained for β_1 and β_2. The values of β_1 and β_2 are restricted to a certain domain for distribution functions. The shaded portion of Figure 1 indicates the domain of these parameters.

It is the responsibility of the programmer to test for an error condition after the first call to PURGE2 for a given distribution. This may be done by the following sequence of instructions

```
COMMON/Z3/LIMIT
.
.
.
CALL PURGE2(N1,N2)

IF(LIMIT .EQ. 0)     (error, PURGE2 is unable to fit
.                     this distribution and calling
.                     program should skip to the next
.                     fit or stop)
```

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

It is possible that overflow conditions may occur either in computing moments from sampling data or in computing moments from generated data. Since both underflows and overflows occur as normal conditions in the program, it operates under no limit on underflow or overflow conditions. No warning is given when normal underflows or overflows occur.

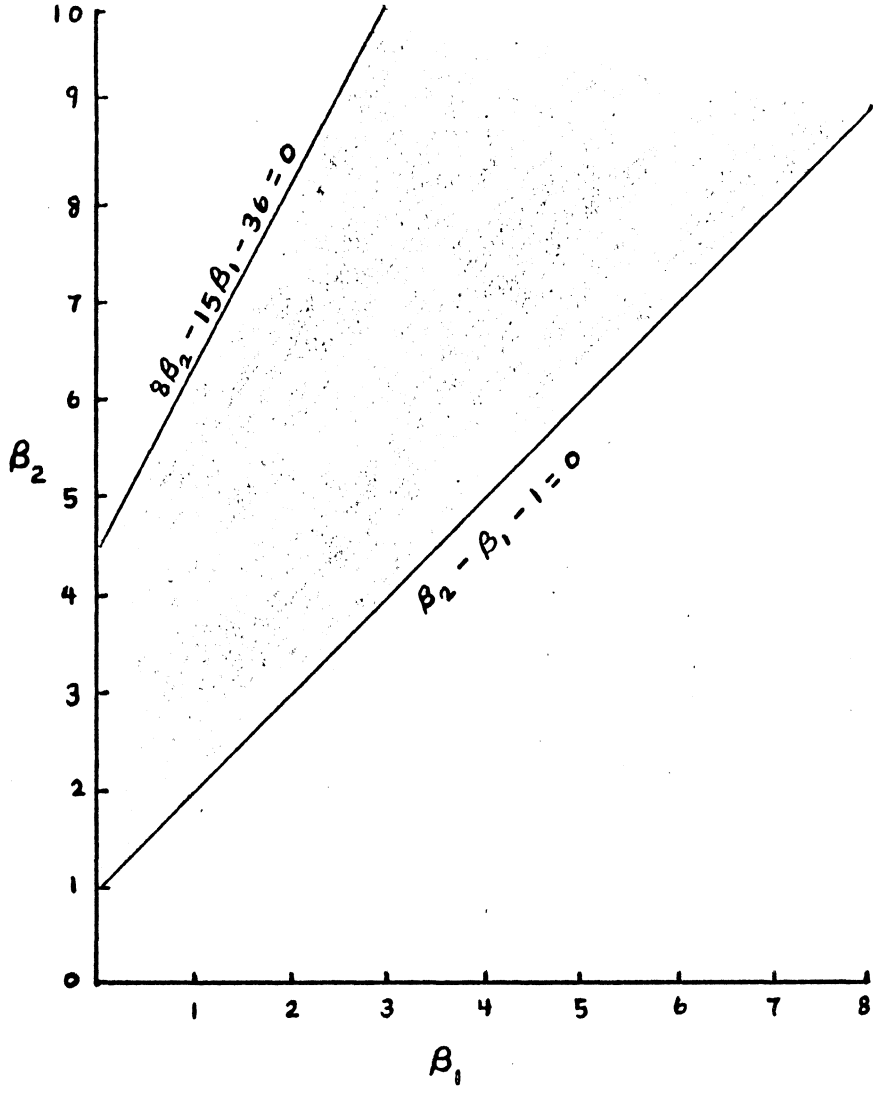


Figure 1. Admissable Region For β_1 and β_2.

The program tests for overflow when computing moments and should overflow occur when computing moments from data, LIMIT is set to zero (as a warning to calling program that PURGE2 was unable to fit a curve) and the diagnostic "TOO MUCH AND / OR TOO LARGE DATUM" is printed. 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, otherwise the program functions normally. Note, it is the responsibility of the programmer to test for overflow or underflow conditions which may occur in his program since it will also operate with no limit on overflow or underflow conditions. A flowchart of the PURGE2 subroutine and a FORTRAN IV listing for an IBM 7040 are given in the Appendix.

5.4 Sample Input and Output

The following computer program

```
COMMON/Z1/AVE/Z2/RDS(100)/Z9/CM2,CM3,CM4,BETA1,BETA2,SKAPPA
CALL PURGE2(1,4)
CALL PURGE2(1,4)
AVE = 50.
CM2 =  1.
CM3 =  2.
CM4 =  9.
CALL PURGE2(3,4)
STOP
END
                    (DATA see figure 2)
```

when used with the input data given in Figure 2 and the PURGE2 subroutine will produce Figures 3-5. Figure 6 illustrates the data deck for a fit from sampling data. Figure 7 gives a force fit of the normal distribution from the given moments. Figure 8 illustrates a data deck

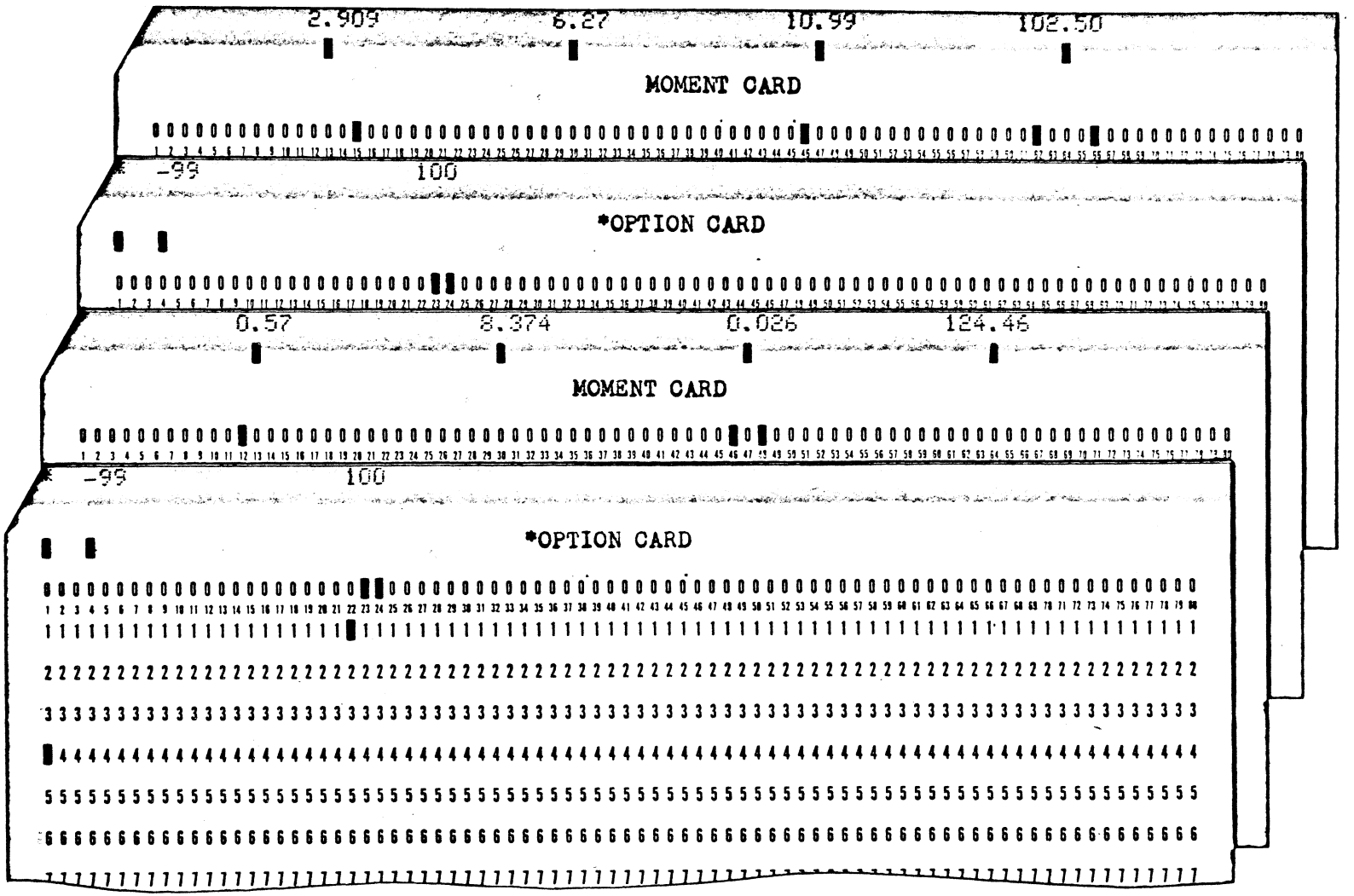


Figure 2. Moment Card Input

PEARSON CURVE TYPE ONE DISTRIBUTION GENERATOR

M1=-5.18874E-02 M2=-5.07221E-02 A1= 4.92270E 00 A2= 4.92875E 00 YE= 9.82661E-02

MOMENTS FROM DATA	
MEAN	0.57000000
VARIANCE	8.37399992
MU(3)	0.02600000
MU(4)	124.46000000
BETA 1	0.00000115
BETA 2	1.77485912
KAPPA	-0.00000038

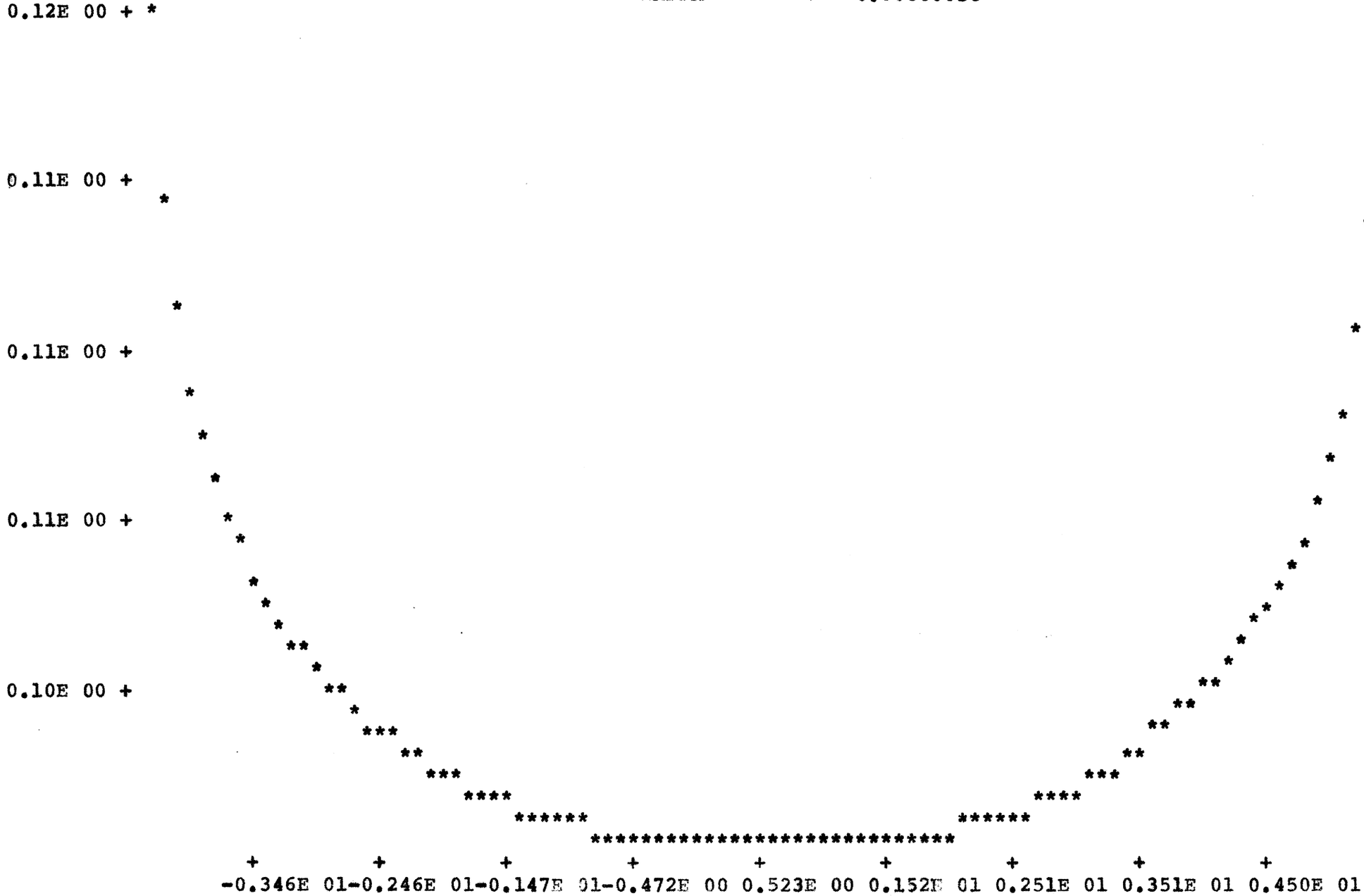


Figure 3. PURGE2 Output

PEARSON CURVE TYPE ONE DISTRIBUTION GENERATOR

M1=-1.08489E-01 M2= 1.08078E 00 A1= 3.26667E 00 A2= 7.62436E 00 YE= 1.24426E-01

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 100
MEAN	2.90900000	2.43103834	
VARIANCE	6.26999992	5.31912140	
MU(3)	10.99000000	14.08958480	
MU(4)	102.50000000	110.11172096	
BETA 1	0.48999626	1.31909683	
BETA 2	2.60728668	3.89182892	
KAPPA	-0.19060951	-0.62067799	

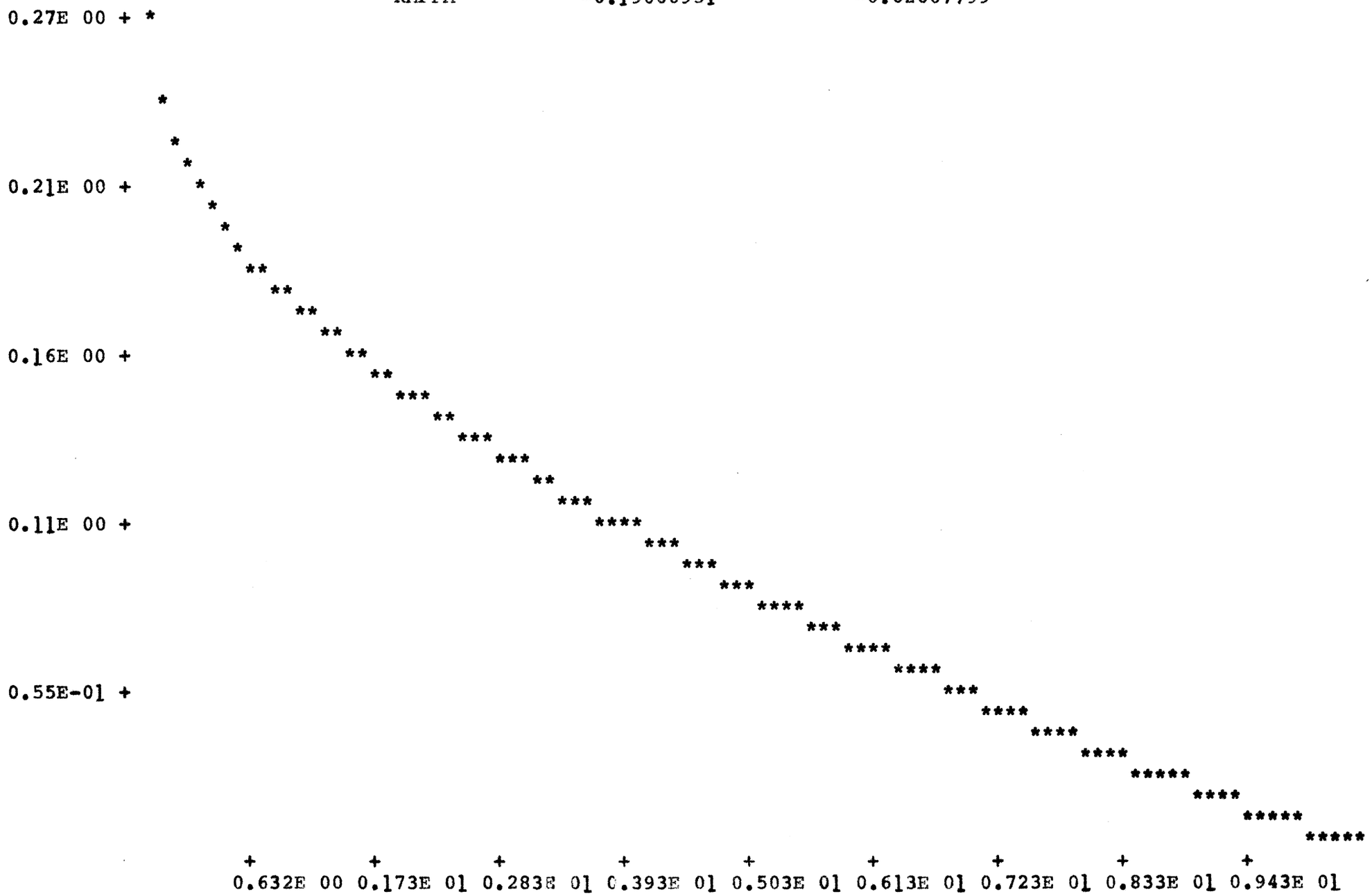


Figure 4. PURGE2 Output

SUBCLASS TYPE TEN DISTRIBUTION GENERATOR

G= 1.00000E 00 P= 0. A= 1.00000E 00 YE= 3.67879E-01

MOMENTS FROM DATA	
MEAN	50.00000000
VARIANCES	1.00000000
MU(3)	2.00000000
MU(4)	9.00000000
BETA 1	4.00000000
BETA 2	9.00000000
KAPPA	**********

(Asterisks indicate Kappa too large to be printed.)

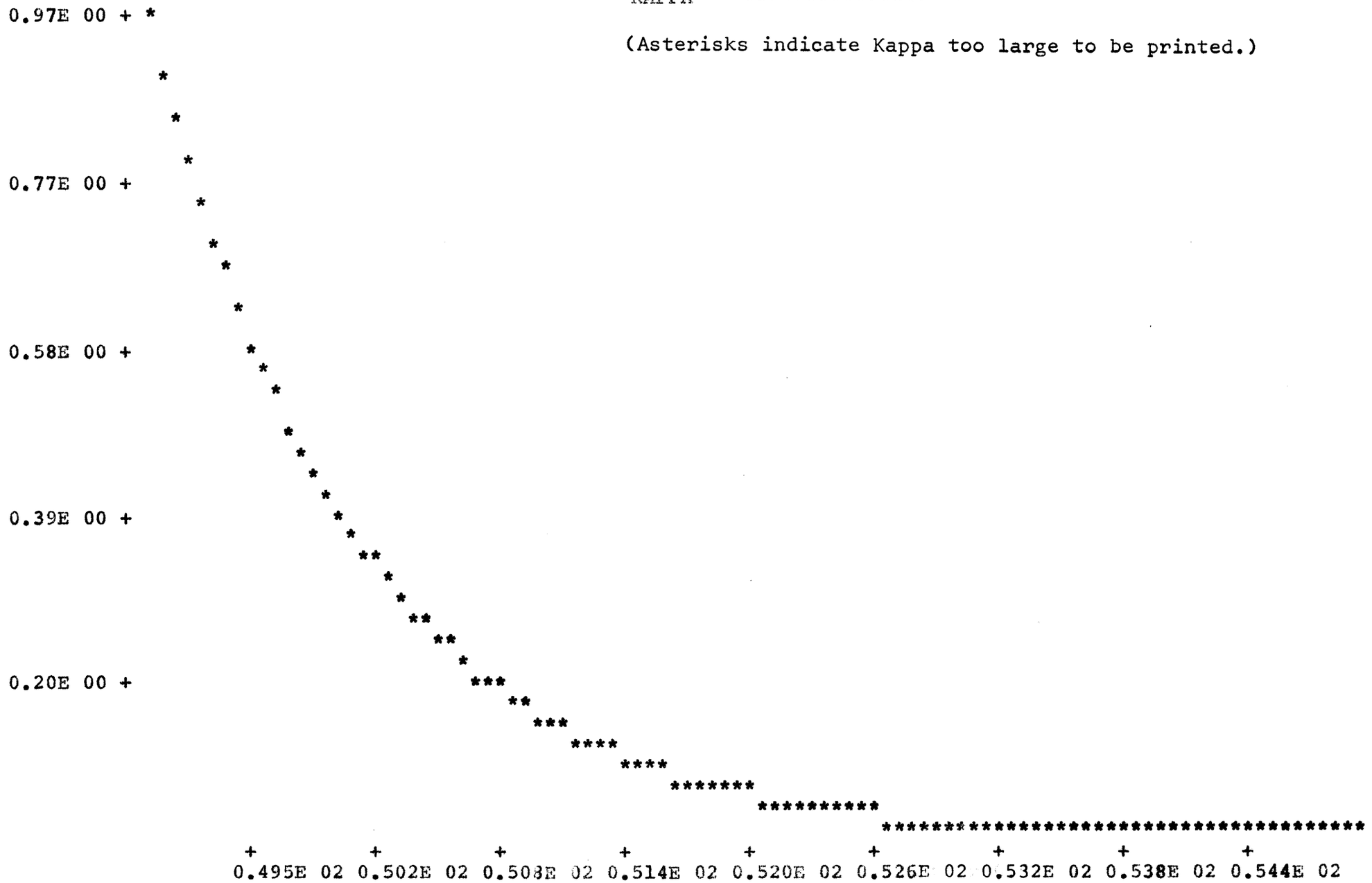


Figure 5. PURGE2 Output

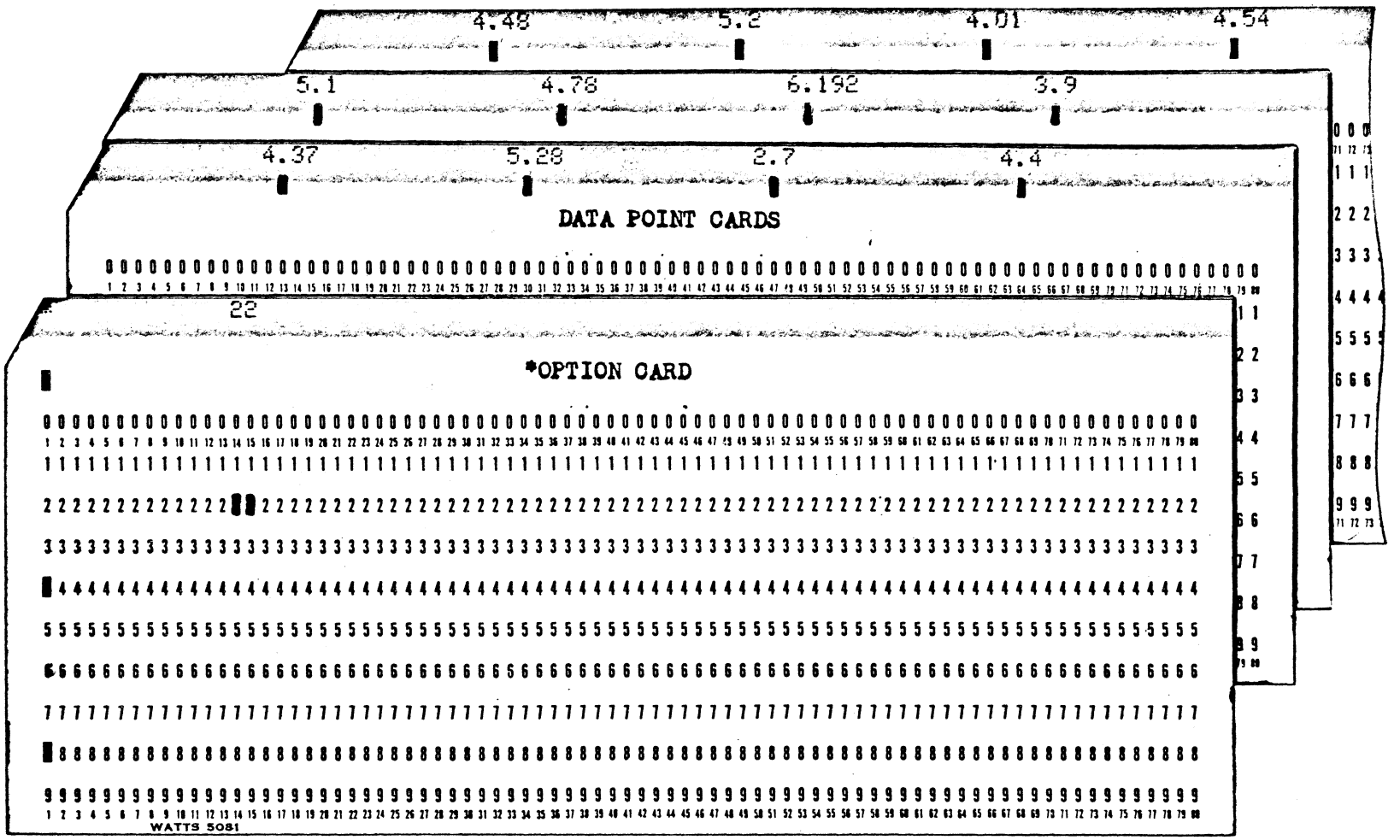


Figure 6. Data Point Card Input

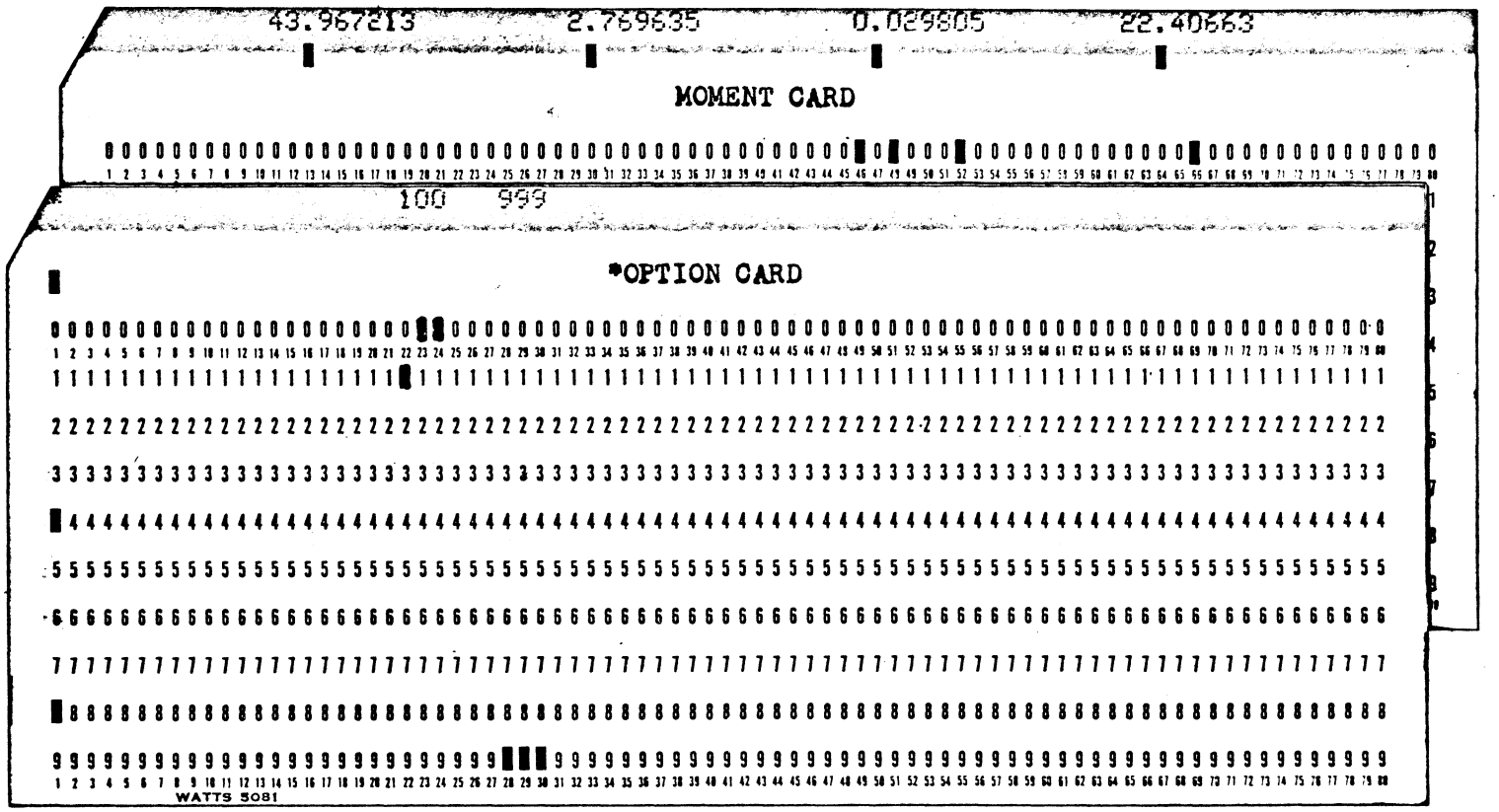


Figure 7. Moment Card Input Using Force Fit Option

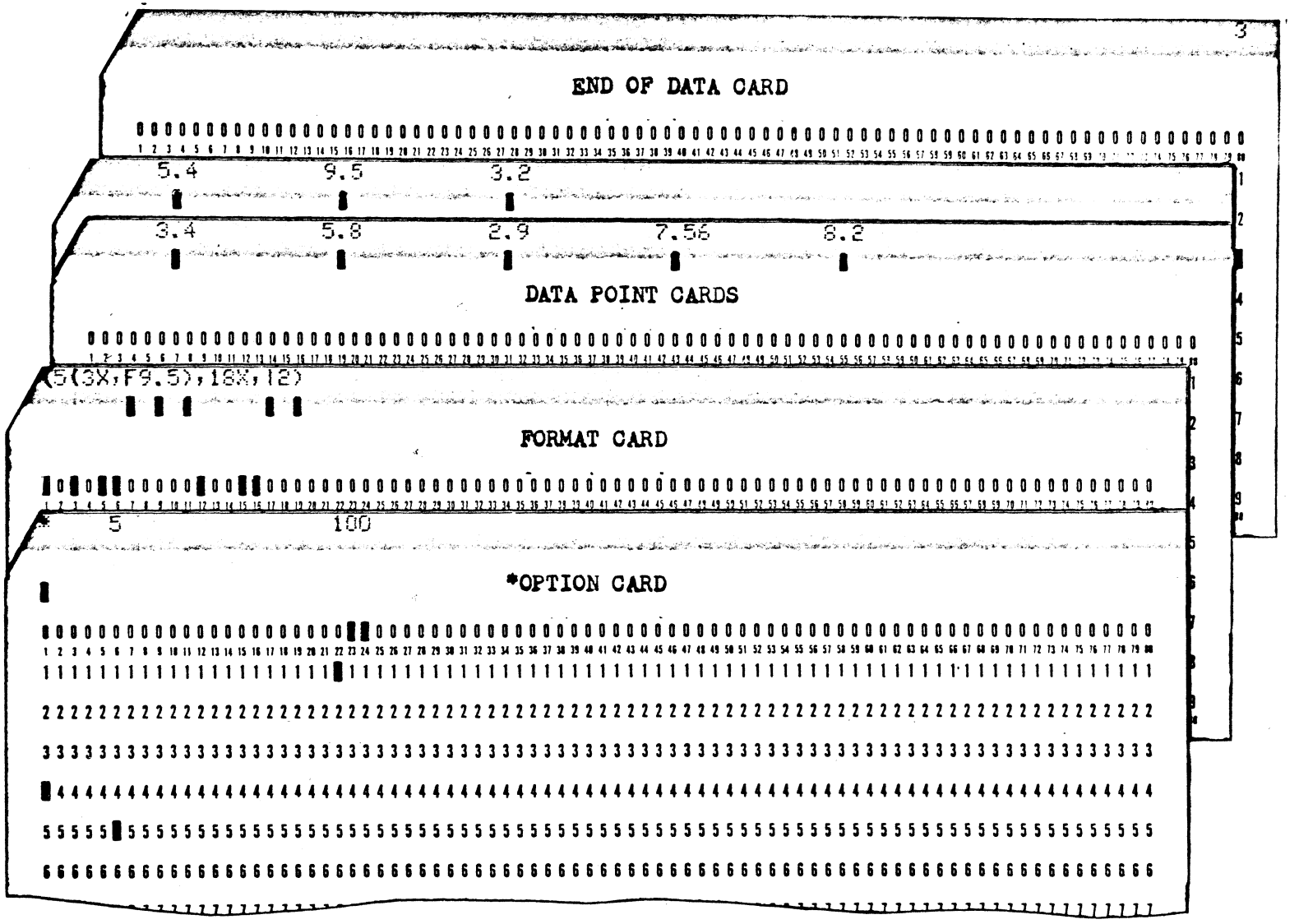


Figure 8. Data Point Card Input With Special Format

using the optional input format.

5.5 Tests of PURGE2

The performance of the aforementioned uniform random number generator is a major factor in the properties of the random numbers generated by PURGE2. Kaercher (1962) reports satisfactory statistical results were obtained from tests made on the uniform random number generator used in PURGE2 at the VPI Computing Center. See Section 5.2.

Several tests were made as a check on the performance of random numbers generated by PURGE2 and the results of all of them seem satisfactory.

Moments

For the X^2 distribution (Pearson's Type III) with 10 d.f., 50,000 random numbers were generated and the calculated moments compared with the true moments as a check on convergence. Table 7 gives values of the mean as well as the second, third and fourth central moments in intervals of 10,000 and also the true moments.

Table 7. Values of Sample Moments from a X^2_{10} Distribution

N	M_1'	M_2	M_3	M_4
10,000	10.0232	20.4682	85.5119	1737.8079
20,000	9.9809	20.4682	81.0195	1667.2398
30,000	9.9778	20.0569	81.2117	1675.9766
40,000	9.9638	19.9039	80.0241	1642.7236
50,000	9.9787	19.9659	79.6566	1634.1745
True	10.0000	20.0000	80.0000	1680.0000

These data suggest that the moments are converging to their true values. The sampling variability is, of course, larger in the higher order moments. Tables 8, 9, and 10 are obtained from the generation of 12,000 numbers from each of three Pearson Type I distributions being bell, J and U shaped respectively. Note, the graph of the distribution used in Table 9 appears as Figure 3 whereas that of Table 10 is Figure 4.

Table 8. Values of Sample Moments from a Bell Shaped Type I Distribution

N	M_1'	M_2	M_3	M_4
3,000	0.070	30.207	207.136	4728.115
6,000	0.093	31.242	221.975	5101.772
9,000	0.052	30.411	204.544	4627.790
12,000	0.0002	29.734	194.979	4352.521
True	0	29.501	190.112	4361.120

Table 9. Values of Sample Moments from a J-Shaped Type I Distribution

N	M_1'	M_2	M_3	M_4
3,000	0.508	8.824	2.633	143.696
6,000	0.597	8.819	1.878	142.630
9,000	0.593	8.805	1.864	142.416
12,000	0.571	8.803	2.193	142.762
True	0.570	8.374	0.026	142.460

Table 10. Values of Sample Moments from a U-Shaped Type I Distribution

N	M_1'	M_2	M_3	M_4
3,000	2.915	6.307	11.254	103.673
6,000	2.925	6.287	11.040	103.028
9,000	2.901	6.129	10.699	99.112
12,000	2.898	6.115	10.687	98.734
True	2.909	6.270	10.990	102.500

Some alternative tests which could be used on PURGE2 are X^2 goodness of fit tests, the Kolmogorov statistic and testing random numbers on a similar problem for which the answer is known. It should be noted that the results of tests on runs up and down made on the uniform random number generator carry over directly here because of the one to one correspondence of the uniform random numbers with those from a Pearson distribution using this method of generation.

The following remarks made by Hull and Dobell (1964) seem appropriate here. "... no finite class of tests can guarantee the general suitability of a finite sequence of (random) numbers. Given a set of tests, there will always exist a sequence of numbers which passes these tests but which is completely unacceptable for some particular application.

The point is that the appropriate tests are entirely dependent on the application for which the random numbers are needed."

VI. MIX SUBROUTINE

The MIX subroutine was written to overcome the limitation, inherent in PURGE2, of being restricted to generating only unimodal Pearson distributions. Under certain conditions, this subroutine will generate random numbers from continuous distributions of almost any shape at speeds up to ten times faster than PURGE2, i.e. more than 100,000 random numbers per minute on the IBM 7040. This great increase in speed is accomplished by using large proportions of uniform and normal random numbers 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 numbers in the mixture increases.

6.1 Description and Purpose

The greater flexibility in the shape of distributions which may be generated by the MIX subroutine is achieved by a probability mixture of any combination of one or two Pearson distributions with a uniform distribution, on any interval, and also a normal distribution with any mean and variance.

The desired Pearson distributions are generated by PURGE2 and cards containing the c.d.f. along with the necessary parameters for generating random numbers are punched in binary for later input to MIX. (Both programs, written in FORTRAN IV, could be combined on a 32K IBM 7040 system to eliminate the card input-output for greater efficiency.)

A control card specifying the proportions of the various dis-

tributions to be mixed followed by the cumulative distribution functions of the Pearson distributions is then fed in as input to MIX. Using the c.d.f. of the desired Pearson distribution, MIX is able to generate random numbers from that distribution in the same way as PURGE2. The proportion of distributions mixed may later be modified by the control program. The MIX subroutine may also be used to generate random numbers from a single Pearson distribution without refitting the curve for each computer run.

Output options included printing theoretical moments along with moments computed from the generated numbers and also a graph of the distribution.

6.2 Method

The density function f of a random variable x, which we wish to generate, may be 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 $\sum_{i=1}^{4} a_i = 1$, $a_i \geq 0$ all i.

To obtain the moments of f, we have

$$EX^r = \int_R x^r f(x)\,dx = \sum_{i=1}^{4} a_i \int_{R_i} x^r f_i(x)\,dx$$

where R_i is the domain of definition of f_i whereupon

$$EX^r = \quad u_r' \quad = \sum_{i=1}^{4} a_i u_{r_i}' \,.$$

The MIX subroutine first computes the moments about zero for each distribution, multiplies them by the appropriate probability, sums them and converts the second thru fourth moments to central moments.

For random number generation, the program generates a uniform (0,1) random number which determines the distribution from which to generate x with distribution f. To illustrate the procedure suppose we have

$$f(x) = .50f_1(x) + .25f_2(x) + .20f_3(x) + .05f_4(x) .$$

A uniform random variable u is generated. If $u < .50$, x is generated from distribution f_1; if $.50 \leq u < .75$, x is generated from f_2; if $.75 \leq u < .95$, x is generated from f_3; and if $.95 \leq u < 1$, x is generated from f_4. After x has been generated, this process is repeated 100 times per call to MIX and if required, moments are calculated from the generated numbers for comparison with their theoretical values.

The speed of generation, in numbers per minute on the IBM 7040 using the uniform and normal random number generators available at the VPI Computing Center, is approximately given by

$$n = 10^4(a_1 + a_2) + 3x10^4a_3 + 10^5a_4$$

where a_1 and a_2 represent probabilities of generating from Pearson distributions, likewise a_3 for the normal and a_4 the uniform. If the a_i are chosen such that a_1 and a_2 are small in comparison with a_3 and a_4 as suggested in papers by Marsaglia (1961 and 1964), very fast methods of generation are obtained.

If desired, a graph of the density function may be obtained from

10,000 random numbers with the required distribution. The numbers are first generated to determine their range. The range is then divided into 100 equally spaced intervals and the numbers are regenerated to determine the frequency in each interval.

At this point an option to smooth the data, using moving averages of order five may be elected. In order not to drop 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

$$x_2 = (x_1 + x_2 + x_3)/3, \qquad x_{99} = (x_{98} + x_{99} + x_{100})/3 .$$

This technique is used to preserve the end points of J and U shaped distributions and appears to give good results.

By summing the area of the 100 rectangles and requiring it to be unity, an ordinate scale is obtained whence the curve is plotted. The scale is more accurately determined if smoothing is used.

6.3 Operating Information

The MIX subroutine is entered by the FORTRAN statement

```
OALL MIX(N1,N2)
```

where N1 and N2 are arguments which describe the option desired.

The options obtained for various values of the arguments are given in Table 11.

Table 11. Values of the Arguments for MIX

Value of N1	Action Taken
1	Read CONTROL CARD and if necessary binary cards for Pearson distribution(s). Generate 100 random numbers from the mixture.
2	Generate 100 random numbers from a mixture of distributions previously described under the N1 = 1 option.

Value of N2	Action Taken
1	Summations of first four powers of all numbers generated from this mixture are kept for later calculation of sample moments.
2	Summations of first four powers of all numbers generated from this mixture are used to calculate sample moments which are then printed.
3	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.)
4	Same as for N2 = 3 except that a graph with smoothing is printed. (Must be last option taken for a given distribution.)
5	Random numbers are generated without calculating summations for computation of moments.

6.31 Data Input Option

For each fit under the N1 = 1 option the following format applies:

The first data card to be read is the CONTROL CARD followed by up to two binary decks produced by PURGE2. If more than one fit per run is desired, the next card would be a new CONTROL CARD followed by the necessary binary decks.

CONTROL CARD

The CONTROL CARD format is given in Table 12. The format for

probabilities is F7.4, a seven digit floating point number. The format for the other variables is F10.5, a ten digit floating point number.

Table 12. Control Card Format

Variable Name	Cols.	Description
P(1)	1-7	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.)
P(2)	8-14	Probability of selection of second Pearson distribution.
G	15-21	Probability of selection of normal distribution.
GM	22-31	Mean of normal distribution.
GV	32-41	Variance of normal distribution.
U	42-48	Probability of selection of uniform distribution.
UL	49-58	Lower endpoint of uniform interval.
UP	59-68	Upper endpoint of uniform interval.
	69-80	Blank.

Note: If there is a number greater than zero but less than one in P(1) or P(2) then a Pearson binary deck produced by PURGE2 must follow the control card. If both are non-zero then two such decks must follow the control card.

6.32 Regeneration

When it is desired to regenerate a sequence of random numbers or continue from the same sequence in a different computer run, the

uniform random number generator must be modified in the manner described in Section 5.33. In addition, the normal distribution generator must be modified if it is used.

The normal random number generator used at the VPI Computing Center was developed and tested by Marsaglia, MacLaren and Bray (1963). It may be modified by placing the following FORTRAN cards in the calling program:

```
      COMMON/VPI001/NUMBR/Z11/Q
      NEXT = NUMBR                    (store starting value of uniform
                                       generator)
      Q = RNEX(DUMMY)                 (store starting value of normal
                                       generator)
      CALL MIX(N1,N2)                 (generate numbers)
      NUMBR = NEXT                    (reset uniform generator to its
                                       original starting value)
      DUMMY = RNST(Q)                 (reset normal generator to its
                                       original starting value)
      CALL MIX(N1,N2)                 (regenerate numbers)
      Q = RNEX(DUMMY)                 (store starting value for normal
                                       generator, needed to continue
                                       sequence from this point)
      WRITE(7) NUMBR, Q               (punch binary card with starting
                                       values of uniform and normal gen-
                                       erators which will continue se-
                                       quence from this point)
```

If it is desired to continue the same sequence in a different computer run, the instructions

```
      COMMON/VPI001/NUMBR/Z11/Q
      READ(5) NUMBR, Q                (read binary card, punched in pre-
                                       vious run, containing starting
```

values of uniform and normal generators needed to continue sequence)

```
DUMMY = RNST(Q)          (set normal generator to its starting value)

CALL MIX(N1,N2)          (continue generation)
```

placed in the calling program will cause the binary card punched in the previous run to be read and the sequence of random numbers to be continued from the point at which it was terminated.

6.33 Variables Available thru COMMON

The calling program may also modify the CONTROL CARD variables by using the FORTRAN COMMON statement. See Table 13 for names and block numbers of variables available thru COMMON.

6.34 Error Conditions

The MIX subroutine tests the probability multipliers (no check is made for erroneously punched negative values) of the distributions to make sure they sum to one. They are rejected 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 same manner as in PURGE2. The subroutine exits printing the diagnostic message "MPYRS DO NOT SUM TO 1, REQ FOR DIST FUN".

Overflow when computing moments from generated data is handled in the same way as in PURGE2 (see Section 5.35).

A flowchart of the MIX subroutine and a FORTRAN IV listing for an IBM 7040 are given in the Appendix.

Table 13. Variables in MIX Available thru COMMON

Block	Variable	Use
Z1	TM1	Theoretical mean of mixture
Z2	TOM(100)	Generated random numbers
Z3	LIMIT	Number of random numbers to be generated per call (100)
Z4	FA(4)	Noncentral moments calculated from generated data*
Z5	FAN(4)	Sums of first four powers of generated numbers*
Z6	N(2)	Number of intervals of the c.d.f.
Z7	KU(2,101)	Centile index for c.d.f.
Z8	S(2,2641)	c.d.f.
Z9	CM2	Theoretical variance of mixture
	CM3	Theoretical third moment of mixture
	CM4	Theoretical fourth moment of mixture
Z10	FCM2	Variance of generated distribution*
	FCM3	Third moment of generated distribution*
	FCM4	Fourth moment of generated distribution*
Z11	Q	Starting value for normal random number generator
Z12	P(1), P(2), G, GM, U, UL, UP	CONTROL CARD variables in the order they appear on it

* Not calculated for N2 = 5 option.

6.4 Sample Input and Output

The following computer program

```
COMMON/Z2/RDS(100)
CALL MIX(1,4)
CALL MIX(1,4)
STOP
END
```

(DATA see Figurure 9)

when used with the input data given in Figure 9 produces Figures 10 and 11. Figure 10 is derived from a mixture of 30% of the U shaped distribution shown in Figure 3 and 70% N(0,1). Figure 11 is a mixture of 25% each of a J shaped distribution with moments 0.051, 4.266, -7.688 and 48.154, the distribution shown in Figure 3 (U shape), N(0,2) and uniform on (-1,0). The smoothing option was used for both graphs.

Figure 12 is a graph of 10,000 observations from a N(0,1) distribution without smoothing whereas Figure 13 is a graph of the same data with smoothing.

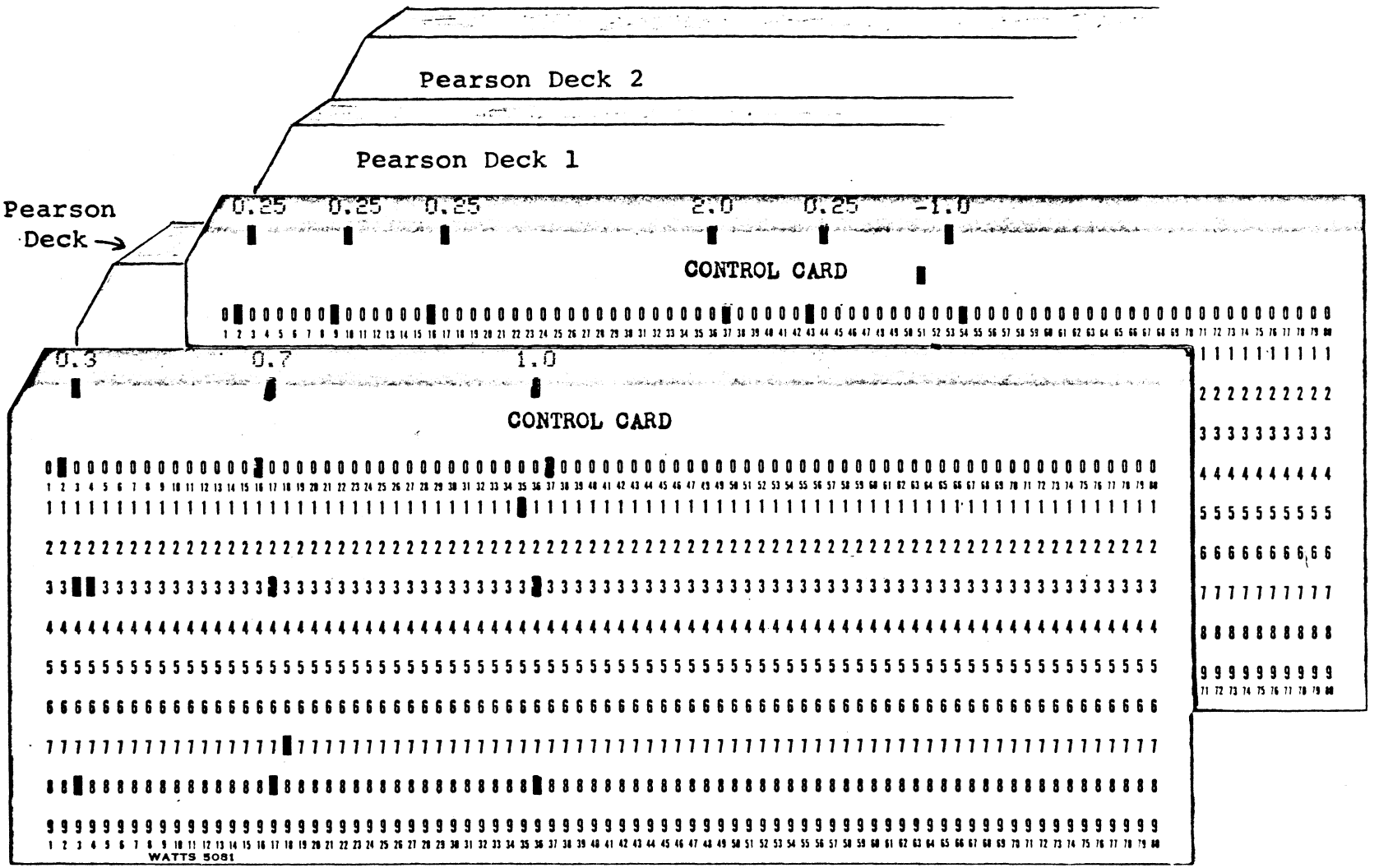


Figure 9. Input For Mix

DISTRIBUTION MIXTURES

CONTROL CARD P1= 0.3000 P2= -0.0000 NORM= 0.7000 MEAN= -0.00000 VAR= 1.00000

UNIF= -0.0000 FROM -0.0000 TO -0.0000

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 10000
MEAN	0.17100000	0.03850844	
VARIANCE	3.28042888	3.34579096	
MU(3)	2.67135952	2.14223446	
MU(4)	41.98112896	45.84792448	

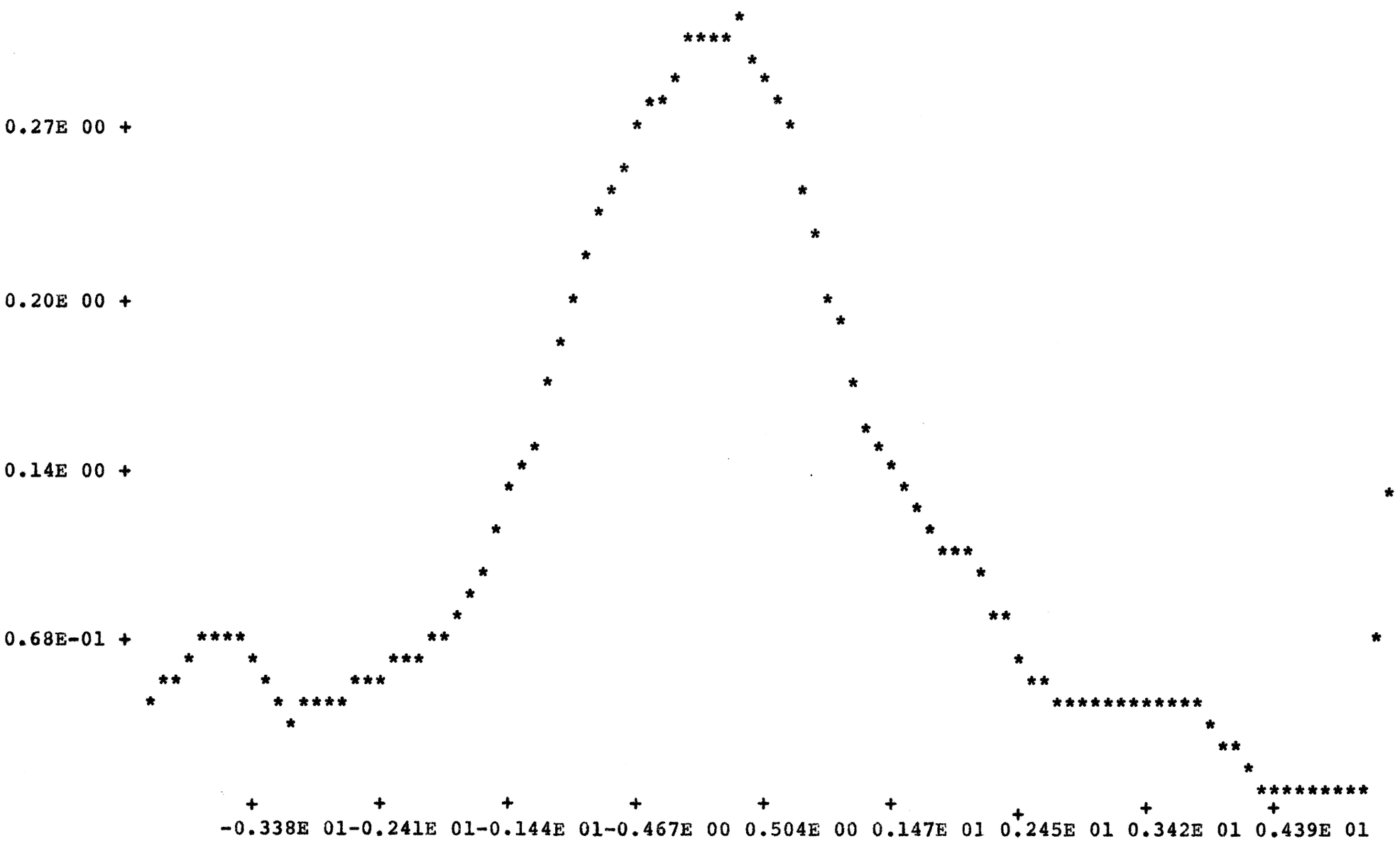


Figure 10. MIX Output

DISTRIBUTION MIXTURES

CONTROL CARD P1= 0.2500 P2= 0.2500 NORM= 0.2500 MEAN= -0.00000 VAR= 2.00000

UNIF= 0.2500 FROM -1.0000 TO -0.0000

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 10000
MEAN	0.03025000	-0.00531474	
VARIANCE	3.82429340	4.12326840	
MU(3)	1.46430850	0.81013203	
MU(4)	49.75215488	57.12495424	

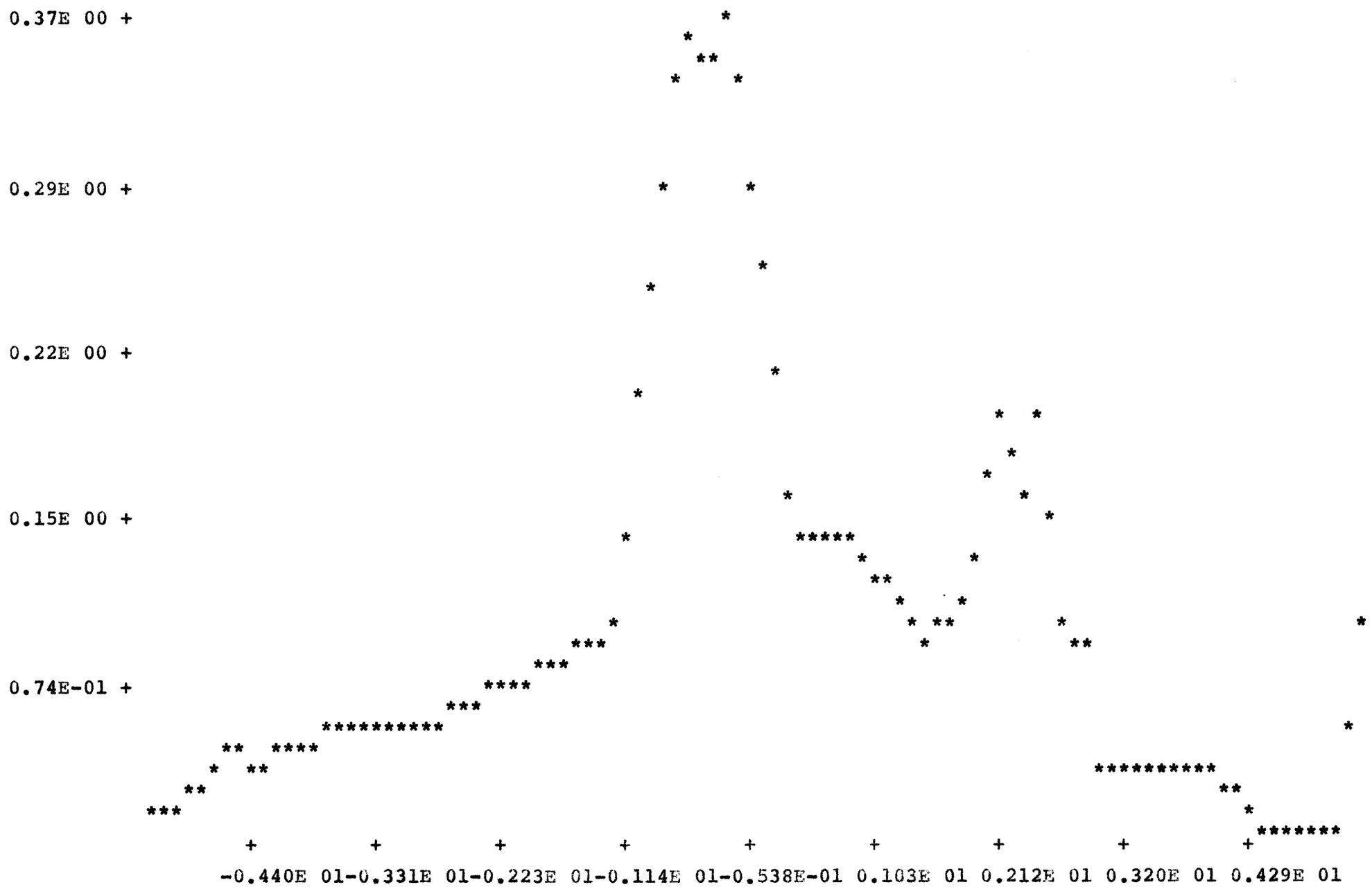


Figure 11. MIX Output

DISTRIBUTION MIXTURES

CONTROL CARD P1= -0.0000 P2= -0.0000 NORM= 1.0000 MEAN= -0.00000 VAR= 1.00000

UNIF= -0.0000 FROM -0.0000 TO -0.0000

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 10000
MEAN	-0.00000000	0.04899041	
VARIANCE	1.00000000	0.97922736	
MU(3)	0.00000000	0.05991855	
MU(4)	3.00000000	2.78555760	

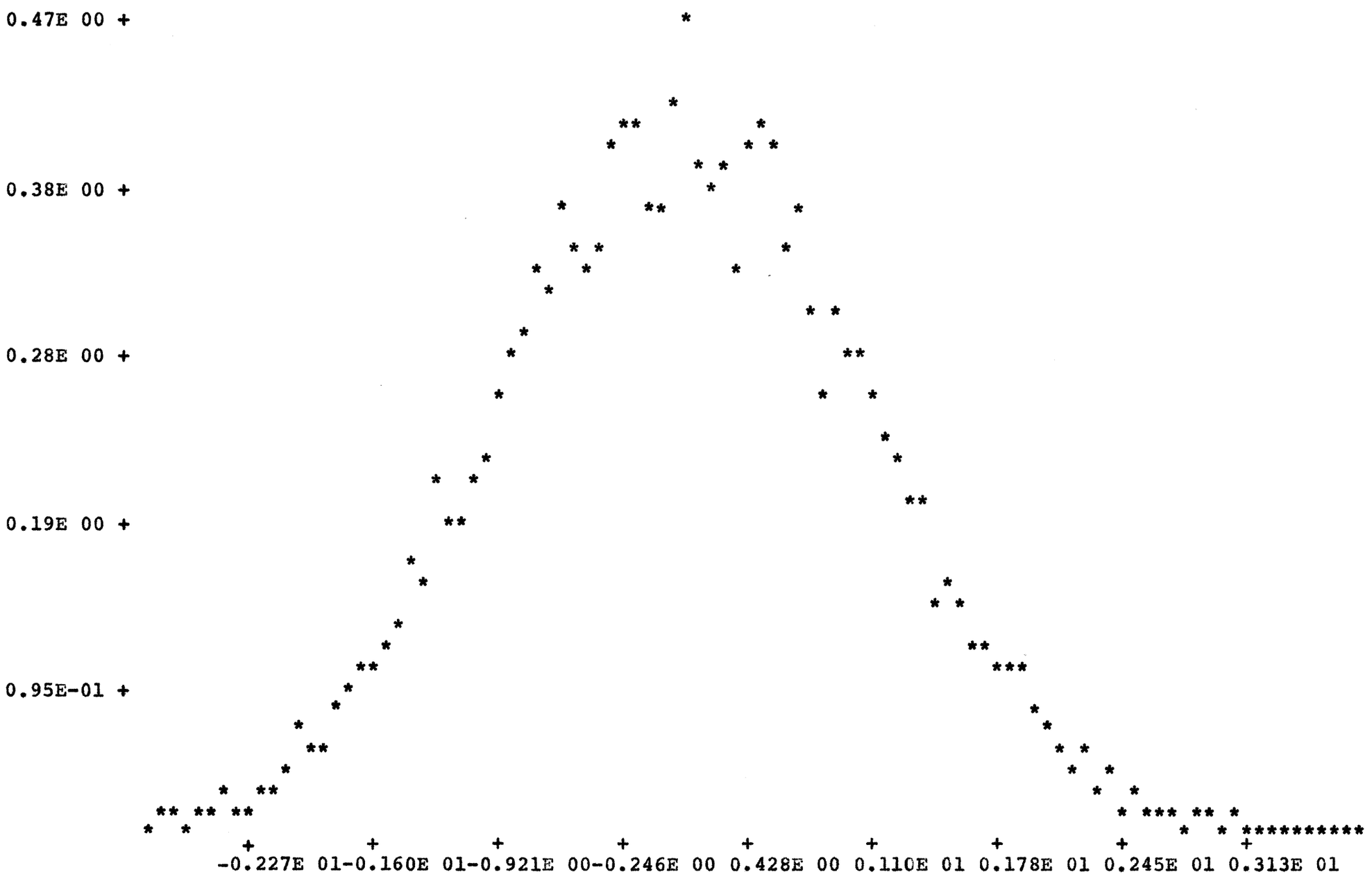


Figure 12. MIX Output

DISTRIBUTION MIXTURES

CONTROL CARD P1= -0.0000 P2= -0.0000 NORM= 1.0000 MEAN= -0.00000 VAR= 1.00000

UNIF= -0.0000 FROM -0.0000 TO -0.0000

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 10000
MEAN	-0.00000000	0.04899041	
VARIANCE	1.00000000	0.97922736	
MU(3)	0.00000000	0.05991855	
MU(4)	3.00000000	2.78555760	

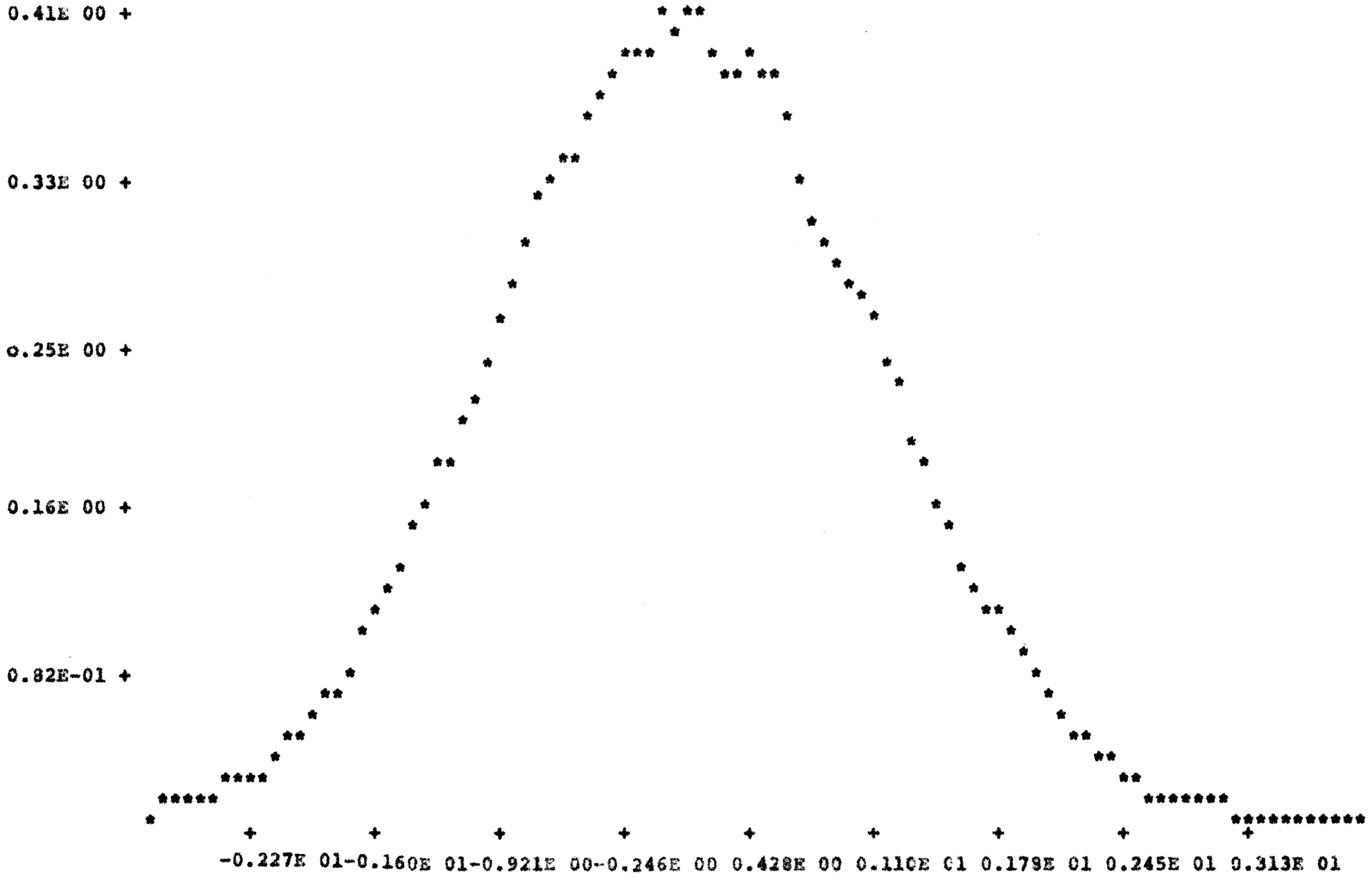


Figure 13. MIX Output

CONTROL CARD P1= 0.4000 P2= -0.0000 NORM= 0.4000 MEAN= 8.00000 VAR= 10.00000

UNIF= 0.2000 FROM 8.0000 TO 10.0000

MOMENTS	FROM ORIGINAL DATA	FROM GENERATED DATA	N= 10000
MEAN	8.99999984	9.03124128	
VARIANCE	12.86666768	12.39445872	
MU(3)	44.00000000	41.10000608	
MU(4)	992.84007936	835.58983680	

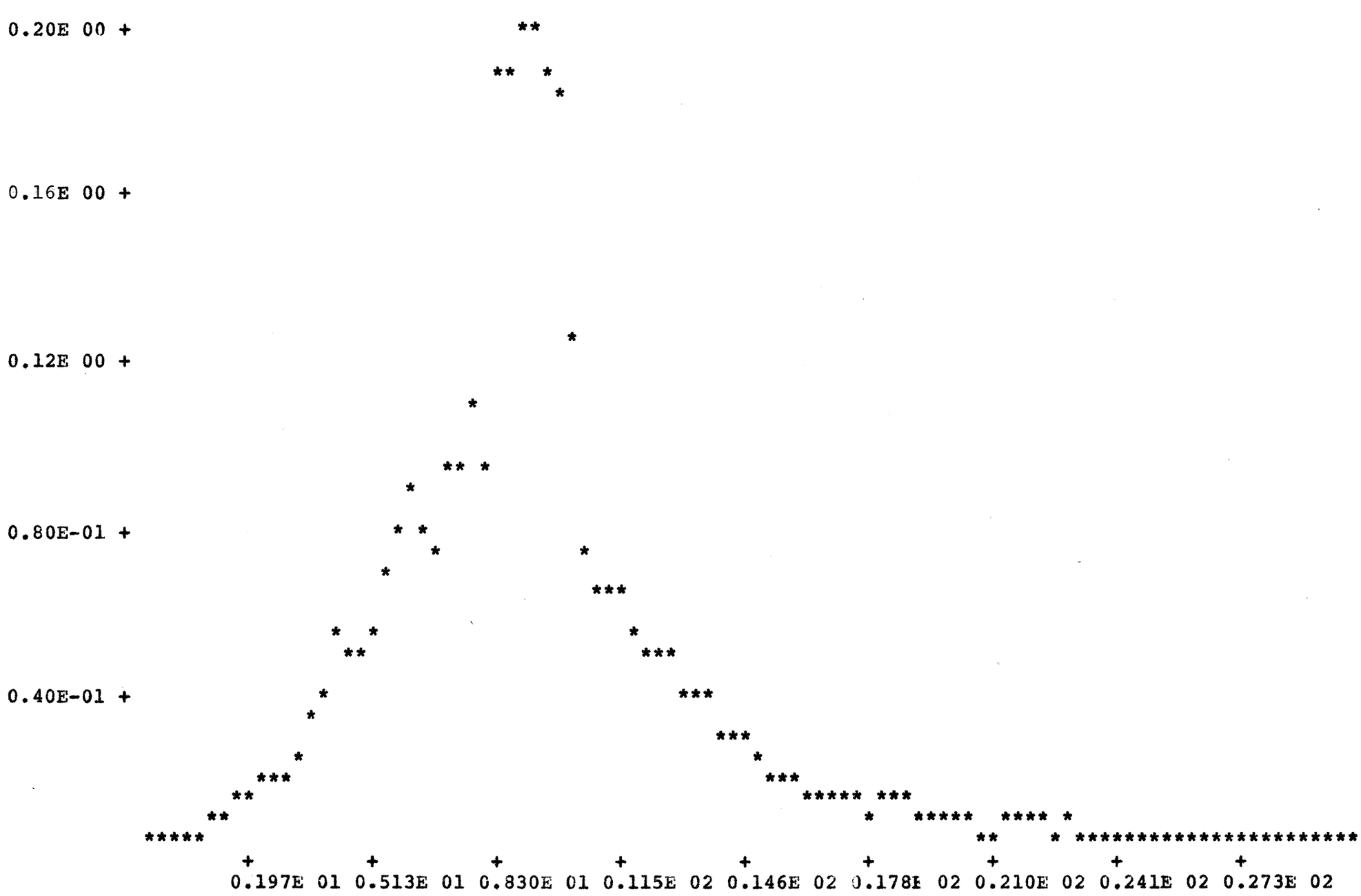


Figure 14. MIX Output

VII. APPLICATION TO INVESTIGATION OF ROBUSTNESS

7.1 Confidence Interval Problem

As a practical example of how the foregoing computer programs may be used, consider the problem of investigating the effects of departure from normality of the parent population on three methods of estimating confidence intervals for the mean μ with confidence coefficient $1 - \alpha$, where the true mean and variance are unknown.

7.2 Methods of Estimation

The methods of confidence interval estimation under consideration are:

(1) the standard procedure using the sample standard deviation given by

$$\bar{x} \pm \frac{s}{\sqrt{n}} t_{(\alpha/2, n-1)} \tag{7.1}$$

where

$$s^2 = \sum_{i=1}^{n} (x_i - \bar{x})^2/(n - 1)$$

and $t_{(\alpha/2,n-1)}$ is the appropriate percentage point of the t-distribution with significance level $\alpha/2$ and $n - 1$ degrees of freedom,

(2) a procedure using the sample mean absolute deviation

$$\bar{x} \pm mK_{(\beta,n)} \tag{7.2}$$

where

$$m = 1/n \sum_{i=1}^{n} |x_i - \bar{x}|$$

is the sample mean absolute deviation of the sample of size n and $K_{(\beta,n)}$

is tabulated by Krutchkoff (1965) for n = 3(1)10, β = .80, .90, .95, .99 and .999,

(3) a procedure, described by Noether (1955), using the sample range

(7.3) $$\bar{x} \pm g_{\alpha} R$$

where R is the range of the sample of size n and g_{α} is the critical value tabulated by Jackson and Ross (1955) for n = 2(1)15, α = .01, .05, and .10.

These methods assume an independent sample of size n from a normal population and under these conditions it is well known in statistical theory that the method using the sample standard deviation gives the confidence interval with the shortest expected length. The sample standard deviation is however, somewhat difficult to calculate without the aid of some type of computing equipment.

The sample mean absolute deviation is much easier to calculate since no square roots are involved and Krutchkoff (1965) shows that the "sacrifice" or fractional increase in interval length is small being on the order of two to five percent except when an excessively high confidence on the basis of very few observations is desired. He also shows there is something to be gained by using the sample mean absolute deviation if one more observation is allowed than with the sample standard deviation. In this case the fractional decrease in length ranges from about four to ninety five percent being largest when an excessively high confidence based on a few observations is desired, the gain produced being larger than the sacrifice.

The range method of estimating confidence intervals is the

easiest to compute. When the underlying distribution is normal, the expected increase in interval length using the range is shown by Noether (1955) to be negligible for all practical purposes.

7.3 Monte Carlo Example

One way of investigating the robustness of the three confidence interval estimation methods, to departures from the normality assumptions, is to use Monte Carlo techniques. As an illustration of the type of results which may be obtained by this technique, the PURGE2 and MIX subroutines are used to generate large sequences of random numbers from various distributions for which the true means are known. These numbers are used to compute (95% normal theory) confidence intervals for sample sizes of five and ten by the three methods. The mean interval lengths are computed and since the true means are known, the confidence levels are also computed.

A summary of the results of these calculations is contained in Tables 15 thru 21. The first column in these tables specifies the total number N of numbers generated for the distribution. Several values of N are given in each table in order to indicate how the data is converging.

Table 14 gives the distributions used in Tables 15 thru 21 along with the moments of the distribution. Table 15, computed from a N(0,1) distribution, is provided for comparison with the other tables.

The graphs of the distributions used in Tables 19 and 20 appear

as Figures 3 and 4 respectively. Table 21 is computed from a mixture of 40% X^2_{10}, 40% N(8,10) and 20% uniform on (8,10). The graph of this distribution is obtained from the MIX subroutine without smoothing and appears in Figure 14.

Table 14. Types of Curves Used in Tables 15 Thru 21

Table	Distribution	Shape	μ'_1	μ_2	μ_3	μ_4
14	Normal	Bell	0.	1.000	0.	3.000
15	Type III (X^2_{10})	Bell	10.000	20.000	80.000	1680.000
16	Type I	Bell	0.	29.501	190.112	4361.120
17	Type I	J	0.051	4.266	-7.688	48.154
18	Type I	U	0.570	8.374	0.026	124.460
19	Type I	L	2.909	6.270	10.990	102.500
20	Mixture	Bell	9.000	12.867	44.000	992.840

Table 15. Confidence Interval Data (normal)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
17	0.949	2.317	0.952	2.403	0.951	2.357
18	0.948	2.317	0.952	2.401	0.949	2.357
19	0.947	2.313	0.952	2.397	0.948	2.353
20	0.947	2.317	0.952	2.401	0.949	2.356
			Sample Size Ten			
17	0.944	1.382	0.947	1.423	0.943	1.404
18	0.942	1.382	0.946	1.422	0.942	1.404
19	0.941	1.381	0.945	1.420	0.941	1.403
20	0.942	1.382	0.946	1.422	0.942	1.405

Table 16. Confidence Interval Data (X^2_{10})

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
20	0.936	6.164	0.938	6.297	0.932	6.225
30	0.938	6.162	0.940	6.292	0.935	6.224
40	0.937	6.144	0.940	6.272	0.937	6.206
50	0.936	6.155	0.939	6.286	0.936	6.218

Table 17. Confidence Interval Data (Type I Bell)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
3	0.928	12.384	0.932	12.810	0.925	12.479
6	0.927	12.601	0.936	13.096	0.927	12.627
9	0.928	12.491	0.934	12.959	0.929	12.551
12	0.926	12.376	0.931	12.825	0.928	12.440

Table 18. Confidence Interval Data (Type I J)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
3	0.895	4.757	0.897	5.051	0.892	4.615
6	0.900	4.803	0.899	5.109	0.896	4.638
9	0.903	4.823	0.902	5.117	0.899	4.669
12	0.905	4.823	0.906	5.113	0.903	4.673

Table 19. Confidence Interval Data (Type I U)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
3	0.938	7.707	0.942	7.544	0.928	6.936
6	0.938	7.108	0.942	7.570	0.934	6.997
9	0.938	7.109	0.941	7.556	0.936	7.017
12	0.941	7.127	0.945	7.575	0.940	7.035

Table 20. Confidence Interval Data (Type I L)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
3	0.932	5.974	0.932	6.300	0.938	5.915
6	0.937	5.930	0.937	6.249	0.938	5.877
9	0.928	5.812	0.931	6.131	0.931	5.752
12	0.930	5.817	0.932	6.135	0.931	5.757

Table 21. Confidence Interval Data (Mixture Bell)

Sample Size Five

$Nx10^{-3}$	s.s.d.		s.m.a.d.		range	
	c.l.	i.l.	c.l.	i.l.	c.l.	i.l.
17	0.964	8.057	0.969	8.198	0.963	8.251
18	0.963	8.067	0.968	8.209	0.963	8.258
19	0.962	8.037	0.967	8.117	0.961	8.231
20	0.963	8.024	0.967	8.164	0.962	8.220
			Sample Size Ten			
17	0.954	4.876	0.961	4.750	0.953	5.170
18	0.956	4.881	0.962	4.752	0.953	5.179
19	0.957	4.866	0.963	4.735	0.955	5.165
20	0.956	4.858	0.962	4.729	0.955	5.159

7.4 Conclusions

The data in Tables 15 thru 21 indicate that none of the confidence interval estimation methods are greatly influenced by departures from normality.

The lowest confidence levels are about 90% and occur with the J shaped distribution used in Table 18. For this distribution, the confidence levels for each method do not appear to be significantly different. Here the shortest mean interval length is given by the range method being about 9% shorter than that of the sample mean absolute deviation and about 3% shorter than that of the sample standard deviation.

The only distribution with mean confidence levels greater than 95% is the mixture given in Table 21. For this distribution, the mean confidence level is about 96% for each method regardless of sample size. For samples of size ten, the sample mean absolute deviation gives the shortest interval length being about 8% shorter than that of the range and about 3% shorter than that of the sample standard deviation. For sample size five, the sample standard deviation gives the shortest mean interval length being about 2% shorter than that of the range and sample mean absolute deviation.

These results are by no means conclusive and the need for further study is indicated.

VIII. ACKNOWLEDGEMENTS

The author wishes to express his appreciation to the following:

Dr. Richard G. Krutchkoff for his time and care spent in providing assistance, suggestions and criticisms of this thesis.

Mr. Whitney L. Johnson for his reading and corrections of the manuscript.

Dr. Boyd Harshbarger for his encouragement and general interest in the work.

Mr. Philip N. Bergstresser for his time and invaluable assistance in computer technology.

IX. BIBLIOGRAPHY

Brown, G. W. (1951). History of RAND'S random digits-summary. Monte Carlo Method, Nat. Bur. Stand., Appl. Math. Series 12, Washington, D. C.: Government Printing Office, pp. 31-32.

Cooper, J. D., Davis, S. A. and Dono, N. R. (1963). Pearson universal random distribution generator (PURGE). IBM SHARE Programs Distributions. New York. G2 IBM 003.

Coveyou, R. R. (1960). Serial correlation in the generation of pseudo-random numbers. Journal of the Association for Computing Machinery, Vol. 7, pp. 72-74.

Elderton, W. P. (1953). Frequency Curves and Correlation. London: Cambridge University Press.

Gorenstein, S. (1966). Random Number Generation for the GPSS. IBM ASDD Technical Report 17-161. New York: International Business Machines Corporation.

Hammersley, J. M. and Handscomb, D. C. (1965). Monte Carlo Methods. New York: John Wiley and Sons.

Hull, T. E. and Dobell, A. R. (1962). Random number generators. SIAM Review, Vol. 4, No. 3, pp. 230-254.

Hull, T. E. and Dobell, A. R. (1962). Mixed congruential random number generators for binary machines. Journal of the Association for Computing Machinery, Vol. 11, pp. 31-40.

Hunter, D. G. N. (1960). Note on a test for repeating cycles in a pseudo-random number generator. Computer Journal, Vol. 3, p. 9.

International Business Machines Corporation. (1959). Random Number Generation and Testing, Reference manual C20-8011. New York.

Jackson, J. E. and Ross, E. L. (1955). Extended tables for use with the "G" test for means. Journal of the Americal Statistical Association. Vol. 50, pp. 416-433.

Kaercher, A. W. (1962). Rectangular random number generator. IBM SHARE Programs Distributions. New York. Film 31, No. 1359.

Kendall, M. G. and Stuart, A. (1961). The Advanced Theory of Statistics, Vol. 2, New York: Hafner.

Kendall, M. G. and Stuart, A. (1963). The Advanced Theory of Statistics, Vol. 1, New York: Hafner.

Krutchkoff, R. G. (1965). The correct use of the sample mean absolute deviation in confidence intervals for a normal variate. Submitted to Technometrics.

Lehmer, D. H. (1951). Mathematical methods in large-scale computing units. Annals of the Computation Laboratory of Harvard University, No. 26, Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery, pp. 141-146.

MacLaren, M. D. and Marsaglia, G. (1965). Uniform random number generators. Journal of the Association for Computing Machinery, Vol. 12, pp. 83-89.

Marsaglia, G. (1961). Expressing a random variable in terms of uniform random variables; Generating exponential random variables. Ann. Math. Statist., Vol. 32, pp. 894-900.

Marsaglia, MacLaren and Bray (1963). Random normal number generator. IBM SHARE Programs Distributions, New York. Film 33, No. 1461.

Marsaglia, G. (1964). Random variables and computers. Third Prague Conf. Inform. Theory, Stat. Decis. Funs., Random Processes, Publ. House Chech. Acad. Sci., Prague. pp. 499-512.

Metropolis, N. (1956). Phase shifts-middle squares-wave equation. Symposium on Monte Carlo Methods, ed. Meyer, H. A., New York: John Wiley and Sons, pp. 29-36.

Muller, Mervin E. (1959). A comparison of methods for generating normal deviates on digital computers. Journal of the Association for Computing Machinery, Vol. 6, pp. 376-383.

Noether, G. E. (1955). Use of the range instead of the standard deviation. Journal of the American Statistical Association, Vol. 50, pp. 1040-1055.

Patnaik, P. B. (1949). The non-central X^2 and F-distributions and their applications. Biometrika, Vol. 36, pp. 202-232.

Pearson, E. S. and Hartley, H. O. (1962). Biometrika Tables for Statisticians. London: Cambridge University Press.

Rotenberg, A. (1960). A new pseudo-random number generator. Journal of the Association for Computing Machinery, Vol. 7, pp. 75-77.

Scarborough, J. B. (1958). Numerical Mathematical Analysis. Baltimore: Johns Hopkins Press.

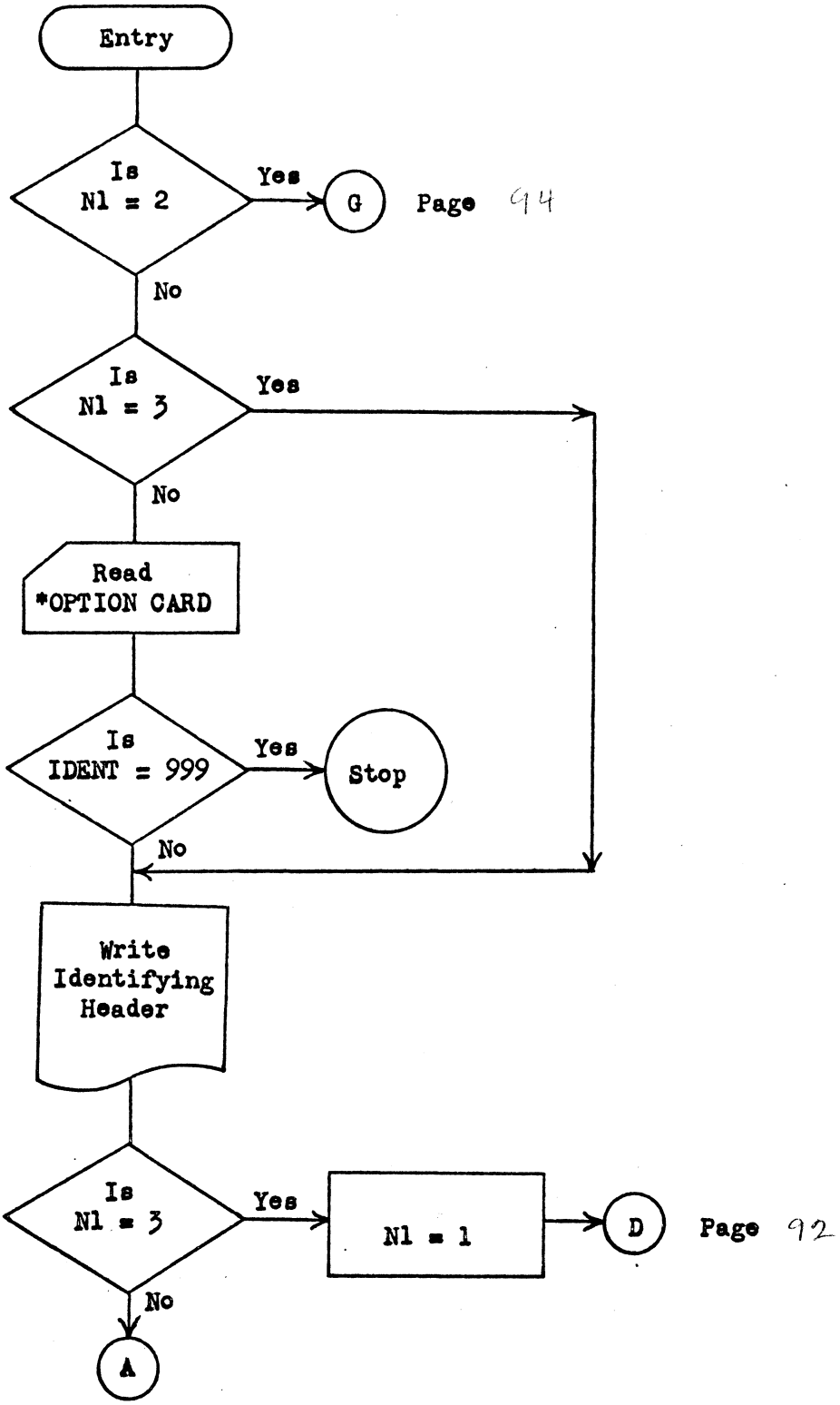
Taussky, O. and Todd, J. (1956). Generation and testing of pseudo-random numbers. Symposium on Monte Carlo Methods, ed. Meyer, H. A., New York: John Wiley and Sons, pp. 15-28.

United States Department of Commerce (1964). Handbook of Mathematical Functions. Nat. Bur. Stand., Appl. Math. Series 55, Washington, D. C.: Government Printing Office, pp. 949-953.

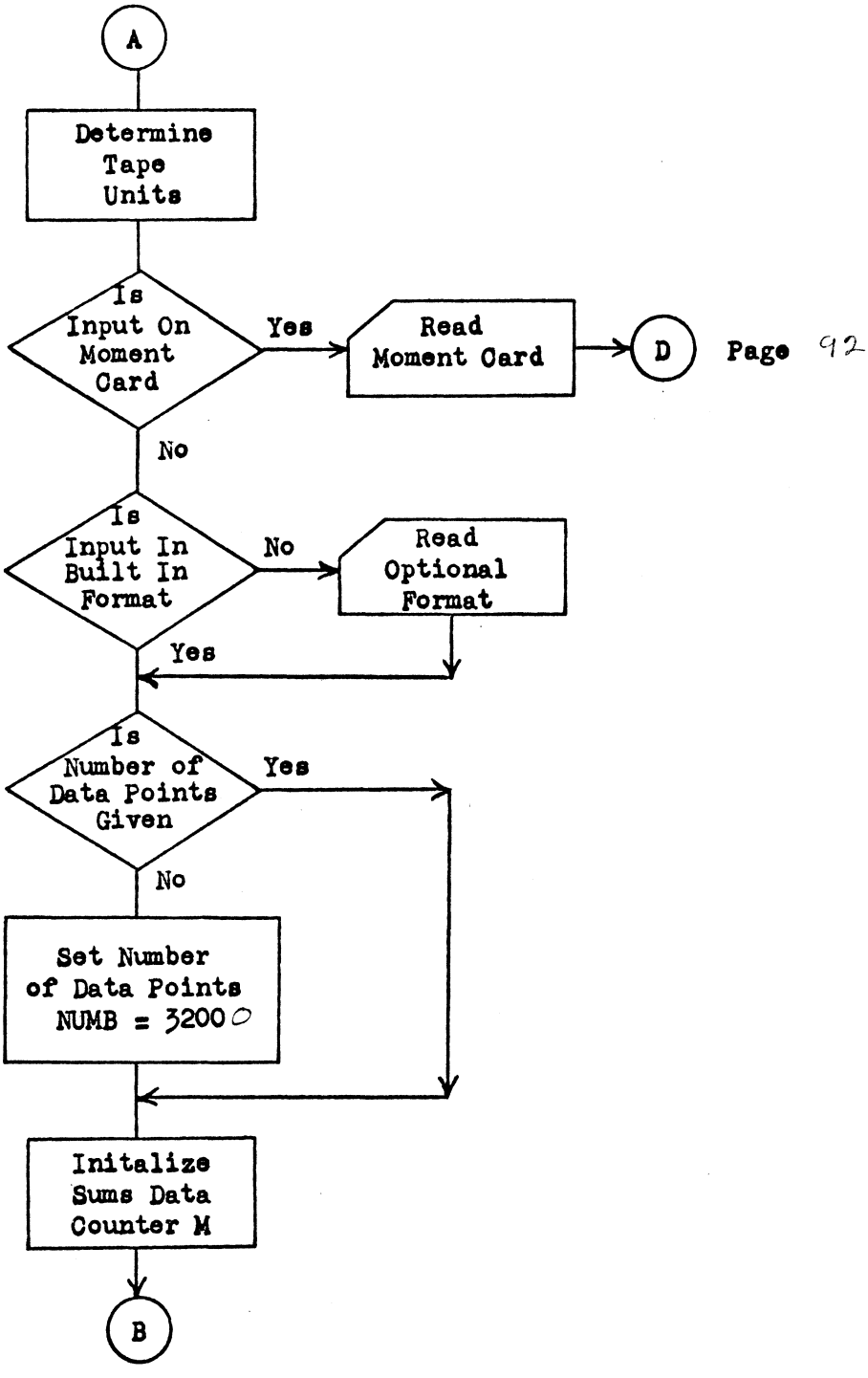
United States Department of Commerce (1965). Computer Literature Bibliography. Nat. Bur. Stand., Misc. Publ. 266, Washington, D. C.: Government Printing Office.

The vita has been removed from the scanned document

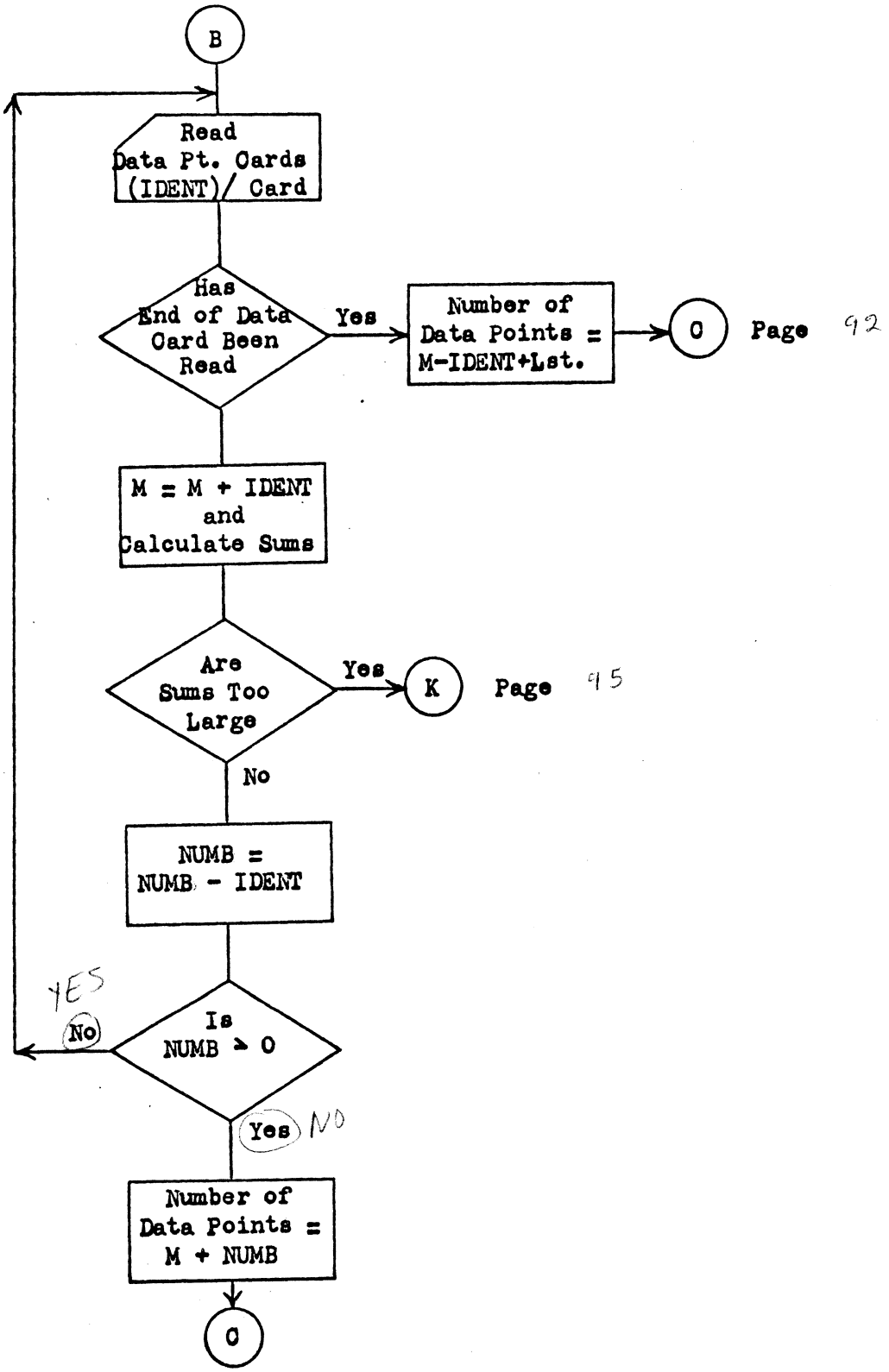
XI. APPENDIX



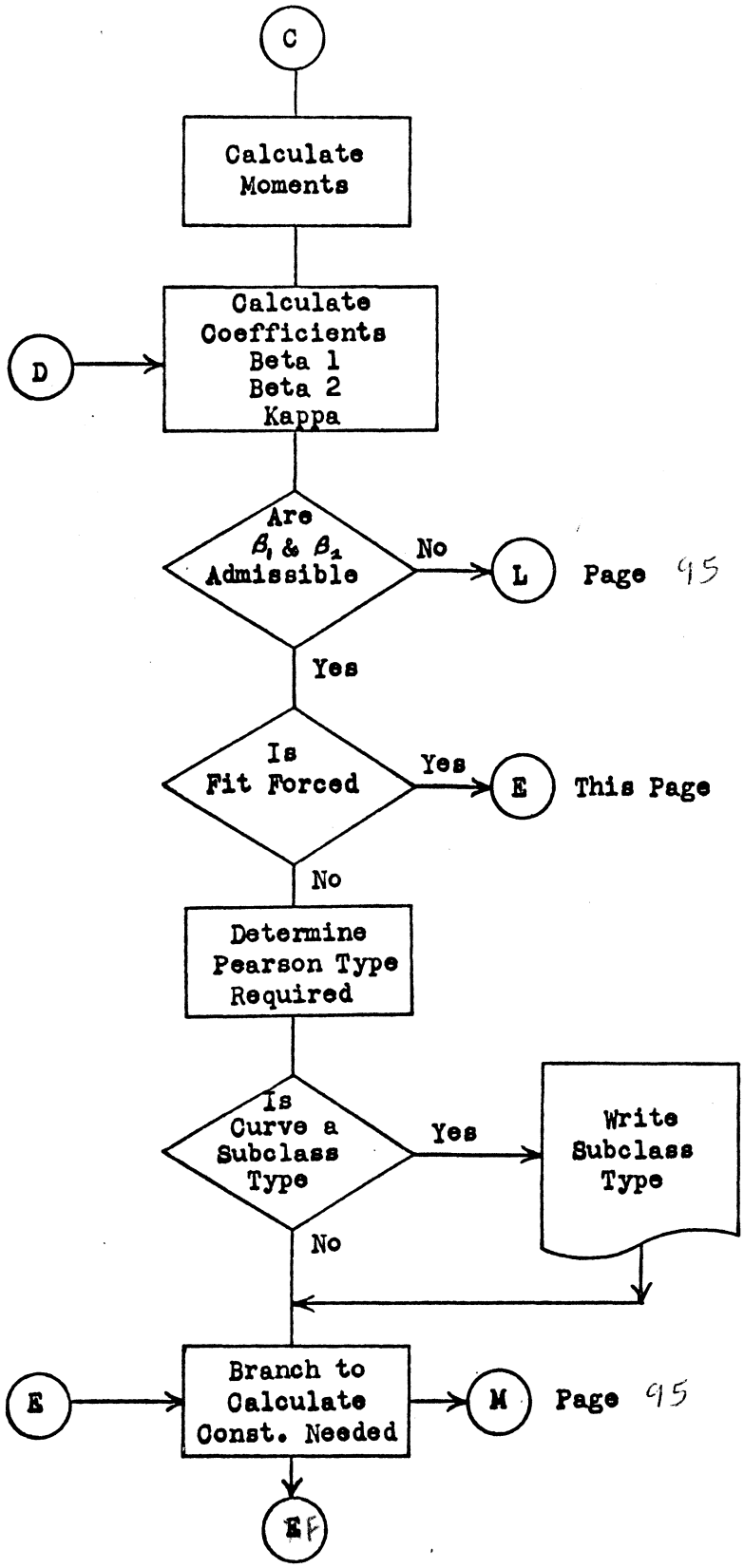
Flowchart For PURGE2



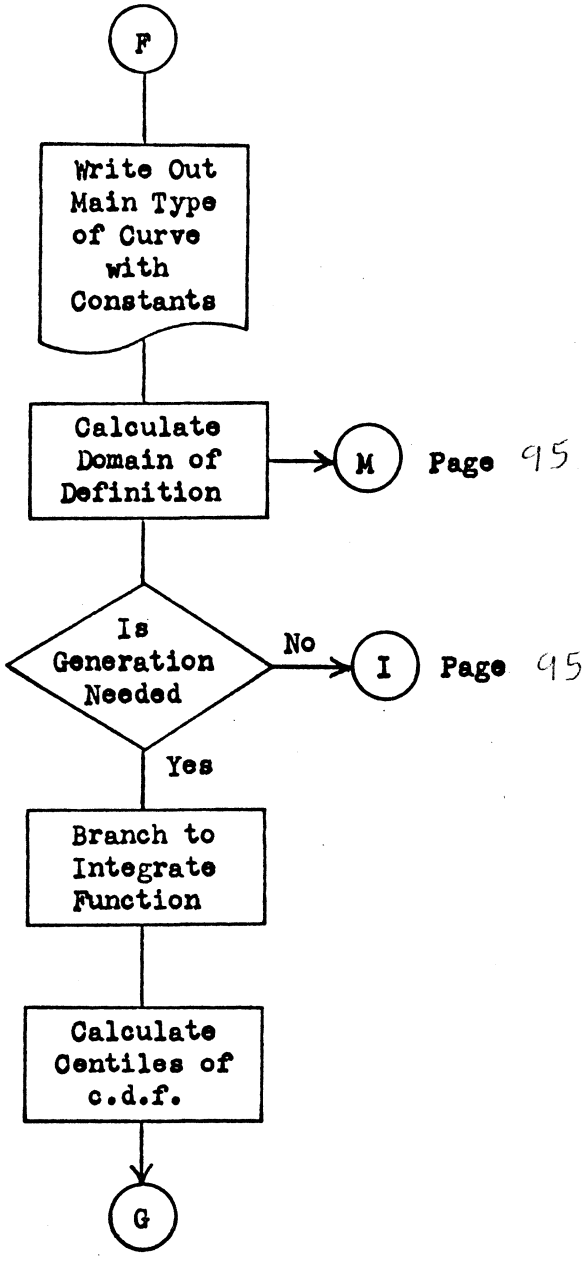
Flowchart For PURGE2, ctd.



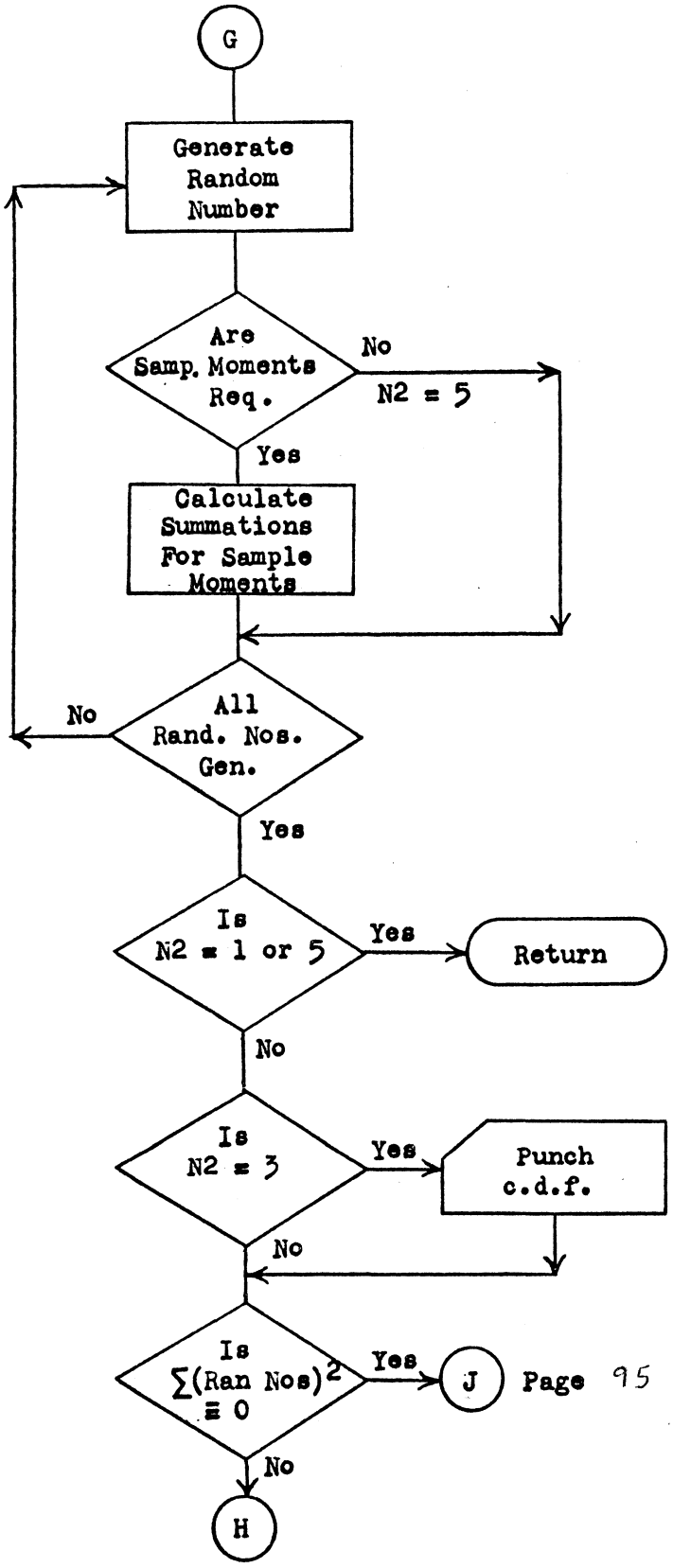
Flowchart For PURGE2, ctd.



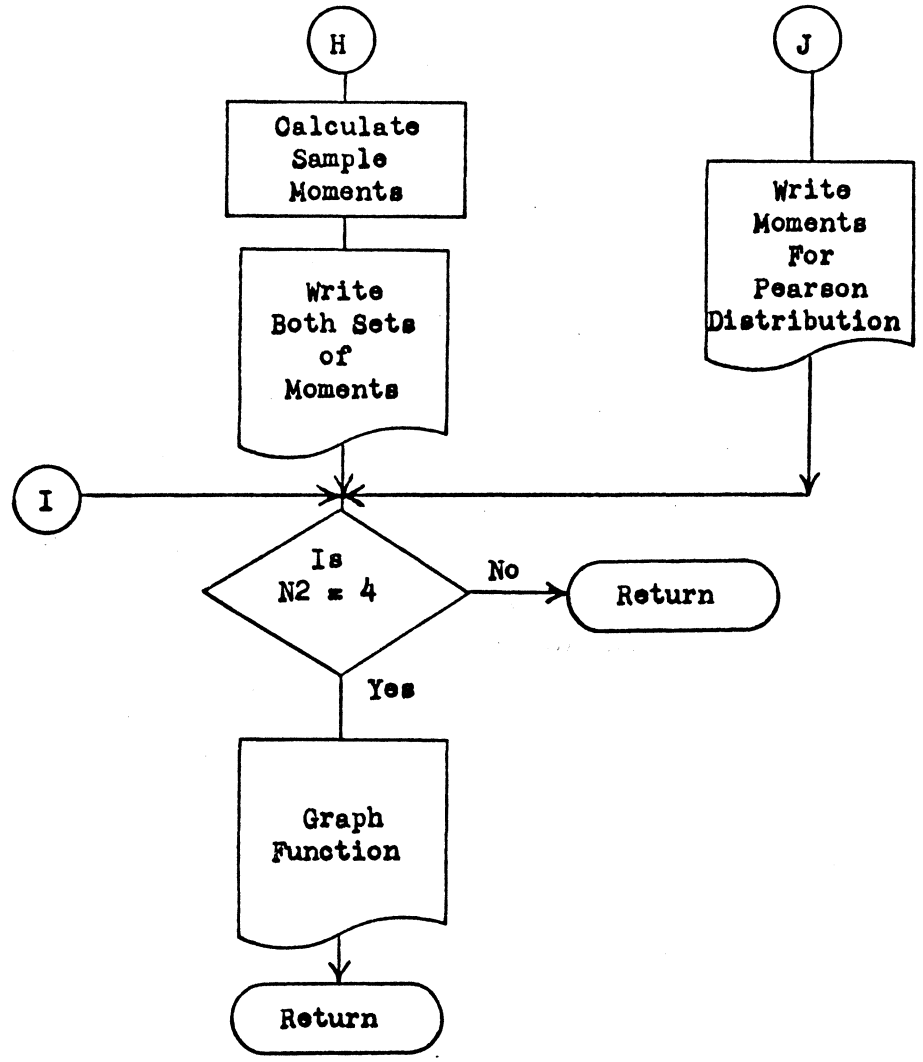
Flowchart For PURGE2, ctd.

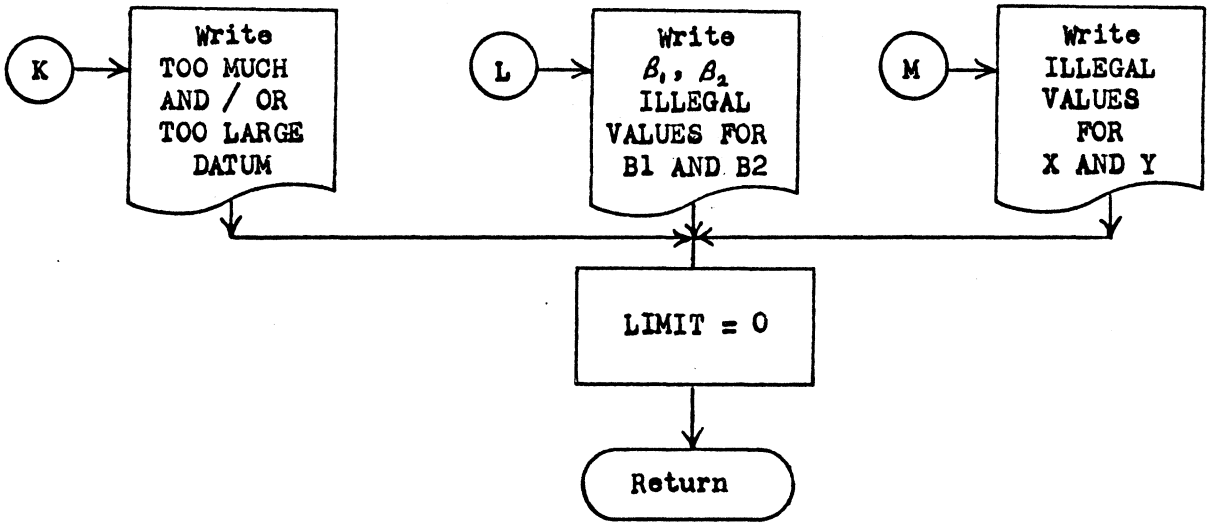


Flowchart For PURGE2, ctd.



Flowchart For PURGE2, ctd.





Flowchart For PURGE2, ctd.

```
C                                                                       PURGE001
      SUBROUTINE PURGE2(MKX,MKZ)                                        PURGE002
C      UNIVERSAL RANDOM DISTRIBUTION GENERATOR                          PURGE003
      N1=MKX                                                            PURGE004
      N2=MKZ                                                            PURGE005
C                                          FUNCTIONS DESCRIBING EACH EQPURGE006
      EXTERNAL NOLMUN,NOLMOV                                            PURGE007
      ATANF(X)=ATAN(X)                                                  PURGE008
      COSF(X)=COS(X)                                                    PURGE009
      EXPF(X)=EXP(X)                                                    PURGE010
      REAL LOGF,LGAMF,NORMAL                                            PURGE011
      LOGF(X)=ALOG(X)                                                   PURGE012
      LGAMF(X)=ALGAMA(X)                                                PURGE013
      SQRTF(X)=SQRT(X)                                                  PURGE014
      ABSF(X)=ABS(X)                                                    PURGE015
      FLOATF(I)=FLOAT(I)                                                PURGE016
      SIGNF(X,Y)=SIGN(X,Y)                                              PURGE017
      PEARIF (Y0,B1,B2,D1,D2,X)=Y0*(1.+X/B1)**D1*(1.-X/B2)**D2          PURGE018
C                                                                       PURGE019
      PEARIV (Y0,B,V,R,D,X)=EXPF(Y0-D*LOGF(1.+(X/B-V/R)**2)-V*ATANF(X/B-PURGE020
     2V/R))                                                             PURGE021
      NORMAL (Y0,C,X)=Y0*EXPF(-X**2/C)                                  PURGE022
      PEARVI (Y0,B1,B2,D1,D2,X)=EXPF(Y0+D2*LOGF(1.+X/B2)-D1*LOGF(1.+X/B1PURGE023
     2))                                                                PURGE024
       PERIII (Y0,G,P,A,X)=Y0*EXPF(P*LOGF(1.+X/A)-G*X)                  PURGE025
       DIMENSION SKUNK(5),                    FM(12),FRM(12)           PURGE026
      DIMENSION S(2641),   KU(101)                                      PURGE027
      COMMON/Z1/AVE/Z2/TOM(100)/Z3/LIMIT/Z4/FA(4)/Z5/FAN(4)/Z6/N/Z7/KU  PURGE028
      COMMON/Z8/S/Z9/CM2,CM3,CM4,BETA1,BETA2,SKAPPA                     PURGE029
      COMMON/Z10/      FCM2,FCM3,FCM4,FBETA1,FBETA2,FKAPPA              PURGE030
       DIMENSION CN(8),TN(8)                                            PURGE031
       DIMENSION A(4),AN(4),X(30)                                       PURGE032
      DIMENSION O(100   ),VAL(100),NA(100)                              PURGE033
      DIMENSION  BOT(100),SID(5)                                        PURGE034
      EQUIVALENCE (S,O),(S(101),VAL),(S(201),NA),(S(301),BOT),(S(401),SIPURGE035
     1D),(S(406),BOX),(S(407),PIP),(S(408),TMP),(X,KU)                  PURGE036
      EQUIVALENCE (A,KU(61)),(AN,KU(69)),(CN,KU(77)),(TN,KU(85)),(FRM,  PURGE037
```

```
     1KU(93))                                                                PURGE038
      DOUBLE PRECISION A,AN,X,Z,W                                            PURGE039
      IF(N1.EQ.2) GO TO 696                                                  PURGE040
  695 LT=0                                                                   PURGE041
      LIMIT=100                                                              PURGE042
C                                                                            PURGE043
C                                                                            PURGE044
      NT1=0                                                                  PURGE045
C                        CHANGE NEXT 2 CARDS FOR SPECIAL TAPE CONFIGURATI    PURGE046
       NINTP=5                                                               PURGE047
       NOUTP=6                                                               PURGE048
C                                                                            PURGE049
C                                                                            PURGE050
C                                                                            PURGE051
       L2=NINTP                                                              PURGE052
       L3=NOUTP                                                              PURGE053
 9831 CALL OVERFL(J)                                                         PURGE054
      IF(N1 .EQ. 3) GO TO 604                                                PURGE055
C                                                                            PURGE056
C                                                READ *CONTROL CARD          PURGE057
 9030  READ           (L2,1)IDENT,NUMB,LIMIT,NORM,ITT,NOUT                   PURGE058
 1     FORMAT (3X,I3,3X,I6,3X,I6,3X,I3,3X,I2,3X,I2)                           PURGE059
      IF(IDENT - 999) 604,605,604                                            PURGE060
604   L2=NINTP                                                               PURGE061
      L3=NOUTP                                                               PURGE062
C                                                                            PURGE063
       WRITE            (L3,9022)                                           PURGE064
 9022  FORMAT (1H1,  92X,22HDISTRIBUTION GENERATOR)                           PURGE065
       DO 600 I=1,2641                                                       PURGE066
600    S(I)=0.                                                               PURGE067
      IF(N1.NE.3) GO TO 2080                                                 PURGE068
      N1=1                                                                   PURGE069
      GO TO 2081                                                             PURGE070
C                                              DETERMINE OPTIONS             PURGE071
C                                              TAPES                         PURGE072
2080   IF (ITT) 2,2,3                                                        PURGE073
 2     ITT=L2                                                                PURGE074
```

```
 3      IF (NOUT) 4,4,5                                                  PURGE075
 4      NOUT=L3                                                          PURGE076
C                                                  TYPE OF INPUT        PURGE077
 5      IF(IDENT) 40,8,6                                                 PURGE078
C                                                  DATA POINT           PURGE079
 6      READ           (L2,7)(FM(I),I=1,12)                              PURGE080
 7      FORMAT (12A6)                                                    PURGE081
        GO TO 10                                                        PURGE082
C                                                  BUILT IN FORMAT      PURGE083
 8     CALL ATHRUZ(FM(1),6H(4X,4F)                                       PURGE084
       CALL ATHRUZ(FM(2),6H17.8,6)                                       PURGE085
       CALL ATHRUZ(FM(3),5HX,I2))                                        PURGE086
        IDENT=4                                                         PURGE087
C                                                  NUMBER OF DP CARDS GIVEN PURGE088
C                                                  OR CALCULATED        PURGE089
 10     IF (NUMB) 11,11,12                                               PURGE090
 11     NUMB=32000                                                       PURGE091
 12     M=0                                                              PURGE092
C                                                  INITIALIZE SUMS      PURGE093
        AN(1)=0.                                                        PURGE094
        AN(2)=0.                                                        PURGE095
        AN(3)=0.                                                        PURGE096
        AN(4)=0.                                                        PURGE097
C                                                  READ DATA POINT CARDS PURGE098
 13     READ           (ITT,FM)(X(I),I=1,IDENT),LST                      PURGE099
        IF (LST) 14,14,20                                               PURGE100
 14     M=M+IDENT                                                        PURGE101
C                                                  CALCULATE SUMS       PURGE102
 15     DO 17 I=1,IDENT                                                  PURGE103
        AN(1)=AN(1)+X(I)                                                PURGE104
       Z=X(I)*X(I)                                                      PURGE105
        AN(2)=AN(2)+Z                                                   PURGE106
       Z=Z*X(I)                                                         PURGE107
        AN(3)=AN(3)+Z                                                   PURGE108
   17   AN(4)=AN(4)+X(I)*Z                                               PURGE109
       CALL OVERFL(J3K)                                                 PURGE110
       GO TO (780,16,16),J3K                                            PURGE111
```

```
  780 LIMIT=0                                                                  PURGE112
 16    NUMB=NUMB-IDENT                                                       PURGE113
C                                                  DETERMINE OF DP CARDS     PURGE114
       IF (NUMB) 25,25,13                                                    PURGE115
 20    M=M-IDENT+LST                                                         PURGE116
       GO TO 27                                                              PURGE117
 25    M=M+NUMB                                                              PURGE118
C                                                  HOW MANY DP               PURGE119
 27    FN=M                                                                  PURGE120
      IF(LIMIT.EQ.0) GO TO 910                                               PURGE121
C                                                  FORM MOMENTS              PURGE122
       DO  30  K=1,4                                                         PURGE123
   30  A(K)=AN(K)/FN                                                         PURGE124
      Z=A(1)*A(1)                                                            PURGE125
       CM2=A(2)-Z                                                            PURGE126
       W=A(1)*A(2)                                                           PURGE127
       Z=Z*A(1)                                                              PURGE128
       CM3=A(3)-3.*W          +2.*Z                                          PURGE129
       CM4=A(4)-4.*A(1)*A(3)+6.*A(1)*W        -3.*A(1)*Z                     PURGE130
      AVE=A(1)                                                               PURGE131
      RCM2=SQRT(CM2)                                                         PURGE132
C                                                                            PURGE133
C                                                  CALCULATE COEFF FOR PEARSON PURGE134
C                                                                            PURGE135
  35  TP=CM2*CM2                                                             PURGE136
      BETA1=CM3**2/(CM2*TP)                                                  PURGE137
      BETA2=CM4/TP                                                           PURGE138
C     DETERMINE IF BETA1 AND BETA2 ADMISSIBLE                                PURGE139
      IF(((BETA1-BETA2+1.)/(-1.414213562).LT.0.).OR.(BETA1.LT.0.).OR.        PURGE140
     1((15.*BETA1-8.*BETA2+36.)/(-17.).GE.0.)) GO TO 750                     PURGE141
       DOP=0.                                                                PURGE142
      SK1=BETA1*(BETA2+3.)**2                                                PURGE143
      SK2=4.*(4.*BETA2-3.*BETA1)*(2.*BETA2-3.*BETA1-6.)                      PURGE144
      IF (SK1) 351,621,351                                                   PURGE145
 351  IF (SK2) 352,353,352                                                   PURGE146
 352  SKAPPA=SK1/SK2                                                         PURGE147
      GO TO 45                                                               PURGE148
```

```
  621 SKAPPA=0.                                                          PURGE149
      GO TO 45                                                           PURGE150
  353 SKAPPA=(10.**10)**10                                               PURGE151
       IF (BETA1-4.) 8008,8007,8008                                      PURGE152
 8007  WRITE            (L3,8006)                                       PURGE153
8006  FORMAT(1H ,35X,17HSUBCLASS TYPE TEN)                               PURGE154
       DOP=1.                                                            PURGE155
       GO TO 45                                                          PURGE156
 8008  WRITE            (L3,8009)                                       PURGE157
 8009 FORMAT(1H ,35X,19HSUBCLASS TYPE THREE)                            PURGE158
       DOP=1.                                                            PURGE159
      GO TO 45                                                           PURGE160
C                                          READ MOMENTS CARD IF NECCESARPURGE161
  40    READ           (ITT,9023) AVE,CM2,CM3,CM4                      PURGE162
  9023  FORMAT (4X,4F17.8)                                             PURGE163
2081    RCM2=SQRT(CM2)                                                 PURGE164
      GO TO 35                                                         PURGE165
C                                          DETERMINE MAIN TYPE         PURGE166
C                                               OR FORCE FIT           PURGE167
 45     IF (NORM-1) 451,50,452                                         PURGE168
 452    IF (NORM-6) 71,111,90                                          PURGE169
 451    IF (DOP) 453,453,9400                                          PURGE170
 453    IF (SKAPPA)50,8001,8002                                        PURGE171
 8001   IF (BETA2-3.) 8004,90,8003                                     PURGE172
 8002   IF (SKAPPA-1.) 71,8005,111                                     PURGE173
 8004   WRITE            (L3,8010)                                     PURGE174
 8010 FORMAT(1H ,35X,17HSUBCLASS TYPE TWO)                             PURGE175
       GO TO 50                                                         PURGE176
 8003   WRITE            (L3,8011)                                     PURGE177
 8011 FORMAT(1H ,35X,19HSUBCLASS TYPE SEVEN)                           PURGE178
       GO TO 71                                                         PURGE179
 8005   WRITE            (L3,8012)                                     PURGE180
 8012 FORMAT(1H ,35X,18HSUBCLASS TYPE FIVE)                            PURGE181
       GO TO 111                                                        PURGE182
C                                                                       PURGE183
C                                          TYPE ONE CONSTANTS           PURGE184
C                                                                       PURGE185
```

```
 50   WRITE            (L3,51)                                                  PURGE186
 51   FORMAT(1H ,30X,22HPEARSON CURVE TYPE ONE)                                 PURGE187
511   R=6.* (BETA2-BETA1-1.)/(6.+3.*BETA1-2.*BETA2)                             PURGE188
      TP=          R*(R+2.)*SQRTF(BETA1/(BETA1*(R+2.)**2+16.*(R+1.)))           PURGE189
      RP=.5*(R-2.+TP)                                                           PURGE190
      RM=.5*(R-2.-TP)                                                           PURGE191
53    IF (CM3) 57,56,56                                                         PURGE192
 56   D2=RP                                                                     PURGE193
      D1=RM                                                                     PURGE194
      GO TO 58                                                                  PURGE195
 57   D1=RP                                                                     PURGE196
      D2=RM                                                                     PURGE197
 58   SCOTT=.5*RCM2       *SQRTF(BETA1*(R+2.)**2+16.*(R+1.))                    PURGE198
      B2=SCOTT*(D2+1.)/(D2+D1+2.)                                               PURGE199
      B1=SCOTT-B2                                                               PURGE200
      IF((D1+1.)*(D2+1.)) 903,903,582                                           PURGE201
582   IF (B1+B2) 583,906,583                                                    PURGE202
583   Y0=SIGNF(1.0,B1+B2)*EXPF(D1*LOGF(D1+1.)-LOGF(ABSF(B1+B2))-LGAMF(D1PURGE203
     2+1.)+D2*LOGF(D2+1.)-LGAMF(D2+1.)+LGAMF(D1+D2+2.)-(D1+D2)*LOGF(D1+DPURGE204
     32+2.))                                                                    PURGE205
585   WRITE            (L3,581)D1,D2,B1,B2,Y0                                   PURGE206
581   FORMAT (4H0M1=,1PE12.5,4H M2=,1PE12.5,4H A1=,1PE12.5,4H A2=,1PE12.PURGE207
     25,4H YE=,1PE12.5)                                                         PURGE208
      V1=B1                                                                     PURGE209
      V2=B2                                                                     PURGE210
      V3=D1                                                                     PURGE211
      V4=D2                                                                     PURGE212
C                                          JAB IS FOR INTEGRATION BRANCHPURGE213
      JAB=1                                                                     PURGE214
      IF(B2    ) 60,906,63                                                      PURGE215
 60   IF (B1) 61,61,906                                                         PURGE216
C                                          RANGES                             PURGE217
 61   H=(-B1-B2)/5120.                                                          PURGE218
      UPPER=-B1                                                                 PURGE219
      EL=B2                                                                     PURGE220
      GO TO 145                                                                 PURGE221
 63   IF (B1) 906,64,64                                                         PURGE222
```

```
 64   H=(B2+B1)/5120.                                                          PURGE223
      UPPER=B2                                                                 PURGE224
      EL=-B1                                                                   PURGE225
      GO TO 145                                                                PURGE226
C                                                                              PURGE227
C                                                TYPE FOUR CONSTANTS           PURGE228
C                                                                              PURGE229
  71  WRITE           (L3,72)                                                  PURGE230
  72  FORMAT(1H ,30X,23HPEARSON CURVE TYPE FOUR)                               PURGE231
  73  R=6.*(BETA2-BETA1-1.)/(2.*BETA2-3.*BETA1-6.)                             PURGE232
  74  D=.5*(R+2.)                                                              PURGE233
  75  V=R*(R-2.)*SQRTF(BETA1)/SQRTF(16.*(R-1.)-BETA1*(R-2.)**2)                PURGE234
 76   B=SQRTF(CM2/16.)*SQRTF(16.*(R-1.)-BETA1*(R-2.)**2)                       PURGE235
       IF (CM3)77,783,783                                                      PURGE236
 783   V=-V                                                                    PURGE237
  77  PHI=ATANF(V/R)                                                           PURGE238
  78  TOP=    ((COSF(PHI))**2/(3.*R)-1./(12.*R)-PHI*V)                         PURGE239
       Y0=LOGF(0.3989423)-LOGF(B)+.5*LOGF(R)+TOP-(R+1.)*LOGF(COSF(PHI))        PURGE240
       IF (Y0-88.9) 782,782,784                                                PURGE241
  782  Y1=EXPF(Y0)                                                             PURGE242
       GO TO 785                                                               PURGE243
  784  Y1=.99999999E 38                                                        PURGE244
  785 WRITE           (L3,781)R,D,V,B,Y1                                       PURGE245
 781  FORMAT (3HOR=,1PE12.5,4H  M=,1PE12.5,4H  V=,1PE12.5,3H A=,1PE12.5,PURGE246
     24H Y0=,1PE12.5)                                                         PURGE247
      JAB=2                                                                    PURGE248
      H=RCM2      /512.                                                        PURGE249
      UPPER=5.* RCM2                                                           PURGE250
      EL=-UPPER                                                                PURGE251
      V1=B                                                                     PURGE252
      V2=V                                                                     PURGE253
      V3=R                                                                     PURGE254
      V4=D                                                                     PURGE255
      GO TO 145                                                                PURGE256
C                                                                              PURGE257
C                                              NORMAL CURVE CONSTANTS          PURGE258
C                                                                              PURGE259
```

```
 90   WRITE            (L3,91)                                                  PURGE260
 91   FORMAT(1H ,30X,29HTHIS IS A NORMAL DISTRIBUTION)                          PURGE261
 92   C=CM2*2.                                                                  PURGE262
      Y0=1./SQRTF(6.2831853*CM2)                                                PURGE263
      WRITE            (L3,1101)C,Y0                                            PURGE264
 1101 FORMAT (3H0C=,1PE12.5,4H Y0=,1PE12.5)                                     PURGE265
      JAB=3                                                                     PURGE266
      H=RCM2       /512.                                                        PURGE267
      UPPER=5.*RCM2                                                             PURGE268
      EL=-UPPER                                                                 PURGE269
      V1=C                                                                      PURGE270
      GO TO 145                                                                 PURGE271
C                                                                               PURGE272
C                                          TYPE SIX CONSTANTS                   PURGE273
C                                                                               PURGE274
 111  WRITE            (L3,112)                                                 PURGE275
 112  FORMAT(1H ,30X,22HPEARSON CURVE TYPE SIX)                                 PURGE276
 113  R=6.*(BETA2-BETA1-1.)/(6.+3.*BETA1-2.*BETA2)                              PURGE277
      CAT=.5*RCM2       *SQRTF(BETA1*(R+2.)**2+16.*(R+1.))                      PURGE278
       IF (CM3) 9836,9837,9837                                                  PURGE279
 9836  CAT=-CAT                                                                PURGE280
 9837 TERM=R*(R+2.)/2.*SQRTF(BETA1/(BETA1*(R+2.)**2+16.*(R+1.)))                PURGE281
      D2=(R-2.)/2.+TERM                                                         PURGE282
      D1=-((R-2.)/2.-TERM)                                                      PURGE283
      TP=              ((D1-1.)-(D2+1.))                                        PURGE284
      B1=CAT*(D1-1.)/TP                                                         PURGE285
      B2=CAT*(D2+1.)/TP                                                         PURGE286
      CAC1=1.                                                                   PURGE287
      Y0=(D2*LOGF(D2+1.)-LGAMF(D2+1.)-LOGF(CAT)+(D1-D2)*LOGF(D1-D2-2.)-DPURGE288
     21*LOGF(D1-1.)+LGAMF(D1)-LGAMF(D1-D2-1.))+LOGF(CAC1)                       PURGE289
      Y1=EXPF(Y0)                                                               PURGE290
      WRITE            (L3,1134)D2,D1,B1,B2,Y1                                  PURGE291
 1134 FORMAT (4H0Q2=,1PE12.5,4H Q1=,1PE12.5,4H A1=,1PE12.5,4H A2=,1PE12.PURGE292
     25,4H YE=,1PE12.5)                                                         PURGE293
      JAB=4                                                                     PURGE294
      V1=B1                                                                     PURGE295
      V2=B2                                                                     PURGE296
```

```
      V3=D1                                                              PURGE297
      V4=D2                                                              PURGE298
      IF (B2    ) 115,906,120                                            PURGE299
  115 H=(-B2+5.*RCM2       )/5120.                                       PURGE300
       UPPER=-B2                                                         PURGE301
      EL=-5.*RCM2                                                        PURGE302
      GO TO 145                                                          PURGE303
  120 UPPER=5.*RCM2                                                      PURGE304
      H=(B2+UPPER         )/5120.                                        PURGE305
      EL=-B2                                                             PURGE306
       GO TO 145                                                         PURGE307
C                                                                        PURGE308
C                                          TYPE THREE CONSTANTS          PURGE309
C                                                                        PURGE310
 9400 G=2.*CM2/CM3                                                       PURGE311
      P=4./BETA1-1.                                                      PURGE312
      A1=(P+1.)/G                                                        PURGE313
      Y0=G*(P+1.)**P/EXPF(P+1.+LGAMF(P+1.))                              PURGE314
      Y0=ABSF(Y0)                                                        PURGE315
      WRITE             (L3,9401)G,P,A1,Y0                               PURGE316
 9401 FORMAT (3HOG=,1PE12.5,3H P=,1PE12.5,3H A=,1PE12.5,4H YE=,1PE12.5)PURGE317
      JAB=5                                                              PURGE318
      V1=G                                                               PURGE319
      V2=P                                                               PURGE320
      V3=A1                                                              PURGE321
      IF (A1) 9402,9403,9403                                             PURGE322
 9402 H=(-A1+5.*RCM2       )/5120.                                       PURGE323
      UPPER=-A1                                                          PURGE324
      EL=-5.*RCM2                                                        PURGE325
      GO TO 145                                                          PURGE326
 9403 UPPER=5.*RCM2                                                      PURGE327
      H=(A1+UPPER         )/5120.                                        PURGE328
      EL = -A1                                                           PURGE329
C                                          INITIALIZE OUTPUT SUMS        PURGE330
C                                          GENERATION                    PURGE331
C                                          BRANCH TO INTEGRATE           PURGE332
 950  FORMAT(1H )                                                        PURGE333
```

```
 145    V0=Y0                                                          PURGE334
        V6=EL                                                          PURGE335
        V7=UPPER                                                       PURGE336
 951    DO 146 KK=1,4                                                  PURGE337
  146   FAN(KK)=0.                                                     PURGE338
         IF (LIMIT) 148,148,147                                        PURGE339
 147    S(1)=0.                                                        PURGE340
  153    GO TO (200,250,300,350,460),JAB                               PURGE341
C                                                                       PURGE342
C                                                TYPE ONE INTEG         PURGE343
C                                                                       PURGE344
   200 IF (D1) 2054,2055,2055                                           PURGE345
  2055 IF (D2) 2056,2057,2057                                           PURGE346
  2057 EXVAL=EL                                                         PURGE347
       ADD1=EL+H                                                        PURGE348
       TP= H+H                                                          PURGE349
       XVAL=EL+TP                                                       PURGE350
       I=2                                                              PURGE351
  205  SKUNK(1)=PEARIF(Y0,B1,B2,D1,D2,EXVAL)                            PURGE352
  2051 SKUNK(2)=PEARIF(Y0,B1,B2,D1,D2,ADD1)                             PURGE353
  2052 SKUNK(3)=PEARIF(Y0,B1,B2,D1,D2,XVAL)                             PURGE354
  2053 S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                        PURGE355
       IF (XVAL+TP-UPPER)210,210,385                                    PURGE356
  210  EXVAL=XVAL                                                       PURGE357
       ADD1=XVAL+H                                                      PURGE358
       XVAL=XVAL+TP                                                     PURGE359
       I=I+1                                                            PURGE360
       GO TO 205                                                        PURGE361
C                                                                       PURGE362
C                                                TYPE ONE L SHAPE       PURGE363
  2054 IF (D2) 7001,3001,3001                                           PURGE364
  3001 H=(UPPER-EL)/5121.                                               PURGE365
       EL=EL+H                                                          PURGE366
       EXVAL=EL                                                         PURGE367
       ADD1=EL+H                                                        PURGE368
       TP= H+H                                                          PURGE369
       XVAL=EL+TP                                                       PURGE370
```

```
      S(2)=0.                                                          PURGE371
      I=3                                                              PURGE372
 3002 SKUNK(1)=PEARIF(Y0,B1,B2,D1,D2,EXVAL)                            PURGE373
      SKUNK(2)=PEARIF(Y0,B1,B2,D1,D2,ADD1)                             PURGE374
      SKUNK(3)=PEARIF(Y0,B1,B2,D1,D2,XVAL)                             PURGE375
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                        PURGE376
      IF (XVAL+TP-UPPER)3003,3003,3004                                 PURGE377
 3003 EXVAL=XVAL                                                       PURGE378
      ADD1=XVAL+H                                                      PURGE379
      XVAL=XVAL+TP                                                     PURGE380
      I=I+1                                                            PURGE381
      GO TO 3002                                                       PURGE382
 3004 S(2)=3./H-S(I)                                                   PURGE383
 4005 N=I                                                              PURGE384
      DO 4006 I=3,N                                                    PURGE385
 4006 S(I)=S(I)+S(2)                                                   PURGE386
      I=N                                                              PURGE387
      GO TO 385                                                        PURGE388
C                                                                      PURGE389
C                                        TYPE ONE U SHAPE              PURGE390
 7001 H=(UPPER-EL)/5122.                                               PURGE391
      EL=EL+H                                                          PURGE392
      UPPER=UPPER-H                                                    PURGE393
      EXVAL=EL                                                         PURGE394
      ADD1=EL+H                                                        PURGE395
      TP= H+H                                                          PURGE396
      XVAL=EL+TP                                                       PURGE397
       I=2                                                             PURGE398
 7002 SKUNK(1)=PEARIF(Y0,B1,B2,D1,D2,EXVAL)                            PURGE399
      SKUNK(2)=PEARIF(Y0,B1,B2,D1,D2,ADD1)                             PURGE400
      SKUNK(3)=PEARIF(Y0,B1,B2,D1,D2,XVAL)                             PURGE401
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                        PURGE402
       IF (XVAL+TP-UPPER)7003,7003,385                                 PURGE403
 7003 EXVAL=XVAL                                                       PURGE404
      ADD1=XVAL+H                                                      PURGE405
      XVAL=XVAL+TP                                                     PURGE406
      I=I+1                                                            PURGE407
```

```
      GO TO 7002                                                        PURGE408
C                                                                       PURGE409
C                                                                       PURGE410
C                                          TYPE ONE J SHAPE             PURGE411
 2056 H=(UPPER-EL)/5121.                                                PURGE412
      UPPER=UPPER-H                                                     PURGE413
      EXVAL=EL                                                          PURGE414
       ADD1=EL+H                                                        PURGE415
      TP= H+H                                                           PURGE416
      XVAL=EL+TP                                                        PURGE417
      I=2                                                               PURGE418
 4002 SKUNK(1)=PEARIF(Y0,B1,B2,D1,D2,EXVAL)                             PURGE419
      SKUNK(2)=PEARIF(Y0,B1,B2,D1,D2,ADD1)                              PURGE420
      SKUNK(3)=PEARIF(Y0,B1,B2,D1,D2,XVAL)                              PURGE421
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                         PURGE422
      IF (XVAL+TP-UPPER)4003,4003,385                                   PURGE423
 4003 EXVAL=XVAL                                                        PURGE424
      ADD1=XVAL+H                                                       PURGE425
      XVAL=XVAL+2.*H                                                    PURGE426
      I=I+1                                                             PURGE427
      GO TO 4002                                                        PURGE428
C                                                                       PURGE429
C                                          TYPE FOUR INTEG              PURGE430
  250 EXVAL=EL                                                          PURGE431
      ADD1=EL+H                                                         PURGE432
      TP= H+H                                                           PURGE433
      XVAL=EL+TP                                                        PURGE434
      I=2                                                               PURGE435
 255  SKUNK(1)=PEARIV (Y0,B,V,R,D,EXVAL)                                PURGE436
      SKUNK(2)=PEARIV (Y0,B,V,R,D,ADD1)                                 PURGE437
      SKUNK(3)=PEARIV (Y0,B,V,R,D,XVAL)                                 PURGE438
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                         PURGE439
      IF (XVAL-UPPER) 260,385,385                                       PURGE440
 260  EXVAL=XVAL                                                        PURGE441
      ADD1=XVAL+H                                                       PURGE442
      XVAL=XVAL+TP                                                      PURGE443
      I=I+1                                                             PURGE444
      GO TO 255
```

```
C                                                                       PURGE445
C                                          NORMAL INTEG                 PURGE446
  300 TP= H+H                                                           PURGE447
      EXVAL=EL-TP                                                       PURGE448
      ADD1=EL-H                                                         PURGE449
      XVAL=EL                                                           PURGE450
      I=2                                                               PURGE451
  305 SKUNK(1)=NORMAL (Y0,C,EXVAL)                                      PURGE452
      SKUNK(2)=NORMAL (Y0,C,ADD1)                                       PURGE453
      SKUNK(3)=NORMAL (Y0,C,XVAL)                                       PURGE454
      S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                         PURGE455
      IF (XVAL-UPPER) 310,385,385                                       PURGE456
  310 EXVAL=XVAL                                                        PURGE457
      ADD1=XVAL+H                                                       PURGE458
      XVAL=XVAL+TP                                                      PURGE459
      I=I+1                                                             PURGE460
      GO TO 305                                                         PURGE461
C                                                                       PURGE462
C                                          TYPE SIX INTRG               PURGE463
  350 EXVAL=EL                                                          PURGE464
      ADD1=EL+H                                                         PURGE465
      TP= H+H                                                           PURGE466
      XVAL=EL+TP                                                        PURGE467
      I=2                                                               PURGE468
 3551 SKUNK(1)=0                                                        PURGE469
      GO TO 3553                                                        PURGE470
 3552 SKUNK(1)=PEARVI (Y0,B1,B2,D1,D2,EXVAL)                            PURGE471
 3553 SKUNK(2)=PEARVI (Y0,B1,B2,D1,D2,ADD1)                             PURGE472
      SKUNK(3)=PEARVI (Y0,B1,B2,D1,D2,XVAL)                             PURGE473
 3559 S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                         PURGE474
      IF (XVAL-UPPER) 360,385,385                                       PURGE475
  360 EXVAL=XVAL                                                        PURGE476
      ADD1=XVAL+H                                                       PURGE477
      XVAL=XVAL+TP                                                      PURGE478
      I=I+1                                                             PURGE479
      GO TO 3552                                                        PURGE480
C                                                                       PURGE481
```

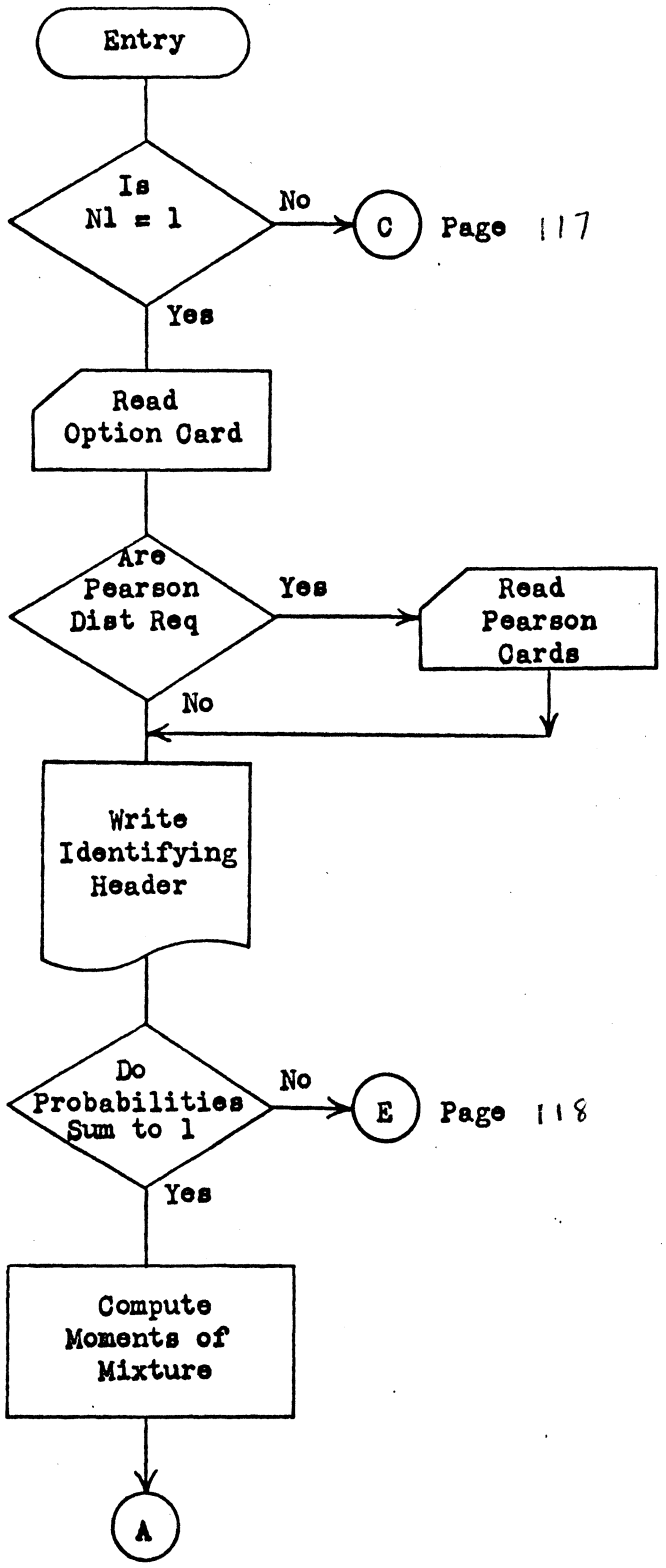
```
C                                                    TYPE THREE INTEG          PURGE482
  460  EXVAL=EL                                                                 PURGE483
       ADD1=EL+H                                                                PURGE484
      TP= H+H                                                                   PURGE485
       XVAL=EL+TP                                                               PURGE486
       I=2                                                                      PURGE487
       SKUNK(1)=0                                                               PURGE488
       GO TO 462                                                                PURGE489
 461   SKUNK(1)=PERIII (YO,G,P,A1,EXVAL)                                        PURGE490
 462   SKUNK(2)=PERIII (YO,G,P,A1,ADD1)                                         PURGE491
       SKUNK(3)=PERIII (YO,G,P,A1,XVAL)                                         PURGE492
       S(I)=S(I-1)+SKUNK(1)+4.*SKUNK(2)+SKUNK(3)                                PURGE493
       IF (XVAL-UPPER) 463,385,385                                              PURGE494
 463   EXVAL=XVAL                                                               PURGE495
       ADD1=XVAL+H                                                              PURGE496
       XVAL=XVAL+TP                                                             PURGE497
       I=I+1                                                                    PURGE498
       GO TO 461                                                                PURGE499
  385  A2B=H/3.                                                                 PURGE500
682    N=I+1                                                                    PURGE501
678    DO 3851 I=1,N                                                            PURGE502
 3851  S(I)=S(I)*A2B                                                            PURGE503
       S(N+1)=1.                                                                PURGE504
C                                                                               PURGE505
C                                                    RANDOM UNIT CALCULATION   PURGE506
 9301  KU(1)=1                                                                  PURGE507
C                                                                               PURGE508
       A1=0.                                                                    PURGE509
       I=1                                                                      PURGE510
       DO 675 J=2,100                                                           PURGE511
       A1=A1+.01                                                                PURGE512
       M=I                                                                      PURGE513
        DO 676 I=M,N                                                            PURGE514
       IF(S(I)-A1)676,675,675                                                   PURGE515
676    CONTINUE                                                                 PURGE516
  675  KU(J)=I                                                                  PURGE517
       KU(101)=N                                                                PURGE518
```

```
C                                                                       PURGE519
      CALL OVERFL(J3K)                                                  PURGE520
696   DO 393 K=1,LIMIT                                                  PURGE521
       RND=RDM(1.0)                                                     PURGE522
      NR=INT(100.*RND)+1                                                PURGE523
  671 M=KU(NR)                                                          PURGE524
      DO 387 I=M,N                                                      PURGE525
      IF(S(I)-RND) 387,390,388                                          PURGE526
387   CONTINUE                                                          PURGE527
388   BUT=FLOAT(I+I-2)-2.*(S(I)-RND)/(S(I)-S(I-1))                      PURGE528
C                                                                       PURGE529
       GO TO 391                                                        PURGE530
  390 BUT=I+I-2                                                         PURGE531
  391 TOM(K)=EL+BUT*H+AVE                                               PURGE532
      IF(N2.EQ.5) GO TO 393                                             PURGE533
683    DO 685 KK=1,4                                                    PURGE534
685   FAN(KK)=FAN(KK)+TOM(K)**KK                                        PURGE535
393    CONTINUE                                                         PURGE536
C                                                                       PURGE537
      LT=LT+LIMIT                                                       PURGE538
      GO TO (952,400,953,400,952),N2                                    PURGE539
952   RETURN                                                            PURGE540
953   WRITE(7)N,KU,KT,EL,H,AVE,CM2,CM3,CM4                              PURGE541
      WRITE(7) (S(I),I=1,N)                                             PURGE542
C                                                CALC OUTPUT MOMENTS    PURGE543
  400 IF(FAN(2).EQ.0.) GO TO 148                                        PURGE544
      TP=FLOAT(LT)                                                      PURGE545
      DO 405 KK=1,4                                                     PURGE546
  405 FA(KK)=FAN(KK)/TP                                                 PURGE547
      FCM2 =FA(2)-FA(1)**2                                              PURGE548
      FCM3 =FA(3)-3.*FA(1)*FA(2)+2.*FA(1)**3                            PURGE549
      FCM4 =FA(4)-4.*FA(1)*FA(3)+6.*FA(1)**2*FA(2)-3.*FA(1)**4          PURGE550
      FBETA1=FCM3**2/FCM2**3                                            PURGE551
      FBETA2=FCM4/FCM2**2                                               PURGE552
      FKAPPA=FBETA1*(FBETA2+3.)**2/(4.*(4.*FBETA2-3.*FBETA1)*           PURGE553
     2              (2.*FBETA2-3.*FBETA1-6.))                          PURGE554
      CALL OVERFL(J3K)                                                  PURGE555
```

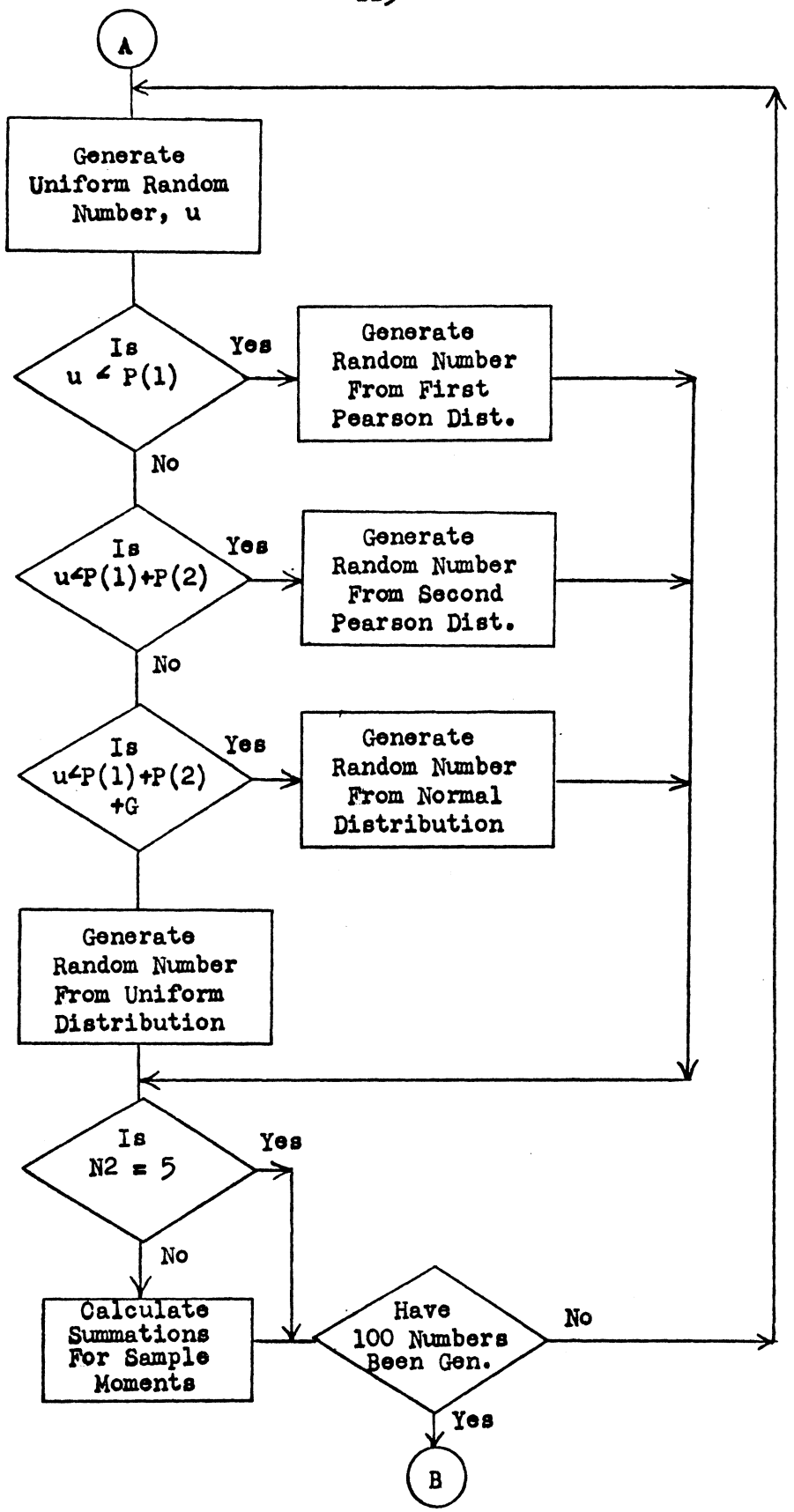
```
      IF(J3K .EQ. 1) WRITE(6,922)                                          PURGE556
  922 FORMAT(6H OVRFL)                                                     PURGE557
C                                          WRITE BOTH SETS OF MOMENTS     PURGE558
      WRITE           (L3,410) LT                                          PURGE559
 410  FORMAT(1H0,27X,32HMOMENTS       FROM ORIGINAL DATA,6X,               PURGE560
     219HFROM GENERATED DATA,6H    N=,I6)                                  PURGE561
  407 WRITE(L3,411)AVE,FA(1),CM2,FCM2,CM3,FCM3,CM4,FCM4,                   PURGE562
     2            BETA1,FBETA1,BETA2,FBETA2,SKAPPA,FKAPPA                 PURGE563
 411  FORMAT (35X,4HMEAN,3X,F18.8,4X,F18.8 /31X,8HVARIANCE,3X,F18.8,4X,    PURGE564
     2            F18.8 /34X,5HMU(3),3X,F18.8,4X,F18.8 /34X,5HMU(4),      PURGE565
     3            3X,F18.8,4X,F18.8 /33X,6HBETA 1,3X,F18.8,4X,F18.8/      PURGE566
     4            33X,6HBETA 2,3X,F18.8,4X,F18.8 /34X,5HKAPPA,3X,         PURGE567
     5            F18.8,4X,F18.8)                                         PURGE568
C                                                                          PURGE569
C                                                                          PURGE570
C                                                                          PURGE571
C                                                                          PURGE572
681   IF(N2.EQ.4) GO TO 680                                                PURGE573
  412 RETURN                                                               PURGE574
C                                                                          PURGE575
C                                          NO GENERATIN WRITE OUT          PURGE576
 148   WRITE           (L3,149)                                            PURGE577
 149   FORMAT(1H0,51X,17HMOMENTS FROM DATA  )                              PURGE578
       WRITE           (L3,150)AVE ,CM2,CM3,CM4,BETA1,BETA2,SKAPPA         PURGE579
 150   FORMAT (54X,4HMEAN,4X,F18.8 /50X,8HVARIANCE,4X,F18.8 /53X,5HMU(3)PURGE580
     2   ,4X,F18.8 /53X,5HMU(4),4X,F18.8 /52X,6HBETA 1,4X,F18.8 /52X,6HBPURGE581
     3ETA 2,4X,F18.8 /53X,5HKAPPA,4X,F18.8)                               PURGE582
       IF(N2 .EQ. 4) GO TO 680                                             PURGE583
       RETURN                                                              PURGE584
C                  GRAPH ROUTINE                                           PURGE585
680   WRITE(L3,950)                                                        PURGE586
      DO 80 I=1,100                                                        PURGE587
 80   CALL ATHRUZ(O(I   ),1H )                                             PURGE588
      DO 81 I=1,5                                                          PURGE589
  81  SID(I)=0.                                                            PURGE590
      BOX=(V7-V6)/99.                                                      PURGE591
      Q=V6+BOX                                                             PURGE592
```

```
      PIP=0.                                                            PURGE593
      SMAX=0.                                                           PURGE594
      SMIN=(10.**10)**10                                                PURGE595
      DO 82 I=2,99                                                      PURGE596
      GO TO (83,84,85,86,87),JAB                                        PURGE597
   83 VAL(I)=PEARIF(V0,V1,V2,V3,V4,Q)                                   PURGE598
      GO TO 88                                                          PURGE599
   84 VAL(I)=PEARIV(V0,V1,V2,V3,V4,Q)                                   PURGE600
      GO TO 88                                                          PURGE601
   85 VAL(I)=NORMAL(V0,V1,Q)                                            PURGE602
      GO TO 88                                                          PURGE603
   86 VAL(I)=PEARVI(V0,V1,V2,V3,V4,Q)                                   PURGE604
      GO TO 88                                                          PURGE605
   87 VAL(I)=PERIII(V0,V1,V2,V3,Q)                                      PURGE606
   88 BOT(I)=Q+AVE                                                      PURGE607
      Q=Q+BOX                                                           PURGE608
      IF(VAL(I)-SMAX) 93,93,89                                          PURGE609
   89 SMAX=VAL(I)                                                       PURGE610
   93 IF(VAL(I)-SMIN) 94,82,82                                          PURGE611
   94 SMIN=VAL(I)                                                       PURGE612
   82 CONTINUE                                                          PURGE613
      N=0                                                               PURGE614
      DO 95 K=1,5                                                       PURGE615
      E=8*(6-K)                                                         PURGE616
      SID(K)=SMIN+E/39.*(SMAX-SMIN)                                     PURGE617
      DO 95 J=1,8                                                       PURGE618
      N=N+1                                                             PURGE619
      NA(1)=1                                                           PURGE620
      NA(2)=1                                                           PURGE621
      M1=0                                                              PURGE622
      DO 96 I=2,99                                                      PURGE623
      TMP    =(VAL(I)-SMIN)/(SMAX-SMIN)*39.+1.                          PURGE624
      L=TMP                                                             PURGE625
      L=41-L                                                            PURGE626
      IF(N.NE.L) GO TO 96                                               PURGE627
      CALL ATHRUZ(O(I),1H*)                                             PURGE628
      M1=M1+1                                                           PURGE629
```

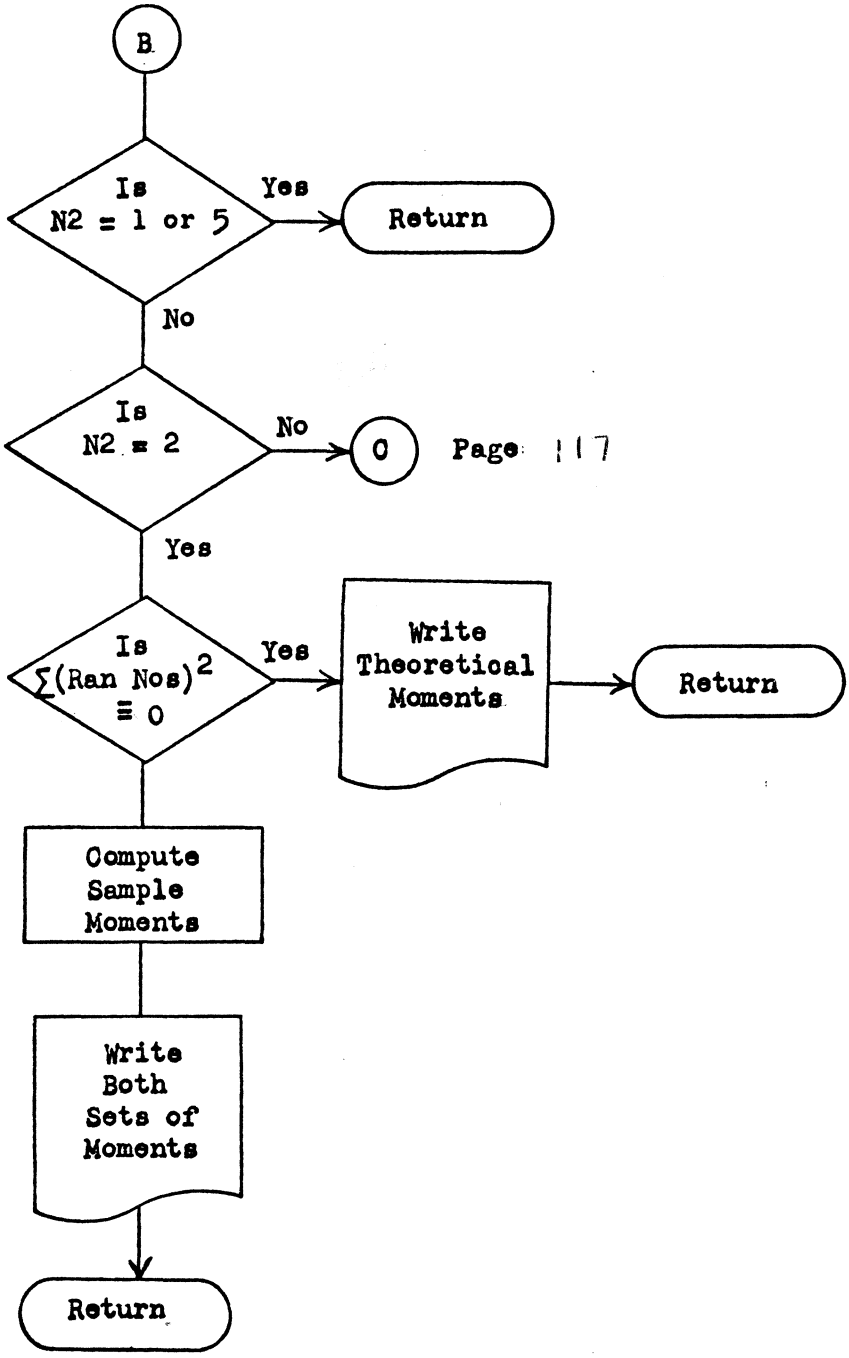
```
      NA(M1)=I                                                          PURGE630
   96 CONTINUE                                                          PURGE631
      GO TO (97,98,98,98,98,98,98,98),J                                 PURGE632
   97 WRITE (L3,99) SID(K),(O(I),I=1,100)                               PURGE633
   99 FORMAT(2X,E9.2,2H +,100A1)                                        PURGE634
      GO TO 65                                                          PURGE635
   98 WRITE(L3,66) (O(I),I=1,100)                                       PURGE636
   66 FORMAT(13X,100A1)                                                 PURGE637
   65 DO 95 I=1,M1                                                      PURGE638
      L=NA(I)                                                           PURGE639
   95 CALL ATHRUZ(O(L),1H )                                             PURGE640
C                                                                       PURGE641
C                                                                       PURGE642
      WRITE           (L3,67)(BOT(I),I=10,90,10)                        PURGE643
   67 FORMAT ( /13X,9(9X,1H+)/20X,9E10.3)                               PURGE644
      RETURN                                                            PURGE645
C                                        ERROR NOTIFICATION             PURGE646
  750 WRITE(L3,751) BETA1,BETA2                                         PURGE647
  751 FORMAT(3X,2F18.8,29H ILLEGAL VALUES FOR B1 AND B2)                PURGE648
      GO TO 983                                                         PURGE649
  903 WRITE           (L3,904)D1,D2                                     PURGE650
  904 FORMAT (3X,2F18.8,29H ILLEGAL VALUES FOR M1 AND M2)               PURGE651
       GO TO 983                                                        PURGE652
  906 WRITE           (L3,907)B1,B2                                     PURGE653
  907 FORMAT (3X,2F18.8,29H ILLEGAL VALUES FOR A1 AND A2)               PURGE654
       GO TO 983                                                        PURGE655
  910  WRITE           (L3,911)                                         PURGE656
  911  FORMAT (3X,33HTOO MUCH AND / OR TOO LARGE DATUM)                 PURGE657
C                                        SKIP          TO NXT PASS     PURGE658
  983 LIMIT=0                                                           PURGE659
      RETURN                                                            PURGE660
  605 STOP                                                              PURGE661
      END                                                               PURGE662
```



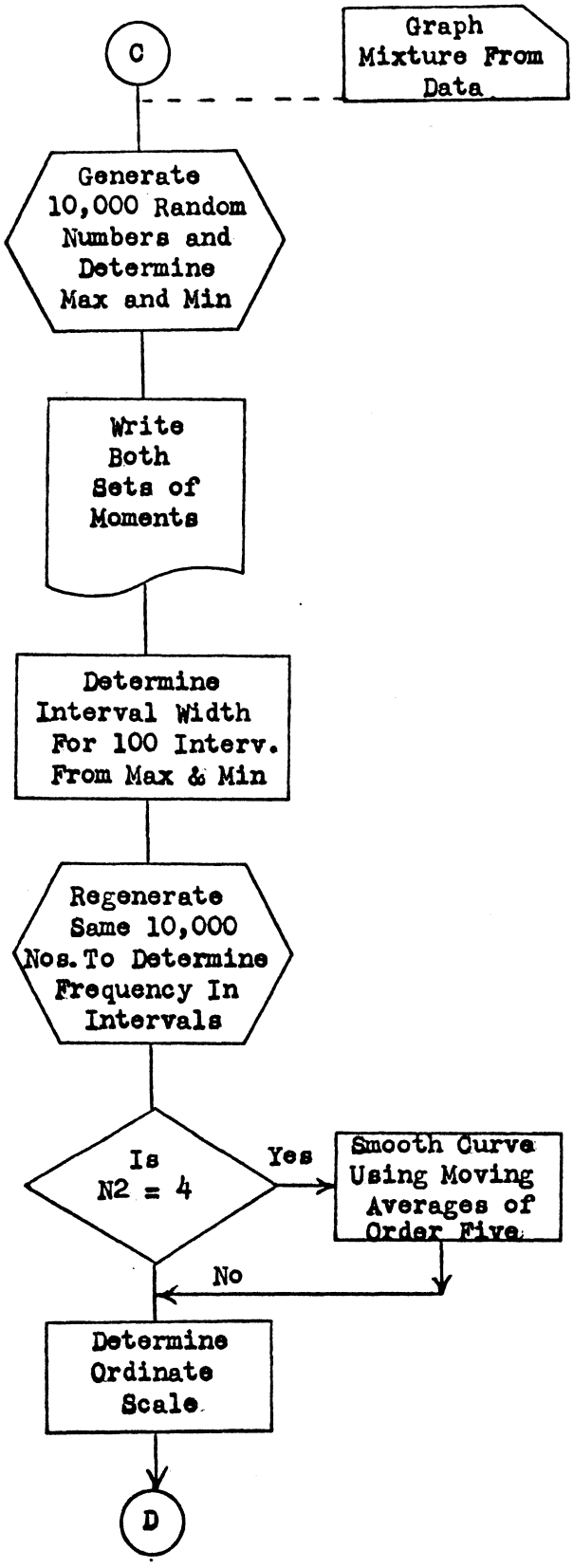
Flowchart For MIX



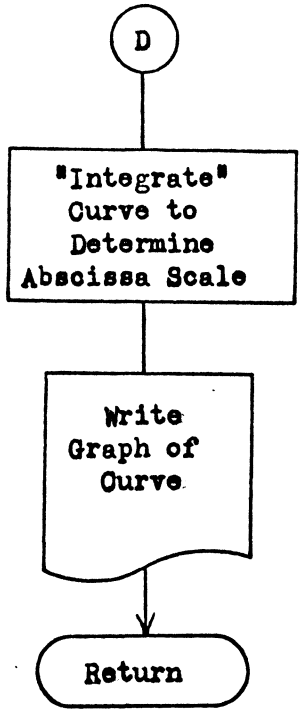
Flowchart For MIX, ctd.

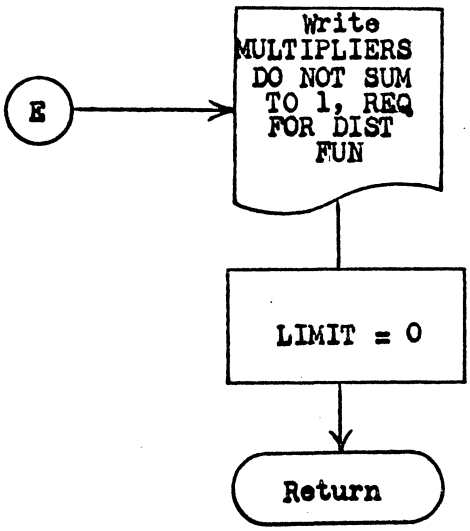


Flowchart For MIX, ctd.



Flowchart For MIX, ctd.





Flowchart For MIX, ctd.

```
C                                                                       MIX001
      SUBROUTINE MIX(MKX,MKZ)                                           MIX002
      N1=MKX                                                            MIX003
      N2=MKZ                                                            MIX004
      EXTERNAL NOLMUN,NOLMOV                                            MIX005
C                              FOR MIX OF DIST                          MIX006
C     FUNCTIONS TO COMPUTE NONCENTRAL MOMENTS                           MIX007
      RM2(U1,U2)       =U2+U1**2                                        MIX008
      RM3(U1,U2,U3)  =U3+3.*U1*U2+U1**3                                 MIX009
      RM4(U1,U2,U3,U4)=U4+4.*U1*U3+6.*U2*U1**2+U1**4                    MIX010
      EXTERNAL NUMBR                                                    MIX011
      DATA NR1,Q/1,0./                                                  MIX012
      DIMENSION O(100),NA(100),BOT(9),VAL(101),SID(5),NAL(100)          MIX013
      DIMENSION S(2,2641),N(2)          ,KT(2),EL(2),H(2),C1(2),C2(2),  MIX014
     1C3(2),C4(2),P(2),FAN(4),FA(4),KU(2,101)                           MIX015
      EQUIVALENCE(S,O),(S(1,100),NA),(S(1,200),NAL)                     MIX016
      COMMON/Z1/TM1/Z2/TOM(100)/Z3/LIMIT/Z4/FA/Z5/FAN/Z6/N/Z7/KU/Z8/S/  MIX017
     1Z9/CM2,CM3,CM4/Z10/FCM2,FCM3,FCM4/Z11/Q/Z12/P     ,G,GM,U,UL,UP   MIX018
      LIMIT=100                                                         MIX019
      GO TO (15,19),N1                                                  MIX020
C                              READ OPTION CARD                         MIX021
   15 READ(5,101) P(1),P(2),G,GM,GV,U,UL,UP                             MIX022
  101 FORMAT(3F7.4,2F10.5,F7.4,2F10.5)                                  MIX023
      IF(P(1).GT. 1.) STOP                                              MIX024
      LT=0                                                              MIX025
      DO 126 L=1,4                                                      MIX026
  126 FAN(L)=0.                                                         MIX027
      ASSIGN 60 TO KY                                                   MIX028
      IF((P(2) .EQ. 0.) .AND. (G.EQ. 0.) .AND. (U .EQ. 0.)) ASSIGN 51   MIX029
     1TO KY                                                             MIX030
      M=1                                                               MIX031
      IF(P(1) .EQ. 0.) GO TO 110                                        MIX032
      L=1                                                               MIX033
      IF(P(2) .NE. 0.) L=2                                              MIX034
      DO 102 I=1,L                                                      MIX035
      READ(5) N(I),(KU(I,J),J=1,101),KT(I),EL(I),H(I),C1(I),C2(I),C3(I),MIX036
     1C4(I)                                                             MIX037
```

```
      K=N(I)                                                                MIX038
  102 READ(5) (S(I,J),J=1,K)                                                MIX039
  110 WRITE(6,125) P(1),P(2),G,GM,GV,U,UL,UP                                MIX040
  125 FORMAT(1H1,92X,21HDISTRIBUTION MIXTURES/13HOCONTROL CARD,6H   P1=,    MIX041
     1,F10.4,6H   P2=,F10.4,6H NORM=,F10.4,6H MEAN=,F12.5,6H  VAR=,F12.5    MIX042
     2/1HO,12X,6H UNIF=,F10.4,6H  FROM,F10.4,6H    TO,F10.4)               MIX043
      IF(ABS(1.-P(1)-P(2)-G-U) .GE. .001) GO TO 100                         MIX044
      IF(G .EQ. 0.) GO TO 40                                                MIX045
C                              FOR NORMAL                                   MIX046
      IF(NR1 .NE. 1) GO TO 160                                              MIX047
      NR1=0                                                                 MIX048
      BG=RNST(Q)                                                            MIX049
  160 BG=SQRT(GV)                                                           MIX050
      DG=3.*GV**2                                                           MIX051
   40 IF(U .EQ. 0.) GO TO 41                                                MIX052
C                              COMPUTE CENTRAL MOMENTS FOR UNIFORM          MIX053
      AU=(UL+UP)/2.                                                         MIX054
      BU=(UP-UL)**2/12.                                                     MIX055
      DU=(UP-UL)**4/80.                                                     MIX056
C                              COMPUTE NONCENTRAL MOMENTS FOR MIX           MIX057
   41 TM1=P(1)*C1(1) + P(2)*C1(2) + G*GM + U*AU                             MIX058
      TM2=P(1)*RM2(C1(1),C2(1)) + P(2)*RM2(C1(2),C2(2)) + G*RM2(GM,GV)      MIX059
     1+ U*RM2(AU,BU)                                                        MIX060
      TM3=P(1)*RM3(C1(1),C2(1),C3(1)) + P(2)*RM3(C1(2),C2(2),C3(2)) +       MIX061
     1G*RM3(GM,GV,0.) + U*RM3(AU,BU,0.)                                     MIX062
      TM4=P(1)*RM4(C1(1),C2(1),C3(1),C4(1)) + P(2)*RM4(C1(2),C2(2),C3(2)    MIX063
     1,C4(2)) + G*RM4(GM,GV,0.,DG) + U*RM4(AU,BU,0.,DU)                     MIX064
C                              COMPUTE CENTRAL MOMENTS                      MIX065
      REWIND 0                                                              MIX066
      WRITE(0) NUMBR                                                        MIX067
      CM2=TM2-TM1**2                                                        MIX068
      CM3=TM3-3.*TM1*TM2+2.*TM1**3                                          MIX069
      CM4=TM4-4.*TM1*TM3+6.*TM2*TM1**2-3.*TM1**4                            MIX070
C                              GENERATE RANDOM NOS.                         MIX071
      P(2)=P(2)+P(1)                                                        MIX072
      G=G+P(2)                                                              MIX073
      UR=UP-UL                                                              MIX074
```

```
      CALL OVERFL(J3K)                                                  MIX075
   19 GO TO (20,20,503,504,16),N2                                       MIX076
   16 N1=3                                                              MIX077
   20 DO 393 K=1,LIMIT                                                  MIX078
      GO TO KY,(60,51)                                                  MIX079
   52 IF(RN1 .GE. G) GO TO 53                                           MIX080
      TOM(K)=(RNOR(0.)*BG)+GM                                           MIX081
      GO TO 675                                                         MIX082
   53 TOM(K)=RDM(1.0)*UR+UL                                             MIX083
      GO TO 675                                                         MIX084
   60 RN1=RDM(1.0)                                                      MIX085
      IF(RN1 .GE. P(1)) GO TO 50                                        MIX086
      M=1                                                               MIX087
      GO TO 51                                                          MIX088
   50 IF(RN1 .GE. P(2)) GO TO 52                                        MIX089
      M=2                                                               MIX090
   51 RND=RDM(1.0)                                                      MIX091
      NR=INT(100.*RND)+1                                                MIX092
      L=KU(M,NR)                                                        MIX093
  671 J=N(M)                                                            MIX094
      DO 387 I=L,J                                                      MIX095
      IF(S(M,I)-RND) 387,390,388                                        MIX096
  387 CONTINUE                                                          MIX097
388   BUT=FLOAT(I+I-2)-2.*(S(M,I)-RND)/(S(M,I)-S(M,I-1))                MIX098
C                                                                       MIX099
      GO TO 391                                                         MIX100
  390 BUT=I+I-2                                                         MIX101
  391 TOM(K)=EL(M)+BUT*H(M)+C1(M)                                       MIX102
  675 GO TO (683,683,393),N1                                            MIX103
  683 DO 685 KK=1,4                                                     MIX104
  685 FAN(KK)=FAN(KK)+ TOM(K)**KK                                       MIX105
  393 CONTINUE                                                          MIX106
      LT=LT+LIMIT                                                       MIX107
      GO TO (952,400,501,502,952),N2                                    MIX108
  952 RETURN                                                            MIX109
C                              CAL OUTPUT MOMENTS                       MIX110
  400 TP=LT                                                             MIX111
```

```
      DO 405 KK=1,4                                                    MIX112
  405 FA(KK)=FAN(KK)/TP                                                MIX113
      FCM2=FA(2)-FA(1)**2                                              MIX114
      FCM3=FA(3)-3.*FA(1)*FA(2)+ 2.*FA(1)**3                           MIX115
      FCM4=FA(4)-4.*FA(1)*FA(3) +6.*FA(1)**2*FA(2)-3.*FA(1)**4         MIX116
      CALL OVERFL(J3K)                                                 MIX117
      IF(J3K .EQ. 1 ) WRITE(6,922)                                     MIX118
  922 FORMAT(6H OVRFL)                                                 MIX119
      IF(FAN(2).EQ.0.) GO TO 148                                       MIX120
C                              WRITE BOTH SETS OF MOMENTS              MIX121
      WRITE(6,410) LT                                                  MIX122
  410 FORMAT(1H0,27X,32HMOMENTS        FROM ORIGINAL DATA,6X,19HFROM GENE  MIX123
     1RATED DATA,6H    N=,I6)                                          MIX124
  407 WRITE(6,411) TM1,FA(1),CM2,FCM2,CM3,FCM3,CM4,FCM4                MIX125
  411 FORMAT(35X,4HMEAN,3X,F18.8,4X,F18.8/31X,8HVARIANCE,3X,F18.8,4X,  MIX126
     2F18.8/34X,5HMU(3),3X,F18.8,4X,F18.8/34X,5HMU(4),3X,F18.8,4X,F18.8/  MIX127
     3)                                                                MIX128
      GO TO 412                                                        MIX129
  148  WRITE            (6 ,149)                                       MIX130
  149  FORMAT(1H0,51X,17HMOMENTS FROM DATA  )                          MIX131
       WRITE            (6 ,150)TM1 ,CM2,CM3,CM4                       MIX132
  150  FORMAT (54X,4HMEAN,4X,F18.8 /50X,8HVARIANCE,4X,F18.8 /53X,5HMU(3)  MIX133
     2   ,4X,F18.8 /53X,5HMU(4),4X,F18.8)                              MIX134
  412 IF(N2.EQ.4) GO TO 75                                             MIX135
      RETURN                                                           MIX136
C                              GRAPH ROUTINE                           MIX137
  503 ASSIGN 904 TO KS                                                 MIX138
      GO TO 500                                                        MIX139
  504 ASSIGN 903 TO KS                                                 MIX140
      N2=3                                                             MIX141
  500 DO 1 I=1,100                                                     MIX142
    1 VAL(I)=0.                                                        MIX143
      VAL(101)=0.                                                      MIX144
      SMIN=TM1                                                         MIX145
      SMAX=TM1                                                         MIX146
      Q=RNEX (Z)                                                       MIX147
      REWIND 0                                                         MIX148
```

```
      WRITE(0) NUMBR                                                    MIX149
      LT=0                                                              MIX150
      DO 39 I=1,4                                                       MIX151
   39 FAN(I)=0.                                                         MIX152
      NN=0                                                              MIX153
   70 NN=NN+1                                                           MIX154
      IF(NN .EQ. 101) GO TO 71                                          MIX155
      GO TO 20                                                          MIX156
  501 DO 3 NR=1,100                                                     MIX157
      IF(TOM(NR) .LT. SMIN) SMIN=TOM(NR)                                MIX158
      IF(TOM(NR) .GT. SMAX) SMAX=TOM(NR)                                MIX159
    3 CONTINUE                                                          MIX160
      GO TO 70                                                          MIX161
   71 FIL=(SMAX-SMIN)/100.                                              MIX162
      Z =RNST(Q)                                                        MIX163
      REWIND 0                                                          MIX164
      READ(0) NUMBR                                                     MIX165
      N1=3                                                              MIX166
      N2=4                                                              MIX167
      NN=0                                                              MIX168
      GO TO 400                                                         MIX169
   75 NN=NN+1                                                           MIX170
      IF(NN .EQ. 101) GO TO 76                                          MIX171
      GO TO 20                                                          MIX172
  502 DO 7 NR=1,100                                                     MIX173
      J=(TOM(NR)-SMIN)/FIL + 1.                                         MIX174
    7 VAL(J)=VAL(J)+1.                                                  MIX175
      GO TO 75                                                          MIX176
   76 VAL(100)=VAL(100)+VAL(101)                                        MIX177
      VL=VAL(50)                                                        MIX178
      VS=VL                                                             MIX179
      VLT=0.                                                            MIX180
      X=VAL(1)                                                          MIX181
      Y=VAL(2)                                                          MIX182
      GO TO KS,(903,904)                                                MIX183
  903 DO 10 I=3,98                                                      MIX184
      TMP=VAL(I)                                                        MIX185
```

```
      VAL(I)=(X+Y+VAL(I)+VAL(I+1)+VAL(I+2))/5.                    MIX186
      X=Y                                                         MIX187
   10 Y=TMP                                                       MIX188
      VAL(2)=(VAL(1)+VAL(2)+VAL(3))/3.                            MIX189
      VAL(99)=(VAL(98)+VAL(99)+VAL(100))/3.                       MIX190
  904 DO 901 I=1,100                                              MIX191
      CALL ATHRUZ(O(I),1H )                                       MIX192
      IF(VAL(I)  .GT. VL) VL=VAL(I)                               MIX193
      IF(VAL(I).LT.VS) VS=VAL(I)                                  MIX194
  901 CONTINUE                                                    MIX195
C                DET Y SCALE                                      MIX196
      DO 13 I=1,100                                               MIX197
   13 VLT=VLT+VAL(I)                                              MIX198
      R=39./(VL-VS)                                               MIX199
      A=(1./(FIL*VLT*R))*8.                                       MIX200
      DO 11 I=2,99                                                MIX201
   11 NAL(I)=(VAL(I)-VS)*R + 1.5                                  MIX202
      DO 12 I=1,9                                                 MIX203
      F=I*10                                                      MIX204
   12 BOT(I)=SMIN+F*FIL                                           MIX205
      M=41                                                        MIX206
      DO 95 K=1,5                                                 MIX207
      E=6-K                                                       MIX208
      SID(K)=A*E                                                  MIX209
      DO 95 J=1,8                                                 MIX210
      M=M-1                                                       MIX211
      NA(1)=1                                                     MIX212
      M1=0                                                        MIX213
      DO 96 I=2,99                                                MIX214
      IF(M .NE. NAL(I)) GO TO 96                                  MIX215
      CALL ATHRUZ(O(I),1H*)                                       MIX216
      M1=M1+1                                                     MIX217
      NA(M1)=I                                                    MIX218
   96 CONTINUE                                                    MIX219
       GO TO (97,98,98,98,98,98,98,98),J                          MIX220
   97 WRITE(6,99) SID(K),(O(I),I=1,100)                           MIX221
   99 FORMAT(2X,E9.2,2H +,100A1)                                  MIX222
```

```
      GO TO 65                                                          MIX223
   98 WRITE(6,66) (O(I),I=1,100)                                        MIX224
   66 FORMAT (13X,100A1)                                                MIX225
   65 DO 95 I=1,M1                                                      MIX226
      L=NA(I)                                                           MIX227
   95 CALL ATHRUZ(O(L),1H )                                             MIX228
      WRITE(6,67) (BOT(I),I=1,9)                                        MIX229
   67 FORMAT (//13X,9(9X,1H+)/20 X,9E10.3)                              MIX230
      RETURN                                                            MIX231
C                              ERROR NOTIFICATION                      MIX232
  100 WRITE(6,120)                                                      MIX233
  120 FORMAT(1H0,30X,39HMPYRS DO NOT SUM TO 1, REQ FOR DIST FUN)        MIX234
      LIMIT=0                                                           MIX235
      RETURN                                                            MIX236
      END                                                               MIX237
```

ABSTRACT

This thesis contains a brief review of some of the work that has been done concerning the generation and testing of pseudo-random numbers. Computer subroutine programs written in FORTRAN IV are given for the generation of pseudo-random numbers from 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 Pearson distribution may be specified either by the first four moments or from sample data, then the parameters of the fitted distribution are printed and, if desired, a graph of the distribution. A graph of the mixture of distributions may be obtained from 10,000 pseudo-random numbers from the mixture.

The speed of generation varies from about 10,000 random numbers per minute (on the IBM 7040), for a Pearson distribution with moments calculated from the generated numbers, to more than 100,000 numbers per minute if mixtures are used.

The subroutines are applied to a Monte Carlo investigation of the robustness of several methods of confidence interval estimation.