# REJECTION METHODS FOR GENERATING RANDOM DEVIATES AND THEIR APPLICATIONS IN SYSTEM RELIABILITY

By

KEUN KUK LEE

Bachelor of Economics
Yonsei University
Seoul, Korea
1968

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1976

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
December, 1977

REJECTION METHODS FOR GENERATING RANDOM DEVIATES AND THEIR APPLICATIONS IN SYSTEM RELIABILITY

Thesis Approved:

J P Chandler
Thesis Adviser

Donald W Grace

J Leroy Folks

Norman N. Durham
Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

## PREFACE

This thesis is a description of the development of fast methods of generating random deviates, and of their applications for the estimation of system reliability through computer simulation.

I am very grateful to have had Dr. J.P. Chandler as my major advisor. I appreciate very much his guidance, assistance, and consideration through all phases of my research. An expression of gratitude is extended to Dr. D.W. Grace and Dr. J.L. Folks for serving on my advisory committee. I would like to express my special appreciation to Dr. Mitchell O. Locks who gave me financial support and constructive comments for this study. Finally, a sincere thanks is extended to my parents, Mr. & Mrs. Sung Wook Lee and my wife, Young Ra, for their interest, encouragement, and support during my college career.

# CHAPTER I

## INTRODUCTION

Many products or production systems are complex systems, and estimating the reliability of the total system is important. As the final complex system is expensive, it is not possible in many cases to test a sufficient number of products to obtain an accurate estimate of system reliability. Because statistical experimentation over the complex system is expensive, it is often necessary to predict or estimate system reliability based on component or subsystem test data.

When a system configuration is defined, a simulation technique can be applied to estimate system reliability by using a Monte Carlo sampling technique based upon historical component data which provides a Bayesian prior distribution.

For a success-failure component, the distribution is often assumed to have Beta probability density function. For a time-to-failure component, the distribution of the failure rate is often assumed to follow a Gamma density function. To achieve the desirable result of the distribution of system reliability, accurate and fast uniform, Beta, and Gamma deviate generators should be provided.

Chapter II describes brief assessments of some of the previous methods for obtaining Beta and Gamma deviates. Chapter III presents fast methods for generating Beta and Gamma deviates. Also a uniform pseudo-random deviate generator is discussed. Chapter IV discusses the relationship between component and system reliability. The

characterization of the distribution of system reliability resulting from using the Monte Carlo simulation technique is also discussed as a part of the conclusions.

# CHAPTER II

## PREVIOUS METHODS OF GENERATING BETA AND GAMMA DEVIATES

There are three basic general techniques (as opposed to specialized "trick" methods) for generating deviates from continuous probability distributions, the "inverse transformation method," the "rejection method," and the "composition method." These methods or some variation of them give random deviates which simulate most of the continuous distributions (36).

For the inverse transformation method, the cumulative distribution function (cdf), corresponding to a probability density function (pdf), can be inverted by using values selected from the uniform distribution. Suppose deviates X are to be sampled from the pdf f(x). Then

$$F(x) = \int_{-\infty}^{x} f(w)\, dw$$

is the cdf corresponding to f(x). If F(x) is continuous and strictly increasing, it takes on all values between 0 and 1, and there is an inverse function $F^{-1}(y)$ such that, if $0 < y < 1$, $y = F(x)$ if and only if $x = F^{-1}(y)$.

An easy way to compute a random variable X with a continuous strictly increasing distribution F(x) is set $X = F^{-1}(U)$. For the probability that X  x is the probability that $F^{-1}(U) \leq x$. Consequently, $F^{-1}(u_i)$ is a random variable that has probability density.

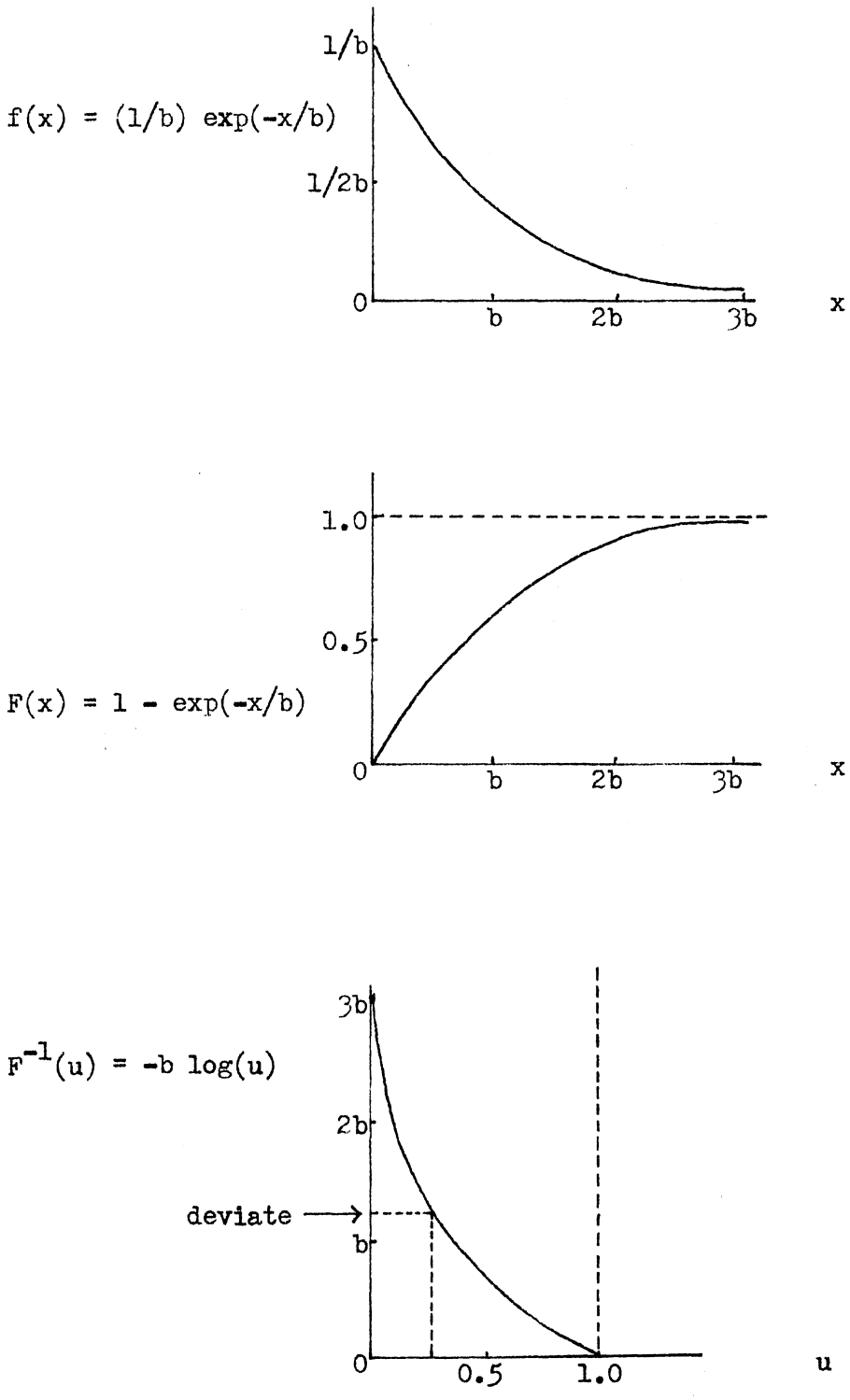



Figure 1. Inverse Transformation Method

Theorem: Let the random variable X have a continuous strictly increasing distribution function F on the interval [a,b]. Then the random variable Y = F(x) has a uniform distribution over the interval [0,1].

Proof: Denote G the distribution function of Y. Then

$$G(y) = P(Y \le y) = P(F(x) \le y)$$

$$= \begin{cases} 0 & , \quad y < 0 \\ P(X \le F^{-1}(y)) = F(F^{-1}(y)) = y & , \quad 0 \le y \le 1 \\ 1 & , \quad 1 \le y \end{cases}$$

where $a \le F^{-1}(y) \le b$. These three random events $\{Y \le y\}$, $\{F(X) \le y\}$, and $\{X \le F^{-1}(y)\}$ are equivalent, therefore their probabilities are equal.

In many cases the inverse transformation method is not practical unless the inverse function $F^{-1}(u_i)$ can be obtained explicitly.

Secondly, if a probability density function f(x) is bounded and x has a finite range, the rejection techniques (44) can be used to generate random deviates. The application of this technique requires the following steps:

1) Normalize the range of f(x) by a scale factor s, such that

$$s \cdot f(x) \le 1, \quad a \le x \le b.$$

2) Generate pairs of uniform random deviates $u_0$ and $u_1$.

3) Define x as a linear function of $u_0$ such as

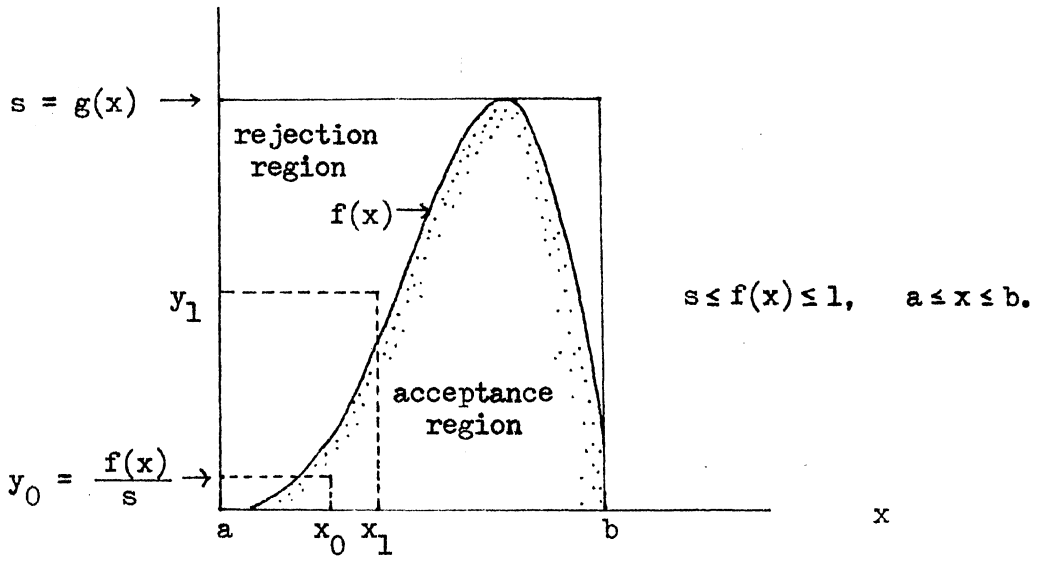
$$x = a + (b - a) \cdot u_0.$$

4) Whenever there is a pair of random deviate that satisfies the conditional probability relationship

$$u_1 \le s \cdot f(x),$$

then accept $u_0$ and use

$$x = a + (b - a)\, u_0$$



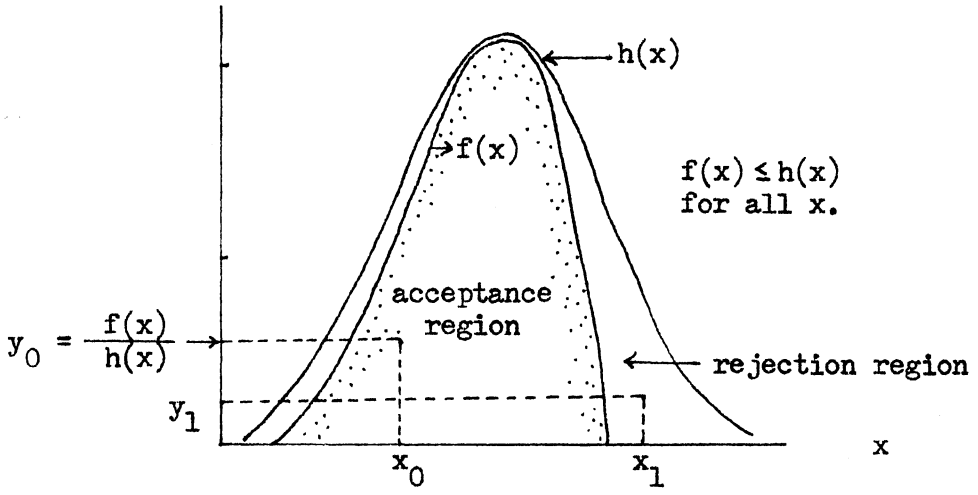




Figure 2. Rejection Method

as random deviate generated. Otherwise, reject $u_0$ and repeat steps from 2.

The theory behind this method is based on the fact that the probability of U being less than or equal to $s \cdot f(x)$ is exactly $f(x)$. Consequently, an x accepted at random from the range (a,b) will have exactly the probability density function $f(x)\,dx$. The expected number of trials before a successful pair is found is equal to $1/s$. This implies that this method may be quite inefficient for certain probability density function.

The composition method (14) samples from a probability density function by composing properly selected density functions which make the desired probability density function f(y) such that

$$f(y) = \int_{-\infty}^{\infty} g_z(y)\, dH(z)$$

where $g_z(y)$ is a probability density function which depends on the parameter z; further H(z) is the cumulative distribution function of random variable Z. In order to generate random deviates Y having a probability density function f(y), one draws a deviate having the cumulative distribution function H(z); then draw a second sample having the probability density function $g_z(y)$.

## Beta Random Variables

A beta random variable X has a probability density function

$$f(x) = \frac{1}{B(a,b)}\, x^{a-1}\,(1-x)^{b-1}, \qquad 0 \le x \le 1, \quad a,b > 0$$

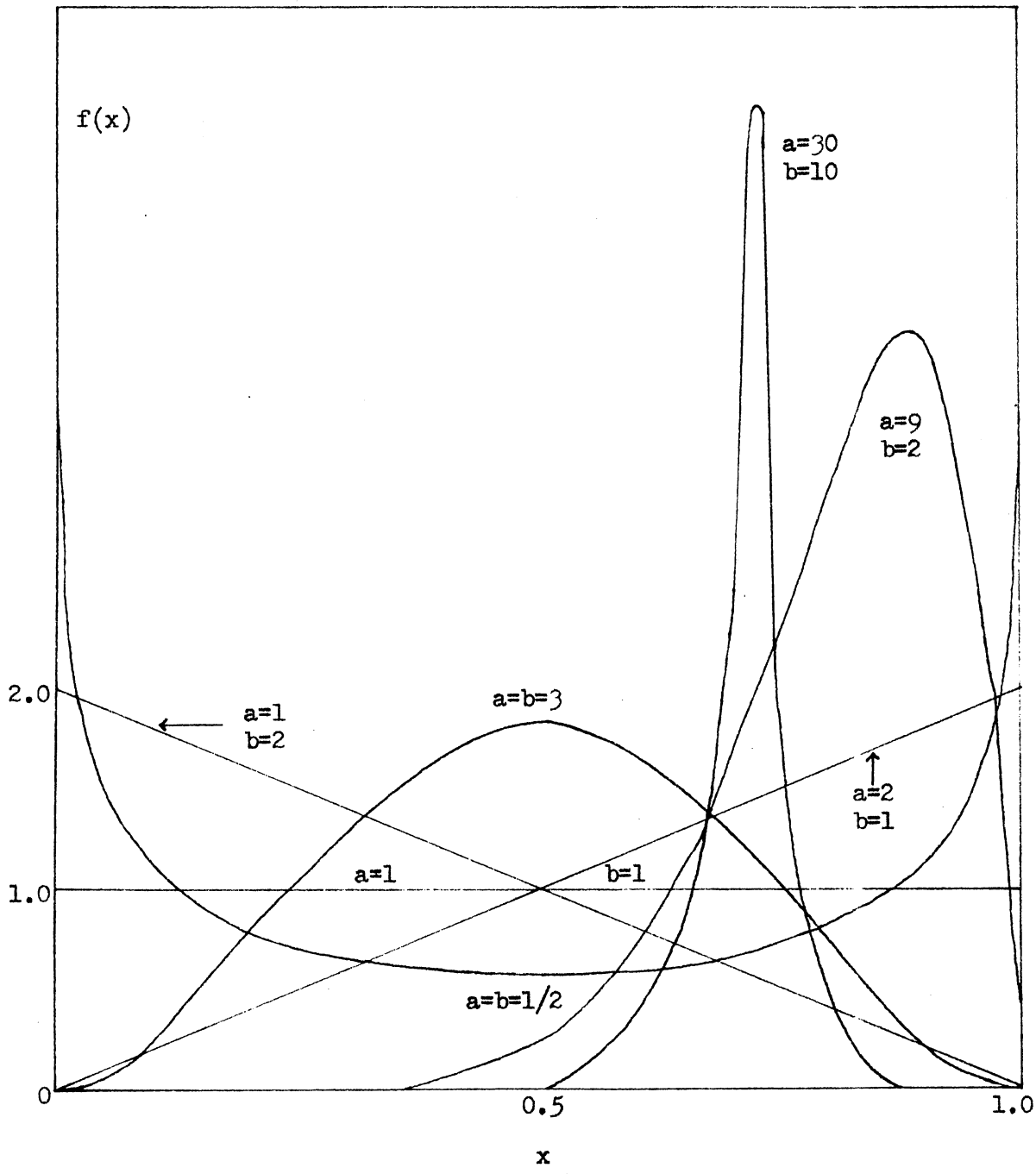



Figure 3. Beta Density Functions

where

$$B(a,b) = \int_0^x t^{a-1} (1 - t)^{b-1} \, dt.$$

The probability density function has mean equal to $a/(a + b)$.

The variance is

$$ab/((a + b)^2 (a + b + 1))$$

and the mode is

$$(a - 1)/(a + b - 2) \quad \text{where } a > 1, \; b > 1.$$

The beta probability density function has various shapes corresponding to the sizes of the parameters.

The probability integral of the probability density function up to x is called the incomplete beta function ratio and is denoted by $I_x(a,b)$, so that

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1} (1 - t)^{b-1} \, dt.$$

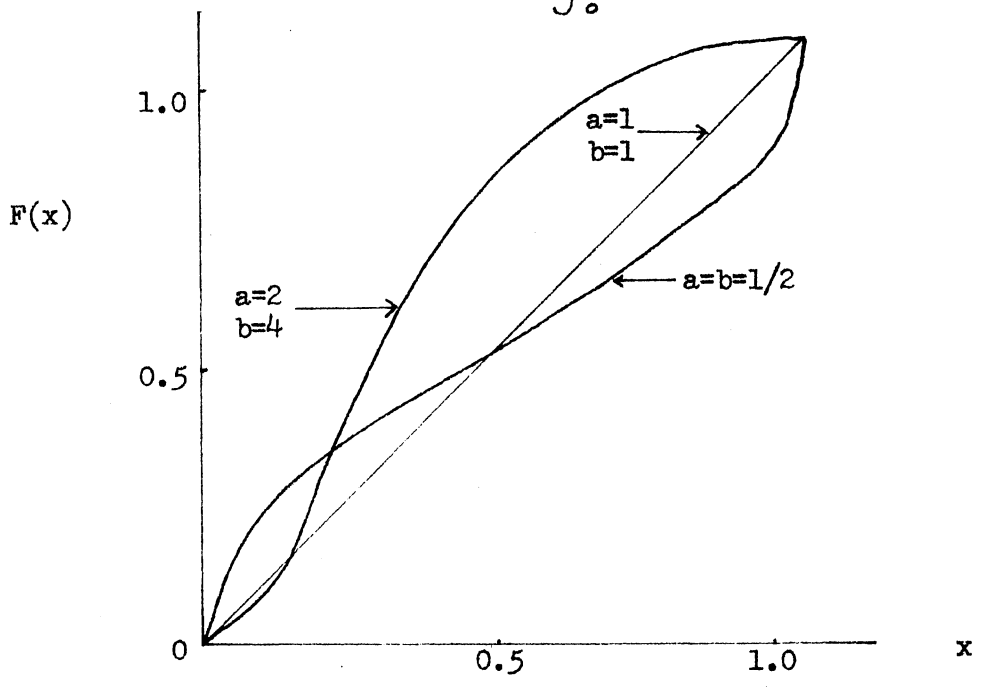



Figure 4. Beta Cumulative Distribution Functions

## Methods for Generating Beta Deviates

Four different previous techniques for obtaining beta deviates are analyed. The methods are from M.D. Johnk (17), an inverse beta probability distribution function subroutine (MDBETI) from International Mathematics and Statistical Libraries (IMSL) (7), an approximation to the inverse beta probability distribution function from National Bureau of Standards (1), and Ahrens-Dieter (2).

### Johnk's Method

Johnk has shown that if X and Y are chosen from independent standard uniform distributions in the interval (0,1) then the conditional distribution $X^{1/a}$ given that

$$X^{1/a} + Y^{1/b} \leq 1,$$

is a standard beta distribution with parameter a and b. In other words, if $U_1$ and $U_2$ are continuous uniform random variables and

$$X = U_1^{1/a}, \quad Y = U_2^{1/b},$$

then if $X + Y \leq 1$,

$$Z = \frac{X}{X + Y}$$

follows a beta distribution with distribution function

$$f(z) = \frac{\Gamma(a + b)}{\Gamma(a)\,\Gamma(b)} \; z^{a-1} \, (1 - z)^{b-1}, \qquad 0 \leq z \leq 1.$$

This method is implemented in FORTRAN in IMSL under the function subroutine name GGBTA as following procedure.

1) $a = 1/a$

2) $b = 1/b$

3) Obtain two uniform deviates, $u_1$ and $u_2$.

4) $x = u_1^{a}$

5) $y = u_2^{b}$

6) If $x + y$ is greater than 1, then go to step 3.

7) Otherwise $d = x/(x + y)$ and accept d as beta deviate.

Though this method is exact and simple to implement, a computer may require an enormous excution time to obtain a deviate if "a" and "b" are large. This means that step 6 requires a lot of rejections to satisfy the condition

$$u_1^{1/a} + u_2^{1/b} \leq 1.$$

Using the compiler WATFIV in IBM 370 Model 158, for the parameter value a = 10, b = 10, the function subroutine GGBTA from IMSL took more than one minute computing time to obtain a beta deviate. This rejection technique consumes a time which increases without bound, to obtain one deviate, as "a" and "b" increase. Even for a = 10 and b = 10 it is completely impractical, obviously.

Inverse Incomplete Beta Distribution Function Subroutine

An IMSL package subroutine MDBETI computes the values of percentage point (deviate), x, such that the probability of a beta (a,b) random variable being not greater than x is p, where the parameters, a and b, and the value of probability, p, are given. MDBETI finds x at the zero of the function $I_x(a,b) - p$ by the bisection method. In the computation procedure MDBETI bisects an interval on x axis and evaluates the probability at x by calling the incomplete

beta probability function subroutine MDBETA in the IMSL Library repeatedly until the absolute value of function $I_x(a,b) - p$ is less than or equal to 1·E - 5. The maximum number of iterations is 30.

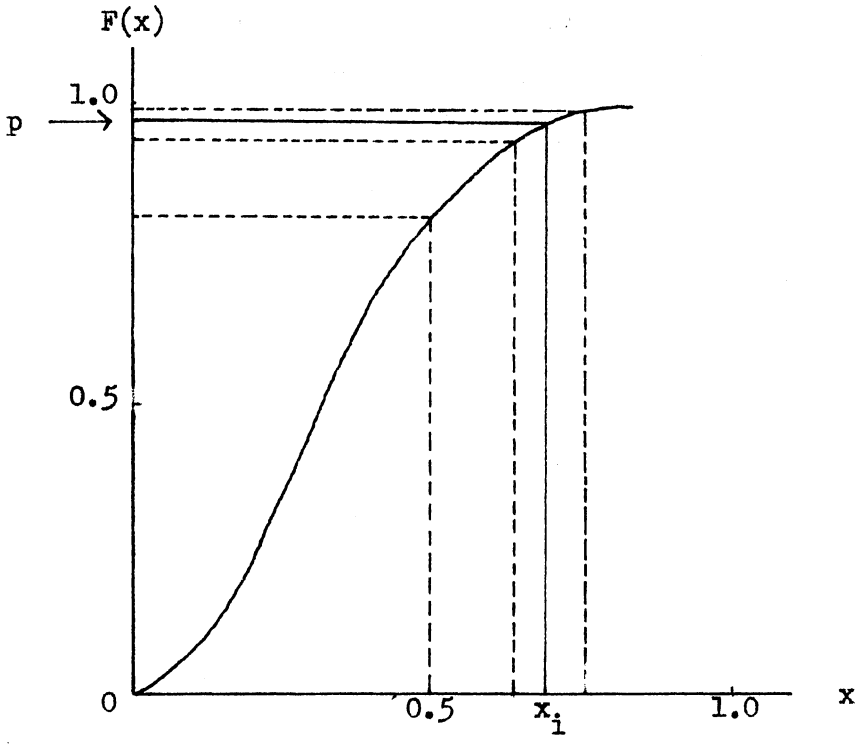



Figure 5. Bisection Searching Procedure

MDBETA is based on the CACM algorithm 179 (Incomplete Beta Ratio) (3). MDBETA computes the incomplete beta ratio using the equation

$$I_x(a,b) = \frac{s \cdot x^a \, \Gamma(r + a)}{\Gamma(r)\,\Gamma(a + 1)} + \frac{x^a (1 - x)^b \cdot \Gamma(a + b) \cdot t}{\Gamma(a)\,\Gamma(b + 1)}$$

where

$$s = \sum_{i=0}^{\infty} \frac{(1 - r)_i \, a}{a + i} \cdot \frac{x^i}{i}$$

where

$$(1 - r)_i = 1, \qquad i = 0$$

$$= (1 - r)(2 - r)\ldots(i - r)$$

$$= \frac{\Gamma(1 + i - r)}{\Gamma(1 - r)}, \qquad i > 0$$

and

$$t = \sum_{i=1}^{[b]} \frac{b(b - 1)\ldots(b - i + 1)}{(a + b - 1)(a + b - 2)\ldots(a + b - i)} \cdot \frac{1}{(1 - x)^i}$$

where $b$ is equal to the largest integer less than b. If $b = 0$, then $t = 0$, and r is defined as

$r = 1$ , if b is an integer

$r = b - [b]$, otherwise.

Since MDBETI uses an iterative searching method, a lot of computation time is required to find a deviate. Even if MDBETI has the convergence criteria for which $|x^* - x|$ is not greater than $1 E - 5$ and $|I_x(a,b) - p|$ is not greater than $1 \cdot E - 4$, where $x^*$ is defined by $I_x^*(a,b) = p$, the absolute error is greater than $1 \cdot E - 5$ at

the tail of the beta cumulative distribution function. This means that the limit of 30 iterations was reached before the convergence criteria were met.

TABLE I

MAXIMUM ERROR FROM MDBETI

| a | b | Input Percentage Point | Probability | Output Percentage Point | Absolute Error |
|---|---|---|---|---|---|
| 1 | 1 | .97 | .97000 | .96999 | .00001 |
| 20 | 20 | .71 | .99718 | .70992 | .00008 |
| 50 | 50 | .66 | .99948 | .69996 | .00004 |
| 50 | 40 | .69 | .99607 | .68999 | .00001 |
| 100 | 100 | .64 | .99997 | .63998 | .00002 |
| 100 | 110 | .62 | .99999 | .61993 | .00007 |

The above table is the result of an error analysis on subroutine MDBETI (21, 22). For this analysis, one hundred percentage points were taken from .01, in increments of .01, up to 1.0, corresponding

to the values used in Pearson's tables (38). Given a set of combinations of integer parameter values of a and b from 1 up to 110, each of the probability values were computed by a beta probability function subroutine, denoted by FBETA (21). Then the computed values of probability were taken as input data for the inverse beta distribution function subroutine, MDBETI, and values of percentage points (random deviates) were computed. The difference in input and output values of the percentage point were considered to be the error.

A series expansion of the incomplete beta function ratio is used for development of the FBETA function subroutine from Harter (15).

$$I_x(a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\,\Gamma(b)} = \int_0^x u^{a-1}\,(1-u)^{b-1}\,du$$

$$= 1 - (1-x)^b \sum_{i=0}^{b-1} \binom{b-1+i}{i} x^i$$

which is derived by the substitution of a hypergeometric function into the incomplete beta function ratio. When a is a positive integer, the equation can be put in the form

$$I_x(a,b) = 1 - (1-x)^b \left[ 1 + bx + \frac{(b+1)!}{2\,(b-1)!}\,x^2 + \right.$$

$$\left. \frac{(b+2)!}{3!\,(b-1)!}\,x^3 + \frac{(b+3)!}{4!\,(b-1)!}\,x^4 + \ldots + \right.$$

$$\frac{(b-2+1)!}{(i-1)\ (b-1)!}\ x^{i-1} + \frac{(b-1+i)!}{i!\,(b-1)!}\ x^{i} +$$

$$\frac{(b+1)!}{(i+1)!\ (b-1)!}\ x^{i+1} + \ldots + \frac{(b-2+a)!}{(a-1)!\ (b-1)!}\ x^{a-1}\Bigg]$$

For simplification of the computational procedure in the computer program, the factor

$$\frac{(b+i-1)!\ x^{i}}{i!\ (b-1)!} \div \frac{(b+i-2)!}{(i-1)!\ (b-1)!} = \frac{(b+i=1)\ x}{i}$$

is multiplied by the (i - 1)-th term to get the value of the i-th term which is added to the sum of the preceding i - 1 terms, i = 2,3, ... , a-1. Underflow or overflow problems are prevented by using logarithmic and exponential transformations in the FBETA function subroutine. The result of the computations in double precision was compared with Pearson's tables (38), agreement occurring within seven digits after the decimal point.

An Approximation to Inverse Beta Distribution Function

Because of difficulty in the inverse transformation of beta cumulative distribution function, use was made of an approximation to the inverse beta distribution function obtained from the National Bureau of Standards (1) which uses Hasting's approximation to the normal percentage point $y_p$ (16), where

$$p = \frac{1}{\sqrt{2\pi}} \int_{y_p}^{\infty} e^{-1/2u^2} \, du,$$

$$y_p = t - \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}$$

$$t = \sqrt{\ln (1/p^2)} \qquad \text{for} \qquad 0 < p \leq .5$$

$$t = \sqrt{\ln \left[1/(1 - p)^2\right]} \qquad \text{for} \qquad .5 < p < 1.0$$

$c_0 = 2.515517$ $\qquad$ $d_1 = 1.432788$

$c_1 = .802853$ $\qquad$ $d_2 = .189269$

$c_2 = .010328$ $\qquad$ $d_3 = .001308$

The value of the percentage point $x_p$ of an incomplete inverse beta distribution function such that $I_{x_p}(a,b) = p$ is

$$x_p = \frac{a}{a + be^{2w}}$$

$$w = \frac{y_p (h + 1)^{1/2}}{h} - \left(\frac{1}{2b - 1} \quad \frac{1}{2a - 1}\right)\left(s + \frac{5}{6} - \frac{2}{3h}\right)$$

$$h = 2\left(\frac{1}{2a - 1} + \frac{1}{2b - 1}\right)^{-1}$$

$$s = \frac{y_p^2 - 3}{6}$$

This function computes a deviate relatively fast. But the function gives too crude an approximation to use. An error analysis also performed by the same procedure used for MDBETI.

TABLE II

MAXIMUM ERROR FROM APPROXIMATION FUNCTION

| a | b | Input Percentage Point | Probability | Output Percentage Point | Absolute Error |
|---|---|---|---|---|---|
| 1 | 1 | .59 | .59000 | .46496 | .12504 |
| 20 | 20 | .52 | .59928 | .49421 | .02579 |
| 50 | 50 | .50 | .50000 | .48246 | .01754 |
| 50 | 40 | .56 | .53061 | .46727 | .09273 |
| 100 | 100 | .50 | .50000 | .48761 | .01239 |
| 100 | 110 | .47 | .42964 | .50436 | .03436 |

Ahrens-Dieter Methods (Beta)

Ahrens and Dieter (2) developed a technique, involving a rejection method from a normal distribution, which generates beta deviates. This method applied the inequality

$$(X/A)^A\ ((1 - x)/B)^B\ C^C \leq \exp\ (-2C(x - A/C)^2)$$

where $A = a - 1$, $B = b - 1$, $C = A + B = a + b - 2$, $a > 1$, $b > 1$ and $0 \leq x \leq 1$. The left hand side of the above inequality is proportional to the beta (a,b) density. The right one is proportional to the density of a normal distribution with mean $m = A/C$ and standard deviation $s = 1/(2\sqrt{C})$. The resulting rejection algorithm reads as following:

1) Set $A = a - 1$, $B = b - 1$, $C = A + B$, $L = C \ln C$, $m = A/C$ and $s = .5/\sqrt{C}$.
2) Take a sample t from the standard normal distribution and compute $x = t \cdot s + m$.
3) If $x < 0$ or $x > 1$ go to step 2.
4) Generate u from uniform distribution.
   If $\ln u > A \ln (x/A) + B \ln ((1 - x)/B) + L + .5\ s^2$ ($u \leq f(x)/g(x)$, where f(x) is beta, g(x) is normal distribution) then reject x and go to step 2. Otherwise accept x as a beta deviate.

The expected number E of trials (step 2) per complete sample works out to

$$E = (1/2\pi/C)^{1/2}\ A^A\ B^B\ \Gamma(a + b)/(C^C\ \Gamma(a)\ \Gamma(b))$$

$$1/2(A + B)/(A\ B)^{1/2}$$

The approximation is based on Stirling's formula and is good except for small a and b. It follows that the method attains maximum efficiency in the symmetrical case a = b. For the symmetric cases the computation time is bounded, but the other cases the time increases as one of the beta parameters is increased. This method is relatively faster than the methods mentioned previously, but it shows an inefficiency in the cases of nonsymmetric beta density because the number of rejection, which falls outside of domain of beta density function, is increased. This technique is definitely inadequate to sample deviates which represent a high or low probability of component reliability.

TABLE III

COMPUTING TIME FOR AHRENS-DIETER METHOD (BETA)

| b \ a | 1 | 2 | 5 | 10 | 20 | 50 | 100 |
|---|---|---|---|---|---|---|---|
| 1 | - | 752 | 939 | 1197 | 1581 | 2319 | 3108 |
| 2 | | 617 | 633 | 693 | 809 | 1108 | 1391 |
| 5 | | | 597 | 612 | 677 | 860 | 1024 |
| 10 | | | | 576 | 600 | 710 | 859 |
| 20 | | | | | 568 | 612 | 739 |
| 50 | | | | | | 596 | 614 |
| 100 | | | | | | | 593 |

Computation is performed to sample 1000 deviates by CDC 6400.
Time unit is micro-second.

## Gamma Random Variable

A gamma random variable Y has a probability density function

$$f(y) = \frac{1}{\Gamma(a)\, b^a}\, y^{a-1}\, e^{-y/b}, \qquad 0 \leq y.$$

This gamma probability density function has mean at ab. The variance is $ab^2$ and the mode is (a - 1)b, where a 1. The standard form of the distribution is obtained by putting b = 1, giving

$$g(y) = \frac{y^{a-1}\, e^{-y}}{\Gamma(a)}, \qquad 0 \leq y.$$

If the random variable X has a gamma distribution with scale parameter a, and also the random variable Y has a gamma distribution with unit scale parameter and shape parameter a, then the variable relationship is X = b Y.

The chi-square random variable Z may be obtained using the variable relationship Z = 2X/b with r degrees of freedom, where a = r/2. The gamma distribution provides a wide variety of functional shapes. The gamma probability density function has the cumulative distribution function

$$F(y) = \int_0^y \frac{u^{a-1}\, e^{-u}}{\Gamma(a)}\, du, \qquad 0 \leq y$$

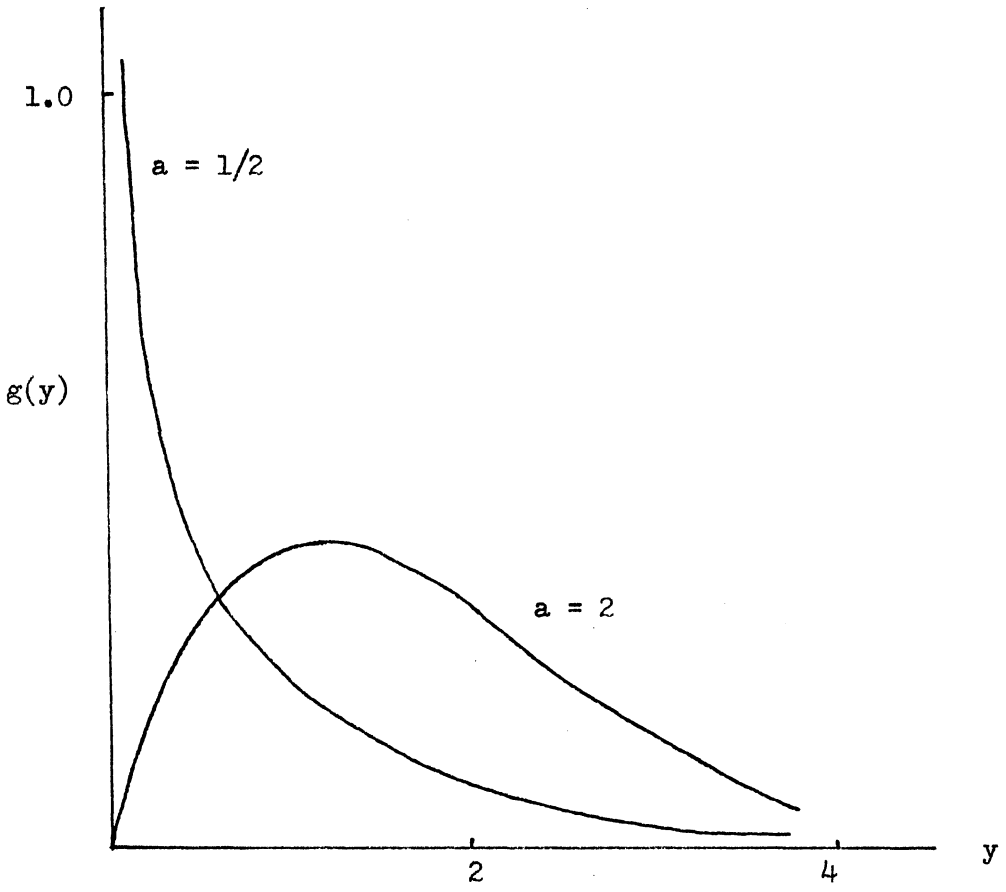



Figure 6. Gamma Probability Density Functions

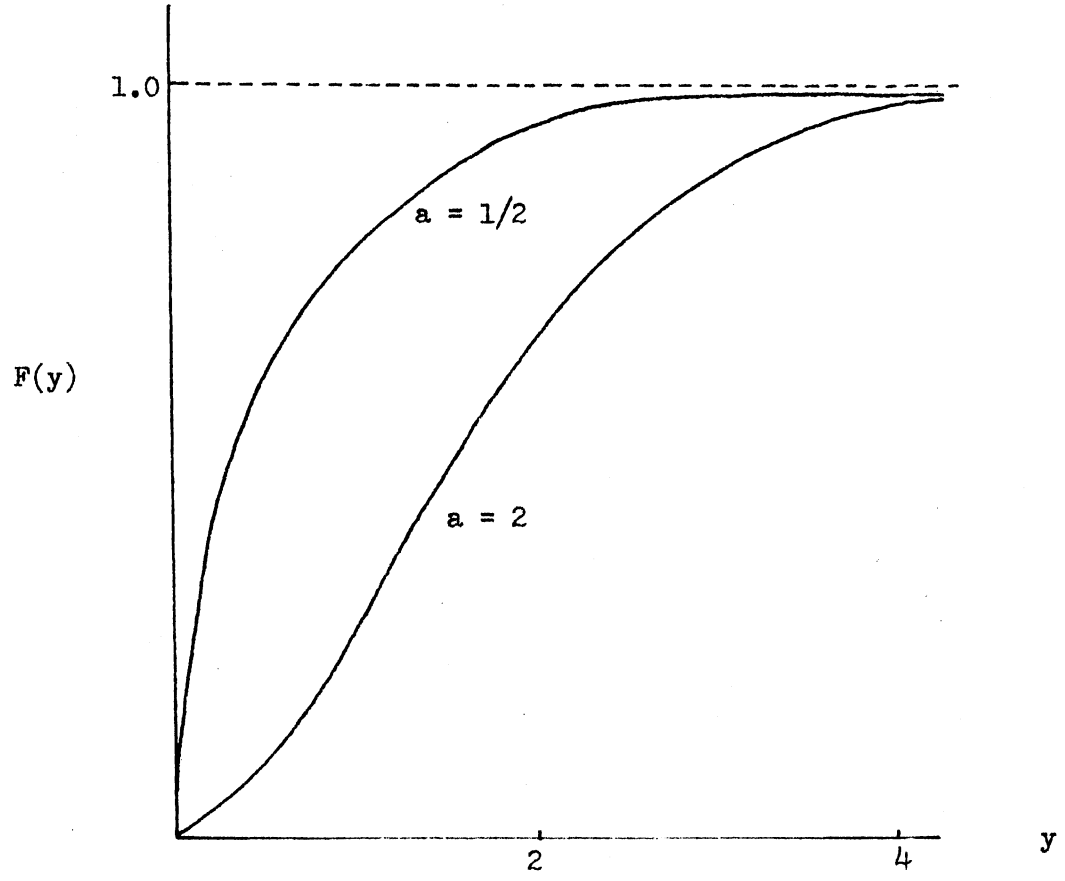



Figure 7. Gamma Cumulative Distribution Functions

## Methods for Generating Gamma Deviate

Four different previous methods from Naylor (36), International Mathematics and Statistical Libraries (7), Greenwood (13), and Ahrens-Dieter (2) are available for analysis.

Naylor's Method

We already know that if the random variables $V_i$ have independent exponential distributions with common parameter u, where $i = 1,2, \ldots , a$, then the sum of the independent exponential random variables is a gamma random variable when a is an integer. Let V have an exponential probability density function

$$f(v) = \frac{1}{b} e^{-v/b}, \qquad 0 \leq v$$

and distribution function

$$F(v) = \frac{1}{b} \int_0^v e^{-u/b} du = 1 - e^{-v/b}$$

Then we use the uniform random variable U and set

$$U = F(v) = 1 - \log(1 - U).$$

Since U is a uniform variable, it is easily seen that 1 - U is also a uniform variable. Hence we may save a step by obtaining an exponential deviate as

$$v_i = -b \log(u_i)$$

A random minimization method (Composition Method) which also can compute an exponentially distributed variable without using the logarithm subroutine is due to Von Neumann, George Marsaglia, and Knuth (20, 27, 44). The idea is based on the fact that if $c = 1/(e - 1)$

and if the random variable n takes value 1, 2, 3, ... with probabilities c, c/2 , c/3 , ... and if independently the random variable m takes values 0, 1, 2, ... with probabilities $1/(ce)$, $1/(ce^2)$, $1/(ce^3)$, ... , then the random variable

$$X = m + \min (u_1, u_2, \ldots , u_n)$$

has the exponential distribution

$$P (X \leq a) = 1 - e^{-a} , \qquad 0 \leq a$$

The computational algorithm is

1) Compute $p_j = 1 - e^j$ and

$$Q_j = (e - 1)^{-1} (1 + \frac{1}{2!} + \ldots + \frac{1}{j!})$$

where j is extended until the values are equal to the largest fraction which can be held in a computer word.

2) Set j = 1, and generate independent uniform deviates, $u_1$ and $u_2$, and set $x = u_2$.

3) If $u_1 < Q_j$, then go to step 4.

4) Set $j = j + 1$, and generate a uniform deviate, $u_j$ ; and if $x > u_j$, set $x = u_j$. Go to step 3.

5) Generate a new uniform deviate, $u_2$, and set j = 1.

6) If $u_3 < P_j$, then accept x as an exponential deviate. Otherwise, set $j = j + 1$, $x = x + 1$ and go to step 5.

This algorithm is reasonably fast; on the average only 2.582 uniform deviates are calculated per single exponential deviate.

It may well be faster than calculating the logarithm of a single uniform deviate.

When the exponential deviates are available, the gamma deviate is

$$y = \sum_{i=1}^{a} v_i = -\frac{1}{b} \sum_{i=1}^{K} \log u_i = -\frac{1}{b} \log \prod_{i=1}^{a} u_i .$$

This technique is exact mathematically, but it is valid only for obtaining random deviates from gamma distributions with an integer shape parameter. As the magnitude of the shape parameter gets larger, not only does the computational time get larger in proportion but also the many successive multiplication of uniform deviates can cause an underflow problem (13).

Function Subroutine GGTMAJ from IMSL

This method is for generating gamma deviate over integral and non-integral values of the gamma shape parameter a. It involves three techniques; one generates a deviate from the beta distribution, the second generates a deviate from the exponential distribution, then the third uses the first and second to generate a gamma deviate.

Suppose that Y is chosen from a gamma distribution with a non-integral shape parameter. We know that the sum of two independent gamma random variables will give another gamma random variable with the sum of the two shape parameter values. Then the variate relation can be expressed as

$$Y = Q + R$$

where Q has an integer parameter, R has a fractional part of shape parameter value such as

$$d = c - [c] ,$$

where [c] is integer part of c. Let Z be an independent beta random variable with parameter d and 1 - d, and V be an independent exponential random variable with unit parameter. Then we have the variate relationship with the scale parameter b,

$$R = bZ \cdot V.$$

To see this we note that

$$f(z,v) = \frac{1}{\Gamma(d)\,\Gamma(1-d)}\, z^{d-1}\,(1-z)^{-d}\, e^{-v}, \qquad 0 \le z \le 1, \quad 0 \le v$$

$$g(r,v) = \frac{b^{-d}}{\Gamma(d)\,\Gamma(1-d)}\, r^{d-1}\,(v - \frac{r}{b})^{-d}\, e^{-v}, \qquad 0 \le r, \quad 0 \le z$$

then

$$h(r) = \int_0^{\infty} g(r,v)\, dv = \frac{b^{-d}}{(d)}\, r\, e^{-r/b}, \qquad 0 \le r$$

has a gamma probability density function with shape parameter d and scale parameter b. Consequently, the desired gamma variate with non-integral shape parameter is obtained by the variate relationship

$$Y = b(X + Z \cdot V).$$

The function subroutine GGTMAJ from IMSL generates gamma deviates as following:

1) Compute [c] as the largest integer value in the shape parameter.
2) Compute the non-integral part by d = c - [c] .
3) Generate a gamma deviate, x, using Naylor's method with integer parameter [c].
4) Generate a beta deviate z with parameter d and 1 - d using Johnk's method.
5) Generate an exponential deviate v.
6) Compute a gamma deviate y = b(x + zv).

This method, using a transformation of the gamma probability density function, is exact. However, it also requires a great deal of time to generate a deviate. The computational time is also proportional to the size of shape parameter.

The Kolmogorov goodness of fit test (8) was applied to this method before a new fast method was available. A set of 99 gamma random deviates were taken from the GGTMAJ function subroutine using a set of parameter values a and b. These gamma deviates were used as input values to compute the values of probability, which should be distributed uniformly in the interval (0,1) if the subroutine is accurate. The Kolmogorov goodness-of-fit test was used to indicate the accuracy of the subroutine by testing the null hypothesis that the distribution is uniform on the interval (0,1). The theoretical uniform distribution function is U(i) = i/99 where i = 1,2, ... , 99.

An accurate gamma probability distribution function subroutine, FGAMMA, was developed to be used in the procedure of Kolmogorov goodness-of-fit test. The gamma density function is defined by

$$h(x) = \frac{t^n x^{n-1} e^{-xt}}{\Gamma(n)} \qquad x, n, t > 0$$

$$p_r(e^{-x} \leq p) = G(p) = \int_0^p \frac{t^n(-\ln y)^{n-1} y^{t-1}}{(n)} \, dy,$$

$$0 < p \leq 1$$

and let $x = -t \ln y$, $y = e^{-x/t}$, $dy = (-1/t) e^{-x/t}$, then

$$p_r(y \leq p) = H(p) = \int_{-t \ln p}^{\infty} \frac{x^{n-1} e^{-x}}{\Gamma(n)} \, dx$$

When n is an integer, after integration and evaluation term by term

$$H(p) = p^t \left[ 1 + (-t \ln p) + \frac{(-t \ln p)^2}{2!} + \frac{(-t \ln p)^3}{3!} \right.$$

$$\left. + \ldots + \frac{(-t \ln p)^i}{i!} + \ldots + \frac{(-t \ln p)^{n-1}}{(n-1)!} \right]$$

In the FGAMMA function subroutine, the factor

$$\frac{(-t \ln p)^i}{i!} \div \frac{(-t \ln p)^{i-1}}{i - 1} = \frac{-t \ln p}{i!}$$

is multiplied by the (i - 1)-th term to obtain the value of the $i^{th}$ term which is added to the sum of preceding i - 1 terms, where

$i = 1, 2, \ldots, n - 1$. This process is repeated until all of the terms of series are exhausted. The underflow or overflow problem is prevented by using logarithmic and exponential transformations. The value of the percentage point p is transformed into $z = e^{-p\sqrt{n}}$ to check with the values in Pearson's tables. Pearson's incomplete gamma distribution function (39) is given by

$$I(u,n) = \int_0^{u\sqrt{n+1}} \frac{x^n e^{-x}}{n} dx, \qquad n > 0, \quad u > 0$$

The result of the computation in double precision was compared with Pearson's table (39). The agreement is good to seven digits after the decimal point.

TABLE IV

KOLMOGOROV TEST FOR GGTMAJ

| a | b | Maximum Absolute Difference* | Observed Significance Level |
|---|---|---|---|
| 1 | 1 | .0666 | $p > .2$ |
| 13 | 4 | .0528 | $p > .2$ |
| 5 | 29 | .0676 | $p > .2$ |
| 50 | 50 | .0595 | $p > .2$ |
| 100 | 100 | .0775 | $p > .2$ |

*Difference between empirical cumulative distribution function and uniform cumulative distribution function.

Ahrens-Dieter Method (Gamma)

Ahrens and Dieter (2) developed a fast method, using a rejection technique from normal and exponential distribution, which generates gamma deviates. A normal density function can not encase any gamma density function curve completely. No matter how the parameter of normal density function is chosen some part of the tail of the gamma density curve sticks out. A combination of a normal density which envelops the bulk of the gamma distribution, with an exponential density which covers the tail of the gamma density makes it possible to encase all of the gamma density. Let f(x), g(x), and h(x) be proportional to standard gamma, normal, and exponential probability density functions respectively.

Let

$$f(x) = (x/(a - 1))^{a-1} \exp(-(x - a + 1)), \quad 0 \le x$$

$$g(x) = \exp(-(x - (a - 1))^2 / (2(a + c\sqrt{a}))), \quad 0 \le x \le b$$

$$h(x) = (b/(a - 1))^{a-1} \exp(-(1 - (a - 1)/b)x), \quad b < x$$

where $c = \sqrt{8/3}$ and $b = a - 1 + (3/2)\, c(a + c\ a)^{1/2}$, a is the gamma shape parameter. Then $f(x) \le g(x)$ for $0 \le x \le b$ and $f(x) \le h(x)$ for $b \le x \le \infty$. The areas under g(x) and h(x) are

$$G = \int_{-\infty}^{+\infty} g(x)\,dx = (2\pi(a + c\sqrt{a}))^{1/2}$$

$$H = \int_{b}^{\infty} h(x)\,dx = (b/(a-1))^{a-1}\,(b/b - (a-1)))\,e^{-(b-(a-1))}$$

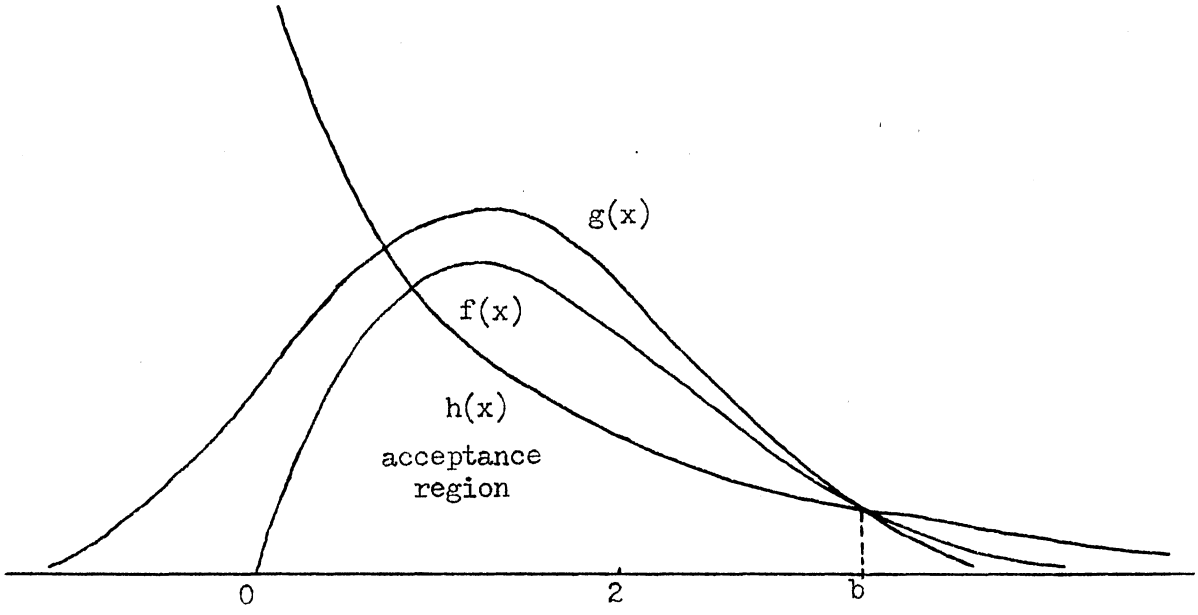
The majoring functions are $g(x)$ in $(-\infty, b]$ and $h(x) + g(x)$ in $[b, \infty)$. However, all x from $g(x)$ are discarded if they lie outside of the interval $[0,b]$.

The algorithm is

1) Generate a uniform deviate $u_1$.
2) If $u_1 \leq H/(G + H)$, then go to step 5.
3) Take a deviate s from the standard normal distribution, $x = m + vs$, $m = a - 1$, $v = \sqrt{(a + c\sqrt{a})}$. If $x < 0$ or $x > b$, then go to step 1.
4) Generate $u_2$. If $u_2 \leq f(x)/g(x)$ then accept x as a gamma deviate. Otherwise, go to step 1.
5) Take a deviate s from the standard exponential distribution and set $x = b + bs/(b - (a - 1))$.
6) $u_2 \leq f(x)/h(x)$, then accept x as a gamma deviate, otherwise go to step 1.

The steps 2, 4, and 6 have been modified only for better computation performance in the final implementation. This method works if the shape parameter value a is greater than 2.53278. An additional algorithm is required for the case of parameter value a is less than

2.53278. The computation time is given in chapter III for the purpose of performance comparision.




f(x) : Proportional to the gamma probability density function
g(x) : Proportional to the normal probability density function
h(x) : Proportional to the exponential probability density function.

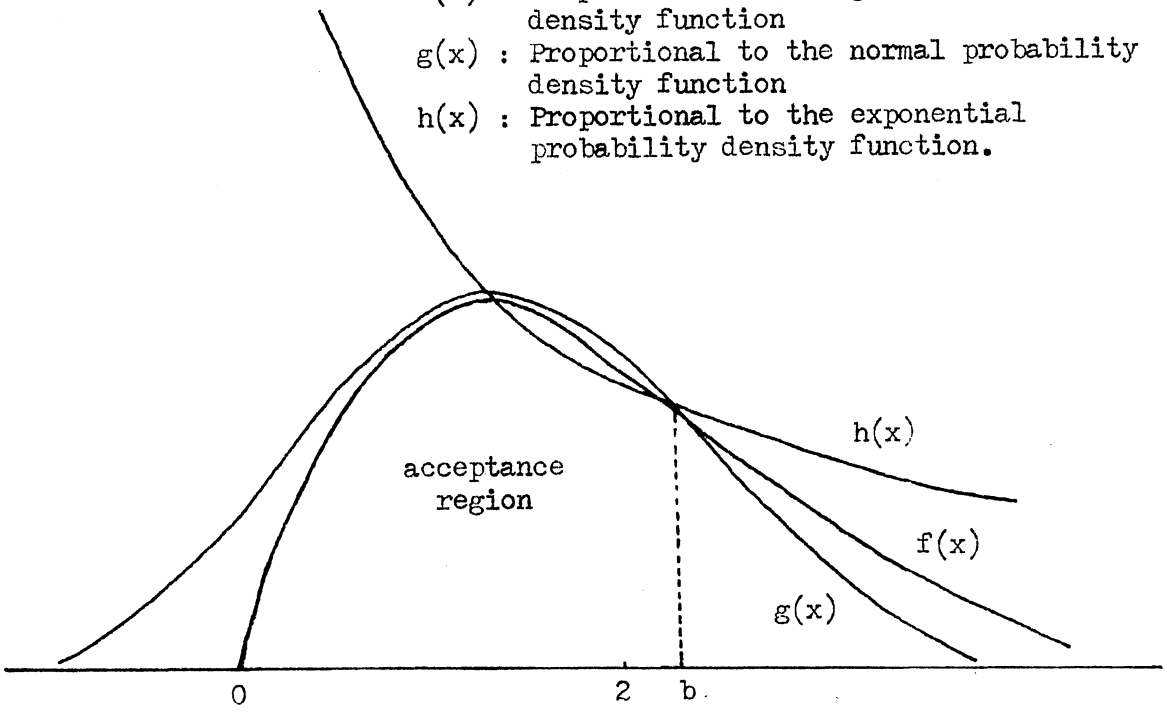



Figure 8. Ahrens-Dieter Method (Gamma)

Greenwood Method

Greenwood (13) also presented a fast rejection method for generating gamma random deviates. Using the Wilson-Hilferty transformation (18, 45), if a random variable Y has standard normal probability density, then

$$Z = d\,(1 - 2/ad + \sqrt{2/9d}\;\; Y)^3$$

has approximately a chi-squared distribution with d degrees of freedom. The standardized chi-squared distribution tends to a unit normal distribution as the number of degrees of freedom increases. Putting d = 2a, the above variate relation gives

$$H = 2a\,(1 - (9a)^{-1} + (9a)^{-1/2}\;Y)^3$$

which has approximately the gamma distribution

$$g(x) = \frac{x^{a-1}\,e^{-x}}{\Gamma(a)}, \qquad x > 0, \quad a > 0$$

A minor correction of the random variable H is given for better accuracy by Pogurova (21) as

$$F = a\,(1 - (9a)^{-1} + (9a)^{-1/2}\;Y)^3,$$

which gives values of percentile points correct to six significant figures for $a \geq 25$.

To generate an exact gamma random deviate the rejection technique is applied

1) Generate a normal deviate y.

2) Compute an approximate gamma deviate by

$$t = a\,(1 - (9a)^{-1} + (9a)^{-1/2}\,y)$$

3) Generate a uniform deviate u.

4) If $u \leq g(t)/f(t)$ then accept t as a gamma deviate. Otherwise, reject t and go to step 1.

A fast and accurate normal deviate generator is also necessary to obtain a gamma deviate with desired conditions. Greenwood used a transformation to polar coordinates.

$$V = R \cos w, \qquad Z = R \sin w.$$

Then we find that

$$S = R^2$$

$$X = (-2 \log S)^{1/2} \cos w$$

and

$$Y = (-2 \log S)^{1/2} \sin w$$

have a normal distribution. An attempt to prevent the cancellation errors from sine and cosine functions are computed by the identities

$$\text{Sin } 2w = 2 \tan w/(1 + \tan^2 w)$$

$$\text{Cos } 2w = (1 - \tan^2 w)/(1 + \tan^2 w), \qquad 0 \leq w \leq \pi .$$

A prevention of discretization error is also attempted in the computation of exponential deviate E = -log u by checking the range of E

such as

$$v \log 10 - \log 2 < E < v \log 10,$$

where

$$10^{-v} < u < 2 \cdot 10^{-v} \qquad s = 8.$$

Then an exponential random deviate is generated as following:

1) Compute c = log (1000) which is 6.9077... .
2) Generate a uniform deviate u.
3) If u > 1/1000 then compute an exponential deviate E = -log u.
4) Otherwise, compute E = -log u.
5) If E + log (1000) is equal to 0, then obtain E as an exponential deviate, otherwise, go back to step 2.

This method for obtaining exponential deviates may show a lack of goodness-of-fit or randomness because of step 5.

Greenwood's direct rejection method is fairly fast. But there is a better and faster method (28) than this technique. The performance comparison is discussed in chapter III.

# CHAPTER III

## FAST METHODS FOR GENERATING BETA, GAMMA, AND NORMAL DEVIATES

Most of the previous methods for generating beta or gamma deviates are inefficient in computation time as the size of a parameter value gets large. Some of the methods restrict their functions or accuracy in certain range of parameter value. The accurate representation of a distribution to be assumed in statistical simulation model is very important in computer simulation. The speed of generating the random deviates, which is derectly related to the cost of computing time, is also crucial where a large scale simulation model requires a large sample. Therefore new techniques are introduced which can overcome the inefficiencies of the previous methods.

### Beta Random Deviate Generator

A new and efficient method for generating beta deviates has been devised by J.P. Chandler and the author. Basically, the rejection method is applied. Several modification are added to reduce the number of rejection and to decrease the computing time. We generalize the rejection method that was mentioned earlier, as follows.
Let $f(x)$ be a continuous probability density function. Let $h(x)$ be another continuous probability density function such that

$$f(x) \leq s \cdot h(x) = g(x)$$

for some constant s 1. Then a random deviate from pdf f(x) is generated as follows:

1) Take a random deviate x from pdf h(x).

2) Generate a uniform random deviate, u.

3) If $u \leq f(x)/g(x)$ then accept x as a random deviate from f(x). Otherwise, reject x and go back to step 1.

Mathematically we can show that

$$P_r\left(X \leq x \;\middle|\; U \leq \frac{f(X)}{g(X)}\right) = F(x).$$

Proof:

$$P_r\left(X \leq x \;\middle|\; U \leq \frac{f(X)}{g(X)}\right)$$

$$= \frac{P_r\left(X \leq x \text{ and } U \leq \frac{f(X)}{g(X)}\right)}{P_r\left(U \leq \frac{f(X)}{g(X)}\right)}$$

$$= \frac{\displaystyle\int_{-\infty}^{+\infty}\left(\int_0^{\frac{f(x)}{g(x)}} 1 \cdot du\right) h(x)\,dx}{\displaystyle\int_{-\infty}^{+\infty}\left(\int_0^{\frac{f(x)}{g(x)}} 1 \cdot du\right) h(x)\,dx}$$

$$= \frac{\displaystyle\int_{-\infty}^{x} \frac{f(x)}{g(x)}\, h(x)\,dx}{\displaystyle\int_{-\infty}^{+\infty} \frac{f(x)}{g(x)}\, h(x)\,dx}$$

$$= \frac{\int_{-\infty}^{x} \frac{f(x)}{s}\, dx}{\int_{-\infty}^{+\infty} \frac{f(x)}{s}\, dx}$$

$$= \int_{-\infty}^{x} f(x)\, dx = F(x).$$

In our procedure the function $g(x)$ is chosen proportional to a Cauchy probability density function. The position of the mode of the function $g(x)$ is adjusted to the position of the mode of the beta pdf $f(x)$ to reduce the number of rejections.

A Cauchy random variable has the probability density function

$$h(x) = \frac{s}{\pi(s^2 + x^2)}, \qquad s > 0 \qquad -\infty < x < \infty.$$

The cumulative distribution function is

$$H(x) = \int_{-\infty}^{x} \frac{s}{\pi(s^2 + u^2)}\, du = 1/2 + {}^{-1} \tan^{-1} \left(\frac{x}{s}\right).$$

Using the inverse transformation, a Cauchy random deviate can easily obtained by

$$X = s \cdot \tan(T),$$

where T is a uniform random variable in $(-\pi/2, \pi/2)$.

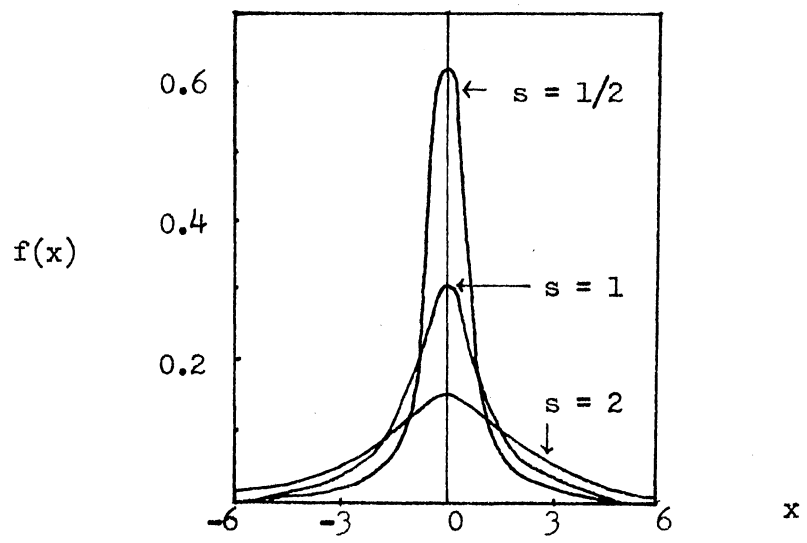



Figure 9. Cauchy Probability Density Function

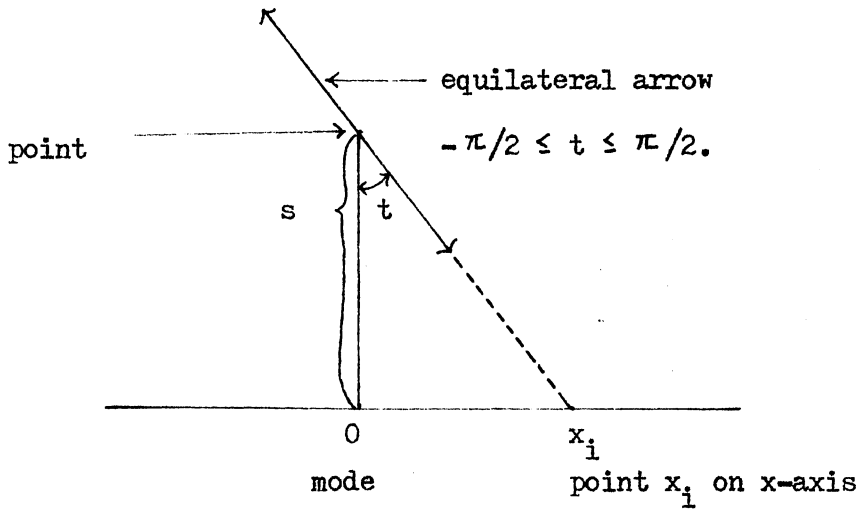



Figure 10. Cauchy Distribution Generator*

* The arrow is spun so that the angle t has uniform density. The Cauchy probability density function refers to the random variable X, representing the coordinates of the points to the arrow leads (33).

The function g(x) proportional to Cauchy pdf is

$$g(x) = c \cdot \frac{s}{\pi(s^2 + x^2)}, \qquad s > 0, \quad c \geq 1, \quad -\infty < x < \infty .$$

The proportionate constant c and the scale parameter s are chosen by solving two equations

$$f(m) \cdot 1.1 = g(m)$$

and

$$f''(m) = 1.1\, g''(m),$$

where m is mode of f(x), so that the height of the function g(x) is 10% greater than the height of the beta pdf and the second derivative (curvature) of g(x) is 10% smaller at the mode of the beta pdf. Then g(x) can encase f(x) completely with a possible small differences for all x in (0,1) when the modes of f(x) and g(x) are fixed at a point. To prevent a Cauchy deviate which falls outside of the interval (0,1), the relation $t = \tan^{-1}(x/s)$ is applied (5).

This new method is implemented as following:

1) Compute the mode of the beta pdf with the values of parameters a and b.
2) Compute the range of t where the Cauchy deviates are to be generated.
3) Generate a Cauchy deviate t from h(x).
4) Generate a uniform deviate u.
5) If $u \leq f(t)/g(t)$ then accept t as a beta deviate. Otherwise, reject t and go back to step 3.

To increase the computational efficiency the constant part of the beta pdf is computed and stored to be used repeatedly until the values of

the beta parameters are changed. Step 5 is performed in a logarithmic transformation to prevent underflow or overflow problems. The computation time required remains bounded and in fact small, as the values of the beta parameter get larger. A subroutine named CBETA is programmed in PL/1 and used in SPARCS (see APPENDIX B).

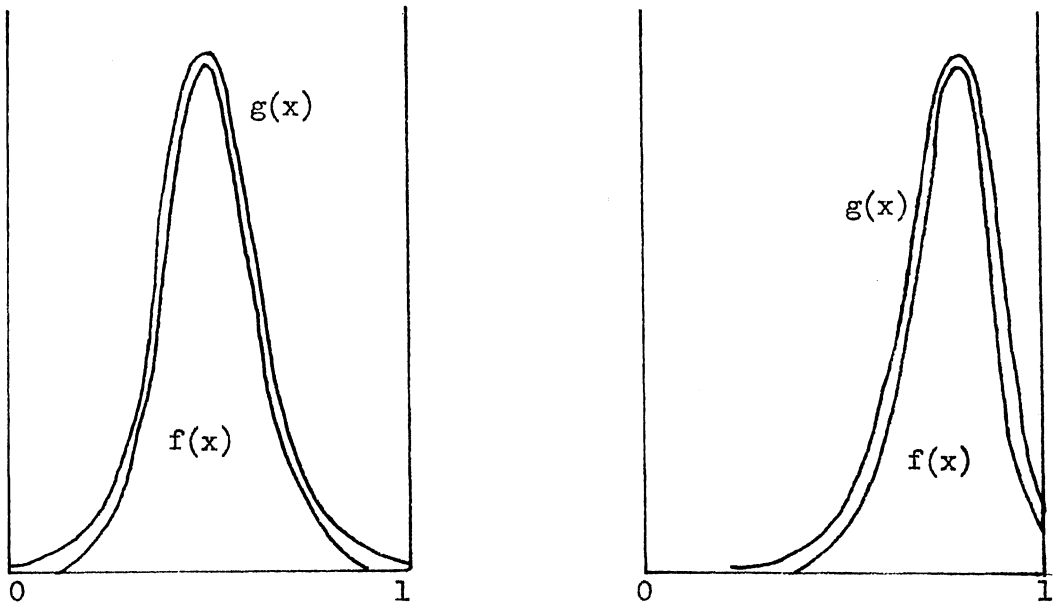



Figure 11. New Method (beta)

Random samples with 100 observations corresponding to each parameter value were taken for the kolmogorov goodness-of-fit test and estimation of average computation time.

TABLE V

KOLMOGOROV TEST FOR CBETA

| a | b | Maximum Absolute Difference | Observed Significance Level |
|---|---|---|---|
| 100 | 2 | .098076 | p > .2 |
| 100 | 5 | .038107 | p > .2 |
| 100 | 8 | .057301 | p > .2 |
| 100 | 10 | .055841 | p > .2 |
| 100 | 30 | .049207 | p > .2 |
| 100 | 50 | .094918 | p > .2 |
| 100 | 80 | .071702 | p > .2 |
| 100 | 100 | .088941 | p > .2 |

TABLE VI

COMPUTING TIME FOR CBETA

| a | b | Average Number of Rejections per Beta Deviate |
|---|---|---|
| 100 | 2 | 1.71 |
| 100 | 5 | 2.34 |
| 100 | 8 | 2.36 |
| 100 | 10 | 2.51 |
| 100 | 30 | 2.81 |
| 100 | 50 | 2.69 |
| 100 | 80 | 3.35 |
| 100 | 100 | 2.99 |

## Gamma Random Deviate Generator

A very fast and exact method for generating gamma deviates exists, called the squeeze method devised by George Marsaglia (28). The method is based upon the rejection technique and Wilson-Hilferty transformation (45). The idea is to squeeze the target density between two functions, the top one "easy to sample from," the bottom one "easy to evaluate" in such a way

$$h(x) \leq g(x) \leq f(x).$$

The top function, f(x), is chosen to be the normal probability density function in this case. The target function g(x) is a probability density function from which we want to sample. Wilson-Hilferty transformation on g(x) gives gamma random variable. The function h(x) is chosen to be an exponential function in this case. The function h(x) is

$$h(z) = \exp\left(-1/2\, u^2 - a(z^3 - v^3) + (3a - 1)(t + t^2 + 1/3\, t^3)\right)$$

for

$$s = a^{-1/2}/3$$

$$u = s - 3$$

$$v = 1 - 3$$

$$t = 1 - v/z$$

a is shape parameter of gamma.

The algotithm is:

1) Generate a standard normal random deviate x.

   Take the Wilson-Hilferty transformation by

$$Z = sX + 1 - s^2,$$

where $s = a^{-1/3}/3$. If $Z \leq 0$, repeat this step.

2) Generate a uniform deviate u.

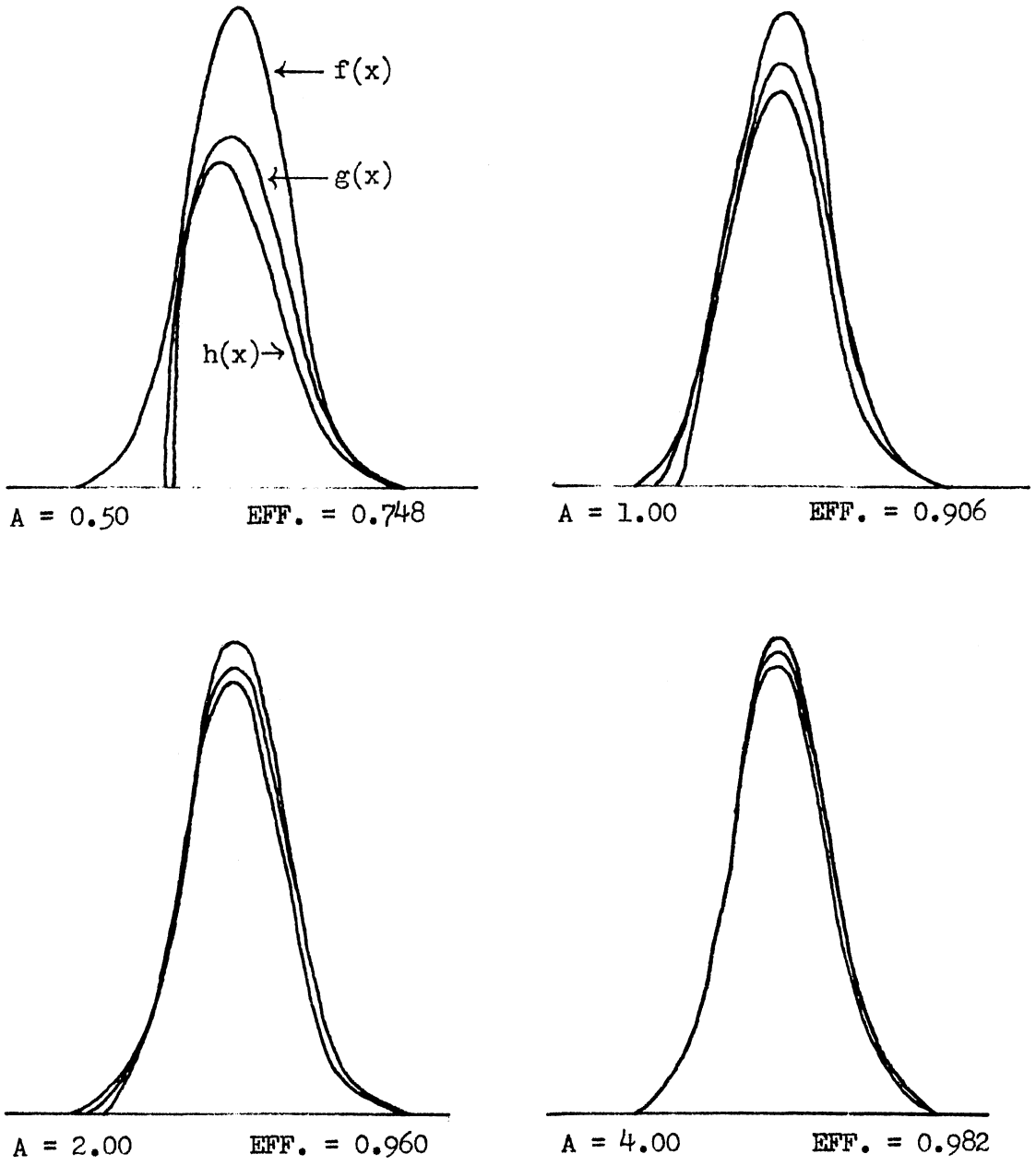
3) If $u \leq \frac{h(z)}{f(z)}$, then accept $w = a \cdot z^3$. Else go to step 4.

4) If $u \leq \frac{g(z)}{f(z)}$, then accept $w = a \cdot z^3$ as a gamma deviate,

else go to step 1.

For computational efficiency the step 2, 3, and 4 are modified in the final implementation.

The essence of the squeeze method lies in the testing procedure for a decision as to whether a normal deviate is qualified to be a gamma deviate. For this procedure the function g(z), and f(z) or h(z) are evaluated whenever an acceptance or rejection decision is made. To economize the function evaluation time for g(z), which is complicated, the function h(z), which is easy to evaluate, is inserted "under" the function g(z). The testing under g(z) is avoided most of time by first testing under h(z). The functions f(z) and h(z) are chosen to be close to g(z) so that the squeeze method can be utilized efficiently. The efficiency, (area under g(z)) / (area under f(z)), is 90% for a = 1, 98% for a = 2, and quickly convergent to 1 as "a" gets larger. Function RGAMMA is programmed in PL/1 and used in SPARCS (see APPENDIX B).




f(x) : pdf
g(x) : pdf
h(x) : exponential function

Figure 12. The Marsaglia Squeeze Method

For the Kolmogorov goodness-fit-test five samples with 100 observations (deviates) were taken corresponding to each parameter value. The parameter values chosen were 1, 5, 25, and 100.

TABLE VII

KOLMOGOROV TEST FOR RGAMMA

| a | Maximum Absolute Difference | Observed Significance Level |
|---|---|---|
| 1 | .056407 | p > .2 |
| | .058373 | p > .2 |
| | .092194 | p > .2 |
| | .104112 | p > .2 |
| | .083168 | p > .2 |
| 5 | .074911 | p > .2 |
| | .076704 | p > .2 |
| | .097174 | p > .2 |
| | .086548 | p > .2 |
| | .059757 | p > .2 |
| 25 | .065501 | p > .2 |
| | .091730 | p > .2 |
| | .079708 | p > .2 |
| | .090348 | p > .2 |
| | .063717 | p > .2 |
| 100 | .060478 | p > .2 |
| | .080398 | p > .2 |
| | .085890 | p > .2 |
| | .043725 | p > .2 |
| | .087178 | p > .2 |

Computation time is reported in microseconds and the squeeze method is compared with Ahrens-Dieter and Greenwood methods in TABLE VIII.

TABLE VIII

COMPARISION OF AVERAGE TIME TO GENERATE GAMMA DEVIATE

| Method | a=1.001 | a=2.533 | a=4 | a=10 | a=100 | a=1000 |
|---|---|---|---|---|---|---|
| Ahren-Dieter | --- | 313 | 293 | 256 | 203 | 193 |
| Greenwood | 258 | 251 | 255 | 230 | 227 | 232 |
| Squeeze Method | 210 | 197 | 191 | 187 | 195 | 183 |

Time unit is one micro-second.

## Normal Random Deviate Generator

An accurate and fast normal random deviate generator is required since the gamma deviate is sampled from normal deviates. The normal distribution is one of the best known and most important probability distribution functions.

$$f(x) = \frac{1}{s\sqrt{2\pi}} \; e^{-1/2\left(\frac{x-m}{s}\right)^2}, \qquad -\infty < x < \infty ,$$

Several techniques are available for generating normal deviates such as a rejection method by transformation to polar coordinates (20), Central Limit approach (36), approximation to the inverse normal distribution (16), Marsaglia's rectangle-wedge-tail method (30), Teichrow's approximation method (42), and others.

For the purpose of optimization between the speed of computing time and memory space requirement, an exact method by Marsaglia and Bary (31) is adopted. They developed a normal random deviate generator by a composite of a decomposition approach (46), probability density function transformation to polar coordinates, and rejection method. A normal random deviate is generated in terms of uniform random variable in such a way: 86 percent of the time, put $X = 2(u_1 + u_2 - 1.5)$, 11 percent of the time, put $X = 1.5\ (u_1 + u_2 - 1)$ and the remaining 3 percent of the time, use a more complicated procedure so that the resulting mixture is correct. In other words, the first two part are composed of simple equations which are computed easily most of the time. The last part is quite complicated, but it rarely needs to be computed. For the area of about 2.55 percent which shares the tails and residue of the normal density function, two techniques are applied to generate

the normal deviates. For $|x| > 3$, a modified polar method is used and a rejection technique used for $|x| < 3$.

Suppose we have random variables $X_1, X_2, \ldots, X_n$, and a sequence $a_1, a_2, \ldots, a_n$ of positive constants satisfying

$$\sum_{n=1} a_n = 1.$$

Then a new random variable Y, which is a mixture of $X_1, X_2, \ldots, X_n$, can be obtained by equating Y to $X_n$. If $X_n$ has a distribution function $F_n(x)$, the distribution function for Y is

$$F(y) = \sum_{n} a_n F_n(x)$$

since

$$F(y) = p_r(Y \leq y) = \sum_{n} a_n p_r(X_n \leq x).$$

If $X_n$ has the probability density function $f_n(x)$ then Y has the probability density function

$$f(y) = \sum_{n} a_n f_n(x).$$

Suppose we are given f(y). If a probability density function g(y) and a positive number a a are given, for all y, then

$$f(y) - a\, g(y) \geq 0.$$

It follows that

$$1 - a = \int_{-\infty}^{\infty} ( f(y) - a\, g(y) )\, dy > 0,$$

where

$$g(y) \neq f(y), \qquad 1 - a > 0, \qquad 0 < a < 1.$$

Now a probability density function h is defined by

$$h(y) = \frac{f(y) - a \cdot g(y)}{1 - a}$$

Then

$$f(y) = a \cdot g(y) + (1 - a)\, h(y)$$

is a decomposition of the random variable Y as a mixture.

Let $X_1 = 2\,(U_1 + U_2 + U_3 - 1.5)$, where $U_1$, $U_2$, $U_3$ are independently distributed U(0,1). The density function for $X_i$ is

$$f_1(x) = \begin{cases} (3 - x^2)/8\,, & -1 \le x \le 1 \\ (3 - x)^2/16, & 1 \le |x| \le 3 \\ 0\,, & 3 \le |x|. \end{cases}$$

Finding $a_1$ is done by calculating the point at which the quotient $f(x) / f_1(x)$ is minimized. Zeros of the derivative occur at x = 0, 1, 2. The minimum value of the ratio occurs for $x = \pm 2$.

Thus

$$a_1 = \frac{f(2)}{f_1(2)} = \frac{16\, e^2}{\sqrt{2\,\pi}} \simeq .8638554642.$$

The residual density function is well approximated by

$$f_2(x) = \begin{cases} (6 - 4x)/9, & |x| \le 1.5 \\ 0\,, & |x| > 1.5 \end{cases}$$

which is the density function of

$$X_2 = 1.5\ (U_1 + U_2 - 1).$$

It can be found that the ratio

$$\frac{f(x) - a_1\ f_1(x)}{f_2(x)}$$

has a minimum value of $a_2 = .110817965$ occurring at $x = .87386312884$.

So far deviates are obtained from the portion of normal curve between $x = -3$ and $x = 3$. But there is a residual error because $f(x) - a_1\ f_1(x) - a_2\ f_2(x)$ is not zero in the interval $[-3,3]$. Before dealing with this error, we deal with the tail portions of $f(x)$ for which $|x| \geq 3$.

The probability that $|X| \geq 3$ is

$$a_3 = 2 \int_{-\infty}^{-3} f(x)\ dx = .002699796063.$$

The sampling for this portion, $f_3(x)$, is done by a rejection method which results from a probability density function transformation to polar coordinates.

Returning to the residual function $f_4(x)$, we have

$$f_4(x) = \frac{f(x) - a_1\ f_1(x) - a_2\ f_2(x) - a_3\ f_3(x)}{1 - a_1 - a_2 - a_3}$$

which is all that is left to make this technique exact.

Then

$$a_4 = 1 - a_1 - 1 - a_2 - a_3 \cong .02262677245.$$

The function $f_4(x)$ has a maximum value of approximately .3181471173. Therefore, if

$$U_1 = 6\ U - 3$$

and

$$V_1 = .3181471173\ V,$$

Then $(U_1\ ,\ V_1)$ is uniformly distributed over the rectangle enclosing the relevant portions of $f_4(x)$.

Then to generate a standard normal deviate x, we take following steps:

1) Generate a uniform deviate $u_1$.

2) If $u_1 > .8638$ then $x = 2\ (u_2 + u_3 + u_4 - 1.5)$.

3) If $.8638 > u_1 > .1107$ then $x = 1.5\ (u_2 + u_3 - 1)$.

4) If $.1107 > u_1 > .0228$ then take x and y by

$$x = 6\ u_2, \qquad y = .358\ u_3.$$

If $y < f_4(x)$ then x is normal deviate, else do this step repeatdly until one deviate is obtained.

5) If $u_1 < .0026997$ then use a rejection method by a modified probability function transformation to polar coordinates.

$$x = v((9 - 2\ \ln(v^2 + w^2))/(v^2 + w^2))^{1/2}$$

$$y = w((9 - 2\ \ln(v^2 + w^2))/(v^2 + w^2))^{1/2}$$

If either x or y is greater than 3 then take either x or y as a normal deviate. v, w are uniform over $[-1\ ,\ 1]$ , conditioned by $v^2 + w^2 < 1$.

$$\sin(t) = w \Big/ \sqrt{v^2 + w^2}\ , \qquad -2\pi < t < 2\pi$$

$$\cos(t) = v \Big/ \sqrt{v^2 + w^2}.$$

The step 2 and 3 are important ones for speed. The step 4 and 5 are important ones for accuracy. The efficiency of step 4 is 47 percent, that is the probability that a point (x,y) chosen uniformly from the rectangle will lie under the curve $f_4(x)$ is 47 percent. The function $f_4(x)$ is the residual density. Step 5 for the tail portion of normal density occurs about 1 time in 400.

If the time needed to carry out decomposed step i is $t_i$ then the expected amount of time to produce a single deviate is given by

$$E = t_0 + \sum_{i=1}^{n} a_i \, t_i$$

where $t_0$ is the time needed to generate the discrete distribution of i.

The function subroutine GAUSF was programmed in PL/1 and used in SPARCS (see APPENDIX B).

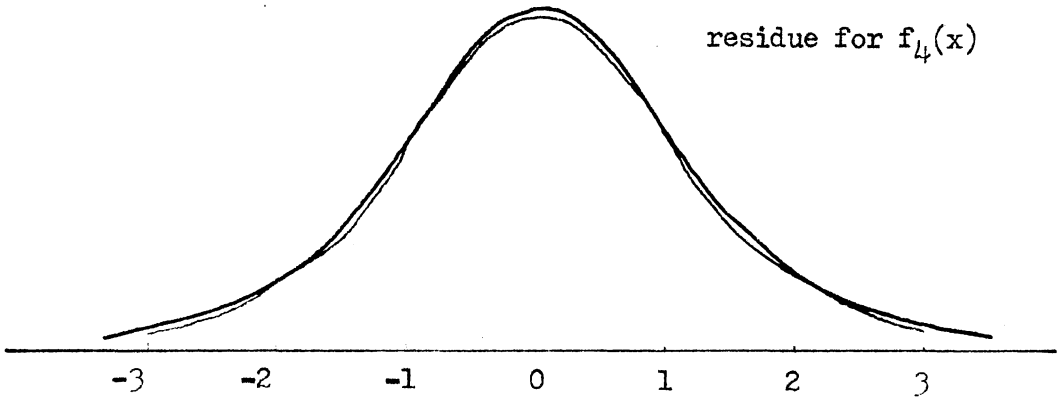



Figure 13. Polynomial Fit to Normal Density

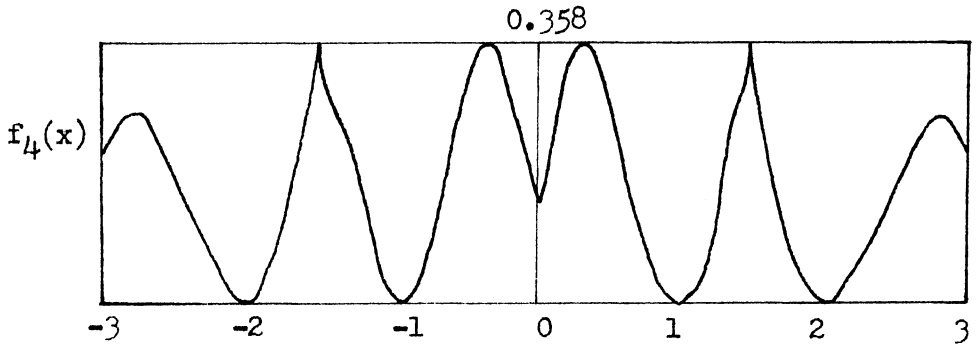



Figure 14. Residual Function

## Uniform Random Deviate Generator

A good uniform random deviate generator is the most important foundation for generating other statistical random variables in the computer. Uniform random deviate is a random fraction uniformly distributed between zero and one. A set of uniform deviate from a random deviate generator has to have the statistical properties of randomness and uniformity. For a digital computer it is most conve nient to calculate a sequence of numbers one at a time as required, by a completely specified rule which is, however, so devised that no reasonable statistical test will detect any significant departure from randomness. This sequence is often called pseudo-random number (deviate).

Numerous random number generators have been proposed for digital computers. A congruential method is often applied to generate the uniform random deviate. The congruential relation is

$$X_i = a\, X_{i-1} \pmod{m} + c$$

where $m = p^e$ and a is a constant. The random number generator is often called multiplicative congruential if $c = 0$ and mixed congruential if $c \neq 0$. m represents the word size of the computer and e is the number of digits in a word. For a binary computer p is 2. There are two reasons for choosing $m = p^e$. First, reduction modulo m is accomplished by truncating and retaining only the low order e digits; and second, conversion to the unit interval [0,1] involves only moving the binary point to the left of the number.

Marsaglia (29) gave a cautionary remark that all congruential methods have a nonrandom characteristic in a few applications.

Then Maclaren and Marsaglia (26) proposed a composite method whereby the numbers from one generator are used to shuffle the numbers from a second. Suppose the first set of number $U_1$ for a congruential generator is picked at random. Then the sequence $U_1$, $U_2$, ... may be considered a sequence of random variables. Moreover, each $U_i$ will be uniform on the set of numbers in [0,1] which can be represented exactly in computer. However, the different $U_i$ are not independent, and it turns out that the distribution of an n-tuple $(U_1, \ldots, U_n)$ may be quite far from the correct distribution. To improve the distribution of n-tuples, two different type of generators are used and the sequence produced by one is shuffled by the other.

The two generators are

$$j_{i+1} = m\, j_i \quad \text{modulo } 2^{24} + 1$$

$$u_{i+1} = A_j\, U_i \quad \text{modulo } 2^{31}$$

Initially 128 odd integer values are stored in an array A. j comes from the random integer in [1,128] . Then compute the uniform deviate using the first congruential relation. A(j) is filled with an integer which ensure long periods. The uniform random deviate generator RANF (6) uses this method, essentially in the form described by Marsaglia and Bray (26). The time to generate a random deviate by this method is about twice the time required with a congruential generator.

Composite generators generally pass all known tests of randomness, even if the component generators are not of the highest possible quality. The multipliers used in RANF have been tested by Van Gelder (43) and found to pass some common tests. If RANF were coded in the simplest possible way, the least significant bit in the mantissa would almost always be zero. In order to use the numbers from RANF for

testing the roundoff properties of numerical algorithms, the least significant bit has been randomized separately (6). RANF is coded in PL/1 and used in SPARCS (see APPENDIX B).

# CHAPTER IV

## APPLICATION OF RANDOM DEVIATE GENERATORS

The random deviate generators, described in previous chapter, are used in *Confidence Assessment of System Reliability which is studied extensively by Locks (35). As a part of the continuation of the research, an improvement of computing time by replacing the random deviate generators and optimization of the SPARCS (Simulation Program for Assessing the Reliability of Complex System developed by Cooley (9)) program was attempted. A brief description of previous study and some characterization of the distribution of system reliability from the results of using the modified SPARCS program are given in this chapter.

Unfortunately, the definitions of "component" and "system" given in the literature are sometimes unclear and very among writers. For component, Gnedenko, Belyayev, and Solovyer (12) called it a unit which means an element, a system, a part of system, or the like. Mann, Schafer, and Singpurwalla (32) used it as an equipment. They defined system as a given equipment configuration. Rosenblatt (41) used subsystem for a definition of component. Miles (34) gives a few definitions on the general concept of a system. One of them is "A group of devices or artificial objects forming a network or used for a

---

* The research is granted by the Air Force System Command, Wright-Patterson Air Force Base, in Dayton, Ohio as project number F33615-74-c-4077. The result of that contract is technical report AFFDL-TR-75-144 (35).

common purpose." As we have various types of machine systems for different purpose, many different shapes of component are also used for a specific system. While some components have a unit hardware structure, another may have several basic elements in it which have a character of a subsystem or system.

A component or system can be described in terms of reliability to eliminate confusion in definitions. Component reliability is a value of probability obtained in testing it as a unit (23) whether it has the character of subsystem or system. System reliability is also a value of probability to be predicted based on the structure of components in system and the value of component reliability obtained by testing of components which form the system.

The SPARCS program is composed of the parts which are system reliability equation generation, Monte Carlo sampling of component reliability, and computation of statistical information. The mean of each prior distribution of component reliability, system reliability computed from mean component reliabilities, average system reliability resulting from Monte Carlo simulations, variance, standard deviation, estimated system mean time between failure (MTBF) based on mean component reliabilities, estimated system MTBF resulting from Monte Carlo simulations, and 11 MTBF and system reliability percentile points are the statistical information. The MTBF is computed as

$$MTBF = t / \ln (1/S),$$

where t is required mission time and S is system reliability. Each system reliability percentile point is the one of sorted reliabilities in ascending order pointed by percentage point. In other words, when

100 system reliabilities generated, then 90 percent of percentile point of system reliability is 90-th element of sorted reliabilities.

## System Reliability Equation Generation

If all of the components in a system must operate properly for the system to function, it is called a series system. A system which will operate if any one of its components performs successfully is called a parallel system. A series-parallel system has both a parallel and a series subsystem in it. Parallel-series, m-out-of-n (m success subsystem out of n subsystems), or modularized system can be formed by a proper logical combination of the basic series or parallel systems (35).

The reliability of a series system is the product of the probabilities of success of each component, if they work independently.

$$P(C_1 \cap C_2 \cap C_3) = P(C_1)\ P(C_2)\ P(C_3)$$

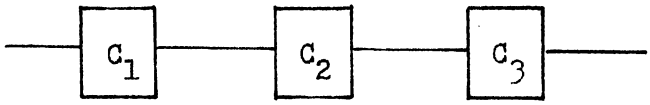



Figure 15. A Series System

The reliability of a parallel system can be obtained from the following equation if each component will work independently.

$$P(C_1 \cup C_2 \cup C_3) = P(C_1) + P(C_2) + P(C_3) - P(C_1 \cap C_2) - P(C_1 \cap C_3) - P(C_2 \cap C_3) + P(C_1 \cap C_2 \cap C_3).$$

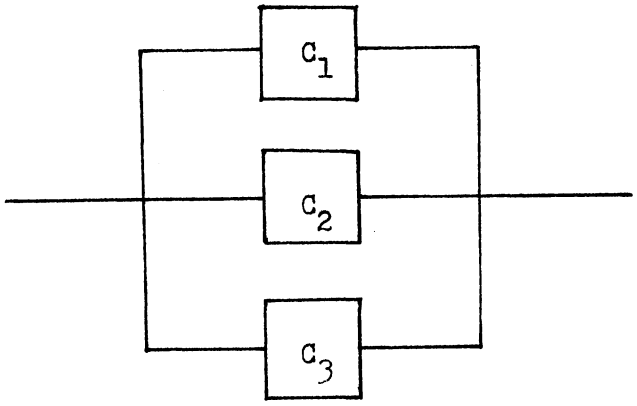



Figure 16. A Parallel System

The reliability of series-parallel, parallel-series, m-out-of-n, or modularized system can be obtained by an appropriate logical combination of the reliabilities of the basic series or parallel systems in it.

In order to apply this probability relation for efficient computa-

tion of system reliability, Poincare's Theorem (10) and a minimal state conception are introduced by Locks (25). The complete set of minimal paths and minimal cuts is called the set of minimal states. A minimal path is a unique path which does not include any other path. A minimal cut is a cut which is not included in any other cut. In other words, if the system is operating properly, at least one minimal path is effective. Therefore, the reliability of system is the probability of at least one minimal path. The unreliability is the probability of at least one minimal cut. There is a complete duality between the concept of minimal path and minimal cut. We have a simple example in the expression of minimal path in terms of binary vectors.

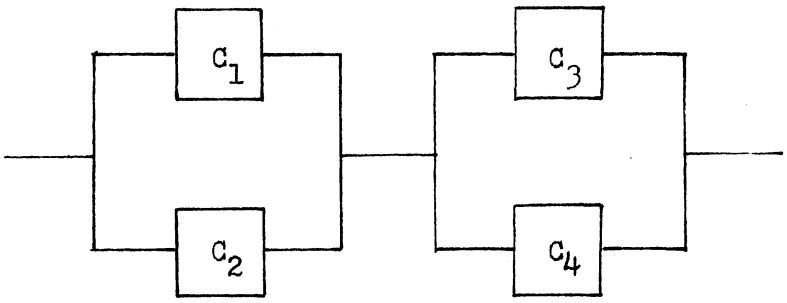



Figure 17. A Series-Parallel System

As we have 4 components in this series-parallel system, there are 4 binary elements in a vector for minimal path such as

$$C_1\ C_3\ \ldots.\ (1010)$$

$$C_1\ C_4\ \ldots.\ (1001)$$

$$C_2\ C_3\ \ldots.\ (0110)$$

$$C_2\ C_4\ \ldots.\ (0101).$$

For the convenient way of generation of terms in system reliability equation, Poincare's Theorem, which is described by Locks (24) is applied.

Suppose that the system has m minimal paths. The system reliability is the probability that the state which the system is in, characterized as a binary vector, contains at least one of these minimal paths. Let $V_i$, i = 1, ... , m, be the probability of the i-th minimal path; for example, if the components are statistically independent, $V_i$ would be the product fo the component reliabilities for those components which characterize the i-th minimal path.

Let $$S_1 = \sum_{i=1}^{m} V_i$$

be the sum of probabilities of m paths taken one at a time. Also let $S_2$ denote the sum of probabilities of $\binom{m}{2}$ intersections formed by taking the minimal paths two at a time, $S_3$ the sum of probabilities of the $\binom{m}{3}$ intersections formed by taking three minimal path at a time, etc. Then, by Poincare's method, as described by Feller (10), the system reliability is

$$R = S_1 + \ldots + (-1)^{m-1} S_m$$

where

$$S_1 = \sum_{i=1}^{m} P(V_i)$$

$$S_2 = \sum_{\{i,j\}} P(V_i \cup V_j)$$

$$\vdots$$

$$S_m = P\,(V_1 \cup V_2 \cup \ldots \cup V_m),$$

where $V_i$ is binary vector of minimal path.

For the four component system, the system reliability equation is generated as follows:

$$R_1 = C_1\,C_3$$

$$\begin{aligned} R_2 &= R_1 + C_1\,C_4 - C_1\,C_3\,C_4 \\ &= C_1\,C_3 + C_1\,C_4 - C_1\,C_3\,C_4 \end{aligned}$$

$$\begin{aligned} R_3 &= R_2 + C_2\,C_3 - C_1\,C_2\,C_3 - C_1\,C_2\,C_3\,C_4 + C_1\,C_2\,C_3\,C_4 \\ &= C_1\,C_3 + C_1\,C_4 + C_2\,C_3 - C_1\,C_3\,C_4 - C_1\,C_2\,C_3 \end{aligned}$$

$$\begin{aligned} R_4 &= R_3 + C_2\,C_4 - C_1\,C_2\,C_3\,C_4 - C_1\,C_2\,C_4 \\ &\quad - C_2\,C_3\,C_4 + C_1\,C_2\,C_3\,C_4 + C_1\,C_2\,C_3\,C_4 \\ &= C_1\,C_3 + C_1\,C_4 + C_2\,C_3 + C_2\,C_4 - C_1\,C_3\,C_4 \\ &\quad - C_1\,C_2\,C_3 - C_1\,C_2\,C_4 - C_2\,C_3\,C_4 + C_1\,C_2\,C_3\,C_4. \end{aligned}$$

Burris (4) programmed a function procedure named EQGEN which generates the system reliability equation. EQGEN takes the binary vectors as a minimal state. Then Boolean OR operations are performed successively. Using the four component system again, initialize the input vectors as $(1010)_1$, $(1001)_2$, $(0110)_3$, $(0101)_4$, and the internal computation steps are simulated as following.

| Step | Operation | Result | Generated Vectors |
|---|---|---|---|
| 1 | $(1010)_1$ OR $(1001)_2$ | $(1011)_5$ | $(1010)_1$ |
| | | | $(1001)_2$ |
| | | | $(1011)_5$ |
| 2 | (1010) OR (0110) | $(1110)_6$ | $(1010)_1$ |
| | (1001) OR (0110) | $(1111)_7$ | $(1001)_2$ |
| | (1011) OR (0110) | $(1111)_8$ | $(1011)_5$ |
| | | | $(0110)_3$ |
| | | | $(1110)_6$ |
| | | | $(1111)_7$ |
| | | | $(1111)_8$ |
| 3 | (1010) OR (0101) | $(1111)_9$ | $(1010)_1$ |
| | (1001) OR (0101) | $(1101)_{10}$ | $(1001)_2$ |
| | (1011) OR (0101) | $(1111)_{11}$ | $(1011)_5$ |
| | (0110) OR (0101) | $(0111)_{12}$ | $(0110)_3$ |
| | (1110) OR (0101) | $(1111)_{13}$ | $(1110)_6$ |
| | | | $(1111)_7$ |
| | | | $(1111)_8$ |
| | | | $(0101)_4$ |
| | | | $(1111)_9$ |
| | | | $(1101)_{10}$ |
| | | | $(1111)_{11}$ |
| | | | $(0111)_{12}$ |
| | | | $(1111)_{13}$ |

If they have same bit patterns and different signs, then cancel them out. Finally we have 9 vectors corresponding to the terms in the equation.

(1010), (1001), (1011), (0110), (1110), (0101),

(1101), (0111), (1111).

When the computation of reliability is performed with the probability equations which are expressed in binary vectors, the computer examines all the bits in a vector one by one. If there is a 1 then multiply the value of component reliability at that position to the previous value, otherwise just skip. Then the sum of all the probability values computed by each vector will be the system reliability. The sign of each coefficient of a term (binary vector) is also computed.

## Monte Carlo Sampling of Component Reliability

There are two types of component. One is success or failure type, the other is a time-to-failure component. The Monte Carlo sampling is performed seperately from each random deviate generator depending on the type of component.

### Success-Failure Component

Usually a component testing procedure is done by a Bernoulli process if testing is independent from the other. Thus, for one trial, the distribution is given as the following, which is called the Bernoulli distribution.

$$f(x) = \begin{cases} p, & x = 1, \quad 0 < p < 1 \\ (1 - p) = q, & x = 0 \\ 0, & \text{otherwise} \end{cases}$$

For convenience, we define a random variable x = 1 if a trial results in success and x = 0 if a trial results in failure, and the probability of success (unknown) is assumed to remain constant from trial to trial.

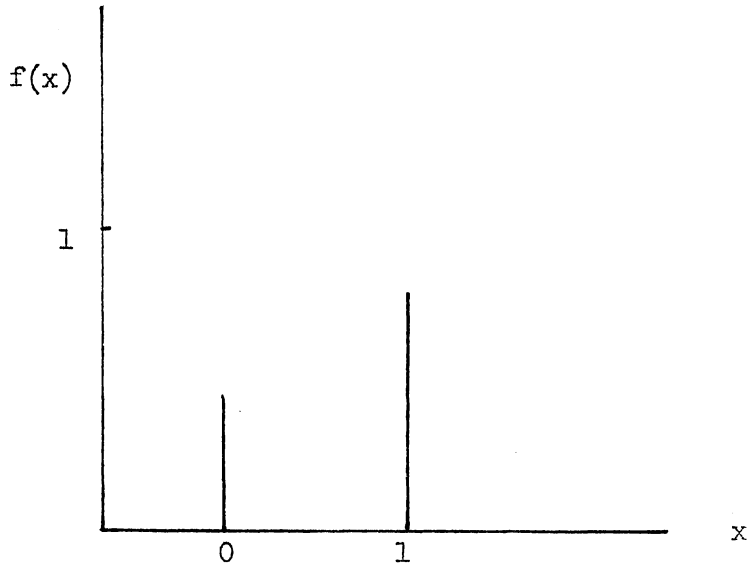



Figure 18. Bernoulli Distribution

The random variable M which denotes the number of successes in n Bernoulli trials has a binomial distribution given by f(m), where

$$f(m) = \begin{cases} \binom{n}{m} p^m (1-p)^{n-m}, & m = 0, 1, 2, \ldots, n \\ & 0 < p < 1 \\ 0, & \text{otherwise.} \end{cases}$$

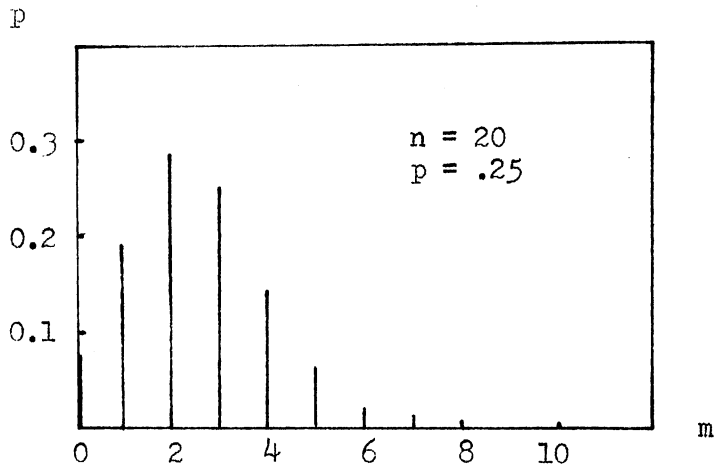



Figure 19. Binomial Distribution

The parameters of a binomial distribution are n and p, where n is positive integer. The value of probability p may be obtained after a given number of Bernoulli trial by taking the ratio of frequency of successes over number of total trials.

For r failures in n tests of a success-failure component, let p be the component reliability. Then the distribution of p is assumed to be the beta prior distribution.

$$f(p) = \frac{p^{r} (1 - p)^{n-r}}{B(r + 1, n - r + 1)} , \qquad 0 < p < 1,$$

where $B(r + 1, n - r - 1)$ is a beta function.

Based on Bayes' Theorem, the conditional distribution

$$f(p \mid m) = \frac{f(p \mid m)\, f(p)}{f(m)}$$

can be generated as a posterior distribution of p given m. It tells us what is known about m given knowledge of the data. Locks (23) and Raiffa and Schlaifer (39) show that a natural conjugate with parameter m of the kernel function, beta probability density function, have exactly the same beta distribution with different parameter values. The parameters of a beta prior distribution are sufficient statistics representing all of the available prior data for the same Bernoulli process. n is the equivalent accumulated prior data, and r is the number of previous occurrences.

Finally the values of reliability from success-failure components are sampled from a beta random deviate generator given the number of success and failures.

Time-to-Failure Component

Each of event can be thought of as a process that generates a number of changes in a fixed interval of time or space. If a process leads to a Poisson distribution, that process is called a Poisson process. The Poisson distribution is

$$f(n) = \begin{cases} \dfrac{(rt)^n \exp(-rt)}{n!}, & n = 0, 1, 2, \ldots \\ 0, & \text{otherwise} \end{cases}$$

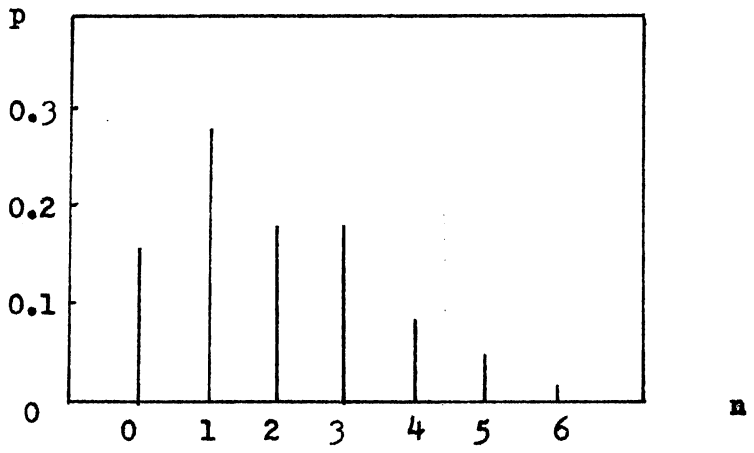



Figure 20. Poisson Distribution

This is the probability of n events during time t when the average number of events per unit of time is r. In other words, r is the expected number of events per unit measure, i.e., the intensity of the process, and t is the unit of time or space during which events are to be counted.

For a time-to-failure component in a Bayesian context, the failure rate r has a gamma probability density function

$$h(r) = \frac{t^n r^{n-1} e^{-rt}}{(n)}, \qquad r, n > 0$$

where $\Gamma(n)$ is the gamma function and n and t are sufficient statistics

for r. When the failure rate r is equal to 1 (scaled in units of required operating time), the parameter t is the accumulated operating time for the component, and n is the number of failure in that time (23). Taking the natural logatithm of both sides of $y = e^{-r}$ gives $r = -\ln(y) = \ln(1/y)$. The result of the transformation gives the component reliability y, a negative log gamma probability density function (23).

$$g(y) = \frac{t^n (\ln (1/y))^{n-1} y^{t-1}}{\Gamma (n)} , \quad 0 < y < 1 \quad n, t > 0$$

The probability density function g(y) has mode at

$$\exp (-(n - 1)/(t - 1)).$$

The mean is

$$(t/(t + 1))^n$$

and the variance is

$$(t/(t + 2))^n - (t/(t + 1)^{2n}$$

The values of failure rate of time-to-failure type component are sampled from a gamma random deviate generator.

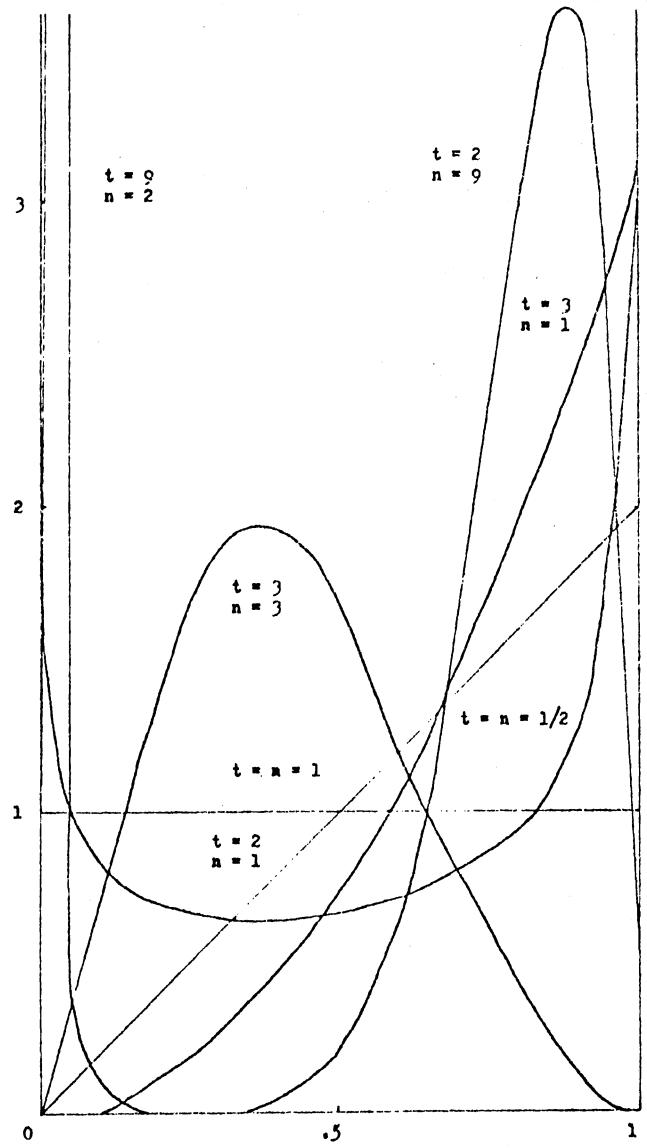
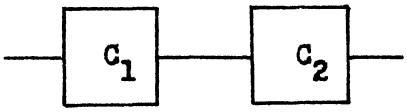
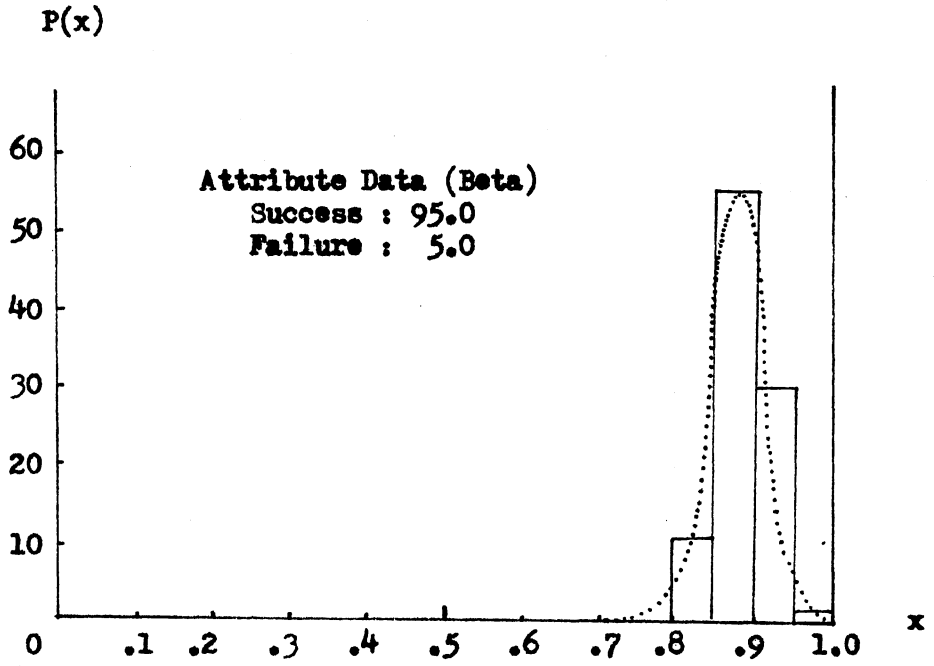



**Figure 21. Negative Log Gamma Density Functions**

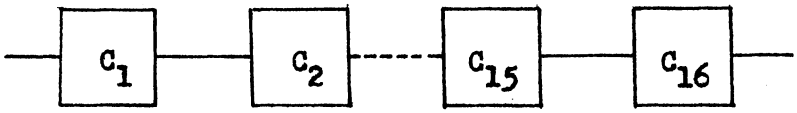
## Statistical Analysis of System Reliability

Using a few simple system donfiguration and one complex system which has two modules in it, the frequency is pictured below by means of histograms. Each sample has 100 observations resulting from 100 simulations. The expected system reliability is computed with each expected value of component reliability. Arbitrary selected historical component reliabilities are assumed identical in the specified system.

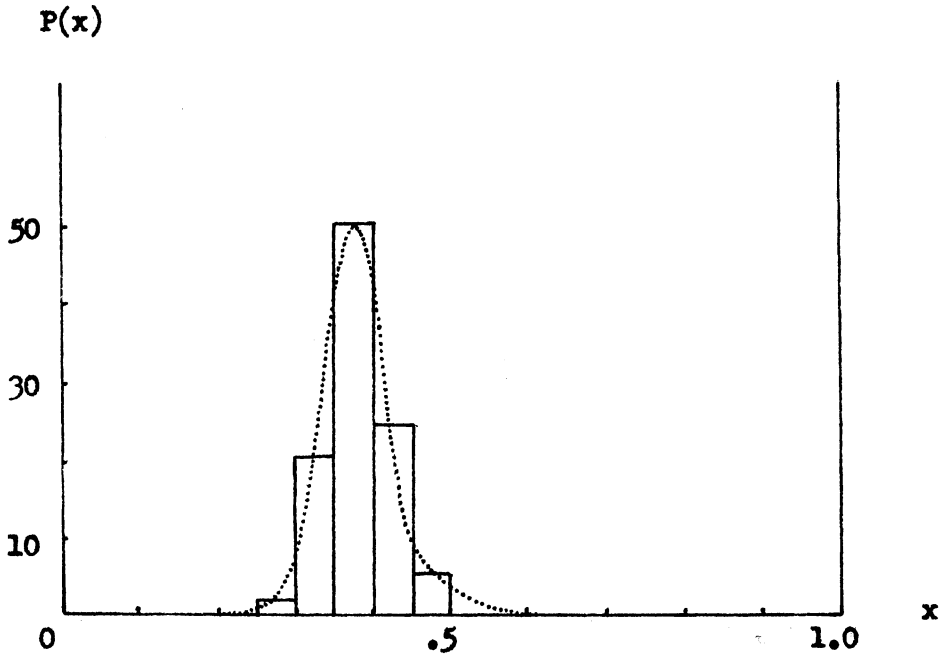


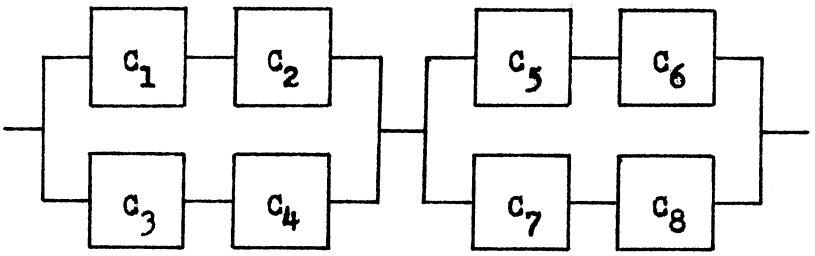

| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .88501 |
| Average of System Reliability | : .88808 |
| Variance | : .00093 |
| Standard Deviation | : .03045 |



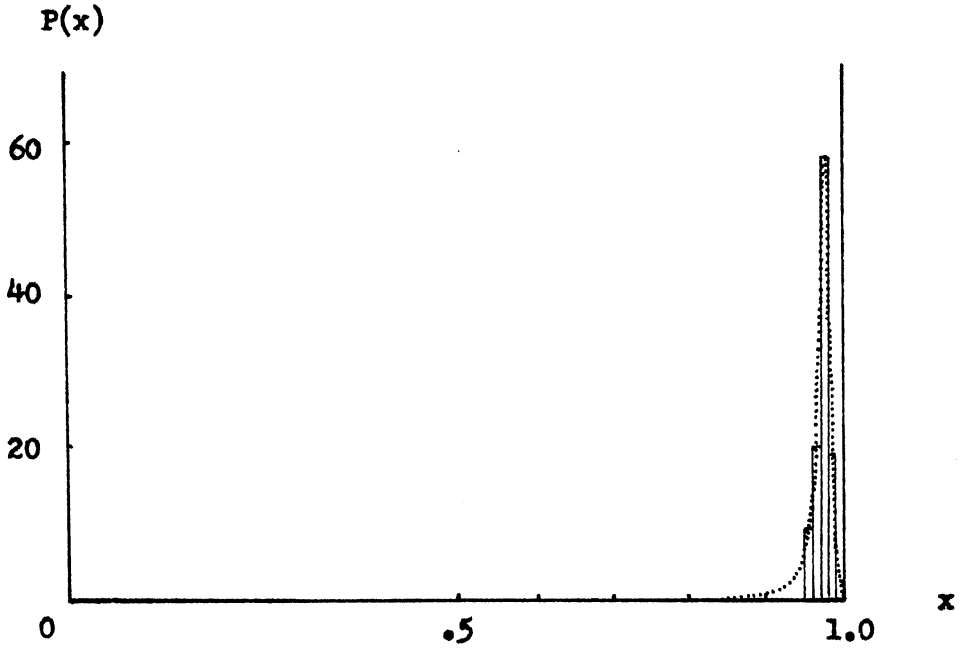

**Attribute Data (Beta)**
**Success : 95.0**
**Failure : 5.0**



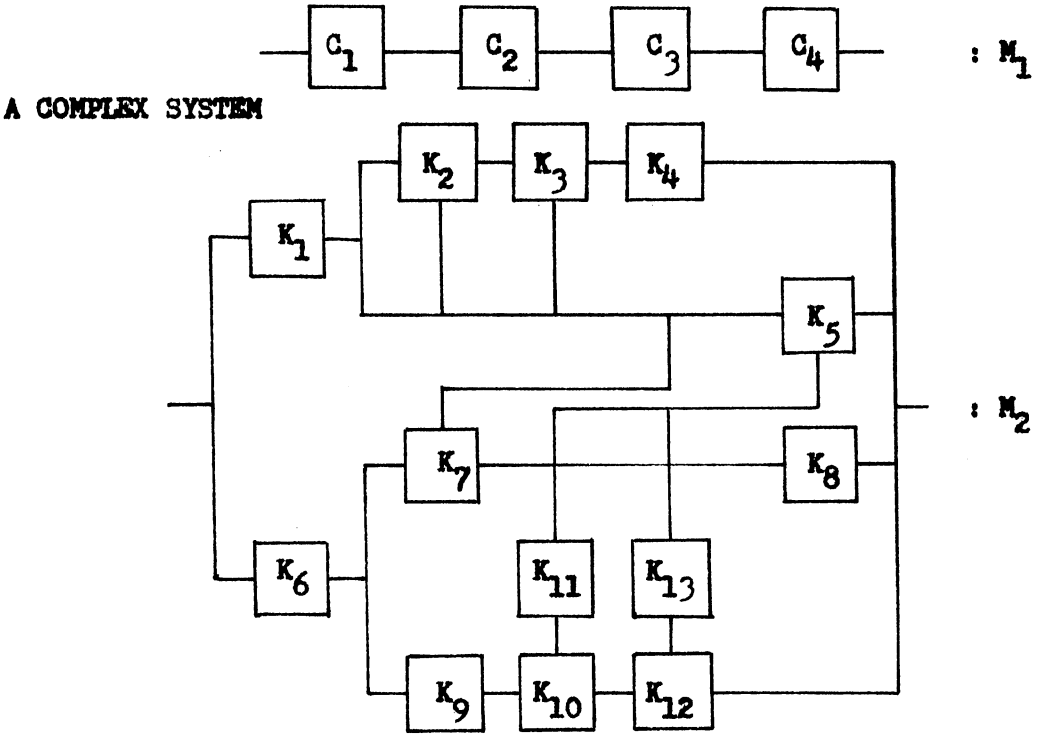

| | |
|---|---|
| **No. of Trials** | : 100 |
| **Expected System Reliability** | : .38871 |
| **Average of System Reliability** | : .38207 |
| **Variance** | : .00150 |
| **Standard Deviation** | : .03874 |



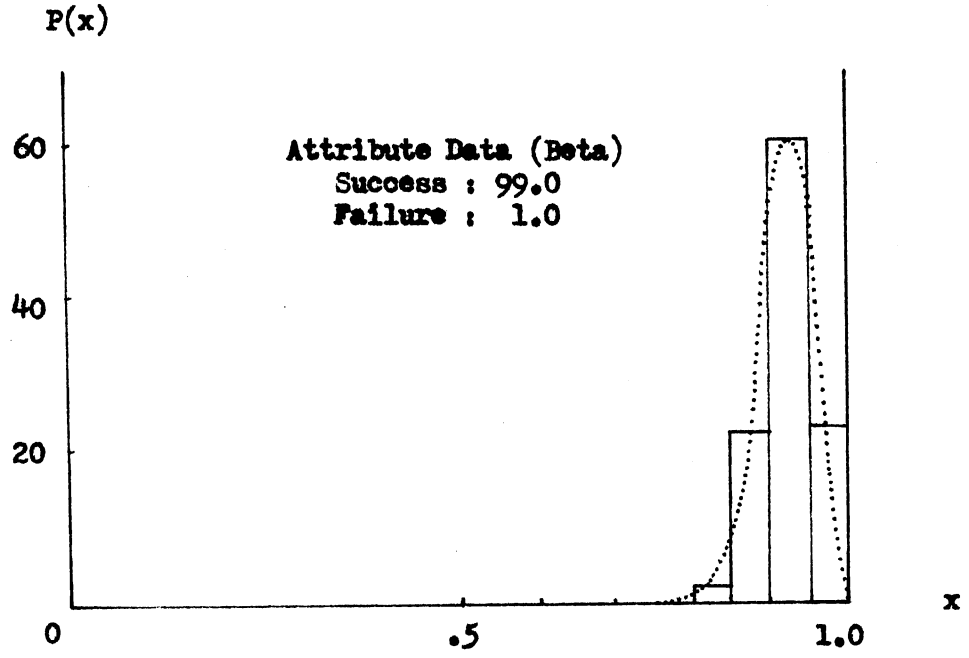

**Attribute Data (Beta)**
**Success : 95.0**
**Failure : 5.0**



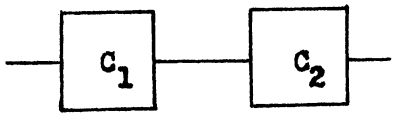

| | |
|---|---|
| **No. of Trials** | : 100 |
| **Expected System Reliability** | : .974093 |
| **Average of System Reliability** | : .973949 |
| **Variance** | : .000054 |
| **Standard Deviation** | : .007324 |



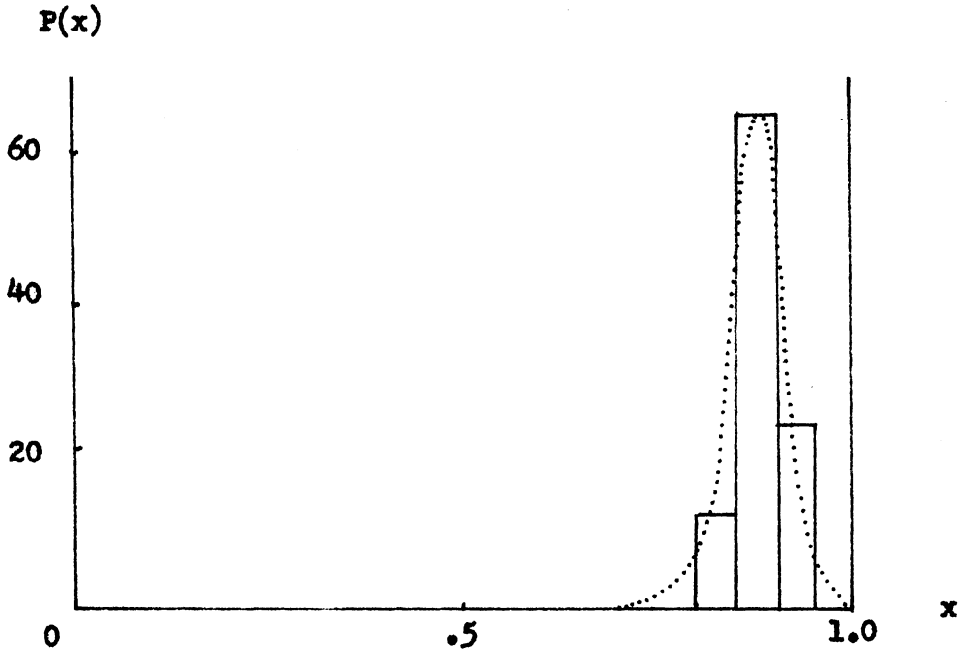

A COMPLEX SYSTEM



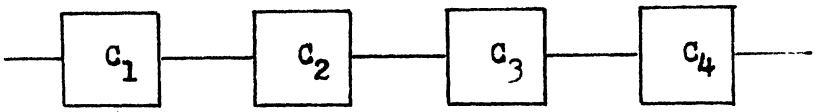

| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .923461 |
| Average of System Reliability | : .924347 |
| Variance | : .000812 |
| Standard Deviation | : .028495 |



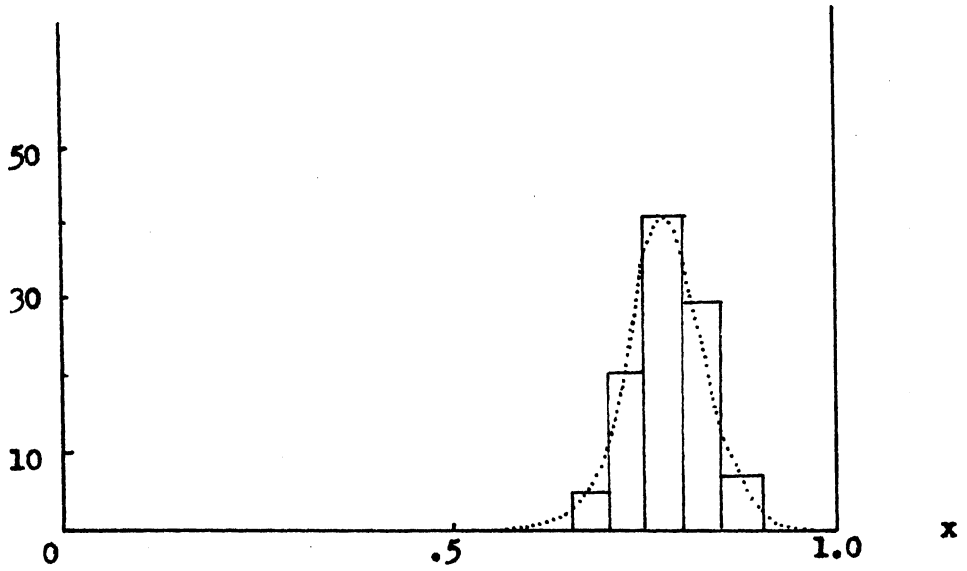

Time-to-Failure Data (Gamma)
Mission Time : 95.0
Failure : 5.0



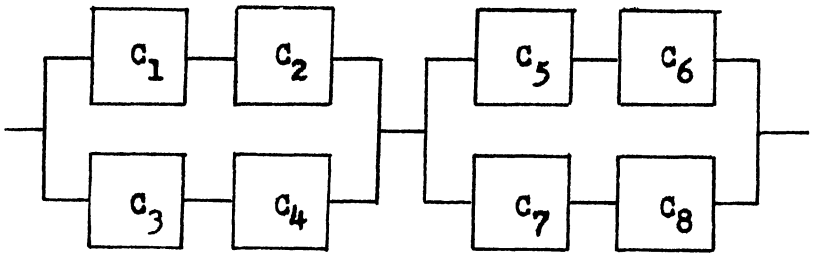

| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .883068 |
| Average of System Reliability | : .882180 |
| Variance | : .000874 |
| Standard Deviation | : .029557 |



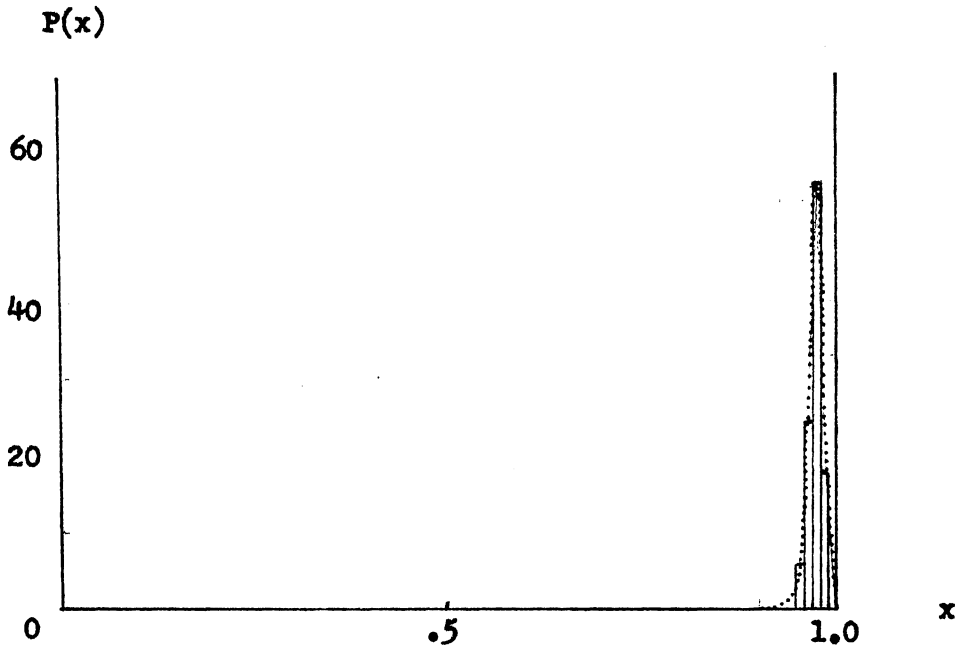

Time-to-Failure Data (Gamma)
Mission Time : 95.0
Failure : 5.0



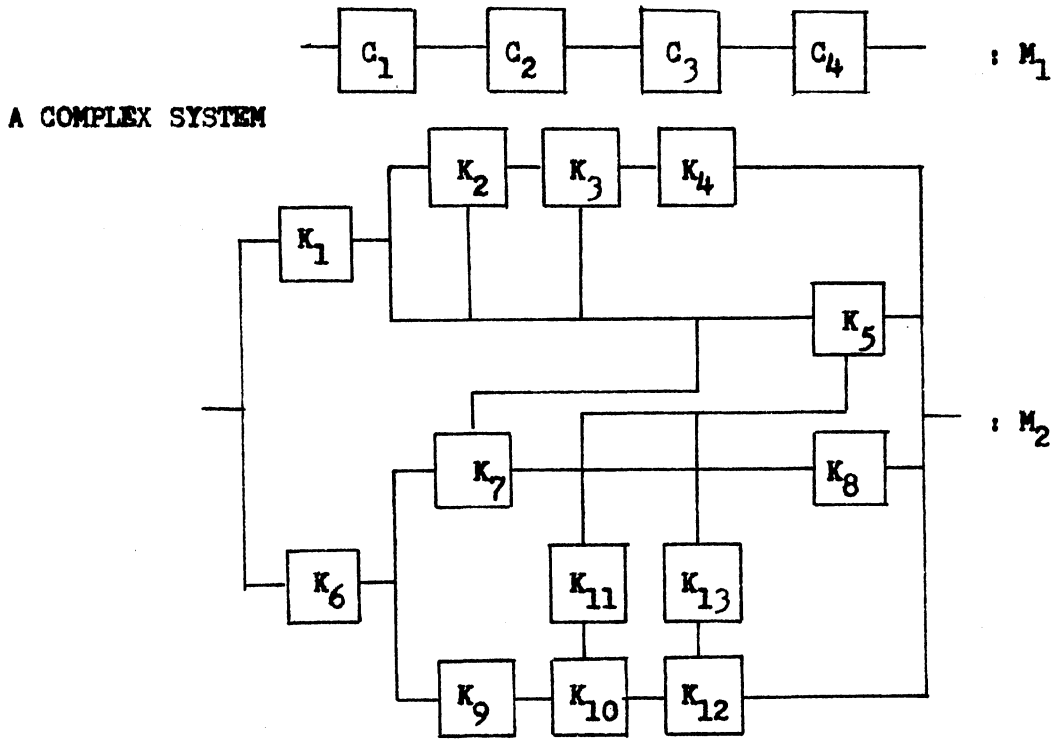

| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .779809 |
| Average of System Reliability | : .781250 |
| Variance | : .001718 |
| Standard Deviation | : .041452 |

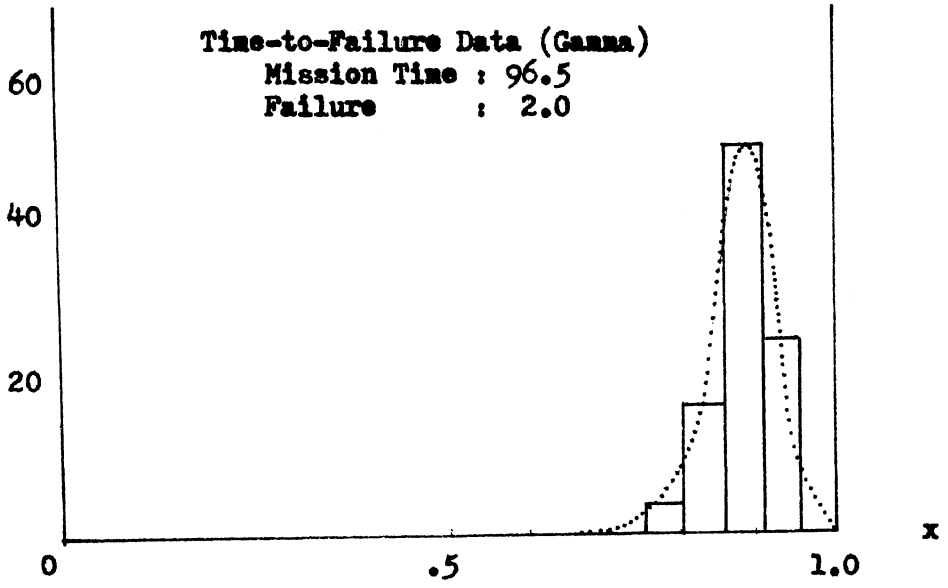



Time-to-Failure Data (Gamma)
Mission Time : 95.0
Failure : 5.0




| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .972841 |
| Average of System Reliability | : .973024 |
| Variance | : .000059 |
| Standard Deviation | : .007683 |




| | |
|---|---|
| No. of Trials | : 100 |
| Expected System Reliability | : .883843 |
| Average of System Reliability | : .880671 |
| Variance | : .001060 |
| Standard Deviation | : .032562 |

Figure 22. Empirical Distribution of System Reliability Resulting Computer Simulation

The pictorial representation does not give us accurate information on the character of the distribution of system reliability. However, the system reliability may have a beta distribution if we consider it as a tested unit (system) like tested component to obtain component reliability. With the assumption that system reliability is also distributed approximately as a beta distribution, an estimation of parameters of the beta distribution is attempted. Fielitz and Myers (11) showed that computational complications are involved in solving the maximum likelihood estimation problem for the beta distribution. The method of moments is shown below for obtaining reasonable estimate of the parameters in the beta distribution. The univariate beta distribution has density function

$$f(x) = \frac{\Gamma(a + b)}{\Gamma(a)\,\Gamma(b)} \quad p^{a-1}\,(1 - x)^{b-1}, \quad 0 \le x \le 1 \quad a, b > 0$$

Then expectation is

$$E(x) = \frac{a}{a + b},$$

Variance is

$$v(x) = \frac{ab}{(a + b)^2\,(a + b + 1)}$$

$$= E(x^2) - (E(x))^2$$

$$E(x^2) = \frac{ab + a^2(a + b + 1)}{(a + b)^2\,(a + b + 1)}$$

Consider the bivariate case where parameters are a, b, and c.

Then

$$E(x_1) = \frac{a}{a + b + c}$$

$$E(x_2) = \frac{b}{a + b + c} \quad \text{are given.}$$

$$E(x_1^2) = \frac{a(b + c) + a^2(a + b + c + 1)}{(a + b + c)^2(a + b + c + 1)}$$

and

$$a + b + c = \frac{a}{E(x_1)} = \frac{b}{E(x_2)}$$

hence

$$b = \frac{a\,E(x_2)}{E(x_1)}$$

and

$$b + c = \frac{a(1 - E(x_1))}{E(x_1)} \quad .$$

substituting

$$E(x_1^2) = \frac{a\left(\frac{a(1 - E(x_1))}{E(x_1)}\right) + a^2\,\frac{a}{E(x_1)} + 1}{\frac{a^2}{(E(x_1))^2} \cdot \left(\frac{a}{E(x_1)} + 1\right)}$$

$$= \frac{\frac{a^2(1 - E(x_1))}{E(x_1)} + a^2\left(\frac{a + E(x_1)}{E(x_1)}\right)}{\frac{a^2(a + E(x_1))}{(E(x_1))^3}}$$

$$= \frac{(1 - E(x_1) + a + E(x_1))\ (E(x_1))^2}{a + E(x_1)}$$

Then

$$(a + E(x_1))\, E(x_1^2) = (E(x_1))^2 + a(E(x_1))^2$$

$$a\, E(x_1^2) - a(E(x_1))^2 = (E(x_1))^2 - E(x_1)\, E(x_1^2)$$

finally

$$\hat{a} = \frac{(E(x_1))^2 - E(x_1)\, E(x_1^2)}{E(x_1^2) - (E(x_1))^2}$$

$$= \frac{E(x_1)\,(E(x_1) - E(x_1^2))}{V(x_1)}$$

The derivation for $\hat{b}$ takes similar procedure.

$$\hat{b} = \frac{(E(x_1) - E(x_1^2))\,(1 - E(x_1^2))}{E(x_1^2) - (E(x_1))^2}$$

$$= \frac{(1 - E(x_1^2))\,(E(x_1) - E(x_1^2))}{V(x_1)}$$

Fielitz and Myers pointed that maximum likelihood has more desirable properties than the method of moments. For the beta distribution, because of the lack of an upper bound on the parameters, the

method of moments appears to present a good way to compute a starting point for a relatively complicated numerical search scheme applied to the likelihood function.

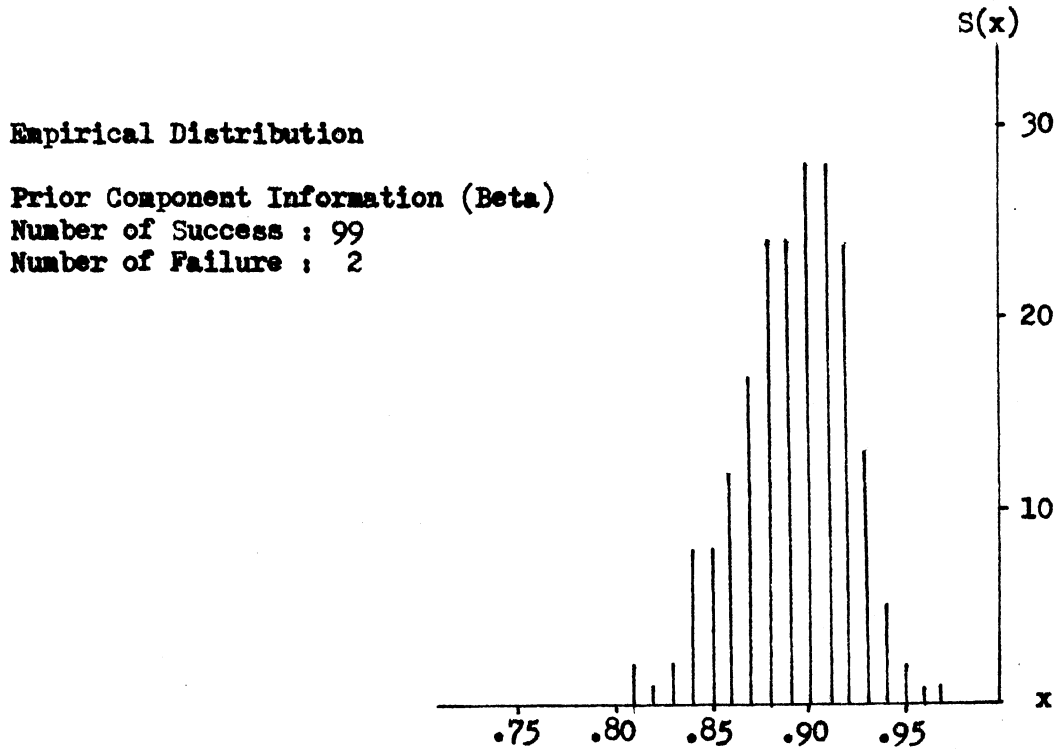
In order to apply the method of moments to estimate beta parameters, four different simulation runs were performed. A complex system, which has two modules in it in series, was chosen. The first module has seven components in series. Three of them are redundant components. The other has thirteen components which form a complicated module. The prior information, which are the values of component reliability, are selected as identical for convenience of recognition. However, the values of prior component data are the ones which could occur often in real situations. The procedure starts from 200 simulations and generates 200 system reliabilities. Then the beta parameters are estimated based upon the computer generated empirical distribution of system reliabilities, using the method of moments.

A comparison is attempted between the empirical distribution and theoretical beta distribution which has the parameters estimated. A set of random beta deviates, which are generated based upon the estimated parameter values, is used as a theoretical beta distribution. Once these two distributions are available, the chi-square goodness-of-fit tests are performed to verify the null hypothesis which tells the equality between the empirical distribution, S(x), and a theoretical distribution, F(x). The htpothesis is

Null hypothesis ($H_0$) : $S(x) = F(x)$ for all x

Alternative hypothesis ($H_1$) : $S(x) = F(x)$ for at least one x.

For the chi-square test, each of 200 observations in a sample are grouped by .01 difference of system reliability. In other words, if the first two digits under the decimal point are the same as in the other



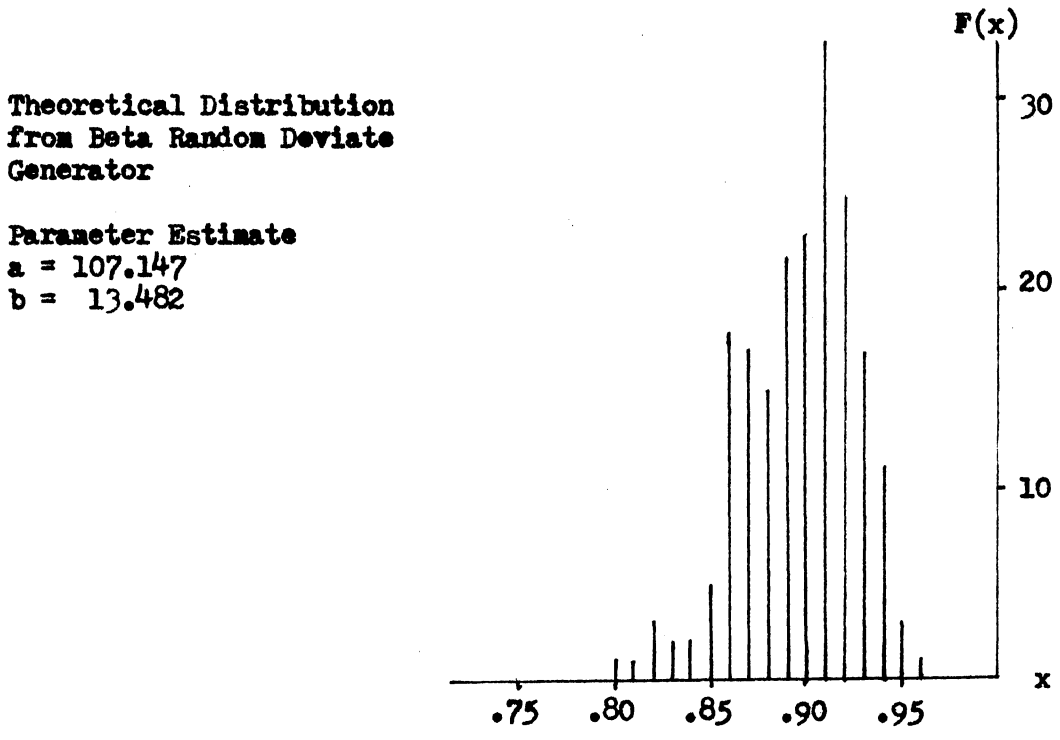




Figure 23. Comparison between Empirical & Hypothesized Distribution I

TABLE IX

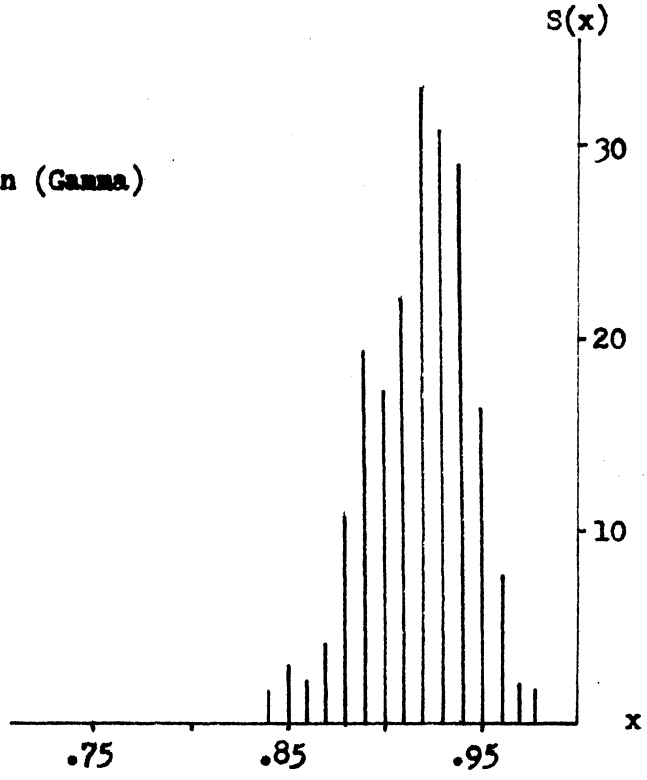
SYSTEM RELIABILITY FREQUENCY DISTRIBUTION I

| | Class | Observed Frequency | Expected Frequency |
|---|---|---|---|
| 1 | .80 | 0 | 1 |
| 2 | .81 | 2 | 1 |
| 3 | .82 | 1 | 3 |
| 4 | .83 | 2 | 2 |
| 5 | .84 | 8 | 2 |
| 6 | .85 | 8 | 5 |
| 7 | .86 | 12 | 18 |
| 8 | .87 | 17 | 17 |
| 9 | .88 | 24 | 15 |
| 10 | .89 | 24 | 22 |
| 11 | .90 | 28 | 23 |
| 12 | .91 | 28 | 33 |
| 13 | .92 | 24 | 25 |
| 14 | .93 | 13 | 17 |
| 15 | .94 | 5 | 11 |
| 16 | .95 | 2 | 3 |
| 17 | .96 | 1 | 2 |
| 18 | .97 | 1 | 0 |

Chi-Square Test Statistic : 15.4
Degrees of Freedom : 15
Observed Significance Level : $.25 < p < .5$
F test for Equality of Variance
Test Statistic : 1.0187
O.S.L : $.25 < p$

Empirical Distribution

Prior Component Information (Gamma)
Missions : 99
Number of Failure : 1




Theoretical Distribution
from Beta Random Deviate
Generator

Parameter Estimate
$\hat{a}$ = 95.186
$\hat{b}$ = 7.935

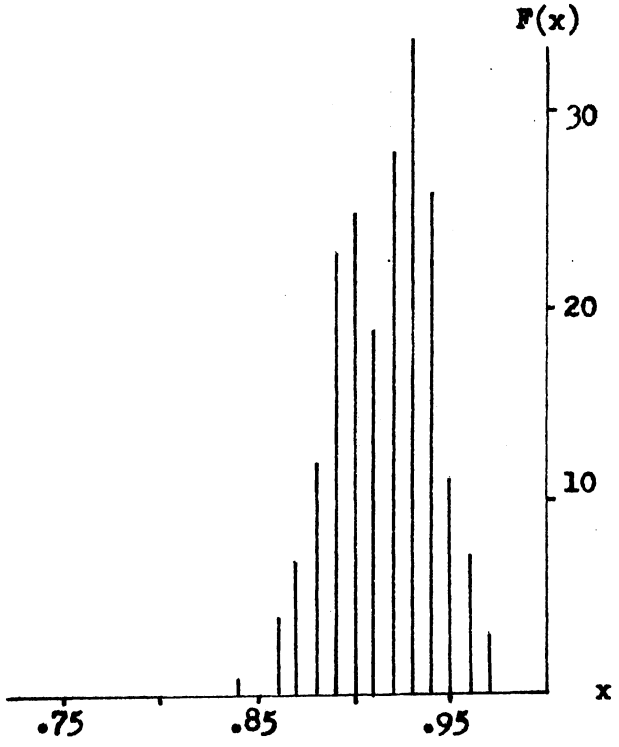


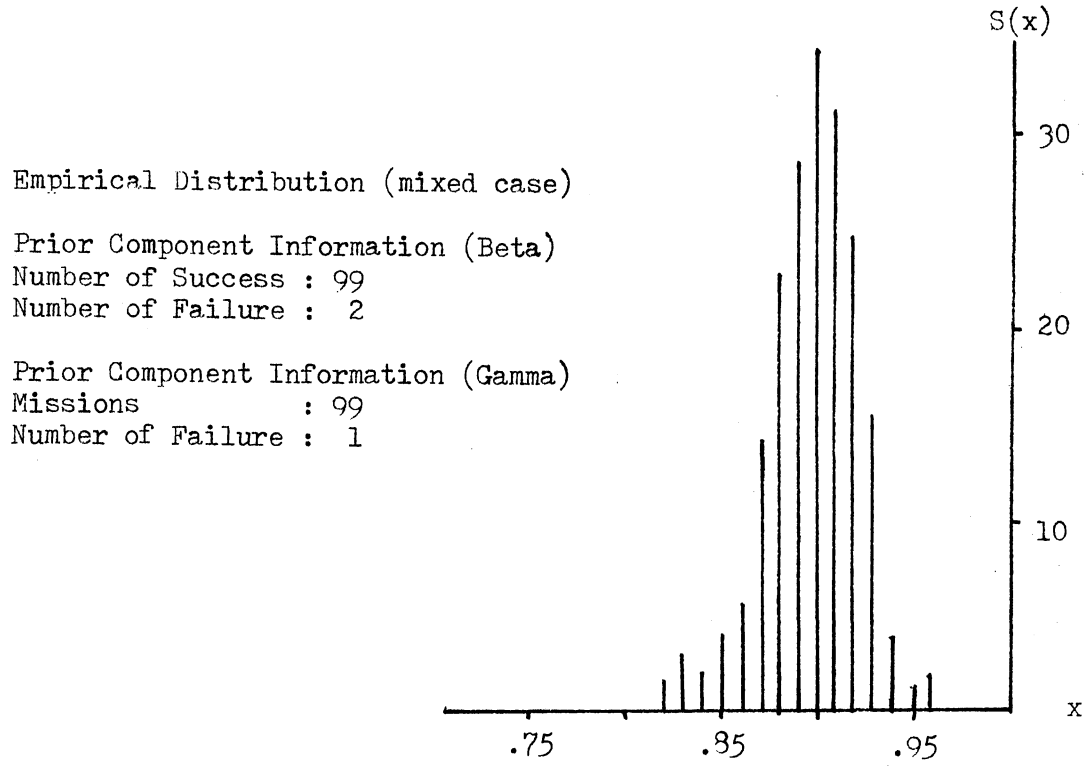

Figure 24. Comparison between Empirical & Hypothesized Distribution II

TABLE X

SYSTEM RELIABILITY FREQUENCY DISTRIBUTION II

| | Class | Observed Frequency | Expected Frequency |
|---|---|---|---|
| 1 | .84 | 2 | 1 |
| 2 | .85 | 3 | 0 |
| 3 | .86 | 2 | 4 |
| 4 | .87 | 4 | 7 |
| 5 | .88 | 11 | 12 |
| 6 | .89 | 19 | 23 |
| 7 | .90 | 17 | 25 |
| 8 | .91 | 22 | 19 |
| 9 | .92 | 33 | 28 |
| 10 | .93 | 31 | 34 |
| 11 | .94 | 29 | 26 |
| 12 | .95 | 16 | 11 |
| 13 | .96 | 8 | 7 |
| 14 | .97 | 2 | 3 |
| 15 | .98 | 1 | 0 |

Chi-Square Test Statistic : 12.266
Degree of Freedom : 12
Observed Significance Level : $.25 < p < .5$
F Test for Equality of Variance
Test Statistic : 1.0160
O.S.L : $.25 < p$



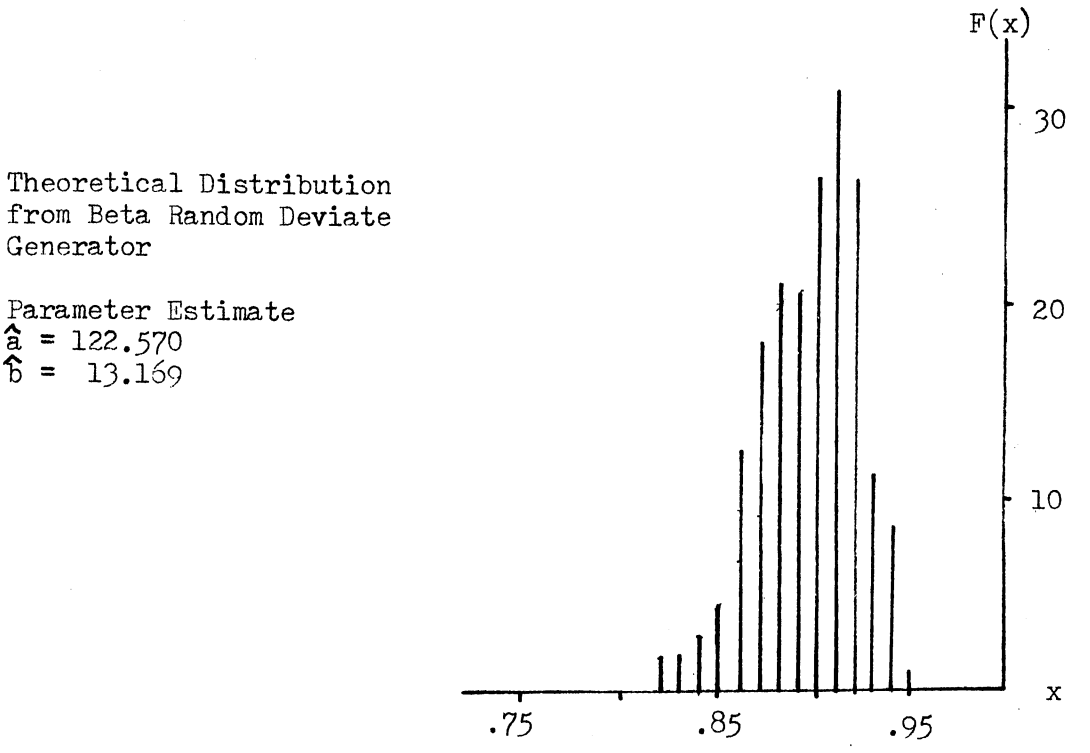




Figure 25. Comparison between Empirical & Hypothesized Distribution III

TABLE XI

SYSTEM RELIABILITY FREQUENCY DISTRIBUTION III

| | Class | Observed Frequency | Expected Frequency |
|---|---|---|---|
| 1 | .82 | 2 | 2 |
| 2 | .83 | 3 | 2 |
| 3 | .84 | 2 | 3 |
| 4 | .85 | 4 | 5 |
| 5 | .86 | 6 | 13 |
| 6 | .87 | 11 | 19 |
| 7 | .88 | 24 | 22 |
| 8 | .89 | 30 | 22 |
| 9 | .90 | 36 | 28 |
| 10 | .91 | 33 | 34 |
| 11 | .92 | 26 | 28 |
| 12 | .93 | 16 | 12 |
| 13 | .94 | 4 | 9 |
| 14 | .95 | 1 | 1 |
| 15 | .96 | 2 | 0 |

Chi-Square Test Statistic : 19.866
Degree of Freedom : 12
Observed Significance Level : $.05 < p < .1$
F Test for Equality of Variance
Test Statistic : 1.2819
Observed Significance Level : $.25 < p$

value of system reliability, the two values are considered as being in the same class.

The degrees of freedom is calculated by c - k - 1, where c is the number of groups and k is number of parameter estimated. k is 2 for this case because two unknown beta parameters are estimated from sample.

The results of the chi-square goodness of fit tests do not give us enough evidence to reject the null hypothesis.

## Conclusion

The newly developed procedure subroutine CBETA (beta random deviate generator) shows good fit by the result of the Kolmogorov goodness-of-fit tests. The computing time is bounded as the value of the parameters of the beta distribution gets larger. Furthermore, CBETA works very efficiently in terms of computing time when high success-failure component reliabilities are generated. Usually, the historical component data contains a high reliability value in a real situation. A beta random deviate is also available from a gamma deviate generator by a transformation of random variable. But the computing time is approximately twice slower than using this beta generator.

The RGAMMA (Marsaglia "Squeeze" method gamma random deviate generator) procedure subroutine also shows good fit by the result of the Kolmogorov goodness-of-fit test. It may be well said that Marsaglia's RGAMMA is the best known technique as an accurate and speedy method to generate gamma deviates presently.

The result of chi-square goodness-of-fit tests between empirical distribution and hypothesized distribution of system reliability

shows that there is not enough evidence to reject the null hypothesis. But some other known statistical tests may be required to support this result strongly.

The SPARCS program is not only improved in computing time by using the fast random deviate generators but also has reduced requirements of core storage for compilation and excution. Every dimension of the arrays is determined by input data except for the variables used for terms and coefficients in the system reliability equation. The storage requirement for the SPARCS program is presently less than 104k bytes.

## SELECTED BIBLIOGRAPHY


( 1) Abramowitz, M. and I.A. Stegun. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. U.S. Dept. of Commerce, NBS Applied Mathematics Series No. 55, June, 1964.

( 2) Ahrens, J.H. and U. Dieter. "Computer Methods for Sampling from the Gamma, Beta, Poisson and Binomial Distributions." Computing, 12, (1974), pp. 223-246.

( 3) Boston, N.E. and E.L. Battiste. "Incomplete Beta Probability Distribution Function." COMM. ACM 17 (1974), pp. 156-157.

( 4) Burris, J.L. "Model for the Analysis of Probabilities of Systems (MAPS)." MBA Research Report, Oklahoma State University, 1972.

( 5) Chandler, J.P. Private Communication. Stillwater, Oklahoma: Oklahoma State University, 1976.

( 6) Chandler, J.P. RANF-Uniformly Distributed Pseudo-Random Numbers. Unpublished Document, Dept. of Computing and Information Science, Oklahoma State University, 1970.

( 7) Computer Subroutine Libraries in Mathematics and Statistics (Library Contents). International Mathematical and Statistical Libraries, Inc., GNB Building, Houston, TX 77036, IMSL LIB-0004p (April, 1974).

( 8) Conover, W.J. Practical Nonparametric Statistics. New York: Wiley & Sons, Inc., (1971), pp. 293-298.

( 9) Cooley, J.W. Simulation Program for Assessing the Reliability of Complex Systems (SPARCS). Ph. D. thesis, Oklahoma State University, 1976.

(10) Feller, W. An Introduction to Probability Theory and Its Applications. New York: John Wiley & Sons, Inc., (1968), pp. 98-101.

(11) Fielitz, B.D. and B.L. Myers. "Estimation of Parameters in Beta Distribution." Decision Sciences, vol. 6 (1975), pp. 1-13.

(12) Gnedenko, B.V., et al. Mathematical Methods of Reliability Theory. New York, Academic Press, 1969.

(13) Greenwood, A.J. "A fast Generator for Gamma-Distributed Random Variables." COMSTAT Proceedings in Computational Statistics. G. Bruckmann, F. Ferschl, L. Schmetterer, Editors, Physica Verlag, Vienna, (1974), pp. 19-27.

(14) Hammersley, J.M. and D.C. Handscomb. Monte Carlo Methods. Methuen, London, 1964.

(15) Harter, H.L. New Tables of The Incomplete of Percentage Points of The Chi-Square and Beta Distributions. Wright-Patterson Air Force Base: Aerospace Research Laboratories, United States Air Force, 1964.

(16) Hasting, C. Jr. Approximations for Digital Computers. Princeton, New Jersey: Princeton University Press, 1955.

(17) Johnk, M.D. Erzeugung Von Betavesteilten und Gammavestrilten Zufellszahlen. Metrica 8, 1, (1964), pp. 5-15.

(18) Johnson, N.L. and S. Kotz. Continuous Univariate Distribution-1 New York, Houghton, Mifflin Co., (1970), pp. 176.

(19) Kahn, H. Applications of Monte Carlo. RAND Report No. RM 1237 AEC, 1954.

(20) Knuth, D.E. The Art of Computer Programming Volume 2: Seminumerical Algorithms. Addison-Wesley, Reading, Mass., 1973.

(21) Lee, K.K. Error Analysis of Various Computing Methods of Generating Beta and Gamma Variates. M.S. Report in Statistics, Oklahoma State University, 1976.

(22) Locks, M.O. "Error Analysis of Various Methods for Generating Beta and Gamma Variates." Proceeding of Computer Science and Statistics: 9th Annual Symposium on the Interface. (April, 1976).

(23) Locks, M.O. Reliability, Maintainability, and Reliability Assessment. Rochelle Park, New Jersey: Spartan Books, 1973.

(24) Locks, M.O. "The Maximum Error in System Reliability Calculations by Using a Subset of The Minimal States." I.E.E.E. Transactions on Reliability, vol. R-24, No. 4 (November, 1971), pp. 231-234.

(25) Locks, M.O. "Exact Minimal State System Reliability Analysis." Proceeding of Computer Science and Statistics: 5th Annual Symposium on the Interface (November, 1971)

(26) Maclaren, M.D. and G. Marsaglia. "Uniform Random Number Generator." Journal of ACM, 12 (1965), pp. 83

(27) Marsaglia, G. "Generating Exponential Random Variables." Ann. Math. Stat. 32 (1961), pp. 899-902.

(28) Marsaglia, G. "The Squeeze Method for Generating Gamma Variates." The McGill Random Number Package 'Super Duper'. School of Computer Science, McGill University, Montreal.

(29) Marsaglia, G. Proceedings of the National Academy of Science, 60 (1968), pp. 5.

(30) Marsaglia, G. "Expressing a random Variable in Terms of Uniform Random Variables." Annals Math. Stat, vol. 32, (1961), pp. 894-898.

(31) Marsaglia, G. and T.A. Bray. "A Convenient Method for Generating Normal Variables." SIAM Rev. 6 (1964), pp. 260-264.

(32) Mann, N.R., R.E. Schafer, and N.D. Singpurwalla. Methods for Statistical Analysis of Reliability and Life Data. New York: John Wiley and Sons, 1974.

(33) Mayer, B.L. and N.L. Enrick. Statistical Functions; A Source of Practical Deviations Based on Elementary Mathematics. The Kent State University Press. Kent State University, (1970), pp. 64-66.

(34) Miles, R.F., Jr. System Concepts-Lectures on Contemporary Approaches to Systems. New York: John Wiley & Sons. Inc., (1973), pp. 3.

(35) Monte Carlo Baysian System Reliability and MTBF Confidence Assessment. Technical Report AFFDL-TR-75-144, Air Force Flight Dynamics Laboratory, Air Force Wright Aeronautical laboratories, Air Force Systems Command, Wright Patterson Air Force Base, Ohio.

(36) Naylor, T.H., et al. Computer Simulation Techniques. New York: Wiley & Sons, Inc., 1966.

(37) Newman, T.G. and Odell P.L. The Generation of Random Variates, Hafner Publishing Co. New York: (1971), PP. 19-26.

(38) Pearson, K. Tables of the Incomplete Beta-Function. Cambridge: Cambridge University Press for the Biometrika Trustees, 1964.

(39) Pearson, K. Tables of the Incomplete -Function. Cambridge: Cambridge University Press for the Biometrika Trustees, 1968.

(40) Raiffa, H. and R. Schlafer. Applied Statistical Decision Theory. Boston, Mass., Graduate School of Business Administration, Harvard University, 1961.

(41) Rosenblatt, J.R. "Confidence Limits for the Reliability of Complex Systems." Statistical Theory of Reliability. Ed. Marvin Zelen. University of Wisconsin Press, (1963), pp. 115-137.

(42) Teichroew, D. Distribution Sampling with High Speed Computers. Ph. D. Thesis, University of North Carolina, 1953.

(43) Von Gelder, A. "Some New Results in Pseudo-Random Number Generations." Journal of the Association of Computing Machinery, Vol. 14 (October, 1967), pp. 785-792.

(44) Von Neumann, J. "Various Technique in connection with Random Digits." Monte Carlo Methods. Nat. Bureau Standards, AMS 12 (1951), pp. 36-38.

(45) Wilson, E.B. and M.M. Hilferty. The Distribution of Chi-Square." Proceedings National Academy of Science, Vol. 17 (1931), pp. 684-688.

# APPENDIX A

## SAMPLE OUTPUT REVISED

## SPARCS PROGRAM

* S P A R C S *
SIMULATION PROGRAM FOR THE ANALYSIS OF THE RELIABILITY OF COMPLEX SYSTEM

| | | |
|---|---|---|
| SYSTEM IDENTIFICATION | SOFTWARE RELIABILITY-UNRELIABILITY COMPUTATION | BROWN & LIPOW |
| NUMBER OF SIMULATIONS | 20 | |
| NUMBER OF MODULES | 2 | |
| NUMBER OF COMPONENTS | 2 | |
| NUMBER OF MINIMAL CUT | 2 | |
| TYPE OF ANALYSIS | UNRELIABILITY | |

SPARCS :: EQUATION GENERATION ROUTINE

THE 2 MINIMAL CUT FOR THE SYSTEM

<B>
<A>

$$U_{SYS} = U_B + U_A - U_A U_B$$

THE 4MINIMAL CUT FOR THE MODULE A

<1>
<3>
<5>
<7>

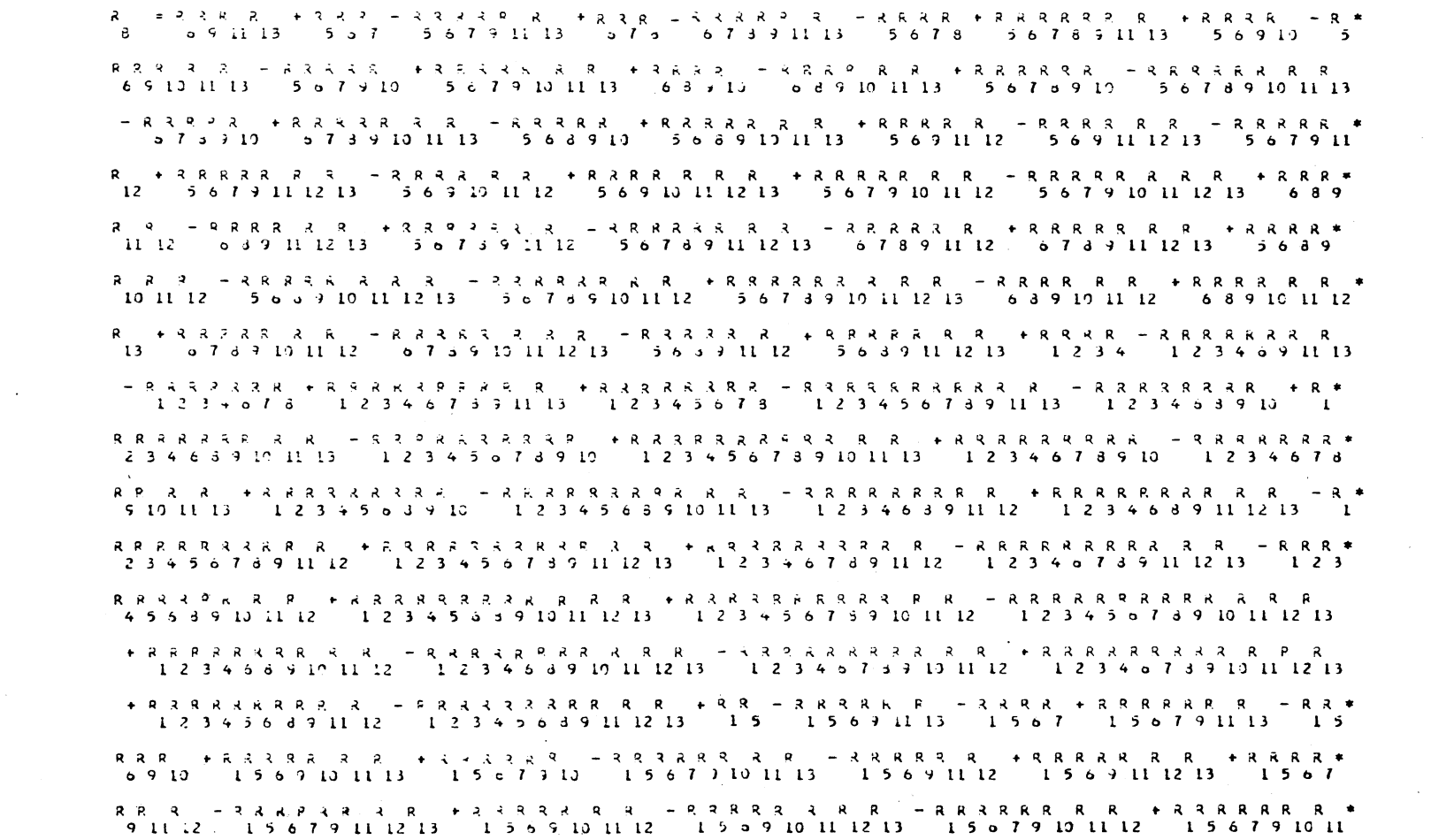
$$U_A = U_1 + U_3 - U_1 U_3 + U_5 - U_1 U_5 - U_3 U_5 + U_1 U_3 U_5 + U_7 - U_1 U_7 - U_3 U_7 + U_1 U_3 U_7 - U_5 U_7 + U_1 U_5 U_7 + U_3 U_5 U_7 - U_1 U_3 U_5 U_7$$

HISTORICAL INFORMATIONS FOR EACH COMPONENT IN MODULE A

| COMPONENT | TYPE | SUCCESS (BETA) EQUIVALENT MISSIONS(GAMMA) | FAILURES | PRIOR DISTRIBUTION MEAN |
|---|---|---|---|---|
| 1 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.969851 |
| 2 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.980392 |
| 3 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.980392 |
| 4 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.980392 |
| 5 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.969851 |
| 6 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.980392 |
| 7 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.980392 |

THE 9MINIMAL PATH FOR THE MODULE 8

<6,9,11,13>
<5,6,7>
<6,7,9>
<5,6,9,10>
<6,8,9,10>
<5,6,9,11,12>
<6,8,9,11,12>
<1,2,3,4>
<1,5>



```
R  R   = R R R R R  * R R R R R R R R  R
 12 13    1 2 3 4 5    1 2 3 4 5 6 9 11 13
```

HISTORICAL INFORMATIONS FOR EACH COMPONENT IN MODULE 3

| COMPONENT | TYPE | SUCCESS (BETA) EQUIVALENT MISSIONS(GAMMA) | FAILURES | PRIOR DISTRIBUTION MEAN |
|---|---|---|---|---|
| 1 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 2 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 3 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 4 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 5 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 6 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 7 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 8 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 9 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 10 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 11 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |
| 12 | ATTRIBUTE(BETA) | 99.00 | 1.00 | 0.98039 |
| 13 | TIME-TO-FAILURE(GAMMA) | 96.50 | 2.00 | 0.96985 |

SPARCS :: SYSTEM SIMULATION ROUTINE

SYSTEM RELIABILITY CALCULATED FROM MEAN COMPONENT RELIABILITIES IS 0.903486; SYSTEM UNRELIABILITY IS 0.096514

AVERAGE SYSTEM RELIABILITY FROM 20 MONTE CARLO TRIALS IS 0.907255 ; AVERAGE SYSTEM UNRELIABILITY IS 0.092745

VARIANCE 0.000534

STANDARD DEVIATION 0.023110

THE MISSION TIME IS 100.000 HOURS

THE ESTIMATED SYSTEM MTBF BASED UPON MEAN COMPONENT RELIABILITIES IS 9.85600300E+02

THE ESTIMATED SYSTEM MTBF BASED UPON MEAN SYSTEM RELIABILITY 20 MONTE CARLO TRIALS IS 1.02741943E+03

| PERCENTILE | RELIABILITY PERCENTILE POINTS | MTBF PERCENTILE POINTS |
|---|---|---|
| 5.0 PERCENT | 0.853755 | 6.532793641E+02HOURS |
| 10.0 PERCENT | 0.878869 | 7.710778618+02HOURS |
| 20.0 PERCENT | 0.893825 | 8.917894922E+02HOURS |
| 25.0 PERCENT | 0.897440 | 9.241562594E+02HOURS |
| 50.0 PERCENT | 0.907436 | 1.030184532E+03HOURS |
| 75.0 PERCENT | 0.922629 | 1.241811094E+03HOURS |
| 80.0 PERCENT | 0.927674 | 1.332300738+03HOURS |
| 90.0 PERCENT | 0.935412 | 1.497703748+03HOURS |
| 95.0 PERCENT | 0.939504 | 1.605911758+03HOURS |
| 97.5 PERCENT | 0.939604 | 1.606361568+03HOURS |
| 99.0 PERCENT | 0.939612 | 1.606743338+03HOURS |

# APPENDIX B

## LIST OF REVISED SPARCS PROGRAM

```
SPARCS2 PROCEDURE OPTIONS(MAIN);
/*                                                                  */
/*                                                                  */
/*    SPARCS2 (SIMULATION PROGRAM FOR ASSESSING THE RELIABILITIES   */
/* OF COMPLEX SYSTEM) IS A SYSTEM OF PL/I PROCEDURES TO ASSESS      */
/* (PROVIDE INTERVAL ESTIMATES FOR) THE RELIABILITY AND MEAN TIME   */
/* BETWEEN FAILURES (MTBF) OF A COMPLEX SYSTEM OF ANY LOGICAL CON-  */
/* FIGURATION.  THE COMPONENTS CAN BE EITHER ATTRIBUTE OR TIME TO-  */
/* FAILURE(TTF), WITH NO RESTRICTION ON THEIR PLACEMENT.  BAYESIAN  */
/* AND MONTE CARLO TECHNIQUES ARE EMPLOYED.  INPUTS ARE MINIMAL     */
/* STATES FOR SYSTEM AND ITS MODULES AND COMPONENT FAILURE HISTORY  */
/* DATA.  MINIMAL STATES ARE MINIMAL PATH, FOR 'RELIABILITY'        */
/* ANALYSIS OR MINIMAL CUTS FOR 'UNRELIABILITY' ANALYSIS.  OUTPUT   */
/* IS A SCHEDULE OF SYSTEM RELIABILITY AND MTBF VALUES AND ASSO-    */
/* CIATED CONFIDENCE LEVELS.  MTBF ANALYSIS IS OPTIONAL.  FOR EACH  */
/* COMPONENT, THE RAW DATA ARE EXPERIENCED FAILURE HISTORIES, ACCU-  */
/* MULATED SUCCESSES AND FAILURES FOR ATTRIBUTES, EXPERIENCED TEST-  */
/* ING TIME (NORMALIZED TO MISSION EQUIVALENT UNITS) AND FAILURES    */
/* FOR TTF COMPONENTS.  THE SYMBOL OF COMPONENT IS NUMBERED          */
/* SUCCESSIVELY 1, 2, 3, ... , 128.  MODULES ARE TO BE SEQUENCED    */
/* ALPHANUMERICALLY A, B, C, ... , Z, A1, B1, C1, ... , Z1, A2, B2,  */
/* C2, ... ,Z2, A4, B4, ... , X4.                                   */
/*                                                                  */
/*                                                                  */
/* REFERENCES:                                                      */
/*                                                                  */
/*     LOCKS, M.O.,     MONTE CARLO BAYAISN SYSTEM RELIABILITY AND  */
/*                      MTBF CONFIDENCE ASSESSMENT, TECHNICAL REPORT*/
/*                      AFFDL-TR-75-1444, AIR FORCE FLIGHT DYNAMICS */
/*                      LABORATORY, AIR FOR SYSTEMS COMMAND, WRIGHT */
/*                      PATTERSON AIR FORCE BASE, OHIO, 1975.       */
/*                                                                  */
/*     LOCKS, M. O.,    RELIABILITY, MAINTAINABILITY, AND AVAILA-   */
/*                      BILITY ASSESMENT, HAYDEN BOOK CO., 1972.    */
/*                                                                  */
/*     LEE, K. K.,      REJECTION METHODS FOR GENERATING RANDOM     */
/*                      DEVIATES AND THEIR APPLICATIONS IN SYSTEM   */
/*                      RELIABILITY, MASTER'S THESIS, COMPUTING AND */
/*                      INFORMATION SCIENCES, OSU, JUNE, 1977.      */
/*                                                                  */
/*     COOLEY, J. W.,   SIMULATION PROGRAM FOR ASSESSING THE        */
/*                      RELIABILITY OF COMPLEX SYSTEMS (SPARCS),    */
/*                      PH.D. THESIS, OSU, 1976.                    */
/*                                                                  */
/*     CHANDLER, J. P.,RAND-UNIFORMLY DISTRIBUTED PSEUDO RANDOM     */
/*                      NUMBERS, UNPUBLISHED DOCUMENT, COMPUTING    */
/*                      AND INFORMATION SCIENCES, OSU, 1970.        */
/*                                                                  */
/*     BURRIS, J. L., MODEL FOR THE ANALYSIS OF PROBABILITIES       */
/*                     OF SYSTEMS (MAPS), MBA RESEARCH REPORT, OSU, */
/*                     1972.                                        */
/*                                                                  */
/*                                                                  */
/* PROCEDURES USED:                                                 */
/*                                                                  */
/*     ALGAMA          : COMPUTES LOG OF GAMMA FUNCTION.            */
/*     BETA            : GENERATES BETA DEVIATES.                   */
/*     COMPUTE         : CALCULATE MODULE OR SYSTEM RELIABILITY.    */
/*     DATAGEN         : GENERATES COMPONET RELIABILITY.            */
/*     EQGEN           : GENERATES BINARY VECTORS FOR EQUATION.     */
/*     EQPUT           : PRINTS PROBABILITY EQUATION.               */
/*     GAUSS           : GENERATES NORMAL DEVIATES.                 */
/*     HDLINE          : PRINTS INFORMATION FOR SYSTEM IDENTIFICATION*/
/*     HISINF          : PRINTS VALUE OF HISTORICAL COMPONENT DATA.  */
/*     INPUT1,2,3,4    : READ INPUT DATA IN.                        */
/*     MEANREL         : COMPUTE MEAN COMPONENT RELIABILITY FROM    */
/*                       HISTORICAL COMPONENT DATA.                 */
/*     RANF            : GENERATES UNIFORM DEVIATES.                */
/*     RGAMA           : GENERATES GAMMA DEVIATES.                  */
/*     SORT            : PERFORMS SHELL SORT ON SYSTEM RELIABILITIES.*/
/*     STAT            : COMPUTES AND PRINTS STATISTICAL INFORMATIONS*/
/*                                                                  */
```

```
/*
/* IMPORTNAT VARIABLES:                                                  */
/*                                                                       */
/*     A(*)            : FIRST PARAMETER FROM HISTORICAL DATA.           */
/*     B(*)            : SECOND PARAMETER FROM HISTORICAL DATA.          */
/*     ATYPE           : FLAG FOR RELIABILITY OR UNRELIABILITY PROCESS   */
/*     AVSM            : AVERAGE SYSTEM RELIABILITY.                     */
/*     CREL(*)         : COMPONENT RELIABILITIES.                        */
/*     COEF(*)         : COEFFICIENTS OF TERMS IN SYSTEM EQUATION.       */
/*     MINPTH(*)       : BINARY VECTORS FOR MINIMAL PATHS.               */
/*     NCOM            : NUMBER OF COMPONENTS IN SYSTEM.                 */
/*     NMOD            : NUMBER OF MODULES IN SYSTEM.                    */
/*     NPATH           : NUMBER OF MINIMAL STATES IN SYSTEM.             */
/*     NTERM           : NUMBER OF TERMS IN EQUATION.                    */
/*     SIMNUM          : NUMBER OF SIMULATIONS DESIRED.                  */
/*     STMREL          : SYSTEM RELIABILITY.                             */
/*     SYSID           : SYSTEM IDENTIFICATION.                          */
/*     SYSVEC          : SYSTEM RELIABILITIES VECTOR.                    */
/*     TERMS           : BINARY  VECTOR OF TERM IN EQUATION.             */
/*     TIME            : MISSION TIME.                                   */
/*     UNIT            : TIME UNIT SUCH AS SECOND, MINUTE, OR HOUR.      */
/*                                                                       */
/* INPUT DATA FORMAT:                                                    */
/*                                                                       */
/*                      -- SYSTEM WITHOUT MODULE --                      */
/*                                                                       */
/* DATA     NUMBER    COLUMN                  DESCRIPTIONS              */
/* SET     OF CARDS  POSITION                                            */
/*                                                                       */
/*                                                                       */
/* FIRST       1       1 - 80    ALPHANUMERIC SYSTEM IDENTIFICATION.     */
/* SECOND      1       FREE      1. NUMBER OF MONTE CARLO TRIALS.        */
/*                     FORMAT    2. SYSTEM MISSION TIME.                 */
/* THIRD               FREE      1. NUMERIC ZERO DENOTES NO MODULE.      */
/*                     FORMAT    2. NUMBER OF COMPONENTS IN SYSTEM.      */
/*                               3. NUMBER OF SYSTEM MINIMAL STATES.     */
/*                               4. 'R' FOR RELIABILITY, 'U' FOR         */
/*                                  UNRELIABILITY ANALYSIS.              */
/*                               5. MISSION TIME UNIT, OPTIONAL.         */
/*                                  (MAXIMUM 10 CHARACTERS)              */
/* FOURTH      N       FREE      1. '0'B FOR GAMMA (TIME TO              */
/*                     FORMAT       FAILURE) OR '1'B FOR BETA            */
/*                                  (SUCCESS-FAILURE).                   */
/*                               2. NUMBER OF MISSIONS (GAMMA) OR        */
/*                                  SUCCESSES (BETA).                    */
/*                               3. NUMBER OF FAILURES.                  */
/* FIFTH               FREE      M MINIMAL STATES AS A STRING OF         */
/*                     FORMAT    N VECTORS SUCH AS '00110'B.             */
/*                               COMPONENTS IN THE MINIMAL STATE ARE     */
/*                               DENOTED BY "1", COMPONENTS NOT IN       */
/*                               MINIMAL STATE BY "0".                   */
/*                                                                       */
/*                                                                       */
/*                     INPUT DATA SETUP EXAMPLE                          */
/*                                                                       */
/*                                                                       */
/*       COLUMN      00000000011111111112222222222333333333344          */
/*       POSITION    12345678901234567890123456789012345678901          */
/*                                                                       */
/*       CARD    1   TWO COMPONENT AND TWO MINIMAL PATH SYSTEM         */
/*               2     100     100                                     */
/*               3     0   2     2     'R'   ' MINUTES'                */
/*               4   '0'B    99     1                                  */
/*               5   '1'B    60     2                                  */
/*               6   '10'B   '01'B                                     */
/*                                                                       */
```

```
/*                                                                    */
/* PROCEDURES USED:                                                   */
/*                                                                    */
/*     ALGAMA          : COMPUTES LOG OF GAMMA FUNCTION.              */
/*     CBETA           : GENERATES BETA DEVIATES.                     */
/*     COMPUTE         : CALCULATE MODULE OR SYSTEM RELIABILITY.      */
/*     DATAGEN         : GENERATES COMPONET RELIABILITY.              */
/*     EQGEN           : GENERATES BINARY VECTORS FOR EQUATION.       */
/*     EQPUT           : PRINTS PROBABILITY EQUATION.                 */
/*     GAUSF           : GENERATES NORMAL DEVIATES.                   */
/*     HDLINE          : PRINTS INFORMATION FOR SYSTEM IDENTIFICATION */
/*     HISINF          : PRINTS VALUE OF HISTORICAL COMPONENT DATA.   */
/*     INPUT1,2,3,4    : READ INPUT DATA IN.                          */
/*     MEANREL         : COMPUTE MEAN COMPONENT RELIABILITY FROM      */
/*                       HISTORICAL COMPONENT DATA.                   */
/*     RANF            : GENERATES UNIFORM DEVIATES.                  */
/*     RGAMA           : GENERATES GAMMA DEVIATES.                    */
/*     SORT            : PERFORMS SHELL SORT ON SYSTEM RELIABILITIES. */
/*     STAT            : COMPUTES AND PRINTS STATISTICAL INFORMATIONS */
/*                                                                    */
/*                     -- SYSTEM WITH MODULES --                      */
/*                                                                    */
/*                                                                    */
/* DATA   NUMBER    COLUMN              DESCRIPTIONS                  */
/* SET   OF CARDS  POSITION                                           */
/*                                                                    */
/* FIRST     1       1 - 80     ALPHANUMERIC SYSTEM IDENTIFICATION.   */
/* SECOND            FREE       1. NUMBER OF MONTE CARLO TREALS.      */
/*                   FORMAT     2. SYSTEM MISSION TIME(OPTIONAL).     */
/* THIRD             FREE       1. NUMBER OF MODULES IN SYSTEM.       */
/*                   FORMAT     2. NUMBER OF MODULES IN SYSTEM.       */
/*                              3. NUMBER OF MINIMAL STATES IN SYSTEM */
/*                              4. 'R' FOR SYSTEM RELIABILITY, 'U'    */
/*                                 FOR SYSTEM UNRELIABILITY.          */
/*                              5. MISSION TIME UNITS  (OPTIONAL).    */
/* FOURTH            FREE       J SYSTEM MINIMAL STATES AS A STRING   */
/*                   FORMAT     BINARY IN MINIMAL STATE ARE DNOTED    */
/*                              BY "1". MODULES NOT IN MINIMAL STATE  */
/*                              BY "0".                               */
/* FIFTH             FREE       1. 'R' FOR MODULE RELIABILITY, 'U'    */
/*                   FORMAT        FOR MODULE UNRELIABILITY.          */
/*                              2. MODULE IDENTIFICATION, 2 CHAR. MAX */
/*                              3. NUMBER OF COMPONENT IN MODULE.     */
/*                              4. NO. OF MINIMAL STATES IN MODULE.   */
/* SIXTH     M       FREE       1. '0'B FOR GAMMA(TIME-TO-FAILURE)    */
/*                   FORMAT        OR '1'B FOR BETA (SUCCESS-FAILURE) */
/*                              2. NUMBER OF MISSIONS(GAMMA) OR       */
/*                                 SUCCESSESS(BETA).                  */
/*                              3. NUMBER OF FAILURES.                */
/* SEVENTH           FREE       M MODULE MINIMAL STATES AS A STRING   */
/*                   FORMAT     OF BINARY N VECTORS SUCH AS '01110'B. */
/*                              COMPONENTS IN MINIMAL STATE ARE       */
/*                              DENOTED BY 1, COMPONENTS NOT IN       */
/*                              MINIMAL STAE BY 0.                    */
/*                                                                    */
/* ** THE FIFTH, SIXTH, AND SEVENTH DATA SETS ARE REPEATED FOR        */
/*    EACH MODULE.                                                    */
/*                                                                    */
/*                    INPUT DATA SETUP EXAMPLE                        */
/*                                                                    */
/*                                                                    */
/*       COLUMN     00000000011111111112222222223333333333444444    */
/*       POSITION   12345678901234567890123456789012345678901234    */
/*                                                                    */
/*       CARD    1  TWO-MODULE AND TWO-COMPONENT SYSTEM ANALYSIS     */
/*               2  20  40                                            */
/*               3    2   0     2       'R'     ' HOURS'            */
/*               4  '01'B  '10'B                                      */
/*               5  'U'   'A'  2    2                                 */
/*               6  '1'B     95.     5.                             */
/*               7  '0'B    120.     4.                             */
/*               8  '01'B  '10'B                                      */
/*               9  'R'   'B'  2    2                                 */
/*              10  '0'B    120.     6.                             */
/*              11  '1'B     85.     2.                             */
/*              12  '10'B  '01'B                                     */
/*                                                                    */
/*                                                                    */
/* LANGUAGE USED   :   PL/I                                           */
/* PRECISION       :   SINGLE                                         */
/* PROGRAMMER      :   K K LEE                                        */
/* DATE PROGRAMMED :   JULY, 1977                                     */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
```

```
/*-------------------------------------------------------------------*/
/*                          MAIN PROCEDURE                           */
/*-------------------------------------------------------------------*/
/*                                                                   */
DCL TIME FLOAT DEC(6),ATYPE CHAR(1), SYSID CHAR(80), UNIT CHAR(10);
DCL (NCOM,NMOD,NPATH,SIMNUM,NCOMS,NSTAT) FIXED BIN(15);
    OPEN FILE(SYSPRINT) PAGESIZE(50);
    ON ENDPAGE(SYSPRINT) PUT FILE(SYSPRINT) PAGE;
AGN:ON ENDFILE (SYSIN) GO TO FINIS;
    CALL  INPUT1 (ATYPE,NCOM,NMOD,NPATH,SIMNUM,SYSID,TIME,UNIT);
IF NMOD=0 THEN  BEGIN;
/*-------------------------------------------------------------------*/
/*              PROCESS FOR SYSTEM WITHOUT MODULES                   */
/*-------------------------------------------------------------------*/
DCL (A(NCOM),AVSM,B(NCOM),CREL(NCOM),STMREL,SYSVEC(SIMNUM),STMCR,
    TOT,TOT2) FLOAT DEC(6),COEF(5000) FIXED BIN(4), IS FIXED BIN(15),
    EQID CHAR(6),MINPTH(NPATH) BIT(NCOM) VAR,NTERM FIXED BIN(15),
    TERMS(5000) BIT(NCOM) VAR,TYPE(NCOM) BIT(1);
    AVSM=0.;  EQID='SYS   ';  TOT = 0.;  TOT2 = 0.;
CALL INPUT2 (NCOM,NPATH,TYPE,A,B,MINPTH);
CALL MEANREL (NCOM,A,B,TYPE,CREL);
CALL HDLINE (SYSID,NMOD,NCOM,NPATH,ATYPE,SIMNUM);
CALL HISINF (NCOM,A,B,TYPE,CREL,EQID);
CALL EQGEN (COEF,MINPTH,NCOM,NPATH, NTERM,TERMS);
CALL EQPUT (ATYPE,COEF,MINPTH,NCOM,NPATH,NTERM,TERMS,EQID,NMOD);
CALL COMPUTE (CREL,COEF,NCOM,NTERM,TERMS,STMREL);
   PUT FILE(SYSPRINT) EDIT ('SPARCS :: SYSTEM SIMULATION ROUTINE')
                           (SKIP(3),A);
   IF ATYPE = 'U' THEN STMREL=1.-STMREL;
   STMCR=STMREL;
   PUT FILE(SYSPRINT) EDIT (' SYSTEM RELIABILITY CALCULATED FROM ',
        'MEAN COMPONENT RELIABILITIES IS ',STMREL,'; SYSTEM ',
        'UNRELIABILITY IS ',1.-STMREL)(SKIP(5),A,A,F(8,6),A,A,F(8,6));

DO IS = 1 TO SIMNUM;
   CALL DATAGEN (A,B,NCOM,TYPE,CREL);
                                                        /*           */
   CALL COMPUTE (CREL,COEF,NCOM,NTERM,TERMS,STMREL);
   IF ATYPE = 'U' THEN STMREL=1.-STMREL;
   SYSVEC(IS)=STMREL;
   AVSM = AVSM + STMREL / SIMNUM;
   TOT = TOT + STMREL;
   TOT2 = TOT2 + STMREL*STMREL;
END;
                                                  /* END OF SIMULATION */
   CALL SORT (SIMNUM,SYSVEC);
   CALL STAT(SIMNUM,SYSVEC,AVSM,ATYPE,TIME,UNIT,TOT,TOT2,STMCR);
   PUT FILE(SYSPRINT) PAGE;
END; /* BEGIN FOR NON MODULE SYSTEM PROCESS */
/*-------------------------------------------------------------------*/
/*              PROCESS FOR SYSTEM WITH MODULES                      */
/*-------------------------------------------------------------------*/
ELSE DO;
        NCOMS=NCOM;
        NSTAT=NPATH;
BEGIN;
   DCL  NSTRM FIXED BIN(15),SMINP(NSTAT) BIT(NCOMS) VAR,
        SOLE(1000) FIXED BIN(4),STERM(1000) BIT(NCOMS) VAR;
   DCL  (AVSM,TOT,TOT2,MODREL(NMOD,SIMNUM),RELMOD(NMOD),SYSVEC(SIMNUM),
        RMDL(NMOD),STMCR) FLOAT DEC(6), EQID CHAR(6),(MSTAT,L,NCO)
                                                  FIXED BIN(15);
        AVSM=0.;  EQID='SYS   ';  TOT = 0.;  TOT2 = 0.;
   CALL INPUT3 (SMINP,NSTAT);
   CALL HDLINE (SYSID,NMOD,NCOMS,NSTAT,ATYPE,SIMNUM);
   CALL EQGEN (SOLE,SMINP,NCOMS,NSTAT,NSTRM,STERM);
   CALL EQPUT (ATYPE,SOLE,SMINP,NCOMS,NSTAT,NSTRM,STERM,EQID,NMOD);
DO L=1 TO NMOD BY 1;
   CALL INPUT4 (ATYPE,EQID,NCO ,MSTAT);
   /*                                                             */
  BEGIN; /* INNER */
   DCL (A(NCO),B(NCO),CREL(NCO),MODREL,STMREL)FLOAT DEC(6);
   DCL COEF(5000) FIXED BIN(4),MINPTH(MSTAT) BIT(NCO) VAR,TYPE(NCO)
       BIT(1),(IS,NTERM) FIXED BIN(15),TERMS(5000) BIT(NCO) VAR;
   CALL INPUT5 (A,B,MINPTH,TYPE,NCO,MSTAT);
   CALL EQGEN (COEF,MINPTH,NCO ,MSTAT,NTERM,TERMS);
   /*                                                             */
   CALL EQPUT (ATYPE,COEF,MINPTH,NCO ,MSTAT,NTERM,TERMS,EQID,NMOD);
   CALL MEANREL (NCO ,A,B,TYPE,CREL);
   CALL COMPUTE (CREL,COEF,NCO ,NTERM,TERMS,RMDL(L));
   /*                                                             */
```

```
      CALL DISINF (NCO ,A,B,TYPE,CREL,EQID);
      PUT FILE(SYSPRINT) SKIP(5);
      PUT FILE(SYSPRINT) EDIT ('SPARCS :: SYSTEM SIMULATION ROUTINE')
          (SKIP(3),A);
      DO IS = 1 TO STANUM;
        CALL DATAGEN (A,B,NCO ,TYPE,CREL);
        CALL COMPUTE (CREL,COEF,NCO ,NTERM,TERMS,MODREL);
        MDRL(L,IS)=MODREL;
      END;
          /*                                                          */
    END; /* INNER BEGIN */
   END,  /* NMOD */
     CALL COMPUTE (NMOD ,SOEF,NCOMS,NSTRM,STERM,STMREL);
     IF ATYPE = 'U' THEN STMREL=1.-STMREL;
     PUT FILE(SYSPRINT) EDIT (' SYSTEM RELIABILITY CALCULATED FROM ',
          'MEAN COMPONENT RELIABILITIES IS ',STMREL,'; SYSTEM ',
          'UNRELIABILITY IS ',1.-STMREL)
          (SKIP(5),A,A,F(8,6),A,A,F(8,6));
     STMCR=STMREL;
    DO IS=1 TO SIMNUM;
     DO L=1 TO NMOD;
      RELMOD(L)=MDRL(L,IS);
     END; /* NMOD */
     CALL COMPUTE (RELMOD,SOEF,NCOMS,NSTRM,STERM,STMREL);
     IF ATYPE = 'U' THEN STMREL=1.-STMREL;
     SYSVEC(IS)=STMREL;
     AVSM = AVSM + STMREL / SIMNUM;
     TOT = TOT + STMREL;
     TOT2 = TOT2 + STMREL*STMREL;
   END; /* SIMNUM */
     CALL SORT (SIMNUM,SYSVEC);
     CALL STAT(SIMNUM,SYSVEC,AVSM,ATYPE,TIME,UNIT,TOT,TOT2,STMCR);
     PUT FILE(SYSPRINT) PAGE;
 END;/* ELSE BEGIN */
END; /* ELSE DO */
GO TO AGN;
/*-------------------------------------------------------------------*/
/*                        END OF MAIN PROCEDURE                      */
/*-------------------------------------------------------------------*/
INPUT1: PROC(ATYPE,NCOM,NMOD,NPATH,SIMNUM,SYSID,TIME,UNIT);
/*                                                                   */
/*-------------------------------------------------------------------*/
/*      THE INPUT1 PROCEDURE READS IN DATA FOR THE SYSTEM            */
/* IDENTIFICATION.                                                   */
/*-------------------------------------------------------------------*/
/*                                                                   */
    DCL (NCOM,NMOD,NPATH,SIMNUM                          ) FIXED BIN(15);
    DCL (TIME                                            ) FLOAT DEC(6);

    DCL ATYPE CHAR (*),UNIT CHAR(*);
    DCL SYSID CHAR (*);
       GET FILE (SYSIN) EDIT (SYSID) (COL(1),A(80));
       GET FILE (SYSIN) LIST (SIMNUM,TIME);
       GET FILE (SYSIN) LIST (NMOD,NCOM,NPATH,ATYPE,UNIT);
    RETURN;
    END  INPUT1;
/*                                                                   */
/*-------------------------------------------------------------------*/
/*      INPUT2 PROCEDURE READS IN HISTORICAL DATA FRO COMPONENTS     */
/* AND MINIMAL STATES IN BINARY REPRESENTATION FOR COMPUTATION       */
/* OF SYSTEM RELIABILITY WITHOUT MODULES.                            */
/*-------------------------------------------------------------------*/
/*                                                                   */
INPUT2: PROC (NCOM,NPATH,TYPE,A,B,MINPTH);
    DCL (NCOM,NPATH,I                                    ) FIXED BIN(15);
    DCL (A(*),B(*)                                       ) FLOAT DEC (6);
    DCL MINPTH(*) BIT  (*) VAR , TYPE(*) BIT(*);
     GET FILE(SYSIN) LIST ((TYPE(I),A(I),B(I) DO I=1 TO NCOM));
     GET FILE (SYSIN) LIST ((MINPTH(I) DO I=1 TO NPATH));
    RETURN;
    END  INPUT2;
INPUT3: PROC (SMINP,NSTAT);
/*                                                                   */
/*-------------------------------------------------------------------*/
/*      INPUT3 READS IN THE MINIMAL STATES (SMINP) FRO SYSTEM WHICH  */
/* HAS THE MODULES IN IT.                                            */
/*-------------------------------------------------------------------*/
/*                                                                   */
    DCL SMINP(*) BIT(*) VAR, NSTAT FIXED BIN(15);
    DCL (I                       ) FIXED BIN(15);
     PUT FILE(SYSPRINT) LIST (NSTAT);
     GET FILE (SYSIN) LIST ((SMINP(I) DO I=1 TO NSTAT));
    RETURN;
    END; /* INPUT3 */
                                                                     */
```

```
/*                                                                    */
/*--------------------------------------------------------------------*/
/*      INPUT4 PROCEDURE READS THE DATA FOR MODULE IDENTIFICATION.    */
/*--------------------------------------------------------------------*/
/*                                                                    */
INPUT4: PROC (ATYP,EQID,NCOM,STATES);
     DCL ATYP CHAR(*),EQID CHAR(*),(NCOM,STATES) FIXED BIN(15);
      GET FILE (SYSIN) LIST (ATYP,EQID,NCOM,STATES);
     RETURN;
     END; /* INPUT4 */
INPUT5: PROC (A,B,MINPTH,TYPE,NCOM,STATES);
/*                                                                    */
/*--------------------------------------------------------------------*/
/*      INPUT5 PROCEDURE READS IN HISTORICAL DATA FOR COMPONENTS      */
/* AND MNIMAL STATES FOR COMPUTATION OF EACH MODULE RELIABILITIES.    */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL (NCOM,STATES,I                                ) FIXED BIN(15);
     DCL (A(*),B(*)) FLOAT DEC(6),MINPTH(*) BIT(*) VAR,TYPE(*) BIT(*);

      GET FILE (SYSIN) LIST ((TYPE(I),A(I),B(I)  DO I=1 TO NCOM));
      GET FILE (SYSIN) LIST ((MINPTH(I) DO I=1 TO STATES));
     RETURN;
     END INPUT5;
MEANREL: PROC (NCOM,A,B,TYPE,ACR);
/*--------------------------------------------------------------------*/
/*      A COMPUTATION PROCEDURE FOR MEAN RELIABILITY OF COMPONENT     */
/* BASED UPON THE HISTORICAL DATA OF EACH COMPONENT RELIABILITY.      */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL (NCOM,I) FIXED BIN(15),(A(*),B(*),ACR(*)) FLOAT DEC(6),
         TYPE(*) BIT(*);
      DO I=1 TO NCOM;
      IF TYPE(I)='1'B THEN ACR(I)=(A(I)+1.)/(A(I)+B(I)+2.);
                      ELSE ACR(I)=(A(I)/(A(I)+2.))**FIXED(B(I)+1.);
      END;
     RETURN;
     END MEANREL;
HDLINE:PROC(SYSID,NMOD,NCOM,NPATH,ATYPE,STMNUM);
/*                                                                    */
/*--------------------------------------------------------------------*/
/* HDLINE PRINTS MAJOR HEADINGS FOR KEY VARIABLES AND SYSTEM ID.      */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL SYSID CHAR (*), ATYPE CHAR (*), STA CHAR(4) VAR;
     DCL (NCOM,NMOD,NPATH,STMNUM,NC    ) FIXED BIN(15);
     IF ATYPE='R' THEN STA='PATH';
                  ELSE STA='CUT ';
     PUT FILE(SYSPRINT) EDIT ('* S I M A R C S *')
         (PAGE,A) ('SIMULATION PROGRAM FOR THE ANALYSIS OF THE ',
         'RELIABILITY OF COMPLEX SYSTEM') (SKIP(1),A,A) ('SYSTEM ',
         'IDENTIFICATION',SYSID) (SKIP(2),A,A,X(13),A)
         ('NUMBER OF STMULATIONS ',STMNUM) (SKIP(1),A,X(13),F(3))
         ('NUMBER OF MODULES ',NMOD) (SKIP(1),A,X(17),F(3))
         ('NUMBER OF COMPONENTS ',NCOM) (SKIP(1),A,X(14),F(3))
         ('NUMBER OF MINIMAL ',STA,NPATH)   (SKIP(1),A,A,X(13),F(3));
     IF ATYPE = 'R' THEN DO;
      PUT FILE(SYSPRINT) EDIT ('TYPE OF ANALYSIS','RELIABILITY')
                              (SKIP(1),A,X(19),A);
     END;
     ELSE DO;
      PUT FILE(SYSPRINT) EDIT ('TYPE OF ANALYSIS','UNRELIABILITY')
                              (SKIP(1),A,X(19),A);
     END;
     RETURN;
     END;  /* HDLINE */
HISINF: PROC (NCOM,A,B,TYPE,ACR,EQID);
/*                                                                    */
/*--------------------------------------------------------------------*/
/*      HISINF PROCEDURE PRINTS THE HISTORICAL INFORMATIONS FOR       */
/* COMPONENT EITHER IN A SYSTEM OR IN MODULES.                        */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL (A(*),B(*),ACR(*)                                ) FLOAT DEC(6),
     DCL TYPE(*) BIT(*),EQID CHAR(*),ID CHAR(13) INIT ('') VAR;
     DCL (NCOM,NC                                         ) FIXED BIN(15);
```

```
     IF EQID = 'SYS    ' THEN ID='SYSTEM';
                      ELSE ID='MODULE '||EQID;
     PUT FILE(SYSPRINT) EDIT ('HISTORICAL INFORMATIONS FOR EACH ',
          'COMPONENT IN ',ID)(SKIP(5),A,A,A)('COMPONENT','  TYPE   ',
          'SUCCESS (MEAN)','    FAILURES    ','    MEANS        ')
          (SKIP(2),A,X(7),A,X(13),A,X(7),A,X(6),A)
          ('EQUIVALENT MISSIONS(GAMMA)')(SKIP(1),COL(35),A);
    DO NC=1 TO NCOM;
     IF TYPE(NC)='1'B THEN DO;
     PUT FILE(SYSPRINT) EDIT (NC,'ATTRIBUTE(BETA)',A(NC),B(NC),ACR(NC))
          (SKIP(2),X(3),F(3),X(7),A,X(12),F(8,2),X(13),F(8,2),X(15),
          F(6,4));     END;
     ELSE DO;
     PUT FILE(SYSPRINT) EDIT (NC,'TIME-TO-FAILURE(GAMMA)',A(NC),B(NC),
          ACR(NC)) (SKIP(2),X(3),F(3),X(7),A,X(5),F(8,2),X(13),F(8,2),
          X(13),F(8,4));   END;
    END;
     RETURN;
     END; /*HISTNF*/
EQGEN:PROC(COEF,MINPTH,NCOM,NPATH,NTERM ,TERMS);
/*
/*-------------------------------------------------------------------------*/
/*      THE PROBABILITY EQUATION, ENTIRELY GENERATED BY THIS EQGEN         */
/* PROCEDURE, USES POINCARE'S METHOD (INCLUSION-EXCLUSION), FOR A          */
/* SYSTEM HAVING N MINIMAL STATES, PROBABILITY EWUATION HAS A MAXIMUM      */
/* OF 2**N - 1 TERMS. HOWEVER, THE EQUATION GENERATED BY EQGEN PROC        */
/* HAS ONLY A FRACTION OF DUPLICATE TERMS AFTER EACH MINIMAL STATE         */
/* IS INTRODUCED AND COMPINED WITH PREVIOUSLY GENERATED TERMS TO FORM      */
/* POSSIBLE NEW TERMS. SIMULTANEOUSLY, THE TERMS THAT HAVE ZERO CO-        */
/* EFFICIENTS ARE REMOVED BEFORE THE NEXT MINIMAL STATE IS INTRO-          */
/* DUCED.                                                                  */
/*-------------------------------------------------------------------------*/
/*                                                                         */
        DCL COEF (*) FIXED BIN( 4);
        DCL MINPTH (*) BIT (*) VAR;
        DCL(I1,I2,I3,I4,I5,I6,I7,KK1,KDUP,NSUB,NDUP,INC2)FIXED BIN(15);
        DCL (NCOM, NPATH,NTERM                          ) FIXED BIN(15);
        DCL TERMS (*) BIT (*) VAR;
        TERMS(1)=MINPTH(1);COEF(1)=1; /* 1ST 3 TERMS OF PROB. EQUA.  */
        IF NPATH=1 THEN DO;
             NTERM=1;  RETURN;  END;
        TERMS(2)=MINPTH(2); COEF(2)=1;
        TERMS(3)=MINPTH(1) | MINPTH(2);
        COEF(3)=-1;
        NTERM=3;
        IF NPATH=2 THEN GO TO ENDEQ;
        NSUB=4;
        DO I1 = 3 TO NPATH;              /* DO LOOP1,  REMAINING TERMS */
        TERMS(NSUB)=MINPTH(I1);
        COEF(NSUB)=1;
        NSUB=NSUB+1;
        DO I2 = 1 TO NTERM;              /* DO LOOP2 */
        TERMS(NSUB)=MINPTH(I1) | TERMS(I2);
        COEF(NSUB)=-COEF(I2);            /* DETERNINE COEFFICIENT */
        NSUB=NSUB+1;
        END;                             /* END LOOP2 */
        NDUP=0;                          /*ACCUMULATE DUPLICATE TERMS*/
        INC2=NSUB-1;
        DO I3=NTERM+2 TO INC2;
        DO I4=3 TO I3-1-NDUP;
        IF TERMS(I4)^=TERMS(I3-NDUP) THEN GO TO ENDI4;
        COEF(I4)=COEF(I4) + COEF(I3-NDUP);
        IF I3-NDUP=NSUB-1 THEN GO TO SUB;
             DO I5=I3-NDUP TO INC2-1-NDUP;
             TERMS(I5)=TERMS(I5+1);
             COEF(I5)=COEF(I5+1);
             END;
   SUB: NSUB=NSUB-1;
        NDUP=NDUP+1;
        GO TO ENDI3;
 ENDI4: END;
 ENDI3: END;
        KDUP=0;         /* REMOVE TERMS WITH ZERO COEFFICIENTS */
        KK1=NSUB-1;
        DO I6=3 TO KK1;
        IF COEF(I6-KDUP)^=0 THEN GO TO ENDI6;
        IF I6-KDUP=NSUB-1 THEN GO TO KSUB;
             DO I7=I6-KDUP TO KK1-1-KDUP;
             TERMS(I7)=TERMS(I7+1);
             COEF(I7)=COEF(I7+1);
             END;
  KSUB: NSUB=NSUB-1;
        KDUP=KDUP+1;
 ENDI6: END;
        NTERM=NSUB-1;
        END;                             /* END LOOP1 */
```

```
ENDEQ: END EQGEN;
EQPUT: PROC (ATYPE,COEF,MINPTH,NCOM,NPATH,NTERM,TERMS,EQID,NMOD);
/*                                                                    */
/*--------------------------------------------------------------------*/
/* PRINT OUT THE SYSTEM RELIABILITY EQUATION BY USING THE BINARY      */
/* VECTOR TERMS(*) AND SIGN VECTOR OF COEF(*) FROM THE FUNCTION       */
/* PROCEDURE EQGEN.                                                   */
/*--------------------------------------------------------------------*/
    DCL (K4,K5,K6,K7,LEN,LEN6,NCOM,NMIN,NTERM,NPATH,NM) FIXED BIN(15);
    DCL SIGN CHAR(3) VAR, STATE CHAR(4) VAR,NMOD FIXED BIN(15);
    DCL (CHAR1,CHAR2,MINP                              ) CHAR(120) VAR;
    DCL MINPTH(*) BIT(*) VAR, TERMS (*) BIT (*) VAR;
    DCL COEF (*) FIXED BIN( 4), ATYPE CHAR(*),EQID CHAR(*);
    DCL C0 CHAR(1), C1 CHAR(2), C2 CHAR(3), C3 CHAR(4);
    DCL KOMPS(128) CHAR(3) VARYING INITIAL('1','2','3',
        '4','5','6','7','8','9','10','11','12','13','14','15','16',
        '17','18','19','20','21','22','23','24','25','26','27',
        '28','29','30','31','32','33','34','35','36','37','38',
        '39','40','41','42','43','44','45','46','47','48','49',
        '50','51','52','53','54','55','56','57','58','59','60',
        '61','62','63','64','65','66','67','68','69','70','71',
        '72','73','74','75','76','77','78','79','80','81','82',
        '83','84','85','86','87','88','89','90','91','92','93',
        '94','95','96','97','98','99','100','101','102','103',
        '104','105','106','107','108','109','110','111','112',
        '113','114','115','116','117','118','119','120','121',
        '122','123','124','125','126','127','128') STATIC;
    DCL MODSY(128) CHAR(3) VARYING INIT('A','B','C',
        'D','E','F','G','H','I','J','K','L','M','N','O','P','Q',
        'R','S','T','U','V','W','X','Y','Z','A1','B1','C1','D1',
        'E1','F1','G1','H1','I1','J1','K1','L1','M1','N1','O1',
        'P1','Q1','R1','S1','T1','U1','V1','W1','X1','Y1','Z1',
        'A2','B2','C2','D2','E2','F2','G2','H2','I2','J2','K2',
        'L2','M2','N2','O2','P2','Q2','R2','S2','T2','U2','V2',
        'W2','X2','Y2','Z2','A3','B3','C3','D3','E3','F3','G3',
        'H3','I3','J3','K3','L3','M3','N3','O3','P3','Q3','R3',
        'S3','T3','U3','V3','W3','X3','Y3','Z3','A4','B4','C4',
        'D4','E4','F4','G4','H4','I4','J4','K4','L4','M4','N4',
        'O4','P4','Q4','R4','S4','T4','U4','V4','W4','X4') STATIC;
    IF ATYPE = 'R' THEN DO;
                        STATE = 'PATH'; C0='R'; C1='R ';
                        C2='R  '; C3='R   ';
    END;
    ELSE DO;
         STATE = 'CUT'; C0='U'; C1='U '; C2='U  '; C3='U   ';
    END;
    IF EQID = 'SYS    ' THEN PUT FILE(SYSPRINT) EDIT
                          ('SPARCS :: EQUATION GENERATION ROUTINE')
                          (SKIP(5),A);
    IF ^(EQID='SYS    ') THEN PUT FILE(SYSPRINT) EDIT ('THE ',NPATH,
      ' MINIMAL ',STATE,' FOR THE ','MODULE ',EQID)(SKIP(3),A,F(3),
       A,A,A,A,A);
    ELSE   PUT FILE(SYSPRINT) EDIT('THE ',NPATH,' MINIMAL ',STATE,
         ' FOR THE'  ,' SYSTEM ') (SKIP(3),A,F(3),A,A,A,A,SKIP(2));
    PUT FILE(SYSPRINT) SKIP(2);
    /*                                                                */
L5: DO K4=1 TO NPATH;
    MINP='';
    MINP='(';
L6: DO K5=1 TO NCOM;  /* OUTPUT FOR MINIMAL STATES */
    IF LENGTH(MINP) > 110 THEN DO;    /* WHEN LINESIZE > 110  */
                                PUT FILE(SYSPRINT) EDIT(MINP)(SKIP
                                    (1),COL(1),A);
                                MINP='';
                              END;
    IF SUBSTR(MINPTH(K4),K5,1)='1'B THEN
       IF EQID='SYS    ' & NMOD ^= 0 THEN MINP=MINP||MODSY(K5)||',';
                                     ELSE MINP=MINP||KOMPS(K5)||',';
    END;  /* DO : L6 */
    NMIN=LENGTH(MINP);
    SUBSTR(MINP,NMIN,1)=')';
    PUT FILE (SYSPRINT) EDIT (MINP)  (SKIP(1),COL(1),A);
    END;  /* DO L5 */
       CHAR1='';  /* DETERMINATION OF COMPONENT SYMBOLS FOR OUTPUT */
       CHAR2=C0;
       CHAR1=CHAR1||EQID;
       CHAR2=CHAR2|| '   = ';
    L7:DO K6=1 TO NTERM;
       IF COEF(K6) > 0 THEN  SIGN=' + ';
                       ELSE  SIGN = ' - ';
       IF K6 = 1 THEN GO TO AG;
       CHAR2 = CHAR2 || SIGN;
       CHAR1 = CHAR1 || '   ';
       IF ABS(COEF(K6)) = 1 THEN GOTO AG;        /* CASE WHEN        */
       CHAR2=CHAR2 || KOMPS(ABS(COEF(K6)));      /* COEFFICIENT > 1  */
       CHAR1=CHAR1 || ' ';
```

```
     AG: DO K7 = 1 TO NCOM;
         IF SUBSTR(TERMS(K6),K7,1)  =  '0'B THEN GO TO L50;
         IF EQ10='SYS    ' & NMOD ^= 0 THEN DO;
            CHAR1=CHAR1||MODSY(K7)||' ';
            LEN6=LENGTH(KOMPS(K7));
         END;
         ELSE DO;
            CHAR1=CHAR1||KOMPS(K7)  ||' ';
            LEN6=LENGTH(KOMPS(K7));
         END;
         IF LEN6=1 THEN CHAR2=CHAR2||C1;
                   ELSE IF LEN6=2 THEN CHAR2=CHAR2||C2;
                                  ELSE CHAR2=CHAR2||C3;
         LEN=LENGTH(CHAR2);
         IF LEN)112 & LEN (120 THEN DO;
           IF K7+1=NCOM & ^SUBSTR(TERMS(K6),K7+1,1) THEN GOTO L8;
           IF K7 ^= NCOM THEN CHAR2=CHAR2||'*';
        L8:PUT FILE (SYSPRINT) EDIT (CHAR2) (SKIP(2),COL(3),A);
           PUT FILE (SYSPRINT) EDIT (CHAR1) (COL(4),A);
           LEN=0;
           CHAR1='';
           CHAR2='';
         END;
         ELSE DO;
              GO TO L50;
         END;
    L50:END; /* DO:AG */
         END; /* DO:L7 */
         PUT FILE (SYSPRINT) EDIT (CHAR2) (SKIP(2),COL(3),A);
         PUT FILE (SYSPRINT) EDIT (CHAR1) (COL(4),A);
         RETURN;
         END;/* EQPUT */
COMPUTE: PROC(REL,COEF,NCOM,NTERM,TERMS,SYSREL);
/*                                                                    */
/*--------------------------------------------------------------------*/
/*       PORCEDURE COMPUTE CALCULATES VALUES OF RELIABILITY OF EACH   */
/* SYSTEM BY USING PROBABILITY EQUATION GENERATED BY PROCEDURE EQGEN.*/
/*--------------------------------------------------------------------*/
/*                                                                    */
         DCL (NCOM,NTERM,KA,KB                          ) FIXED BIN(15),
         DCL (SYSREL,REL(*)                             ) FLOAT DEC( 6),
         DCL (BROD,DREL,DSYSREL                         ) FLOAT DEC(16);
         DCL TERMS(*) BIT (*) VAR;
         DCL COEF (*) FIXED BIN( 4);
         DSYSREL=0.;
         DO KA = 1 TO NTERM;
           BROD=1.0;
           DO KB = 1 TO NCOM;
           IF SUBSTR(TERMS(KA),KB, 1) ^= '1'B THEN GO TO FIN;
           DREL=REL(KB);
           BROD=BROD*DREL;
   FIN:    END;
         DSYSREL=DSYSREL+BROD*COEF(KA);
         END;
         SYSREL=DSYSREL;
         END COMPUTE;
DATAGEN.PROCEDURE   (PORT,FAIL,NCOM,TYPE,REL);
/*                                                                    */
/*--------------------------------------------------------------------*/
/*       DATAGEN PROCEDURE OBTAINS VALUES OF COMPONENT RELIABILITY BY */
/* CALLING CBETA (BETA DEVIATE GENERATOR) OR RGAMA (GAMMA DEVIATE     */
/* GENERATOR).                                                        */
/*--------------------------------------------------------------------*/
/*                                                                    */
```

```
     DCL (REL(*),PORT(*),FAIL(*),AA,B,TEMP                ) FLOAT DEC(6);

     DCL (NCOM,ID                                         ) FIXED BIN(15);
     DCL TYPE(*) BIT (*);
     DO ID=1 TO NCOM;
      IF TYPE(ID) ='0'B THEN DO;
      REL(ID)=0;
     END;
     ELSE DO;
      IF TYPE(ID) = '1'B THEN GO TO BTVR;
                         ELSE IF TYPE(ID) = '0'B THEN GO TO GMVR;
      PUT FILE (SYSPRINT) LIST ('TYPE,PORT,FAIL DESIGNATED IN ERROR');
      EXIT;
     END;
BTVR:AA=PORT(ID) + 1.;  /* SUCCESS-FAILURE COMPONENT RELIABILITY */
     B = FAIL(ID) + 1.;
     REL(ID) = CBETA(AA,B);
     GO TO NXTT;
GMVR:AA=FAIL(ID) + 1.;  /* TIME-TO-FAILURE COMPONENT RELIABILITY */
     B = 1./(PORT(ID)+1.);
     TEMP   = RGAMA(AA) * B   ;
     REL(ID) = EXP (-TEMP);
NXTT:END;
     END DATAGEN;
CBETA: PROC(A,B) RETURNS (FLOAT DEC(6));
/*                                                                    */
/*--------------------------------------------------------------------*/
/*      PROCEDURE CBETA GENERATES RANDOM DEVIATES FROM THE BETA       */
/* DISTRIBUTION WITH PARAMETERS EQUAL TO "A" AND "B" RESPECTIVELY.    */
/* THE METHOD USED IS REJECTION FROM A CAUCHY DISTRIBUTION. JOHNK'S   */
/* METHOD IS ALSO USED FOR THE PARAMETER VALUE <= 1.                  */
/*                                                                    */
/* CHANDLER, J. P., COMPUTING AND INFORMATION SCIENCES, OSU.          */
/*                                                                    */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL PARH FLOAT DEC(6) INIT(1.1);
     DCL PARC FLOAT DEC(6) INIT(1.1);
     DCL PI FLOAT DEC(6) INIT(3.14159);
     DCL A1SAV FLOAT DEC(6) INIT(1.);
     DCL B1SAV FLOAT DEC(6) INIT(1.);
     DCL Z FIXED BIN(15) INIT(0), (A,B) FLOAT DEC(6),
     DCL (A1,B1,ALCNST,ALPA,BMOD,BMOD1) FLOAT DEC(6);
     DCL (FBL,DT,BU,BL,GCL,U,T,GCL      ) FLOAT DEC(6);
     IF A=1. & B=1. THEN RETURN (RANF(Z));
     IF (A>0. & A<=1.) | (B>0. & B<=1.) THEN
                          DO;A1=1./A;          /* JOHNK'S METHOD */
                             B1=1./B;
                          LA:
                             PARC=RANF(Z)**A1;
                             PARH=RANF(Z)**B1+PARC;
                             IF PARH>1. THEN GOTO LA;
                             RETURN(PARC/PARH);
                          END;
     /*                      CHECK FOR ILLEGAL VALUES OF A AND B.     */
     /*                                                               */
     IF A > 0. & B >= 0. THEN GO TO L73; ELSE GO TO L2;
     /*                                                               */
     /*                      HAVE THE VALUES OF A AND B CHANGED SINCE */
     /*                      THE PREVIOUS CALL TO CBETA.              */
     /*                                                               */
 L73:A1=A-1.;
     B1=B-1.;
     IF A1 ^= A1SAV THEN GO TO L74;
     IF B1  = B1SAV THEN GO TO L1;
 L74:A1SAV=A1;                /* RECOMPUTE CERTAIN CONSTANTS.        */
     B1SAV=B1;
     BMOD=A1/(A1+B1);
     BMOD1=1.-BMOD;
     ALCNST=ALGAMA(A+B)-ALGAMA(A)-ALGAMA(B);
     ALPA=PARC*SQRT(2.*BMOD*BMOD1/(A1+B1));
     GCL= LOG(PARH*PI*ALPA)+ALCNST+A1* LOG(BMOD)+B1* LOG(BMOD1);
     BL=ATAN(-BMOD/ALPA);
     BU=ATAN(BMOD1/ALPA);    /* GENERATE BETA RANDOM DEVIATE.       */
  L1:DT=TAN(BL+(BU-BL)*RANF(Z))*ALPA;
     T=BMOD+DT;
     IF T <= 0. | T >= 1. THEN GO TO L1;
     GCL=GCL+ LOG(ALPA/(ALPA**2+DT**2)/PI);
     FBL=ALCNST+A1* LOG(T)+B1* LOG(1.-T);
     IF FBL > GCL THEN GO TO L2;
```

```
 L11:U=RANF(Z);
     IF U (= 0. THEN GO TO L11;
     IF GCL*LOG(U) ) EBL THEN GO TO L1;
     RETURN(1);
 L2:PUT FILE(SYSPRINT) LIST ('* ERROR IN SUBROUTINE GNLTA: ',A,B,BMOD,
                              ALCNST,ALPA,CL,T,GCL,EBL);
     EXIT;
     END;
ALGAMA: PROC (XX) RETURNS (FLOAT DEC(6));
/*                                                                    */
/*--------------------------------------------------------------------*/
/*       ALGAMA PROCEDURE COMPUTES DOUBLE PRECESION NATURAL LOGARITHM*/
/* OF GAMMA FUNCTION OF A GIVEN SINGLE PRECISION ARGUMENT.            */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL    (ZZ,TERM,RZ2)       FLOAT DEC (14),
            (XX,DLNG)           FLOAT DEC (6), ERROR CHAR(1) INIT('0');
   ZZ   =XX ;
   IF XX (= 1.E10
   THEN IF XX (= 1.E-09                 /* XX IS NEAR 0 OR NEGATIVE     */
        THEN DO ;                       /* SET ERROR INDICATOR          */
             ERROR='2' ; PUT FILE(SYSPRINT) DATA (ERROR) SKIP(2);
             DLNG =-1.E75 ;
             GO TO S20 ;
        END;
        ELSE DO ;                       /* XX ) 0 AND ( OR = TO 1.E+10 */
             TERM =1.E0 ;
S10:         IF ZZ (= 10.E0             /* ZZ ( OR = 10                 */
             THEN DO ;                  /* TRANSLATE ARGUMENT           */
                  TERM =TERM*ZZ ;
                  ZZ   =ZZ+1.E0 ;
                  GO TO S10 ;
             END;
             ELSE DO ;                  /* CALC. EQUATION 1             */
                  RZ2  =1.E0/ZZ**2 ;
                  DLNG =(ZZ-0.5E0)*LOG(ZZ)-ZZ+0.918938533204672E0
                       -LOG(TERM)+(1.E0/ZZ)*(.833333333333333E-01
                       -(RZ2*(.277777777777777E-02+(RZ2*
                       (.793650793650793E-03-(RZ2*
                       (.595238095238095E-03)))))) ;
                  GO TO S20 ;
             END;
        END;
   ELSE IF XX (  1.E70 THEN DO;         /* XX ) 1.E+10 AND ( 1.E+70     */
             DLNG =ZZ*(LOG(ZZ)-1.E0) ;/* CALC. EQUATION 2             */
             GO TO S20 ;
        END;
        ELSE DO ;                       /* XX ) OR = 1.E+70             */
             ERROR='1' ;    PUT FILE(SYSPRINT) DATA (ERROR) SKIP(2);
             DLNG =1.E75 ;
        END;
S20:RETURN (DLNG); END;
RGAMA: PROC (A) RETURNS (FLOAT DEC(6));
/*                                                                    */
/*--------------------------------------------------------------------*/
/*     PROCEDURE RGAMA GENERATES A RANDOM GAMMA DEVIATE WITH PARA-    */
/* METER "A", A)1/3.                                                  */
/*                                                                    */
/*     MARSAGLIA, G., SCHOOL OF COMPUTER SCIENCEM MCGILL UNIVERSITY. */
/*     CHANDLER, J. P., COMPUTING & INFORMATION SCIENCES, OSU.        */
/*--------------------------------------------------------------------*/
/*                                                                    */
     DCL B FLOAT DEC(6) INIT (-1.);
     DCL D FIXED BIN (15) INIT(0);
     DCL (A,CL,S,ROOTH ,CS,X,Z,U,E,CD,T,CC,GAMA,ZO     ) FLOAT DEC(6);
     CL=3.*A-1.;
     IF CL(= 0. THEN STOP; /* TEST FOR INVALID VALUE OF -A-.          */
     /*                       HAS THE VALUE OF -A- CHANGED SINCE THE  */
     /*                       PREVIOUS CALL TO RGAMA.                 */
     IF A = B THEN GO TO ONE;
     B=A;                       /*    RECOMPUTE VARIOUS CONSTANTS.     */
     S=1./(3.*SQRT(A));
     ROOTH=SQRT(3.);
     ZO=1.-ROOTH*S;
     CC=A*ZO**3-(S-ROOTH)**2/2.;
     CS=1.-S**2;
ONE: X=GAUSF(D);       /* GAUSF RETURNS A NORMAL(0,1) RANDOM DEVIATE */
     Z=S*X+CS;
     IF Z (= 0. THEN GO TO ONE;
     GAMA=A*Z**3 ;
S1X: U=RANF(D);
     IF U (= 0. THEN GO TO S1X;
     E = - LOG(U);          /* GENERATE E, AN EXPONENTIAL RANDOM DEVIATE*/
     CD=E+X**2/2.- GAMA+CC;
     T=1.-ZO/Z;
     IF CD+CL*T*(1.+T*(1./2.+T/3.)) ) 0. THEN GO TO EGT;
     IF CD+CL* LOG(Z/ZO) ( 0. THEN GO TO ONE;
```

```
EGT.RETURN( GAMA);
    END;
GAUSF:PROC (NDUMY) RETURNS (FLOAT DEC(6));
/*                                                                    */
/*--------------------------------------------------------------------*/
/*     PROCEDURE GAUSF GENERATES NORMAL DEVIATES FROM NORMAL DISTRI-*/
/* BUTION WITH ZERO MEAN AND UNIT VARIANCE. THIS PROCEDURE IS CALLED*/
/* BY RGAMA PROCEDURE.                                                */
/*     MARSAGLIA, G. AND BRAY T. A., SIAM REVIEW 6, 1964, P.260.      */
/*                                                                    */
/*     CHANDLER, J. P., COMPUTING AND INFORMATION SCIENCE, OSU.       */
/*--------------------------------------------------------------------*/
/*                                                                    */
    DCL NARG  FIXED BIN(15) INIT(0), NDUMY FIXED BIN(15);
    DCL NARGA FIXED BIN(15) INIT(0);
    DCL (A,ABX, EX,FAC,G,GAUS ,SUMSQ,VA,VB,X,Y,TMP  ) FLOAT DEC(6);
    DCL (RJUMP,R                                    ) FLOAT DEC(6);
    GAUS = 0.;
    NARGA=NARG;
    RJUMP=RANF(NARG);
    IF RJUMP ( .1362 THEN GO TO L20;
    A=(RJUMP-.1362)/.8638;
    /*                                                                */
    R=RANF(NARG);
    TMP=A+R*RANF(NARGA)-1.5;
    GAUS=TMP*2.;
    GO TO L150;
L20:IF RJUMP ( .0255 THEN GO TO L40;
    A=(RJUMP-.0255)/.1107;
    GAUS=1.5*(A+RANF(NARG)-1.);
    GO TO L150;
L40:IF RJUMP ( .0026997961 THEN GO TO L110;
    X=6.*RANF(NARG)-3.;
    Y=.358*RANF(NARGA);
    EX=17.49731196*EXP(-.5*X**2) ;
    ABX=ABS(X);
    IF ABX ) 1. THEN GO TO L70;
    G=EX-4.73570326*(3.-X**2)-2.15787544*(1.5-ABX);
    GO TO L90;
L70:G=EX-2.36785163*(3.-ABX)**2;
    IF ABX ) 1.5 THEN GO TO L90;
    G=G-2.15787544*(1.5-ABX);
L90:
    IF Y ) G THEN GO TO L150;
    GAUS=X;
    GO TO L150;
L110:
    VA=2.*RANF(NARG)-1.;
    VB=2.*RANF(NARGA)-1.;
    SUMSQ=VA**2+VB**2;
    IF SUMSQ ) 1. THEN GO TO L110;
    IF SUMSQ (= 0. THEN GO TO L110;
    FAC=SQRT((9.-2.* LOG(SUMSQ))/SUMSQ);
    GAUS=VA*FAC ;
    IF ABS(GAUS)  ( 3 THEN GO TO L110;
L150:
    RETURN(GAUS);
    END;
      (NOFIXEDOVERFLOW)
RANF : PROC (NARG) RETURNS (FLOAT DEC (6));
/*                                                                    */
/*--------------------------------------------------------------------*/
/*      THIS PROCEDURE GENERATES PSEUDO-RANDOM NUMBERS, UNIFORMLY     */
/* DISTRUBUTED ON (0,1).  THIS VERSION IS FOR IBM 370. METHOD IS      */
/* COMPOSITE OF THREE MULTIPLICATIVE CONGRUENTIAL GENERATOR. IF RANF  */
/* IS CALLED WITH NARG=0, THE NEXT RANDOM NUMBER IS RETURNED. IF      */
/* RANF IS CALLED WITH NARG^=0, THE GENERATOR RE-INITIALIZED USING    */
/* IABS(2*NARG+1) AND THE FIRST RANDOM NUMBER FROM THE NEW SEQUENCE   */
/* IS RETURNED.                                                       */
/*      MARSAGLIA, G. AND BRAY T. A., COMM. ACM 11, 1968, P.757.      */
/*                                                                    */
/*      CHANDLER, J. P., COMPUTING AND INFORMATION SCIENCES, OSU.     */
/*--------------------------------------------------------------------*/
```

```
/*                                                                    */
        DCL K FIXED BINARY(31)   INITIAL(7654321) STATIC,
        DCL L FIXED BINARY(31)   INITIAL(7654321) STATIC,
        DCL M FIXED BINARY(31)   INITIAL(7654321) STATIC,
        DCL MK FIXED BINARY(31)   STATIC INITIAL(32762),
        DCL ML FIXED BINARY(31)   STATIC INITIAL(32921),
        DCL MM FIXED BINARY(31)   STATIC INITIAL(65541),
        DCL NARG FIXED BIN (15),
        DCL NFIRST BIT(1) STATIC INITIAL('1'B),
        DCL (J,KLM,N(128),NDIV,NR                ) FIXED BIN(31) STATIC,
        DCL (RAN,RDIV                            ) FLOAT DEC(6) STATIC;

        IF NARG ^= 0 THEN DO;          /* RE-INITIALIZE USING NARG.   */
            KLM = ABS(2 * NARG + 1);
            K,L,M = KLM;
        END;
        ELSE DO;
        IF ^ NFIRST THEN GO TO SKP;
        END;
        NFIRST = '0'B;                 /* INITIALISE THE ROUTINE.     */
        NDIV = 16777216;
        RDIV = 32768.0 * 65536.0;
        DO J = 1 TO 128;               /* FILL THE TABLE.             */
           K = K * MK;
           N(J) = K;
        END;
   SKP: L = L * ML;                    /* COMPUTE THE NEXT RANDOM NO. */
        J = 1 + ABS(L) / NDIV;
        M = M * MM;
        NR = ABS(N(J) + L + M);
        RAN = FLOAT(NR) / RDIV;
        K = K * MK;          /* REFILL THE J-TH PLACE IN THE TABLE    */
        N(J) = K;
        RETURN(RAN);
        END RANF;
SORT:PROC(N,X);
/*--------------------------------------------------------------------*/
/* SORTINT PROCEDURE ON THE VALUES OF THE SYSTEM RELIABILITY BY       */
/* USING THE ORIGINAL SHELL  SORTING METHOD.                          */
/*--------------------------------------------------------------------*/
     DCL (N,M,I,J                                        ) FIXED BIN(15),
     DCL (X(*),Y                                         ) FLOAT DEC(6);

     IF N<2 THEN RETURN;
     M = N;
 ONE:M = M/2;
     DO J=1 TO N-M;
     Y = X(J+M);
     DO I=J BY -M TO 1;
     IF X(I)<Y THEN GO TO TWO;
     X(I+M) = X(I);
     END;
 TWO:X(I+M) = Y;
     END;
     IF M>1 THEN GO TO ONE;
     RETURN;
     END; /* SHELL SORT */
 STAT: PROC (SN,SV,AVSM,ATYPE,TIME,UNIT,TOT,TOT2,STMCR);
 /*                                                                    */
 /*--------------------------------------------------------------------*/
 /*     PROCEDURE STAT PROVIDES STATISTICAL INFORMATION FOR ASSESS-    */
 /* MENT OF SYSTEM RELIABILITY. MEAN RELIABILITY, AVERAGE OF SYSTEM    */
 /* RELIABILITY BY SIMULATION, MTBF, AND PERCENTILE POINTS OF SYSTEM   */
 /* RELIABILITY AND MTBF.                                              */
 /*--------------------------------------------------------------------*/
 /*                                                                    */
     DCL PCT(11) FLOAT DEC(6) INIT (.05,.1,.2,.25,.5,.75,.8,.9,.95,.975
         ,.99) STATIC,(VAL(11),MTBF(11),SV(*),AVSM,MTBA) FLOAT DEC(6),

     DCL (I,SN,IV) FIXED BIN(15), MTBFMCR FLOAT DEC(6),
     DCL (TIME,EVAL,CVA,SUM,FLOG,TOT,TOT2,SD,VAR,PR     ) FLOAT DEC(6),
     DCL UNIT CHAR(*),ATYPE1 CHAR(13),ATYPE CHAR(*),STMCR FLOAT DEC(6);

     ATYPE1 = 'RELIABILITY';
     IF SN=1 THEN DO;
     VAR = 0.; SD=0.;
     END;
     ELSE DO;
     PR=1.;
     DO I=1 TO SN;
     PR=PR*(1.-SV(I));
     END;
     FLOG=1.-PR**(1./FLOAT(SN));
     VAR=(TOT2-TOT*TOT/SN)/(SN-1);
     SD = SQRT(VAR);
```

```
      END;
      PUT FILE(SYSPRINT) EDIT (' AVERAGE SYSTEM RELIABILITY FROM ',
          SIMNUM,' MONTE CARLO TRIALS IS ',AVSM,' : AVERAGE SYSTEM ',
          'UNRELIABILITY IS ',1.-AVSM)
          (SKIP(3),A,F(4),A,F(8,6),A,A,F(8,6))
          (' VARIANCE                    '
          ,' ' ,VAR) (SKIP(2),A,A,F(8,6)) (' STANDARD DEVIATION         '
          ,'                ',SD) (SKIP(2),A,A,F(8,6));
      IF TIME ^= 0 THEN
   DO;MTBFMCR=-TIME/LOG(STMCR);
      MTBA=-TIME/LOG(AVSM);
      PUT FILE(SYSPRINT) EDIT ('THE MISSION TIME IS ',TIME,UNIT)
          (SKIP(2),COL(6),A,F(9,3),X(2),A) ('THE ESTIMATED SYSTEM MTBF '
         ,'BASED UPON MEAN COMPONENT RELIABILITIES IS ',MTBFMCR)
          (SKIP(2),COL(6),A,A,F(9,3)) ('THE ESTIMATED SYSTEM MTBF ',
          'BASED UPON MEAN SYSTEM RELIABILITY ',SIMNUM,' MONTE CARLO ',
          'TRIALS IS ',MTBA) (SKIP(2),COL(6),A,A,F(3),A,A,F(9,3))
          (A(TYPE1,'MTBF') (SKIP(6),
          COL(24),A,COL(49),A) ('PERCENTILE','PERCENTILE','PERCENTILE')
          (COL(4),A,COL(25),A,COL(46),A) ('POINTS','POINTS') (COL(27),
          A,COL(48),A);
   END;
      ELSE DO;
          PUT FILE (SYSPRINT) EDIT ('PERCENTILE','PERCENTILE') (SKIP(3),
          COL(4),COL(28),A) ('POINTS') (COL(27),A);
      END;
      /*                                                              */
      DO I=1 TO 11;                  /* PERCENTILE COMPUTATION */
      VAL(I)=SN*PCT(I)+.5;
      IV=FIXED(VAL(I));
      FVAL=FLOAT(IV);
      IF FVAL ^= VAL(I) THEN DO;
      VAL(I)=VAL(I)-FVAL;
      IF IV=0 THEN DO;
      SZR=SV(1)*2.-SV(2);
      CVA=SV(2)-SV(1);
      VAL(I)=CVA*VAL(I);
      VAL(I)=SZR+VAL(I);
      END;
      ELSE DO;
      IF IV=SN THEN DO;
      CVA=SV(IV)-SV(IV-1);
      END;
      ELSE DO;
      CVA=SV(IV+1)-SV(IV);
      END;
      VAL(I)=CVA*VAL(I);
      VAL(I)=SV(IV)+VAL(I);
      END;
      END;
      ELSE DO;
      VAL(I)=SV(IV);
      END;
      IF TIME ^= 0. THEN DO;
      MTBF(I)=-TIME/LOG(VAL(I));
      PUT FILE(SYSPRINT) EDIT (PCT(I)*100.,'PERCENT',VAL(I),MTBF(I),
          UNIT)(SKIP(2),COL(1),F(7,1),COL(10),A,COL(26),F(8,6),X(9),
          F(15,0),A);
      END;
      ELSE DO;
      PUT FILE(SYSPRINT) EDIT (PCT(I)*100.,'PERCENT',VAL(I)) (COL(2),
          A,COL(26),F(8,6));
      END;
   EXIT:
      END;
      END; /* STAT */
FINISH: END  SPARCS2;
READY
```

VITA[2]

Keun Kuk Lee

Candidate for the Degree of

Master of Science

Thesis: REJECTION METHODS FOR GENERATING RANDOM DEVIATES AND THEIR APPLICATIONS IN SYSTEM RELIABILITY

Major Field: Computing and Information Sciences

Biographical:

Personal Data: Born in Masan, Korea, August 8, 1946, the son of Mr. and Mrs. Sung Wook Lee.

Education: Attended elementary school in Seoul, Korea; attended junior high school in Seoul, Korea; graduated from Dongsung Senior High School, Seoul, Korea, in 1963; received the Degree of Bachelor of Science from Yonsei University, Seoul, Korea, in 1968; with a major in Economics; received Master of Science in Statistics from Oklahoma State University, Stillwater, Oklahoma, in 1976; completed the requirements for the Master of Science degree in Computing and Information Science at Oklahoma State University, in December, 1977.

Professional Experience: Junior program and system analyst in Shinjin Motor Co., Seoul, Korea, 1968-70; graduate research assistant in Department of Statistics, Oklahoma State University, Stillwater, Oklahoma, 1973-74; graduate research assistant in the Department of Management Science, Oklahoma State University, Stillwater, Oklahoma, from May, 1974 to May, 1977.