THE APPROXIMATION OF JOINT DISTRIBUTION FUNCTIONS FOR APPLICATION IN PROBABILISTIC MECHANICAL DESIGN

by

BRUCE EUGENE SWANSON

B.S., Kansas State University, 1985

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

[Signature]
Major Professor
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. LITERATURE REVIEW</td>
<td>5</td>
</tr>
<tr>
<td>Probabilistic Machine Design</td>
<td>5</td>
</tr>
<tr>
<td>Monte Carlo Simulation</td>
<td>6</td>
</tr>
<tr>
<td>Fast Probability Integration</td>
<td>10</td>
</tr>
<tr>
<td>Discrete Simulation</td>
<td>11</td>
</tr>
<tr>
<td>III. METHOD AND PROCEDURE</td>
<td>14</td>
</tr>
<tr>
<td>Cell Development</td>
<td>16</td>
</tr>
<tr>
<td>Discrete Point Representation</td>
<td>17</td>
</tr>
<tr>
<td>Simulation</td>
<td>20</td>
</tr>
<tr>
<td>Distribution Function</td>
<td>21</td>
</tr>
<tr>
<td>Density Functions</td>
<td>22</td>
</tr>
<tr>
<td>Example</td>
<td>24</td>
</tr>
<tr>
<td>IV. PROGRAM DESCRIPTION</td>
<td>38</td>
</tr>
<tr>
<td>Discrete Simulation</td>
<td>39</td>
</tr>
<tr>
<td>Density Function Approximation</td>
<td>41</td>
</tr>
<tr>
<td>Data Management</td>
<td>42</td>
</tr>
<tr>
<td>Graphics</td>
<td>42</td>
</tr>
<tr>
<td>V. COMPARISON</td>
<td>43</td>
</tr>
<tr>
<td>Problem</td>
<td>45</td>
</tr>
<tr>
<td>Exact Solution</td>
<td>45</td>
</tr>
<tr>
<td>Monte Carlo Simulation vs. Discrete Simulation</td>
<td>50</td>
</tr>
<tr>
<td>VI. CANTILEVER I-BEAM</td>
<td>72</td>
</tr>
<tr>
<td>Problem</td>
<td>72</td>
</tr>
<tr>
<td>Solution</td>
<td>76</td>
</tr>
</tbody>
</table>
APPENDIX E ................................. 180
Flowchart for DISCSIM Program ........... 181

APPENDIX F .................................. 183
Sample Output from the DISCSIM Program ... 184

APPENDIX G .................................. 189
Notes about the Comparison of the Monte Carlo Simulation vs. the Discrete Simulation ...... 190
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Normal Distribution Approximation Using Seven Discrete Points</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Discrete Approximation of Random Variables X and Y Using Five Cells</td>
<td>28</td>
</tr>
<tr>
<td>3.3</td>
<td>Distribution of 25 Discrete Z Locations Z=(X*Y**2)/12</td>
<td>30</td>
</tr>
<tr>
<td>6.1</td>
<td>Random Variables for a Cantilever I-Beam</td>
<td>75</td>
</tr>
<tr>
<td>7.1</td>
<td>Bounding Probabilities of Failure of a Bar in Tension</td>
<td>100</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Normal Distribution Divided into Seven Cells</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Seven Cell Discrete Approximation of a Normal Distribution</td>
<td>19</td>
</tr>
<tr>
<td>3.3</td>
<td>Five Cell Discrete Approximation of a Uniform Distribution</td>
<td>26</td>
</tr>
<tr>
<td>3.4</td>
<td>Five Cell Discrete Approximation of a Triangular Distribution</td>
<td>27</td>
</tr>
<tr>
<td>3.5</td>
<td>Z Distribution of 25 Discrete Locations</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>Z Distribution (Z=X*Y**2)/12</td>
<td>31</td>
</tr>
<tr>
<td>3.7</td>
<td>Smoothed Cubic Spline Z Distribution, (Z=X*Y**2)/12, Tolerance=1.0</td>
<td>32</td>
</tr>
<tr>
<td>3.8</td>
<td>Smoothed Cubic Spline Z Distribution, (Z=X*Y**2)/12, Tolerance=1.5</td>
<td>33</td>
</tr>
<tr>
<td>3.9</td>
<td>Joint Probability Histogram (Z=X*Y**2)/12</td>
<td>35</td>
</tr>
<tr>
<td>3.10</td>
<td>Cubic Spline Density Function (Z=X*Y**2)/12, Tolerance=1.0</td>
<td>36</td>
</tr>
<tr>
<td>3.11</td>
<td>Cubic Spline Density Function (Z=X*Y**2)/12, Tolerance=1.5</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>Approximation and Exact Joint Probability Distribution Functions</td>
<td>44</td>
</tr>
<tr>
<td>5.2</td>
<td>Exact Joint Probability Distribution Function</td>
<td>49</td>
</tr>
</tbody>
</table>
5.3 Exact Joint Probability Density Function

5.4 Joint Probability Distribution Function for the Discrete Simulation using 50 Discrete Points per Random Variable...

5.5 Joint Probability Distribution Function for the Discrete Simulation using 50 Discrete Points per Random Variable and the Exact Solution

5.6 Joint Probability Distribution Function for the Monte Carlo Simulation using 325 Unique R Locations

5.7 Joint Probability Distribution Function for the Monte Carlo Simulation using 325 Unique R Locations and the Exact Solution

5.8 Joint Probability Distribution Function for the Discrete Simulation using 100 Discrete Points per Random Variable...

5.9 Joint Probability Distribution Function for the Discrete Simulation using 100 Discrete Points per Random Variable and the Exact Solution

5.10 Joint Probability Distribution Function for the Monte Carlo Simulation using 1275 Unique R Location

5.11 Joint Probability Distribution Function for the Monte Carlo Simulation using 1275 Unique R Locations and the Exact Solution

5.12 Joint Probability Distribution Function for the Discrete Simulation using 200 Discrete Points per Random Variable...

5.13 Joint Probability Distribution Function for the Discrete Simulation using 200 Discrete Points per Random Variable and the Exact Solution...
5.14 Joint Probability Distribution Function for the Monte Carlo Simulation using 5049 Unique R Location

5.15 Joint Probability Distribution Function for the Monte Carlo Simulation using 5049 Unique R Locations and the Exact Solution

5.16 Mean Squared Error vs. Number of Unique R Locations for the Monte Carlo and Discrete Simulation

5.17 Maximum Deviation vs. Number of Unique R Locations for the Monte Carlo and Discrete Simulation

5.18 Mean Squared Error vs. Computer Processing Unit Time for the Monte Carlo and Discrete Simulation

5.19 Maximum Deviation vs. Computer Processing Unit Time for the Monte Carlo and Discrete Simulation

6.1a Cantilever I-Beam Subjected to a Uniform Load

6.1b Free-Body Diagram of a Cantilever I-Beam Subjected to a Uniform Load

6.2 Cross Section of the Cantilever I-Beam

6.3 Joint Probability Distribution Function of the Maximum Moment of a Cantilever I-Beam

6.4 Joint Probability Density Function of the Maximum Moment of a Cantilever I-Beam

6.5 Joint Probability Distribution Function of the Moment of Inertia of a Cantilever I-Beam

6.6 Joint Probability Density Function of the Moment of Inertia of a Cantilever I-Beam
6.7 Joint Probability Distribution Function of the Maximum Deflection of a Cantilever I-Beam.......................... 82
6.8 Joint Probability Density Function of the Maximum Deflection of a Cantilever I-Beam.......................... 83
6.9 Joint Probability Distribution Function of the Slope at the End of a Cantilever I-Beam.......................... 84
6.10 Joint Probability Density Function of the Slope at the End of a Cantilever I-Beam.......................... 85
6.11 Joint Probability Distribution Function of the Maximum Stress of a Cantilever I-Beam.......................... 87
6.13 Joint Probability Distribution Function of the Maximum Shear Stress of a Cantilever I-Beam.......................... 89
6.14 Joint Probability Distribution Function of the Maximum Shear Stress of a Cantilever I-Beam.......................... 90
6.15 Joint Probability Distribution Function of the Ratio $S_{act}/S_{yield}$ of a Cantilever I-Beam.......................... 92
6.14 Joint Probability Distribution Function of the Ratio $S_{act}/S_{yield}$ of a Cantilever I-Beam.......................... 93
7.1 Known Probability Density Functions for Random Variables 1 and 2.......................... 95
7.2 Upper and Lower Bounding Curves of a Hypothetical Joint Distribution Function. 97
7.3 The Exact Solution and the Bounding Curves for the Probabilistic Pythagorean Random Variable Using 100 Discrete Points... 99
CHAPTER I

INTRODUCTION

Design engineers have begun to appreciate the limitations of the traditional or classical method of design. For example, the classical criteria for strength-limited design is that the overall strength of the system must be greater than the maximum applied load to the system multiplied by a factor of safety. This safety factor is usually a value between 1.3 and 6, depending on the structure and the materials. Although this approach is very simple in application, it often leads to designs that are overdesigned in terms of weight and bulk. Haugen [1] points out that "Since safety factors are not a performance-related measure there is no way by which an engineer can know whether his designs are near optimum or overconservative."

Designers are now starting to realize that applied loads, dimensions, material properties, etc. are all subject to variation. From this realization has emerged
a new design method called Probabilistic Design. Probabilistic Design treats the design inputs as random variables with each random variable having its own probability density function. After the variables have been defined, the problem can be solved using probability mathematics. The results can then be analyzed to determine if the design satisfies the operational and economic requirements with an acceptable reliability of success.

Although formulating the functional relationships of the random variables into a joint probability integral form is usually possible, evaluating this integral can be a very complex mathematical operation. Springer [2] has written an entire book on the subject of algebraic operations of random variables. Many of the chapters of his book deal with deriving the joint distribution and joint density functions of rather simple functions which contain random variables. Although the mathematical techniques that he describes are certainly important from a theoretical standpoint, there are some disadvantages in practical applications. The designer needs to have a working knowledge of differential and integral calculus, statistical inference, and probability distribution theory. Another drawback is that there are many
functions for which it is impossible to obtain the exact solution to the probability density function. Many times the system is so involved and cumbersome that the average engineer does not have the time or the knowledge to solve the problem.

An alternative to the traditional mathematical approach to Probabilistic Design is a technique known as Monte Carlo Simulation. Law and Kelton [3] define Monte Carlo simulation as "...a scheme employing random numbers which is used for solving certain stochastic or deterministic problems where the passage of time plays no substantive role". The random numbers are generated so that they simulate the physical random process of the probabilistic problem. The system is simulated many times so the frequencies of the events can be studied.

Monte Carlo [4] simulation originated as a tool for the development of the atomic bomb during World War II. The scientists used Monte Carlo simulation to simulate the randomness of neutron diffusion in fissile material. In the last thirty years Monte Carlo techniques have been used for operational research and nuclear physics. Since the motions of neutrons are random, the nuclear physicists have used Monte Carlo to simulate the performance of nuclear reactors. The physicists can
simulate the reactor design without the dangers and expense of an actual physical experiment.

Although the Monte Carlo technique has been around for over 50 years, its application in mechanical engineering design has been limited. The purpose of this thesis is to develop a simulation algorithm that is capable of approximating the joint probability distribution function of complex algebraic expressions encountered in design engineering. This computer algorithm is similar to the Monte Carlo method of simulation. However, this new technique does not use generated random numbers to represent the probability density functions of the individual random variables. Instead, the program divides each density function into a given number of discrete sections with equal probabilities and then takes all possible outcomes into consideration before assembling the joint distribution and density curves.

The literature review that follows is an overall review of probabilistic design and how simulation has been used in different engineering applications.
CHAPTER II

LITERATURE REVIEW

Probabilistic Machine Design

Bury [5] compared the probabilistic machine design approach to the traditional design method of a simple tension link with a given safety factor, material property, applied load, and cross sectional area. Bury pointed out that many design engineers are dissatisfied [6] with the traditional method because applied loads, material properties, and actual machining dimensions can vary dramatically from one application to another. While the traditional method assumes the input variables to be deterministic, the probabilistic method assumes the variables to be random with unique probability density functions. Once the density functions have been determined, the problem can then be solved as a probabilistic event. However, he points out that this probabilistic approach can only be taken if data is available to properly describe the random variables.
Balkey, Meyer, and Witt [7] discussed the importance of using Probability Structural Mechanics as a tool for evaluating the reliability and structural risk of components and structures. Probability Structural Mechanics, or PMS, combines the traditional methods of structural mechanics with the probabilistic methods to determine the probability of structural failure. They pointed out the necessity of including uncertainties in the mechanical and structural design process. They concluded that "...as components and structures become more sophisticated, failure mechanisms will be probabilistically modeled from the beginning of the design process, and potential design improvements will be evaluated to assess their effects on reducing overall risk."

Monte Carlo Simulation

Haugen [1] discussed Monte Carlo simulation throughout his text. One rather simple example was a simulation of strain energy in a statically loaded wire. Haugen solved the problem by Monte Carlo simulation and by an approximate partial derivative method and compared the results. After 200 simulations the strain energy
values were plotted on normal and lognormal probability paper.

Elishakoff [8] used Monte Carlo simulation to predict the buckling time of an elastic bar. The initial imperfections of the bar were assumed to be normally distributed with a given mean and autocorrelation function. The problem was then simulated many times to determine the reliability of the structure. A histogram and distribution of the buckling time was plotted. The results were then compared to the previously published works of Linberg [9]. The Monte Carlo solution was very similar to the results obtained by Linberg.

Elishakoff continued his work of simulating structural imperfections using the Monte Carlo method. His later paper [10] discussed buckling of finite uniform columns resting on a "softening" nonlinear elastic foundation. The initial imperfections were again assumed to be normally distributed. The system was simulated many times and the reliability was plotted vs. nondimensional actual load. The nondimensional buckling loads were then plotted as a histogram and analyzed.

Elishakoff's latest research [11] was concerned with the effect of nonsymmetric random imperfections on the reliability of axially compressed cylindrical shells. He
used the Monte Carlo process to obtain the reliability functions of shells with asymmetric and axisymmetric imperfections. Using the results of his previous works, Elishakoff was able to use measured initial imperfections (variance-covariance matrices and the mean vectors) as a direct input for the Monte Carlo simulation.

Martin [12] developed vectorized and scalar particle transport Monte Carlo algorithms to simulate the transport of photons in a high-density, high-temperature plasma. The behavior of a photon was simulated by drawing random samples from probability distributions that describe the actual physical process. The results were tabulated to chart the overall tallies. Both algorithms were compared to a reference Monte Carlo code from Lawrence Livermore National Laboratory.

Ramsay [13] studied the variability of deflections of ten reinforced concrete fixed-ended T-beams and slabs. The dimensions of the beams, as well as the material properties and type of loadings were assumed to be random variables. A mathematical model was developed to describe the characteristics of the beams and slabs. The model considered the change of stiffness and the redistribution of moment along the length of the structure. Each of the structures was analyzed for four
different loading conditions. For each loading condition, 500 deflection values were generated using Monte Carlo simulation. These values were then converted to deflection ratios and statistically analyzed. The results of the simulation were compared to the deflections of those computed using the ACI 318-63 and ACI 318-71 Code procedures [14,15]. He concluded that the major causes of variability in deflections are the variability of the beam stiffness and the variability of the concrete strength.

Dao-Thien and Massoud [16] discussed the probabilistic distributions of stress and strength relationships. They presented a detailed mathematical background of a method used to obtain the stress and strength distribution functions. The density and distribution functions were written in the form of multiple integrals. A computer program was developed that assists the engineer in selecting a probability distribution (normal, log-normal, or Weibull) using linear regression. The program picks the distribution that minimizes the error function. An example was presented that illustrated a typical structural problem in the aerospace industry. The system was simulated 20,000 times using generated random numbers. The normal,
log-normal, and Weibull distributions were compared to the generated stress and strength data. The program picked the Weibull distribution as the best approximation of the design stress and the normal distribution for the design strength. The two distributions were compared to the histogram frequency distributions generated by the Monte Carlo method.

Fast Probability Integration

Hasofer-Lind [17], Rackwitz-Fiessler [18], Chen-Lind [19], and Wu [20] have developed a technique called Fast Probability Integration (FPI). The FPI method [21] assumes that all of the random variables are independent with known probability density functions. The random variable's density functions are transformed into equivalent normal random variables using a curve-fitting routine. The design point is then approximated using the scheme of normal tail approximation [22]. Wirsching and Wu [23] have tested various combinations of linear and nonlinear functions containing both normal and non-normal random variables. They report that the FPI method is more accurate and approximately 10 to 100 times faster than the Monte Carlo technique for approximating probabilities.
of complicated functions involving several random variables. They point out that while the Monte Carlo method may be useful as a research tool, the large amount of computer processing unit time needed to approximate low values of probability tends to make the Monte Carlo method expensive and impractical.

Discrete Simulation

Lockwood [24] introduced a new method of computing the probability density functions in turbulent flows. Lockwood pointed out that the traditional method of probabilistic design has several shortcomings. The primary disadvantage is that the "...computational burden increases more or less exponentially with dimensionality." He added that he has developed a more economical approach to the solution of the probability density function of the transport equation. He applied his method to a fully turbulent jet having a given inlet pipe Reynolds number and temperature. He assumed the temperature to be a random variable. To calculate the temperature's probability density function, he divided the range into a finite number of discrete points. The various relations were applied to each discrete point. The total solution
was obtained by summing each discrete point. He compared the predicted probability density functions with the measured probability density functions and determined that the agreement was remarkably good. He concluded that his new method has an advantage over the traditional Monte Carlo simulation technique. Since the discrete method does not use generated random numbers, there is no statistical errors associated with the generation of the random numbers, as is the case with the Monte Carlo technique.

In this study, the probability density functions are divided into discrete points similar to the study conducted by Lockwood. However, this paper goes on to approximate the joint distribution function of algebraic expressions containing several independent random variables with known density functions. The methods and procedures for this simulation are discussed in Chapter 3. The computer programs that are used to approximate the joint probability distribution and density functions are described in Chapter 4. In Chapter 5 the discrete approximation is compared to the traditional Monte Carlo technique for a joint probability distribution that can be solved exactly. Chapter 6 contains an example of how this probabilistic method can
be used as a tool by engineers to aid in the design and development of structures and components. In Chapter 7 a different technique is discussed that determines the upper and lower bounds of a joint probability distribution using the discrete approximation. The conclusions and recommendations for further study are given in Chapters 8 and 9 respectively.
CHAPTER III

METHOD AND PROCEDURE

The purpose of this thesis is to develop a simulation algorithm that is capable of approximating the joint probability distribution function of complex algebraic expressions containing several independent random variables with known density functions. If the complex algebraic expression is represented by a variable $Z$, then $Z$ can be written as

$$Z = Z(X_1, X_2, \ldots, X_n) \quad (3.1)$$

where $X_1, X_2, \ldots, X_n$ are continuous, independent random variables with individual probability density functions $f_1(x_1), f_2(x_2), \ldots, f_n(x_n)$ respectively. Since the variables in this study are assumed to be independent, the joint probability density function of $Z$ can be written as the product of the $X$ random variables' density functions.
\[ f(z) = f_1(x_1)f_2(x_2) \cdots f_n(x_n) \quad (3.2) \]

The joint probability distribution function of the random variable for any n-dimensional set of \( Z \) can be written as,

\[
F(z) = \int \int \ldots \int_{\mathbb{R}^n} f_1(x_1)f_2(x_2) \cdots f_n(x_n) \, dx_1 \cdots dx_n \quad (3.3)
\]

If the random variables are bounded by the intervals \((a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\), the probability

\[
P(a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2, \ldots, a_n \leq x_n \leq b_n) = 
\int_{a_1}^{b_1} \int_{a_2}^{b_2} \ldots \int_{a_n}^{b_n} f(x_1, x_2, \ldots, x_n) \, dx_1 \cdots dx_n \quad (3.4)
\]

Although formulating the density functions into a multiple integral form similar to equations 3.3 and 3.4 is usually possible, integrating the function over the range of the \( Z \) event can be a very complex mathematical operation.
In this chapter, a method that approximates the joint probability distribution function is discussed. The method, called discrete simulation, has four steps:

1. Cell Development
2. Discrete Point Representation
3. Simulation
4. Distribution Approximation

**CELL DEVELOPMENT**

In order to approximate the continuous density function of each random variable, the known density functions are divided into \(n\) sections or cells. Each cell is computed to have equal area, thus equal probability of occurrence.

\[
\begin{align*}
1. \quad & \int_{a_0}^{a_1} f(x)\,dx = \frac{1}{n} \\
2. \quad & \int_{a_1}^{a_2} f(x)\,dx = \frac{1}{n} \\
& \vdots \\
n. \quad & \int_{a_{n-1}}^{a_n} f(x)\,dx = \frac{1}{n} 
\end{align*}
\]
Figure 3.1 shows an example of a normal random variable density function divided into seven cells with each cell having equal area. For sake of illustration, the density function has been divided into only seven cells. For practical applications the density function can contain anywhere from 30 to 200 cells.

**DISCRETE POINT REPRESENTATION**

After the function has been divided into equal areas, each cell is represented by a single discrete point. This point is located at the cell's center of gravity where:

\[
\int_{a_0}^{a_{cg}} f(x) dx = \int_{a_{cg}}^{a_1} f(x) dx = \frac{1}{2n} \quad (3.6)
\]

and has a magnitude of 1/n. Figure 3.2 shows a normal density function divided into seven cells with its center of gravity locations. Table 3.1 lists the center of gravity locations of a normal distribution approximation for seven cells.
FIGURE 3.1
NORMAL DISTRIBUTION
DIVIDED INTO 7 CELLS

PROBABILITY DENSITY FUNCTION, f(x)
FIGURE 3.2
7 CELL DISCRETE APPROXIMATION
OF A NORMAL DISTRIBUTION
PROBABILITY DENSTITY FUNCTION, f(x)
Table 3.1
Normal Distribution Approximation
Using Seven Discrete Points

Distribution:  Normal
Mean:  0.0
Variance:  1.0
Standard Deviation:  1.0
Number of Cells:  7

<table>
<thead>
<tr>
<th>Cell Number</th>
<th>Center of Gravity Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1.4655158</td>
</tr>
<tr>
<td>2</td>
<td>-0.7914320</td>
</tr>
<tr>
<td>3</td>
<td>-0.3656621</td>
</tr>
<tr>
<td>4</td>
<td>0.0000000</td>
</tr>
<tr>
<td>5</td>
<td>0.3656621</td>
</tr>
<tr>
<td>6</td>
<td>0.7914320</td>
</tr>
<tr>
<td>7</td>
<td>1.4655158</td>
</tr>
</tbody>
</table>

SIMULATION

Once each variable's density function has been chosen, all possible combinations of the expression are simulated. For example, if the joint probability distribution function of the equation

\[ Z = A \times B \]

is desired, and if the density functions of A and B are simulated using 100 and 150 discrete points respectively. The total possible combinations of Z is \( 100 \times 150 = 15,000 \).
DISTRIBUTION FUNCTION

After the outcomes $Z$, have been computed, the data is sorted from smallest to largest. Each outcome has a probability of $1/m$, where $m$ is the number of total outcomes. The joint cumulative distribution function can then be assembled by summing the ordered value's probabilities.

$$F(z) = \begin{cases} 
0 & i < z_i \\
F(z_{i-1}) + \frac{1}{m} & z_{i-1} \leq z < z_i \\
1 & z_m \leq z 
\end{cases} \quad (3.7)$$

If the values between the discrete points need to be known, a linear empirical distribution function can be written that interpolates between the known points:

$$G(z) = \begin{cases} 
0 & z < z_0 \\
G(z_{i-1}) + \frac{z - z_{i-1}}{z_1 - z_{i-1}} (G(z_1) - G(z_{i-1})) & z_{i-1} \leq z < z_i \\
1 & z_m \leq z 
\end{cases} \quad (3.8)$$

The computer program that performs the distribution approximation is listed in the Appendix.
DENSITY FUNCTIONS

Although developing a joint density function is not the primary purpose of this paper, two methods are used with reasonable success. These methods are only used to give an approximate shape of the joint probability density function and not an exact solution. The fitting of density functions is a complex problem in itself. The two methods are: histograms and cubic splines.

Histograms:

A histogram is simply a graphical estimate of the density function corresponding to the known data. To construct a histogram, the range of data is divided into k equal intervals. The data is then placed in its appropriate interval. Once all of the data has been sorted, the proportion of the values in each interval is determined.

The histogram is constructed using Sturge's [25] rule. The number of cells k is determined by the equation

\[ k = 1 + 3.3 \times \log_{10} m \]  

(3.9)
The range $r$, is estimated by the relationship

$$r = x_{\text{max}} - x_{\text{min}} \quad (3.10)$$

And the width $w$, of each interval in the histogram is

$$w = r/k \quad (3.11)$$

Although the histogram is easy to construct, the main disadvantage is the loss of information from grouping the data. Because of this, the histogram is used only as a reference to the possible density function's shape.

Cubic Splines:

A cubic spline is a set of cubic polynomials which is joined together to form a continuous function. A cubic spline subroutine is used to compute the first derivative of the function. The subroutine also determines a "smoother" approximation of the distribution function. The spline method uses a cubic interpolating function to approximate a "smooth" line through a given set of data points by minimizing the integral of the second derivative squared of the cubic function. A tolerance is imputed that allows the programmer to
tighten or loosen the curve fit. A detailed description of cubic splines can be found in reference [26].

The computer program listed in Appendix B performs the density approximations outlined above.

EXAMPLE

Given: Function \( Z = \frac{(X \times Y^3)}{12} \), where \( X \) and \( Y \) are continuous independent random variables. \( X \) is a uniform random variable with given probability density function:

\[
f(x) = \begin{cases} 
1/3 & 1 < x < 4 \\
0 & \text{otherwise}
\end{cases}
\]

\( Y \) is a triangular random variable with a given probability density function:

\[
f(y) = \begin{cases} 
\frac{2(y-4)}{5.25} & 1 < y < 2.5 \\
\frac{2(4-y)}{5.25} & 2.5 < y < 4 \\
0 & \text{otherwise}
\end{cases}
\]

Find: The joint probability distribution of random variable \( Z = Z(X,Y) \).

For simplicity each cell is approximated using only five cells. The area of each cell is equal to 1/5. The
bound for the first cell of variable $X$ is found by using the relationship:

$$\int_{a_0}^{a_1} f(x) \, dx = \text{Area of the cell}$$

$$\begin{align*}
\int_{1}^{3/2} \frac{1}{3} \, dx &= \frac{1}{5} \\
\frac{x}{3} \bigg|_1^{3/2} &= \frac{1}{5} \\
\frac{a_1}{3} &= \frac{8}{5}
\end{align*}$$

The center of gravity for cell 1 is computed using equation 3.6

$$\int_{1/3}^{a_{cg}} \frac{1}{3} \, dx = \frac{1}{10}$$

$$\frac{a_{cg}}{3} = \frac{13}{10}$$

The remaining locations can be found using the same method. Figures 3.3 and 3.4 show the two density functions and their respective centers of gravity. Table 3.2 lists the center of gravity locations for both random variables.
Figure 3.3
5 CELL DISCRETE APPROXIMATION
OF A UNIFORM DISTRIBUTION

Probability Density Function, f(x)
FIGURE 3.4
5 CELL DISCRETE APPROXIMATION
OF A TRIANGULAR DISTRIBUTION
Table 3.2

Discrete Approximation of Random Variables X and Y Using Five Cells

<table>
<thead>
<tr>
<th>Random Variable: X</th>
<th>Random Variable: Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distribution: Uniform</td>
<td>Distribution: Triangular</td>
</tr>
<tr>
<td>Mean: 2.5</td>
<td>Mean: 2.3333</td>
</tr>
<tr>
<td>Variance: 0.75</td>
<td>Variance: 0.5139</td>
</tr>
<tr>
<td>Number of Cells: 5</td>
<td>Number of Cells: 5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Center of Gravity Location</th>
<th>Cell Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.300000</td>
<td>1</td>
</tr>
<tr>
<td>1.900000</td>
<td>2</td>
</tr>
<tr>
<td>2.500000</td>
<td>3</td>
</tr>
<tr>
<td>3.100000</td>
<td>4</td>
</tr>
<tr>
<td>3.700000</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Center of Gravity Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.336660</td>
</tr>
<tr>
<td>1.949138</td>
</tr>
<tr>
<td>2.370829</td>
</tr>
<tr>
<td>2.745010</td>
</tr>
<tr>
<td>3.275431</td>
</tr>
</tbody>
</table>

After computing all possible values, the 25 unique Z locations are sorted. Each Z location has a probability of 1/25. Figure 3.5 shows the 25 discrete Z locations. Using equations 3.7 and 3.8 a joint probability distribution function is assembled. Table 3.3 shows the sorted Z locations and their distribution values. Figure 3.6 shows the assembled Z distribution function.

Figures 3.7 and 3.8 show the Z distributions of the spline smoothing subroutine for a tolerance of 1.0 and 1.5 respectively.
FIGURE 3.5
Z DISTRIBUTION OF 25 DISCRETE LOCATIONS $Z = (X \times Y \cdot 3)/12$
Table 3.3

Distribution of 25 Discrete Z Locations
\( Z = (X*Y^**2)/12 \)

<table>
<thead>
<tr>
<th>Number</th>
<th>Z Location</th>
<th>Z Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.2587170</td>
<td>0.0400000</td>
</tr>
<tr>
<td>2</td>
<td>0.3781249</td>
<td>0.0800000</td>
</tr>
<tr>
<td>3</td>
<td>0.4975327</td>
<td>0.1200000</td>
</tr>
<tr>
<td>4</td>
<td>0.6169406</td>
<td>0.1600000</td>
</tr>
<tr>
<td>5</td>
<td>0.7363484</td>
<td>0.2000000</td>
</tr>
<tr>
<td>6</td>
<td>0.8022129</td>
<td>0.2400000</td>
</tr>
<tr>
<td>7</td>
<td>1.1724650</td>
<td>0.2800000</td>
</tr>
<tr>
<td>8</td>
<td>1.4436524</td>
<td>0.3200000</td>
</tr>
<tr>
<td>9</td>
<td>1.5427172</td>
<td>0.3600000</td>
</tr>
<tr>
<td>10</td>
<td>1.9129693</td>
<td>0.4000000</td>
</tr>
<tr>
<td>11</td>
<td>2.1099535</td>
<td>0.4400000</td>
</tr>
<tr>
<td>12</td>
<td>2.2407525</td>
<td>0.4800000</td>
</tr>
<tr>
<td>13</td>
<td>2.2832214</td>
<td>0.5200000</td>
</tr>
<tr>
<td>14</td>
<td>2.7762546</td>
<td>0.5600000</td>
</tr>
<tr>
<td>15</td>
<td>3.2749459</td>
<td>0.6000000</td>
</tr>
<tr>
<td>16</td>
<td>3.4425557</td>
<td>0.6400000</td>
</tr>
<tr>
<td>17</td>
<td>3.8068655</td>
<td>0.6800000</td>
</tr>
<tr>
<td>18</td>
<td>4.1088568</td>
<td>0.7200000</td>
</tr>
<tr>
<td>19</td>
<td>4.3091393</td>
<td>0.7600000</td>
</tr>
<tr>
<td>20</td>
<td>5.3433328</td>
<td>0.8000000</td>
</tr>
<tr>
<td>21</td>
<td>5.5638804</td>
<td>0.8400000</td>
</tr>
<tr>
<td>22</td>
<td>6.3775262</td>
<td>0.8800000</td>
</tr>
<tr>
<td>23</td>
<td>7.3208952</td>
<td>0.9200000</td>
</tr>
<tr>
<td>24</td>
<td>9.0779101</td>
<td>0.9600000</td>
</tr>
<tr>
<td>25</td>
<td>10.8349250</td>
<td>1.0000000</td>
</tr>
</tbody>
</table>
FIGURE 3.6
Z DISTRIBUTION
Z = (X*Y^3)/12
FIGURE 3.7
'SMOOTHED' CUBIC SPLINE
Z DISTRIBUTION, Z=(X*Y^3)/12
TOLERANCE=1.0
Figure 3.8
'SMOOTHED' CUBIC SPLINE
Z DISTRIBUTION, Z=(X*Y^3)/12
TOLERANCE=1.5
The number of cells for the histogram is found using equation 3.9.

\[ k = 1 + 3.3 \times \log_{10} 25 \]

\[ k = 6 \]

Using equation 3.10, the range is equal to

\[ r = 10.8349 - 0.2587 \]

\[ r = 10.5762 \]

And each cell has a width of

\[ w = \frac{10.5762}{6} \]

\[ w = 1.7627 \]

Figure 3.9 shows the histogram approximation for the function. The cubic spline density approximations are plotted in Figures 3.10 and 3.11. The cubic spline approximations are rough. However, if the number of cells per density function were increased, the cubic spline curves would tend to be smoother with less wiggle.

It should be noted that this is a simplified case. In practice there could be an unlimited number of random variables with each random variable having 30 to 200 cells, resulting in literally thousands of calculations.
FIGURE 3.9
JOINT PROBABILITY
HISTOGRAM
$z = (x^2y^3)/12$
FIGURE 3.10
CUBIC SPLINE
DENSITY FUNCTION
\[ z = \frac{(x^2 - y^3)}{12} \]
TOLERANCE = 1.0
FIGURE 3.11
CUBIC SPLINE
DENSITY FUNCTION
$z = (x \cdot y^3) / 12$
TOLERANCE = 1.5
CHAPTER IV

PROGRAM DESCRIPTION

In this chapter, the computer programs that approximate the joint probability distribution and density functions are discussed. These programs are written in Fortran 77 on a Harris H-800 Super Mini-Computer. Because of storage problems, the simulation package had to be broken down into four separate programs. These programs are:

1. Discrete Simulation
2. Density Function Approximation
3. Data Management
4. Graphics

A complete listing of all the programs can be found in the Appendix.
The discrete simulation program approximates the joint probability distribution function of two random variables with known probability density functions. Before executing the program, the desired function must be inserted in the "Function" subroutine. When run, the program prompts the user for the type of distribution of each random variable and the number of discrete points used to approximate each random variable's density function. The density functions available to the user are:

1. Normal: The normal distribution can be used to represent quantities such as measured errors.
2. Triangular: The triangular distribution is used in the absence of data to approximate a rough model of a random variable's density function.
3. Uniform: The uniform distribution is usually used as a first model for a variable that is suspected of being random.
4. Special: The special distribution is used if the user has a distribution function stored on file. e.g. If a distribution function was generated from experimental or measured results.

After the program has approximated the joint probability distribution function, the mean, variance, locations of the discrete points for each of the two random variables, and the joint probability distribution function of the algebraic expression is printed to work file "W2". The joint probability distribution is also printed to a file (designated by the user) that is compatible with the density program, the special distribution subroutine, the data management program, and the plotting programs.

Although this program simulates only two random variables, a joint probability distribution function of an algebraic expression containing more than two random variables can be approximated. To do this, the function must be separated into groups of two random variables. Using the special distribution subroutine, the joint probability distribution function of each group of two random variables can be combined to form the joint probability distribution function of the algebraic
expression. If an expression contains \( n \) random variables, the program must be run \( n-1 \) times to simulate the joint probability distribution function of the algebraic expression.

DENSITY FUNCTION APPROXIMATION

Program Name: DENSITY

The density function program is a compatible program used to approximate the density function after the joint probability distribution function has been simulated using the discrete simulation program. The density function is approximated using the first derivative of a cubic spline curve fit and a histogram (See Chapter 3). The density program prompts the user for the name of the joint distribution file. The user is able to control the tolerance of the cubic spline curve fit. With careful adjustment of the tolerance, a smooth density curve can be approximated. The histogram data is printed to work file "W1" and the spline data is printed to a file (designated by the user) that is compatible with the plotting programs.
DATA MANAGEMENT

Program Name: MANAGE

The data management program is used if a problem occurs with the calculation of the cubic spline approximation program. One of the drawbacks of the cubic spline curve fit is that if the data is grouped too close together, the program will halt because of a floating point overflow. The data management program sorts through the data and eliminates any data points that are closer than 0.000001 along the x axis.

GRAPHICS

Program Names: GRAPH

The plotting program is written using Precision Visuals' DI3000 computer graphics package. Program GRAPH plots the density and distribution functions on a Selanar Hirez 100 graphics terminal. The user is asked for the input data file name, title of the graph, and the X and Y axis labels. The plot is dumped to an Hewlett Packard 7470A plotter.
CHAPTER V

COMPARISON

In this chapter, the discrete simulation method discussed in Chapters 3 and 4 is compared to the Monte Carlo method. The joint distribution function of an equation is simulated using both methods and the results are compared to the exact solution of the joint distribution function. The accuracy of each method is measured using two parameters: Mean squared error and maximum deviation.

The mean squared error (MSE) is (See Figure 5.1),

\[
\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \Delta h_i^2
\]

where \( n \) is the number of \( r \) locations. The maximum deviation is equal to the largest absolute value of \( \Delta h \).
Figure 5.1
Approximation and Exact Joint Probability Distribution Functions

Joint Probability Distribution Function, $F(r)$

approximation

exact distribution

$r_{i-1}$ $r_i$ $r_{i+1}$
PROBLEM

Consider the equation of a circle,

\[ r = \sqrt{x^2 + y^2} \quad 0 \leq r \leq \infty \]

where variables \( X \) and \( Y \) are normally distributed independent random variables. Each variable has a mean of zero and a standard deviation equal to one. Find the joint probability distribution function.

EXACT SOLUTION

The marginal probability density functions of \( X \) and \( Y \) are,

\[
 f_X(x) = \begin{cases} 
 \frac{1}{2\pi\sigma^2} \exp \left[ -\frac{(x-\mu)^2}{2\sigma^2} \right] & \text{for } -\infty < x < \infty \\
 0 & \text{otherwise}
\end{cases}
\]

\[
 f_Y(y) = \begin{cases} 
 \frac{1}{2\pi\sigma^2} \exp \left[ -\frac{(y-\mu)^2}{2\sigma^2} \right] & \text{for } -\infty < y < \infty \\
 0 & \text{otherwise}
\end{cases}
\]

when \( \mu_X = 0, \sigma_X^2 = 1 \) and \( \mu_Y = 0, \sigma_Y^2 = 1 \), the marginal
probability density functions become,

\[
\begin{align*}
    f_x(x) & = \begin{cases} 
        \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{x^2}{2} \right) & \text{for } -\infty < x < \infty \\
        0 & \text{otherwise}
    \end{cases} \\
    f_y(y) & = \begin{cases} 
        \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{y^2}{2} \right) & \text{for } -\infty < y < \infty \\
        0 & \text{otherwise}
    \end{cases}
\end{align*}
\]

Since \( X \) and \( Y \) are mutually exclusive independent random variables, their joint probability density function is equal to the product of the two marginal probability density functions. Thus,

\[
f(x,y) = f_x(x)f_y(y)
\]

So,

\[
f(x,y) = \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{x^2}{2} \right) \cdot \frac{1}{\sqrt{2\pi}} \exp \left( -\frac{y^2}{2} \right) = \frac{1}{2\pi} \exp \left( -\frac{(x^2+y^2)}{2} \right)
\]
and the joint probability distribution is,

\[
F(x,y) = \iint_D f(x,y) \, dx \, dy
\]

\[
= \frac{1}{2\pi} \int_0^\infty \int_0^{\sqrt{r^2 - y^2}} e^{-\frac{(x^2 + y^2)}{2}} \, dx \, dy
\]

in polar form this can be written as,

\[
F_R(r) = P(R \leq r) = \frac{1}{2\pi} \int_0^{2\pi} \int_0^r e^{-\frac{r^2}{2}} \, r \, dr \, d\theta
\]

to integrate by parts let,

\[
u = r^2/2 \quad \text{and} \quad du = rdr
\]

\[
P(R \leq r) = \frac{1}{2\pi} \int_0^{2\pi} \int_0^u e^{-u} \, du \, d\theta
\]
\[
\frac{1}{2\pi} \int_0^{2\pi} \frac{-\exp(-u)}{d\theta} \int_0^{2\pi} \left[ -\exp(-\frac{r^2}{2}) + 1 \right] d\theta 
\]

\[
= \frac{1}{2\pi} \left[ -\exp(-\frac{r^2}{2}) + 1 \right] \bigg|_0^{2\pi} 
\]

\[
Pr(R \leq r) = 1 - \exp(-\frac{r^2}{2}) 
\]

The joint probability distribution function is,

\[
F_R(r) = \begin{cases} 
0 & r \leq 0 \\
1 - \exp(-\frac{r^2}{2}) & 0 < r \leq \infty 
\end{cases} 
\]

The exact joint probability distribution function is plotted in Figure 5.2

Differentiating the joint probability distribution function yields the joint probability density function.
Figure 5.2
Exact Joint Probability Distribution Function
\[ f_R(r) = \begin{cases} 0 & r \leq 0 \\ r \exp(-r^2/2) & 0 < r < \infty \end{cases} \]

Figure 5.3 shows a plot of the exact joint probability density function.

MONTE CARLO SIMULATION vs. DISCRETE SIMULATION

A Monte Carlo computer program was developed to compare the accuracies of the two simulations with the exact solution. The Monte Carlo program generates random numbers that are normally distributed to approximate the X and Y random variable's density function. The subroutines used for generating the joint cumulative distribution function are the same subroutines used in the discrete approximation program.

The programs were run three times. The discrete simulation was run first. Each time an increasing number of discrete points was used to represent each random variable's density function. After each run, the number of unique r locations was noted. The Monte Carlo program was then run using the same number of unique r locations. A detailed explanation of the Monte Carlo method is located in Appendix G.

On the first run, 50 discrete points were used to approximate each random variable's density function.
After all possible combinations were calculated, there were 325 unique r locations. Figure 5.4 shows the approximated joint probability distribution function of the discrete approximation. In Figure 5.5, the distribution function is plotted along with the exact solution. The first half of the discrete approximation is almost identical to the exact distribution function. However, in the upper 20% of the curve, the discrete approximation deviates quite a bit from the exact solution. The cause of the deviation could be the small number of discrete points used to represent the random variables density function. Because a relatively small number of points were used, the tails of the density functions were not adequately represented. The mean squared error and the maximum deviation for this approximation was 0.000028 and 0.0196446 respectively.

The Monte Carlo Simulation was then run using 325 unique r locations. The joint distribution function is plotted in Figure 5.6. Figure 5.7 shows the joint probability distribution function for the Monte Carlo approximation and the exact solution. As can be seen, the Monte Carlo approximation is not as good as the discrete approximation. The mean squared error is 0.0014709 and the maximum deviation is 0.0739843.
Figure 5.5
Joint Probability Distribution Functions for the Discrete Simulation using 50 Discrete Points per Random Variable and the Exact Solution
Figure 5.6
Joint Probability Distribution Function for the Monte Carlo Simulation using 325 Unique R Locations
Figure 5.7
Joint Probability Distribution Functions for the Monte Carlo Simulation using 325 Unique R Locations and the Exact Solution
On the second run, 100 discrete points were used to approximate the X and Y random variable's density functions. This simulation resulted in 1275 unique r locations. Figure 5.8 shows the approximation of the second discrete approximation. In Figure 5.9, the discrete distribution curve is plotted with the exact solution. The curve still has a small dip at the upper portion of the curve. However, the maximum deviation has decreased to 0.0101403 and the mean squared error is 0.0000041. The 100 point approximation is an improvement over the 50 point approximation.

Figure 5.10 shows the Monte Carlo approximation of the joint distribution function for 1275 unique r locations. The distribution curve is plotted with the exact solution in Figure 5.11. Although the accuracy of the Monte Carlo simulation has improved, the approximation is still not as good as the discrete approximation. The maximum deviation is equal to 0.0314508 and the mean squared error is 0.0002180.

On the final run, 200 discrete points were used to approximate each variable's density function. This resulted in 5049 unique r locations. The joint distribution function is plotted in Figure 5.12. Figure 5.13 shows the approximated joint distribution function with the exact solution. The two distribution functions
Figure 5.8
Joint Probability Distribution Function for the Discrete Simulation using 100 Discrete Points per Random Variable
Figure 5.9
Joint Probability Distribution Functions for the Discrete Simulation using 100 Discrete Points per Random Variable and the Exact Solution
Figure 5.10
Joint Probability Distribution Function
for the Monte Carlo Simulation using
1275 Unique R Locations
Figure 5.11
Joint Probability Distribution Functions for the Monte Carlo Simulation using 1275 Unique R Locations and the Exact Solution
Figure 5.12
Figure 5.13
Joint Probability Distribution Functions for the Discrete Simulation using 200 Discrete Points per Random Variable and the Exact Solution
appear to be virtually identical. The dip that was present in the 50 and 100 point approximation is no longer noticeable. The error has been reduced to 0.0000006 and the maximum deviation is 0.0051283.

The Monte Carlo approximation for 5049 unique $r$ locations is plotted in Figures 5.14 and 5.15. The Monte Carlo approximation has improved with an increase in $r$ locations. Nevertheless, the discrete approximation is still more accurate than the Monte Carlo approximation. The mean squared error is 0.0000302, which is fifty times larger than the discrete approximation's error. The maximum deviation is three times larger than the discrete approximation at 0.0151759.

In Figure 5.16 and 5.17 the mean squared error and the maximum standard deviation is plotted as a function of unique $r$ locations for both the discrete simulation program and the Monte Carlo program. In Figure 5.16, the discrete approximation approaches zero with little variation as the number of unique $r$ locations increases. The Monte Carlo approximation decreased to zero also, however the distribution is scattered and unpredictable. In Figure 5.17, the maximum deviation of the two simulations decrease as the number of unique $r$ locations increases. Again, the Monte Carlo data is scattered from one value to the next. The general trend of the Monte
Figure 5.14
Joint Probability Distribution Function for the Monte Carlo Simulation using 5049 Unique R Locations
Figure 5.15
Joint Probability Distribution Functions for the Monte Carlo Simulation using 5049 Unique R Locations and the Exact Solution
Figure 5.16
Mean Squared Error vs. Number of Unique R Locations for the Monte Carlo and Discrete Simulation

Monte Carlo

Discrete

Mean Squared Error (x10,000,000)
Figure 5.17

Maximum Deviation vs.
Number of Unique R Locations
for the Monte Carlo and Discrete Simulation

Maximum Deviation

Monte Carlo

Discrete

Number of R Locations
Carlo simulation is a decrease in the mean squared error and the maximum deviation as the number of r locations increase. The large scatter of error and deviation of the Monte Carlo method is caused by the statistical error associated with the generation of the random numbers. Because the generated numbers are random, the variation from one r location to the next is unpredictable.

In Figures 5.18 and 5.19 the mean squared error and the maximum deviation is plotted again for both the discrete simulation and the maximum deviation. However, in Figures 5.18 and 5.19 these values are plotted as functions of computer processing unit time. The figures are very similar to the previous two plots. The discrete simulation data is constantly decreasing and very predictable, while the Monte Carlo data is quite scattered.

A way to minimize the Monte Carlo scatter is to use a larger number of r locations. Increasing the number of r locations would lead to an increase of computing time and money. Therefore, the main advantage of the discrete simulation over the Monte Carlo simulation is that a predictable and more accurate representation of the joint probability distribution function can be approximated using fewer calculations and less computing time.
Figure 5.18
Mean Squared Error vs. Computer Processing Unit Time for the Monte Carlo and Discrete Simulation

Mean Squared Error (×100000)

CPU Time (Seconds)

Monte Carlo

Discrete
CHAPTER VI

CANTILEVER I-BEAM

The beam problem discussed in this section was chosen to illustrate how the discrete simulation program could be used in an engineering application. Each random variable's density function was simulated using 100 discrete points. The "special" distribution subroutine was used to construct the joint probability density functions of algebraic expressions containing more than two random variables.

PROBLEM

The cantilever I-beam, shown in Figures 6.1 and 6.2, is made by welding three steel (ASTM A7) plates together. Determine:

a) The joint probability density and distribution functions for:

1. The maximum moment M
2. The deflection at B
3. The slope at B

72
Figure 6.1a
Cantilever I-Beam
Subjected to a Uniform Load, $W_o$

Figure 6.1b
Free-Body Diagram
of a Cantilever I-Beam
Subjected to a Uniform Load, $W_o$
Figure 6.2
Cross Section of the Cantilever I-Beam
4. The maximum stress

5. The maximum shear

b) The reliability of the I-beam.

The dimensions and metal properties of the I-beam are assumed to be random variables with known probability density functions. These values are listed in Table 6.1.

Table 6.1
Random Variables for a Cantilever I-Beam

<table>
<thead>
<tr>
<th>Random Variable</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Units</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>6.00</td>
<td>0.2500</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>b</td>
<td>0.50</td>
<td>0.0625</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>c</td>
<td>0.50</td>
<td>0.0625</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>d</td>
<td>8.00</td>
<td>0.2500</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>e</td>
<td>6.00</td>
<td>0.2500</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>f</td>
<td>0.50</td>
<td>0.0625</td>
<td>inches</td>
<td>Normal</td>
</tr>
<tr>
<td>E</td>
<td>30E6</td>
<td>4.5E5</td>
<td>psi</td>
<td>Normal</td>
</tr>
<tr>
<td>L</td>
<td>16.00</td>
<td>1.0000</td>
<td>feet</td>
<td>Normal</td>
</tr>
<tr>
<td>W</td>
<td>45.00</td>
<td>33.3333</td>
<td>lb/inch</td>
<td>Uniform</td>
</tr>
<tr>
<td>Su</td>
<td>65.6E3</td>
<td>2.43E3</td>
<td>psi</td>
<td>Normal</td>
</tr>
<tr>
<td>Sy</td>
<td>42.7E3</td>
<td>4.86E3</td>
<td>psi</td>
<td>Normal</td>
</tr>
</tbody>
</table>
**SOLUTION**

a) 1. The maximum moment for a cantilever beam with a uniform load can be expressed as,

\[ M_{\text{max}} = \frac{W_0 L^2}{2} \]  \hspace{1cm} (6.1)

Figures 6.3 and 6.4 show the joint probability distribution and density functions of the maximum moment. The maximum moment ranges from 59,877.02 to 78,638.19 foot-pounds with a mean value of 69,121.85 foot-pounds.

2. The maximum deflection at point B is equal to

\[ \gamma_{\text{max}} = \frac{W_0 L^4}{8EI} \]  \hspace{1cm} (6.2)

Before the deflection can be calculated, the joint probability distribution function of the moment of inertia must be approximated. Using the Parallel-Axis Theorem, the moment of inertia can be expressed as the equation,

\[ I = \frac{(ab^3)}{12} + ab(d/2 + b/2)^2 + \frac{(cd^3)}{12} + \frac{(ef^3)}{12} + fe(f/2 + d/2)^2 \]  \hspace{1cm} (6.3)
Figure 6.3
Joint Probability Distribution Function of the Maximum Moment of a Cantilever I-Beam
Figure 6.4
Joint Probability Density Function of the Maximum Moment of a Cantilever I-Beam

Joint Probability Density Function

Maximum Moment (foot-pounds)
Figures 6.5 and 6.6 show the probability distribution and density functions of the moment of inertia. Once the moment of inertia's distribution has been determined, the joint probability density and distribution functions for the maximum deflection are approximated. Figures 6.7 and 6.8 show the joint probability distribution and density functions of the deflection at point B. The expected value of the deflection is 1.977 inches with a minimum possible value of 1.4 and a maximum value of 2.82 inches.

3. The slope \( \phi \), at point B can be written as,

\[
\phi = \frac{W_o L^3}{6EI}
\]

The joint probability distribution and density functions are plotted in Figures 6.9 and 6.10. The discrete simulation predicts that the mean value of the slope at point B is 0.786 degrees, and the range of possible slope values is from 0.56 to 1.11 degrees.

4. The maximum stress of the cantilever I-beam occurs at the furthest distance from the neutral axis. The equation for the maximum stress is

\[
S_{\text{max}} = \frac{M(d/2+f)}{I}
\]
Figure 6.5
Joint Probability Distribution Function
of the Moment of Inertia
of a Cantilever I-Beam
Figure 6.6
Joint Probability Density Function
of the Moment of Inertia
of a Cantilever I-Beam
Figure 6.7
Joint Probability Distribution Function of the Maximum Deflection of a Cantilever I-Beam
Figure 6.8
Joint Probability Density Function
of the Maximum Deflection
of a Cantilever I-Beam
Figure 6.9
Joint Probability Distribution Function of the Slope at the End of a Cantilever I-Beam

Slope at the End (degrees)

Joint Probability Distribution Function

1.00  0.80  0.60  0.40  0.20  0.00

0.75  0.70  0.65  0.60  0.55  0.50  0.45  0.40  0.35  0.30  0.25  0.20  0.15  0.10  0.05  0.00
Figure 6.10
Joint Probability Density Function of the Slope at the End of a Cantilever I-Beam
The expected value of the maximum stress is 28,944.26 psi. It is interesting to note that the range of possible maximum stress values differ by as much as 22,000 psi. The joint probability density and distribution functions are plotted in Figures 6.11 and 6.12.

5. The maximum value of the first moment of area occurs at the neutral axis. Since the width of the beam is thin at the neutral axis, the maximum shearing stress occurs at this point. The equation for the maximum shear stress can be written as

\[ \tau_{\max} = \frac{W_0 L (ab(b/2+d/2) + cd^2/8)}{I_c} \]  \hspace{1cm} (6.6)

The expected value of the maximum shearing stress is 2,280.38 psi, and the range of possible shear values is from 1,171 to 4,441 psi. The joint probability density and distribution functions are plotted in Figures 6.13 and 6.14.

b) The reliability of the beam is equal to unity minus the probability of failure. If the stress in the beam exceeds the yield stress, the beam is presumed to have failed. Using this criteria, if the ratio of the maximum
Figure 6.11
Joint Probability Distribution Function of the Maximum Stress of a Cantilever I-Beam
Figure 6.12
Joint Probability Density Function of the Maximum Stress of a Cantilever I-Beam
Figure 6.13
Joint Probability Distribution Function of the Maximum Shear Stress of a Cantilever I-Beam
Figure 6.14
Joint Probability Density Function of the Maximum Shear Stress of a Cantilever I-Beam
stress in the beam to the yield stress is greater than one, failure occurs. The joint probability and density functions for the stress ratio are plotted in Figures 6.15 and 6.16. Using the joint probability distribution, the reliability of the system can be determined. From Figure 6.15 the probability of the stress ratio being less than or equal to one is equal to approximately 98%. Therefore, the probability of failure is roughly 2% and the probable reliability of the I-beam structure is 98%.
Figure 6.15
Joint Probability Distribution Function of the Ratio $S_{act}/S_{yield}$ of a Cantilever I-Beam

Stress in the Beam divided by Yield Stress
Figure 6.16
Joint Probability Density Function of the Ratio $\frac{S_{act}}{S_{yield}}$ of a Cantilever I-Beam

Stress in the Beam divided by Yield Stress
CHAPTER VII

BOUNDING

During the development of the discrete simulation algorithm, a new technique was investigated that approximates the upper and lower bounds of a joint probability distribution function of an algebraic expression containing several independent random variables. Although a formal proof of the bounding theory has not been written, the results of the program have been compared with several known solutions with good results. In this chapter, the procedure used to obtain the upper and lower bounds of a joint probability distribution function is discussed.

Using the same method of cell development as the discrete simulation technique discussed in Chapter 3, two curves are developed that bound the exact joint probability distribution function of an algebraic expression. The bounding approximation divides the two known random variables' density functions into a given
number of cells. The upper and lower bounds of each cell are noted as well as the cell's center of gravity (See Figure 7.1).

M and n are arbitrary cell divisions within the random variables', $X_1$ and $X_2$, known density functions. Since each cell is represented using three locations,
there are nine possible outcomes that need to be considered.

\[
\begin{align*}
Z_1 &= Z(X_1(a_{n-1}), X_2(b_{m-1})) \\
Z_2 &= Z(X_1(a_{n-1}), X_2(b_{cg})) \\
Z_3 &= Z(X_1(a_{n-1}), X_2(b_m)) \\
Z_4 &= Z(X_1(a_{cg}), X_2(b_{m-1})) \\
Z_5 &= Z(X_1(a_{cg}), X_2(b_{cg})) \\
Z_6 &= Z(X_1(a_{cg}), X_2(b_m)) \\
Z_7 &= Z(X_1(a_n), X_2(b_{m-1})) \\
Z_8 &= Z(X_1(a_n), X_2(b_{cg})) \\
Z_9 &= Z(X_1(a_n), X_2(b_m))
\end{align*}
\]

The nine values are compared with each other. The maximum value is stored in the lower \(Z\) bounding array and the minimum value is stored in the upper \(Z\) bounding array. Each bounding distribution is then assembled independently using the same distribution and sort subroutines used for the discrete approximation. Figure 7.2 shows the upper and lower bounds of a hypothetical joint distribution function.
The bounding program was compared with the exact solution of the probabilistic pythagorean random variable discussed in Chapter 5. The bounding approximation bounded both the exact solution as well as the approximated joint probabilistic distribution function.
Figure 7.3 shows the exact solution and the upper and lower bounding curves using 100 cells per random variable.

The bounding routine was also compared to Wu's fast probability integration (FPI) approximation of the probability of failure of a bar in tension. The probability of failure, \( P_f \) is symbolized as

\[
P_f = P(g \leq 0) \quad (7.2)
\]

where \( g \) is equal to the equation

\[
g = R - \left(\frac{4T}{\pi D^2}\right) \quad (7.3)
\]

\( R \) and \( D \) are independent normal random variables. \( R \) has a mean of 170.0 N/mm and a variance of 625.0 N/mm. \( D \) has a mean and variance of 29.4 mm and 9.0 mm respectively. \( T \) is equal to 50 kN. Table 7.1 lists Wu's approximation as well as the exact solution and the upper and lower bounding approximations. An example of the output using ten cells per random variable is listed in the Appendix.

Table 7.1 shows that as the number of cell divisions per density function increase for each random variable,
Figure 7.3
The Exact Solution and the Bounding Curves for the Probabilistic Pythagorean Random Variable Using 100 Discrete Points

- Upper Bound
- Lower Bound

Exact Solution (Rayleigh)
the upper and lower bounds converge toward the exact solution.

Table 7.1

Bounding Probabilities of Failure of a Bar in Tension

<table>
<thead>
<tr>
<th>Number of Discrete Points/ Density Function</th>
<th>Lower Bound</th>
<th>Exact Solution</th>
<th>FPI Approximation</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.00030</td>
<td>0.00230</td>
<td>0.0029</td>
<td>0.0114</td>
</tr>
<tr>
<td>120</td>
<td>0.00032</td>
<td></td>
<td></td>
<td>0.0098</td>
</tr>
<tr>
<td>150</td>
<td>0.00043</td>
<td></td>
<td></td>
<td>0.0083</td>
</tr>
<tr>
<td>200</td>
<td>0.00061</td>
<td></td>
<td></td>
<td>0.0068</td>
</tr>
</tbody>
</table>

The subroutines, Bookbound and Outputbound, that approximate the upper and lower bounds are listed in the Appendix.
CHAPTER VIII

CONCLUSIONS

The purpose of this study was to develop a computer simulation package that is capable of approximating the joint probability distribution function of a complex algebraic expression containing several independent random variables. A discrete simulation technique was developed and implemented on a Harris H-800 Super Mini-Computer.

In Chapter 3 the discrete simulation technique is discussed. Although the discrete simulation method is similar to the Monte Carlo technique, the discrete method divides each random variable's known density function into a given number of n discrete sections with equal areas. Each area is then represented by a single discrete point with a probability of $1/n$. Using the discrete points, all possible combinations of the algebraic expression are simulated. After all the outcomes have been computed, the joint distribution
function is assembled by sorting the data and summing the ordered values probabilities. The approximate shape of the density function is simulated using cubic splines and a histogram.

The computer programs that are used to approximate the joint probability distribution and density functions are described in Chapter 4. Because of storage problems, the simulation package had to be divided into four separate programs. The programs are listed in the Appendix.

In Chapter 5 the discrete approximation is compared to the traditional Monte Carlo technique for a joint probability distribution that can be solved exactly. The joint probability distribution function of the equation of a circle is approximated using both methods and the results are compared to the exact solution of the joint distribution function. The programs were run three times. Each time an increasing number of points was used to represent each random variable. The joint probability distributions are plotted vs. the exact solution for each case. As expected, the accuracy of each method improves with an increase of number of points used to represent each random variable's density function. However, the accuracy of the discrete simulation is better for each
case than the Monte Carlo simulation. The mean squared error and the maximum deviation is plotted as a function of number of unique outcomes and computer processing unit (CPU) time. The discrete simulation plots are very well behaved and consistently decreasing with an increasing number of outcomes and CPU time, while the Monte Carlo data is scattered and erratic.

In Chapter 6 an example of how the discrete simulation method can be used as a tool by engineers to aid in the design and development of structures and components is presented. A cantilever I-beam is analyzed. Each variable is assumed to be random with a known probability distribution. Using the discrete simulation programs, the joint probability and density functions are approximated for the maximum moment, the deflection at the end of the beam, the slope at the end of the beam, the maximum stress, the maximum shear, and the reliability of the beam.

In Chapter 7 a different technique is discussed that determines the upper and lower bounds of a joint probability distribution function using the discrete approximation. Although a formal proof has not been developed for the bounding theory, the program was run for several known cases with good results.
In summary, a discrete simulation algorithm was developed to approximate the joint probability distribution function of complex algebraic expressions containing several independent random variables. Although there are a few methods available for approximating the joint probability distribution functions of algebraic expressions, they usually involve complex mathematical operations that are too cumbersome for the average design engineer. The discrete simulation method developed in this study does not involve any complex mathematical operations, is easy to use, and is capable of being used in a wide variety of engineering applications.
CHAPTER IX

RECOMMENDATIONS FOR FURTHER STUDY

There are many possible extensions of the discrete simulation method that can be investigated following this study. A few suggestions are to:

1. Develop a faster sorting routine which would decrease computing time.
2. Expand the program to allow the use of more discrete points to represent each random variable's density function.
3. Modify the program so that the variables are represented using double precision.
4. Investigate the accuracy of the discrete simulation program for an algebraic expression containing more than two random variables in which an exact solution exists.
5. Develop a formal proof of the bounding theory discussed in Chapter 7.
LIST OF REFERENCES


14. ACI Committee 318, "Building Code Requirements for Reinforced Concrete (ACI 318-63)," American Concrete Institute, Detroit, 1963, 144 pp.

15. ACI Committee 318, "Building Code Requirements for Reinforced Concrete (ACI 318-71)," American Concrete Institute, Detroit, 1971, 78 pp. Also ACI 318-77, 1977, 104 pp.


APPENDIX A

DISCRETE SIMULATION PROGRAM
C ***********************************************************************
C * PROGRAM NAME: DISCSIM (DISCRETE SIMULATION AND BOUNDING) *
C * WRITTEN BY: BRUCE SWANSON *
C * DATE: MARCH 26, 1987 *
C * PURPOSE: PROGRAM "DISCSIM" APPROXIMATES THE JOINT PROBABILITY *
C * DISTRIBUTION FUNCTION OR THE UPPER AND LOWER BOUNDING *
C * CURVES OF A DISTRIBUTION FUNCTION FOR AN ALGEBRAIC *
C * EXPRESSION CONTAINING TWO INDEPENDENT RANDOM VARIABLES. *
C * THE RANDOM VARIABLE'S DENSITY FUNCTIONS ARE APPROXIMATED *
C * USING A FINITE NUMBER (2-200) OF DISCRETE LOCATIONS. THE *
C * FUNCTION MUST BE INSERTED IN SUBROUTINE "FUNCTION". *
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES BOOK *
C * BOOKBOUND *
C * CELLR *
C * DIST *
C * DISTRIBUTION *
C * FUNCTION *
C * NORMAL *
C * NORMINV *
C * OUTPUT *
C * OUTPUTBOUND *
C * SORT *
C * SPECIAL *
C * TRIANGLE *
C * UNIFORM *
C *
C * DESCRIPTION OF PARAMETERS: *
C * INPUT ARGUMENTS: *
C * ANSB - CHARACTER*1 - IF ANSB IS EQUAL TO 'Y' THEN THE *
C * PROGRAM COMPUTES THE UPPER AND LOWER BOUNDS OF *
C * THE JOINT DISTRIBUTION FUNCTION AS WELL AS THE *
C * DISCRETE APPROXIMATION FUNCTION. IF ANSB IS *
C * EQUAL TO 'N' THEN THE PROGRAM COMPUTES ONLY THE *
C * DISCRETE APPROXIMATION OF THE JOINT *
C * DISTRIBUTION FUNCTION. *
C * DISREP(NUM) - CHARACTER*10 - DISREP CONTAINS THE *
C * TYPE OF PROBABILITY DISTRIBUTION FOR VARIABLE *
C * NUM (NORMAL, UNIFORM, TRIANGULAR, SPECIAL). *
C * DISTR(NUM) - INTEGER - THE NUMBER OF THE DISTRIBUTION *
C * FOR RANDOM VARIABLE NUM (1=NORMAL, 2=UNIFORM, *
C * 3=TRIANGULAR, 4=SPECIAL). *
C * NAME - CHARACTER*30 - NAME IS THE INPUT DATA FILE NAME *
C * OF THE SPECIAL DISTRIBUTION FUNCTION FOR *
RANDOM VARIABLE NUM.

NAME2 - CHARACTER*30 - NAME2 IS THE OUTPUT DATA FILE
NAME OF THE JOINT DISTRIBUTION FUNCTION.

NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
LOCATIONS THAT APPROXIMATE THE PROBABILITY
DENSITY FUNCTION OF RANDOM VARIABLE NUM.

NUM - INTEGER - THE NUMBER OF THE RANDOM VARIABLE.
VARIANCE - REAL - THE VARIANCE VALUE FOR RANDOM
VARIABLE NUM.

OUTPUT ARGUMENTS:

A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N
REPRESENT THE CENTER OF GRAVITY LOCATIONS AND
EVEN VALUES OF N INDICATE THE CELL BOUNDARIES.

B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
DENSITY FUNCTION.

COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
FUNCTION.

LB(NCELLS( ),2) - REAL - CONTAINS THE LOWER Bound
LOCATION FOR CELL NCELLS( ).

LB_COUNT - INTEGER - LB_COUNT IS THE NUMBER OF UNIQUE
LB( , ) LOCATIONS OF THE LOWER Bound
DISTRIBUTION FUNCTION.

MEAN(NUM) - REAL - THE MEAN VALUE FOR RANDOM VARIABLE
NUM.

UB(NCELLS( ),2) - REAL - CONTAINS THE UPPER Bound
LOCATION FOR CELL NCELLS( ).

UB_COUNT - INTEGER - UB_COUNT IS THE NUMBER OF UNIQUE
UB( , ) LOCATIONS OF THE UPPER Bound
DISTRIBUTION FUNCTION.

Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
PROBABILITY DISTRIBUTION FUNCTION.

Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO Z(COUNT,1).

Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES Z(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE Z VALUE OF THE LOWER Bound
OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZMEAN - REAL - THE MEAN OF THE JOINT PROBABILITY
DISTRIBUTION FUNCTION.
C *
ZSTANDARD_DEV - REAL - THE STANDARD DEVIATION OF THE
JOINT PROBABILITY DISTRIBUTION Z.
ZUB(COUNT,1) - REAL - THE Z VALUE OF THE UPPER BOUND
OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
ZUB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZUB(COUNT,1).
ZUB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZUB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.
ZVARIANCE - REAL - THE VARIANCE OF THE JOINT
PROBABILITY DISTRIBUTION FUNCTION.
C *
C  **********************************************************************************************************
C INTEGER COUNT,BX,DISTR,NCELLS,NUM,N,UBCOUNT,LBCOUNT
REAL A,B,Z,AREA,PROB,X,TOLERANCE,F,D,SX,W
REAL VARIANCE,MEAN,ZMEAN,ZVARIANCE,UB,LB
CHARACTER DISREP*10,NAME*20,NAME2*20,ANSB*1
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
OPEN (UNIT=23,FILE=`W2')
320 WRITE (3,*)
WRITE (3,300)
300 FORMAT (2X,`SIMULATION AND BOUNDING (Y/N) ?')
READ (0,310) ANSB
310 FORMAT (A1)
IF (ANSB.NE.``Y'.AND.ANSB.NE.``N') THEN
GOTO 320
END IF
DO 10 NUM=1,2
CALL DIST
CONTINUE
10 IF (ANSB.EQ.``Y') THEN
CALL BOOKBOUND
CALL OUTPUTBOUND
ELSE
CALL BOOK
CALL OUTPUT
END IF
CLOSE 23
STOP
END
SUBROUTINE BOOK
C
C **********************************************************************
C * PROGRAM NAME: SUBROUTINE BOOK
C * WRITTEN BY: BRUCE SWANSON
C * DATE: MARCH 4, 1987
C *
C * PURPOSE: SUBROUTINE BOOK SIMULATES THE JOINT DISTRIBUTION FUNCTION
C * OF TWO RANDOM VARIABLES FOR THE DISCRETE APPROXIMATION.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES DISTRIBUTION
C * FUNCTION
C *
C * USAGE: CALL BOOK
C *
C * DISCRITION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
* GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
* PROBABILITY DENSITY FUNCTION.
C *
NCCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
* LOCATIONS USED TO APPROXIMATE THE PROBABILITY
* DENSITY FUNCTION OF RANDOM VARIABLE NUM.
C *
NUM - INTEGER - VARIABLE NUM IS THE NUMBER OF THE
* RANDOM VARIABLE.
C *
C * OUTPUT ARGUMENTS:
C *
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
* LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
* FUNCTION.
C *
TIME - INTEGER*3 - VARIABLE TIME IS THE TOTAL CPU TIME
* OF SIMULATION.
C *
Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
* PROBABILITY DISTRIBUTION FUNCTION.
C *
Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
* THAN OR EQUAL TO Z(COUNT,1).
C *
Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
* TIMES Z(COUNT,1) WAS REPEATED DURING THE
* SIMULATION.
C *
ZMEAN - REAL - THE MEAN OF THE JOINT PROBABILITY
* DISTRIBUTION FUNCTION.
C *
ZVARIANCE - REAL - THE VARIANCE OF THE JOINT
* PROBABILITY DISTRIBUTION FUNCTION.
C *
C **********************************************************************
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/BOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
REAL B,Z,SX,F,D,WORK,A,VARIANCE,MEAN,ZMEAN,ZVARIANCE,TOTAL
REAL ZZ,UB,LB
INTEGER COUNT,FLAG,AX,X,Y,NCCELLS,NUM,N,DISTR,UBCOUNT,LBCOUNT
INTEGER*3 TIME
CHARACTER DISREP*10,ANSB*1
COUNT=0
AX=0
TOTAL=0.0
CALL STIME
Z(0,1)=-1.0E30

SIMULATE ALL POSSIBLE COMBINATIONS OF THE FUNCTION Z

DO 10 X=1,NCCELLS(1)
   DO 20 Y=1,NCCELLS(2)
      AX=AX+1
      CALL FUNCTION (B(X,1),B(Y,2),ZZ)
      CALL SORT (ZZ,Z,COUNT,AX)
      TOTAL=TOTAL+ZZ
20 CONTINUE
10 CONTINUE

END OF THE SIMULATION

CALL WTIME
CALL WTIME2 (TIME)
WRITE (3,*) TIME
ZMEAN=TOTAL/FLOAT(NCELLS(1)*NCCELLS(2))

CALCULATE THE JOINT PROBABILITY DISTRIBUTION FUNCTION

CALL DISTRIBUTION (Z,COUNT,ZMEAN,ZVARIANCE,NCCELLS(1),
*  
RETURN
END

NCELLS(2))
SUBROUTINE BOOKBOUND

C ************************************************************************************************
C * PROGRAM NAME: SUBROUTINE BOOKBOUND
C * WRITTEN BY: BRUCE SWANSON
C * DATE: MARCH 4, 1987
C *
C * PURPOSE: SUBROUTINE BOOK SIMULATES THE JOINT DISTRIBUTION FUNCTION
C * OF THE TWO RANDOM VARIABLES.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES DISTRIBUTION
C * FUNCTION
C *
C *
C * USAGE: CALL BOOK
C *
C *
C * DISCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
BOUND(3,2) - REAL - ARRAY BOUND IS A TEMPORARY VARIABLE
THAT COMPUTES THE MINIMUM AND MAXIMUM
COMBINATIONS OF THE TWO CELLS REPRESENTING THE
TWO RANDOM VARIABLES.
B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
DENSITY FUNCTION.
LB(NCELLS(),2) - REAL - CONTAINS THE LOWER BOUND
LOCATION FOR CELL NCELLS().
NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
LOCATIONS USED TO APPROXIMATE THE PROBABILITY
DENSITY FUNCTION OF RANDOM VARIABLE NUM.
NUM - INTEGER - VARIABLE NUM IS THE NUMBER OF THE
RANDOM VARIABLE.
UB(NCELLS(),2) - REAL - CONTAINS THE UPPER BOUND
LOCATION FOR CELL NCELLS().
*
C * OUTPUT ARGUMENTS:
C *
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
FUNCTION.
LBCOUNT - INTEGER - LBCOUNT IS THE NUMBER OF UNIQUE
LB( , ) LOCATIONS OF THE LOWER BOUND
DISTRIBUTION FUNCTION.
UBCOUNT - INTEGER - UBCOUNT IS THE NUMBER OF UNIQUE
UB( , ) LOCATIONS OF THE UPPER BOUND
DISTRIBUTION FUNCTION.
TIME - INTEGER*3 - VARIABLE TIME IS THE TOTAL CPU TIME
OF SIMULATION.
**Z(COUNT,1)** - REAL - THE Z VALUE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**Z(COUNT,2)** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO Z(COUNT,1).
**Z(COUNT,3)** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES Z(COUNT,1) WAS REPEATED DURING THE SIMULATION.

**ZLB(COUNT,1)** - REAL - THE Z VALUE OF THE LOWER BOUND OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZLB(COUNT,2)** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZLB(COUNT,1).
**ZLB(COUNT,3)** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZLB(COUNT,1) WAS REPEATED DURING THE SIMULATION.

**ZMEAN** - REAL - THE MEAN OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZUB(COUNT,1)** - REAL - THE Z VALUE OF THE UPPER BOUND OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZUB(COUNT,2)** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZUB(COUNT,1).
**ZUB(COUNT,3)** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZUB(COUNT,1) WAS REPEATED DURING THE SIMULATION.

**ZVARIANCE** - REAL - THE VARIANCE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

**ZSETUP** - REAL - THE Z VALUE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

**ZLB** - REAL - THE Z VALUE OF THE LOWER BOUND OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZLB** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZLB.
**ZLB** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZLB WAS REPEATED DURING THE SIMULATION.

**ZSETUP** - REAL - THE Z VALUE OF THE UPPER BOUND OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZSETUP** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZSETUP.
**ZSETUP** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZSETUP WAS REPEATED DURING THE SIMULATION.

**ZMEAN** - REAL - THE MEAN OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZUB** - REAL - THE Z VALUE OF THE UPPER BOUND OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZUB** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZUB.
**ZUB** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZUB WAS REPEATED DURING THE SIMULATION.

**ZVARIANCE** - REAL - THE VARIANCE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

**ZSETUP** - REAL - THE Z VALUE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.
**ZSETUP** - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO ZSETUP.
**ZSETUP** - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES ZSETUP WAS REPEATED DURING THE SIMULATION.
FORMAT (2X,'START SIMULATION')
CALL STIME
Z(0,1)=-1.0E30

SIMULATE ALL POSSIBLE COMBINATIONS OF THE FUNCTION Z

DO 10 X=1,NCELLS(1)
  WRITE (3,*) X
  DO 20 Y=1,NCELLS(2)
    AX=AX+1
    LBX=LBX+1
    UBX=UBX+1
    CALL FUNCTION (B(X,1),B(Y,2),ZZ)
    CALL SORT (ZZ,Z,COUNT,AX)
    ZMIN=1.0E38
    ZMAX=-1.0E38
    BOUND(1,1)=UB(X,1)
    BOUND(2,1)=LB(X,1)
    BOUND(3,1)=B(X,1)
    BOUND(1,2)=UB(Y,2)
    BOUND(2,2)=LB(Y,2)
    BOUND(3,2)=B(Y,2)
    DO 8000 K=1,3
      DO 8010 KK=1,3
        CALL FUNCTION (BOUND(K,1),BOUND(KK,2),ZT)
        IF (ZT.LT.ZMIN) THEN
          ZMIN=ZT
        END IF
        IF (ZT.GT.ZMAX) THEN
          ZMAX=ZT
        END IF
      8010 CONTINUE
    8000 CONTINUE
    CALL SORT(ZMIN,ZUB,UBCOUNT,UBX)
    CALL SORT(ZMAX,ZLB,LBCOUNT,LBX)
    TOTAL=TOTAL+ZZ
    UBTOTAL=UBTOTAL+ZMIN
    LBTOTAL=LTOTAL+ZMAX
  20 CONTINUE
10 CONTINUE

END OF THE SIMULATION
CALL WTIME
CALL WTIME2 (TIME)
WRITE (3,*) TIME
WRITE (3,124)

FORMAT (2X, "END OF SIMULATION")
ZMEAN=TOTAL/FLOAT(NCELLS(1)*NCELLS(2))
UBMEAN=UBTOTAL/FLOAT(NCELLS(1)*NCELLS(2))
LMEAN=LBTOTAL/FLOAT(NCELLS(1)*NCELLS(2))

CALCULATE THE JOINT PROBABILITY DISTRIBUTION FUNCTIONS
FOR THE DISCRETE APPROXIMATION AND THE TWO BOUNDING CURVES.

CALL DISTRIBUTION (Z,COUNT,ZMEAN,ZVARIANCE,NCELLS(1),
NCELLS(2))
* CALL DISTRIBUTION (ZLB,LBCOUNT,LBMEAN,VBVARIANCE,NCELLS(1),
NCELLS(2))
* CALL DISTRIBUTION (ZUB,UBCOUNT,UBMEAN,UBVARIANCE,NCELLS(1),
NCELLS(2))
RETURN
END
SUBROUTINE CELLR

C ******************************************************************************
C *
C * PROGRAM NAME: SUBROUTINE CELLR
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 3, 1987
C *
C * PURPOSE: SUBROUTINE CELLR COMPUTES THE CENTER OF GRAVITY LOCATIONS
C * FOR A GIVEN CELL WITH A KNOWN PROBABILITY DISTRIBUTION.
C *
C * USAGE: CALL CELLR
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N
* REPRESENT THE CENTER OF GRAVITY LOCATIONS, AND
* THE EVEN VALUES OF N INDICATE THE CELL
* BOUNDARIES.
C *
ANSB - CHARACTER*1 - IF ANSB IS EQUAL TO 'Y' THEN THE
* PROGRAM COMPUTES THE UPPER AND LOWER BOUNDS OF
* THE JOINT DISTRIBUTION FUNCTION AS WELL AS THE
* DISCRETE APPROXIMATION FUNCTION. IF ANSB IS
* EQUAL TO 'N' THEN THE PROGRAM COMPUTES ONLY THE
* DISCRETE APPROXIMATION OF THE JOINT
* DISTRIBUTION FUNCTION.
C *
NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
* LOCATIONS THAT APPROXIMATE THE PROBABILITY
* DENSITY FUNCTION OF RANDOM VARIABLE NUM.
C *
NUM - INTEGER - VARIABLE NUM IS THE NUMBER OF THE
* RANDOM VARIABLE.
C *
C * OUTPUT ARGUMENTS:
C *
B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
* GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
* DENSITY FUNCTION.
C *
LB(NCELLS( ),2) - REAL - CONTAINS THE LOWER BOUND
* LOCATION FOR CELL NCELLS( ).
C *
UB(NCELLS( ),2) - REAL - CONTAINS THE UPPER BOUND
* LOCATION FOR CELL NCELLS( ).
C *
C * OUTPUT ARGUMENTS:
C *
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
* LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
* FUNCTION.
C *
121
COMMON FIRST / A(0:400), B(0:400, 2), Z(0:40000, 3), COUNT
COMMON SECOND / DISTR(2), NCELLS(2), VariANCE(2), MEAN(2), DISREP(2)
COMMON THIRD / NUM, ZMEAN, ZVARIANCE, UB(0:400, 2), LB(0:400, 2), ANSB
COMMON / CBOUND / ZLB(0:40000, 3), ZUB(0:40000, 3), UBCOUNT, LBCOUNT
INTEGER BX, NCELLS, NUM, COUNT, DISTR, UBCOUNT, LBCOUNT
REAL A, B, Z, VARIANCE, MEAN, ZMEAN, ZVARIANCE, UB, LB
CHARACTER DISREP*10, ANSB*1
WRITE (3, 2)
2 FORMAT (2X, 'ENTERED CELLR')
BX = 0
DO 80 K = 1, NCELLS(NUM)*2, 2
   BX = BX + 1
   B(BX, NUM) = A(K)
80 CONTINUE
C
C ----------------- FOR BOUNDING -----------------
C
IF (ANSB.EQ. 'Y') THEN
   BX = 1
   DO 10 I = 0, NCELLS(NUM)*2, 2
      LB(BX, NUM) = A(I)
      UB(BX, NUM) = A(I+2)
      BX = BX + 1
10 CONTINUE
END IF
C
C ---------------- END FOR BOUNDING -----------------
C
RETURN
END
SUBROUTINE DIST
C ***********************************************************************
C *
C * PROGRAM NAME: SUBROUTINE DIST
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 3, 1987
C *
C * PURPOSE: SUBROUTINE DIST Initializes the type of probability
C * distribution for each random variable.
C *
C * USAGE: CALL DIST
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES CELLR
C * NORMAL
C * NORMINV
C * TRIANGEL
C * UNIFORM
C * SPECIAL
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
DISTR(NUM) - INTEGER - The number of the distribution
FOR RANDOM VARIABLE NUM (1=NORMAL, 2=UNIFORM,
3=TRIANGULAR, 4=SPECIAL).
NUM - INTEGER - The number of the random variable.
*
C *
C * OUTPUT ARGUMENTS:
C *
A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N
REPRESENT THE CENTER OF GRAVITY LOCATIONS AND
EVEN VALUES OF N INDICATE THE CELL BOUNDARIES.
DISREP - CHARACTER*10 - DISREP CONTAINS THE TYPE OF
PROBABILITY DISTRIBUTION FOR VARIABLE NUM
(NORMAL, UNIFORM, TRIANGULAR, SPECIAL).
NCELLS(NUM) - INTEGER - The number of discrete
LOCATIONS THAT APPROXIMATE THE PROBABILITY
DENSITY FUNCTION OF RANDOM VARIABLE NUM.
*
C *
C ***********************************************************************
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUH,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
INTEGER DISTR,NCELLS,COUNT,UBCOUNT,LBCOUNT
REAL VARIANCE,MEAN,A,B,Z,ZMEAN,ZVARIANCE,UB,LB
CHARACTER DISREP*10,ANSB*1

123
WRITE (3,*)
WRITE (3,10)

FORMAT (2X,'DISTRIBUTIONS')
WRITE (3,20)

FORMAT (2X,'1 NORMAL')
WRITE (3,30)

FORMAT (2X,'2 UNIFORM')
WRITE (3,40)

FORMAT (2X,'3 TRIANGULAR')
WRITE (3,50)

FORMAT (2X,'4 SPECIAL')
WRITE (3,80) NUM

FORMAT (2X,'ENTER NUMBER OF DISTRIBUTION FOR VARIABLE',13)
READ (0,*) DISTR(NUM)

IF (DISTR(NUM).GT.4) GOTO 5
IF (DISTR(NUM).EQ.1) THEN
  DISREP(NUM)='NORMAL'
  CALL NORMAL
ELSE
  IF (DISTR(NUM).EQ.2) THEN
    DISREP(NUM)='UNIFORM'
    CALL UNIFORM
  ELSE
    IF (DISTR(NUM).EQ.3) THEN
      DISREP(NUM)='TRIANGULAR'
      CALL TRIANGLE
    ELSE
      IF (DISTR(NUM).EQ.4) THEN
        DISREP(NUM)='SPECIAL'
        CALL SPECIAL
      ELSE
        GOTO 5
      END IF
    END IF
  END IF
END IF
END IF
RETURN
END
SUBROUTINE DISTRIBUTION (Z,COUNT,MEAN,VARIANCE,NC1,NC2)
C ***************************************************************
C * * PROGRAM NAME: SUBROUTINE DISTRIBUTION *
C *
C * WRITTEN BY: BRUCE SWANSON *
C *
C * DATE: MARCH 23, 1987 *
C *
C * PURPOSE: SUBROUTINE DISTRIBUTION COMPUTES THE JOINT DISTRIBUTION *
C * FUNCTION OF RANDOM VARIABLE Z. *
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE *
C *
C * USAGE: CALL DISTRIBUTION(Z,COUNT,MEAN,VARIANCE,NC1,NC2) *
C *
C * DESCRIPTION OF PARAMETERS: *
C * INPUT ARGUMENTS: *
C * COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z *
C * LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION *
C * MEAN - REAL - THE MEAN OF THE JOINT PROBABILITY *
C * DISTRIBUTION FUNCTION. *
C * NC1 - INTEGER - THE NUMBER OF DISCRETE LOCATIONS THAT *
C * APPROXIMATE THE PROBABILITY DENSITY FUNCTION *
C * OF RANDOM VARIABLE 1. *
C * NC2 - INTEGER - THE NUMBER OF DISCRETE LOCATIONS THAT *
C * APPROXIMATE THE PROBABILITY DENSITY FUNCTION *
C * OF RANDOM VARIABLE 2. *
C * Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT *
C * PROBABILITY DISTRIBUTION FUNCTION. *
C * Z(COUNT,2) - REAL - THE NUMBER OF TIMES THE VALUE OF *
C * TIMES Z(COUNT,1) WAS REPEATED DURING THE *
C * SIMULATION. *
C *
C * OUTPUT ARGUMENTS: *
C * VARIANCE - REAL - THE VARIANCE OF THE JOINT PROBABILITY *
C * DISTRIBUTION FUNCTION. *
C * Z(COUNT,3) - REAL - THE PROBABILITY OF Z BEING LESS *
C * THAN OR EQUAL TO Z(COUNT,1). *
C * TIMES Z(COUNT,1) WAS REPEATED DURING THE *
C * SIMULATION. *
C *
C ***************************************************************
C
DIMENSION Z(0:40000,3)
REAL Z,ZT,MEAN,VARIANCE
INTEGER COUNT,NC1,NC2

125
ZT=0.0
VARIANCE=0.0
DO 80 I=1,COUNT
    Z(I,3)=Z(I,2)
    ZT=ZT+Z(I,3)
    Z(I,2)=ZT/FLOAT(NC1*NC2)
    VARIANCE=((Z(I,1)-MEAN)**2.0)*(Z(I,3)/FLOAT(NC1*NC2)) + VARIANCE
80 CONTINUE
RETURN
END
SUBROUTINE FUNCTION (X1,X2,ANSZ)
C *-------------------------------------------------------------------------------------------------------------------------------
C * PROGRAM NAME: SUBROUTINE FUNCTION
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 23, 1987
C *
C * PURPOSE: SUBROUTINE FUNCTION COMPUTES THE VALUE OF ANSZ GIVEN
C * THE VARIABLES X1 AND X2.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE
C *
C * USAGE: CALL FUNCTION (X1,X2,ANSZ)
C *
C * DISCRITION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C * X1 - REAL - THE VALUE OF THE FIRST VARIABLE.
C * X2 - REAL - THE VALUE OF THE SECOND VARIABLE.
C *
C * OUTPUT ARGUMENTS:
C * ANSZ - REAL - THE VALUE OF THE FUNCTION GIVEN THE
C * VALUES OF THE TWO RANDOM VARIABLES X1 AND X2.
C *
C *-------------------------------------------------------------------------------------------------------------------------------
C
REAL X1,X2,ANSZ
PI=3.141592654

-------------------------------------------------------------------------------------------------------------------------------

ENTER FUNCTION TO BE SIMULATED
  X1 = RANDOM VARIABLE 1
  X2 = RANDOM VARIABLE 2
  ANSZ = JOINT RANDOM VARIABLE

-------------------------------------------------------------------------------------------------------------------------------

ANSZ=(X1/2.0)+X2
ANSZ=SQRT(X1**2+X2**2)
ANSZ=X1*X2*12.0
ANSZ=2.5-(X1*X2)
ANSZ=(X1/2.0+X2/2.0)
ANSZ=(X1*X2**2.0)/2.0
ANSZ=(X1*(X2**3.0))/6.0
ANSZ=X1*(X2**2.0/8.0)
ANSZ=X1*(1/X2)
ANSZ=X1-((4*50000)/(3.141592654*X2**2))
ANSZ = X1 / (1 + X2)
ANSZ = LOG10((1 + X1) / X2)
ANSZ = X1 - ((4.0 * 50000) / (PI * X2))
ANSZ = SQRT(X1**2 + X2**2)

RETURN
END
SUBROUTINE NORMAL
C ******************************************************************************
C  PROGRAM NAME: SUBROUTINE NORMAL
C  WRITTEN BY: BRUCE SWANSON
C  DATE: MARCH 3, 1987
C  PURPOSE: SUBROUTINE NORMAL FINDS THE NCELL LOCATION ON THE X AXIS
         GIVEN A PROBABILITY (AREA) FOR A NORMAL RANDOM VARIABLE.
C  SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES CELLR
         NORMINV
C  USAGE: CALL NORMAL
C  DISCRIPION OF PARAMETERS:
C  INPUT ARGUMENTS:
C      MEAN(NUM) - REAL - THE MEAN VALUE FOR RANDOM VARIABLE NUM.
C      NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
      LOCATIONS THAT APPROXIMATE THE PROBABILITY
      DENSITY FUNCTION OF RANDOM VARIABLE NUM.
C      NUM - INTEGER - THE NUMBER OF THE RANDOM VARIABLE.
C      VARIANCE - REAL - THE VARIANCE VALUE FOR RANDOM VARIABLE NUM.
C  OUTPUT ARGUMENTS:
C      A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N
      REPRESENT THE CENTER OF GRAVITY LOCATIONS AND
      THE EVEN VALUES OF N INDICATE THE CELL
      BOUNDARIES.
C      B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
      GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
      DENSITY FUNCTION.
C ******************************************************************************

COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
REAL AREA,A,X,PROB,KAREA,MEAN,VARIANCE,B,Z
REAL ZMEAN,ZVARIANCE,UB,LB
INTEGER NCELLS,K,NUM,COUNT,DISTR,UBCOUNT,LBCOUNT
CHARACTER DISREP*10,ANSB*1
WRITE (3,*)

129
WRITE (3,*)
WRITE (3,20) NUM
20 FORMAT (2X,'NORMAL DISTRIBUTION FOR VARIABLE',I3)
WRITE (3,30)
30 FORMAT (2X,'ENTER MEAN')
READ (0,*) MEAN(NUM)
WRITE (3,40)
40 FORMAT (2X,'ENTER STANDARD DEVIATION')
READ (0,*) VARIANCE(NUM)
VARIANCE(NUM)=VARIANCE(NUM)**2.0
WRITE (3,3) NUM
3 FORMAT (2X,'ENTER NUMBER OF CELLS FOR VARIABLE',I3)
READ (0,*) NCELLS(NUM)
AREA=1/(FLOAT(NCELLS(NUM))*2.0)
DO 10 K=1,NCELLS(NUM)*2.0
   KAREA=AREA*(FLOAT(K))
   P=KAREA
   CALL NORMINV(P,X,D,IE)
   A(K)=X*SQRT(VARIANCE(NUM))+MEAN(NUM)
10 CONTINUE
A(0)=-A(NCELLS(NUM)*2)
CALL CELLR
RETURN
END
SUBROUTINE NORMINV (P,X,D,IE)
C ***************************************************************
C * PROGRAM NAME: NORMINV
C * PURPOSE: SUBROUTINE NORMINV WAS SUPPLIED FROM THE SCIENTIFIC
C * SUBROUTINE PACKAGE (SSP). THE SSP CONTAINS A COLLECTION
C * OF MATHEMATICAL AND STATISTICAL ROUTINES FOR THE
C * HARRIS COMPUTER SYSTEM.
C ***************************************************************
C
SUBROUTINE NORMINV
C
PURPOSE
COMPUTES X = P**(-1)(Y), THE ARGUMENT X SUCH THAT Y = P(X) =
THE PROBABILITY THAT THE RANDOM VARIABLE U, DISTRIBUTED
NORMALLY(0,1), IS LESS THAN OR EQUAL TO X. F(X), THE
ORDINATE OF THE NORMAL DENSITY, AT X, IS ALSO COMPUTED.
C
USAGE
CALL NORMINV (P,X,D,IER)
C
DESCRIPTION OF PARAMETERS
P - INPUT PROBABILITY.
X - OUTPUT ARGUMENT SUCH THAT P = Y = THE PROBABILITY THAT
    U, THE RANDOM VARIABLE, IS LESS THAN OR EQUAL TO X.
D - OUTPUT DENSITY, F(X).
IER - OUTPUT ERROR CODE
    = -1 IF P IS NOT IN THE INTERVAL (0,1), INCLUSIVE.
    X=D=.999999E+38 IN THIS CASE
    = 0 IF THERE IS NO ERROR. SEE REMARKS, BELOW.
C
REMARKS
MAXIMUM ERROR IS 0.00045.
IF P = 0, X IS SET TO -(10)**38. D IS SET TO 0.
IF P = 1, X IS SET TO (10)**38. D IS SET TO 0.
C
SUBROUTINES AND SUBPROGRAMS REQUIRED
NONE
C
METHOD
BASED ON APPROXIMATIONS IN C. HASTINGS, APPROXIMATIONS FOR
DIGITAL COMPUTERS, PRINCETON UNIV. PRESS, PRINCETON, N.J.,
1955. SEE EQUATION 26.2.23, HANDBOOK OF MATHEMATICAL
FUNCTIONS, ABRAMOWITZ AND STEGUN, DOVER PUBLICATIONS, INC.,
C
131
REAL X,D,P,T2,T
INTEGER IE
IE=0
X=1.7E+38
D=X
IF(P)1,4,2
1 IE=-1
   GO TO 12
2 IF (P-1.0)7,5,1
4 X=-1.7E+38
5 D=0.0
   GO TO 12
7 D=P
   IF(D-0.5)9,9,8
8 D=1.0-D
9 T2=ALOG(1.0/(D*D))
   T=SQRT(T2)
   X=T-(2.515517+0.802853*T+0.010328*T2)/(1.0+1.432788*T+0.189269*T2
   +0.001308*T*T2)
   IF(P-0.5)10,10,11
10 X=-X
11 D=0.3989423*EXP(-X*X/2.0)
12 RETURN
END
SUBROUTINE OUTPUT
C * ******************************************************************************
C * PROGRAM NAME: SUBROUTINE OUTPUT
C * WRITTEN BY: BRUCE SWANSON
C * DATE: MARCH 3, 1987
C * PURPOSE: SUBROUTINE OUTPUT PRINTS THE CENTER OF GRAVITY LOCATIONS
C * FOR RANDOM VARIABLES A AND B, AND THE JOINT PROBABILITY
C * DISTRIBUTION OF VARIABLE Z TO FILE W2. OUTPUT ALSO PRINTS
C * THE JOINT DISTRIBUTION FUNCTION TO A GIVEN FILE SO THE
C * JOINT DISTRIBUTION CAN BE INPUT AGAIN AS A "SPECIAL"
C * DISTRIBUTION FUNCTION.
C *
C * SUBRoutines AND SUBProGRAMS REQUIRED: NONE
C *
C * USAGE: CALL OUTPUT
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
DENSITY FUNCTION.
C *
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
FUNCTION.
C *
DISREP(NUM) - CHARACTER*10 - DISREP CONTAINS THE
TYPE OF PROBABILITY DISTRIBUTION FOR VARIABLE
NUM (NORMAL, UNIFORM, TRIANGULAR, SPECIAL).
C *
MEAN(NUM) - REAL - THE MEAN VALUE FOR RANDOM VARIABLE
NUM.
C *
NAME2 - CHARACTER*30 - NAME2 IS THE OUTPUT DATA FILE
NAME OF THE JOINT DISTRIBUTION FUNCTION.
C *
NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
LOCATIONS THAT APPROXIMATE THE PROBABILITY
DENSITY FUNCTION OF RANDOM VARIABLE NUM.
C *
NUM - INTEGER - RANGE OF 1 TO 2, VARIABLE 'NUM' IS THE
NUMBER OF THE RANDOM VARIABLE.
C *
VARIANCE(NUM) - REAL - THE VARIANCE VALUE FOR RANDOM
VARIABLE NUM.
C *
Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
PROBABILITY DISTRIBUTION FUNCTION.
C *
Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO Z(COUNT,1).
C *
Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES Z(COUNT,1) WAS REPEATED DURING THE
**SIMULATION.**

- **ZMEAN** - REAL - THE MEAN OF THE JOINT PROBABILITY DISTRIBUTION.
- **ZVARIANCE** - REAL - THE VARIANCE OF THE JOINT PROBABILITY DISTRIBUTION.

**OUTPUT ARGUMENTS:**

- **ZSTANDARD_DEV** - REAL - THE STANDARD DEVIATION OF THE JOINT PROBABILITY DISTRIBUTION Z.

```
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
REAL Z,MAXDEV,B,MEAN,VARIANCE,A,ZMEAN,ZVARIANCE,UB,LB
INTEGER NCELLS,COUNT,DISTR,NUM,UBCOUNT,LBCOUNT
CHARACTER DISREP*10,NAME*20,NAME2*20,ANSB*1
```

**PRINT DATA TO FILE "W2"**

```
WRITE (23,*)
WRITE (23,*)
WRITE (23,70)
70  FORMAT (2X,`RANDOM VARIABLE: 1`,18X,`RANDOM VARIABLE: 2`)
WRITE (23,80) DISREP(1),DISREP(2)
80  FORMAT (2X,`DISTRIBUTION: `,A,11X,`DISTRIBUTION: `,A)
WRITE (23,90) MEAN(1),MEAN(2)
90  FORMAT (2X,`MEAN: `.F12.6,18X,`MEAN: `.F12.6)
WRITE (23,100) VARIANCE(1),VARIANCE(2)
100 FORMAT (2X,`VARIANCE: `.F12.6,14X,`VARIANCE: `.F12.6)
     STANDARD_DEV1=SQRT(VARIANCE(1))
     STANDARD_DEV2=SQRT(VARIANCE(2))
WRITE (23,101) STANDARD_DEV1,STANDARD_DEV2
101 FORMAT (2X,`STANDARD DEVIATION: `.F12.6,4X,
     *`STANDARD DEVIATION: `.F12.6)
WRITE (23,170) NCELLS(1),NCELLS(2)
170 FORMAT (2X,`NUMBER OF CELLS: `,I4,15X,`NUMBER OF CELLS: `,I4)
WRITE (23,*)
WRITE (23,110)
110 FORMAT (6X,`X LOCATION`,26X,`X LOCATION`)}
WRITE (23,120)
120 FORMAT (4X,`FOR VARIABLE 1`,22X,`FOR VARIABLE 2`)}
```
IF (NCELLS(1).GT.NCELLS(2)) THEN
   INCELLS=NCELLS(1)
ELSE
   INCELLS=NCELLS(2)
END IF

DO 160 I=1,INCELLS
   IF (I.LE.NCELLS(1).AND.I.LE.NCELLS(2)) THEN
      WRITE (23,130) B(I,1),B(I,2)
      FORMAT (4X,F12.6,23X,F12.6)
   ELSE
      WRITE (23,140) B(I,1)
      FORMAT (4X,F12.6)
   ELSE
      WRITE (23,150) B(I,2)
      FORMAT (37X,F12.6)
   END IF
160 CONTINUE
WRITE (23,*)
WRITE (23,161)

DO 10 I=1,COUNT
   WRITE (23,20) I,Z(I,1),Z(I,2)
10 CONTINUE
WRITE (3,116)
116 FORMAT (2X,'ENTER DISTRIBUTION DATA FILE NAME')
READ (0,117) NAME2
117 FORMAT (A20)
OPEN (UNIT=25,FILE=NAME2)
WRITE (25,538) NCELLS(1),NCELLS(2)
538 FORMAT (I5,I5)
WRITE (25,189) COUNT
189 FORMAT (I5)
DO 89 I=1,COUNT
   WRITE (25,112) Z(I,1),Z(I,2),Z(I,3)
112 FORMAT (F20.7,F12.7,F12.7)
89 CONTINUE
CLOSE 25
30 RETURN
END
SUBROUTINE OUTPUTBOUND

C **************************************************
C * PROGRAM NAME: SUBROUTINE OUTPUTBOUND*
C * WRITTEN BY: BRUCE SWANSON*
C * DATE: MARCH 3, 1987*
C *
C * PURPOSE: SUBROUTINE OUTPUT PRINTS THE CENTER OF GRAVITY LOCATIONS*
C * FOR RANDOM VARIABLES A AND B, AND THE JOINT PROBABILITY*
C * DISTRIBUTION OF VARIABLE Z TO FILE W2. OUTPUT ALSO PRINTS*
C * THE JOINT DISTRIBUTION FUNCTION TO A GIVEN FILE SO THE*
C * JOINT DISTRIBUTION CAN BE INPUT AGAIN AS A "SPECIAL"*
C * DISTRIBUTION FUNCTION.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE*
C *
C * USAGE: CALL OUTPUT*
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY*
DENSITY FUNCTION.
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION*
FUNCTION.
DISREP(NUM) - CHARACTER*10 - DISREP CONTAINS THE
TYPE OF PROBABILITY DISTRIBUTION FOR VARIABLE
NUM (NORMAL, UNIFORM, TRIANGULAR, SPECIAL).
MEAN(NUM) - REAL - THE MEAN VALUE FOR RANDOM VARIABLE
NUM.
NAME2 - CHARACTER*30 - NAME2 IS THE OUTPUT DATA FILE
NAME OF THE JOINT DISTRIBUTION FUNCTION.
NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
LOCATIONS THAT APPROXIMATE THE PROBABILITY
DENSITY FUNCTION OF RANDOM VARIABLE NUM.
NUM - INTEGER - RANGE OF 1 TO 2, VARIABLE 'NUM' IS THE
NUMBER OF THE RANDOM VARIABLE.
VARIANCE(NUM) - REAL - THE VARIANCE VALUE FOR RANDOM
VARIABLE NUM.
Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
PROBABILITY DISTRIBUTION FUNCTION.
Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO Z(COUNT,1).
Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES Z(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE Z VALUE OF THE LOWER Bound
OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

ZLB(COUNT,1) - REAL - THE LOWER Bound

ZLB(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
THAN OR EQUAL TO ZLB(COUNT,1).

ZLB(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
TIMES ZLB(COUNT,1) WAS REPEATED DURING THE
SIMULATION.

OUTPUT ARGUMENTS:

ZSTANDARD_DEV - REAL - THE STANDARD DEVIATION OF THE
JOINT PROBABILITY DISTRIBUTION Z.

*********************************************************************************

COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/BOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
REAL Z,MAXDEV,B,MEAN,VARIANCE,A,ZMEAN,ZVARIANCE,UB,LB
INTEGER NCELLS,COUNT,DISTR,NUM,UBCOUNT,LBCOUNT
CHARACTER DISREP*10,NAME*20,NAME2*20,ANSB*1

PRINT DATA TO FILE "W2"

WRITE (23,70)
70 FORMAT (14X,'RANDOM VARIABLE: 1',34X,'RANDOM VARIABLE: 2')
WRITE (23,80) DISREP(1),DISREP(2)
80 FORMAT (14X,'DISTRIBUTION: ',A11,27X,'DISTRIBUTION: ',A)
WRITE (23,90) MEAN(1),MEAN(2)
90 FORMAT (14X,'MEAN: ',F12.6,34X,'MEAN: ',F12.6)
WRITE (23,100) VARIANCE(1),VARIANCE(2)
100 FORMAT (14X,'VARIANCE: ',F12.6,30X,'VARIANCE: ',F12.6)
STANDARD_DEV1 = SQRT(VARIANCE(1))
STANDARD_DEV2 = SQRT(VARIANCE(2))
WRITE (23, 101) STANDARD_DEV1, STANDARD_DEV2
101 FORMAT (14X, "STANDARD DEVIATION: ", F12.6)
     * "STANDARD DEVIATION: ", F12.6)
WRITE (23, 170) NCELS(1), NCELS(2)
WRITE (23, *)
WRITE (23, 110)
110 FORMAT (6X, "X1 MINIMUM", 7X, "CENTER OF", 6X, "X1 MAXIMUM",
       * 10X, "X2 MINIMUM", 7X, "CENTER OF", 6X, "X2 MAXIMUM")
WRITE (23, 120)
       * 10X, "GRAVITY", 10X, "VALUE")
WRITE (23, 123)
123 FORMAT (6X, "----------------", 4X, "----------------", 4X, "----------------"
       * , 8X, "----------------", 4X, "----------------", 4X, "----------------")
IF (NCELS(1).GT.NCELS(2)) THEN
  INCELS = NCELS(1)
ELSE
  INCELS = NCELS(2)
END IF
DO 160 I = 1, INCELS
   IF (I.LE.NCELS(1).AND.I.LE.NCELS(2)) THEN
      WRITE (23, 130) LB(I, 1), B(I, 1), UB(I, 1), LB(I, 2), B(I, 2),
      * UB(I, 2)
130 FORMAT (6X, F12.7, 4X, F12.7, 4X, F12.7, 8X, F12.7, 4X, F12.7,
             4X, F12.7)
   ELSE
      IF (I.LE.NCELS(1)) THEN
         WRITE (23, 140) LB(I, 1), B(I, 1), UB(I, 1)
      140 FORMAT (6X, F12.7, 4X, F12.7, 4X, F12.7)
      ELSE
         WRITE (23, 150) LB(I, 2), B(I, 2), UB(I, 2)
      150 FORMAT (39X, F12.7, 4X, F12.7, 4X, F12.7)
   END IF
160 CONTINUE
WRITE (23, *)
WRITE (23, *)
WRITE (23, 161)
161 FORMAT (38X, "************ Z FUNCTION ************")
WRITE (23, 11) COUNT
11 FORMAT (38X, "NUMBER OF UNIQUE Z LOCATIONS: ", I5)
WRITE (23, 34) ZMEAN
WRITE (23, 162) ZVARIANCE
162 FORMAT (38X, "VARIANCE: ", F12.5)
ZSTANDARD_DEV = SQRT(ZVARIANCE)
139
WRITE (23, 163) ZSTANDARD_DEV
WRITE (23, 13) FORMAT (38X, 'STANDARD DEVIATION: ', F12.5)
WRITE (23,*)
WRITE (23, 12) FORMAT (3X, 'MEAN: ', F12.5)
WRITE (23,*)
ICOUNT = -9999
IF (UBCOUNT GT ICOUNT) THEN
  ICOUNT = UBCOUNT
ELSE IF (LBCOUNT GT ICOUNT) THEN
  ICOUNT = LBCOUNT
ELSE ICOUNT = COUNT
END IF
END IF
DO 10 I = 1, ICOUNT
IF (I LE UBCOUNT .AND. I LE LBCOUNT .AND. I LE COUNT) THEN
  WRITE (23, 20)
ELSE IF (I LE LBCOUNT .AND. I LE COUNT .AND. I GT UBCOUNT) THEN
  WRITE (23, 720)
ELSE IF (I LE LBCOUNT .AND. I GT COUNT .AND. I LE UBCOUNT) THEN
  WRITE (23, 730)
ELSE IF (I GT LBCOUNT .AND. I LE COUNT .AND. I LE UBCOUNT) THEN
  WRITE (23, 740)
ELSE IF (I GT LBCOUNT .AND. I LE COUNT .AND. I LE UBCOUNT) THEN
  WRITE (23, 770)
END IF
10 CONTINUE
END
ELSE
    IF (I.LE.LBCOUNT.AND.I.GT.COUNT.
        AND.I.GT.UBCOUNT) THEN
        WRITE (23,750) I,ZLB(I,1),ZLB(I,2)
        FORMAT (1X,I4,35X,F12.7,4X,F12.7)
    ELSE
        IF (I.LE.COUNT.AND.I.GT.LBCOUNT.
            AND.I.GT.UBCOUNT) THEN
            WRITE (23,760) I,Z(I,1),Z(I,2)
            FORMAT (1X,I4,3X,F12.7,4X,F12.7)
        ELSE
            WRITE (23,770) I,ZUB(I,1),ZUB(I,2)
            FORMAT (1X,I4,67X,F12.7,4X,F12.7)
        END IF
    END IF
END IF
END IF
END IF
END IF
END IF
END IF
END IF
CONTINUE

PRINT DISTRIBUTION DATA TO NAME2 FILE

WRITE (3,116)
FORMAT (2X,'ENTER DISTRIBUTION DATA FILE NAME')
READ (0,117) NAME2
FORMAT (A20)
OPEN (UNIT=25,FILE=NAME2)
WRITE (25,538) NCELLS(1),NCELLS(2)
FORMAT (15,I5)
WRITE (25,189) UBCOUNT,LBCOUNT
FORMAT (15,I5)
IF (UBCOUNT.GT.LBCOUNT) THEN
    ICOUNT=UBCOUNT
ELSE
    ICOUNT=LBCOUNT
END IF
DO 89 I=1,ICOUNT
    IF (I.LE.LBCOUNT.AND.I.LE.UBCOUNT) THEN
        WRITE (25,112) ZUB(I,1),ZUB(I,2),ZLB(I,1),ZLB(I,2)
        FORMAT (F12.6,1X,F12.6,1X,F12.6,1X,F12.6)
    ELSE
        IF (I.LE.LBCOUNT) THEN
            WRITE (25,113) ZLB(I,1),ZLB(I,2)
            FORMAT (F12.6,1X,F12.6,1X,F12.6,1X,F12.6)
        ELSE
            WRITE (25,114) ZUB(I,1),ZUB(I,2)
            FORMAT (F12.6,1X,F12.6,1X,F12.6,1X,F12.6)
        END IF
89  CONTINUE
113           FORMAT (12X,1X,12X,1X,F12.6,1X,F12.6)
           ELSE
           WRITE (25,114) ZUB(I,1),ZUB(I,2)
114           FORMAT (F12.6,1X,F12.6)
           END IF
           END IF
89           CONTINUE
           CLOSE 25
30           RETURN
           END
SUBROUTINE SORT(ZZ,Z,COUNT,AX)

C ***********************************************************************
C *
C * PROGRAM NAME: SUBROUTINE SORT
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 23, 1987
C *
C * PURPOSE: SUBROUTINE SORT DOES A BINARY SORT OF THE ARRAY Z.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE
C *
C * USAGE: CALL SORT(ZZ,Z,COUNT,AX)
C *
C * DISCRPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
Ax - INTEGER - THE NUMBER OF Z VALUES PLUS ONE.
C *
ZZ - REAL - THE Z VALUE THAT NEEDS TO BE INSERTED INTO
C *
ARRAY Z.
C *
C * OUTPUT ARGUMENTS:
C *
COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE
C *
VALUES CONTAINED IN ARRAY Z.
C *
Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
C *
PROBABILITY DISTRIBUTION FUNCTION.
C *
Z(COUNT,2) - REAL - THE NUMBER OF TIMES THE VALUE OF
C *
Z(COUNT,1) HAS BEEN REPEATED DURING THE
C *
SIMULATION.
C *
C ***********************************************************************
C
DIMENSION Z(0:40000,2)
REAL ZZ,Z
INTEGER COUNT,AX
Z(0,1)=-1.0E30
IF (ZZ.GT.Z(AX-1,1).OR.AX.LE.1) THEN
  Z(AX,1)=ZZ
  Z(AX,2)=1
  COUNT=COUNT+1
RETURN
END IF

C
C
START BINARY SORT
C
C
C
IXMIN=1
IXMAX=AX-1
JUMP=NINT((IXMAX-IXMIN)/2.0)+IXMIN
DIFFER=ABS(ZZ-Z(JUMP,1))
IF (ZZ.EQ.Z(JUMP,1).OR.DIFFER.LE.1.0E-7) THEN
  Z(JUMP,2)=Z(JUMP,2)+1
  AX=AX-1
  RETURN
END IF
IF (ZZ.LT.Z(JUMP,1).AND.ZZ.GT.Z(JUMP-1,1)) THEN
  DO 30 K=AX,JUMP+1,-1
       Z(K,1)=Z(K-1,1)
       Z(K,2)=Z(K-1,2)
  CONTINUE
  Z(JUMP,1)=ZZ
  Z(JUMP,2)=1
  COUNT=COUNT+1
ELSE
  IF (ZZ.LT.Z(JUMP,1)) THEN
    IXMAX=JUMP
  ELSE
    IXMIN=JUMP
  END IF
  GOTO 40
END IF
RETURN
END
SUBROUTINE SPECIAL
C ******************************************************************************
C * PROGRAM NAME: SUBROUTINE SPECIAL * *
C * WRITTEN BY: BRUCE SWANSON * *
C * DATE: MARCH 4, 1987 * *
C * PURPOSE: SUBROUTINE SPECIAL READS A DISTRIBUTION FUNCTION STORED IN * *
C * A DATA FILE. SPECIAL THEN FINDS THE NCELL LOCATION ON THE * *
C * X AXIS GIVEN A PROBABILITY (AREA) FOR THE DISTRIBUTION * *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINE CELLR * *
C * USAGE: CALL SPECIAL * *
C * DISCRPTION OF PARAMETERS: * *
C * INPUT ARGUMENTS: * *
C * NAME - CHARACTER*30 - NAME IS THE INPUT DATA FILE NAME * *
C * OF THE SPECIAL DISTRIBUTION FUNCTION FOR * *
C * RANDOM VARIABLE NUM. * *
C * NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE * *
C * LOCATIONS THAT APPROXIMATE THE PROBABILITY * *
C * DENSITY FUNCTION OF RANDOM VARIABLE NUM. * *
C * NUM - INTEGER - VARIABLE NUM IS THE NUMBER OF THE * *
C * RANDOM VARIABLE. * *
C * OUTPUT ARGUMENTS: * *
C * A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N * *
C * REPRESENT THE CENTER OF GRAVITY LOCATIONS AND * *
C * THE EVEN VALUES OF N INDICATE THE CELL * *
C * B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF * *
C * GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY * *
C * DENSITY FUNCTION. * *
C * MEAN(NUM) - REAL - MEAN(NUM) IS THE MEAN VALUE OF * *
C * RANDOM VARIABLE NUM. * *
C * VARIANCE(NUM) - REAL - VARIANCE(NUM) IS THE VARIANCE OF * *
C * RANDOM VARIABLE NUM. * *
C ********************************************************************************

REAL A,TOTAL,DS,AREA,KAREA,MEAN,VARIANCE,B,UB,LB,P
INTEGER NUM,INC,NCELLS,COUNT,DISTR,NUMBER,UCOUNT,LCOUNT
INTEGER NC1,NC2
CHARACTER NAME*30,DISREP*10,ANSB*1
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT

145
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,2MEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CSPPECIAL/DS(40000,3)
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
WRITE (3,*)
WRITE (3,10) NUM
10 FORMAT (2X,"SPECIAL DISTRIBUTION FOR VARIABLE",13)
WRITE (3,20)
20 FORMAT (2X,"ENTER DATA DISTRIBUTION FILE NAME")
READ (0,30) NAME
30 FORMAT (A30)
WRITE (3,110) NUM
110 FORMAT (2X,"ENTER NUMBER OF CELLS FOR VARIABLE",13)
READ (0,*) NCELLS(NUM)
OPEN (UNIT=26,FILE=NAME)
READ (26,120) NC1,NC2
120 READ (26,40) NUMBER
40 FORMAT (I5)
TOTAL=0.0
DO 50 I=1,NUMBER
   READ (26,60) DS(I,1),DS(I,2),DS(I,3)
50 CONTINUE
MEAN(NUM)=TOTAL/FLOAT(NUMBER)
VARIANCE(NUM)=0.0
DO 90 I=1,NUMBER
   P=DS(I,3)/FLOAT(NC1*NC2)
   VARIANCE(NUM)=((DS(I,1)-MEAN(NUM))**2)*P+VARIANCE(NUM)
90 CONTINUE
CLOSE 26
INC=0
AREA=1/(FLOAT(NCELLS(NUM))*2.0)
DO 70 K=1,NCELLS(NUM)*2
   KAREA=AREA*(FLOAT(K))
100 INC=INC+1
   IF (DS(INC,2).GE.KAREA) THEN
      A(K)=((KAREA-DS(INC-1,2))*(DS(INC,1)-DS(INC-1,1)))/
            (DS(INC,2)-DS(INC-1,2)+DS(INC-1,1))
   ELSE
      GOTO 100
   END IF
70 CONTINUE
CALL CELLR
RETURN
END
SUBROUTINE TRIANGLE

C *****************************************************************************
C * PROGRAM NAME: SUBROUTINE TRIANGLE
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 4, 1987
C *
C * PURPOSE: SUBROUTINE TRIANGLE FINDS THE NCELL LOCATION ON THE X AXIS
C * GIVEN A PROBABILITY (AREA) FOR A TRIANGULAR RANDOM
C * VARIABLE.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINE CELLR
C *
C * USAGE: CALL TRIANGLE
C *
C * DISCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
C * AA - REAL - VARIABLE AA IS THE MINIMUM VALUE OF THE
C * TRIANGULAR DENSITY FUNCTION.
C *
C * BB - REAL - VARIABLE BB IS THE MAXIMUM VALUE OF THE
C * TRIANGULAR DENSITY FUNCTION.
C *
C * CC - REAL - VARIABLE CC IS THE MEDIAN VALUE OF THE
C * TRIANGULAR DENSITY FUNCTION.
C *
C * NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
C * LOCATIONS THAT APPROXIMATE THE PROBABILITY
C * THE PROBABILITY DENSITY FUNCTION OF RANDOM
C * VARIABLE NUM.
C *
C * NUM - INTEGER - VARIABLE NUM IS THE NUMBER OF THE
C * RANDOM VARIABLE.
C *
C * OUTPUT ARGUMENTS:
C *
C * A(N) - REAL - WHERE N=NCCELLS*2, ODD VALUES OF N
C * REPRESENT THE CENTER OF GRAVITY LOCATIONS AND
C * THE EVEN VALUES OF N INDICATE THE CELL
C * BOUNDARIES.
C *
C * B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF
C * GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY
C * DENSITY FUNCTION.
C *
C * MEAN(NUM) - REAL - MEAN(NUM) IS THE MEAN VALUE OF
C * RANDOM VARIABLE NUM.
C *
C * VARIANCE(NUM) - REAL - VARIANCE(NUM) IS THE VARIANCE OF
C * RANDOM VARIABLE NUM.
C *
C *****************************************************************************
C
REAL AREA,KAREA,A,B,MEAN,VARIANCE,AA,BB,CC,TAREA,Z
REAL ZMEAN,ZVARIANCE,UB,LB

147
INTEGER NCELLS, NUM, K, COUNT, DISTR, UB_COUNT, LB_COUNT
CHARACTER DISREP*10, ANSB
COMMON/FIRST/A(0:400), B(0:400,2), Z(0:40000,3), COUNT
COMMON/SECOND/DISTR(2), NCELLS(2), VARIANCE(2), MEAN(2), DISREP(2)
COMMON/THIRD/NUM, Z_MEAN, Z_VARIANCE, UB(0:400,2), LB(0:400,2), ANSB
COMMON/CBOUND/ZLB(0:40000,3), ZUB(0:40000,3), UB_COUNT, LB_COUNT

5 WRITE (3,*)
WRITE (3,*) NUM
WRITE (3,10) NUM
10 FORMAT (2X, 'TRIANGULAR DISTRIBUTION FOR VARIABLE', I3)
WRITE (3,20)
20 FORMAT (2X, 'PARAMETERS: A<C<B')
WRITE (3,30)
30 FORMAT (2X, 'RANGE: (A,B)')
WRITE (3,40)
40 FORMAT (2X, 'ENTER "A" (MINIMUM) VALUE')
READ (0,*) AA
WRITE (3,50)
50 FORMAT (2X, 'ENTER "C" (MIDDLE) VALUE')
READ (0,*) CC
IF (CC-AA.LT.0) THEN
GOTO 5
END IF
WRITE (3,60)
60 FORMAT (2X, 'ENTER "B" (MAXIMUM) VALUE')
READ (0,*) BB
IF (BB-CC.LT.0) THEN
GOTO 5
END IF
WRITE (3,70) NUM
70 FORMAT (2X, 'ENTER NUMBER OF CELLS FOR VARIABLE', I3)
READ (0,*) NCELLS(NUM)
AREA=1/(FLOAT(NCELLS(NUM))*2.0)
TAREA=(CC-AA)*(2/(BB-AA))/2
DO 80 K=1,NCELLS(NUM)*2.0
KAREA=AREA*(FLOAT(K))
IF (KAREA.GT.TAREA) THEN
A(K)=BB-SQRT((1-KAREA)*(BB-AA)*(BB-CC))
ELSE
A(K)=SQRT(KAREA*(BB-AA)*(CC-AA))+AA
END IF
CONTINUE
80 MEAN(NUM)=(AA+BB+CC)/3
VARIANCE(NUM)=(AA**2+BB**2+CC**2-AA*BB-AA*CC-BB*CC)/18
CALL CELLR
RETURN
END
SUBROUTINE UNIFORM

C ***********************************************************************
C * PROGRAM NAME: SUBROUTINE UNIFORM                          *
C * WRITTEN BY: BRUCE SWANSON                                     *
C * DATE: MARCH 4, 1987                                            *
C * PURPOSE: SUBROUTINE UNIFORM FINDS THE NCELL LOCATION ON THE X AXIS *
C * GIVEN A PROBABILITY (AREA) FOR A UNIFORM RANDOM VARIABLE.      *
C * SUBRoutines and Subprograms REQUIRED: SUBROUTINE CELLR          *
C * USAGE: CALL UNIFORM                                           *
C * DISCRIPTION OF PARAMETERS:                                    *
C * INPUT ARGUMENTS:                                              *
C * MAXX - REAL - MAXX IS THE MAXIMUM POSSIBLE VALUE FOR           *
C * THE UNIFORM RANDOM VARIABLE.                                 *
C * MINX - REAL - MINX IS THE MINIMUM POSSIBLE VALUE FOR           *
C * THE UNIFORM RANDOM VARIABLE.                                 *
C * NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE                 *
C * LOCATIONS THAT APPROXIMATE THE PROBABILITY                   *
C * DENSITY FUNCTION OF RANDOM VARIABLE NUM.                      *
C * NUM - INTEGER - VARIABLE 'NUM' IS THE NUMBER OF THE            *
C * RANDOM VARIABLE.                                              *
C * OUTPUT ARGUMENTS:                                             *
C * A(N) - REAL - WHERE N=NCELLS*2, ODD VALUES OF N                *
C * REPRESENT THE CENTER OF GRAVITY LOCATIONS AND                 *
C * THE EVEN VALUES OF N INDICATE THE CELL BOUNDARIES.            *
C * B(NCELLS,NUM) - REAL - THE DISCRETE POINT (CENTER OF          *
C * GRAVITY) REPRESENTATIONS FOR NUM'S PROBABILITY                *
C * DENSITY FUNCTION.                                            *
C * MEAN(NUM) - REAL - MEAN(NUM) IS THE MEAN VALUE OF              *
C * RANDOM VARIABLE NUM.                                          *
C * VARIANCE(NUM) - REAL - VARIANCE(NUM) IS THE VARIANCE OF        *
C * RANDOM VARIABLE NUM.                                          *
C * ***********************************************************************

REAL AREA,KAREA,MINX,MAXX,A,B,MEAN,VARIANCE,Z,ZMEAN,ZVARIANCE
REAL UB,LB
INTEGER NCELLS,NUM,K,COUNT,DISTR,UBCOUNT,LBCOUNT
CHARACTER DISREP*10,ANSEP*1
COMMON/FIRST/A(0:400),B(0:400,2),Z(0:40000,3),COUNT
COMMON/SECOND/DISTR(2),NCELLS(2),VARIANCE(2),MEAN(2),DISREP(2)
COMMON/THIRD/NUM,ZMEAN,ZVARIANCE,UB(0:400,2),LB(0:400,2),ANSB
COMMON/CBOUND/ZLB(0:40000,3),ZUB(0:40000,3),UBCOUNT,LBCOUNT
WRITE (3,*)
WRITE (3,*)
WRITE (3,10) NUM
10 FORMAT (2X,"UNIFORM DISTRIBUTION FOR VARIABLE",I3)
WRITE (3,30)
30 FORMAT (2X,"RANGE (MINX,MAXX)")
WRITE (3,40)
40 FORMAT (2X,"ENTER MINX VALUE")
READ (0,*), MINX
WRITE (3,50)
50 FORMAT (2X,"ENTER MAXX VALUE")
READ (0,*), MAXX
IF (MAXX-MINX.LT.0) THEN
   GOTO 5
END IF
WRITE (3,60) NUM
60 FORMAT (2X,"ENTER NUMBER OF CELLS FOR VARIABLE",I3)
READ (0,*), NCELLS(NUM)
AREA=1/(FLOAT(NCELLS(NUM))*2.0)
DO 20 K=1,NCELLS(NUM)*2.0
   KAREA=AREA*(FLOAT(K))
   A(K)=MINX*KAREA*(MAXX-MINX)
20 CONTINUE
MEAN(NUM)=(MINX+MAXX)/2.0
VARIANCE(NUM)=((MAXX-MINX)**2)/12.0
CALL CELLR
RETURN
END
APPENDIX B

DENSITY FUNCTION PROGRAM
PROGRAM NAME: DENSITY

WRITTEN BY: BRUCE SWANSON

DATE: MARCH 26, 1987

PURPOSE: PROGRAM "DENSITY" APPROXIMATES THE JOINT PROBABILITY DENSITY FUNCTION USING A CUBIC SPLINE AND A HISTOGRAM. THE DISTRIBUTION DATA IS READ IN FROM A FILE DETERMINED BY THE USER. IF THE PROGRAM FAILS DURING THE CUBIC SPLINE APPROXIMATION, THE "MANAGE" PROGRAM MUST BE USED TO ADJUST THE DATA.

SUBROUTINES AND SUBPROGRAMS REQUIRED: SUBROUTINES HISTOGRAM OUTPUTD SMOOTH

DESCRIPTION OF PARAMETERS:

INPUT ARGUMENTS:

COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

NAME - CHARACTER*30 - NAME IS THE INPUT/OUTPUT DATA FILE NAME OF THE DISTRIBUTION/DENSITY FUNCTION.

NCCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE LOCATIONS THAT APPROXIMATE THE PROBABILITY DENSITY FUNCTION OF RANDOM VARIABLE NUM.

Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION.

Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS THAN OR EQUAL TO Z(COUNT,1).

Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF TIMES Z(COUNT,1) WAS REPEATED DURING THE SIMULATION.

OUTPUT ARGUMENTS:

H(N,1) - REAL - THE Z LOCATION OF THE NTH CELL OF THE HISTOGRAM.

H(N,2) - REAL - THE FRACTION OF Z LOCATIONS WITHIN THE RANGE OF H(N,1).

NUMBER - REAL - THE NUMBER OF TOTAL DIVISIONS WITHIN THE HISTOGRAM.

RANGE - REAL - THE RANGE OF VALUES OF THE HISTOGRAM.

R(N) - REAL - THE "SMOOTHED" VALUES OF THE Z PROBABILITY DISTRIBUTION FOR VARIABLE N AFTER THE CUBIC SPLINE APPROXIMATION.
R1(N) - REAL - THE DENSITY FUNCTION AFTER THE CUBIC *
SPINE APPROXIMATION.
SIZE - REAL - THE SIZE OF EACH CELL WITHIN THE *
HISTOGRAM.

INTEGER COUNT, NCELLS
REAL Z
CHARACTER NAME*30
COMMON/MAIN/Z(20000,4), COUNT, NCELLS(2), H(200,2), R(20004), *
Rl(20004), NUMBER

WRITE (3, 10)
10 FORMAT (2X, 'ENTER DISTRIBUTION DATA FILE NAME')
READ (0, 20) NAME
20 FORMAT (A30)
OPEN (UNIT=23, FILE=NAME)
READ (23, 60) NCELLS(1), NCELLS(2)
60 FORMAT (I5, I5)
READ (23, 30) COUNT
30 FORMAT (I5)
DO 40 I = 1, COUNT
   READ (23, 50) Z(I, 1), Z(I, 2), Z(I, 3)
50 FORMAT (F20.7, F12.7, F12.7)
40 CONTINUE
CLOSE 23
CALL HISTOGRAM
CALL SMOOTH
CALL OUTPUTD
STOP
END
SUBROUTINE HISTOGRAM

C ******************************************************************************
C *
C * PROGRAM NAME: SUBROUTINE HISTOGRAM
C *
C * WRITTEN BY: BRUCE SWANSON
C *
C * DATE: MARCH 26, 1987
C *
C * PURPOSE: SUBROUTINE "HISTOGRAM" BUILDS A HISTOGRAM FROM A
C * DISTRIBUTION FUNCTION STORED IN A DATA FILE.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE
C *
C * USAGE: CALL HISTOGRAM
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
C  COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z
C  LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION
C *
C  NAME - CHARACTER*30 - NAME IS THE OUTPUT DATA FILE NAME
C  OF THE HISTOGRAM.
C *
C  NCELLS(NUM) - INTEGER - THE NUMBER OF DISCRETE
C  LOCATIONS THAT APPROXIMATE THE PROBABILITY
C  DENSITY FUNCTION OF RANDOM VARIABLE NUM.
C *
C  Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT
C  PROBABILITY DISTRIBUTION FUNCTION.
C *
C  Z(COUNT,2) - REAL - THE PROBABILITY OF Z BEING LESS
C  THAN OR EQUAL TO Z(COUNT,1).
C *
C  Z(COUNT,3) - REAL - THE NUMBER OF TIMES THE VALUE OF
C  TIMES Z(COUNT,1) WAS REPEATED DURING THE
C  SIMULATION.
C *
C *
C * OUTPUT ARGUMENTS:
C *
C  H(N,1) - REAL - THE Z LOCATION OF THE NTH CELL OF THE
C  HISTOGRAM.
C *
C  H(N,2) - REAL - THE FRACTION OF Z LOCATIONS WITHIN THE
C  RANGE OF H(N,1).
C *
C  NUMBER - REAL - THE NUMBER OF TOTAL DIVISIONS WITHIN
C  THE HISTOGRAM.
C *
C  RANGE - REAL - THE RANGE OF VALUES OF THE HISTOGRAM.
C *
C  SIZE - REAL - THE SIZE OF EACH CELL WITHIN THE
C  HISTOGRAM.
C *
C *
C ********************************************************************************
C
REAL RANGE, SIZE, H, Z

154
INTEGER NUMBER,COUNT,NCELLS
CHARACTER NAME*30

* COMMON/MAIN/Z(20000,4),COUNT,NCELLS(2),H(200,2),R(20004),
* R1(20004),NUMBER

NUMBER=NINT(1+3.3*LOG10(FLOAT(NCELLS(1)*NCELLS(2))))
RANGE=Z(COUNT,1)-Z(1,1)
SIZE=RANGE/FLOAT(NUMBER)

DO 10 I=1,NUMBER
   H(I,1)=SIZE*FLOAT(I)+Z(1,1)
   H(I,2)=0.0
10 CONTINUE
J=1

DO 20 I=1,COUNT
   DIFFER=ABS(Z(I,1)-H(J,1))
   IF (Z(I,1).LE.H(J,1).OR.DIFFER.LE.1.0E-6) THEN
      H(J,2)=H(J,2)+Z(I,1)
   ELSE
      J=J+1
      GOTO 30
   END IF
20 CONTINUE

DO 40 I=1,NUMBER
   H(I,2)=H(I,2)/(NCELLS(1)*NCELLS(2))
40 CONTINUE

WRITE (3,50)
50 FORMAT (2X,'ENTER HISTOGRAM DATA FILE NAME')
READ (0,60) NAME
60 FORMAT (A30)
OPEN (UNIT=23,FILE=NAME)
WRITE (23,70) NUMBER
70 FORMAT (15)
DO 90 I=1,NUMBER
   WRITE (23,80) H(I,1),H(I,2)
80 FORMAT (F12.7,F12.7)
90 CONTINUE
CLOSE 23
RETURN
END
**SUBROUTINE OUTPUTD**

C *********************************************************
C * PROGRAM NAME: SUBROUTINE OUTPUTD                       *
C * WRITTEN BY: BRUCE SWANSON                              *
C * DATE: MARCH 26, 1987                                   *
C * PURPOSE: SUBROUTINE OUTPUTD PRINTS THE JOINT PROBABILITY DENSITY FUNCTION TO A FILE DESIGNATED BY THE USER. *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE              *
C * USAGE: CALL OUTPUTD                                    *
C * DESCRIPTION OF PARAMETERS:                            *
C * INPUT ARGUMENTS:                                      *
C * COUNT - INTEGER - COUNT IS THE NUMBER OF UNIQUE Z LOCATIONS OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION. *
C * NAME - CHARACTER*30 - NAME IS THE OUTPUT DATA FILE NAME OF THE APPROXIMATED JOINT PROBABILITY DENSITY FUNCTION. *
C * Z(COUNT,1) - REAL - THE Z VALUE OF THE JOINT PROBABILITY DISTRIBUTION FUNCTION. *
C * R(N) - REAL - THE "SMOOTHED" VALUES OF THE Z PROBABILITY DISTRIBUTION FOR VARIABLE N AFTER THE CUBIC SPLINE APPROXIMATION. *
C * R1(N) - REAL - THE DENSITY FUNCTION AFTER THE CUBIC SPINE APPROXIMATION. *
C *********************************************************

INTEGER NCELLS,COUNT,NUMBER
REAL Z,RANGE,SIZE,H,R,R1
CHARACTER NAME*30
COMMON/MAIN/Z(20000,4),COUNT,NCELLS(2),H(200,2),R(20004),R1(20004),NUMBER

WRITE (3,60)
FORMAT (2X,"ENTER SPLINE DATA FILE NAME")
READ (0,70) NAME
70 FORMAT (A30)
OPEN (UNIT=23,FILE=NAME)
WRITE (23,80) COUNT
80 FORMAT (I5)
DO 90 I=1,COUNT
  WRITE (23,100) Z(I,1),R(I),R1(I)
90 CONTINUE
CLOSE 23
RETURN
END
SUBROUTINE SMOOTH

***********************************************************************

* PROGRAM NAME: SUBROUTINE SMOOTH

* PURPOSE: SUBROUTINE SMOOTH APPROXIMATES THE DENSITY FUNCTION FROM
* A GIVEN SET OF DISTRIBUTION DATA.

***********************************************************************

SANDIA MATHEMATICAL PROGRAM LIBRARY
APPLIED MATHEMATICS DIVISION 2546
SANDIA LABORATORIES
ALBUQUERQUE, NEW MEXICO 87185
CONTROL DATA 6600/7600 VERSION 8.1 AUGUST 1980

***********************************************************************

* ISSUED BY *
* SANDIA LABORATORIES, *
* A PRIME CONTRACTOR *
******
* UNITED STATES *
* DEPARTMENT *
* OF *
* ENERGY *

***********************************************************************
---NOTICE--- ********************

* THIS REPORT WAS PREPARED AS AN ACCOUNT OF WORK SPONSORED
* BY THE UNITED STATES GOVERNMENT. NEITHER THE UNITED *
* STATES NOR THE UNITED STATES DEPARTMENT OF ENERGY, *
* NOR ANY OF THEIR EMPLOYEES, *
* NOR ANY OF THEIR CONTRACTORS, SUBCONTRACTORS, OR THEIR *
* EMPLOYEES, MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR *
* ASSUMES ANY LEGAL LIABILITY OR RESPONSIBILITY FOR THE *
* ACCURACY, ******* *
* COMPLETENESS *
* OR USEFULNESS *
* OF ANY *
* INFORMATION, *
* APPARATUS, *
* PRODUCT *
* OR PROCESS *
* DISCLOSED, *
* OR REPRESENTS *
* THAT ITS *
* USE WOULD NOT *
* INFRINGE *
* PRIVATELY *
* OWNED *
* RIGHTS. *

***********************************************************************
SMOO FITS A SMOOTH SPLINE THROUGH A GIVEN SET OF DATA POINTS
BY MINIMIZING THE INTEGRAL OF THE SECOND DERIVATIVE SQUARED,
SUBJECT TO THE CONSTRAINT THAT

\[ \sum_{i=1}^{N} \left( \frac{r(i)-y(i)}{dy(i)} \right)^2 \leq s \]

WHERE \( r(i) \) IS THE ORDINATE OF THE SMOOTH SPLINE AT \( x(i) \).
SMOO RETURNS THE VALUES OF THE SPLINE FUNCTION, \( r \),
ITS FIRST DERIVATIVE, \( rl \), AND ITS SECOND DERIVATIVE, \( r2 \),
evaluated at the abscissas of the given data points.
THE RESULTING SPLINE, DEFINED BY THE ARRAYS \( x \), \( r \), AND \( r2 \),
may then be interpolated (if desired) using SPLINT.
FOR AN EXACT SPLINE FIT SEE SUBROUTINE SPLIFT.

DESCRIPTION OF ARGUMENTS

INPUT ARGUMENTS --
- \( N \) - NUMBER OF DATA VALUES (AT LEAST 3)
- \( X \) - ABSCISSA ARRAY (INCREASING ORDER)
- \( Y \) - ORDIINATE ARRAY
- \( DY \) - ARRAY OF ERROR ESTIMATES. \( dy(i) \) SHOULD BE AN ESTIMATE
  OF THE ERROR (ACTUALLY, THE STANDARD DEVIATION) IN \( y(i) \).
  Thus, the units of \( dy \) are the same as the units of \( y \).
  Larger values of \( dy(i) \) allow a looser, smoother fit.
  Smaller values of \( dy(i) \) cause a tighter fit.
  Setting \( dy(i)=0 \) at all points results in an exact fit. (see SPLIFT)
  By appropriately adjusting \( dy(i) \) at each point, the spline
  can be made tight at critical points and loose at others.
- \( S \) - SHOULD NORMALY = \( N \). (NOTE-- \( S \) IS FLOATING POINT - DONT
  USE \( N \) DIRECTLY FOR \( S \).) IF YOU WISH TO TIGHTEN OR LOOSEN
  THE SPLINE FIT BY MULTIPLYING EACH ELEMENT OF \( dy \) BY
  SOME FACTOR \( F \), YOU MAY ALTERNATIVELY SIMPLY MULTIPLY
  \( S \) BY \( F**2 \).

OUTPUT ARGUMENTS --
- \( A,B,C,D \) - CUBIC BETWEEN \( x(i) \) AND \( x(i+1) \) IS
  \[ a(i) + b(i)*h + c(i)*h**2 + d(i)*h**3 \]
  WHERE \( h \) IS DESIRED ABCISSA MINUS \( x(i) \).
- \( R \) - ARRAY OF SMOOTH SPLINE VALUES
- \( RL \) - ARRAY OF SMOOTH SPLINE DERIVATIVES
- \( R2 \) - ARRAY OF SMOOTH SPLINE SECOND DERIVATIVES
- \( T,T1,U,V \) - WORK ARRAYS
- \( IERR \) - A STATUS CODE
  --NORMAL CODE
  =1 MEANS THE REQUESTED SPLINE WAS COMPUTED.
  --ABNORMAL CODE
  =2 MEANS EITHER \( N \) IS LESS THAN 3, OR \( S \) IS NEGATIVE,
  OR THE X-AXIS VALUES ARE MISORDERED.

\( X,Y,DY,A,B,C,D \) MUST BE DIMENSIONED AT LEAST \( N \)
\( R,RL,R2,T,T1,U,V \) MUST BE DIMENSION AT LEAST \( N+2 \)
THE ORIGINAL N1 WAS FIXED AT 1 TO AVOID WASTED WORK ARRAY SPACE
THE ORIGINAL N2 IS CALLED N HERE
ALL WORK ARRAY INDICES ARE 1 LARGER THAN IN THE ALGOL,
TO AVOID A ZERO SUBSCRIPT.

COMMON/MAIN/Z(20000,4),COUNT,NCELLS(2),HHH(200,2),R(20004),
   R1(20004),NUMBER
COMMON/SIMO/T(20004),T1(20004),U(20004),V(20004),R2(20004),
   *X(20000),Y(20000),DY(20000),A(20000),B(20000),C(20000),D(20000)
INTEGER COUNT
REAL AAA,BBB,Z,R2,X,Y,DY,A,B,C,D
WRITE (3,66)
66 FORMAT(2X,"ENTER TOLERANCE FOR SPLINE FIT")
   READ (0,*) F
   S=.05**2
   S=FLOAT(COUNT)
   N=COUNT
DO 15 I=1,COUNT
   X(I)=Z(I,1)
   Y(I)=Z(I,2)
   DY(I)=F
15 CONTINUE
C CHECK INPUT DATA
C
IERR = 1
IF (N.LT.3) GO TO 10
IF (S.LT.0.0) GO TO 10
DO 5 I=2,N
   IF (X(I)-X(I-1)) 10,10,5
5 CONTINUE
GO TO 20
10 WRITE (3,76)
76 FORMAT (2X,"PROGRAM BOMBED BECAUSE OF MISSORDER")
   IERR = 2
   RETURN
20 CONTINUE
C INITIALIZE
C
R(1) = 0.0
R(2) = 0.0
R1(N+1) = 0.0
R2(N+1) = 0.0
R2(N+2) = 0.0
U(1) = 0.0
U(2) = 0.0
U(N+1) = 0.0
U(N+2) = 0.0
P = 0.0
M2 = N-1
C
H = X(2)-X(1)
IF (H.EQ.0.0) THEN
WRITE (3,439) X(2),X(1)
FORMAT (2X,'X(2)=',F12.7,'X(1)=',F12.7)
END IF
F = (Y(2)-Y(1))/H
DO 100 I=2,M2
G = H
H = X(I+1)-X(I)
E = F
F = (Y(I+1)-Y(I))/H
A(I) = F-E
T(I+1) = 2.0*(G+H)/3.0
T1(I+1) = H/3.0
R2(I+1) = DY(I-1)/G
R(I+1) = DY(I+1)/H
100 R1(I+1) = -DY(I)/G - DY(I)/H
C
DO 200 I=2,M2
B(I) = R(I+1)*R(I+1) + R1(I+1)*R1(I+1) + R2(I+1)*R2(I+1)
C(I) = R(I+1)*R1(I+2) + R1(I+1)*R2(I+2)
200 D(I) = R(I+1)*R2(I+3)
F2 = -S
C
NEXT ITERATION
C
250 DO 300 I=2,M2
R1(I) = F*R(I)
R2(I-1) = G*R(I-1)
R(I+1) = 1.0/(F*B(I)+T(I+1)-F*R1(I)-G*R2(I-1))
U(I+1) = A(I)-R1(I)*U(I)-R2(I-1)*U(I-1)
F = P*C(I)+T1(I+1)-H*R1(I)
G = H
300 H = D(I)*P
C
DO 400 J=2,M2
I=M2-(J-2)
400 U(I+1) = R(I+1)*U(I+1) - R1(I+1)*U(I+2) - R2(I+1)*U(I+3)
E=0.0
H=0.0
C
DO 500 I=1,M2
G = H
H = (U(I+2)-U(I+1))/(X(I+2)-X(I))
V(I+1) = (H-G)*DY(I)*DY(I)
500 E = E+V(I+1)*(H-G)
V(N+1) = -H*DY(N)*DY(N)
G = V(N+1)
E = E-G*H
G = F2
F2 = E*P*P
IF ((F2.GE.S).OR.(F2.LE.G)) GO TO 650
F = 0.0
H = (V(3)-V(2))/(X(2)-X(1))
C
DO 600 I=2,M2
160
G = H
H = (V(I+2)-V(I+1))/(X(I+1)-X(I))
G=H-G*R(I)*R(I)-R2(I-1)*R(I-1)
F = F+G*R(I+1)*G
600 R(I+1) = G
H = E-P*F
IF (H.LE.0.0) GO TO 650
P = P+(S-F2)/((SQRT(S/E)+P)*H)
GO TO 250
C
C FINISH
C
650 DO 700 I=1,N
A(I)=Y(I)-P*V(I+1)
700 C(I)=U(I+1)
C
DO 800 I=1,M2
H=X(I+1)-X(I)
D(I)=(C(I+1)-C(I))/(3.0*H)
800 B(I)=(A(I+1)-A(I))/H-(H*D(I)+C(I))*H
B(N)=0.0
D(N)=0.0
C
C OUTPUT DERIVATIVES, ETC.
C
DO 900 I=1,M2
R(I) = A(I)
R1(I) = B(I)
900 R2(I) = 2.0*C(I)
R(N) = A(N)
H = X(N)-X(M2)
R1(N)=(3.0*D(M2)*H+2.0*C(M2))*H+B(M2)
R2(N) = 0.0
DO 16 I=1,COUNT
Z(I,3)=R(I)
Z(I,4)=R1(I)
16 CONTINUE
RETURN
END

161
APPENDIX C

DATA MANAGEMENT PROGRAM
PROGRAM NAME: MANAGE

Written by: Bruce Swanson

Date: March 6, 1987

Purpose: Program "MANAGE" is used if a problem occurs with the calculation of the cubic spline density approximation. This program reads in a distribution data file and finds the smallest abscissa and prints out the value to the screen. It then eliminates any data points that are closer than 0.000001 along the x axis.

Subroutines and subprograms required: None

Description of parameters:

Input arguments:
- ANSWER - Character*1 - (Y or N) an input variable used to indicate if the user wants to perform data management to the distribution file.
- COUNT - Integer - number of unique z locations.
- NAME - Character*30 - name is the input data file name of the joint distribution function.
- NAME2 - Character*30 - name2 is the output data file name of the joint distribution function.
- NCELLS(1 or 2) - Integer - the number of discrete locations that approximate the probability density function of random variable 1 or 2.
- SMALL - Real - the smallest x increment value of the joint distribution function.
- Z(COUNT,1) - Real - the abscissa value of the joint probability distribution function.
- Z(COUNT,2) - Real - the probability of z being less than or equal to Z(COUNT,1).

Output arguments:
- INC - Real - the number of unique z locations after the data management has been performed.
- ZD(INC,1) - Real - the abscissa value of the joint probability distribution function after data management has been performed.
- ZD(INC,2) - Real - the probability of z being less than or equal to Z(INC,1).

*******************************************************************************
REAL Z,DIFF1,DIFF2,SMAH, ZD,TOLERANCE
INTEGER COUNT,NCELLS1,NCELLS2
CHARACTER NAME*20, ANSWER*1, NAME2*20
COMMON/Main/Z(2,10000), ZD(2,10000)
TOLERANCE=.00001
SMALL=99999
WRITE (3,10)
10 FORMAT (2X, 'ENTER INPUT FILE NAME')
READ (0,20) NAME
20 FORMAT (A20)
OPEN (UNIT=23, FILE=NAME)
READ (23,30) NCELLS1,NCELLS2
30 FORMAT (15,15)
READ (23,40) COUNT
40 FORMAT (15)
DO 50 I=1,COUNT
   READ (23,60) Z(1,I), Z(2,I), DUMMY
50 CONTINUE
DO 70 I=2, COUNT
   DIFF1=ABS(Z(1,I)-Z(1,I-1))
   IF (DIFF1.LT. SMALL) THEN
      SMALL=DIFF1
   END IF
70 CONTINUE
WRITE (3,80) SMALL
80 FORMAT (2X, 'SMALLEST X INCREMENT=',F12.7)
CLOSE 23
WRITE (3,90)
90 FORMAT (2X, 'PERFORM DATA MANAGEMENT? (Y/N)'
READ (0,100) ANSWER
100 FORMAT (A1)
IF (ANSWER.EQ. 'N') THEN
   GOTO 9999
END IF
WRITE (3,200)
200 FORMAT (2X, 'ENTER OUTPUT FILE NAME')
READ (0,210) NAME2
210 FORMAT (A20)
OPEN (UNIT=24, FILE=NAME2)
INC=1
ZD(1,1)=Z(1,1)
ZD(2,1)=Z(2,1)
DO 110 I=2, COUNT
   DIFF1=ABS(Z(1,I)-Z(1,I-1))
   IF (DIFF1.LT. TOLERANCE) THEN
      GOTO 110
   ELSE
      \n   164
ZD(1,INC) = Z(1,I)
ZD(2,INC) = Z(2,I)
INC = INC + 1
END IF

110 CONTINUE
INC = INC - 1
WRITE (24,220) NCELLS1, NCELLS2
220 FORMAT (I5, I5)
WRITE (24,130) INC
130 FORMAT (I5)
DO 120 I = 1, INC
  Z(I,1) = ZD(I,1)
  Z(I,2) = ZD(I,2)
  WRITE (24,140) ZD(1,I), ZD(2,I)
140 FORMAT (F20.7, F12.7)
120 CONTINUE
CLOSE 24
9999 STOP
END
APPENDIX D

GRAPHICS PROGRAM
REAL MATL,MITL,XX,YY,ZZ,ERROR,MAXDEV
INTEGER COUNT,INCELLS,ICOUNT,ITIME,ANS
CHARACTER XLABEL*30,YLABEL*30,GTITLE*35,NAME*20
LOGICAL AUTOSC
COMMON/ONE/XX(6,10000),YY(6,10000),COUNT
COMMON/MISC/INCELLS(10000),ERROR(10000),MAXDEV(10000),
ICOUNT(10000),ITIME(10000)

WRITE (3,10)
10 FORMAT (2X,"PLOTT:","")
WRITE (3,20)

167
FORMAT (2X,'1. DISTRIBUTION FUNCTION')
WRITE (3,30)
FORMAT (2X,'2. DENSITY FUNCTION')
WRITE (3,40)
FORMAT (2X,'ENTER NUMBER FOR DESIRED CURVE')
READ (0,*) ANS
IF (ANS.EQ.1) THEN
  CALL GVTDISTRIBUTION
ELSE
  IF (ANS.EQ.2) THEN
    CALL GVTDENSITY
  ELSE
    GOTO 1
  END IF
END IF
ICURVES=1
WRITE (3,*)
DO 6000 I=1,COUNT
  WRITE (3,*) XX(1,I),YY(1,I)
6000  CONTINUE
WRITE (3,90)
90 FORMAT (2X,'ENTER TITLE')
READ (0,100) GTITLE
100 FORMAT (A35)
WRITE (3,92)
92 FORMAT (2X,'ENTER X LABEL')
READ (0,93) XLABEL
93 FORMAT (A30)
WRITE (3,94)
94 FORMAT (2X,'ENTER Y LABEL')
READ (0,96) YLABEL
96 FORMAT (A30)
MATL=0.03
MITL=0.01
IDEV=2
AUTOSC=.TRUE.
CALL PLOTT (D1,D2,D3,D4,MATL,MITL,XLABEL,YLABEL,
*GTITLE,XX,YY,COUNT,ICURVES,IDEV,AUTOSC)
END
SUBROUTINE AUTOSCL (XX,YY,K,XMAX,XMIN,YMAX,YMIN,MATSX,MITSX,
@ MATSY,MITSY,AUTOSC,XSTART,YSTART,
@ DX,DY)

C ********************************************
C * PROGRAM NAME: SUBROUTINE AUTOSCL
C * WRITTEN BY: DONALD A. SMITH
C ********************************************

REAL XX(6,10000),YY(6,10000),MATSX,MATSY,MITSX,MITSY,DX(4),
@ DY(4)

LOGICAL AUTOSC
IF (AUTOSC) THEN
XMAX=XX(1,1)
XMIN=XX(1,1)
YMAX=YY(1,1)
YMIN=YY(1,1)
WRITE (3,2222)
2222 FORMAT (2X 'ENTERED AUTOSCL')
DO 10 I=1,K
    IF (XX(I,1).GT.XMAX) XMAX=XX(I,1)
    IF (XX(I,1).LT.XMIN) XMIN=XX(I,1)
    IF (YY(I,1).GT.YMAX) YMAX=YY(I,1)
    IF (YY(I,1).LT.YMIN) YMIN=YY(I,1)
10 CONTINUE
ENDIF
A=ABS((XMAX-XMIN)/5.)
MATSX=A
C=ABS((YMAX-YMIN)/5.)
MATSY=C
IF ((XMAX.GT.0.) .AND. (ABS(XMAX-XMIN).LT.ABS(XMAX))) THEN
    XSTART=XMIN
    GOTO 14
ENDIF
IF ((XMAX.LE.0.) .AND. (ABS(XMAX-XMIN).LT.ABS(XMIN))) THEN
    XSTART=XMAX
    ELSE
    XSTART=0.
ENDIF
14 CONTINUE
IF ((YMAX.GT.0.) .AND. (ABS(YMAX-YMIN).LT.ABS(YMAX))) THEN
    YSTART=YMIN
    GOTO 30
ENDIF
IF ((YMAX.LE.0.) .AND. (ABS(XMAX-XMIN).LT.ABS(XMIN))) THEN
    YSTART=YMAX
ELSE
    YSTART=0.
ENDIF

30 CONTINUE
D WRITE (3,*) 'TICKS CALC DX'
210 CONTINUE
DX(1)=(XMAX-XSTART)/MATSX
DX(2)=(XMIN-XSTART)/MATSX
MITSX=MATSX/5.
DX(3)=(XMAX-XSTART)/MITSX
DX(4)=(XMIN-XSTART)/MITSX
DO 200 I=1,4
    DX(I)=INT(DX(I))
    DX(I)=REAL(DX(I))
    DX(I)=ABS(DX(I))
    IF ((I.LT.3) .AND. (DX(I).GT.20.)) THEN
        MATSX=MATSX*1.2
        GOTO 210
    ENDIF
200 CONTINUE
D WRITE (3,*) 'TICKS CALC DY'
311 CONTINUE
DY(1)=(YMAX-YSTART)/MATSY
DY(2)=(YMIN-YSTART)/MATSY
MITSY=MATSY/5.
DY(3)=(YMAX-YSTART)/MITSY
DY(4)=(YMIN-YSTART)/MITSY
DO 300 I=1,4
    DY(I)=INT(DY(I))
    DY(I)=REAL(DY(I))
    DY(I)=ABS(DY(I))
    IF ((I.LT.3) .AND. (DY(I).GT.20.)) THEN
        MATSY=1.2*MATSY
        GOTO 311
    ENDIF
300 CONTINUE
RETURN
END
SUBROUTINE GVTDENSITY
C ***********************************************************************
C *
C PROGRAM NAME: SUBROUTINE GVTDENSITY *
C *
C WRITTEN BY: BRUCE SWANSON *
C *
C DATE: MARCH 27, 1987 *
C *
C PURPOSE: SUBROUTINE "GVTDENSITY" READS IN THE DATA FORMATTED FOR *
C THE JOINT DENSITY FUNCTION. *
C *
C SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE *
C *
C DESCRIPTION OF PARAMETERS:
C *
INPUT ARGUMENTS:
C *
COUNT - INTEGER - THE NUMBER OF POINTS TO BE PLOTTED. *
GRAPH. *
XX(N,M) - REAL - THE X VALUE FOR VARIABLE M, CURVE *
NUMBER N, FOR THE DENSITY FUNCTION. *
YY(N,M) - REAL - THE Y VALUE FOR VARIABLE M, CURVE *
NUMBER N, FOR THE DENSITY FUNCTION. *
*
REAL MATL,MITL,XX,YY,ZZ,ERROR,MAXDEV
INTEGER COUNT,INCELLS,ICOUNT,ITIME
CHARACTER XLABEL*30,YLABEL*30,GTITLE*35,NAME*20
COMMON/ONE/XX(6,10000),YY(6,10000),COUNT
COMMON/MISC/INCELLS(10000),ERROR(10000),MAXDEV(10000),
ICOUNT(10000),ITIME(10000)
*
WRITE (3,51)
51 FORMAT (2X,'PLOTTING ROUTINE FOR DENSITY FUNCTION')
WRITE (3,50)
50 FORMAT (2X,'ENTER DATA FILE NAME')
READ (0,60) NAME
60 FORMAT (A20)
OPEN (UNIT=23,FILE=NAME)
READ (23,987) COUNT
987 FORMAT (15)
WRITE (3,*), COUNT
DO 30 I=1,COUNT
   READ (23,628) XX(I,1),YY(I,1)
30 CONTINUE
RETURN
END
SUBROUTINE GVTDISTRIBUTION

C ***************************************************************
C * PROGRAM NAME: SUBROUTINE GVTDISTRIBUTION
C * WRITTEN BY: BRUCE SWANSON
C * DATE: MARCH 27, 1987
C *
C * PURPOSE: SUBROUTINE "GVTDISTRIBUTION" READS IN THE DATA FORMATED
C * FOR THE JOINT DISTRIBUTION FUNCTION.
C *
C * SUBROUTINES AND SUBPROGRAMS REQUIRED: NONE
C *
C * DESCRIPTION OF PARAMETERS:
C *
C * INPUT ARGUMENTS:
C *
C * COUNT - INTEGER - THE NUMBER OF POINTS TO BE PLOTTED.
C *
C * GRAPH.
C *
C * XX(N,M) - REAL - THE X VALUE FOR VARIABLE M, CURVE
C *
C * NUMBER N, FOR THE DISTRIBUTION FUNCTION.
C *
C * YY(N,M) - REAL - THE Y VALUE FOR VARIABLE M, CURVE
C *
C * NUMBER N, FOR THE DISTRIBUTION FUNCTION.
C *
C ***********************************************************************
C
REAL MATL,MTL,XX,YY,ZZ,ERROR,MAXDEV
INTEGER COUNT,INCELLS,ICOUNT,ITIME
CHARACTER XLABEL*30,YLABEL*30,GTITLE*35,NAME*20
* COMMON/ONE/XX(6,10000),YY(6,10000),COUNT
* COMMON/MISC/INCELLS(10000),ERROR(10000),MAXDEV(10000),
*    ICOUNT(10000),ITIME(10000)
* WRITE (3,5000)
5000 FORMAT (2X,"PLOTTING ROUTINE FOR THE DISTRIBUTION FUNCTION")
WRITE (3,50)
50 FORMAT (2X,"ENTER DATA FILE NAME")
READ (0,60) NAME
60 FORMAT (A20)
OPEN (UNIT=23,FILE=NAME)
READ (23,4001) IDUMMY1,IDUMMY2
4001 FORMAT (I5,I5)
READ (23,987) COUNT
987 FORMAT (I5)
WRITE (3,*) COUNT
DO 30 I=1,COUNT
   READ (23,628) XX(I,1),YY(I,1),DUMMY
30 FORMAT (F20.7,F12.7,F12.7)
WRITE (3,204) XX(1,1),YY(1,1)
204 FORMAT (F12.7,F12.7)

30 CONTINUE
RETURN
END
SUBROUTINE LABEL (XMAX, XMIN, YMAX, YMIN, DX, DY, MATSX, MATSY, XSTART, YSTART)

C ***************************************************************
C *
C * PROGRAM NAME: SUBROUTINE LABEL
C *
C * WRITTEN BY: DONALD A. SMITH
C *
C *
C *
C ***************************************************************
C
CHARACTER CNT*10
REAL DX(20), DY(20), MATSX, MATSY
XZ = ABS(XMAX - XMIN)
YZ = ABS(YMAX - YMIN)
CALL JCOLOR(0)
CALL JMOVE (XMIN, YSTART)
CALL JRMOVE (-XZ*.1, 0.)
CALL JJJUST (3, 2)
CALL JFONT (1)
A = (4./100.)*XZ
B = (3./100.)*YZ
CALL JSIZE (A, B)
DO 50 I = 1, 2
  IF (DY(I).LT.0.9) GOTO 60
  DO 70 CTT = 0., DY(I)
    WRITE (CNT, '(F10.2)') CTT*MATSY + YSTART
    CALL JHSTRG (CNT)
    CALL JRMOVE (MATSX, 0.0)
    70 CONTINUE
  CONTINUE
  CALL JMOVE (XMIN, YSTART)
  CALL JRMOVE (-XZ*.1, 0.)
  MATSY = -MATSY
50 CONTINUE
CALL JBASE (0., 1., 0.)
CALL JMOVE (XSTART, YMIN)
CALL JRMOVE (0., -YZ*.1)
CALL JSIZE (B*3./2., A*2./3.)
D WRITE (3,*) DX(1), 'DX1', DX(2), 'DX2'
D WRITE (3,*) MATSX, 'MATSX', YMIN, 'YMIN', YMAX, 'YMAX'
DO 10 I = 1, 2
  IF (DX(I).LT.0.9) GOTO 20
  DO 30 CTT = 0., DX(I)
    WRITE (CNT, '(F10.2)') CTT*MATSX + XSTART
    CALL JHSTRG (CNT)
    CALL JRMOVE (MATSX, 0.0)
10 CONTINUE
30 CONTINUE
30 CONTINUE
20 CONTINUE
   CALL JMOVE(XSTART,YMIN)
   CALL JRMOVE(0.,-YZ*.1)
   MATSX=-MATSX
10 CONTINUE
   CALL JBASE (1.,0.,0.)
   RETURN
   END
SUBROUTINE PLLABEL (XMAX,XMIN,YMAX,YMIN,XLABEL,YLABEL,GTITLE)
C******************************************************************************************
C *
C * PROGRAM NAME: SUBROUTINE PLLABEL
C *
C * WRITTEN BY: DONALD A. SMITH
C *
C *
C ЧАРАКТЕР XLABEL*30,YLABEL*30,GTITLE*35
CALL JPEDGE (2)
CALL JCOLOR(0)
C CALL JRECT (XMIN,YMIN,XMAX,YMAX)
C=ABS(( 6./100.)*(XMAX-XMIN))
D=ABS((5./100.)*(YMAX-YMIN))
C CALL NOBLNK (GTITLE)
CALL JSIZE (C,D)
CALL JJUST (1,2)
C CALL JMOVE (XMIN,YMAX)
CALL JMOVE (-9.*((XMAX-XMIN)/19.),YMAX)
CALL JRMOVE (0.,D*1.25)
CALL JHSTRG (GTITLE)
CALL JSIZE (0.66*C,C.66*D)
CALL JMOVE (XMIN,YMIN)
CALL JRMOVE (0.,-D*.4/.05)
CALL JHSTRG (XLABEL)
CALL JMOVE (XMIN,YMIN)
CALL JRMOVE (-C*.4/.06,0.)
CALL JBSE (0.,1.,0.)
CALL JSIZE (0.66*D,0.66*C)
C CALL JBSE (0.,1.,0.)
CALL JHSTRG (YLABEL)
CALL JBSE (1.,0.,0.)
RETURN
END
SUBROUTINE PLOTT (XMAX, XMIN, YMAX, YMIN, MATL, MITL, @
  XLABEL, YLABEL, GTITLE, XX, YY, IDATN, @
  IDATNY, IDEV, AUTOSC)

C **********************************************************************
C *
C * PROGRAM NAME: SUBROUTINE PLOTT
C *
C * WRITTEN BY: DONALD A. SMITH
C *
C *
C *
C **********************************************************************

REAL MATL, MITL, MATSX, MITSX, MATSY, MITSY, DX(4), DY(4), @
XX(6,10000), YY(6,10000), MALX, MALY, MILX, MILY
LOGICAL AUTOSC
CHARACTER XLABEL*30, YLABEL*30, GTITLE*35

IF (AUTOSC) WRITE (3,*) 'AUTOSC TRUE PLOTT SUB''
  CALL AUTOSCL (XX(1,1), YY(1,1), IDATN, XMAX, XMIN, YMAX, YMIN, MATS, @
  MITSX, MATSY, MITSY, AUTOSC, XSTART, YSTART, DX, DY)

IF (AUTOSC) THEN
  WRITE (3,*) 'JUST DID AUTO SCALE'
ELSE
  WRITE (3,*) 'NO AUTO SCALE'
ENDIF

CALL JBEGIN
CALL JDINIT (IDEV)
CALL JDEVON (IDEV)
CALL JVPORT (-0.3, 0.7, -0.2, 0.8)

IF (ABS(XMAX-XMIN).LT.1) XMIN=XMAX-.1
IF (ABS(YMAX-YMIN).LT.1) YMIN=YMAX-.1

CALL JWINDO (XMIN, XMAX, YMIN, YMAX)
CALL JDEVWN (IDEV, -1.0, 1.0, -0.7, 1.0)
CALL JOPEN

WRITE (3,*) 'OPENED WINDOW'
IW=10000
MALY=MATL*ABS(XMAX-XMIN)
MALX=MATL*ABS(YMAX-YMIN)
MILY/MITL*ABS(XMAX-XMIN)
MILX/MITL*ABS(YMAX-YMIN)

WRITE (3,*) '1ST TICKS CALL'
CALL TICKS (XMAX, XMIN, YMAX, YMIN, MATSX, MATSY, MALX, MALY, DX(1), @
  DY(1), IW, XSTART, YSTART)

WRITE (3,*) '1ST LABEL CALL'
CALL LABEL (XMAX, XMIN, YMAX, YMIN, DX(1), DY(1), MATSX, MATSY, @
  XSTART, YSTART)
IW=8000

177
D WRITE (3,*) '2ND TICKS CALL'
CALL TICKS (XMAX, XMIN, YMAX, YMIN, MITSX, MITSY, MILX, MILY, 
& DX(3), DY(3), IW, XSTART, YSTART)
D WRITE (3,*) 'PLLABEL CALL'
CALL PLLABEL (XMAX, XMIN, YMAX, YMIN, XLABEL, YLABEL, GTITLE)
D CALL JJUST (2, 2)
RR=(6./100.)*ABS(XMAX-XMIN)
TT=(6./100.)*ABS(YMAX-YMIN)
D CALL JSIZE (RR, TT)
D CALL JJUST (2, 2)
D CALL JSIZE (RR*2./6., TT*2./6.)
DO 500 J=1, IDATNY
C CALL JCOLOR(J)
IF (J.GT.1) THEN
   CALL JLSTYL(3)
END IF
D CALL JMOVE (XX(IDATNY,1), YY(IDATNY,1))
D WRITE (3,*) XX(IDATNY,1), YY(IDATNY,1)
DO 100 I=2, IDATN
D IF ((XX(J,I).LE.XMAX) .AND. 
& (XX(J,I).GE.XMIN) .AND. 
& (YY(J,I).LE.YMAX) .AND. 
& (YY(J,I).GE.YMIN)) THEN
   CALL JDRAW (XX(J,I), YY(J,I))
END IF
D WRITE(3,*) XX(J,I), YY(J,I)
D WRITE(3,*) J, 'FIELD', I, 'COUNT'
100 CONTINUE
500 CONTINUE
C CALL JCLOSE
CALL JPause (IDev)
CALL JDEVOF (IDev)
CALL JDEND (IDev)
CALL JEND
RETURN
END
SUBROUTINE TICKS (XMAX,XMIN,YMAX,YMIN,MATSX,MATSY,MLX,MLY,
@ DX,DY,IW,XSTART,YSTART)
C

REAL MATSX,MATSY,MLX,MLY,DX(20),DY(20)
CALL JLSTYL (0)
CALL JCOLOR (7)
CALL JLWIDE (20000)
CALL JMOVE (XMIN,YSTART)
CALL JDRAW (XMAX,YSTART)
CALL JMOVE (XSTART,YMAX)
CALL JDRAW (XSTART,YMIN)
CALL JMOVE (XSTART,YSTART)
D
WRITE (3,*) XSTART,'XSTART',YSTART,'YSTART'
CALL JLWIDE (IW)
DO 20 I=1,2
DO 10 SP=1.,DX(I)
   CALL JRMOVE (MATSX-MLX/2)
   CALL JRDRAW (0.,MLX)
   CALL JRMOVE (0.,-MLX/2)
10 CONTINUE
15 CONTINUE
CALL JMOVE (XSTART,YSTART)
MATSX=-MATSX
20 CONTINUE
CALL JMOVE (XSTART,YSTART)
D
WRITE (3,*)'TICKS CALC DY'
DO 30 I=1,2
DO 40 SP=1.,DY(I)
   CALL JRMOVE (-MLY/2.,MATSY)
   CALL JRDRAW (MLY,0.)
   CALL JRMOVE (-MLY/2.,0.)
40 CONTINUE
45 CONTINUE
CALL JMOVE (XSTART,YSTART)
MATSY=-MATSY
30 CONTINUE
RETURN
END
APPENDIX E

FLOWCHART FOR THE DISCSIM PROGRAM
Subroutine OUTPUT
Input: NAME2
Output: DISREP, MEAN, VARIANCE, STANDARD DEV, NCELLS, COUNT, ZMEAN, Z, ZVARIANCE, ZSTANDARD DEV

Subroutine OUTPUTBOUND
Input: NAME2
Output: DISREP, MEAN, VARIANCE, STANDARD DEV, NCELLS, COUNT, ZMEAN, Z, ZVARIANCE, ZSTANDARD DEV, ZLB, ZUB
APPENDIX F

SAMPLE OUTPUT OF THE DISCRETE SIMULATION PROGRAM
**Random Variable: 1**  
**Distribution:** Normal  
**Mean:** 170.000000  
**Variance:** 425.000000  
**Standard Deviation:** 25.000000  
**Number of Cells:** 19

<table>
<thead>
<tr>
<th>XI Minimum Value</th>
<th>Center of Gravity</th>
<th>XI Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-338.79635416</td>
<td>140.9567140</td>
<td>137.9567811</td>
</tr>
<tr>
<td>137.9567811</td>
<td>144.0872129</td>
<td>148.9635820</td>
</tr>
<tr>
<td>148.9635820</td>
<td>153.1452715</td>
<td>156.89799532</td>
</tr>
<tr>
<td>156.89799532</td>
<td>160.3780729</td>
<td>163.6766685</td>
</tr>
<tr>
<td>163.6766685</td>
<td>166.8554752</td>
<td>170.00000025</td>
</tr>
<tr>
<td>170.00000025</td>
<td>173.1345248</td>
<td>176.3233317</td>
</tr>
<tr>
<td>176.3233317</td>
<td>179.6219271</td>
<td>181.1009467</td>
</tr>
<tr>
<td>181.1009467</td>
<td>185.8547825</td>
<td>191.0364180</td>
</tr>
<tr>
<td>191.0364180</td>
<td>195.9107871</td>
<td>202.0432189</td>
</tr>
<tr>
<td>202.0432189</td>
<td>211.1302860</td>
<td>338.7968616</td>
</tr>
</tbody>
</table>

**Random Variable: 2**  
**Distribution:** Normal  
**Mean:** 29.400000  
**Variance:** 9.000000  
**Standard Deviation:** 3.000000  
**Number of Cells:** 10

<table>
<thead>
<tr>
<th>X2 Minimum Value</th>
<th>Center of Gravity</th>
<th>X2 Maximum Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>-49.6556234</td>
<td>24.4643637</td>
<td>25.5548137</td>
</tr>
<tr>
<td>25.5548137</td>
<td>26.2907055</td>
<td>26.8756298</td>
</tr>
<tr>
<td>26.8756298</td>
<td>27.5774326</td>
<td>27.8279944</td>
</tr>
<tr>
<td>27.8279944</td>
<td>28.2453687</td>
<td>28.6412002</td>
</tr>
<tr>
<td>28.6412002</td>
<td>29.0238570</td>
<td>29.4000000</td>
</tr>
<tr>
<td>29.4000000</td>
<td>29.7761430</td>
<td>30.1587998</td>
</tr>
<tr>
<td>30.1587998</td>
<td>30.5546313</td>
<td>30.9720565</td>
</tr>
<tr>
<td>30.9720565</td>
<td>31.4225674</td>
<td>31.9243702</td>
</tr>
<tr>
<td>31.9243702</td>
<td>32.5092945</td>
<td>33.2451863</td>
</tr>
<tr>
<td>33.2451863</td>
<td>33.93356343</td>
<td>49.4556234</td>
</tr>
</tbody>
</table>

******* Z Function ********
**Number of Unique Z Locations:** 100  
**Mean:** 94.25145  
**Variance:** 771.93585  
**Standard Deviation:** 27.78198

<table>
<thead>
<tr>
<th>Cell No.</th>
<th>Discrete Approximation</th>
<th>Lower Bounds Distribution</th>
<th>Z</th>
<th>Upper Bounds Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>22.5014176</td>
<td>0.0100000</td>
<td>49.8188862</td>
<td>0.0100000</td>
</tr>
<tr>
<td>2</td>
<td>35.7563186</td>
<td>0.0200000</td>
<td>55.7483597</td>
<td>0.0200000</td>
</tr>
<tr>
<td>3</td>
<td>37.2209165</td>
<td>0.0300000</td>
<td>60.3503526</td>
<td>0.0300000</td>
</tr>
<tr>
<td>4</td>
<td>13.8538503</td>
<td>0.0400000</td>
<td>60.8256572</td>
<td>0.0400000</td>
</tr>
<tr>
<td>5</td>
<td>40.7769751</td>
<td>0.0500000</td>
<td>64.3046270</td>
<td>0.0500000</td>
</tr>
<tr>
<td>6</td>
<td>69.0723945</td>
<td>0.0600000</td>
<td>65.7551607</td>
<td>0.0600000</td>
</tr>
<tr>
<td>7</td>
<td>71.9358175</td>
<td>0.0700000</td>
<td>67.9441992</td>
<td>0.0700000</td>
</tr>
<tr>
<td>8</td>
<td>53.2765557</td>
<td>0.0800000</td>
<td>65.7620284</td>
<td>0.0800000</td>
</tr>
<tr>
<td>9</td>
<td>54.0097765</td>
<td>0.0900000</td>
<td>71.3571536</td>
<td>0.0900000</td>
</tr>
<tr>
<td>90</td>
<td>131.3336585</td>
<td>0.9000000</td>
<td>312.9776360</td>
<td>1.0000000</td>
</tr>
<tr>
<td>91</td>
<td>131.4349777</td>
<td>0.9100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>132.5551793</td>
<td>0.9200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>133.5567277</td>
<td>0.9300000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>135.4735513</td>
<td>0.9400000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>139.3271784</td>
<td>0.9500000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>141.9125800</td>
<td>0.9600000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>142.9394477</td>
<td>0.9700000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>144.6544796</td>
<td>0.9800000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>150.8930502</td>
<td>0.9900000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>157.1307569</td>
<td>1.0000000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX G

NOTES ABOUT THE COMPARISON OF THE MONTE CARLO SIMULATION VS. THE DISCRETE SIMULATION
METHODS OF COMPARISON

The Monte Carlo program was written using some of the same subroutines that are used in the discrete simulation program. These subroutines are: DISTRIBUTION, FUNCTION, OUTPUT, and SORT. Subroutine BOOK is also used, however, a few lines of code had to be modified for the generation of the random numbers. The random numbers are generated using a normal random number generator provided by Harris Computer Systems.

Because the two methods are slightly different, a method of comparison had to be devised. Three methods were possible:

1. Use the same number of points to represent the random variables' density functions.
2. Use the same number of possible combinations of outcomes to represent the joint random variable's distribution function.
3. Use the same number of unique outcomes to represent the joint random variable's distribution function.
Method 1 was thought to give a distinct advantage to the discrete approximation program, while method 2 would give an advantage to the Monte Carlo simulation. Method 3 was chosen to be used as the method of comparison because it is somewhat of a compromise between methods 1 and 2. However, this method does give the discrete simulation a slight advantage over the Monte Carlo method for the example discussed in Chapter 5.

The discrete simulation program was run first each time the two methods were compared. After each discrete approximation, the number of unique outcomes \( m \), was noted. The joint distribution function was then approximated using the Monte Carlo program by generating \( m \) pairs of random variables \( X \) and \( Y \).

CPU TIME COMPARISON

Figures 5.18 and 5.19 were plotted to give an overall comparison of the computer processing unit time for both programs. A binary sorting routine was used for both programs. Since the outcomes generated by the discrete method are somewhat ordered, the discrete values are sorted quicker than the data that is created randomly from the Monte Carlo program. It should be noted that
these times are largely dependent on the efficiency of the sorting routine. Quicker times could be achieved by using a faster method of sorting the outcomes. For example, a quick sort subroutine would decrease the run time of the Monte Carlo program since this method sorts random numbers faster than ordered numbers. Also, a heap sort could be used to decrease the CPU time for the ordered discrete values.
ACKNOWLEDGEMENTS

I would like to express my appreciation to Dr. Fredric C. Appl, Professor of Mechanical Engineering, for his guidance and encouragement throughout the course of this study.

I am also grateful to Dr. Hugh S. Walker and Dr. Mark S. McNulty for serving as graduate committee members.

Appreciation is also extended to the Department of Mechanical Engineering for their financial support throughout my graduate study.

My thanks are also due to Donald A. Smith for the graphics routines he wrote for the Harris computer system.

And finally, I would like to thank my parents for their moral and financial support during all of my years of study.
VITA

Bruce Eugene Swanson

Candidate for the Degree of

Master of Science

Thesis: THE APPROXIMATION OF JOINT DISTRIBUTION FUNCTIONS FOR APPLICATION IN PROBABILISTIC MECHANICAL DESIGN

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Kansas City, Kansas, January 22, 1963, the son of Eugene A. and Sylvia J. Swanson.

Education: Graduated from Shawnee Mission East High School in 1981; received the Bachelor of Science degree from Kansas State University in 1985; completed the requirements for the Masters of Science degree from Kansas State University in 1987.

Honors: Tau Beta Pi - All Engineering Honorary Pi Tau Sigma - Mechanical Engineering Honorary

Professional Experience: Two summers as an Engineer III in the Metal Products Department at Allied/Bendix, Kansas City Division. Have accepted employment with Sandia National Laboratories in Albuquerque, New Mexico, in the Rocket Systems Division II.
THE APPROXIMATION OF JOINT DISTRIBUTION FUNCTIONS FOR APPLICATION IN PROBABILISTIC MECHANICAL DESIGN

by

BRUCE EUGENE SWANSON

B.S., Kansas State University, 1985

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987
ABSTRACT

In this study, a discrete simulation technique is developed to approximate the joint probability distribution functions of complex algebraic expressions containing several independent random variables. The discrete simulation package is written in Fortran on a Harris H-800 Super Mini-Computer.

The discrete approximation is compared to the traditional Monte Carlo technique for a joint probability distribution function that can be solved exactly. The accuracy of the discrete simulation is found to be better than the Monte Carlo simulation.

A cantilever I-beam is analyzed to illustrate how the discrete simulation package can be used by design engineers in a variety of engineering applications.

A bounding program is also discussed that bounds the exact joint probability distribution function of an algebraic expression. Although a formal proof has not been developed, the results of several different examples show considerable promise for further study.