Investigation and Evaluation of Random Number Generators for Digital Implementation

1984

Ylberto V. Ruiz
University of Central Florida

Find similar works at: https://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation
https://stars.library.ucf.edu/rtd/4677

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lee.dotson@ucf.edu.
INVESTIGATION AND EVALUATION OF RANDOM NUMBER GENERATORS FOR DIGITAL IMPLEMENTATION

BY

YLBERTO VIDAL RUIZ
B.S.E.E., University of Miami, 1978

THESIS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering in the Graduate Studies Program of the College of Engineering at the University of Central Florida
Orlando, Florida

Fall Semester
1984
The continuous improvement in the speed of digital components in conjunction with reduction of size has brought about a revolutionary age of microprocessors. Mathematical functions, which at one time could only be implemented by complex analog circuitry, can now be easily implemented via microprocessors and high density digital components.

Principles of random number generation must be understood in order to implement pseudo-random algorithms in a digital random frequency generator (DRFG) design. Chapter 1 is a discussion of several types of random number algorithms which have been used in the past and outlines the deficiencies and advantages associated with each individual algorithm. In particular, problems such as cycling and maximum period deficiency are discussed. The discussions in Chapter 1 lead to the selection of a random number algorithm which can be used in a DRFG design.

There are other characteristics which should be observed in the evaluation of acceptable random number algorithms. In Chapter 2 three tests are described which can be applied in order to test the algorithm for the well known uniformity and independence criteria. These tests
are implemented in a Fortran program which is used to evaluate the algorithm selected in Chapter 1. The random number generator evaluation program (RNGEP) listing is presented in Appendix B. The results of the tests applied to the DRFG random number algorithm are presented in Appendix C.
ACKNOWLEDGMENTS

I would like to thank the members of my graduate committee, Richard N. Miller, Brian E. Petrasko, Robert L. Walker, Richard C. Harden, and Donald C. Malocha for their time and efforts expended during my course work. A special thanks to Coleman Research Corporation for their support and allowing me to use their facilities and equipment throughout the preparation of the thesis. Most importantly, a special thanks to my family who had to put up with my frequent absence while I was pursuing my graduate degree.
# TABLE OF CONTENTS

List of Tables ................................................. v
List of Figures .................................................. vi

INTRODUCTION .................................................. 1

Chapter

I. PSEUDO-RANDOM ALGORITHMS .............................. 3
   Middle Square Algorithm ................................ 4
   Multiplicative Congruential Algorithm .................. 8
   Linear Congruential Algorithm ............................ 9

II. PSEUDO-RANDOM ALGORITHM EVALUATION ............... 11
   Statistical Measurements ................................. 11
   Statistical Tests (Uniformity) ........................... 18
   Statistical Test (Independence) .......................... 24

III. CONCLUSIONS ............................................... 28

APPENDICES

Appendix A. Statistical Tables ............................... 29
Appendix B. Random Number Generator Evaluation Program
   Listing ...................................................... 33
Appendix C. Random Number Algorithm Test Results ........ 45

REFERENCES ................................................... 61
LIST OF TABLES

I. Kolmogorov-Smirnov Acceptance Limits .................. 30
II. Chi-Square Acceptance Limits ............................ 31
III. Normal Distribution Constants .......................... 32
## LIST OF FIGURES

1. Uniform Probability Density Distribution .......... 12  
2. Discrete Cumulative Distribution Function .......... 17  
3. Case 1 Discrete Cumulative Distribution ............ 48  
4. Case 1 White Noise Test ................................ 49  
5. Case 2 Discrete Cumulative Distribution ............ 53  
6. Case 2 White Noise Test ................................ 54  
7. Case 3 Discrete Cumulative Distribution ............ 56  
8. Case 3 White Noise Test ................................ 57  
9. Case 4 Discrete Cumulative Distribution ............ 59  
10. Case 4 White Noise Test ................................ 60
INTRODUCTION

The continuous improvement in the speed of digital components in conjunction with reduction of size has brought about a revolutionary age of microprocessors. Mathematical functions, which at one time could only be implemented by complex analog circuitry, can now be easily implemented via microprocessors and high density digital components. Problems such as:

1) offsets
2) biases
3) thermal drift
4) component tolerances
5) high power consumption

associated with analog design can be almost completely eliminated by the use of digital components. Designs, which at one time included complex relay logic and both active and passive filter synthesis, have been replaced by advanced computer logic and digital filter techniques. Digital autopilots are exemplary of such advances in engineering technology. Similar advances are essential in the field of communications. Outdated analog random frequency generators can now be replaced by the powerful computational and complex logical capabilities of present microprocessor technology. Utilizing microprocessors for the
renovation of such equipment also eliminates the familiar problems of packaging and cost.

Principles of random number generation must be understood in order to implement pseudo-random algorithms in a digital random frequency generator (DRFG) design. Chapter 1 is a discussion of several types of random number algorithms which have been used in the past and outlines the deficiencies and advantages associated with each individual algorithm. In particular, problems such as cycling and maximum period deficiency are discussed. The discussions in Chapter 1 lead to the selection of a random number algorithm which can be used in a DRFG design.

There are other characteristics which should be observed in the evaluation of acceptable random number algorithms. In Chapter 2 three tests are described which can be applied in order to test the algorithm for the well known uniformity and independence criteria. These tests are implemented in a Fortran program which is used to evaluate the algorithm selected in Chapter 1. The random number generator evaluation program (RNGEP) listing is presented in Appendix B. The results of the tests applied to the DRFG random number algorithm are presented in Appendix C.
CHAPTER I

PSEUDO-RANDOM ALGORITHMS

In many engineering applications it is often necessary to model random variables. One such application is a Monte Carlo Simulation where selected parameters are allowed to approximate random variation using Gaussian, Uniform or Exponential distributions. A random draw is then made on each selected variable using a pseudo-random number algorithm. These parameters are then utilized within the simulation and describe the behavior of particular systems when their values vary from the nominal. Another application is a microprocessor based digital random frequency generator. In a digital random frequency generator a random draw is made from a pseudo-random number algorithm in order to generate random counts. These random count values, which are stored in individual digital counters, produce signals which randomly vary in their frequency content.

An algorithm which is capable of generating a random sequence of numbers is defined as a random number algorithm. Such algorithms have periods of cycling (i.e., they have a period such that random numbers are generated effectively but repeat after each multiple of the period). In digital simulations this period is determined by the maximum word length and this period is referred to as the modulus of the generator. For example if a register internal to a microprocessor is 16 bits long
the modulus of an algorithm implemented within the microprocessor is $2^{16}$ or 65536. Since these algorithms do not have infinite cycling periods they are called pseudo-random algorithms.

The three types of pseudo-random algorithms which are typically considered are:

a) Middle Square  
b) Multiplicative Congruential  
c) Linear Congruential

**Middle Square Algorithm**

The middle square algorithm is implemented by squaring the previous number (or the seed) and extracting the middle $k$ digits of the squared result ($X$) generated by the algorithm. The middle $k$ digits are arbitrarily chosen with one restriction. After the middle $k$ digits are extracted from $X$, an equal number of digits must remain on the ends of $X$. If this condition is not satisfied, $X$ must be pre-multiplied by ten before extracting the middle $k$ digits. The extracted number ($Y$) is then divided by $10^k$ to generate a fractional number between 0 and 1. An example of the computational sequence of the middle square algorithm is shown below:
Assume: \( k = 4 \)

seed = 314

\[ 10^k = 10000. \]

Computations:

1st pass

\[ x = 98596 \]
\[ XP = 985960 \]
\[ y = 8596 \]
\[ z(1) = .8596 \]

2nd pass

\[ x = 73891216 \]
\[ y = 8912 \]
\[ z(2) = .8912 \]

3rd pass

\[ x = 79423744 \]
\[ y = 4237 \]
\[ z(3) = .4237 \]
One of the deficiencies of the middle square algorithm is that it frequently generates sequences of numbers that approach zero rapidly. For example if the seed 6992 is used, the random numbers generated approach zero very rapidly. This sequence is shown below:

6992
8880
8544
9999
9800
400
6000
0000
0000

Another commonly encountered deficiency of this method is that it has varying cycling periods and these periods can be very short, depending on the seed used. For example if a seed value of 1516 is selected the random sequence of numbers repeats after the 32\textsuperscript{nd} sample. The values of the samples shown below illustrate the problem. The underlined numbers indicate repeated samples.
<table>
<thead>
<tr>
<th>Sample Number</th>
<th>Random Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1516</td>
</tr>
<tr>
<td>4</td>
<td>1536</td>
</tr>
<tr>
<td>7</td>
<td>4090</td>
</tr>
<tr>
<td>10</td>
<td>6641</td>
</tr>
<tr>
<td>13</td>
<td>2396</td>
</tr>
<tr>
<td>16</td>
<td>8370</td>
</tr>
<tr>
<td>19</td>
<td>4537</td>
</tr>
<tr>
<td>22</td>
<td>7683</td>
</tr>
<tr>
<td>25</td>
<td>3033</td>
</tr>
<tr>
<td>28</td>
<td>3585</td>
</tr>
<tr>
<td>31</td>
<td>9875</td>
</tr>
<tr>
<td>34</td>
<td>1406</td>
</tr>
</tbody>
</table>

These two commonly encountered problems suggest that the middle square algorithm is not very applicable. The investigation of an applicable pseudo-random algorithm is continued with the multiplicative congruential algorithm.
Multiplicative Congruential Algorithm

The multiplicative congruential algorithm is implemented by multiplying the previously sampled number (or seed) by a constant \( a \) and dividing the result by \( 2^k \) (modulo \( k \)), where \( k \) represents the maximum binary word length of a register inside a microprocessor. In terms of binary register operations this represents shifting the binary digits \( k \) times to the right of the decimal point.

The multiplicative congruential algorithm is shown below:

\[ X_{n+1} = aX_n \pmod{k} \]

Since the maximum word length is \( k \) digits long, the updated random number is also truncated after \( k \) digits. For example if a register has a maximum word length of 4 bits, a seed value of 1, and a constant \( a \) with value of 5 the following calculations would result:

<table>
<thead>
<tr>
<th>Sample Number</th>
<th>Decimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MOD(5*1)/16 = .3125</td>
<td>0101 * 0001 = 0000.0101</td>
</tr>
<tr>
<td>2</td>
<td>MOD(5*5)/16 = .5625</td>
<td>0101 * 0101 = 0001.1001</td>
</tr>
<tr>
<td>3</td>
<td>MOD(5*9)/16 = .8125</td>
<td>0101 * 1001 = 0010.1101</td>
</tr>
<tr>
<td>4</td>
<td>MOD(5*13)/16 = .0625</td>
<td>0101 * 1101 = 0100.0001</td>
</tr>
</tbody>
</table>
The operations involved in both decimal and binary are shown in the previous example. Notice the underlined binary values represent the truncated numbers \( (X_n) \) which are used in the arithmetic computation. It should also be observed that if one more sample had been taken, the sequence of numbers would begin to cycle. This example demonstrates that since there are \( 2^k \) possible numbers in a register of \( k \) binary digits, these algorithms fail to generate a maximum period of \( 2^k \), which is expected from a good algorithm. Furthermore, only certain values of \( a \) and \( X_0 \) will work effectively. Another algorithm commonly used involves adding a constant to the value derived by the multiplicative congruential algorithm and is discussed in the section which follows.

**Linear Congruential Algorithm**

The linear congruential algorithm (LCA) is a general form of the multiplicative congruential algorithm. The LCA adds a bias \( b \) to the product of \( a \) and \( X_n \). The constant \( b \) is the main reason why the multiplicative congruential algorithm does not satisfy the maximum period of \( 2^k \). This is also the reason why the LCA is preferred for many engineering applications. The LCA is given below:

\[
X_{n+1} = aX_n + b \quad \text{(modulo } k)\]

Not all values of \( a \) and \( b \) will produce a maximum period. In order to select appropriate values of \( a \), \( b \), and \( k \), specific rules must be
followed. One of the rules given by Yakowitz (1977), is that \( b \) and \( k \) must be relatively prime. This means that \( b \) and \( k \) have no factors in common other than 1 (e.g., 5 and 2 are relatively prime). In the multiplicative congruential algorithm, \( b \) is zero and has any number as a factor. Hence, this rule is violated. One example of a LCA is shown below (Abramowitz and Stegun 1964):

\[
X_{n+1} = 129X_n + 1 \quad \text{(modulo 16)} \tag{1}
\]

There are various other types of numerical algorithms that can be discussed. However, the three algorithms previously discussed are the most popular. Furthermore the LCA, which generates a maximum period of \( 2^k \), is sufficient for the cycling demand of the pseudo-random number algorithm. Other criteria of goodness are uniform probability distributions of the sampled random numbers and independence between samples generated by the algorithm. Statistical tests must be performed on the algorithm given by equation (1) to test for uniformity and independence. Definitions of acceptance criteria for the selected pseudo-random number algorithm will be given in the following chapter.
CHAPTER II
PSEUDO-RANDOM ALGORITHM EVALUATION

Statistical Measurements

The two principal parameters which represent qualitative criteria of goodness of random number generating algorithms are uniformity and independence (whiteness). A series of tests must be performed on the sequence of numbers generated by the algorithm in order to evaluate acceptance criteria of uniformity and independence for the selected algorithm. In order to perform such tests methods of computing statistical parameters must be reviewed and incorporated into a digital simulation. These tests must include a theoretical description of statistical parameters of the uniform probability distribution as well as descriptions of confidence parameters for the statistical results. Specifically, the highly accepted Kolomogorov-Smirnov Statistic (K-S Statistic) will be investigated as a method of applying specifications to the results obtained from random number algorithms (Lindgren and McElrath 1978).

In a DRFG design, it is desirable to have an equal likelihood of any count being drawn at any one time in order to generate uniformly distributed random frequencies. Furthermore, the same requirement may be demanded for the duty cycle of the signal in order to generate random signal activation and deactivation times. In order to achieve both
uniformly distributed random activation/deactivation times and counts, a uniform distribution of the generated numbers is essential. The uniform density distribution must be defined by

$$f(x) = \frac{1}{x_2 - x_1}, \quad x_2 \leq x \leq x_1$$  \hspace{1cm} (2)$$

$$= 0 \quad \text{otherwise,}$$

since the probability of any outcome over all possibilities defined by

$$p(x) = \int_{-\infty}^{\infty} f(x) dx = 1$$  \hspace{1cm} (3)$$

must be satisfied. The quantity $x_2 - x_1$ is defined to be the density range and is shown in Figure 1.
If the probability from a discrete point \( a \) to a point \( b \) is to be determined, given the probability distribution \( p(x) \), the above equation becomes

\[
p(x) = \int_a^b f(x) \cdot dx \quad ; \quad a \leq x \leq b
\]

(4)

and for the uniform density distribution

\[
p(x) = \int_a^b \frac{dx}{x_2 - x_1} = \frac{b - a}{x_2 - x_1} \quad ; \quad a \leq x \leq b.
\]

(5)

Furthermore, if \( b = x_2 \) and \( a = x_1 \),

\[
p(x) = 1 \quad ; \quad a \leq x \leq b.
\]

(6)

The mean (\( \mu \)) and variance (\( \sigma^2 \)) of a continuous density distribution \( f(x) \) is given respectively by

\[
\mu(x) = \int_{-\infty}^{\infty} xf(x) \cdot dx
\]

(7)

\[
\sigma^2(x) = \int_{-\infty}^{\infty} x^2 f(x) \cdot dx - \mu^2(x)
\]

(8)
For the uniform density distribution given in Figure 1, the mean and variance is shown as follows:

\[
\mu(x) = \int_{x_1}^{x_2} \frac{x}{x_2 - x_1} \, dx \\
= \frac{(x_2 + x_1)}{2} 
\]

(9)

\[
\nu(x) = \int_{x_1}^{x_2} \frac{x^2}{x_2 - x_1} \, dx - \frac{(x_2 + x_1)^2}{4} \\
= \frac{(x_2 - x_1)^2}{12} 
\]

(10)

(10a)

and the standard deviation (\(\sigma\)) is defined as the square root of the variance,

\[
\sigma(x) \overset{A}{=} \nu(x)^{1/2} 
\]

(11)

\[
\sigma(x) = \frac{x_2 - x_1}{2\sqrt{3}} 
\]

(11a)

The cumulative distribution function (CDF) is defined by

\[
F(x) \overset{A}{=} \int_{-\infty}^{x} f(x) dx 
\]

(12)
The continuous CDF for the uniform density function given by equation (2) is computed by substituting equation (2) into equation (12) as shown below:

\[
F(x) = \begin{cases} 
0 & ; -\infty < x < x_1 \\
\frac{x - x_1}{x_2 - x_1} & ; x_1 \leq x < x_2 \\
1 & ; x_2 \leq x < +\infty
\end{cases}
\]

(13)

Inferences regarding statistical parameters based on a finite number of observations must often be made. In such situations a discrete version of the previously derived equations is used. These equations will be utilized in a program to compute necessary statistical parameters.

The arithmetic mean of a finitely sampled system is defined by the following:

\[
\mu_D = \frac{1}{n} \sum_{i=1}^{n} u_i ,
\]

(14)

where \( n \) equals the total number of samples and \( u_i \) equals each sampled data magnitude. Notice the similarity between equation (14) and (9).
where \( n \) is equivalent to the density range \((x_2 - x_1)\) and \( \sum u_i \) is equivalent to \( \int x \, dx \). A measure of dispersion of any discrete system, the variance, is defined by

\[
V_D = \frac{\sum_{i=1}^{n} u_i^2 - (\sum_{i=1}^{n} u_i)^2/n}{n-1}.
\]

(15)

Often, a more useful measure of dispersion is the standard deviation, which is defined exactly the same as for continuous systems:

\[
\sigma = \left( V_D \right)^{1/2}
\]

(16)

The cumulative distribution for a finitely sampled system is defined by:

\[
F_c(x) = \sum_{n=1}^{i} p(x_n), \quad 0 \leq F_c(x) \leq 1
\]

(17)

where \( i \) represents the total number of intervals and \( p(x_n) \) is the sampled probability within that interval. For example, assume a die is rolled ten times and the number of times that the numbers 1 through 6 appears on the face of the die is counted and tabulated. Each face of the die is the interval for this example. The results are tabulated
below and the plot of the corresponding discrete CDF is given in Figure 2.

<table>
<thead>
<tr>
<th>Number on face (n)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of occurences</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Probability $p(x_n)$</td>
<td>.2</td>
<td>.1</td>
<td>.3</td>
<td>.2</td>
<td>.1</td>
<td>.1</td>
</tr>
<tr>
<td>Cum. Probability $F_c(x)$</td>
<td>.2</td>
<td>.3</td>
<td>.6</td>
<td>.8</td>
<td>.9</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Figure 2. Discrete Cumulative Distribution Function.
Statistical Tests (Uniformity)

A method of determining the acceptance of the sampled data must be created to assure that the observed sample is typical of any sample taken from a population. That is, a method of establishing some "degree of confidence" that observed data is typical must be defined. Such confidence coefficients (McClave and Dietrich 1979) evolve from one of the most important theorems of statistical theory, the Central Limit Theorem. The Central Limit Theorem states that the mean ($\bar{x}$) of a sampled set taken from a population will have normal distribution properties for large sample sizes. These properties are valid regardless of the population's probability distribution (McClave and Dietrich 1979). By computing the area under the normal probability distribution curve (which is well known) and comparing the measured statistics to the hypothetical statistics, a probability that the measured statistics will be the same or approximately equal to the hypothetical value can be computed. This probability is expressed in terms of percent or standard deviations. Hence, the confidence coefficient is the probability that a measured value from the population lies within the hypothetical area of the normal probability distribution. Similarly, a confidence coefficient can be computed for the area under a Chi-square ($X^2$) probability distribution.

The Chi-square distribution typically provides information about a sampled variance when compared to the population's variance. A statistical test which is based on the Chi-square distribution is the Chi-square Test. However, the Chi-square test can be utilized to
establish a measure of uniformity for a given sampled data set containing n partitions. The measure of uniformity is computed taking the difference between the sampled frequency and expected frequency of observations which fall in each of the n partitions. The sum of the square of these differences over the n partitions (Chi-square statistic) is compared to the area under the Chi-square distribution for a given confidence coefficient. The equation below clarifies the procedure used to compute the Chi-square statistic ($X^2$).

$$X^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$  \hspace{1cm} (18)

where,

- $O_i =$ measured number of outcomes for the $i^{th}$ partition
- $E_i =$ expected number of outcomes for the $i^{th}$ partition

If the computed $X^2$ value exceeds the theoretical value for a given confidence interval, the uniformity hypothesis is rejected. If the computed $X^2$ value is less than or equal to the theoretical value the hypothesis is accepted. For example, assume some set of randomly generated numbers ranging from zero to one fall in the partitions $x_1$ through $x_5$, such that
and \( x_1 \) through \( x_5 \) represent the values generated by the pseudo-random number generator. These values can be utilized in a DRFG to formulate random counts which consequently generate signals containing random frequencies.

The particular Chi-square statistic of interest can be defined by

\[
X^2_1 = \frac{5}{i} \sum_{j=1}^{5} (N_j(i) - N_H)^2
\]

where,

\( i = \text{sample size} \)

\( N_j(i) = \text{sampled quantity of random values within the } j^{\text{th}} \text{ partition.} \)

\( N_H = \text{hypothesized quantity of random values within the } j^{\text{th}} \text{ partition.} \)

The hypothesized parameter \( N_H \) is dependent on sample and partition size and is defined by

\[
N_H = \frac{i}{p}
\]

where

\( p = \text{number of partitions.} \)

Notice that if the pseudo-random number generator does not exhibit uniform probability statistical properties, the value \( N_j(i) \) will differ
from the hypothesized value and produce a Chi-square value which rejects
the uniform hypothesis. In order to obtain the Chi-square value which
tests the hypothesis, the confidence level and degrees of freedom (one
less than the number of partitions) must be obtained. The author will
select a 90 percent confidence interval with four degrees of freedom (5
partitions). A table of the Chi-square values for selected acceptance
percentiles and degrees of freedom are presented in Appendix A (Lindgren
and McElrath 1978). The value obtained from this table for the desired
degrees of freedom and confidence interval is 7.78. Hence, a
pseudo-random number generator exhibiting acceptable uniformity
characteristics must produce a Chi-square test value less than 7.78.

Similarly, the Kolomogorov-Smirnov test for uniformity (Yakowitz
1977) is used to measure the sample cumulative distribution against the
hypothetical uniform cumulative distribution and establish an acceptance
criteria, given a confidence interval for the sampled cumulative
distribution. Hence, the parameter of interest in the K-S test is the
difference between the hypothetical and sample cumulative distribution
functions. The difference is denoted by $D_K$. The maximum value on $D_K$
($D_{KM}$) is compared to a tabulated hypothetical difference for various
confidence intervals. Appendix A, contains the hypothetical values for
up to 80 samples (Lindgren and McElrath 1978). If the number of samples
exceeds 80, the asymptotic approximation is used. The maximum numerical
difference for any confidence interval ($x$) is defined by:
\( D_{km} = \text{Max} \left| F_s(z) - F_h(z) \right|, \text{ all } z \) (19)

where,

\( F_s(z) \) - sampled cumulative distribution for variable \( z \)

\( F_h(z) \) - hypothetical cumulative distribution for variable \( z \)

For a selected confidence interval \( x \), if the tested parameter is within the interval of acceptance, the parameter is accepted and will be within the acceptance interval \( x \) percent of the time.

The hypothetical acceptance difference \( (D_{ax}) \) (Yakowitz 1977) given by the asymptotic formula for samples greater than or equal to 35 and for the 90 percent confidence interval \( (x = .90) \) is:

\[
D_{a .90} = \frac{1.22}{\sqrt{n}}
\] (20)

\( n \) = number of samples.

Since the theoretical uniform CDF is known, the difference between test and hypothetical values is an indication of the random number generator's uniformity. This difference should approach zero as the number of samples approaches infinity if the random number generator possesses uniform properties.

For example, assume a random number generator exists with the capability of generating numbers within a range of 1. The sorted random
numbers generated by the pseudo-random number algorithm are shown below:

<table>
<thead>
<tr>
<th>Sample Number</th>
<th>Sorted Random Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0081 0.0136 0.0294 0.0427 0.0433</td>
</tr>
<tr>
<td>6</td>
<td>0.1598 0.1605 0.1931 0.1969 0.2028</td>
</tr>
<tr>
<td>11</td>
<td>0.2090 0.2109 0.2208 0.2249 0.2285</td>
</tr>
<tr>
<td>16</td>
<td>0.2501 0.2638 0.2792 0.2967 0.3381</td>
</tr>
<tr>
<td>21</td>
<td>0.3611 0.3941 0.4143 0.4436 0.4707</td>
</tr>
<tr>
<td>26</td>
<td>0.4744 0.4786 0.4977 0.5076 0.5804</td>
</tr>
<tr>
<td>31</td>
<td>0.5877 0.6155 0.6188 0.7110 0.7148</td>
</tr>
<tr>
<td>36</td>
<td>0.7149 0.7382 0.7384 0.7499 0.7946</td>
</tr>
<tr>
<td>41</td>
<td>0.7970 0.8163 0.8166 0.8298 0.8434</td>
</tr>
<tr>
<td>46</td>
<td>0.8695 0.8865 0.8869 0.9138 0.9590</td>
</tr>
</tbody>
</table>

The maximum numerical difference occurs at the 19th sample and is computed by:

\[
D_{km} = \left| .2967 - .3800 \right| = .0833
\]

and the K-S value for the 90 percent confidence coefficient (10 percent significance level) from Table I is computed by the asymptotic approximation of equation (20) and is equal to 0.1725. Hence, the K-S test for uniformity is passed.

The frequency counts for the five partitions discussed for the Chi-square test is given below:

\[
x_1 = 9 \\
x_2 = 13 \\
x_3 = 9 \\
x_4 = 10 \\
x_5 = 9
\]
The Chi-square statistic is computed from equation (18a) as:

\[ x^2 = \frac{(9 - 10)^2}{10} + \frac{(13 - 10)^2}{10} + \frac{(9 - 10)^2}{10} + \frac{(10 - 10)^2}{10} + \frac{(9 - 10)^2}{10} \]

\[ x^2 = 1.20 \]

This value is well within the 90% hypothetical \( x^2 \) value of 7.78.

Thus, the tested random number algorithm passes both tests for uniformity. These tests were performed on the algorithm defined by equation (1) for various seed values and sample sizes. The results are presented in Appendix C.

**Statistical Test (Independence)**

The previous sections have described methods for evaluating the uniformity of random number algorithms. However, algorithms must also be evaluated for independence ("whiteness"), since a random number algorithm can pass uniformity acceptance tests and generate values which are correlated to previous values. One test for whiteness evolves from measured statistical values generated by a white noise process. Since an unbiased white noise process contains a zero mean value, the signal strength is measured by its average power or variance (\( S_T \) within a
"record length" \( T \) by the relationship:

\[
S_T = \frac{1}{T} \int_{-T/2}^{T/2} x^2(t) dt \tag{21}
\]

or

\[
S_T = \sum_{m=-\infty}^{\infty} |X_m|^2 \tag{22}
\]

where, \( X_m \) is the complex amplitude at the \( m \text{th} \) harmonic frequency \( f_m = \frac{m}{T} \). The complex amplitude can be measured by the Fourier transform of the signal \( x(t) \) given by:

\[
X_m = \frac{1}{T} \int_{-T/2}^{T/2} x(t) e^{-2\pi f_m t} dt. \tag{23}
\]

For discrete signal observations at times \( t = -n\Delta, -(n-1)\Delta, \ldots, (n-1)\Delta \), where \( \Delta \) is the time between observations and \( n \) is the total number of observations \( (N) \) divided by two \( (n = N/2) \), the variance is measured by:

\[
S_T = \sum_{m=-n}^{n-1} |X_m|^2 \tag{24}
\]

\[
X_m = \frac{1}{N} \sum_{t=-n}^{n-1} x_t e^{-j2\pi mt/N\Delta} \tag{25}
\]

and by allowing the record length to approach infinity, equations (21) and (22) are defined as the variance of a signal and is evaluated by:

\[
\sigma^2 = \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{T/2} x^2(t) dt \tag{26}
\]
\[ \sigma^2 = \lim_{T \to \infty} \sum_{m=-\infty}^{\infty} \frac{1}{T} \mathbb{E}[X_m^2] \]  

(27)

and,

\[ T |X_m|^2 = P_{zz}(f) = \frac{1}{T} \left| \int_{-T/2}^{T/2} x(t)e^{-2\pi ft} dt \right|^2 \]

(28)

where, \( P_{zz}(f) \) is defined as the sample power spectrum. The sample spectrum for the discrete version is defined as:

\[ P_{zz}(f) = \frac{1}{N\Delta} \left| \sum_{t=-n}^{n-1} x_t e^{-2\pi ft} \right|^2 \]

\[ = \frac{1}{N\Delta} \left\{ \left( \sum_{t=-n}^{n-1} x_t \cos 2\pi ft \right)^2 + \left( \sum_{t=-n}^{n-1} x_t \sin 2\pi ft \right)^2 \right\} \]

\( -1/2\Delta < f < 1/2\Delta \)

(29)

Since the power spectrum for wideband white noise is constant for all frequencies, the sample integrated spectrum is expected to be linear with frequency. The integral of the sample power spectrum is defined as:

\[ I(f_k) = \frac{1}{N\Delta} \sum_{k=1}^{N/2} P_{zz}(f_k) \]

(30)

where,

\[ f_k = \frac{k}{N\Delta} \quad (k = 0,1,2,\ldots,N/2) \]
In order to normalize $I(f_k)$ with respect to the sample power spectrum variance, the quantity $I(f_k)$ is divided by $S_x^2$. The normalized integrated sample spectrum can now be defined as:

$$I(f_k) = \frac{1}{N \Delta S_x^2} \sum_{k=1}^{N/2} P_{zz}(f_k)$$  \hspace{1cm} (31)$$

where,$$S_x^2 = \text{measured variance.}$

The integrated sample power spectrum for a system containing uncorrelated noise is a "ramp" function with a slope of 1. The K-S test can be applied by superimposing the upper and lower significance level boundaries on the integrated power spectrum. The discrete CDF of the integrated power spectrum must lie between the upper and lower significance level limits in order to pass the K-S test for white noise. The results of the white noise K-S test for the algorithm defined by equation (1) for various seed values and sample sizes are illustrated in Appendix C.

The tests mentioned in the previous sections establish a criteria of goodness for both uniformity and independence. These equations have been incorporated in a FORTRAN VII program (RNGEP) which evaluates any random number algorithm at given significance levels. The results given in Appendix C were generated by the program RNGEP.
CHAPTER III
CONCLUSIONS

The methods for evaluating pseudo-random number algorithms presented in this thesis have been incorporated in a Fortran VII program (RNGEP) presented in Appendix B. These methods provide the analyst with adequate means of evaluating pseudo-random number algorithms against the uniformity and independence criteria described in this thesis. Particularly, the Chi-square test for uniformity and K-S test for both uniformity and independence criteria are adequate for proper evaluation of Middle Square, Multiplicative Congruential and Linear Congruential algorithms which may be considered for engineering applications. Furthermore, these tests being general in nature are applicable in the evaluation of any random number algorithm.

The algorithm defined by equation (1) was evaluated for several sample sizes and seed values. Based on the results presented in Appendix C, the algorithm defined by equation (1) is adequate for implementation in engineering concepts requiring random number generation. Specifically in the case of the DRFG discussed in the introduction, the algorithm described by equation (1) is ideally suited for implementation. However, other algorithms can be tested for the uniformity and independence criteria by inserting the algorithm in RNGEP and evaluating it for several seed values and sample sizes to assure that goodness criteria are fulfilled independent of seed values or sample sizes.
APPENDIX A

STATISTICAL TABLES
### TABLE I. KOLMOGOROV-SMIRNOV ACCEPTANCE LIMITS

<table>
<thead>
<tr>
<th>Sample size (n)</th>
<th>.20</th>
<th>.15</th>
<th>.10</th>
<th>.05</th>
<th>.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.900</td>
<td>.925</td>
<td>.950</td>
<td>.975</td>
<td>.995</td>
</tr>
<tr>
<td>2</td>
<td>.684</td>
<td>.726</td>
<td>.776</td>
<td>.842</td>
<td>.929</td>
</tr>
<tr>
<td>3</td>
<td>.565</td>
<td>.597</td>
<td>.642</td>
<td>.708</td>
<td>.829</td>
</tr>
<tr>
<td>4</td>
<td>.494</td>
<td>.525</td>
<td>.564</td>
<td>.624</td>
<td>.734</td>
</tr>
<tr>
<td>5</td>
<td>.446</td>
<td>.474</td>
<td>.510</td>
<td>.563</td>
<td>.669</td>
</tr>
<tr>
<td>6</td>
<td>.410</td>
<td>.436</td>
<td>.470</td>
<td>.521</td>
<td>.618</td>
</tr>
<tr>
<td>7</td>
<td>.381</td>
<td>.405</td>
<td>.438</td>
<td>.486</td>
<td>.577</td>
</tr>
<tr>
<td>8</td>
<td>.358</td>
<td>.381</td>
<td>.411</td>
<td>.457</td>
<td>.543</td>
</tr>
<tr>
<td>9</td>
<td>.339</td>
<td>.360</td>
<td>.388</td>
<td>.432</td>
<td>.514</td>
</tr>
<tr>
<td>10</td>
<td>.322</td>
<td>.342</td>
<td>.368</td>
<td>.409</td>
<td>.486</td>
</tr>
<tr>
<td>11</td>
<td>.307</td>
<td>.326</td>
<td>.352</td>
<td>.391</td>
<td>.468</td>
</tr>
<tr>
<td>12</td>
<td>.295</td>
<td>.313</td>
<td>.338</td>
<td>.375</td>
<td>.450</td>
</tr>
<tr>
<td>13</td>
<td>.284</td>
<td>.302</td>
<td>.325</td>
<td>.361</td>
<td>.433</td>
</tr>
<tr>
<td>14</td>
<td>.274</td>
<td>.292</td>
<td>.314</td>
<td>.349</td>
<td>.418</td>
</tr>
<tr>
<td>15</td>
<td>.266</td>
<td>.283</td>
<td>.304</td>
<td>.338</td>
<td>.404</td>
</tr>
<tr>
<td>16</td>
<td>.258</td>
<td>.274</td>
<td>.295</td>
<td>.328</td>
<td>.391</td>
</tr>
<tr>
<td>17</td>
<td>.250</td>
<td>.266</td>
<td>.286</td>
<td>.318</td>
<td>.380</td>
</tr>
<tr>
<td>18</td>
<td>.244</td>
<td>.259</td>
<td>.278</td>
<td>.309</td>
<td>.370</td>
</tr>
<tr>
<td>19</td>
<td>.237</td>
<td>.252</td>
<td>.272</td>
<td>.301</td>
<td>.361</td>
</tr>
<tr>
<td>20</td>
<td>.231</td>
<td>.246</td>
<td>.264</td>
<td>.294</td>
<td>.352</td>
</tr>
<tr>
<td>25</td>
<td>.210</td>
<td>.220</td>
<td>.240</td>
<td>.264</td>
<td>.320</td>
</tr>
<tr>
<td>30</td>
<td>.190</td>
<td>.200</td>
<td>.220</td>
<td>.242</td>
<td>.290</td>
</tr>
<tr>
<td>50</td>
<td>.190</td>
<td>.230</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>.170</td>
<td>.210</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>.160</td>
<td>.190</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>.150</td>
<td>.180</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Asymptotic Formula</th>
<th>( \sqrt{n} )</th>
<th>( \sqrt{n'} )</th>
<th>( \sqrt{n''} )</th>
<th>( \sqrt{n} )</th>
<th>( \sqrt{n'} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.07</td>
<td>1.14</td>
<td>1.22</td>
<td>1.36</td>
<td>1.63</td>
<td></td>
</tr>
</tbody>
</table>
TABLE II. CHI-SQUARE ACCEPTANCE LIMITS

<table>
<thead>
<tr>
<th>Degrees of Freedom</th>
<th>Confidence Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.010</td>
</tr>
<tr>
<td>1</td>
<td>.000</td>
</tr>
<tr>
<td>2</td>
<td>.020</td>
</tr>
<tr>
<td>3</td>
<td>.115</td>
</tr>
<tr>
<td>4</td>
<td>.297</td>
</tr>
<tr>
<td>5</td>
<td>.554</td>
</tr>
<tr>
<td>6</td>
<td>.872</td>
</tr>
<tr>
<td>7</td>
<td>1.24</td>
</tr>
<tr>
<td>8</td>
<td>1.65</td>
</tr>
<tr>
<td>9</td>
<td>2.09</td>
</tr>
<tr>
<td>10</td>
<td>2.56</td>
</tr>
<tr>
<td>11</td>
<td>3.05</td>
</tr>
<tr>
<td>12</td>
<td>3.57</td>
</tr>
<tr>
<td>13</td>
<td>4.11</td>
</tr>
<tr>
<td>14</td>
<td>4.66</td>
</tr>
<tr>
<td>15</td>
<td>5.23</td>
</tr>
<tr>
<td>16</td>
<td>5.81</td>
</tr>
<tr>
<td>17</td>
<td>6.41</td>
</tr>
<tr>
<td>18</td>
<td>7.01</td>
</tr>
<tr>
<td>19</td>
<td>7.63</td>
</tr>
<tr>
<td>20</td>
<td>8.26</td>
</tr>
<tr>
<td>21</td>
<td>8.90</td>
</tr>
<tr>
<td>22</td>
<td>9.54</td>
</tr>
<tr>
<td>23</td>
<td>10.2</td>
</tr>
<tr>
<td>24</td>
<td>10.9</td>
</tr>
<tr>
<td>25</td>
<td>11.5</td>
</tr>
<tr>
<td>26</td>
<td>12.2</td>
</tr>
<tr>
<td>27</td>
<td>12.9</td>
</tr>
<tr>
<td>28</td>
<td>13.6</td>
</tr>
<tr>
<td>29</td>
<td>14.3</td>
</tr>
<tr>
<td>30</td>
<td>15.0</td>
</tr>
<tr>
<td>40</td>
<td>22.1</td>
</tr>
<tr>
<td>50</td>
<td>29.7</td>
</tr>
<tr>
<td>60</td>
<td>37.5</td>
</tr>
</tbody>
</table>

For J degrees of freedom greater than 60, use \( X^2 = 0.5(Z_p + (2J - 1))^2 \)

where, \( Z_p \) is the corresponding constant of the standard normal distribution (given in Table III).


<table>
<thead>
<tr>
<th>Confidence Interval</th>
<th>$Z_p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.800</td>
<td>0.8416</td>
</tr>
<tr>
<td>.850</td>
<td>1.0364</td>
</tr>
<tr>
<td>.900</td>
<td>1.2816</td>
</tr>
<tr>
<td>.950</td>
<td>1.6449</td>
</tr>
<tr>
<td>.960</td>
<td>1.7507</td>
</tr>
<tr>
<td>.970</td>
<td>1.8808</td>
</tr>
<tr>
<td>.980</td>
<td>2.0537</td>
</tr>
<tr>
<td>.990</td>
<td>2.3263</td>
</tr>
<tr>
<td>.995</td>
<td>2.5758</td>
</tr>
<tr>
<td>.999</td>
<td>3.0902</td>
</tr>
</tbody>
</table>
APPENDIX B

RANDOM NUMBER GENERATOR EVALUATION PROGRAM LISTING
**RANDOM NUMBER GENERATOR EVALUATION PROGRAM**

(RNGEP)

**RNGEP** is an interactive evaluator of pseudo random number algorithms.

**PROGRAM INPUT:**

MODULUS, SAMPLE SIZE, SEED, SIG LEVEL, PRINT OPTION, STOP FLAG

**PROGRAM OUTPUT:**

SAMPLE MEAN
SAMPLE VARIANCE
SAMPLE STANDARD DEVIATION
CHI-SQUARE AND K-S TEST RESULTS

**ARGUMENT AND FUNCTION ARRAYS**

**ARGUMENTS**

DATA ARGTAB/.01, .05, .10, .15, .20,
* 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
* 25., 30., 35., 40., 50., 60., 70., 80./
**DATA VALKS**

<table>
<thead>
<tr>
<th>SIGNIFICANCE LEVEL [HORIZONTAL]</th>
<th>SAMPLE SIZE [VERTICAL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>.01</td>
<td>.05</td>
</tr>
<tr>
<td>* .995, .975, .950, .925, .900,</td>
<td></td>
</tr>
<tr>
<td>* .929, .842, .776, .726, .684,</td>
<td></td>
</tr>
<tr>
<td>* .829, .708, .642, .597, .565,</td>
<td></td>
</tr>
<tr>
<td>* .734, .624, .564, .525, .494,</td>
<td></td>
</tr>
<tr>
<td>* .669, .563, .510, .474, .446,</td>
<td></td>
</tr>
<tr>
<td>* .618, .521, .470, .436, .410,</td>
<td></td>
</tr>
<tr>
<td>* .577, .486, .438, .405, .381,</td>
<td></td>
</tr>
<tr>
<td>* .543, .457, .411, .381, .358,</td>
<td></td>
</tr>
<tr>
<td>* .514, .432, .388, .360, .339,</td>
<td></td>
</tr>
<tr>
<td>* .486, .409, .368, .342, .322,</td>
<td></td>
</tr>
<tr>
<td>* .468, .391, .352, .325, .307,</td>
<td></td>
</tr>
<tr>
<td>* .450, .375, .338, .313, .295,</td>
<td></td>
</tr>
<tr>
<td>* .433, .361, .325, .302, .284,</td>
<td></td>
</tr>
<tr>
<td>* .418, .349, .314, .292, .274,</td>
<td></td>
</tr>
<tr>
<td>* .404, .338, .304, .283, .266,</td>
<td></td>
</tr>
<tr>
<td>* .391, .328, .295, .274, .258,</td>
<td></td>
</tr>
<tr>
<td>* .380, .318, .286, .266, .250,</td>
<td></td>
</tr>
<tr>
<td>* .370, .309, .278, .259, .244,</td>
<td></td>
</tr>
<tr>
<td>* .361, .301, .272, .252, .237,</td>
<td></td>
</tr>
<tr>
<td>* .352, .294, .264, .246, .231,</td>
<td></td>
</tr>
<tr>
<td>* .320, .264, .240, .220, .210,</td>
<td></td>
</tr>
<tr>
<td>* .290, .242, .220, .200, .190,</td>
<td></td>
</tr>
<tr>
<td>* .270, .230, .210, .190, .180,</td>
<td></td>
</tr>
<tr>
<td>* .250, .210, .000, .000, .000,</td>
<td></td>
</tr>
<tr>
<td>* .230, .190, .000, .000, .000,</td>
<td></td>
</tr>
<tr>
<td>* .210, .170, .000, .000, .000,</td>
<td></td>
</tr>
<tr>
<td>* .190, .160, .000, .000, .000,</td>
<td></td>
</tr>
<tr>
<td>* .180, .150, .000, .000, .000/</td>
<td></td>
</tr>
</tbody>
</table>

**** CHI-SQUARE TABLES

**STANDARD NORMAL DISTRIBUTION**

- DATA ARGTB2 / .80, .85, .90, .95, .96, .97, .98, .99, .995, .999/
- DATA VZNORM / .8416, 1.0364, 1.2816, 1.6449, 1.7507, 1.8808, 2.0537, 2.3263, 2.5758, 3.0902/
### CHI-SQUARE DISTRIBUTION

DATA NARG1 /5, 33/

DATA ARGTB1 / .80, .90, .95, .975, .990,
* 1., 2., 3., 4., 5., 6., 7., 8., 9., 10.,
* 40., 50., 60. /

DATA VALCHI /

<table>
<thead>
<tr>
<th>CONFIDENCE PERCENTILE [HORIZONTAL]</th>
<th>DEGREES OF FREEDOM [VERTICAL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.80</td>
<td>0.90</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>* 1.64, 2.71, 3.84, 5.02, 6.63,</td>
<td>* 8.56, 10.60, 12.60, 14.40, 16.80,</td>
</tr>
<tr>
<td>* 3.22, 4.61, 5.99, 7.38, 9.21,</td>
<td>* 9.80, 12.00, 14.10, 16.00, 18.50,</td>
</tr>
<tr>
<td>* 4.64, 6.25, 7.81, 9.35, 11.30,</td>
<td>* 11.00, 13.40, 15.50, 17.50, 20.10,</td>
</tr>
<tr>
<td>* 7.29, 9.24, 11.10, 12.80, 15.10,</td>
<td>* 13.40, 16.00, 18.30, 20.50, 23.20,</td>
</tr>
<tr>
<td>* 14.60, 17.30, 19.70, 21.90, 24.70,</td>
<td>* 15.80, 18.50, 21.00, 23.30, 26.20,</td>
</tr>
<tr>
<td>* 17.00, 19.80, 22.40, 24.70, 27.70,</td>
<td>* 18.20, 21.10, 23.70, 26.10, 29.10,</td>
</tr>
<tr>
<td>* 19.30, 22.30, 25.00, 27.50, 30.60,</td>
<td>* 20.50, 23.50, 26.30, 28.80, 32.00,</td>
</tr>
<tr>
<td>* 21.60, 24.80, 27.60, 30.20, 33.40,</td>
<td>* 22.80, 26.00, 28.90, 31.50, 34.80,</td>
</tr>
<tr>
<td>* 23.90, 27.20, 30.10, 32.90, 36.20,</td>
<td>* 25.00, 28.40, 31.40, 34.20, 37.60,</td>
</tr>
<tr>
<td>* 26.20, 29.60, 32.70, 35.50, 38.90,</td>
<td>* 27.30, 30.80, 33.90, 36.80, 40.30,</td>
</tr>
<tr>
<td>* 28.40, 32.00, 35.20, 38.10, 41.60,</td>
<td>* 29.60, 33.20, 36.40, 39.40, 43.00,</td>
</tr>
<tr>
<td>* 30.70, 34.40, 37.70, 40.60, 44.30,</td>
<td>* 31.80, 35.60, 38.90, 41.90, 45.60,</td>
</tr>
<tr>
<td>* 32.90, 36.70, 40.10, 43.20, 47.00,</td>
<td>* 34.00, 37.90, 41.30, 44.50, 48.30,</td>
</tr>
<tr>
<td>* 35.10, 39.10, 42.60, 45.70, 49.60,</td>
<td>* 36.20, 40.30, 43.80, 47.00, 50.90,</td>
</tr>
</tbody>
</table>
C
* 47.30, 51.80, 55.80, 59.30, 63.70,
* 58.20, 63.20, 67.50, 71.40, 76.20,
* 69.00, 74.40, 79.10, 83.30, 88.40/
C
C** INITIALIZATION OF VARIABLES
C
STOPFL = 0
1 DO 5 I = 1,5000
   ARG(I) = 0.
   CARG(I) = 0.
   CREP(I) = 0.
   ARGREP(I) = 0.
   ARGGRAN(I) = 0.
   RK(I) = 0.
   POWER(I) = 0.
   POWPLT(I) = 0.
   AKSPT1(I) = 0.
   AKSPT2(I) = 0.
5 POWINT(I) = 0.
   PI  = 3.141592654
   NSAM = 0
   SUMN = 0.
   SUMN2 = 0.
   AMEAN = 0.
   VAR  = 0.
   SDEF = 0.
   VKS  = 0.
   THCDF = 0.
   AMXDIF1 = 0.
   AMXDIF2 = 0.
   CHI  = 0.
   CHISQ = 0.
   ZNORM = 0.
   X1   = 0.
   X2   = 0.
   X3   = 0.
   X4   = 0.
   X5   = 0.
   NPRT = 0
   POWERL = 0.
   POWERI = 0.
C
C** READ INPUT DATA INTERACTIVELY
   IF( STOPFL .NE. 1 ) WRITE (7,100)
998 WRITE (7,101)
   WRITE (7,*)'?'
   READ (5,* ) MODU, NSAMP, SEED, SIGLEV, PRINTOP, STOPFL
   NSEED = SEED
N11 = SEED
IF( NSAMP .GT. 5000 ) WRITE(7,*)
' MAX SAMPLE SIZE OF 5000 EXCEEDED!!!'
IF( NSAMP .GT. 5000 ) GO TO 998
IF( SIGLEV .LT. .01 )
* WRITE (6,*),' MIN SIGNIFICANCE LEVEL OF .01 EXCEEDED:
* RE-ENTER DATA'
IF( SIGLEV .GT. .20 )
* WRITE (6,*),' MAX SIGNIFICANCE LEVEL OF .20 EXCEEDED:
* RE-ENTER DATA'
IF( SIGLEV .LT. .01 .OR. SIGLEV .GT. 20 ) GO TO 998

C
C** BEGIN SAMPLING RANDOM NO. ALGORITHM
C
DO 10 I = 1,NSAMP
REPFL = 0.
C
C** RANDOM NO. ALGORITHM
C
NX1 = 129*N11 + 1
C
C** CHECK FOR HARDWARE OVERFLOW
C
N11 = MOD(NX1,MODU)
NSAM = NSAM + 1
NPRT = NPRT + 1
IF( NSAMP .LE. 1000 .OR. NPRT .LT. 500 ) GO TO 652
WRITE(6,*),' SAMPLES HAVE BEEN PROCESSED'
NPRT = 0
C
C** DIVIDE BY MODULUS
C
652 AX11 = FLOAT(N11)/FLOAT(MODU)
C
C** STORE RANDOM NUMBERS FOR LISTING
ARGRAN(NSAM) = AX11
C
C** SORT RANDOM NUMBERS IN ORDER OF MAGNITUDE
C
DO 900 II=1,NSAM
IF( AX11 .LE. ARG(II) ) GO TO 925
900 CONTINUE
ARG(NSAM) = AX11
GO TO 10
925 DO 950 J=1,NSAM-II
JJ = NSAM - J
950 ARG(JJ+1) = ARG(JJ)
ARG(II) = AX11
10 CONTINUE

C** COMPUTE QUANTITY OF REPEATED NUMBERS
C
J = 1
DO 510 I=1,NSAMP
   IF(ARG(I) .NE. ARG(I+1)) GO TO 500
C
C** REPEATED ARGUMENT COUNT
   CARG(J) = CARG(J) + 1.
   GO TO 510
500 CARG(J) = CARG(J) + 1.
C
C** STORE REPEATED ARGUMENTS
   ARGREP(J) = ARG(I)
J = J + 1
510 CONTINUE
NPOINT = J - 1
DO 700 I = 1,NPOINT
   DO 650 JJ = 1,1
   C** COMPUTE CUMULATIVE PROBABILITY DISTRIBUTION
   CREP(I) = CREP(I) + CARG(JJ)/NSAMP
700 CONTINUE
C
C** SET REMAINDER OF CDF ARRAY TO 1.0
   DO 750 I = NPOINT+1,NSAMP
   CREP(I) = 1.
750 CONTINUE
C
C** REARRANGE CDF PLOT VARIABLES
   NPTPL1 = 2*NPOINT + 2
   A(1) = 0.
   DO 800 I=2,NPOINT+1
   II = 2*(I-1)
      A(II ) = ARGREP(I-1)
      A(II+1) = ARGREP(I-1)
800 CONTINUE
   A(NPTPL1) = 1.0
   B(1) = 0.
   B(2) = 0.
   DO 850 I=1,NPOINT
      II = 2*(I-1) + 3
      B(II ) = CREP(I)
      B(II+1) = CREP(I)
850 CONTINUE
   DO 878 I = 1,NPTPL1
      GRAPH1(I,1) = A(I)
      GRAPH1(I,2) = B(I)
878 CONTINUE
C** OUTPUT STATEMENT
WRITE(6,103)
WRITE(6,102) NSAMP,MODU,NSEED
C
C** COMPUTE SUM OF SAMPLED NUMBERS (STATISTICAL)
   DO 212 I = 1,NSAMP
   212 SUMN = SUMN + ARG(I)
C
C** COMPUTE SQUARED SUM OF SAMPLED NUMBERS (STATISTICAL)
   DO 211 I = 1,NSAMP
   211 SUMN2 = SUMN2 + ARG(I)*ARG(I)
C
C** COMPUTE MEAN VALUE
C
   AMEAN = SUMN/NSAMP
C
C** COMPUTE VARIANCE
C
   VAR = ( SUMN2 - SUMN*SUMN/NSAMP )/(NSAMP-1)
C
C** COMPUTE STANDARD DEVIATION
C
   SDEV = SQRT( VAR )
C
C** OUTPUT STATEMENT
WRITE(6,140) AMEAN
WRITE(6,141) VAR
WRITE(6,142) SDEV
C
C** KOLOMOGOROV-SMIRNOV TEST
C
   RNSAMP = NSAMP
   PCT = 100.*( 1. - SIGLEV )
   DO 851 I = 1,NSAMP
      THCDF = THCDF + 1./NSAMP
      AMXDIF1 = AMAX1(ABS(THCDF-ARG(I)),ABS(THCDF-CREP(I)))
      AMXDIF2 = AMAX1(AMXDIF2,AMXDIF1)
   851 CONTINUE
   IF ( NSAMP .GT. 80 ) GO TO 852
   IF ( NSAMP .GT. 35 .AND. SIGLEV .GT. .05 ) GO TO 852
   CALL TABL2(SIGLEV,RNSAMP,ARGTAB,VALKS,NARG,O,VKS)
   GO TO 853
   852 IF( SIGLEV .EQ. .01 ) VKS = 1.63/SQRT(RNSAMP)
   IF( SIGLEV .EQ. .05 ) VKS = 1.36/SQRT(RNSAMP)
   IF( SIGLEV .EQ. .10 ) VKS = 1.22/SQRT(RNSAMP)
   IF( SIGLEV .EQ. .15 ) VKS = 1.14/SQRT(RNSAMP)
   IF( SIGLEV .EQ. .20 ) VKS = 1.07/SQRT(RNSAMP)
C  
853 IF ( AMXDIF2 .GT. VKS ) WRITE (6,109) PCT  
     IF ( AMXDIF2 .LE. VKS ) WRITE (6,110) PCT  
     WRITE(6,108) VKS,AMXDIF2  
C  
C** CHI-SQUARE TEST  
   DOF = 4.0  
   DO 854 I = 1,NSAMP  
      IF( ARG(I) .GE. 0.0 .AND. ARG(I) .LT. 0.2 ) X1 = X1 + 1  
      IF( ARG(I) .GE. 0.2 .AND. ARG(I) .LT. 0.4 ) X2 = X2 + 1  
      IF( ARG(I) .GE. 0.4 .AND. ARG(I) .LT. 0.6 ) X3 = X3 + 1  
      IF( ARG(I) .GE. 0.6 .AND. ARG(I) .LT. 0.8 ) X4 = X4 + 1  
      IF( ARG(I) .GE. 0.8 .AND. ARG(I) .LE. 1.0 ) X5 = X5 + 1  
854 CONTINUE  
   PARTSZ = RNSAMP/5.0  
   CHISQ1 = ( X1 - PARTSZ )**2.0  
   CHISQ2 = ( X2 - PARTSZ )**2.0  
   CHISQ3 = ( X3 - PARTSZ )**2.0  
   CHISQ4 = ( X4 - PARTSZ )**2.0  
   CHISQ5 = ( X5 - PARTSZ )**2.0  
   CHISQ = ( CHISQ1 + CHISQ2 + CHISQ3 + CHISQ4 + CHISQ5 )/PARTSZ  
   PCTILE = 1. - SIGLEV  
   IF ( DOF .GT. 60. ) GO TO 855  
   CALL TABL2 ( PCTILE,DOF,ARGTB1,VALCH1,NARG1,O,CHI )  
   GO TO 856  
855 CALL TABL1 ( PCTILE, ARGTB2, VZNORM, 10, 0, ZNORM )  
   CHI = 0.5*( ZNORM + SQRT(2.*DOF - 1.))**2.0  
856 IF ( CHISQ .GT. CHI ) WRITE (6,111) PCT  
   IF ( CHISQ .LE. CHI ) WRITE (6,112) PCT  
   WRITE(6,113) CHI,CHISQ  
C  
C** WHITENESS TEST  
C  MAXIMUM VALUE OF FREQUENCY(FREQK) IS 0.5  
KMAX = NSAMP/2  
   DO 870 I = 1, KMAX+1  
      K = I-1  
      RK(I) = K  
      FREQK = RK(I)/RNSAMP  
      POWERL = 0.  
      POWERI = 0.  
      DO 871 J = 1,NSAMP  
         T = -NSAMP/2 - 1 + J  
         POWERL = POWERL + ARGRAN(J)*COS(2.*PI*FREQK*T)  
         POWERI = POWERI + ARGRAN(J)*SIN(2.*PI*FREQK*T)  
871 CONTINUE
C CALCULATE SAMPLE POWER SPECTRUM(PZZ) VS. K FREQUENCY
C POWER(1) --> K=0,POWER(2) --> K=1,...,POWER(N/2+1) --> K=N/2,
C MULTIPLY SAMPLE SPECTRUM BY TWO TO ACCOUNT FOR NEGATIVE SIDE OF
C SPECTRUM
     POWER(I) = 2.0*(POWERL*POWERL + POWERI*POWERI)/RNSAMP
870 CONTINUE
C
C COMPUTE SPECTRUM INTEGRAL
     POWINT(I) = 0.
     DO 873 I = 2,KMAX+1
     POWINT(I) = POWINT(I-1) + POWER(I)
873 CONTINUE
     RKMAX = KMAX
     IF ( RKMAX .GT. 80 ) GO TO 875
     IF ( RKMAX .GT. 35 .AND. SIGLEV .GT. .05 ) GO TO 875
     CALL TABL2(SIGLEV,RKMAX,ARGTAB,VALKS,NARG,O,VKS)
     GO TO 876
875 IF( SIGLEV .EQ. .01 ) VKS = 1.63/SQRT(RKMAX)
     IF( SIGLEV .EQ. .05 ) VKS = 1.36/SQRT(RKMAX)
     IF( SIGLEV .EQ. .10 ) VKS = 1.22/SQRT(RKMAX)
     IF( SIGLEV .EQ. .15 ) VKS = 1.14/SQRT(RKMAX)
     IF( SIGLEV .EQ. .20 ) VKS = 1.07/SQRT(RKMAX)
876 DO 874 I = 1,KMAX+1
     POWPLT(I) = POWINT(I)/VAR/RNSAMP
C
C COMPUTE K-S CONFIDENCE LIMITS
     AKSPT1(I) = 2.*RK(I)/RNSAMP + VKS
     AKSPT2(I) = 2.*RK(I)/RNSAMP - VKS
C
C PLOT INTEGRATED SPECTRUM AND CONFIDENCE LIMITS
     GRAPH2(I,1) = RK(I)
     GRAPH2(I,2) = POWPLT(I)
     GRAPH2(I,3) = AKSPT1(I)
     GRAPH2(I,4) = AKSPT2(I)
874 CONTINUE
C
C** PRINT RESULTS
C
     IF( PRINTOP .LE. 0. ) GO TO 99
     WRITE(6,150)
     WRITE(6,160)
     DO 90 K = 1,NSAMP,5
     90 WRITE(6,170) ARGRA(K),ARGRA(K+1),ARGRA(K+2),
          * ARGRA(K+3),ARGRA(K+4)
     WRITE(6,151)
     WRITE(6,161)
DO 91 K = 1,NSAMP,5
91 WRITE(6,170) ARG(K),ARG(K+1),ARG(K+2),ARG(K+3),ARG(K+4)
99 IF( STOPFL .EQ. 1.0 ) NEXRUN = 1
   IF( STOPFL .EQ. 0.0 ) NEXRUN = 2
   NOPL1 = 2
   NOPL2 = 4
   NPTPL2 = KMAX+1
   WRITE(15) NOPL1,NPTPL1,NEXRUN
   WRITE(15) ((VLABL1(I,J),I=1,4),J=1,NOPL1)
   WRITE(15) ((GRAPH1(I,J),I = 1,NPTPL1),J=1,NOPL1)
   WRITE(16) NOPL2,NPTPL2,NEXRUN
   WRITE(16) ((VLABL2(I,J),I=1,4),J=1,NOPL2)
   WRITE(16) ((GRAPH2(I,J),I = 1,NPTPL2),J=1,NOPL2)
   IF( STOPFL .LE. 0. ) GO TO 1
   CALL EXIT
C** OUTPUT FORMATS
C
100 FORMAT(,' WELCOME TO RNGEP')
101 FORMAT(,' PLEASE ENTER: ',/,' MODULO, SAMPLE SIZE, SEED,
   * SIGNIF LEV, PRINT(0=NO:1=YES), STOP(0=NO:1=YES)')
102 FORMAT(,'NUMBER OF SAMPLES = ',I4,
   * MODULUS = ',I6,
   * SEED = ',I6)
103 FORMAT(,'******************** RNGEP STATISTICAL RESULTS
   ******************')
106 FORMAT(,' K-S TABLE VALUE = ',F5.3,
   * COMPUTED VALUE = ',F7.4)
107 FORMAT(,' ALGORITHM FAILS K-S TEST FOR ',F4.1,
   * PCT. CONFIDENCE INTERVAL!!!')
109 FORMAT(,' ALGORITHM PASSES K-S TEST FOR ',F4.1,
   * PCT. CONFIDENCE INTERVAL!!!')
110 FORMAT(,' ALGORITHM FAILS CHI-SQUARE TEST FOR ',F4.1,
   * PCT. CONFIDENCE INTERVAL!!!')
112 FORMAT(,' ALGORITHM PASSES CHI-SQUARE TEST FOR ',F4.1,
   * PCT. CONFIDENCE INTERVAL!!!')
113 FORMAT(,' CHI-SQUARE TABLE VALUE = ',F8.3,
   * COMPUTED VALUE = ',F8.3)
114 FORMAT(,' CHI-SQUARE TABLE VALUE = ',F8.3,
   * COMPUTED VALUE = ',F12.3)
115 FORMAT(,' SAMPLES OUT OF ',I4,
   * TOTAL COMPLETED')
116 FORMAT(,'MEAN = ',1F6.4)
117 FORMAT(,'VARIANCE = ',1F10.5)
118 FORMAT(,'STANDARD DEV = ',1F10.5)
120 FORMAT(,'SEQUENTIAL RANDOM NUMBERS')
121 FORMAT(,'SORTED RANDOM NUMBERS')
122 FORMAT(,'-------------------------')
123 FORMAT(,'---------------------')
170 FORMAT(,'12X,5(2X,F7.4))
END
**SINGLE TABLE LOOK-UP SUBROUTINE**

```fortran
SUBROUTINE TBL1 (A, ARG, TBL, NARG, KEEP, FN)
    DIMENSION ARG(1), TBL(1)
    IF(KEEP.GE.1) GO TO 90
    I=1
    10 I=I+1
    IF (A.LE.ARG(I)) GO TO 20
    IF (I.LT.NARG) GO TO 10
    20 J1=I-1
    PCT1=(A-ARG(J1))/(ARG(I)-ARG(J1))
    90 FN=TBL(J1)+PCT1*(TBL(I)-TBL(J1))
    RETURN
END
```

**DOUBLE TABLE LOOK-UP SUBROUTINE**

```fortran
SUBROUTINE TBL2 (A, B, ARG, TBL, NARG, KEEP, FN)
    DIMENSION NARG(2), ARG(1), TBL(1)
    IF(KEEP.GE.1) GO TO 90
    N1=NARG(1)
    I=1
    10 I=I+1
    IF (A.LE.ARG(I)) GO TO 20
    IF (I.LT.N1) GO TO 10
    20 J1=I-1
    PCT1=(A-ARG(J1))/(ARG(I)-ARG(J1))
    I=N1+1
    IO=I+1
    IARG=N1+NARG(2)
    30 I=I+1
    IF (B.LE.ARG(I)) GO TO 40
    IF (I.LT.IARG) GO TO 30
    40 K1=(I-IO)*N1
    PCT2=(B-ARG(I-1))/(ARG(I)-ARG(I-1))
    I1=J1+K1
    I2=I1+1
    90 A1=TBL(I1)+PCT2*(TBL(I1+N1)-TBL(I1))
    A2=TBL(I2)+PCT2*(TBL(I2+N1)-TBL(I2))
    FN=A1+PCT1*(A2-A1)
    RETURN
END
```
APPENDIX C

RANDOM NUMBER ALGORITHM TEST RESULTS
Random Number Generator Evaluation Program Case 1 Results

************************ RNGEP STATISTICAL RESULTS ************************

NUMBER OF SAMPLES = 50  MODULUS = 65536  SEED = 100

MEAN = 0.4759  VARIANCE = 0.08245  STANDARD DEV = 0.28715

ALGORITHM PASSES K-S TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
K-S TABLE VALUE = 0.173  COMPUTED VALUE = 0.0833

ALGORITHM PASSES CHI-SQUARE TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
CHI-SQUARE TABLE VALUE = 7.780  COMPUTED VALUE = 1.200

SEQUENTIAL RANDOM NUMBERS

0.1969  0.3941  0.8434  0.7946  0.4977
0.2028  0.1598  0.6188  0.8298  0.0427
0.5076  0.4744  0.1931  0.9138  0.8865
0.3611  0.5877  0.8163  0.2967  0.2792
0.0136  0.7499  0.7382  0.2285  0.4707
0.7148  0.2109  0.2090  0.9590  0.7110
0.7149  0.2208  0.4786  0.7384  0.2501
0.2638  0.0294  0.7970  0.8166  0.3381
0.6115  0.8869  0.4143  0.4436  0.2249
0.0081  0.0433  0.5804  0.8695  0.1605
Random Number Generator Evaluation Program Case 1 Results
(Cont’d)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0081</td>
<td>0.0136</td>
<td>0.0294</td>
<td>0.0427</td>
<td>0.0433</td>
</tr>
<tr>
<td>0.1598</td>
<td>0.1605</td>
<td>0.1931</td>
<td>0.1969</td>
<td>0.2028</td>
</tr>
<tr>
<td>0.2090</td>
<td>0.2109</td>
<td>0.2208</td>
<td>0.2249</td>
<td>0.2285</td>
</tr>
<tr>
<td>0.2501</td>
<td>0.2638</td>
<td>0.2792</td>
<td>0.2967</td>
<td>0.3381</td>
</tr>
<tr>
<td>0.3611</td>
<td>0.3941</td>
<td>0.4143</td>
<td>0.4436</td>
<td>0.4707</td>
</tr>
<tr>
<td>0.4744</td>
<td>0.4786</td>
<td>0.4977</td>
<td>0.5076</td>
<td>0.5804</td>
</tr>
<tr>
<td>0.5877</td>
<td>0.6115</td>
<td>0.6188</td>
<td>0.7110</td>
<td>0.7148</td>
</tr>
<tr>
<td>0.7149</td>
<td>0.7382</td>
<td>0.7384</td>
<td>0.7499</td>
<td>0.7946</td>
</tr>
<tr>
<td>0.7970</td>
<td>0.8163</td>
<td>0.8166</td>
<td>0.8298</td>
<td>0.8434</td>
</tr>
<tr>
<td>0.8695</td>
<td>0.8865</td>
<td>0.8869</td>
<td>0.9138</td>
<td>0.9590</td>
</tr>
</tbody>
</table>
DISCRETE CUMULATIVE DISTRIBUTION
(50 SAMPLES; MODULO 65536; SEED 100)

Figure 3. Case 1 Discrete Cumulative Distribution.
WHITE NOISE KOLMOGOROV-SMIRNOV TEST
(50 SAMPLES; MODULO 65536; SEED 100)

Figure 4. Case 1 White Noise Test.
Random Number Generator Evaluation Program Case 2 Results

************************* RNGEP STATISTICAL RESULTS *************************

NUMBER OF SAMPLES = 100      MODULUS = 65536      SEED = 200

MEAN = 0.5278
VARIANCE = 0.09272
STANDARD DEV = 0.30451

ALGORITHM PASSES K-S TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
K-S TABLE VALUE = 0.122      COMPUTED VALUE = 0.0867

ALGORITHM PASSES CHI-SQUARE TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
CHI-SQUARE TABLE VALUE = 7.780      COMPUTED VALUE = 1.800
Random Number Generator Evaluation Program Case 2 Results

(Cont'd)

<table>
<thead>
<tr>
<th>SEQUENTIAL RANDOM NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3937 0.7863 0.4308 0.5773 0.4758</td>
</tr>
<tr>
<td>0.3762 0.5285 0.1829 0.5891 0.9973</td>
</tr>
<tr>
<td>0.6575 0.8196 0.7337 0.6497 0.8177</td>
</tr>
<tr>
<td>0.4877 0.9096 0.3334 0.0092 0.1870</td>
</tr>
<tr>
<td>0.1167 0.0483 0.2319 0.9175 0.3550</td>
</tr>
<tr>
<td>0.7945 0.4859 0.6793 0.6246 0.5719</td>
</tr>
<tr>
<td>0.7711 0.4723 0.9254 0.3805 0.0876</td>
</tr>
<tr>
<td>0.2966 0.2575 0.2204 0.4353 0.1521</td>
</tr>
<tr>
<td>0.6209 0.0916 0.8143 0.0389 0.0155</td>
</tr>
<tr>
<td>0.9940 0.2245 0.9569 0.4413 0.9276</td>
</tr>
<tr>
<td>0.6659 0.9062 0.8984 0.8925 0.1387</td>
</tr>
<tr>
<td>0.8867 0.3867 0.8887 0.6426 0.8985</td>
</tr>
<tr>
<td>0.9063 0.9161 0.1778 0.9415 0.4572</td>
</tr>
<tr>
<td>0.9748 0.7443 0.0158 0.0393 0.0647</td>
</tr>
<tr>
<td>0.3420 0.1213 0.6526 0.1858 0.9710</td>
</tr>
<tr>
<td>0.2581 0.2972 0.3382 0.6312 0.4261</td>
</tr>
<tr>
<td>0.9730 0.5219 0.3227 0.6254 0.6801</td>
</tr>
<tr>
<td>0.7368 0.0454 0.8560 0.4185 0.9829</td>
</tr>
<tr>
<td>0.7994 0.1177 0.1881 0.2603 0.5846</td>
</tr>
<tr>
<td>0.4108 0.9889 0.5690 0.4010 0.7350</td>
</tr>
</tbody>
</table>
## Random Number Generator Evaluation Program Case 2 Results

(Cont'd)

**SORTED RANDOM NUMBERS**

```
| 0.0092 | 0.0155 | 0.0158 | 0.0389 | 0.0393 |
| 0.0454 | 0.0483 | 0.0647 | 0.0876 | 0.0916 |
| 0.1167 | 0.1177 | 0.1213 | 0.1387 | 0.1521 |
| 0.1778 | 0.1829 | 0.1858 | 0.1870 | 0.1881 |
| 0.2204 | 0.2245 | 0.2319 | 0.2575 | 0.2581 |
| 0.2603 | 0.2966 | 0.2972 | 0.3227 | 0.3334 |
| 0.3382 | 0.3420 | 0.3550 | 0.3762 | 0.3805 |
| 0.3867 | 0.3937 | 0.4010 | 0.4108 | 0.4185 |
| 0.4261 | 0.4308 | 0.4353 | 0.4413 | 0.4572 |
| 0.4723 | 0.4758 | 0.4859 | 0.4877 | 0.5219 |
| 0.5285 | 0.5690 | 0.5719 | 0.5773 | 0.5846 |
| 0.5891 | 0.6209 | 0.6246 | 0.6254 | 0.6312 |
| 0.6426 | 0.6497 | 0.6526 | 0.6575 | 0.6659 |
| 0.6793 | 0.6801 | 0.7337 | 0.7350 | 0.7368 |
| 0.7443 | 0.7711 | 0.7863 | 0.7945 | 0.7994 |
| 0.8143 | 0.8177 | 0.8196 | 0.8560 | 0.8867 |
| 0.8887 | 0.8925 | 0.8984 | 0.8985 | 0.9062 |
| 0.9063 | 0.9096 | 0.9161 | 0.9175 | 0.9254 |
| 0.9276 | 0.9415 | 0.9569 | 0.9710 | 0.9730 |
| 0.9748 | 0.9829 | 0.9889 | 0.9940 | 0.9973 |
```
DISCRETE CUMULATIVE DISTRIBUTION
(100 SAMPLES; MODULO 65536; SEED 200)

Figure 5. Case 2 Discrete Cumulative Distribution.
Figure 6. Case 2 White Noise Test.
Random Number Generator Evaluation Program Case 3 Results

************************* RNGEP STATISTICAL RESULTS *************************

NUMBER OF SAMPLES = 500 MODULUS = 65536 SEED = 300

MEAN = 0.5081
VARIANCE = 0.08397
STANDARD DEV = 0.28977

ALGORITHM PASSES K-S TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
K-S TABLE VALUE = 0.055 COMputed VALUE = 0.0305

ALGORITHM PASSES CHI-SQUARE TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
CHI-SQUARE TABLE VALUE = 7.780 COMPUTED VALUE = 4.160
Figure 7. Case 3 Discrete Cumulative Distribution.
WHITE NOISE KOLMOGOROV-SMIRNOV TEST
(500 SAMPLES; MODULO 65536; SEED 300)

Figure 8. Case 3 White Noise Test.
Random Number Generator Evaluation Program Case 4 Results

************************ RNGEP STATISTICAL RESULTS ************************

NUMBER OF SAMPLES = 1000  MODULUS = 65536  SEED = 400

MEAN = 0.4984
VARIANCE = 0.08327
STANDARD DEV = 0.28856

ALGORITHM PASSES K-S TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
K-S TABLE VALUE = 0.039  COMPUTED VALUE = 0.0076

ALGORITHM PASSES CHI-SQUARE TEST FOR 90.0 PCT. CONFIDENCE INTERVAL!!!
CHI-SQUARE TABLE VALUE = 7.780  COMPUTED VALUE = 0.170
Figure 9. Case 4 Discrete Cumulative Distribution.
Figure 10. Case 4 White Noise Test.
REFERENCES


