UNIFORM RANDOM NUMBER GENERATORS

By William R. Farr
Preliminary Design Office

November 10, 1971

NASA

George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama
Methods are presented for the generation of random numbers with uniform and normal distributions. Subprogram listings of Fortran generators for the Univac 1108, SDS 930, and CDC 3200 digital computers are also included. The generators are of the mixed-multiplicative type, and the mathematical method employed is that of Marsaglia and Bray.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>THEORETICAL CONSIDERATIONS</td>
<td>1</td>
</tr>
<tr>
<td>FORTRAN CONGRUENT GENERATORS</td>
<td>4</td>
</tr>
<tr>
<td>GENERATION OF NORMAL VARIABLES</td>
<td>6</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>7</td>
</tr>
<tr>
<td>APPENDIX A: FORTRAN LISTING OF UNIVAC 1108 UNIFORM GENERATOR</td>
<td>8</td>
</tr>
<tr>
<td>APPENDIX B: FORTRAN LISTING OF CDC 3200 UNIFORM GENERATOR</td>
<td>9</td>
</tr>
<tr>
<td>APPENDIX C: FORTRAN LISTING OF SDS 930 UNIFORM GENERATOR</td>
<td>10</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>11</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>12</td>
</tr>
</tbody>
</table>
INTRODUCTION

With the increased use of simulation and Monte Carlo methods in all the various disciplines of engineering and science, the need for sequences of numbers that appear to be drawn from particular probability distributions has become increasingly more important. The production of these sequences of numbers, or so-called "pseudo-random" numbers, must be simple, fast, accurate, and, most importantly, reproducible. The purpose of this paper is to present automated procedures for the production of these pseudo-random numbers consistent with the above criteria. In automating the random number generators, emphasis was placed upon the uniform distribution.

Most computer library subprogram generators are coded in complex machine language and therefore tend to increase, rather than alleviate, any confusion existing on the generation of random variables. Presented herein are Fortran subprograms for the generation of uniform pseudo-random numbers on the unit interval \([0, 1)\). The generators are of the mixed-multiplicative congruential type, and the method is that of Marsaglia and Bray [1]. The subprograms described are for the Univac 1108, the CDC 3200, and the SDS 930 digital computers. A method for generating normal random variables from uniform variables is also included.

THEORETICAL CONSIDERATIONS

There are many methods for generating uniform random variables, each having inherent advantages and disadvantages depending upon its utilization [2]. The mixed-multiplicative congruential generator is discussed herein. Because of varied opinions concerning the appropriateness of this type of generator for certain Monte Carlo applications [3-5], the validity of these particular applications will not be addressed. This discussion will consist of the basic theory of this type of generator and how it may be used conveniently via Fortran subprograms by the method of Marsaglia and Bray [1].
The congruential method is basically an arithmetic recurrence relation involving integers in which each new number is generated from the previous number by some deterministic approach. The recurrence relation is initiated by some initial value and, at some subsequent point, will redevelop forming a closed loop. The length of this closed loop is the period of the generator and, hopefully, is nearly equal to the total integer population of the machine, which is designated by m.

The deterministic approach of the congruential generators is the relation,

\[ X_{i+1} = aX_i + c \mod (m) \quad (0 \leq X_i < m) , \]

which means that the expression \( aX_i + c \) is to be divided by \( m \) and \( X_{i+1} \) is set equal to the remainder. To illustrate this, let \( m \) (modulus) = 25, \( a \) (multiplier) = 7, and \( c = 1 \), and let \( X_0 = 3 \) be the initial value for

\[ X_1 = 7 \times 3 + 1 \mod (25) \quad X_1 = 22 , \]

\[ X_2 = 7 \times 22 + 1 \mod (25) \quad X_2 = 5 , \]

\[ X_3 = 7 \times 5 + 1 \mod (25) \quad X_3 = 11 , \text{ etc.} \]

Numbers on \([0, 1)\) can be obtained by dividing by \( m \). The method usually is called multiplicative if \( c = 0 \) and mixed if \( c \neq 0 \). The modulus \( m \) is normally taken as \( 2^n \) for an \( n \)-bit binary machine and \( 10^n \) for an \( n \)-digit decimal machine. The constants \( a \) and \( c \) are chosen to provide speed, a long period, and good statistical results [6].

A basic problem inherent in congruence method generators is that choice of \( X_0 \), \( a \), and \( c \) which will insure a maximum period. The following theorem presented by Hull and Dorbell [2] solves this problem.
Theorem: The sequence defined by the congruence relation

\[ X_{i+1} = aX_i + c \mod (m) \]

has full period \( m \), provided

1. \( c \) is relatively prime to \( m \)
2. \( a = 1 \mod (p) \) if \( p \) is a prime factor of \( m \),
3. \( a = 1 \mod (4) \) if \( 4 \) is a factor of \( m \).

Thus, if \( m \) is a power of \( 2 \), as on a binary machine, we need only to have \( c \) an odd number and \( a = 1 \mod (4) \). The proof of this theorem is included in Reference [2].

The most favorable aspect of the congruence generators is the characteristic that a sequence of random digits may be reproduced by simply starting the generator with the same initial value. In the paper by Stockmal [7], algorithms are presented that evaluate

\[ X_i = f \left( i \right) \]

or

\[ i = f^{-1} \left( X_i \right) , \]

where \( X_i \) is the \( i \)th element of a congruential random number sequence.

These algorithms can be very useful when only certain parts of a Monte Carlo simulation need to be repeated and the generation of the entire sequence of random variables is not required.

Congruence generators have been widely used, and the results have been favorable. An extensive list of references may be found in Reference 2.
Marsaglia and Bray [1] developed a method of generating uniform random variables by the congruence method utilizing a single Fortran statement. They also describe Fortran programs that mix several such generators.

Initially, consider the handling of integers by the SDS 930 digital computer. Here, Fortran integers are stored in 24 bits, and the multiplication of 2 integers produces a 24-bit integer mod, $2^{24}$. When used in algebraic expressions, the sign on an integer $I$ is determined by the relation

\[
I = \begin{cases} 
1 & \text{if } 0 \leq I < 2^{23} \\
-2^{24} + I & \text{if } 2^{23} \leq I \leq 2^{24} - 1
\end{cases}.
\]

which has a range from $-2^{23}$ to $2^{23} - 1$. We may therefore use the single Fortran statement $I = I \times A$ for each random integer on $-2^{23} < I < 2^{23} - 1$ to produce a new random integer thus giving the congruence relation

\[X_{i+1} = a \times X_i \mod (2^{24}) \]

Finally dividing by $2^{24}$ and adding $1/2$ will produce a uniform variate on $[0, 1)$.

The process is more complicated in the Univac 1108 digital computer where 1's complement arithmetic is used. Here, multiplication of two 36-bit integers yields the product mod ($2^{36}$) but the sign is handled by

\[
m(I) = \begin{cases} 
1 & \text{if } 0 \leq I < 2^{35} \\
-2^{36} + I + 1 & \text{if } I \geq 2^{35}
\end{cases}.
\]
Therefore, steps must be taken to subtract the integer 1 at certain times to represent the proper remainders of \(2^{36}\). The following instruction, which is a correction by Grosenbaugh [8] to Marsaglia and Bray's original Fortran instruction, handles this requirement satisfactorily:

\[
I = I \times K + \text{MINO} (0, \text{ISIGN}(K-1, L))
\]

The two functions MINO and ISIGN are standard Fortran library routines. MINO determines the minimum value of a series of integer quantities, and ISIGN transfers the sign of the second argument to the absolute value of the first argument. The constant \(K\) can be chosen for maximum period as shown by Van Gelder [6].

The CDC 3200 digital computer handles integers exactly as does the Univac 1108. However, the CDC 3200 stores integers as 24 bits, therefore the only change required is the power of 2 in all equations.

These one-line Fortran generators may be incorporated directly into programs easily as they are, but Marsaglia and Bray obtained better statistical results upon combining several generators. As an example, consider the following SDS 930 generator:

\[
\begin{align*}
L &= L \times ML \\
M &= M \times MM \\
J &= 1 + \text{IABS}(L)/2^{16} \\
U &= 1/2 + [N(J) + L + M]/2^{24} \\
K &= K \times MK \\
N(J) &= K
\end{align*}
\]

The array \(N\) is a 128-element array filled with random numbers previously assigned by a one-line generator. In the procedure, \(J\) is used to choose from the \(N\) array; \(J\) comes from the random integer \(L\) after division by the appropriate power of 2. The desired random variable \(U\) is formed from the
sum of the randomly chosen $N$ array element, the random integer $L$ used to find $J$, and a third additional random integer $M$. The used element of $N$ is replaced by the random integer $K$. Similar procedures are used for the Univac 1108 and CDC 3200 generators.

Subprogram listings are given for the Univac 1108, CDC 3200, and SDS 930 generators in Appendices A, B, and C, respectively. All three subprograms are called in the exact same manner, so that a computer program may be converted to run on several machines by simply inserting the appropriate generator. Initially, the subprograms must be given a non-zero integer to establish the internal 128-cell array. An odd number in the millions gives excellent statistical results. The initial call to a generator does not produce a useful result. Uniform random variables on the unit interval $[0, 1)$ may be obtained by calling the generator with a zero integer argument. To repeat a sequence of variables, the generator is simply reinitialized with the same initial value.

GENERATION OF NORMAL VARIABLES

As in the case of the uniform pseudo-random number generator, there are a variety of ways to generate normally distributed random variables. A method of normal variable generation by Box and Muller [9, 10] is discussed here. Their method is simple, fast, and requires very little memory storage.

The following method is used to generate a pair of random deviates $(X_1, X_2)$ from the same normal distribution starting from a pair of uniformly distributed random variables $(U_1, U_2)$ distributed $[0, 1)$.

\[
X_1 = (-2 \ln U_1)^{\frac{1}{2}} \cos (2 \pi U_2) \\
X_2 = (-2 \ln U_1)^{\frac{1}{2}} \sin (2 \pi U_2)
\]

The pair $(X_1, X_2)$ will be normally distributed $[0, 1]$ and are very reliable in the tails of the distribution. The estimated speed is 6.01 milliseconds per variable on a CDC 3200 digital computer.
CONCLUSION

Methods have been shown and subprograms have been developed for the fast, simple, and reproducible generation of pseudo-random numbers for uniform and normal probability distributions. The accuracy of the numbers must not be assumed to be 100 percent for all utilizations, because potential error sources do exist. Marsaglia [5] presents results that indicate that every multiplicative generator has a defect that makes it unsuitable for certain Monte Carlo applications. Other remote error possibilities are stated in the references. However, for most applications the generators presented herein give accurate results.
APPENDIX A

FORTRAN LISTING OF UNIVAC 1108 UNIFORM GENERATOR

FUNCTION RAN(JJ)
C
UNIFORM RANDOM NUMBER GENERATOR FOR THE UNIVAC 1108
C
C
FIRST CALL MUST BE OF THE FORM X = RAN(J), WHERE J IS AN
C
INITIAL INTEGER VALUE. SUBSEQUENT CALLS MUST BE OF THE FORM
C
X = RAN(0).
C
DIMENSION N(128)
J=JJ
IF (J.EQ.0) GO TO 2
DO 1 ISL=1,128
J=J*2**227157*MINO(0,ISIGN(227156,J))
1 N(I)=J
L=JJ
M=JJ
K=JJ
2 L=L*2**274693*MINO(0,ISIGN(274692,L))
M=M*2**43133*MINO(0,ISIGN(243132,M))
I=ABS(L)/2**68435456+1
RAN=.5*FLOAT(N(I)+L+M)*1.4551952E-10
K=K*2**49149*MINO(0,ISIGN(249148,K))
N(I)=K
RETURN
END
FUNCTION RAN(JJ)

    UNIFORM RANDOM NUMBER GENERATOR FOR THE CDC 3200

    FIRST CALL MUST BE OF THE FORM X = RAN(J), WHERE J IS AN
    INITIAL INTEGER VALUE. SUBSEQUENT CALLS MUST BE OF THE FORM
    X = RAN(J).

    DIMENSION N(128)
    J=JJ
    IF (J.EQ.0) 3,1
1   U=2 I=1,128
    J=J*227157*MIN0(0,ISIGN(227156,J))
2   L=L*274693+MIN0(0,ISIGN(274692,L))
    M=M*243133+MIN0(0,ISIGN(243132,M))
    I=ABS(L)/65536+1
    RAN=.5*FLOAT(N(I)+L*M)*596046447E-07
    K=K*249149*MIN0(0,ISIGN(249148,K))
    N(I)=K
    RETURN
END

      RAN  10
      RAN  20
      RAN  30
      RAN  40
      RAN  50
      RAN  60
      RAN  70
      RAN  80
      RAN  90
     RAN 100
     RAN 110
     RAN 120
     RAN 130
     RAN 140
     RAN 150
     RAN 160
     RAN 170
     RAN 180
     RAN 190
     RAN 200
     RAN 210
     RAN 220
     RAN 230
     RAN 240
     RAN 250-
APPENDIX C

FORTRAN LISTING OF SDS 930 UNIFORM GENERATOR

FUNCTION RAN(JJ)
C UNIFORM RANDOM NUMBER GENERATOR FOR THE SDS 930
C
C FIRST CALL MUST BE OF THE FORM X = RAN(J), WHERE J IS AN
C INITIAL INTEGER VALUE. SUBSEQUENT CALLS MUST BE OF THE FORM
C X = RAN(0).
C
DIMENSION N(128)
J=JJ
IF (J) 1,3,1
1 DO 2 J=1,128,1
J=J*65539
2 N(I)=J
L=JJ
M=JJ
K=JJ
3 L=L*4357
M=M*9197
I=IABS(L)/65536
HAN=0.5*FLOAT(N(I)+L*M)*0.59604644E-07
K=K*10757
N(I)=K
RETURN
END

RAN 10
RAN 20
RAN 30
RAN 40
RAN 50
RAN 60
RAN 70
RAN 80
RAN 90
RAN 100
RAN 110
RAN 120
RAN 130
RAN 140
RAN 150
RAN 160
RAN 170
RAN 180
RAN 190
RAN 200
RAN 210
RAN 220
RAN 230
RAN 240
RAN 250-
REFERENCES


BIBLIOGRAPHY


UNIFORM RANDOM NUMBER GENERATORS

By William R. Farr

The information in this report has been reviewed for security classification. Review of any information concerning Department of Defense or Atomic Energy Commission programs has been made by the MSFC Security Classification Officer. This report, in its entirety, has been determined to be unclassified.

This document has also been reviewed and approved for technical accuracy.

ROBERT M. CROFT
Acting Chief, Flight Mechanics Branch

WILLIAM D. GOLDSBY, JR.
Deputy Chief, Flight Performance and Mission Analysis Division

ERICH E. GOERNER
Director, Preliminary Design Office

JAMES T. MURPHY
Acting Director, Program Development
**DISTRIBUTION**

<table>
<thead>
<tr>
<th>INTERNAL</th>
<th>PD-DO-P</th>
<th>Mr. Goldsby</th>
<th>S&amp;E-AERO-Y</th>
<th>Mr. Vaughn</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIR</td>
<td>PD-DO-PF</td>
<td>Mr. Croft</td>
<td>S&amp;E-AERO-YA</td>
<td>Mr. Kaufman</td>
</tr>
<tr>
<td></td>
<td>Dr. Rees</td>
<td>Mr. Sumrall</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PD-DO-PF</td>
<td>Mr. Lowery</td>
<td>S&amp;E-COMP-SD</td>
<td>Mr. Zeannah</td>
</tr>
<tr>
<td>DEP-T</td>
<td>Mr. Perkison</td>
<td>Mr. Solmon</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>AD-S</td>
<td>Mr. Farr (10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dr. Stuhlinger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PD-DIR</td>
<td>PD-DO-PM</td>
<td>A&amp;TS-MS-H</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mr. Murphy</td>
<td>Mr. Leonard</td>
<td>A&amp;TS-MS-IL</td>
<td>Miss Robertson (8)</td>
</tr>
<tr>
<td></td>
<td>PD-SQA-DIR</td>
<td>Mr. Gold</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mr. Huber</td>
<td>Mr. Wheeler</td>
<td>A&amp;TS-MS-IP</td>
<td>Mr. Ziak (2)</td>
</tr>
<tr>
<td></td>
<td>PD-SQA-V</td>
<td>Mr. Bradford</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mr. Orillion</td>
<td>Mr. Young</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PD-DO-E</td>
<td>PD-DO-PA</td>
<td>A&amp;TS-PAT</td>
<td>Mr. L. D. Wofford, Jr.</td>
</tr>
<tr>
<td></td>
<td>Mr. Digesu</td>
<td>Mr. Allen</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PD-DO-ES</td>
<td>Mr. Jump</td>
<td>A&amp;TS-TU (6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mr. Cole</td>
<td>Mr. Burton</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dr. Thomason</td>
<td>Mr. French</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mr. Heyer</td>
<td>Mr. Zimmerman</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mrs. Andrews</td>
<td>Mr. Sullivan</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mr. Sparks</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTERNAL</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scientific and Technical Information Facility (25)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>P. O. Box 33</td>
<td>College Park, Maryland 20740</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attn: NASA Representative (S-AK/RKT)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lockheed Missiles and Space Company</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology Drive</td>
<td>Huntsville, Alabama 35805</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASA Headquarters</td>
<td>Washington, D. C. 20546</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASA Ames Research Center</td>
<td>Mountain View, California 94035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASA – Langley Research Center</td>
<td>Hampton, Virginia 23365</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jet Propulsion Laboratory</td>
<td>Pasadena, California 91103</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NASA – Lewis Research Center</td>
<td>21000 Brookpark Road</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cleveland, Ohio 44135</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer Sciences Corporation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S&amp;E-COMP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attn: Mr. Ron Hall</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Attn: Mr. Emmett Jones</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Northrop Space Laboratories</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technology Drive</td>
<td>Huntsville, Alabama 35805</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>