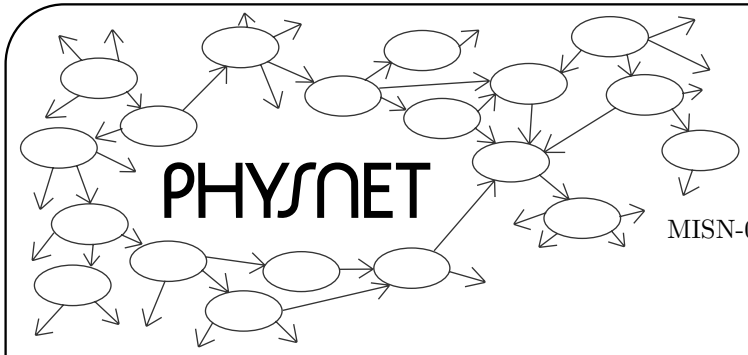

MISN-0-354

GENERATION OF RANDOM NUMBERS BY COMPUTER

.19700	.90515	.29637	.42424	.62188
.80175	.75286	.80151	.53337	.51323
.15799	.03256	.72655	.84777	.12528
.89599	.52696	.27464	.89032	.37010
.41673	.62908	.47612	.01816	.32402
.73241	.41984	.57256	.71374	.37059
.95392	.45476	.81347	.71203	.09305
.43602	.03183	.61504	.90368	.38676
.85522	.81951	.72021	.55431	.30741
.13181	.49309	.72784	.30454	.22312
.96899	.90173	.63091	.83001	.15793
.05521	.36955	.21409	.45866	.32517
.51244	.66619	.88525	.18418	.42796
.61058	.89813	.60649	.77825	.37236
.42235	.32286	.23807	.02280	.49419
.38841	.83093	.98102	.09220	.88208
.66314	.14170	.61791	.51744	.83447

Project PHYSNET Physics Bldg. Michigan State University East Lansing, MI

GENERATION OF RANDOM NUMBERS BY COMPUTER

by

Robert Ehrlich

1. Pseudorandom Sequences
a. Introduction 1
b. Pseudorandom Numbers 1
c. Period of Pseudorandom Number Sequences 1
d. Distributions of Pseudorandom Numbers 2

2. The Power Residue Method
a. Description of the Method 2
b. Example of the Power Residue Method 2
c. Optimum Choice of the Parameters 3
d. A Pseudorandom Number Sequence 3

3. Statistical Test of Randomness
a. Frequency Distribution Tests 5
b. Examples of Frequency Distribution Tests 5
c. Frequency of Strings Test 7
d. No Absolute Tests of Randomness 7

4. Distribution Functions
a. Introduction 8
b. Uniform Distribution For An Interval 8
c. Non-Uniform Probability Distributions 9

5. Generating Pseudorandom Numbers
a. Input 9
b. Output 10
c. Handling Overflows on Different Computers 10

6. Procedures 11
a. Length of Period 11
b. Frequency Distribution Test 11
c. Frequency of Strings Test 12
d. Non-Uniformly Distributed Random Numbers 12

Acknowledgments 12

A. Professional Random Number Discussions 12

B. Fortran, Basic, C++ Programs 13

Title: **Generation of Random Numbers by Computer**

Author: R. Ehrlich, Physics Dept., George Mason Univ., Fairfax, VA 22030; (703)323-2303.

Version: 3/18/2002 Evaluation: Stage 0

Length: 2 hr; 20 pages

Input Skills:

1. Write (or modify) and run programs using advanced programming techniques such as arrays and nested loops in FORTRAN (MISN-0-347) or in BASIC.

Output Skills (Knowledge):

K1. Vocabulary: sequence of random numbers, probability density function, pseudorandom, period of a sequence of pseudorandom numbers, power residue method, frequency distribution test, frequency of strings test, frequency distribution, non-uniformly distributed, pseudorandom numbers.

K2. State the conditions for the longest period of a pseudorandom number sequence generated using the power residue method.

K3. State the criteria for a good pseudorandom number generator.

Output Skills (Project):

P1. Enter and run a computer program which uses the power residue method to generate a pseudorandom number sequence and test the sequence obtained for the two criteria of a good random number generator.

P2. Modify and run the computer program in P1 to generate pseudorandom numbers which have an exponential distribution function.

External Resources (Required):

1. A computer with BASIC, FORTRAN, or C++.

GENERATION OF RANDOM NUMBERS BY COMPUTER

by

Robert Ehrlich

1. Pseudorandom Sequences

1a. Introduction. This module introduces the concept of a "sequence of random numbers," and the generation of random number sequences on a computer. Sequences of random numbers are made use of in the so-called Monte Carlo method of problem solving. In other modules the Monte Carlo method is used to find the center of mass of an object, simulate a nuclear chain reaction, and simulate the decay of a sample of radioactive nuclei.

1b. Pseudorandom Numbers. A sequence of numbers is said to be random if the value of each successive number cannot be predicted based on the preceding numbers. Such numbers could, in principle, be generated using some process normally considered to be random, such as rolling a pair of dice. In practice, we will primarily be interested in arithmetic methods for generating random numbers that can be implemented on a computer. These sequences are referred to as "pseudorandom," since the numbers are generated according to an arithmetic procedure in which each number is uniquely determined from the ones that preceded it, and therefore they cannot be considered truly random. A successful method for generating pseudorandom numbers yields a number sequence that would appear indistinguishable from a truly random sequence in the sense that no regularities of any kind would appear in the sequence beyond what the laws of chance would predict.

1c. Period of Pseudorandom Number Sequences. In most methods for obtaining a sequence of pseudorandom numbers, each number in the sequence is used to find the next one using some arithmetic procedure. Thus the entire sequence is determined given the first number. Any arithmetic procedure used to generate pseudorandom numbers on a computer yields a sequence that must repeat itself after some number of terms. The reason is connected with the finite number of different numbers that can be represented on a computer owing to the finite size of each memory word. Let n designate the number of different numbers that can be represented. Any sequence of $n + 1$ pseudorandom numbers

must then include at least one duplication which would cause the entire sequence to repeat, in the event that each number determines the next. The "period" of the pseudorandom sequence is the number of terms before a repetition occurs. In practice we want to select a method for generating pseudorandom number sequences that has a period as large as possible.

1d. Distributions of Pseudorandom Numbers. We normally seek to generate a sequence of pseudorandom numbers such that the relative frequency of the numbers obeys some particular distribution function. Most commonly, it is desired to generate a pseudorandom sequence in which the numbers are equally likely to have any value in some interval, for example, the interval 0 to 1. Such a distribution is said to be flat or uniform since the number of numbers in each sub-interval would be equal.

2. The Power Residue Method

2a. Description of the Method. The "power residue method" is widely used and can generate a sequence of pseudorandom integers, which has both a long period and good statistical properties. Given a first integer x_0, commonly called the process's "seed," each successive number in the sequence can be found from the preceding one, according to

$$x_n = cx_{n-1} \pmod{M}, \tag{1}$$

where c and M are two constant integers. The notation $y = z \pmod{M}$ means that if z exceeds M, then the modulus M is subtracted from z as many times as necessary in order that $0 < y < M$; that is, y and z differ by some multiple of M (for example, $5752 \pmod{100} = 52$.

2b. Example of the Power Residue Method. As an example, let us take $M = 32$, $c = 5$, and $x_0 = 1$. We can then repeatedly use Eq.(1) to find the entire sequence:

$$\begin{array}{lcccccl} x_1 & = & 5x_0 \pmod{32} & = & (5)(1)\text{-}(0)(32) & = & 5 \\ x_2 & = & 5x_1 \pmod{32} & = & (5)(5)\text{-}(0)(32) & = & 25 \\ x_3 & = & 5x_2 \pmod{32} & = & (5)(25)\text{-}(3)(32) & = & 29 \\ x_4 & = & 5x_3 \pmod{32} & = & (5)(29)\text{-}(4)(32) & = & 17 \\ \vdots & & & & & & \end{array}$$

The first 12 numbers in this sequence are:

5, 25, 29, 17, 21, 9, 13, 1, 5, 25, 29, 17

We notice that the sequence repeats after only eight numbers, which would be much too small a period for the sequence to be of any practical use. In general, the length of the period depends on all three parameters M, c, x_0, so they must be chosen carefully to insure a long period.

2c. Optimum Choice of the Parameters. It may be shown that to obtain as long a period as possible, the following conditions should be satisfied for the parameters M, c, and x_0:

(1) M is one more than the largest integer than can be represented on the computer: (on a binary computer of b bits, for which each integer is stored in a single word, we have $M = 2^{b-1}$,

(2) x_0 = any odd integer.

In addition, in order that the sequence have the best statistical properties, it is further recommended that the value of c be close to $M^{1/2}$, although the period is not thereby lengthened.

2d. A Pseudorandom Number Sequence. In many applications involving random numbers, we do not want a sequence of pseudorandom integers; instead we want a sequence of numbers all of which lie in the interval from zero to one. This can be achieved by dividing each integer in the sequence by $M-1$, which is the largest possible integer. The sequence of numbers listed in Table 1 are the first 300 numbers generated using the power residue method with the constants M, c, x_0 chosen as stipulated in the preceding section. A cursory look at the numbers listed reveals no repetition of the sequence which means that the period exceeds 300 numbers. Additionally, there is no obvious departure from randomness evident. For example, there does not appear to be more numbers above 0.5 than below 0.5. Beyond such superficial observations there are a number of statistical tests that we can apply to pseudorandom number sequences to see whether they have the same properties of number sequences that are genuinely random.

Table 1. A Pseudorandom Number Sequence Generated Using the Power Residue Method.

N=300 M=32768 x0=13 c=199					
.07895	.71087	.45836	.21036	.86120	.37394
.41246	.07657	.23716	.19327	.46031	.59905
.20688	.16886	.60192	.77776	.77038	.30070
.83728	.61272	.92761	.58879	.16636	.10392
.67858	.03311	.58922	.25138	.02329	.63372
.10581	.05509	.96179	.39055	.71618	.51506
.49400	.30363	.42027	.63128	.61998	.37297
.21812	.40373	.33970	.59856	.10971	.83245
.65319	.98041	.09507	.91742	.56041	.51848
.17417	.65862	.06137	.21281	.34703	.05606
.15610	.06339	.61364	.10978	.84460	.07022
.97400	.81976	.12680	.23331	.42808	.18595
.00278	.55266	.97595	.20841	.47252	.02823
.61754	.88714	.53600	.66008	.35289	.22208
.19321	.44816	.18198	.21329	.44420	.39250
.10483	.86077	.28892	.49309	.12143	.16446
.72741	.74993	.23179	.12442	.75964	.16300
.43590	.74065	.38560	.73235	.73376	.01309
.60533	.45793	.12534	.94183	.41881	.33976
.61071	.52678	.82604	.37785	.18979	.76800
.82702	.57219	.86267	.66546	.42174	.92279
.62926	.21915	.61022	.42961	.48961	.42912
.39244	.09268	.44371	.29533	.76843	.91205
.49156	.81780	.73815	.88763	.63317	.99652
.30161	.01944	.86853	.83148	.45885	.30754
.19761	.32267	.20981	.75188	.62047	.47014
.55455	.35246	.13706	.27384	.49303	.10929
.74743	.73382	.02524	.02237	.45152	.85003
.15122	.09171	.24937	.62249	.87097	.31730
.14096	.05118	.18442	.69915	.12632	.13614
.09165	.23722	.20542	.87738	.59264	.93158
.37828	.27482	.68737	.78216	.64489	.32853
.37584	.78900	.00522	.03848	.65807	.95209
.45933	.40471	.53404	.27140	.00717	.42717
.00375	.74700	.64879	.10587	.06723	.37883
.38414	.44084	.72448	.16691	.21323	.43205
.97546	.11124	.13608	.07950	.82018	.21183
.15268	.38322	.25864	.46867	.26304	.34318
.29087	.88177	.46715	.95941	.91687	.45109

.76501	.23185	.13657	.17667	.15659	.16056
.95007	.05850	.64196	.74554	.35728	.09659
.22105	.98675	.35826	.29093	.89392	.88421
.95300	.64153	.66051	.43791	.14145	.14835
.52086	.64788	.92370	.81146	.47496	.51408
.29966	.63079	.52281	.03653	.26939	.60637
.66442	.21525	.83288	.73821	.89978	.05020
.99011	.02628	.22886	.54143	.74108	.47063
.65172	.68889	.08579	.07120	.16831	.49260
.02426	.82806	.77868	.95257	.55651	.74114
.48277	.06876	.68249	.81048	.28062	.84124

3. Statistical Test of Randomness

3a. Frequency Distribution Tests. Tests can help determine the extent to which the numbers in a pseudorandom sequence are consistent with a uniform (flat) frequency distribution from zero to one (assuming this to be their range). Let us divide up the interval from zero to one into M equal size subintervals and count the number of numbers which lie in each subinterval. The number of numbers in the j^th subinterval is designated x_j, and the average number in all subintervals is designated $\bar{x}$. The frequency distribution test then consists in observing to what extent the deviations $d_j = x_j - \bar{x}$ are consistent with the laws of statistics.

3b. Examples of Frequency Distribution Tests. As an illustration, the first 300 numbers generated using the power residue method with modulus $M = 2^{15} = 32768$, constant $c = 199$, and seed $x_0 = 13$ are listed in Table 1. The frequency distribution, or histogram, in Fig. 1 shows the number of numbers in each of the ten subintervals between zero and one. The figure also indicates the deviations from a uniform distribution (exactly 30 in each subinterval). The laws of statistics applied to truly random sample predict that as long as the average number of numbers in each subinterval is greater than about 25, the probability of finding x numbers in a subinterval is given by the Gaussian distribution,

$$f(x) = \exp\left(\frac{-(x-\bar{x})^2}{2\sigma^2}\right), \quad \sigma = \bar{x}^{1/2},$$

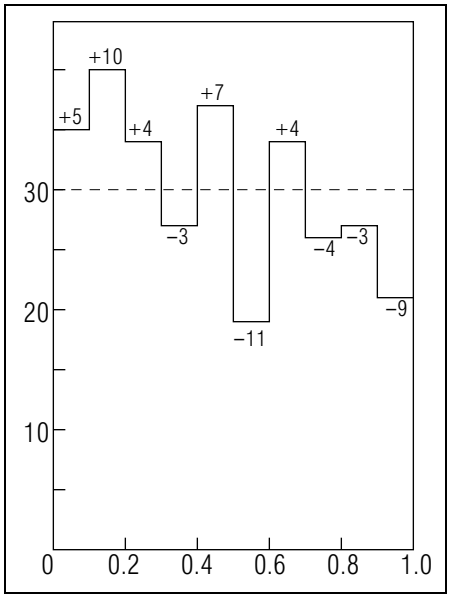

Figure 1. Frequency distribution of pseudorandom numbers of Table 1, in 10 bins.

where σ is the "standard deviation." According to this distribution, the most likely number of numbers in any subinterval is the average number,

$$\bar{x} = \sum^{M} x_j/M,$$

and there is a 68% probability of finding x within a one standard deviation range of the average value: $x-\sigma$ to $x+\sigma$ (which corresponds to deviations in the range $-\sigma$ to $+\sigma$). For an average number $\bar{x} = 30$, we find a standard deviation $\sigma = 30^{1/2} \cong 5.5$ As can be seen in Fig. 1, six out of ten of the deviations are less than 5.5, which is in fairly good agreement with the expected 68%. Better agreement would be expected if we had more pseudorandom numbers in the sample. Another test is to compute the root mean square (rms) deviation for all subintervals

$$d_{rms} = \left(\frac{1}{M}\sum^{M} d_j^2\right)^{1/2} = \left(\frac{1}{M}\sum^{M}(x_j - \bar{x})^2\right)^{1/2}. \qquad (2)$$

For a large number of subintervals we expect d_{rms} to approach the standard deviation σ.

In the present example we determine d_{rms} using $M = 10$ and $d_1 = +5$, $d_2 = +10, \cdots, d_{10} = -9$, as indicated in Fig. 1. The computed value $d_{rms} = 6.7$ is reasonably close to the standard deviation, $\sigma = 5.5$. (Reasonably close is here defined in terms of the expected "deviation of the deviation" given by: $\pm\sigma M^{1/2} = \pm 1$). Better agreement would undoubtedly be obtained if we had more pseudorandom numbers in the sample. Frequency distribution tests are not sensitive to the order of pseudorandom numbers in the sequence, only to their relative frequency. For example, suppose some number-generation method had produced the same 300 numbers that are listed in Table 1, but in monotonically increasing order. We certainly would not regard that as a random sequence but we would still obtain the same random-looking histogram shown in Fig. 1.

3c. Frequency of Strings Test. Some information on the order of numbers in a sequence can be found by looking for "strings" of various length. A string of length k consists of consecutive numbers which all have some specific property. We may, for example, look for strings of numbers which are all above the mean value (1/2) or all below the mean value. We can then compare the number of strings of length k found with the number of such strings that would be expected for a truly random sequence:

$$\text{expected number of strings of length } k = \frac{M - k + 3}{2^{k+1}}. \qquad (3)$$

3d. No Absolute Tests of Randomness. The difficulty in determining that a sequence of numbers has no regularities of any kind makes finding a good random number generator as much an art as a science. Should a particular sequence of pseudorandom numbers give satisfactory results for the two preceding tests we would have some confidence in the method used to generate the sequence. However, it would certainly not prove that the sequence has all the statistical properties of a truly random sequence. Unfortunately there are an unlimited number of possible regularities that a sequence of numbers might have that would escape detection in these particular tests. A sequence which appears to be random according to several tests may reveal its nonrandom character only in some special application. Even for a particular technique, such as the power residue method, the randomness of the sequence may vary according to the choice of the constants M, c, x_0.

4. Distribution Functions

4a. Introduction. Let z represent a random variable that can take on values in the interval $-\infty < z < +\infty$. Let f(z) be a piecewise continuous function such that the probability that the variable has a value between z and $z + dz$ is given by $f(z)\,dz$. In that case, $f(z)$ is called the "probability density function" (or the "distribution function") for the variable z. The fact that z is required to take on some value requires that:

$$\int_{-\infty}^{\infty} f(z)\,dz = 1 \qquad (4)$$

Uniformly distributed pseudorandom numbers in the range zero to one satisfy the condition:

$$\begin{aligned} \text{f(z)} &= 1 \text{ for } 0 < z < 1 \\ &= 0 \text{ otherwise} \end{aligned}$$

We wish to find a way to generate pseudorandom numbers r which are distributed according to other probability density functions. Let $F(z)$ be the cumulative distribution function:

$$F(z) = \int_{-\infty}^{Z} f(z')\,dz' \qquad (5)$$

corresponding to a particular distribution function $f(z')$ for which we want to generate a pseudorandom number sequence. $F(z)$ is equally likely to take on any value between zero and one. The equation

$$F(z) = r = \text{ uniformly distributed pseudorandom number}$$

may therefore be inverted to solve for z:

$$z = F^{-1}(r)$$

The variable z then has the desired distribution function $f(z)$.

4b. Uniform Distribution For An Interval. Given a pseudorandom number uniformly distributed in the interval zero to one, we may easily find a random number that has a different range a to b. The desired probability density function has the form:

$$\begin{aligned} \text{f(z)} &= \text{c (a constant)} && a < z < b \\ &= 0 && \text{otherwise} \end{aligned}$$

The condition:

$$\int_{-\infty}^{\infty} f(z)\,dz = 1$$

then requires that $c = 1/(b-a)$. Thus we have for the cumulative distribution function $F(z)$:

$$F(z) = \int_{-\infty}^{Z} f(z')dz' = \frac{z-a}{b-a}$$

Whereupon the requirement $F(z) = r$ can be immediately solved for z, giving:

$$z = r(b-a) + a \tag{6}$$

It is clear that the ends of the interval $(0,1)$ for r correspond to the ends of the interval (a,b) for z.

4c. Non-Uniform Probability Distributions. As an example of how pseudorandom numbers can be generated according to a particular non-uniform probability distribution, consider the function:

$$\begin{aligned} f(z) \quad &= e^{-az} \qquad z \geq 0 \\ &= 0 \qquad\quad z < 0 \end{aligned}$$

and the corresponding cumulative distribution function found by the required integration:

$$F(z) = \frac{1}{a}(1 - e^{-az})$$

In this case, the requirement $F(z) = r$ can also be inverted to solve for z yielding:

$$z = -\frac{1}{a}\ell n\,(1-ar). \tag{7}$$

Thus, the variable z will be distributed according to $F(z) = e^{-az}$, provided r is a pseudorandom number uniformly distributed on the interval zero to one.

5. Generating Pseudorandom Numbers

5a. Input. The program to generate uniformly distributed pseudorandom numbers first reads numerical values for the parameters `NUM`, `N`, `X`, and `C`, where

`N`	=	number of pseudorandom numbers to be generated
`M`	=	the modulus
`X`	=	x_0, the seed, the first number in the sequence
`C`	=	c, the constant appearing in the power residue formula

5b. Output. The program uses the power residue method to generate `N` pseudorandom numbers having a uniform distribution between zero and one. The program generates the random numbers in sets of six, in order that each set of six numbers can be printed on a single line. Thus, instead of a single loop from 1 to `N`, the program uses two nested loops, the outer one going from 1 to `N`/6, and the inner one going from 1 to 6. To obtain each pseudorandom number from the preceding one, we need to compute the product, cx (modulo M). This can be done by first computing the new x from the preceding one using $x \leftarrow cx$. Then, to find x (modulo M), we can use

$$x \leftarrow x - M\,(x/M), \tag{8}$$

where all operations are carried out according to the rules of integer arithmetic, i.e., quantities are truncated to the next lower integer after each arithmetic operation. Finally, to obtain a pseudorandom number between zero and one, we divide x by $M-1$, its maximum possible value. After the program finds six consecutive numbers, R_1, R_2, ..., R_6, it prints them on a line and proceeds to the next set of six. When the entire sequence of `N` numbers has been printed, the program reads the next data card if any remain. The sample results shown in Table 1 have been already discussed. That output can be generated using these values for the parameters:

`N`	`M`	`X`	`C`
300.	32768.	13.	899.

5c. Handling Overflows on Different Computers. A complication arises in the way particular computers handle a multiplication in which the product of two numbers exceeds the largest number that can be represented on the computer. This results in what is called an overflow. Due to the way most computers carry out integer arithmetic in FORTRAN, if there is an overflow, the leading (most significant) part of the result is dropped. In this case, the result is computed modulo $(I_{max}+1)$, apart from its sign which may come out negative, if an overflow into the leading (sign) bit occurs. Thus, by using $x \leftarrow |cx|$, we get a result which is automatically computed modulo $(I_{max}+1)$. This implies that the subsequent computation of x(modulo M) using Eq. 6 can only be correct if $I_{max}+1$ is divisible by M. On a binary computer this restricts M to powers of

two. In other words, if $M = 2^b$, then the possible initial loss of leading bits (due to overflow) in no way affects the computation of x (modulo M), in which all but the lower order b bits are dropped. Since overflows may not affect the sign bit on some computers it is possible that your output may not look exactly like that in Table 1. You may find that roughly half the numbers agree while the other half are equal to the values listed subtracted from 1.0 .

6. Procedures

6a. Length of Period. Run the original version of the program using the values for c, M and x_0 specified in Table 1 and see if your 300 pseudorandom numbers are the same as those in the Table. If some of the numbers agree and some disagree, see the discussion of overflow handling on different computers in Sect. 5c. Next, you should modify the program so that it keeps a count of how many numbers are generated before the first number in the sequence repeats, thereby determining the length of the period of the sequence. Also, delete the printout of the individual numbers.

6b. Frequency Distribution Test. Modify the program so that it generates a sequence of pseudorandom numbers and keeps a count of how many numbers lie in each of ten equal subintervals from zero to one. Use the parameters $M = 32768$, $X = 13$ and $C = 199$. Make runs of 300 numbers each, and for each run compute the deviations in each subinterval from the mean (30), and also compute the root mean square deviation from the mean. For each run you should print out only the number of numbers in each of the ten intervals and the rms deviation from the mean. Modify the program so that the above is accomplished automatically for a series of 25 runs without needing to enter new parameters each time. Be sure that the parameter X is not reset each time, otherwise you will recalculate the same set of random numbers 25 times. After 25 sets of rms deviations have been found compare them with what would be expected from the laws of statistics: $\sigma = 30^{1/2} \cong 5.5$. You should find that roughly half of the rms deviations are above this value and half are below. According to theory, the power residue method gives pseudorandom number series having the best statistical properties when the parameter C is chosen close to $M^{1/2}$. Redo the series of 25 runs using a value for C far from the recommended value (perhaps $C = 5$). How are the rms deviations affected by this change?

6c. Frequency of Strings Test. A string of length k is defined as the number of k consecutive pseudorandom numbers which are all either above the mean value (0.5) or below the mean value. Make 3 runs of 100 pseudorandom numbers each. You should be able to count by hand the number of strings of length $k = 2, 3, \cdots, 8$ in each run and then construct a table showing the number of strings of length $k = 2, 3, ..., 8$ for each of the 3 runs. Also show in the table the expected number of strings of length k that would be expected for a truly random sequence based on Eq 3.

6d. Non-Uniformly Distributed Random Numbers. Modify the program so that it generates pseudorandom numbers having a distribution function $f(z) = e^{-z}$. This can be accomplished using uniformly distributed random numbers, r, and then calculating z using Eq. 7 with $a = 1$:

$$z = -\ell n\,(1 - r)$$

The variable z will then have the desired distribution, as explained in Sect. 4c. Modify the program to keep track of how often the pseudorandom number z lies in each of ten successive equal size intervals from 0 to 3. Run the program to generate a sequence of 1000 random numbers and plot (by hand) a histogram showing the number of z values in each interval. If you make the plot on semilog paper it will be easy to compare your histogram with the expected distribution function e^{-z} which should be a unit slope straight line.

Acknowledgments

Preparation of this module was supported in part by the National Science Foundation, Division of Science Education Development and Research, through Grant #SED 74-20088 to Michigan State University.

A. Professional Random Number Discussions

For professional-level material, we recommend that you visit the Web site of Cambridge University Press, where you will find some 15 coordinated books and disks for sale under the collective name "Numerical Recipes." Go to `http://www.cup.org` and type "numerical recipes" (without the quotes) in the Search box. The books you will find there (and in libraries) contain authoritative material, programs, and references to sources in the

literature. Some titles are:

1. *Numerical Recipes*, Press, William H.; Teukolsky, Saul A.; Vetterling, William T., and others.
2. *Numerical Recipes in FORTRAN 90, The Art of Parallel Scientific Computing*, Press, William H.; Vetterling, William T.; Metcalf, Michael Publication (Sept. 1996).
3. *Numerical Recipes in C++*, Press, William H., Teukolsky, Saul A., Vetterling, William T., Flannery, Brian P. (2nd ed., 2002).
4. *Numerical Recipes Routines and Examples in BASIC*, Sprott, Julien C., In association with Numerical Recipes Software.

B. Fortran, Basic, C++ Programs

All programs are at

`http://www.physnet.org/home/modules/support_programs`

which can be navigated to from the home page at

`http://www.physnet.org`

by following the links: → `modules` → `support programs`, where the programs are:

`m354p1f.for`, Fortran;
`m354p1b.bas`, Basic;
`m354p1c.cpp`, C++;
`lib351.h`, needed Library for C++ program;

MODEL EXAM

1-3. See Output Skills K1-K3.

Examinee:
On your computer output sheet(s):

(i) Mark page numbers in the upper right corners of all sheets.

(ii) Label all output, including all axes on all graphs.

On your Exam Answer Sheet(s), for each of the following parts of items (below this box), show:

(i) a reference to your annotated output; and

(ii) a blank area for grader comments.

When finished, staple together your sheets as usual, but include the original of your annotated output sheets just behind the Exam Answer Sheet.

4. Submit your hand-annotated output from using the power residue method to generate a pseudorandom number sequence. Be sure it shows:
   a. the degree of agreement with the text table reached on a check run and reasons for any discrepancy;
   b. the number of numbers generated before a repeat of the first number in the sequence.
   c. the count of numbers lying in each of 10 equal subintervals from zero to one, using the parameters $M = 32768$, $X = 13$, $C = 199$, and with 300 numbers in each of 25 runs, where the program prints out only the number of numbers in each subinterval and the rms deviations from each subinterval's mean number of numbers, and a comparison with what is expected from the law of statistics.
   d. for (c) above, a series of 25 runs using a value of C far from the recommended value and an interpretation of how the rms deviations are affected.
5. Submit your hand-annotated output for the frequency-of-strings test. Make sure it shows:

a. a table containing your hand count of the number of strings of length 2, 3, 4, ...,8 for 3 runs each having 100 pseudorandom numbers;

b. in the same table, the numbers expected for a truly random sequence.

6. Submit your hand-annotated output for the exponentially distributed pseudorandom numbers.

   a. a hand-plotted histogram showing the number of values in each of ten successive intervals from 0 to 3, with the total number of values being 1000.

   b. a comparison of the results in (a) to the input exponential distribution.

INSTRUCTIONS TO GRADER

If the student has submitted copies rather than originals of the computer output, state that on the exam answer sheet and **immediately stop grading the exam and give it a grade of zero.**