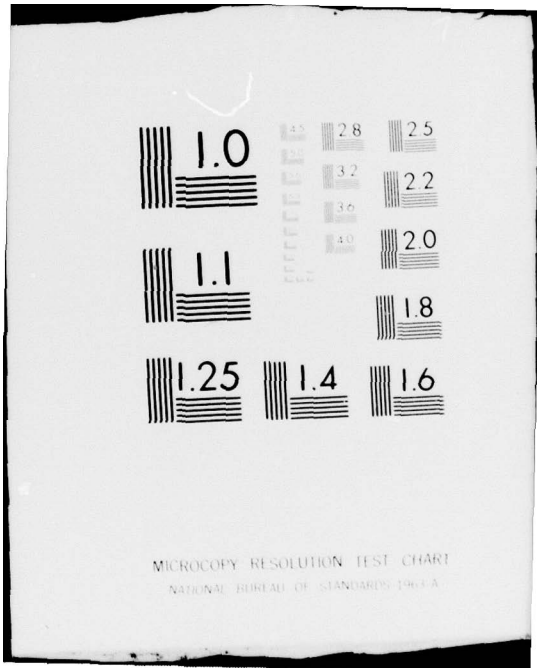
AD-A079 242 ARMY PERSONNEL RESEARCH OFFICE WASHINGTON DC F/G 9/2
COMPUTER GENERATION OF RANDOM NUMBERS.(U)
MAY 65 W D LARKIN
UNCLASSIFIED APRO-RM-65-3 NL

| OF |
AD A079242L

END
DATE FILMED
2-80
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Research Memorandum 65-3

COMPUTER GENERATION OF RANDOM NUMBERS

May 1965

U. S. ARMY PERSONNEL RESEARCH OFFICE

Army Project Number
2J024701A723

MICS b-11

(9) Research Memorandum 65-3

(6) COMPUTER GENERATION OF RANDOM NUMBERS.

(10) By W. D. Larkin

(14) HPRO-RM-65-3

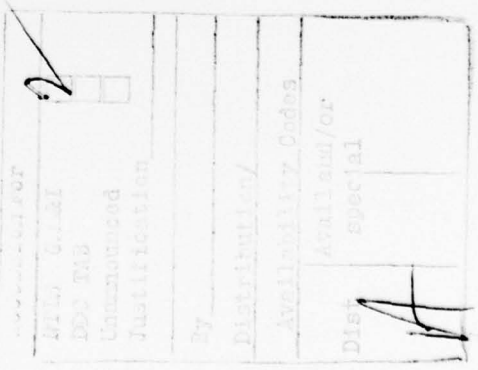
Submitted by:
Cecil D. Johnson
Chief, Statistical Research
and Analysis Laboratory

(12) 18

Approved by:
J. E. Uhlaner
Director of
Laboratories

(11) May 1965

Research Memorandums are informal reports on technical research problems. Limited distribution is made, primarily to personnel engaged in research for the U. S. Army Personnel Research Office.



040 650 Gu

COMPUTER GENERATION OF RANDOM NUMBERS

INTRODUCTION

A random number is a quantity which is specified probabilistically, and a random number generator is a device which outputs numbers under the constraints of probability laws. The probability laws governing the sequence describe (1) the statistical properties of the output sample, and (2) the relationship between successive numbers in the sequence. The present memorandum reports progress in the development of computer techniques for generating random numbers.

The probability law governing the output sample often takes the form of a theoretical probability distribution function which the output as a whole is expected to satisfy. The usual sequential requirement is that successive numbers in the output sequence be statistically independent. There are, of course, numerous possible rules to govern non-independent sequences.

Random numbers have their greatest use in simulation. A random number generator is, in fact, a program to simulate sample distributions. A simulated sample is often very useful in solving psychometric problems, because its properties can be controlled more easily than can those of empirical samples. (See, for example, the study of an allocation technique applied to current aptitude input by Boldt, Wiskoff, and Fitch, 1960.) However, the real utility of random number generators is achieved in simulation of theoretical systems and probabilistic control processes. A typical theoretical system often studied by simulation is a mathematical model of behavior. Reasonable models of behavior often turn out to be mathematically intractable. However, computer programs may be written which simulate the functional operations of the model. It would then be possible, with a judicious choice of parameters, to run many empirical simulations of the model. In this way, the model may be tested so thoroughly that explicit mathematical solutions of the model equations are no longer needed. Random number generators are essential in this endeavor, because most psychological models are written in the language of probabilities.

An example of a probabilistic control process is a psychological experiment in which the stimulus presentations (or the administered rewards) have some chance dependence on other events. Experiments sometimes require that stimuli be chosen "at random", or with some probabilistic dependence on the subject's last response, or at certain "random" intervals in time. For example, studies of multi-person interaction often use a complicated reward structure in which the outcome administered to any individual depends on the joint response pattern of the group as well as upon chance events set up by the experimenter. The probabilistic character of these contingencies can often be achieved by a random number program. In fact, there are many interesting experiments which cannot be done without the computer's probabilistic control.

METHODS

The core of most random number programs is a simple, repeatable arithmetic operation. In almost every program, this core operation produces numbers which are uniformly distributed along the interval from zero to one. This is by no means a necessary first step, but it makes possible a variety of second steps, such as converting uniformly distributed numbers into other distributions. For example, it is much easier to convert uniform random variables into, say, a beta distribution than to construct the beta distribution directly. Therefore, most random number programs will have two parts: a subroutine to generate uniform random numbers and a master program to make them conform to some probability distribution.

Of the many computer methods for generating uniformly distributed numbers, none are perfect. All produce a repeating finite sequence, the terms of which can be perfectly predicted from the initial conditions. Nevertheless, the sequences may be extremely long (so that, for practical purposes, they do not repeat), and they may possess most of the statistical properties of a truly random sequence. For this reason, the numbers produced are sometimes called "pseudo-random". A discussion of alternative methods for generating uniform random numbers is beyond the scope of the present memorandum, but the one most commonly used will be briefly described.[1]

THE POWER RESIDUE METHOD

An arithmetic process that produces a long, erratic sequence can be invented easily; to make the terms independent and equi-probable is much more difficult. The power residue method comes close to satisfying all these requirements. It relies on computations of the residues (remainders) of successive powers of a number, x modulo[2] M, where M is large compared to x. In particular, if x and M are relatively prime[3], the sequence $u_n = x^n \pmod{M}$, or the sequence $u_n = a.x^n \pmod{M}$ where a is also relatively prime to M, is a repeating sequence with random properties.

---

[1] Alternate methods are given in Hamming (1962); IBM Reference Manual C20-8011; and Meyer (1956).

[2] The expression $u = v \pmod{M}$ means that $(u - v)$ is divisible by M. The operation $u.v \pmod{M}$ is read "multiply u by v, divide the product by M, and take the least positive remainder".

[3] Two numbers, u and v, are relatively prime if their greatest common divisor is one.

The residue sequence depends on the relationship between x and M: if $M = 2^b$ for some positive integer b, and if x is a suitably chosen odd integer, there will be 2^{b-2} terms produced before the sequence repeats. This fact has some practical importance. To make the sequence as long as possible in a binary computing machine, it is convenient to choose M to be the maximal word. If there are b "bits" available, the maximal computer word is 2^b. (On the GE 225, $M = 2^{19}$, and the maximal sequence length is therefore 2^{17}.) This choice has the virtue that the division operation in the formation of residues is eliminated: when the computer multiplies x^n by x^{n-1}, the product "overflows" (i.e., $x^{n+(n-1)} > 2^{19}$), and is automatically set equal to the residue.

The power residue method has been programmed, tested, and incorporated in large random number programs. Three versions exist: one written in GE 225 machine language (GAP), one written in FORTRAN for the GE 225, and one written in machine language for the IBM 7090. A statistical analysis of the results has been completed and will be reported separately. All the programs perform well on tests of uniformity and independence.

METHODS FOR CONVERSION

Independent uniformly distributed random numbers are rarely all that is needed to solve a problem. Most simulations require that the numbers satisfy some additional probability law. Therefore, general methods are needed for converting uniform random numbers into other more complex probability distributions. Two conversion techniques can be distinguished: (1) integration methods and (2) composition methods.

Integration Methods. Let f(x) be a continuous probability density function which is being simulated and let {u} be a sequence of uniformly distributed random numbers on the unit interval. The u's are interpreted as percentile scores. For each value of u, we wish to find a number, v, which is at the uth percentile of the distribution f(x). The v's will then have the required f(x) distribution. For example, if f(x) is the standard normal distribution, and if u is 0.975, then the value of v which is greater than exactly $97\frac{1}{2}\%$ of the normal distribution turns out to be v = 1.96. The value 1.96 can be found in the usual tables of the normal distribution (although, of course, tables would not be used with the random number generator). Mathematically, the example from the normal distribution is expressed as follows: If

$$u = 0.975 = \int_{-\infty}^{v} f(x)dx = \int_{-\infty}^{v} (2\pi)^{-\frac{1}{2}} e^{-x^2/2}\, dx,$$

then v = 1.96. More generally, given u, one must be able to find v such that the area under f(x), from its lower limit to v, equals u:

$$u = \int_{-\infty}^{v} f(x)dx \qquad [1]$$

It is this integration problem that gives the method its name.

Whether v is easily determined depends on the density function f. Some functions, such as the exponential,

$$f(x) = \lambda e^{-\lambda x} \quad , \lambda > 0 \qquad [2]$$

or the Laplace,

$$f(x) = \frac{1}{2} \lambda e^{-\lambda |x|} \quad , \lambda > 0 \qquad [3]$$

are easily integrated, and an exact solution to equation [1] can be programmed. This programming has been done for the exponential and the Laplace distributions. Samples from these random number routines can be quickly generated. Because their accuracy depends only on the residue calculations, the samples conform well to theoretical distributions. The exponential distribution occurs often in models of latency mechanisms (e.g., models for visual search time or detection response time). It is also a component of more complex distributions, such as the gamma, and is likely to be one of the most useful random number programs.

The generality of the integration method depends on how easily Equation [1] can be solved. When an exact solution cannot be found, some approximation technique is often useful. This is the case when f is a normal distribution. The most convenient method for integrating the normal distribution on a computer is to use an approximation formula (Hamming, 1962; Ralston and Wilf, 1962). Usually, use of this formula entails calculating the first few terms of an infinite series. The normal distribution subroutine--a FORTRAN program for the GE 225 or IBM 7090--calculates the area under any continuous segment of the normal curve, using an approximation formula. This program is quite fast, as it must be when large samples are needed, and it is accurate: the output agrees with the tabled normal distribution to six decimal places.

When neither an analytic solution nor an approximate formula is available for Equation [1], the integration may be carried out by an iterative procedure. Several iterative techniques were investigated for converting uniform random numbers into a gamma distribution,

$$f(x) = \frac{\lambda}{\Gamma(r)} (\lambda x)^{r-1} e^{-\lambda x} , \qquad [4]$$

where $\lambda > 0$, $\Gamma(r) = \int_0^\infty x^{r-1} e^{-x} dx$, and $r > 0$. All the iterative methods use a series of quadratures of the area under the probability curve. At each step, an ordinate grid is overlayed on the area to be integrated. This grid divides the area into four-sided geometric figures defined by two adjacent ordinates on the grid, the segment of the x-axis between the ordinates, and a line segment near the probability curve chosen to simplify computations. The total area under the curve is approximated by the sum of the small grid areas. Figure 1 illustrates this method. On each iteration, the number of grid lines is increased, making the approximation closer to the true area until two successive iterations give estimates which differ by less than a pre-set tolerance. In the FORTRAN program that uses the method shown in Figure 1, N rectangles are used to determine

$$A_N = \frac{V}{N} \sum_{i=0}^{N-1} f\left[\frac{V}{N} \left(i + \frac{1}{2} \right) \right] \simeq \int_0^V f(x)\, dx \qquad [5]$$

on the first iteration, and $N \cdot 2^{k-1}$ are used on the kth iteration.

The most time-consuming step in computing Equation [5] is calculation of the ordinates, f(i). For the gamma distribution, Equation 4, the GE 225 computer used about 1.5 minutes to compute 1000 values of f(i). The number required to reach a tolerance of 1% varied from about 100 ordinates to 10,000 ordinates, depending on the parameter r. Iterative routines that used Simpson's rule or trapezpoidal approximations were equally time consuming.[4] These results force the conclusion that although iterative routines are useful for integration, they are not fast enough to provide random numbers in quantity. Methods are needed which will not require as many operations. Monte Carlo methods may satisfy this requirement.

A Monte Carlo method is a procedure by which random processes simulate or approximate mathematical functions. The best-known use of random processes in computers is Monte Carlo integration: to find the area under a

---

[4] Ralston and Wilf (1962) give more thorough presentations of iterative quadrature methods.

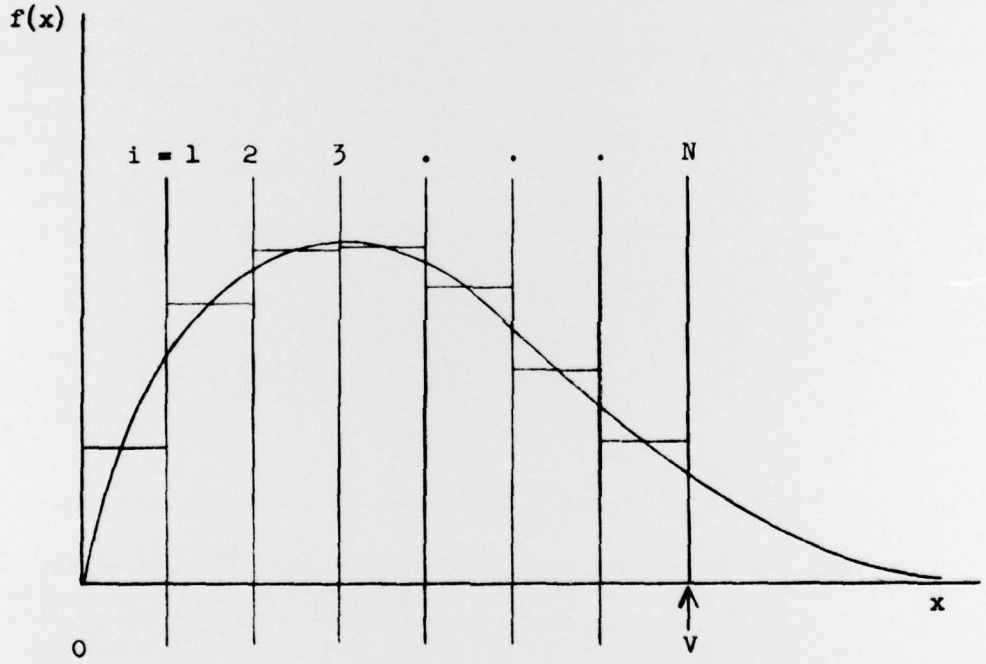


Figure 1. A step in an integration method: approximating $\int_0^V f(x)\,dx$ by a series of rectangles.

curve, the computer scatters points at random in a space of known area which includes the region to be integrated as a proper subspace. The proportion of points falling inside the integration region is the estimate of its area. To generate probability distributions, we use a variation of this technique attributed to J. Von Neumann. Pairs of independent uniform random numbers (x,y) are generated: x in the domain of the probability density function f, and y in its range. Each x is a "candidate" for inclusion in the sample of random numbers having the f(x) distribution. If, for a given x, $f(x) > y$, x is "accepted" in the sample; otherwise x is discarded, and a new (x,y) pair is generated. Figure 2 illustrates this procedure. The proportion of x values included in the sample depends on the shape of f. If the function is highly peaked, its range will be large compared to the probability ordinates at most values of x, and, because a large percentage of y values will then exceed the ordinates, many x values will be discarded. The efficiency of the method depends on the probability that x is not discarded, which is,

$$p(y < f(x)) = 1 - \frac{M \cdot d - \int_D f(x)}{M \cdot d}, \qquad [6]$$

where D is the (interval) domain of f, d is its length, and M is the value of f at its mode.

When this Monte Carlo procedure was used to generate the gamma distribution, computation time per random number was decreased over the iterative methods by a factor of about 1000. However, for certain parameter values, the gamma distribution is highly peaked; and, although computation time is satisfactory, the method is not as efficient as it could be. Efficiency is important, not because of small differences in computation time, but because an inefficient Monte Carlo program may use a large portion of the power residue sequence to generate a small sample. Generating large samples may entail frequent changes in the basic uniform number sequence; these changes should be avoided whenever possible.

To increase the efficiency of the gamma routine, a modified program was written. In this program, the x-domain of f is divided into subintervals of equal length. For each subinterval, values of y are generated only in the subset of the range of f which is supported by the x subinterval. This restriction on the y values increases the efficiency, because the factor M·d (Equation 6) is replaced by a sum of small rectangular areas whose total approaches $\int_D f(x)dx$ as their number increases. With even a small number of intervals, the efficiency is improved by a factor of 2 or 3. Table 1 illustrates the difference between the two methods.

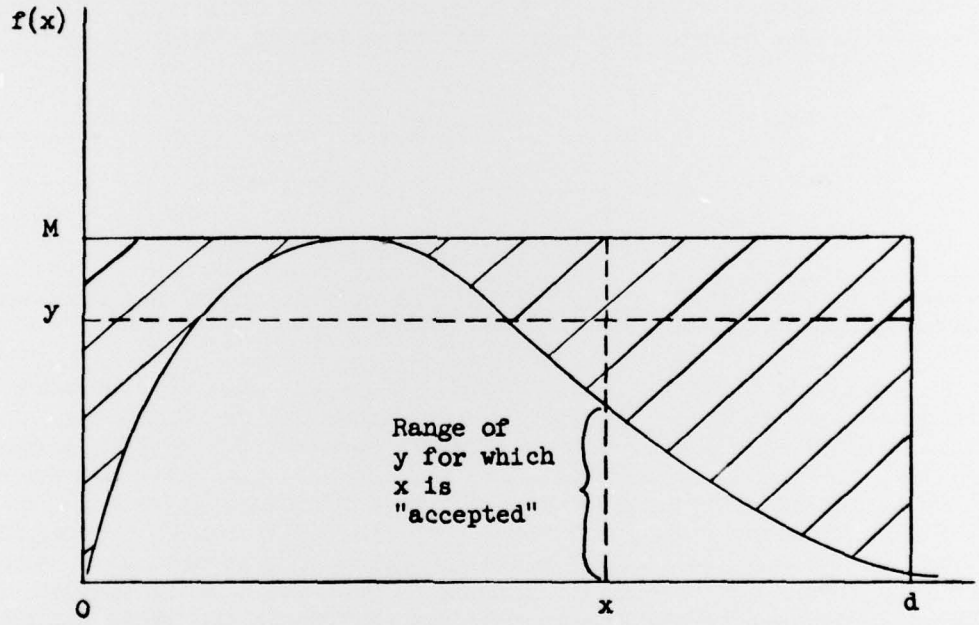


Figure 2. A Monte-Carlo method. A pair of independent random numbers determines a point (x,y). If y falls in the shaded area, x is discarded; if y falls beneath the curve, x is added to the sample.

Table 1

EFFECT OF REFINEMENT ON MONTE CARLO GENERATION OF THE GAMMA DISTRIBUTION[a]

	r	Pairs of Random Numbers Needed[b]	Accuracy of Fit to Gamma Distribution[c]
Original Method	1.0	24,717	-1.57
	3.0	13,521	2.19
	5.0	13,810	-0.35
Modified Method	1.0	6,414	-0.03
	3.0	6,300	-0.73
	5.0	6,000	0.34

[a] Each row of the table represents a simulation run. The parameter r (see Equation 4) determines the "shape" of the distribution. When r is unity, the distribution is highly skewed, with a mode at zero. As r increases, the mode (which is at the point $\frac{r-1}{\lambda}$) moves to the right, and the distribution becomes flatter. In each simulation, λ was unity, and the same random number sequence was used.

[b] The table entry is the number of pairs needed to generate a sample of 5000.

[c] The output was a 100-cell histogram, similar to the one shown in Figure 4. Chi-square goodness-of-fit values are approximately normally distributed when the number of cells is 30 or more. The table entry is the deviation, in standard deviation units, of the computed chi-square value from its theoretical mean. A 95% confidence interval, symmetric about the theoretical mean, would have its boundaries at ±1.96 standard deviations.

Composition Methods. Occasionally, it is easier to convert one distribution into another by a method other than integration. This would certainly be the case if we desired the gamma distribution only for integral values of its parameter r, because the sum of r independent identically distributed exponential random variables has the required gamma distribution. There are many special methods which can be characterized by their reliance on composition operations, such as addition, multiplication, or convolution. In the case of the normal distribution, the central limit theorem, which guarantees the asymptotic normality of sums

of independent random variables, suggests an alternate way to produce normally distributed numbers. The uniformly distributed numbers, $\{x\}$, $0 < x < 1$ have mean $\frac{1}{2}$ and variance $\frac{1}{12} = \int_0^1 (x - \frac{1}{2})^2 \, dx$. If 12 independent values of x are added, the distribution of the sum will have mean 6 and unit variance. This distribution approximates a normal distribution very closely, even though only 12 random variables have been used. Its main drawback is that, being bounded, the fit to the normal is not good in the tails. Figure 3 shows the output of a subroutine which sums 24 uniform variables. This distribution has been transformed to approximate the unit normal by subtracting 12 from each sum and multiplying the result by $\sqrt{\frac{1}{2}}$.

The familiar application of the Poisson distribution to "completely random" events (e.g., telephone traffic) suggests a special method to convert uniform random numbers to a Poisson distribution. Suppose that, in a unit of time, the number n of occurrences of an event (e.g., telephone calls) is a Poisson distributed random variable, i.e.,

$$P(n = k) = \frac{v^k}{k!} e^{-v}, \qquad [7]$$

where $v > 0$, $k = 0, 1, 2, \ldots$

It follows (Feller, 1962) that, if $t_1, t_2, \ldots$ are the times at which the events occur, the inter-occurrence times, $I_1 = t_1$, $I_2 = t_2 - t_1$, $I_3 = t_3 - t_2 \ldots$ are independently exponentially distributed. Thus, if the computer generates a sequence of exponential random variables, $y_1, y_2, \ldots$ and adds them, the smallest value of n such that

$$\sum_{i=1}^{n+1} y_i > 1 \qquad [8]$$

will be Poisson distributed. This method can be simplified by noting that by integrating Equation 2, we can express the y's in terms of uniform numbers;

$$y_i = \frac{-\ln (x_i)}{v} \qquad [9]$$

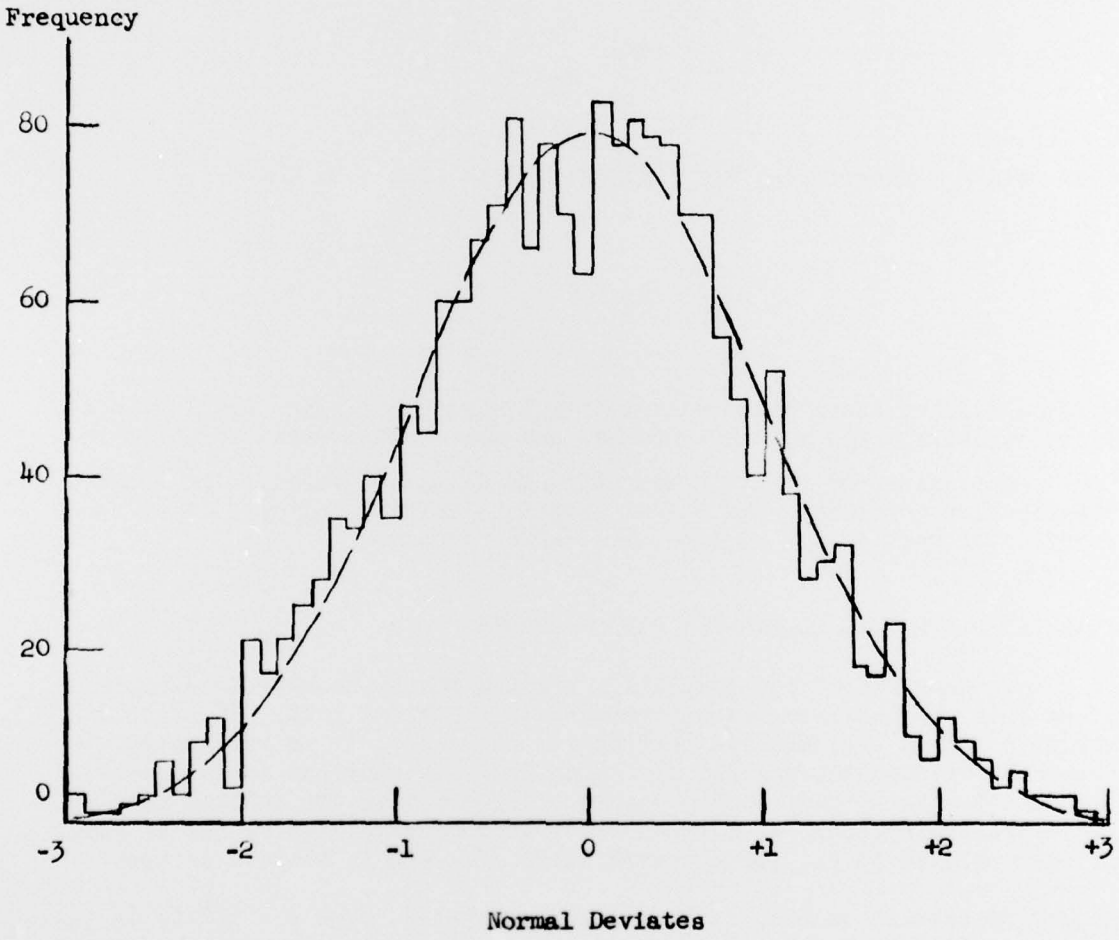


Figure 3. Output histogram of the sum of 24 uniformly distributed random variables. A normal distribution with zero mean and unit variance is drawn through the histogram, which represents a sample of 2000 observations.

where x_i is a uniform random number and v is the parameter of the exponential distribution. Inequality 8 can now be written

$$\sum_{i=1}^{n+1} \frac{-\ln (x_i)}{v} > 1$$

or,

$$\frac{1}{v} \ln \prod_{i=1}^{n+1} x_i < 1$$

and, taking exponentials, the smallest values of n such that

$$\prod_{i=1}^{n+1} x_i < e^{-v} \qquad [10]$$

will be Poisson distributed with mean and variance v. For small values of v, relatively few x_i's are needed, and the routine works efficiently. For larger values of v, the computation is slow, but because the Poisson distribution approaches the normal distribution with increase in v, the inefficient part of the routine need never be used.

CORRELATED AND NON-INDEPENDENT RANDOM NUMBERS

One requirement of a simulation program may be to reproduce correlations that are observed between numbers in empirical data. If, for example, a matrix of simulated test scores is needed, it is very often desirable to simulate the row or column intercorrelations as well as the distribution properties of the scores. Suppose R is the desired matrix of intercorrelations. If a vector A of numbers generated by the computer is uncorrelated (i.e., the expected value of $1/n\ A'A$, where n is the number of sets of values, is the identity matrix), then $B = AR^{\frac{1}{2}}$ is a transformation of A which has the same distribution properties, but its expected intercorrelation matrix is

$$\begin{aligned} B'B &= R^{\frac{1}{2}} A' A R^{\frac{1}{2}} \\ &= R^{\frac{1}{2}} R^{\frac{1}{2}} \\ &= R. \end{aligned} \qquad [11]$$

Thus, by a simple multiplication operation--but one that often could not be done without a high-speed computer--the random number output reflects the intercorrelations of the empirical data. This method, and the adequacy of the programs using it, are discussed in Boldt, Wiskoff, and Fitch (1960).

The correlation adjustment leaves successive score vectors statistically independent, because the parameters of the generating process are left unchanged. When dependent random numbers are required, the generating algorithm must adjust its parameters on trial (or time period) t as a function of the output sequence through trial t - 1.

Probability mechanisms with time-dependent parameters--known as stochastic processes--are common elements of simulation models. The simplest sequential adjustments of parameters are those that depend only on the trial numbers, t, and not on the magnitude of the random numbers themselves. For example, suppose an exponential random number generator is altered so that the parameter λ is proportional to the unfilled portion of the desired sample. If N is the total sample size, the first value of λ is $\lambda_1 = kN$, for some fixed k. On the second trial, only N - 1 numbers are required, and $\lambda_2 = k\ (N - 1)$. Continuing in this way we have,

$$\lambda_t = k(N - t + 1),\ t = 1,2,\ldots N. \qquad [12]$$

Because λ becomes smaller each time a number is added to the sample, successive numbers are drawn from exponential distributions (Equation 2) with steadily increasing means (since the mean of the exponential distribution is inversely related to λ). If we identify the random number output with, say, response latency, the generator is a model of an organism which responds at a decreasing rate. This program has found useful application in simulation of visual search.

CONCLUSIONS

The techniques just described are building blocks for a wide variety of random-number programs. In each case, the method has been programmed and tested for efficiency and accuracy. A chi-square testing program was written to evaluate the goodness of fit of the generated data to theoretical functions. In the case of the gamma distribution (which involved the most extensive programming work), typical chi-square values for samples of 10,000 numbers were significant beyond the 0.01 probability level. This indicates that Monte Carlo techniques preserve the basic accuracy of the uniform power residue core program and can therefore be expected to operate satisfactorily in future applications.

To evaluate a random number program, two questions must be asked: "Do the data fit the desired distribution?" and "Are successive numbers independent?". All considerations of program efficiency are secondary to these basic goals, because, in application, no amount of computation speed can compensate for error, bias, or unwanted sequential dependencies. The independence question is at once the most difficult to answer and the most common source of trouble. The problem may become acute when successive random numbers enter into recursive or progressive computations that magnify error. Basically, the difficulty stems from systematic variations in the uniform core program. Although the power residue method is the best available, it is known to have cyclic and compensatory properties that depend on the initial number and multiplier chosen. The effect of these regularities in a sequence that ought to be irregular is not always easy to detect. Figure 4 shows one instance when the malfunction was discovered immediately, because the output histogram was composed of relatively small class intervals. If the interval size had been doubled, the fault could have escaped notice. In principle, the mathematics of number theory could be used to discover which choices of initial conditions produce the "best" random sequence, but the work would be so complex and tedious that no one is likely to attempt it. The only recourse is to trial and error. Attempts to reduce sequential effects by composition of independent generators have been successful (Maclaren and Marsaglia, 1965).

Occasionally, random numbers are needed in large quantities, rather than one by one, and the principal requirement is that they closely approximate a distribution function. Some integration problems have this requirement. The independence of successive numbers in the sequence can be sacrificed if the total sample distribution can be made precise. If the desired distribution is uniform, or is based on an underlying uniform sample, the output of the random number generator can be improved in this respect. For example, if a sample of 1000 numbers is desired in a uniform distribution over the unit interval, it can be achieved by selecting each number in the list 0.001, 0.002,...1.000 exactly once. A program, called RANPER, has been written to make such selections in a random order. Briefly, this program uses the power residue subroutine to randomly select the numbers, one at a time; each selection is then removed from the list of available numbers and the final output is a permutation of the list. Because the numbers are selected without replacement, the selection probabilities change as the list diminishes. Consequently, the output is no longer random: it is a haphazardly constructed, but perfectly uniform sample.

The diverse applications of random number generators make it likely that new techniques will be found and put to use in future work. To date, there have been more suggested applications for these programs than the organization has resources to pursue. The present memorandum thus reports only the beginning of a growing research field. Whether the new developments will simply be more efficient refinements of basic techniques or fundamentally different approaches is a question that will be answered only with continued investigation.

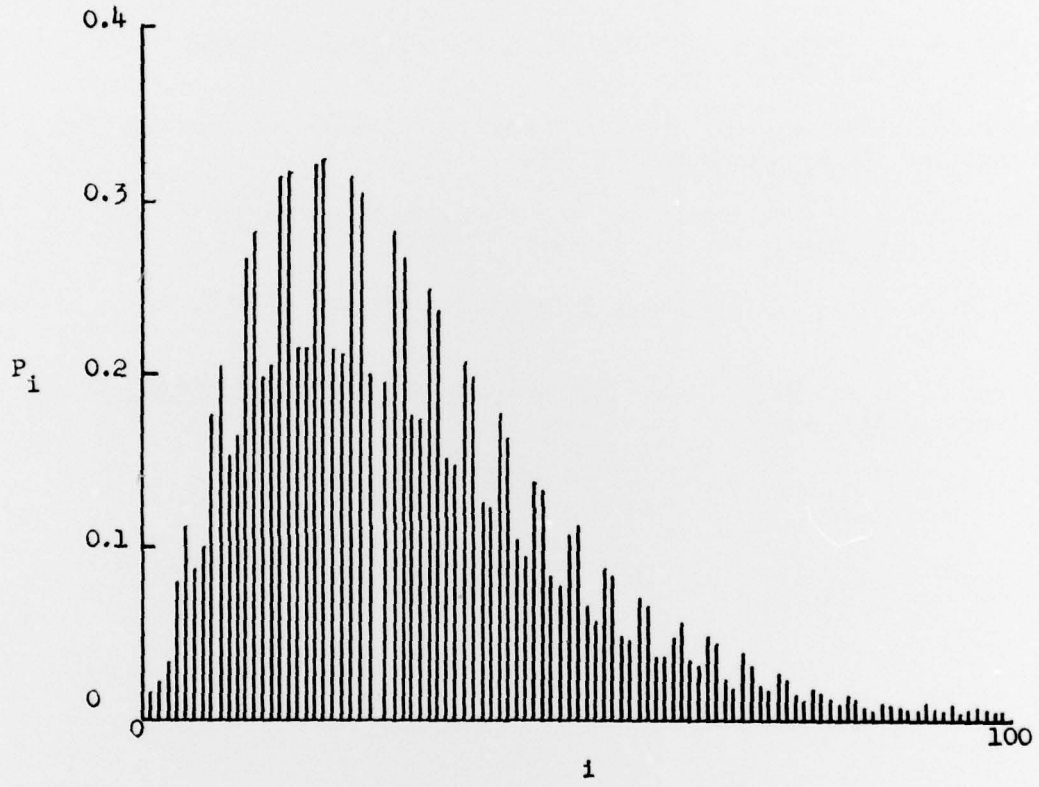


Figure 4. Output histogram from the gamma distribution generator. Each vertical line represents the sample proportion P_i in the ith class interval. The parameters for this run were $\lambda=1$, $r=3$, and the sample size was 5000. The histogram shows the effect of a pronounced regularity in the power residue computations for the basic uniformly distributed random numbers: the sample proportions should conform to a function that lies between the tall and short segments.

REFERENCES

Boldt, R. F., Wiskoff, M. F., and Fitch, D. J. An allocation technique applied to current aptitude input. USAPRO Research Memorandum 60-19. November 1960.

Feller, W. An introduction to probability theory and its applications. Vol. 1, (2nd. ed.), New York: Wiley, 1962.

Hamming, R. W. Numerical methods for scientists and engineers. New York: McGraw-Hill, 1962.

International Business Machines Corporation. Random number generation and testing. Reference Manual C20-8011.

MacLaren, M. D. and Marsaglia, G. Uniform random number generators. J. Assoc. Comp. Mach., 12, No. 1, 1965, 83-89.

Meyer, H. A. (Ed.) Symposium on Monte Carlo methods. New York: Wiley, 1956.

Ralston, A. R. and Wilf, H. S. Mathematical methods for digital computers. New York: Wiley, 1962.