NOTES

EXPRESSING A RANDOM VARIABLE IN TERMS OF UNIFORM RANDOM VARIABLES

By G. Marsaglia

Boeing Scientific Research Laboratories

Summary. This note suggests that expressing a distribution function as a mixture of suitably chosen distribution functions leads to improved methods for generating random variables in a computer. The idea is to choose a distribution function which is close to the original and use it most of the time, applying the correction only infrequently. Mixtures allow this to be done in probability terms rather than in the more elaborate ways of conventional numerical analysis, which must be applied every time.

Introduction. We are concerned with procedures for generating sequences of numbers which will serve as independent determinations of a random variable x with specified distribution function F. Currently, the most satisfactory method is to use an arithmetic procedure for generating a sequence of numbers $u_1, u_2, \cdots$ which serve as "independent" determinations of a uniform (0, 1) random variable, and then to generate the required x in terms of the u's. There is no, or very little, probability theory concerned with the u's—they are generated recursively, say by putting $u_{i+1} = \alpha u_i + \beta \pmod m$, where α, β and m are chosen to make the resulting sequence meet the user's requirements for "randomness." See references [1] and [2].

If we are willing to grant the adequacy of such procedures and take as our starting point a sequence

$$u_1, u_2, u_3, \cdots$$

of independent uniform (0, 1) random variables, then we may use some probability theory in searching for methods for expressing a random variable x with distribution function F in terms of the u's, guided, of course, by the suitability of such methods for use in programs for digital computers. A summary of existing methods for generating a normal random variable is given by Muller in [3]. We will not go into details of the various methods on record, but point out that the fastest method in Muller's summary is one of his own [4] that takes, using a unit familiar to programmers, about 120 cycles and provides F to within a certain accuracy, while programs based on the methods outlined below will be more accurate and have average running times on the order of 15-20 cycles.

Methods. Suppose we have a method M_1 for providing a random variable y_1 with distribution function G_1, and method M_1 takes 10 cycles. Suppose we also have a method M_2 for providing a random variable y_2 with distribution function G_2, and method M_2 takes 500 cycles. If we can represent F as a mixture of G_1

Received November 8, 1960.

and G_2, say

$$F(a) = .99G_1(a) + .01G_2(a),$$

then we may use $u_1, u_2, u_3, \cdots$ to provide a random variable x whose distribution is F as follows: if $u_1 < .99$, we use method M_1 on $u_2, u_3, \cdots$ to generate y_1 and put $x = y_1$. If $.99 \leqq u_1 \leqq 1$, we use method M_2 on $u_2, u_3, \cdots$ to generate y_2 and put $x = y_2$. The average running time will then be $(.99)10 + (.01)500 = 14.9$ cycles, plus the time necessary to test $u_1 < .99$.

This illustrates the basic principle of the simple device which provides programs with very short average running times—we represent F as a mixture of distributions,

$$F(a) = p_1G_1(a) + p_2G_2(a), \tag{1}$$

in such a way that p_1 is close to 1 and the time to generate a random variable y_1 with distribution G_1 is small; then most of the time we put $x = y_1$. Even though G_2, the correcting distribution, may be quite complicated and difficult to handle, we still have a short average running time, since G_2 must be handled so infrequently.

In searching for representations of F such as (1), if we have a G_1 which shows promise, then we find the largest p_1 so that $F - p_1G_1$ is monotone. It seems better to work with densities, and find p_1 so that $f(x) - p_1g_1(x)$ is non-negative. Furthermore, if we make g_1 a rectangle or a mixture of rectangles, we can expect to have very short running times, especially if the rectangles are chosen so as to exploit the particular features of the computer in question.

We give some examples which will serve to illustrate the above remarks.

EXAMPLE 1. Let

$$h(x) = ce^{-\frac{1}{2}x^2} \int_0^{(9-x^2)^{\frac{1}{2}}} e^{-\frac{1}{2}y^2}\, dy, \qquad 0 \leqq x \leqq 3. \tag{2}$$

We will see in Example 2 how h may be used in generating a normal random variable. Suppose we want to express a random variable z with density h in terms of the members of the sequence $u_1, u_2, u_3, \cdots$. We write

$$h(x) = \frac{1832}{2048}h_1(x) + \frac{164}{2048}h_2(x) + \frac{52}{2048}h_3(x),$$

where the terms in the mixture are drawn in Figure 1, h_1 and h_2 are mixtures of rectangles and h_3 is the residuum. Then:

(i) The rectangles which make up h_1 have base $\frac{1}{8}$ and altitude an integral multiple of $\frac{1}{32}$. A random variable z_1 with density h_1 may be formed by putting $z_1 = a_i + \frac{1}{8}u_2$ where a_i is chosen with probability $\frac{1}{256}$. z_1 may be formed in about 10 cycles.

(ii) The rectangles in h_2 all have the same area, so that a random variable z_2 with density h_2 may be formed by putting $z_2 = c_i + b_iu_2$, where the pairs (c_i, b_i) are chosen with equal probability. Time, about 20 cycles.

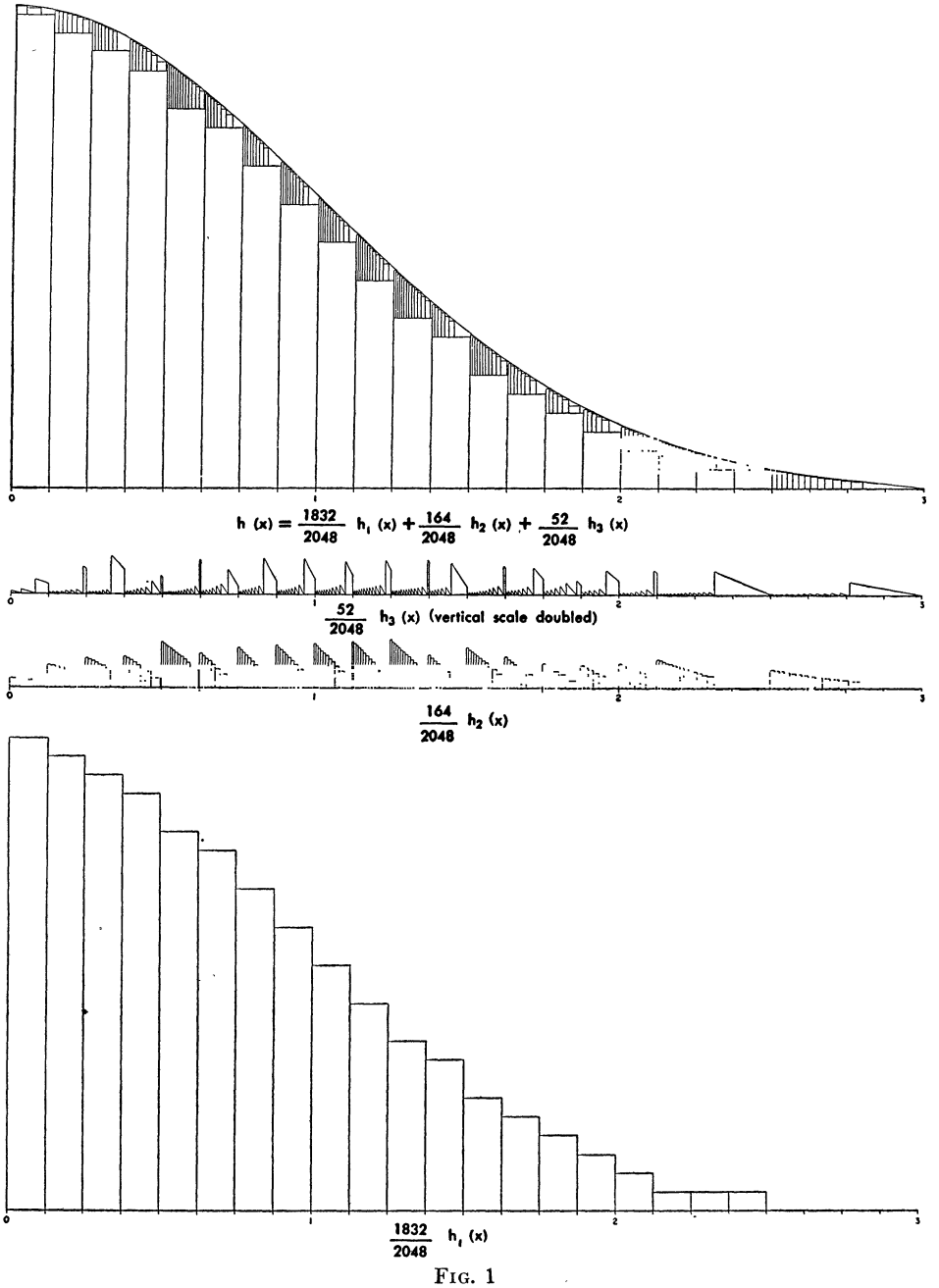
0
1
2
3
h (x) = 1832/2048 h1 (x) + 164/2048 h2 (x) + 52/2048 h3 (x)
52/2048 h3 (x) (vertical scale doubled)
164/2048 h2 (x)
1832/2048 h1 (x)

FIG. 1

(iii) It is more difficult to generate z_3 with density h_3. Most of the "teeth" in h_3 may be replaced by triangles, so that $z_3 = e_i + d_i \min(u_2, u_3)$ will do, where the pair (e_i, d_i) is chosen with a certain probability. The probabilities associated with these pairs are all different so that the computer must spend some time finding the appropriate one, and a few other parts of h_3 must be handled by even slower methods, but the entire procedure shouldn't take more than 200 cycles.

Thus, to generate a random variable z with density h, we use u_1 to choose one of the above methods, generate a number, and call it z. The average time is around 15 cycles.

EXAMPLE 2. The normal distribution. We describe a method for generating a random variable x with the absolute normal density:

$$f(x) = \frac{2}{(2\pi)^{\frac{1}{2}}} e^{-\frac{1}{2}x^2}, \qquad 0 \leqq x.$$

A random $\pm$ may be attached later. The tail of f offers some difficulties which we avoid by using a suggestion of D. MacLaren. If x and y are independent with density f then the distribution of x, given that $x^2 + y^2 \leqq 9$, is h in (2). Hence we put $p = 1 - e^{-4.5} \cong .989$ and write

$$f(x) = ph(x) + (1 - p)t(x),$$

where

$$t(x) = \frac{1}{1 - p}[f(x) - ph(x)].$$

To generate a random variable x with density f in terms of $u_1, u_2, \cdots$, we test: is $u_1 < p$? Then:

(j) If $u_1 < p$, use the method of Example 1 on $u_2, u_3, \cdots$ to generate x.

(jj) If $p \leqq u_1 < 1$, put

$$x = u_2 \left(\frac{9 + 2\rho}{u_2^2 + u_3^2}\right)^{\frac{1}{2}}, \tag{3}$$

where ρ has the exponential distribution and $u_2^2 + u_3^2 \leqq 1$. The right side of (3) has density t. We may use any of several methods for generating ρ. The time necessary to generate x in this way is relatively long, but we can afford to be extravagant since we use this method only 1 percent of the time.

Remarks. The quick parts of the mixtures above are based on representing a random variable in the form $a + bu$ where a and b are discrete random variables and u is uniform on the interval (0, 1). It is easy to show that any random variable may be so represented, in much the same way as the fundamental result in analysis that a measurable function is the limit of a sequence of simple functions. The problem is to choose the discrete distributions of a and b in a suitable way to ensure short running times, consistent with the number of

storage locations which can be allotted for the program. Only a certain number of the probabilities for the values of a and b can be stored; if a and b are to have an infinite set of values, then the machine must compute the probabilities from some point on, or else a and b can be assigned a finite set of values and then the residual portion of the distribution can be treated by other means, as in the case of h_3 of Example 1.

At any rate, there is a wide variety of reasonable ways of assigning distributions to a and b. The one chosen in Example 1 requires a moderate amount of storage, less than 1000 locations, and is quite fast. It can be made even faster by increasing the number of rectangles in h_1 or h_2, at the expense of additional storage space. An assignment of distributions for a and b which requires less storage space than that suggested above is suggested by the rectangles in Figure 2. The idea there is to let $x = u, u + 1, \frac{1}{2}u, \frac{1}{2}u + 1, \frac{1}{2}u + 2, \cdots$, with the greatest frequencies possible. A program based on that resolution of h generates a random variable x with density h by putting $x = u$ about 48 percent of the time, $x = u + 1$ about 11 percent of the time, $x = u/2$ about 11 percent of the time, and so on.

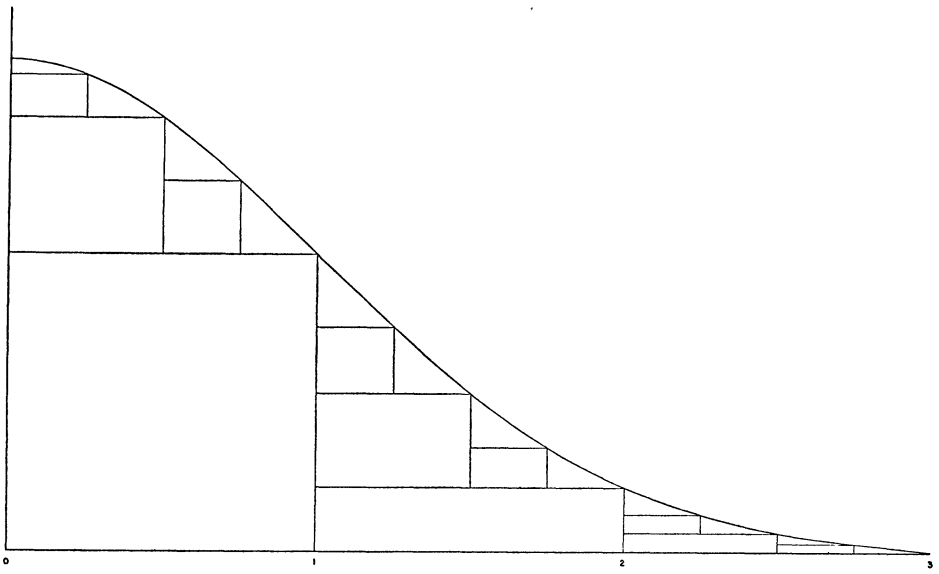

FIG. 2

REFERENCES

[1] O. TAUSSKY AND J. TODD, "Generation and testing of pseudo-random numbers," *Symposium on Monte Carlo Methods*, John Wiley and Sons, New York, 1956, pp. 15–27.

[2] A. ROTENBERG, "A new pseudo-random number generator," *J. Assn. Comp. Mach.*, Vol. 7 (1960), pp. 75–77.

[3] MERVIN E. MULLER. "A comparison of methods for generating normal deviates on digital computers," *J. Assn. Comp. Mach.*, Vol. 6, (1959), pp. 376–383.

[4] MERVIN E. MULLER, "An inverse method for the generation of random normal deviates on large-scale computers," *Math. Tables Aids Comp.*, Vol. 12 (1958), pp. 167–174.