The Alias and Alias-Rejection-Mixture Methods for Generating Random Variables from Probability Distributions

Richard A. Kronmal, Ph.D.,
Arthur V. Peterson, Ph.D

Department of Biostatistics,
University of Washington
Seattle, Washington

Abstract

The alias-rejection-mixture method is a general and exact method for the computer generation of random variables from an arbitrary discrete, continuous, or mixed probability distribution. The method is based on two ingredients: (1) Walker's alias method, an ingenious and efficient method for generating discrete random variables, and (2) the rejection-mixture method, which is a modification of the standard rejection method that eliminates the need to repeat steps of the algorithm. The generation of random variables by the alias-rejection-mixture method requires operations that are simple and few and, remarkably, approximately the same for all distributions.

1. INTRODUCTION

This paper reviews the recent work on a new general exact method for generating random variables from an arbitrary discrete, continuous or mixed distribution. This method, called the alias-rejection-mixture method, requires for each random variable generated operations that are simple and few and furthermore that are approximately the same for all distributions.

In Section 2 we review some of the most commonly used general methods for the generation of continuous or discrete random variables and indicate their relation to the alias-rejection-mixture method. Section 3 reviews both the alias method, the discrete version of alias-rejection-mixture method, and the fundamental theorem that demonstrates its generality. The alias method is combined with the rejection-mixture method to give rise to the alias-rejection-mixture method, as described in Section 4. Also discussed in Section 4 is an efficient variant of the method called the uniform alias-rejection-mixture method. Finally some concluding remarks on open issues and areas for further work are made in Section 5.

2. BACKGROUND

2.1 A short review of basic methods used in the computer generation of random variables

Four basic methods are used, often in creative ways, in almost all algorithms for the computer generation of random variables from a specified distribution $F(x) \equiv \Pr(X \leq x)$. The methods rely on the initial generation of one or more pseudo-independent uniform random numbers which are conveniently generated on a computer by various methods, notably by the linear congruential method, e.g., Knuth (1) or Ahrens and Dieter (2), on most computers.

2.1.1 The inverse-distribution (ID) method

For each realization desired, perform the following steps:

ID1. Generate $U \sim$ uniform $(0,1)$

ID2. Set $X \leftarrow F^{-1}(U)$

It is easily shown that, for F strictly increasing with no jumps, the random variable X generated by this method has the desired distribution $F(\cdot)$. Further, the method is easily modified to work for

CH1437-3/79/0269-0280$00.75
1979 Winter Simulation Conference

arbitrary distributions $F(\cdot)$ without the above restriction.

Unfortunately, for only a few distributions does this simple method, when used alone, yield efficient algorithms. For many distributions (e.g., normal, t, gamma, beta) the inverse-distribution function $F^{-1}(\cdot)$ is a computationally-difficult integral. Even for distributions for which the inverse-distribution function $F^{-1}(\cdot)$ is easily expressible (e.g., $F^{-1}(U) = -\ln(1 - U)$ for the unit exponential distribution), the algorithm based on it may not be efficient. (In the exponential example, execution of the logarithm function involves a series computation that is relatively time-consuming.)

2.1.2 The mixture method

Choose a mixture

$$F(x) = \sum_{i=1}^{n} p_i \cdot F_i(x),$$

where

$$\sum_{i=1}^{n} p_i = 1,$$

$F_i(\cdot)$ distribution functions, to represent $F(\cdot)$. For each realization desired, perform the following steps:

M1. [Choose at random an element of the mixture]

Generate I ~ discrete $(p_1, p_2, p_3, \cdots p_n)$

M2. [Generate a variable from the chosen element of the mixture]

Generate $X \sim F_I(\cdot)$ (independent of of I)

It is profitable to choose a mixture so that the frequently-chosen distributions $F_i(\cdot)$ (i.e., those for which the p_i's are large) are easy to generate. The integer variable I is generated [from a uniform (0,1) variable] by one of the discrete-variable generating techniques (topic to be discussed later), and the variable $X \sim F_i(\cdot)$ is generated from one or more uniform (0,1) variables by some convenient technique using other methods. Commonly, the mixture is chosen so that a frequently-chosen distribution $F_i(\cdot)$ is particularly simple to generate, for example, either a uniform distribution or a triangular (i.e., "sum of two independent uniform random variables") distribution.

Algorithms based on the mixture method have been proposed for various distributions (e.g., normal, t, Cauchy, etc.), with the chosen mixture tailored (often cleverly) to the desired distribution. In these algorithms there are typically one or two high-probability elements of the mixture that are easy to generate (e.g., uniform, or triangular, densities). The other elements of the mixture are typically "leftover" odd-ball distributions, for which the rejection method is often chosen.

The mixture method is of particular interest since it is one of the building blocks of both the alias method and the alias-rejection-mixture method, the subjects of this paper. However, unlike the use of the mixture method in other applications, the alias method and alias-rejection-mixture (ARM) method provide a general prescription for the specification of the mixture that is applicable (and routinely and efficiently applied) to most distributions.

2.1.3 The (acceptance-) rejection method

To generate random variables from a density (or probability mass) function $f(\cdot)$, choose any function $g(\cdot) \geq f(\cdot)$. For each realization desired, perform the following steps as often as necessary for acceptance:

R1. [Prepare for the acceptance-rejection step.]

(a) Generate a variable X from the density $c \cdot g(\cdot)$.

(b) Generate U ~ uniform (0,1), independent of X.

R2. [Acceptance-rejection].

(a) If $U \leq \frac{f(X)}{g(X)}$, accept X.

(b) Otherwise, repeat steps R1 and R2.

The strategy for obtaining an efficient algorithm is to choose a dominating function $g(\cdot)$ that

(a) (when normalized) is a density (such as a uniform density) from which random variables are easily generated, and

(b) is close to the density $f(\cdot)$, [i.e., has a probability of acceptance (= c) of close to 1].

Unfortunately, these two aims are often conflicting, and often a dominating function that is a compromise between these two aims is the best choice.

Discussed later in this paper (Section 4.1) will be a modificaton of the rejection method that avoids the need to ever repeat steps of the algorithm (and thus increases speed). This new method will be the cornerstone of the alias-rejection-mixture method.

2.1.4 Methods using special functions

Some of the common parameteric families of distributions have special properties that allow the generation of random variables from some special function of one or more uniform random variables. These methods are often clever, and occasionally even competitive with other methods. Their main limitation is that each is applicable only to a certain very restricted family of distributions.

There are several other basic methods that might have been added to the list of basic methods, but are not. In addition to the the four basic methods described above, there are several other important basic methods: the comparison method [von Neumann (3) , Forsythe (4)], the method of polynomial sampling [e.g., Ahrens and Dieter (5)], the ratio-of-uniforms method [Kinderman and Monahan (6)], and the rectangle-wedge-tail method [Marsaglia, MacLaren and Bray (7)]. These methods, which have been successfully applied to several specific distributions, are not described here only in the interest of brevity.

2.2 Developments to date in the use of the basic methods

A very brief overview of how these methods have been used to date in developing algorithms for generating random variables from various distributions is now given, first for discrete distributions, then for continuous distributions.

2.2.1 Discrete distributions

The developments for discrete distributions, except for a few specialized algorithms, have been largely along the following two general lines:

(a) The direct use of the inverse distribution (ID) method

Consider the inverse distribution method for the discrete distribution $p(x)$, where for simplicity of exposition, and without loss of generality, the mass points of the distribution are taken to be the integers $x = 1,2,\ldots,n$. For discrete distributions, the inverse distribution method generates the integer $x = i$ if $F^{-1}(U) = i$; that is, $x = i$ is generated iff

$$P_{i-1} < U \leq P_i \qquad (1)$$

where

$$P_i \equiv \Pr(X \leq i) = \sum_{x \leq i} p(x), \quad i = 1,2,\ldots,n.$$

As seen from expression (1), the inverse distribution method for a discrete distribution thus involves searching for the interval $I_i \equiv (P_{i-1},P_i]$ in which the uniform $(0,1)$ variable U lies. The speed of the inverse distribution method thus depends on the speed of the search. Two points can be made in this regard. First, the speed of the method generally decreases as the number of discrete values n of the distribution (which is the number of intervals I_i , $i = 1,2,\ldots,n$ that are searched) increases. Second, the speed can depend markedly on the search strategy employed. The commonly used linear search, in which intervals I_i are searched sequentially $i = 1,2,\ldots,n$, (by first comparing U with P_1, then P_2, and so on, and setting $x = i$ as soon as $U < P_i$), is not the fastest, even when the event probabilities $p(1)$, $p(2)$ are renumbered so that $p(1) \geq p(2) \geq p(3)\ldots$ A faster search procedure is described by Atkinson (8). Even for this method, the speed decreases as the number of discrete values of the distribution increases.

(b) The use of mixture methods

One clever mixture method is Marsaglia's (9) table method. This method will not be described here, except to say that it is based on the b-ary representation of the probabilities $p(1),p(2),\ldots,p(n)$, where b is a selected integer base, usually $b = 2^m$, m integer. The speed of Marsaglia's table method depends both on the number of outcomes of the distribution and the actual distribution $p(\cdot)$. Depending on the discrete distribution $p(\cdot)$, the average number of comparisons may be as small as 1 or as large as approximately $\log_b(n)$. This method requires large tables. Ahrens and Dieter (5) discuss the performance of this method for the Poisson distribution.

A recent dramatic advance in the generation of random variables from discrete distributions is a clever method discovered by Walker (10,11,12) for computer generation of pseudorandom variables from an arbitrary discrete probability distribution with a finite number of outcomes. This method is descibed below in Section 3.

2.2.2 Continuous distributions

For continuous distributions, the developments have in general been along the lines of combining the basic methods in ingenious ways that result in simple and/or efficient algorithms for certain distributions.

Many innovative methods have been discovered by creatively combining various of the tools of the inverse-distribution method, the mixture method, the rejection method, the special-function method, and the other basic methods in ways suitable to the specified distribution under consideration.

A drawback of almost any of these methods proposed to date for generating random variables from continuous distributions is that it is applicable only to one, or a few, distributions. For the practicing researcher using Monte Carlo and simulation methods, this means that many different algorithms, each applicable to a narrow class of distributions, must be obtained, tested, and maintained.

Some effort in the direction of efficiently applying one method to a wide class of distributions has been accomplished, notably:

(a) applications of the von Neumann comparison [Forsythe (4)] method to those densities of the exponential family $f(x) = c \cdot e^{-B(x)}$, where $B(x)$ is an increasing function of $x \varepsilon (\emptyset, \infty)$. Applications of this method include the exponential distribution [Forsythe (4), Ahrens and Dieter (5), and Ahrens and Dieter (13)], the normal distribution [Forsythe (4), Brent (14), Ahrens and Dieter (5), and Ahrens and Dieter (13)], and the beta and gamma distributions [Atkinson and Pearce (15)]. Unfortunately, a method for routinely applying the method to distributions in the (restricted) exponential class of densities has not yet been discovered and programmed; implementations of the comparison method have to date been done on a case-by-case basis, albeit all based on the comparison method.

(b) applicatons of the rectangle-wedge-tail [Marsaglia, MacLaren, and Bray (7)] method. This method is one that appears, at least in principle, to be of wide applicability (much wider than the von Neumann-Forsythe method). It is based on expressing the distribution as a mixture of numerous elements, most of which are equiprobable uniform distributions. The determination of the mixture requires a good deal of effort, and as far as we know it has been applied only to the normal distribution [Marsaglia, MacLaren, and Bray (7); Knuth (1,)], and the exponential distribution.

We will describe in this paper the alias-rejection-mixture method [Kronmal and Peterson (16)], which is efficient and routinely applied to almost any continuous (or mixed) distribution. We feel that this method offers considerable promise of filling a need for a general, routinely applicable, yet efficient, method for generating random variables from continuous distributions.

3. THE ALIAS METHOD

The alias method, due to Walker (10,11,12), is a clever new method for efficiently generating random variables from any discrete distribution with a finite number of outcomes. This method is related to rejection methods, but is better because the "rejected" random numbers are not discarded but instead replaced by "aliases." The alias method appears to be a significant improvement over previous methods for generating discrete random variables, especially in simplicity and speed. This method is a special case of the rejection-mixture method, to be discussed in section 4.1 below, which is the cornerstone of the alias-rejection-mixture method, the subject of this paper.

A modified version of the alias method can be described in terms of mixtures [Kronmal and Peterson (17)]. The following theorem about representing discrete distributions as equiprobable mixtures states the relevant result on which the alias method is based. For convenience and without loss of generality, the mass points of the discrete distribution $p(x)$ will be $x = 1,2,\ldots,n$.

Theorem

Any discrete distribution $p(\cdot)$ with a finite number (n) of outcomes can be expressed as an equiprobable mixture of n 2-point distributions $q_i(\cdot)$, $i = 1,2,\ldots,n$. Furthermore, the elements $q_i(\cdot)$, $i = 1,2,\ldots,n$ can be numbered such that, for $i = 1,2,\ldots,n$, the value i is a mass point of $q_i(\cdot)$.

The proof, given in (17), involves demonstrating the construction of a set of n 2-point distributions $q_i(\cdot)$, $i = 1,2,\ldots,n$, that are elements of such a mixture, and thus provides a prescription for finding and expressing the $q_i(x)$ in terms of the specified distribution $p(x)$.

Once a representation of p(x) in terms of an equiprobable mixture

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} q_i(x)$$

where the $q_i(x)$ are 2-point distributions, has been determined, it is used by the alias method to generate random variables from $p(\cdot)$ as follows:

The Alias algorithm

For each random variable desired from the specified distribution

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} q_i(x),$$

perform the following two steps:

A1. [Choose at random an element of the mixture.]

Generate I ∿ uniform (1,2,...,n).

A2. [Choose at random one of the two values of the distribution $q_I(\cdot)$.]

(a) Generate U ∿ uniform (Ø,1), independent of I.

(b) If $U \leq q_I(I)$, return I. Otherwise, return A_I.

The $q_i(i)$ and A_i, i = 1,2,...,n, are constants that describe the 2-point distributions $q_i(\cdot)$, i = 1,2,...,n whose equiprobable mixture is equal to the specified distribution $p(\cdot)$. The fractional constant $q_i(i)$ is the probability of the mass point i assigned by the 2-point distribution $q_i(\cdot)$, and the integer constant A_i, called an "alias," is the mass point of the 2-point distribution $q_i(\cdot)$ that is not i. A total of 2n storage locations are needed for the constants $q_i(i)$ and A_i, i = 1,2,...,n, required by the method.

Note that the discrete uniform (1,2,...,n) random variable I is returned by the method unless the comparison in Step A2b fails (i.e., "rejection" occurs). If the comparison fails, the alias A_I is returned.

The beauty of the alias method is that, in sharp contrast to the various discrete search procedures used in the inverse-distribution method in common use for discrete distributions, only one comparison is needed per variable generated. Thus, regardless of the number of mass points of the specified discrete distribution, the method is very efficient: once the table of 2n constants is computed, which can be done with a number of operations proportional to n, the total of the operations required to generate each random variable are:

(a) generation of two uniform random numbers [or only one if the uniform (Ø,1) variable and the uniform (1,2,...,n) variable are stripped from one uniform (Ø,n) variable],

(b) either one (if "acceptance" in the comparison step), or two (if "rejection") table look-ups, and

(c) one comparison.

In summary, the alias method for generating random variables from an arbitrary specified discrete probability distribution is exact and requires a very few number of steps. Generation of the tables, performed only once, is proportional to only the first power of the number of values of the discrete distribution. The generation of the random variables requires, in addition to the generation of one uniform random variable, only one comparison and either one or two table look-ups (regardless of the number of mass points in the distribution!). The method, in general, requires fewer operations than other methods suggested to date, including specialized methods for generating random variables for common families of discrete distributions such as binomial, Poisson, and geometric distributions. Of course, since the table-generating portion of the alias method is a required overhead, it may not be the preferred method if the generation of only a small number of random variables is desired. Because of its exactness and speed, in addition to its generality, simplicity, and portability, the alias method is a highly promising one for generating large numbers of random variables from any discrete distribution.

4. THE REJECTION-MIXTURE AND THE ALIAS-REJECTION-MIXTURE METHODS

The alias-rejection-mixture (ARM) method [Kronmal and Peterson (16)] is a promising general exact method for generating random variables from an arbitrary continuous, discrete, or mixed distribution.

The ARM method combines the features of two methods: the alias method for generating random variables from discrete distributions (Section 3), and a modified rejection method, called the rejection-mixture method (16), described below. In sharp contrast to the simple rejection method, the rejection-mixture method does not require repeat of steps of the algorithm.

4.1 Rejection-mixture method

To generate random variables from $f(x)$, $a \le x \le b$, make preparations as follows:

(a) Choose a decomposition of $f(\cdot)$ into subdensities $g_1(\cdot)$ and $g_2(\cdot)$:

$$f(\cdot) = g_1(\cdot) + g_2(\cdot)$$

(b) Choose a function

$$h(x) \ge g_1(x), \quad a \le x \le b,$$

that:

(i) dominates the first subdensity $g_1(x)$, and

(ii) is a density.

The rejection-mixture algorithm

For each random variable desired, perform the following two steps:

RM1. [Prepare for the acceptance-"rejection" step.]

(a) Generate $X \sim h(\cdot)$.

(b) Generate $U \sim$ uniform $(0,1)$.

RM2. [Acceptance-"rejection"]

(a) If $U \le \frac{g_1(X)}{h(X)}$, then return X.

(b) Otherwise, generate Y from the density $c \cdot g_2(\cdot)$, and return Y.

It is noteworthy that "rejection" in step RM2 in the rejection-mixture method is similar to the aliasing in the alias method; the difference is that here "rejection" specifies to return an alias random variable $Y \sim c \cdot g_2(\cdot)$, whereas in the alias method "rejection" specifies to return the alias constant A_I.

Note that the rejection-mixture method offers the flexibility of two choices in the design of the method for a specified $f(\cdot)$:

(a) a choice of the decomposition of the density $f(\cdot)$ into subdensities $g_1(\cdot)$ and $g_1(\cdot)$, and

(b) a choice of the density $h(\cdot)$ that dominates $g_1(\cdot)$.

The strategies behind making these choices, which are crucial to the efficient operation of the method, are similar to those behind the choice of dominating function in the simple acceptance-rejection method. They will not be discussed here, except to say that the aims are that both the densities $h(\cdot)$ and $c \cdot g_2(\cdot)$ are easy to generate variables from. Note that whereas in the simple rejection method it is quite advantageous to make large the probability of acceptance in order to avoid repeating steps of the algorithm, it is not particularly advantageous in the rejection-mixture method, since upon "rejection" in the rejection-mixture method steps are not repeated but rather a (hopefully) simply-generated random variable is generated. Thus, unlike the simple-rejection method, in the rejection-mixture method it is not particularly advantageous to choose the dominating function close to the function it dominates, and thus more flexibility is provided in the rejection-mixture method in the choice of dominating function.

Like the simple-rejection method, the rejection-mixture method is applicable to continuous, discrete, or mixed densities. Although in most cases (when generating $Y \sim c \cdot g_2(\cdot)$ is faster than repeating steps) it is more efficient than the simple-rejection method, the rejection-mixture method suffers from the same limiting feature that the function $g_1(\cdot)$ must be evaluated each time. Even if upper and lower constant bounding functions for $g_1(\cdot)/h(\cdot)$ are used for the initial comparisons to minimize the proportion of times when evaluation of $g_1(\cdot)/h(\cdot)$ is necessary, their value is limited in the usual instances in which $g_1(\cdot)/h(\cdot)$ is not nearly constant. Non-constant bounding functions can be used, but these typically require more time to evaluate than constant ones. Thus, when used as the sole tool, the rejection-mixture method usually suffers the same fate as the simple-rejection method--for most distributions, when used alone it is not competitive with other methods.

However, when combined with the alias method in the alias-rejection-mixture method, as described below, the rejection method will become more useful. By choosing large enough the number n of judiciously chosen densities $f_i(\cdot)$ that are elements of an equiprobable mixture

$$f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x),$$

the densities $f_i(x)$ can be chosen, it appears, to be essentially flat. When the rejection-mixture method is applied to

each of them, and when upper and lower constant bounding functions are used, it appears that evaluation of any nonconstant functions can be avoided almost completely.

4.2 The alias-rejection-mixture (ARM) method

The alias-rejection-mixture method (ARM) is a combination of Walker's alias method and the rejection-mixture method.

The preparations needed to generate random variables from a specified density $f(x)$, $a \le x \le b$ (where a,b may be $\mp \infty$, respectively) are as follows:

(a) Express the density $f(\cdot)$ as a mixture (usually, but not necessarily, equiprobable) of n densities $f_i(\cdot)$, $i = 1,2,\ldots,n$:

$$f(\cdot) = \sum_{i=1}^{n} p_i \cdot f_i(\cdot) \qquad (2)$$

$$0 \le p_i \le 1, \quad i = 1,2,\ldots,n$$

$$\sum p_i = 1$$

(b) For each of the densities $f_i(\cdot)$ of the mixture, choose a decomposition into subdensities:

$$f_i(\cdot) = g_{i,1}(\cdot) + g_{i,2}(\cdot), \qquad (3)$$

$$i = 1,2,\ldots,n.$$

(c) For each of the subdensities, $g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$, of the first term on the right-hand side of eq. (3), choose a function $h_i(\cdot) \ge g_{i,1}$ that dominates $g_{i,1}(\cdot)$, and is a density, i.e.,

$$\int_a^b h_i(x)\,dx = 1.$$

Once the above choices of the p_i, $f_i(\cdot)$, $g_{i,1}(\cdot)$, $g_{i,2}(\cdot)$, and $h_i(\cdot)$, $i = 1,2,\ldots,n$, are made, they are used to generate random variables from $f(\cdot)$ as follows:

Alias-rejection-mixture algorithm

For each random variable desired from the specified distribution $f(\cdot)$, perform the following three steps:

ARM1. [Choose at random an element of the mixture eq. (2)]

Generate $I \sim$ discrete distribution $P(I = i) = p_i$, $i = 1,2,\ldots,n$. [If $p_i = 1/n$ for all i, then $I \sim$ uniform $(1,2,\ldots,n)$.]

ARM2. [Prepare for the acceptance-"rejection" step]

a. Generate $X \sim h_I(\cdot)$.
b. Generate $U \sim$ uniform $(0,1)$.

ARM3. [Acceptance-"rejection"]

If $U \le \dfrac{g_{I,1}(X)}{h_I(X)}$, then return X.

Otherwise, generate $Y \sim C_I \cdot g_{I,2}(\cdot)$, where

$$C_I \equiv 1 \Big/ \int_a^b g_{I,2}(x)\,dx,$$

and return Y.

Comments

(1) This method is truly a combination of the alias method and the rejection-mixture method. Step 1 in ARM is similar to step 1 in the alias method, the only difference being that ARM has added flexibility in choosing the p_i. (In most cases, however, it will not be advantageous to use this flexibility, for the reason that uniformly distributed integers are much more easily generated on a computer than nonuniformly distributed integers). Steps 2 and 3 in ARM are similar to steps 1 and 2 in the rejection-mixture method, the only difference being that ARM does the acceptance-"rejection" procedure conditionally on the randomly-selected member I of the mixture (eq. 2).

(2) The method has been described above as applicable to arbitrary continuous or mixed distributions. Actually, it is applicable also to purely discrete distributions, but here (the sole use of) the alias method is generally more efficient.

(3) There are some similarities between the ARM method and the rectangle-wedge-tail method for generating normal random variables. In particular, the ARM method uses the feature of equiprobable mixtures.

However, unlike the rectangle-wedge-tail method, the ARM method uses the alias and rejection-mixture ideas for generating random variables from the chosen element of the mixture.

(4) The ARM method allows many choices to be made in the preparation portion of the method:

(a) a choice of integer n of elements of the mixture

(b) a choice of mixture weights p_i, $i = 1,2,\ldots,n$,

(c) a choice of mixture elements $f_i(\cdot)$, $i = 1,2,\ldots,n$,

(d) for each $f_i(\cdot)$, $i = 1,2,\ldots,n$, chosen in 4c, a choice of subdensities $g_{i,1}(\cdot)$ and $g_{i,2}(\cdot)$, and

(e) for each subdensity $g_{i,1}(\cdot)$ chosen in 4d, a choice of the dominating density $h_i(\cdot) \geq g_{i,1}(\cdot)$.

The choices are of course crucial for the speed of the method. However, the choices of any of the above items affects the ability to make good choices of the other items, and so the search for a good combination of choices is not always straightforward. The restrictions on the choices are noted in Comment 5 below, and the general aims of any strategy for making the choices are noted in Comment 6. Below we shall discuss one particularly promising strategy, called the uniform alias-rejection-mixture method, that seems to result in remarkably few and simple operations and that offers promise of a strategy that can be routinely applied to any distribution.

(5) There are several conditions that the choices must satisfy in order to have a valid method that works as described:

(a) the items n, p_i, and $f_i(\cdot)$, $i = 1,2,\ldots,n$ must be chosen so that the mixture (2) is a bona fide one for $f(\cdot)$. That is, n a positive integer, $\emptyset \leq p_i \leq 1$, $f_i(\cdot)$ densities, and

$$\sum_{i=1}^{n} p_i \cdot f_i(\cdot) = \text{the specified } f(\cdot).$$

(b) the subdensities $g_{i,1}(\cdot)$ and $g_{i,2}(\cdot)$ must be chosen so that $g_{i,1}(\cdot) + g_{i,2}(\cdot) = f_i(\cdot)$, $i = 1,2,\ldots,n$, and so that they are bona fide subdensities, i.e., $g_{i,1}(x) \geq \emptyset$, $g_{i,2}(x) \geq \emptyset$, for all i and x.

(c) the functions $h_i(\cdot)$ must be chosen so that

(i) $h_i(\cdot) \geq g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$

(ii) $\int_a^b h_i(x)dx = 1$, $i = 1,2,\ldots,n$.

(6) The general aims of any strategy for choosing the quantities in 4 above are:

(a) (Efficiency in step ARM1) That the mixture probability distribution $P(I = i)$, $i = 1,2,\ldots,n$ is one from which random variables are efficiently generated. Usually this would mean either (i) that the number n of elements of the mixture is small, say 2 or 3, or (ii) that the distribution $P(I = i)$ is uniform: $P(I = i) = 1/n$, $i = 1,2,\ldots,n$. The latter choice is made below in UARM. It is crucial that the method (fortunately) retains its flexibility of application to any density, despite this choice of a uniform density of $P(I = i)$.

(b) (Efficiency in step ARM2) That the dominating densities $h_i(\cdot)$, $i = 1,2,\ldots,n$, are ones from which random variables are efficiently generated. For example, one desirable choice would be uniform densities, the simplest of all densities to generate. This choice is made below in UARM again, fortunately without, it seems, sacrificing any needed flexibility.

(c) (Efficiency in step ARM3) That the quantities $g_{i,1}(x)$ and $h_i(x)$, $i = 1,2,\ldots,n$, are easy to compute, and that the densities $c_i \cdot g_{i,2}(\cdot)$, $i = 1,2,\ldots,n$, are ones from which random variables are efficiently generated. For example, the choice of uniform densities for the $h_i(\cdot)$, mentioned above as a desirable choice with regard to efficiency of step ARM2, is also a good one to make the $h_i(\cdot)$ easy to compute. Also, one desirable choice for the densities $c_i \cdot g_{i,2}(\cdot)$ would be uniform densities. This choice is made below in UARM.

Unfortunately, since one of the conditions on the choices requires that the mixture of densities $f_i(\cdot)$ is the specified density $f(\cdot)$, not all subdensities can be chosen arbitrarily; our convention is to select the subdensities $g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$, to be responsible for the main feature (shape) of the specific density $f(\cdot)$. Thus, typically the subdensities $g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$, will not be easy to compute, since typically $f(\cdot)$ is not. However, the need to compute these subdensities will be avoided (at the expense of only one or two comparisons and table lookups) by incorporating boundary functions into the alias-rejection-mixture method, as discussed below.

(7) In a fashion similar to that used in the simple acceptance-rejection method, or in the rejection-mixture method, constant

(or linear, or piecewise linear, at some expense in computing time) lower and/or upper bounding functions L_i and U_i for the quantities $g_{i,1}(\cdot)/h_i(\cdot)$, $i = 1,2,\ldots,n$, can be specified (and stored), and then computation of $g_{I,1}(X)/h_I(X)$ in step ARM3a is largely avoided by modifying this step as follows to compare the uniform (0,1) random variable U with L_I and U_I first. The step ARM3 then becomes:

ARM3′ Perform the following steps until a value is returned:

a′. If $U \le L_I$, then return X.

b′. If $U > U_I$, then return $Y \sim c\, g_{I,2}(\cdot)$.

c′. If $U \le \dfrac{g_{I,1}(X)}{h_I(X)}$, then return X.

d′. Otherwise, return $Y \sim c_I \cdot g_{I,2}(\cdot)$.

Note that step ARM3c′, which involves a computation of the function $g_{I,1}(X)/h_I(X)$, is performed only if the comparisons in steps ARM3a′ and ARM3b′ both fail. If the constants L_I and U_I are close to $g_{I,1}(X)/h_I(X)$, then the computation of $g_{I,1}(X)/h_I(X)$ will largely be avoided. Also note that each of the steps ARM3a′ and ARM3b′ consists of comparing the uniform variate U with a tabled constant.

(8) If ARM is modified as shown in Comment 7 to use upper and lower constant bounding functions L_I and U_I for the functions $g_{I,1}(\cdot)/h_I(\cdot)$, $i = 1,2,\ldots,n$, then a further convenient modification can also be made: the "alias" random variable Y in step ARM3d′ need not be specified to have the same distribution as the "alias" random variable Y in step ARM3b′. That is, in ARM3d′ we could return $Y \sim d_I \cdot g_{I,3}(\cdot)$, where $g_{I,3}(\cdot)$ is chosen so that

$$f_I(\cdot) = g_{I,1}(\cdot) + g_{I,2}(\cdot) + g_{I,3}(\cdot),$$

that is, a decomposition into 3 subdensities instead of 2 (thus two "alias" subdensities instead of only one). This gives added flexibility at no cost in additional steps. Furthermore, if the random variable Y in step ARM3d′ is specified to have one distribution <u>regardless of I</u>, which is practical when the total of the masses of the subdensities $g_{i,3}(\cdot)$, $i = 1,2,\ldots,n$, is small, then it appears that the method gains a slight advantage in speed (because table look-up is avoided in the few instances when step ARM3d′ is performed). Also realized is a gain in simplicity and efficiency of the initial portion of the algorithm that computes the required constants (because computation of $P[U > g_{I,1}(X)/h_I(X)]$, which requires an integration of the density, is not needed; computation of the total mass of the subdensities $g_{i,3}(\cdot)$, $i = 1,2,\ldots,n$, $\sum_i P(g_{i,1}(X)/h_i(X) \le U \le U_i)$, which requires no integration, suffices).

It appears (details not given here) that the alias-rejection mixture method can be applied to generate random variables from any continuous or mixed distribution. For any distribution it offers a wide range of allowable choices of mixture elements and subdensities in the search for an efficient combination of choices. Described in the next section is an exceptionally promising special case of the ARM method, called the <u>uniform</u> alias-rejection-mixture method, which embodies a strategy for choosing the mixture elements and subdensities that appear to be generally applicable, and whose operations are few, simple, and even, to a large extent, independent of the specified distribution $f(\cdot)$.

4.3 <u>The uniform alias-rejection-mixture method</u>

The uniform alias-rejection-mixture (UARM) method, a specialized version of the ARM method with bounding functions, is a general exact method for generating random variables from continuous (or mixed) distributions, whose operations are few and simple and approximately the same for all distributions.

The UARM method is the ARM method (with bounding functions) specialized by the following choices, designed to attain operations that are simple and few:

(a) equal (= 1/n) mixture weights p_i, $i = 1,2,\ldots,n$.

(b) subdensities $g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$, of the form

$$g_{i,1}(x) = (n \cdot f(x) - d_i) \cdot I(x \in I_i),$$

whose supports are adjacent intervals $I_i \equiv (a + (i-1)w,\ a + iw)$ of equal width w (except possibly for $g_{1,1}(\cdot)$ and $g_{n,1}(\cdot)$, which would handle the left and right tails). Note that the subdensities $g_{i,1}(\cdot)$, $i = 1,2,\ldots,n$, have the same shape as the parent density $f(\cdot)$ in each of the intervals I_i, $i = 1,2,\ldots,n$. Thus, they have the primary responsibility for ensuring that the mixture

$$\frac{1}{n} \sum_{i=1}^{n} (g_{i,1}(\cdot) + g_{i,2}(\cdot) + g_{i,3}(\cdot))$$

does indeed equal the density $f(\cdot)$. Note, however, that the subdensities $g_{i,1}(\cdot)$ do not have this entire responsibility,

because they are missing a uniform contribution $d_i \cdot I(x \in I_i)$ that is regained (see below) from some of the subdensities $g_{i,2}(\cdot)$ and $g_{i,3}(\cdot)$, $i = 1,2,\ldots,n$.

(c) alias subdensities $g_{i,2}(\cdot)$, $i = 1,2,\ldots,n$, that are all uniform and with support the intervals I_{j_i}, where the integer j_i ($1 \le j_i \le n$) indicates the alias interval for mixture density $f_i(\cdot)$.

(d) (if the second-alias subdensities $g_{i,3}$, $i = 1,2,\ldots,n$, referred to in comment 8 of Section 4.2 are used) alias subdensities $g_{i,3}(\cdot)$, $i = 1,2,\ldots,n$, that are all uniform with common support (c,d).

(e) dominating densities $h_i(\cdot)$, $i = 1,2,\ldots,n$, that are all uniform on the intervals I_i, i.e., $h_i(x) = 1/w \cdot I[x \in I_i]$, [except possibly for $h_1(\cdot)$ and $h_n(\cdot)$, which dominate the tail element subdensities $g_{1,1}(\cdot)$ and $g_{n,1}(\cdot)$].

(f) lower and upper bounding functions L_i and U_i that are the best possible: i.e.,

$$L_i \equiv \inf_{x \in I_i} w \cdot g_{i,1}(x), \text{ and}$$

$$U_i \equiv \sup_{x \in I_i} w \cdot g_{i,1}(x).$$

Once the remaining choices (n, a, w, and $g_{i,1}(\cdot)$, j_i, $i = 1,2,\ldots,n$), which depend on the specified $f(x)$, are made, they are used to generate random variables from $f(\cdot)$ as follows:

The Uniform alias-rejection-mixture algorithm

For each random variable desired from the specified distribution $f(\cdot)$, perform the following three steps:

UARM1. [Choose at random an element of the mixture]

Generate $I \sim$ discrete uniform $(1,2,\ldots,n)$

UARM2. [Prepare in part for the acceptance - "rejection" step]

Generate $U \sim$ uniform $(0,1/w)$

UARM3. [Acceptance-"rejection"]

(a) if $U \le L_I$, then return $X \sim$ uniform (I_i).

(b) If $U > U_I$, then return $Y \sim$ uniform (I_{j_i}).

(c) Generate $X \sim$ uniform I_I. If $U \le w \cdot g_{I,1}(X)$, then return X.

(d) Otherwise, return $Y \sim$ uniform (c,d).

These steps are remarkably few and simple. The steps require the following operations per random variable generated:

Step UARM1: 1 uniform random variable

Step UARM2: 1 uniform random variable

Step UARM3: either 1 or 2 table look-ups and comparisons, plus 1 uniform random variable and table look-up, plus computation of the subdensity $g_{I,1}(\cdot)$, another comparison, and another uniform random variable (Y) a fraction of the time. We will note below that by choosing n large enough it appears that the computation of the subdensity $g_{I,1}(\cdot)$, the comparison of U with $g_{I,1}(\cdot)$, and the generation of the uniform random variable Y will be performed an arbitrarily small fraction ε of the time.

The total of the operations required on the average per variable is, thus, as follows, where ε denotes the small fraction (discussed below) of the time that Step UARM3c is reached:

(1) uniform random variables: $3 + \varepsilon$ random variables.

[This can be reduced to $2 + \varepsilon$ if a uniform $(0,1)$ variable and a uniform $(1,2,\ldots,n)$ variable are stripped from one uniform $(0,n)$ variable, and can be reduced to nearly $1 + \varepsilon$, if other standard tricks (such as use of conditional uniformity) are used.]

(2) comparisons: between $1 + \varepsilon$ and $2 + \varepsilon$

(3) table look-ups: between 2 and 3

(4) computation of density: ε

It is noteworthy that the type and number of operations needed per random variable generated does not depend on the distribution considered, provided that n can be chosen large enough. Furthermore, not only are the operations few and simple on the average (per variable), they are essentially uniformly few and simple for each variable generated. Although the fact that each variable is efficiently generated is not an important consideration in most applications, it might possibly become relevant on machines where computations which rely on each other are

done in parallel on different central processing units.

Now that the general UARM method has been described, the only remaining consideration is how, for a specified $f(\cdot)$, to choose:

(1) the number of n of mixture elements $f_i(\cdot)$, $i = 1,2,\ldots,n$,

(2) the constants a and w that define the n adjacent intervals,

(3) the subdensities $g_{i,1}(\cdot)$, $i = 1,\ldots,n$, whose supports are the intervals $I \equiv (a + (i - 1)w, a + iw)$, and

(4) the indicators j_i, $i = 1,2,\ldots,n$, of the alias intervals I_{j_i} for mixture elements $f_i(\cdot)$,

(5) the common support (c,d) of the second-alias subdensities $g_{i,3}(\cdot)$, $i = 1,2,\ldots,n$.

Research is underway on good strategies for routinely and efficiently making these choices. Already developed is a strategy for choosing n, the alias indicators j_i, $i = 1,2,\ldots,n$, and the suppressing constants d_i, $i = 1,2,\ldots,n$, that has been successfully applied to the normal distribution [Kronmal and Peterson,(16)]. This strategy does not depend on any special feature of the normal distribution, and appears to be applicable to any continuous distribution.

5. CONCLUSION

In this paper we have reviewed the alias and alias-rejection mixture methods for generating discrete and continuous random variables respectively, and we have presented algorithms for implementing them. The alias method is applicable to any discrete distribution, and the alias-rejection-mixture method appears to be applicable to any continuous or mixed distribution. Furthermore, each method involves relatively few and simple operations, and when implemented result in programs that are short and fast.

Additional investigation is needed in methods to optimize and generalize the choices necessary in implementing the alias-rejection-mixture algorithm. Among the open questions are: (1) how many subdensities should be used; (2) how should appropriate subdensities (and the resulting table of constants) be routinely computed? (3) can the method be routinely tailored to improve performance for specific distributions; and (4) can the method be modified in such a way as to allow for frequent changes in parameter value without requiring costly total regeneration of the table constants.

Finally, work is in progress on implementing both the alias and UARM methods for specific families of distributions including the Poisson, binomial, geometric, and multinomial (using the alias method) and the normal (16), t, exponential, gamma, beta, (using the UARM method). This includes running timing studies for each distribution to compare these new methods against other widely used methods. Details and results of these implementations will be reported in the future.

BIBLIOGRAPHY

1. Knuth, D. E. (1969). The Art of Computer Programming, Vol. 2, Reading Mass.: Addison-Wesley.

2. Ahrens, J. H. and Dieter, U. (1973a). Uniform Randum Numbers, unpublished manuscript, Institut for Math. Statistik, Technische Hochschule in Graz, Austria.

3. von Newmann, J. (1963). Various techniques used in connection with random digits. In Collected Works, 5, Pergamon Press, New York, 768-770.

4. Forsythe, G. E. (1972). von Neumann's comparison method for random sampling from the normal and other distributions. Math. Comput., 26, 817-826.

5. Ahrens, J. H. and Dieter, U. (1973b). Non-Uniform Random Numbers, unpublished manuscript, Institut fur Math. Statistik, Technische Hochschule in Graz, Austria.

6. Kinderman, A. J. and Monahan, J. F. (1977). Computer generation of random variables using the ratio of uniform deviates. ACM Trans. Math Software, 3, 257-260.

7. Marsalia, G., MacLaren, M. D. G., and Bray, T. A. (1964). A fast procedure for generating normal random variables. Commun. ACM, 6, 37-38.

8. Atkinson, A. C. (1979). The computer generation of Poisson random variables. Appl. Statist., 28, 29-35.

9. Marsaglia, G. (1963). Generating discrete random variables in a computer. Commun. ACM, 6, 37-38.

10. Walker, A. J. (1974a). Fast generation of uniformly distributed pseudo-random numbers with floating point representation. Elec. Lett., 10, 553-554.

11. Walker, A. J. (1974b). New fast method for generating discrete random numbers with arbitrary frequency distribution. Electronics Letters, 10, 127-128.

12. Walker, A. J. (1977). An efficient method for generating discrete random variables with general distributions. ACM Trans. Math. Software, 3, 253-256.

13. Ahrens, J. H. and Dieter, U. (1974). Computer methods for sampling from gamma, beta, Poisson, and binomial distributions. Computing (Arch. Electron. Rechnen), 12, 223-246.

14. Brent, R. P. (1974). A Gaussian pseudo random number generator. Commun. Ass. Comput. Mach., 17, 704-706.

15. Atkinson, A. C. and Pearce, M. C. (1976). The computer generatin of beta, gamma, and normal random variables. J. R. Statist. Soc. A., 139, 431-448.

16. Kronmal, R. A. and Peterson Jr., A. V. (1979). The alias-rejection-mixture method for generating random variables from continuous distributions. Submitted.

17. Kronmal, R. A. and Peterson, Jr., A. V. (1979). On the alias method for generating random variables from a discrete distribution. The Amer. Statist. (in press).