Generating Non-uniform Random Variables: Infinite Precision Procedures and Computational Complexity

by

Herman Rubin[1]
Purdue University

Technical Report #86-58

Department of Statistics
Purdue University

December 1986

[1] Research supported by the National Science Foundation under Grant DMS-8401996.

*

Generating Non-uniform Random Variables: Infinite Precision Procedures and Computational Complexity

Herman Rubin
Purdue University

Much has been published about the generation of non-uniform random variables from uniform information. The typical paper on this subject frequently produces a procedure without regard for the computational costs involved in the amount of information processed, and, with few exceptions, only real input is used. This automatically means that, as accuracy increases, the processing cost must increase at least linearly in the accuracy desired. However, this need not be so! For example, one could generate a normal random variable with mean exactly e and variance exactly π, and accomplish this with finite expected computational cost. Of course, to get the result to n-bit accuracy, it is still necessary to copy n bits, corrected for the bits that are produced by the procedure.

It may be asked how can this be done, considering that the parameters and values cannot even be represented on the machine? The answer is to produce an interval such that the random variable is conditionally uniform over the interval; then if more accuracy is needed, it can be obtained by the acquisition of random bits, but no additional computation need be done. One can argue that this may be sufficiently costly that the result should be "filled out" to full accuracy immediately; this may or may not be true, as can be seen from the discussion. Another question is as to the ease of implementing these procedures on computers; here the results vary from easy to very difficult, but possible. In fact, computer hardware is suggested by these procedures–on some computers, it may be profitable to use another computer better designed for this purpose to do the job. It may be very undesirable to have portable procedures. Also, it is necessary to have excellent independence properties for the input random bits; I do not believe that most " reasonable" pseudo-random generators have any chance of achieving this. Of course, I have never believed that, where anything other than situations in which quasi-random numbers are preferable to random numbers, that a machine generated stream should even be considered by itself.

What is an infinite precision random variable? How often are they available, and how costly are they? We shall, for absolutely continuous random variables, give a complete answer as to the availability, and we shall give examples of their utility in the lattice case and the absolutely continuous case. We only have partial answers as to the cost. Of course, for a discrete random variable there is always an infinite precision procedure; nevertheless, incomplete specification of the result can be profitable, and can lead to computationally efficient methods for complete specification. Infinite precision methods can even be applied in customary acceptance-rejection methods to good effect.

Let f be the density of a random variable, or even a random vector. An infinite precision procedure for attaining this distribution is given by the construction of a sequence p_i of probabilities, each of which is a power of $\frac{1}{2}$, and a sequence of assignments of bits in each of the cases. For example, let $f(x)$ be $2x$ on the unit interval, and 0 elsewhere. Then

if p_i is 2^{-i}, and the i-th assignment is 0 to the left of the binary point and 1 to the i-th bit to the right of the binary point, this density is achieved. We see that the condition is that the density must be the sum of a sequence of functions, and each function must be a power (positive or negative) of 2 on an interval (rectangular parallelopiped) whose endpoints have coordinates which are successive multiples of (possibly different) powers of 2.

For an absolutely continuous random variable, a necessary and sufficient condition that there is an infinite precision procedure for generating random variables with that distribution is that there is a lower semi-continuous density. This follows because each density of the form above is, if set 0 at the boundary, is a sum of lower semi-continuous functions. Conversely, each non-negative lower semi-continuous function can be seen to be, except on the union of a countable set of boundaries of parallelopipeds, a sum of such functions.

If the density is also lower semi-continuous in a set of parameters, this approach can be extended to obtain infinite precision procedures even if the parameters must be calculated; this is useful also in obtaining multivariate infinite precision procedures by the use of marginals and conditionals.

Not all densities have infinite precision procedures. The following example shows that a density f bounded by 2 on the unit interval can have an infinite precision representation, while $2 - f$ has the property that there is no non-negative lower semi-continuous function other than 0 which it dominates. To do this, let A be a Cantor set of measure $\frac{1}{2}$ on the unit interval. The the function which is 2 on A and 0 otherwise is totally not infinite precision, while the reverse is infinite precision.

Let us now attack the problem of cost. If an infinite precision procedure for a univariate density f can be achieved using a finite expected number of bits, the Wiener-Shannon information $I(f)$ in the exponent in the floating point representation must be at most 2 more than the expected number of bits. If f is unimodal with mode 0, there is a procedure whose expected number of bits required is at most $I(f) + C$, where C is a fairly small number, probably around 5 or 10.

One may ask about optimal infinite precision procedures for generating a random variable. In the discrete case, there is an optimal procedure in terms of bits used. However, in the density case, there are different definitions of optimal, and also many of the procedures which are optimal by these criteria may be extremely difficult to obtain, and may possibly have an infinite expected cost. The three optimality criteria that we have considered are: minimize the expected number of bits needed for generation, minimize the expected number of bits produced to the rignt of the binary point (this may be negative), and minimize the expected number of bits needed to produce high-accuracy random variables. The first clearly exists, but there does not seem to be any method, in general, of finding it. For the last two, one can produce explicit algorithms to generate them. They are bit efficient and quite easy to describe algorithmically, but extremely difficult to produce practically. The method given for the density $2x$ is optimal in all respects, but for other distributions, such as the exponential, these optimality criteria are difficult to achieve. It is possible to

produce an excellent procedure for the exponential, which is not optimal in bit utilization, which achieves the second criterion.

The problem with obtaining the bit-optimal procedures is that the probability of not terminating after k bits is on the order of $2^{-k/2}$, and the number of distinct regions unfinished is on the order of $2^{k/2}$. Smoothness properties of the density may enable one to carry out the procedure with finite expected cost. It is generally possible to produce computationally non-optimal procedures which achieve the minimum expected number of bits output.

However, there are suboptimal procedures which are not too much worse. For example let f be a decreasing density on $(0,\infty)$. Then

$$2^k f(2^k) \geq \int_{2^k}^{2^{k+1}} f(x)dx \geq 2^k f(2^{k+1}).$$

Now if we first select an interval with probabilities proportional to $2^k f(2^k)$, $-\infty < k < \infty$, and in each interval $(2^k, 2^{k+1})$ we proceed by dividing the area under $y = f(2^k)$ in half horizontally and vertically, at least one quarter is either under the density or over the density. The quarters not decided can be quartered, etc. Thus the expected number of bits required after the selection of the interval is at most 8. In the event of rejection, this is repeated. The expected number of bits is this procedure is at most $2I(f) + 24$. The constant 2 cannot be improved below $1/\log 2$. However, if we choose each interval with its probability, we can show that if we use the previous procedure within each interval the expected number of bits is bounded by $I(f) + C$. In no way is this procedure optimal.

There are some surprises. For example, if we generate exponential random variables, it is cheaper in every respect to produce the entire integer part rather than merely specifying for each bit whether it is 0 or random; this latter procedure produces the minimum expected number of bits output, but the output procedure has a much higher amount of information.

We can even use acceptance-rejection procedures to generate infinite precision random variables in practice. For example, suppose we want to generate normal$(0,\frac{1}{2})$ random variables. We shall put the sign on at the end, and use the half uniform, half exponential distribution as the initial distribution and use infinite precision exponentials, which we only compute partially, for test purposes. There is no problem in setting up the procedure.

Let us illustrate some of the procedures for generating the important special case of exponential random variables. We shall need for all the procedures given here geometric random variables with $p = \frac{1}{2}$ for efficiency; on some machines, because right shifts are negative shifts, we should use negative geometric random variables. In what follows, we shall use G to denote such a random variable. The procedures we indicate are a modification of the von Neumann method, an improvement of that, and a method based on the duality between the exponential and Poisson distributions.

The modification of the von Neumann method is based on the following algorithm: Let X be uniform (0,1), and let $Y_0 = X$, and continue taking independent uniform (0,1)

random variables Y_i until the Y-sequence is no longer decreasing. If the first i for which $Y_i > Y_{i-1}$ is odd, accept X, else repeat. The integer part N is the number of rejections until an acceptance, or the number of trials less 1. This can be improved by making X uniform $(0,\frac{1}{2})$, provided M, the multiple of $\frac{1}{2}$to be added to X, is 1 less than the sum of K G's, where K is the number of trials. In both of these situations we only retain in our calculations those bits of X which are known to be 1 and those bits of Y which are known to be 0. On acceptance, a bit of X can be known to be 0. We use the fact that the location of the first bit where a uniform (0,1) random variable differs from a random variable independent of it can be realized by G.

For the Poisson-based method, we use the fact that the exponential distribution is the waiting time for a Poisson process with rate 1. Thus the integer part can be taken to be the number of Poisson random variables which are 0 before a non-zero value occurs, and the fractional part is the minimum of that number of independent uniform (0,1) random variables. Of course, we may generate that minimum by generating only the locations where the uniform random variables differ. A method for doing this which is within 0.75 of the information bound can be found in Rubin [1986]. That method should, of course, be modified to take greatest advantage of the features of whichever computer on which it is implemented.

Another method of generating exponential random variables is based on the fact that we can profitably use exponential random variables to test for the creation of exponentials. Thus to generate exponential random variables mod A, generate uniform random variables mod A and subtract from previously obtained exponential random variables. If the difference is positive, accept the uniform and save the difference for reuse as an exponential random variable, and if negative, discard both. Of course, the multiple of A must be obtained, and if A is 1, $\frac{1}{2}$, or $\log 2$, this is relatively easy. If we use the method based on exponential heads described below, this is relatively easy to implement, especially on microcomputers; there are some independence problems, which are undoubtedly almost irrelevant if large numbers of random variables are generated at a time, and which can be handled in any case. The methods are not as bit-efficient as those previously described.

Generating most other random variables is usually somewhat more difficult. The author has obtained procedures for normally distributed random variables which are combinations of methods similar to the von Neumann method above for part of the range, use an acceptance-rejection method of that type over another part, and a more conventional acceptance-rejection method in the tail. It is usually easy to modify acceptance-rejection methods to be infinite precision. An interesting example is the metaproblem of distributions with a concave logarithm of density. In this case, it is possible to show that there is an algorithm whose expected number of bits used is bounded. The bound is probably in the neighborhood of 15 or 20.

If we have a lattice distribution with concave logarithm of the probability, similar results are true. For such a distribution, examples being hypergeometric, binomial, Poisson, or negative binomial with shape parameter ≥ 1, this enables us to generate random variables with the expected number of bits used not too much greater than $(\log_2(2\pi eV)/2$,

where V is the variance. For a binomial with $n = 10^6$, this number is at most 11.013.

We may make use of the fact that the bits of an exponential random variable are independent. Suppose we are conducting an acceptance-rejection test using exponentials for testing, and p is the rejection probability. Then p of the exponentials are lost. Now if we use, say, only the first 10 bits of the fractional part of the exponential, what happens? The expected number of exponentials lost per trial increases by at most 0.001, and usually about 0.0005; the expected proportion of times that the tail is used is at most 0.001; and except when the tail is used, the bits needed to produce the tail are, of course, not used. One can also come up with cute methods of producing the tails, which are very cheap. The actual test can be done in low precision fixed-point arithmetic.

Of course, one could use the infinite-precision versions of exponential random variables in which only a near-minimal amount of information is available for the exponential random variable used, and possibly for the one for which the acceptance-rejection procedure is being used. Only in that case is it likely to pay, since it is easy to reuse exponential random variables upon acceptance; this causes the number of exponential random variables used to be the number of rejections. If the truncated method described above is used, this number is increased by less than the maximal truncation "error." As the other approach uses nearly 2 bits per trial and is computationally more difficult, it seems that one should use the truncated approach. Something similar can be done if testing is done with uniform random variables; here we must work harder.

There are also situations in which one must use some bit inefficiency to obtain reasonable computational efficiency. For example, let it be desired to obtain a random variable uniform on the interval $(0, a)$, where $a = (2(\arctan \frac{1}{2} + \arctan \frac{1}{3})/(3 \arcsin \frac{1}{2})$, and each of the functions is given by a power series. Now *we* know that $a = 1$, but suppose it were not known (there are cases in which a computable real number can have a given value, but this fact is not provable). We can compute a within a provable error of 2^{-k}, and we easily ascertain that $0 < a < 2$. Then, except with probability 2^{-k}, we can determine by examining at most $k + 1$ bits whether a uniform (0,2) random variable is in the desired interval. Since we can determine a to this accuracy easily with the amount of computation at worst on the order of k^2, we can clearly obtain a procedure which attains the desired goal, and which has a finite expected amount of computation and uses a finite expected number of random bits for the computation.

(One way to obtain a normal random variable with mean e and variance π is to first get an infinite precision version of a standard normal random variable, which can be expanded to give as output a uniform random variable in some binary rational interval I. It is now only necessary to obtain a random variable uniform in $e + \sqrt{\pi} \times I$, which the reader should be able to accomplish. Of course, there are other ways in which the desired result can be obtained.)

REFERENCES

Rubin, H. (1986) An efficient method of producing infinite-precision exponential random variables. Purdue University Technical Report #86-39.