

Atomic Scale Simulations

Sampling Methods

Today we discuss how to sample probability distributions. Namely how to transform a uniform prn (udrn) to that of another distribution.

Discrete Distributions

Any discrete distribution can be sampled by constructing the cumulant. Suppose we want to sample the integer k where $1 \leq k \leq N$ with probability p_k . An example might be to sample the roll of several dice. A possible sampling algorithm for any discrete distribution is as follows:

1. Before the MC sampling begins, make the table: $F_k = F_{k-1} + p_k$ with $F_0 = 0$. Since p_k is a probability we must have $0 \leq p_k \leq 1$. Note that F_k is monotonically increasing with $F_N = 1$.
2. Sample u , a uniform rn in $(0, 1)$. Find the unique k that satisfies $F_{k-1} < u < F_k$. Unless p_k has a suitable analytic form, one must solve this inequality using the bisection method. That is start with a range $[1, N]$, bisect this range and decide which subinterval the solution is in, recursively until k is determined.

There are many special cases where one can avoid the bisection. However, the operation involved in step 2 will take in general $\log_2(N)$ steps. If this is time-consuming for large N one can use an improved algorithm (see Knuth) which involves a more elaborate table set up (of order $N \log_2(N)$ steps but which takes only a few operations to sample, independent of the value of N). Essentially we sample uniformly in the range of 1 to N but correct for the fact that P_k is non-uniform by correcting the sampling. This "Walker" aliasing method is as follows:

1. Set up tables Y_k and J_k as follows. Form the list of pairs $(p_1, 1), (p_2, 2) \dots$ and sort them in ascending order of p_i obtaining the pairs $(q_1, a_1), (q_2, a_2) \dots$. Set $n = N$ and repeat the following procedure until $n = 0$. Set $Y_{a_1} = Nq_1$ and $J_{a_1} = a_n$. Delete the pairs (q_1, a_1) and (q_n, a_n) from the list. Insert $(q_n - (1/N - q_1), a_n)$ into the list in the proper place. Decrease n by 1.
2. Sample u , a uniform rn in $(0, 1)$ and divide $1 + N * u = k + x$ with k and integer and $0 < x < 1$.
3. If $x < Y_k$ our sample is k
4. Otherwise our sample is J_k .

Continuous Distributions

To sample a distribution for a continuous variable x , use the method above by letting $N \rightarrow \infty$. This is called the *mapping method*. Suppose we want to sample $p(x)dx$. Then the mapping method is:

1. Analytically or numerically form the cumulative distribution:

$$F(x) = \int_{-\infty}^x dy p(y) \quad (1)$$

2. Sample a uniform rn $u \in (0, 1)$.

3. Solve the equation:

$$F(x) = u \quad (2)$$

for x . Then x is sampled from $p(x)dx$.

This is a completely general method. We can generalize this further if u is sampled from something other than the uniform distribution. If $g(u) = x$ then the probabilities of x and u are related by the equation:

$$p(x)dx = p(u)du. \quad (3)$$

Hence:

$$p(x)|dg/du| = p(u) \quad (4)$$

The inverse cumulant is a solution to this differential equation.

Two common examples of using the mapping method are:

1. $p(x) = (a+1)x^a$ for $0 < x < 1$ with $-1 < a$. Then $F(x) = x^{a+1}$ and hence $x = u^{1/(a+1)}$. Although the form is simple, in general it requires computing two special functions, the log and exponential. We will see another way of doing this.
2. $p(x) = \exp(-x)$ for $0 < x$. Then $F(x) = 1 - \exp(-x)$ and $x = -\ln(u)$.

The mapping method is a straightforward procedure, but it may not be the most efficient or simple. The problem is that the inversion may be awkward to carry out, although with a tabulation it can always be made quite fast though the tables of the cumulant may not fit into fast memory. The other problem with mapping is to handle the tails of the distribution.

The Rejection Method

Another general method is the *rejection method*. The procedure is as follows:

1. Sample x from $q(x)dx$. ($q(x)$ is some pdf perhaps obtained from the mapping method).
2. Accept the value x with probability $r(x)$. To do this we compare $r(x)$ to u (a udrn) and accept if $r(x) < u$. Otherwise we go back to 1 until a number is found.

What is the distribution of accepted x 's? It is proportional to $q(x)r(x)dx$. Hence to sample $p(x)$, we must choose $r(x) = cp(x)/q(x)$ with c some constant; c must be chosen so that $r(x) \leq 1$ so that $r(x)$ is a probability. We find that $1/c$ is the average number of times that the loop is executed. The optimal value of c is $c = \min(q(x)/p(x))$. Only if we could sample $p(x)$ directly would the rejection rate be zero.

The *efficiency* of a sampling method is simply the rate at which it can deliver random numbers. For the mapping method it is $1/CPUtime$ for a single rn. For the rejection method it is $c/CPUtime$ for a single trial rn. One can use an approximate mapping to make a q close to p , and then a final rejection step (with a c close to one) to sample the exact distribution. The rejection method is an illustration of the Monte Carlo rule: *under sampling can be very bad*. One can get very inefficient sampling because of a single region where $q(x) \ll p(x)$.

Composition Methods

Mappings can be considered to be unary operations on a single random number. What if we do something with 2 or more random numbers, such as add or multiply them together. These are composition methods. Consider adding together several random numbers. Suppose x is sampled from $p(x)dx$ and y from $q(y)dy$. Then the sum $z = x + y$ is sampled from the convolution:

$$p(z)dz = \int dx p(x)q(z-x) \quad (5)$$

The characteristic function is the Fourier transform of p : $G(k) = \int dx p(x)e^{ikx}$. By the convolution property of Fourier transforms: $G_z(k) = G_x(k)G_y(k)$. By inverse Fourier transforming one can find the distribution of z . In fact, this is the way we proved the central limit theorem. Multiplication of random numbers can be treated by using logarithms (to convert to additions). Another interesting operation is to take the maximum of several uniform random numbers. For example, two random numbers will give $p(x) = x/2$.

Sampling the normal distribution

This is a common but moderately difficult function to sample. Many different ways have been devised. Here are a few of the ways to sample a ndrn.

1. Sample N uniformly distributed rns u_i in $(0,1)$ and set $\eta = \sqrt{\frac{12}{N}}[\sum_{k=1}^N (u_k - 1/2)]$. Then η has mean zero and unit variance and for large N will approach a normal random number. In fact N can be fairly small. (For $N = 12$ we don't even need to do the division.) The disadvantage is that one is losing randomness in the pseudo-random number stream by grouping things by N 's so the correlation between successive ndrn's may be much higher than between the udrn's. Also it is impossible for the magnitude of η to exceed $\sqrt{3/N}$ so large excursions are not only rare but impossible.
2. The direct mapping method would be to invert the cumulative distribution, that would involve computation of the inverse error function. You can find algorithms for doing that in the literature.
3. There is a much simpler procedure which involves generating a pair of normal distributed random numbers $(\eta_1, \eta_2) = (r \cos(\theta), r \sin(\theta))$ where $\theta = 2\pi u_1$ and $r = \sqrt{-2 \ln(u_2)}$. (This is often called the Box-Mueller method.) So for the price of computing 2 udrn and 4 special functions we can generate 2 ndrns.
4. We can improve this last method by sampling the angle (really the x and y components) directly using the rejection method. The pair (x, y) is sampled in the square $-1 < x, y < 1$ and values of $r^2 = x^2 + y^2 > 1$ are rejected (new pairs are sampled until a satisfactory pair is generated). Then the two ndrn numbers generated are $\sqrt{-2 \ln(u_2)/(x^2 + y^2)}(x, y)$. Hence at the price of rejecting a few pairs (acceptance rate is $\pi/4$), one does not need to compute any trig functions, only a log and square root.

Multivariate Normal Distribution

How can we generate a multivariate normal distribution? We want pairs or triplets of ndrn which are correlated. Here is an algorithm for random vectors $\mathbf{x}$ with D components with a given (known) covariance matrix $\nu_{i,j}$.

1. (Before the sampling) do a Choleski decomposition of the matrix ν . This means to find a "square root" S of $\nu = SS^T$. One can assume that S is a lower triangular matrix so that $S_{ij} = 0$ if $j > i$. Then to find S from ν we loop $i = 1, D$

$$S_{ii} = \sqrt{\nu_{ii} - \sum_{k=1}^i S_{ik}^2} \quad (6)$$

$$S_{ij} = \nu_{ij} - \left[\sum_k S_{ik} S_{jk} / S_{jj} \right] \quad (7)$$

2. sample the vector $\mathbf{y}$ from an uncorrelated normal distribution with mean zero and unit variance.

3. Then $\mathbf{x} = S\mathbf{y}$ is a correlated normally distributed RN with zero mean.

Bias

Let us another problem, namely for any non-linear function $G(F_1, F_2 \dots)$, one will bias the result. First let us work it out for a function of a single variable $G(F)$. We imagine that F has been computed with MC but has some statistical error. We may really be interested in some function of F such as $\ln(F)$ or $\sqrt{F}$. The bias is defined as:

$$Bias = \langle G(F) \rangle - G(\langle F \rangle) \quad (8)$$

The bias is a systematic error so that if we use finite size samples we will be systematically too low or too high. Let us expand F around its mean value $\langle F \rangle$ up to second order:

$$Bias = \frac{d^2 G}{2dF^2} \langle (F - \langle F \rangle)^2 \rangle \quad (9)$$

Hence the bias is caused by the curvature of G . Note that the bias is always order N^{-1} so for large number of samplings it should be smaller than the statistical error which is order $N^{-1/2}$. One can correct for the bias by computing the variance.

Suppose we are doing correlated sampling of several variables as we will describe next time. Then the bias will equal:

$$\langle G \rangle_N \neq G(\langle F_1 \rangle_N, \langle F_2 \rangle_N \dots) \quad (10)$$

where the average is over samples of size N . Our estimate of G may be systematically too high or too low. To get an estimate of this bias, we can do a Taylor expansion of G about the exact values of $\langle F_1 \rangle, \langle F_2 \rangle \dots$

$$\langle G \rangle_N - G(\langle F_1 \rangle_N, \langle F_2 \rangle_N \dots) = \sum_{i,j} \frac{d^2 G(F_1, F_2 \dots)}{dF_i dF_j} \big|_{\langle F_1 \rangle \dots} covar(F_1, F_2) \quad (11)$$

References

Knuth, "Art of Computer Programming"

Kalos and Whitlock, "Monte Carlo Methods"

Abramowitz and Stegun, "Handbook of Mathematical Functions"

"Numerical Recipes"
Rubenstein, "Simulation and the Monte Carlo Method"

©D. M. Ceperley, 2000.