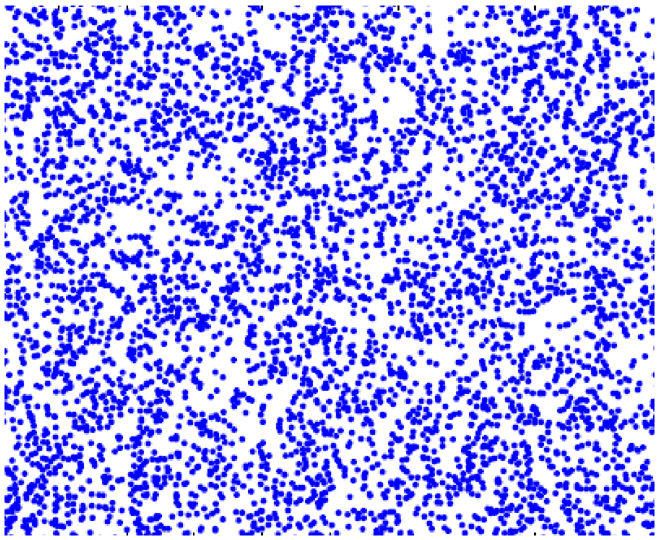
The Monte-Carlo Method

Paul J. Atzberger



Comments should be sent to:
atzberg@math.ucsb.edu

Introduction to Monte-Carlo Methods

The solution of many problems in mathematics can be expressed in terms of an integration of a function. One is often interested in obtaining a numerical value from such expressions, but this is often difficult or tedious to obtain analytically. For integrals of the form

$$I = \int_{\Omega} f(\mathbf{x}) d\mathbf{x}$$

in which Ω is the domain of integration, the integral I can be related to an expectation of a random variable with respect to some probability measure. For probability measures of a random variable X that have a density $\rho(\mathbf{x})$ the expectation can be expressed as:

$$E(f(\mathbf{X})) = \int_{\Omega} f(\mathbf{x})\rho(\mathbf{x}) d\mathbf{x}.$$

The integral I can be expressed in terms of an expectation in a number of different ways. One rather general approach is to use a density having the feature that $\rho(\mathbf{x}) > 0$ whenever $f(\mathbf{x}) \neq 0$. This gives that:

$$\begin{aligned} I &= \int_{\Omega} f(\mathbf{x}) d\mathbf{x} = \int_{\Omega} \frac{f(\mathbf{x})}{\rho(\mathbf{x})} \rho(\mathbf{x}) d\mathbf{x} \\ &= E\left(\frac{f(\mathbf{X})}{\rho(\mathbf{X})}\right) \\ &= E\left(g(\mathbf{X})\right). \end{aligned}$$

where $g(\mathbf{x}) = \frac{f(\mathbf{x})}{\rho(\mathbf{x})}$. In the case of a domain of integration Ω which is finite, we can always use the random variable $\mathbf{X}$ uniformly distributed on Ω with density $\rho(\mathbf{x}) = \frac{1}{|\Omega|}$ to obtain:

$$\begin{aligned} I &= \int_{\Omega} f(\mathbf{x}) d\mathbf{x} = \int_{\Omega} \frac{f(\mathbf{x})}{\frac{1}{|\Omega|}} \frac{1}{|\Omega|} d\mathbf{x} \\ &= |\Omega| E(f(\mathbf{X})). \end{aligned}$$

The utility of expressing the integral in terms of an expectation derives from the *Law of Large Numbers*, which states that for a collection of independent identically distributed random variables $\{\mathbf{X}_i\}_{i=1}^{\infty}$:

$$E\left(g(\mathbf{X})\right) = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{X}_i).$$

This offers a way to estimate the numerical value of I, in particular:

- generate N random variates $\{\mathbf{X}_i\}_{i=1}^{N}$ with distribution $\rho(\mathbf{x})$ on Ω.
- approximate the expectation using the *Law of Large Numbers* $I \approx \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{X}_i)$.

This gives a probabilistic approach to estimating the quantity I. This general class of methods are called *Monte-Carlo Methods* and were proposed for statistical sampling in the 1940's by S. Ulam. The approach is nicknamed after a famous Monaco casino in the Mediterranean.

Accuracy of the Monte-Carlo Method

The Monte-Carlo method has an accuracy which can be estimated as:

$$\text{error} = \left| \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{X}_i) - I \right|$$

$$\begin{aligned} &= \left| \frac{\sigma_g}{\sqrt{N}} \left(\frac{\sum_{i=1}^N g(\mathbf{X}_i) - NI}{\sigma_g \sqrt{N}} \right) \right| \\ &\approx \left| \frac{\sigma_g}{\sqrt{N}} \eta(0,1) \right| \end{aligned}$$

where

$$\sigma_g^2 = \int_\Omega \left(g(\mathbf{x}) - I \right)^2 \rho(\mathbf{x}) d\mathbf{x}$$

and $\eta(0,1)$ denotes a standard normal random variable (Gaussian random variable) with mean zero and variance 1. The last approximation was obtained by using the *Central Limit Theorem*, which states that for a sum of i.i.d random variables Y_i with mean μ and finite variance σ^2:

$$\frac{\sum_{i=1}^N Y_i - N\mu}{\sigma\sqrt{N}} \rightarrow \eta(0,1), \text{ as } N \rightarrow \infty.$$

This shows that asymptotically the error converges at a rate $O(\frac{1}{\sqrt{N}})$, independent of the dimensionality of the problem considered. Furthermore, the convergence rate in the Monte-Carlo method is strongly influenced by the prefactor σ_g which depends on the function $f(\mathbf{x})$ and the sampling distribution with density $\rho(x)$ that is used. The prefactor σ_g presents the primary avenue by which the convergence rate can be improved. We shall discuss a number of approaches by which one can attempt to reduce the size of σ_g.

Monte-Carlo Methods in Practice

Pseudo-Random Number Generation

Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin. - John von Neumann

In order to utilize the Monte-Carlo method in practice we must devise a means by which to generate "random" numbers. Obtaining true random samples of course can not be achieved from a deterministic computation, instead what is sought in practice are algorithms which generate sequences of numbers which while deterministic share many of the features of random sequences when subjected to statistical tests. We shall discuss several algorithms for generating pseudo-random samples for the uniform random variable by attempting to generate suitable sequences of integers in the range $[0, 1, 2, \cdots, M]$. The integers $n_k \in [0, M]$ in the sequence are are then used to obtain pseudo-random samples for the uniform random variable by setting $u_k = n_k/M$. For a general discussion of random number generation see (2–4).

Linear Congruential Generator

The linear congruential generator attempts to create a sequence of numbers in the range $[0, M]$ by using the recurrence:

$$n_k \quad = \quad a \cdot n_{k-1} + b \bmod M$$

where n_0, often referred to as the "seed" of the algorithm, must be provided to determine the sequence.

In order to obtain a sequence with many features expected for uniform random samples the parameters a and b must be chosen carefully with a sufficiently large M.

Example 1: A Bad Generator

$$a = 9, b = 1, M = 1103.$$

This choice of parameters leads to poor results, the generated numbers fail to completely sample the numbers $0, \cdots, M$ and furthermore exhibits significant correlations between subsequently generated numbers, see figure 1 which plots (n_{k-1}, n_k) for $N = 20,000$ samples. Given the small value of M the sequence also repeats at least every 1103 samples.

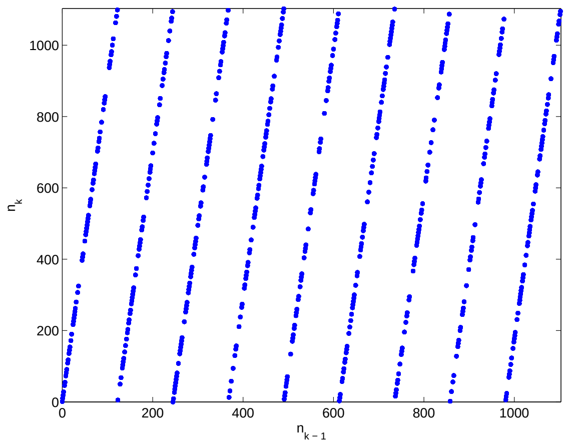


Figure 1: Linear Congruential Generator

Example 2: A Better Generator

$$a = 1230, b = 1201, M = 10001.$$

While this choice of parameters leads to a sequences which appears to be much better with respect to pair correlations, one still must be careful that other correlations do not arise in the samples, see figure 2 which plots (n_{k-1}, n_k) for $N = 20,000$ samples. In general several statistical tests should be performed for the samples, see (3).

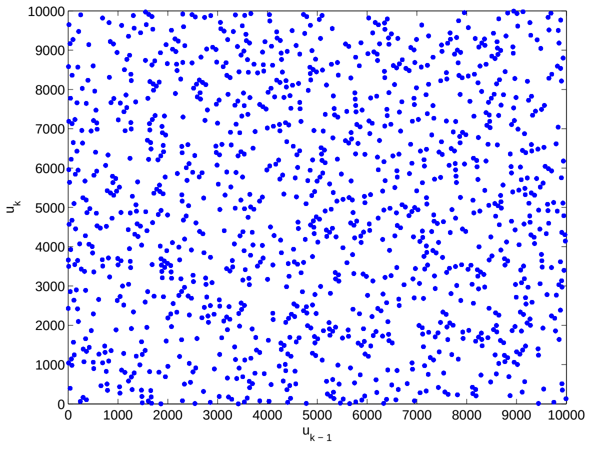


Figure 2: Linear Congruential Generator

Example 3: Matlab 4.0 The LCG was used in early versions of matlab (before 1995) in the routine rand() for uniform random variates, see (1). The parameters for this LCG were

$$a = 7^5 = 16807$$

$$\begin{aligned} b &= 0 \\ M &= 2^{31} - 1 = 2147483647 \end{aligned}$$

see (1; 2). In figure 3 a plot is given of (n_{k-1}, n_k).

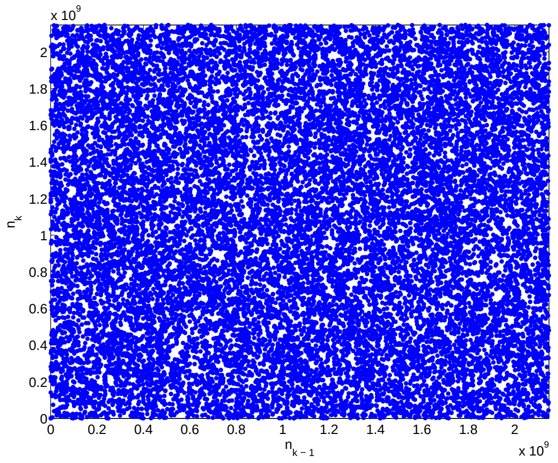


Figure 3: Linear Congruential Generator

Lagged Fibonocci Generators

A random number generator can be obtained by using the recurrence:

$$n_k = a \cdot n_{k-\nu} + b \cdot n_{k-\mu} + c \bmod M$$

where now $n_0, n_1, \cdots, n_\mu$ must be provided to determine the sequence. For the special case of $a = 1$, $b = 1$, $\nu = 1$, and $\mu = 2$ we obtain a recurrence which generates a Fibonocci sequence modulo M.

Example: Fibonocci Generator

$$a = 1, b = 1, \nu = 1, \mu = 2, M = 10001.$$

This generator appears to give a good sampling when compared to the LCG with comparable parameters, see figure 4.

Example: Matlab 5.0 Matlab version 5 uses a lagged Fibonocci generator along with a bit-shift random number generator to obtain a sequence of samples for a uniform random variable.

$$a = 1, b = 1, \nu = 17, \mu = 5.$$

The effective period of the generator is 2^{1492}, which ensures for practical purposes that sequence will not repeat. It is hard to imagine an application which would be carried out computionally that requires anything near 2^{1492} random samples. Also there is evidence to support, that the code will generate almost all floating-point values in the range $[eps/2, 1 - eps/2]$, where eps $= 2^{-52}$, where "eps" stands for "epsilon precision" which is the distance from 1.0 to the next floating point value and indicates the relative accuracy of the floating calculations done on a machine. See figure 5 for plot of the samples, to make a rough comparison with the previous examples we plot $n_k = (round(rand(20000) * M)), M = 10001$.

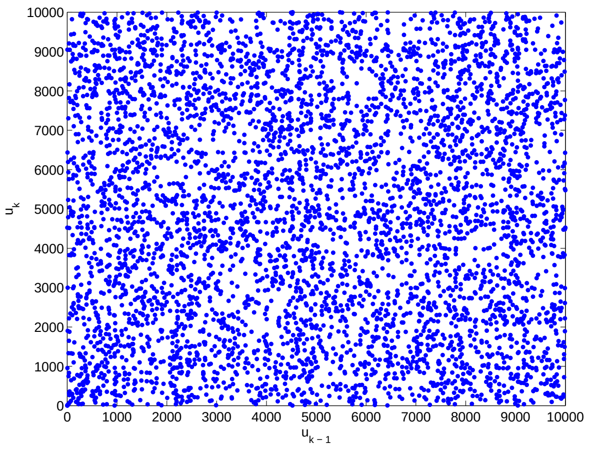


Figure 4: Fibonocci Random Number Generator

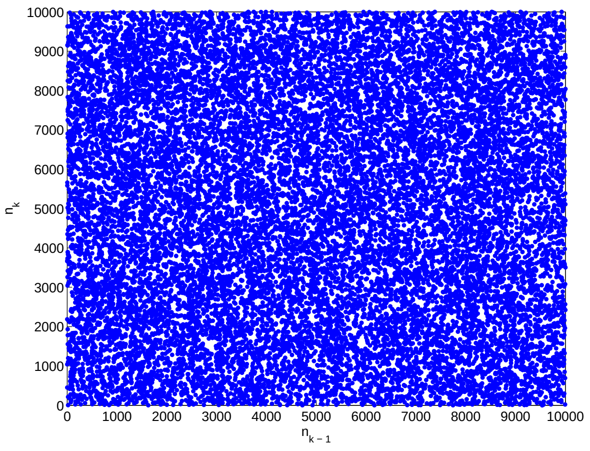


Figure 5: Matlab Random Number Generator: rand()

Transformation Methods

Samples for non-uniform random variables can be obtained from the samples generated for the uniform random variable. We now discuss two results which show how transforming a random variables changes the underlying probability distribution. The following two theorems describe this in terms of the cumulative probability distribution function and the probability density function.

Theorem: Cumulative Distribution Transformation: Let $F_X(x) = \Pr\{X \leq x\}$ be the cumulative probability distribution function for a random variable X. Then for a uniformly distributed random variable U, the random variable obtained from $Z = F_X^{-1}(U)$ has the same probability distribution as X.

Proof: We shall use that the cumulative probability distribution function for a uniform random variable is $F_U(u) = \Pr\{U \leq u\} = u$.

Now consider the probability distribution for the random variable Z,

$$\begin{aligned} F_Z(z) &= \Pr\{Z \leq z\} \\ &= \Pr\{F_X^{-1}(U) \leq z\} \end{aligned}$$

$$\begin{aligned} &= \Pr\{U \leq F_X(z)\} \\ &= F_X(z) \end{aligned}$$

where the third equality uses the monotonicity of $F_X(x)$ and the last equality follows by the preceding remark. This shows that Z and X have the same probability distribution. ■.

Theorem: Probability Density Transformation Let $\rho_X(x)$ be the probability density function for a general n-dimensional random variable $X \in \mathbb{R}^n$. Then the random variable $Z = h(X)$ obtained from an invertible transformation $h : \mathbb{R}^n \to \mathbb{R}^n$ has the probability density

$$\rho_Z(z) = \rho_X(h^{-1}(z)) \left| \frac{dh^{-1}(z)}{dz} \right|$$

where the Jacobian of h^{-1} is defined as

$$\left| \frac{dh^{-1}(z)}{dz} \right| = \det \begin{bmatrix} \frac{\partial h_1^{-1}}{\partial z_1} & \frac{\partial h_1^{-1}}{\partial z_2} & \cdots & \frac{\partial h_1^{-1}}{\partial z_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial h_n^{-1}}{\partial z_1} & \frac{\partial h_n^{-1}}{\partial z_2} & \cdots & \frac{\partial h_n^{-1}}{\partial z_n} \end{bmatrix}.$$

Proof:
By definition of the random variable Z and invertibility of h we have:

$$\Pr\{Z \in h(A)\} = \Pr\{X \in A\}.$$

From the definition of the probability density we have:

$$\begin{aligned} \Pr\{X \in A\} &= \int_A \rho_X(x)dx \\ \Pr\{Z \in h(A)\} &= \int_{h(A)} \rho_Z(z)dz. \end{aligned}$$

By the change of variable $z = h(x)$ we have:

$$\Pr\{X \in A\} = \int_{h(A)} \rho_X(h^{-1}(z)) \left| \frac{dh^{-1}(z)}{dz} \right| dz$$

This implies that for any set A we have:

$$\int_{h(A)} \rho_Z(z)dz = \int_{h(A)} \rho_X(h^{-1}(z)) \left| \frac{dh^{-1}(z)}{dz} \right| dz.$$

This requires that

$$\rho_Z(z) = \rho_X(h^{-1}(z)) \left| \frac{dh^{-1}(z)}{dz} \right|$$

almost everywhere. ■.

Generating Exponentially Distributed Random Variables

An exponentially distributed random variable with rate λ has the probability density $\rho(x) = \lambda e^{-\lambda x}$. The *Cumulative Distribution Transformation Theorem* can be applied to obtain from sample generated for the uniform random variable, samples for the exponentially distributed random variable. The cumulative probability distribution function is given by

$$F_X(x) = \int_0^x \lambda e^{-\lambda x'} dx' = \left[1 - e^{-\lambda x}\right].$$

In this case, the cumulative distribution function can be easily inverted to obtain

$$F_X^{-1}(u) = -\frac{1}{\lambda}\log(1-u).$$

The exponentially distributed random variable X is then obtained from the uniform random variable U in this case by using the transformation:

$$X = -\frac{1}{\lambda}\log(1-U).$$

Generating Normally Distributed Random Variables: Box-Muller

A Normally distributed random variable has probability density $\rho_X(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$. In this case the cumulative distribution function is not readily invertible analytically:

$$F_Z(x) = \int_0^x \frac{1}{\sqrt{2\pi}} e^{-\frac{x'^2}{2}} dx'.$$

Instead we shall make use of the *Probability Density Transformation Theorem* by considering the 2-dimension random variable $Z = [X_1, X_2]$ where both X_1 and X_2 are normally distributed and independent. The probability density in this case is given by $\rho_Z(x_1,x_2) = \frac{1}{2\pi}\exp\left(-\frac{x_1^2+x_2^2}{2}\right)$. In order to obtain Z from the independent uniform random variables U_1 and U_2 using a transformation of the form $[x_1,x_2] = h(u_1,u_2)$, we have from the *Probability Density Transformation Theorem* that:

$$\rho_Z(x_1,x_2) = \rho_U(h^{-1}(x_1,x_2))\left|\frac{\partial h^{-1}(x_1,x_2)}{\partial(x_1,x_2)}\right|$$

A useful approach is to formally express this relation as

$$\begin{aligned}\rho_Z(x_1,x_2)dx_1dx_2 &= \rho_U(u_1,u_2)\left|\frac{\partial(u_1,u_2)}{\partial(x_1,x_2)}\right|dx_1dx_2 \\ &= \rho_U(u_1,u_2)du_1du_2 \\ &= du_1du_2\end{aligned}$$

where we used that $[u_1,u_2] = h^{-1}(x_1,x_2)$ and that $\rho_U(u_1,u_2) = 1$.

This suggests the following strategy, try to find a sequence of changes of variable, which under the formal rules $g(x)df(x) = g(x)f'(x)dx$ gives a prefactor which is identically one, $g(x)f'(x) = 1$. The *Probability Density Transformation Theorem* ensures that the uniform random variables under such a transformation will give the random variables with density ρ_Z. We now show how this approach can be applied to obtain an appropriate transformation for the normal random variable.

By making the change of variable to polar coordinates (r,θ), with $x_1 = r\cos(\theta)$ and $x_2 = r\sin(\theta)$, we have by the *Probability Density Transformation Theorem* that:

$$\begin{aligned}\frac{1}{2\pi}e^{-\frac{x_1^2+x_2^2}{2}}dx_1dx_2 &= \frac{1}{2\pi}e^{-\frac{r^2}{2}}rdrd\theta. \\ &= d\left(-e^{-\frac{r^2}{2}}+C_1\right)d\left(\frac{\theta}{2\pi}+C_2\right)\end{aligned}$$

where the last line uses the formal rule of differentials for smooth functions $df(x) = f'(x)dx$.

This suggests making the change of variable $u_1 = 1 - e^{-\frac{r^2}{2}}$, $u_2 = \frac{\theta}{2\pi}$ which yields:

$$\frac{1}{2\pi} e^{-\frac{r^2}{2}} r dr d\theta \quad = \quad du_1 du_2.$$

The constants were set to $C_1 = 1$ and $C_2 = 0$ to ensure that $u_1, u_2 \in [0, 1]$.

The transformation $h : (u_1, u_2) \to (x_1, x_2)$ can be obtained by inverting each of the changes of variable above, which yields:

$$x_1 = h_1(u_1, u_2) = \sqrt{-2\log(1 - u_1)}\cos(2\pi u_2)$$
$$x_2 = h_2(u_1, u_2) = \sqrt{-2\log(1 - u_1)}\sin(2\pi u_2).$$

This can be simplified slightly by using that $U_1' = 1 - U_1$ is also distributed uniformly in $[0, 1]$. This gives the *Box-Muller Algorithm* for generating normally distributed random variates:

Box-Muller Algorithm:
If no random variates saved then

1. Generate two independent uniform random variates U_1, U_2.
2. Let $X_1 = \sqrt{-2\log(U_1)}\cos(2\pi U_2)$ and $X_2 = \sqrt{-2\log(U_1)}\sin(2\pi U_2)$.
3. Return X_1 now and save X_2 for the next call to this routine.

else

1. Return the saved variate X_2 from the previous call to this routine.

■

Sampling by Rejection

For some probability densities $g(x)$ it may be difficult to determine analytically an appropriate transformation. In some cases one would also like to sample with relative weights given by a function $g(x)$ without knowing explicitly the normalization constant $Z_0 = \int g(x)dx$ which would give the probability density $\rho(x) = \frac{1}{Z_0}g(x)$ corresponding to the weights. We now discuss approaches by which the desired random variates can be obtained by generating candidate samples which are either accepted or rejected to obtain the desired distribution.

Basic Rejection Method

One approach that can be taken is to sample the probability density by generating variates (X, Y) uniformly in a box and accepting only those which fall in the area under the graph of $g(x)$, see figure 6.

This procedure can be formalized as follows:

1. Sample (X, Y) uniformly in a box $x_1 \leq X \leq x_2$, $0 \leq Y \leq y_2$.
2. If $Y \leq g(X)$ then accept and return $Z = X$, otherwise return to step 1.

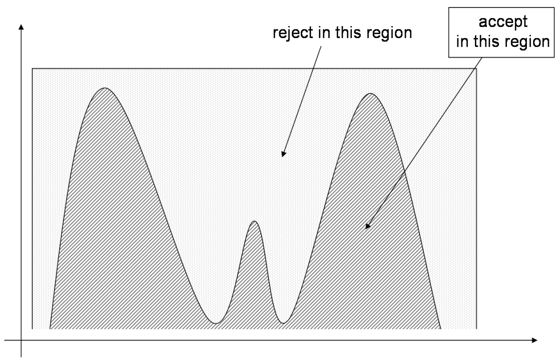


Figure 6: Rejection Method

It can be shown, provided $g(x) > 0$ on some interval, that with probability one a variate will eventually be generated, but this may occur after many rejections making the method inefficient. We define the efficiency of the method as:

$$\text{efficiency} \quad = \quad \frac{\text{number of } X \text{ variates accepted}}{\text{number of } X \text{ variates generated}}.$$

General Rejection Method

If $g(x)$ is sharply peaked the smallest box enclosing the graph of $g(x)$ may be a poor approximation resulting in many rejections and inadequate sampling of the distribution near the peak. One way the efficiency of the method can be improved is to use regions more general than boxes on which to generate the uniform variates. One approach is to sample in the area under the graph of another function $f(x)$, with $g(x) \leq f(x)$, and for which random variates distributed according to $f(x)$ can be readily computed.

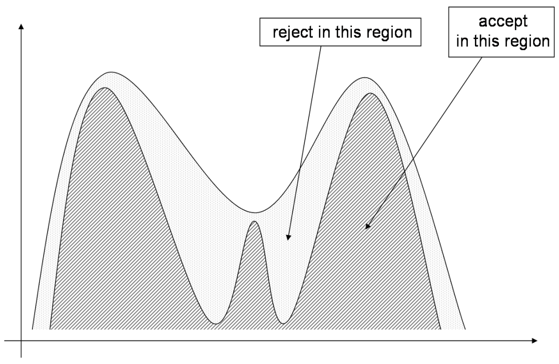


Figure 7: General Rejection Method

This can be formalized as the follow procedure:

1. Generate a variate X having probability density $\rho(x) = \frac{f(x)}{\int f(x)dx}$.

2. Generate a uniform variate $U \in [0,1]$ and accept only if $U \leq \frac{g(x)}{f(x)}$ setting $Z = X$. If rejected return to step 1.

Variance Reduction Techniques

An important consideration in designing effective Monte-Carlo methods is to formulate the estimate for the integrals in terms of the expectation of random variables that have variances as small as possible. The Monte-Carlo method gives the estimate:

$$I = E\left(g(x)\right) \approx \frac{1}{N}\sum_{i=1}^{N} g(X_i).$$

As discussed in the section on the accuracy of the method the error can be estimated by:

$$\text{error} = \frac{\sigma_g}{\sqrt{N}}$$

where

$$\sigma_g^2 = \int_\Omega \left(g(\mathbf{x}) - I\right)^2 \rho(\mathbf{x}) d\mathbf{x}.$$

In the Monte-Carlo estimate recall the random variables X_i were generated having probability density $\rho(x)$.

This suggests one approach by which the Monte-Carlo method rate of convergence can be improved. In particular, to use a probability density $\rho(x)$ for which generation of variates X_i is not too difficult while making σ_g^2 small.

Importance Sampling

Importance sampling is concerned with the choosing $\rho(x)$ for the random variates X_i so that regions which contribute significantly to the expectation of $g(X)$ are sampled with greater frequency. Thus regions where $f(x)$ is large should be sampled more frequently than those regions where $f(x)$ is comparatively very small. A practical consideration in choosing $\rho(x)$ is that the distribution be amenable to efficient generation of the pseudo-random variates.

In the case that $f(x) > 0$ a probability density always exists which gives a Monte-Carlo method having zero error, $\sigma_g = 0$. This is obtained by considering the definition of g:

$$g(x) = \frac{f(x)}{\rho(x)}.$$

From this it follows immediately that $\rho(x) = \frac{f(x)}{I}$ gives a $\sigma_g^2 = 0$. In general, efficiently generating variates with such a density requires I, which if already known would preclude performing the Monte-Carlo estimate in the first place. However, this result suggests a strategy to reduce the variance. In particular, one should try to choose a density which approximates $f(x)/I$ as close as possible.

Example: Let us consider estimating the integral

$$I = \int_{-4}^{4} \frac{e^{-x^2}}{(|x^2-1| + 0.01)^{1/2}} dx.$$

To efficiently estimate the integral using the Monte-Carlo method we see that one approach is to try to sample as closely as possible with a probability density approximating the optimal density $\rho_0(x) = f(x)/I$. We shall now discuss two Monte-Carlo estimates of the integral. In the first we use a standard Gaussian to sample. In the second we use a mixture probability density consisting of a linear combination of Gaussian densities.

Sampling with a Standard Gaussian:
If the Monte-Carlo method uses the density $\rho_1(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$, the prefactor is $\sigma_g = 2.2853$. See Figure 8

for a plot of the density $rho_1(x)$ and the optimal density $\rho_0(x)$. From the plot we see that the Monte-Carlo estimate should be improved if the samples are distributed with higher frequency at the two peaks and in the region where $f(x)$ is non-negligible.

Sampling with a Gaussian Mixture:
We now discuss a probability distribution which generates samples at the two peaks and in the regions where $f(x)$ is non-negligible more frequently. We can readily generate any random variates which have a density which is given as a linear combination of probability densities of form:

$$\rho_{\text{mix}}(x) = \sum_j \alpha_i \rho_j(x).$$

where $\sum_j \alpha_j = 1$. This is done by dividing the interval into subintervals of size α_j and generating the uniform variate $U \in [0, 1]$. If U falls in the j^{th} interval we generate Y_j with density ρ_j and let $X = Y_j$. Thus at the expense of generating an additional uniform variate we can easily generate variates with multi-modal probability distributions.

For the function above we use for example the density with $\sigma_1^2 = 1/100$, $\sigma_2^2 = 1$, $\sigma_3^2 = 1/100$

$$\rho_2(x) = \alpha_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{(x-1)^2}{2\sigma_1^2}} + \alpha_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{(x+1)^2}{2\sigma_2^2}} + \alpha_3 \frac{1}{\sqrt{2\pi\sigma_3^2}} e^{-\frac{x^2}{2\sigma_3^2}}.$$

with $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, $\alpha_3 = 0.8$. Sampling with this density we have $\sigma_g = 1.2184$.

For a plot of the optimal density $\rho_0(x)$, Gaussian density $\rho_1(x)$ and mixture density $\rho_2(x)$, see figure 8. We find that the second method will converge with the prefactor σ_g about half that of the first method. Since the rate of convergence is the inverse to the square root of N we find that the second method requires only $1/2^2 = 1/4$ the number of samples as the first method for a comparable accuracy. We remark that even though the second method required generating the extra uniform random variate, this was more than made up for in the reduction in variance.

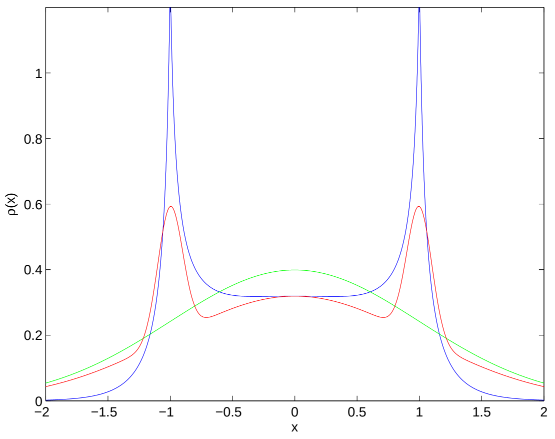


Figure 8: Importance Sampling

References

[1] C. Moler, Numerical Computing with MATLAB, Electronic edition: The MathWorks, Inc., Natick, MA, 2004.

[2] S. K. Park and K. W. Miller, *Random Number Generators: Good ones are hard to find*, Comm. ACM Vol. 32, 1988.

[3] LEcuyer, P., *Random number generation.* In: Gentle, J. E., Haerdle, W., Mori, Y. (Eds.), Handbook of Computational Statistics. Springer- Verlag, Berlin, pp. 3570, chapter II.2, 2004.

[4] W. H. Press and Saul A. Teukolsky and W. T. Vetterling and B. P. Flannery, *Numerical Recipes*, Cambridge University Press, 2002

[5] Cai, D., *Computational Finance Notes*, http://www.cims.nyu.edu, 2004.

[6] Goodman, , *Finance Class Notes*, http://www.cims.nyu.edu, 2003.