Introduction to Monte Carlo methods*

Jonathan Goodman

May 4, 1999

1 Introduction and motivation

A Monte Carlo method is a numerical procedure that deliberately uses random numbers (or numbers that are supposed to mimic random numbers) to compute something. Usually, the quantity being computed is itself not random, although it may be defined in terms of random numbers, for example, the average of a random variable. Equally common is the introduction of random numbers to solve a problem that did not have randomness in its original definition, such as the ground state energy of an atom described by the Schrödinger equation. In some cases we are not interested in averages but in actual random events. This would be the case, for example, in using randomization to make artificial pictures of trees or clouds. I call this process "simulation" to distinguish it from Monte Carlo. The primary difference between simulation and Monte Carlo is that in Monte Carlo, cheating is allowed. Most of advanced Monte Carlo (importance sampling, stratified sampling, etc.) are forms of cheating.

Usually, Monte Carlo methods are used to overcome the "curse of dimensionality" that prohibits the use of "deterministic" methods. For example, suppose you want to compute

$$\int \cdots \int f(x_1, \ldots, x_d) dx_1 \cdots dx_d \ ,$$

where $x = \left(x^1, \ldots, x^d\right)$, and the dimension of integration, d, is large. Such integrals arise in many fields with d in the hundreds or millions or more. A quadrature formula would approximate the integral by a sum

$$\int \cdots \int f(x_1, \ldots, x_d) dx_1 \cdots dx_d \approx \Delta x^d \sum_{i_1=1}^{l} \sum_{i_2=1}^{l} \cdots \sum_{i_d=1}^{l} f(x^1_{i_1}, \ldots, x^d_{i_d}) \ . \tag{1}$$

If we use l quadrature points in each coordinate direction, the total number of points is $n = l^d$. For $l = 10$ and $d = 30$, this is infeasible on present or likely future computers. All grid based methods have the property that the work scales exponentially with the dimension.

Suppose, on the other hand that you can write $f(x) = g(x)\rho(x)$ where $\rho(x)$ is a probability density function for a d dimensional random variable. Suppose also that you can make many independent samples, X_k, from this probability density. Then you can make the Monte Carlo approximation

$$\int f(x)dx = \int g(x)\rho(x)dx = E_\rho\left[g(X)\right] \approx \frac{1}{n}\sum_{k=1}^{n} g(X_k) \ .$$

The error in this approximation usually is on the order of $n^{-1/2}$, which is not great; but it is much better, for large d, than the quadrature formula. In many cases, there are Monte Carlo methods with the property that the work needed to reach a specified accuracy, ϵ, grows polynomially in $1/\epsilon$ and d.

*These are course notes for Scientific Computing, given Spring 1999 at the Courant Institute of Mathematical Sciences at New York University.

A slightly more quantitative way to understand the potential advantage of Monte Carlo for high dimensional integration is to think of error and work estimates. A Monte Carlo method typically has error on the order of $1/\sqrt{n}$ using n samples. The deterministic method (1) may have an order of accuracy of, say, 4. In that case, the error would be proportional to Δx^4. In terms of the total number of points used, that is proportional to $1/l^4 = 1/n^r$ with $r = 4/d$. This says that Monte Carlo has a better power of n (is more accurate) than Simpson's rule in $d = 9$ or more dimensions.

2 Background probability

This section covers background in probability mainly to establish common terminology. A random variable will usually be denoted by a capital letter. A d dimensional random variable,

$$X = (X_1, \ldots, X_d) \ ,$$

has components, X_1, …, X_d, that are one dimensional random variables. Specific possible values of a random variable are often denoted by the corresponding random variable. The probability density function for X is written $\rho(x)$, or $\rho_X(x)$ if there is a need to say what random variable it is the density of. This may be called the "joint density" of the random variables X_1, …, X_d. An event, A, is a set of possible outcomes. We say that A occured if $X \in A$. The probability of this is

$$\mathrm{P}(A) = \mathrm{P}(X \in A) = \int_A \rho(x)dx \ .$$

We often work with the special case when A is a very small set of total volume dx centered about a point x, so small that ρ has neglegible variation over A. Then

$$\mathrm{P}\left(X \in A\right) = \rho(x)dx \ . \tag{2}$$

This relation is particularly useful for computing the effect of transformations and mappings on random variables.

The expected value of a random value $F(X)$ is written

$$\mathrm{E}\left[F(X)\right] = \langle F(X) \rangle = \overline{F(X)} = \int F(x)\rho(x)dx \ .$$

The random variables X_1 and X_2 are independent if the joint density function, $\rho(x_1, x_2)$, is a product of two one variable density functions:

$$\rho(x_1, x_2) = \rho_1(x_1) \cdot \rho_2(x_2) \ .$$

3 Pseudo Random Number Generators

Almost all random quantities in Monte Carlo algorithms are in the end made by manipulating hypothetical independent uniformly distributed random variables, ξ. That is, we assume that we have random variables, ξ_1, ξ_2, …, that are independent and all have probability density $\rho_u(x) = 1$ if $0 \leq x \leq 1$, and $\rho_u(x) = 0$ otherwise. For most of the course, we will just assume that such ξ can be made and not worry about how it's done. In fact, it is impossible to do it exactly. Any deterministic computer algorithm will produce numbers that are correlated with each other to some extent. There are stories about Monte Carlo computations that got the wrong answer because of correlations in the pseudo random numbers. The best random number generators have correlations that are almost impossible to detect and make almost any Monte Carlo method work correctly, as if the numbers were truly random.

Almost all pseudo random number generators in use today work in the following way. There is a "seed", $s = (s^1, \ldots, s^m)$ that consists of m 32 bit computer integers. There is a mapping, $s' = F(s)$, that produces a new seed from a given one and a mapping, $\xi = G(s)$, that produces a number in the interval $[0, 1]$ from a seed. The mappings, F, and G, are usually given by some modular arithmetic. To use such a random number generator, you specify an initial seed, s_1, and then produce the sequence ξ_k by repeatedly using F and G:

$$\xi_k = G(s_k) \ , \qquad s_{k+1} = F(s_k) \ .$$

In C, this would be done[1] (with $m = 2$), by `RanSet(int s1, int s2)` to set the seed, and `xi = Rand()` to get a new uniform random variable independent of the previous ones. The mappings F and G are carried out by the procedure `Rand()`. There probably will be a routine `RanGet(int *s1, int *s2)` that sets `s1` and `s2` to be the current seed. This allows programmers to do a long Monte Carlo run in several batches.

To use a pseudo random number generator, it is important to remember to set the seed in the beginning and not to set it again. If you do two runs with the same initial seed and the same code, you will get exactly the same result. (Pseudo random numbers are not really "random".) This can be very helpful in debugging computer programs. A pseudo random number generator with $m = 1$ (a 32 bit generator) should not be used because it must cycle (produce the same numbers in the same order again) in no more than $k = 2^{32} = 4$ billion steps. Monte Carlo computations often use more than this.

4 Direct sampling

We think of the computer "random number generator" as an oracle, producing independent random variables, ξ_1, ξ_2, ..., each uniformly distributed in the interval $[0,1]$. Direct sampling methods are methods that use these ξ_k to produce independent random variables with other probability distributions. The notation $X \sim \rho(x)$ will mean that the random variable X has a probability density function $\rho(x)$. If X is a random variable, we say that X_k "is a sample of X" if X_k has the same density X has. Direct sampling means producing independent samples of a given random variable.

There are sampling problems for which direct methods are completely infeasible. Fortunately, there are indirect, or "dynamic", sampling methods for many such problems. Dynamic sampling methods produce samples that are not independent and have the correct probability density only in the limit $k \to \infty$. In this way dynamic sampling methods resemble iterative methods for solving equations. We never get the "exact" answer, but our answer can easily be correct to sixteen decimal digits. The best known dynamic sampling method is the Metropolis algorithm.

Direct sampling methods that are entirely deterministic (given the assumed uniform random variables) are called "mapping methods". One of the most famous mapping methods is the Box Muller method for generating standard normals. Rejection methods are the other main family of direct sampling methods. They are interesting and useful, and they are precursors to the Metropolis method.

4.1 Mapping methods

Mapping methods are methods that make a random variable by applying deterministic operations (i.e. mappings) to other random variables. They are the closest Monte Carlo relatives to direct methods in numerical analysis; they produce an exact answer, exactly independent random variables exactly distributed by ρ (assuming exact arithmetic and a perfect random number generator), in a deterministic amount of time.

The term "mapping method" is often used to refer to the special case of a one dimensional random variable, X, given as a function of a single uniform random variable, ξ. Suppose that the function $x = \phi(t)$ is defined and monotone for t in the range $0 < t < 1$ and that $X = \phi(\xi)$. We need to determine the probability density function, $\rho(x)$, for X. This can be done using the one dimensional version of our previous characterization of ρ, (2):

$$\rho(x)dx = \text{Prob}\left[X \in (x, x + dx)\right] \ . \tag{3}$$

To use this, we suppose that a number, x is in the range of ϕ (otherwise, $\rho(x) = 0$). Since ϕ is monotone[2], there is a unique t with $x = \phi(t)$. Moreover, ϕ maps an interval of length dt around t to an interval of length dx around x, where $dx = \phi'(t)dt$. The probability that X lands in the interval of length dx around x is the same as the probability that ξ is in the interval of size dt around t. This leads to the formula

$$\rho(x) = 1/\phi'(t) \ , \quad \text{where} \quad x = \phi(t) \ . \tag{4}$$

[1]The actual procedure and variable names will depend on the random number generator you use.

[2]This is the only place where the monotonicity assumption is used

This one dimensional mapping method can be explained in terms of the "distribution function"

$$F(x) = \text{Prob}\,[X < x] = \int_{-\infty}^{x} \rho(x')dx' \quad .$$

This function is monotone increasing (or, at any rate, nondecreasing) and maps the real line to the interval $[0,1]$. Define ϕ to be the inverse function for F, that is, $x = \phi(t)$ meand that $t = F(x)$. Then $X = \phi(\xi)$ gives $X \sim \rho$. Indeed, if $F(x) = t$, which is the same as $\phi(t) = x$, then $\text{Prob}[X < x] = \text{Prob}[\xi < t] = t = F(x)$, as claimed. Thus, the algorithm for finding an X with density ρ is to generate a uniform ξ and then solve $F(X) = \xi$. If we can compute the indefinite integral $F(x)$ and solve the equation $F(x) = t$ to find x for given $t \in [0,1]$, then we can sample from ρ. This is the case for the exponential random variable (see below).

If we don't have closed form expressions for F or ϕ, we can tabulate them. The work that it takes to make the table by numerical integration will be dwarfed by the time taken to generate thousands or millions of samples. Tabulation may not be practical when ρ depends on several parameters whose values are not known in advance or change from sample to sample. For example, we can generate gaussian random variables if we can compute the inverse of the error function

$$\text{erf}(x) = \sqrt{\frac{2}{\pi}} \int_0^x e^{-y^2/2} dy \quad .$$

I recently downloaded a very fast and accurate shareware routine for computing the inverse of $t = \text{erf(x)}$ (you supply t, it returns x) over the internet. This would give a competitive (in speed, not elegance) algorithm to Box Muller (see below).

4.1.1 The Exponential Random Variable

An exponential random variable with mean μ is a random variable with density

$$\rho(x) = \frac{1}{\mu} e^{-x/\mu} \quad , \qquad \text{if } x > 0, \text{ and } \rho(x) = 0 \text{ otherwise.} \tag{5}$$

If X is an exponential with mean 1 then $Y = \mu X$ is exponential with mean μ (check this), so we need only generate an exponential with mean 1. Exponential random variables are important partly because they are a good way to simulate a continuous time Markov process with discrete state space.

It is natural to think of an exponential random variable as the random amount of time one waits for something to happen. The defining property of the *exponential* waiting time is that it has no knowledge of the past (This is the Markov property.). If you have waited a time, t and the event has not happened ($T > t$), then it is as though you had not waited at all. If a light bulb failure has an exponential distribution, than any bulb that has not failed is good as new. Such models are used commercially, for example to predict the MTBF (mean time before failure) of hard disks. The manufacturer runs a number of drives and records the failures in a few months. These observations are used to fit an exponential probability density and determine an empirical parameter, μ, which then appears in the advertisements. If a company actually tested its drives until half of them failed (on the order of 4 years) before advertising them, it would miss the market completely.

The formula (5) is a consequence of the Markov property. Suppose $\rho(t)$ is the density function for a random variable, T, with the Markov property. The probability of breaking immediately within time dt is $\rho(0)dt$. The Markov property states that this is also the probability of breaking in time interval $(t, t+dt)$, given that it has lasted until time t. From the formula for conditional probability, this leads to

$$\begin{aligned}
\rho(0)dt &= \text{Prob}\,[T \in (t, t+dt) \mid T > t] \\
&= \frac{\text{Prob}\,[T \in (t, t+dt) \ \text{ and } \ T > t]}{\text{Prob}\,[T > t]} \\
&= \frac{\rho(t)dt}{1 - F(t)} \quad ,
\end{aligned}$$

where $F(t) = \int_{-\infty}^{t} \rho(t')dt'$ is the distribution function for ρ. But, $\rho(t) = -F'(t)$ and $F(0) = 0$, so we have the differential equation and boundary condition for F

$$\rho(0)\,(1 - F(t)) = -F'(t) \quad , \qquad F(0) = 0 \quad ,$$

which leads to $F(t) = 1 - e^{-t/\mu}$ and then to (5).

The direct sampling method for the exponential random variable is easier to explain than the exponential random variable itself. If $X = -\log(\xi)$ and ξ is uniform, then (3) shows that X is exponential. Direct application of the probability distribution formalism discussed above leads to the formula $X = -\log(1-\xi)$, which also works, since $1 - \xi$ is also a uniformly distributed random variable.

4.1.2 The Box Muller Algorithm for Normals

A Gaussian random variable (also called "normal") with mean μ and variance σ^2 is a random variable with density function

$$\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \quad .$$

The "standard normal" random variable is a Gaussian with mean 0 and variance 1.

The Box Muller method, which is a clever mapping method, makes two independent standard normal random variables from two independent uniforms. The trick is that X and Y are independent standard normals if and only if their joint density function is

$$\rho(x,y) = \frac{1}{2\pi} e^{-(x^2+y^2)/2} \quad .$$

If we write (X,Y) in polar coordinates, $X = R\cos(\Theta)$, $Y = R\sin(\Theta)$ then (X,Y) will have the density (3) if R and Θ have the joint density

$$\tilde{\rho}(r,\theta) = \frac{1}{2\pi} r e^{-r^2/2} \quad ,$$

where, of course, θ is restricted to a range such as $0 < \theta < 2\pi$. This can be sampled by taking Θ to be uniformly distributed in the interval $[0, 2\pi]$ (i.e. $\Theta = 2\pi\xi_1$), and R by the mapping formula $R = \sqrt{-2\log(\xi_2)}$.

The Box Muller algorithm makes independent standard normals in pairs. The first two uniforms, ξ_1, and ξ_2, make the first two standard normals, X_1, and X_2. Then ξ_3 and ξ_4 make X_3 and X_4, and so on. Many large Monte Carlo codes avoid repeated function calls to random number generators by generating large lists of random numbers (say, $10,000$) each call. The Box Muller can easily be used to fill list of standard normal random variables.

4.1.3 Green's Function for the Klein Gordon Operator

The Klein Gordon operator is

$$-\triangle + m^2 \quad .$$

Klein and Gordon used it (actually, it's square root) in an attempt to make a relativistic version of Schrödinger's wave equation in quantum mechanics. The same equation was used by Debye and Hückel (where it is called the Debye and Hückel equation) to study screening of charges by a dialectric material. It has many other applications, including Monte Carlo, where the dimension may be much larger than 3. The Green's function for the Klein Gordon operator, $G(x)$, is defined to be the solution of

$$-\triangle G + m^2 G = \delta(x) \quad . \qquad (6)$$

The solution in 3 dimensions is $G(x) = \frac{1}{|x|} e^{-m|x|}$. The solution in 2 dimensions is a "modified Bessel function": $G(x) = K_0(m\,|x|)$. By integrating both sides of (6), and assuming that G decays rapidly for large $|x|$, we find that

$$\int G(x)dx = \frac{1}{m^2} \quad .$$

The probability density function we want to sample is $\rho(x) = \frac{1}{m^2}G(x)$. We will shortly see that $G(x) > 0$ for all X. We may at first be discouraged since we don't even have a formula for ρ.

The trick for sampling ρ, and for many related density functions, is to find a suitable integral representation. Integral representations for special functions in mathematical physics can usually be found from physical properties of the definition of the function. In this case, consider a generalization of (6) to more general right hand sides and operators:

$$Au = f \ . \tag{7}$$

Here A represents the Klein Gordon operator, u the Green's function, and f the delta function. If A is positive definite (as it is in the Klein Gordon case), we solve (7) using the dynamical system

$$\dot{v} = -Av \ , \quad v(0) = f \ . \tag{8}$$

If $v(t) \to 0$ quickly enough as $t \to \infty$, then

$$u = \int_{t=0}^{\infty} v(t)dt \tag{9}$$

satisfies (7). This can be checked by applying A to both sides of (9) and using the relations (8). In the case of the Klein Gordon operator, (8) becomes

$$\partial_t v = \triangle v - m^2 v \ , \quad v(x,0) = \delta(x) \ . \tag{10}$$

If the m^2 term were not present, the solution would be given by the fundamental solution of the heat equation in d dimensions, namely

$$\frac{1}{(4\pi t)^{d/2}} e^{-|x|^2/4t} \ .$$

The "decay term", $-m^2 v$, is handled by including an additional exponential decay factor. Therefore, the solution to (10) is

$$v(x,t) = \frac{1}{(4\pi t)^{d/2}} e^{-|x|^2/4t} e^{-m^2 t} \ .$$

Finally, we can take the integral (9) to get the desired integral representation for G, and therefore (after adding a factor $1/m^2$) ρ:

$$\rho(x) = \frac{1}{m^2} \int_{t=0}^{\infty} m^2 e^{-m^2 t} dt \cdot \frac{1}{(4\pi t)^{d/2}} e^{-|x|^2/4t} \ . \tag{11}$$

The integral representation (11) suggests a strategy for sampling ρ. Notice that the first factor on the right is an exponential density with mean $1/m^2$ while the second is a gaussian density in d dimensions with each component having variance $2t$. The direct sampling algorithm is now: first, pick a random time, T, from the exponential density function with mean $1/m^2$ using the log mapping above; second, pick $X = (X_1, \ldots, X_d)$ by taking Y_k to be independent standard normal random variables made using the Box Muller algorithm, and $X_k = \sqrt{2T}Y_k$.

In this sampling algorithm, the random variable, T is not reported to the user, but it makes the method work. We have turned a seemingly hard d dimensional sampling problem into a much easier $d+1$ dimensional sampling problem. Some of the most effective innovative Monte Carlo methods developed in recent years, among them umbrella sampling and cluster algorithms, are based on clever enlargements of the sampling space.

4.2 Sampling by Rejection

Rejection methods sit between the above truly direct methods and dynamic sampling methods such as the Metropolis algorithm. They produce exactly independent samples with the exact probability density specified, but the number of steps needed to do this cannot be exactly predicted in advance. There is considerable freedom in designing a rejection algorithm to sample a given density, ρ.

The Monte Carlo practitioner occasionally must spend some time and effort optimizing parameters or otherwise tinkering to get a rejection method that is reasonably efficient[3]. Rejection methods are often in the innermost loop of a Monte Carlo code, so their efficiency determines the running time of the code as a whole.

To generate a random variable with density $\rho(x)$, the rejection method uses independent random variables sampled from an auxiliary density, $\rho_0(x)$ and a "acceptance probability", $p(x)$. The method has two steps

Trial: generate a "trial" random variable, $X \sim \rho_0$. All trials are independent.

Rejection: "accept" the trial with probability $p(X)$. If X is accepted, it is the random variable generated by the algorithm. If X is rejected, go back to the trial step, generate a new (independent) X.

Accepting with probability p is done on the computer by comparing p to an independent uniform random variable. If p is larger (an event with probability p), accept. This trial and rejection process is repeated until a random variable is accepted. If ζ is the probability of getting an acceptance on any given trial, then the expected number of trials needed to get an acceptance is $1/\zeta$.

The following ghastly one line code C does all this:

```
while( unif() > acc_prob( X = trial()) );
```

This assumes that `float unif()` when called, returns a uniformly distributed random number, that `float acc_prob( float x)` returns the acceptance probability for trial variable `x`, and that `float trial()` returns a sample from the trial density, ρ_0. A code that works from lists could look like this:

```
while( unif[u\count++] > acc_prob( X = trial[t_count++]) ) {
    if ( u_count >= U_LIST_SIZE ) unif_refil();
    if ( t_count >= T_LIST_SIZE ) trial_refil();
  }
```

In the second code fragment, `float unif[U_LIST_SIZE]` and `float trial[U_LIST_SIZE]` are arrays rather than subroutines. the procedures `void unif_refil()` and `void unif_refil()` refill the lists, using random number generators.

We can determine the probability density function for the eventual accepted X using the laws of conditional probability. It is given by

$$\begin{aligned}
\rho(x)dx &= \text{Prob}[\text{ accepted } X \in (x, x+dx)] \\
&= \text{Prob}[\text{ trial } X \in (x, x+dx) \mid \text{accepted } X] \\
&= \frac{\text{Prob}[\text{ trial } X \in (x, x+dx) \text{ and accepted } X\ ]}{\text{Prob}[\text{ got an acceptance }]} \\
&= \frac{1}{\zeta}\rho_0(x)dx \cdot p(x) \ ,
\end{aligned}$$

where ζ is the probability of getting an acceptance on a given trial, as above. Putting this together gives

$$p(x) = \zeta \frac{\rho(x)}{\rho_0(x)} \ . \tag{12}$$

In order for $p(x)$ as given in (12) to be a probability, it must be between 0 and 1. In order for this to be possible (with a fixed ζ), we must have

$$\rho(x) \le \frac{1}{\zeta}\rho_0(x) \ .$$

This requirement limits the possibilities for ρ_0; the tails of the trial distribution must be at least as "fat" as the tails of the distribution you want to sample. For example, one can sample a standard normal by rejection from an exponential (the ratio $e^{-x^2/2}/e^{-x}$ is bounded), but one cannot sample an exponential by rejection from a gaussian (the ratio $e^{-x}/e^{-x^2/2}$ is not).

[3]This is not fun.

The rejection algorithm has the advantage that it can be applied even when the probability densities are known only up to a multiplicative constant. This situation arises, for instance, whenever the Gibbs Boltzmann distribution (the "canonical ensemble") is used. In that notation, suppose $\phi(x)$ and $\phi_0(x)$ are two energy functions with corresponding probability densities

$$\rho(x) = \frac{1}{Z} e^{-\phi(x)} \quad , \quad \text{and} \quad \rho_0(x) = \frac{1}{Z_0} e^{-\phi_0(x)} \quad .$$

If we write the acceptance probability also in exponential form:

$$p(x) = e^{-\psi(x)} \quad ,$$

then the discussion leading to (12) now gives the formula

$$\psi(x) = \phi(x) - \phi_0(x) + \alpha \quad ,$$

where

$$\alpha = \log(Z) - \log(Z_0) - \log(\zeta) \tag{13}$$

should be taken as small as possible, subject to the constraint that $\psi(x) \geq 0$ for all x (so the $p(x)$ is a probability). If we work with the functions, ϕ, ϕ_0, and ψ, then only α need ever be known, not Z, Z_0, or ζ.
Example: Let us consider the problem of generating a standard normal by rejection from an exponential. To begin with, we will sample from the exponential density with mean one and do rejection to get a random variable whose density function is the positive half of the gaussian, that is

$$\rho(x) = \begin{cases} \frac{2}{\sqrt{2\pi}} e^{-x^2/2} & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \quad , \text{ and } \quad \rho_0(x) = \begin{cases} e^{-x} & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \quad .$$

From the treatment where the constants are supposed to be known we get acceptance probability

$$p(x) = \zeta \frac{\sqrt{2}}{\sqrt{\pi}} e^{x - x^2/2} \quad .$$

The largest possible ζ that gives $p(x) \leq 1$ for all $x > 0$ is

$$\zeta = \frac{\sqrt{\pi}}{\sqrt{2}} e^{-1/2} = .7602 \quad ,$$

which is also the efficiency of the rejection method: the probability of getting an acceptance on a given trial is 76%.
Example. We want to sample from the density

$$\rho(x) = \frac{1}{Z} e^{-x^4/4}$$

by rejection from a gaussian. At first we consider rejection from a standard normal. In that case,

$$\psi(x) = \frac{x^4}{4} - \frac{x^2}{2} + \alpha \quad . \tag{14}$$

To make sure $\psi \geq 0$, we compute that the minimum of ψ is taken at $x = \pm 1$ ($\psi' = 0 \Rightarrow x^3 = x$). Thus, if $\alpha = -1/4$, then $min_x \psi(x) = 0$, as needed.

We try to improve the efficience of this rejection method by rejecting from a normal with variance $\sigma^2 \neq 1$. This makes $\phi_0(x) = x_2/x\sigma^2$, so (14) becomes

$$\psi(x) = \frac{x^4}{4} - \frac{x^2}{2\sigma^2} + \alpha \quad ,$$

which now is minimized at $x = \pm 1/\sigma$, so $\alpha = -1/4\sigma^4$. To optimize the acceptance probability, ζ, by the best choice of σ. we use $Z_0 = \sqrt{2\pi\sigma^2}$, so , from (11),

$$\log(\zeta) = \log(Z) + \frac{1}{4\sigma^4} - \log(\sigma) + \frac{1}{2}\log(2\pi) \ .$$

To maximize this expression, we differentiate with respect to σ and set the derivative to zero. Since Z and 2π are independent of σ, this gives $\sigma_{\text{opt}} = 1$: our standard normal rejection was already optimal.

A fortunate property of the rejection algorithm is that it may be applied even when certain constants of integration are unknown. This situation arises often in physics and chemistry, for example, when we have an "energy", $\phi(x)$, and an "inverst temperature", β. The probability density then is given by Gibbs' formula:

$$\rho_\beta(x) = \frac{1}{Z([beta)} e^{-\beta\phi(x)} \ . \tag{15}$$

The normalization constant, $Z(\beta)$, which is chosen so that ρ_β has total integral equation to one:

$$Z(\beta) = \int e^{-\beta\phi(x)} dx \ ,$$

is the "partition function". Often we want to sample ρ_β without knowing $Z(*\beta)$. This is possible using the rejection algorithm.

The example

$$\phi(x) = \frac{x^4}{4}$$

will clarify the method. We wish to sample by rejection from a Gaussian with mean zero and variance σ^2 (to be determined later) as a trial. We attempt rejection from a Gaussian because we already know how to produce Gaussians.

As before, the acceptance probability is determined by

$$\rho_\beta(x) = \frac{1}{\zeta} p(x) \frac{1}{\sqrt{2\pi\sigma^2}} e^{-x^2/2\sigma^2} \ , \tag{16}$$

where ζ is teh efficiency of the rejection method in this case. Using the formula for $\rho_\beta(x)$ and taking the log of both sides of (16) leads to

$$-\log(Z) - \frac{\beta x^4}{4} = -\log(\zeta) + \log(p(x)) - \frac{1}{2}\log(2\pi\sigma^2) - \frac{x^2}{2\sigma^2} \ ,$$

or, collectin terms,

$$\log(p(x)) = \frac{x^2}{2\sigma^2} - \frac{\beta x^4}{4} + C \ . \tag{17}$$

Here C is an unknown constant that depends on σ an dβ but not on x. Requiring that $p(x) \leq 1$ for all x leads to

$$\frac{x^2}{2\sigma^2} - \frac{\beta x^4}{4} + C \leq 0 \quad \text{for all } x. \tag{18}$$

Assuming taht β is positive, the left side has a local minimum at $x = 0$ and symmetrically positioned local maxima. These local maxima are found by solving

$$\partial_x \left(\frac{x^2}{2\sigma^2} - \frac{\beta x^4}{4} \right) = 0 \quad ,$$

which leads to

$$x = \pm \frac{1}{\sqrt{\beta\sigma^2}} \ .$$

5 Data analysis and error bars

Every Monte Carlo computation must include an *error bar*, which is an estimate of the order of magnitude of the difference between the computer result and the actual answer. In some cases, error bars are difficult to construct, but there is a simple approach that is appropriate to most Monte Carlo computations using direct sampling methods. This is the error bar constructed from the sample variance through the central limit theorem.

Most data analysis and error estimation methods in Monte Carlo are taken from statistics. After all, statistics is the study of inference from random data, which is what we are trying to do in Monte Carlo. Many of the hardest problems in statistics do not arise in Monte Carlo data analysis. For example, the data sets in Monte Carlo computations are usually large enough for large sample size approximations, such as the central limit theorem, to be accurate. There are no missing data points, nor have the results been mistranscribed.

The simplest version of the central limit theorem is about sums of independent random variables drawn from the same density. Suppose X is a random variable with mean $\mathrm{E}(X) = \mu$ and variance $\mathrm{var}(X) = \mathrm{E}[(X-\mu)^2] = \sigma_x^2$, and that $X_1, \ldots, X_n$ are independent samples of X. The Monte Carlo estimate of μ is the sample mean

$$\hat{\mu} = \overline{X} = \frac{1}{n}\sum_{k=1}^{n} X_k \quad . \tag{19}$$

Statisticians often use a hat to indicate a statistical estimate of a variable. Calculating from (19) and the assumption that the X_k are independent, we find the variance of the estimator

$$\mathrm{var}(\hat{\mu}) = \frac{1}{n}\sigma_x^2 \quad . \tag{20}$$

The central limit theorem says that, for large n, $\hat{\mu}$ is very nearly a gaussian random variable. One gaussian random variable with mean μ and variance σ^2 may be written $\mu + \sigma Z$ where Z is a standard normal random variable. In our situation, the central limit theorem may be stated[4]

$$\hat{\mu} \approx \mu + \frac{\sigma_X}{\sqrt{n}} Z \quad \text{where } Z \sim \mathcal{N}(0,1). \tag{21}$$

This shows that the error is on the order of $1/\sqrt{n}$, as has been indicated often above.

Error bars reported in Monte Carlo computations are usually "one standard deviation" error bars. In this case, the multiplier of Z in (21). This is not a bound for the error, but an indication of the likely size of the error. The actual mean will be outside a one standard deviation "error bar" about a third of the time. If this troubles you, you may use two standard deviation error bars (to include μ 95% of the time) or even three standard deviation error bars. If you do this, you should say so very clearly, as most Monte Carlo practitioners expect reported error bars to be one standard deviation error bars. If the data are presented graphically, the plot should have a big dot indicating $\hat{\mu}$ and a vertical line extending from $\hat{\mu} - \sigma_x/\sqrt{n}$ to $\hat{\mu} + \sigma_x/\sqrt{n}$. This line, or bar, is the error bar. If you report the data in a table, you should write, say $23.45 \pm .321$ to indicate that $\hat{\mu}$ is 23.45 and $\sigma_x/\sqrt{n}$ is .321.

In practice, it is unlikely that you know σ_X if you don't know μ. Instead, we estimate σ_X from the Monte Carlo data. Since σ_X^2 is the expected value of $(X-\mu)^2$ a reasonable estimator might be

$$\widehat{\sigma_x^2} = \frac{1}{n}\sum_{k=1}^{n} (X_k - \hat{\mu})^2 \quad . \tag{22}$$

Some statisticians say that we should use $1/(n-1)$ instead of $1/n$ in (22), with the argument that (22) as it stands is slightly biased: the expected value of $\widehat{\sigma_x^2}$ is equal to σ_x^2 only with $1/(n-1)$. I answer that if n is so small that that makes a difference, you have too few samples to be meaningful. A generally satisfactory error bar is (21) with σ_X estimated from (22).

[4]The notation $\mathcal{N}(\mu, \sigma^2)$ refers to a normal, or Gaussian, random with mean μ and variance σ^2. Thus, $Z \sim \mathcal{N}(0,1)$ means Z is a standard normal random variable.

6 Variance reduction methods

Given the formula (21), there are two ways to get a smaller error. One is to increase n. This calls for more waiting, or a bigger computer budget, but not more thinking. The other approach is to reduce σ_X. How can this be done? Find a different random variable, Y with the same expected value but a different (hopefully lower) variance. The books on Monte Carlo discuss many clever approaches to variance reduction: importance sampling, antithetic variates, control variates, and stratified sampling to name a few.

6.1 Importance sampling

Importance sampling is the most blatant form of cheating in Monte Carlo. You know that if $X \sim \rho_0(x)$, then $A = E[F(X)]$ is the number you want. However, you prefer to sample a different probability density, $\rho_1(x)$. This is allowed, provided that we suitably modify the "observable", F, as the following algebra shows.

$$\begin{aligned}
A &= E_{\rho_0}\left[F(X)\right] \\
&= \int F(x)\rho_0(x)dx && \text{(the definition)} \\
&= \int F(x)\frac{\rho_0(x)}{\rho_1(x)}dx && \text{(the main idea)} \\
&= \int \tilde{F}(x)\rho_1(x)dx \\
A &= \mathrm{E}_{\rho_1}\left[\tilde{F}(X)\right] && \text{where } \tilde{F}(x) = F(x)\frac{\rho_0(x)}{\rho_1(x)}. \qquad (23)
\end{aligned}$$

People often write $\mathrm{E}_{**}[\cdots]$ to denote the expected value of the quantity $\cdots$ under conditions $**$. Here, for example, $\mathrm{E}_{\rho_0}[\cdots]$ means the expectation value when $X \sim \rho_0$.

Importance sampling is often used when most of the contribution to A comes from rare events. For example, suppose we want to compute $\mathrm{E}[Z^8]$ when Z is a standard normal. From Figure 1 and Figure 2, it is clear that about 10% of the integral defining $\mathrm{E}[Z^8]$ comes from $z >\approx 3.8$. The probability of this is on the order of 10^{-4}. That is, if you take $n = 10,000$ samples, you have a reasonable chance of underestimating $\mathrm{E}[Z^8]$ by more than 10%. The general rule of thumb that the error is on the order of $1/sqrtn$ would predict 1% errors rather than 10%. Figure 3 has results of this experiment. Note that 17 out of 48 times (about 35% of the times) the error bar interval does not contain the exact answer.

To fix this situation, we might to choose a density, $\rho_1(z)$ that has more weight in the regions about $z \approx 3$ where the integrand is large. One simple way to attempt this is to choose $\rho_1(z)$ to correspond to a gaussian with mean 0 and standard deviation 3. This would mean, using (23)

$$\rho_1(z) = \frac{1}{3\sqrt{2\pi}}\exp(-z^2/(2\cdot 9)) \quad , \qquad \tilde{F}(z) = z^8 \cdot 3\exp(-\frac{4}{9}z^2) \ . \tag{24}$$

From this formula, it is clear what importance sampling is doing in this case. First we change the rules to allow many more samples to be generated in the most important places, then we compensate by those samples vey little weight. These two steps preserve the expected value but reduce the variance considerably.

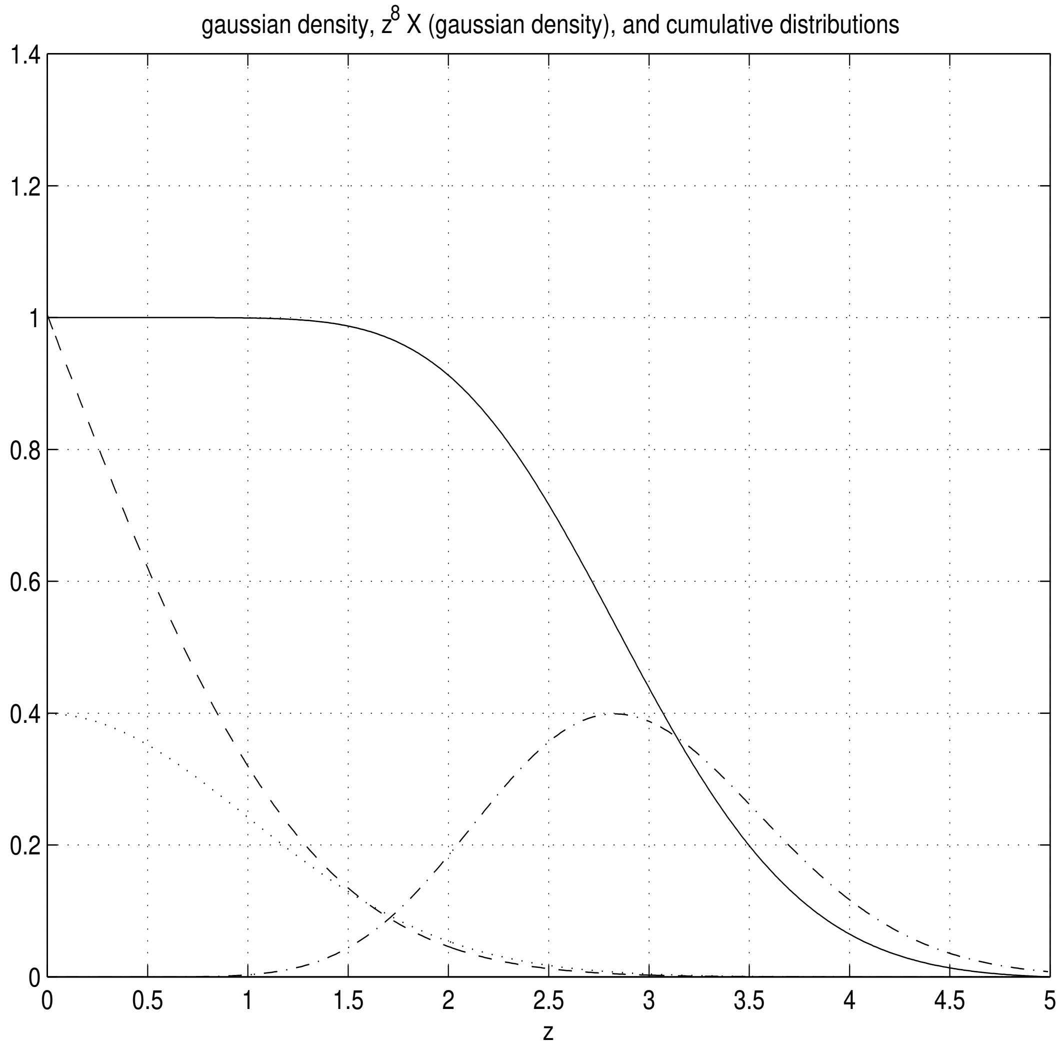

Figure 1: Illustration of the need for importance sampling. The dotted line is the standard normal density, $\rho_0(z)$ and the dot-dashed line is the density of the integrand $z^8\rho_0(z)$, normalized to have the same maximum height as $\rho_0(z)$. The dashed line is $2\int_z^\infty \rho_0(z'dz'$, the two sided cumulative density of the standard normal. The solid line is $2\int_z^\infty z'^8\rho_0(z')dz'$, normalized to be 1 when $z=0$.

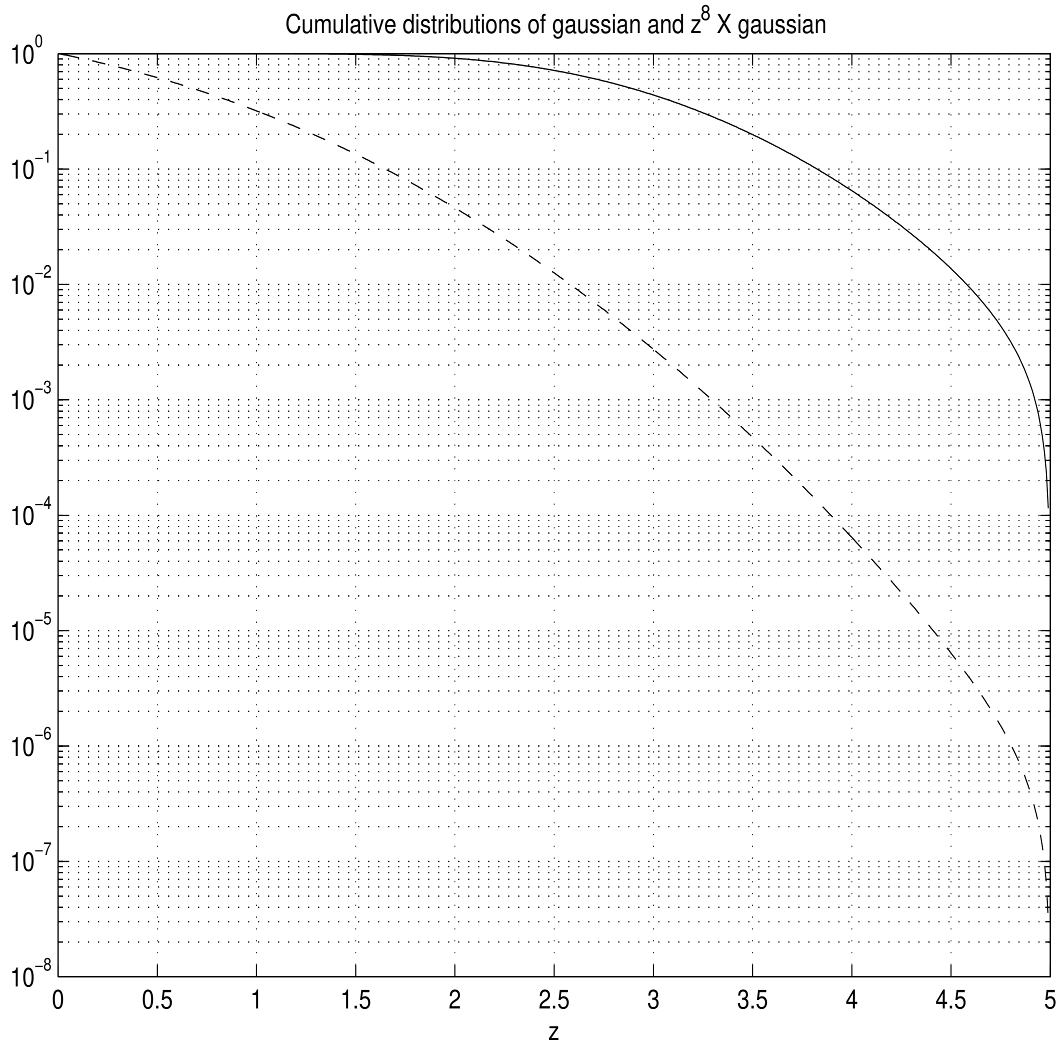

Figure 2: Illustration of the need for importance sampling. A semilog plot of the two distribution functions. Note that when $z = 3.8$, the $z^8\rho_0(z)$ distribution is about .1 while the $\rho_0(z)$ distribution is about $1/10,000$. This means that there is about one chance in $10,000$ of getting a sample where it needs to be to sample the top 10% of the mass of the integrand, making it very likely that the Monte Carlo answer is at least 10% too low if we use just $10,000$ samples.

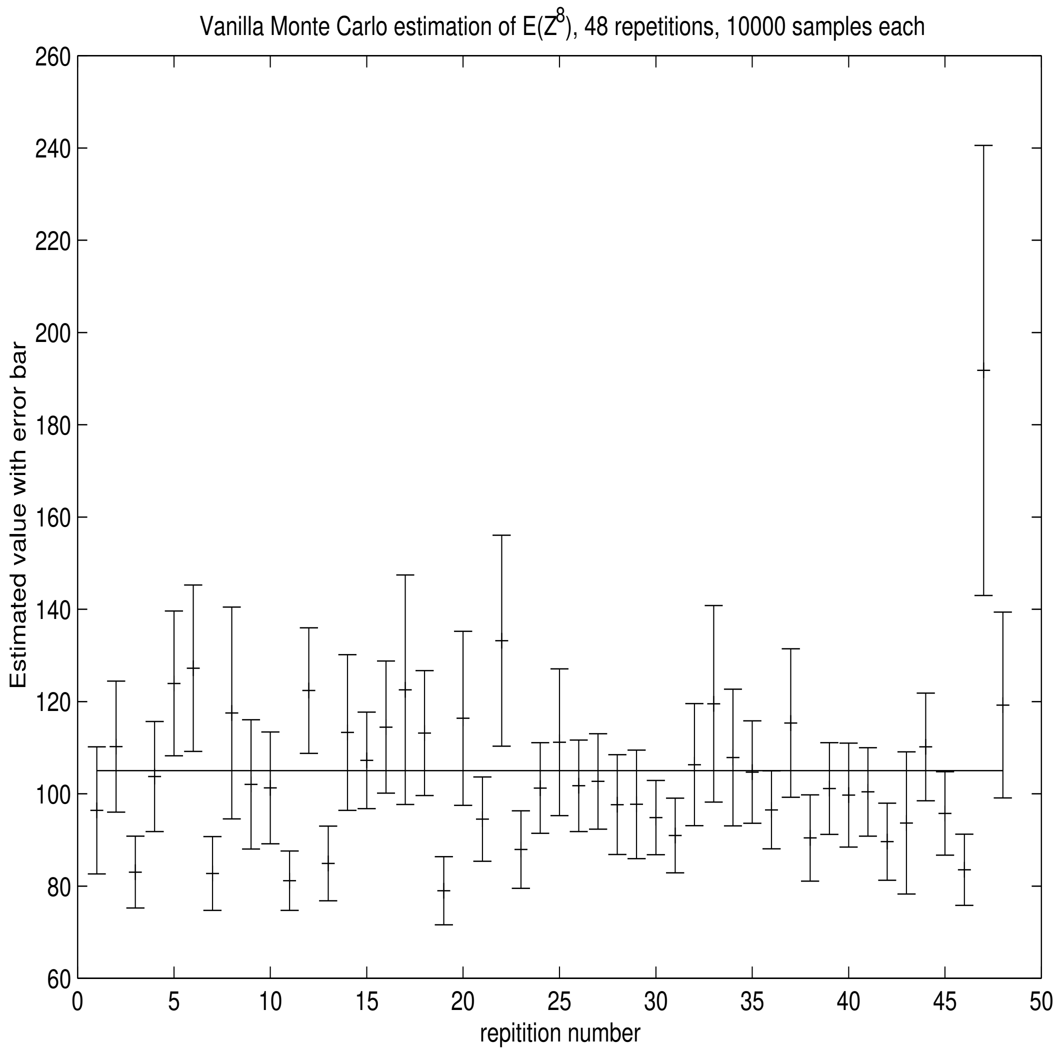

Figure 3: Results of vanilla Monte Carlo for $\mathrm{E}(Z^8)$. The figure shows 48 indpenedent experiments. Each experiment used $10,000$ samples. Typical errors are on the order of 10%, a fortuitous agreement with my analysis. The error bars are one standard deviation error bars. The exact answer is 105, which is indicated by the horizontal line.

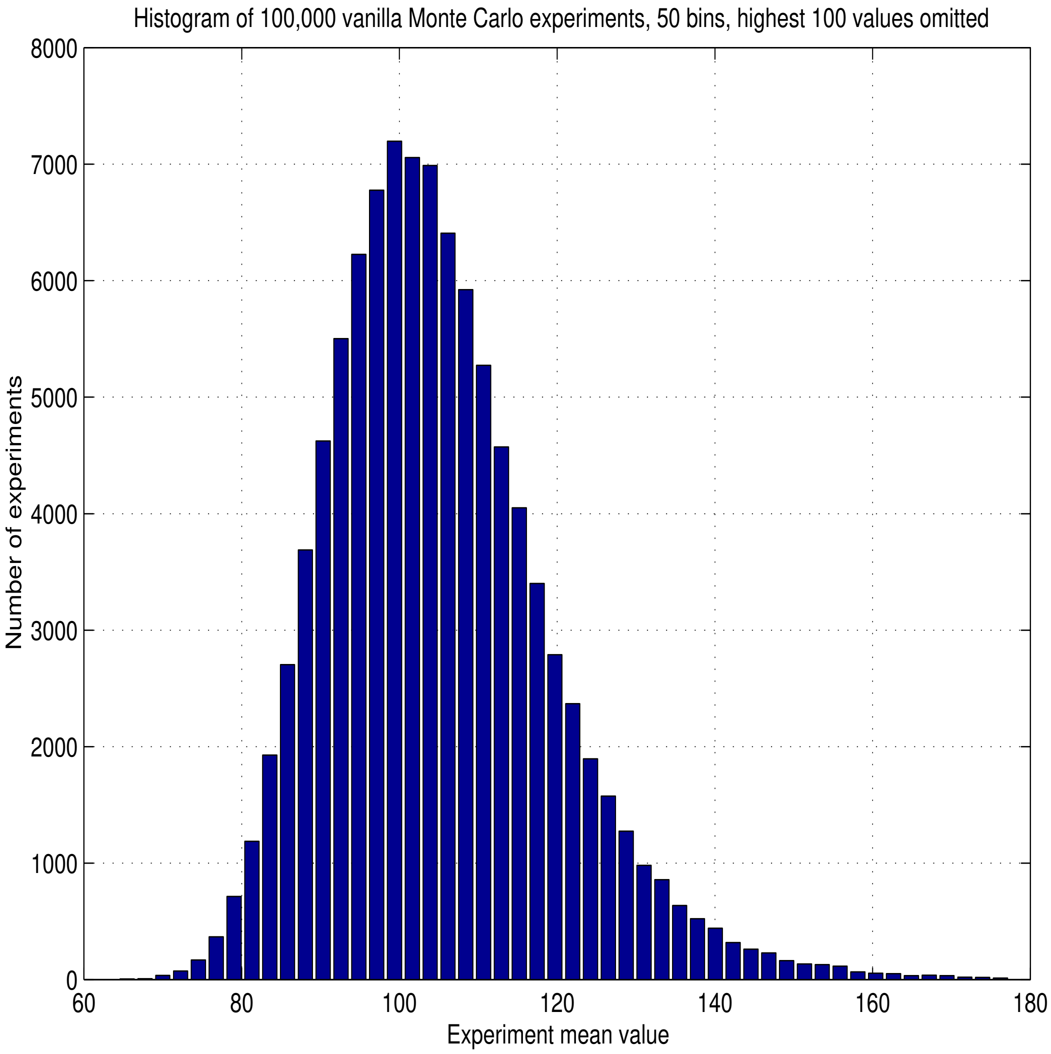

Figure 4: A continuation of the experiment from Figure 3. This time there are $100,000$ repetitions of the $10,000$ sample Monte Carlo computation. Note the departure of this histogram from gaussian, with the left tail larger than the right one. Also note that the hundred largest values were left out. The largest of these was 381, which is far from the right answer.