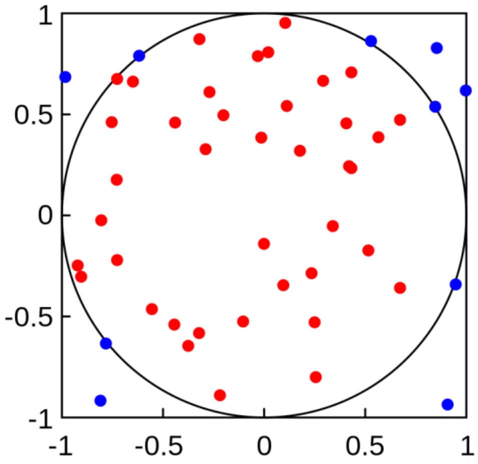
Submitted By:
Dr. Gargi Tiwari
Guest Asst. Prof. (Physics)

MPHYCC-5 Modelling and Simulation

MONTE CARLO APPLICATIONS

We have an idea of how to use the computer to generate pseudorandom numbers, we build some confidence that we can use these numbers to incorporate the element of chance into a simulation. We can do this first by simulating a random walk. After that, we can show how knowing the statistics of random numbers leads to the best way to evaluate multidimensional integrals.

An illustration of Monte Carlo integration



In this example, the domain D is the inner circle and the domain E is the square. Because the square's area (4) can be easily calculated, the area of the circle (π*1.02) can be estimated by the ratio (0.8) of the points inside the circle (40) to the total number of points (50), yielding an approximation for the circle's area of 4*0.8 = 3.2 ≈ π.

Monte Carlo Estimator: Evaluating Integrals

Before you start, it is important that you understand the **law of the unconscious statistician**. It states that the expected value of a function of a random variable f(X) can be defined as:

$$E[f(X)]=\int f(X)PX(X)dX$$

Where PX is the probability distribution of the random variable X. This is hopefully something you understand well. You need to be perfectly comfortable with this idea to understand Monte Carlo integration.

At this point you should also be familiar with the concept of **variance** and **standard deviation.**

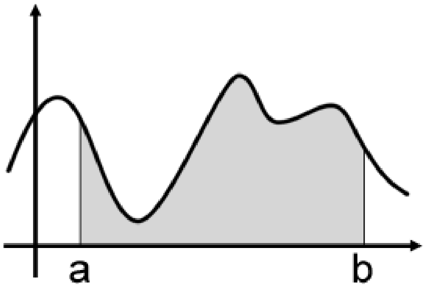
$$\mathrm{Var}(X) = E[(X-E[X])^2],$$

$$=E[X^2]-E[X]^2.$$

We will be using these formulas further down, so it also important that you understand them.

The principle of a basic Monte Carlo estimation is this: imagine that we want to integrate a one-dimensional function f(x) from a to b such as:

$F=\int_a^b f(x)dx.$



© www.scratchapixel.com

Figure 1: the integral over the domain [a,b] can be seen as the area under the curve.

As you may remember, the integral of a function f(x) can be interpreted as calculating the area below the function's curve. This idea is illustrated in figure 1. Now imagine that we just pick up a random value, say x in the range [a,b], evaluate the function f(x) at x and multiply the result by (b-a). Figure 2 shows what the result looks like: it's another rectangle (where f(x) is the height of that rectangle and (b-a) its width), which in a way you can also look at a very crude approximation of the area under the curve. Ofcourse, we may get it more or less right. If we

© www.scratchapixel.com

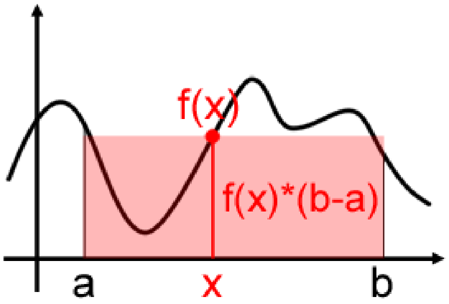
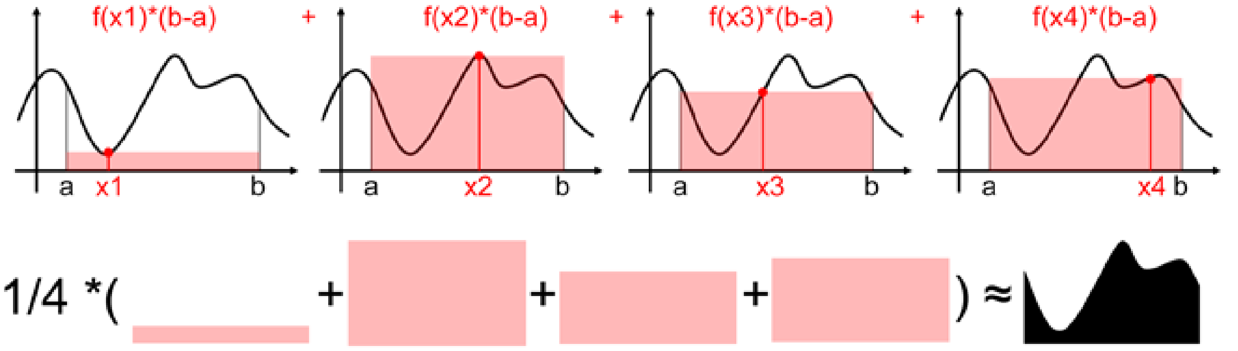


Figure 2: the curve can be evaluated at x and the result can be multiplied by (b - a). This defines a rectangle which can be seen as a very crude approximation of the integral.

evaluate the function at x1 (figure 3) we quite drastically underestimate this area. If we evaluate the function at x2, we overestimate the area. But as we keep evaluating the function at different random points between a and b, adding up the area of the rectangles and averaging the sum, the resulting number gets closer and closer to the actual result of the integral. It's not surprising in a way as the rectangles which are too large compensate for the rectangles which are too small. And in fact, we will soon give the proof that summing them up and averaging their areas actually converges to the integral "area" as the number of samples used in the calculation increases. This idea is illustrated in the following figure. The function was evaluated in four different locations. The result of the function as these four values of x randomly chosen, are then multiplied by (b-a), summed up and averaged (we divide the sum by 4). The result can be considered as an approximation of the actual integral.



© www.scratchapixel.com

Figure 3

Of course, as usual with Monte Carlo methods, this approximation converges to the integral result as the number of rectangles or samples used increases. We can formalize this idea with the following formula:

$$\langle F_N \rangle = (b-a) 1/N \sum_{i=0}^{N-1} f(X_i)$$

Where N here, is the number of samples used in this approximation. In mathematical notation (and statistics), $\langle S \rangle$ represents the average of all the elements in S $\langle F^N \rangle$ is an approximation of F using N samples. This equation is called a basic Monte Carlo estimator. The random point in the interval [a,b] can easily be obtained by multiplying the result of a random generator producing uniformly distributed numbers in the interval [0,1] with (b-a):

$X_i = a + \xi(b-a)$, where ξ is uniformly distributed between zero and one.

The PDF of the resulting X_is is $1/(b-a)$. Since, the random numbers are produced with equiprobability (each number is produced with the same probability than the others), we just divide 1 by the total number of outcomes as in the case of a dice. However in this example, the function is continuous (as opposed to discrete), so we divide 1 by the interval [a,b]. It is important here to note that:

$$\Pr(\lim_{N \to \infty} \langle F^N \rangle = F) = 1.$$

As N approaches infinity, our Monte Carlo approximation converges (in probability) to the right answer (the probability is 1). Note also that $\langle F^N \rangle$ is a random variable, since it's actually made up of a sum of random numbers. We can now proof that the expected value of $\langle F^N \rangle$ is equal to F:

$E[\langle F^N\rangle]= E[(b-a)1/N \sum^{N-1}_{i=0} f(xi)],$
$= (b-a)1/N\ E[f(x)],$
$= (b-a)1/N \sum^{N-1}_{i=0}\int^b_a f(x)pdf(x)dx$
$= \ 1/N \sum^{N-1}_{i=0}\int^b_a f(x)dx,$
$=\int^b_a f(x)dx,$
$=F$
Remember that the pdf is equal to 1/(b-a) thus it cancels out the term (b-a) on the right inside of the integral sign (line 3).

Generalization to Arbitrary PDF
Now, as mentioned above, the formula we used for the Monte Carlo estimator is basic. Why? Because it only works if the PDF of the random variable X is uniform. However, we can extend Monte Carlo integration to random variables with arbitry PDFs. The more generic formula is then:

$$\langle F^N\rangle=1/N \sum_{i=0}^{N-1} f(Xi)/pdf\ (Xi).$$

This is the more generalized form of the Monte Carlo estimator, and the one you should remember.

To be clear, the pdf in the denominator is the same as the pdf of the random variable X.
As with the basic Monte Carlo estimator, to be sure that this formula is valid, we need to check that this estimator has the correct expected value. Let's check:

$$E[\langle F^N\rangle] =E[1/N \sum_{i=0}^{N-1} f(X_i)/pdfX_i)],$$

$$=1/N \sum_{i=0}^{N-1} E[f(X_i)/pdf(X_i)],$$

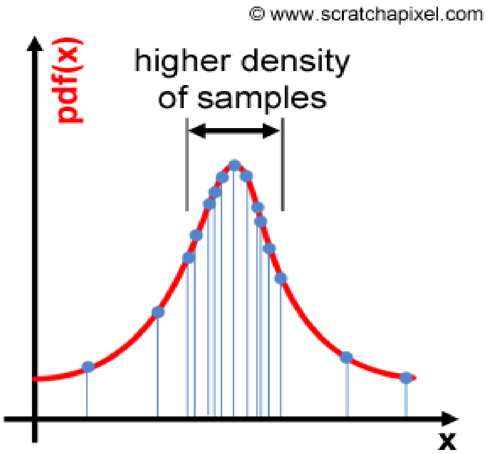
$$=1/N\sum_{i=0}^{N-1}\int_\Omega (f(x)/pdf(x))\ pdf(x)dx,$$

$$=1/N\sum_{i=0}^{N-1}\int_\omega f(x)dx,$$

$$= F$$

This is the most important result of everything we have studied so far, and is the backbone of almost every algorithm. If you don't understand this algorithm, you won't understand monte carlo ray tracing. With the rendering equation this is probably the second most important equation. Now, you will ask why would I ever want to draw samples from any other distribution than a uniform distribution? And that would be a very good question. The weak answer is "because maybe you can only use a given random generator to produce samples and that this generator has a non-uniform PDF". Thus, at least, if that's the case, we just demonstrated that you can still use a Monte Carlo integration, as long as you don't forget to divide f(Xi) by pdf(Xi).

Let's get an intuition as to why dividing f(x) by pdf(x) is necessary (for non-constant PDFs). As you know, the pdf gives the probability that a random variable X get some value xi. When you draw samples from an arbitrary PDF, samples aren't uniformly distributed: more samples are generated where the PDF is high and reversely, fewer samples are generated where the PDF is low (see adjacent figure). In a monte carlo integration though, the samples need to be uniformly distributed. If you generate a high concentration of samples in some region of the function (because the PDF is high in this region), the result of the Monte Carlo integration will be clearly biased. Dividing f(x) by pdf(x) though will counterbalance this effect. Indeed, when the pdf is high (which is also where more samples are generated) dividing f(x) by pdf(x) will decrase the "weight" of these samples in the sum. So in essence, we compensate for the higher concentration of samples, by weighting down their contribution. On the other hand, when the pdf is low, fewer samples are generated (the density of samples is lower than in region where the pdf is high).



But, if we divide f(x) by a low value (1 divided by 0.1 for instance), the contribution of these samples is scaled up. We compensate for the lesser density of samples by giving them more weight. That's in essence, what the division of f(x) by pdf(x) does. It weights the samples' contribution to compensate for the fact that samples generated from an arbitrary PDF won't be uniformly distributed (assuming the PDF is not constant of course): it scales down samples generated in regions of higher density (where the PDF is high), and scales up samples generared in regions of lesser density (where the PDF is low).