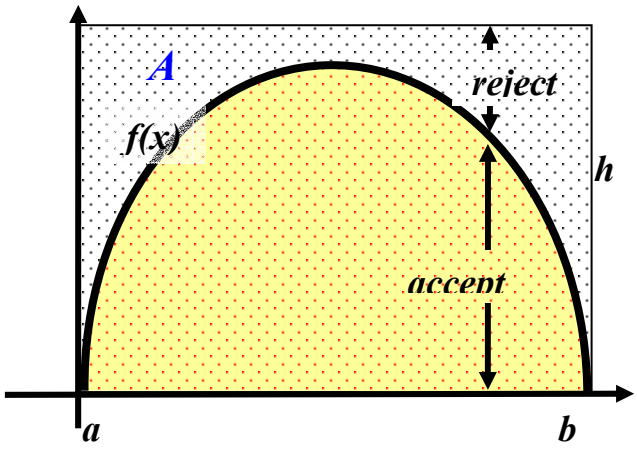
Monte Carlo Random Sampling



The Monte Carlo random sampling method generates what can be called "***artificial data***" of a prescribed form, represented, e.g., by a functional relation $y(x)$ between independent (x) and dependent (y) variables. The simplest Monte Carlo simulation method is called the ***Rejection Method***. It is used in applications with normally distributed data sets in the *MATHCAD* code mathcad\MonteCarlo-01i.mcd. The method is very useful in the numerical evaluation of multi-dimensional integrals of possibly complicated functions of several variables, which may not have analytical primitive functions.

In the following, this method is illustrated with the simple case of a numerical integration of a function ***f(x)*** of one variable x over the interval **[*a, b*]**. As shown in the sketch, one draws a graph of the function ***f(x)*** and encloses that part of the function to be integrated within a (not necessarily, but ***preferably the smallest***) rectangle of area ***A***, in the example $A = h \cdot (b - a)$. Then, a large number of pairs of real numbers $\{x_i, y_i\}$ are chosen randomly within the area ***A***. Each pair $\{x_i, y_i\}$ is tested as to whether or not $y_i \le f(x_i)$, i.e., whether or not the point $\{x_i, y_i\}$ lies within the area ***below*** the curve ***f(x)*** (*see figure above*). If this is the case, the point is accepted for calculating the integral $\int_a^b dx f(x)$, otherwise, the point is rejected. Counting the number N_{acc} of accepted points out of the total number N_{tot} of random points drawn, one has a measure of the

integral, since the area under the curve scales to the total area like the ratio N_{acc}/N_{tot}

$$\int_a^b dx\, f(x) = \frac{N_{acc}}{N_{tot}} \cdot A \qquad (1)$$

This is a plausible example of the general Monte Carlo integration rule

$$\int_a^b dx f(x) \approx \langle f \rangle (b-a) \pm \sigma_f (b-a) \qquad (2)$$

with the statistical error involving the standard deviation

$$\sigma_f \approx \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}} \qquad (3)$$

The averages and errors (see also: Prob-Moments.doc) are defined with respect to the functional values at the x coordinates of the N sampling points $\{x_i, y_i\}$:

$$\langle f \rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad \text{and} \quad \langle f^2 \rangle = \frac{1}{N}\sum_{i=1}^{N} f^2(x_i) \qquad (4)$$

The ± term in Equ. 2 with the standard deviation of Equ. 3 represents only an error *estimate* for the value of the integral obtained with N points. For a more accurate error determination of the Monte Carlo integration method, books on numerical analysis should be consulted.

It is straight-forward to check the consistency of the example with the general method defined in Equ. 2. Realizing that, for the example, $\langle f \rangle = h \cdot (N_{acc} / N_{tot})$, and inserting this into Equ. 2, one obtains the earlier result of Equ.1. At the same time, it is obvious from Equ. 2 that

$$f(x) \approx \langle f(x) \rangle \pm \sigma_f \tag{5}$$

with

$$\langle f(x) \rangle = \frac{N_{acc}(x)}{N_{tot}(x)} \cdot h \tag{6}$$

Here, the sampling is done at a particular abscissa value *x,* i.e., the set $\{(x_i, y_i) | x_i = x\}$ is chosen and the number N_{acc} of acceptable points is counted out of a total of N_{tot} draws. Then, the number of accepted points is proportional to the average of the function *f* at the argument *x,* namely

$$N_{acc}(x) = \langle f(x) \rangle \frac{N_{tot}(x)}{h} \tag{7}$$

It is approximately equal to the actual value of the function *f* at argument *x,*

$$N_{acc}(x) = \langle f(x) \rangle \frac{N_{tot}(x)}{h} \pm \sigma_f \frac{N_{tot}(x)}{h} \tag{8}$$

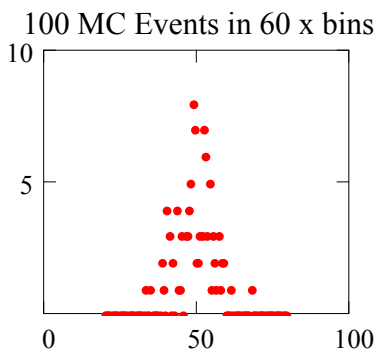
where the second term indicates again the estimated uncertainty.

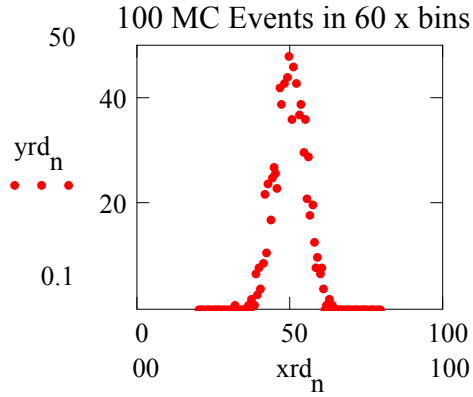
It follows from the above discussion that ***randomly chosen pairs $\{x_i, y_i\}$ with $y_i \le f(x_i)$ have a probability distribution given by the function f(x)***,

$$P(x) = \frac{N_{acc}(x)}{N_{tot}(x)} \propto f(x) \qquad (9)$$

For example, there are many pairs with x-values where the magnitude of f is large, and very few pairs represent the function where it is small. It also does not matter, in which sequence the various arguments x are sampled. So, normally the pairs $\{x_i, y_i\}$ are chosen at random with respect to both x and y. Since $P(x)$ is given by the ***fraction*** of the total number of points drawn, it is already ***properly normalized*** to unity. This is the essence of Monte Carlo random sampling.

The principle is applied to a normal (***Gaussian distributed***) sample of x values with an average value of $\langle x \rangle = x_0 = 50$ and a standard deviation of $\sigma_x = 5$. The two figures above represent the same Gaussian distribution, on the left a very small sample of only 100 events in ± 30 bins about the average, and on the right *1000* events over the same range of x values. In either figure, the frequency

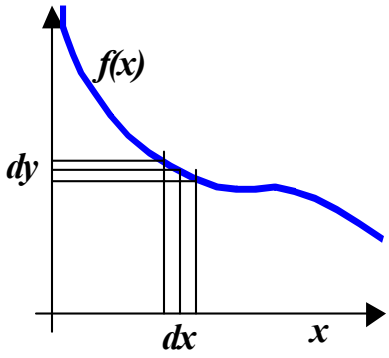




$P(x)$ of finding the value x in the sample of accepted points (cf. Equ. 9) is plotted *vs.* x. It is obvious that a small sample gives only a rough idea of the general shape of the function to be simulated numerically, here a Gaussian, with significant ***statistical errors***. Increasing the sample size by a factor 10, to *1000*, leads to a much

more accurate representation, as illustrated by the figure on the right.

The rejection method discussed above is universally applicable. However, depending on the functional relationship to be simulated, a large fraction of the randomly drawn pairs $\{x_i, y_i\}$ may be rejected and, hence, useless. If this is the case, the method is not very effective and the computing time needed to generate a large statistical sample may become very long. In special cases, the alternative ***Transformation Method*** may be faster. This latter method makes use of the bunching bunching effected by an appropriate reflection of a randomly distributed set $\{x_i\}$ of x values at a function $f(x)$. As illustrated in the sketch, if the x-values are uniformly distributed, the values $y = f(x)$ are compressed, as given by the slope dy/dx. The smaller this slope, the stronger the bunching of the corresponding y-values:



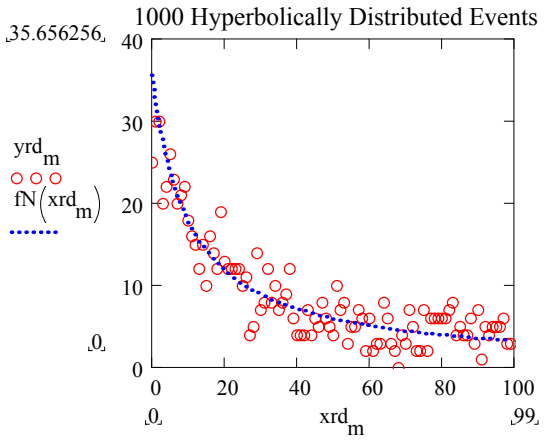
$$dy = f'(x) \cdot dx \qquad (10)$$

where f' is the derivative of f. If the x-values are random, i.e., $P(x) = const.$, then the y-values are distributed according to

$$P(y) = P(x) \cdot \left(\frac{dy}{dx}\right)^{-1} \propto \left(f'(x)\right)^{-1} \qquad (11)$$

In other words, according to Equ. 11, choosing a random sample $\{x_i\}$, the values of the set $\{y_i = f(x_i)\}$ simulate the function $(f'(x))^{-1}$.

Suppose that one wishes to simulate a probability distribution $P(x) = f(x)$ instead of $(f'(x))^{-1}$. If it is possible to obtain a function $g(x)$, such that



$$(g'(x))^{-1} = f(x) \qquad (12)$$

i.e., if it is possible to find the primitive function (integral) of $[f(x)]^{-1}$,

$$g(x) = \int^{x} dx' \frac{1}{f(x')} \qquad (13)$$

then the set $\{y_i = g(x_i)\}$ built upon the random set $\{x_i\}$ has the desired probability distribution $P(x) = f(x)$.

Consider the example of a hyperbolical probability distribution

$$P(x) = f(x) = \frac{a}{b+x} \qquad (14)$$

to be simulated. Obviously, the inverse of this function can easily be integrated, yielding

$$g(x) = \int^{x} dx \left[\frac{b+x}{a}\right] = \frac{b}{a}x + \frac{1}{a}x^2 \qquad (15)$$

Now, choose a random set $\{x_i\}$ which can be done with the code mathcad\MonteCarlo-01i.mcd and construct the set $\{y_i = g(x_i)\}$,

also done in this code for the above example. The resulting frequency distribution $\{y_i\}$ is plotted as open circles $\{\mathrm{yrd_m}\}$ in the figure. The analytical function *f(x)* defined in Equ. 14 is indicated by the dashed curve (*fN*). Obviously, the random points outline the function well, but show a statistical scatter given by the number of events drawn.

Brief discussions of Monte Carlo Methods can be found in

R.H. Landau and M.J. Paez, *Computational Physics*, Wiley Interscience, New York, 1997

N. J. Giordano, *Computational Physics,* Prentice Hall, Upper Saddle River, 1997