

Suppose we want to compute the following multidimensional integral:

$$A = \frac{\int f(\vec{x})w(\vec{x})d^N x}{\int w(\vec{x})d^N x}. \text{ For example, } w(\vec{x}_1, \vec{x}_2) = \exp\left(-\frac{e^2/kT}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}}\right)$$

(which happens to be the weight function you need for the extra credit project).

While we “know” $w(x)$ at any given point x (that is, “locally”), we do not have a good intuition for how $w(x)$ behaves globally. In particular, we don’t even know the normalization of the weight function (that is, the value of A). Thus we cannot use von Neumann rejection or other methods.

To compute A by Monte Carlo, we want to generate a sequence of points $\{x_i\}$ weighted by $w(x_i)$. We do this by the following algorithm:

- (1) We start at some point x_i . We then generate a trial step to $x_{\text{trial}} = x_i + \Delta x$, where Δx is a random step (the size of the step is a parameter to be set).
- (2) Compute the ratio $r = w(x_{\text{trial}}) / w(x_i)$; also, generate d , a uniformly distributed random number between 0 and 1.
- (3) If $r \geq d$, then let $x_{i+1} = x_{\text{trial}}$, (accept) else $x_{i+1} = x_i$ (reject).
- (4) Go to step 1.

Discussion. We can show that this procedure satisfies the *detailed balance equation*, that is,

$$w(x) P(x \rightarrow x') = w(x') P(x' \rightarrow x), \text{ where } P(x \rightarrow x') \text{ is the probability to go from } x \text{ to } x'.$$

Suppose $w(x') > w(x)$. Then, from the above, $r > 1$, and the probability $P(x \rightarrow x') = 1$.

Conversely, from the reverse process, probability $P(x' \rightarrow x) = r = w(x)/w(x') < 1$.

Putting together.

$$w(x) P(x \rightarrow x') = w(x) \times 1 = w(x)/w(x') \times w(x') = r w(x') = P(x' \rightarrow x) w(x').$$

One way to think about this: if $w(x') > w(x)$, then one is guaranteed to accept the step. On the other hand, if $w(x')$ is much smaller than $w(x)$, there is only a low probability to accept the step. This means that the Metropolis tends to produce a lot of points in areas of high weight and few points in areas of low weight—without having to know the absolute normalization of the weight function.

Sweeping. Experience shows that for high-dimensional problems, making a step in all dimensions simultaneously tends to lead to very high or very low acceptance rates. Thus, one often sweeps, which means, making a step in only one coordinate, accepting or rejecting that step, and then going on to the next coordinate.

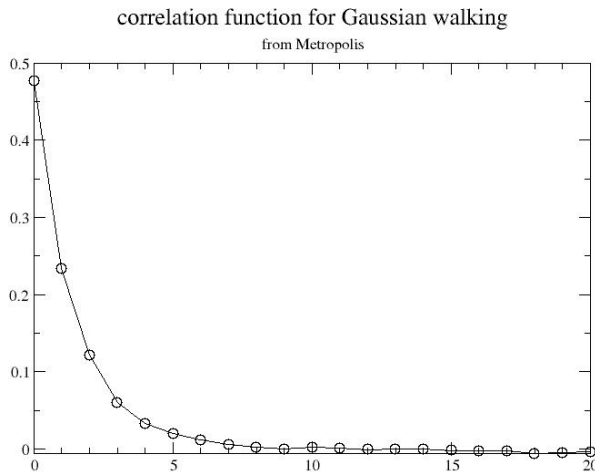
Correlation: Because one only takes small steps, x_i and x_{i+1} are correlated. In order to have statistically meaningful results, one needs to only use uncorrelated samples, say x_i and x_{i+n} . The question is, what value of n is needed to decorrelate?

To answer this, one must do a run to compute the *correlation function*:

$$C_n = \frac{\langle f_i f_{i+n} \rangle}{\langle f_i \rangle^2} - 1 = \frac{\frac{1}{N} \sum_i f_i f_{i+n}}{\left(\frac{1}{N} \sum_i f_i \right)^2} - 1$$

From this one obtains a plot of C_n which

decreases in n ; when it approaches zero, that is the appropriate value of n . There is a code, *correlation.f* which can be used to compute the correlation function.



In the above picture, a good value of n to decorrelate would be between 5 and 8. This means, when actually doing a production run, one would use only every 5th sample in the integral.

Thermalization. Similar to correlation, when one chooses an initial position it is seldom truly random. Therefore one must let Metropolis run for a while in order to let the system "thermalize." Typically this must be several times the correlation length.