10.34 Fall 2015
Metropolis Monte Carlo Algorithm

The Metropolis Monte Carlo method is very useful for calculating many-dimensional integration. For e.g. in statistical mechanics in order to calculate the prosperities of the system you are required to use ensemble average. The ensemble average of any property $\underline{B}$ is given by:

$$\langle B \rangle = \frac{\int \cdots \int B\left(\underline{r}^N, \underline{p}^N\right) w_N\left(\underline{r}^N, \underline{p}^N\right) d\underline{r}^N d\underline{p}^N}{\int \cdots \int w_N\left(\underline{r}^N, \underline{p}^N\right) d\underline{r}^N d\underline{p}^N}$$

where w_N is the statistical weighting function, r^N is configuration space (a 3N dimensional vector of spatial coordinates), p^N is momentum space (a 3N dimensional vector of momentum). Both the numerators as well as the denominator have 6N integrals to be computed. We can use Monte Carlo method to evaluate these integrals. This consists of simply summing over random points sampled according to the probability distribution. Rewriting the above expression in terms of the probability distribution $P(\underline{r}^N, \underline{p}^N)$:

$$\langle B \rangle = \int \cdots \int B\left(\underline{r}^N, \underline{p}^N\right) P\left(\underline{r}^N, \underline{p}^N\right) d\underline{r}^N d\underline{p}^N$$

where P is defined as:

$$P\left(\underline{r}^N, \underline{p}^N\right) = \frac{w_N\left(\underline{r}^N, \underline{p}^N\right)}{\int \cdots \int w_N\left(\underline{r}^N, \underline{p}^N\right) d\underline{r}^N d\underline{p}^N}$$

If we had a simple explicit form for P we could try to sample directly from that distribution, and then evaluate B at these points. However, since the denominator of the expression for P is so difficult to evaluate, instead we would like to use a method that works directly with w. Now let us see how to implement the Metropolis Monte Carlo Method to solve the integral using the weighting function. Following is the algorithm:

<u>Metropolis Monte Carlo Method:</u>

We have to generate a random sequence $\underline{q}^{[1]}$, $\underline{q}^{[2]}$, $\underline{q}^{[3]}$ …of states to solve for the integral. We start with some value for $\underline{q}$ and then make moves to different states. For the kth iteration (move), you are at a state $\underline{q}^{[k]}$ and have a scalar "Sum" defined, where this scalar holds the value of the integral we are attempting to solve:

1) Randomly generate a step $\Delta\underline{q}$ (e.g. using Ndim random numbers from rand function, $\Delta\underline{q}$(n) =Δ*(2*rand-1), for n=1..Ndim, where Δ is the maximum allowable displacement in any of the coordinates in successive iterations)

```
2) Propose a new state $\underline{q}^{[proposed]} = \underline{q}^{[k]} + \underline{\Delta q}$

3) Compute $w(\underline{q}^{[proposed]})$ and $w(\underline{q}^{[k]})$,
   If $w(\underline{q}^{[proposed]}) > w(\underline{q}^{[k]})$ then
           $\underline{q}^{[k+1]} = \underline{q}^{[proposed]}$
   else if $w(\underline{q}^{[proposed]}) / w(\underline{q}^{[k]}) >$ rand
           $\underline{q}^{[k+1]} = \underline{q}^{[proposed]}$
   else
           $\underline{q}^{[k+1]} = \underline{q}^{[k]}$

4) Sum = Sum + $B(\underline{q}^{[k+1]})$

5) $\langle B \rangle$ = Sum/(No. of random points)
```

Note: "rand" is a uniformly-distributed random number from 0 (zero) to 1 (one).

The goal of the Metropolis MC method is to generate N states of $\underline{q}$ such that:

$$\lim_{N\to\infty} \frac{N_{q_i}}{N} = w(q_i) \text{ for all } i$$

where the variable N_{qi} represents the number of molecules in state q_i and the weighting function w is known. Two questions that may be on your mind are:

- How do we know that the Metropolis MC method actually achieves this goal?
- Why do we use the particular acceptance criteria: $w(\underline{q}^{[proposed]}) / w(\underline{q}^{[k]}) >$ rand?

Discussion:

Suppose we have a state in our system, q_i. If we wanted to calculate the total number of q_i states in our system, N_{qi}, we would need to worry about two terms:

(1) The state q_i moved from another state in our system $q_{j\neq i}$:

$$\sum_{j\neq i} N_{qj} P(q_i \mid q_j)$$

(2) Our system tried to move out of state q_i but remained there:

$$N_{qi}\left[1 - \sum_{j\neq i} P(q_j \mid q_i)\right]$$

The $P(q_i|q_j)$ expressions are conditional probabilities, which represent the probability of moving to state q_i given that we were in state q_j. Summing these two expressions gives us the total number of states in our system in state q_i:

$$N_{qi} + \Delta N_{qi}$$

where ΔN_{qi} is defined as:

$$\Delta N_{qi} = \sum_{j \neq i} N_{qj} P(q_i \mid q_j) - N_{qi} \sum_{j \neq i} P(q_j \mid q_i)$$

$$= \sum_{j \neq i} N_{qj} P(q_i \mid q_j) - N_{qi} P(q_j \mid q_i)$$

We know the values of the conditional probabilities from our acceptance criteria:

$$P(q_i \mid q_j) = \begin{cases} w(q_i)/w(q_j) & \textit{if } w(q_i) \leq w(q_j) \\ 1 & \textit{otherwise} \end{cases}$$

Looking at one of the other states in our system, q_k:

$$\Delta N_{qi} = N_{qk} P(q_i \mid q_k) - N_{qi} P(q_k \mid q_i)$$

If $w(q_i) < w(q_k)$:

$$\Delta N_{qi} = N_{qk} \frac{w(q_i)}{w(q_k)} - N_{qi}$$

If $w(q_k) < w(q_i)$:

$$\Delta N_{qi} = N_{qk} - N_{qi} \frac{w(q_k)}{w(q_i)}$$

When $\Delta N_{qi} = 0$, both of the preceding equations tell us:

$$\frac{N_{qk}}{w(q_k)} = \frac{N_{qi}}{w(q_i)} = \text{constant}$$

If we choose this constant to be N, the total number of molecules, then we've reached the "goal" of the Metropolis MC method (as stated approximately half-way down the second page):

$$\frac{N_{qi}}{N} = w(q_i)$$

Notice, there is nothing special about states *i*, *j*, or *k* in this derivation. Thus, the preceding equation is true for all states *i* in our system. Another way to think about the above expression is that there will be no change in the system ($\Delta N_{qi} = 0$) once the relative populations of all states (N_{qi} / N) reach their expected probability (w_{qi}) … a MC way of saying "system has reached equilibrium."

Returning to our expressions for ΔN_{qi}:

$$\Delta N_{qi} = N_{qk} \frac{w(q_i)}{w(q_k)} - N_{qi}$$

$$\Delta N_{qi} = N_{qk} - N_{qi} \frac{w(q_k)}{w(q_i)}$$

Imagine $\Delta N_{qi} > 0$:

$$N_{qk} \frac{w(q_i)}{w(q_k)} - N_{qi} > 0$$

$$N_{qk} \frac{w(q_i)}{w(q_k)} > N_{qi}$$

$$\frac{w(q_i)}{w(q_k)} > \frac{N_{qi}}{N_{qk}}$$

This expression tells us there are more molecules in state q_k (and less molecules in state q_i) than what we would expect based on the ratio of their probabilities. Thus, ΔN_{qi} is increasing to counter that effect. Similarly, imagine $\Delta N_{qi} < 0$:

$$N_{qk} \frac{w(q_i)}{w(q_k)} - N_{qi} < 0$$

$$N_{qk} \frac{w(q_i)}{w(q_k)} < N_{qi}$$

$$\frac{w(q_i)}{w(q_k)} < \frac{N_{qi}}{N_{qk}}$$

In this case, there are more molecules in the state q_i (and less in state q_k) than we would expect based on their probabilities. However, ΔN_{qi} is decreasing to counter this. Thus, the system is always trying to reach equilibrium.

Simple Example with Metropolis MC Method:

Suppose we want to solve following integral using Metropolis Monte Carlo Method:

$$\int_0^1 x^2(1.5-x)dx$$

Here, we have $f(x) = x^2$ and weighting function $w(x) = 1.5 - x$. We have to generate random sequence of x values in (0,1) and accept them on the basis of weighting function. Let us start with $x_0 = 0.25$ (arbitrarily chosen). Generate N random numbers between 0 and 1. For each number x_new, compute (w(x_new)). If (w(x_new)) > =(w(x(i-1))), then x(i) = x_new, else (w(x_new)/ w(x(i-1)))>rand, then x(i) = x_new, else x(i) = x(i-1). Sum (f(x(i))) for all i and divide by N to get the value of the integral.