Introduction to Polymer Physics
Prof. Dr. Prateek Kumar Jha
Department for Chemical Engineering
Indian Institute of Technology-Roorkee

Lecture-24
Monte Carlo Algorithm: Detailed Balance, Metropolis Algorithm

Welcome so in the last class we have ended up discussing about the Monte Carlo simulations and we give an example of these bead spring model and I already showed you the basic algorithm of a Monte Carlo simulation. So today I will take it further and first tell you like how do we get the acceptance and rejection probability and then I will tell you some practical tips for a Monte Carlo simulation.

So just to recall the steps of the Monte Carlo simulation, so we start from a configuration which represent

$\{\vec{r}_i\}$

It indicate in the positions of bead then I do trial displacement for a bead chosen randomly. So the key here is that is we should always choose with uniform probability that is like all the beads must be chosen with equal probability if for example we are choosing if we are doing a large number of Monte Carlo simulation. It should not be that a particular bead is chosen with lesser probability compared to others of course that will be true only when we do a large number of Monte Carlo steps and then we compute energy change due to trial that is –

$$\Delta U = U\left(\{\vec{r}_i'\}\right) - U\left(\{\vec{r}_i\}\right)$$

So keep in mind all the energy that we have discussed like the elastic energy and the excluded volume interactions and Lennard- Jones interactions. They all depend on distances as soon as any bead is moved all these interactions are changed and that is why we are interested in the energy change from the move and then accept or reject the move using some function of ΔU if I accept basically I set r_i to r_i' and if I reject of course I do nothing. Then I repeat 2-4 until some sort of convergence that I will elaborate in this lecture. So the first thing here is to determine

what the condition of acceptance or rejection should be and that is where we will use a condition that is known as detailed balance that is very essential to the Monte Carlo simulation.
The detailed balance essentially requires that when the system reaches equilibrium of course when we start with random configuration we do not hope that we are at equilibrium but if I keep on running it we do hope the system will reach equilibrium. When the system is at equilibrium the number of forward transitions should be equal to number of backward transition if something is going in that direction with some probability the reverse should also happen with the same probability or the number of transitions should be the same in order to at equilibrium. So in most Monte Carlo algorithm with demand the detailed balance to be true, detailed balance by itself does not give me the function P, but it tells me which functions will be workable functions which functions I can use that satisfies detailed balance criteria.

So the basic condition is the number of forward transitions equal to the number of backward transitions. So let us say that we talk about two particular configurations- one of them I call 'o' and one of them I call 'n', the old and new configuration. Then the system at equilibrium will have certain probability to be in configuration 'o' and certain probability to be in configuration 'n' and we do not know essentially what the probabilities is so let us say N(o) means the probability for configuration 'o' to occur and N(n) be the probability for n to occur at equilibrium this is very important. So we don't start from equilibrium but we want the detailed balance to be valid from beginning of the configuration.

So now of course when I am doing this trial displacement there must be a probability of going from old confirmation to the new configuration. So let us say Π (o to n) represents transition probability from o to n and similarly the reverse can also happen (n to o) and we have transition probability form n to o.

So now what the detailed balance demand is the number of forward transitions and the number of backward transitions. The number of forward transition the forward here is old to new and backward is new to old. So the number of forward transitions depend on the number of states that

correspond to the old state multiplied by the probability of going from o to n or must be proportional to the probability of being in the old state multiplied by the probability of going from old to new. So the forward transitions will N(o) multiplied by 𝛑(o-n) so only those configuration which are already at o can go to n with the probability 𝛑(o-n) and then similarly only those which are at n can go to o with probability 𝛑(n-o). So here-

$$N(o)\pi(o \to n) = N(n)\pi(n \to o)$$

This is the condition of detailed balance and now we will see how does it help to give the acceptance or rejection criteria. So let us look at what does the transition probability is equal to, so the probability of going from o to n depends on our algorithm, depends on how many trials we make that are going from o-n and how many of those trials I accept. So let us say if α(o to n) represents the number of trials from o-n or more specifically the probability of trials from o to n. You can make of course many trials only the probability of trials that goes from o to n should be considered multiplied by what is the probability of accepting those trials. So while α depends on how I am doing the trial, acceptance will depend on the energy condition that I employ to reject or accept the trial. So now-

$$\text{equation 1 for trials } o$$
$$\pi(o \to n) = \alpha(o \to n)\, acc(o \to n) \quad (¿n)$$

$$\text{equation 2 for trials } n$$
$$\pi(n \to o) = \alpha(n \to o)\, acc(n \to o) \quad (¿o)$$

Using both equation 1 & 2 we write the detailed balance as-

$$\frac{N(n)}{N(0)} = \frac{\pi(o \to n)}{\pi(n \to o)} = \frac{\alpha(o \to n)\, acc(o \to n)}{\alpha(n \to o)\, acc(n \to o)}$$

This sets the rule for the Monte Carlo algorithm, we can have some liberty in how I am choosing a trial move, and we can have some liberty in how are we accepting or rejecting the move. But in the end whatever we do it must satisfy this particular condition that comes from the detailed balance.

The most common algorithm is the Metropolis algorithm that we will discuss next. So in the metropolis algorithm what we do assume is I will make trials randomly in all possible directions for the purpose of making trials I will not differentiate between any configurations. For example I have 10 configuration possible then I can make trials both forward and backward with equal probability only the acceptance criteria can differ. So in this case I will assume α to be the same for both forward and backward transition and now I am left with-

$$\alpha(o \to n) = \alpha(n \to o)$$

$$\frac{N(n)}{N(o)} = \frac{acc(o \to n)}{acc(n \to o)}$$

So now finally we already have discussed something about the Boltzmann distribution the probability of the system being in the particular configuration depends on what is known as the Boltzmann factor actually proportional to it. So in fact if I look at the ratio of probability of the new and old configuration they will be essentially a ratio of their Boltzmann weights. That is for the new one divided by the same thing for the old one.

$$\frac{\exp(-\beta U(n))}{\exp(-\beta U(o))} = \frac{N(n)}{N(o)} = \frac{acc(o \to n)}{acc(n \to o)} = \exp(-\beta \Delta U)$$

Where, $\Delta U = U(n) - U(o)$ and $\beta = \frac{1}{k_B T}$. Here T is the absolute temperature and k_B is the Boltzmann constant and the whole term is Boltzmann Factor. The other way again to remind you this is only happen at equilibrium although we start making that particular criteria in the beginning detailed balance is only satisfied at equilibrium that is a demand that we are making from the simulation.

So with this now we have a rule to perform the metropolis algorithm Monte Carlo-

$$\frac{acc(o \to n)}{acc(n \to o)} = \exp(-\beta \Delta U)$$

Here again I can have some flexibility in terms of how exactly I am going to choose acceptance probabilities them self but this particular ratio must satisfy this particular relation that we have got from detailed balance.

So in the metropolis case we assume-

$$acc(o \rightarrow n) = \exp(-\beta \Delta U)$$

so when delta U is higher than 0 that means we are going to a higher energy state I will use exponential of minus beta delta U and I will use one delta u is less than 0, if I am going downhill in the energy we always accept the move if we are going uphill we accept the probability that is given by $\exp(-\beta \Delta U)$. Now to satisfy the detailed balance that we have here the reverse probability should simply be $\exp(+\beta \Delta U)$. Now you can see if I take a ratio for delta u greater than 0 case I will get exponential of minus beta delta u. If I take the ratio for delta u less than 0 case again satisfy the detailed balance condition. So, this becomes my metropolis algorithm, my acceptance criteria is-

$$P(\Delta U) = Min\left[1, \exp\left(\frac{-\Delta U}{k_B T}\right)\right]$$

Here $\Delta U = U(\{\vec{r}_i'\}) - U(\{\vec{r}_i\})$. Alright, so now if I go back to my Monte Carlo algorithm we have figured out how to accept or reject the move. So I have looked that the probability of acceptance rejection will be just my 1 - of that and now we are pretty much all set to do the simulations. Now there are few practical tips here that we will now discuss.

The first tip is like how exactly we are going to choose the trial displacement particularly how exactly do we program that to question. So let us first start with that, so what should be the trial so first condition is that I will randomly choose a bead that we can do by any kind of a random number generator but the second thing is like what should be how exactly we update the coordinates. So we want to do for the chosen bead-

$$\vec{r}_i \rightarrow \vec{r}_i'$$

$$x_i \to x_i^{'}, y_i \to y_i^{'}, z_i \to z_i^{'}(¿ cartesian\, coordinates)$$

Now there are many ways to do it, the trick here is we must choose displacements that goes in all possible directions because ultimately you start to random configuration you don't know what the ultimate equilibrium is right. So it is always better that we do not bias our random displacement our displacement should be able to explore the entire space. So to do that the way we will do it is we start from the old positions set some maximum displacement and generate numbers-

$$x_i \to x_i^{'} = x_i + \Delta_1 (r_1 - 0.5)$$

$$y_i \to y_i^{'} = y_i + \Delta_2 (r_2 - 0.5)$$

$$z_i \to z_i^{'} = z_i + \Delta_3 (r_3 - 0.5)$$

Here r_1, r_2, and r_3 are random numbers between 0 and 1. All these quantities will vary from -0.5 to 0.5 and we are not biasing the displacement into any particular direction and it helps us to reach equilibrium faster.

The second one is Δ_1 , Δ_2 Δ_3 they essentially set the maximum displacements. So if the system is anyway isotropic we do not we do not really bias into any particular direction. So we set-

$$\Delta_1 = \Delta_2 = \Delta_3 = 2 \times the\, maximum\, displacement\, which\, goes - 0.5\Delta ¿ + 0.5\Delta$$

So now you think of like what exactly will happen in the simulation if I for example set delta to be very high so go back to the bead spring model and let us say if I stretch a bead if I pull a bead to very large distance that we will lead to large elongation in the spring that of course will mean large energy change and if it is a large energy change in the case of spring it is large positive energy change, that large positive energy change if I put into our acceptance condition will give rise to very little acceptance probability right, so large value of delta will be giving rise to little

acceptance probability because every move is resulting in a large stretching of spring or for example separation of beads to very large distances in those moves are typically not accepted because of the metropolis condition of acceptance that is one problem.

If on the other hand if I start with a very small delta now you will see what will happen if I just pull up bead slightly then the spring elongation will change only slightly that would mean that the energy change will be like very small. If energy change is very small it is more likely the move to get accepted. So every trial move you will do will be more likely to be accepted ok that is also not quite preferred because it will take you like a very long simulation to actually get equilibrium state if you are very far from it because although you are accepting all the moves in every move your only moving by little say not going towards the equilibrium value fast enough. So in this case we have large acceptance probability and it turns out both of them are not efficient simply because in the first case I will not accept most trials. In the second case I will accept every trial pretty much and in both the cases I am going very slowly towards equilibrium.

So as a rule of thumb delta should be such that acceptance probability or rate goes in the range 40% to 60% just a rough number in every Monte Carlo algorithm is not follow this. But if you are doing a Monte Carlo case and if it is taking very longer you look at how many moves are getting accepted and may be happens rate is very small or very high that is one thing. The second thing is always a good practice for any Monte Carlo code you vary the delta value to figure out what is acceptance rates of moves that is and you find out delta value where it is between 40% to 60% that is when the program will be most efficient this is like I would say really a rule of thumb.

So this is like a very important practical tip when we are doing a Monte Carlo simulation. In the next lecture we will discuss couple more tactical tips that we must consider if I am doing a Monte Carlo simulation.

Thank you.