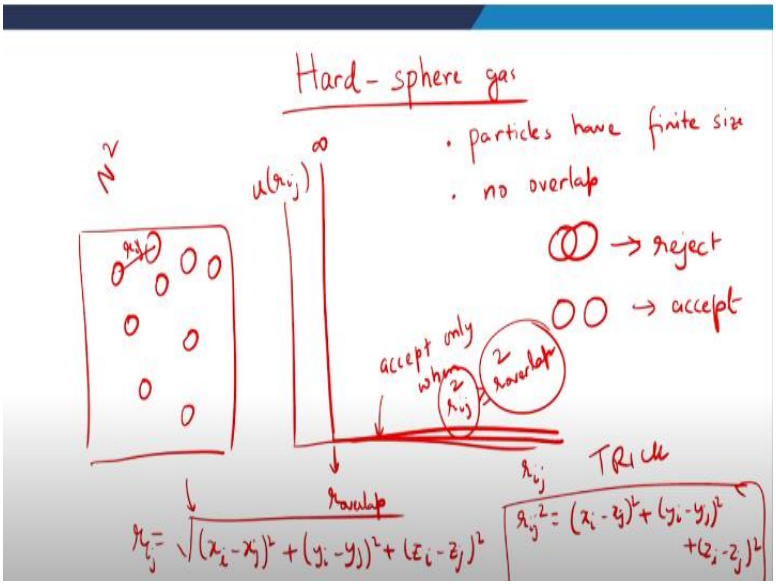
Advanced Thermodynamics and Molecular Simulations
Prof. Prateek Kumar Jha
Department of Chemical Engineering
Indian Institute of Technology, Roorkee

Lecture - 40
Numerical Implementation of Monte Carlo Simulation: Python Examples V

Hello all of you. So, in the last lecture we started discussing the Monte Carlo simulation of multi particles systems. We have already done the ideal gas example and I showed you how to visualize and make a movie of Monte Carlos simulations. We were already in the middle of the molecular simulation. We discussed the beauty of it already because once you have very nicely playing movies, they pretty much tell you how the system is evolving that is much more revealing than compared to the just the map of it that we discussed earlier.

So, today, I will take the discussion a bit more realistic we discuss the hard sphere and soft sphere simulations where in addition to the movements there is also some energetic interaction between the particles.

(Refer Slide Time: 01:17)



So, let us start with first the hard sphere case. So, there are basically the hard sphere case. There is only one change from the ideal gas model and that is particles have a finite size and they cannot overlap. So, apart from that as long as the particles are not beginning to overlap, there is no such problem, there is no interaction but if there is an overlap happening then we should

reject that move because in Monte Carlo we have an option of accepting and rejecting a move. So, you may imagine why I am doing in molecular dynamics, why it is a problem of molecular dynamics because in molecular dynamics we cannot reject any move. So, we have to make movements that prevent overlapping that is more difficult than making random movements and rejecting all the moves that give rise to overlap. Therefore, we can simulate hard sphere in Monte Carlo but not in molecular dynamics.

So, the other problem is that since the energy function is discontinuous the forces are not defined, that is a derivative of the energy. So, in Monte Carlo, it is not a problem. So, the way we should do it is we should follow pretty much the ideal gas model approach. But whenever an overlap is happening we should reject and whenever an overlap is not happening we should accept, because there is no energy apart from the hard sphere energy.

So, with this particular idea if I want to simulate one of the things that I have to compute now is the distance between the particles. Because the hard sphere interaction is like this. So, my u r_{ij} versus r_{ij} will be something like that. That is it is going to be infinity. When we are less than the overlap distance it is going to be zero beyond that. In other words, we need to accept only when r_{ij} greater than equal to r overlap.

So, therefore we have to compute the distances for every pair of particles and figure out if any of them are less than the overlap distance. If it is less we reject it, if it is higher we have no other problem. So, in any particular frame or at any particular Monte Carlo step if there is even a single overlap we should not accept that particular movement. So, now it turns out that I can define clearly the r as something like-

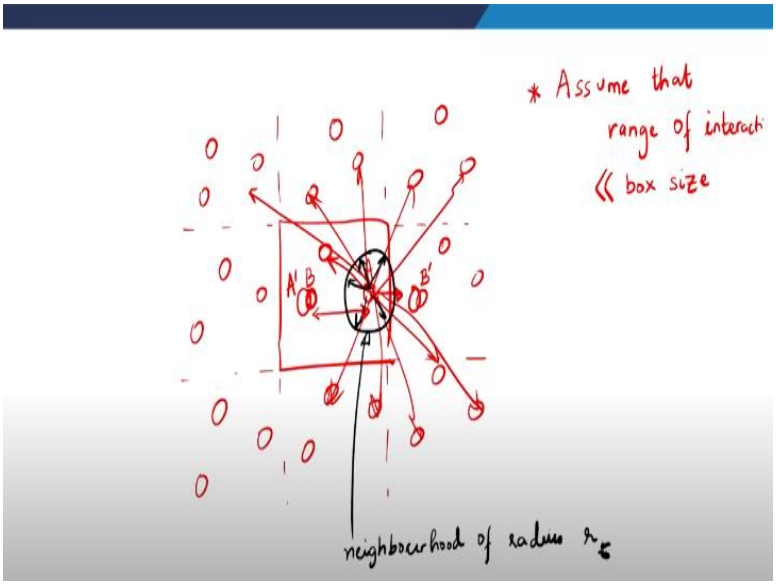
$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

So, this involves a square root operation and I was telling you in some other lecture that square root is difficult for the computers, it takes more time. So, we should as a rule avoid as many square roots as we can because those will improve the efficiency of the code because you may imagine since we have some n square pairs, we are going to have n square square root operation after every Monte Carlo step. So, in order to avoid that we have to remove that particular operation. It turns out it is very easy in this case. Instead of saying this I can say r_{ij}^2 is greater than or equal to r overlap square and this quantity is anyway constant.

So, therefore, you simply can work with the squares of the distances as it turns out that even in more complicated cases it is often possible to work with only the squares of distances as opposed to the actual value. So, I will simply evaluate the square of that is I would say one of the tricks that we do in Monte Carlo simulation.

$$r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2$$

(Refer Slide Time: 06:40)



Now there is one more point that we have to be bothered about and that is the interaction between the particle and its images in the system because what I was telling you is that even if we are simulating only that central simulation box I assume that the same configuration is being repeated in all the image boxes that we have. Now you may well say that the images essentially was the same particle it was moving. So, why exactly we are bothered about images apart from the fact that one particle will leave then the other will enter only in that case the image would be important in other cases it is not important and the answer is, in fact, no because let us say for example if this particle wants to cross the box this particle is actually pretty far from it.

So, you may imagine that this particle has no influence on small movement of particles. So, let us say if this is A and this is B so when A crosses the box B is on the other side. So, what role does B has to play in that? But A can very well cross it and overlap with B prime that is the image of that should be allowed and the answer is no because we have the whole idea of images is that we should allow free movement through every simulation box. So, B prime is actually the same particle as B. So, if I overlap with B prime then this basically corresponds to an

overlap of an A prime with B because as A comes out another A comes from the opposite end.

So, therefore what I was trying to convince you is it is not that we have to consider the interaction between the particles only in the box but in principle we should also consider the interactions with all possible images everywhere. And now suddenly the problem becomes completely intractable because just imagine you already had n square pairs of particles with particles in the simulation box and now if I also account for the particle image interaction, we are pretty much going to like infinitely many possible pairs to worry about if indeed the particle image interactions are important. So, two things help us here.

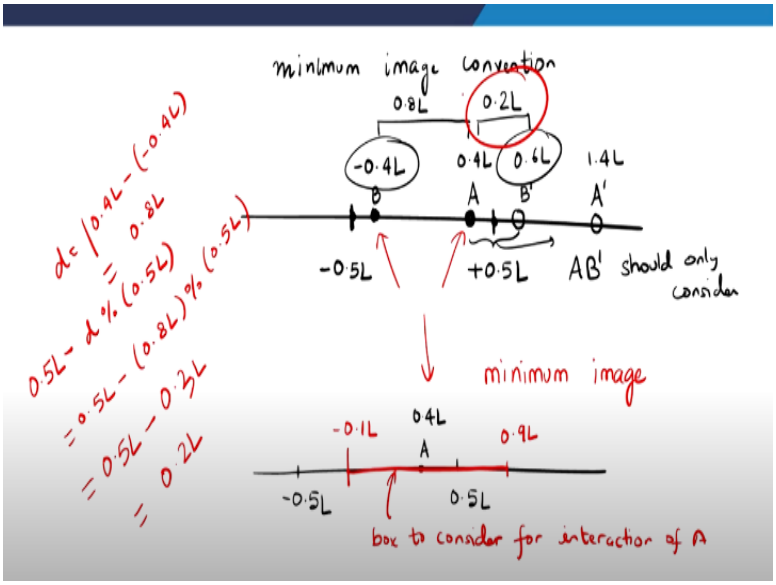
The first thing is we assume that range of interaction is much less than the box size. So, beyond that small range where the interactions work, the interactions have no role to play. So, therefore if that range is much higher than box size then in that case we should only look in the neighborhood of A. So, I should not go too far from A here. I should look at only the small neighborhood of A and compute interactions in that regime. So, let us say for example that is defined as a neighborhood of some radius r_i or r_c.

So, what is that neighborhood in the case of a hard sphere interaction? It is not very difficult to see. So, beyond that r overlap distance there is pretty much no problem. So, if the particles are at a distance more than r overlap why should we care? So, we should only look at the distances which are less than the overlap distance only then we see an interaction in those cases we are rejecting the move.

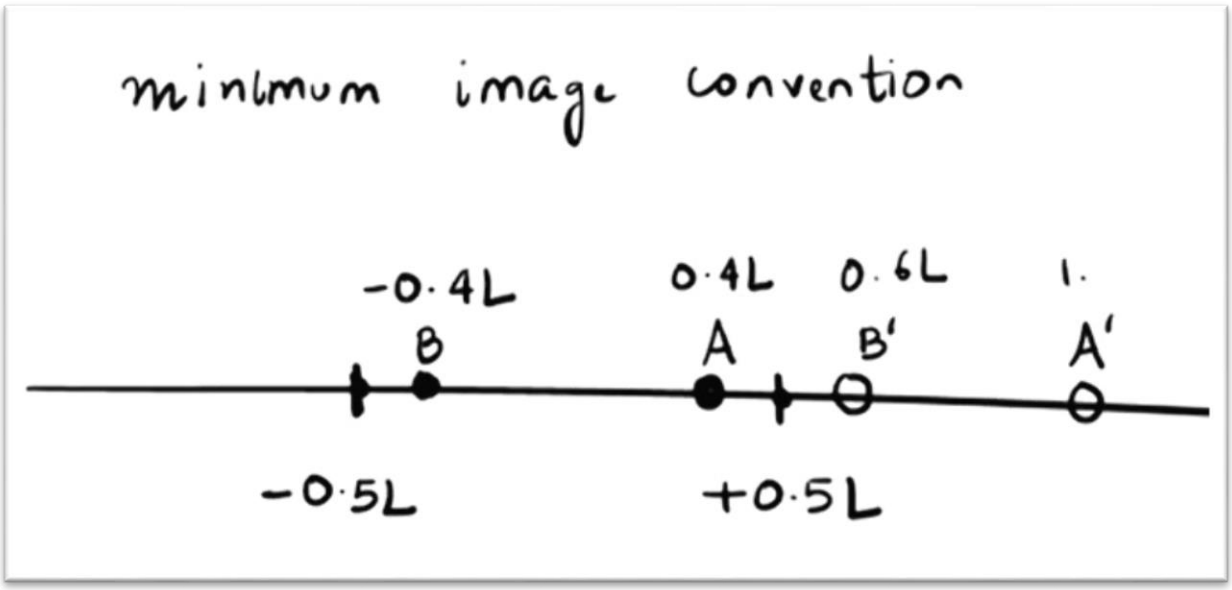
In general I was telling you that interactions can be of various shapes and then the interactions may not decay to zero even at long distances and in those cases we need to make an assumption and I will come to that in the next example of a soft sphere gas but for the hard case it is no big deal.

So, therefore we can only consider in the neighborhood. So, that solves the problem of particles with all the images because many of the images are going to be beyond the interaction range and therefore essentially we will be looking at only the nearest image of the particle. So, the next thing then is then how to account for the nearest image of a particle? And I will show you an example in one dimension. The same idea we can employ in the three dimensional case.

(Refer Slide Time: 12:25)



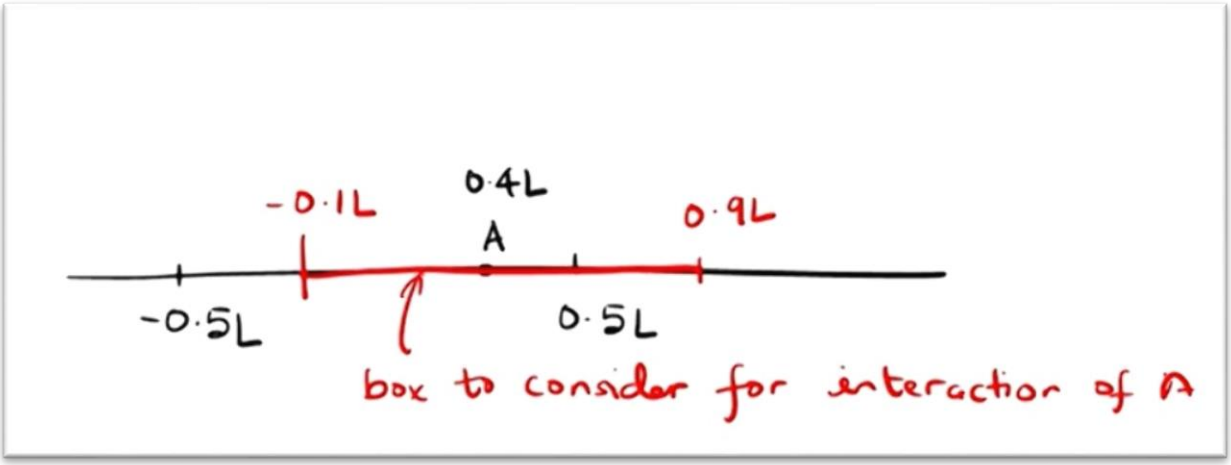
And the idea is called the minimum image convention. So, let us say that this is my original box. And I am looking at two particles located in different locations. That is my A and B. And A is located at 0.4 L and B is located at minus 0.4 L. So, now clearly A and B are going to have their images in the adjacent box. So, you will have another B prime that is L distance to the right of B that has to be minus 0.4 L + L, that is 0.6 L and then we will have an A prime that is 0.4 L + 1 that is at 1.4 L.



So, now if I look at the A particle near to it there are two options for B. One is B that is located in the original box and the image of B that is located on the adjacent box. So, if I look at the distance between A and B prime here, that is actually 0.2 L. On the other hand if I look at the distance between A and B, that is actually 0.8 L and therefore I should only consider AB prime

because we are interested in the pairs that are nearest to each other because pairs which are farther are going to have no interaction because I have assumed that the range of interaction should be much smaller than the box size. So, only the nearest combination of the particle-particle or particle-image is going to matter and any further than that is going to be not important.

So, a better way to do that is I can say that I will look at a box that is centered around A. So, A is at 0.4 L. So, instead of looking in the original box, I can look in a box centered at A of same length as the box. So, I define a new box here that is centered at A. So, it is - 0.5 L from 0.4 L. That is - 0.1 L to 0.5 L to the right that is 0.9 L. So, this becomes the box to consider for interactions of A.



So, for every particle that is located in the original box - 0.5 to 0.5 L, we are going to have an image in the new box. So, therefore instead of like looking in the original box, I can look in the new box for a particle and this I can do for every particle in the system. And the reason why we can do that another way to think about it is that our choice of origin as the center of the simulation box is arbitrary even if I shift that origin to somewhere else then also I will have the same system because the system repeats itself. So, you have the same cube stacked over each other. So, even if I make the center of the cube as the center of the box center of the phase as the center of the box, it does not really matter because the same thing will repeat itself. As long as the box size is L any box is a valid box because it repeats itself.

If there was some other boundary condition like fixed boundary condition it is no longer true that really comes as a virtue of the periodic boundary condition because anything that leaves

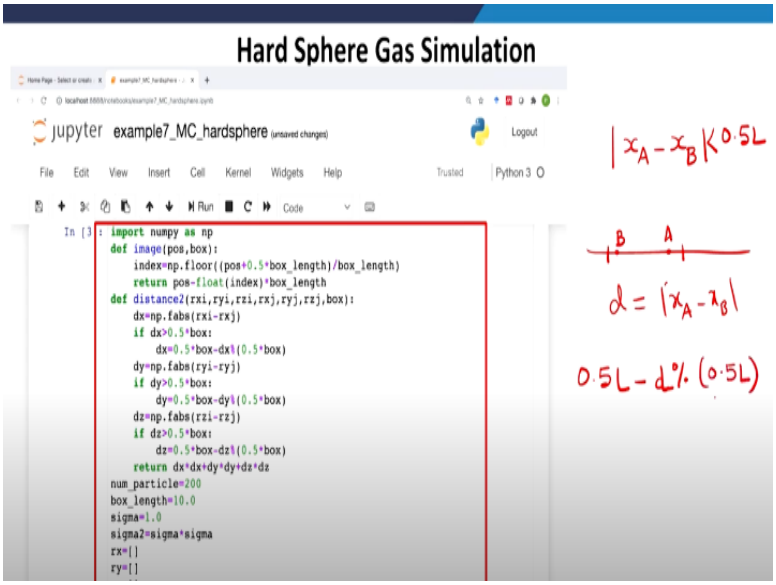
the image of that comes from the other end. So, therefore the location of the center of the box is completely arbitrary.

Now, you may imagine that this is going to be complicated because for every particle in the system and there are n particles we have to think of a different box. And that is clearly not the way we should do it because then you have n possible boxes and I have to worry about the coordinates of the n possible boxes. It turns out that all that effort is not really needed you can just imagine that this idea is what we are using but for the computation of distance, we can simply do a small arithmetic manipulation with the original particle positions in the original box using that I can find the distance for the case of the minimum image and the way that works and I am using in the code here is that I am saying that I will compute something like absolute value of xA – xB

So, there are A and B located in the original box. I compute the absolute distance. If it happens to be less than 0.5 L then we can clearly see that it is going to be present in the new box centered around A because around the box centered at A with length of L, I am starting from 0.5 L to the left of A and going until 0.5 L to the right of A. So, if xA - xB is already less than 0.5 L we simply can assume that the distance even for the minimum image case is going to be xA - xB in this case.

But if this is not true then we have to compute the minimum image in the box keep in mind that you can only have one image in the box. If the particle is there image cannot be there, if the image is there then the particle cannot be there. There cannot be two images in the box because every image is located at L distance from each other. So, there cannot be two images in the box. So, if xA - xB turns out to be less than 0.5 L, then the particle is located the B particle located in the minimum image of A.

(Refer Slide Time: 18:51)

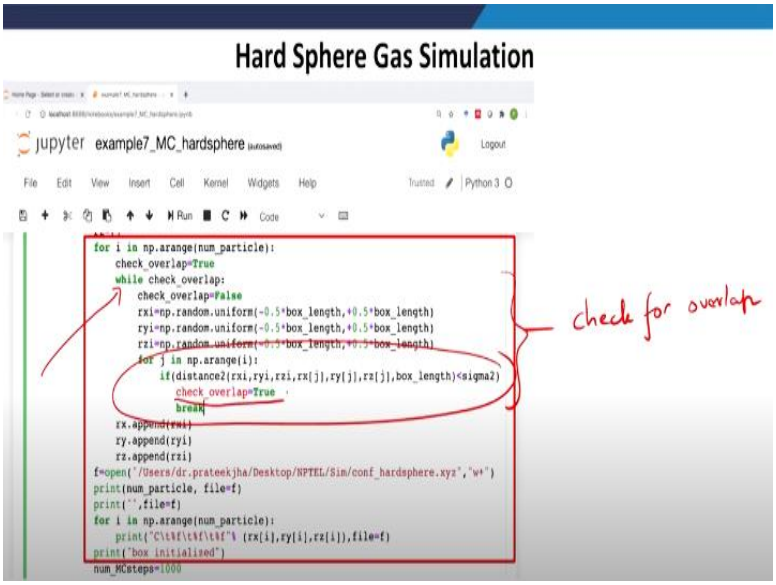


And therefore I should simply take the distance as that particular value. If this is not true then what we do is a manipulation like 0.5 L minus the d modulo 0.5 L. And modulo here gives the remainder once we divide d with 0.5 L and let us see how it works in the case that we were doing.

So, in this case clearly the original position of B was not lying in the minimum image around A. So, what we do is we compute 0.5 L minus the quantity d modulo 0.5 L. The d in this case is absolute value of 0.4 L minus 0.4 L and that is going to be equal to 0.8 L. So, we will have d modulo of 0.5 L and that is 0.5 L - 0.8 L modulo 0.5 L. So, if I divide 0.8 L with 0.5 L, the remainder is going to be 0.3 L. So, this is 0.5 L minus 0.3 L. That is equal to 0.2 L and this is the distance with the nearest image of B starting from A.

So, this really works and we can show that it works even if I jump multiple boxes to the right and left and this is the idea we always use whenever we compute the distance. There are other ways to code that but this is one of the example. So, now I have to do it for all the three directions x, y and z and this is what we are doing along x, y and z direction and then I want to start the Monte Carlo simulation.

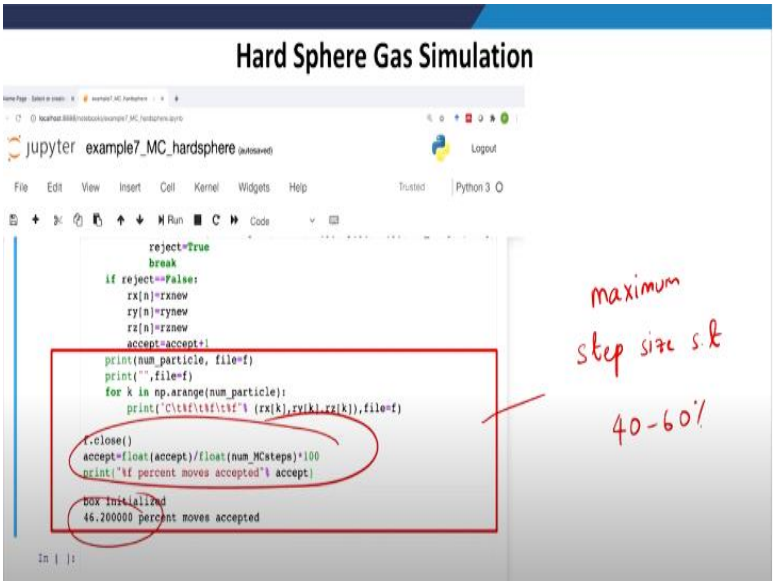
(Refer Slide Time: 23:04)



So, first of all, I have to generate the coordinates and then I have to start moving it. So, the only change from the ideal gas code will come here is I have to basically check for the overlap between the particles. So, whenever I am going to move the particle I will have some sort of a check for whether it is an overlapping or not and if it is overlapping in that case we will not accept the move, otherwise we will accept the move. So, you see I have added a while loop here. So, what the while loop is doing is it is running it until overlap is not found. So, when overlap is found, check overlap is true. So, I am running it until I can generate particles without any overlap and that has to be done only in the initial step because in the initial step I want to generate positions of all the particles so that they are not overlapping to begin with.

Now I start the Monte Carlo simulation. So, I want to again save files and all that as I have been doing and then what will change here is the condition of accepting and rejecting unlike the ideal gas case where accepted every move in this case I will reject according to a distance criteria. If distance is higher than the overlap distance and we compute the square of the distances as I was telling you we will accept it. If distance happens to be less than that then we reject it. So, σ2 in here in the code refers to the overlap distance squared and this is what we are doing. So, now if I accept it, I will store the new coordinates. If I reject it then I do not store the new coordinates. I simply move to the next system.

(Refer Slide Time: 25:15)



However, we keep track of how many moves are we accepting. So, I basically have a counter that does a plus 1 every time we accept and I keep count of how many steps we have taken that is the initial number of steps I have started with and I can get the percentage of moves I am accepting. So, in this particular case, I am for example accepting 46% moves in this case. Now, let us say for example, if very few moves are being accepted what can we possibly do.

So, it could be that my step size is large because if the step size is large, there is more likelihood of overlap because if your step is very small then the particle is only moving in the vicinity of it. So, therefore the overlaps are less likely. If I give larger displacements overlap become more likely and therefore there are more rejections which are happening. So, in that case I should reduce these step sizes.

On the other hand, if I am accepting almost all the moves, it means that I am probably not having overlaps at all. So, it will do no harm to actually increase the step size and the reason why we want to increase it is because if I increase the steps size then I would be simulating behavior over longer distances within less time because every Monte Carlo step is costing us computational time.

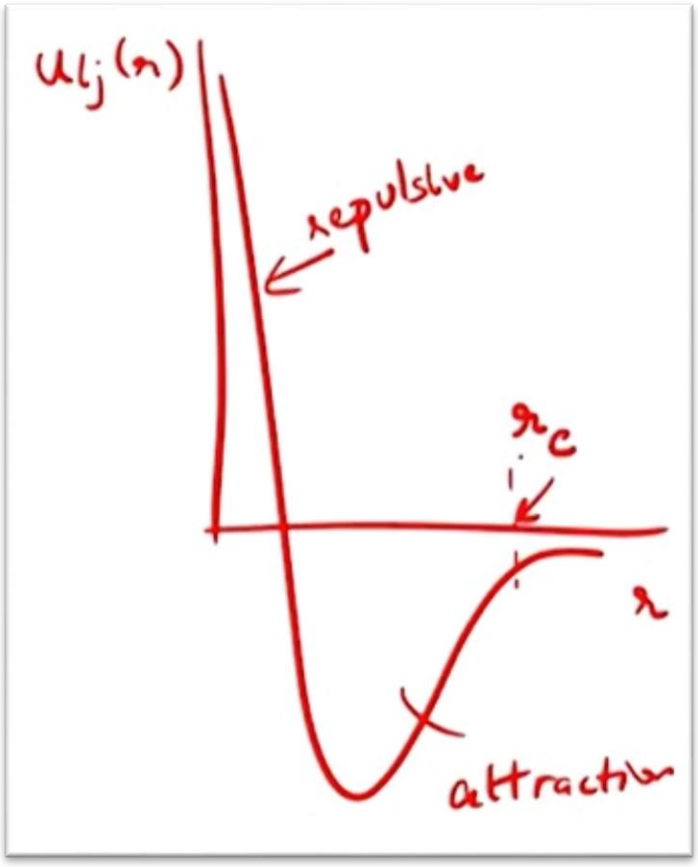
So, with smaller step size, it will take more time to simulate something as opposed to the larger step size. So, again, just like for example that we have discussed earlier of error versus computational cost. In here as well, our choice of step size is determined by how much moves are we accepting. If we are accepting all the moves we should basically increase the step size. If we are rejecting all the moves we should reduce the step size and in general as some called

kind of a thumb rule we should have the step size such that we are accepting like 40 to 60% of the move.

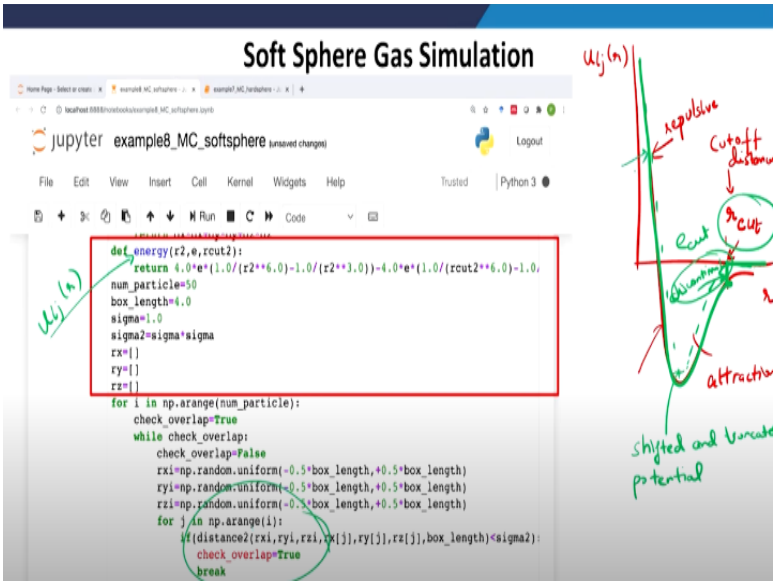
Now, in the example here what we are saying is we do not have really a constant step size. We have a maximum possible value of step size. So, we can tweak that in the example when we had a constant step size you can simply replace step size. Ultimately it does not have any physical significance so as to speak because in Monte Carlo we are interested in sampling configurations and the configurations are not really connected to each other by any means. The way we are doing it, our objective is to sample different states of the system. So, we are not trying to do any dynamics so that larger step size will give me more error or something of that sort it is just our convenience of choosing a step size to generate different configurations and we should choose it such that it is efficient.

So, if I am using lower step size then I am sampling less efficiently, I am exploring the free space slowly. If I am using a larger step size then I am sampling more efficiently but it also depends on how many moves are being accepted. And you should go with a thumb line like 40 to 60% to identify what can be a good step size for the problem. This is not to say that the equilibrium behavior will change if you are accepting only 20% moves. It may take longer to equilibrate but you should get the same equilibrium behavior provided you will run long enough.

So, whatever we are doing here is simply increasing the efficiency of the code or making the equilibrium faster. It has no consequence on the equilibrium behavior as such. The next example we do is instead of like just rejecting moves that are overlapping we actually add an interaction between the particles and the interaction that we add is of the form popularly known as the Lennard Jones interaction. And it is something like this.



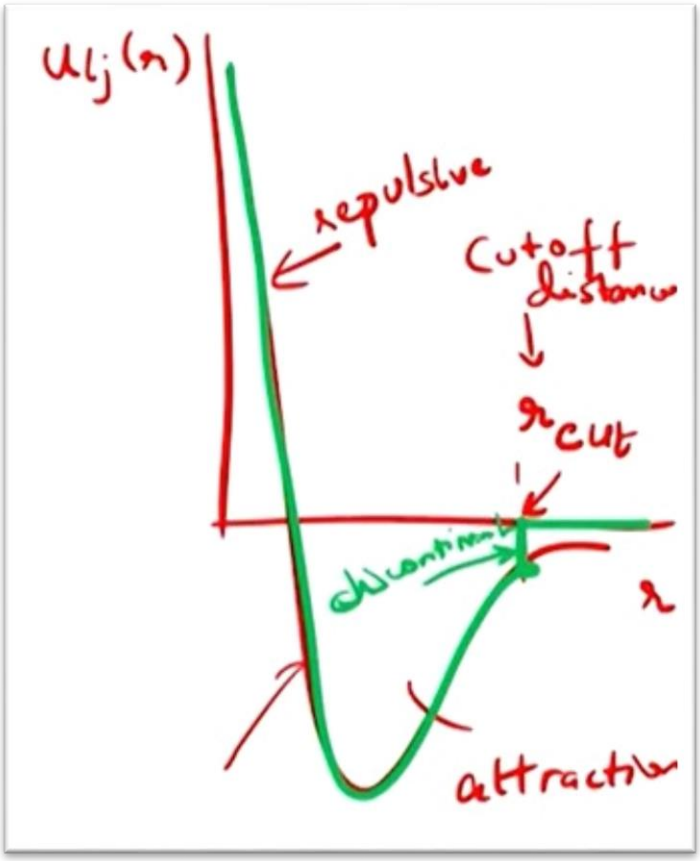
(Refer Slide Time: 29:45)



So, it has a repulsive component and it has an attraction component and if you think about it, it falls over very long distances. If you really want u_{ij} to be 0 but we can imagine a finite value of r beyond which the interactions are very small. Let me call that as r_c.

Earlier I was saying you when I was discussing the hard sphere case that our interaction range should be much smaller than the box size. So, why not manually choose an interaction range that will make it happen and there is one advantage of doing that tells me that I can define something called a cut off distance. Unlike the actual value of the r_c in the example that I was telling you it tells me when the interaction goes to 0. I will use an arc cut that is a cut off distance that is something that we can choose and therefore if I want to if I have computational power I can run for larger value of r cut and that will be more accurate in comparison to smaller value of r cut if I really want to capture this particular interaction. This is the first point that I wanted to make here.

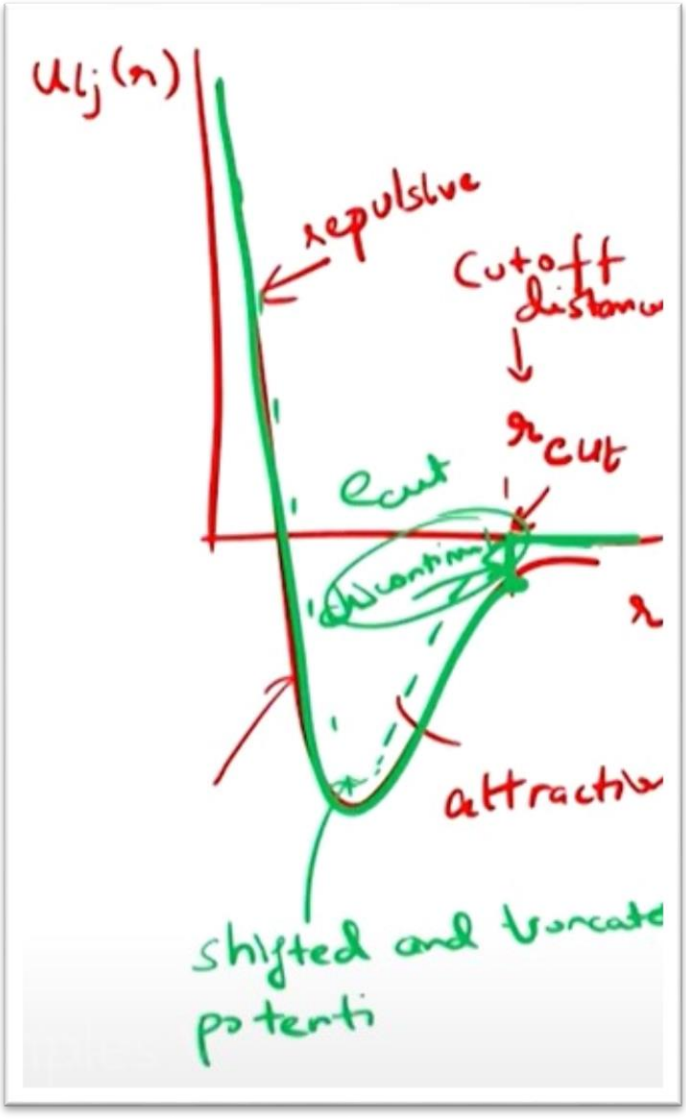
The second point is let us say if I do that what will happen is my new interaction will look like this because beyond the distance r cut I am assuming that there is no interaction. I am not even counting those interactions. So, it goes to zero beyond that. So, I am approximating the actual interaction that is the red line with the green line and where we have a small discontinuity at this particular point.



As such it is not a problem in Monte Carlo simulation because we are not interested in forces.

But it turns out that it can give you some discontinuity in the final result because there is some jump that has been artificially introduced by the r cut.

So, the practice is that when you do that you also shift the potential a bit so that it kind of becomes smooth at this so it really falls down to zero. So, whatever that small discontinuity we are having let me call it e cut that I basically subtract from the potential or add to the potential so that it falls down to zero smoothly and the dotted line is what I call a shifted and truncated potential.

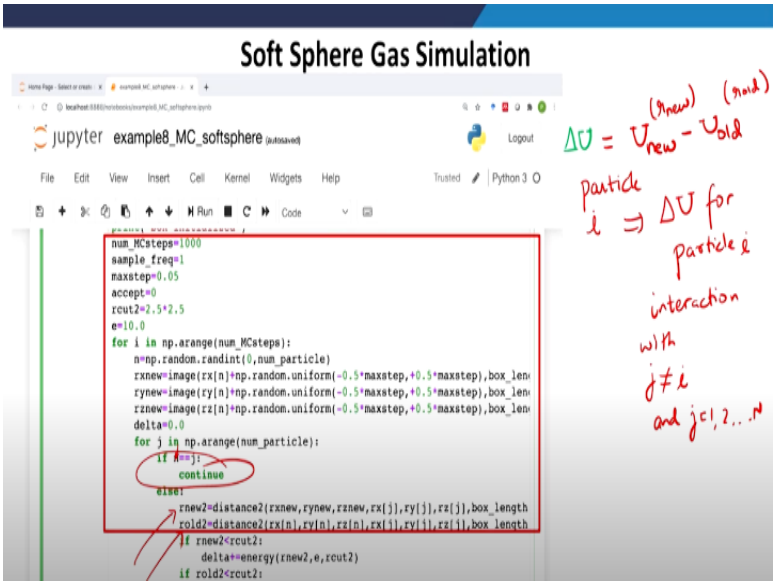


I want to leave the discussion of the actual form of potential later when we do the molecular dynamics. But in this particular example, we are using the shifted and truncated Lennard Jones potential that has this r cut variable built into this that is we are considering interactions only within a cutoff distance. It really reduces the number of pairs, for which in the forces because

if the distances happen to be more than cut off we simply do not compute the forces or energies in the case of Monte Carlo here.

So, now that is one change that we have. So, we have defined an energy function that is basically computing my u_{ij} between the pair of particles and now in the code again I can generate the initial configuration using the previous code of hard sphere that is what I am doing right here. So, I have an overlap check built into this because even in this case the overlaps are coming with large energetic penalty. It is not like hard sphere. It is not going to infinity but the energetic penalty is high. So, why to do that? So, we begin with a configuration that has no overlap and we do the Monte Carlo simulation. In that case we also add the energetics condition.

(Refer Slide Time: 34:25)



So, I am interested in finding the ΔU which is equal to the U in the new configuration minus U in the old configuration. This is what we are interested in finding.

$$\Delta U = U_{new} - U_{old}$$

So, this will depend on the r in the new configuration and r in the old configuration. Now in a given Monte Carlo step I am still moving only one particle. So, only the interaction of that particle with every other particle will change. So, I will not compute the entire U. I will only compute the energy difference that has resulted in the pair interactions of that particular particle but that particle with every part in the system. Let us say if I am moving particle i, I will compute ΔU for particle i interactions with j not equal to i and j equal to 1 to N.

So, I can go over all the particles except that particular particle and see how much the ΔU has changed and for each of those cases I have to again compute the pair interactions where i is fixed but j is changing. And this is what we are doing in this particular code everything else is very similar to the hard sphere code, except when I am accepting or rejecting the move I do not have a simple overlap condition criteria. I have to first compute the energy difference that will result from the move that is why I am finding the new and old distances.

First of all this case is when i is equal to j here. So, in the code i is presented by m. So, i I am using for something else. So, then this is where you can see in the code so I am putting the condition of cut off. So, I check for whether the distances are less than cut off then I do this otherwise I do this and so on. And then finally I apply the metropolis criteria. So, if Δenergy is less than 0 that means I am going to a lower energy state. So, I clearly accept the move. On the other hand if the ΔU is higher than 0, then I accept with a probability of exponential of minus ΔU by k_B T. So, the Δ in this code is the entire thing, Δu by k_B T.

$$probabilty = \exp\left[-\left(\frac{\Delta U}{k_B T}\right)\right]$$

And it has to be done with this probability. So, where to implement that is I generate a random number between 0 to 1 and I find whether this quantity is more than that random number I generate in the range 0 to 1.

So, let say, for example, if this quantity is equal to 0.1 what is going to happen? So, if I generate between the range 0 to 1 then if the value that comes out is more than 0.1 in that case this condition is not met when values are less than 0.1 then only this is met. And what is the probability that value is going to be less than 0.1? If I generate in the range 0 to 1 is 0.1. So, therefore I am able to accept a move with a probability of 0.1. Now if the probability becomes 0.2 and I generate in the range 0 to 1 then the number I generate will be in the range 0 to 0.2 with a probability of 0.2 because I am generating in the range 0 to 1 and I am generating it uniformly.

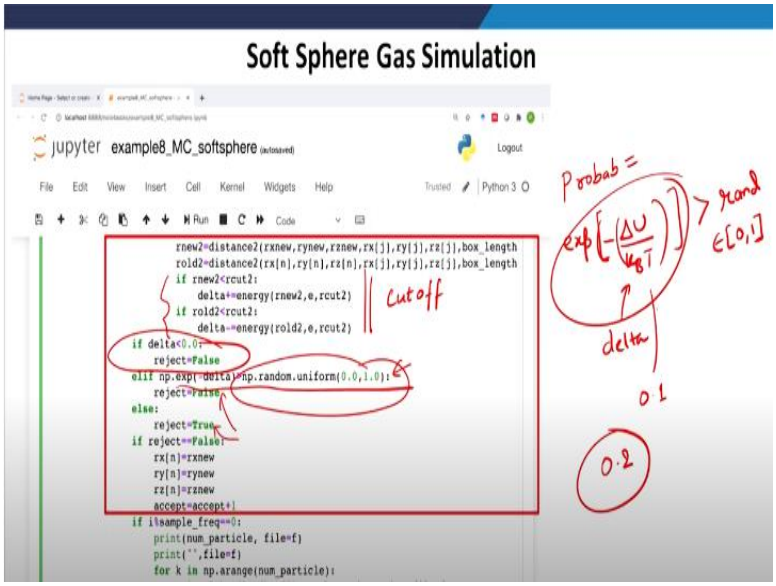
So, it can be just anywhere between 0 to 1. So, the probability for 0 to 0.2 ranges 0.2 probability for 0 to 0.5 range will be 0.5 and so on. Clearly as the probability increases we will have more range that we can cover. At probability equal to 1 it is true always because every number

between 0 to 1 is less than 1.

So, this is what this condition is doing with the again, this is a very nice mathematical trick that implements the condition that I want to accept with a certain probability without worrying about like too much math, how many outcomes are there, what is the total number of outcomes and so on. A very beautiful trick takes care of the fact that I have to accept the moves with a certain probability.

So, in that case, I accept that I have a flag that is counting rejections if it is true, then we reject otherwise we accept. So, if it is not true, then we expect the rejection or the reject the move. So, and then I store the new coordinates only when the move is accepted just like the earlier case and we pretty much print the new coordinates in the file as earlier.

(Refer Slide Time: 36:50)



So, compared to the hard sphere code, there are only few changes in soft sphere code that is why I did not go through this code entirely and that is instead of simply rejecting moves that are resulting in an overlap, now we find the energy change ΔU that energy change gives me the metropolis criteria of accepting or rejecting move. If ΔU is less than 0 we always accept, if ΔU is higher than 0 then we accept with the probability of exponential of minus ΔU by k_B T. And in this case we already work in k_B T units. So, my delta in the code is ΔU by k_B T.

So, with that particular discussion, I want to conclude the discussion on Monte Carlo scheme. I will revisit the soft sphere simulation problem in the context of molecular dynamics wherein we will discuss the idea of the cutoff and we will Lennard Jones interactions in somewhat more

detail.

So, keep in mind that all the discussions of the interactions or cut offs or any kind of numerical scheme that I am using is true for both Monte Carlo and molecular dynamic simulation because many of them are related to computational efficiency. So, if the same feature is happening in MD simulation, we should use that particular feature. So, from that perspective in terms of the computer we can use the same tricks in Monte Carlo and molecular dynamic simulations provided that they are applicable in those situations, same applies to a choice of boundary conditions. So, when I apply for example, the periodic boundary condition and we discuss the idea of the images, minimum images all of this will translate, also in the case of the molecular dynamic simulations because the system remains the same. So, this entire specification of the interactions or the boundary conditions are what is characterizing my system. So, system specification will remain the same even the way we initialize it can also remain the same between molecular dynamics and Monte Carlo simulation what however is changing is how we are evolving the system.

So, in molecular dynamics, we do not have any acceptance and rejection and in Monte Carlo we have acceptance and rejection. In Monte Carlo there is no notion of time. In molecular dynamics we have a notion of time but the prescription of the system the way we are building the system, the way we are defining the boundary conditions are going to be pretty much the same and therefore I am postponing some of the stuff that I wanted to discuss in the Monte Carlo simulation for discussion in the molecular dynamics part of the course but keep in mind that the same idea will also apply in Monte Carlo.

So, with that I want to conclude the discussion today, thank you.