

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

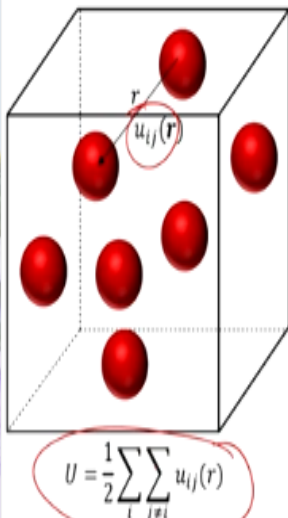
**Lecture - 39**

**Numerical Implementation of Monte Carlo Simulation: Python Examples IV**

Hello all of you so in the last lecture we have discussed the effect of boundary conditions in Monte Carlo simulations. In today's lecture, we will extend the idea to multi particle simulations that is where the system the Monte Carlo simulation is actually employed. The one particle example was only for the simulation of diffusion, but most circumstances you want to do a multi particle system that is what is the main I would say most used Monte Carlo simulation framework for thermodynamic problems.

**(Refer Slide Time: 01:17)**

**Monte Carlo Simulations for Equilibrium Behavior**



$$U = \frac{1}{2} \sum_i \sum_{j \neq i} u_{ij}(r)$$

**General Algorithm**

- Try to move particles one by one in a random direction.
- Accept/reject moves based on some criterion on the total energy  $\Delta U$  due to the move, e.g., Metropolis criterion

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

- Keep on doing this until equilibration is achieved, determined by monitoring of a desired physical property, which should converge.

So, this is a typical statement of the problem that we have in the multi particle case. So, you have some n number of particles and they are interacting with some sort of pair potentials that is  $u_{ij}$  in the example and all these pair energies if I sum over all the possible pairs of particles, they give me the total energy of the system.

$$U = \frac{1}{2} \sum_i \sum_{j \neq i} u_{ij}(r)$$

Before I proceed to the Monte Carlo simulation part, let us look at that approximation in a bit more detail for the time being.

**(Refer Slide Time: 01:43)**

$$U = u_{12} + u_{13} + u_{14} + u_{23} + u_{24} + u_{34}$$

$$= \frac{1}{2} \sum_i \sum_{j \neq i} u_{ij}$$

$$u_{ij}(\vec{r}_{ij}) \equiv u_{ij}(|\vec{r}_{ij}|) = u_{ij}(r_{ij})$$

$$\left. \begin{array}{l} u_{123} \\ u_{134} \\ u_{134} \\ u_{124} \\ u_{1234} \end{array} \right\}$$

So, what I essentially say is just to sake of argument let us say we have 1, 2, 3, 4 particles, we are saying that there are in total like  ${}^4C_2$  pairs that is 6 pairs of particles and we are summing over the pair energies of all of them because the self does not form a pair so  $u_{11}$  is not a valid pair interaction, there is no still self-interaction, but everything else is possible. So,  $U_{ij}$  where  $i$  is not equal to 0.

So, we have then  $U = u_{12} + u_{13} + u_{14} + u_{23} + u_{24} + u_{34}$ , keep in mind that we are not double counting. So,  $u_{12}$  is same as  $u_{21}$ , the energy is the same. Of course the forces are in opposite direction, but the energy is going to be same this is what is essentially-

$$U = \frac{1}{2} \sum_i \sum_{j \neq i} u_{ij}$$

However, it turns out that these are not the only energies that contribute in this problem of course, we are considering no external agency apart from that. But even if there is no external agency apart from that then also there is something else called multi body interaction. So, that means that if I look at this collection of system and this is very simple system already. There are also three body interactions, let us say  $u_{123}$  the together form a three body interaction. So, if I look in the energy of the 1, 2, 3 system. It is not represented completely by just the three

two body interactions  $u_{12}$ ,  $u_{23}$ , and  $u_{31}$  but you also have some three body interaction  $u_{123}$  that will appear when the three particles are put together.

So, how many of them that we have in this case so we can have four possible three body interactions. That is  $u_{123}$ . Let me count all of them and add them together  $u_{123} + u_{234} + u_{134} + u_{124}$ . So, one of them is lacking in all these four guests. And then in addition to that when the 4 come together then you also have a 4 body interaction and that is something like this. So, all the four together form a four body interaction,  $u_{1234}$ .

So, now you imagine that if I have some  $n$  particle system, then you have 2 body interaction, 3 body interaction, 4 body interaction, 5 body interaction until some  $n$  body interaction by using only the pair interactions we are basically not considering all the multi body interactions apart from the 2 body interactions. It turns out that in most cases particularly in the simulations of liquids and gases especially dilute solutions of liquids and gases in almost every condition.

This higher body interactions are not really so important but this is no longer true for many solids particularly where electrostatic interactions are very important. So, you may recall from electrostatics that you have things called multipole moments, which will appear when many charges are put together. They are not present between a pair of charges they only appear where when they are multiple charges present more than two.

So, in those cases, this will be a significant problem by that we are not considering the higher multi body interactions but that really makes the problem completely intractable if I start counting all the multi volume interactions. Unfortunately in even those cases it is found that the pair interactions are the most significant contribution to the problem that is the first point I wanted to make.

The second point I wanted to make is that I am considering kind of a symmetric form where, for two particles with the distance vector or displacement vector between them  $\vec{r}$ , the energy of that is same as  $u_{ij}$  of the magnitude of  $\vec{r}$  that we can write as simply  $r$  without a vector sign.

$$U_{ij}(\vec{r}) \equiv U_{ij}(|\vec{r}|) = U_{ij}(r)$$

And that means that it does not matter where the if the particles are like this or like that. So, clearly the  $r$  will change the  $\vec{r}$  vector will change, but the magnitude of that will not change in

this case that is not a problem in when the particles are spherical or assumed to be spherical but let us see for example you have particles which are not spherical. Let us say like this, now the orientation of the particles become important in that case apart from the  $r$  we also have to somehow capture the effect of the orientation of the particle. In this case I am connecting the axis of the two ellipsoids and  $\theta$  is the angle subtended by the two axis. Now that  $\theta$  will also be a part of the  $u_{ij}$  function. So, we have to now look at  $u_{ij}(r, \theta)$ , so now we are considering two vector components in some sense we are compute. Now we are not taking the scalar distance, but we are taking the vector distance as to speak.

There can be other possibilities although we are mostly dealing with equilibrium situations here. But let us say if we look at none equilibrium situations, let us say some electric field is present. In that case electric field is directional in nature. So, in that case again, we cannot look at the distances we have to look at the displacement vectors because whenever the field is directional we have to look at the effect of direction as well. But for the all the examples that I am doing in the class actually the most of them for Monte Carlo simulations and molecular dynamics as well we will be considering that  $U_{ij}$  is a function of  $r$  that is the scalar distance not the vector distance but keep in mind that this does not have to be always true.

So, going back here, so let us say we take that for granted. Now one of the methods to stimulate this is I try to move every particle one by one in random direction. So, in the earlier examples I have been doing I was looking at the motion of a single particle and it was moving first with like a constant step size but later on we said we can relax that assumption. In this case all the particles individually are moving in pretty much the same way as what we had for the one particle system. The only difference is as they are moving we are also re-computing the forces and basically energies in the Monte Carlo simulations, both force and energy is related. So, force is simply-

$$F = -\vec{\Delta}U$$

And based on the energy difference that will happen if I move this particle, we accept or reject a movement that is something that was not present in the one particle system. The particle continued to move because there was no resistance for its movement. In the multi particle system the movement may not be allowed if this is coming at a cost of high energetic penalty. If the energy of system increases significantly then we will either not allow the movement or except with a lower probability.

On the other hand if the energy is decreasing then we will always accept the move and we have discussed the same idea earlier in the framework of the probability densities and all that and we said that we can define what is known as metropolis criteria in terms of the energies that we have defined right here this is what the metropolis criteria comes down to.

So, we compute a minimum-

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

And why do we do that, because when  $\Delta u$  is less than 0, the energy is decreasing then we always accept the move. But if I put  $\Delta u$  less than 0 in this formula that number will exceed 1. But the probability has a maximum value of 1. So, for  $\Delta U$  less than 0 we take the probability to be 1 and for  $\Delta u$  higher than 0 clearly this number is going to be less than 1 and for  $\Delta U$  I would say close to infinity this number is going to be 0. So, this makes my probability well defined in the range 0 to 1.

Now clearly when we begin the simulation we do not know what is the equilibrium state because if we knew the equilibrium then what is the point of doing the simulation. Simulation basically aims to attain the equilibrium behavior and if we know that then why are we doing it. So, therefore we will start with some initial configuration that we know may not be at equilibrium.

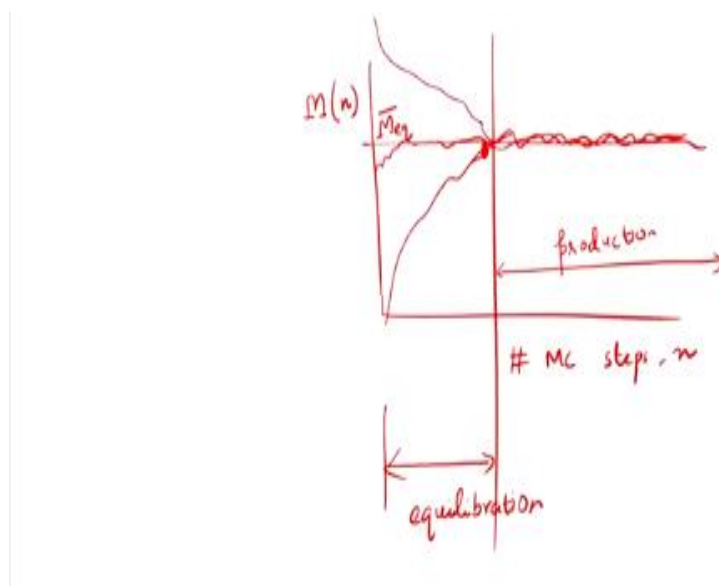
In other words, we may generate with a probability density or whatever that is not same as the stationary solution in the phase space. So, the way we are generating a configuration because I was telling you that if I read on my Monte Carlo simulation, I will not get the same configuration. The same way when I generate a configuration or initial value for initial state, it is going to be different in every none of the simulation. So, if those initial states are being generated using some kind of a probability density in the phase space that will not be close to the stationary solution of that and therefore we will not be in the equilibrium state. So, therefore we have to start from some configuration and continue running the Monte Carlo simulation until we say that equilibration has been achieved.

The criteria we already know the criteria we know that my  $\rho$  of  $\Gamma(t)$  should converge to  $\rho$  of  $\Gamma$ . It turns out that even though this is rigorously valid, this has very less practical significance because in order to find the  $\rho$  I need to do many-many simulations and see which part of the

phase space the system is and how it is moving and so on and so forth. There is an easier way of doing it and that is we simply monitor a physical property that I am interested and clearly since when we begin with we are not in equilibrium, or we may not be in equilibrium that property will not be the equilibrium value and therefore we will have very large fluctuations or increase or decrease in the property when we begin. When we reach the equilibrium state then we should start fluctuating around an average value. This is how we defined the equilibrium earlier.

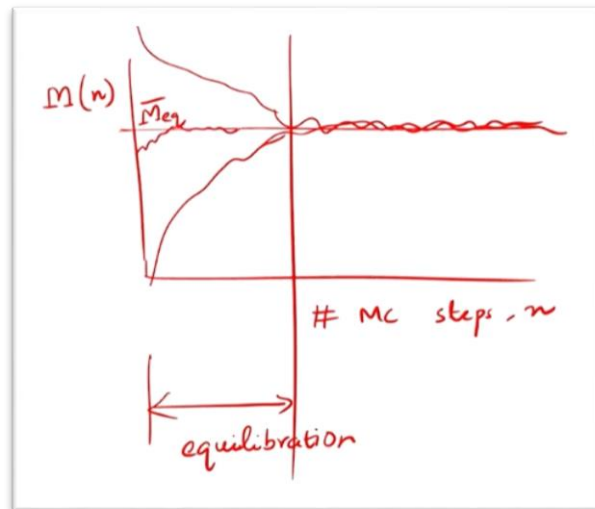
So, we should start fluctuating around some evidence value, of course, the system will not stop moving. So, they are going to be fluctuations but we have to wait until we see that the fluctuations above the average are somewhat compensated by the fluctuations below the average and this is what we call the equilibration in the Monte Carlo simulation.

**(Refer Slide Time: 15:10)**



What this comes down to is something like this, so let us say for example if I am tracking a property  $M$  as a function of my Monte Carlo steps  $n$  and this is my number of MC steps and the same idea will apply also in molecular dynamics but then we will have a time axis as opposed to a number of Monte Carlo step axis. And then in the very beginning it may so happen that the property is starting from a lower value because we do not know the equilibrium state or a higher value in comparison to the equilibrium average value for a system. So, in the very beginning the system will try to approach the equilibrium value the system will tend to go towards the stationary state. And as long as we are doing the thing correctly, it will really get

there. But then eventually it will start fluctuating around that equilibrium state. So, the initial part is what is referred as the equilibration in the Monte Carlo simulation.



And the equilibration phase you may imagine will have strong dependence on the initial condition. If we started closer to the equilibrium state let say here then we can quickly get to the equilibrium value. If we started farther from the equilibrium value then we can will take longer. And since in most cases we do not know what the equilibrium value is, it is typically I would say just our sheer luck in most cases or we can make some educated guess to begin with that will make our job easier. But nonetheless we do not have so much control on how much time the system takes to equilibrate but really have to run long enough so that it is equilibrating.

Now we can use some efficient sampling strategies we can make some smart movements in the Monte Carlo simulation while satisfying detail balance that will make the code more efficient will have a faster equilibration as opposed to slower equilibration in the case of like simpler moves that we have discussed. So, that part we can always improve based on the quality of move I am making the step size that I am choosing and so on and so forth and the key thing here is that the properties are evaluated not in the equilibration phase, but in what is known as the production phase that will occur after the equilibration has been achieved.

This can be a separate run starting from a configuration that I obtain after an equilibration has been achieved signified by the fact that the values are now fluctuating around an evidence value or we can just do a single long run and discard some initial part that we can attribute to the equilibration And along those points in the production range we will compute some sort of evidence that I am interested in and that should be the equilibrium value.

So, clearly since we are doing a finite size system and since it is stochastic simulation it is not that every Monte Carlo simulation run will give you the same answer. So, again, we have to look at the error versus computational cost trade off. So, we will accept the errors within some tolerance that we can specify provided that the computation can be performed within a reasonable time that we can decide and depends on the computer that we have or we can afford.

So, with this particular idea let us now start looking at some of the basic codes. But this particular schematic is more or less same for any Monte Carlo simulation. The form of  $U$  can be complicated, the Monte Carlo movements I am making can be complicated because I can choose movements that can give me a faster equilibration or what is known as an advanced sampling strategy or whatever which satisfies the detailed balance, but it gives me a half faster equilibration all of that we can do but nonetheless the basic schematically remains the same we do not have to use metropolis criteria always we can use some other criteria but it is going to be a function of  $\Delta U$  nonetheless and we always start with initial state and we always wait for the equilibration and we monitor the property of interest that is a general scheme that we follow irrespective of the problem that we are trying to solve in the thermodynamics case.

**(Refer Slide Time: 20:14)**

### Ideal Gas Simulation

```
f=open('/Users/br-prateek/Desktop/SPTEL/Sim/cont_idealgas.py','w')
print(num_particle, file=f)
print("", file=f)
for i in np.arange(num_particle):
    print("C:\nX:\nY:\nZ", (rx[i],ry[i],rz[i]),file=f)

num_MCSteps=10000
maxstep=100
for i in np.arange(num_MCSteps):
    n=np.random.randint(0,num_particle-1)
    rx[n]=maxp[0]+np.random.uniform(-0.5*maxstep,0.5*maxstep),box_len
    ry[n]=maxp[1]+np.random.uniform(-0.5*maxstep,0.5*maxstep),box_len
    rz[n]=maxp[2]+np.random.uniform(-0.5*maxstep,0.5*maxstep),box_len
    print(num_particle, file=f)
    print("", file=f)
    for i in np.arange(num_particle):
        print("C:\nX:\nY:\nZ", (rx[i],ry[i],rz[i]),file=f)

f.close()
```



So, let us start with a simple example of an ideal gas simulation. In the case of an ideal gas, the advantage is that my  $U$  is equal to 0 there is no interaction between the systems. So, in this case essentially if I look at our metropolis criteria the first term exponential of minus  $\Delta U$  by  $k_B T$  that is also equal to 1. So, in that case pretty much it means that we will accept every possible move.

So, this means that in our system we are basically every particle in the system is moving in the same way as in the one particle simulation we did earlier because there is no resistance to their movement, every move is being accepted because they can overlap or other particles the particles are point like and there is no preference to go towards in particular direction because there is no energetics involved in here.

So, let us say I want to simulate that one advantage is that I do not have to worry about computation of the forces. So, it is going to be a fast simulation. So, simulate that I am doing this particular codes, so I start with importing the numpy library for math functions as I have been doing and then I define this particular thing called an image function and we the idea of the image function is the following.

First of all, let us see why we are defining a function, so it turns out that if I have to do the same arithmetic operation many-many times in the code, it is more convenient to define a function that will do that. That is really comes for those of you who are used to see your C++, that is a very common thing we define a function and then we simply call that function within the code. The same thing can be done here. We simply define the function in the top and call it many times in the code that really makes it very easy to make changes to the code or for debugging because let us say if you figure out that one expression here is wrong, let us say we change the plus to minus or minus to plus if we are not using a function if we are simply putting the same expression again and again in the code. I have to make the same change in every appearance of that expression.

So, therefore if we have defined the function we can make the correction only once and since another places we are calling the function the change will be automatically carried out, that is the advantage of using a function. So, in this case we are defining an image function and what essentially it achieves is it gives me the coordinates of image that lies within a simulation box after the correction by periodic boundary condition, you may recall the expression that I am using I compute an index then I subtract something like index multiplied by a with  $l$  from the

from the position. This is how we corrected for the periodic boundary condition in the previous example, same thing that we are doing here.

Now I am working with 100 particles in a box of size 10. I declare some arrays to store the x, y and z coordinates of all the particles in the system. Now those coordinates are defined for a particular time in reality but we start with we have to give some initial state and this is what we first do so we initialize the simulation.

So, in that part of the thing what we do is we randomly generate the coordinates of particles in the box that is I would say a good guess at least for the ideal gas simulation. Even if we do not know the initial configuration, we know that for an ideal gas the probability of being anywhere in this space is pretty much equal to 1 and therefore we can randomly generate the configuration. It turns out that even if, we are doing that for an ideal gas we are already at equilibrium. Because this is what an ideal gas behaves like. So, our initialization method itself is giving me the equilibrium behavior. So, the very trivial example, but nonetheless it sets the ground for the case when we start accounting for the use.

So, now I am generating the random positions of particles. So, at the end of this for loop that is looping over all the particles in the system, I will have the initial coordinates of the particles.

So, now I create a file on my computer, so you see I am giving a folder location exact folder location may differ depending on whether using windows or Linux. But this is specific to your system where you are putting the file or where you want to save the file. So, I want to create a file `conf_ideal_gas_dot_x_y_g` in the folder that is `users/dr.Prateekjha/Desktop/Nptel/sim` and so on.

Now in that particular file, I want to first print the number of particles and I will come to the file format in a minute and then I want to print the coordinates of all the particles. This is what we are doing here. So, I print all the coordinates, so I am looping over all the particles and I am printing the coordinates. This is how I want to format this. So, I want to have C written then I should have my x coordinate then I should have my y coordinate then I should have my z coordinate and we should have coordinate of one particle in a line. It turns out that this particular formatting is important when I try to visualize this that we can see in future slides.

So, then this is how we have already initialized it and we have saved these coordinates in there. I do not want to do visualization as part of the python code here. And the reason is that we are doing visualization in a separate software also we can imagine that when I will do Monte Carlo simulation I will have too many coordinates to write why to do any kind of visualization along with Monte Carlo code it is going to slow down the code. So, it is better to save the file and then we can see the coordinates later.

Then I start doing my Monte Carlo simulation. So, I want to run for 10,000 steps. I define a maximum step of 2. So, at any given time, I can make like steps of 2 length in any particular direction and then I generate the new position as the old position plus some random displacement and as soon as we compute the new position, I compute its image in the box. So, this entire thing is called within by the image function. So, I pass on this particular position to the image function and this gives me the image of the particle in the central simulation box. If we were interested in the mean square displacement, we will also worry about storing the original position in this case. I am not interested in that. So, I am only saving the images that will lie in the center box that is for minus 0.5 to plus 0.5. So, I do it for the x y and z coordinate for a particle index n and that is an important point here.

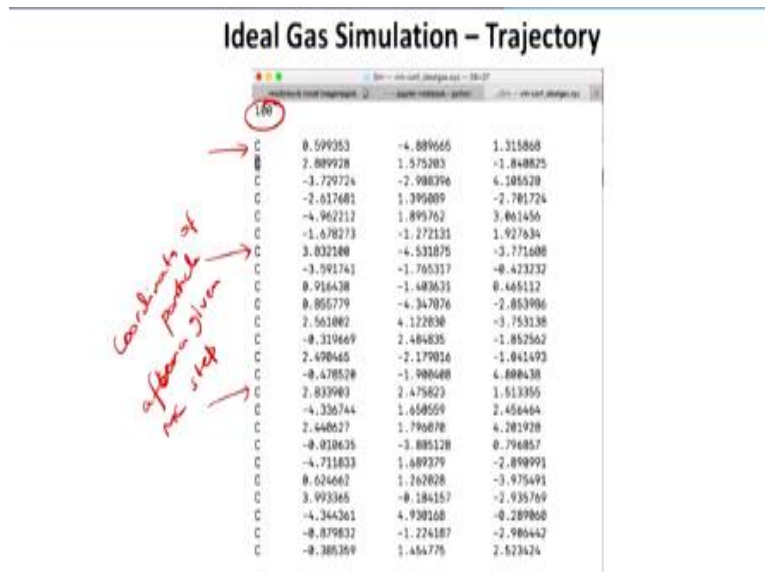
So, for in every Monte Carlo move I am first generating n as a random number random integer, if you see look at it around in function between 0 to n minus 1. So, particles are indexed 0, 1, 2 to n minus 1. So, within every Monte Carlo step, I am picking one of the numbers between 0 to n minus 1 that amounts to picking one of the particle to move within the Monte Carlo simulation and the reason why I am doing this is if we have not done it then I would be moving every particle one by one that is not what physically happens, no particle waits for the other to move. In reality, it is more natural that any particle can move at any given time.

So, therefore we are picking a particle randomly and we are moving it. It turns out that this is an example of an advanced sampling strategy, the more random is our method of doing Monte Carlo simulation, the more closer we will be near the equilibrium state any sort of order, let us say defining the arrangement of I will move, this first that second this third is going to move us far from the equilibrium state.

So, now we again save the coordinates in the same file as new lines in there. Again, the number of particles and the new coordinates in there, so therefore the file would that then contain

basically a trajectory of the entire system at a given Monte Carlo step. I am printing coordinates of all the particles in the system and this I am repeating for every Monte Carlo step in the process. And the printing command therefore remains the same as earlier when I wrote the initial coordinates. In the same file I am appending the coordinates after every Monte Carlo step and then simply close the file.

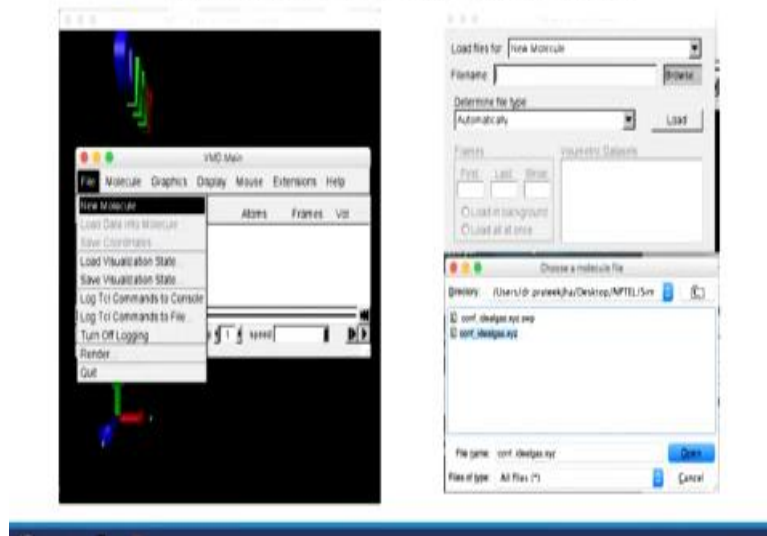
**(Refer Slide Time: 30:51)**



So, this is how the file looks like. I am printing in the number of particles and each of the lines are represents the coordinates of particles for a given MC step or after a given MC steps. In the beginning I put the print by initial coordinates and then I print after every Monte Carlo step. This is what it contains these are my x y and z coordinates of the particle.

**(Refer Slide Time: 31:30)**

## Ideal Gas Simulation – Visualization in VMD

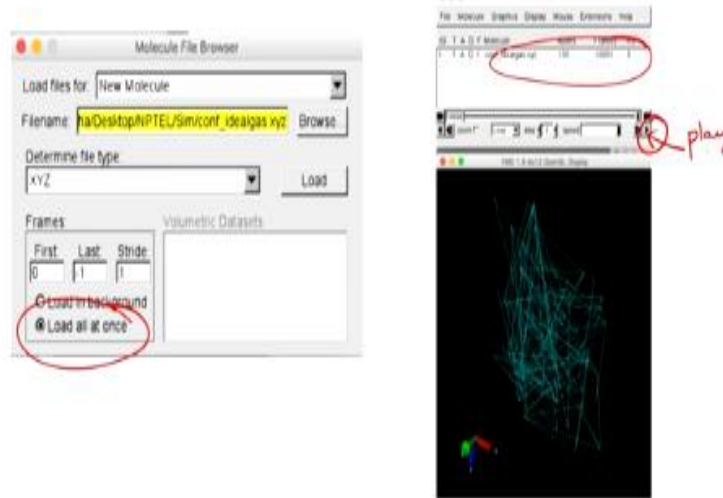


Now I want to visualize this particular system and we use an open source software called VMD. There are many other ways to visualize this. So, if you if you see here this particular file format that I have chosen that starts with something like c then takes a tab prints the x coordinate and takes a tab prints the y coordinate, takes a tab prints the z coordinate and every coordinate appears in a new line and the fact that we give the number of particles as the first line of the code and after every Monte Carlo step we again print the number of particles.

All of this is particular to the file that VMD can read many number of file formats and depending on the convenience or depending on what features we want, we will choose that particular file format and we simply print our coordinates in that particular file format. So, all of that we are doing right here is just to make the visualization easier in this particular case. So, now if I if I install VMD and load it, you have you can go to files choose a new molecule and load the particular xyz file that is from the folder that we are in you have saved in and you want to load all at once because you may have for example 10,000 steps.

**(Refer Slide Time: 32:57)**

## Ideal Gas Simulation – Visualization in VMD

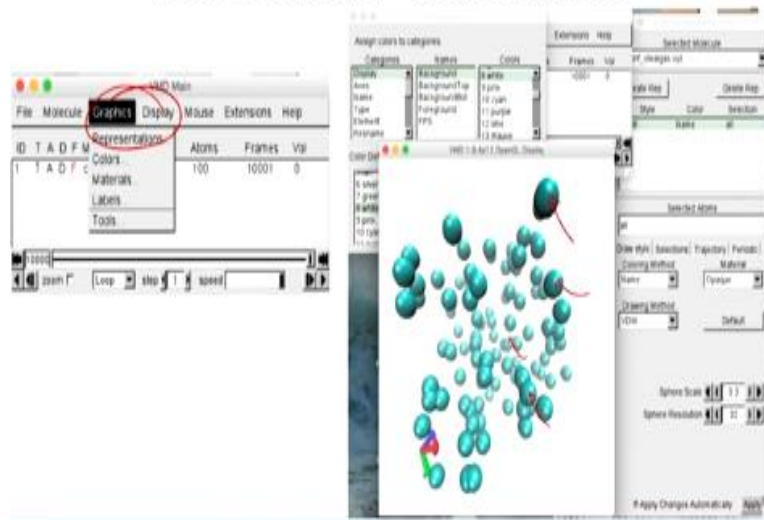


So, 10,000 Monte Carlo steps and initial set of coordinates so in total you will have 10,000 to 1 frames. So, as to speak interest actually we have every frame contains the coordinates of all the particle after a Monte Carlo step. This is what comes in then. So, it is showing all the all the frames if I want to play a movie. I can simply play this particular thing and when you load it in VMD in the beginning, it does not look so good because it by default chooses a line drawing option, it connects every point with a line you want it to look better the particles would look like is spheres you can play with the background and all of this is possible by going to the graphics menu to representation and I am doing quite a lot of settings here just for this to look better.

So, now I see these are my particles all of them are particles. The particles which look smaller are farther down and particles which are bigger are simply closer to from the direction, I am looking at it. All of the particles are of the same size. In fact, for ideal gas the size is 0 but I want to visualize this. So, therefore I have given a finite size and you can play with lot of settings for this to be a very nice movie in fact, you can also set the box in this case the box is of the length 10. So, I can specify the box length, it will so a box around centered at origin with a length of 10. And you can also play a movie and store the movie. In fact, there is a whole set of visualization tools in VMD such as the movie maker using which I can make a movie a beautiful movie.

**(Refer Slide Time: 33:53)**

## Ideal Gas Simulation – Visualization in VMD



So, if you have seen any presentation on molecular simulation, they always saw this colorful movie. Then it is not very difficult to create those movies using the movie maker tool VMD. So, with this particular example, I want to close the discussion here. I will discuss some more complicated Monte Carlo simulations where I also account for the interaction between the particles. This is going to be what we discussed in the next lecture.

Thank you.