Advanced Thermodynamics and Molecular Simulations

Prof. Prateek Kumar Jha
Department of Chemical Engineering
Indian Institute of Technology, Roorkee

Lecture - 47
MD Simulations – Efficiency and Parallelization, Sampling and Averaging, Analysis of Simulation Trajectories

Hello all of you, so in the last class we have discussed the MD algorithm and now I will come to the point of like how exactly the MD simulation is done on computers, mostly since the computations are time intensive, we have to resort to parallel simulations some very simple simulations you can program on your laptop or run your laptop, but in most cases you are you have to use high performance computing typically workstations or very nice computers with large number of cores.

It turns out that this doing simulations parallel or making use of the parallel computing is easier for molecular dynamics then compared to the Monte Carlo simulation and the reason for this is that in molecular dynamics we are computing forces which are somewhat local computation, especially when we are dealing with short range interactions as opposed to Monte Carlo where we are computing energies, which is actually a global variable defined over the entire system and, it turns out that it is easier to distribute work into different cores in an MD simulation because we are doing local computation as opposed to Monte Carlo simulation.

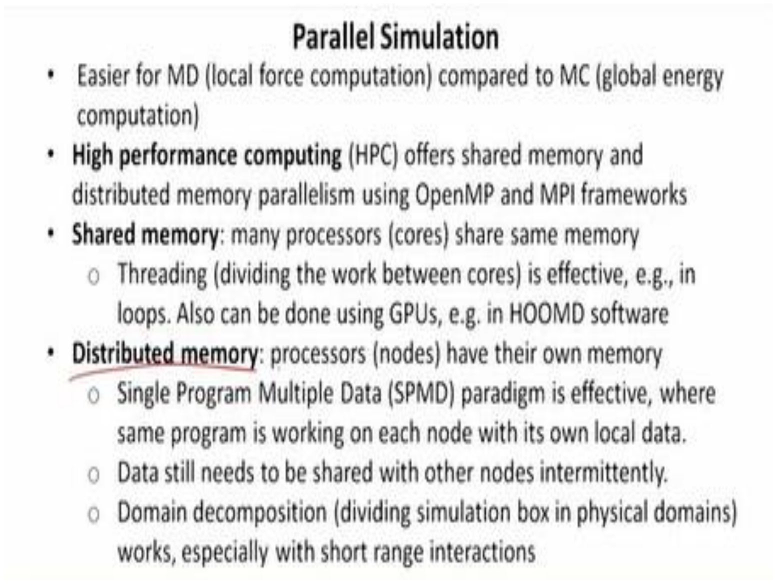
So, in high performance computing whenever we are using multiple cores, there are essentially two different paradigms one is a shared memory paradigm and one is a distributed memory paradigm. So, it used to be that distributed memory was more common because, computers did not have too many cores earlier people used to have use many, many computers and collect them and perform computation over that. Nowadays, workstations contains good number of cores and all these cores share the same memory and therefore shared memory is also becoming very popular actually it is picking up more space now than compared to the distributed memory cores earlier.

So, there are essentially two kinds of frameworks in both of these one is an open MP framework and an MPI framework, there are some others as well, but these two are most commonly used and since the basic idea how the memory sharing happens is different in these two the parallelism has to be done somewhat differently in these two. So, since they are memory case all the processors share same memory so, essentially you run a single code and that code assigns work to multiple cores on your machine and then the cores do their job send their data back to the same machine and by doing that what essentially we are doing is known as a threading kind of an operation.

So, in a particular single code you can create threads the essentially divides the work into various cores of your computer and then whatever work they do they give it back to you and then you compile that and so as the output.

Now with the use of GPUs it is becoming even more easier you can much I would say very inexpensive hardware because GPUs can then that can do threading very efficiently, this is how most gaming things work, in fact, there is a whole software for MD simulation called HOOMD that essentially is coded in graphical processing units.

(Refer Slide Time: 03:55)

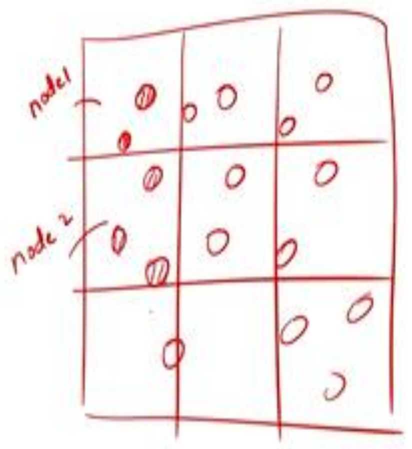


Parallel Simulation

- Easier for MD (local force computation) compared to MC (global energy computation)
- **High performance computing** (HPC) offers shared memory and distributed memory parallelism using OpenMP and MPI frameworks
- **Shared memory**: many processors (cores) share same memory
  - Threading (dividing the work between cores) is effective, e.g., in loops. Also can be done using GPUs, e.g. in HOOMD software
- **Distributed memory**: processors (nodes) have their own memory
  - Single Program Multiple Data (SPMD) paradigm is effective, where same program is working on each node with its own local data.
  - Data still needs to be shared with other nodes intermittently.
  - Domain decomposition (dividing simulation box in physical domains) works, especially with short range interactions

In the distributed memory paradigm you do not have the same memory, so you have different nodes or different processors having their own memory and essentially you create typically copies of code that run on all of these nodes and this is what is known as an SPMD paradigm that is single program multiple data paradigm. So, every processor or every node is running it is own version of the program it is working with it is own local data and every so often the data has to be shared between the nodes whenever we want to compile the results or whenever there is a need of data from the other nodes and one way to do that is we simply divide the simulation box into physical domains something like if you have a simulation box like this and let us say these are my molecules I can divide them into for example, 9 domains and all these 9 domains are simulated on different nodes of the HPC platform you have and let us say for example these two particles there motion is being solved in node 1. These three are being done in node 2 and so on clearly the particles may move from one domain to the other and, whenever it does that then that information has to be transmitted between two nodes and therefore we can perform some computation over every node but we also have to do some communication between the two nodes that you have.

(Refer Slide Time: 04:48)



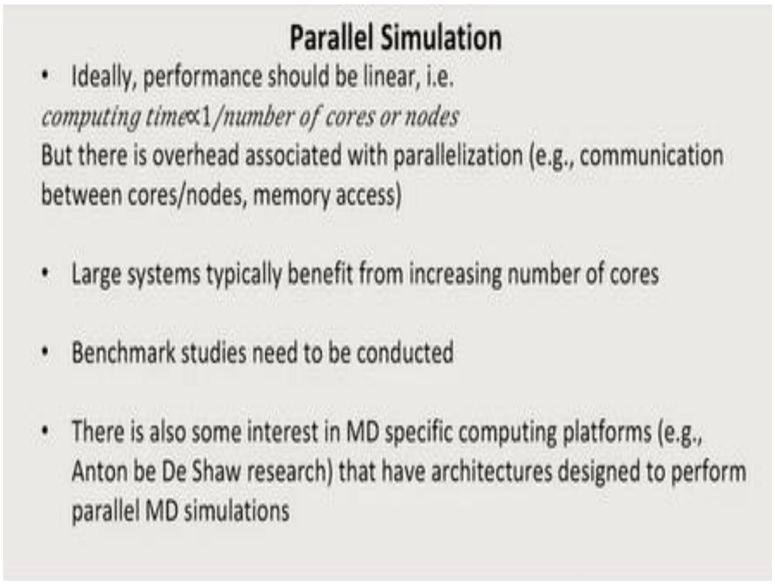
So, it turns out that domain decomposition is particularly effective when you work with short range interactions because in that case is the interactions that are present between the particles are typically confined within a single physical domain we create they are not really long range as in electrostatic interactions in which case the interactions themselves in compass multiple domains in which case is become kind of challenging but for short range interactions it is greatly effective and this is the I would say most commonly used paradigm in most software's.

So, ideally what we really want is if I double the number of cores I should double the computing speed that means the given stimulation should take half the time. So, the computing time ideally should decrease with increase in the number of cores or the number of nodes depending on the fair or distributed computing. It turns out that it is never true and the reason why it is not true is because although all the cores are doing their jobs. So, naturally the amount of effort every core is putting in is equal and, therefore using two cores means double the effort, but we also have some effort because of the communication between the cores and that communication is not present when we are doing a serial code on a single core.

So, as we start using more and more core we are also spending plenty of time communicating between the cores or nodes. So, it turns out that this if the systems are really large then they can benefit immensely from increasing the number of cores. On the other hand if systems are small then increasing the number of cores does not give rise to so much increase in computing efficiency. In fact, in some cases it is found that computing time actually become larger when you increase the number of cores because in those cases the communication takes more time than the actual computation.

So, therefore in whatever system we are trying to simulate we should think of doing some benchmark studies on HPC platforms, identify which parallelism to go for, whether it is shared or distributed and then how many cores are good enough for the application that we are doing? It is not always the case that if I keep doubling the number of cores my simulation becomes automatically faster. In fact, there are limitations to how many cores that we can use while still getting a good improvement in the computing efficiency. It turns out that 32 cores is ideal in most of the cases anything beyond that gives rise to pretty much no increase in computing efficiency, but in certain cases when we are doing very large systems, then in those cases we can go until 512 or 1024 cores and even higher. It really depends on the system size and they are interactions that are present in the system.

(Refer Slide Time: 09:02)



Parallel Simulation

- Ideally, performance should be linear, i.e.

$computing\ time \propto 1/number\ of\ cores\ or\ nodes$

But there is overhead associated with parallelization (e.g., communication between cores/nodes, memory access)

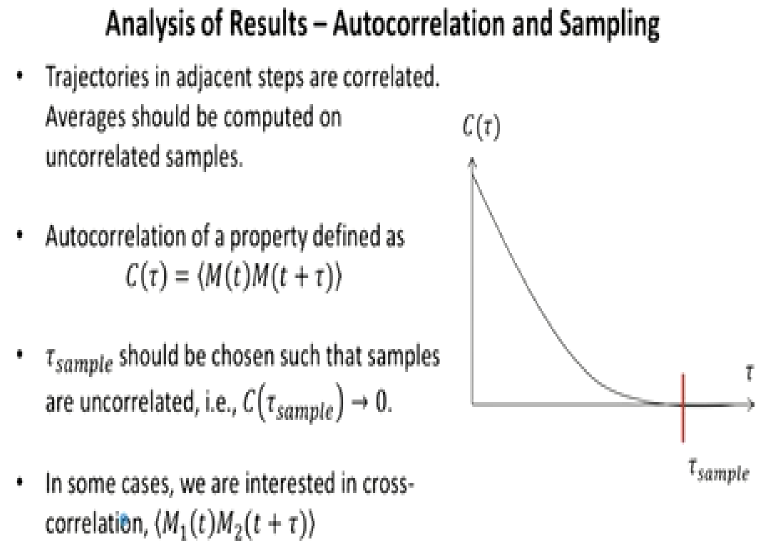
- Large systems typically benefit from increasing number of cores

- Benchmark studies need to be conducted

- There is also some interest in MD specific computing platforms (e.g., Anton be De Shaw research) that have architectures designed to perform parallel MD simulations

There is also some interest recently on making computers that are designed to do molecular dynamics. So, the architecture of the computer it is designed in a manner that it can do MD simulations most efficiently. For example the whole idea of dividing systems into various physical domains can be done at the hardware label as well we can design our cores in a manner that they will basically do parts of the MD simulation. So, that the hardware is integrated with the software that will definitely provide more efficiency of the MD simulations. For example, the ANTON software of the De Shaw research does.

(Refer Slide Time: 09:44)

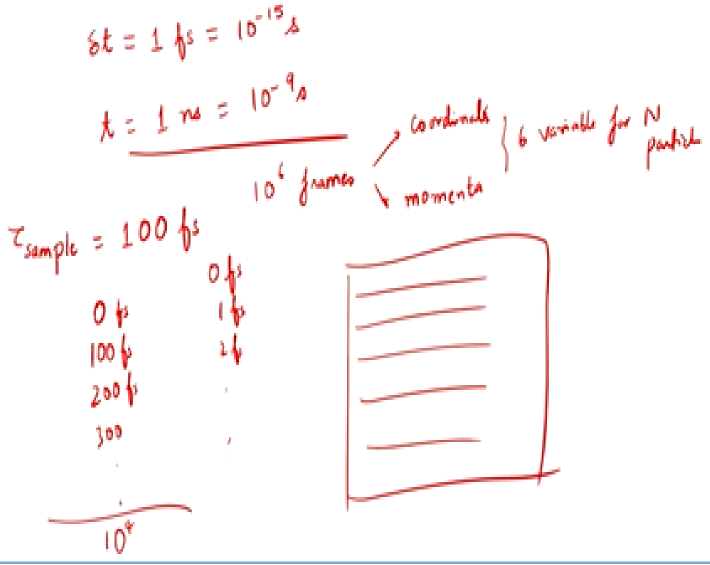


Analysis of Results – Autocorrelation and Sampling

- Trajectories in adjacent steps are correlated. Averages should be computed on uncorrelated samples.
- Autocorrelation of a property defined as $C(\tau) = \langle M(t)M(t+\tau)\rangle$
- τ_{sample} should be chosen such that samples are uncorrelated, i.e., $C(\tau_{sample}) \to 0$.
- In some cases, we are interested in cross-correlation, $\langle M_1(t)M_2(t+\tau)\rangle$

So, the next thing that comes in is how do we analyse the results of the molecular dynamic simulations? So, the first thing an MD gives you is essentially trajectories of the system that means we get basically a whole collection of the coordinates and momenta of particle at different time steps. Now, it turns out that you will have too many of the frames if you start collecting every time step. So, let us say for example we are working with δT of 1 femtosecond that is 10^{-15} second and let us say I want to simulate for time is equal to 1 Nanosecond that is 10^{-9} second this would mean if I collect my coordinates and momenta are in every time a step I will have 10^{6} frames and each of these frames will contain the coordinates and the momenta that is in total 6 variables for the number of particles in the system.

(Refer Slide Time: 10:22)



So, imagine that we are writing the coordinates and momenta in simple text file and we have like 6 elements in the text file in every row for every particle in a single frame. So, for one frame, you will have n rows containing 6 entries each but then you will have 10^6 such frames for this very simple simulation. Even that text file will be so huge that we will take plenty of the memory on your computer and, let us say even if we save that ultimately we have to compute properties using that data. So, any analysis tool will have to 10^6 frames and compute properties using that and it turns out this is not really worth doing in most cases.

In fact, this is not recommended to be done in any case and the reason is because many of these frames are actually going to be correlated we are interested in computing averages. Averages should be computed over samples which are essentially uncorrelated and therefore we should first check when the correlation vanishes in the system. The reason why we say that the frames are correlated is because let us say if I look in a timely step of say one frame to second within that small time step the previous coordinates will only change by little bit.

Let us say one or two particles or few particles or a collection of particles will move only by little bit in one femtosecond and therefore the coordinates in the new file after δt will pretty much be very close to the old file whenever that happens we say that these two are correlated.

If I compute properties for the previous frame and the new frame you will find there is some degree of correlation. And to find how the correlation changes between two frames, we can define a quantity called auto correlation that is defined in this particular wave that is I find a property at time t and I find a property at time $t + \tau$. Now, for the same value of τ I can do it for different value of t and compute the average. So, that we get a function C of τ, that is the essentially tells me the time at after which the correlation has this particular value.

So, Cτ is the value of the correlation after τ time has elapsed starting from any given point in the MD simulation. So, clearly when the tau is small we are looking at closely spaced points in the MD simulation. The immediately next point will have τ equal to δt the following point will have τ equal to τ t and so on as we look further in time in the MD simulation we are looking at larger values of τ and naturally immediately after δt the next frame is clearly highly correlated but as we go frame by frame after so many frames the correlations should start to decay and, that is when we will start to get independent sample.

So, although after 1 femtosecond or 10 femtosecond or 100 femtosecond that 2 coordinates may be quite similar but if I go to say 1000 femtosecond or from larger value that correlations should become smaller and when will that happen is can be seen by looking at the autocorrelation function versus τ, precisely what we do?

So, we choose a sampling frequency when this C of τ goes to 0. So, τ sample is chosen at a point where the C τ 0 again for any given system we can do some benchmark study and find an auto correlation time find what τ sampling should be appropriate or if we have done a similar system in the past pretty much we can use that particular data.

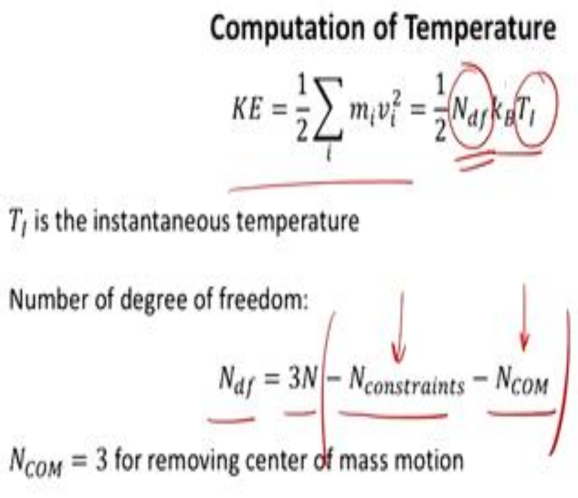
The whole idea is we want to ensure that the samples I am averaging over are uncorrelated. So, in that case in the previous picture, I will put some sampling frequency. Let us say the sampling frequency is 100 femtosecond. So, in this case, I will save data for 0, 0 femtosecond, 100 femtosecond, 200, 300 and so on unlike in the original case we are doing 0 1 2 and so on.

So, clearly now we have only 10^4 frames that is 100 times less in size than compared 10^6 frame not only we have less number of frames but also we have less correlation between the frames and therefore we can hope to get better averaging. So, we always should keep sampling frequency reasonably large not only because it will save our storages but also because we really want to get independent samples.

In some cases we are also interested in cross correlation. Let us say we have two properties M 1 and M 2, we can look at the correlation of the M 1 value at time t and M 2 value at time t plus τ. So, it tells me how the M 2 value at a later time is correlated with respect to M 1 value at a previous time. This is particularly useful for computation of properties like diffusion coefficient, viscosity, and so on we can come to that later but this is also something that one can compute using the same ideas and this now contains the correlation between two different properties.

So, using these ideas, we save our trajectories and then, we often need to find two things in our MD simulation and these two things are temperature and pressure. If we are doing in the canonical ensemble, we want to control the temperature. So, we have to keep track of how the temperature is changing. If we are working in say and NVE ensemble then temperature can be an output variable where we can find what is the average value of temperature. Similarly, if we are working in the NPT ensemble, we want to control pressure.

(Refer Slide Time: 18:24)



Computation of Temperature

$$KE = \frac{1}{2}\sum_i m_i v_i^2 = \frac{1}{2}N_{df} k_B T_I$$

T_I is the instantaneous temperature

Number of degree of freedom:

$$N_{df} = 3N - N_{constraints} - N_{COM}$$

$N_{COM} = 3$ for removing center of mass motion

So, therefore we so monitor what the pressure is if we are doing in the other ensembles, we can pressure can be an output variable so we want to get the average pressure. In both these cases, we can get the temperature from the kinetic energy and, the way it comes out to be is the kinetic energy is defined as the sum over the kinetic energy of all the molecules and this is proportional to k_BT multiplied with the instantaneous temperature that we have defined in the earlier class-

$$KE = \frac{1}{2}\sum_i m_i v_i^2 = \frac{1}{2}N_{df} k_B T_I$$

but multiplied with some number of degree of freedom and this particular point is important that I do not put N there instead what I do is I define the number of degree of freedom as 3 times the number of molecules minus the number of constraints and minus 3 for removing centre of mass motion.

$$Number\ of\ degree\ of\ freedom, N_{df} = 3N - N_{constraints} - N_{COM}$$

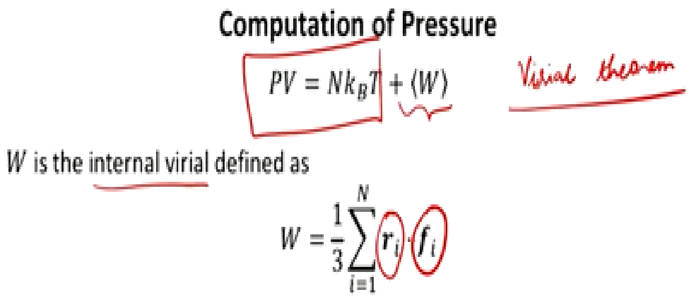
And there is a reason why we do that. So, if there was no constraints on the system then clearly 3 N represent the degree of freedom. So, every molecule can move in three possible directions x y z unless we are looking at rotation in that case, there will be additional degree of freedom, but we have three transnational degree of freedom for every molecule, having said that we need to have some constraints in some of the MD simulations let us say I want to constrain to the bond length to a particular value, let us say I want to constrain the bond angle to a particular value.

Those may be required in cases where the bond length is known to fluctuate very less or the bond vibrations are happening at time scales less than 1 femtosecond or δt that I am using. So, in those cases we have to incorporate the constraints when we compute the number of degree of freedom so we subtract the number of constraints.

Similarly, whenever we have a conservation of momentum in those cases, we have to make sure that the system overall does not have a translational motion. So, therefore we make sure that the centre of mass is not at all moving in the system or if it is moving we correct for the movement by bringing centre of mass back to its original position every so often in the simulation. So, in both these cases we therefore decrease also by 3 that is the 3 coordinates of the centre of mass and these degree of freedoms comes in the definition of the temperature.

So, we can get the instantaneous temperature using that and as we have discussed in the last lecture this is can be used in a thermostat for controlling temperature, but when we are working within say an NVE ensemble then this is the quantity we can average over to find the average temperature of the system.

(Refer Slide Time: 21:14)



Computation of Pressure

$$\boxed{PV = Nk_BT} + \langle W \rangle$$

Virial theorem

W is the internal virial defined as

$$W = \frac{1}{3}\sum_{i=1}^{N} r_i \cdot f_i$$

Similarly, we can find the pressure, for computation of pressure we use the idea of a virial theorem, which essentially says that the pressure basically contains an ideal gas-like contribution.

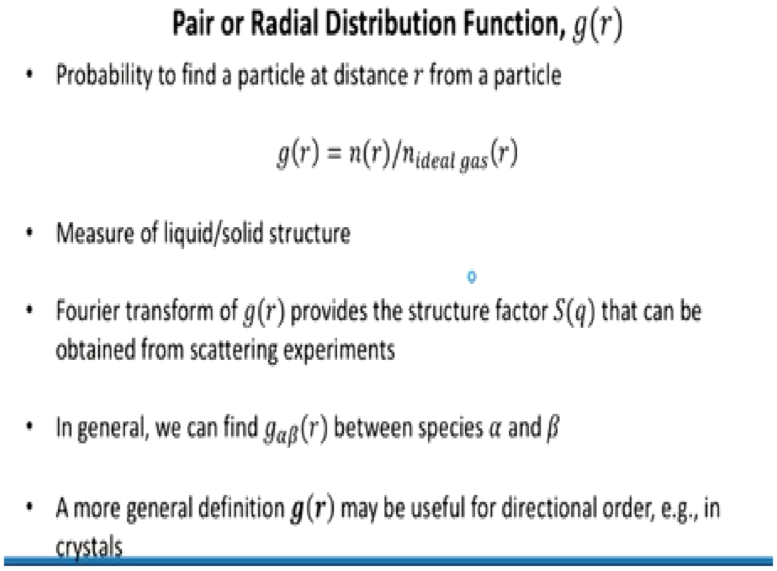
$$PV = Nk_BT + < W >$$

So, if I do not have the second term what we have is an ideal gas law that is the pressure that is present because of the molecular motion and addition to that there is some pressure coming because of inter particle forces and that particular quantity is called as w that is the internal virial that is because of the inter particle forces between the molecules and, it is defined as one third of the sum over the dot products of the position with the force acting on every particle.

$$W = \frac{1}{3}\sum_{i=1}^{N} r_i \cdot f_i$$

And all these information is available from us from the molecular dynamics because this is what we are solving for the solving for the positions the velocities and acceleration or the forces. So, we clearly have this information present in our trajectory and I can use that to find the pressure and, again I can use that to control the pressure when I am using a Barrostat or I can use that to find the average pressure when I am working in for example, the canonical ensemble.

(Refer Slide Time: 22:35)



Pair or Radial Distribution Function, $g(r)$

- Probability to find a particle at distance r from a particle

$$g(r) = n(r)/n_{ideal\ gas}(r)$$

- Measure of liquid/solid structure
- Fourier transform of $g(r)$ provides the structure factor $S(q)$ that can be obtained from scattering experiments
- In general, we can find $g_{\alpha\beta}(r)$ between species α and β
- A more general definition $\boldsymbol{g}(\boldsymbol{r})$ may be useful for directional order, e.g., in crystals

The next thing that I am interested in typically is something that characterizes the structure of the system when I say structure I am not referring to the molecular structure, but structure of the system at a larger length scale or a mesoscale structure or the state of the system and that can be found using a function known as the pair distribution function or the radial distribution function and, essentially it is defined as the probability to find a molecule at distance r from a reference molecule and we can also define it as the number of molecules present at the distance r from a reference molecule divided by the number of molecules that would have been present if the system would have been an ideal gas and both of this represents the probability.

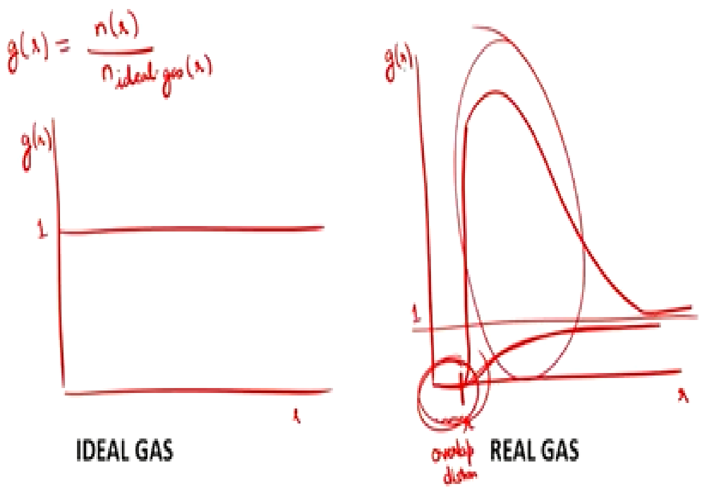
$$g(r) = \frac{n(r)}{n_{idealgas}(r)}$$

So, clearly if the system has strong interactions, you will have particles closer to each other and, in those cases we can imagine that g of r will take large values. On the other hand, in the case of an ideal gas the g of r would go to 1. So, essentially the magnitude of the g of r at a distance r demonstrates or contains the information regarding the order or the structure of the system. It turns out that if one Fourier transforms this g of r we get something known as structure factor that is typically obtained from the scattering experiments. So, whatever liquid or solid structure that we get from experiments, we can verify using molecular simulations or we can predict using molecular simulations by simply computing g of r and then doing a Fourier transform for that.

Now, g of r as it is was defined for a monoatomic or single component system. If, we have different species in the system we can define in general some $g_{\alpha\beta}$ between species α and species β. For example I look at species β at distance r from species α and we look at the probability of that and that will give me the g of αβ (r). In cases where we have some directional order we can think of an vector analog of this equation, so I replace the r with vector r or g with a vector or tensor g and that we also contain a directional order but in most cases the only work with the radial distribution function particularly in the case of liquid simulations.

(Refer Slide Time: 25:21)

Examples of $g(r)$

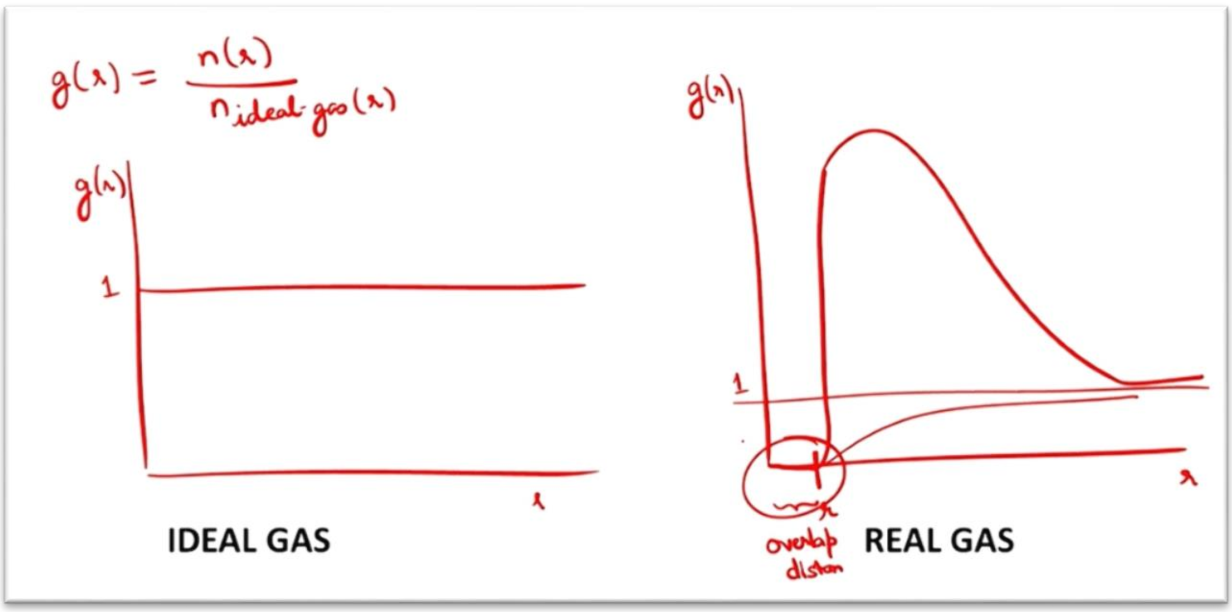


So, just to give you some examples of typical radial distribution function. So, since g of r by definition is-

$$g(r) = \frac{n(r)}{n_{idealgas}(r)}$$

this is going to be equal to 1 for all r values.

So, that means the probability of finding another molecule at distance r from any molecule is equal to a constant value. So, there is no particular preference or no particular affinity for the molecule to be at a distance from any reference molecule we are looking at. On the other hand, when we are dealing with a real gas it turns out that since, we are not allowing for an overlap there will be some small r values that would be completely not permitted because the overlaps are not permitted in the real gas and therefore we will have something like this where this is the, overlap distance over long distances, it should go back to 1 because our overlap distances there is no effect of the overlap but our short distances it should be close to 0. It may go like what I have drawn or it may go like something like this but the point being that at very short distances the probability has to be 0.

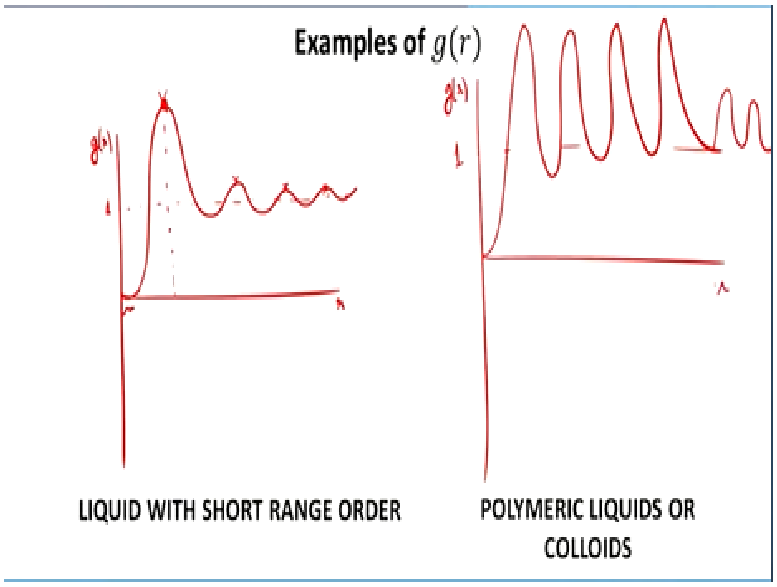


Now, the reason why I am drawing different shapes here is because the order also depends on the concentration. It turns out let us say if I think of a hard sphere system. The hard sphere system in dilute concentration would have a different structure as opposed to larger concentration; it turns out if I go to higher concentration in the hard sphere system, that is what I am using for simulating real gas I actually can form crystals and once I form crystals I can basically have large order but once that starts to happen then we are in the solid state, we are no longer in the gaseous state but before that, I first have to form some kind of liquid that will happen at certain concentration that will have more order than the gas but less order than the liquid or even before that we are somewhere between the gas and the liquid where we are trying to become ordered while still remaining a gas.

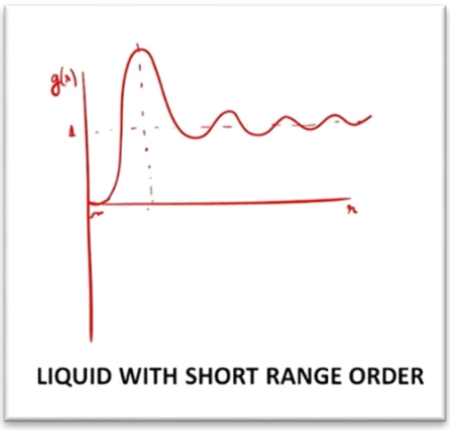
So, therefore the magnitude of the g of r, essentially contains information of how much order is present. It is not only black and white that every gas has to be disordered every liquid has to be more ordered and so on there is a whole spectrum of the development of the order as we go from a gaseous state to a liquid state and liquid state to a solid state and g of r essentially tries to capture that.

So, therefore I have tried to draw two different lines here in both these cases I am eliminating the possibility of the overlap but then in one case it is going to one in one particular way in the other case it is going to one in some different way and all this will depend on concentration of the hard sphere gas actually, it also depends on the temperature when I do it for different temperatures, you may have different amounts of order.

(Refer Slide Time: 29:39)



If I look at a liquid then it typically looks like this.



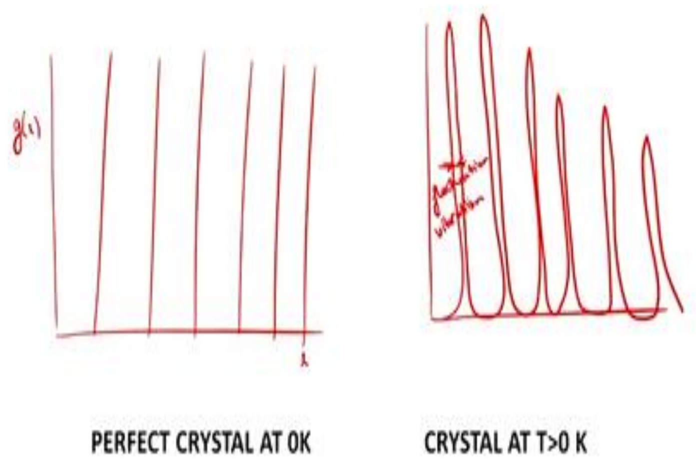
So, again, we have a reason of overlap where the molecules cannot really overlap but then there is a clear peak in the g of r that essentially it represents that there is a certain point in the g of r versus r diagram where the probability is relatively higher and the reason why this happens is because of the presence of the neighbouring molecules of the liquid that kind of forms the first neighbours of the molecules. So, although in liquid, it is not like crystalline or a very clear order but is still every molecule with surrounded by the other molecules by relatively weaker interactions and you will have first nearest neighbour second nearest neighbour and so on.

So, the average distance of the first neighbouring cell where the first nearest neighbours are located is what is represented by the first peak here. Now, you will have neighbours of those neighbours the first neighbours of those or the second neighbour of the reference particle that we started with that will give rise to another peak. Another peak and another peak and so on as we start looking at second neighbour third neighbourhood neighbour and so on. The reason why the peaks are decreasing in magnitude as r is increasing is because the interaction is short range the solvent - solvent interaction or liquid - liquid interaction is filled over shorter distances. So, typically the reference molecule only affects the presence of it is first neighbouring cell, it does not interact with second neighbouring cell and therefore the positions are relatively uncorrelated or less coordinated in comparison to the first neighbouring cell. If you think of polymeric liquids or collides, they are somewhere in between a liquid and a solid so therefore again they may have somewhat more order than compared than comparison to liquids.

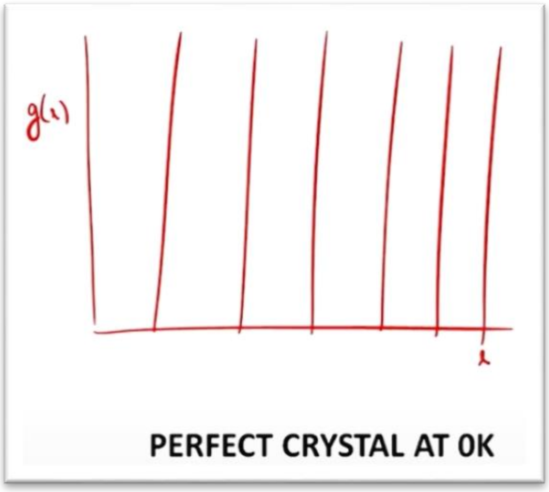
So, for example, they can have relatively larger peaks before they decay down and ultimately I am just doing some arbitrary cases here in reality there is a whole range of possibilities there but the key point is that the positions of the peaks of the g of r and the range of g of r over with the g of r is significantly different from what dictates how much order is present in the system because 1 is when it is an ideal gas like behaviour any departure from 1 is essentially showing you some presence of order in the system and that has to be looked at in more detail.

(Refer Slide Time: 33:02)

Examples of $g(r)$

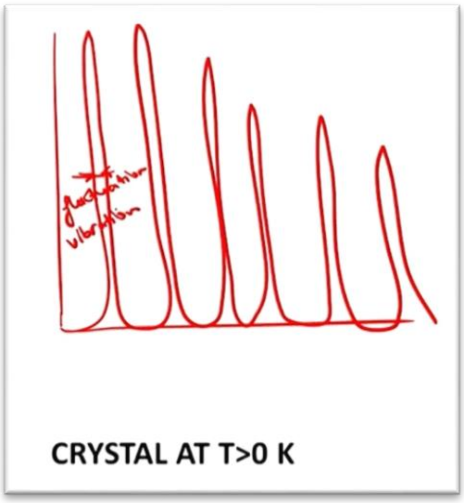


If you have a perfect crystal at zero Kelvin then what do we expect we expect that the molecules are not at all vibrating, so their positions are fixed and their neighbours are located at the fixed distance from it, because that is the lattice spacing in a crystalline structure if it is more than zero Kelvin then it is no longer true, because although the unit cell dimensions remains more or less same but the molecules themselves are moving a bit and, therefore the distance between the two molecules will not be constant but for a perfect crystal it has to be the case because there is no vibration taking place, so for that case the molecules can only be present at certain distances from a particle and that corresponds to the lattice spacing.



So, the first neighbour may be somewhere here second may be here third may be here and so on. So, there are discrete r locations which are highly probable and then there is no probability in between because the molecules are not allowed to vibrate.

For crystals at temperatures more than zero Kelvin this looks pretty similar to that except that we have to account for vibrations, so we still have peaks but the peaks have certain small width that account for the fluctuations of the particles, fluctuations or vibrations of the particles in the crystal.

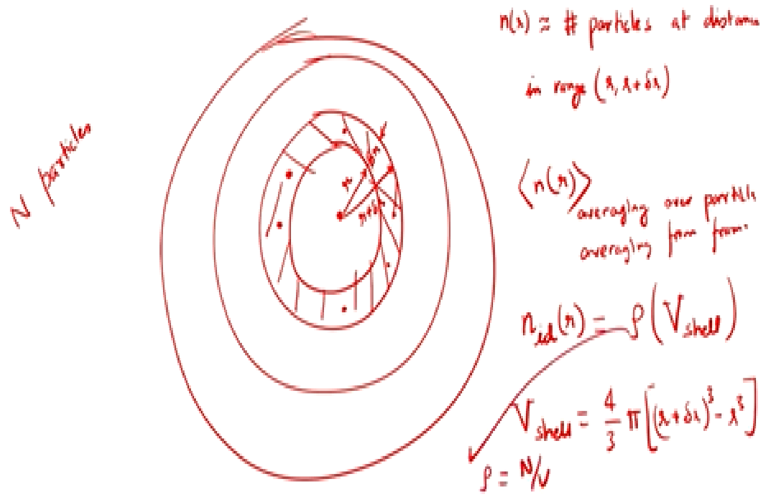


So, they look pretty similar to what crystal except that the peaks are broader and as you may imagine the peaks become broader and broader we are going from a solid state to a liquid like state because you may have noticed here the peaks here are pretty broad both in these cases and the in the example of gas the g of r was even broader. So, if when we look at the g of r, we look at all these features and try to infer the order of the structure from these features present in the g of r and, if we have experimental data we can also Fourier transform it get structure factor and compare that to experiments and that is the beauty of g of r and that is something that is only obtainable from molecular simulations.

It turns out that although I can get the structure factor from g of r by doing the Fourier transform inverse of that is not theoretically possible it is difficult for impossible to do in most of the cases. So, therefore it only comes from the molecular simulations it is not obtainable from experiments or other means and therefore it is really useful to get the liquid structure or solid structure information from molecular simulations that can be easily obtained from the trajectories that we have and, the way it is done actually on the algorithm point of view is the following.

(Refer Slide Time: 36:28)

Computing $g(r) = n(r)/n_{ideal\ gas}(r)$ from simulation trajectory


N particles
n(r) = # particles at distance in range (r, r+δr)
⟨n(r)⟩ averaging over particles averaging from frames
n_id(r) = ρ(V_shell)
V_shell = 4/3 π[(r+δr)^3 - r^3]
ρ = N/V

So, let us say, for example, if I am looking at any reference atom in the system we can imagine that there will be small cells around it and this construction is purely imaginary. So, between distance r and r + δr, we can imagine a spherical cell in three dimensions and we can count the number of particles that lie within the spherical cell.

So, if I want to find the g of r, I look at the number of particles at distance in range r and r + δr clearly as we decrease the δr, this definition becomes more and more rigorous, at the same time as we decrease the δr, you will have fewer and fewer particles present in that particular spherical cell. So, whenever I am evidencing over many trajectories, it is going to give me better answers or better averages when I am using δr good enough so, that there is reasonably large number of particles present in there because it is that quantity that we are averaging for.

So, basically using multiple, multiple points in the simulations we can basically get an average of the n of r, we can average over the frames and we can also average over the particles. So, we can think of let us say if you have n particles in the simulation then I can imagine cells like this around each of the particle and therefore I can get the number of pairs essentially which are at distance in the range r 2, r + δr and then we have to divide by the number of particles and all that.

So, essentially we are averaging over the particles in the system and we are also averaging over frames. This is like essentially two kinds of averaging we are employing and the reason why we are doing the first one is because we are not interested in any particular particle in the system we are interested in actually knowing that what is the probability of finding a particle at distance r from another particle, we are not labelling the particle in any particular way and therefore it makes sense that I average over the particles and since molecular dynamics ultimately gives you averages it is better to use as many ways of averaging as possible if you have many particles you have more quantities to evidence for and therefore you will have a smaller fluctuations and the final result.

At the same time, we can also average over the frames in the MD simulations or if we are doing multiple simulations we can also average over different simulations. Now, once we have done that then we can also compute the number of molecules in the case of the ideal gas and that can be easily done I simply multiply the density in the case of an ideal gas with the volume of the spherical cell and, the volume of cell is something like-

$$V_{shell} = \frac{4}{3}\pi[(r + \delta r)^3 - r^3]$$

and the density in the case of the ideal gas is simply-

$$\rho = \frac{N}{V}$$

because in the ideal gas case the density everywhere is uniform so throughout the box we have a constant density, that is N by V and I am defining the number density here. So, we already have the average n of r and we have an id r, so therefore we can find-

$$g(r) = \frac{< n(r) >}{n_{id}(r)}$$

So, this procedure when I am looking over this smaller cells of thickness δr is called basically a binning procedure and each of these cells basically corresponds to bins essentially what we are creating is an histogram of how many particles are present in different bins of the system and by using the right bin size we can make sure that the averages are proper and we can get the correct representation of g of r.

In many cases what you will figure out is that if you are not doing everything properly then g of r exhibit large fluctuations. And, when that happens if I repeat the simulation you will get a different answer and that is not really because something was went wrong in the simulation it is simply because the averaging procedure was poor.

So, we have to really ensure that we are using correct averaging in the case of the molecular dynamics and in this case the correct addressing can basically refers to the correct choice of δr and averaging over all the particles and averaging over all the frames in the simulations and I say all the frames it is clearly all the frames sampled at the sampling frequency τ sample because anyway, we do not want all the all the correlated frames in the simulation.

So, with this I want to conclude this lecture. In the next lecture, we will discuss some more properties that we can evaluate from MD simulations and after that, I will discuss some case studies of molecular dynamics taken from our research, thank you.