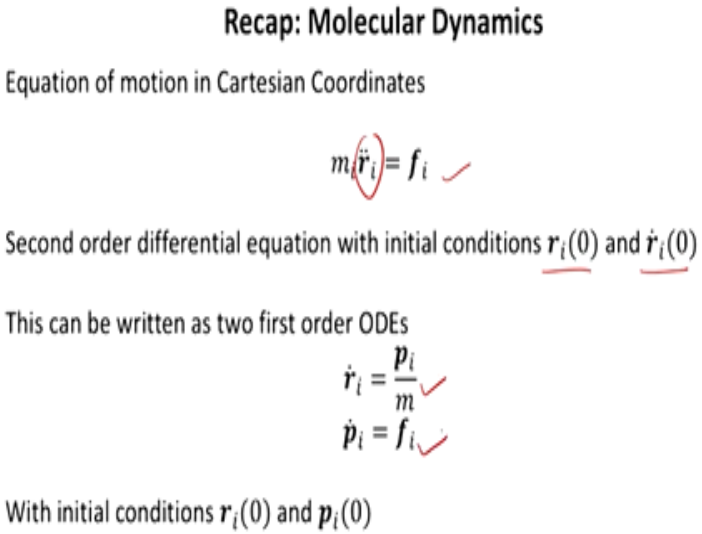
Advanced Thermodynamics and Molecular Simulations
Prof. Prateek Kumar Jha
Department of Chemical Engineering
Indian Institute of Technology, Roorkee

Lecture - 46
Numerical Implementation of MD; Thermostat and Barostat

Hello all of you. So, in the last couple of lectures we have been discussing the molecular dynamics or MD scheme. Before that we have looked at the particle simulations and the Monte Carlo methods. So, in this lecture we will start talking about the numerical implementation of the molecular dynamics scheme.

(Refer Slide Time: 00:44)



So, as we have set out in the last lecture. We can write the equation of motion in Cartesian coordinates for molecular dynamics as the following. So, where r_i double dot is the acceleration the second derivative of position of particles with respect to time.

$$m_i\ddot{r}_i = f_i$$

And this equation itself is a second order differential equation. So, we need two initial conditions for the position and the velocity at time t equal to 0. And we discussed that we can write it as two first order ordinary differential equations.

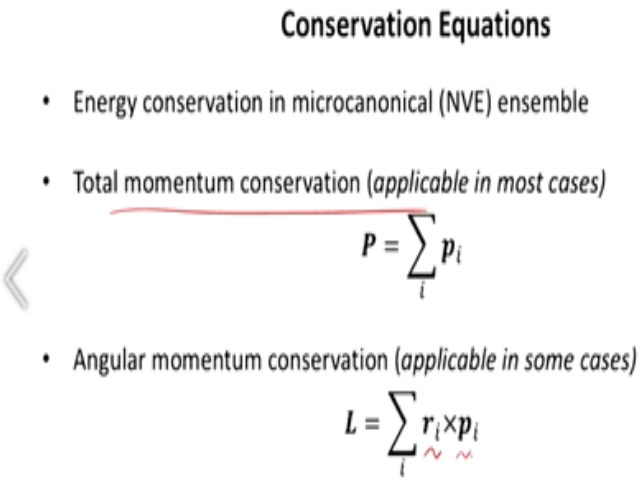
$$\dot{r}_i = \frac{p_i}{m}$$

$$\dot{p}_i = f_i$$

One for the definition of momenta and other for the Newton's laws of motion essentially a representation of the equation of motion in terms of first order ordinary differential equation.

In addition to this, in most cases we also have some conservation equations that has to be valid. For example, when we are working in the macro canonical or NVE ensemble the energy of the system has to be conserved.

(Refer Slide Time: 01:44)

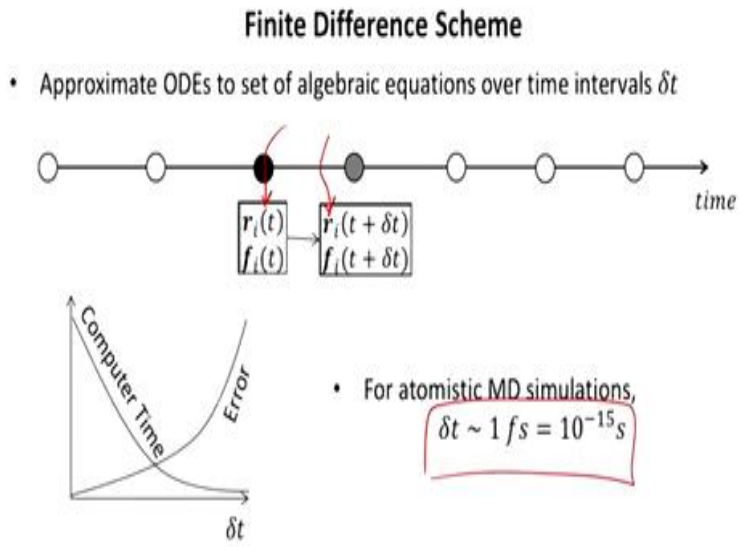


Similarly, in many cases we may have some total momentum conservation that is if I sum over the momenta of all the particles in system that has to be constant. In certain cases, we also have conservation of angular momentum that is defined as the position dot crossed with the momentum of particle again summed over all the particles. It is not true in every MD case but in certain cases this also has to be there.

So, these conservation equations should be satisfied along with the equation of motion and we should make sure that whatever algorithm we are building is satisfying the conservation equation. It turns out that it is not always valid for the numerical algorithms. And in those cases, it is really important that our departure from the conservation equation should be somehow corrected for or accounted for in how we analyze the results or how we implement the algorithm.

So, this is basically the scheme that we use in molecular dynamics. It is preferred to use a simpler finite difference scheme because this really has to be something, that has to be robust over I would say computation over long times. So, any particular scheme containing many steps and all that is typically may be giving rise to some more accuracy but it is not robust enough and simpler schemes are always preferred. Having said that it is preferred that we should do it for the time steps as small as possible, because as we increase the time step the error in the scheme or the difference from the accurate integration if we could do the integration increases and therefore, we should ideally do it for δt going to 0. It turns out that as we decrease the time step in order to simulate the same time, we require to, perform more MD steps and therefore, the computer time also increases as we decrease the time step and it is always a compromise therefore of what time step we can use.

(Refer Slide Time: 04:03)



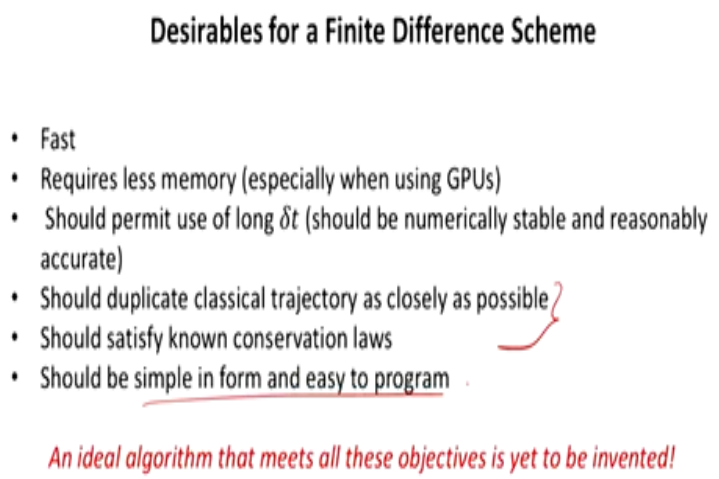
It turns out that for atomistic MD simulations the typically used time steps are of the order of 1 femtosecond that is 10 to the power minus 15 seconds.

$$\delta t \sim 1\ fs = 10^{-15} seconds$$

Any steps higher than that result in typically numerical errors which cannot be ignored or some instability in the computational algorithm. So, having said that the; basic idea remains the same as with any finite difference scheme that is, if I know the position and velocity at time t, I can basically discretize the two first order differential equations. And use that to get the position and velocity at time t + δt. This is essentially what we are trying to achieve from finite difference scheme.

$$r_i(t) \rightarrow r_i(t + \delta t)$$
$$f_i(t) \rightarrow f_i(t + \delta t)$$

(Refer Slide Time: 04:59)



And what is desired from the scheme I have already said it has to be robust. But more importantly it has to be fast. It should require less memory that is typically not a consideration in most CPU machines. But nowadays it has become important or it has become I would say possible or feasible to do computation using GPUs which are much cheaper alternative to CPUs but there is more memory limitation. So, in order to implement a finite difference scheme that can harness the power of GPUs, we have to make algorithm that is requiring less memory that is true for GPUs. For CPUs it is typically not a big deal but nonetheless we should prefer algorithms that require less memory. It should permit the use of longer time steps that is, if I use longer time step. First of all, the integration should be numerically stable we should not get any divergence or any kind of instability in the simulation.

Having said that it also has to be reasonably accurate and these are both these things are something that we must keep in mind in order to choose the time step of our simulations. And finally, we should be able to duplicate the classical trajectory as closely as possible. And also, our algorithm should be able to satisfy the conservation laws and these two things are actually the most critical part that determines whether algorithm is good algorithm or bad algorithm.

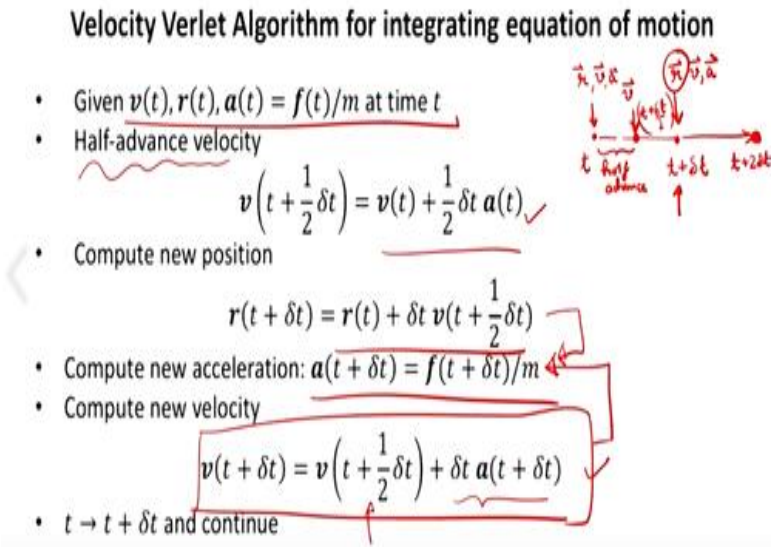
If there is large violation from the conservation laws, if there are significant departures from the trajectory in those cases the algorithm is definitely not a good algorithm. And finally, it has

to be simple in form and ease to program that is again not a big concern now a days because mostly, we use software's for MD calculations but having said that for programmers or developers at least this is also an important point to keep in mind.

So, it turns out that an ideal algorithm that meets all these objectives to sufficient precision is yet to be invented. And we always make some compromise one way or the other depending on the application that we have in mind or the machine we are working with we really have to choose an algorithm that is best out of the possibilities. And there is no one algorithm I can say that can meet all the objectives in all the possible application or all the possible architectures.

So, one of the most common schemes that is implemented in most software's is the velocity verlet scheme for integrating equation of motion. It is a finite difference scheme. And the way it works is like. Let us say for example, if I look at time t and I know the position, the velocity and acceleration that is I know the force at time t because if I know the force, I can find the acceleration of all the particles. Then for every particle I can go ahead and find the position and velocity at time t + δt by doing the following.

(Refer Slide Time: 07:52)



So, first what we do is we half advance the velocity that means if I am having the initial time point as t and the new time point as t + δt. So, the objective is that I know the position, velocity, acceleration here and I want to find the position, velocity here. Then in this case what I first do is I pick the midpoint of the interval. And compute the velocity at the midpoint this is what we mean by half advance. And this clearly is given as the velocity at time t + half δt multiplied with the acceleration this has to be the case.

$$Half\ advance\ velocity = v\left(t + \frac{1}{2}\delta t\right) = v(t) + \frac{1}{2}\delta t\ a(t)$$

So, for this interval essentially between t 2 t + δt by 2, I am assuming the acceleration to be constant and equal to the acceleration at time t. And then I use the simple equation of rectilinear motion and we get the expression of the velocity at time t + half δt.

And using this velocity that is the velocity midway between t and t + δt, I now find the position at t + δt we could have used the velocity at time t in place of velocity at time t + δt that is the standard Euler scheme we could have used that. It turns out that the procedure of half advancing velocity and using the velocity at midpoint instead provides more accuracy and stability to the algorithm.

So, then we simply compute the new position as the old plus delta t multiplied with velocity at the midpoint.

$$r(t + \delta t) = r(t) + \delta t(v + \frac{1}{2}\delta t)$$

And then I find the new acceleration at time t + δt. And finally, I compute the velocity at time t + δt as the velocity at t + half δt + δt multiplied by acceleration at time t + δt.

$$v(t + \delta t) = v\left(t + \frac{1}{2}\delta t\right) + \delta t\ a(t + \delta t)$$

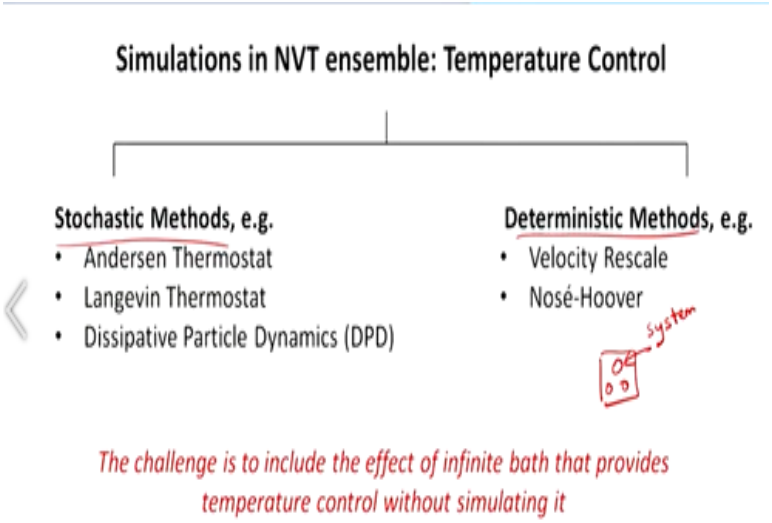
So, in the very beginning we are kind of doing some sort of an explicit scheme and towards the end we are doing some sort of an implicit scheme that means that in the beginning in the first part of this the new velocity depends on only the previous velocity and acceleration but towards the end the new velocity depends on also the new acceleration. So, in some sense it is an implicit scheme. But in reality, we have already computed the acceleration using the force value at time t + δt and I can find the forces because we have already found the position.

So, it really works as a nice trick of finding the velocity at midpoint and using that to first update the position then update the acceleration and finally update the velocity. And it turns out that it really works very well for most of the cases and then once we have done that then we have found the position and velocity and also acceleration at the new time.

Now I set that equal to my old time and use that to find the position, velocity and acceleration at the following time step and that would be at t + 2 δt and we can keep on doing it until we have done the simulation for the time over which we want to simulate the system.

So, the next thing in an MD simulation is in sometimes we have to control temperature. So, what we have discussed so far was assuming that we are doing simulation in the NVE or micro canonical ensemble.

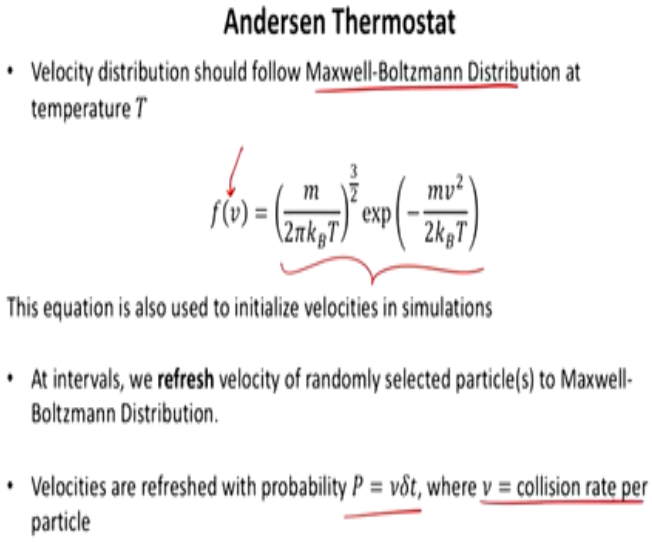
(Refer Slide Time: 13:03)



If we want to simulate in the canonical or NVT ensemble, then we have to have some mechanism to control temperature that means we have to have some mechanism to simulate the presence of a bath that can control the temperature of the system and if you recall we are not really simulating bath in our molecular dynamics we are only simulating the system although in the thermodynamic ensemble picture we had discussed the system was in infinite bath and I am looking at many systems in that infinite bath, when I am doing an MD simulation I am only simulating the system not the bath and so there has to be some mechanism essentially to simulate the effect of an infinite bath if I want to control the temperature.

And there are two clause of methods to achieve that, one is the stochastic method and second the Deterministic method. In the Stochastic methods comes methods like Andersen Thermostat, Langevin Thermostat and Dissipative Particle Dynamics there are some standard methods that are being used. On the deterministic method there are methods like Velocity Rescale or Nose-Hoover method and there are others as well.

(Refer Slide Time: 14:04)



So, just to give you an example of the stochastic method. In the Andersen Thermostat what we essentially do is we use the fact that velocity distribution at equilibrium at temperature t should essentially follow the Maxwell Boltzmann distribution that is well established in thermodynamics and that goes like the following.

$$f(v) = \left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} \exp\left(-\frac{mv^2}{2k_B T}\right)$$

So, the probability density to have a velocity of magnitude v is given by this particular function and in fact, this function is what we use to generate the initial momenta of the particles in the simulation. So, if I want to simulate a temperature t, I first generate an initial distribution using the Maxwell Boltzmann distribution.

Now what is going to happen is as the system evolves the distribution will start to depart from the Maxwell Boltzmann distribution because of the collisions between the molecules and so, in order to control the temperature then what we can then do is we try to basically refresh the velocity in a manner that basically goes back to the Maxwell Boltzmann distribution at that temperature.

So, essentially, we start with a Maxwell Boltzmann distribution and then as the system evolves and we start to depart from there that means our temperatures depart from the temperature I want to put I will basically try to refresh the velocity of particles so that we can recover the Maxwell-Boltzmann distribution and the way we do that is we do not refresh velocity of all the particles we randomly select some particles with certain probability every so often with certain

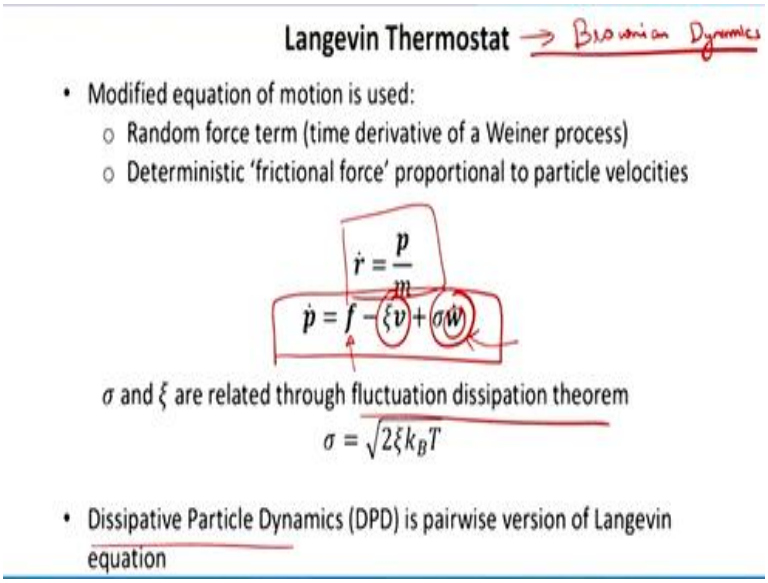
time intervals. And then we refresh their velocity that is we reset their velocities in order to get to the Maxwell Boltzmann distribution and this can be done with a probability of some $\nu \delta t$. For example, where; ν is the collision rate per particle.

So, the more refreshing we are doing the easier would be to control temperature. But then farther we will depart from the true dynamics because every time we do this kind of refreshing we are also not quite stimulating the true dynamics because true dynamics was only there in the NVE ensemble, when there was no external interference with the motion of the particles. Now since we are randomly choosing particles and providing them velocities this is clearly not following the true dynamics in a sense it simulates the effect of bath but it does not quite capture the true dynamics as in the NVE ensemble. We can only hope that the classical trajectory for the NVT ensemble case would be reproduced at least to some approximation, when we are doing the NVT ensemble simulation.

Having said that; in many applications the classical trajectory is not of our interest we want to use the MD as a method to sample points in the phase space. So, as long as the motivation is not to look in detail how we are going from one state to the other. We can still use MD to sample points on the phase space and that part of it is unaffected by our application of thermostat.

The next method that is also a Stochastic Langevin Thermostat. This is also used in the non-equilibrium simulations like Brownian dynamics but since we are focusing on the equilibrium behavior now let us keep the discussion of Brownian dynamics later.

(Refer Slide Time: 17:49)



So, in this case what we do is we slightly modify the equation of motion itself. So, we use the definition of momenta as it is, it is mass times velocity. But the Newton's law of motion we represent by a modified equation where in addition to the force term that was always there, we also add two extra terms.

$$\dot{r} = \frac{p}{m}$$

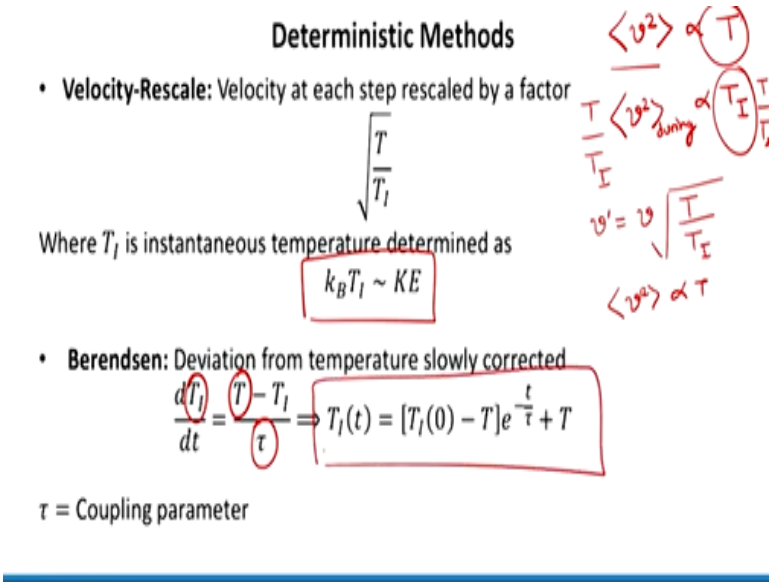
$$\dot{p} = f - \zeta v + \sigma \dot{w}$$

And the first term kind of represents some sort of a frictional force or a drag force that is acting on the particle and the last term represents some kind of thermal motion that is introduced by using a random force. And both of these are essentially simulating the effect of bath. It turns out that we can relate sigma with zeta using a theorem called the fluctuation dissipation theorem. w here is found using what is called a Weiner process in random number theory and using this particular idea we can also basically establish that the temperature will remain controlled at temperature we want to do. So, we want to simulate the effect of temperature or the effect of bath by addition of these two terms in our equation of motion.

There is another method called dissipative particle dynamics that is essentially same as the Langevin Thermostat. Except that we represent the Langevin equation in terms of the pair wise forces. So, the pair potentials can be written such as you will have additional terms because of the frictional force and additional term because of thermal part that is the sigma W dot part in this particular equation.

So, next comes, the deterministic methods of temperature control. So, one of the; I would say most intuitive method is to basically rescale velocity at every step by some kind of a factor and the idea to do that is the following.

(Refer Slide Time: 20:12)



So, let us say if I start the simulation with the Maxwell Boltzmann distribution and we ensure that the v^2 is proportional to temperature. As the simulation proceeds, we will start to deviate from that particular temperature. And what this would mean is v^2 will not remain proportional to this temperature, the v^2 that is during simulation proportional to some other temperature and we give it a name instantaneous temperature. That is to say that we can define a quantity called instantaneous temperature that represents basically the mean square velocity. In other words, we can use the equation like this where the kinetic energy of particle can be written as something like Boltzmann constant multiplied with the instantaneous temperature.

$$k_B T \sim KE$$

And since we have a way to figure out the new temperature, we basically rescale the velocities such that I basically multiply this both sides with T by T_I and the new velocity therefore would be something like the velocity multiplied with square root of T by T_I. And this would ensure that the mean square of the new velocity will again go like proportional to T.

$$v' = v\sqrt{\frac{T}{T_I}}$$

$$< v^2 > \propto T$$

And this will pretty much ensure that we are going back to the temperature that we wanted to work with.

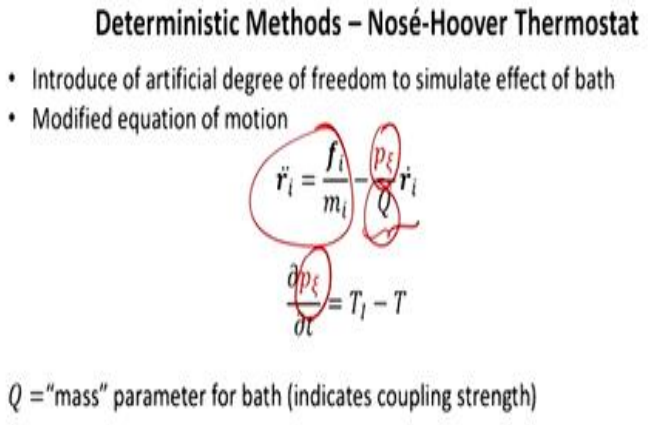
The next method is the Berendsen method wherein what we do is? We again use the idea of instantaneous temperature and the TI and T as my target temperature or the; controlling temperature. But now what I do is basically, I correct for the deviation from T rather slowly using some kind of a exponential method. In velocity rescale method it is quite drastic change because we rescale velocity of all the particles. In this case we do somewhat slower change and we keep a parameter τ that essentially controls how slow we are going to do the change.

$$\frac{dT_I}{dt} = \frac{T - T_I}{\tau} => T_I(t) = [T_I(0) - T]e^{-\frac{t}{\tau}} + T$$

If τ is larger then, we are correcting for the deviation slowly it is called what is known as weak coupling. On the other hand, if τ is small, we are correcting for the temperature deviation very quickly and in that case, we call it a strong coupling.

And in both these cases we basically have an exponential decay of TI to the target temperature T. And τ is essentially a time constant that determines how much time it roughly takes to get there actually the first order time constant to be precise.

(Refer Slide Time: 23:42)



Then finally there is a Nose-Hoover Thermostat. So, in this case we what we do is we modify the equation of motion and actually introduce a new degree of freedom that simulates the presence of bath. So, in this case you already have the first term of the equation in the previous

slides as well but now I include a new term where I have some variable to represent the momenta of some arbitrary bath variable.

$$\ddot{r}_i = \frac{f_i}{m_i} - \frac{p\zeta}{Q}\dot{r}_i$$

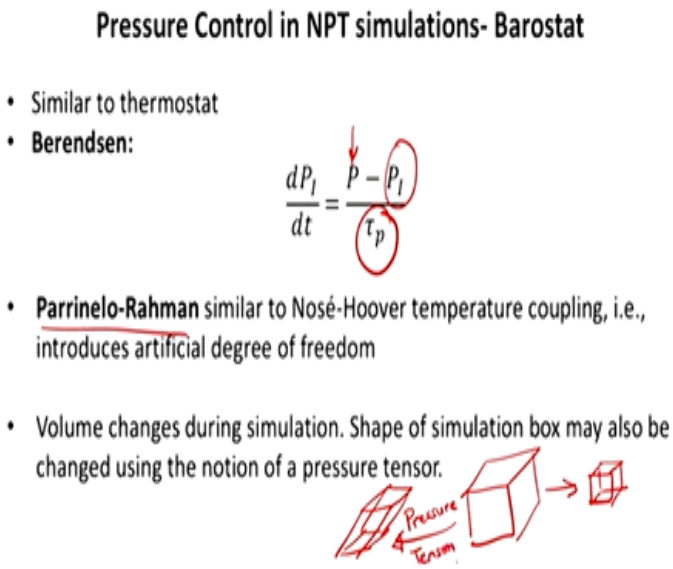
$$\frac{\partial p\zeta}{\partial t} = T_I - T$$

And for that variable, I write another equation that gives me how it changes with time it does not have a physical meaning as such except that using doing this helps to control temperature to a target value and just like tau in the previous case we have a parameter called Q in this case that is the called the mass parameter of bath. Because it has a unit of mass that again, does some kind of coupling with the bath by variation of Q, I can go from a strong coupling regime that means we can control temperature pretty quickly to a weak coupling regime where we can control temperature rather slowly.

So, similar to the temperature control we also will need to control pressure if we work in the NPT ensemble and just like we have used thermostat for temperature control we use something called a barostat for pressure control. And the idea remains pretty similar to thermostat just like the Berendsen Thermostat we can also use a Berendsen Barostat.

$$\frac{dP_I}{dt} = \frac{P - P_I}{\tau_p}$$

(Refer Slide Time: 25:07)



We can define an instantaneous pressure that represents the pressure at a point in the simulation and P is the target pressure or the controlling pressure where I want to go to. And τP again is some sort of a coupling parameter. So, for a small τP we will have a strong coupling and for large tau P we will have a weak coupling because it will take longer for the pressure to come back to the target value.

And similar to the Nose-Hoover temperature coupling where we introduced an artificial variable for the bath. There is a Parrinelo-Rahman method that introduces an artificial degree of freedom to account for the effect of the bath.

In that case the bath is kind of applying the pressure on the system or controlling the pressure of the system just like the bath was controlling the temperature of the system in the case of thermostat and since we are doing in the NVT ensemble the volume will change in the simulation the simulation box volume will has to be changed and while doing that the box may not even remain cubic all the time. So, you can think of like two ways of deforming it.

It can be isotropic where it remains cubic or it can be an anisotropic change where the box may also change in shape and both of these are possible in most software's depending on the application we can go for the isotropic pressure or we can have anisotropic deformation as well. In the anisotropic case what we make use of something called a pressure tensor, where essentially the pressure or the stress may vary along different faces of the simulation box.

So, I told you about the strong coupling and the weak coupling so, which one to use? So, it turns out that a strong coupling is typically not preferred in the equilibration phase and the reason is that when we are using the strong coupling any deviation from the velocity that takes the temperature away from the controlling temperature is suddenly controlled by the thermostat and same is true for the volume change deformation. So, it turns out that it is not really helping in the equilibrium of the system because the system will equilibrate only by relaxing to different configuration and typically, the relaxation process involves sufficient deformation or sufficient velocity changes and the strong coupling kind of delays that equilibration process.
On the other hand, when I am doing a production run then strong coupling will give you lower fluctuations in the properties and therefore it is more preferred in the production run.

On the other hand, weak coupling provides relatively poor control but it takes lesser time to equilibrate because the system is evolving naturally kind of there is lesser interference by the Thermostat or the Barostat. So, in most cases it is preferred that we start with weak coupling relax the system to the temperature that we are interested in get near the equilibrium state. And then finally use the strong coupling in some cases, it is even better to use strong coupling from the beginning itself if the equilibration time is relatively small but in other cases where equilibration is an issue it is a good idea to start with weak coupling and then once the equilibration is over then switch to the strong coupling, because you will have large fluctuations in the case of weak coupling that is not good for the prediction of properties, for prediction properties in production phase, we tend to prefer the strong coupling.

So, with that I want to conclude today's lecture what we have discussed is the numerical implementation of the molecular dynamics scheme. We have discussed the finite difference scheme especially the velocity verlet scheme. And then we discussed how can we control temperature and pressure in a molecular-dynamic simulation.

In the next class I will discuss how exactly we can parallelize the molecular dynamics and what exactly we can compute out of the MD simulations, thank you.