Computational Chemistry & Classical Molecular Dynamics

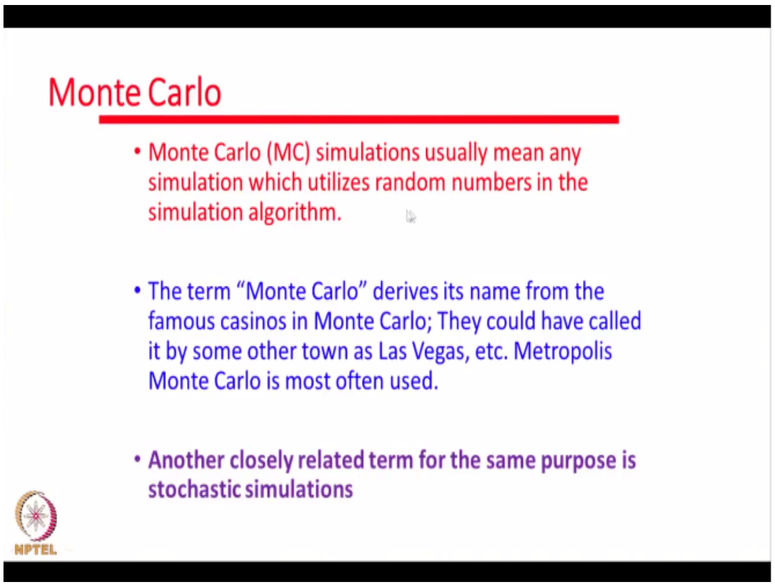
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology – Bombay

Lecture - 32
Classical MD-5, Execution of Programs on Liquid Argon

Hello and welcome to this lecture on Monte Carlo simulations. In the last lecture, we considered several details of molecular dynamics, not only we discussed the algorithm; we also defined what are pair correlation functions that is the radial distribution function as well as the time correlation function. So this time, we will do what are called Monte Carlo simulations.

And Monte Carlo simulation really mean simulations which use lot of random numbers in the simulation algorithm.

(Refer Slide Time: 00:46)



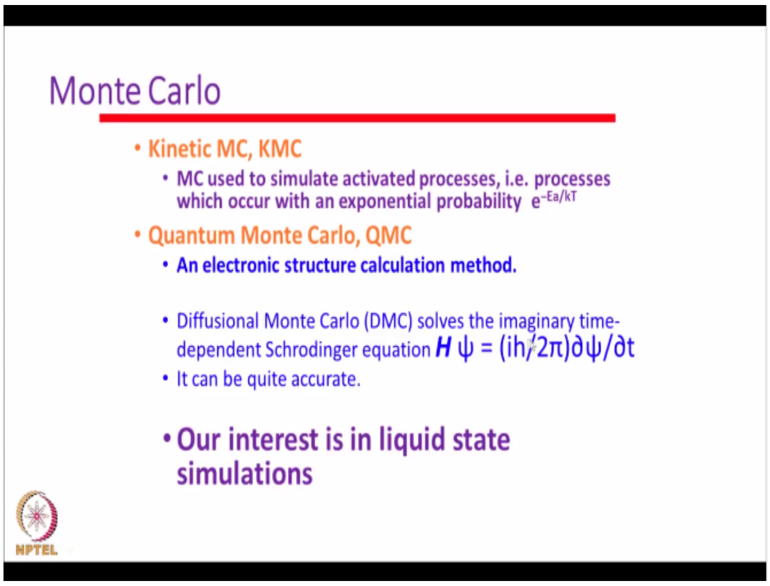
Monte Carlo

- Monte Carlo (MC) simulations usually mean any simulation which utilizes random numbers in the simulation algorithm.
- The term "Monte Carlo" derives its name from the famous casinos in Monte Carlo; They could have called it by some other town as Las Vegas, etc. Metropolis Monte Carlo is most often used.
- Another closely related term for the same purpose is stochastic simulations

So they will not use equations of motion. They will just use random numbers to move all the particles based on an algorithm and the term Monte Carlo came because of the casinos in Monte Carlo but they could have also been called as Las Vegas or any other name. So the correct name would be Metropolis Monte Carlo because it was Metropolis and others who gave that algorithm.

So another name for Monte Carlo simulation could also be stochastic simulations. Stochastic means any process that uses random numbers. So what are these simulations now or what are the uses? There are many, many types of Monte Carlos okay.

(Refer Slide Time: 01:27)



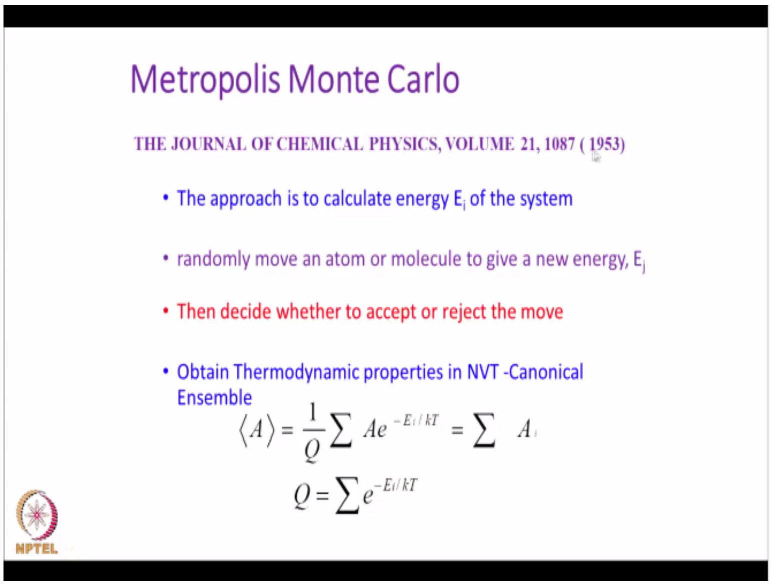
Monte Carlo

- Kinetic MC, KMC
  - MC used to simulate activated processes, i.e. processes which occur with an exponential probability $e^{-Ea/kT}$
- Quantum Monte Carlo, QMC
  - An electronic structure calculation method.
  - Diffusional Monte Carlo (DMC) solves the imaginary time-dependent Schrodinger equation $H\,\psi = (ih/2\pi)\partial\psi/\partial t$
  - It can be quite accurate.
- Our interest is in liquid state simulations

So this kinetic Monte Carlo can be used to simulate an activated process okay or a process which occurs with an exponential probability. An activated process occurs with a probability of e to the –energy/kT. So this is an activated process. So there is also what is quantum Monte Carlo that can be used to calculate electronic structure okay. Now there is also what is called diffusional Monte Carlo. It solves this time dependent Schrodinger equation.

But we are not interested in all this (FL). So our interest is in liquid state simulations. So what we do, we will just consider classical liquids and how to simulate the structure of that liquid. So Monte Carlo simulation uses random numbers, so therefore they will not tell me anything which is a function of time. So they will only give me the structure or the equilibrium distribution function okay.

(Refer Slide Time: 02:24)



Metropolis Monte Carlo

THE JOURNAL OF CHEMICAL PHYSICS, VOLUME 21, 1087 (1953)

- The approach is to calculate energy E_i of the system
- randomly move an atom or molecule to give a new energy, E_j
- Then decide whether to accept or reject the move
- Obtain Thermodynamic properties in NVT -Canonical Ensemble

$$\langle A \rangle = \frac{1}{Q}\sum Ae^{-E_i/kT} = \sum A_i$$

$$Q = \sum e^{-E_i/kT}$$

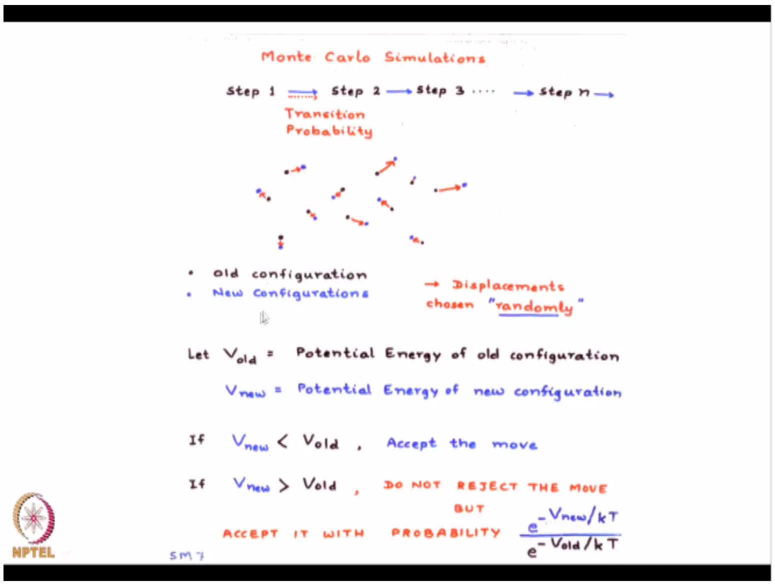
So let us see what is the strategy here. So this particular Metropolis Monte Carlo, you can see that paper in journal of Chemical Physics, volume 21, 1087, 1953, a very remarkable paper okay. So their aim was to calculate the average energy of the system by moving all particles randomly okay. The strategy is take a set of configurations, move particles randomly and see whether after you move the particle, the new move is acceptable or not so okay.

So once you find out whether the new move is acceptable or not, so for each step you will have physical quantities like Helmholtz free energy or internal energy or every physical quantity will have a value at each step. When you average over the entire configuration and divide by the number of steps, you will get the average value for that okay. So this Metropolis Monte Carlo follows the NVT ensemble.

That is the number of particles is fixed, volume is fixed and the entire calculation is done at a fixed temperature okay. So the Monte Carlo simulations correspond to NVT calculations whereas the molecular dynamics correspond to NVE. That is a different ensemble because normally temperature should not be fixed in a molecular dynamics, so that corresponds an NVE ensemble.

So this is my canonical ensemble, averages can be calculated at the end of the simulation by just summing over individual steps.

(Refer Slide Time: 03:59)


Monte Carlo Simulations

Step 1 $\longrightarrow$ Step 2 $\longrightarrow$ Step 3 .... $\longrightarrow$ Step n $\longrightarrow$

Transition Probability

- old configuration
- New configurations

$\rightarrow$ Displacements chosen "randomly"

Let V_{old} = Potential Energy of old configuration

V_{new} = Potential Energy of new configuration

If $V_{new} < V_{old}$, Accept the move

If $V_{new} > V_{old}$, DO NOT REJECT THE MOVE BUT

ACCEPT IT WITH PROBABILITY $\dfrac{e^{-V_{new}/kT}}{e^{-V_{old}/kT}}$

SM 7

So now let us see the logic of the Monte Carlo simulation. So I start with step 1, go to step 2, step 3, step n, do this entire simulation. To go from step 1 to step 2, I need I use what is called a transition probability okay. So this is essentially a stochastic algorithm. The essential feature in this stochastic algorithm is the transition probability from step 1 to step 2. So how do I calculate the transition probability?

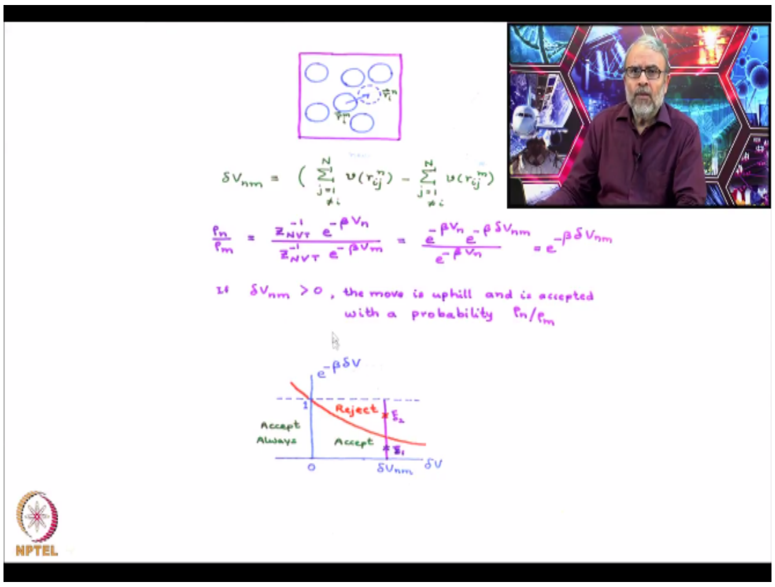
So this is the strategy, you have an old configuration which is determined by this blue dots and each of the particle I will move a little bit. I will move a little bit in different direction by random moves. How do I generate these random moves? Generate random numbers in the x and y and z directions and move that a little bit. So you can either move all particles at a time or you can move only one particle at a time.

So once I move my particles, I have got a new configuration. So V old is the potential energy of the old configuration. V new is the potential energy of the new configuration. That is V old is of this original dots and V new is the all the new positions. So when I have two energies now, so what is the strategy? If the new potential energy is<the old potential energy accept that move okay.

So whenever I have a new energy<the old that is an acceptable move. Now if the new potential energy is>the old potential energy, do not reject that move okay. If you reject that move, that will not be correct but accept that move with the given probability. So how do you calculate whether to accept the new one or not okay. Accept the new particle, you take the ratio, new potential energy/kT e to the –new potential energy/kT/e to the –V old/kT okay.

So if this ratio has a certain value, the next slide will tell you how to accept or reject with this new move. So this is the slide now, so what I am going to do, this position i, I have moved it to a new position.

(Refer Slide Time: 06:10)



$$\delta V_{nm} = \left(\sum_{\substack{j=1 \\ \neq i}}^{N} v(r_{ij}^{n}) - \sum_{\substack{j=1 \\ \neq i}}^{N} v(r_{ij}^{m}) \right)$$

$$\frac{\rho_n}{\rho_m} = \frac{Z_{NVT}^{-1}\, e^{-\beta V_n}}{Z_{NVT}^{-1}\, e^{-\beta V_m}} = \frac{e^{-\beta V_n}\, e^{-\beta \delta V_{nm}}}{e^{-\beta V_n}} = e^{-\beta \delta V_{nm}}$$

If $\delta V_{nm} > 0$, the move is uphill and is accepted with a probability ρ_n/ρ_m

When I have moved it to a new position, I have a new potential energy. So I took this delta V is the difference in the potential energy. So now if delta V is>0, so that means okay. So look at this graph, whenever the new potential energy is<the old potential energy, this ratio will be>1 okay. This ratio will be>1, so I will accept that move. So whenever the new potential energy is<the old potential energy that ratio e to the –delta V will be<1.

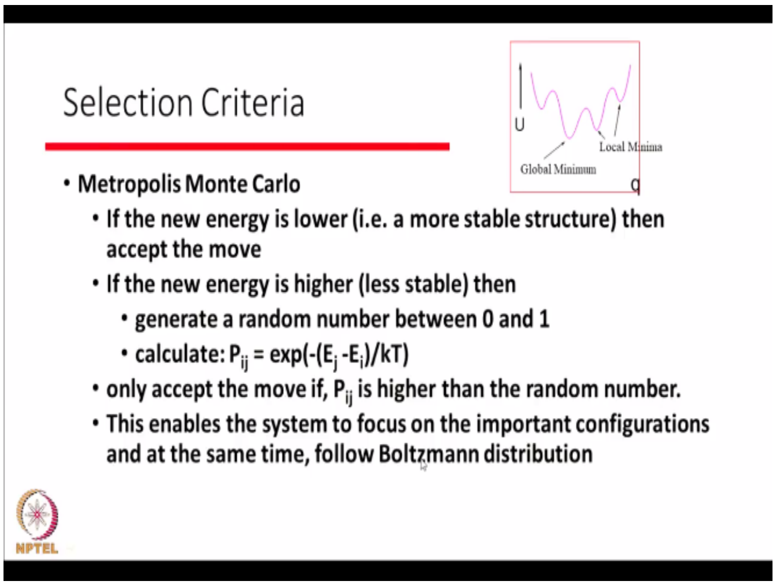
So whenever this ratio is<1, suppose it is at this point, obtain a new random number. When I obtain a new random number, if that random number is<this red line, accept the move. If the new random number is>this red line, reject the move. So what this means physically is that if the new potential energy is very high compared to the old potential energy, then e to the –delta V will be very small.

So very high energy configurations will be accepted with very low probability, very low energy configurations will be accepted with high probability and whenever the new potential energy is greater is less than the present one, then it is certainly accepted. So this is the strategy. So energy configurations with very high energy are accepted with low probability, configurations with low energy are accepted with high probability.

So what it means is that the distribution function we get is exactly like a Boltzmann distribution. What is the meaning of a Boltzmann distribution? The Boltzmann distribution says probability is proportional to e to the –beta*energy. So higher energy lower probability, lower energy higher probability, so therefore this particular strategy of Metropolis allows us to sample configuration based on a Maxwell-Boltzmann distribution.

So that is the meaning of this Metropolis Monte Carlo simulation okay.

(Refer Slide Time: 08:13)



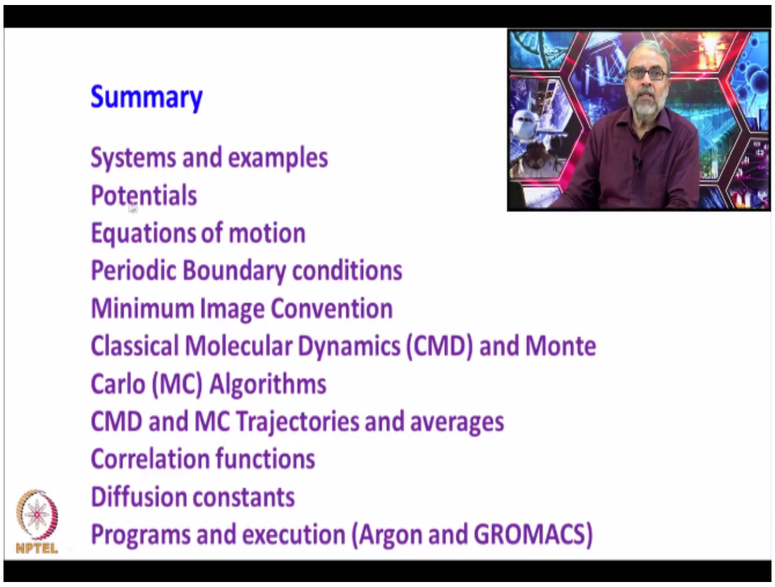
Selection Criteria

- Metropolis Monte Carlo
  - If the new energy is lower (i.e. a more stable structure) then accept the move
  - If the new energy is higher (less stable) then
    - generate a random number between 0 and 1
    - calculate: $P_{ij} = \exp(-(E_j - E_i)/kT)$
  - only accept the move if, P_{ij} is higher than the random number.
  - This enables the system to focus on the important configurations and at the same time, follow Boltzmann distribution

So I have repeated the whole thing here. What are the selection criteria? I will list them again. If the new energy is lower, more stable structure then accept the move. If the new energy is higher that is less stable then generate a random number between 0 and 1, calculate Pij is=e to the-Ej-Ei. Only accept the move if Pij is higher than the random number. If the random number is<Pij, accept.

If the random number is>Pij, then do not accept. So this allows me as I mentioned earlier that the whole distribution you get will sample all the important configurations corresponding a Boltzmann distribution.

(Refer Slide Time: 08:57)



Summary

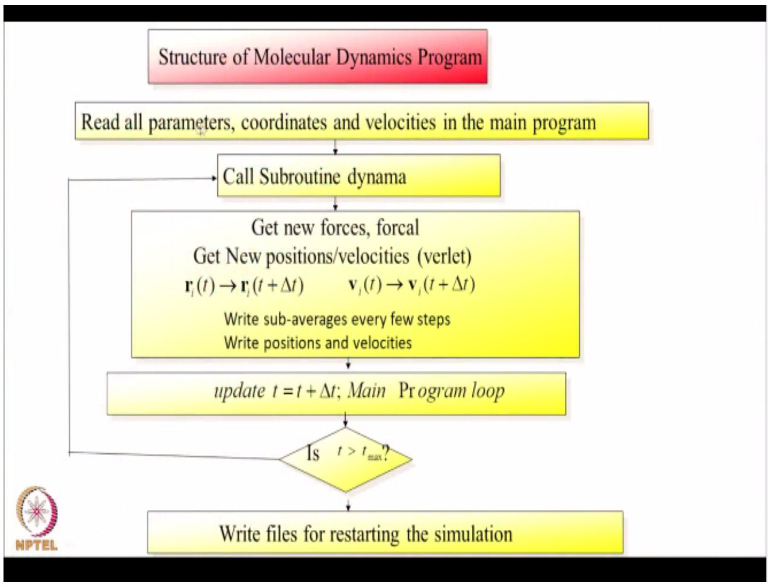
Systems and examples
Potentials
Equations of motion
Periodic Boundary conditions
Minimum Image Convention
Classical Molecular Dynamics (CMD) and Monte Carlo (MC) Algorithms
CMD and MC Trajectories and averages
Correlation functions
Diffusion constants
Programs and execution (Argon and GROMACS)

So this is my Monte Carlo simulation. Now let me summarize what I have done so far. I started with systems and examples of molecular dynamics. I considered the potentials. I considered the equations of motion. We discussed periodic boundary conditions. We need periodic boundary conditions because I want to simulate an infinite system; I want to simulate a liquid.

So you cannot simulate a liquid with 50 or 100 or 200 particles, it has to be much, much larger because liquids typically Avogadro number of particles, so to have an infinite system I use a periodic boundary condition. So it not only removes the edge effects, it allows me to simulate real liquid. So minimum image convention, I only consider interaction of a central particle with all the remaining n-1 particles with this particle at the center, so that is the minimum image convention.

We considered in detail MD classical molecular dynamic simulations as well as Monte Carlo simulations okay. We discussed trajectories and averages. We discussed spatial correlation functions as well as time correlation functions, diffusion constant through mean square displacements and what we will do? We will execute a program now of Monte Carlo as well as molecular dynamics. So before I execute I will tell you what is now the strategy of that program.

(Refer Slide Time: 10:18)



Structure of Molecular Dynamics Program

Read all parameters, coordinates and velocities in the main program

Call Subroutine dynama

Get new forces, forcal
Get New positions/velocities (verlet)
$\mathbf{r}_i(t) \rightarrow \mathbf{r}_i(t+\Delta t)$ $\quad \mathbf{v}_i(t) \rightarrow \mathbf{v}_i(t+\Delta t)$
Write sub-averages every few steps
Write positions and velocities

update $t = t + \Delta t$; *Main Program loop*

Is $t > t_{\max}$?

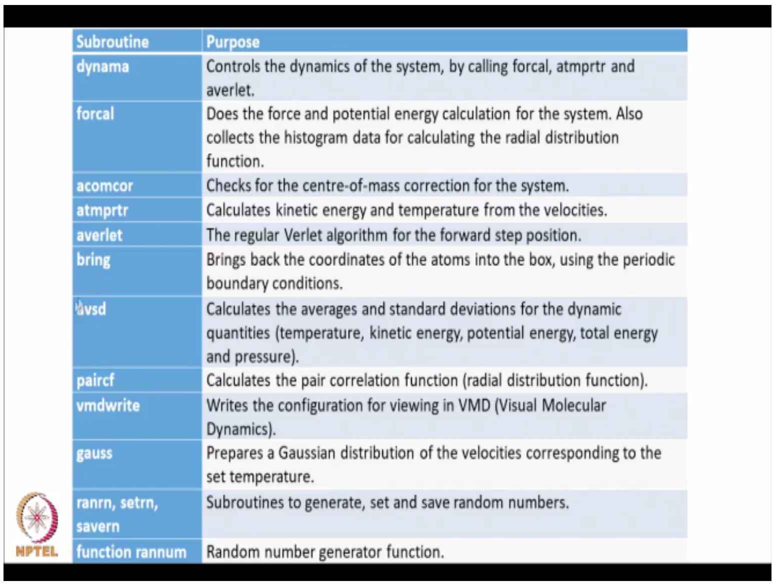
Write files for restarting the simulation

So that strategy of the program would be the structure is in the main program I read all the parameters like coordinates, velocities, I read in the main program. Then, calculate, then call a subroutine called dynama. Dynama is a dynamic subroutine. So what this subroutine does? It calculates other subroutines. What are them? Forcal, calculate the force okay. Verlet is a subroutine which gives me new positions and velocities okay.

Then, every few steps write the averages and every step try to write position and velocities, so that will complete my one step. Once I complete to at one step, I update my t to t+delta t okay. So this is done in the main program. Then, if this new time is much greater than the maximum time, then stop. If it is less I go back and keep on calculating my new positions and new velocities until you do from the starting time till the end time.

So starting time typically is 0 picosecond, ending time typically 50 or 100 nanoseconds. So this is how we will do the simulation and at the end of the simulation write files so that I can restart the calculation. So next slide illustrates what are all the subroutines here.

(Refer Slide Time: 11:35)



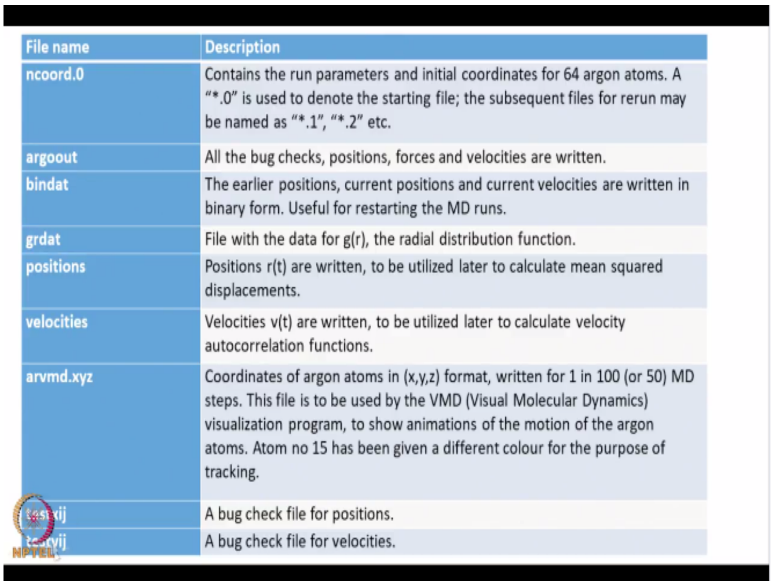
Subroutine	Purpose
dynama	Controls the dynamics of the system, by calling forcal, atmprtr and averlet.
forcal	Does the force and potential energy calculation for the system. Also collects the histogram data for calculating the radial distribution function.
acomcor	Checks for the centre-of-mass correction for the system.
atmprtr	Calculates kinetic energy and temperature from the velocities.
averlet	The regular Verlet algorithm for the forward step position.
bring	Brings back the coordinates of the atoms into the box, using the periodic boundary conditions.
avsd	Calculates the averages and standard deviations for the dynamic quantities (temperature, kinetic energy, potential energy, total energy and pressure).
paircf	Calculates the pair correlation function (radial distribution function).
vmdwrite	Writes the configuration for viewing in VMD (Visual Molecular Dynamics).
gauss	Prepares a Gaussian distribution of the velocities corresponding to the set temperature.
ranrn, setrn, savern	Subroutines to generate, set and save random numbers.
function rannum	Random number generator function.

As I mentioned to you dynama, it controls the dynamics of the system and it calls you forcal. Forcal is the force calculation. This subroutine is to calculate temperature and Verlet is to calculate new positions and velocities. Forcal, it calculates the potential energy as well as the force okay. Now acomcor, it checks the center of mass of the entire system. You know as you keep on doing the simulation, the entire simulation box should not start moving to the right and left.

So you make sure of that by correcting for the center of mass. Verlet calculates you Verlet algorithm, so whenever a particle goes outside the box, you bring it back for the periodic boundary condition. Avsd calculates you averages, paircf calculates pair correlation function, this vmdwrite it writes the configuration for a VMD program, gauss is a subroutine to calculate Gaussian random numbers okay.

This one is for normal random numbers. There is a function random number generative also.

(Refer Slide Time: 12:39)

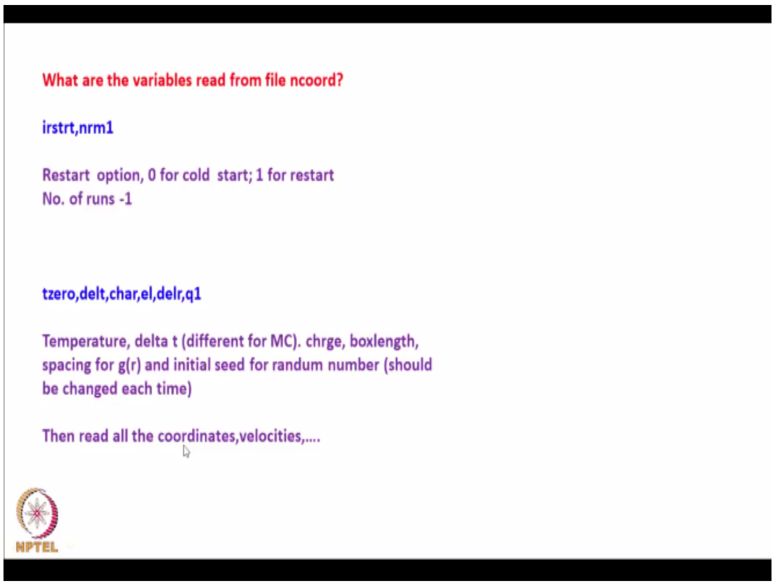


File name	Description
ncoord.0	Contains the run parameters and initial coordinates for 64 argon atoms. A "*.0" is used to denote the starting file; the subsequent files for rerun may be named as "*.1", "*.2" etc.
argout	All the bug checks, positions, forces and velocities are written.
bindat	The earlier positions, current positions and current velocities are written in binary form. Useful for restarting the MD runs.
grdat	File with the data for g(r), the radial distribution function.
positions	Positions r(t) are written, to be utilized later to calculate mean squared displacements.
velocities	Velocities v(t) are written, to be utilized later to calculate velocity autocorrelation functions.
arvmd.xyz	Coordinates of argon atoms in (x,y,z) format, written for 1 in 100 (or 50) MD steps. This file is to be used by the VMD (Visual Molecular Dynamics) visualization program, to show animations of the motion of the argon atoms. Atom no 15 has been given a different colour for the purpose of tracking.
testxij	A bug check file for positions.
testvij	A bug check file for velocities.

Then, what about the files for data files, ncord or ncord0 contains all the initial positions as well as certain parameters okay. So there are certain parameters to decide whether to start from a fixed configuration or to start from a previous configuration. Argout, it gives you several outputs. Bindat gives you a restart configuration. Grdat is for pair distribution function. Positions are to calculate mean square displacement.

Velocities are for time correlation function and these other things are to check for errors in the program okay.

(Refer Slide Time: 13:17)



What are the variables read from file ncoord?

irstrt,nrm1

Restart option, 0 for cold start; 1 for restart
No. of runs -1

tzero,delt,char,el,delr,q1

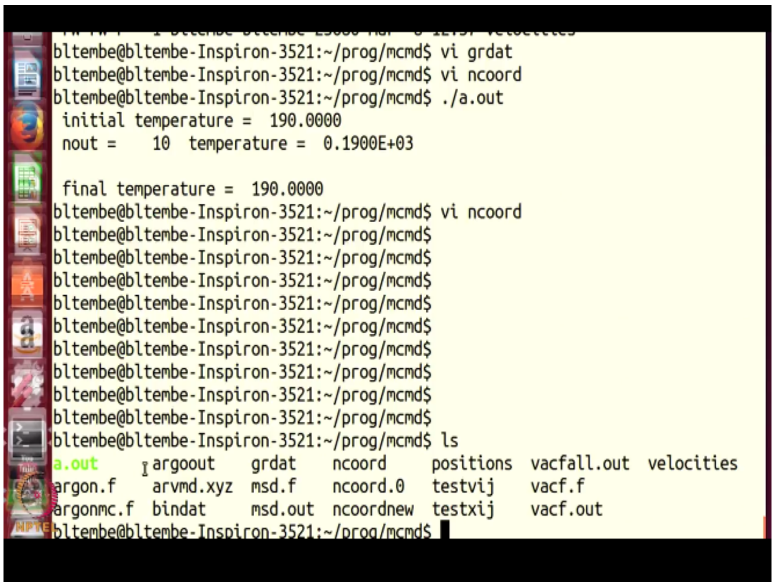
Temperature, delta t (different for MC). chrge, boxlength, spacing for g(r) and initial seed for randum number (should be changed each time)

Then read all the coordinates,velocities,....

So now let us see how to run this program. So the file data file is containing file ncoord. So this ncoord has 3 data and the position. So first line will tell me whether I will restart from a frozen configuration or restart from an old configuration. This is a restart option, so the next one will be temperature, delta t, charge, box length, delta r for calculating GFR and the q1 is an initial random number.

So these are all the variables, I will show you in the program. Then, finally it reads all the coordinates and velocities. So what I will do now, I will actually shift to the program, execute Monte Carlo as well as molecular dynamics and then conclude this part of my lecture. I am now going to demonstrate execution of the molecular dynamics as well as the Monte Carlo program. All my files are in this directory mcmd okay.

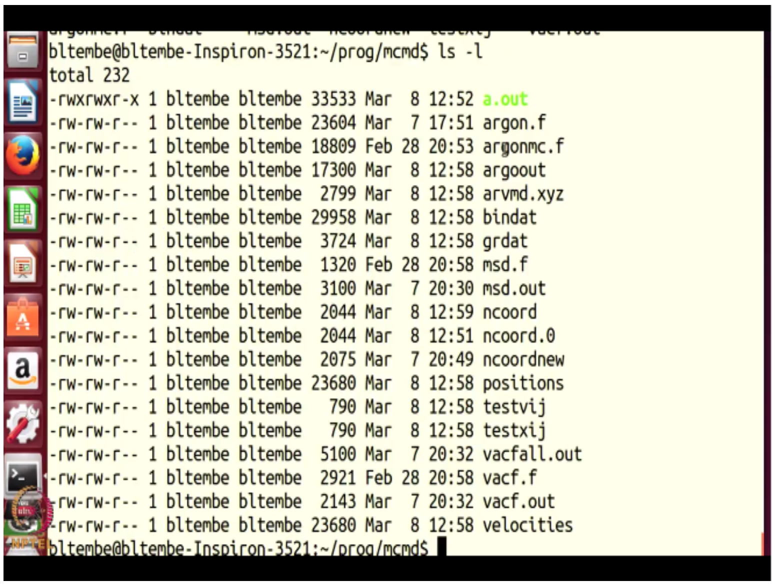
(Refer Slide Time: 14:22)



```
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ vi grdat
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ vi ncoord
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ ./a.out
 initial temperature =  190.0000
 nout =    10  temperature =  0.1900E+03

 final temperature =  190.0000
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ vi ncoord
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ ls
a.out      argoout    grdat    ncoord     positions  vacfall.out  velocities
argon.f    arvmd.xyz  msd.f    ncoord.0   testvij    vacf.f
argonmc.f  bindat     msd.out  ncoordnew  testxij    vacf.out
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
```

So let me just do ls, so when I do ls you get a list of all the files, you see how which are my files okay.

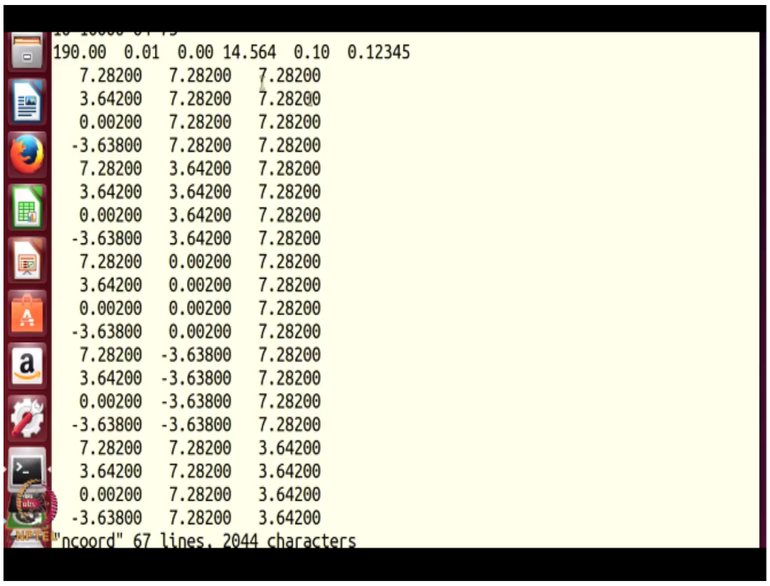
(Refer Slide Time: 14:35)



```
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ ls -l
total 232
-rwxrwxr-x 1 bltembe bltembe 33533 Mar  8 12:52 a.out
-rw-rw-r-- 1 bltembe bltembe 23604 Mar  7 17:51 argon.f
-rw-rw-r-- 1 bltembe bltembe 18809 Feb 28 20:53 argonmc.f
-rw-rw-r-- 1 bltembe bltembe 17300 Mar  8 12:58 argoout
-rw-rw-r-- 1 bltembe bltembe  2799 Mar  8 12:58 arvmd.xyz
-rw-rw-r-- 1 bltembe bltembe 29958 Mar  8 12:58 bindat
-rw-rw-r-- 1 bltembe bltembe  3724 Mar  8 12:58 grdat
-rw-rw-r-- 1 bltembe bltembe  1320 Feb 28 20:58 msd.f
-rw-rw-r-- 1 bltembe bltembe  3100 Mar  7 20:30 msd.out
-rw-rw-r-- 1 bltembe bltembe  2044 Mar  8 12:59 ncoord
-rw-rw-r-- 1 bltembe bltembe  2044 Mar  8 12:51 ncoord.0
-rw-rw-r-- 1 bltembe bltembe  2075 Mar  7 20:49 ncoordnew
-rw-rw-r-- 1 bltembe bltembe 23680 Mar  8 12:58 positions
-rw-rw-r-- 1 bltembe bltembe   790 Mar  8 12:58 testvij
-rw-rw-r-- 1 bltembe bltembe   790 Mar  8 12:58 testxij
-rw-rw-r-- 1 bltembe bltembe  5100 Mar  7 20:32 vacfall.out
-rw-rw-r-- 1 bltembe bltembe  2921 Feb 28 20:58 vacf.f
-rw-rw-r-- 1 bltembe bltembe  2143 Mar  7 20:32 vacf.out
-rw-rw-r-- 1 bltembe bltembe 23680 Mar  8 12:58 velocities
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
```

Let me do ls-l okay. My argon.f is my molecular dynamics program, argonmc.f is my Monte Carlo program. Then, msd.f calculates me mean square displacements and vacf.f calculates me velocity autocorrelation function. So what I will do? I will illustrate first the molecular dynamics program. So how do I do that?

(Refer Slide Time: 15:02)



```
190.00  0.01  0.00 14.564  0.10  0.12345
   7.28200   7.28200   7.28200
   3.64200   7.28200   7.28200
   0.00200   7.28200   7.28200
  -3.63800   7.28200   7.28200
   7.28200   3.64200   7.28200
   3.64200   3.64200   7.28200
   0.00200   3.64200   7.28200
  -3.63800   3.64200   7.28200
   7.28200   0.00200   7.28200
   3.64200   0.00200   7.28200
   0.00200   0.00200   7.28200
  -3.63800   0.00200   7.28200
   7.28200  -3.63800   7.28200
   3.64200  -3.63800   7.28200
   0.00200  -3.63800   7.28200
  -3.63800  -3.63800   7.28200
   7.28200   7.28200   3.64200
   3.64200   7.28200   3.64200
   0.00200   7.28200   3.64200
  -3.63800   7.28200   3.64200
"ncoord" 67 lines, 2044 characters
```

I compile gfortran argon.f that is my molecular dynamics program. So it has compiled now. So let me see all the data is in ncoord. So let me see vi ncoord okay. So this is data. The first one is the restart option, I discussed in the earlier class and this 1 is to read the data. The next I have this 10 tells me subaverages, how many subaverages I have. The next one is the number of steps.

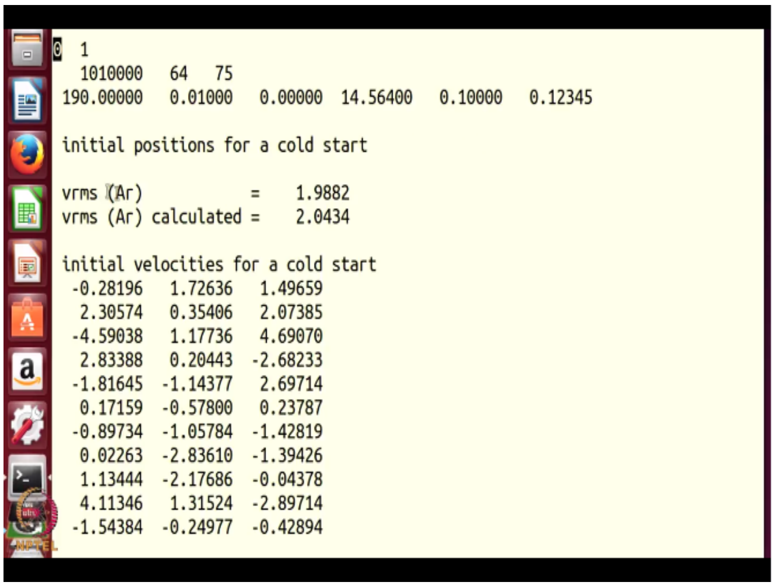
This is the number of particle 64 okay, 64 is the number of particles and 75 is how many boxes I put my GFR. See I want to calculate GFR, there are 75 boxes that is 7.5 angstrom, each bin for GFR is 0.1 angstrom. Then, temperature is 190 this is my temperature, 0.01 picosecond is my time step okay, then third one is a charge, our system has no charge, so I put 0. Then, 14.564 is the box length, 0.1 is the spacing for GFR and 0.12345 is a random number.

And then the next 64 lines since there are 64 particles there are 64 positions. These positions are all lattice positions now okay. So now "**Professor - student conversation starts**" Can we start from any random number? You can start with any random number. Main thing is you should every time you should change a random number because if you keep the same random number you will get the same sequence, so good programming wants to change it to some other number. **"Professor - student conversation ends."**

And we know that when you start with the given random number, to get the same random number it takes a very, very long time okay. So the algorithm is such that the random numbers will not repeat for at least several lakh steps. So even if you give the same random number, it will not be too bad but it is not a good thing. Always change the random number so that ultimately you want to have a different configuration.

Your Monte Carlo should have, it should span the whole space, always it should be different okay. So now I will run that MD program now.

(Refer Slide Time: 17:30)



```
0  1
  1010000   64   75
190.00000   0.01000   0.00000  14.56400   0.10000   0.12345

initial positions for a cold start

vrms (Ar)            =    1.9882
vrms (Ar) calculated =    2.0434

initial velocities for a cold start
 -0.28196   1.72636   1.49659
  2.30574   0.35406   2.07385
 -4.59038   1.17736   4.69070
  2.83388   0.20443  -2.68233
 -1.81645  -1.14377   2.69714
  0.17159  -0.57800   0.23787
 -0.89734  -1.05784  -1.42819
  0.02263  -2.83610  -1.39426
  1.13444  -2.17686  -0.04378
  4.11346   1.31524  -2.89714
 -1.54384  -0.24977  -0.42894
```

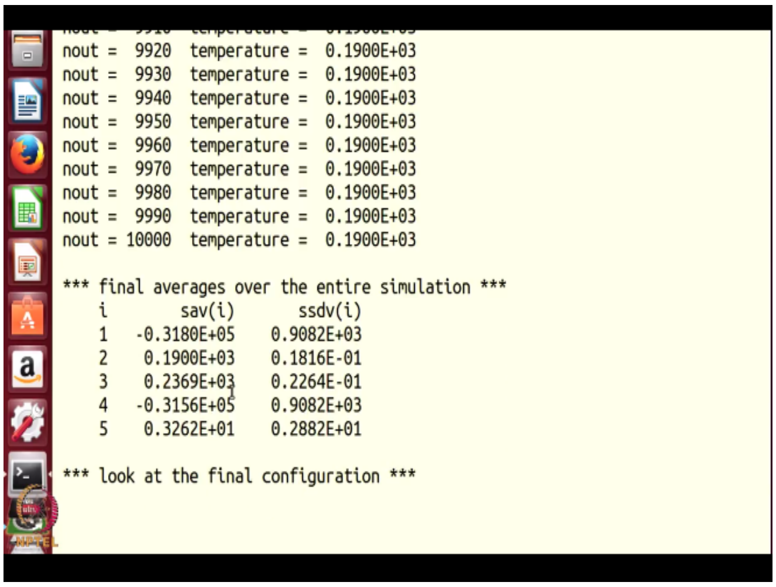
So a.out, so now it is saying, okay so I made a mistake, dot slash I did not give, dot slash. So now it is doing all these calculations, it will do for 10,000 steps okay. So when it does 10,000 steps, it will stop my okay. So it has done the 10,000 steps. So I can see that, so this is my output file. So what does this output file contain? My initial data whatever I fed in initial data okay. Then, initial okay I am not able to see it okay.

The output file contains this argout, all the initial data I fed. This is from a cold start because restart was 0 okay. Then, I calculate velocities from a random distribution okay at temperature t. So this is my root mean square velocity from the Gaussian distribution. Then, it writes all the initial velocities for a cold start, it writes the initial velocities okay. It writes all the initial velocities, there are 64 of them.

Then, temperature, it lists all the forces, it lists all the forces on each of the atom okay. Then, it calculates the temperature in every subaverage okay, every 10 steps it is calculating the temperature. So it is listing the temperature for the entire simulation. There will be 10,000 steps. So I am going to write every 10 steps. So it will write 1000 times. "**Professor - student conversation starts**" Here we are not giving anything to keep the temperature constant?

No, there is a rescaling, we rescale temperature every few steps that is already there okay. "**Professor - student conversation ends**." So we rescale every step, every few steps. So now these are my averages, during the simulation these are the final averages over the entire sample.

(Refer Slide Time: 19:40)

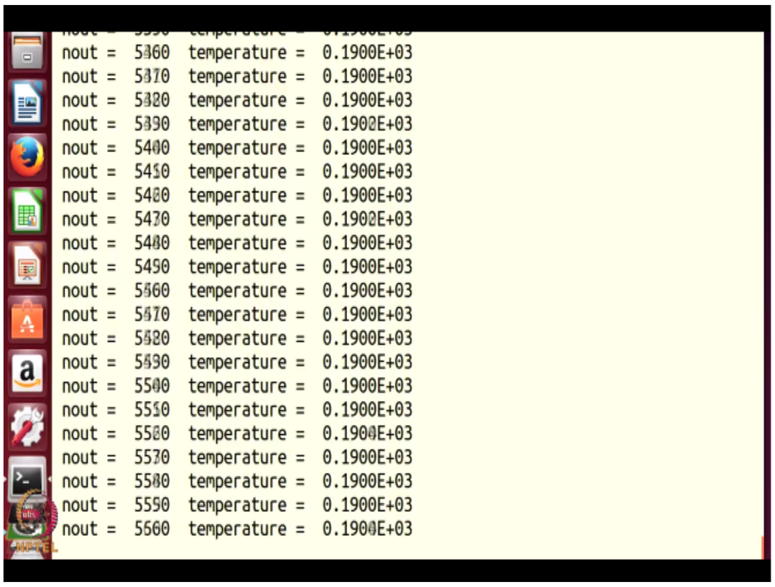


```
 nout =  9920  temperature =  0.1900E+03
 nout =  9930  temperature =  0.1900E+03
 nout =  9940  temperature =  0.1900E+03
 nout =  9950  temperature =  0.1900E+03
 nout =  9960  temperature =  0.1900E+03
 nout =  9970  temperature =  0.1900E+03
 nout =  9980  temperature =  0.1900E+03
 nout =  9990  temperature =  0.1900E+03
 nout = 10000  temperature =  0.1900E+03

 *** final averages over the entire simulation ***
     i        sav(i)       ssdv(i)
     1   -0.3180E+05    0.9082E+03
     2    0.1900E+03    0.1816E-01
     3    0.2369E+03    0.2264E-01
     4   -0.3156E+05    0.9082E+03
     5    0.3262E+01    0.2882E+01

 *** look at the final configuration ***
```

So you can go to the program, you will see that the first one will be the average potential energy okay. Second one will be the temperature. Third one will be kinetic energy. So all the averages are calculated during the simulation and finally this is my final configuration, final forces okay. These are my final forces, then final positions okay, then final velocities okay. So and this is random number.

So now I come out of this okay. So if I want to restart from this configuration, I change my ncoord. So okay that is I have made a wrong statement. I will come out, vi ncoord, so what I do here, I change the first 0 to 1, you are not able to see that okay. I change let me show that.

(Refer Slide Time: 20:48)

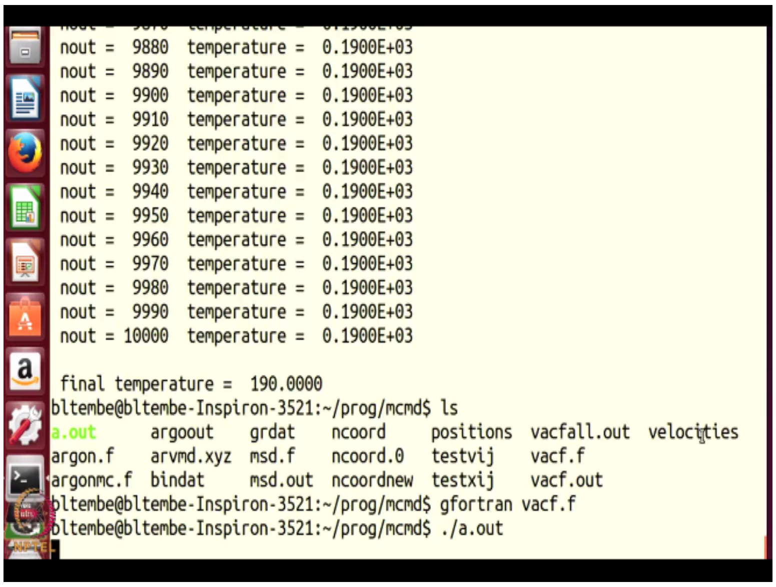


```
nout =  5360  temperature =  0.1900E+03
nout =  5370  temperature =  0.1900E+03
nout =  5380  temperature =  0.1900E+03
nout =  5390  temperature =  0.1902E+03
nout =  5400  temperature =  0.1900E+03
nout =  5410  temperature =  0.1900E+03
nout =  5420  temperature =  0.1900E+03
nout =  5430  temperature =  0.1902E+03
nout =  5440  temperature =  0.1900E+03
nout =  5450  temperature =  0.1900E+03
nout =  5460  temperature =  0.1900E+03
nout =  5470  temperature =  0.1900E+03
nout =  5480  temperature =  0.1900E+03
nout =  5490  temperature =  0.1900E+03
nout =  5500  temperature =  0.1900E+03
nout =  5510  temperature =  0.1900E+03
nout =  5520  temperature =  0.1900E+03
nout =  5530  temperature =  0.1900E+03
nout =  5540  temperature =  0.1900E+03
nout =  5550  temperature =  0.1900E+03
nout =  5560  temperature =  0.1900E+03
```

So the first one was 0 for a cold start, now I will replace it with 1 okay. So I have replaced that, now I will run a.out again, dot slash, so this time when I run it is running from the last configuration of the last simulation. When I did 10,000 steps, so remember in Verlet, I need present step and the old step correct. So what it is writing in that file bindat, it will write the old position, current position and the current velocity.

So for a restart, I use bindat file which has current position, past position and velocity, so then I can do Verlet know. For Verlet I need V new=two times V current-V old, so that is the MD run. Now I want to calculate velocity out of correlation function. So what I will do?

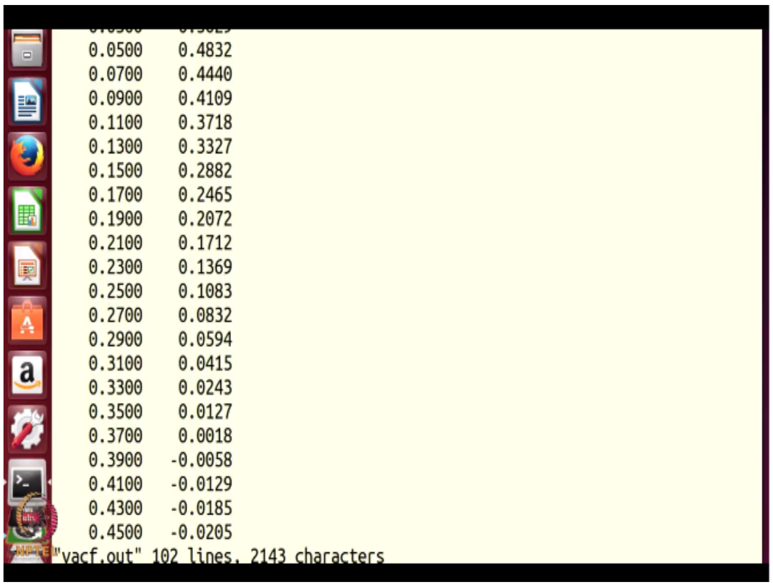
(Refer Slide Time: 22:07)



```
nout =  9880  temperature =  0.1900E+03
nout =  9890  temperature =  0.1900E+03
nout =  9900  temperature =  0.1900E+03
nout =  9910  temperature =  0.1900E+03
nout =  9920  temperature =  0.1900E+03
nout =  9930  temperature =  0.1900E+03
nout =  9940  temperature =  0.1900E+03
nout =  9950  temperature =  0.1900E+03
nout =  9960  temperature =  0.1900E+03
nout =  9970  temperature =  0.1900E+03
nout =  9980  temperature =  0.1900E+03
nout =  9990  temperature =  0.1900E+03
nout = 10000  temperature =  0.1900E+03

 final temperature =   190.0000
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ ls
a.out      argoout    grdat    ncoord     positions  vacfall.out  velocities
argon.f    arvmd.xyz  msd.f    ncoord.0   testvij    vacf.f
argonmc.f  bindat     msd.out  ncoordnew  testxij    vacf.out
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ gfortran vacf.f
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$ ./a.out
```

Let me compile that program, gfortran vacf.f, so I compiled the velocity autocorrelation program. Now I do dot slash a.out, so this program uses all the velocities, look at this line, see it writes all the velocities can you see this velocities here. That velocities has all the velocities for 10,000 steps. Positions will have all the positions for 10,000 steps. So I can execute. Now it has already completed.

(Refer Slide Time: 22:46)

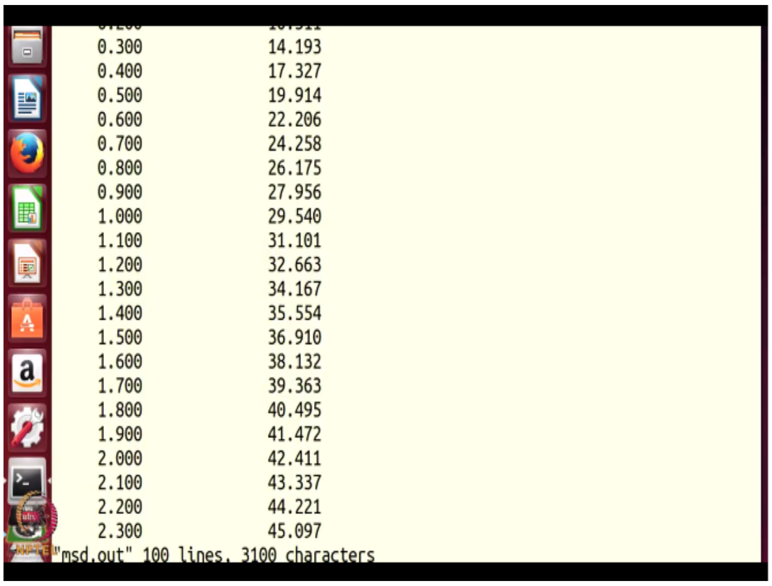


```
0.0500    0.4832
0.0700    0.4440
0.0900    0.4109
0.1100    0.3718
0.1300    0.3327
0.1500    0.2882
0.1700    0.2465
0.1900    0.2072
0.2100    0.1712
0.2300    0.1369
0.2500    0.1083
0.2700    0.0832
0.2900    0.0594
0.3100    0.0415
0.3300    0.0243
0.3500    0.0127
0.3700    0.0018
0.3900   -0.0058
0.4100   -0.0129
0.4300   -0.0185
0.4500   -0.0205
"vacf.out" 102 lines, 2143 characters
```

So let us see, so I did vacf, so vacf.out will have the output for the velocity autocorrelation function, so let me edit that vi vacf.out. So these are all that my velocity autocorrelation function (()) (23:02) 0.1 this is the value, 0.3 this is value. So as the time increases to around 0.39, it has taken a negative value. Remember, we had shown a plot of autocorrelation function, it starts with 1 goes to a negative value, comes up again and finally goes to 0.

So this is my velocity autocorrelation function program, so finally see it has gone to 0 at about 2 picoseconds. It has gone to 0 and it also calculates a diffusion constant using that relation integral, this is the diffusion constant. So now I can also do mean square displacement.

(Refer Slide Time: 23:41)



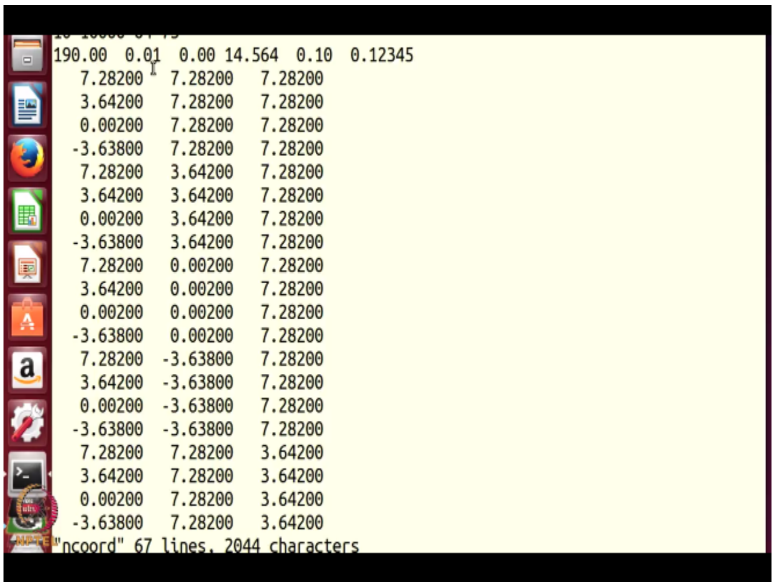
```
0.300          14.193
0.400          17.327
0.500          19.914
0.600          22.206
0.700          24.258
0.800          26.175
0.900          27.956
1.000          29.540
1.100          31.101
1.200          32.663
1.300          34.167
1.400          35.554
1.500          36.910
1.600          38.132
1.700          39.363
1.800          40.495
1.900          41.472
2.000          42.411
2.100          43.337
2.200          44.221
2.300          45.097
"msd.out" 100 lines, 3100 characters
```

How do I do that? gfortran, so I am compiling the mean square displacement program, so it will calculate mean square displacement as a function of time. So just as in the earlier case, I had a plot of velocity autocorrelation versus time. Now I will have a mean square displacement versus time. Once this program is over, I will tell you how it calculates the mean square displacement okay.

So what was that file now? msd.out, you see that msd.out. So I read that file vi msd.out so at each time step it is calculating the mean square displacement. So if I plot this mean square displacement versus time, I can take the slope and calculate the diffusion constant. So remember that the mean square displacement become 6dt at large times. So this is how you can calculate the mean square displacement by plotting this mean square displacement versus time.

And I would also urge you to look at a software called xmgrace. Xmgrace allows you to take a file from your data and plot on your screen. Otherwise, you will have to take the Scilab, you can plot in Scilab know. You can take all this is Scilab and plot or you can plot using xmgrace software.

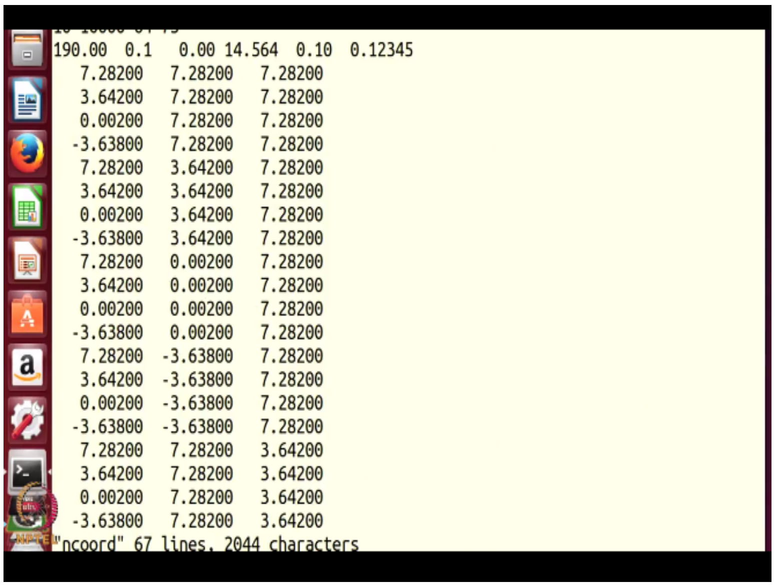
(Refer Slide Time: 25:20)



```
190.00  0.01   0.00 14.564  0.10  0.12345
    7.28200   7.28200   7.28200
    3.64200   7.28200   7.28200
    0.00200   7.28200   7.28200
   -3.63800   7.28200   7.28200
    7.28200   3.64200   7.28200
    3.64200   3.64200   7.28200
    0.00200   3.64200   7.28200
   -3.63800   3.64200   7.28200
    7.28200   0.00200   7.28200
    3.64200   0.00200   7.28200
    0.00200   0.00200   7.28200
   -3.63800   0.00200   7.28200
    7.28200  -3.63800   7.28200
    3.64200  -3.63800   7.28200
    0.00200  -3.63800   7.28200
   -3.63800  -3.63800   7.28200
    7.28200   7.28200   3.64200
    3.64200   7.28200   3.64200
    0.00200   7.28200   3.64200
   -3.63800   7.28200   3.64200
"ncoord" 67 lines, 2044 characters
```

So the last program I will illustrate now is Monte Carlo program, so for that I will do I will compile the Monte Carlo program, gfortran I am compiling the Monte Carlo program okay. So I have compiled the Monte Carlo program, so in the Monte Carlo program there is no time. So what we do, see there is no time. So in my old program this 0.01 was delta t, so now this delta t is the displacement in Monte Carlo so what I will do?

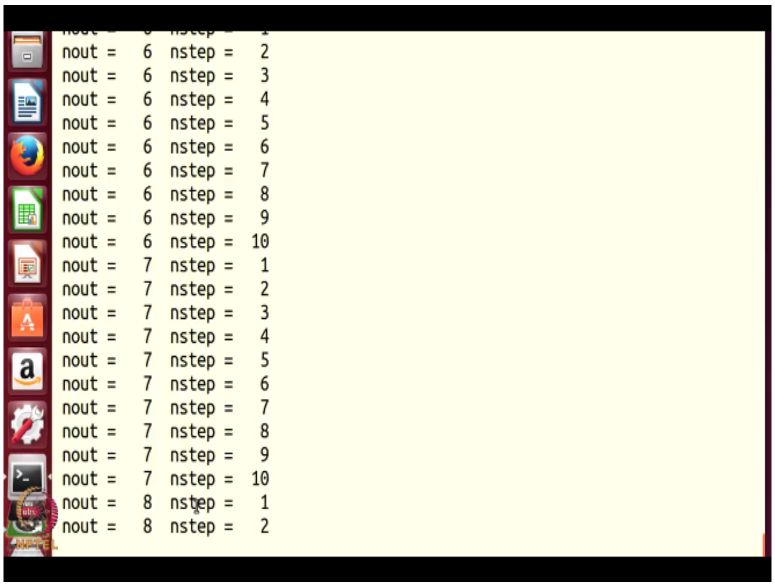
(Refer Slide Time: 26:10)



```
190.00  0.1    0.00 14.564  0.10  0.12345
    7.28200   7.28200   7.28200
    3.64200   7.28200   7.28200
    0.00200   7.28200   7.28200
   -3.63800   7.28200   7.28200
    7.28200   3.64200   7.28200
    3.64200   3.64200   7.28200
    0.00200   3.64200   7.28200
   -3.63800   3.64200   7.28200
    7.28200   0.00200   7.28200
    3.64200   0.00200   7.28200
    0.00200   0.00200   7.28200
   -3.63800   0.00200   7.28200
    7.28200  -3.63800   7.28200
    3.64200  -3.63800   7.28200
    0.00200  -3.63800   7.28200
   -3.63800  -3.63800   7.28200
    7.28200   7.28200   3.64200
    3.64200   7.28200   3.64200
    0.00200   7.28200   3.64200
   -3.63800   7.28200   3.64200
"ncoord" 67 lines, 2044 characters
```

I will change it to instead of 0.01 I want 0.1 okay. How do I delete that? So I have made 0.1, so I have made, so I have added escape, so I have added 0.1 as the maximum displacement in each step, 0.1 angstrom. So no particle will move more than 0.1 angstrom, so I will escape. Did I compile the Monte Carlo? Now let us see again. So it is alright. So I escape, argon and c I have compiled, now let me execute dot slash a.out.

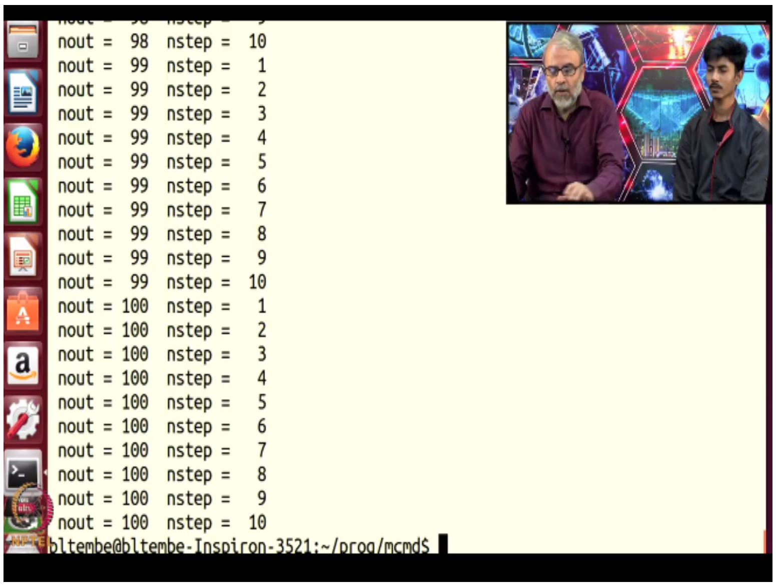
(Refer Slide Time: 27:15)



```
nout =   6  nstep =   2
nout =   6  nstep =   3
nout =   6  nstep =   4
nout =   6  nstep =   5
nout =   6  nstep =   6
nout =   6  nstep =   7
nout =   6  nstep =   8
nout =   6  nstep =   9
nout =   6  nstep =  10
nout =   7  nstep =   1
nout =   7  nstep =   2
nout =   7  nstep =   3
nout =   7  nstep =   4
nout =   7  nstep =   5
nout =   7  nstep =   6
nout =   7  nstep =   7
nout =   7  nstep =   8
nout =   7  nstep =   9
nout =   7  nstep =  10
nout =   8  nstep =   1
nout =   8  nstep =   2
```

So now it is doing this Monte Carlo calculation for 10,000 steps. So this will take a very long time. So should we do it for smaller steps, so that we can demonstrate quickly? **"Professor - student conversation starts."** Yes, sir. **"Professor - student conversation ends."** So I can stop it by doing control z, this has stopped. So what I will do instead of 10,000 steps, I will just do for 100 steps okay.

(Refer Slide Time: 27:57)



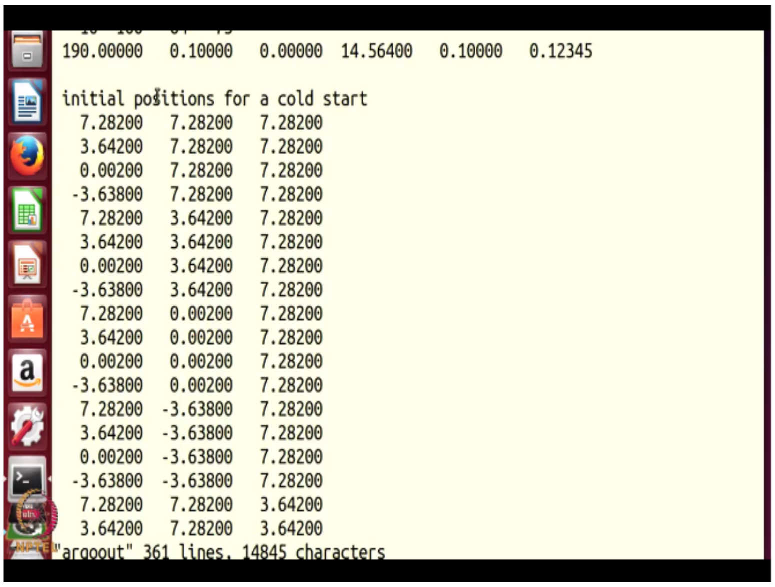
```
nout =  98  nstep =  10
nout =  99  nstep =   1
nout =  99  nstep =   2
nout =  99  nstep =   3
nout =  99  nstep =   4
nout =  99  nstep =   5
nout =  99  nstep =   6
nout =  99  nstep =   7
nout =  99  nstep =   8
nout =  99  nstep =   9
nout =  99  nstep =  10
nout = 100  nstep =   1
nout = 100  nstep =   2
nout = 100  nstep =   3
nout = 100  nstep =   4
nout = 100  nstep =   5
nout = 100  nstep =   6
nout = 100  nstep =   7
nout = 100  nstep =   8
nout = 100  nstep =   9
nout = 100  nstep =  10
bltembe@bltembe-Inspiron-3521:~/prog/mcmd$
```

So this is mainly to illustrate, so since it is going to do 100 steps, it will be over very quickly. So what we are doing here? We are executing the Monte Carlo program not the molecular dynamics okay. So this will be over very shortly. So it will be 100*10, so 1000 steps and in Monte Carlo simulation it will not calculate for you any velocities, it will only calculate new positions, no forces are calculated because nothing moves.

Because of a force it is just a random movement, so at the end of this you will see that argout will have an output file which has only the final configuration. The velocities have no meaning okay. So what I urge you now since we have used these different programs, look through the program and check all the names of subroutines. So check all the names of subroutines.

So the name will tell you, forcal will calculate forces, Verlet will calculate positions, MC will calculate MC steps using random numbers, so that allows you and look at all the read and write statements, what files are written to, what files are read from and once you know this you will get a good feel for all these programs okay. So it should stop at 100, so I will just edit okay. So I wrote a wrong thing.

(Refer Slide Time: 29:20)



```
190.00000   0.10000   0.00000  14.56400   0.10000   0.12345

initial positions for a cold start
  7.28200   7.28200   7.28200
  3.64200   7.28200   7.28200
  0.00200   7.28200   7.28200
 -3.63800   7.28200   7.28200
  7.28200   3.64200   7.28200
  3.64200   3.64200   7.28200
  0.00200   3.64200   7.28200
 -3.63800   3.64200   7.28200
  7.28200   0.00200   7.28200
  3.64200   0.00200   7.28200
  0.00200   0.00200   7.28200
 -3.63800   0.00200   7.28200
  7.28200  -3.63800   7.28200
  3.64200  -3.63800   7.28200
  0.00200  -3.63800   7.28200
 -3.63800  -3.63800   7.28200
  7.28200   7.28200   3.64200
  3.64200   7.28200   3.64200
"argoout" 361 lines, 14845 characters
```

So this is the output of the Monte Carlo. So you see that 0.1 was the spacing that 0.1 is given here, this is my initial cold start okay. "**Professor - student conversation starts**." Because when I do Monte Carlo, see what I have done I have used the old program only, I have not changed anything. Instead of a forcal, I have a MC, all the rest. So what I will do I will change all the unnecessary things and just make it a Monte Carlo program. **"Professor - student conversation ends."**

See you will see that whenever you write a new program, you want to start from an old one correct, it is like when you move to a new room from an old room you carry all the old things from the old room. So this program is the MD program where the force is removed, in place of forces I calculate Monte Carlo maths okay. So in assignments we will give you many, many exercises where you can run this program for other situations.

Now what do you think you will do suppose you want to instead of argon you want to do krypton what will you do? "**Professor - student conversation starts**." We have to change the temperature. No, I want to do at the same temperature, how will you do? Change the coordinates. But you have to change the parameters know. If argon has sigma of 3.4 krypton will have a parameter of 4.0.

So if you change the Lennard-Jones parameter, you will get the krypton okay. So you can do for many, many systems now. **"Professor - student conversation ends."** So what I will urge you, you look at all the subroutines, look at all the read and write files and execute it for different situations okay. So from next time onwards, we will use a Gromacs program which is far more elaborate but through this MD program you will know what each step in your MD means okay.

It is always good to know every step in a program. A good programmer knows every line in his program whereas an average programmer take someone else program and runs it okay. So hope you will enjoy and we will continue with Gromacs simulations in the next lecture. Thank you.