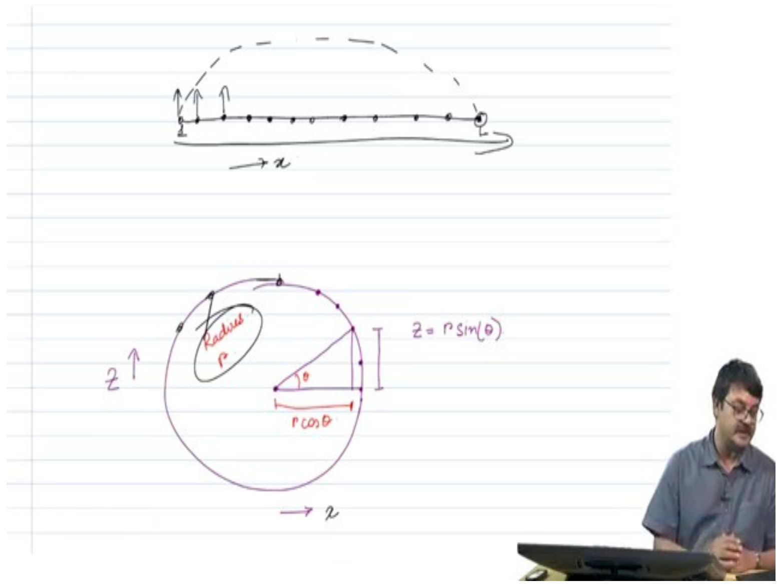
Computational Physics

Dr. Apratim Chatterji
Dr. Prasenjit Ghosh
Department of Physics
Indian Institute of Science Education and Research, Pune

Lecture - 30
Coupled Differential Eqn. Visualisation & Making Movie

(Refer Slide Time: 00:18)



Hello, everybody. So, in the last class we discussed basically Runge-Kutta for array of particles which were organized in a straight line right and basically we are looking at the vibration of a string which had been discretized. And the position of each of these particles was fixed and they could basically displace along the y-direction; so, this being x right.

And to solve the differential equation suppose there were 50 particles we would have 50 first order differential equations for the positions and 50 first order differential equations for the velocity and we were solving them in tandem so, they were coupled differential equations. In addition what we had was periodic boundary conditions. So, the basically the left neighbour of particle number 1 was particle number L, 50 say and the right neighbour of particle number L would be 1. So, they were base essentially periodic boundary conditions.

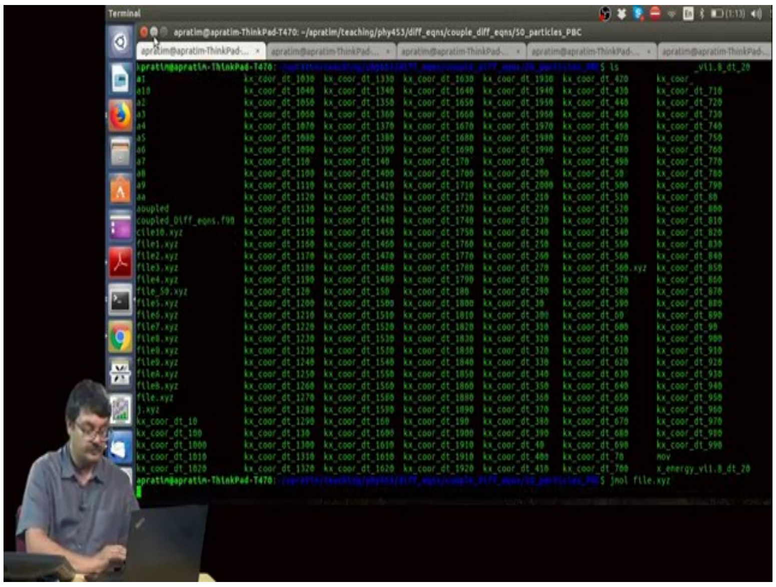
And what we are looking at essentially is basically the displacement perpendicular to the line; so, how much are they moving, what is the kind of wave like motion there is right and what we had said suppose the initial condition is one of the particles we give it an initial displacement and then we see how the wave develops. So, you have basically 100 differential equations which was solving in tandem which means essentially that you are recording the position of each particle as a function of time 100 of them and velocity of each particle as a function of time and so on right.

The question is how do we visualize it? Of course, we can calculate it we can, but how do we visualize it and what is the code line. So, we will be discussing this in a hands on manner in today’s class. One change what I am doing since we have periodic boundary condition what I am doing is basically not putting them in a line we can do that as well, but I am putting them in a circle. So, the position is not along x and you will see that it will help you visualize this because else is difficult to visualize that 1 is connected to L.

So, what I am doing is I am putting all these particles on a circle like this of circle of radius r say right and basically all these particles are placed on the circle as Z equal to Z being this direction and suppose this is x-direction and the particles shall be displaced perpendicular to the plane of the screen. So, it is basically if this is the circle the particles will may displace like this. Now, the particles in are placed as r Z equal to r sin theta and x equal to r cos theta, this is not changing. Their y displacement is changing right.

Without much I do, now let us move and show you how this particle will move in a movie and then I shall show you the code and tell you how to make the movies which is of course, important to visualize also the data which you have generated.

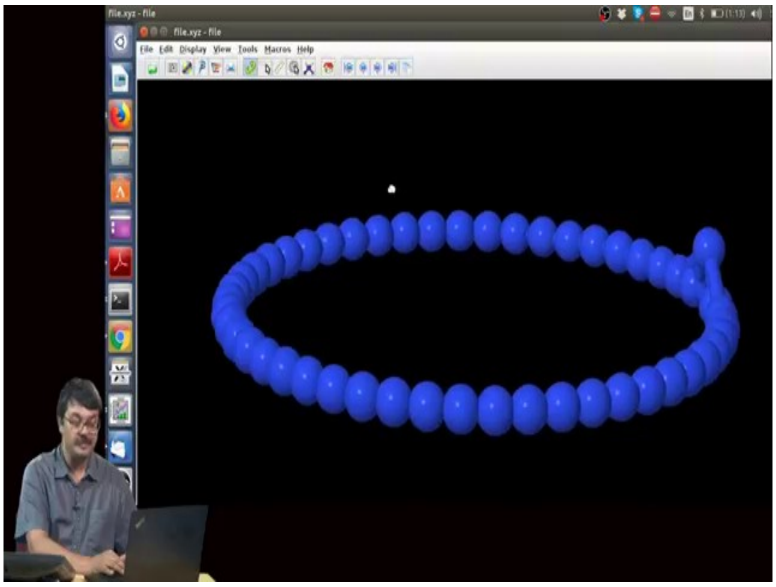
(Refer Slide Time: 03:50)



```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC
apratim@apratim-ThinkPad-T470: [illegible]$ ls                                                                _vl1.0_dt_20
a1                      kx_coor_dt_1030  kx_coor_dt_1330  kx_coor_dt_1630  kx_coor_dt_1930  kx_coor_dt_420      kx_coor_
a10                     kx_coor_dt_1040  kx_coor_dt_1340  kx_coor_dt_1640  kx_coor_dt_1940  kx_coor_dt_430      kx_coor_dt_710
a2                      kx_coor_dt_1050  kx_coor_dt_1350  kx_coor_dt_1650  kx_coor_dt_1950  kx_coor_dt_440      kx_coor_dt_720
a3                      kx_coor_dt_1060  kx_coor_dt_1360  kx_coor_dt_1660  kx_coor_dt_1960  kx_coor_dt_450      kx_coor_dt_730
a4                      kx_coor_dt_1070  kx_coor_dt_1370  kx_coor_dt_1670  kx_coor_dt_1970  kx_coor_dt_460      kx_coor_dt_740
a5                      kx_coor_dt_1080  kx_coor_dt_1380  kx_coor_dt_1680  kx_coor_dt_1980  kx_coor_dt_470      kx_coor_dt_750
a6                      kx_coor_dt_1090  kx_coor_dt_1390  kx_coor_dt_1690  kx_coor_dt_1990  kx_coor_dt_480      kx_coor_dt_760
a7                      kx_coor_dt_110   kx_coor_dt_140   kx_coor_dt_170   kx_coor_dt_20    kx_coor_dt_490      kx_coor_dt_770
a8                      kx_coor_dt_1100  kx_coor_dt_1400  kx_coor_dt_1700  kx_coor_dt_200   kx_coor_dt_50       kx_coor_dt_780
a9                      kx_coor_dt_1110  kx_coor_dt_1410  kx_coor_dt_1710  kx_coor_dt_2000  kx_coor_dt_500      kx_coor_dt_790
aa                      kx_coor_dt_1120  kx_coor_dt_1420  kx_coor_dt_1720  kx_coor_dt_210   kx_coor_dt_510      kx_coor_dt_80
aoupled                 kx_coor_dt_1130  kx_coor_dt_1430  kx_coor_dt_1730  kx_coor_dt_220   kx_coor_dt_520      kx_coor_dt_800
coupled_Diff_eqns.f90   kx_coor_dt_1140  kx_coor_dt_1440  kx_coor_dt_1740  kx_coor_dt_230   kx_coor_dt_530      kx_coor_dt_810
cile10.xyz              kx_coor_dt_1150  kx_coor_dt_1450  kx_coor_dt_1750  kx_coor_dt_240   kx_coor_dt_540      kx_coor_dt_820
file1.xyz               kx_coor_dt_1160  kx_coor_dt_1460  kx_coor_dt_1760  kx_coor_dt_250   kx_coor_dt_550      kx_coor_dt_830
file2.xyz               kx_coor_dt_1170  kx_coor_dt_1470  kx_coor_dt_1770  kx_coor_dt_260   kx_coor_dt_560      kx_coor_dt_840
file3.xyz               kx_coor_dt_1180  kx_coor_dt_1480  kx_coor_dt_1780  kx_coor_dt_270   kx_coor_dt_560.xyz  kx_coor_dt_850
file4.xyz               kx_coor_dt_1190  kx_coor_dt_1490  kx_coor_dt_1790  kx_coor_dt_280   kx_coor_dt_570      kx_coor_dt_860
file_50.xyz             kx_coor_dt_120   kx_coor_dt_150   kx_coor_dt_180   kx_coor_dt_290   kx_coor_dt_580      kx_coor_dt_870
file5.xyz               kx_coor_dt_1200  kx_coor_dt_1500  kx_coor_dt_1800  kx_coor_dt_30    kx_coor_dt_590      kx_coor_dt_880
file6.xyz               kx_coor_dt_1210  kx_coor_dt_1510  kx_coor_dt_1810  kx_coor_dt_300   kx_coor_dt_60       kx_coor_dt_890
file7.xyz               kx_coor_dt_1220  kx_coor_dt_1520  kx_coor_dt_1820  kx_coor_dt_310   kx_coor_dt_600      kx_coor_dt_90
file8.xyz               kx_coor_dt_1230  kx_coor_dt_1530  kx_coor_dt_1830  kx_coor_dt_320   kx_coor_dt_610      kx_coor_dt_900
file0.xyz               kx_coor_dt_1230  kx_coor_dt_1500  kx_coor_dt_1830  kx_coor_dt_320   kx_coor_dt_610      kx_coor_dt_910
file9.xyz               kx_coor_dt_1240  kx_coor_dt_1540  kx_coor_dt_1840  kx_coor_dt_330   kx_coor_dt_620      kx_coor_dt_920
fileA.xyz               kx_coor_dt_1250  kx_coor_dt_1550  kx_coor_dt_1850  kx_coor_dt_340   kx_coor_dt_630      kx_coor_dt_930
fileB.xyz               kx_coor_dt_1260  kx_coor_dt_1560  kx_coor_dt_1860  kx_coor_dt_350   kx_coor_dt_640      kx_coor_dt_940
file.xyz                kx_coor_dt_1270  kx_coor_dt_1580  kx_coor_dt_1880  kx_coor_dt_360   kx_coor_dt_650      kx_coor_dt_950
j.xyz                   kx_coor_dt_1280  kx_coor_dt_1590  kx_coor_dt_1890  kx_coor_dt_370   kx_coor_dt_660      kx_coor_dt_960
kx_coor_dt_10           kx_coor_dt_1290  kx_coor_dt_160   kx_coor_dt_190   kx_coor_dt_380   kx_coor_dt_670      kx_coor_dt_970
kx_coor_dt_100          kx_coor_dt_130   kx_coor_dt_1600  kx_coor_dt_1900  kx_coor_dt_390   kx_coor_dt_680      kx_coor_dt_980
kx_coor_dt_1000         kx_coor_dt_1300  kx_coor_dt_1610  kx_coor_dt_1910  kx_coor_dt_40    kx_coor_dt_690      kx_coor_dt_990
kx_coor_dt_1010         kx_coor_dt_1310  kx_coor_dt_1610  kx_coor_dt_1910  kx_coor_dt_400   kx_coor_dt_70       mov
kx_coor_dt_1020         kx_coor_dt_1320  kx_coor_dt_1620  kx_coor_dt_1920  kx_coor_dt_410   kx_coor_dt_700      x_energy_vl1.0_dt_20
apratim@apratim-ThinkPad-T470: [illegible]$ jmol file.xyz
```

So, what I am going to use is basically use a software called Jmol which is freely available and you can be installed in any Linux systems. So, here I have the position coordinates of all the 50 particles stored as a function of time in file dot xyz.

(Refer Slide Time: 04:18)



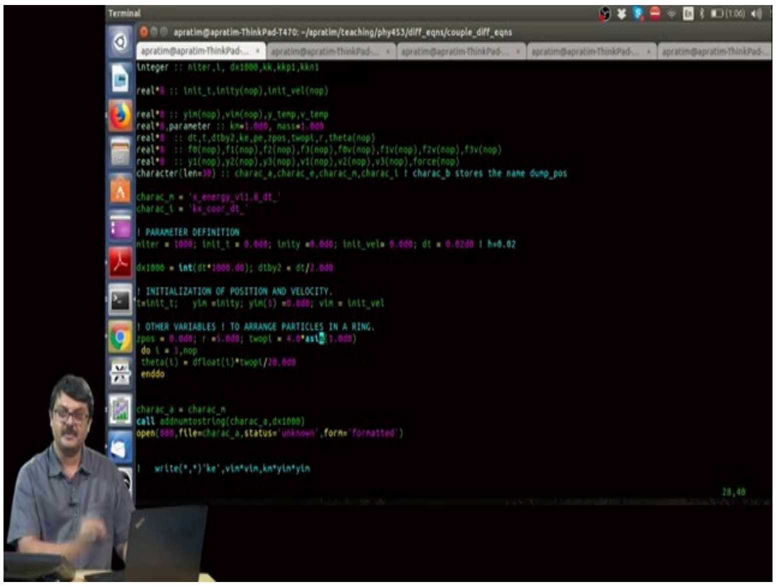
So, basically here you have 50 particles placed in a circle as r cos theta, r sin theta and this direction this direction is the y-direction and here you see that one particle has been displaced. So, the initial condition if you remember was that y is equal to 0 for all particles except one and suppose this is that 1th particle whose position has been displaced velocity has also 0 for all the particles and because of this displacement let us see what happens and the way I do it in this Jmol is go to tools, this animate and do it once.

So, now what you see is this particle had been vibrating because we have basically have a spring and what we have is this vibration moves along this arm and this arm and after some time all these particles coupled differential equations they start to move. I will just play this movie once again for you. So, here there was this particle which was displaced and it displaces the neighbouring particles, there is a force acting.

So, they are all doing basically simple harmonic motion in the y-direction and after some point of time the displacement reaches this point the other end after 25 particles and of course, you can ask questions like how much time would it take for this displacement for this perturbation to reach the other end; how can you change the speed of the wave propagating along this. So, this is this motion is the one which you basically Fourier transform and do your normal mode analysis. Here we are actually visualizing how 50 particles would oscillate right and you can change the frequency and so on so forth by changing Kappa, mass and so on and so forth right.

So, well and good. So, this is how you would have visualize 50 particles in the solution of 100 different differential equations ah. So, to make this movie so, now, what I am going to discuss is how will you land up making this movie. First of all I will show you the code itself. I understand it is bit involved though it looks simple, but I will just show you the code and then tell you in details how to land up to this movie.

(Refer Slide Time: 07:02)



```
integer :: niter,i, dx1000,kk,kkp1,kkn1

real*8 :: init_t,inity(nop),init_vel(nop)

real*8 :: yim(nop),vim(nop),y_temp,v_temp
real*8,parameter :: km=1.0d0, mass=1.0d0
real*8  :: dt,t,dtby2,ke,pe,zpos,twopi,r,theta(nop)
real*8  :: f0(nop),f1(nop),f2(nop),f3(nop),f0v(nop),f1v(nop),f2v(nop),f3v(nop)
real*8  :: y1(nop),y2(nop),y3(nop),v1(nop),v2(nop),v3(nop),force(nop)
character(len=30) :: charac_a,charac_e,charac_m,charac_l ! charac_b stores the name dump_pos

charac_m = 'x_energy_v11.0_dt_'
charac_l = 'kx_coor_dt_'

! PARAMETER DEFINITION
niter = 1000; init_t = 0.0d0; inity =0.0d0; init_vel= 0.0d0; dt = 0.02d0 ! h=0.02

dx1000 = int(dt*1000.d0); dtby2 = dt/2.0d0

! INITIALIZATION OF POSITION AND VELOCITY.
t=init_t;   yim =inity; yim(1) =0.0d0; vim = init_vel

! OTHER VARIABLES ! TO ARRANGE PARTICLES IN A RING.
zpos = 0.0d0; r =5.0d0; twopi = 4.0*asin(1.0d0)
 do i = 1,nop
 theta(i) = dfloat(i)*twopi/20.0d0
 enddo


charac_a = charac_m
call addnumtostring(charac_a,dx1000)
open(800,file=charac_a,status='unknown',form='formatted')


!   write(*,*)'ke',vim*vim,km*yim*yim
```

So, just; so, here is the program the program is called rk 4 coupled. Coupled for coupled differential equation and here the code it is a slightly different version. I have only 20 particles, but you can easily change it to 50, no big deal right. So, you have 20 particles in this version of the code arranged along a circle and the number of iterations you can choose and I have chosen number of iterations here to be 1000 and choose 10000s then you can watch the particles moving around for a longer period of time, look at long time behaviour.

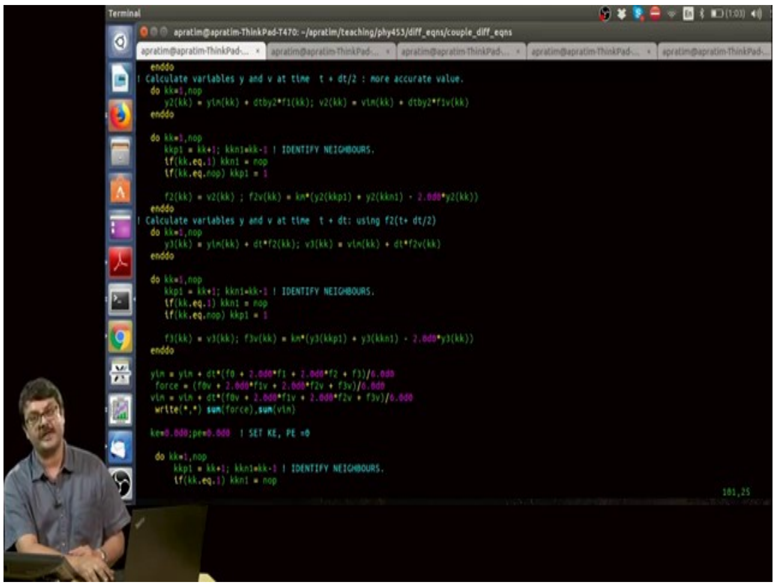
But, number of iterations I have chosen is 1000, h – the integration interval over which you integrate Runge-Kutta I have chosen to be 0.2 sorry, 0.02 I have denoted it by dt and what I have put in is that initial time t equal to 0, initial time equal to 0; initial displacement I have put it equal to 0, initial velocity equal to 0, the positions and velocities are stored in two arrays called vim and the length of these arrays is nop; nop is number of particles which is 20, km is 1, kappa is 1 and mass is 1. So, all those things have been chosen.

Then the initialization of position and velocity so, basically I say time equal to 0 init, then yim this that array all of them is equal to 0, inity has been defined to be 0 just here, but only the first particle yim the first particle, the displacement of the first particle in the y direction has been given to be 0.8, you can just give it to 0.3 as well does not matter. Initial velocity this is also an array remember vim is an array. So, all the 20 boxes in that array have been defined as init vel; init vel was 0 here.

For ease of calculation I have defined quantities like dt by 2 h by 2 which we use in our Runge-Kutta method right. I do not want to calculate this every time it takes computing power and here what I have done is basically to put particles in a circle. In a circle it is 2 pi; so, this 2 pi has been divided by 20. So, that basically you change theta 20 times and you have the position of each of these particles by taking r cos theta and r sin theta, no big deal. So, that is the x and z direction.

If you had 50 particles you would divide 2 pi by 50 and put the position of each of these particles 50 of them at theta equal to theta 1, theta 2, theta 3, theta 4 up till theta 50 which would complete the circle right fine. So, that is the basic initialization.

(Refer Slide Time: 10:27)


Terminal
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns

```
  enddo
! Calculate variables y and v at time  t + dt/2 : more accurate value.
  do kk=1,nop
     y2(kk) = yim(kk) + dtby2*f1(kk); v2(kk) = vim(kk) + dtby2*f1v(kk)
  enddo

  do kk=1,nop
     kkp1 = kk+1; kkn1=kk-1 ! IDENTIFY NEIGHBOURS.
     if(kk.eq.1) kkn1 = nop
     if(kk.eq.nop) kkp1 = 1

     f2(kk) = v2(kk) ; f2v(kk) = km*(y2(kkp1) + y2(kkn1) - 2.0d0*y2(kk))
  enddo
! Calculate variables y and v at time  t + dt: using f2(t+ dt/2)
  do kk=1,nop
     y3(kk) = yim(kk) + dt*f2(kk); v3(kk) = vim(kk) + dt*f2v(kk)
  enddo

  do kk=1,nop
     kkp1 = kk+1; kkn1=kk-1 ! IDENTIFY NEIGHBOURS.
     if(kk.eq.1) kkn1 = nop
     if(kk.eq.nop) kkp1 = 1

     f3(kk) = v3(kk); f3v(kk) = km*(y3(kkp1) + y3(kkn1) - 2.0d0*y3(kk))
  enddo

  yim = yim + dt*(f0 + 2.0d0*f1 + 2.0d0*f2 + f3)/6.0d0
   force = (f0v + 2.0d0*f1v + 2.0d0*f2v + f3v)/6.0d0
  vim = vim + dt*(f0v + 2.0d0*f1v + 2.0d0*f2v + f3v)/6.0d0
   write(*,*) sum(force),sum(vim)

  ke=0.0d0;pe=0.0d0  ! SET KE, PE =0

   do kk=1,nop
      kkp1 = kk+1; kkn1=kk-1 ! IDENTIFY NEIGHBOURS.
      if(kk.eq.1) kkn1 = nop
                                                                    101,25
```

Here I am calculating all the different values of theta i which will be there for 20 particles. Here is the actual code do i equal to 1 to niter; niter being number of iterations which I have chosen to be 1000. So, you choose it to be a 500, 10000 whatever time is being incremented by time dt right; dt will be 0.02 that is what I have chosen. And then what I am doing here there is this loop. So, from kk equal to 1 to number of particles that is exactly what we discussed right.

Loop number of particles so, you have to identify the neighbours. So, suppose kk is equivalent to I in my previous lecture so, i-th particle. So, here you are talking about the kk-th particle then the two neighbours on the two sides are kk plus 1 and kk minus here, but you have to be careful of the periodic boundary conditions. If kk equal to the first particle then the neighbour on the left is l, l is number of particles 20 here right. If kk equal to number of particles this is the last particle in that chain, then the right neighbour kk plus 1 is equal to 1 right that is how you do your periodic boundary condition. Here I am calculating f 0 kk for each of the particles the loop is over kk and that is f0 for the position it is dy dx equal to vim.

So, f 0 is nothing, but dy dt that is vim kk that is what I have written and this is for dv dt. So, you have to calculate f 0v which is nothing by km and that entire formula that we showed. This is the right neighbour, this is the left neighbour and this is yim kk itself. Then what do we do? Then we basically calculate y1 refer to your previous lectures if you have forgotten what it is y1 to be you update it by dtby2 right dt by 2 dt is the same as h, h by 2 and similarly you update v1 you update the position this is this temporary position and you update it so v1 is also updated by dt by 2.

So, you have updated the positions and the temporary velocities of all the particles, the temporary position and temporary velocity just to calculate the local slope after having incremented the position and the velocity or any variable. Here at these new values of y 1 and v1 we are calculating f1k and f 2 f 1k and basically f 1v right for all the particles. So, there is a loop. There is a loop over all the particles again we identify neighbours carefully keeping in mind periodic boundary condition using this f 1 we calculate y 2 and v2.

So, this is again some increment; increment by h by 2 after calculating that for each of the particle we calculate essentially the next one which is f 2 for each of the particles and f 2v for each of the particles, then we calculate the new positions and so on so forth right. So, this goes on and after calculating f 0, f 1, f 2 and f 3 for all the particles what we do is this is your actual position update those were only basically incrementing y1 to different values and then trying to calculate the averaging of these slopes.

So, this is the one where we actually update. So, this is the actual new position of the particles which is yim which was the initial velocity the beginning the dt into f 0 plus 2 f 0 f 1 2 f 2 and f 3 except remember that f 3, f 2, f 1, f 0 all of these are arrays and similarly yim is an array right. So, Fortran allows you to sum all the elements of the array like this and you can even calculate the force. What is the total force? You can calculate acting on each of these particles force is also an array and then this is an update the new velocities the other set of differential equations where all of these are of course, arrays, vim is an array.

So, what do you have? You have the position and the velocity of each of the particles after time h right which are stored again in yim and vim. So, using that we can now calculate the new position and velocities time 0 plus 2h right and this entire process will go on in a loop depending upon number of iterations fine. But, to see this data we need to save the positions and the velocities at each time to be able to see the movie right ok.

So, even before saving the positions what I have done is basically calculate the kinetic energy for each of these particles here KE equal to KE plus half into vim into vim half mv square for each of the particles and similarly, I have calculated the potential energy for each of these particles and basically write down as a function of time the total energy, the kinetic energy and the potential energy of all the particles and you know that the total energy has to be conserved. It has to be absolutely straight with an error of 0.02 to the power of 4 rk 4 right.

So, this is just for a cross check that you are maintaining your energy momentum conservation. You can similarly check for the momentum of the particles and that has to be should be 0 right. And after that what we are doing is using so, what we want is that we want to save the position of all the particles at every say 10 steps there is no point in saving the position of all the particles every step because in one step after one iteration h equal to 0.02 the particles would have moved relatively little.

So, to able to visualize the motion you do not want to basically store the position every time step you might you may if you want to, but it would make more sense to basically store the position of the particles every 10 iterations the you can see the particle movement when you make it in a movie. So, what you want is each the position of each of the particles should be saved in a file with a time label right.

So, suppose you say that your file name primary name is suppose position and then after 10 iterations you want to save it you want to add the word 10 to the character variable position and say the position of all the particles are saved after 10 iterations in the file called position dot 10 ok. Similarly, the position of all the particles after 20 iterations are saved in a file say called position dot 20 underscore 20. The position of all the particles after 30 iterations are saved in a file called position dot underscore 30.

So, then if you have the position of all the particles at different instances and time then these data you can all join them in one file and you can play the movie. So, the question is how do you do it? For that you need the position of all these particles and I will tell you how to first of all add suppose any string is there any character variable position and how do you add an integer 10, 20, 30, 40 so on so forth.

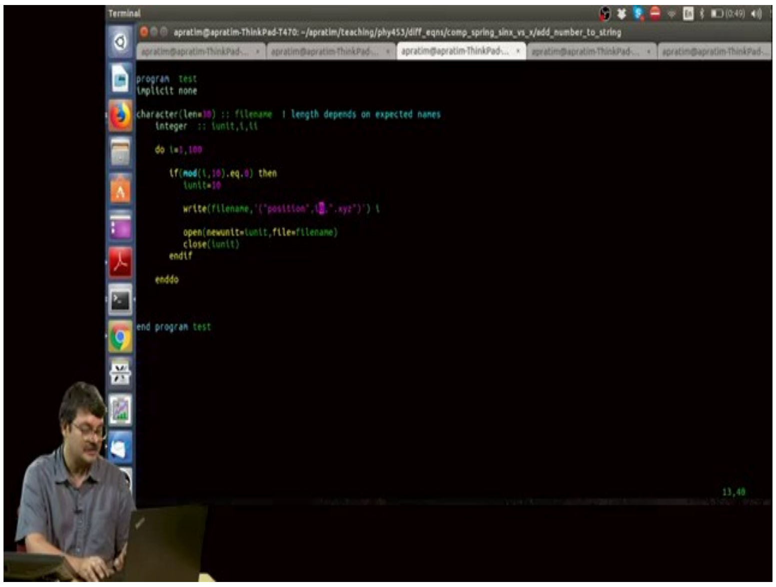
And here what I am doing is basically writing down at every instant in time I am opening a new file ok. So, file equal character a, where the value of character a shall keep on changing at one time it will at after 10 iterations every position underscore 10; after 20 iterations the value of character a will take position underscore 20; after 30 iterations the value of the character a is a character variable string variable will be positioned underscore 30.

So, every time though I am using the same loop a file will be opened with a different name and within that I am writing down N; N actually stands for nitrogen because the software that I will be using Jmol basically it is a chemistry software, so, it needs to be told which atom you want to plot. For us we do not care about atoms we just care for the sphere right or a point particle. So, N one has to put just to say nitrogen you can put in oxygen it will just be denoted by a different colour.

Then, r cos theta r sin theta corresponding to x and z and those will be unchanged with respect to time, but the so, the loop is over number of particles. For each particle the value of this and this will be different, but the y this value for each particle will change as a function of time right. So, as you saw in the movie the position of all the particles along the circle remain the same, but all the displacement was only perpendicular to the plane of the circle.

So, this that is why this quantity and this quantity will remain fixed as a function of time whereas, yim will changed as what you are calculating is a function of time anyway. And similarly if you want you can also write in the velocity then you close the file. The question is how did I do this addition of a number 10, 20, 30, 40 to a some variable string variable say position just to show that part. So, that part I do here in subroutine. I will not show you the subroutine, but I will teach you how to do that.

(Refer Slide Time: 22:44)



apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/comp_spring_sinx_vs_x/add_number_to_string

```
program  test
implicit none

character(len=30) :: filename  ! length depends on expected names
   integer  :: iunit,i,ii

   do i=1,100

      if(mod(i,10).eq.0) then
         iunit=10

         write(filename,'("position",i0,".xyz")') i

         open(newunit=iunit,file=filename)
         close(iunit)
      endif

   enddo


end program test
```

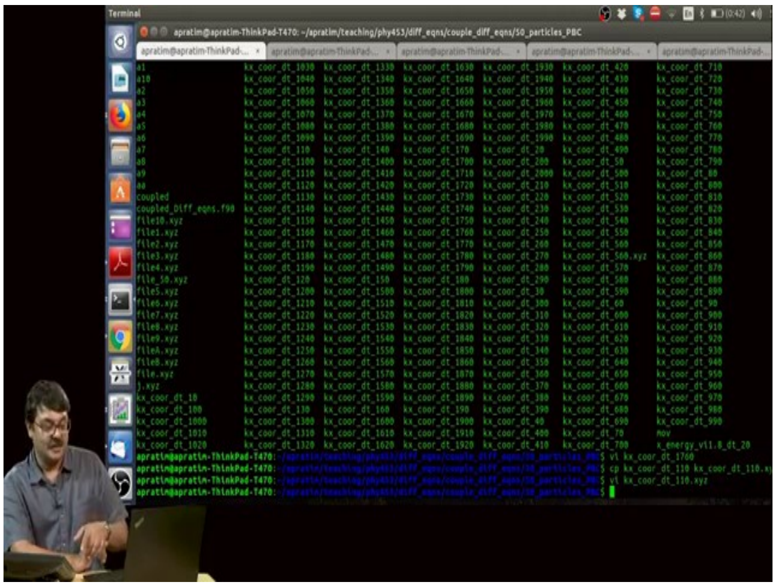
13,40

And, that is done in, so this is a test program which you should try out and then the same lines you can just put in a subroutine called add number to string right and each time its variable, the character variable will be different. So, here you have a character variable called file name, you can call it character a does not matter called filename and that is of length 30 right and suppose do i equal to 1 to 100 this is equivalent to number of iterations.

If mod i, 10; so, i keeps on changing from 1 to 100 or 1 to 1000 or 1 to 10000 number of iterations and every time that mod i, comma 10 equal to 0, that will happen you every 10 steps then you say i unit equal to 10 ok. And if you write this line write filename and whatever you put; so, here I am teaching you the method whatever you put between this double comma and this double comma and here this is i 0 you write it as i 0 this stands for integer and you are here you are writing i right this is what was changing.

So, only if this enter this line only every 10 steps right and if you run this you will see that you will get a variable called position value of i 10, 20, 30, 40, dot xyz right. So, here you have written position then i0 here after a comma and this value, different values of i will be written at this position right and then dot xyz will be added. I have already run it, you can have a look.

(Refer Slide Time: 24:56)



```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC
a1                     kx_coor_dt_1030  kx_coor_dt_1330  kx_coor_dt_1630  kx_coor_dt_1930  kx_coor_dt_420      kx_coor_dt_710
a10                    kx_coor_dt_1040  kx_coor_dt_1340  kx_coor_dt_1640  kx_coor_dt_1940  kx_coor_dt_430      kx_coor_dt_720
a2                     kx_coor_dt_1050  kx_coor_dt_1350  kx_coor_dt_1650  kx_coor_dt_1950  kx_coor_dt_440      kx_coor_dt_730
a3                     kx_coor_dt_1060  kx_coor_dt_1360  kx_coor_dt_1660  kx_coor_dt_1960  kx_coor_dt_450      kx_coor_dt_740
a4                     kx_coor_dt_1070  kx_coor_dt_1370  kx_coor_dt_1670  kx_coor_dt_1970  kx_coor_dt_460      kx_coor_dt_750
a5                     kx_coor_dt_1080  kx_coor_dt_1380  kx_coor_dt_1680  kx_coor_dt_1980  kx_coor_dt_470      kx_coor_dt_760
a6                     kx_coor_dt_1090  kx_coor_dt_1390  kx_coor_dt_1690  kx_coor_dt_1990  kx_coor_dt_480      kx_coor_dt_770
a7                     kx_coor_dt_110   kx_coor_dt_140   kx_coor_dt_170   kx_coor_dt_20    kx_coor_dt_490      kx_coor_dt_780
a8                     kx_coor_dt_1100  kx_coor_dt_1400  kx_coor_dt_1700  kx_coor_dt_200   kx_coor_dt_50       kx_coor_dt_790
a9                     kx_coor_dt_1110  kx_coor_dt_1410  kx_coor_dt_1710  kx_coor_dt_2000  kx_coor_dt_500      kx_coor_dt_80
aa                     kx_coor_dt_1120  kx_coor_dt_1420  kx_coor_dt_1720  kx_coor_dt_210   kx_coor_dt_510      kx_coor_dt_800
coupled                kx_coor_dt_1130  kx_coor_dt_1430  kx_coor_dt_1730  kx_coor_dt_220   kx_coor_dt_520      kx_coor_dt_810
coupled_Diff_eqns.f90  kx_coor_dt_1140  kx_coor_dt_1440  kx_coor_dt_1740  kx_coor_dt_230   kx_coor_dt_530      kx_coor_dt_820
file10.xyz             kx_coor_dt_1150  kx_coor_dt_1450  kx_coor_dt_1750  kx_coor_dt_240   kx_coor_dt_540      kx_coor_dt_830
file1.xyz              kx_coor_dt_1160  kx_coor_dt_1460  kx_coor_dt_1760  kx_coor_dt_250   kx_coor_dt_550      kx_coor_dt_840
file2.xyz              kx_coor_dt_1170  kx_coor_dt_1470  kx_coor_dt_1770  kx_coor_dt_260   kx_coor_dt_560      kx_coor_dt_850
file3.xyz              kx_coor_dt_1180  kx_coor_dt_1480  kx_coor_dt_1780  kx_coor_dt_270   kx_coor_dt_560.xyz  kx_coor_dt_860
file4.xyz              kx_coor_dt_1190  kx_coor_dt_1490  kx_coor_dt_1790  kx_coor_dt_280   kx_coor_dt_570      kx_coor_dt_870
file_50.xyz            kx_coor_dt_120   kx_coor_dt_150   kx_coor_dt_180   kx_coor_dt_290   kx_coor_dt_580      kx_coor_dt_880
file5.xyz              kx_coor_dt_1200  kx_coor_dt_1500  kx_coor_dt_1800  kx_coor_dt_30    kx_coor_dt_590      kx_coor_dt_890
file6.xyz              kx_coor_dt_1210  kx_coor_dt_1510  kx_coor_dt_1810  kx_coor_dt_300   kx_coor_dt_60       kx_coor_dt_90
file7.xyz              kx_coor_dt_1220  kx_coor_dt_1520  kx_coor_dt_1820  kx_coor_dt_310   kx_coor_dt_600      kx_coor_dt_900
file8.xyz              kx_coor_dt_1230  kx_coor_dt_1530  kx_coor_dt_1830  kx_coor_dt_320   kx_coor_dt_610      kx_coor_dt_910
file9.xyz              kx_coor_dt_1240  kx_coor_dt_1540  kx_coor_dt_1840  kx_coor_dt_330   kx_coor_dt_620      kx_coor_dt_920
fileA.xyz              kx_coor_dt_1250  kx_coor_dt_1550  kx_coor_dt_1850  kx_coor_dt_340   kx_coor_dt_630      kx_coor_dt_930
fileB.xyz              kx_coor_dt_1260  kx_coor_dt_1560  kx_coor_dt_1860  kx_coor_dt_350   kx_coor_dt_640      kx_coor_dt_940
file.xyz               kx_coor_dt_1270  kx_coor_dt_1570  kx_coor_dt_1870  kx_coor_dt_360   kx_coor_dt_650      kx_coor_dt_950
j.xyz                  kx_coor_dt_1280  kx_coor_dt_1580  kx_coor_dt_1880  kx_coor_dt_370   kx_coor_dt_660      kx_coor_dt_960
kx_coor_dt_10          kx_coor_dt_1290  kx_coor_dt_1590  kx_coor_dt_1890  kx_coor_dt_380   kx_coor_dt_670      kx_coor_dt_970
kx_coor_dt_100         kx_coor_dt_130   kx_coor_dt_160   kx_coor_dt_190   kx_coor_dt_390   kx_coor_dt_680      kx_coor_dt_980
kx_coor_dt_1000        kx_coor_dt_1300  kx_coor_dt_1600  kx_coor_dt_1900  kx_coor_dt_40    kx_coor_dt_690      kx_coor_dt_990
kx_coor_dt_1010        kx_coor_dt_1310  kx_coor_dt_1610  kx_coor_dt_1910  kx_coor_dt_400   kx_coor_dt_70       mov
kx_coor_dt_1020        kx_coor_dt_1320  kx_coor_dt_1620  kx_coor_dt_1920  kx_coor_dt_410   kx_coor_dt_700      x_energy_v11.8_dt_20
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC$ vi kx_coor_dt_1760
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC$ cp kx_coor_dt_110 kx_coor_dt_110.xy
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC$ vi kx_coor_dt_110.xyz
apratim@apratim-ThinkPad-T470:~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC$
```

So, basically if you; so basically if you run it you will have position 10 dot xyz, position 20 dot xyz, position 30 dot xyz and so on so forth. And if you had given i equal to 1000, then you would have hundred files with these different names each with an ending dot xyz right. So, that is how you add numbers to strings. That is the general principle if you did not have that dot xyz at the end then that would not have been added we would have got only position 10 or 20.

And if you had put in an underscore such as like this, let us compile. Now, you have some other files with position underscore 50, position underscore 60, position underscore 10 and so on so forth right. So, you can even put in names at the end. Why I did xyz I will explain in a second right. Basically, Jmol the software Jmol with which I made the movie accepts files or recognizes files only if it ends in dot xyz ok.

So, position underscore 10 gives you basically the time index it tells you that you are storing the position at time 10, 20, 30, 40, 50 and your valid dot xyz because or it will allow the software called Jmol to basically to recognize the files and it can be uploaded. So, what you do is essentially so, in the couple differential equation I have basically generated these coordinates. Just let me go to 50 particle, it has just the same thing it has the same kind of names and I am storing the position of these 50 particles in various instances of time right.

So, you here you see 10, 100, 1000, 1010, 1020 here 110 and so on so forth. And of course, I have times at time 120 and time 20 here and 30 here and 40 here and so on so forth. What is inside the files? See, here is the nitrogen, these are the position this is the x-coordinate, this is the z-coordinate and this is the y-coordinate which keeps on changing after each times. So, these still first two columns sorry first three columns including nitrogen they will remain the same, but this is the this column the fourth column will keep on changing at different times that will have different values ok.

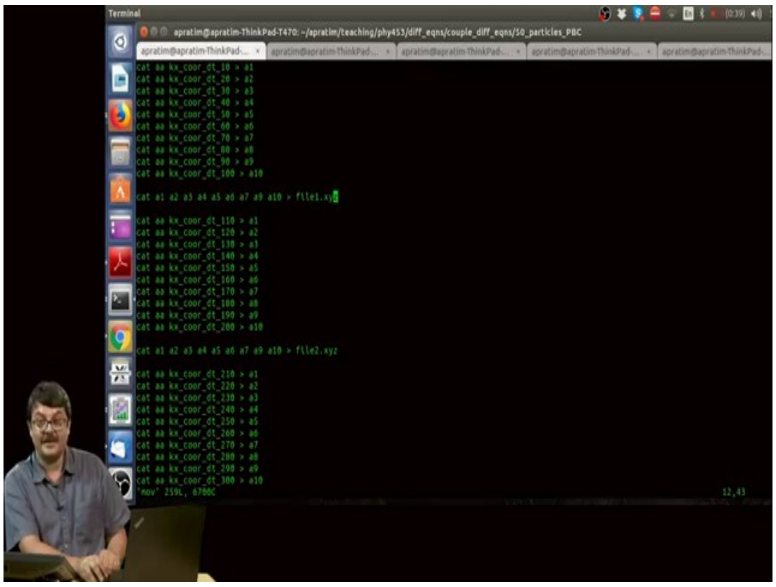
Now, what one needs to do you have these files at different instances of time, now they have to be basically added together into make one big file. How do one do that? One can do it mechanically, but there is another subtlety here if you want to make the movie basically what first of all you have to make these into xyz.

So, the Jmol will not be able to recognize a file like this, you have to name it something like xyz and you must also tell Jmol how many particles are there because the Jmol the software does not know; does not know that you have run the code with 50 particles you could have 50 particles, 20 particles, 100 particles it does not know. So, what has to be done is you write that there are 50 particles. So, the total length of this file becomes 52 right N vi.

So, this is 50 particles give a gap this is important and then you have the position and coordinates of 50 particles then each of these files can be added up into one big file which I have done. So, which you have to do it for each of the is at the top of each of these files you have to write 50. Now, if you doing it for 10 – 20 files doing it manually is possible though boring.

But, if you have to do it for 1000 files you want to add them together you really do not want to do it manually that is open every file, write 50 on the top then save it then go to the next file write 50 at the top, save it. It is too boring; you can mechanize it at least in Linux you can mechanize this entire step. What I have done is basically write a file called aa where the you have nothing except 50 and a blank line.

(Refer Slide Time: 31:13)



```
apratim@apratim-ThinkPad-T470: ~/apratim/teaching/phy453/diff_eqns/couple_diff_eqns/50_particles_PBC
cat aa kx_coor_dt_10 > a1
cat aa kx_coor_dt_20 > a2
cat aa kx_coor_dt_30 > a3
cat aa kx_coor_dt_40 > a4
cat aa kx_coor_dt_50 > a5
cat aa kx_coor_dt_60 > a6
cat aa kx_coor_dt_70 > a7
cat aa kx_coor_dt_80 > a8
cat aa kx_coor_dt_90 > a9
cat aa kx_coor_dt_100 > a10

cat a1 a2 a3 a4 a5 a6 a7 a9 a10 > file1.xyz

cat aa kx_coor_dt_110 > a1
cat aa kx_coor_dt_120 > a2
cat aa kx_coor_dt_130 > a3
cat aa kx_coor_dt_140 > a4
cat aa kx_coor_dt_150 > a5
cat aa kx_coor_dt_160 > a6
cat aa kx_coor_dt_170 > a7
cat aa kx_coor_dt_180 > a8
cat aa kx_coor_dt_190 > a9
cat aa kx_coor_dt_200 > a10

cat a1 a2 a3 a4 a5 a6 a7 a9 a10 > file2.xyz

cat aa kx_coor_dt_210 > a1
cat aa kx_coor_dt_220 > a2
cat aa kx_coor_dt_230 > a3
cat aa kx_coor_dt_240 > a4
cat aa kx_coor_dt_250 > a5
cat aa kx_coor_dt_260 > a6
cat aa kx_coor_dt_270 > a7
cat aa kx_coor_dt_280 > a8
cat aa kx_coor_dt_290 > a9
cat aa kx_coor_dt_300 > a10
"mov" 259L, 6700C                                                    12,43
```

Then, using the cat command of Unix I am adding. So, please look up what is the cat command in Unix cat aa then this file where after the position of all these particles and nitrogen written so, I am adding cat in cat stands for concatenate. So, add aa or rather cat aa and this and store it in file a1 then cat aa and second position and stored it in file a2, then you do it for the third and the fourth and the fifth and the sixth and the tenth file which is stored in a10 then you cat a1, a2, a3, a4 all of these in file1 dot xyz right. Similarly, for 110 to 120 you keep on adding aa to each of these files try it out what cat does and then again concatenate all of these 10 files to file2 dot xyz

So, what does file1 and file2 etcetera have? So, just to show you; so, let us look at what does a1, a2, a3, a4 have, vi a1; a1 it has been concatenated and this two has been mechanically added. You do not have to do it each file by file you just write that file and run it you will get a1 and similarly, you have a2. So, this is 50, but this has the y coordinate at time 20 and this has the y coordinate at time 30 and so on so forth right.

Now, if you remember I was adding each of these files and writing it in file1 dot xyz. What does file1 dot xyz have? It has 50, but after that it has again. So, this is basically the position at time 10, from here you have added a2. So, this is basically the position at time 20 and this is the position at time 30 and this is the position at time 40. So, where you have basically added all of individual each of these files a1, a2, a3, a4 a5 right and created file1 dot xyz. Similarly, you have created file2 dot xyz which is basically the position of all these files from time 110 to 200 and then file3 dot xyz we will have the position of all the particles from time 210 to 300 and so on so forth right.

So, you have basically file1 so, basically you have all these different file1, file2, file3, file4 and you can now concatenate all of them this file1 to file1 dot xyz file2 dot xyz you can concatenate all of them at the end concatenate all of them into fileB say or fileA so this is one file right and this is the file. So, how do you run Jmol? You type in jmol and whichever file has all the movies over say 10000 iterations so, that has all been made into one big file and Jmol file dot xyz.

So, this in my case it is file I have just named it file when I previously prepared it right you can whatever name. So, this is a concatenation of file1, file2, file3, file4 and so on so forth. Jmol file dot xyz and if you do this just type this in you get this movie and then you just rotate it go to Tools – Animate – Once and you have your movie. What is the message? Even if you had 100 particles you could automate this entire step of concatenating of storing the files, concatenating them and see it in Jmol you can visualize your data in a movie.

Of course, for quantitative analysis you also need the position of y for each of these particles as a function of time, you could also choose to write it not that you write the position of all the particles at a particular time. The other hand you could also say that in one file I shall write the position of one particle as a function of time right, then also you can also do that in a very similar way that I wrote where basically suppose pos1 would dot dat instead of dot xyz you would have pos position of first particle 1 for first particle dot dat as a function of time.

So, you can just use that add num to string and so, it depends upon which number you want to add and you can also save the position of each particle in a different file as a function time. At the moment what I have done is written down the position of all the particles at a particular time in a file depending upon what analysis you want to do, what do you want to do with the data you can also write it the other way right.

What I am teaching you is how to solve the differential equation so that you have x rather y, all the particles as a function of time and your physics of the problem will tell you; will tell you what you want to do with the solution right. And basically this will take a bit of time for you to get adjusted because the code was involved after that you take the data, do some processing to do it to plot it in movie called Jmol just install it, it will take some time, but you will get used to it.

So, if you have further questions you can always ask the TA. So, I decided to add a bit of addendum which will help you to make the movies to explain the various commands that I used to compile the file which was used to finally, plot the movie in Jmol. So, first of all I will explicitly show you the commands of cat.

So, suppose you have two files suppose f1 which has suppose aa written and bb whatever written and another file called f2 which has some numbers written right. It does not really matter and what cat does is if you cat f1 and f2 basically what it does is write down the contents of file f1 and then the contents of file f2. So, here I made a mistake, but here if you see it has written down the contents of file f1 and here this is the contents of file f2. Now, if you cat f1 and f2 and save it in some other file say f3, then you do not get the output on the screen instead if you open f3 you will see that the contents of both the files have been concatenated and saved in f3.

And this is exactly what I am using in this file called mov. So, what I have done here is cat concatenate aa and just to remind you that aa contains the total number of particles in the simulation which is just 50 and a blank line and nothing else and mov and move the this file and this file say basically saves the coordinates of all these 50 particles at different times at times 10, 20, 30, 40, 50 and so on so forth right up till the length of the movie and this cat command is basically adding 50 and a blank line to the position file and saving it in a1 and then saving it in a2 and a3 and a4 and all these commands have been written in a file called mov right.

So, in mov has all these lines and then basically a1, a2, a3, a4 etcetera are themselves being concatenated and saved in a file called file1 dot xyz and then from 110 to 200 again I am saving them in a1, a2, a3, a4, a5 to a10 and this in turn is being saved. So, basically the positions of all the particles from time 110 to 200 is being saved in a file called file2 dot xyz and the position of all the particles from time 210 to time 300 are being saved in a file called file 3xyz and so on so forth and right at the end what I am doing is concatenating file1 dot xyz, file2 dot xyz and all of these into a file called fileB dot xyz.

So, basically fileB has all the positions in the right format of all the particles from right from time t equal to 10 to time t till say time t equal to 2000 and so, all that those have been concatenated appropriately right. And then one can use fileB dot xyz or even to basically plot the movie. Just to show you or what does fileB dot xyz contain, it has this 50 blank the nitrogen which is already I am saving in the code and the positions and if you go to 50 lines after 50 lines you again have 50 and so on so forth.

Now, so, all these commands have been written in this file called mov right ah, but this is not an executable file. So, if I just make it executable by writing chmod plus x this is a Unix command mov. So, from this text file in mov it becomes a executable file right and now I can execute it. It is only after this step that I can execute it and it will automatically create fileB dot xyz and fileA dot xyz which I can now upload and make the movie and see the movie in Jmol right.

So, basically what I have done instead of type so many commands which I have written in mov type it every time I have saved them in a file called mov, made it executable and I just type mov and it gives me the file which can be used as a movie. Now, if I change the condition if I change the solution of the differential equation and appropriately suppose I give it a velocity and so on and so forth, I will have a different set of coordinates at different times in just one step this dot mov I can make the corresponding movie file dot xyz and plot it ok. So, this is I thought I would just add a primer to whatever is shown in the main lecture.

Thanks.