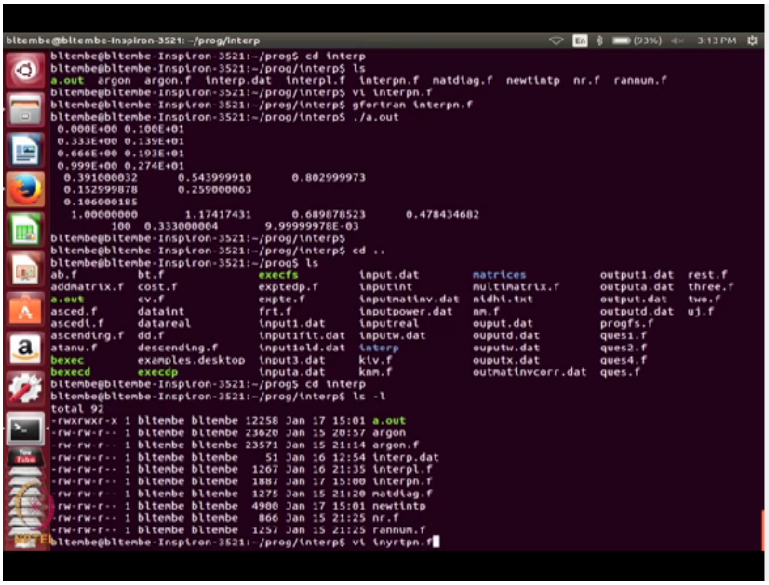
Computational Chemistry & Classical Molecular Dynamics
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology - Bombay

Lecture – 15
Interpolation Methods 2: Newton's and Lagrange Interpolation

Hello and welcome to this next lecture in our computational chemistry course. In this lecture, I am going to demonstrate the 2 interpolating methods. One is the Newton's interpolation and the next one is the Lagrange interpolation. We have discussed the formula in the last lecture. So I will not go too much into the formula again. Let us directly start with our execution of the program. So now you look at our screen.

(Refer Slide Time: 00:46)



On our screen, I am in a directory, already in an interpolation directory. You look at the last lines here, interp. I am in the interpolation directory. So why did I create the directory? Let me go back to the previous directory, cd.., it makes me go to the previous directory and I do ls. ls command, list me all the files in that program. See there are so many files in that directory. So when you have so many files, it is impossible to keep track of which file is where.

So in order to get around that problem, I have created 2 directories, one is interp directory and the other one is matrices directory. In the next lecture, I will be discussing matrices. So I have created a separate directory where all the files related to matrices are there. The present directory

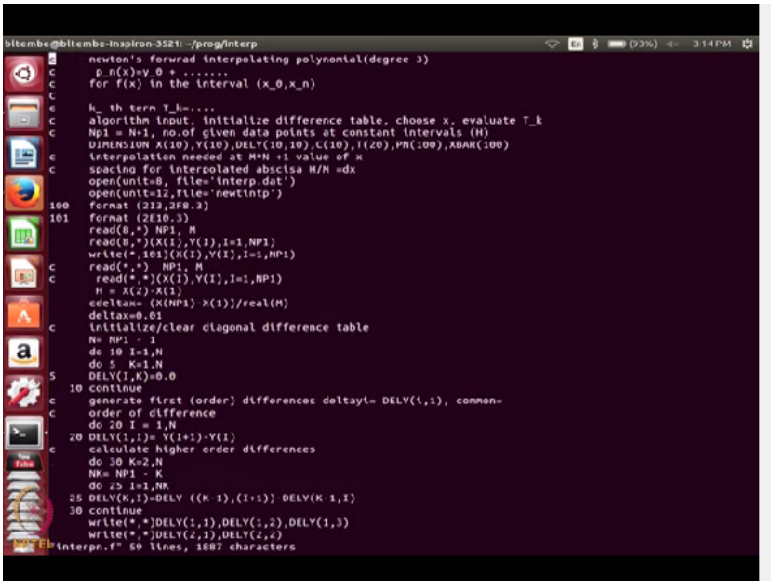
has become very cluttered. And what are these a.out, bexec, execdp, execfs, these were the files that are executable files which we use to execute our earlier programs. Remember a.out is the default name for the executable file.

Whereas if you want instead of a.out, some other name, you have to change your gfortran compiler statement. When you change the compiler statement, you can write your executable file with different names such as execfs, execdp, execcd, execc, as well as a.out. You can change the names. So now I have everything in my interp directory. So I will go to that directory, cd interp. So this is the directory and I have already gone to that directory.

Now let us do ls -l. ls -l that is ls space -l. Remember the space. If there is no space it will give an error. Ls -l, so it will give me all the files in the directory. The details it will give you, okay. There are several files. Of interest today are interpl.f. This is a Lagrange interpolating polynomial file and interpn.f, this is the Newton's interpolation program. .f because it is an extension, it is a Fortran program.

So I will go to this interpn.f. Then we will look at all the details, compile and execute, okay. Then there are some data files as well. What are those data files? interp.dat, that is one data file. newtintp, this is a data file which is the output of that program. So let me edit this particular file interpn.f. What is my command. vi interpn.f. So this is my Newton's interpolation program. I have made some mistake. So let me correct it, interpn.f, so I go to this.

(Refer Slide Time: 03:50)



So I have gone to this program. So whenever I use that vi command, I am in the edit mode. If I want to insert thing, I have to type i and then it will start inserting, okay. So now let us look at this program again. I have looked at some lines in the last lecture already. So first 6 or 7 are comment cards. It just say Newton's interpolating polynomial of degree 3. Pn2=y0 and so on. Then for function fx in the interval x0 to xn, this is my interval.

Interpolation polynomial is always in the interval between the starting value of x, x0 and the last value which is xn, okay. So what is my algorithm? Read all the input, initialize the difference table, chose x and evaluate the polynomial for a large number of data points at a constant interval, okay. Now this particular interpolation method, there are n+1 data points and these points are at a constant interval difference, okay.

So now this dimension statement, X10, so there are 10 data point to be read, okay. So this is a one dimensional array with 10 variables in that dimension. So Y10, again it is a one dimensional array with 10 values. Then del Y is my difference table. Remember we discussed first order difference, second order, third order. So I collect all that in my difference table. Then C is my coefficient table.

This T, PN and X bar are all additional variables which I will not use right away, okay. So now go to this open unit=8, file=interpole .dat, interp.dat, this is my data file and 12, open 12,

file='newtintp'). Remember in an open statement, open (unit=12, file=', then the name of the file, '). Any error will give you problems, okay. Now so then I have 2 format statements, okay. And I tried to execute this with format statements.

There are some problems. So I have avoided the format in the read statements. So what am I going to read in this program. Read 8,*NP1, M. NP1 was 1+N, NP1. So if there are polynomial of order 3, then NP1 will be 4. 4 data points and M is the number of points at which I want the interpolating polynomial. This N can be any number. So these 2 I am going to read from my file 8.

* means there is no specific format, okay. Then read 8,*XiYi, i going from 1 to NP1. In the last lecture I mentioned that whenever I give in bracket, okay, Xi,Yi, i going from 1,NP1. It is an implicit do loop. It is as if I am writing a do loop and in that do loop, for each value of i, I am going to read Xi and Yi. So this, I would have done do some 10 i going from 1 to NP1. Read Xi Yi and 10 continue.

This is how I would be reading in the do loop. So this is the same as a do loop. Then I am going to write, whatever I read, I am going to write on the screen. So write*,101 Xi,Yi i going from 1 to NP1. Now 101 is a format statement. What is that format statement? Look at this 101 format 2E10.3. What is this 2E10.3? E10.3 means it is a format, E format 3 letters after the decimal point. 0.3 means 3 letters are the decimal points.

Total space will be 10 values, decimal point takes 1 point, so there will be 6 values to write the number on the left side of the decimal. So there are 2 numbers. So 2E10.3. On each line, it will read, it will write 2 values. So I have read Xi Yi and I have written Xi Yi, on the screen, I want to write now. Where is my write statement? Write*,101, I write so that what is the advantage of this?

It knows what my data are. Because if I do not do this, suppose I read not from interpolate.dat but from other file. It will be interpolating with the data that it reads, okay. So it is always good to write on the screen or into some other file whatever data you are using, okay. So this is my

write statement. Now the next 2 are comment cards. They are not relevant. Now I want to define which is H.

H is the spacing. Spacing between adjacent point. H=X2-X1. Remember our data was in the class 0.33 0.66, 0.99, I have. So for the difference I can take the difference between any 2 adjacent points, okay, any 2 adjacent points. And now I want to determine at how many points I want to interpolate. Remember I had read this M. M is the number of points at which I want to interpolate.

So how do I get the values for the X. So the delta X will be the final value of X-the first value/real M. What is the meaning of real M? Suppose M is 100, real M will be 100.0. This real function converts an integer into a real number. So I want to always divide real number by a real number. It is always good to convert. The computer may still do it right but it is always good to use the variables in the exact type that you are going to write the program for.

So now I want to initialize the diagonal difference table. What is the difference table? Difference between the value of the function at a new point-the value of the function at an earlier point, okay. So I am going to diagonalize a difference table. So if there are n+1 data points, the difference table will be 1 less than that. What I mean is? Suppose I have X0, X1, X2, X3. The difference which will be X1-X0, that is fx1-fx0, fx2-fx1, fx3-fx2.

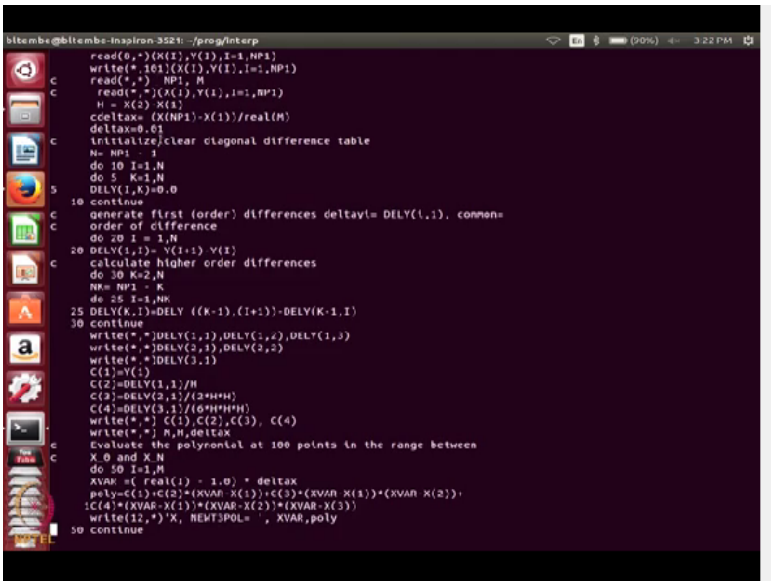
There will be only 3 differences. So therefore, that N is NP1-1. So if NP1 is 4, I have subtracted 1, so I have N=3 now. So now I am going to calculate the difference table. So before I calculate the difference table, it is always a good idea to initialize all my values to 0. So I want to calculate this differences but before I want to calculate, I want to set all of them to 0 so that del YiK=0, this is my initialization.

This is always called initialization. It is always a good practice to initialize your variables, okay. So now I will generate the first order difference table. So now look at that. Where does it start. Now do 20, i going from 1 to N. So this is my difference table. Del Y 1,i will be Yi+1-Yi. So del Y, the first number is 1. That means it is a first order difference. First order difference when i=1,

it will take Y2-Y1.

So this is the first order difference at the first point. At the second point, when i=2, this is Y3-Y2. So this is the first order difference at the second point. i=3, first order difference at the third point. So in del Y, the first variable refers to the order of difference. So through this particular loop, I have calculated all the 3 first order differences. So once I have the first order differences, now I can calculate the next differences. There are higher order differences, okay. So what are those?

(Refer Slide Time: 12:33)



Second order and third order differences. How will I calculate now? Look at this do loop. So since the first order differences have already been calculated, in the new differences, the first variable in this del Y, the first variable in del Y cannot be 1 anymore because I have already calculated the first order differences. So now I need to calculate the second and third order. So my do loop will be do 30 K going from 2 to N.

2 will be the second order difference. 3 will be the third order difference and so on. So second order and so on. So for higher order differences, there will be less differences than the starting one. For the first order differences, there were 2, there were 3. There will be only 2 second order differences. So how do I ensure there are 2 second order differences? When K=2, that NK is NP1-K.

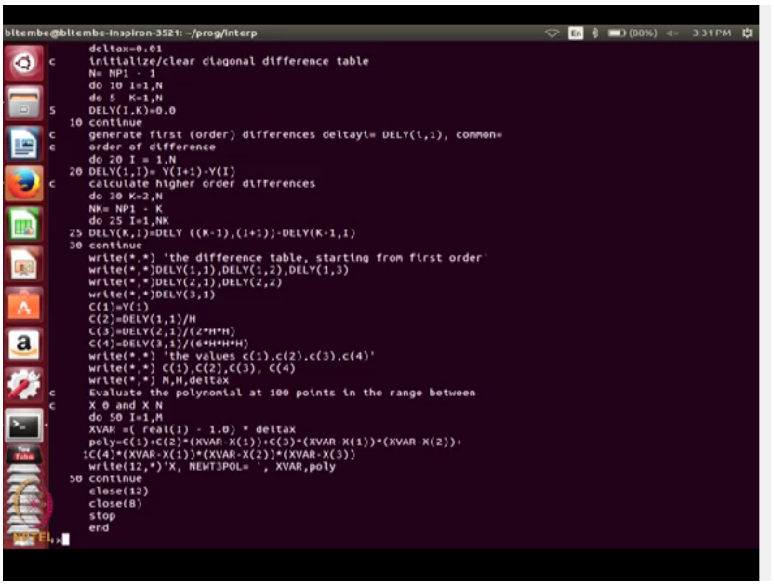
So K is 2, so NP1-K is 2, so there will be only 2 second order differences. So the 2 second order differences are calculated here. Do 25, i going from 1 to NK. NK is 2 because I have subtracted K=2 from NP1 which was 4. So 4-2 is 2. So I am going to calculate now 2 second order differences. What are they now? Del Y of K,i. K is already 2 now. It is a second order difference. When K is 2, I want i=1.

So the first second order difference will be the first order difference at i+1 point, - the first order difference at the ith point, okay. So i+1 will be 2 and i will be 1 and K-1 is 1 now. In the first loop, k was 2. So this is 1. So del Y, the difference between 2 first order differences gives me the first second order difference, K=2 at the first point. When i=2, it gives me the second second order difference.

So that is how I calculated the second second order difference. So this way I have calculated 2 second order differences. My next job is to calculate the only one third order difference. So then now K is 3. First time I calculated K=2. On the second order differences, when K=3, NP1-K is 3. So NP1 is 4. K is 3. So NK will be 3. So this is the third order difference. So the third order difference, del Y, now K=3, so the first third order difference will be the difference 2 second order differences.

So this loop study this carefully. It gives you all the higher order differences. And once I calculate all the higher order differences, it will write them on the screen. So it will write, these are the first order differences. These are the 2 second order differences. These is the 1 third order difference. So now when it writes from the screen, it will write a lot of data to make sure I understand what I am doing. So I shall go here. I am going to add this thing.

(Refer Slide Time: 15:49)



So I am going to write on the screen. So I have just typed, I went to this e, typed a, then it went to the next. Let me again show what I have done. So I was at the character e. At that time, I pressed, type a. That means then it goes to the right side, it allows me to insert something, okay. Then I type enter, okay. Then I type enter okay, then I want to write something on the screen. What I want to write on the screen?

I will say write(*,*. So that means I do not have to worry about format. Then 'the, I am typing the difference table, okay, starting from first order, okay.' So I have inserted this line so that now I know what the output is, okay. So after I come up to this, I type, I press escape again so that I am in the edit mode, okay. So through these lines, I am writing the 3 differences, first order difference, second order difference, third order difference.

Then I calculate my 4 coefficients, C1 was =Y1, the value of the function. The second coefficient was the first order difference/H. The third coefficient was the second order difference/2H square and the fourth coefficient was third order difference/6H cube. I calculate all these differences, then let me again write on the screen, okay. I go to that write bracket, type a, then I will insert.

So I want to write on the screen, okay, the values, the 4 values of the coefficients, it will write on my screen. So this is my statement. After I insert, I again type escape, okay. So now let us just execute the program. So these are all the remaining lines. What it will do now? Once I calculate

C1, C2, C3, C4; I want to evaluate the polynomial at those points. How do I evaluate the polynomial at those points?

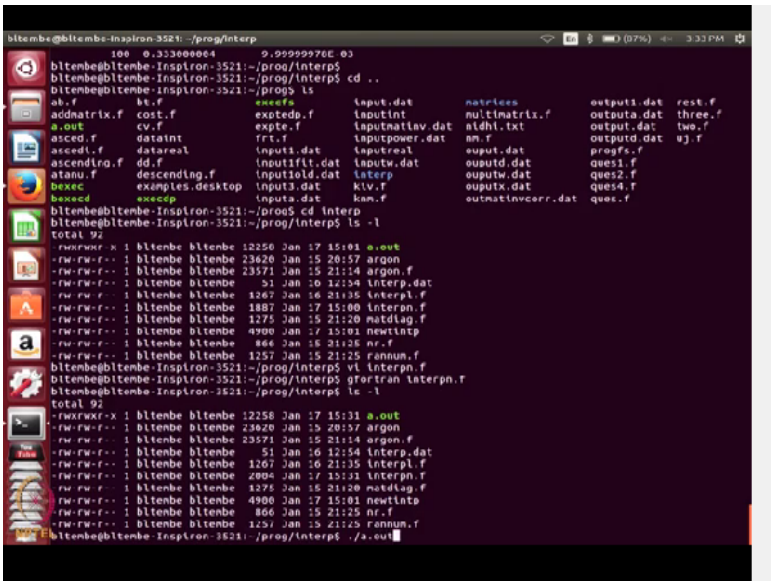
So there is a do loop i going from 1 to N. So I want to evaluate the 100 points. So what is the value of X at those 100 points? Real i, i was an integer. Real i will be a real number. -1*delta X. So the first value will be 0 because when i=1, real i=1.0. 1.0-1 is 0. Delta X is already the value I have told you. That is 0.1. In our case, I am using 0.1 for delta X. So first value will be 0. Second value of X variable will be real of 2, that is 2-1*delta X.

So the next value will be 0.1. The third value will be 0.2. Fourth value will be 0.3. Why 0.3? Because real of 4, that is 4-1 is 3, *0.1, that will be 0.3. So I will calculate for all values of X variable, the value of the polynomial. What is the value of the polynomial? C1+C2*X-X1+C3*X-X1*X-X2+, see the next line. This 1 in the sixth column is a continuation. See this is one line in this program.

So I am continuing that line in the next line. So 1 in the sixth column that is continuation. That is, so I have added the first 3 terms+C4*X-X1*X-X2*X-X3. So these 2 together constitute 1 line. So once I calculate the value of the polynomial, I will write it on file 12, X,newt3p, this is the Newton's polynomial at X. So that is XVAR and poly. So these are the values of X and the polynomial.

I am going to write on the screen, okay. So this is my program. Then finally that do 50 is a loop, that ends. Then I close unit12 and unit8. We are open to read something. It is always good to close whichever files we have opened. Even if you do not use the close command, it will still work but it is always good to be safe. Stop and end. So now I will escape, okay. Now type x, that means it saves this, okay. Enter.

(Refer Slide Time: 21:01)



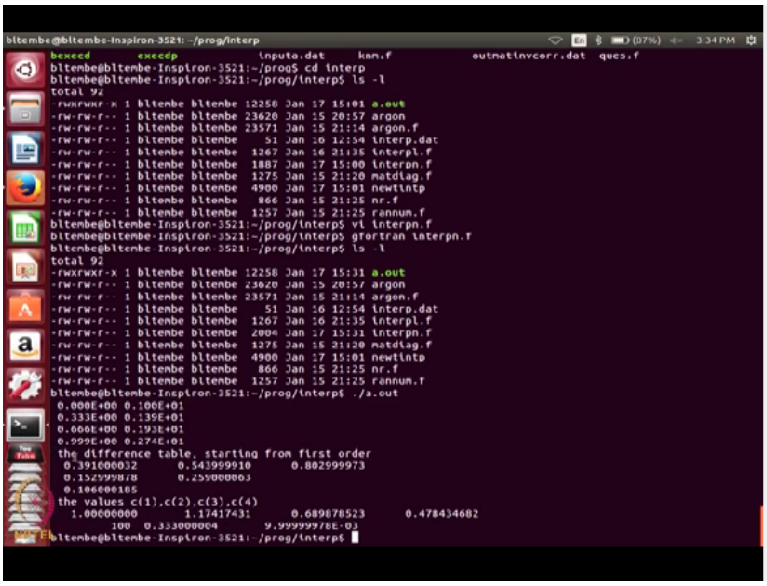
So I have saved whatever little changes I have added. Now I will compile. How do I compile? gfortran interpn.f. So I am compiling this particular program, okay. So it has compiled without any error. Now I will execute. How will I execute? Through a.out. Now look at this. The a.out here gave you a time of 15.01. So this was before I compiled. After I compiled, this will be a new a.out.

How will I know it is a new a.out? You see the time. This is was 1501. So let us do ls -l, enter. You will see that it shows 1531. So the time 1501 is replaced by 1531 because it is a new file. All other times are the same. See this argon, 2057, 2057; argon.f 2114 21. It has not changed and interpn.f, this was at 3:00 p.m., so since I saved it again, you see that interpn.f, that is 1531. So since these 2 programs are new, their times have changed.

So that means the old values, old files have been replaced by the new files. So this is one way to know, it is always good to take screen shots if you want and whenever a program is changed by the new times, you know that the program is a different one. So I have compiled, this is my new a.out. So therefore, another good point is whenever you have an a.out, you do not know which a.out it is. It is new one or the old one.

So it is good to keep track of them. I will execute now, ./a.out.

(Refer Slide Time: 22:53)



So now the moment I execute, it has executed. So what it is doing? First it is writing all the values of Xi and the values of the function. Remember I mentioned to you it is always good to write whatever you have read on the screen. So it has read the values of X, 0.333. In our class, we did 0.33 and 0.66. Here I have done a little better, 0.333, 0.666 and 0.999. These are my values of X.

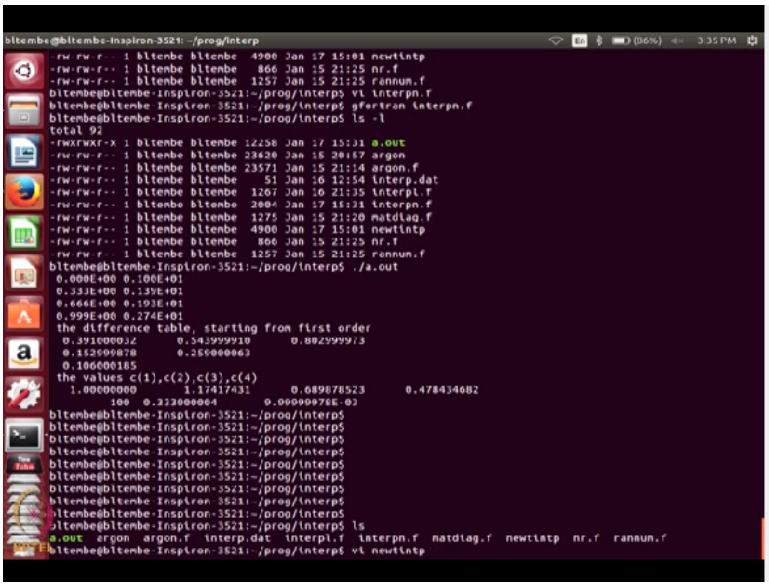
These are my values of i. Remember it is an E format. What is the meaning of E format? So there are total 10 spaces, okay. 1 2 3 4 5 6 7, okay. E+00, that is 4 values. These 3 0s that is 7, point and this 0, so it has taken 10 spaces. The first one and the next 9. For the second one also, 10 spaces. Which are those 10 places? The blank one 0.100, so these are 6 values, +E+01, 10 points for the second, 10 spaces for the first.

It has all the values of Xi and Yi. Now remember in the program I added this line, the difference tables starting from first order. So I have a first order difference. I have a second order difference. I have a third order difference. In the class, I wrote as columns and rows. Now these are, this is my first order difference, second order difference and third order difference. So these are the 3 differences.

Then I also have typed in the program to write al the values of the coefficients. So these are the values of the coefficients. 1, 1.17, 0.6898 and 0.478. These are the 3, these are the 4 values of the

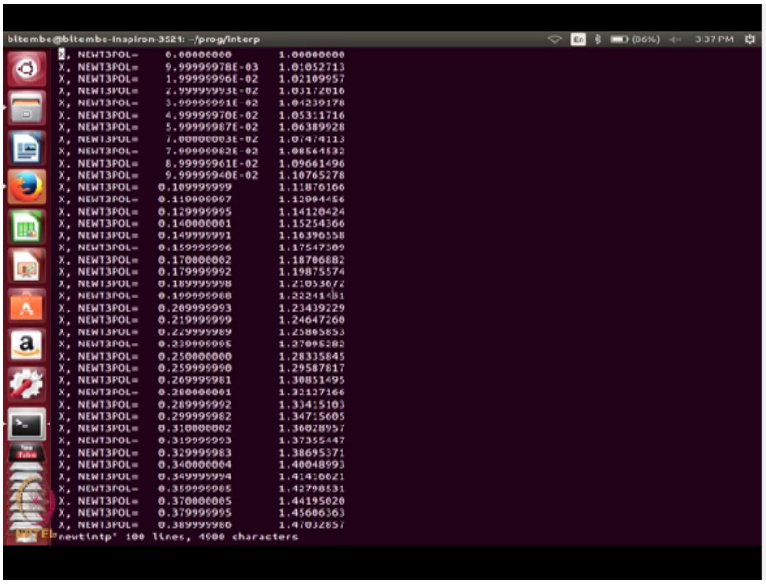
coefficients. This 100.33 and 99, let us find out after some time what these are? Now see since in the program I did not ask it to write what this is, I do not, I have no idea what this is? So I have to go back to the program and check that line. So before I do that, let us see what are the results now?

(Refer Slide Time: 25:09)



So the results were put in newtintp, this is the output file. So let us edit that. vi newtintp, so I am going to see that file.

(Refer Slide Time: 25:26)

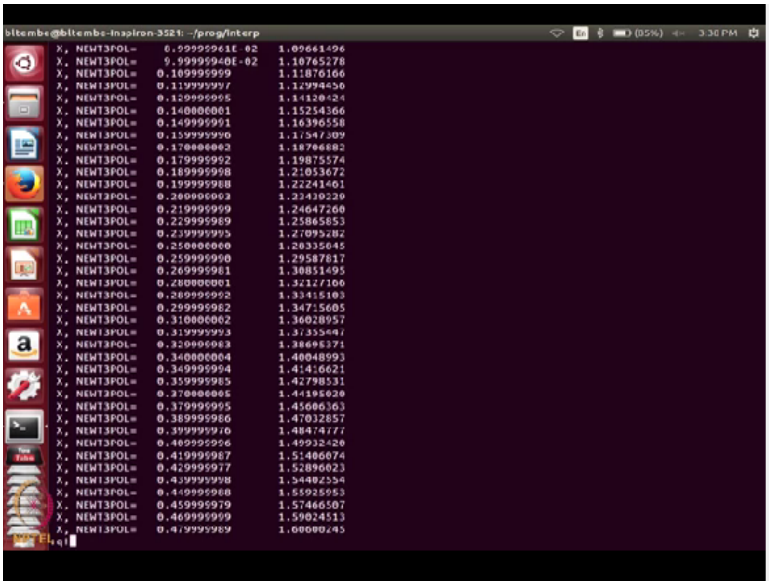


This is the file in which all the interpolation data that I created is written. So what will it write? So look at the first line. X, NEWT3POL value of X and the value of the polynomial. 0 and 1, my

first value was 1, X was 0, Y was 1. And in my data, after the first data .001, my next data point was .33. See this is the next data, .33 and 1.4 was my value of my function. So it has calculated actually my, in my function, I had value for .333 and a value of the function.

In the interpolated values, I have .33 and .34. There is no .33. It does not matter because my polynomial calculates for all values of X between 0 and 1. So you see that my input data was 1.39 at .33. So you see that all the values lower than 1.39 are for values of X<.33. So values of X goes from 0, .1, up to 0, this is .01 up to .1. Then .2, .33, okay. So these are my values of fx. So let us see the remaining values.

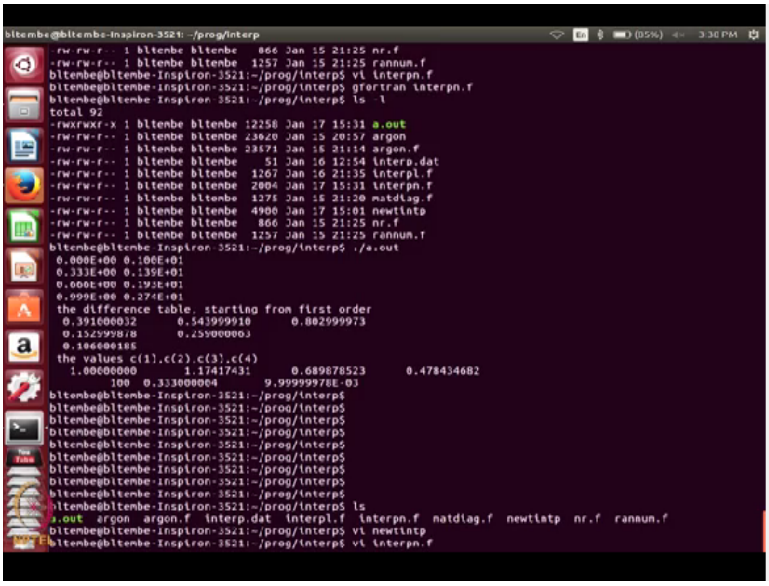
(Refer Slide Time: 26:55)



As I go down, so my last value is .99 and the value of the function is 2.71. So this is my interpolated polynomial. This is my independent variable X. Now you will see that instead of .99, my value of X that is written is .9899995. So as far as we are concerned, it is a recurring 999. So instead of .98999, I can read it as .99 as well. So this is my interpolated polynomial. So before I go to the Lagrange, let us find out what that last line was, okay.

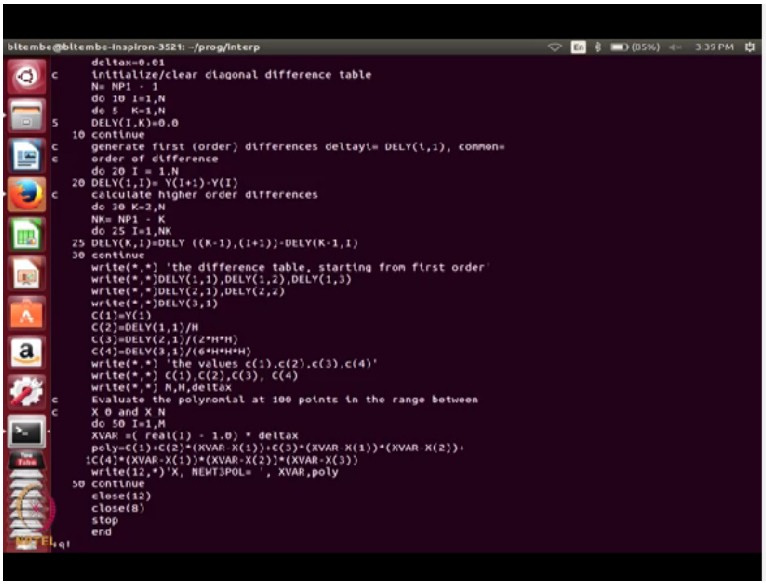
So how do I come out of this? Shift : q factorial. So I come out of this.

(Refer Slide Time: 27:44)



So what I want to know, see remember the program wrote 100.333 and 9.99 10-3. Let us see what this is. So I will go to vi interp.f. I go back to that program. And see what was that last statement.

(Refer Slide Time: 28:02)

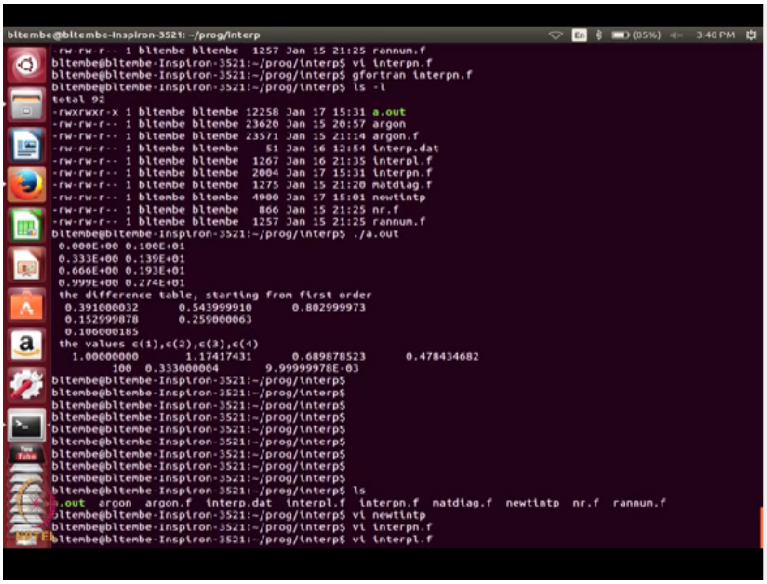


So here is that statement, okay. Remember we wrote C1 C2 C3 C4 and then it wrote M H and delta X, okay. What it wrote? M was 100, H was .33 and delta X was .01. .01 is spacing. Since I have 100 points, I have taken delta X to be .01. So .01*M will be, 100 points are there. I am calculating the 100 points with a spacing of .01, that is .01 .02 .1 .2.

All the 100 values of X are obtained by multiplying delta X to that 100 values. Let us look at that

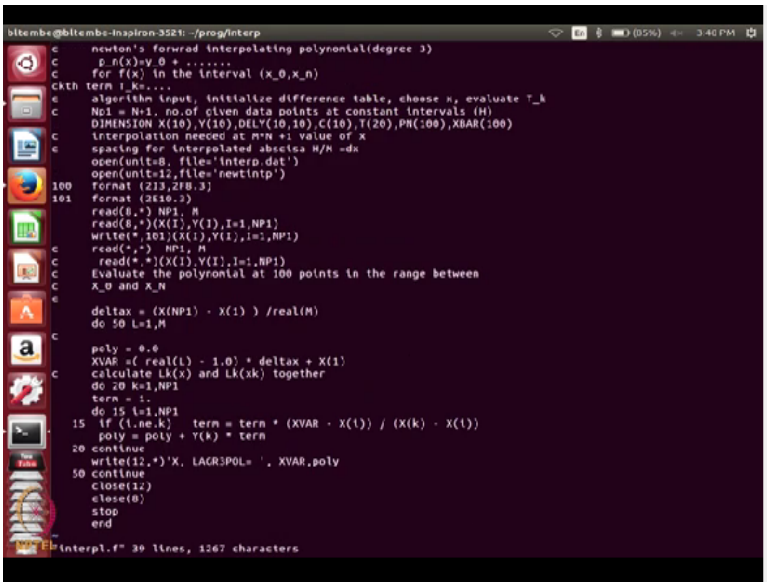
again. See this my real i goes from 1 to 100. So 100*delta X will be, last value will be .99 because the first value was 1-1*.01, first value was 0. So the 100th value is .99, that is the value at which I calculated my Newton's polynomial. So now what I want to do? Let us now look at Lagrange method.

(Refer Slide Time: 29:29)



So Lagrange method, I have already, interpl.f, okay.

(Refer Slide Time: 29:44)

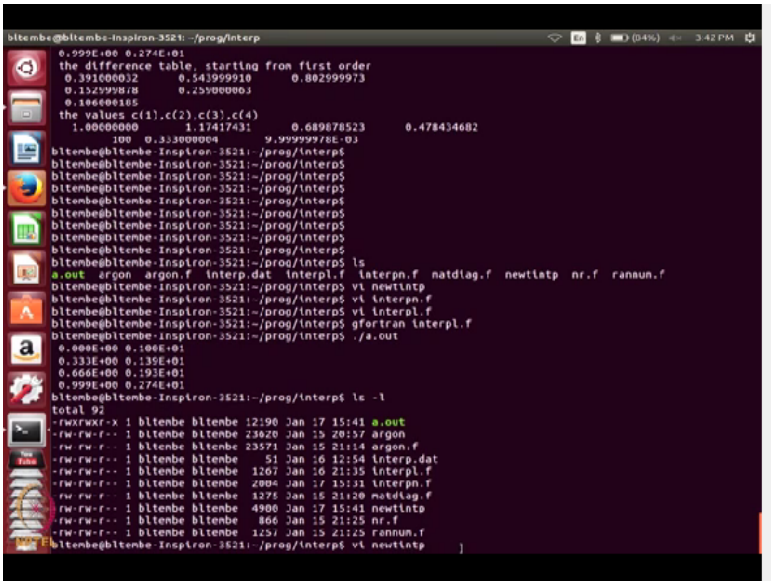


Now see this is my program for Lagrange interpolation. All the things up to this delta X are the same, okay. Same input files, same dimensions, everything is the same except now X variable, okay will be X1, that is my first value which was 0, +real L-1*delta X. So X variable is the same.

Now I want to calculate Lkx and Lk together, okay. So look at these lines. Do 20 i going from 1 to NP1, term=1, do 15 i going from 1,NP1.

If i is not equal to k, okay, then term=term*X variable-Xi*Xk-Xi. This is exactly, when I add all the terms, it will be Yk, see this is Yk*LkX/LkXk. This is exactly the same as the formula I have shown. So this is my Lagrange interpolating polynomial. It will calculate in exactly the same way. So I will, this program is available to you. I want you to practice it. I will come out.

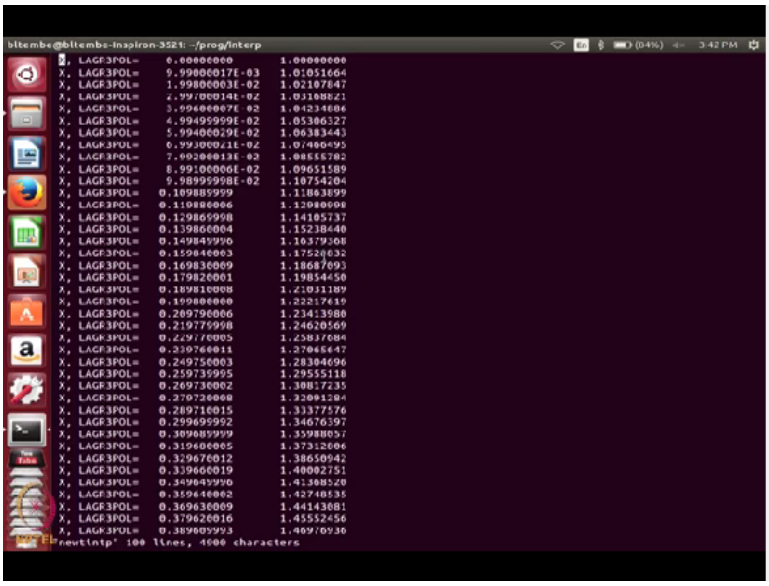
(Refer Slide Time: 31:01)



So I will compile it. interpl.f, so this is my Lagrange interpolating polynomial, okay. I am compiling the Lagrange one, okay. When I type this a.out, it is the Lagrange one that is executed, okay. Wait. This is executing, okay. So we will view the output. So before I view the output, let us do ls -l. Now you will see that the a.out corresponds to 15.41.

Earlier it was 15.31. So this is the new a.out which is the execution of the Lagrange interpolation polynomial. So to find the results, my data file is not changed. I have called it newtintp. So vi, I should have actually called it with a different name, newtintp, so I should have called it Lagrange interpolation but I have not changed it, okay.

(Refer Slide Time: 32:19)



So these are now the values of X and the values of Y. So here I have called it instead of Newton interpolation, Lagrange third order polynomial. These are the values of X and these are the values of Y. So you will see that both programs give exactly the same result. So now let me conclude what I did in today's lecture. I executed the Newton's interpolation polynomial. I executed the Lagrange interpolating polynomial.

Both gave me 100 values between X0 and X1. Any other method also will give you the same result because this is the unique interpolating polynomial. So in the next class, we will see what are the errors in these polynomials and then finally go to matrix methods, okay. These matrix methods are very important in chemistry because it helps you to calculate eigenvalue and eigen vectors of your Schrodinger equation. So I will conclude here. Thank you.