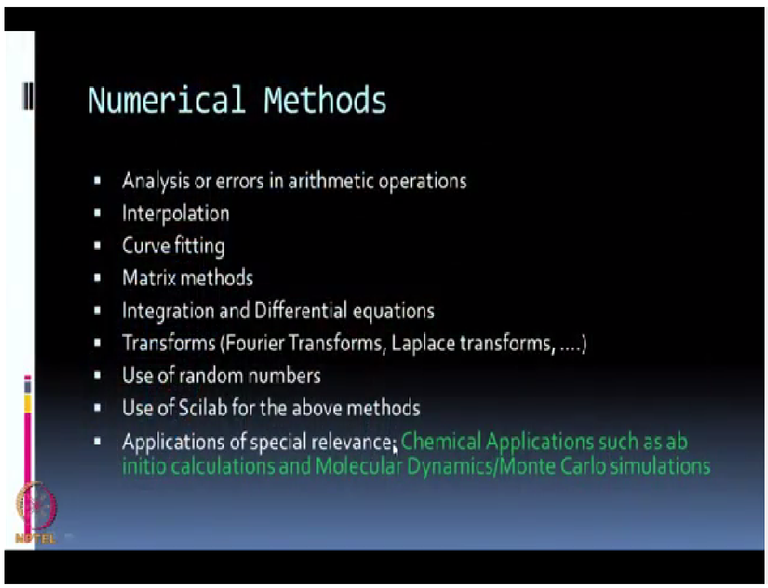
Computational Chemistry & Classical Molecular Dynamics
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology - Bombay

Lecture – 14
Interpolation Methods - 1

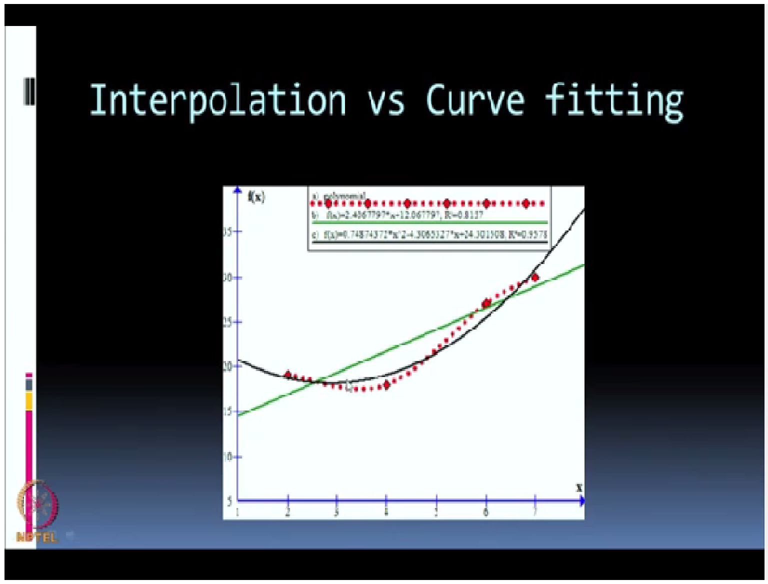
Hello and welcome to today's lecture. Last 2 sessions we had on practicals. That is we compiled several programs and executed them. Now we will receive our discussion on numerical methods and describe methods for interpolation today. So before I start interpolation, let us again summarize what our numerical methods are there. Now you see that we were discussing numerical methods.

(Refer Slide Time: 00:43)



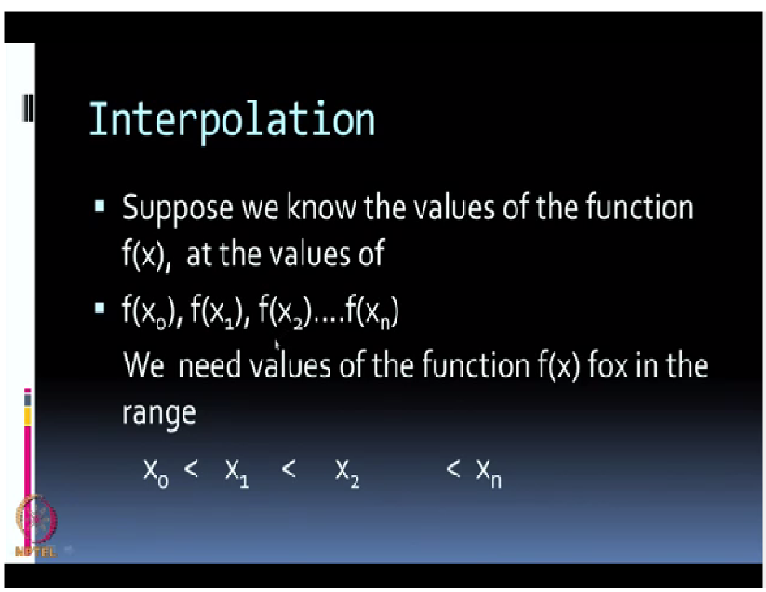
And what do these numerical methods involve? Analysis of errors in arithmetic operations, interpolation, curve fitting, matrix methods, integration and differential equations, transforms, use of random numbers, use Scilab and applications of special relevance. This is what numerical methods are.

(Refer Slide Time: 01:04)



And we already commented last time that interpolation involves fitting a curve through the data points. In this particular graph, you see that there is a red diamond, second red diamond, third and fourth red diamond. These are 4 points. Through these 4 points, I want to pass that curve. The dotted curve is the one which is passing through those points that is called an interpolating curve, okay. So whereas the other curves are fitting curves. We will do fitting in the next session. Today, we will be discussing only interpolation.

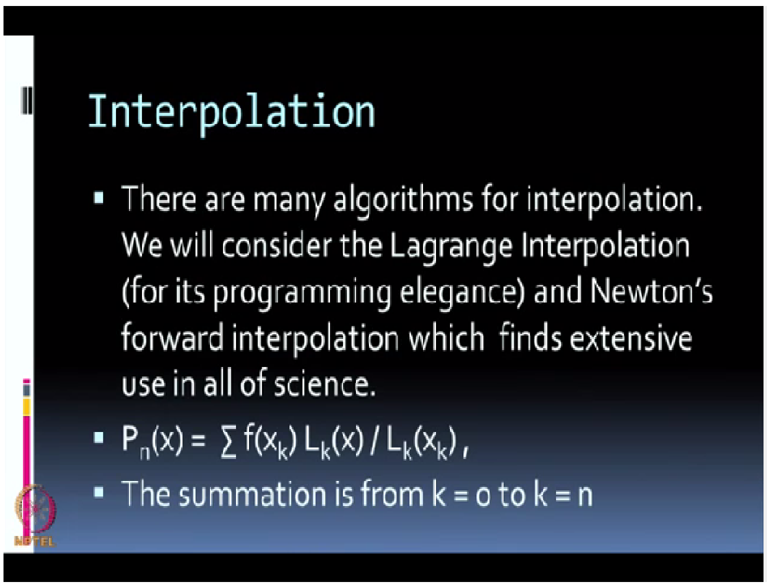
(Refer Slide Time: 01:43)



So what is the problem of interpolation? Suppose I have values of x0, x1, x2 and x3. These are my independent variables on the x axis. And the y axis I have fx0, fx1, fx2 and fxn. These are my points on the y axis. What we need is a function that passes through those points, fx0, fx1, fx2 and fxn.

We want a function to pass through those points and at points in between that, that is in the range between x0 and x1, x1 and x2, these are points intermediate between the points on the x axis. For those intermediate points, we want values of fx. So experiment or my data does not have those values of fx for intermediate values. The interpolating curve gives me the values at those intermediate points, okay.

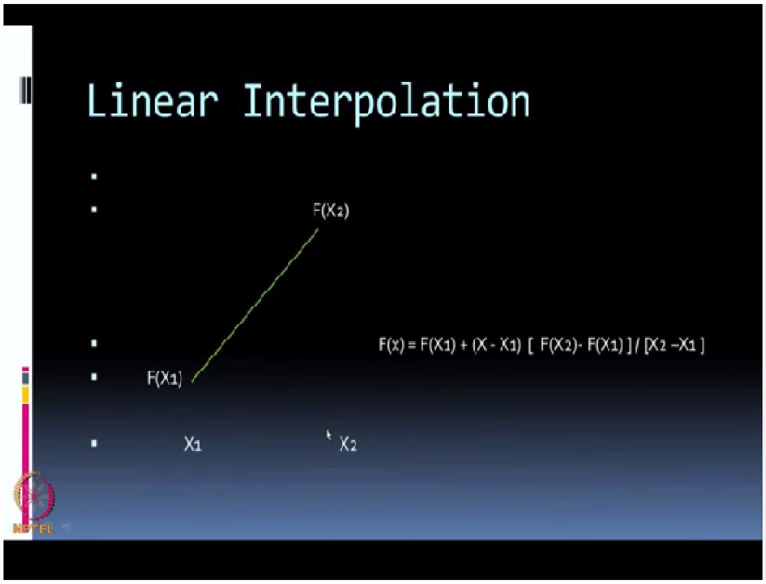
(Refer Slide Time: 02:41)



So we will consider 2 methods of interpolation. There are several methods. One is LaGrange interpolation. One is a Newton's interpolation. But Newton's interpolation finds extensive use in many many applications. So we will do the program for Newton's interpolation first and we will also consider LaGrange interpolation. So what does LaGrange interpolation involve? Is that the nth order polynomial, this Pnx, n refers to the order of the polynomial.

The nth order polynomial is given in terms of fxk which are the values of the function at known points k. So normally we take k to go from 0 to n. So 0 to n means if there are, if n is 10, 0 to n would be 11. So those 11 values of fxk, Lkx evaluated at those 11 points, Lkx evaluated at each one of the values of k,/Lkxk. So the formula is given in the later slide. So before I proceed, I want to define what is a linear interpolation?

(Refer Slide Time: 03:47)



This particular slide demonstrates what is linear interpolation. x1 and x2 are the points on the x axis and this line is a line that has values fx1 at x=x1 and fx2 at x=x2. So between these 2 points, I do not have any data. And I want to interpolate using a straight line that is for all values between x1 and x2, I want to say that the function is a straight line and I want values of fx at those points.

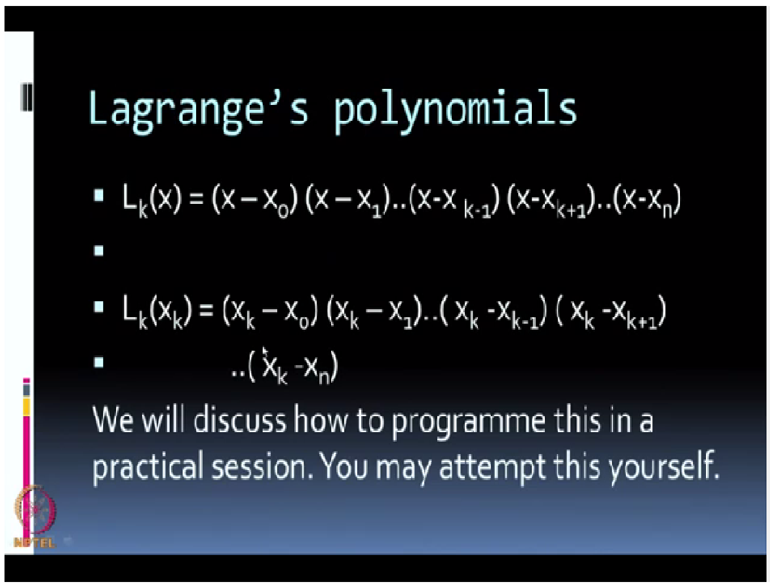
So how do I get the values at those points? You know that the equation of a straight line is y=mx+C. So m is the slope, okay; C is the intercept and y is my function, y=mx+C. So how do I write the interpolating polynomial here? fx which is the value of the function at any point between x1 and x2. That fx is fx1 which is the starting value of the function, fx1, +the slope of this.

What is the slope of this fx2-fx1/x2-x1, that is my slope, fx2-fx1/x2-x1. This is the slope, times x-x1. So when I do this, when x=x1, you know that this term is 0, then fx=fx1, that is my starting point. When x=x2, this x2-x1, there is a denominator x2-x1, both of them cancel. All that remains is fx2-fx1. fx1 cancels with this fx1. Function becomes fx2. When x=x2, the function is fx2.

When x=x1, the function is fx1. At all intermediate points, my function is defined through this formula. Now this particular function will work well beyond x2 also. Suppose I extend this line beyond, that function will work very well beyond this as well as below fx1. Because a line extends all the way on the right side as well as all the way on the lower side. The straight line will extend and this function will give me values of the function at all points for all values of x.

But it may so happen that beyond x2 and below x1, it may not be a straight line. So straight line is not a good method for interpolation if x1 and x2 are far apart. If x1 and x2 are very near, straight line interpolation is very good. So now suppose I have more than 2 points, what is a good polynomial that I can fit that is what I am going to describe next, okay.

(Refer Slide Time: 06:33)



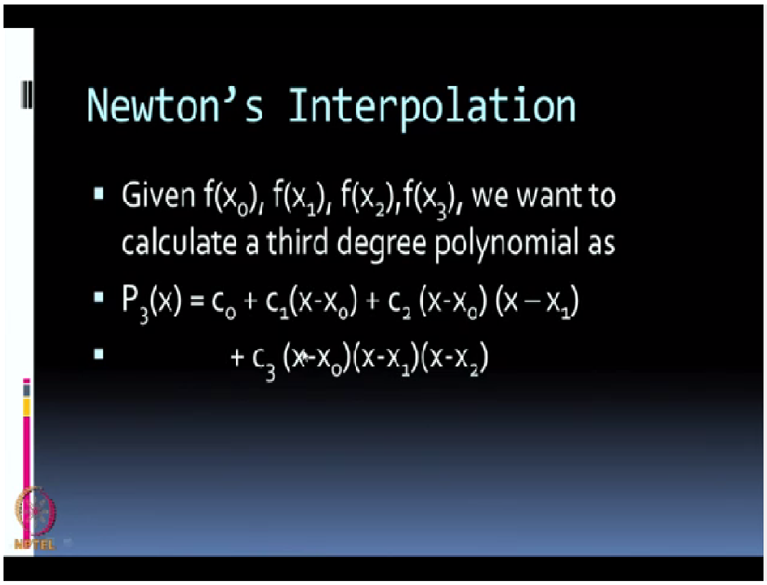
b

So remember LaGrange interpolation is a LaGrange polynomial. I already said in the last slide. Let us go back. It is fxk*Lkx/Lkxk. I am just going to define Lkx and Lkxk, okay. So what is Lkx? Lkx is x-x0*x-x1*x-xk-1, only the kth term is not there. That is x-xk is not here. Because I want Lkx, x-xk is not there. x-xk+1 up to x-xn. All the products are there. There are n products except x-xk.

So that is my, this is the polynomial. This is a polynomial of the nth order because there are n terms. What are those n terms? n, n-1, n-2, k-1, k+1. k is not there up to x-x0. This is a polynomial of order n. Then Lkxk is a factor where in place of x, I have xk, okay. So what is Lkxk? xk-x0, xk-x1, xk-xk-1, just as here. xk-xk is not there because xk-xk will be 0. So that is not there.

xk-xk+1 up to xk-xn. So this is, Lkx is a polynomial. Lkxk is just a term. So my polynomial is fxk*Lkx/Lkxk. We will discuss this program a little later today. You may also try to program it yourself before looking at our solution. Because it is only when you attempt, you are on programs, you will get a better mastery, okay. So now this is my LaGrange polynomial.

(Refer Slide Time: 08:26)



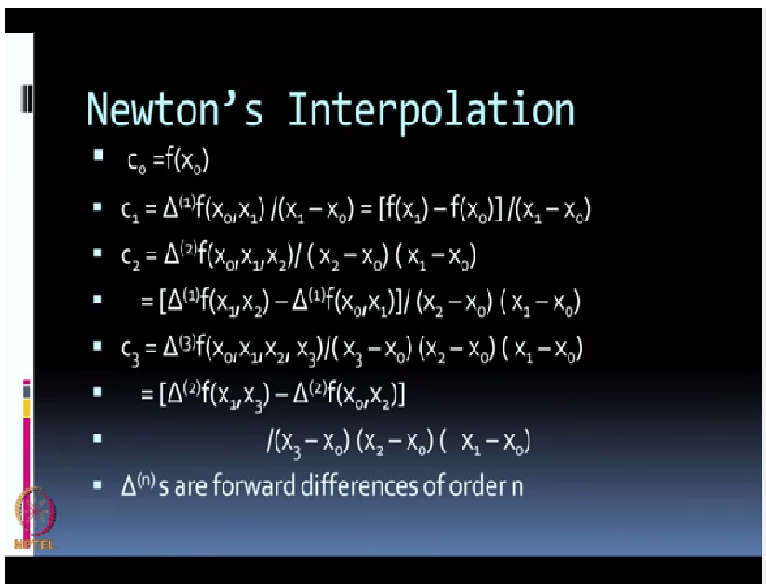
The next one I want to consider is a Newton's interpolation. This is slightly different from LaGrange interpolation. So the formula is a lot simpler. Now I want to consider a third degree polynomial. What is a third degree polynomial? x occurs to the third power. So in my problem, 4 values of x are given, x0, x1, x2, x3. 4 values of x are given and 4 values of the functions are given, fx0, fx1, fx2, fx3.

Now I want a polynomial in the range x0 to x3. So whenever you find a function between x0 and x3, it is called interpolation. Whenever you want a value below x0 and beyond x3, it is called extrapolation. We are not too keen on extrapolation because extrapolation has far too many errors compared to interpolation and we have no knowledge of the function beyond x3 and no knowledge of the function less than x0.

So we will not worry about extrapolation. It is just interpolation. The polynomial, Newton's polynomial is given by P3x=C0, we want to determine this constant, +C1*x-x0+C2*x-x0*x-x1+C3*x-x0*x-x1*x-x2. So there are 4 terms here. The first term is a constant. Second term is a linear term, x-x0. The third term is a quadratic term because x square will appear here. And the last one is a cubic term, x cube will appear here.

So this is an interpolating polynomial for these 4 points. Whenever you have 4 points, you get a third order polynomial. Whenever you have 2 points, you get a straight line. Whenever you have 3 points, you get a parabola. So I would want you to work how to get a parabola given x0, x1, x2 and fx0, fx1, fx2. I will leave that as an exercise. Now we want to program for this P3x, so we want to determine the formulae for C0, C1, C2 and C3. That is the whole strategy in Newton's interpolating polynomial. So now let us look at the formula.

(Refer Slide Time: 10:41)



The formula is given here. This is the Newton's interpolating formula. We also call it a forward interpolation. Forward interpolation because you will go from x0 to x1, x1 to x2 and so on. So my C0, the first constant is the value of the function at x=x0, okay. So this is my C0. C1 is given by delta 1fx0x1/x1-x0. Now this delta 1fx0x1 is called a forward difference. What do I mean by the forward difference?

I have a function, I take the value of the function at x1, subtract the value of the function at x0. So this is the difference between fx1 and fx0. This is the forward difference. So first order difference will be fx1-x0. Second order difference will be, okay, I take 2 first order differences and subtract to get a second order difference. That is what is given here. C1 is given in terms of first order difference/x1-x0.

C2 is given by the second order difference/x2-x0 and x1-x0. What is my second order difference? Second order difference is I take the first order difference between x1 and x2. What is the first order difference between x1 and x2? fx2-fx1. That is my first order difference between x1 and x2, okay. Then delta 1fx0x1 will be the first order difference between x1 and x0, that is fx1-fx0.

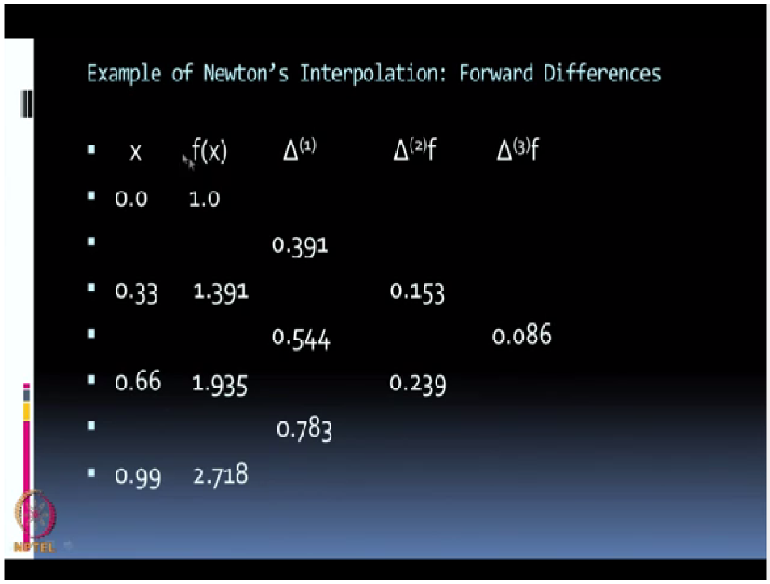
So first order difference at point x1, first order difference at point x2, the difference of the 2,/x2-x0 and x1-x0. So my C2 term is a coefficient for the second value. C2 is given by this. C3 is given in terms of a third order difference. What is a third order difference now? Third order difference will be the difference between 2 second order differences. What is fx0x2? Fx0x2 is the second order difference which is already given here.

This is my second order difference at x2. Now there is again a second order difference at x3. So just as I have a second order difference at x2, I can calculate a second order difference at x3. So the difference between 2 second order differences is my third order difference. So that third order difference,/x3-x0 x2-x0 and x1-x0. So this is my C3. So this is the discussion on my Newton's interpolating polynomial of a third order polynomial.

Now suppose I want to extend to higher orders. If I want to extend to higher orders, I will need higher order coefficients. Say for C3, I had a third order forward difference. For C4, I will have a fourth order forward difference. C5, I will have a fifth order difference. So one of the main strategies in interpolation is you do not use very high order polynomials. When you use very high order polynomials, the excess power is to very large values, like 10, 11, 12. So large polynomials have lot of fluctuation between adjacent points.

Because you know that at nth order polynomial has n 0s. So higher order polynomials go through 0 many many times. So they are not good for interpolation. So best would be the third order interpolation and this is the Newton's formula that I have just indicated. Now my next thing would be, I will take an example. I will take an example of a Newton's third order interpolating polynomial, that is what my next slide is going to be, okay. So look at this example. This example has data x, 4 values of x and 4 values of fx.

(Refer Slide Time: 14:54)



The first value of x is 0. The second value is 0.33. The third value is 0.66 and the fourth value is 0.99. So these are my 4 values of x and in the second column, I have 4 values of the function. What is my function? When x=0, the function is 1. When x=0.33, the function is 1.391. When x=0.66, my value is 1.935. And when x=0.99, my value is 2.718. So these are my 4 values of x and 4 values of fx.

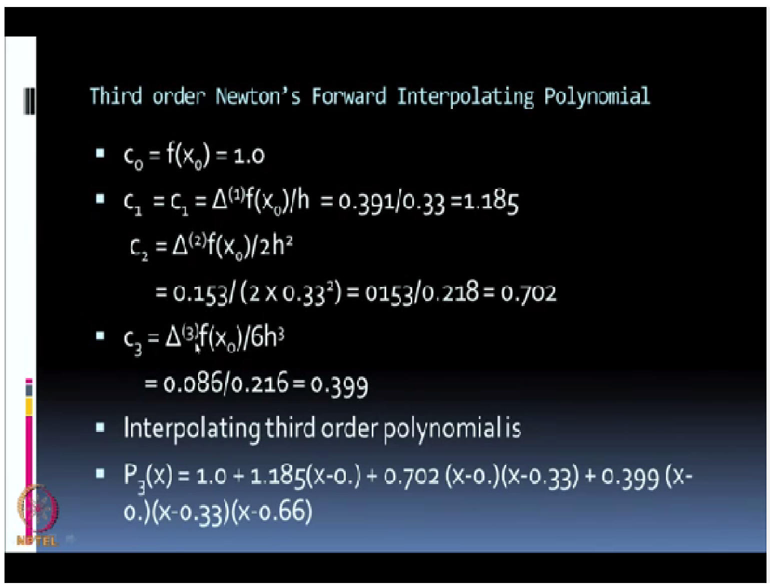
So what is a first order difference. First order difference is the difference between fx1-fx0. So these are the 2 adjacent points. This is my first order difference, okay. This is the first order difference between x1 and x0. x1 and x0, this is fx1-x0. The second order difference is fx3-, sorry fx2-fx1. The first order difference, this is the third one, fx3-fx2. So I have 3 first order difference between 4 points of the value of the function fx.

These are my 3 first order differences. How do I get my second order difference now? Second order difference will be the difference between the first order difference at the second point-the first order difference at the first point, okay. This is the first order difference. The difference between the 2 first order differences is my second order difference. So between the first 2 first order differences, I got a second order difference.

Between the second 2 first order differences, I got the second second order difference. So there are 2 second order differences and finally I have a third order difference. What is the third order difference? It is the difference between 2 second order differences. This is my second order difference between the 3 points. This is my second order difference between the first 3 points. So the difference is my third order difference.

So once I have all these differences, now I can build a polynomial.

(Refer Slide Time: 17:04)

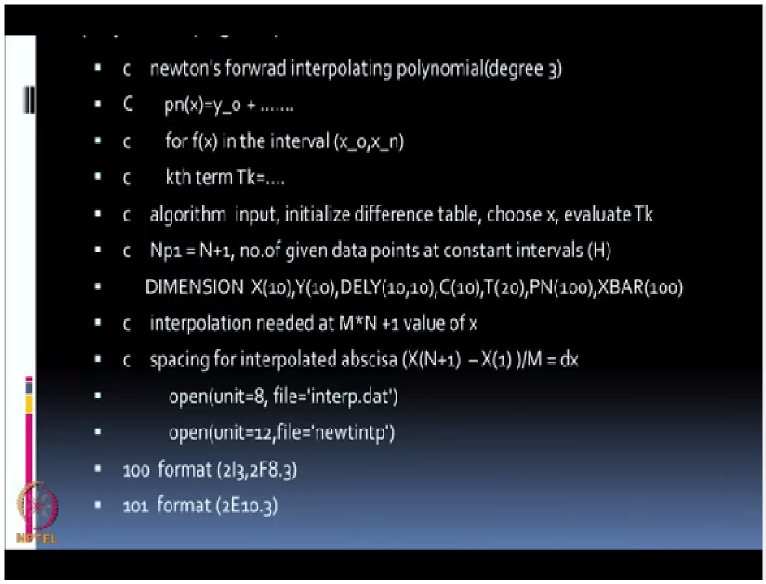


My next slide will show you how I build that polynomial. Remember to build the polynomial, I need the values of C0, C1, C2 and C3. These are my values. What is C0? Fx0 which is 1.0. C2 is second order difference at x0/2h square. C3 is the third order difference/6h cube. This is how I calculate my differences, okay. So the first order C1 was first order difference/h, that is 0.391/0.33, that is 1.185.

C2 was my second order difference/2 times h square. Now this h is the difference between adjacent points, okay. So 0.702 is my second coefficient. Now third coefficient is the third order difference at x0/6h cube. So 0.086/0.216, 0.399. So I have the 3 values of my coefficients and how do I get my polynomial now? Polynomial is P3x=1.

1 is nothing but value of C0, +x-x0*1.185, C1, +0.702 which is the second value, *x-x0*x-x1. This is my x0, x1. So this is x-x0, x-x1. Finally the third value, 0.399, that is my C3, *x-x0*x-x1*x-x2. So now this is a polynomial which is an interpolating polynomial between these 4 points of data that I had. And with this function, I can calculate the values of the polynomial.

(Refer Slide Time: 18:55)



So now my next strategy would be, how do I program this. I will be describing all the lines of the program. Then I will actually execute the program and show you in my next hour. So before I go further, we want to make some points on these polynomials. I already mentioned that if you have 4 points, you can have a third order polynomial between these 4 points. Now if any function passes through these 4 points, any other way of getting this interpolating polynomial will give me the same polynomial.

There is a unique polynomial which will pass through these 4 points. I mean there is a theorem in algebra, you can actually prove that but I will not do this in my lecture. So between 4 points, there is a unique interpolating polynomial. So whether I do LaGrange interpolation or Newton's interpolation, I will get exactly the same result. So there is an interpolation, uniqueness of interpolation.

So now I will, let us see the difference between LaGrange method and Newton's method. In

Newton's method, what was required is that it is far more effective if the axes are spaced equally between each other. If you recall our data, what was our data? 0.33, 0.66, 0.99. If the data is even equally spaced, the whole process becomes very easy, okay. Whereas for leg range interpolation, there is no requirement of any kind.

You can have any values of x0, x1, x2, x3 and you can have an interpolating polynomial. So Newton's method is far more effective if they are equally spaced, okay. So now I shall start looking at the program. Now look at the program. I want to start the interpolating polynomial, okay. So this is my formulae for all the coefficients. This is how the program begins. So look at the program carefully.

The first line is the comment statement. It is always a good idea to have comment statements in the program because the moment you look at the program, you know what it is going to do. So the first line is the comment which says that the Newton's forward interpolating polynomial degree 3. And how is the comment card written? The first character is a c. Do not consider these squares here.

The left hands square are more the part of the ppt. Your actual first program, actual first column in the program starts with this column. This is the first column in your programming lines. First one is a comment card. Second one is also comment. Pnx=y0+c1+c2, this is the polynomial, okay. So for fx in the interval x0 to xn, that is my polynomial. Kth term, let us call it tk, okay. So what algorithm I am going to use in my algorithm?

There will be input, okay. And then I will initialize the difference table. So remember, there are these, these are all forward differences, first order difference, second order difference, third order difference which we have called delta 1, delta 2, delta 3. I will initialize the differences. Then once I initialize the differences, I have all values of x and once I have all values of the table, I can calculate the value of the polynomial for any given x, okay.

And now our strategy would be, we are always going to use N+1, that is if it is, if there are 4 points, then I have a third order polynomial. This N would also refer to your degree of the polynomial. So you have 4 data points here, okay at constant intervals. H is the interval between adjacent point, okay. Np1 is N+1. So these are all the comment cards which is part of the requirement for our own convenience.

Now I will start giving all the variables in the program. So what are my variables in the program? X is a variable, that is how many data points I have. This 10 is arbitrary. Instead of 10, I could have given 50 or 100. I could have given 4 also because I have 4 data points but normally I give larger dimensions because you may have large amount of data and you want to interpolate between adjacent points in that large amount of data.

So it is always good to declare dimensions which are much larger. So x is of dimension 10, that means the variable x can take 10 values. Y also is 10 because if there are 10 values of x, there will be 10 values of y. Then this del y is my difference table. This is the forward difference table. So the maximum I will have is 10,10, right. Since there are only 10 data points, I will not even, when there are 10 data points, there will be 9 first order differences, there will be 8 second order differences, so maximum will be 10, so I have just defined a del y.

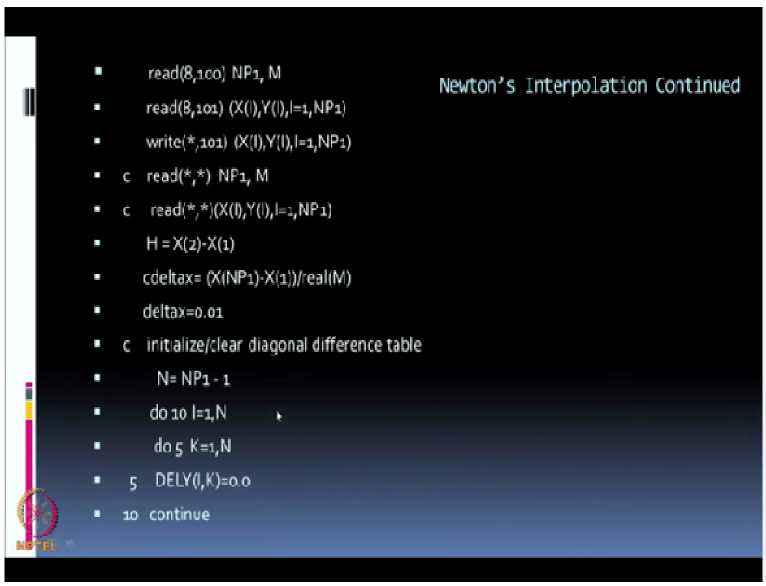
This is an array, 10,10. cs are for the coefficients. cs are the coefficients, I have c0, c1, c2 and so on, okay. So then these Ts, PNs, X bar, these are all, X bar is a value at which I want to calculate the interpolated value. So these are not so important for our present purpose. So let us say I want to interpolate for some 100 points. What are these 100 points? Between the last value of X and the first value of X, I had 4 data points in my program.

So what I want to do here? Is to calculate the value of the interpolating polynomial at those 100 points between X0 and XN. So the number of points will be much larger than the data points I have because I have only 4 data points. At those 4 values of X, I already know fx. So there is nothing to interpolate for the known values of those points. So I want to interpolate between all the Xs in the intermediate values.

So in this case, I have chosen 100 values. So for that purpose, I divide the interval between XN and X0 into 100 values of X. So that is my division. So now I need to read the data from files again. So open unit=8, file=interpolate.dat. This is my input file for my interpolation. Then whatever results I get, I am going to put in the output file. That is open file=Newton interpolation. So I am, for the output, I am giving newtintp as the output file, interp.dat as an input file.

Now why do I use this term interp and newtintp? So it tells me the moment I see interp, I know that it is a data for interpolation. And newtintp, so it has something to do with Newton's interpolation. So in this case, so file which has access by number 8, is the input file. File that is expressed by unit 12, will be the output file.

(Refer Slide Time: 26:13)



So my next things are the read statements, okay. I will read, so what I have to read? I have to read NP1, that is number of points +1, okay. If N was 3, our number of points was 4, okay. NP1 is N=1 and M is the number of points at which I want the interpolated polynomial, okay. So this Newton's interpolation method is not part of a program. So do not give any importance to this. This particular thing is not part of the program, okay.

So this is just for our reference that in this slide, I am discussing the Newton's interpolation method. So my next line was read 8 from that file 8 and NP1 and M. These are 2 integers. So I am going to read those 2 integers from file number 8. Next thing I want to read all the values of Xi and Yi. So how many are there? 4 values of Xi, 4 values of Yi, I going from 1 to NP1, okay.

So now this particular read statement, I am reading many many points without using a do loop.

So what this involves? This is called an implicit do loop. What is an implicit do loop? In the bracket, a do loop is already implicit because I want to say Xi,Yi, I going from 1 to NP1. NP1 is N+1. So there are NP1 points, i goes from 1 to NP1 and these values will be read. So a do loop is implicit. The moment I put in bracket Xi,Yi, I going from 1 to NP1, it will read all those NP1 values of Xi and Yi.

Make sure you put these commas correctly. Because if any comma is missing, the do loop is not going to work properly, okay. So this is how I will read Xi and Yi, okay. Next I am going to read from the, so in this particular case, after I read Xi and Yi, I want to write on the screen. I want to write on the screen what are those values because many times you read something from a file, you do not know what is read.

So if whatever is read is shown on your screen, then you know exactly what you have read. So it is always a good idea whatever you read from a file, write the thing on the screen or write in some other file so that you can crosscheck that whatever you did was alright. So what I will do, I will conclude this lecture at this point. The next lecture, we will actually execute this program. We will execute the Newton's interpolating polynomial program.

We will execute the LaGrange interpolating program. And then you will see the results for yourself on the screen. So what we have discussed in this lecture? We have discussed the methods of interpolation. So there are many many methods. But we will use those methods which are easy for programming purpose.

So Newton interpolation is good because it is useful in many other areas such as integration, differential equations and so on. LaGrange's interpolation, it is a very nice thing to write a program for but Newton's is better because I can analyze the errors as well in this particular interpolation method. So I will conclude now. And the next lecture will be executing the entire program. Thank you.