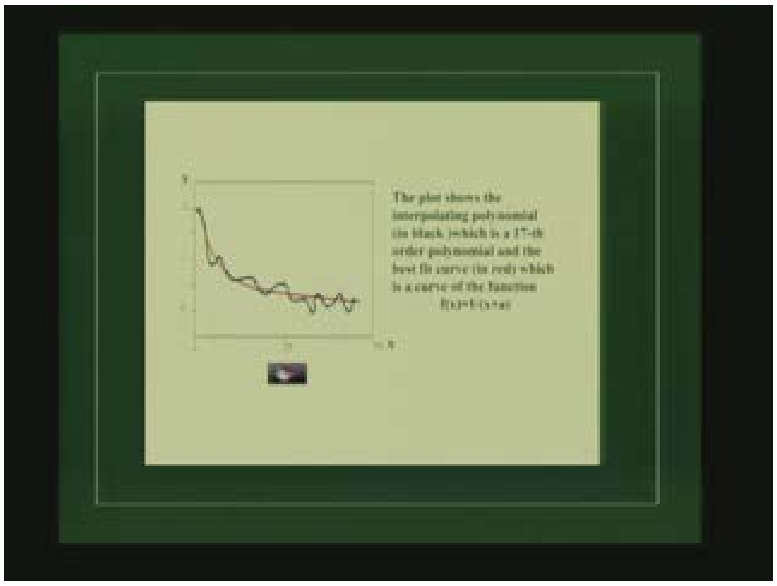
Numerical Methods and Programming
P. B. Sunil Kumar
Department of Physics
Indian Institute of Technology, Madras
Lecture -10
Error in Interpolation Polynomial

Today, we will continue with our discussion on interpolation. So we are again discussing interpolation, interpolation of a discrete set of data points, to get values of the function, approximate values of the function in between. We saw that we use, if you have a set of data points given like in this blue curve, blue point here, this set of data points, and then I could actually draw two sets of lines through this, one set which goes through, on average, through all the points, and the other set actually touches all the points.

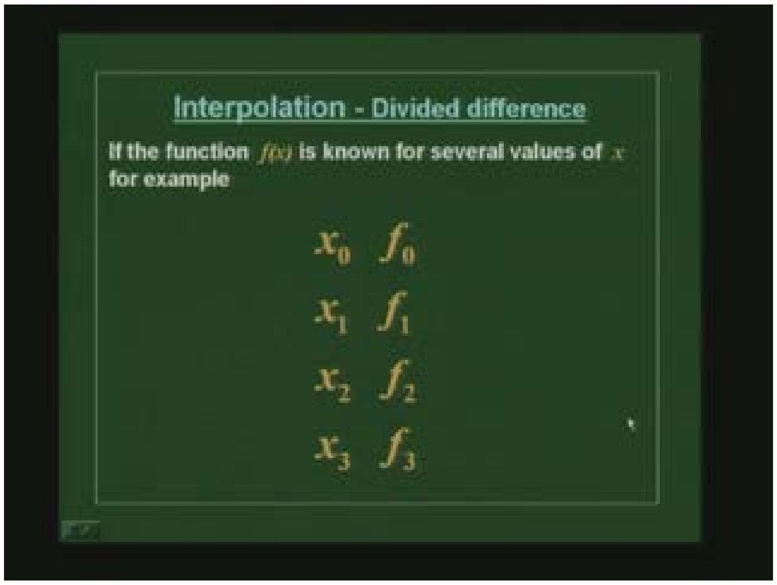
(Refer Slide Time: 01:54)



The plot shows the interpolating polynomial (in black) which is a 17-th order polynomial and the best fit curve (in red) which is a curve of the function $f(x)=1/(x+a)$

So, the red curve here is what we would call a fit, and the blue curve, and the black curve here which goes through the blue points is what we would call an interpolating curve. So, in this particular example we have a function which is 1 by x plus a as the red curve which goes through all these points, which just goes through these points, but does not touch all the points, but while this is a more realistic approximation of a curve. But again, I warned in the last lecture that this interpolating curve which goes through all the points is still an approximate function. It is not the real function. And then we looked at a particular form of interpolation called the Newton’s method, Newton’s form. So it was Newton’s form which we saw, and then we wrote what is known as the divided difference method to write, to actually write down the formula.

(Refer Slide Time: 02:56)



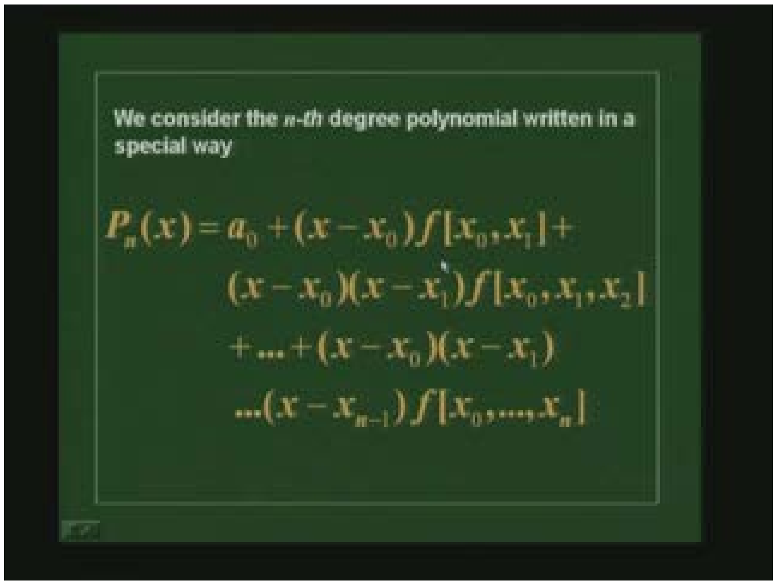
Interpolation - Divided difference

If the function $f(x)$ is known for several values of x for example

$$\begin{array}{cc} x_0 & f_0 \\ x_1 & f_1 \\ x_2 & f_2 \\ x_3 & f_3 \end{array}$$

So in the divided difference method we had a set of values, a set of values for the variables x_0, x_1, x_2, x_3. That is the, and the independent variable, and then you have the function values at these points as f_0, f_1, f_2, f_3, and then we constructed a polynomial of this form.

(Refer Slide Time: 03:34)



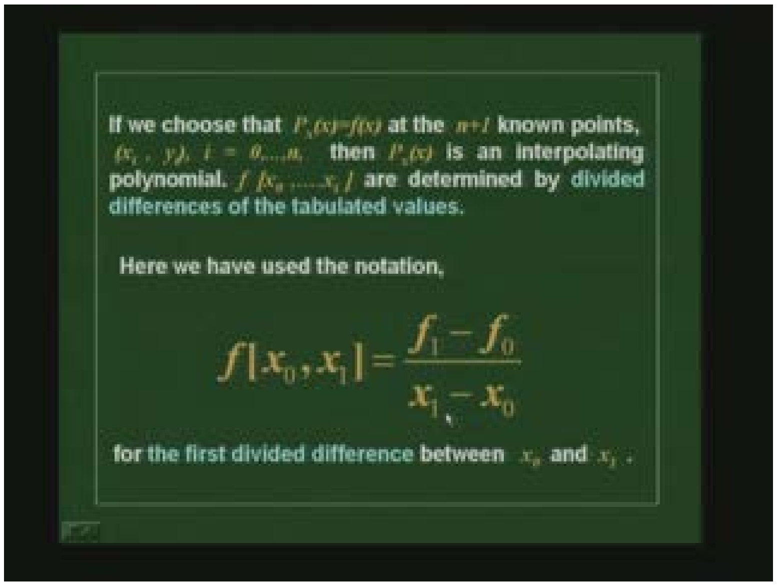
We consider the *n-th* degree polynomial written in a special way

$$P_n(x) = a_0 + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \dots + (x - x_0)(x - x_1) \dots (x - x_{n-1})f[x_0, \dots, x_n]$$

So this is the Newton's form of the polynomial which we wrote which goes through all the x_0, x_1, and all the points up to n. So this is the form of Newton's polynomial. So we wrote x minus x_0 into f of this square bracket x_0 x_1. So this is actually the coefficient of this term, and this is the coefficient of the next order term, and this is the coefficient of the nth order term, etcetera, n minus 1th term, etcetera.

Now to determine these coefficients we use the divided difference which I just mentioned, and wrote on each of these coefficients as f of x_0 x_1 as f_1 minus f_0 by x_0, x_1 minus x $_0$, and then we constructed this first order divided difference between all these points and all these function values, that is, f_1 minus f_0, f_2 minus f_1 divided by x_2 minus x_1, f_3 minus f_2 divided by x_3 minus x_2, etcetera, and then as a next term we constructed further divided differences.

(Refer Slide Time: 04:50)



If we choose that $P_n(x)=f(x)$ at the $n+1$ known points, (x_i, y_i), $i = 0,\dots,n$, then $P_n(x)$ is an interpolating polynomial. $f[x_0,\dots,x_i]$ are determined by divided differences of the tabulated values.

Here we have used the notation,

$$f[x_0,x_1]=\frac{f_1-f_0}{x_1-x_0}$$

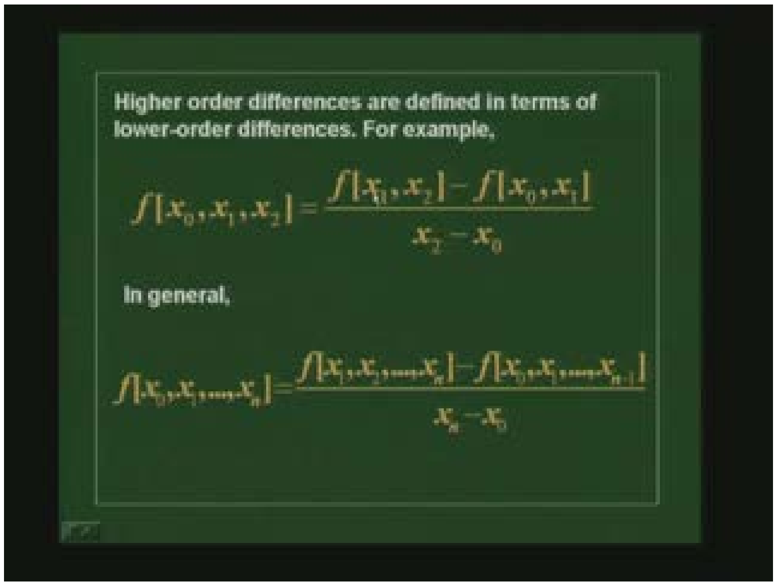
for the first divided difference between x_0 and x_1.

So the first divided difference, and then we had all the first divided differences given by this form, that is, in general, f "x_s x_t". This is, will be f_t minus f_s divided by x_t minus x_s.

That is the definition of the first order divided difference, and the second order divided difference is as I said, is given by the difference of the first order divided differences, that is, f "x_1 x_2" minus f "x_0 x_1" divided by x_2 minus x_0 gives you f of x_0 x_1 x_2. So in general, we had f the nth order divided difference, or the n minus 1th order actually. So it is going x $_0$ x $_1$ into x $_n$ as the n minus 2, the difference of n minus 2 order divided differences. That is, f going from x_1 to x_2, x_0 to x_n minus 1 divided by x_n minus x_0. So that is the way divided difference table. So to summarize, the table would be given something like this. We said this is, we saw this in the last lecture that this is, we have a set of data points x_0, x_1, x_2, x_3, x_4, and then we had the function values which are f_0, f_1, f_2, f_3, f_4.

We had the first order divided differences, that is, f_1 minus f_0 divided by x_1 minus x_0 as f "x_0 x_1" f_2 minus f_1 divided by x_2 minus x_1 as f "x_1 x_2" f_3 minus f_2 divided by x_3 minus x_2 as f "x_2 x_3" and f_4 minus f_3 divided by x 4 minus x_3 as f "x_3 x_4", etcetera. That is the first order divided difference. And then we had second order divided differences which is the difference of these first order differences, that is, f "x_0 x_1 x $_2$" as f "x_1 x_2" minus f "x_0 x_1" divided by x_2 minus x_0 and f "x_2 x_3" minus f "x_1 x_2 divided by x_3 minus x_1 is given by this.

(Refer Slide Time: 05:50)



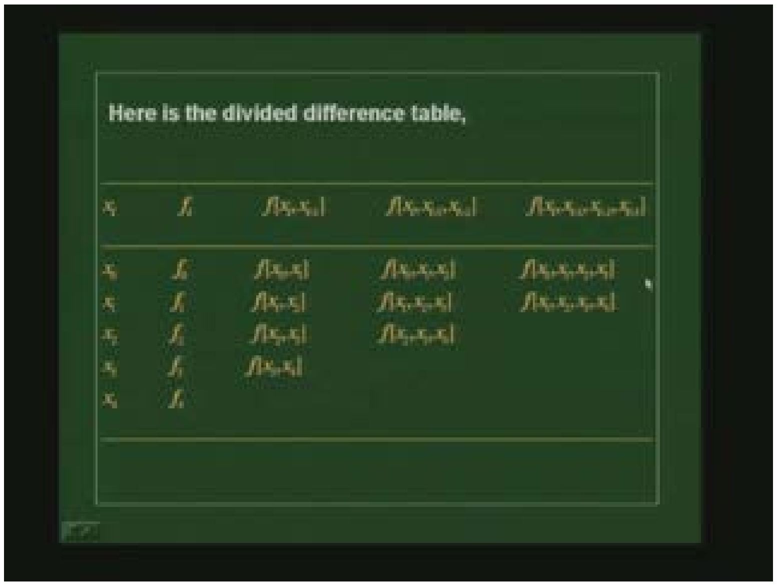
Higher order differences are defined in terms of lower-order differences. For example,

$$f[x_0,x_1,x_2]=\frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$$

In general,

$$f[x_0,x_1,\ldots,x_n]=\frac{f[x_1,x_2,\ldots,x_n]-f[x_0,x_1,\ldots,x_{n-1}]}{x_n-x_0}$$

(Refer Slide Time: 07:30)



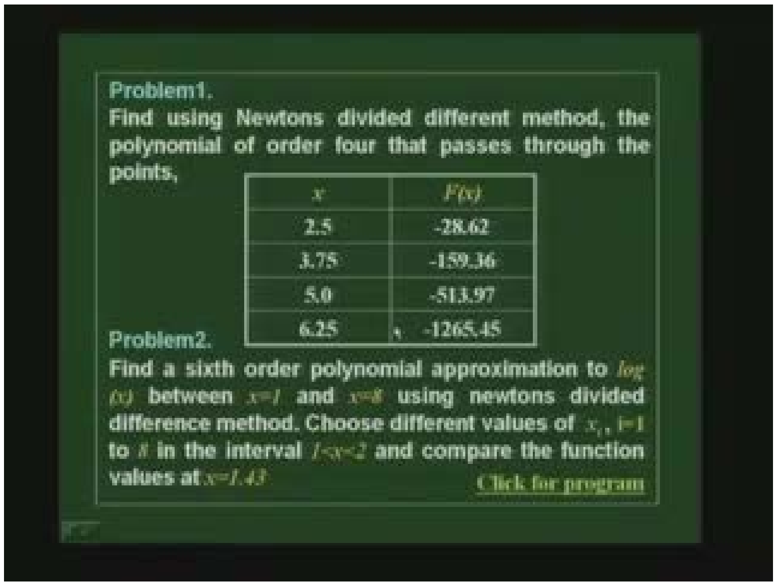
Here is the divided difference table,

x_i	f_i	$f[x_i,x_{i+1}]$	$f[x_i,x_{i+1},x_{i+2}]$	$f[x_i,x_{i+1},x_{i+2},x_{i+3}]$
x_0	f_0	$f[x_0,x_1]$	$f[x_0,x_1,x_2]$	$f[x_0,x_1,x_2,x_3]$
x_1	f_1	$f[x_1,x_2]$	$f[x_1,x_2,x_3]$	$f[x_1,x_2,x_3,x_4]$
x_2	f_2	$f[x_2,x_3]$	$f[x_2,x_3,x_4]$	
x_3	f_3	$f[x_3,x_4]$		
x_4	f_4			

And f "x_3 x_4" minus f "x_2 x_3" divided by x_4 minus x_2 is given by this, and then further divided difference is given by this, and then you can have one more, one more term which is the difference between these two divided by x_4 minus x_0. That will be the last term of this. So once you have the divided difference table we can construct the polynomial which we saw as, which we now just saw. So we will have the Newton's form polynomial given by p_n of x, as we just saw, as "a_0" plus x minus x_0 into f of x_0, x_1 and f of x minus x_0 into x minus x $_1$ into f of x_0 x_1 x_2, etcetera, that is what we saw. And we also saw that this can be written in a nested form, that is, we could write it as a_0 plus x minus x_0 into f of x_0, this is in square bracket, please note this is simple, so in square bracket, plus x minus x_1 into f of x_0 x_1 x_2 and plus x minus x_2 into f of x_0 x_1 x_2 x $_3$, and continuing that way we could write it in this and this is easier to implement numerically.

There are other forms of this same polynomial. We also saw that given a set of n points the polynomial which goes through that, that is, if given a set of n points starting from 0, x and minus 1, the polynomial of order n minus 1 which goes through all these points is a unique polynomial. But this is one of the forms. There are many other forms of writing the same polynomial, but the polynomial is unique. But you could write it in different ways. So, here is a problem which you would see, how, we would see later towards the end of today's lecture, that we would see this thing actually being implemented, and we will actually construct the Newton's form polynomial and use the divided difference to construct a polynomial which goes through all these points.

(Refer Slide Time: 10:01)



Problem1.

Find using Newtons divided different method, the polynomial of order four that passes through the points,

x	$F(x)$
2.5	-28.62
3.75	-159.36
5.0	-513.97
6.25	-1265.45

Problem2.

Find a sixth order polynomial approximation to $log(x)$ between $x=1$ and $x=8$ using newtons divided difference method. Choose different values of x_i, $i=1$ to 8 in the interval $1<x<2$ and compare the function values at $x=1.43$

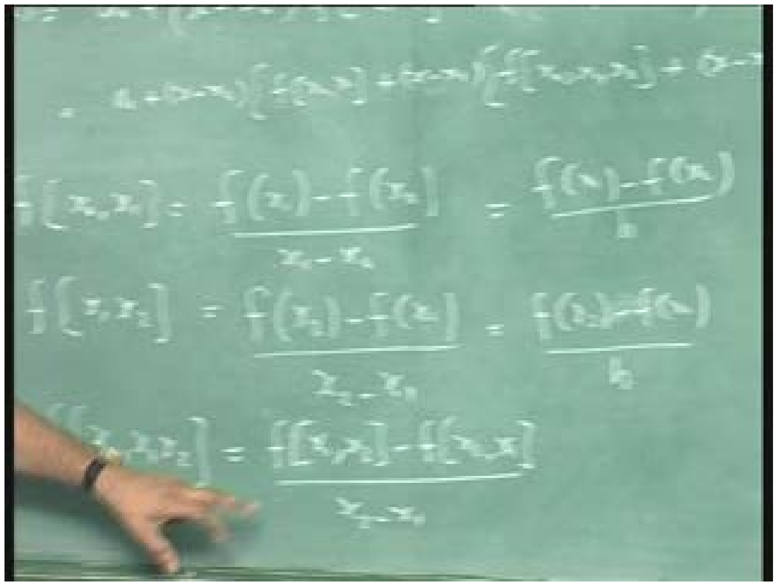
Click for program

So now, going from here, as we said, we just wrote this in this form, and let us say, a special case of this is that where we had these differences, x_0, x_1, x_2, x_3 etcetera, written. And let us consider a case where x 0 minus x $_1$ or x $_1$ minus x $_0$ is same as x_2 minus x_1, is same as x_3 minus x_1. That is, we could write any nth order object as any nth order position, as x_0 plus simply n times some interval h. So, if this kind of, in this special case where the distance between any two points is the same, that is, they are equally spaced, this polynomial can be written in a slightly different form. This is basically the divided difference for this f to obtain f x_0 x_1, f x_0 x_1 x_2, f x_0 x_1 x_2 x_3 etcetera, can be written in slightly different form.

We just saw that f_0, this is the way we write this quantity, is that f of x_0 x_1 we wrote as f of x_1 minus f of x_0 divided by x_1 minus x_0. We said that, and we would say that the same first order difference as x_1 x_2, the first order differences as f of x_2 minus f of x_1 divided by x_2 minus x_1, x_1 minus x_0, x_2 minus x_1, they are all the same. They are equal to h. So we do not need to write this in the results table, and we could just put that into this function here, this part here. So if that is true, that is, and then I would have f of x_0

x_1 x_2, you remember, is we were writing this as f of x_0 x_1 x_2 minus f of, f of f_1 x_2 minus f of x_0 x_1 divided by x $_2$ minus x $_1$. That is what we were writing before, and we know that this can be written as f x_1 minus x_2, if you remember, f x_1 x_2 here is simply f of x_2 minus f of x_1 divided by h x_2 minus x_1 is the h, and while this also is f of x_1 minus f of x_0 divided by h.

(Refer Slide Time: 13:21)



$$= a_0 + (x-x_0)\{f[x_0,x_1] + (x-x_1)\{f[x_0,x_1,x_2] + (x-$$

$$f[x_0,x_1] = \frac{f(x_1)-f(x_0)}{x_1-x_0} = \frac{f(x_1)-f(x_0)}{h}$$

$$f[x_1,x_2] = \frac{f(x_2)-f(x_1)}{x_2-x_1} = \frac{f(x_2)-f(x_1)}{h}$$

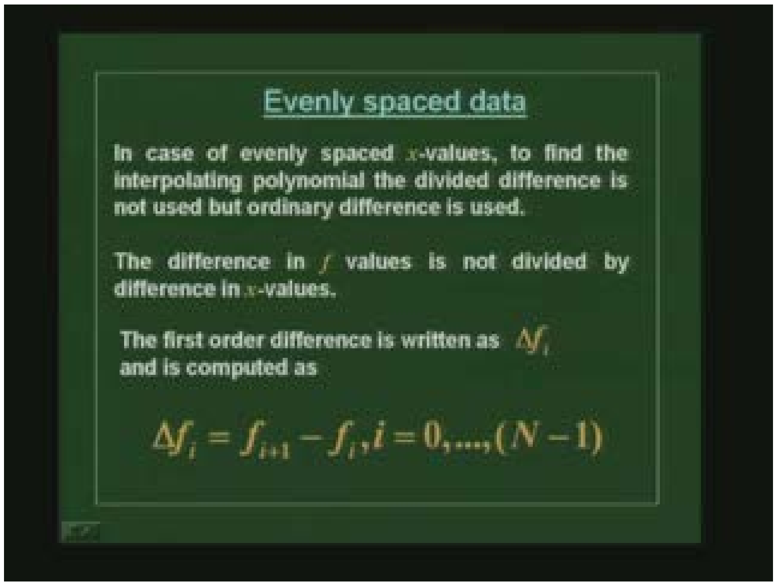
$$f[x_0,x_1,x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0}$$

So I can substitute all that into this and write this as f of x_2. I am substituting for this f of x_1 x_2 from this, and f of x_0 x_1 from that, and I can write that as f of x_2 minus f of x_1 minus f of x_1 plus f of x_0 divided by, so we have x_2 minus x_1 here, this is x_2 minus x_0, x_2 minus x_0 is 2 h. There is h coming from here, so it will be just 2 h square.

So we can rewrite this thing in a slightly different fashion and we can simply, we can write it as f of x_2 minus 2 f of x_1 plus f of x_0, and we can substitute that here, so when you substitute that these formulas for f of x_0 x_1 and f of x_0 x_1 x_2 using these functions, and we write them here. We can write it as, we can write then this quantity as p_n of x equal to "a_0" plus, and then I would write this quantity as f of x_0 minus f of x_1 as I said. So we will write x minus x $_0$ into f of x_1 minus f of x_0 now, divided by x_0 minus x_1, x_1 minus x_0 which is h. I will put it here, and then I will write this as x minus x_0 into x minus x_1. So x is the point at which you want to evaluate this polynomial.

So we will write x minus x_0. So x is the point at which you have to evaluate the polynomial, and x_0 x_1 x_2, etcetera, are the points at which the data is tabulated. So we will write x $_0$ minus x $_1$, x minus x_0, x minus x_1 into, now this polynomial it will be written, this coefficient, we are going to substitute this, so we write f of x_2 minus f of 2 f of 2 times f of x_1 plus f of x_0 divided by 2 h square. That, we will put in here. So that is the form we are going to write, and we will continue a series from here.

(Refer Slide Time: 16:15)



Evenly spaced data

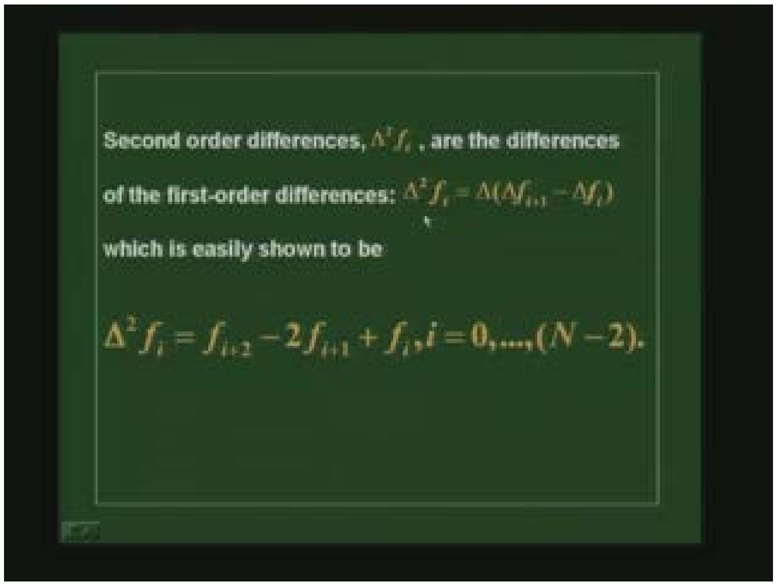
In case of evenly spaced x-values, to find the interpolating polynomial the divided difference is not used but ordinary difference is used.

The difference in f values is not divided by difference in x-values.

The first order difference is written as Δf_i and is computed as

$$\Delta f_i = f_{i+1} - f_i, i = 0, \ldots, (N-1)$$

(Refer Slide Time: 16:17)

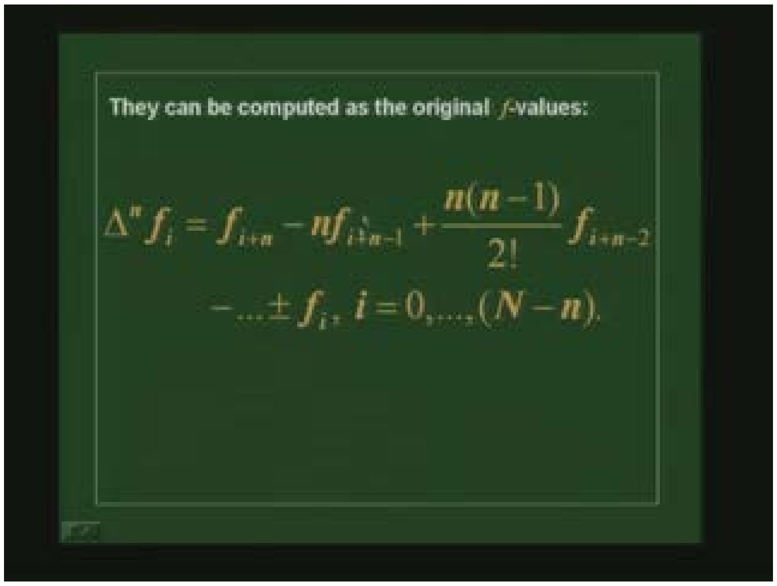


Second order differences, $\Delta^2 f_i$, are the differences of the first-order differences: $\Delta^2 f_i = \Delta(\Delta f_{i+1} - \Delta f_i)$ which is easily shown to be

$$\Delta^2 f_i = f_{i+2} - 2f_{i+1} + f_i, i = 0, \ldots, (N-2).$$

Now that notation which is being used, which was shown here, is that we will say that f of i plus 1 minus f of i as delta f_i, and the second order difference as del square f $_i$, and which is delta of delta f of i plus 1 minus delta f_i. so that is del square f_i which is what we have written on the board and we would write in general the higher order differences del to the power n f of i.

(Refer Slide Time: 16:43)



They can be computed as the original f-values:

$$\Delta^n f_i = f_{i+n} - n f_{i+n-1} + \frac{n(n-1)}{2!} f_{i+n-2} - \ldots \pm f_i, \; i = 0, \ldots, (N-n).$$

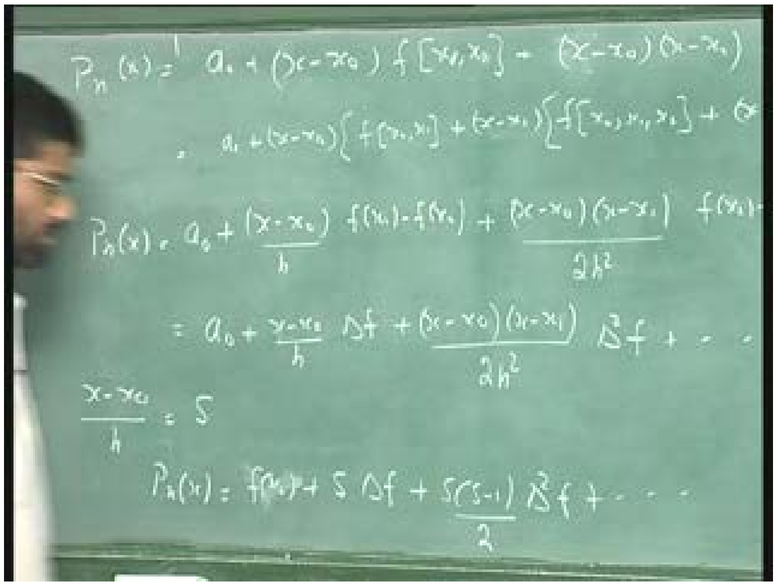
That is the nth order divided difference, now will be written in this form, that is, f_i, n plus 1 n times f_i plus n minus 1 and plus n into n minus 1 by 2 factorial into f_i plus n minus 2 up to f_i, i going from all the way to 0 to n minus n.

So the notation which we are using is basically plus you would say that is delta f and delta f into f of x_1 minus f of x_0. That is what we, x minus x_0 divided by h into delta f, and then you will have x minus x_0 divided by 2 h square into del square f, and we will continue the series in this fashion. So we would call this as, if I call x minus x_0 by h as s, then I can, and then I will introduce this new notation which is x minus x_0 by h which we would call that as s.

Then I can show I am just rearranging this data here, rearranging the points here, that p n of x can be written a_0 which is f of x_0 which is actually f_0. So we will just, f_0 plus, we will write this as f_0 plus s into delta f plus s into s minus 1 by 2 into del square f, and we can continue writing in this form, and you would write n as s into s minus 1 into s minus, we can enter the last term, would be s into s minus 1 into s minus of n minus 1 whole divided by n factorial times del n of “f”. So that is what we would see here. So I can finally write the whole polynomial in this fashion. So that is what I have written on the board now.

That is, f_0 plus s delta f_0 s into s minus 1 by 2 factorial del square $f_{0.}$ Next term would be s into s minus 1 into s minus 2 by 3 factorial into del 3 f_0, and continue up to s into s minus 1, s minus n plus 1, that is s minus of n minus 1 divided by n factorial del n f_0. So remember the notation which we use here is s as x s minus x_0 by h, with h given by delta x the uniform x value.

(Refer Slide Time: 18:36)



$$P_n(x) = a_0 + (x - x_0) f[x_1, x_0] + (x - x_0)(x - x_1)$$

$$= a_0 + (x - x_0)\{ f[x_0, x_1] + (x - x_1)[f[x_0, x_1, x_2] + \cdots$$

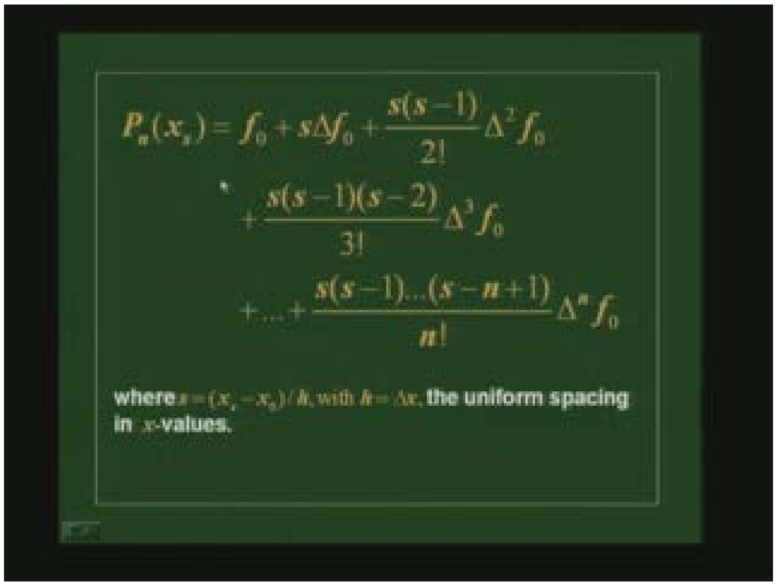
$$P_n(x) = a_0 + \frac{(x - x_0)}{h} (f(x_1) - f(x_0)) + \frac{(x - x_0)(x - x_1)}{2h^2} f(x_2) \cdots$$

$$= a_0 + \frac{x - x_0}{h} \Delta f + \frac{(x - x_0)(x - x_1)}{2h^2} \Delta^2 f + \cdots$$

$$\frac{x - x_0}{h} = s$$

$$P_n(x) = f(x_0) + s \Delta f + \frac{s(s-1)}{2} \Delta^2 f + \cdots$$

(Refer Slide Time: 19:15)



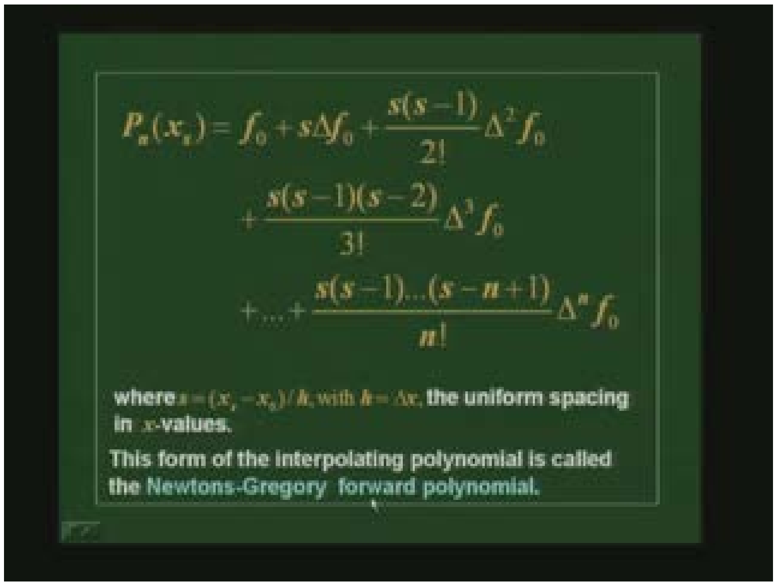
$$P_n(x_s) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \ldots + \frac{s(s-1)\ldots(s-n+1)}{n!}\Delta^n f_0$$

where $s = (x_s - x_0)/h$, with $h = \Delta x$, the uniform spacing in x-values.

That is, x_s here is the point at which we want to evaluate the polynomial. So what we used here on the board is that we evaluated the polynomial at x, and we have x_0 x_1 x_2 x_3 as the point in which it is tabulated, and we use x minus x_0 by h as s, and I can write the polynomial as of order n, n minus 1 as p n of x as f of x_0 plus s delta f plus s into s minus 1 by 2 del square f, etcetera.

This is true when h here is, remember h is equal to x_i plus 1 minus x_i, so that is equally spaced data. So now, this form of the polynomial is known as the Newton-Gregory forward polynomial, and we would again see the implementation of this in a program in a short while.

(Refer Slide Time: 20:48)



$$P_n(x_s) = f_0 + s\Delta f_0 + \frac{s(s-1)}{2!}\Delta^2 f_0 + \frac{s(s-1)(s-2)}{3!}\Delta^3 f_0 + \ldots + \frac{s(s-1)\ldots(s-n+1)}{n!}\Delta^n f_0$$

where $s = (x_s - x_0)/h$, with $h = \Delta x$, the uniform spacing in x-values.

This form of the interpolating polynomial is called the Newtons-Gregory forward polynomial.

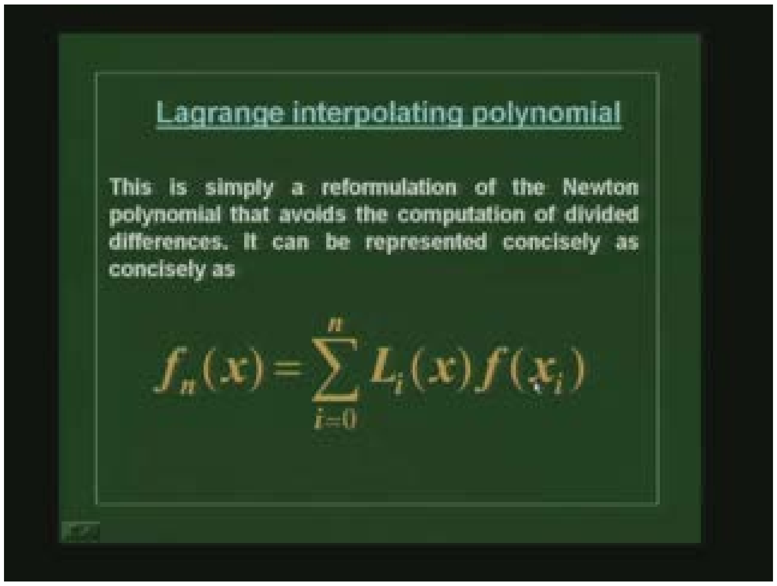
So now, there is, as I said, the polynomial is unique, but there are many ways of writing this polynomial. So, and we saw one way of writing this, and this is called the Newton's form, and we write this particular form because it is easy for analysis.

For example, if you want to get an estimate for the error in the polynomial, as I said, even though this polynomial function, it passes through all the points, all the n points which we have tabulated, it is still an approximate function of the real data. So the question is what is the error in this approximation that can be estimated in this particular form? We will also see that error estimation in a little while, but there are these, however, this particular form is a little difficult to implement in a code because we need to evaluate all these coefficients f of x_0 x_1, f of x_0 x_1 x_2, etcetera.

Even though there is a systematic way of doing that, by using the divided difference method, it is still rather cumbersome. So, we can use a slightly different form of this. Instead of writing this polynomial in this fashion, we can, we could write this nth order polynomial in a slightly different way, and that is what we would see in the Lagrange interpolating polynomial form.

So we will see that now. So in the Lagrange interpolation formula we would write in the polynomial simply in a symbolic way as this. We use f n of x as the nth order polynomial, and we would write it as l_i of x f of x_i, so function evaluated at x_i into some coefficient l_i of x. x is not a coefficient, it is some other function l $_i$ of x. It is also, this is evaluated at the point x where we want to evaluate the polynomial is, so the trouble with this is that every time we want to evaluate the function f of x, the polynomial at any point x, we have to reconstruct this. The idea comes from this that we can write, we use the same notation. So we will write it as f n of x to represent the Lagrange form.

(Refer Slide Time: 23:00)



Lagrange interpolating polynomial

This is simply a reformulation of the Newton polynomial that avoids the computation of divided differences. It can be represented concisely as concisely as

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i)$$

So there we would write this as a polynomial. Let us say we take a function which goes through 3 points. So we take n as 3, that is, we have, we had x_0 as f of x_0 as f_0, the function x_1 and f of f_1 and x_2 as f_2, and then we would write the polynomial in this fashion, that is, we will write x minus x_1 into x minus x_2 into, let me call it a 1. We will see in just a minute what that a 1 is, and then x minus x $_0$ into x minus x_2 into a 2 plus x minus x_0 into x minus x_1 into a 3. So we have 3 points: x_0, x_1, x_2, and the function values of those points are f_3, f_1, f_2, the function values, not the polynomial values.

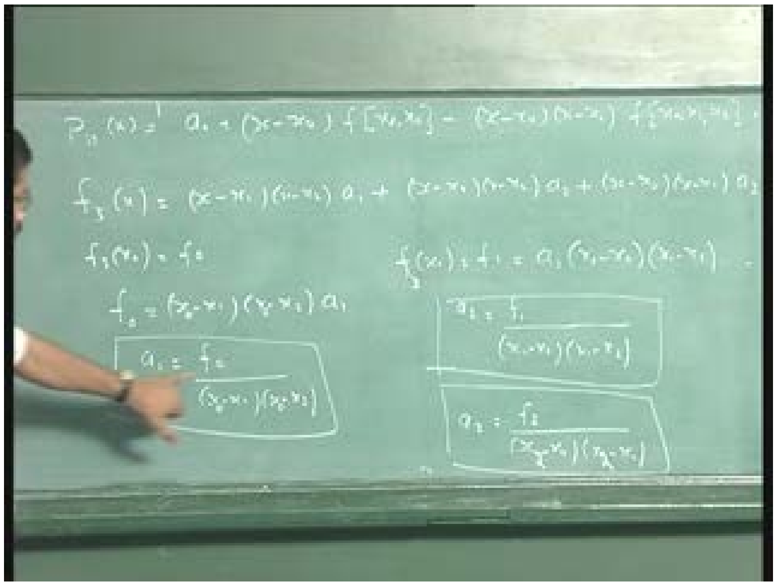
So that is what we have, and then I wrote this, this polynomial here which goes through x_1 x_2 and x_3.That means, I have f_3 of x_1 is f $_1$ or f $_3$ of x $_0$ equal to f $_0$, etcetera. It should satisfy that. So that should give us, since it should pass through all these 3 points, I can evaluate a_1, a_2 and a_3, and let us see what do we get. So, if you just insist that this polynomial f_3 at x_0 is f_0, then we would get from this as f_0 is equal to x minus x_1 into x minus x_2 into a $_1$, because at x equal to x_0 this and this go to 0. So we have only this term left.

This is a function value at x equal to x_0 is f_0.I could say that f $_0$ is x minus x $_1$ plus x minus x_2 plus a_1. So that gives me what a 1 is. I can evaluate a_1 as f_0 divided by x minus x_1 into x minus x_2. So then I can substitute that here, and similarly I could get a_2 to see that x equal to x_1 and then a_2 should be given by, since at x equal to x $_1$, so from f of x_n, that is, in this case 3 f x_1 is f_1. We will get from this polynomial as a_2 into x, I made a mistake here: f_0 is, f of x_0 is f_0, f_0 is x_0 minus x_1. so that is, f_0 minus x_1, x minus x_2.

So I repeat that here: so f of f_3 is my polynomial and evaluated at x_0, so that is, f_3 at x_0, that is, f_0, my value. So it evaluated at x $_0$. It will be x_0 minus x_1, x_0 minus x_2 into a $_1$. So a_1 is f_0 by x_0 minus x_1 into x_0 minus x_2. Similarly here, I can now evaluate this polynomial at x_1. Now I evaluate this at x_1, and I evaluate at x $_1$. I get f_3 of x_1, which is f_1, and that will be a_2 times x_1 minus x $_0$ into x $_1$ minus x $_2$, or my "a $_2$" here is f 1 divided by x_1 minus x_0 into x_1 minus x_2. So I can write it like this.

So I can evaluate all the coefficients now; a_1 here, a_2 here, and in a very simple similar fashion, you can get a 3 as f_2 divided by x_2 minus x_0 into x_2 minus x_1, a_3, would be x_2 minus x_0 into x_3 minus x_0 into x_3 minus x_1. Substitute x_3 that is the value I would get as f of x_0, f of x_2. This is correct; x_2 minus x_1. So I get all the three coefficients from this by evaluating the function at x_0 x_1 and x_2. I can get the coefficients, a_1, a_2, a_3, which I just substitute here.

(Refer Slide Time: 28:04)



$f_3(x_0) = f_0$
$f_0 = (x_0 - x_1)(x_0 - x_2)\, a_1$
$a_1 = \frac{f_0}{(x_0 - x_1)(x_0 - x_2)}$
$a_2 = \frac{f_1}{(x_1 - x_0)(x_1 - x_2)}$
$a_3 = \frac{f_2}{(x_2 - x_0)(x_2 - x_1)}$

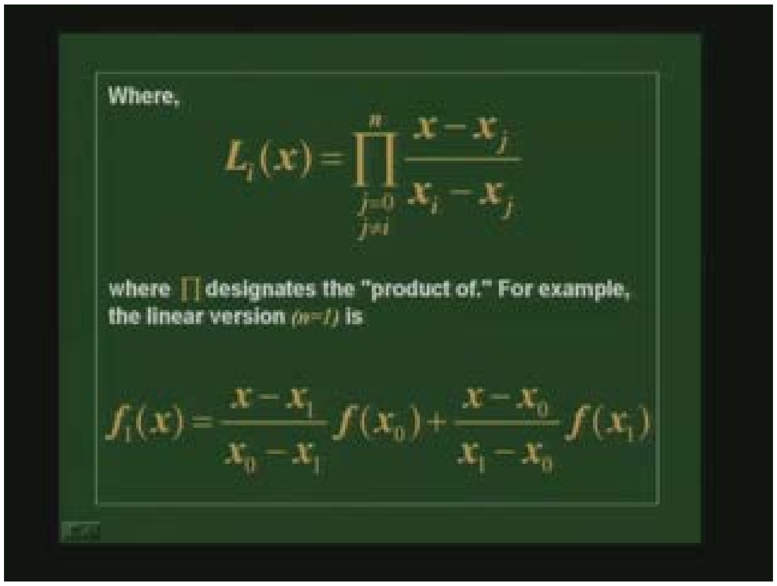
So this, now if you substitute that whole function into this value, we could write this polynomial now as x minus x_1 into x minus x_2 divided by a_1 was this. So it is x_0 minus x_1 into x_0 into x_0 minus x_2 into f at x_1 which we call f_1, and then I have x minus x_0 into x minus x_2, a_2 is this, which is x_1 minus x_0 into x_1 minus x_2, so which is then f of x. So, f evaluated at x_0 here, and f evaluated at x_1 plus x minus x_0 into x minus x_1 divided by x_2 minus x_0 into x_2 minus x_1, and f evaluated at x_2.

That is the polynomial which we are getting. So that is the final form of the polynomial which we have, and that is exactly what is written here in this fashion. So we have, it is written as, so in this particular case we would have n equal to 2, that is going from 0 2 and 3, and we will have, so we can see what l_i of x would be. So l_i of x is basically representing these functions. So that is what this would be, what I call now as l_0, and l_1, and l_2. So then I can write this as sigma l_i of x. It is a function of x, as you can see, into f of x_i. So that is what we have written here, l_i of x and f of x_i.

So the l_i of x, then it can be written as phi, that is the product x minus x_j divided by x_i minus x_0, where j runs over all points from 0 to n except j equal to i. So when you say l 0, j goes from 1 to n, and when you say l_1, j goes from 0 to n, except i equal to, j equal to 1. So that is, we can see that from the, this from is exactly the same as the form which is given here. When you take the l_0, for example, we are finding the product between x minus x_1 divided by x_0 minus x_1 into x minus x_2 divided by x_0 minus x_2. So I started with

l_0, i is 0, l 0. So i is 0. So x_i x_0 minus x_j, and x minus x_j, where j goes from 0 to, it goes from 1 to 2. So, x minus x_j x_0 minus x_j into x minus x_j by x_0 minus x_j. j is 1 here, j is 2 here, and we come to 1.

(Refer Slide Time: 30:17)



Where,

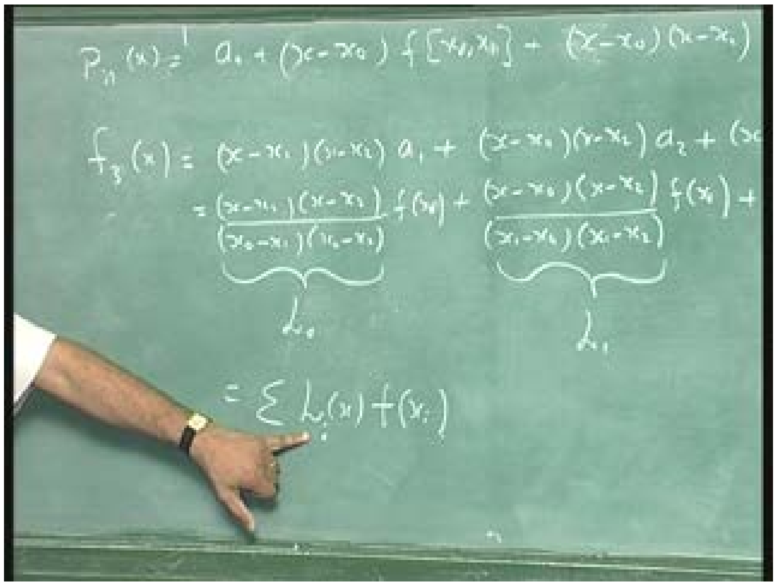
$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

where $\prod$ designates the "product of." For example, the linear version $(n=1)$ is

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

It is x minus x_0 divided by x_1 minus x_0, that is, x minus x_j divided by x_1 minus x_0, because i is 1 multiplied by x minus x $_2$ by x_1 minus x_2.We skip x minus x_1. That is because i is 1. So j will go to 0 and 2. So that is what we have seen here. So we have the first order term, the first order polynomial here, and the second order function would be polynomial, would be given by something like this.

(Refer Slide Time: 30:50)



$$P_n(x) = a_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1)$$

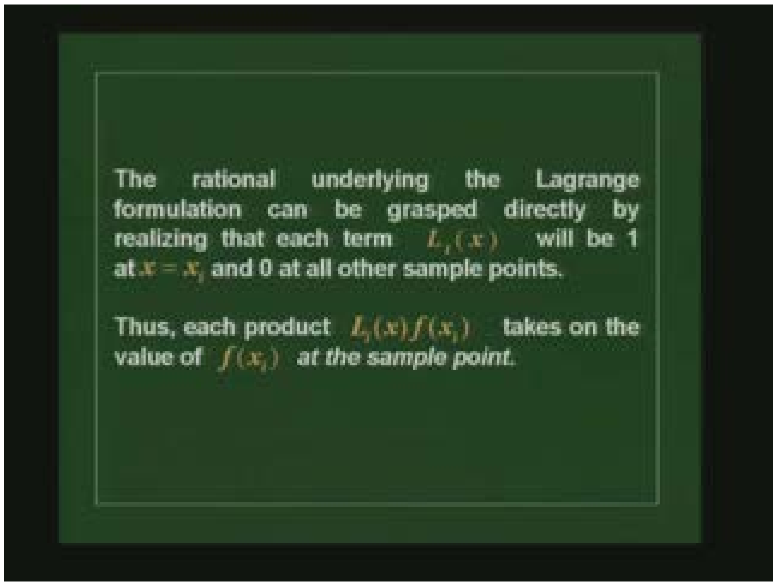
$$f_3(x) = (x - x_1)(x - x_2)\, a_1 + (x - x_0)(x - x_2)\, a_2 + (x$$

$$= \underbrace{\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}}_{L_0} f(x_0) + \underbrace{\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}}_{L_1} f(x_1) +$$

$$= \sum L_i(x) f(x_i)$$

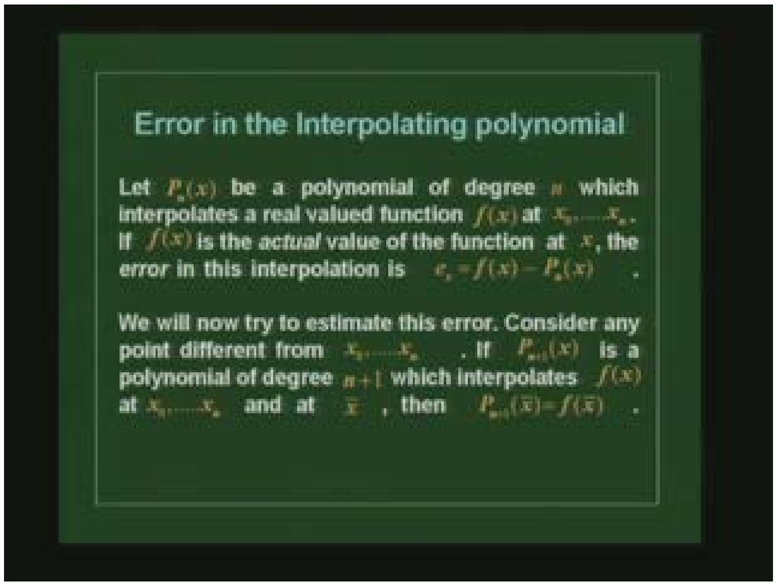
That is what we have on the board, etcetera. In general, we could write the nth order polynomial which we saw l_i of x into f of x_i. So the only trouble here is that each time we have to compute, as we just saw that, each time when we evaluate this polynomial, each time we have to compute the function, this li function, until this point. So, if you have changed the point at which we are going to evaluate this polynomial, and then we have to again compute each of these coefficients, unlike the case where we have the Newton's form where once we have evaluated these coefficients at given value of points, we could just go on, evaluate the polynomial at every point which you want.

(Refer Slide Time: 32:15)


The rational underlying the Lagrange formulation can be grasped directly by realizing that each term $L_i(x)$ will be 1 at $x = x_i$ and 0 at all other sample points.

Thus, each product $L_i(x) f(x_i)$ takes on the value of $f(x_i)$ *at the sample point.*

(Refer Slide Time: 33:20)

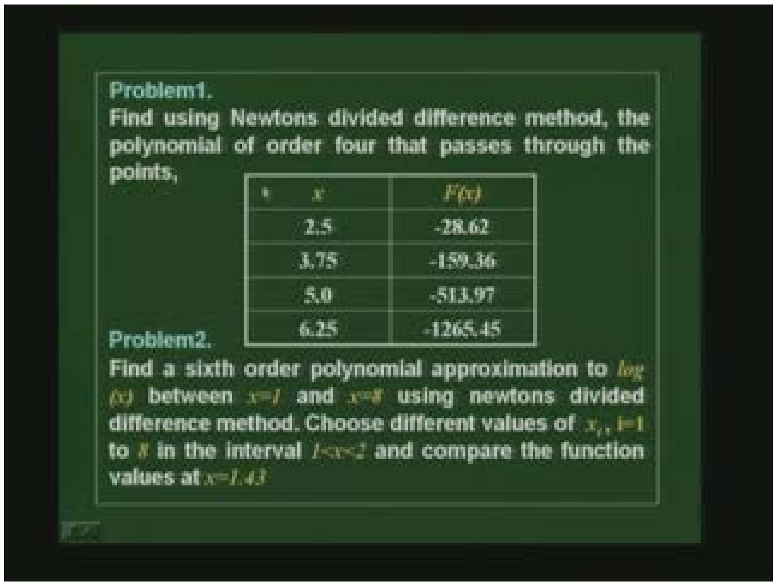

Error in the Interpolating polynomial

Let $P_n(x)$ be a polynomial of degree n which interpolates a real valued function $f(x)$ at $x_0, \ldots, x_n$. If $f(x)$ is the *actual* value of the function at x, the *error* in this interpolation is $e_n = f(x) - P_n(x)$.

We will now try to estimate this error. Consider any point different from $x_0, \ldots, x_n$. If $P_{n+1}(x)$ is a polynomial of degree $n+1$ which interpolates $f(x)$ at $x_0, \ldots, x_n$ and at $\bar{x}$, then $P_{n+1}(\bar{x}) = f(\bar{x})$.

That is the slight difference between these two, but this is easy to implement. This is much easier to implement on a numerical, on a program than this, but this is more amenable to error analysis. So that is the basic difference between these two. We will see the implementation of both of these in a short while. As I said that now both are, now we can do some error analysis.

(Refer Slide Time: 33:53)



Problem1.
Find using Newtons divided difference method, the polynomial of order four that passes through the points,

x	$F(x)$
2.5	-28.62
3.75	-159.36
5.0	-513.97
6.25	-1265.45

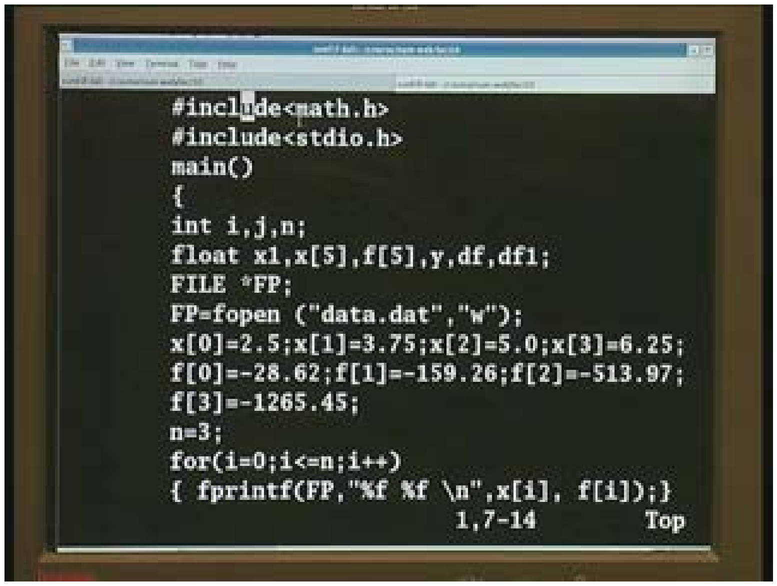
Problem2.
Find a sixth order polynomial approximation to $log(x)$ between $x=1$ and $x=8$ using newtons divided difference method. Choose different values of x_i, $i=1$ to 8 in the interval $1<x<2$ and compare the function values at $x=1.43$

We should know since these are approximate polynomials, approximate functions to the real data, we need to know what is the error in this function and that can be evaluated by using this form of the polynomial, the Newton form, and that is what we would see in the next, in the next. So, before we go into actually evaluating the error in an interpolating polynomial using the Newton's form, we would just see the implementation of the Newton's form in a program, an algorithm, which would implement the Newton's form in a program. So here is a problem which we discussed in the last lecture. That is, we have a set of points "2.5", "3.75", "5.0", "6.25", and the function values tabulated at these points, and we want a polynomial.

So, we will have a polynomial of order 3 since there are 4 points here which pass through these points. We will try to get this polynomial used in the Newton's form. We will try to implement this here. That is what we would try to do now. So here is a program which does that. So it is a very simple program.

I have not used anything complicated in this thing. You can write this in a more optimized and better way, using rather involved c functions. Here, we look at a very simple implementation of this program. So we have tabulated, so I have now the main program, and then I have variables x of 5 dimension, 5 f. So these are function values, and these are the independent variables, and these are 2 arrays, and I have tabulated all the function values here, that is, "2.5" is x of 0, and the function value "minus 28.62", and x of 1 "3.7" minus f function value, "minus 151.26", etcetera.

(Refer Slide Time: 34:25)



```
#include<math.h>
#include<stdio.h>
main()
{
int i,j,n;
float x1,x[5],f[5],y,df,df1;
FILE *FP;
FP=fopen ("data.dat","w");
x[0]=2.5;x[1]=3.75;x[2]=5.0;x[3]=6.25;
f[0]=-28.62;f[1]=-159.26;f[2]=-513.97;
f[3]=-1265.45;
n=3;
for(i=0;i<=n;i++)
{ fprintf(FP,"%f %f \n",x[i], f[i]);}
                          1,7-14        Top
```

So they are exactly the same, the problem given here. So we have just tabulated all these values, and I just opened a file here. So this is also, I also want to use this program to demonstrate some programming. So here I have used, I have opened a file called “data.dat”, and in which, using this file pointer, fp, I open this file, and then I would write this function, all the function values, the independent variable, and the function values, into that data.dat here.

So, n is 3.There are 0 to 3, so I go from 0 to 3. So there are 4 data points, and I write all of them using, since they are floating points, I use percentage f here, percentage f, and I will write down all the functions, all the values of x and the function values, and then after finishing that right now, I close this function. I close this data file which is fp function pointer fp pointing to which is data.dat, that is just writing out that file. I just stored the tabulated values into this file, and now I have had to construct the divided difference, so you remember, the divided difference, we just go back and see the divided difference again. Here is the divided difference table.

So here is the table. So we have the, the x values, and the function values, and then we have to construct this divided difference table and the coefficients of the polynomial, remember, are f_0, f of x_0 x_1, f of x_0 x_1 x_2, f of x_0 x_1 x_2 x_3.That is the coefficients of the polynomial are.

So even though we have to evaluate, we have to calculate all these differences. To calculate these differences, so if you want to calculate these values, say, f_0, f x_0 x_1, f x_0 x_1 x_2, f x_0 x_1 x_2 x_3, we need first to evaluate all those differences because these are differences of these difference functions but we do not need to store them. We do not need to store all of this because we need store only these coefficients, because that is the coefficients we are going to use. So, since we are going to use only these coefficients, the top line, we do not need to store all these values, but we need them as intermediate points. So we need only to evaluate the top points. So that is what we see. So this program demonstrates how we actually do this. So we have, I call them, I have these functions here. So there is f value. I have the function itself tabulated in this array.

So this function itself is not important to me, once I have computed the function values at different points, are not important to me after once I have calculated the coefficients of using the divided difference. So I can use this same array here to store the coefficients of the divided difference, or the divided differences are stored into this particular array itself. That is what we are doing here. So I compute the first. So I have two loops here, j and i. So j is the outer loop, and i is the inner loop. So we go first level.

These calculations are done using the i loop, and the j loop shifts me from these calculation to these calculations, and then to this, and then to the last one. So what I am going to do is, as you can see, I choose j equal to 0. So that is the first level of computation, and then I run i going from 1 0 to n, i goes from 0 to n. j is 0. So i goes from 0 to n. So I will take 0, n in this case is 3. So I get 0, 1, 2, 3 and then I take the difference between this f 1 minus f_0 and divide it by x_1 minus x_0, and store it in, that gives me this, which I store basically inside f_0 itself. So that is what the points is. I hope I can show this to you here.

So I took j equal to 0, and I store the f of j f of 0. I am storing, I am just temporarily storing it in some variable called df. I store this f of 0 there, and then I go to the next step and I run this i loop, and then I am storing this f of 0 again, or this df here, which is actually the 0th divided difference.

We just simply, f of 0 itself, as df 1, and then I take the first divided difference here, so j is 0, remember, and i is 1, and I take f of 1 minus f of 0, j is 0, i is 1. So it is f of 1, i is 1 here. f of 1 minus f of 0 divided by x of 1 minus x of 0. That, I have now. I put that into the variable called df, the first divided difference, or the first points f_0 and f_1, and this line is executed only if i is greater than 1. Since i is not greater than 1, we ignore it now. We go back here, and now i is 2. So when i is 2, we come back here, and now this value which we computed, that is, f_1 minus f_0 divided by x_1 minus x_0 is now stored into df_1, and I compute here now, since i is 1, j is 0, i is 2, j is 0, that is, f_2 minus f1 divided by x_2 minus x_1. i is 2, so j is 0, so x_2 minus x_1.

So that is the first divided difference between the second and third variables, that is 1 and 2, second and third variables. So that is the second divided difference, and now since i is greater than 1, so that df 1 is now stored as f_0, Remember, i is 2 here. i is greater than 1. j is 0, i is 2, so f_1, I will use the function value f_1 which I do not need anymore. I store, I use that point, f_1, to store df 1 which was f_1 minus f_0 divided by x_1 minus x_0. This is clear. So we are, what I have just done is the following. I have just used, I have computed f_1 minus f_0 divided by x_1 minus x_0, and I stored it in a temporary variable, and then I used f_2 minus f_1 by x_2 minus x_1, and once I have done this second computation, that is, f_2 minus f_1 divided by x_2 minus x_1, I do not need f $_1$ anymore, because I have done f_1 minus f_0 by x_1 minus x_0, and I have done f_2 minus f_1 by x_2 minus x_1.

So I do not need f_1 anymore. So I choose the f_1 to store this particular variable into that. So I store f_1 in f_1, I store this. Now similarly, in the next term, as I go over the i loop here, as I go further in i loop, I use f, I compute f_3 minus f_2 divided by x_3 minus $x_{2,}$ again the first divided difference between x_3 and x_2, and then I would, that is, I would, I am computing f_3 minus f_2 divided by x_3 minus x_2, and in the previous step I had computed f_2 minus f_1 by x_2 minus x_1. so now I do not need this f $_2$ anymore.

So the value which I had computed as f_2 minus f_1 by x_2 minus x_1, that is, f x_1 x_2, I will store in f $_1$. Similarly, f x_2 x_3, I will store in f 2. So, at the end of the first i loop, I have now f_0 and f_1, storing f x_0 x_1 f_2, storing f x_1 x_2 and f_3, storing f x_2 x_3, n is already 3.If I have n equal to 4, and f 4 will store f x_3 x_4, so that is what I will have. So I have f_0 and first divided differences stored to these variables.

So that is what this program is basically doing. So I do not need to have, this is important because we will be using large number of data points at some points. This is only for n equal to 3, but even if I will be doing for large number of data points later, and we cannot have an array storage, storing for every intermediate difference which is not needed. So we do not need to use those data points. So we do not need them. We do not use those intermediate divided differences. So we do not need to store them. So here we are using the function itself to store the divided difference, and then we go to j equal to 1. That is the second divided difference, and again, we do exactly the same thing.

So here so now j equal to n minus 1, is what we are going to go here. So when j is equal to 1 now, we are computing the next divided difference. Now you remember I can use exactly the same functional form here. I do not need to change when I want j equal to 1. I do not need to change this program because this part, because now, my f is the first divided difference.

Now my f is storing the first divided difference, and so I can use that f itself to compute the second divided difference as the difference between the first divided differences, so that is what is done here. So when j is 1, so again we go from i equal to 1 to n minus j. Remember, I start from i equal to 1 and f of 1 is storing my first divided difference, and f of 2 is the first divided difference between the points 3 and 2, and f of 3 is storing between that, between 4 and 3, etcetera, 3 and 2, etcetera. So I can use this thing here. So I just use i equal to 1 and now get f of 2. j is now 1. So I can take f of 2 minus f of 1. j is 2 now, i is 1. j is 1 now, and i is 1.

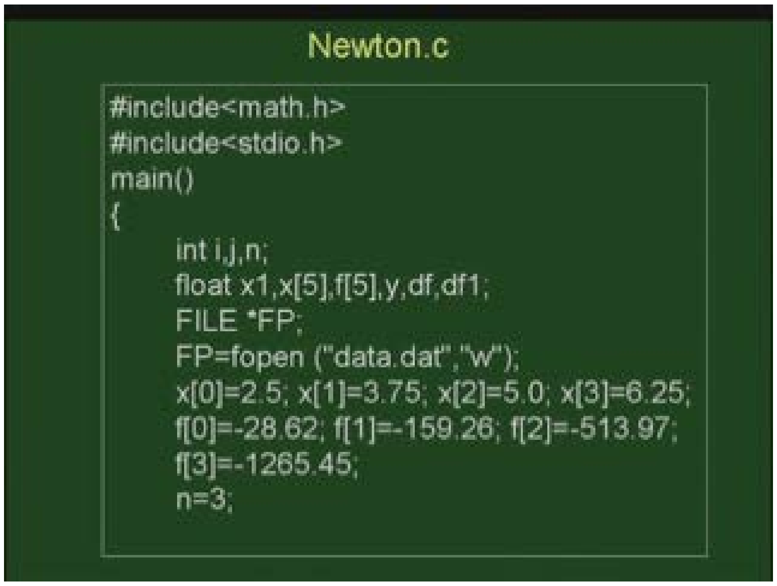
So it is i_1 plus 1, it is 2, it is f of 2 minus f of 1 divided by, now here you note, in the second divided difference we need f of, second divided difference is f of x_1 x_2 minus f of x_0 x_1 divided by x_2 minus x_0. So that has to come correct. So that is what is here. I is now 1. I is 1 and j is 1, so it is x_2 minus x_0. i is 1, so x_i minus 1, that is 0, x_2 minus x_0 is computed correctly, and then we go to i equal to 2. When I go to i equal to 2 again I will do now f_3 minus f_2 divided by, again note, it is i is 2, j is 1, so x_3 minus x_1. So the difference is 2 here, x_3 minus x_1, etcetera. And again, I am storing them, I am going to store them in second order divided differences into the same function.

Now, what will I store into i now, i greater than 1? when i is 2, i greater than 1. So i goes here, and now my f_2, i is 2 here, j is 1 and j minus 1.So this is f $_2$. So, f $_2$ will store now the second divided difference between the first two divided differences. So f $_1$ is not touched. So f_0 and my function value x_0, and now f_1 that is f of 1, the first value, have now the first divided difference. And now I am storing f_2, the second divided difference. So that is what is done. So, remember here, I just go through this again. So I had f_1 minus f $_0$ divided by x_1 minus x_0 stored here, and into this f_1 and f_2 minus f_1 by x $_2$ minus x_1 stored into f_1 f_3 minus f_2 f_3 minus f_2 minus, divided by x_3 minus x_2 was stored in f $_3$.

So then, now I am going to compute the difference between these two, that is, f of x_1 minus x_2 x_3 minus f of x_0 x_2 divided by x_3 minus x_0. Now where will I store that? I will store that into f_2, that is what I am doing. I am storing this divided difference into f_2 because f_1 is storing this one which I need, which is my coefficient for the polynomial. This is the coefficient of the polynomial. This also I need. So that is stored here. So the difference between these two is stored here, and the difference between these two is stored here, f_3, and now we go to the next step. When I go to the step, that is j, now takes the value in this program, j now takes the value 2. So j_0 is done, j_1 is done, j_2 it takes again we have the function values to evaluate. Now j is 2. So we now, i is 1,

Hence, we are going to do f_3 minus f_2. So notice that now we have only these three left these two left. So we are going to do f_3 minus f_2. That is what we are going to do now. So again, i loop now goes from 1 to n is 3, j is 2, so only 1. i loop is only 1 to 1, that is only one, because there is only one value which is divided, which is left, which is f_3 minus f_2 that will be divided by x_3 minus x_0. That is our last coefficient.

(Refer Slide Time: 50:30)



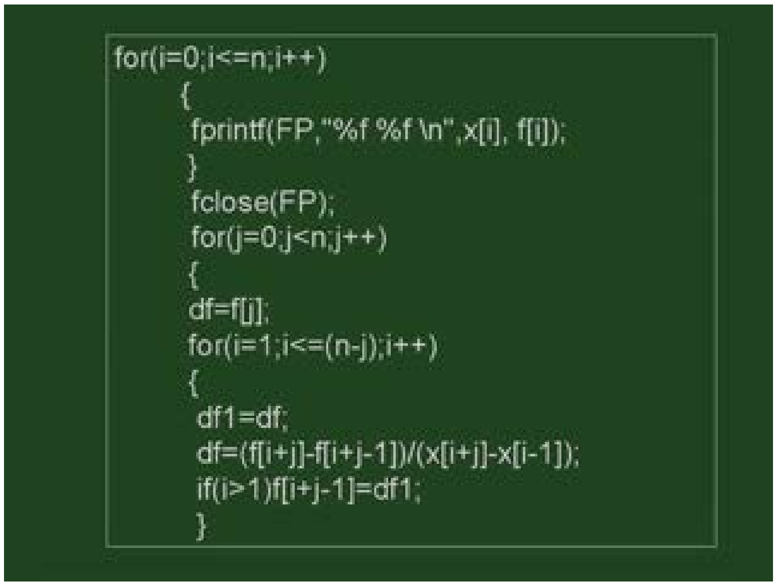
Newton.c

```
#include<math.h>
#include<stdio.h>
main()
{
        int i,j,n;
        float x1,x[5],f[5],y,df,df1;
        FILE *FP;
        FP=fopen ("data.dat","w");
        x[0]=2.5; x[1]=3.75; x[2]=5.0; x[3]=6.25;
        f[0]=-28.62; f[1]=-159.26; f[2]=-513.97;
        f[3]=-1265.45;
        n=3;
```

So we remember, we had only 0, 1 and 2. 0, 1, 2 and 3 we had, we computed, we have the function values as f_0, f_1, f_2, f_3, we computed the difference between the first differences, and we stored them as f_1 f_2 and f_3. So these three values, that is, f of x_0 x_1 f of

x_1 x_2, f of x_2 x_3 were stored into f_1 f_2 and f_3, and then we computed the difference between these two, and that was, that is, we have f x_0 x_1 and x_2 and f x_1 x_2 x_3, and that we stored in f $_2$ and f $_3$. Now we compute the difference between these two, and that will be stored in that f of x_0 x_1 x_2 x_3, and that will be stored in f 3. So that is the program.

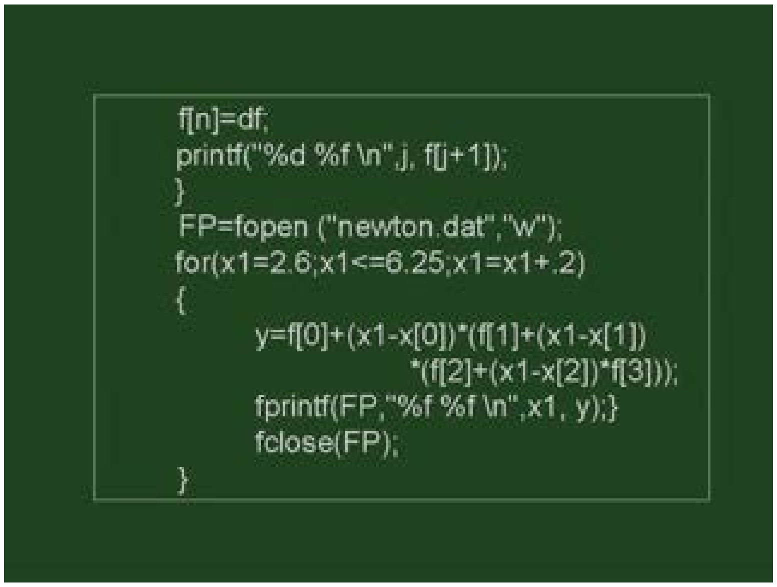
(Refer Slide Time: 50:38)



```
for(i=0;i<=n;i++)
       {
        fprintf(FP,"%f %f \n",x[i], f[i]);
        }
        fclose(FP);
        for(j=0;j<n;j++)
        {
        df=f[j];
        for(i=1;i<=(n-j);i++)
        {
         df1=df;
         df=(f[i+j]-f[i+j-1])/(x[i+j]-x[i-1]);
         if(i>1)f[i+j-1]=df1;
         }
```

The program uses the array we stored the function value to store the divided differences intermediate divided differences, which at the end will become just the coefficients of your Newton's polynomial. So that is what we have done. At the end of this j i loop, at the i and j loops, we will have all the f values storing those coefficients f_1, f_0, f_1, f_2 and f_3. So that is what I printed-off here.

(Refer Slide Time: 50:48)



```
f[n]=df;
printf("%d %f \n",j, f[j+1]);
}
FP=fopen ("newton.dat","w");
for(x1=2.6;x1<=6.25;x1=x1+.2)
{
        y=f[0]+(x1-x[0])*(f[1]+(x1-x[1])
                    *(f[2]+(x1-x[2])*f[3]));
        fprintf(FP,"%f %f \n",x1, y);}
        fclose(FP);
}
```

So I just want to print out here in this statement printf, it will print it on the screen, the j values and the corresponding coefficients of the polynomial. We will just print out that, and once we have that, we will just see that first here. We can print that out. So we compile this program and then we will print that out. Now that is the coefficients. The coefficients are 0, 1 and 2.

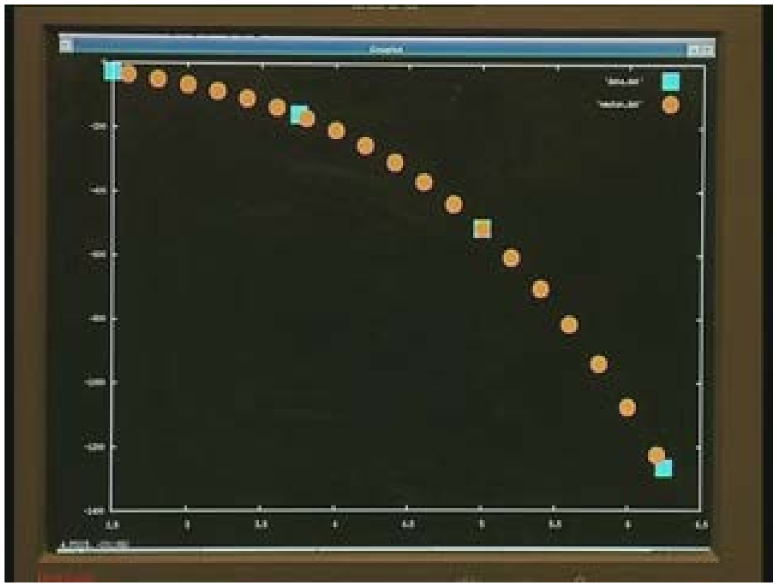
So we are just, I just use them here to show that. So, 104 minus 71 and f_4. So that is the function values $f_{0,}$ $f_{1,}$ f_2 which we, which you would use. So basically, you would use f_0, f_1 and f_2 to construct our polynomial. That is what we do here. So we will see that. Once we have that polynomial thing, so now we have to use the nested form. So you remember the nested form which we wrote down. So we are going to use that nested form to compute that, that is, we have the function value to be evaluated.

Now I am going to evaluate the function values from "2.6" to "6.25" at steps of .2. That is what this says. So the floating point variable called x_1. It starts from "2.1" and it goes up to "6.25", x 1 less than or equal to "6.25" in steps of .2. So it will go in steps of .2, and for every value, so since I have the coefficients already calculated, now I can evaluate this polynomial at any point I want, in between "6.25" and "2.5" because that is the value at which we have tabulated the points and the function, and then there is nested form here, that is the function value which I call y, is f of 0 plus x_1 minus x_0, x_1 is the value at which we want to evaluate the polynomial into f of 1 minus f of 1 plus x minus x_1 into f of 2 plus x minus x_2 into f of 3. So that is what we are going to print out. So we have this polynomial here, and you remember this is the nested form which we have written, that is f of 0 x minus x_1 open a bracket. Then f of 1 plus x minus x_1, open another bracket, that is here, and then f of 2 plus x minus x_2 and multiply that by f of 3, and close all the brackets here, and I am printing out these values here.

So I am printing out the values x_1 and the function value. So we are printing out this into a file. I have opened a file called "Newton.dat" here, using the file pointer fp, and I print out. I am putting all these function values, and the point at which we are evaluating the function into that file, and now we just plot this file to show you how that is. So what we will do is we will plot this value, this plot, this data file called Newton.dat, which has our interpolated values, and we will plot also this file which had our original given values. So we have two data files now called data.dat, which has the values given to you, and Newton.dat which has the interpolated values, and we will plot them and then see. That is what we are going to see. So, I will use since I have run the program, I have these two files created. Now I will just plot that in them. So now, here is the plot of this thing.

So that is the plot. It does not matter what the axis is. The axis is the function values. Now, note that these points, these green points here, these bluish-green points, are the 1, 2, 3 and 4. There are these 4 values, and these are the tabulated values. So 1, 2, 3 and 4 are the tabulated values which you have given in the square symbol.

(Refer Slide Time: 55:08)



The square symbol, I give the tabulated values, and this circular symbol is my interpolated values. So you can see that the interpolated values goes smoothly through this tabulated values, and in one point where the interpolated value, where we use the polynomial to evaluate the function exactly at the tabulated value, we have exactly the same result. So that is implementation of the Newton's form.

We will also see in the next class, may be, so the implementation of the Lagrange form, and how we can get the error of this polynomial which we got as an interpolating polynomial for these data points.