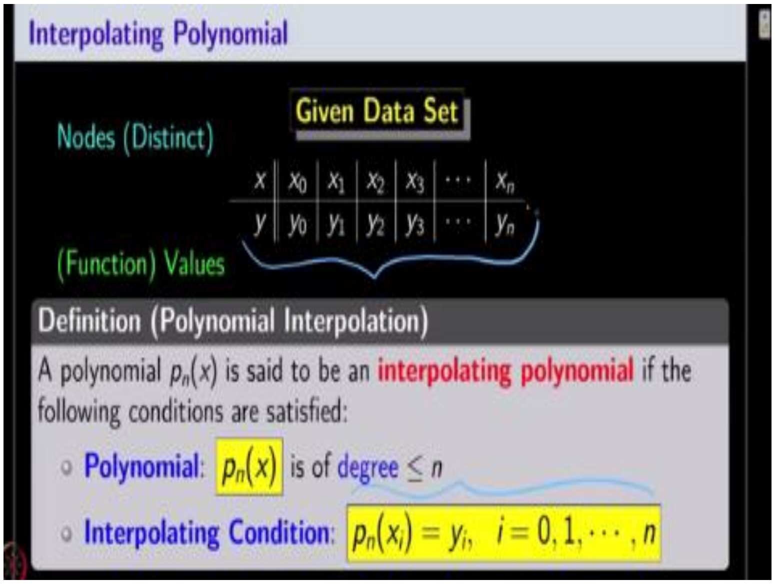
Numerical Analysis
Prof. S. Baskar
Department of Mathematics
Indian Institute of Technology, Bombay

Lecture - 39
Polynomial Interpolation: Lagrange and Newton Forms

Hi, we are learning polynomial interpolation of a given data set. In this we have learned that for a given data set we can always find a polynomial interpolation and such a polynomial interpolation will always be unique for a given data set. This is what we have learned so far. In this class we will learn two ways to construct such a polynomial interpolation for a given data. One is the Lagrange interpolation and another one is the Newton's interpolation.

(Refer Slide Time: 00:55)



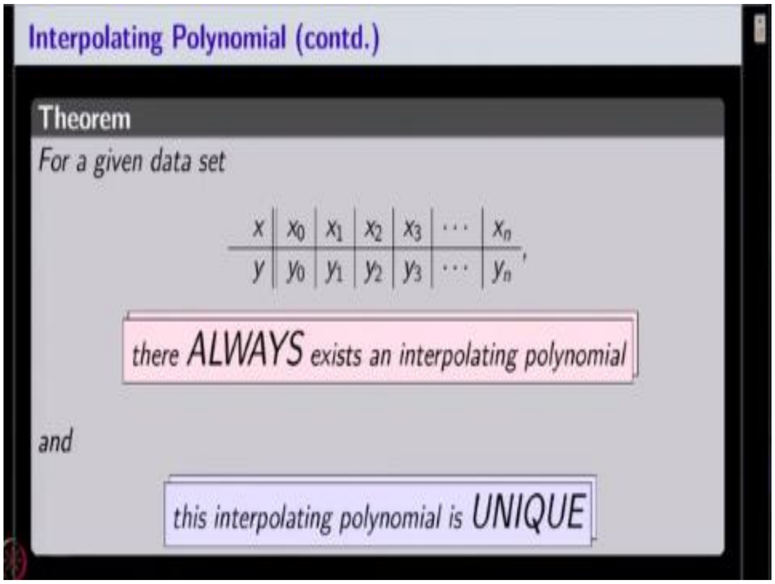
Let us quickly recall what is an interpolating polynomial for a given data. We are given a data set; it means we are given a set of nodes and the corresponding set of values at these nodes. These values may be coming from a function or these values may be generated through some source like experiments. Now the question is can we find a polynomial say $p_n(x)$ such that $p_n(x_i) = y_i$. So, that is what is called the polynomial interpolation.

More precisely a polynomial $p_n(x)$ is said to be an interpolating polynomial for the given set of data that is more important, if its degree is less than or equal to n remember from where we are catching this n? It is coming from the number of nodes that are given in our data. If you are given

$n + 1$ node starting from the index 0 and goes till n then the degree of the polynomial should be n that is what we have to remember.

So, the polynomial $p_n(x)$ should be of degree less than or equal to n and it also satisfies the interpolation conditions given by this. So, if these two conditions are satisfied by a polynomial, then that polynomial will be called as the interpolating polynomial for this set of data that is what we have seen in the last class.

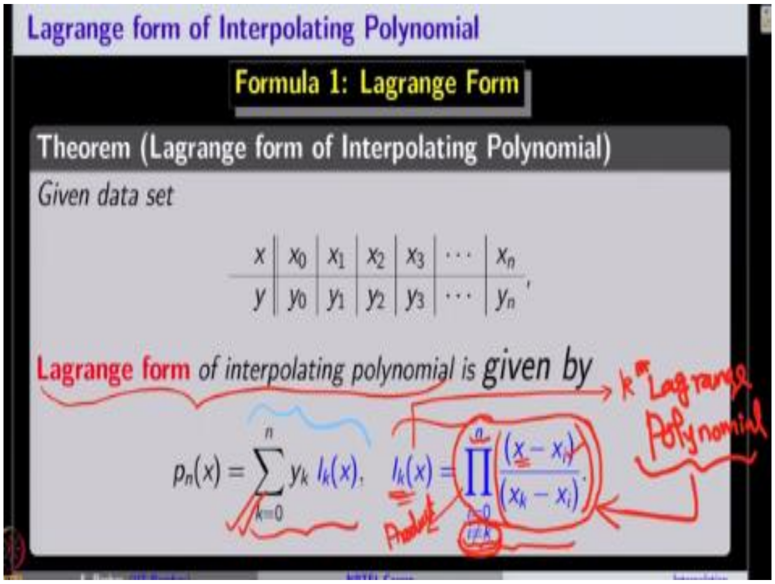
(Refer Slide Time: 02:53)



We have seen that for a given data set there always exist an interpolating polynomial and not only that, such an interpolating polynomial is also unique for a given data set that is what we have proved in the last class. But we never constructed such a polynomial in our last class. We only proved the existence by formulating a Vandermonde system and showing that the Vandermonde matrix is invertible. Today we will give two different formulas to construct these polynomials.

Remember although we give two different formulas finally those two formulas will lead to the same interpolating polynomial for a given set of data. Why? Because we have proved that such an interpolating polynomial is unique. It is only that the methods or the form in which we construct the polynomial is different but they both will lead to same polynomial.

(Refer Slide Time: 03:57)



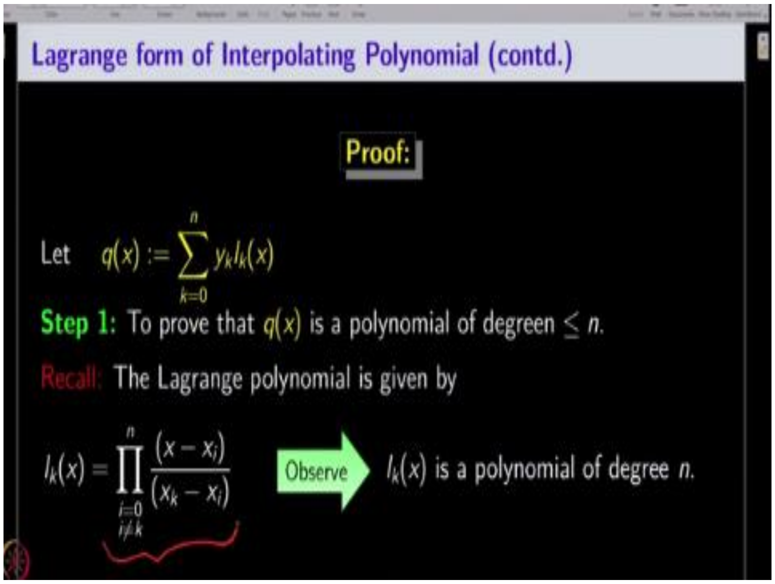
Let us start with the first method called Lagrange form of interpolating polynomial. We are given a data set like this. The Lagrange form of interpolating polynomial is given by this formula. You can observe that this is the linear combination of some polynomials. How these polynomials are given? These are given by $l_k(x)$ and its expression is given like this. This is nothing but the product.

So, product of these terms where you have I, here i runs from 0 to n but it excludes $i = k$ that k is sitting here and this is called the kth Lagrange polynomial. Note the difference between Lagrange form of interpolating polynomial that is given by this expression and the Lagrange polynomial means this polynomial. You can see that it is a polynomial of degree n, why? Because in this product we have $n + 1$ terms but we are excluding one term here which is nothing but the kth term.

And that is called the kth Lagrange polynomial. Therefore, the product runs with $n + 1$ index but in between one index is removed. So, it has n terms in the product and each term contributes to x therefore its degree is n. And this polynomial is the linear combination of nth degree polynomials there are $n + 1$ such polynomials. Therefore, this is a polynomial of degree less than or equal to n. Now we have to prove that it also satisfies the interpolating condition.

Once you do that then the uniqueness says that your polynomial interpolation is nothing but this form. Let us try to prove this.

(Refer Slide Time: 06:28)



Again, I will tell you what we have to prove. Lagrange's suggests that the interpolating polynomial will look like this where all this l_k's are the Lagrange polynomials. Remember, the space of all polynomials of degree less than or equal to *n* forms a vector space. And we are trying to get our interpolating polynomial as a member from this vector space and you can also show that the Lagrange polynomials form a basis for this space.

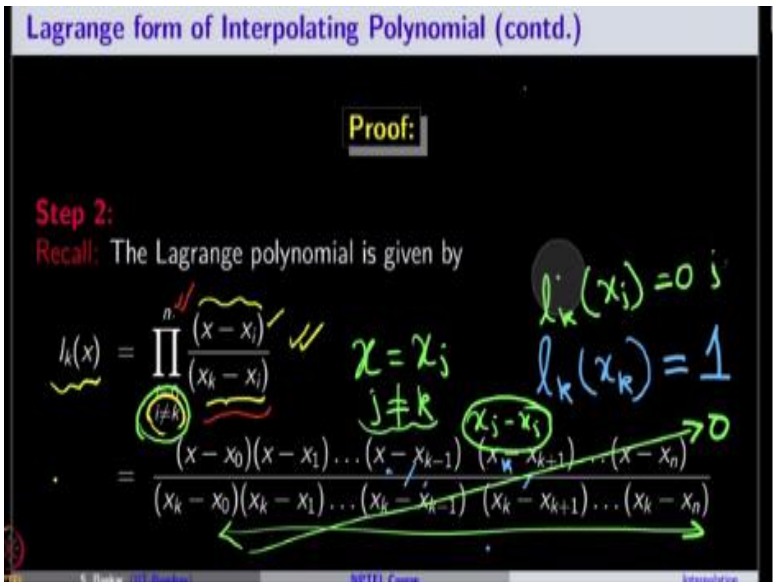
And therefore, our polynomial $p_n(x)$ in fact can be written as the linear combination of this basis elements that is the overall idea behind this. So, as a proof what we will do is, first we will show that this expression is nothing but a polynomial of degree less than or equal to *n* that is not very difficult to show. And next is that we have to show that this $q(x)$ which is now a polynomial of degree less than or equal to *n* satisfies all the interpolation conditions.

Thereby $q(x)$ will also be an interpolating polynomial to the given data set. Already we have $p_n(x)$ as our interpolating polynomial. We do not know how it looks like but that is our assumption that we always have this polynomial for a given set of data. Now we constructed another polynomial like this, thanks to Lagrange. Now the uniqueness says that these two should be equal, that is how we will prove this theorem.

As the first step, we have to prove that $q(x)$ is a polynomial of degree less than or equal to n and that I have already told you how to show that you take each Lagrange polynomial you can see that each Lagrange polynomial is actually a polynomial of degree n. So, that is not very difficult for us to observe from here, you can easily observe and then q is written as a linear combination of all these polynomials of degree n.

Therefore, q itself is a polynomial of degree less than or equal to n. So, that proves the first property of an interpolating polynomial.

(Refer Slide Time: 09:06)



Let us now try to prove the second property, that is, q satisfies all the interpolation conditions. How to do that? Again, take this Lagrange polynomial $l_k(x)$ equal to this this is the *k*th Lagrange polynomial. Let us closely look at this polynomial and try to understand how it looks like. I am just writing this expression by just putting this product explicitly instead of using this notation that is all I am doing; I am not doing anything.

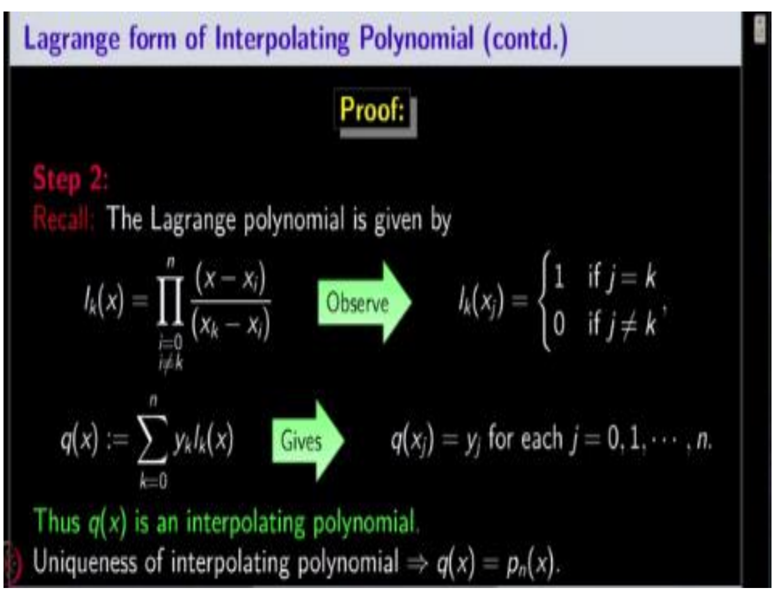
Therefore, you can see that *i* runs from 0 to n therefore when *i* = 0 you have $x - x_0$ that is coming from here divided by $x_k - x_0$ that is coming from the denominator into when *i* = 1 the numerator is $x - x_1$ and the denominator is $x_k - x_1$ like that it goes till $x_k - 1$ then you see that we are just excluding the term *i* = *k*. We do not want to include that term into our product. Therefore, there is supposed to be one more term $x - x_k$ and similarly here $x_k - x_k$ and that is actually removed in our product in this definition.

Of course, that is quite nice that we have to remove it because otherwise this term would have become 0 it would have made the entire term to be not defined at all. Therefore, that *k*th term is actually removed from this product and then it jumps from k - 1 to k + 1 and then it goes just like how it went here it goes till $x - x_n$ to $x_k - x_n$. So, this is how the *k*th Lagrange polynomial looks like. Now let us take $x = x_k$, can you see what happens?

It means you are putting $x = x_k$ here that will get cancelled with this term again here $x = x_k$ if you put this term also get cancelled. Similarly, all the terms will get cancelled and you will get this term equal to 1. It means $l_k(x_k)$ = 1, this is what we are seeing. Similarly, what happens if we take *x* = some x_j for $j \neq k$, just see what happens. If you take $x = x_j$ for $j \neq k$ then see you already excluded *k* from this term.

Therefore, this *j* will be one of these terms. This *j*th term will be sitting in this product at some stage and that will make $x_j - x_j$ in the numerator and that will make the entire thing to become 0. So, therefore you can see that $l_k(x_j)$ = 0 when $j \neq k$.

(Refer Slide Time: 13:06)



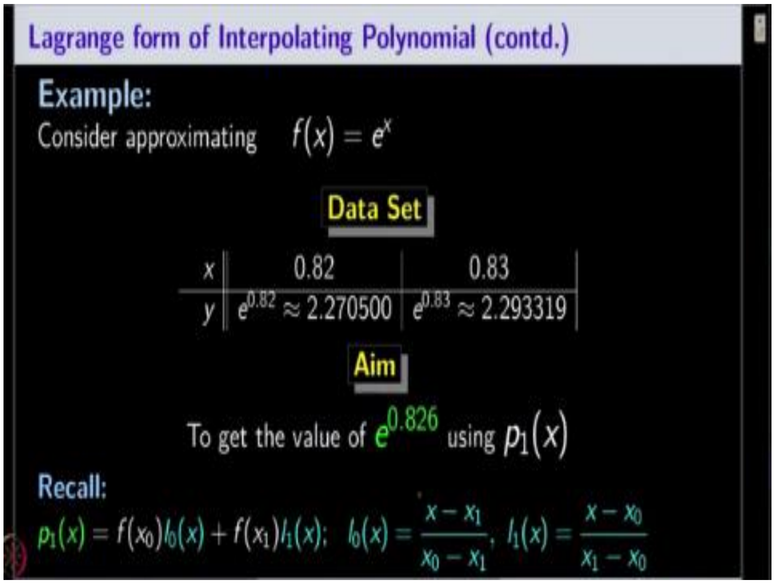
So, that is what I am writing here $l_k(x_j) = 1$ if $j = k$ and 0 if $j \neq k$. Now come back to our polynomial that Lagrange is suggested as, well this can be written as $y_0 l_0(x) + y_1 l_1(x) + y_2 l_2(x) + \cdots + y_n l_n(x)$. Now let us take *x* is equal to say x_1 what happens here? Then on the left hand side you have $q(x_1)$ and that is equal to $l_0(x_1)$. If you see when $k \neq j$ then $l_k(x_j)$ is 0.

Therefore, this term will become 0 and $l_1(x_1)$ will come, again when $k = j$, $l_k(x_j)$ is 1. Therefore, this will give us the value 1 and all other will give us the value 0 and therefore you will be left out with y_1. So, that shows that $q(x_1) = y_1$ this is just an example, I have taken. You can similarly show that $q(x_0) = y_0$ and so on. In general, you can show that $q(x_j) = y_j$ for each j = 0 to *n*. This is precisely the interpolation condition.

Therefore, you can see that q is a polynomial of degree less than or equal to *n* and also it satisfies the interpolation conditions that shows that $q(x)$ is an interpolating polynomial, we have $p_n(x)$ as the interpolating polynomial on one hand. Now Lagrange is suggested a formula and we called it as $q(x)$ and we have shown that that formula is also giving an interpolating polynomial. Now by uniqueness that formula that we got should be equal to $p_n(x)$ which is the interpolating polynomial.

Now from this theorem, what we get is a nice formula to get an interpolating polynomial for the given data set and it is of course unique. Therefore, that is the interpolating polynomial for the given data set that is what we understand.

(Refer Slide Time: 16:28)



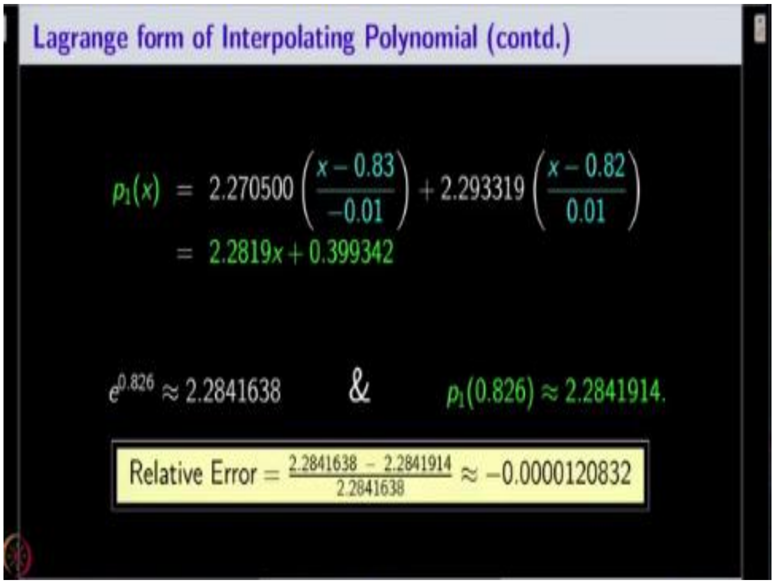
Let us take an example. Let us take this nice function. If you recall we have taken an example in the last class where interpolating polynomial was not successful in giving a good approximation. Now we are taking a nice function $f(x) = e^x$, let us see how the interpolating polynomial works. Now I can construct and show you because we know one formula to construct the interpolating polynomial.

Therefore, we will go to construct the polynomial explicitly. To keep our construction very simple I will take only two nodes. If I take two nodes, I will have to construct the polynomial that is the interpolating polynomial of degree one, that is linear polynomial and my aim is to find the value of the exponential function at the point 0.826 which is something in between these two points. So, I want to find the value of the exponential function at 0.826.

The idea is to first construct the linear interpolating polynomial and then find the value of that linear interpolating polynomial at the point 0.826 and just take that as the approximation to the corresponding exponential value that is the idea. now use the Lagrange form of the interpolating polynomial to get the expression for $p_1(x)$. For that what we have to do? We have to first find the 0th Lagrange polynomial and then first Lagrange polynomial.

And then write this linear combination where $f(x_0)$ is this and $f(x_1)$ is this value. So, let us write this. What is $l_0(x)$? $l_0(x)$ should be $x - x_1$ divided by $x_0 - x_1$. You just see the general formula of $l_k(x)$ and try to understand why in this case $l_0(x)$ is given like. Similarly, $l_1(x)$ is given like this.

(Refer Slide Time: 19:08)

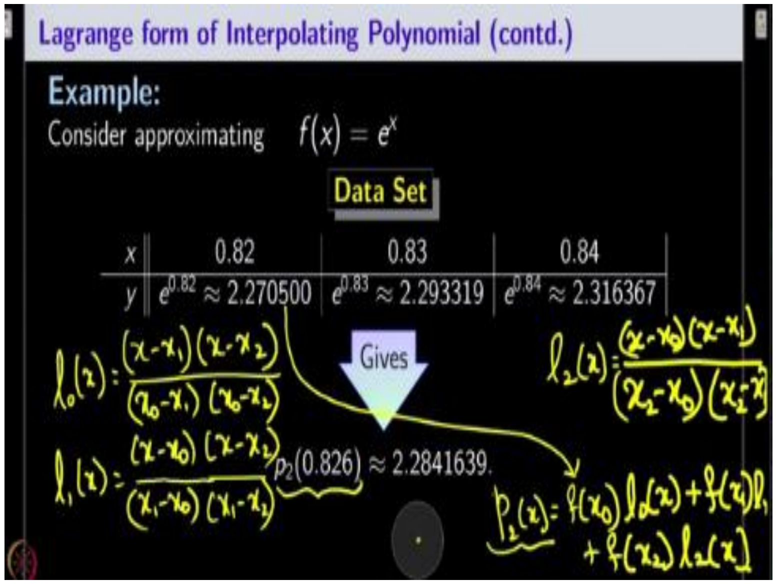


Once you get l_0 and l_1, you simply write the linear interpolating polynomial as $f(x_0)l_0(x) + f(x_1)l_1(x)$ so that is very simple. So, this is the Lagrange form of interpolating polynomial. I just simplified it, in general you do not need to simplify; you can just leave it like this. But I have just simplified it and this is precisely the linear polynomial interpolating the given data set. Now we are interested in finding $e^{0.86}$, that is what the problem that we post for ourselves.

Now we will use the interpolating polynomial to get the value and this is a nice story, a successful story. You can see the relative error in $p_1(x)$ which is the approximate value when compared to the exact value. The relative error is pretty small even for the linear polynomial. You see that is the good news for us. Of course, the function is very good and also the nodes are very close to each other.

You can see that the nodes are very close to each other that is one good thing and second thing is this function is a very nice function you can see that it is a real analytic function. Therefore, often it goes well with such approximations that is the idea.

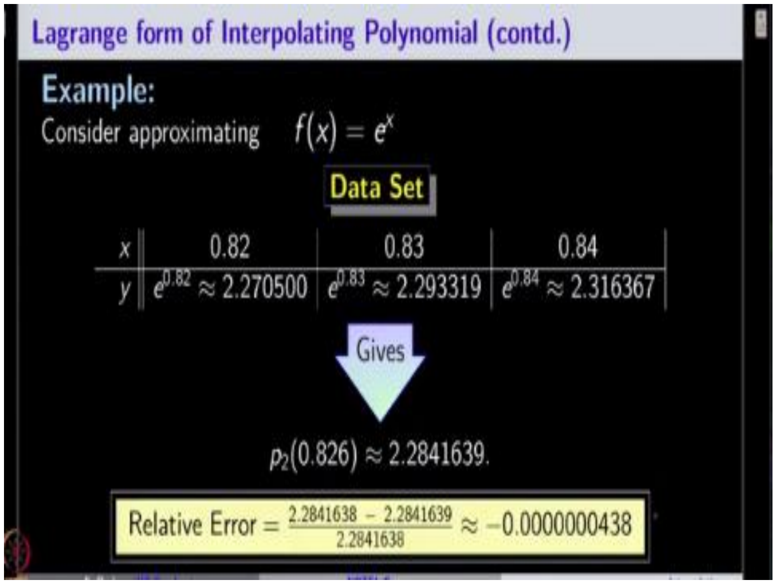
(Refer Slide Time: 20:50)



Just to have a feeling let us go and insert one more point into our data set. Previously you had this and this. Now we are having this, x_1 as this and x_2 as this. Now we have this data set and thereby we will have a quadratic polynomial as our interpolating polynomial for this data set. Remember this will be different from the previous one because the data set itself is different now. Now you can construct the quadratic polynomial. How will you do that?

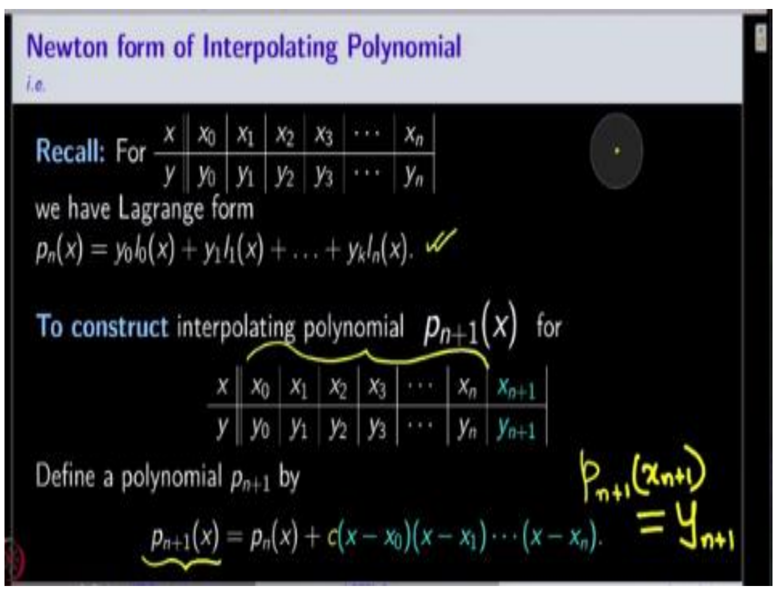
You first have to find $l_0(x)$. Now it will be $x - x_1$, see x_0 term you have to leave therefore $x - x_1$ will be there, $x - x_2$ will be there divided by $(x_0 - x_1)(x_0 - x_2)$. Similarly, $l_1(x)$ will have all the terms removing $x - x_1$ term that is $\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$. Similarly, $l_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$. Once you have this, then $p_2(x) = f(x_0)l_0(x) + f(x_1)l_1(x) + f(x_2)l_2(x)$. Once you have $p_2(x)$ you plug in x = 0.26 and that will give you this value.

(Refer Slide Time: 23:03)



You can see that the relative error is much smaller than what we have obtained from the linear interpolating polynomial. So, this is a successful story for us in applying the interpolating polynomial to approximate a function at some given nodes. So, this is the discussion about the Lagrange form of interpolating polynomial. Let us now pause on to Newton's form of interpolating polynomial.

(Refer Slide Time: 23:34)



And see how Newton proposes a formula for the interpolating polynomial. Remember whether you apply Lagrange’s formula or Newton's formula when you finally simplify that polynomial both these formulas will lead to the same polynomial that is what the uniqueness says. Now given

Lagrange polynomial why are we actually worried about another formula, that is the question. If you see Lagrange form of interpolating polynomial is given like this.

This is simply the linear combination of the basis element of the space of all polynomials of degree less than or equal to *n*. In that way Lagrange form is very nice especially, to use it in doing any theoretical study. On the other hand, Lagrange form of interpolating polynomial is not that preferred for computational purpose mainly for one reason, that is, often we come across a situation where we considered a data set.

And constructed a polynomial $p_n(x)$ and then saw that the approximation that we get out of $p_n(x)$ is not enough for our problem. So, we wanted to have a better approximation therefore we will include more data into our data set and try to get a higher degree interpolating polynomial. Such situations occur quite often in applications. In such case you see even if you put one point into your data set all the Lagrange polynomials have to be constructed right from the scratch.

You cannot make use of your effort in computing $p_n(x)$ in order to compute $p_{n+1}(x)$. All these computational efforts have to be thrashed and $p_{n+1}(x)$ have to be constructed from the scratch because all the Lagrange polynomials have to be reconstructed even if you put one node point more into your data set. That is a big disadvantage of writing the interpolating polynomial in the form of the Lagrange.

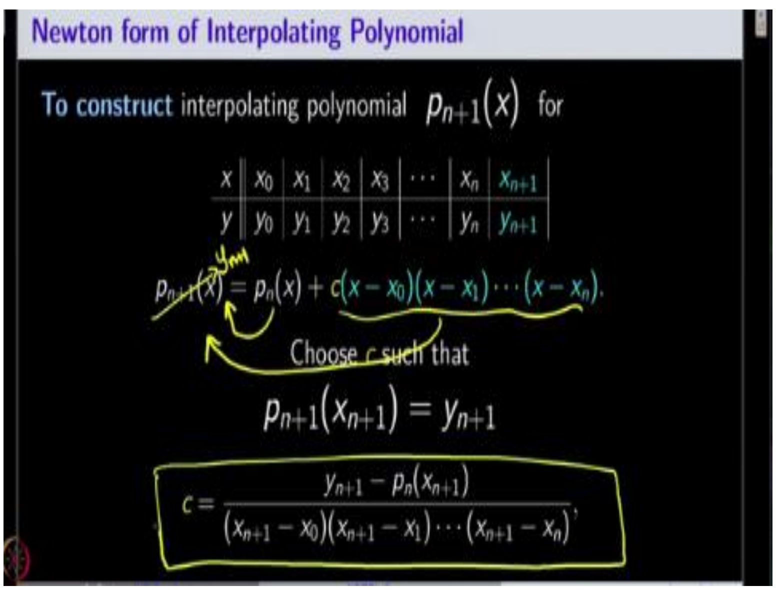
Now that is where Newton's form is quite handy because in the Newton's form what we can do is, we can write $p_{n+1}(x) = p_n(x) + \ extra\ terms.\ R$emember this is when you already computed $p_n(x)$. Therefore, this is already known to you plus you will simply put some more extra terms to get the higher degree polynomial $p_{n+1}(x)$. In that way most of your effort is now saved from your previous computation and you have to put new effort only to obtain this extra term.

Now the Newton form of interpolating polynomial is to just see what is this extra term. We propose this extra term in this form and the question is why should I put this form. You can see that if you propose this form for the extra term then if you put $x = x_0$ then this term went to zero because of this term and then you will be left out with $p_n(x_0)$ and you know that p_n is already an interpolating polynomial for this data.

And therefore, this $p_n(x_0)$ will give you y_0. And in that way the interpolation condition for $p_{n+1}(x_0)$ is readily achieved through p_n. Similarly, you can achieve the interpolating condition for x_1, x_2 up to x_n if your extra term is like this. Now the only problem is you have to make sure that $p_{n+1}(x)$ satisfies the last interpolation condition that is $p_{(n+1)}(x_{n+1}) = y_{n+1}$. This is the only remaining interpolation condition that we have to ensure.

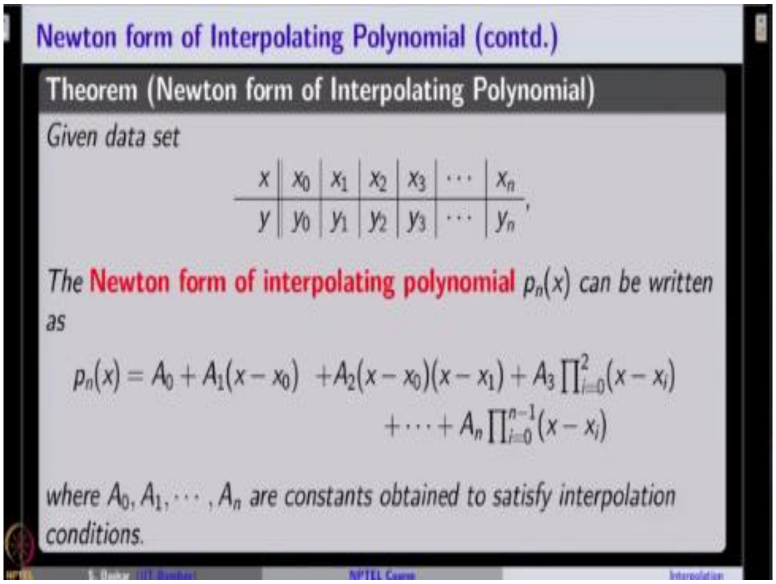
But for that we are including a free variable here. You see that is why we have inserted a free variable *c* here. Now what you can do is you put $x = x_{n+1}$ in this expression. Remember that may not give any known value to this it will come as its own value when you put $p_n(x_{n+1})$ it will not give y_{n+1} because p_n is not going to interpolate the node point x_{n+1}. It will only interpolate the points x_0 to x_n therefore this will be some value. But then you can choose your *c* such that $p_{(n+1)}(x_{n+1}) = y_{n+1}$. How will you do that?

(Refer Slide Time: 29:23)



You have to choose your *c* such that the last interpolation condition is satisfied that is not very difficult. What you will do is you will put instead of $p_{(n+1)}$ you will put y_{n+1}then you bring this to the other side and all this can then be brought to the other side to get your value for our expression for the unknown c. So, in that way you also obtained the interpolation condition for the last node.

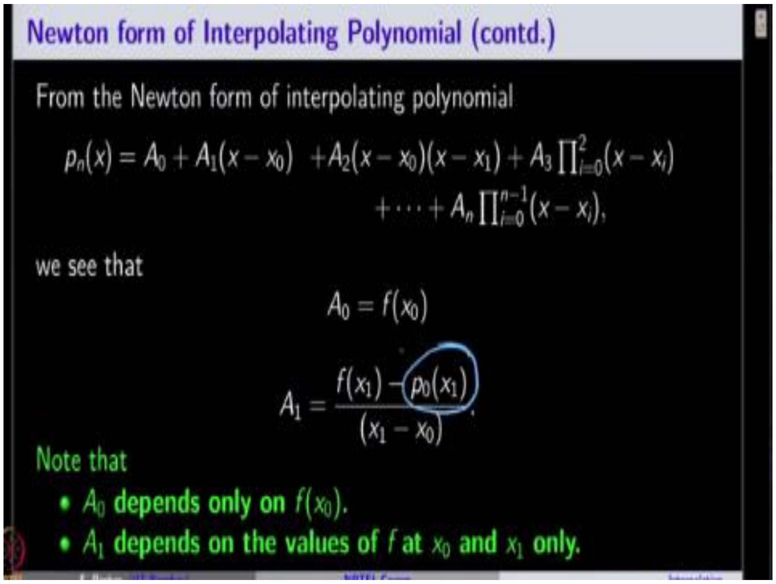
(Refer Slide Time: 29:59)



So, that gives us the Newton's form of interpolating polynomial. The advantage is that in case if you go to add one more point to an existing data for which you have already computed the interpolating polynomial then Newton's form will help us to just do an extra computation and keep the previously computed interpolating polynomial and to get the higher degree polynomial that is the advantage in the Newton's form.

Computationally, in this sense Newton's form is little efficient than Lagrange form. The Newton's form of interpolating polynomial will look like this where all these A_i's or constants that has to be obtained using the interpolation conditions. Let us see how these constants A_i are looking like.

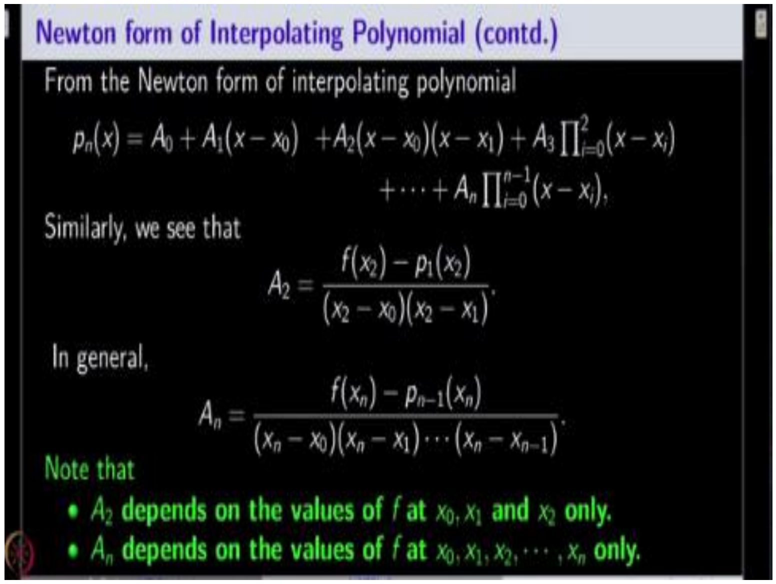
(Refer Slide Time: 30:59)



The Newton's form is proposed in this form. Now if you put $x = x_0$ you can see that all these terms will vanish because all these terms involve $x - x_0$ So, therefore all these terms will become 0 and you will be left out with A_0 and you put x_0 and that you force it to be $f(x_0)$ or y_0 and that is how you will get the value of A_0. So, you can see that A_0 depends on the function f and its value at x_0.

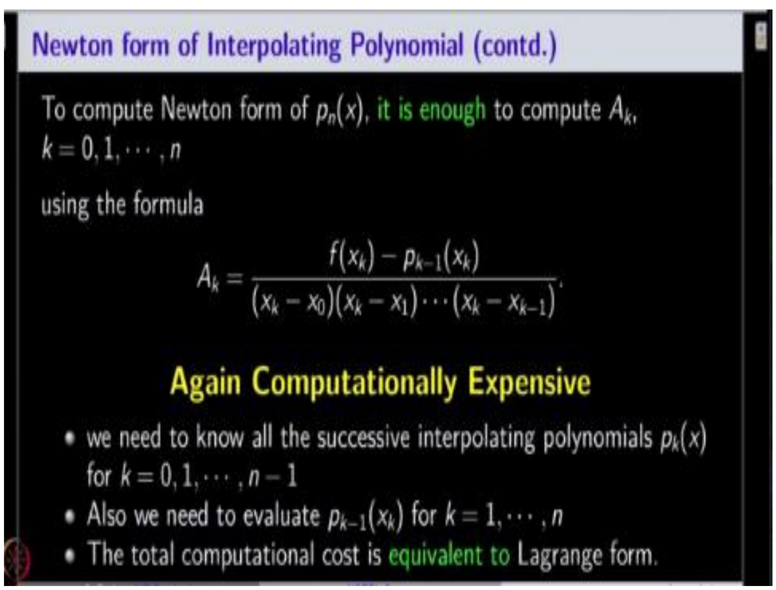
Similarly, to get A_1 you put $x = x_1$ so all these terms will go to zero because all these terms involve $x - x_1$ other than these two terms. You already know that this is $f(x_0)$. From here you can get A_1 as $f(x_1) - p_0(x_1)$, see $p_0(x_1)$ is nothing but $f(x_0)$ that is nothing but A_0 that is how it is coming. Therefore A_0depends only on $f(x_0)$ and A_1 depends on $f(x_1)$ and it also depends on the lower degree polynomial.

(Refer Slide Time: 32:30)



So, this will go on with all the other coefficients A_i's, A_2 will look like this which will depend on the lower degree polynomial and in general A_n is given like this. This is what precisely we called as c in our previous slide where A_n depends on the immediate lower degree polynomial. So, this is how the coefficients of the Newton's form of interpolating polynomial is obtained.

(Refer Slide Time: 33:07)



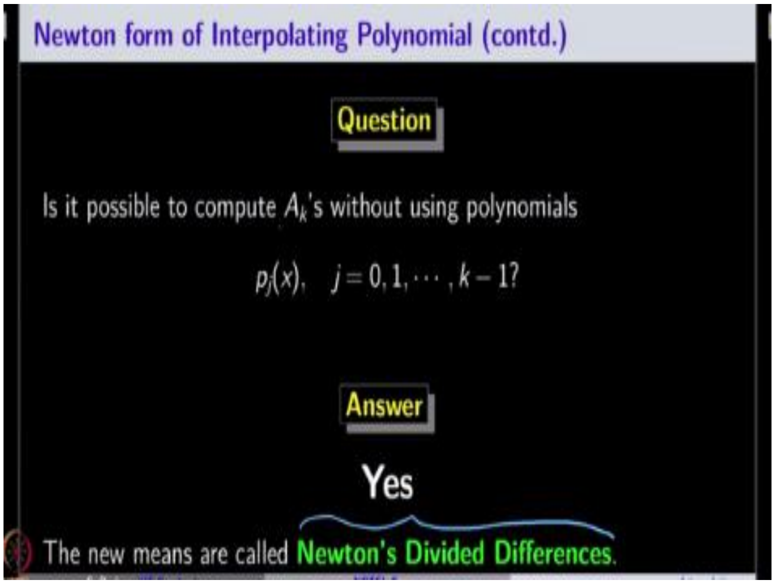
Therefore, to compute Newton's form of the interpolating polynomial p_n for a given data set it is enough for us to compute the coefficients $A_0, A_1, \cdots, A_n$ where each coefficients expression is given like this. You can observe that it depends on the immediate lower degree polynomial. This is actually again not computationally efficient because in order to compute all this coefficient you have to remember all the lower degree polynomials.

Again, that will lead to computationally very expensive algorithm that is again a bad news from the way we have posed the Newton's form of interpolating polynomial. Again, I will tell you we need to know all the successive interpolating polynomials in order to construct the coefficients of the Newton form of the interpolating polynomials. Also, we need to evaluate this polynomial, remember evaluating polynomial is not that easy.

It is a very expensive, computational cost is involved in that. And therefore, finally if you just see the total cost involved in the Newton's form in the way we have posed it is no way going to be better than the Lagrange form. Now that seems to be little depressing because we have put effort to construct another form of interpolating polynomial exactly putting what is needed for us and made our polynomial in that way.

But still we see that the coefficients involve the lower degree polynomials and that again increased our computational cost.

(Refer Slide Time: 35:06)



So, now the question is can we somehow rewrite these coefficients in such a way that we can reduce the computational cost that is the question. The answer is yes, so there is another interesting concept called Newton's divided differences. We can see that this A_k's are precisely the Newton's divided differences and we can derive a formula for all these divided differences that is all these A_k's which does not involve the evaluation of the polynomials of lower degrees.

That is the good part of the Newton's form. We will discuss the Newton's divider difference formulas in the next class, thank you for your attention.