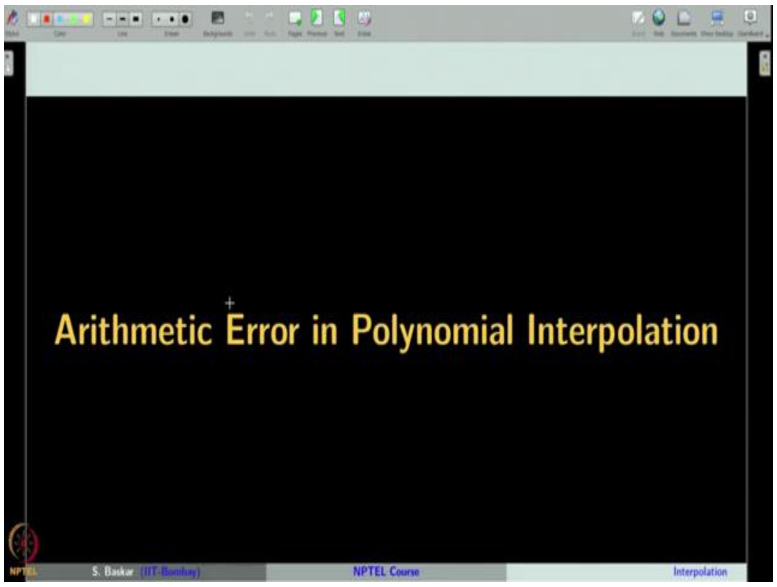
Numerical Analysis
Prof. S. Baskar
Department of Mathematics
Indian Institute of Technology – Bombay

Lecture – 42
Polynomial Interpolation: Arithmetic Error in Interpolating Polynomials

Hi, we are learning Error in Polynomial Interpolation of a Univariate Function. In this we have learned mathematical error and it is estimates. In this class we will learn, Arithmetic Errors and Total Errors and their Estimates.

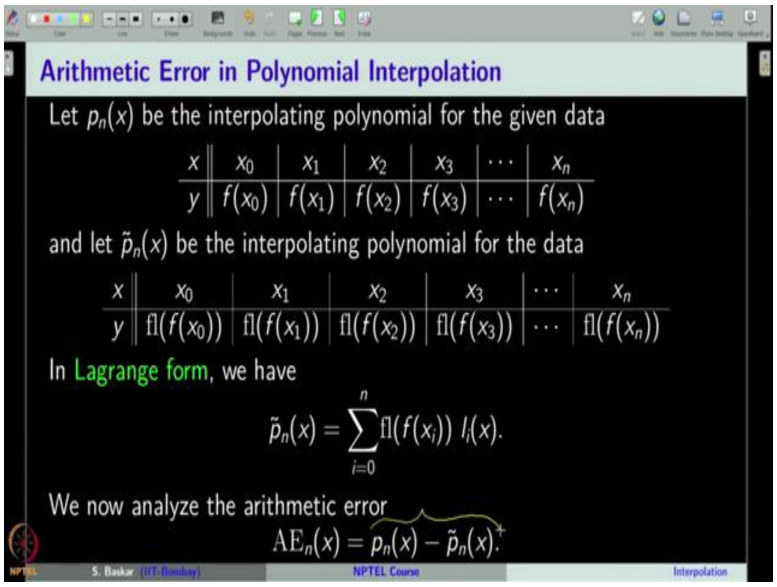
(Refer Slide Time: 00:39)



Arithmetic Error in Polynomial Interpolation

Arithmetic errors will come into the polynomial interpolation when we go to construct the interpolating polynomial on a computer.

(Refer Slide Time: 00:48)



Arithmetic Error in Polynomial Interpolation

Let $p_n(x)$ be the interpolating polynomial for the given data

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
y	$f(x_0)$	$f(x_1)$	$f(x_2)$	$f(x_3)$	$\cdots$	$f(x_n)$

and let $\tilde{p}_n(x)$ be the interpolating polynomial for the data

x	x_0	x_1	x_2	x_3	$\cdots$	x_n
y	$\mathrm{fl}(f(x_0))$	$\mathrm{fl}(f(x_1))$	$\mathrm{fl}(f(x_2))$	$\mathrm{fl}(f(x_3))$	$\cdots$	$\mathrm{fl}(f(x_n))$

In Lagrange form, we have

$$\tilde{p}_n(x) = \sum_{i=0}^{n} \mathrm{fl}(f(x_i))\, l_i(x).$$

We now analyze the arithmetic error

$$\mathrm{AE}_n(x) = \overbrace{p_n(x) - \tilde{p}_n(x)}.$$

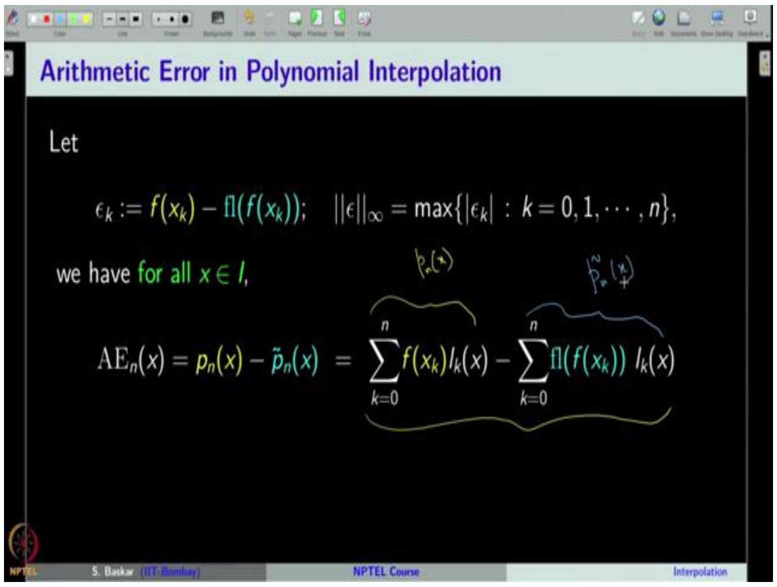
S. Baskar (IIT-Bombay) NPTEL Course Interpolation

That is, when we are given a data set. We are supposed to construct the polynomial $p_n(x)$ as the interpolating polynomial of the data set. However, when we go to feed this data on a computer, the computer makes the floating point approximation. And due to that the data set that is taken by the computer to construct the interpolating polynomial will be slightly different from what we wanted it to consider?

Say, for instance, if you are generating the data set from a function by giving only the nodes, even if we choose our nodes in such a way that there is very less arithmetic error. Even then when the computer evaluates the value of the function at a node point it tends to make arithmetic error. That we generally denote by $fl(f(x))$. Therefore, the interpolating polynomial that the computer gives for the data that it considered will be different from $p_n(x)$.

And we are denoting it by $\tilde{p}_n(x)$. In the Lagrange form $\tilde{p}_n(x)$ is given by $\sum_{i=0}^{n} fl(f(x_i))$, that is the function value evaluated with floating point arithmetic into Lagrange polynomial. And the arithmetic error is defined as the difference between the polynomial that we want minus the polynomial that is obtained on the computer. Now, our interest is to find an estimate for the arithmetic error.

(Refer Slide Time: 02:51)



Arithmetic Error in Polynomial Interpolation

Let

$$\epsilon_k := f(x_k) - \mathrm{fl}(f(x_k)); \quad \|\epsilon\|_\infty = \max\{|\epsilon_k| : k = 0, 1, \cdots, n\},$$

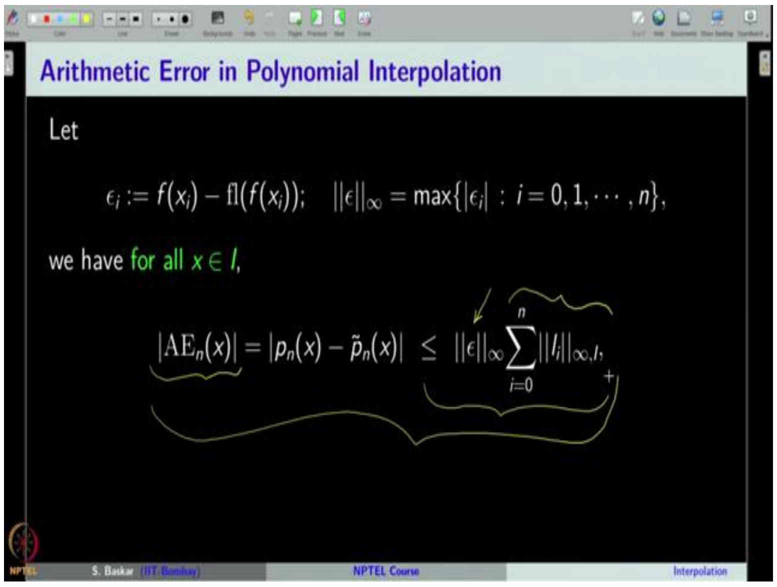
we have for all $x \in I$,

$$\mathrm{AE}_n(x) = p_n(x) - \tilde{p}_n(x) = \underbrace{\overbrace{\sum_{k=0}^{n} f(x_k) l_k(x)}^{p_n(x)} - \overbrace{\sum_{k=0}^{n} \mathrm{fl}(f(x_k))\, l_k(x)}^{\tilde{p}_n(x)}}$$

For that, let us first give a notation for the error that is committed by the computer due to floating point arithmetic. Let us call this error as ϵ_k where ϵ_k is the error involved in $fl(f(x_k))$ when compared to $f(x_k)$ where x_k is a node in our data set. Now, in that way, we have a vector of errors corresponding to each node and let us denote it by the vector epsilon.

Which is nothing but ϵ_0 which is the error committed at the node x_0, ϵ_1 and so on up to ϵ_n. And now we will define the norm of this vector epsilon by taking the maximum of the absolute values of all the errors epsilon case. And that is the infinite norm. Now, for all *x* the arithmetic error is given by $p_n(x) - \tilde{p}_n(x)$ which can be written in the Lagrange form like this where this is the Lagrange form of $p_n(x)$ and this is the Lagrange form of $\tilde{p}_n(x)$.

(Refer Slide Time: 04:29)



Arithmetic Error in Polynomial Interpolation

Let

$$\epsilon_i := f(x_i) - \mathrm{fl}(f(x_i)); \quad \|\epsilon\|_\infty = \max\{|\epsilon_i| : i = 0, 1, \cdots, n\},$$

we have for all $x \in I$,

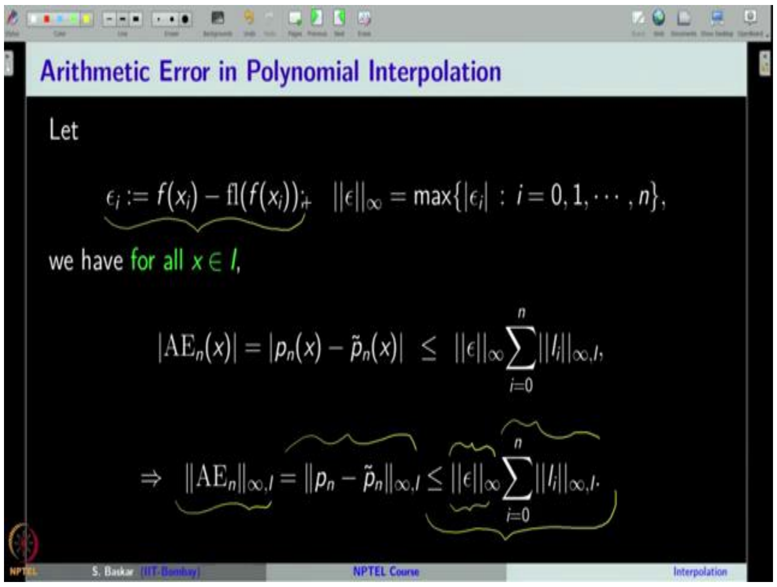
$$\underbrace{\underbrace{|\mathrm{AE}_n(x)|} = |p_n(x) - \tilde{p}_n(x)| \leq \|\epsilon\|_\infty \underbrace{\overbrace{\sum_{i=0}^{n}}\|l_i\|_{\infty, I},}_{+}}$$

This can be rewritten like this and now we will put this notation in place of this. That is instead of this we will put ϵ_k and to get the estimate for the arithmetic error, now we will take the modulus on both sides and take the modulus on the right hand side into the summation. Thereby we will have a less than or equal to sign and we have $\sum_{k=0}^{n}|\epsilon_k||l_k(x)|$.

Now, what we will do as a next step? We can replace this by its infinite now and thereby we will again have less than or equal to. And finally, we will also take the infinite norm over this function. And thereby we will have mod arithmetic error evaluated at *x* is less than or equal to the maximum error that is committed in the function value into the sum of the maximum norm of the Lagrange polynomials.

Now, you can see that the right hand side is independent of *x* and therefore, this inequality holds for all *x* in the interval (a, b). And that implies that it also holds for that x at which the arithmetic error attains it is maximum.

(Refer Slide Time: 06:21)



Arithmetic Error in Polynomial Interpolation

Let

$$\underbrace{\epsilon_i := f(x_i) - \mathrm{fl}(f(x_i))}; \quad \|\epsilon\|_\infty = \max\{|\epsilon_i| : i = 0, 1, \cdots, n\},$$

we have for all $x \in I$,

$$|\mathrm{AE}_n(x)| = |p_n(x) - \tilde{p}_n(x)| \leq \|\epsilon\|_\infty \sum_{i=0}^{n} \|l_i\|_{\infty,I},$$

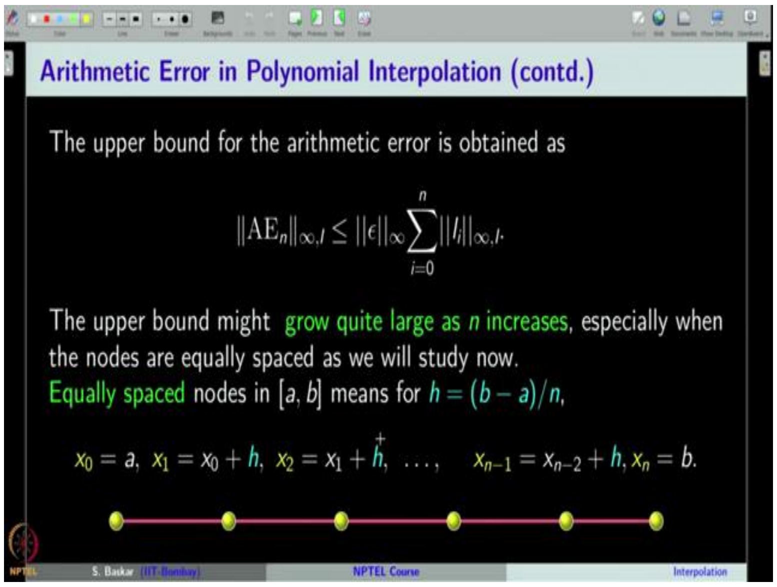
$$\Rightarrow \underbrace{\|\mathrm{AE}_n\|_{\infty,I}} = \overbrace{\|p_n - \tilde{p}_n\|_{\infty,I}} \leq \underbrace{\overbrace{\underbrace{\|\epsilon\|_\infty}} \overbrace{\sum_{i=0}^{n} \|l_i\|_{\infty,I}}}.$$

S. Baskar (IIT-Bombay) NPTEL Course Interpolation

Therefore, this inequality can also be written as maximum norm of the arithmetic error which is by definition this. And now, we have shown that that quantity is less than or equal to this number. You can see that once you have an idea of how this error looks like. For instance, if you know that your computer will do some nth digit rounding approximation then you know to get an estimate for this.

And also, this can be computed once the nodes are given to us. Because the Lagrange polynomials do not depend on the function value, they only depend on the nodes that we choose. Therefore, the upper bound can be evaluated explicitly once we know how we will be committing error with the function values. In that way, we got an estimate for the arithmetic error.

(Refer Slide Time: 07:28)



Arithmetic Error in Polynomial Interpolation (contd.)

The upper bound for the arithmetic error is obtained as

$$\|\mathrm{AE}_n\|_{\infty,I} \leq \|\epsilon\|_\infty \sum_{i=0}^{n} \|l_i\|_{\infty,I}.$$

The upper bound might grow quite large as n increases, especially when the nodes are equally spaced as we will study now.

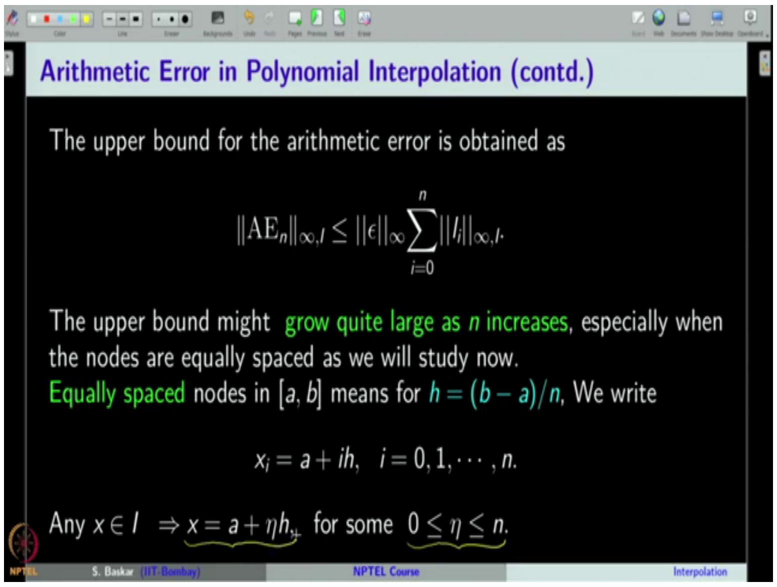
Equally spaced nodes in $[a,b]$ means for $h = (b-a)/n$,

$$x_0 = a,\ x_1 = x_0 + h,\ x_2 = x_1 + h,\ \ldots,\quad x_{n-1} = x_{n-2} + h, x_n = b.$$

And that estimate is given like this. Now we will see that the upper bound, that is this quantity grows more rapidly as we increase n. Especially, this happens when we use equally spaced nodes well, what is mean by equally spaced nodes? Let us see we are given an interval, (a, b) by equally spaced nodes. We mean a partition of the interval, (a, b) in which the length of all the sub intervals are equal.

Say, for instance, we have the length of the sub intervals, as $h = \frac{b-a}{n}$. If we have n partitions in our interval, if so then we will have x_0 as a, x_1 as $x_0 + h$ that is $a + h$ and $x_2 = x_1 + h$ which can also be written as $a + 2h$. And similarly, you can go on generating the nodes by adding h to the previous node. That is what we mean by equally spaced nodes.

(Refer Slide Time: 08:58)



Arithmetic Error in Polynomial Interpolation (contd.)

The upper bound for the arithmetic error is obtained as

$$\|\mathrm{AE}_n\|_{\infty,I} \le \|\epsilon\|_\infty \sum_{i=0}^{n} \|l_i\|_{\infty,I}.$$

The upper bound might grow quite large as n increases, especially when the nodes are equally spaced as we will study now.

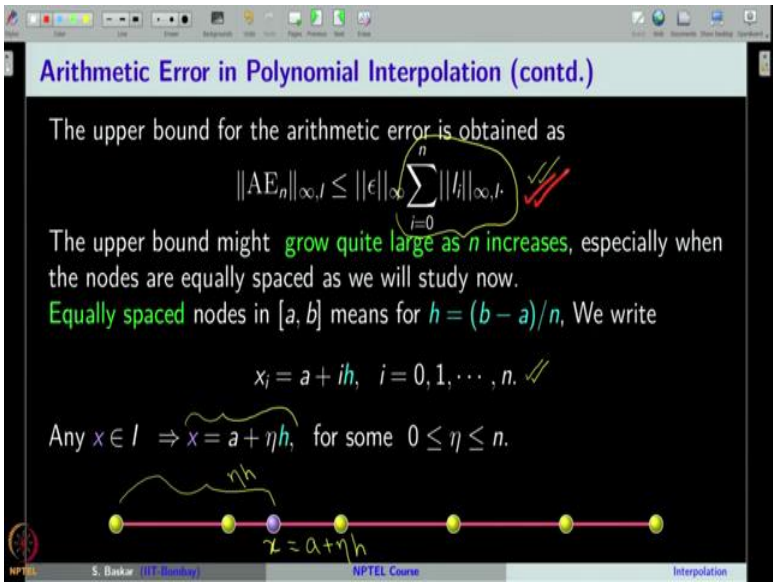
Equally spaced nodes in $[a,b]$ means for $h=(b-a)/n$, We write

$$x_i = a + ih, \quad i = 0,1,\cdots,n.$$

Any $x \in I \Rightarrow \underbrace{x = a+\eta h}$, for some $\underbrace{0 \le \eta \le n}$.

And now, these equally spaced nodes can also be written like this. That is x_i is obtained as $a + i\,h$ where i is the index in the nodes. Now, for any x in the interval (a,b) we can find an $0 \le \eta \le n$, not necessarily an integer such that any point x can be written as $a + \eta h$.

(Refer Slide Time: 09:32)



Arithmetic Error in Polynomial Interpolation (contd.)

The upper bound for the arithmetic error is obtained as

$$\|\mathrm{AE}_n\|_{\infty,I} \le \|\epsilon\|_\infty \sum_{i=0}^{n} \|l_i\|_{\infty,I}.$$

The upper bound might grow quite large as n increases, especially when the nodes are equally spaced as we will study now.

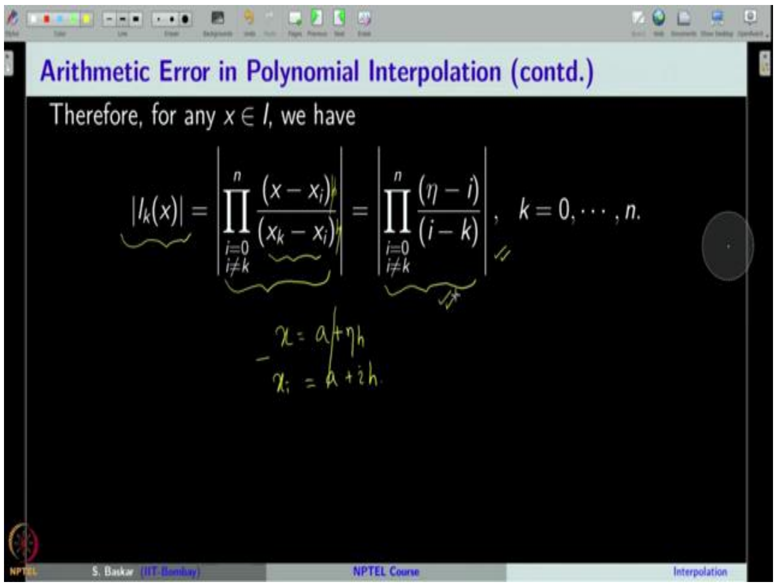
Equally spaced nodes in $[a,b]$ means for $h=(b-a)/n$, We write

$$x_i = a + ih, \quad i = 0,1,\cdots,n.$$

Any $x \in I \Rightarrow \overbrace{x = a+\eta h}$, for some $0 \le \eta \le n$.

How it looks like? Suppose this is your x then this is your ηh therefore, x can be written as $a + \eta h$. Now, we will keep in mind how x_i and any point x in the interval (a,b) can be written if we are considering the equally spaced nodes. Now, let us try to get an idea of how this sum of the Lagrange polynomials behave when we work with equally spaced nodes in order to get an idea of how rapidly the upper bound increases.

(Refer Slide Time: 10:23)



Arithmetic Error in Polynomial Interpolation (contd.)

Therefore, for any $x \in I$, we have

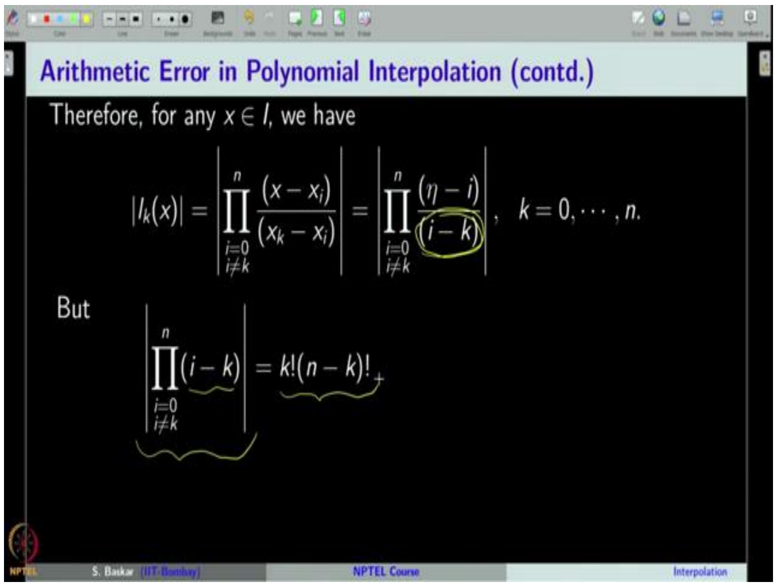
$$\underbrace{|l_k(x)|} = \underbrace{\left|\prod_{\substack{i=0\\i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)}\right|} = \underbrace{\left|\prod_{\substack{i=0\\i\neq k}}^{n} \frac{(\eta-i)}{(i-k)}\right|}, \quad k=0,\cdots,n.$$

$$\begin{aligned} x &= a+\eta h \\ - \; x_i &= a+ih \end{aligned}$$

For that we will first take the Lagrange polynomial. If you recall, the Lagrange polynomial is defined like this. Now, we know $x = a + \eta h$ and $x_i = a + ih$. Therefore, if you take the difference between them, this gets cancelled. And similarly, you will also have $k - i\, h$, so h here and h here gets cancelled and you will have this expression. This is more specific for equally space nodes.

And that is how this h is getting cancelled because it is equally spaced nodes. Therefore, the Lagrange polynomial in the equally spaced nodes can be written by this formula.

(Refer Slide Time: 11:19)



Arithmetic Error in Polynomial Interpolation (contd.)

Therefore, for any $x \in I$, we have

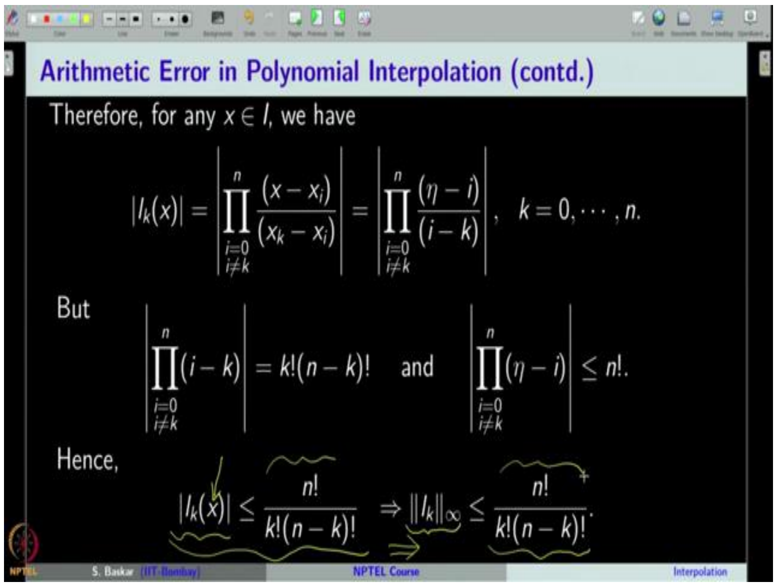
$$|l_k(x)| = \left|\prod_{\substack{i=0\\i\neq k}}^{n} \frac{(x-x_i)}{(x_k-x_i)}\right| = \left|\prod_{\substack{i=0\\i\neq k}}^{n} \frac{(\eta-i)}{(i-k)}\right|, \quad k=0,\cdots,n.$$

But

$$\underbrace{\left|\prod_{\substack{i=0\\i\neq k}}^{n} (i-k)\right|} = \underbrace{k!(n-k)!}$$

Now, let us take the denominator term. You can see that modulus of the product of this can be written as $k!\,(n-k)!$. It is a very elementary formula which we all know.

(Refer Slide Time: 11:36)



Arithmetic Error in Polynomial Interpolation (contd.)

Therefore, for any $x \in I$, we have

$$|l_k(x)| = \left| \prod_{\substack{i=0\\ i\neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} \right| = \left| \prod_{\substack{i=0\\ i\neq k}}^{n} \frac{(\eta - i)}{(i - k)} \right|, \quad k = 0, \cdots, n.$$

But

$$\left| \prod_{\substack{i=0\\ i\neq k}}^{n} (i - k) \right| = k!(n-k)! \quad \text{and} \quad \left| \prod_{\substack{i=0\\ i\neq k}}^{n} (\eta - i) \right| \leq n!.$$

Hence,

$$|l_k(x)| \leq \frac{n!}{k!(n-k)!} \Rightarrow \|l_k\|_\infty \leq \frac{n!}{k!(n-k)!}.$$

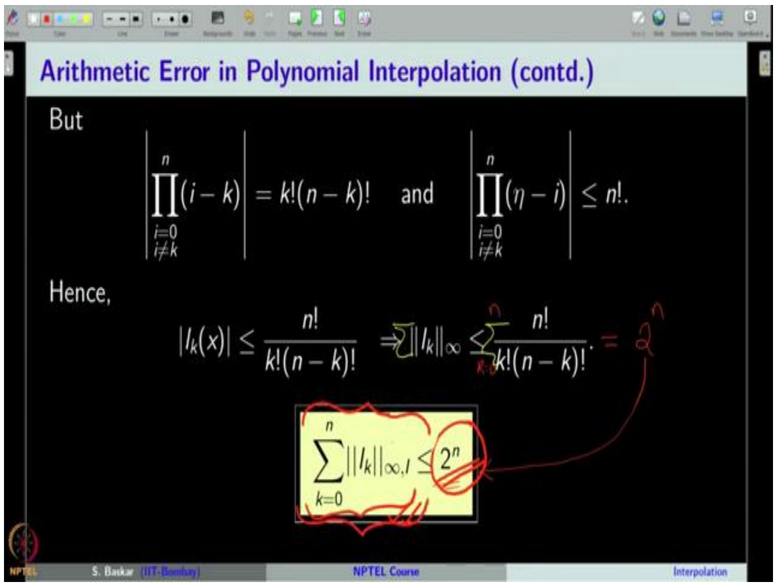
S. Baskar (IIT-Bombay) NPTEL Course Interpolation

And also you can see that the numerator term that is the product of $\eta - i$ can be made less than or equal to n factorial. Now, taking these two estimates in mind, we can write modulus of the Lagrange polynomial as less than or equal to. Because the numerator is less than or equal to that is why it is coming less than or equal to $n!$ that is coming from the numerator divided by this one.

And that is equal therefore, we can still retain less than or equal to here and that is how we get this upper bound for the Lagrange polynomial. And that when you take the infinite norm over the Lagrange polynomial that will be less than or equal to the same quantity. Why? Because the right hand side is independent of x and therefore, this holds for all x means it also holds for that x at which the infinite norm for l_k attains.

I am again and again telling this idea because this is very important and it comes often in our course. That is why I am every time repeating that. Therefore, this indeed implies that $||l_k||_\infty$ is also less than or equal to this upper bound.

(Refer Slide Time: 13:06)



Arithmetic Error in Polynomial Interpolation (contd.)

But

$$\left|\prod_{\substack{i=0\\i\neq k}}^{n}(i-k)\right| = k!(n-k)! \quad \text{and} \quad \left|\prod_{\substack{i=0\\i\neq k}}^{n}(\eta-i)\right| \le n!.$$

Hence,

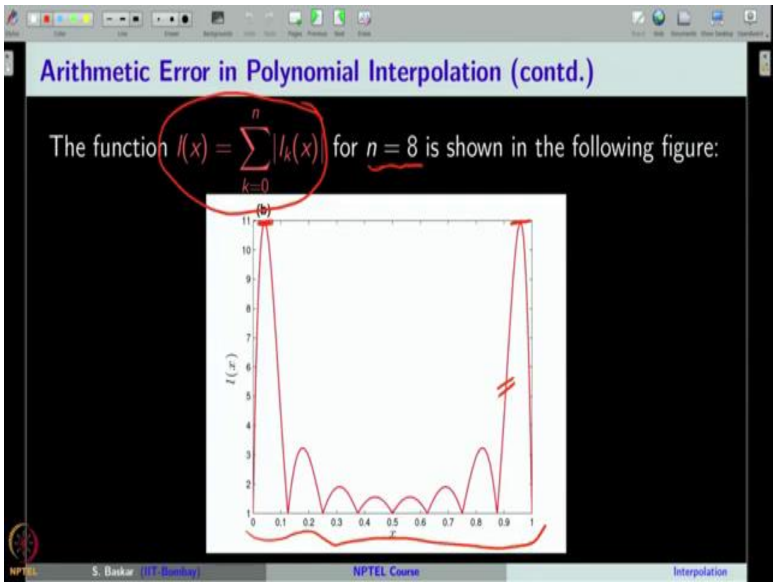
$$|l_k(x)| \le \frac{n!}{k!(n-k)!} \Rightarrow \sum \|l_k\|_\infty \le \sum_{k=0}^{n} \frac{n!}{k!(n-k)!} = 2^n$$

$$\boxed{\sum_{k=0}^{n} \|l_k\|_{\infty,I} \le 2^n}$$

Now, what we will do is, we will take summation on both sides over k = 0 to n then the summation k = 0 to n is equal to 2^n. So, we will use that to get the upper bound for this sum as 2^n. That shows that the bound of this quantity is growing exponentially. Just remember that this quantity which we have shown to grow exponentially, is sitting on the upper bound of the arithmetic error.

That is why we made this statement that the upper bound grows quite rapidly because it is growing exponentially. That is what we have seen here. Well, only the upper bound is growing very rapidly does it mean that this quantity will also grow rapidly, well that is not always true. It only gives us the chance that this may also grow rapidly because by growing this quantity makes a room for this guy also to grow. But we will also see numerically that this quantity also grows rapidly.

(Refer Slide Time: 14:33)



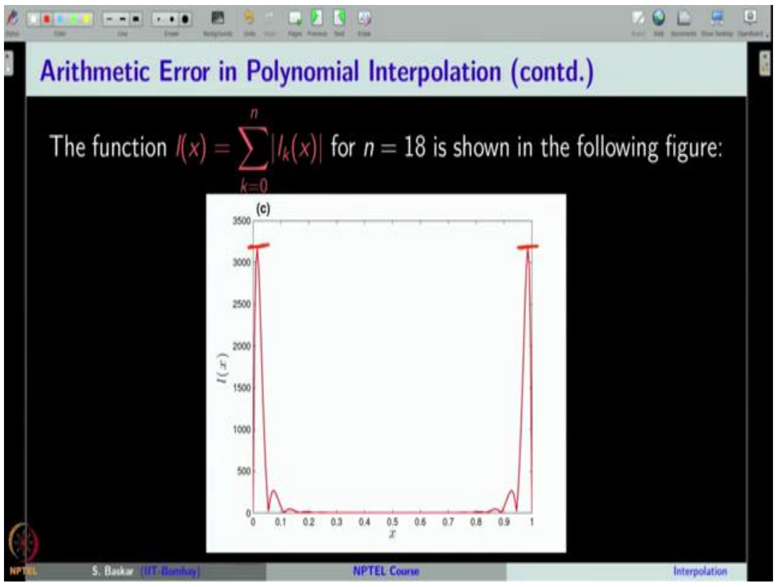
Arithmetic Error in Polynomial Interpolation (contd.)

The function $l(x) = \sum_{k=0}^{n} |l_k(x)|$ for $n = 8$ is shown in the following figure:

For that let us take the function $l(x)$ and we will plot this function for each n. Let us start with n = 2 the graph of the function $l(x)$ is shown in this figure where the red line indicates the graph of the function $l(x)$ given by this. Remember what we have is the infinite norm of this. Therefore, we are just trying to understand how the graph of this function looks like. For n = 2 this is the graph just observe the maximum of this one.

For n = 8 now, I am showing the graph when n = 8 in this expression, by choosing equally spaced nodes, you can get the graph of the function $l(x)$ as shown in this red solid line. You can see that the maximum of this function is around 11, whereas when it is n = 2, it was 1.25 now it has become 11.

(Refer Slide Time: 15:58)

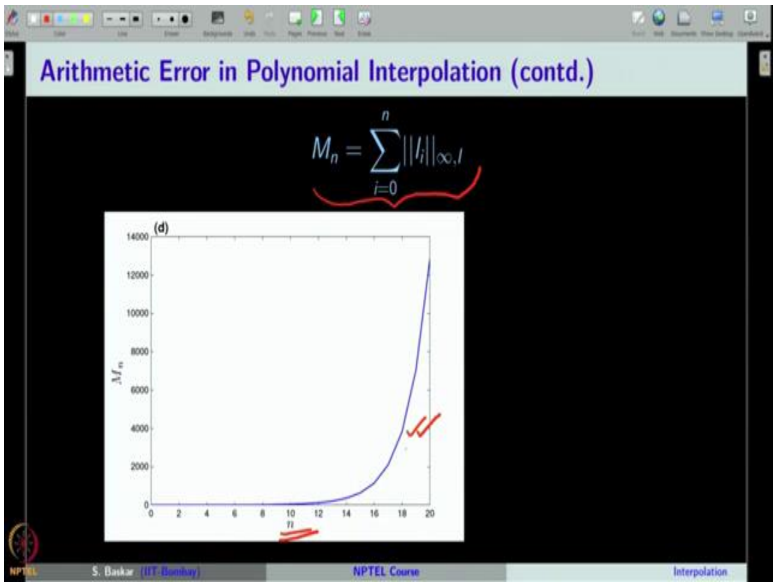


Arithmetic Error in Polynomial Interpolation (contd.)

The function $l(x) = \sum_{k=0}^{n} |l_k(x)|$ for $n = 18$ is shown in the following figure:

Now, let us further increase the value of *n* from 8 to 18. You can see the maximum value of this function $l(x)$ is around 3000, so, it is growing very rapidly.

(Refer Slide Time: 16:11)

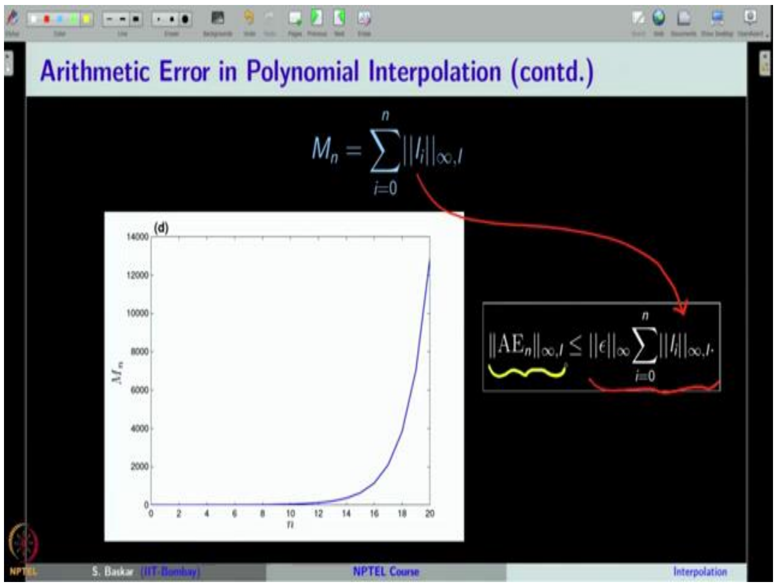


Arithmetic Error in Polynomial Interpolation (contd.)

$$M_n = \underbrace{\sum_{i=0}^{n} \|l_i\|_{\infty,I}}$$

In fact, we can see this quantity which is precisely sitting on the right hand side that is on the upper bound of our arithmetic error. It is graph as a function of *n* is shown in the blue solid line. You can see that the graph is growing exponentially. So that is actually a bad news that you have the possibility that the arithmetic error can grow exponentially. That is what we are seeing from this upper bound.

(Refer Slide Time: 16:45)



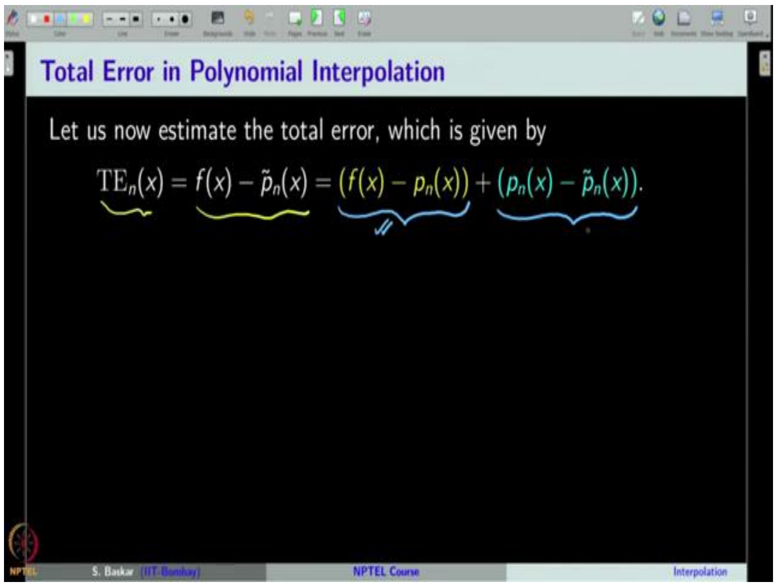
Arithmetic Error in Polynomial Interpolation (contd.)

$$M_n = \sum_{i=0}^{n} \|l_i\|_{\infty,I}$$

$$\boxed{\|\mathrm{AE}_n\|_{\infty,I} \leq \|\epsilon\|_\infty \sum_{i=0}^{n} \|l_i\|_{\infty,I}.}$$

You see this is what is sitting here and that is growing exponentially. Again that gives us a possibility that arithmetic error may also grow drastically as you go on increasing *n*.

(Refer Slide Time: 17:03)



Total Error in Polynomial Interpolation

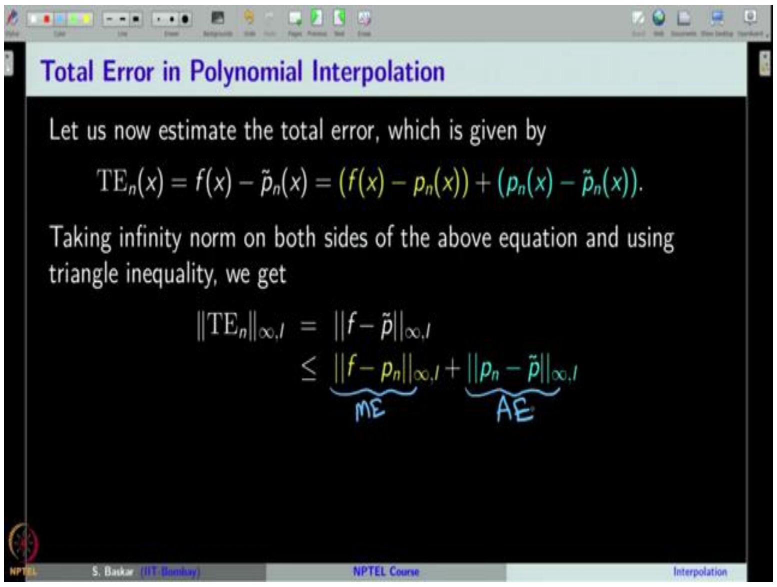
Let us now estimate the total error, which is given by

$$\underbrace{\mathrm{TE}_n(x)} = \underbrace{f(x) - \tilde{p}_n(x)} = \underbrace{(f(x) - p_n(x))}_{\checkmark\checkmark} + \underbrace{(p_n(x) - \tilde{p}_n(x))}.$$

Well, with this note, we will also see how the total error behaves recall that the total error is defined like this. And that can be written as the sum of the mathematical error and the arithmetic error. Now, to find an estimate for the total error we will again take modulus on both sides. Remember we know an estimate for the mathematical error in the last class. We have derived an expression for the mathematical error.

And we have also learned to estimate mathematical error using that expression in certain examples. Therefore, we have an idea of how to estimate mathematical error at least in certain particular cases. We also derived an estimate for the arithmetic error in our previous slide.

(Refer Slide Time: 17:59)



Total Error in Polynomial Interpolation

Let us now estimate the total error, which is given by

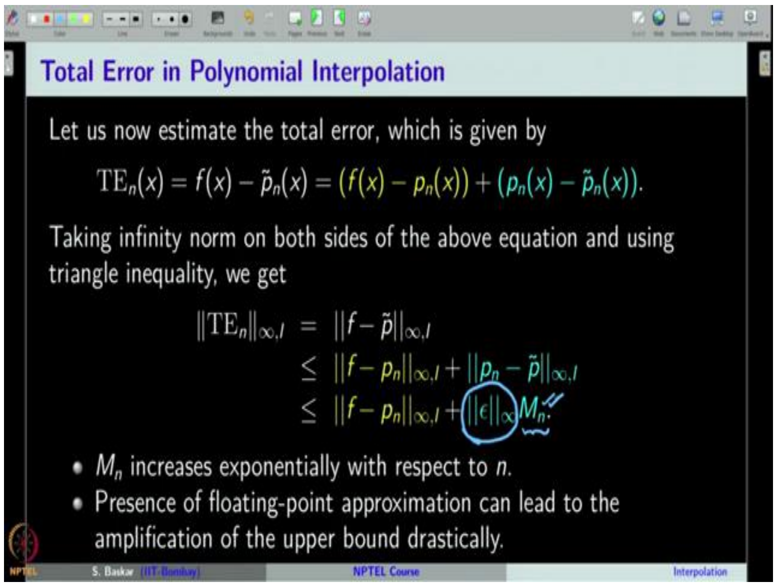
$$\mathrm{TE}_n(x) = f(x) - \tilde{p}_n(x) = (f(x) - p_n(x)) + (p_n(x) - \tilde{p}_n(x)).$$

Taking infinity norm on both sides of the above equation and using triangle inequality, we get

$$\begin{aligned} \|\mathrm{TE}_n\|_{\infty,I} &= \|f - \tilde{p}\|_{\infty,I} \\ &\le \underbrace{\|f - p_n\|_{\infty,I}}_{ME} + \underbrace{\|p_n - \tilde{p}\|_{\infty,I}}_{AE} \end{aligned}$$

Therefore, with this knowledge, we can now get an estimate for the total error. Remember the infinite norm of the total error is equal to the infinite norm of this function, $f - \tilde{p}$. And now that is less than or equal to the infinite norm of the mathematical error plus the infinite norm of the arithmetic error.

(Refer Slide Time: 18:25)



Total Error in Polynomial Interpolation

Let us now estimate the total error, which is given by

$$\mathrm{TE}_n(x) = f(x) - \tilde{p}_n(x) = (f(x) - p_n(x)) + (p_n(x) - \tilde{p}_n(x)).$$

Taking infinity norm on both sides of the above equation and using triangle inequality, we get

$$\begin{aligned} \|\mathrm{TE}_n\|_{\infty,I} &= \|f - \tilde{p}\|_{\infty,I} \\ &\le \|f - p_n\|_{\infty,I} + \|p_n - \tilde{p}\|_{\infty,I} \\ &\le \|f - p_n\|_{\infty,I} + \|\epsilon\|_{\infty} M_n. \end{aligned}$$

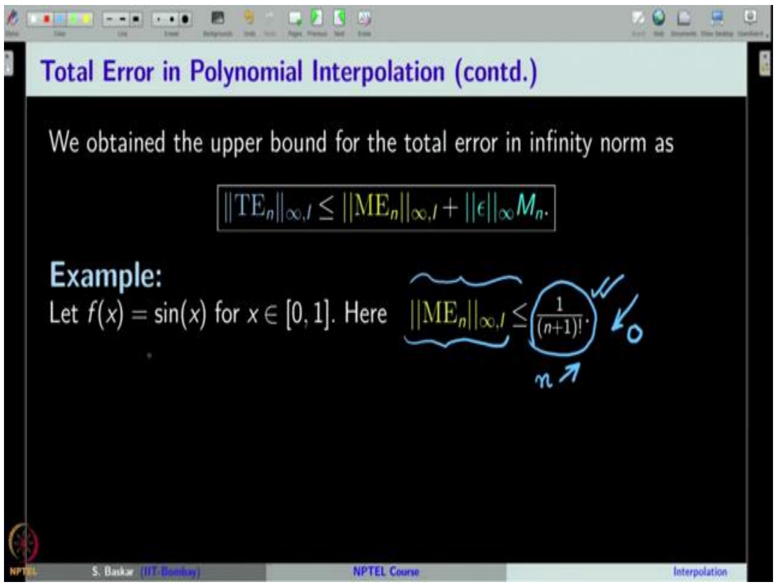
- M_n increases exponentially with respect to n.
- Presence of floating-point approximation can lead to the amplification of the upper bound drastically.

S. Baskar (IIT-Bombay) NPTEL Course Interpolation

Now, we have a way to estimate this, let us not get into that. But in this lecture we have seen that the arithmetic error can be dominated by this term, where this quantity is actually a badly behaving quantity. In the sense that as *n* increases, this M_n gross exponentially. And that shows that even the total error can also grow exponentially.

And that gives us a possibility that even if this floating point error is very small by choosing appropriately large value of *n*, you may land up with a large total error because of this quantity M_n. That is what is the bad news from this error analysis of the interpolating polynomial. And this is especially a serious problem when we are working with equally spaced nodes. That is more important. Let us try to illustrate this through an example recall that this is the estimate that we have for the total error.

(Refer Slide Time: 19:47)



Total Error in Polynomial Interpolation (contd.)

We obtained the upper bound for the total error in infinity norm as

$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \|\mathrm{ME}_n\|_{\infty,I} + \|\epsilon\|_{\infty} M_n.}$$

Example:

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\underbrace{\|\mathrm{ME}_n\|_{\infty,I}}$ $\le \frac{1}{(n+1)!}.$ $\checkmark \swarrow 0$

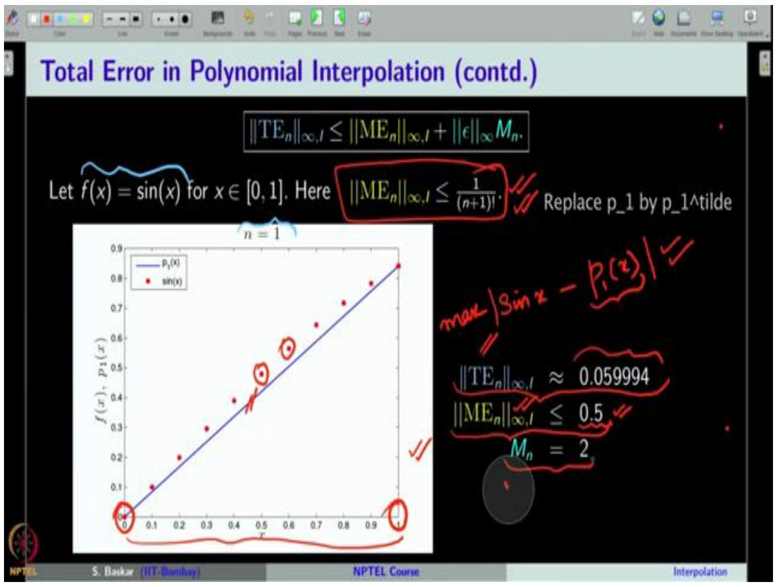
$n \nearrow$

And now let us try to see how the total error behaves in the polynomial interpolation, when we go on increasing n and tries to approximate a function. As an example, let us take the well-known function $f(x) = sin(x)$ and let us do this experiment in the interval $(0,1)$. And see how the total error behaves, as we go on increasing the degree of the interpolating, polynomial n.

Recall that from the expression of the mathematical error, we can see that the infinite norm of the mathematical error is in fact bounded by this number. From here you can see that as you go on increasing n, the upper bound is decreasing to 0 nicely. Therefore, as far as the mathematical error is concerned, the polynomial interpolation is going to behave well as we go on increasing n.

Remember all these problems that we have noticed in our analysis is when you keep on increasing n, that is when you go to approximate the function for a larger and larger value of n. We tend to increase the arithmetic error. That is what we have understood from this analysis. And we are trying to see this behavior through this example.

(Refer Slide Time: 21:36)



Total Error in Polynomial Interpolation (contd.)

$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \|\mathrm{ME}_n\|_{\infty,I} + \|\epsilon\|_\infty M_n.}$$

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|\mathrm{ME}_n\|_{\infty,I} \le \frac{1}{(n+1)!}$. Replace p_1 by p_1^tilde

$\max |\sin x - p_1(x)|$

$\|\mathrm{TE}_n\|_{\infty,I} \approx 0.059994$

$\|\mathrm{ME}_n\|_{\infty,I} \le 0.5$

$M_n = 2$

S. Baskar (IIT-Bombay) NPTEL Course Interpolation

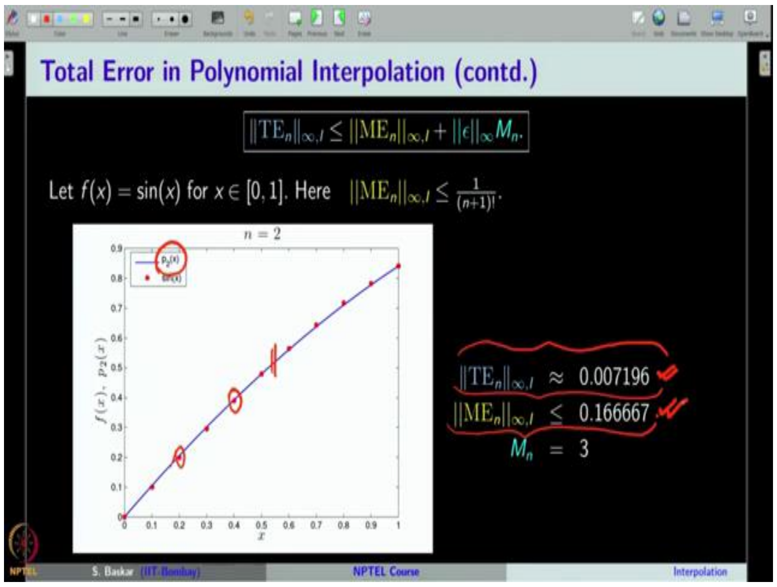
Let us start with approximating this sin function by the linear polynomial interpolation, that is $p_1(x)$. In this graph the blue solid line represents the graph of the interpolating polynomial, p_1 and the red dots represents the graph of the sin function in the interval $(0,1)$. Note that the polynomial $p_1(x)$ is obtained by taking the nodes as 0 and 1. What is the total error? Total error is nothing but $sin\, x - \tilde{p}_1(x)$.

What you do? You find the value of sin at many points in the interval $(0,1)$ and also you find the value of $\tilde{p}_1(x)$. At many points in the interval $(0,1)$ and then take the modulus of that and take the maximum over all these numbers. And that is how, I obtained this total error in this way of computing it happens to be in my calculation as 0.06 approximately. And I am taking that as the total error involved in this computation.

Well, we can also get an estimate for the mathematical error using this estimation, where you have to put n = 2 and therefore, we have 0.5 as the upper bound for the mathematical error involved in the polynomial interpolation. Remember, this in general cannot be obtained on a computer it is only coming theoretically through this inequality.

It means what if we would have computed $p_1(x)$ without involving any arithmetic error, then we would have got the polynomial interpolation with error as something less than 0.5. That is what it says. In fact, on a computer we had a better approximation. We know precisely that the total error is much less than 0.5. Well, we can also find M_1 and that is equal to 2.

(Refer Slide Time: 24:12)



Total Error in Polynomial Interpolation (contd.)

$$\boxed{\|TE_n\|_{\infty,I} \le \|ME_n\|_{\infty,I} + \|\epsilon\|_{\infty} M_n.}$$

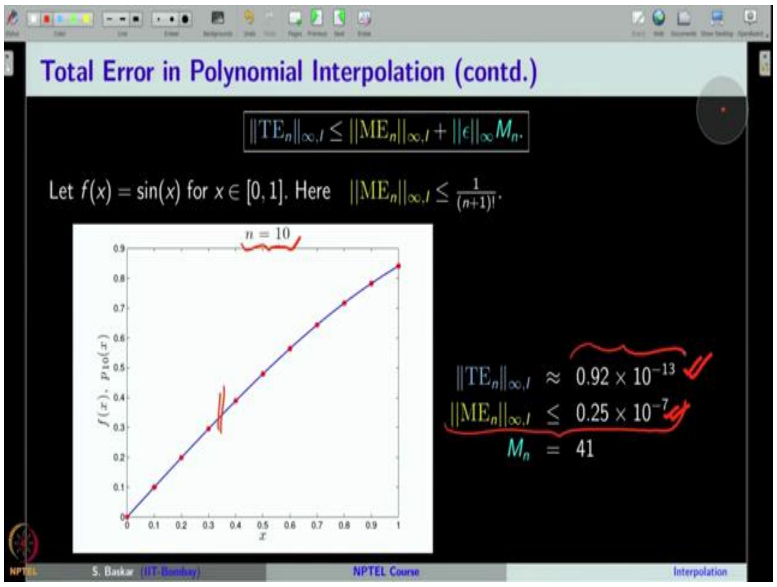
Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|ME_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$\|TE_n\|_{\infty,I} \approx 0.007196$
$\|ME_n\|_{\infty,I} \le 0.166667$
$M_n = 3$

Let us go to increase n from 1 to 2 and thereby we get the quadratic interpolation for the sin function and that is shown again in blue solid line whereas this graph of sin function in the interval $(0,1)$ is just denoted in the red dots. Again, you can see that the polynomial p_2 is approximating sin function quite well, on the graph. And the total error now is something like 10^{-2}.

And again, mathematical error also indicates that we will get a better approximation than p_1. However, our precise total error value shows that we have a really a nice approximation. Then what is actually predicted theoretically for the mathematical error, Let us go to increase again the degree of the polynomial.

(Refer Slide Time: 25:15)



Total Error in Polynomial Interpolation (contd.)

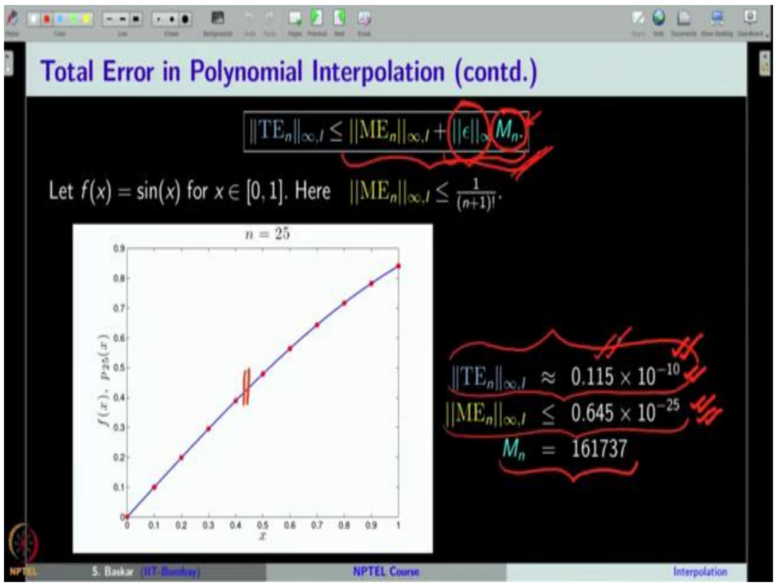
$$\boxed{\|TE_n\|_{\infty,I} \le \|ME_n\|_{\infty,I} + \|\epsilon\|_{\infty} M_n.}$$

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|ME_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$\|TE_n\|_{\infty,I} \approx 0.92 \times 10^{-13}$
$\|ME_n\|_{\infty,I} \le 0.25 \times 10^{-7}$
$M_n = 41$

Let me go a bit fast and show you what I got for n = 10. This time again graphically you can see that $p_{10}(x)$ is approximating sin function very well, with the total error of something nearby 10^{-13}. Again, through the theoretical estimate, the mathematical error is supposed to be less than or equal to something 10^{-7}. Of course, computationally we are doing very well up to n = 10.

(Refer Slide Time: 25:55)



Total Error in Polynomial Interpolation (contd.)

$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \|\mathrm{ME}_n\|_{\infty,I} + \|\epsilon\|_\infty M_n}$$

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|\mathrm{ME}_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$\|\mathrm{TE}_n\|_{\infty,I} \approx 0.115 \times 10^{-10}$

$\|\mathrm{ME}_n\|_{\infty,I} \le 0.645 \times 10^{-25}$

$M_n = 161737$

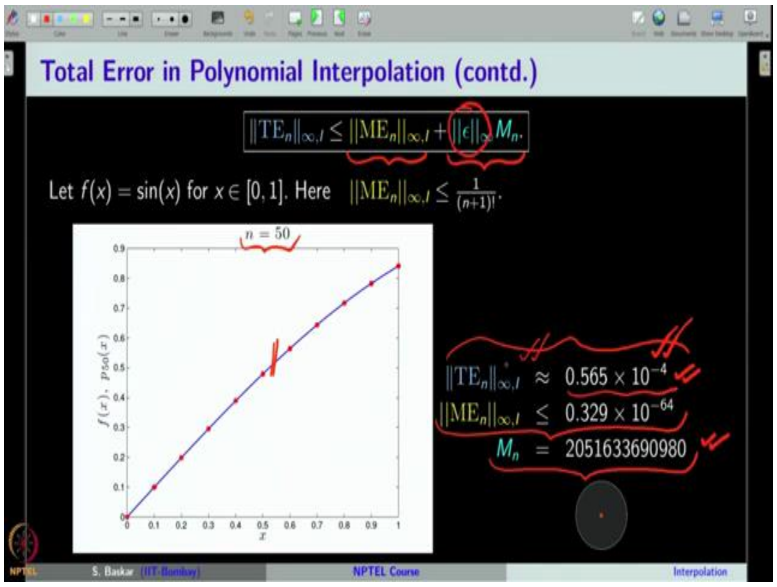
S. Baskar (IIT-Bombay) NPTEL Course Interpolation

Now, let us go to n = 25. Graphically we are doing very well. But carefully observe what is happening with it total error? The total error is now something 10^{-10}. If you go back with polynomial of degree 10, we had 10^{-30}. Now, by increasing the polynomial degree from 10 to 25, actually our total error became bad. However, you can see that the mathematical error indicates that you are supposed to get a very accurate approximation as far as the mathematical construction is concerned.

That is what the mathematical error says but what we got computationally is something worse than what we are supposed to get as indicated by the mathematical error. It means you can see that something is going wrong. Where are we going wrong? Well, it is purely because of the arithmetic error you can see what is the value of M_n, $M_n$where n = 25 is something like 1,65,000. That big number is now multiplied, maybe with a very small number.

We are making a very small rounding error but that is now multiplied with a bigger number and that is giving a bigger room for your total error to increase. And that is what computationally happening here also. However, graphically if you see you do not see anything bad graphically p_{25} is also approximating the sin function nicely. Let us further go to increase the degree of the polynomial and see what happens.

(Refer Slide Time: 27:59)



Total Error in Polynomial Interpolation (contd.)

$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \underbrace{\|\mathrm{ME}_n\|_{\infty,I}} + \underbrace{\|\epsilon\|_{\infty} M_n}.}$$

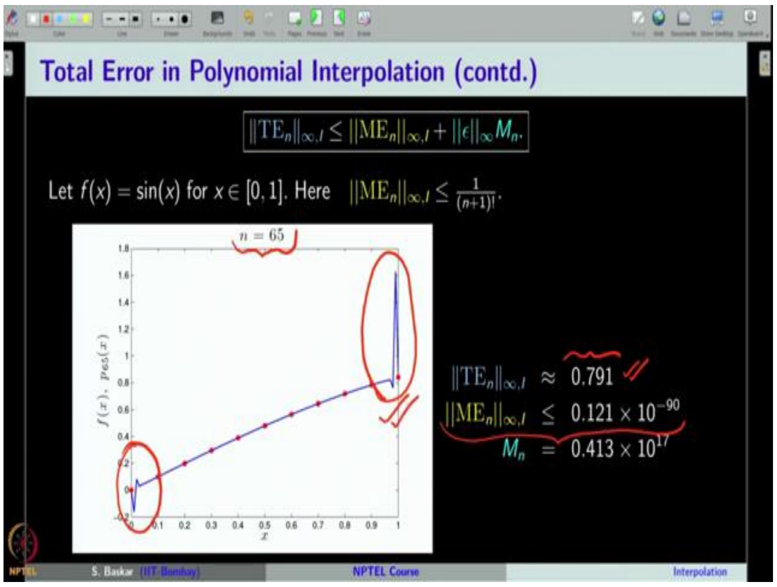
Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|\mathrm{ME}_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$$\|\mathrm{TE}_n\|_{\infty,I} \approx 0.565 \times 10^{-4}$$
$$\|\mathrm{ME}_n\|_{\infty,I} \le 0.329 \times 10^{-64}$$
$$M_n = 2051633690980$$

When I take *n* = 50, again graphically we do not see anything bad. But total error now further increased from 10^{-10} to 10^{-4} where the mathematical error indicates that your polynomial interpolation is supposed to give much better approximation than what you obtained from p_{25}. But that is not happening in reality on a computer.

Again, you can see why such a drastic amplification of the total error happened when mathematical error is very nice. Your arithmetic error is actually spoiling the approximation, the reason is mainly because you can see what is the value of M_n. And that is giving a room for your total error to increase and your total error is indeed increasing.

(Refer Slide Time: 29:00)


Total Error in Polynomial Interpolation (contd.)

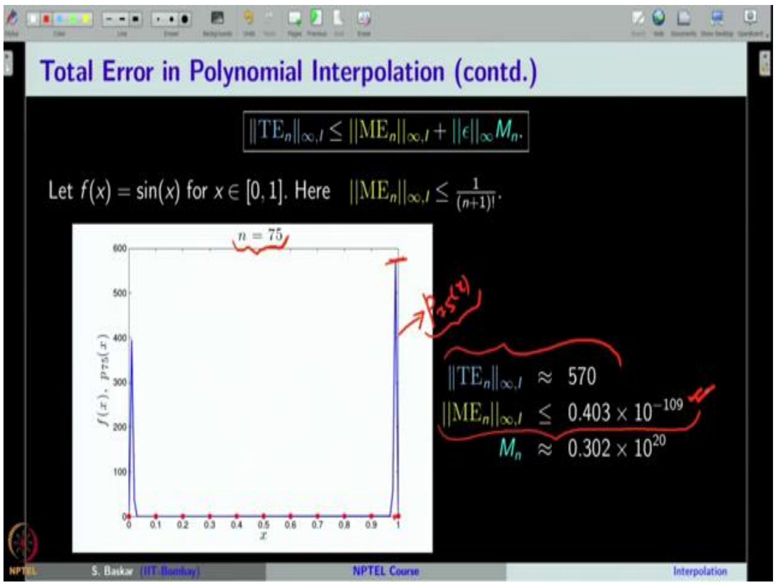
$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \|\mathrm{ME}_n\|_{\infty,I} + \|\epsilon\|_\infty M_n.}$$

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|\mathrm{ME}_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$$\|\mathrm{TE}_n\|_{\infty,I} \approx 0.791$$
$$\|\mathrm{ME}_n\|_{\infty,I} \le 0.121 \times 10^{-90}$$
$$M_n = 0.413 \times 10^{17}$$

Now, from here let us further increase the degree of the interpolating polynomial to n = 65. Now, you can see that even on the graph, you see that the approximation is very bad, especially you can notice error at the boundaries of the interval. You can see the total error now is considerably very big, whereas the mathematical error says that your polynomial interpolation should be almost as accurate as the sin function. But in reality, it is not happening like that.

(Refer Slide Time: 29:41)


Total Error in Polynomial Interpolation (contd.)

$$\boxed{\|\mathrm{TE}_n\|_{\infty,I} \le \|\mathrm{ME}_n\|_{\infty,I} + \|\epsilon\|_\infty M_n.}$$

Let $f(x) = \sin(x)$ for $x \in [0,1]$. Here $\|\mathrm{ME}_n\|_{\infty,I} \le \frac{1}{(n+1)!}$.

$$\|\mathrm{TE}_n\|_{\infty,I} \approx 570$$
$$\|\mathrm{ME}_n\|_{\infty,I} \le 0.403 \times 10^{-109}$$
$$M_n \approx 0.302 \times 10^{20}$$

Let us again go one more step higher and take n = 75. You can see that the graph of the polynomial $p_{75}(x)$ is shown in blue colour. Just imagine that if you are using $p_{75}(x)$ as an approximation for sin function then what you get is the value of sin function is something greater than 500 at some point in the interval $(0,1)$ which is obviously absurd. And you can also see the total error is now grown to 570.

Whereas, the mathematical error still says that you are almost accurate, exactly capturing the sin function. So, this example shows that even if you have a tool which is mathematically very good. The situation may be entirely different when you go to implement it on a computer. So that shows the importance of understanding and analysing the numerical methods before going into the implementation of the methods. And this is very important.

Otherwise, you may be computing and believing something as the solution to your problem which is no way near to your solution. This can actually lead to some disasters. This also shows the importance of analysis not only mathematically, but also you have to understand the analysis behind the computation of the method. With this note, let us close this class. Thank you for your attention.