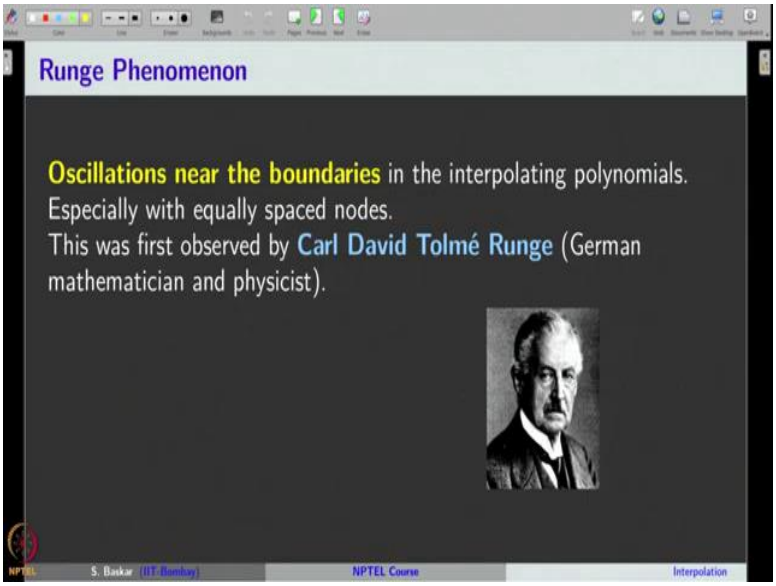
Numerical Analysis
Prof. S. Baskar
Department of Mathematics
Indian Institute of Technology – Bombay

Lecture – 44
Polynomial Interpolation: Runge Phenomenon and Piecewise Polynomial Interpolation

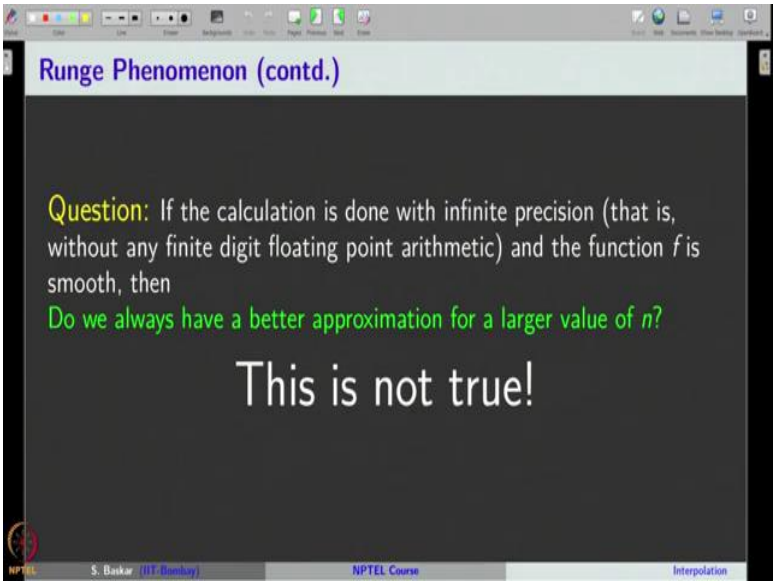
Hi, we are discussing errors in Polynomial Interpolation. In this lecture, we will discuss an interesting phenomenon called Runge Phenomenon. Runge Phenomenon is also connected to errors in polynomial interpolations. Then we will also see that there are two ways that we can minimize errors in polynomial interpolation. One way to minimize the errors in polynomial interpolation is to construct polynomial interpolations on unequally spaced nodes and another approach is to go for piecewise polynomial interpolations.

(Refer Slide Time: 00:59)



Let us first discuss the Runge phenomenon. Runge phenomenon is all about certain oscillations near the boundaries in interpolating polynomials of some functions. Especially when we go to construct interpolating polynomials on equally spaced nodes. This phenomenon was first observed by a German mathematician and physicist named Carl Runge.

(Refer Slide Time: 01:27)

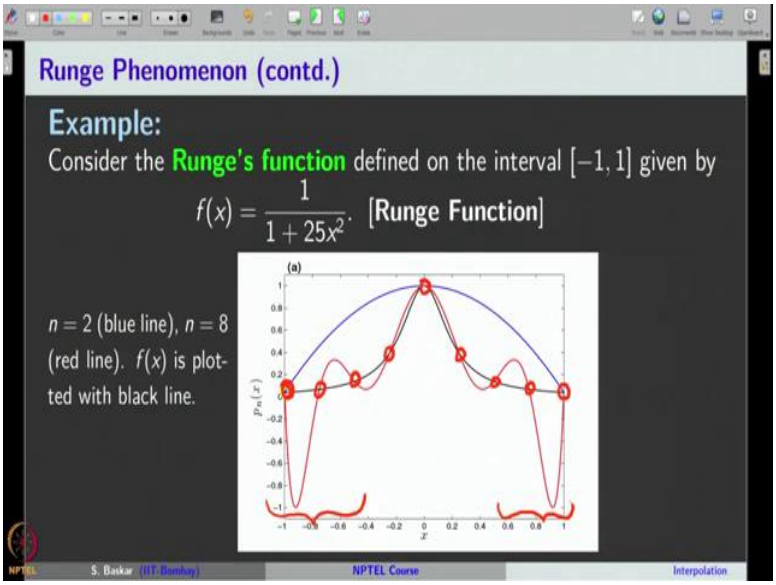


In fact, we have already observed the Runge phenomenon in the example that we have discussed in the last class. If you recall, we have discussed an example where we constructed interpolating polynomial for sin function. There we observe that when we go on increasing the degree of the polynomial, we observed oscillations appearing near the boundaries of the interval of interest.

We attributed this phenomenon to arithmetic error in that example. That is because in that particular example, mathematical error was well behaving because it was bounded by $\frac{1}{(n+1)!}$. Therefore, as n tends to infinity, the upper bound of the mathematical error was tending to 0. And that makes the absolute value of the mathematical error also tends to 0.

On the other hand, we have seen that arithmetic error was growing drastically, especially when we construct the polynomial interpolation on equally spaced nodes. Now, the question is: are such oscillations are always due to arithmetic error? In other words, if we make the calculation of this interpolating polynomial using infinite precision computation, do we always get a good approximation? That is the question. The answer is again, no.

(Refer Slide Time: 03:06)



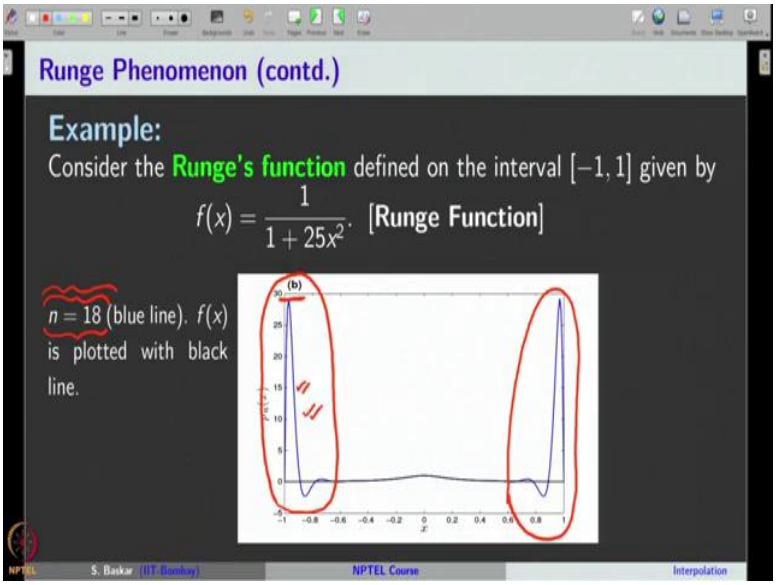
Runge has given an example where you can observe that oscillations can occur even for polynomials of degree much less. Let us consider an example where $f(x)$ is given by $\frac{1}{1+25x^2}$, this is an example of a Runge function. In this figure we are constructing and showing two interpolating polynomials of the function $f(x)$. One is the quadratic polynomial which is shown in blue solid line.

You can see that quadratic polynomial is approximating the Runge function rather in a poor way. This is rather understandable because we are giving the information about the function only at these three node points. Therefore, it is understandable for why the quadratic polynomial is not approximating the Runge function nicely. Next is, let us takes nine nodes and construct $p_8(x)$.

That is interpolating polynomial of degree 8. Now, we are giving the information about the function at nine different node points. But still you can see that $p_8(x)$ which is shown in the red solid line is not approximating the Runge function that well. In fact, we can see that the polynomial is oscillating between the node points. And this oscillation is very nicely seen towards the boundary of the interval, which is also called edges.

This is what is called the Runge phenomenon. Let us go to increase the degree of the polynomial and see how the interpolating polynomial is behaving near the boundary.

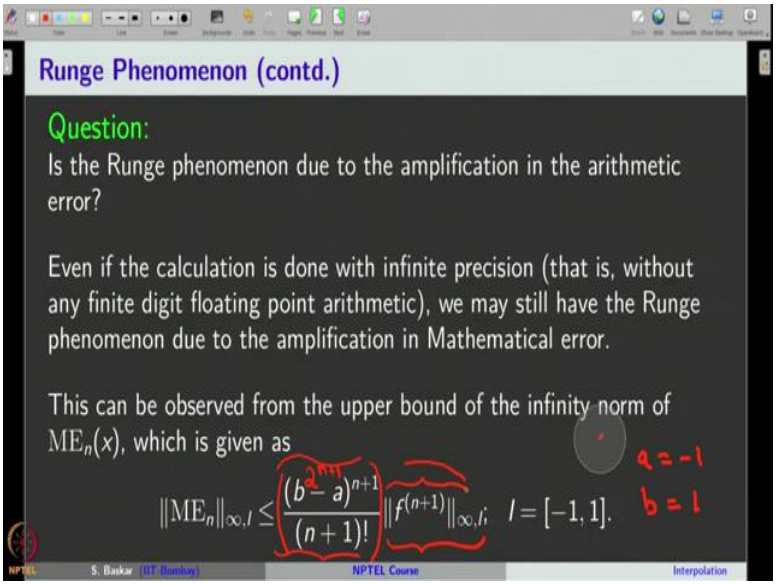
(Refer Slide Time: 05:15)



Let us take $n = 18$ and thereby we are constructing an 18th degree polynomial interpolating the Runge function and it is shown in this blue, solid line. You can see that the polynomial is oscillating widely near the boundary. This is typically what is called the Runge phenomenon. In other words, we say that the Runge phenomenon is clearly visible in the interpolating polynomial of degree 18 for this function.

And in fact, you can see that the total error is something near to 30 which is clearly not acceptable as a good approximation.

(Refer Slide Time: 06:08)



Now, the question is, the Runge phenomenon purely due to the amplification of the arithmetic error? This is a quite natural question that one can ask after seeing the lost example in our previous lecture. But the fact is, even if you construct the interpolating polynomial with infinite

precision, we may still observe the Runge phenomenon that is because in the Runge function the amplification of the error.

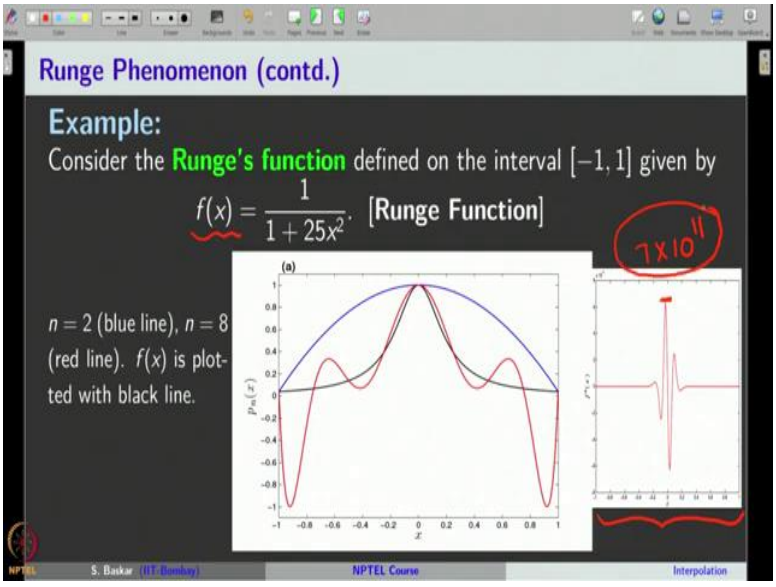
That is the oscillation near the boundaries is actually happening due to the mathematical error mainly. It may be happening due to arithmetic error also. But you see, we are only considering 18th degree polynomial and we are constructing this polynomials on a 64 bit processor. Mostly 64 bit processors have the capacity to handle the rounding errors for such a small *n*.

Therefore, this Runge phenomenon may be coming more because of the mathematical error. Let us try to have a close look at the mathematical error. If you recall, in one of our last lectures, we have derived an expression for the mathematical error involved in the polynomial interpolation. And we have also derived an upper bound for the mathematical error.

Let us take this upper bound and let us view this expression in two parts one is $\frac{(b-a)^{n+1}}{(n+1)!}$. In our example, $a = -1$ and $b = 1$. Therefore, this is $\frac{2^{n+1}}{(n+1)!}$. You can clearly see that this part of the error is a well behaved expression, especially as you go on increasing the value of *n*. Therefore, if at all something is going wrong it must be going from this part of the upper bound.

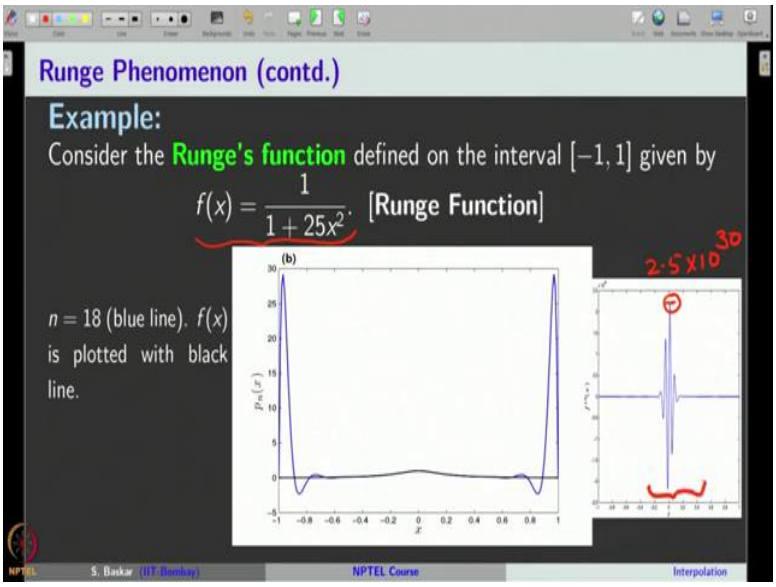
Let us try to have a close look at how the $(n+1)$ order derivative of the function *f* behaves. Recall the infinite norm of any function is nothing but the maximum of the modulus of the function.

(Refer Slide Time: 08:48)



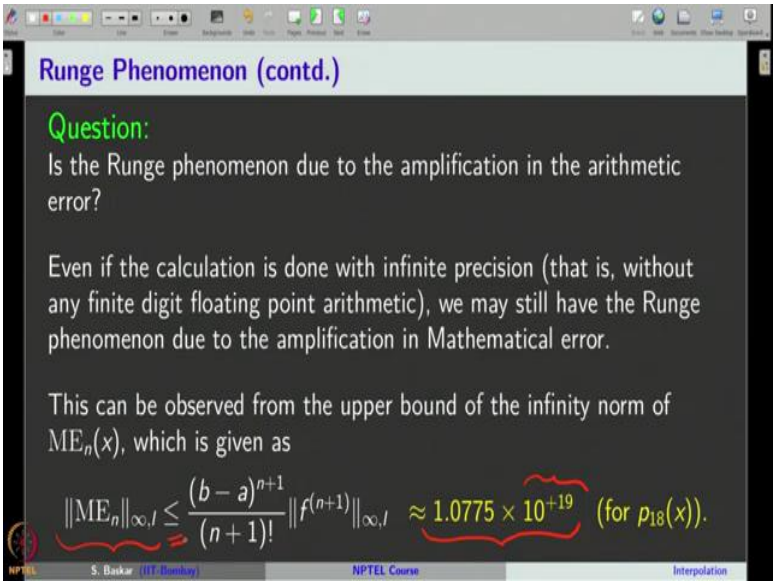
In this small window, I have plotted the graph of the 19th order derivative of the Runge function. Here you can see that Runge function is attaining it is maximum somewhere near the midpoint of the interval. And the maximum is something like 7×10^{11}. So, it is quite a large value.

(Refer Slide Time: 09:23)



Let us go to see how the 19th derivative of the function *f* looks like? Again, you can see that the 19th order derivative of the Runge function has its maximum very near to the midpoint. And it is value is something like 2.5×10^{30}.

(Refer Slide Time: 09:49)



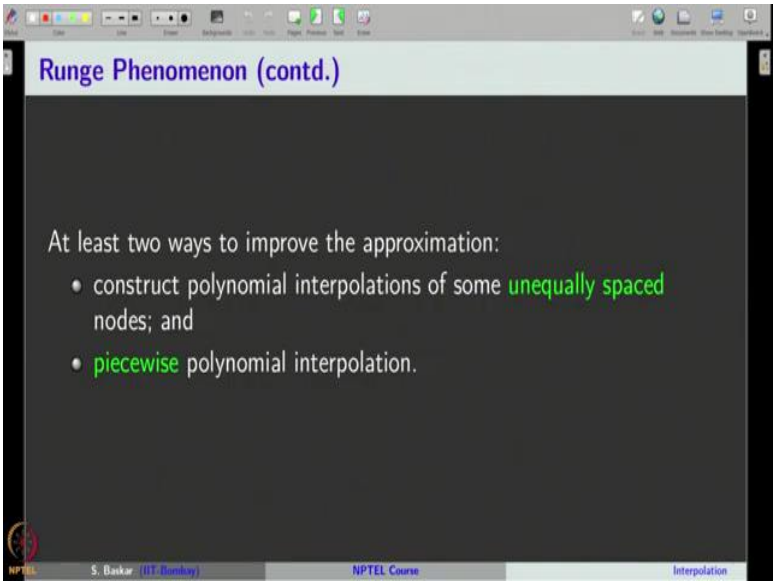
So, these are quite big. In fact, you can precisely compute the upper bound of the 19th order derivative of the Runge function. And it is approximately given by 10^{19}. That gives us a severe warning that the mathematical error may be as large as 10^{19}. It only gives a warning. It need

not be as big as this because it is just an inequality. Nevertheless, there is a possibility that the mathematical error can be as big as 10^{19}.

Therefore, the key takeaway is that if you go on increasing the degree of the interpolating polynomial at certain stage, surely, your arithmetic error will start increasing quite rapidly and spoils the accuracy of the interpolating polynomial. Especially, when you are working on the equally spaced nodes. Even if you are not taking *n* to be very large, there are certain functions like Runge functions.

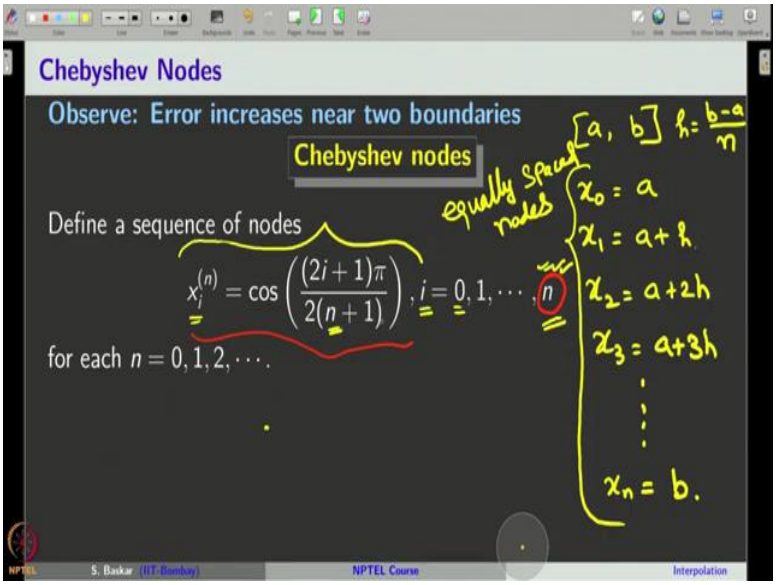
If the derivatives are growing rapidly somewhere near the midpoint of the interval. Then again there is a severe warning that the mathematical error may grow drastically. These are the key takeaways such drastic increase in the error generally happens with equally spaced nodes.

(Refer Slide Time: 11:33)



There are at least two ways that we can improve the approximation of interpolating polynomials. One is to construct the interpolating polynomial with some appropriately chosen and equally spaced nodes. And another approach is to go for piecewise, polynomial interpolations. Let us take up these two approaches one by one.

(Refer Slide Time: 12:01)



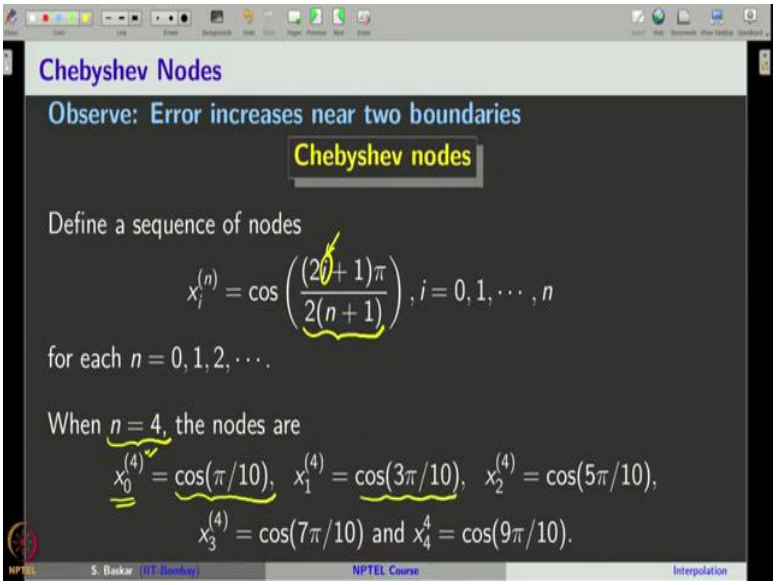
Let us first see how we can construct some unequally spaced nodes in order to improve the approximation of the interpolating polynomial? Observe that the error drastically increases near the boundaries of the interval which is commonly called as edges. Therefore, one idea is to put more points near the boundaries of the interval. When compared to the number of points that is nodes located away from the boundaries of the interval.

One such set of nodes is called the Chebyshev nodes. Chebyshev nodes are given like this. You are given a *n* for that you have to generate *n* + 1 nodes. If you recall equally spaced nodes are generated as $x_0 = a$ where the interval is $[a, b]$ and then $x_1 = a + h$, where *h* is given by $\frac{b-a}{n}$. Then $x_2 = a + 2h$, $x_3 = a + 3h$ and so on up to $x_n = a + nh$, which is nothing but *b*.

So, this is kind of formula that we use to generate equally spaced nodes. Now, what we are doing is you are given *n* now we are constructing *n* unequally spaced nodes, using this formula. For each i, x_i is given like this in order to specify that these nodes are generated with some value of *n* we will also including this in our notation. We will use the notation as superscript bracket *n* remember this is not to indicate derivative.

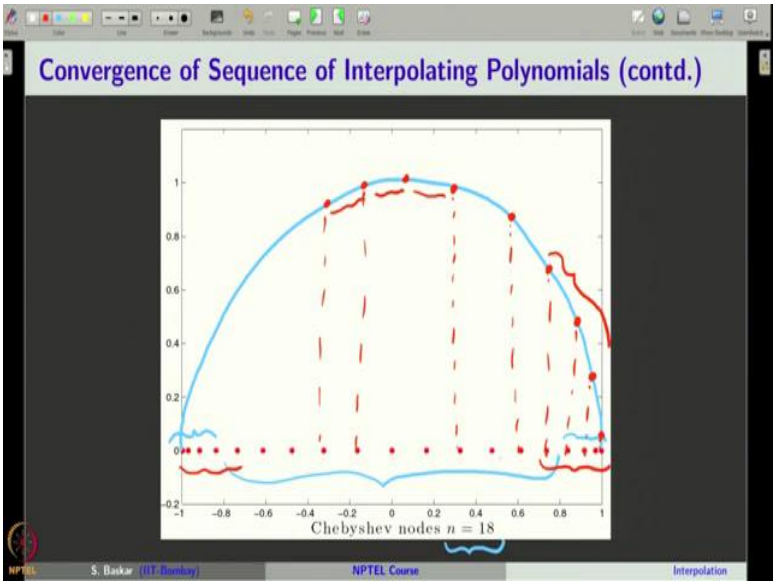
Generally, we use the same kind of notation to denote derivative of a function. Here, we are just using it to mention that this set of nodes is constructed with the value *n*. How these are defined? For each *i* you have $cos\left(\frac{(2i+1)\pi}{2(n+1)}\right)$. So, this is how the Chebyshev nodes are defined.

(Refer Slide Time: 14:48)



For instance, if you take $n = 4$ then you need to generate five node points. And these are denoted by $x_0^{(4)}$ and so, on. And they are given by $cos(\pi/10)$, that is, first you are taking $i = 0$ here, therefore, it is $cos(\pi/10)$ because $n = 4$. Similarly, $x_1^{(4)}$ is obtained by taking $i = 1$ and that is given by $cos(3\pi/10)$ and so on.

(Refer Slide Time: 15:33)

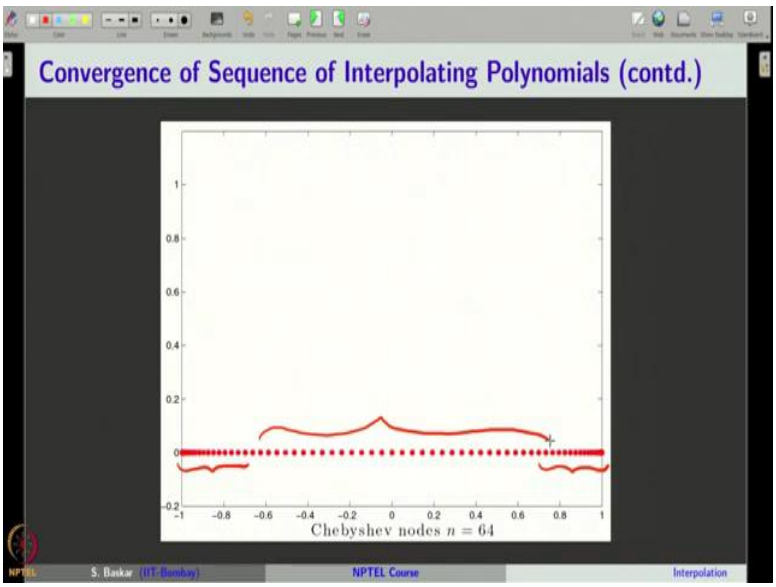


Let us see how these nodes are placed in the interval [–1, 1]. In this figure, we are showing the Chebyshev nodes for $n = 4$ in the red dots. Similarly, you can construct the submissive nodes for any given n. Let us construct the Chebyshev node for $n = 18$. In this case, we will get 19 node points and they are given like this. You can observe that the nodes are more clustered near the boundary.

Whereas they are sparsely placed when you come away from the boundary this is the main idea of the Chebyshev nodes. What the formula for the Chebyshev nodes is doing is, you just take the unit circle and then the points are located equally spaced along this unit circle. And then what you are doing is you are projecting them onto the *x* axis? So, this is why you can see that the points which are near the boundary they are getting more clustered here.

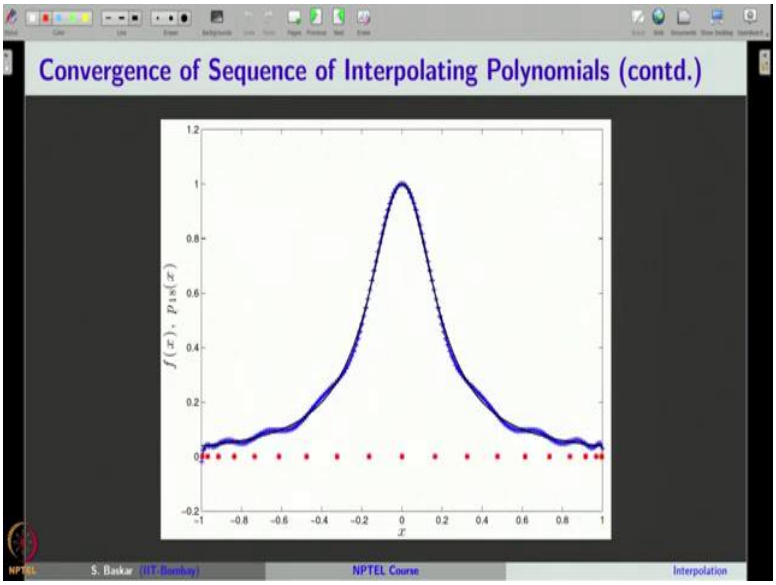
And similarly here, whereas if you go towards the centre of the interval, you can see that these are equally spaced. But when you project them on the *x* axis, they have unequally spaced position. That is what the idea of Chebyshev nodes.

(Refer Slide Time: 17: 19)



Let us see how Chebyshev nodes are placed when $n = 32$. You can write a small python code to generate these nodes. You can see again that the nodes are more clustered near the boundary. And you can see for $n = 64$, very clearly the points are getting accumulated near the boundaries when compared to the points which are placed away from the boundary. They are little sparsely placed. Let us try to construct the interpolating polynomial now using Chebyshev nodes.

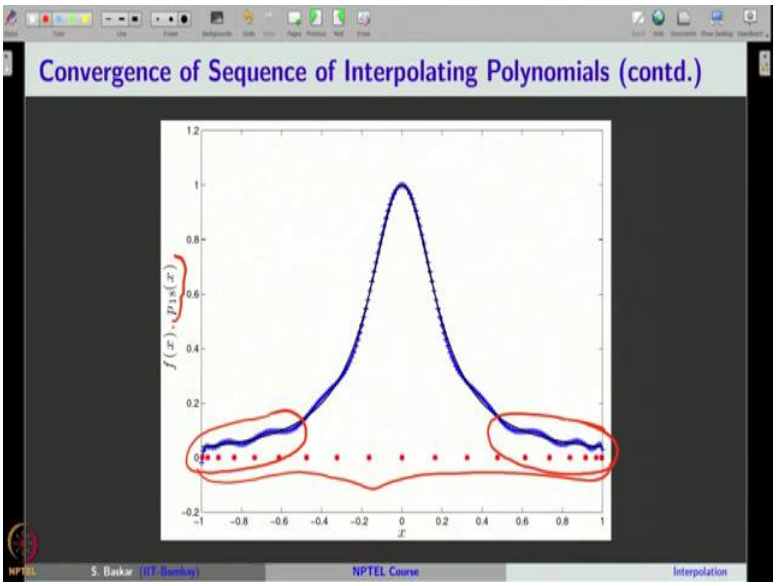
(Refer Slide Time: 18:04)



Let us take $n = 4$ and we have 5 corresponding Chebyshev nodes like how we constructed with equally space nodes. In a similar way, we can also construct interpolating polynomial, either using Lagrange form or Newton's form. Remember in these two forms we have never put any conditions on how these nodes have to be distributed in an interval? They can be distributed in any form you want.

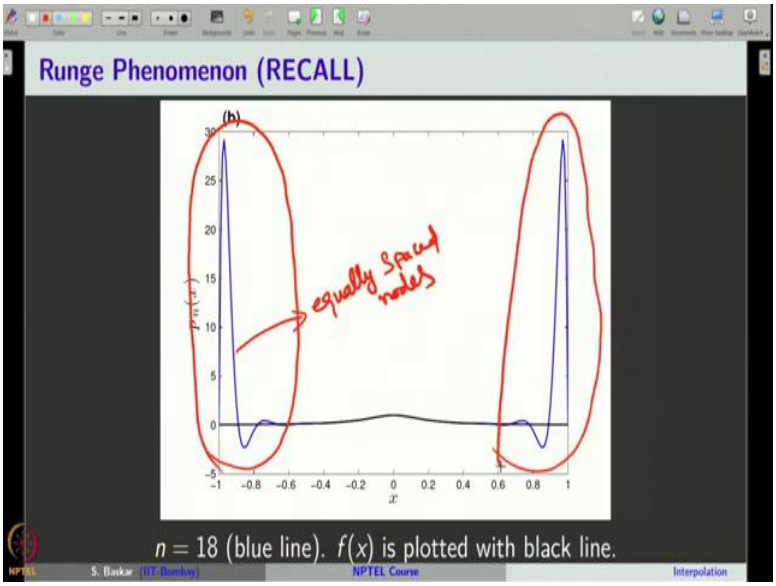
So, in particular, if you have equally spaced nodes, they suffer from the Runge phenomenon. That is why we are looking for an alternate idea, we are going for the Chebyshev nodes. You can see that again, $n = 4$ with Chebyshev nodes is showing some kind of Runge phenomenon near the boundary. Let us go to construct the interpolating polynomial of little higher degree but now using Chebyshev nodes.

(Refer Slide Time: 19:13)



We are now constructing 18th degree polynomial using Chebyshev nodes. We are now given 19 node points, not equally spaced but Chebyshev nodes. You can see that the Runge phenomenon is still observed, but not as bad as what we got with the equally spaced nodes.

(Refer Slide Time: 19:41)

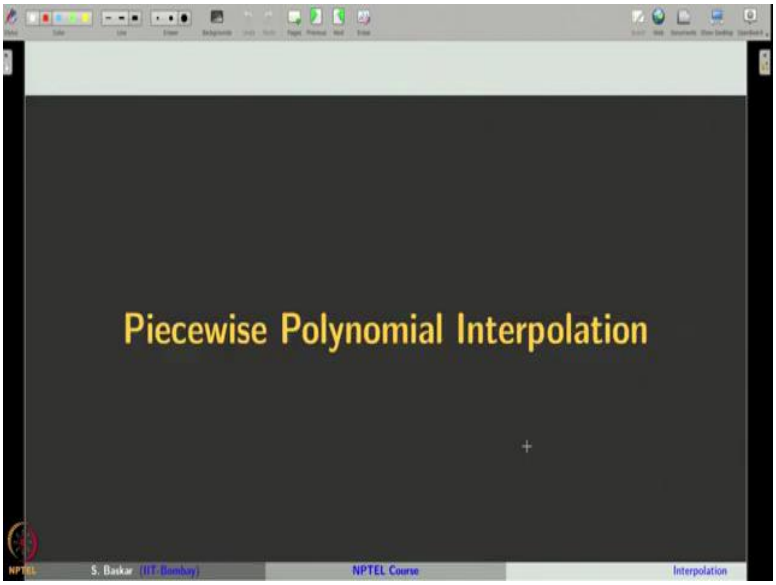


If you recall, we have shown $p_{18}(x)$ with equally spaced nodes. This is with equally spaced nodes. You can see such a bad oscillation near the boundaries. That is the Runge phenomenon is very much visible. Whereas the same 18th degree polynomial now constructed with Chebyshev nodes has improved the approximation drastically. Let us go ahead and compute 32 degree polynomial of the Runge function.

Now, using Chebyshev nodes you can see the power of clustering more points near the boundary when compared to the points distributed towards the centre of the interval. You can see that the Runge phenomenon, at least visually is almost suppressed. Of course, you can see a small error here but this is due to some bug in the program. It is not the numerical error. Next, you can see the polynomial of degree 64 constructed using Chebyshev nodes.

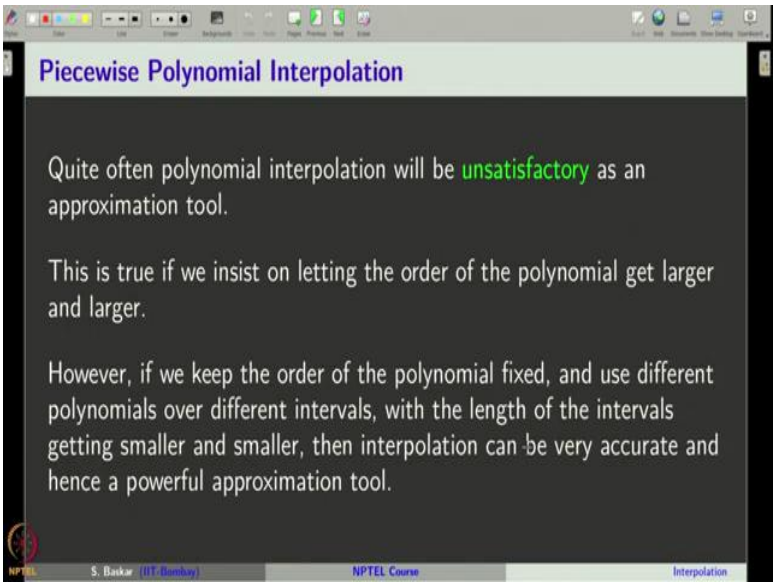
You can see that the approximation is pretty good. From here you can see the power of using unequally spaced nodes. In particular, it is the Chebyshev nodes. The main idea of Chebyshev nodes is to put more points near the boundaries of the interval. And to balance that you can have less points away from the boundary. If you would have constructed the polynomial p_{64} using equally spaced nodes, it would have gone much more worse.

(Refer Slide Time: 21:37)



Now, let us go to learn another approach to improve the approximation in polynomial interpolation. This is due to piecewise polynomial interpolations.

(Refer Slide Time: 21:49)

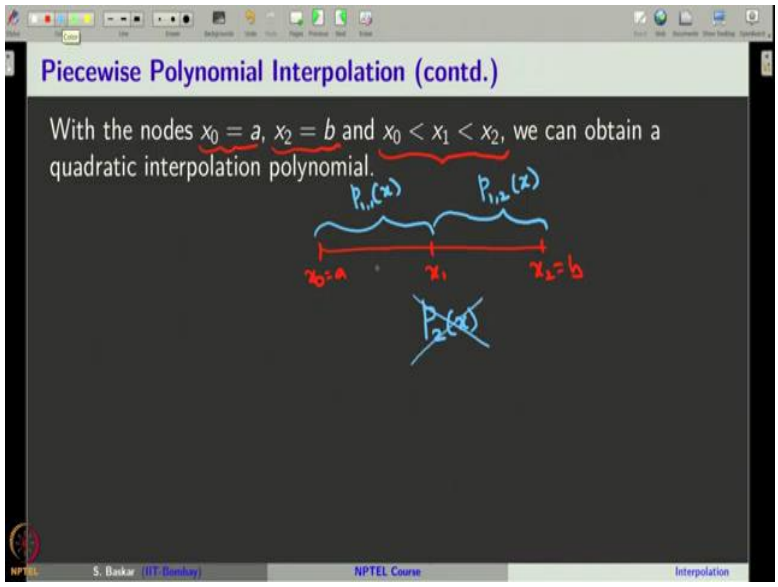


We have seen that when we go for polynomial interpolation, that is one polynomial on the entire interval. The approximation is not that satisfactory, especially when we work with equally spaced nodes and also when you go to construct interpolating polynomials with high degree. Now, in certain functions you cannot avoid giving more points. Especially, when you are working with rapidly varying function something like the Runge functions.

You need to give more points in order to capture all the variations of the function. In that way, if you go for the polynomial interpolation then it increases the degree of the polynomial. And even the arithmetic error will start playing a vital role in amplifying the error. In such cases,

what you can do is, you can fix the degree of the polynomial but you can apply the polynomial interpolation rather piecewise. That is the idea of piecewise polynomial interpolation.

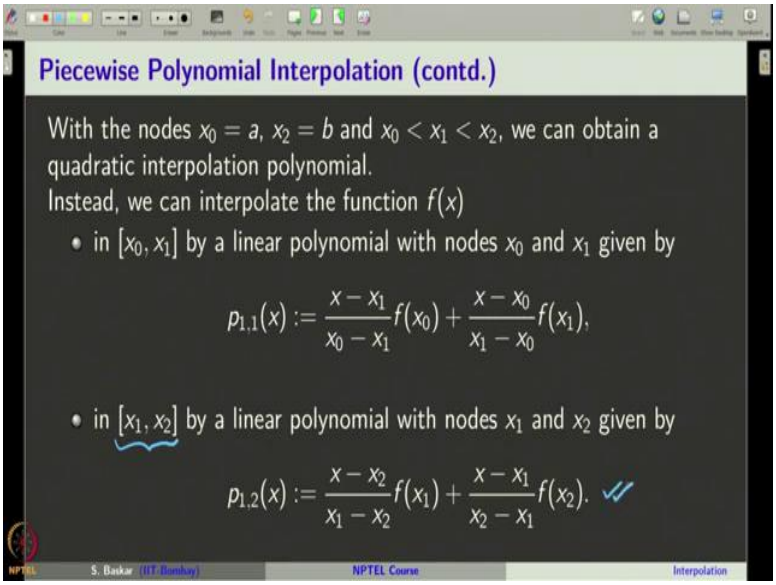
(Refer Slide Time: 23:05)



Let us just take an example and see this idea more clearly. Let us take three nodes x_0, x_1 and x_2 with $x_0 = a$ and $x_2 = b$. Therefore, you have x_0 which is equal to a then x_1 and then $x_2 = b$. Now, you can obtain a quadratic, polynomial interpolating a given function in this interval. An alternate idea is not to go for one single quadratic polynomial interpolating this data set.

Rather you go for a linear interpolating polynomial in this interval. Let us call it as $p_{1,1}(x)$. And then you go for another linear polynomial interpolation in this interval. Let us call it as $p_{1,2}(x)$. So, the idea is not to go for quadratic polynomial rather go for two different linear interpolating polynomials.

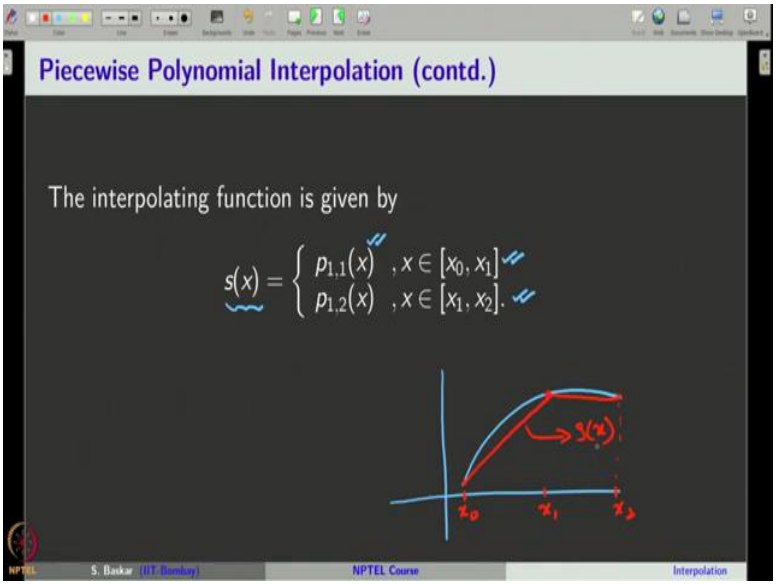
(Refer Slide Time: 24:18)



That is the idea of piecewise interpolation. Let us make this discussion more precise. What you do is, suppose you are given a data set with nodes as x_0, x_1, x_2 then consider the first sub interval $[x_0, x_1]$. You can write the linear interpolating polynomial, either in the Newton's form or the Lagrange form. Here I am showing the Lagrange form of the linear interpolating polynomial.

And this is only for the interval $[x_0, x_1]$ and then you take the second part of the interval. That is the second sub interval of the partition construct another linear interpolating polynomial. I have used a Lagrange form again here.

(Refer Slide Time: 25:07)

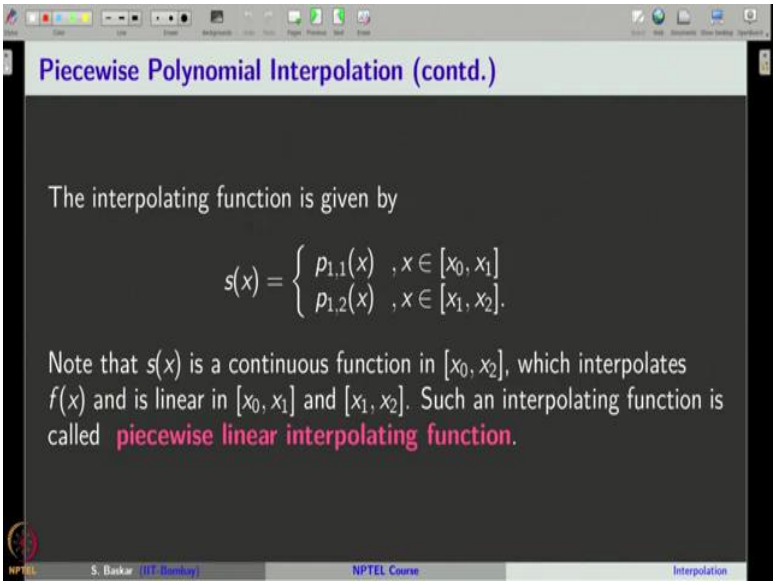


And then define the interpolating function. Let us denote it by $s(x)$ as $p_{1,1}(x)$ when $x \in [x_0, x_1]$ and $p_{1,2}(x)$ when $x \in [x_1, x_2]$. That is, suppose your function looks like this, now you

want to construct an approximation with three node points. So, what you are doing is you are taking this interval and making the linear interpolation here.

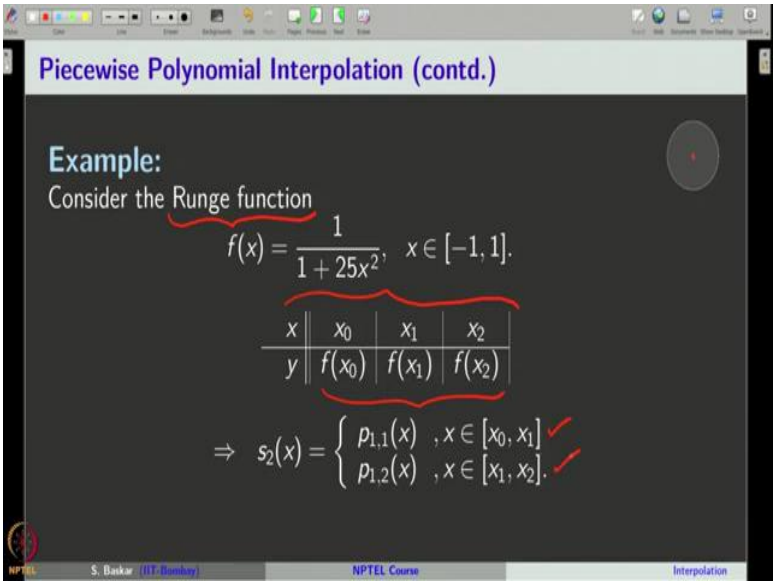
And then take the second part of this interval and make another linear interpolation and this is what is called $s(x)$.

(Refer Slide Time: 25:59)



And this is what is called the piecewise linear interpolating function. Once you understand this idea, you can also generalize it to piecewise quadratic interpolation, piecewise cubic interpolation and so on.

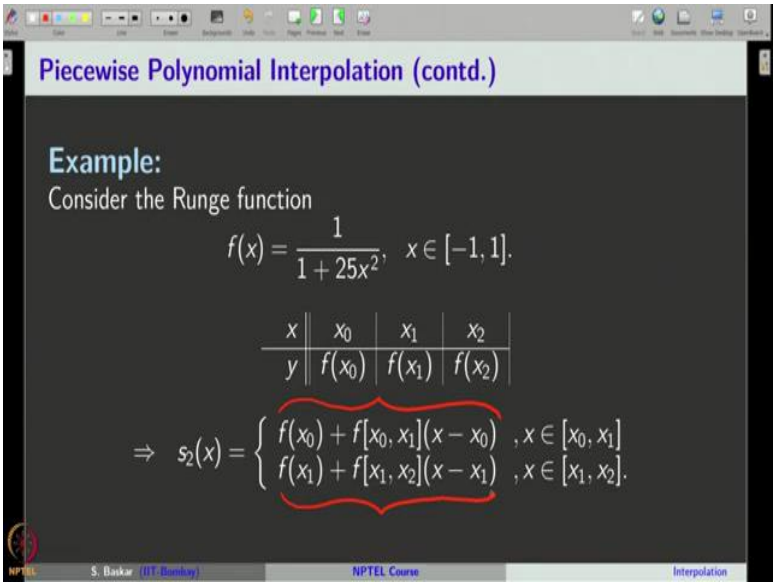
(Refer Slide Time: 26:14)



Let us take an example. Let us consider again the same Runge function. We have seen that one single polynomial interpolation with equally spaced nodes is not going to be a good idea to

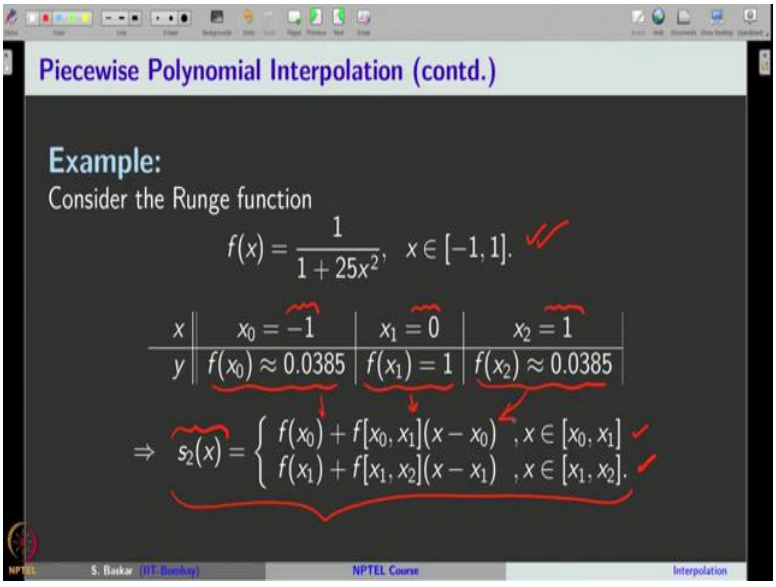
approximate functions like Runge functions. Let us take a data set consisting of three node points and the corresponding function values on them. They can be equally spaced nodes. And our interest is to constrict a linear interpolating polynomial.

(Refer Slide Time: 26:52)



Let me just put the Newton's form of interpolating polynomial here. There is no reason you can use either Lagrange or Newton, since I have already shown the expression for Lagrange in the previous slide. Now, I have shown the Newton's interpolating polynomial formula here. You can use anything you want.

(Refer Slide Time: 27:16)

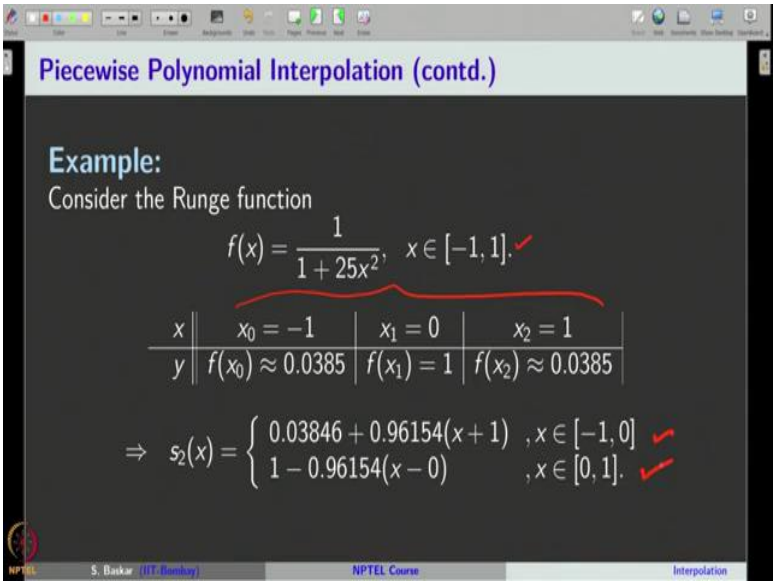


If you are given some values to x_0, x_1 and x_2 and correspondingly $f(x_0), f(x_1)$ and $f(x_2)$ can be calculated from this function. And thereby you can get an explicit expression for the piecewise linear interpolation. Since you have two pieces of linear interpolations, we will use

the notation $s_2(x)$ here. Let me take the nodes as equally spaced nodes so, $x_0 = -1$, $x_1 = 0$ and $x_2 = 1$.

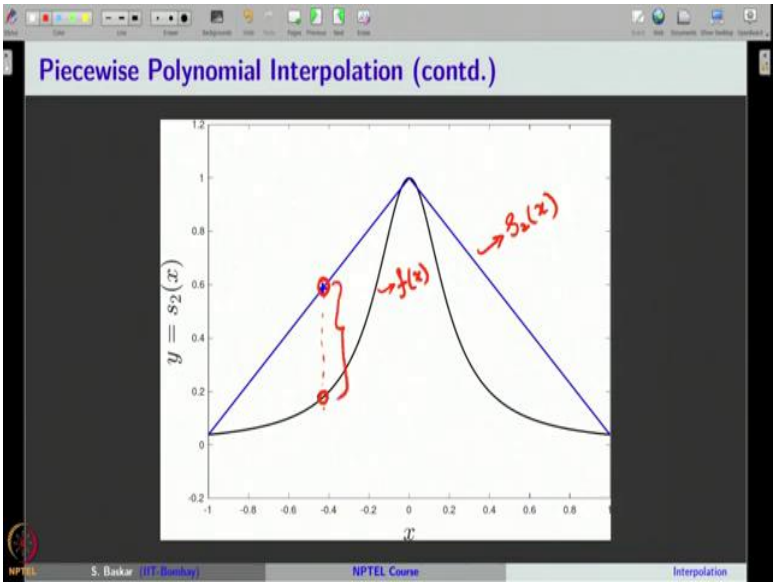
And find the corresponding values of the function f at these nodes and you can now plug in these values into this expression.

(Refer Slide Time: 28:04)



I leave it to you to do this calculation. You can see that the piecewise linear polynomial, interpolating the function $f(x)$ are these node points is given like this.

(Refer Slide Time: 28:23)

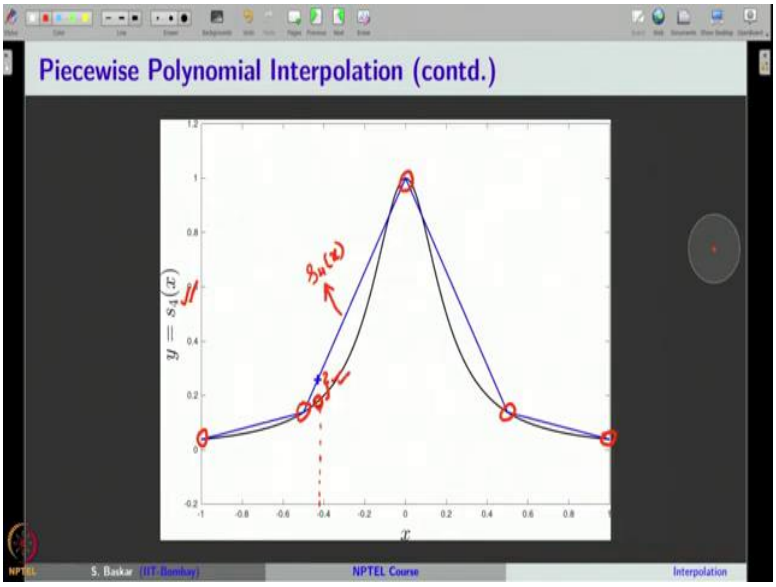


Let us see how this function looks like graphically. Here, the black solid line is the Runge function. And the blue solid line is the graph of the piecewise linear polynomial interpolating the Runge function $f(x)$. Of course, the approximation is not that good. It is understandable

again. Let us just take a point on the graph of the Runge function it is corresponding value from s_2 is given like this.

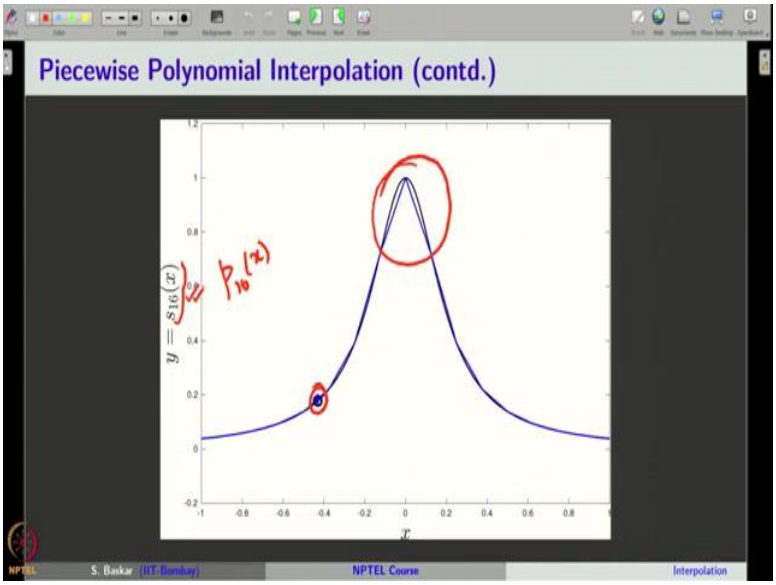
Let us go on increasing the number of nodes in the interval. Now, remember when we increase the number of nodes in the interval, we are not going to increase the degree of the polynomial. Rather we will be increasing the number of pieces in the piecewise polynomial interpolation. Let us observe how these points are located on the graph of the corresponding functions?

(Refer Slide Time: 29:29)



Let us take four pieces. It means we have five node points located here. And the corresponding piecewise linear interpolation is shown in the blue colour $s_4(x)$. You can see when compared to s_2, s_4 is approximating the values pretty well, at least at the point where we have chosen to observe the approximation. You can see this is the total error involved in s_2. The total error involved in s_4 is this.

(Refer Slide Time: 30:20)

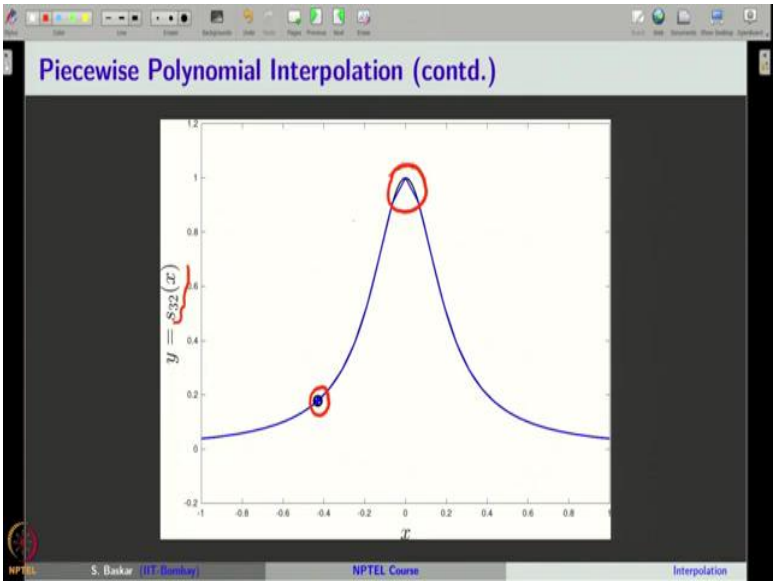


Let us now, go to increase the number of nodes in the interval. Now, I am taking s_8 it means I have taken nine nodes. And therefore, there are eight pieces of linear interpolating polynomials and the total error is now decreased drastically. And then I have gone to 16 you can see that your approximation is tending to the exact value. See this is what is mean by the approximation s_2 is tending to the exact value as you go on increasing the number of node points.

If you recall, if you would have constructed $p_{16}(x)$, you can go back and see it had a very poor approximation Chebyshev gave a better approximation and suppress the Runge phenomenon significantly. Similarly, piecewise linear polynomial interpolation is also suppressing the Runge phenomenon significantly. However, it is not able to approximate the function in certain places.

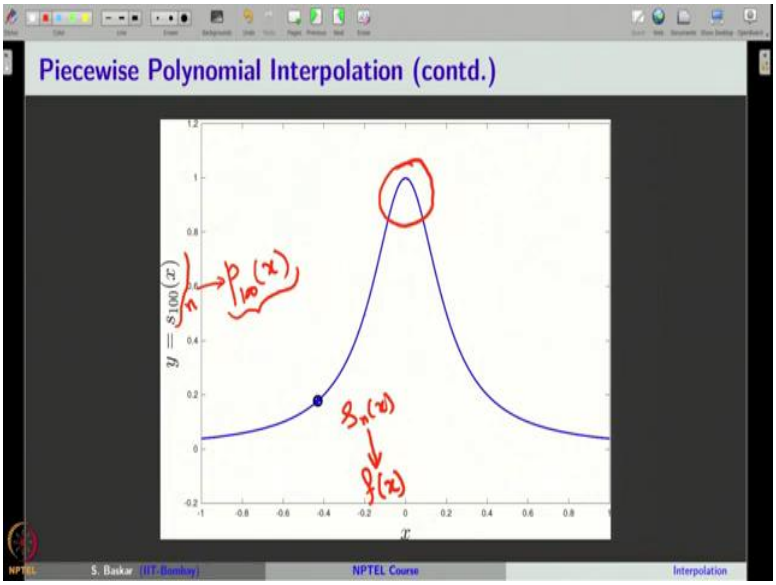
Especially, at those places where the function has higher curvature this is again understandable because we are doing piecewise linear interpolation. If you would have gone for piecewise quadratic interpolation the approximation would have been little better than this.

(Refer Slide Time: 31:55)



But let us go on with increasing the number of nodes in our data set. Now, we are having 33 node points, thereby we have 32 linear interpolation polynomials and we constructed s_{32} with that. Well, you can see that s_{32} is approximating the function pretty well and better in this region also, when compared to s_{16}.

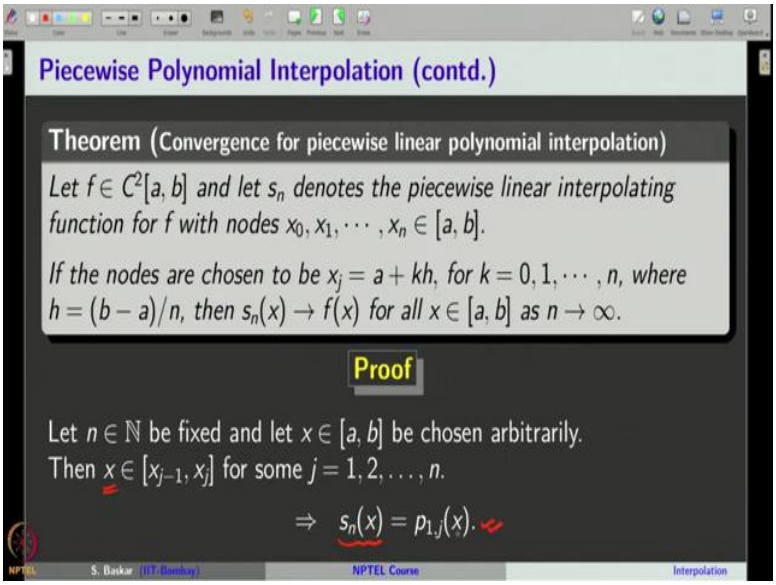
(Refer Slide Time: 32:23)



Then I have just jumped to s_{100}, just to show that as you go on increasing and equivalently, you will be calculating $p_{100}(x)$. It would have gone significantly bad in its approximation. If you would have used the polynomial interpolation, of course here I am doing with equally spaced nodes that also you should remember. I am not constructing it with Chebyshev nodes.

I am constructing this piecewise linear polynomial with equally spaced nodes. You can see that Runge phenomenon is not at all occurring and the $s_n(x)$ is pretty converging to $f(x)$ at every point. That is what is interesting here.

(Refer Slide Time: 33:15)



Let us try to state the convergence theorem for piecewise linear polynomial interpolation. For this, you need to assume that $f \in C^2[a,b]$ and let s_n denotes the piecewise linear interpolating function *f* with nodes as $x_0, x_1, \cdots, x_n$. They can be equally space nodes, no problem as far as you are working with piecewise polynomial interpolation.

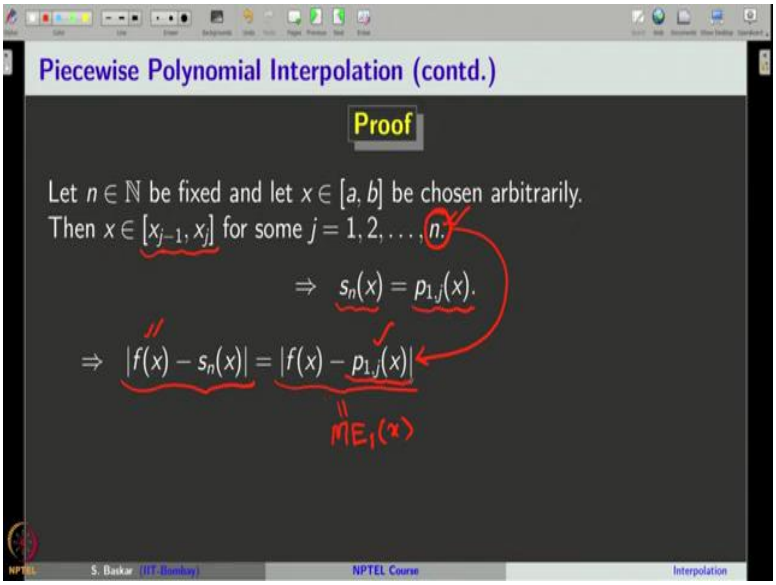
And the interpolation degree is not too high also, you should also take care of that. Here we are stating this theorem only for piecewise linear polynomial interpolation. In fact, the theorem is specifically choosing equally spaced nodes. Even then you can say that the sequence of piecewise linear polynomial $s_n(x) \to f(x)\ \forall\ x \in [a,b]$ as $n \to \infty$.

This is quite encouraging because one thing is you are working with equally spaced nodes and another one is, even you take larger and larger values of *n*, you still can have convergence that is what the theorem says. Let us quickly try to prove this theorem. The idea of proving this theorem is not something new. You have already seen it in one of the previous lectures.

Take any *n* and take any point $x \in [a,b]$ chosen arbitrarily. Now, once you choose this *n*, your partition is fixed. You have $x_0, x_1, \cdots, x_n$. These are fixed equally spaced. So, once you take a

x it will surely sit in one of the sub internals. That is sure, let us say, x is sitting in $[x_{j-1}, x_j]$ for some j. Then for that x, $s_n(x)$ will be $p_{1,j}(x)$, that is how $s_n(x)$ is defined.

(Refer Slide Time: 35:39)



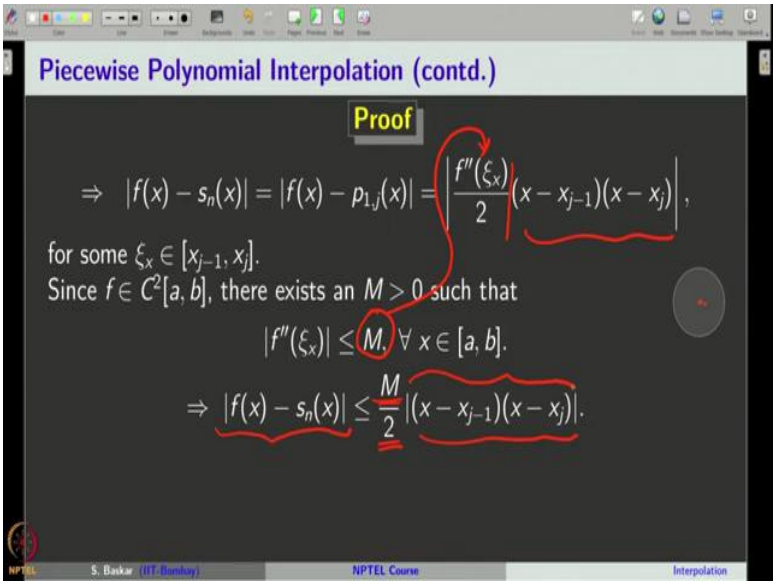
Therefore, if you take the mathematical error involved in $s_n(x)$, when compared to the original function then it can be written as $|f(x) - p_{1,j}(x)|$ because your x belongs to this interval and in this interval $s_n(x)$ is a linear polynomial. That is why this is equal to this. Once you have this you know how this looks like. If you recall, we have derived an expression for the mathematical error involved in a interpolating polynomial of degree any n.

Here in particular $n = 1$, remember, it is in one piece the degree of the polynomial is one. You should not get confused with the number of nodes attached to the degree of the polynomial. That is only in the case of polynomial interpolation. Now, we are working with piecewise polynomial interpolation, whatever may be the value of n your degree is not changed here.

(Refer Slide Time: 36:48)

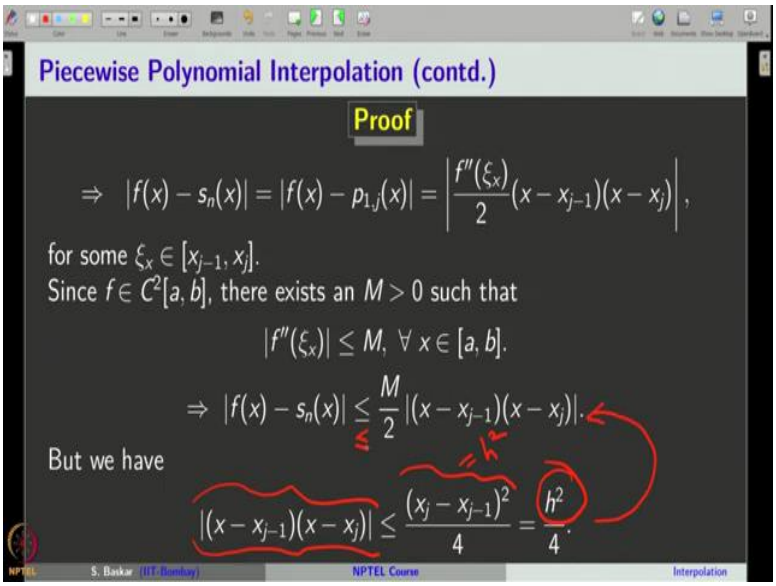
Therefore, you can use the mathematical error in the linear polynomial interpolation. If you recall, the expression is given like this where ξ is an unknown real number lying between x_{j-1} and x_j. Let us try to find an upper bound, that is, an estimate for this. Since, we have assumed that f is a C^2 function, you can bound it by some positive number M. And therefore, you can freeze this ξ by replacing $f''(\xi_x)$ by M.

(Refer Slide Time: 37:28)



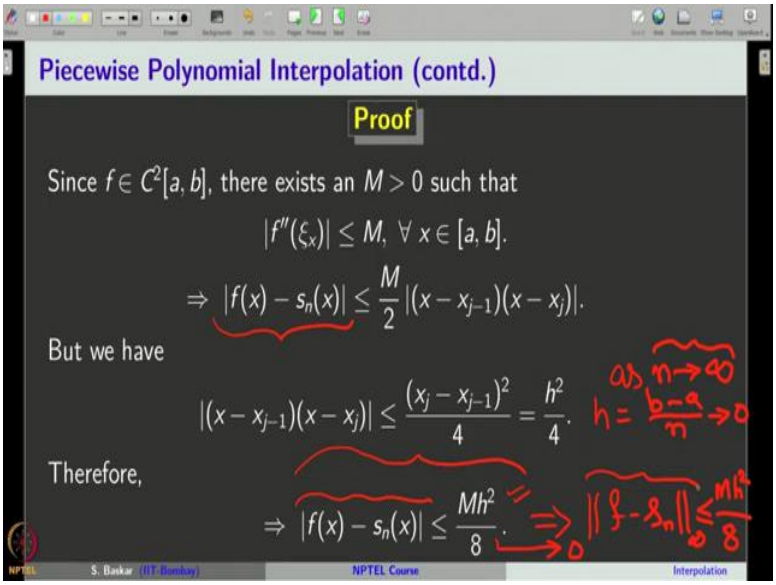
So, you can write $|f(x) - s_n(x)|$ is less than or equal to, I am just replacing this here and therefore it is $\frac{M}{2}$. And then I am having this term as it is here. Now, we will not consider this as an estimate because it involves the variable *x* here.

(Refer Slide Time: 37:56)



In order to eliminate this variable x, we will go to find the maximum of this function. Note that this is a quadratic polynomial with a modulus here. And you can clearly see that it attains it is maximum at the midpoint of the interval $[x_{j-1}, x_j]$ and it is value is given like this. Remember, we are working with equally spaced nodes and therefore this is equal to h^2. That is what I am writing here and now you can replace this here with again a less than or equal to sign.

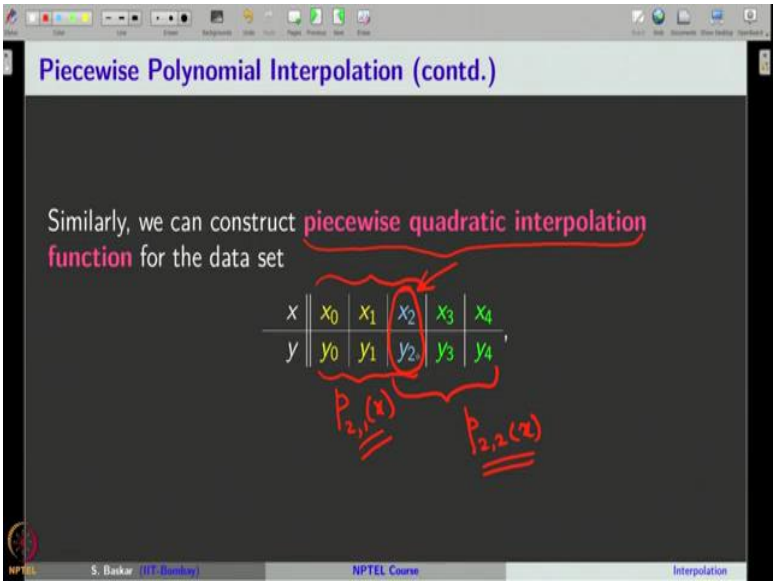
(Refer Slide Time: 38:33)



And therefore, we finally got the estimate of your mathematical error involved in the piecewise linear polynomial interpolation. And that estimate is given by $\frac{Mh^2}{8}$, remember this holds for any n and for any x in the interval $[a, b]$. Therefore, in fact you can say that this is nothing but $||f - s_n||_\infty \leq \frac{Mh^2}{8}$. Now, you can see, as $n \to \infty$, what happens?

Well, as $n \to \infty$, since $h = \frac{b-a}{n}$, you can see that $h \to 0$ as $n \to \infty$. That means, the right hand side goes to 0, as $n \to \infty$. That means the left hand side will also go to 0 rather uniformly. And that proves that $s_n \to f$ and this completes the proof of the theorem.

(Refer Slide Time: 39:44)



In fact, you can extend this idea of constructing piecewise linear interpolating polynomial to piecewise quadratic interpolating polynomial, cubic interpolating polynomial and so on. In

fact, you can also extend the idea of proving convergence theorem for these cases also. I will just remark that if you are going for a quadratic polynomial interpolation. Then you at least need three points in order to construct a quadratic polynomial.

Why? Because to construct a quadratic interpolating polynomial you need three points. Therefore, when you have a data set and you want to construct a piecewise quadratic polynomial function, you have to group the nodes in this way. That is, you need three points to construct one piece of the quadratic interpolating polynomial. Therefore, this will be $p_{2,1}(x)$ and for the next piece you have to take these nodes and construct $p_{2,2}(x)$.

Remember this will be common both for this as well as for this. This is also the case with linear polynomial, you can go and observe that. And you will have the continuity of the piecewise polynomial because of this one. Otherwise, it will have a gap between two nodes we should not have this. So, you should carefully choose the nodes to construct the piecewise interpolating polynomials.

Similarly, for cubic polynomial you have to group the nodes with four nodes in one piece. And the boundary nodes should overlap for both the pieces on the either side of that point. So, this is how you can construct piecewise interpolating polynomials of any degree. With this we will end this lecture and thank you for your attention.