Finite Element Method and Computational Structural Dynamics

Prof. Manish Shrikhande
Department of Earthquake Engineering
Indian Institute of Technology, Roorkee

Lecture - 06
Polynomial Interpolation and Numerical Quadrature - III

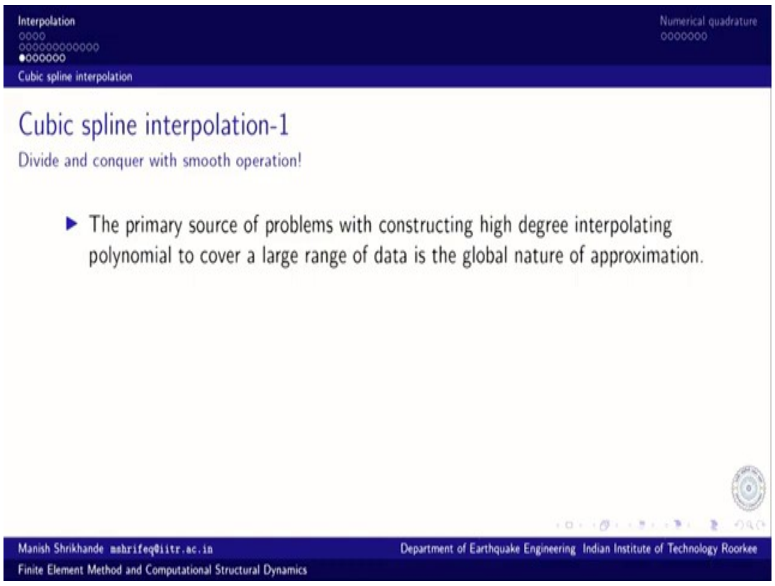
Hello. So, in our last lecture we discussed about a different interpolation families starting with Lagrange interpolation, simple polynomial interpolation based on function values at nodes of interpolation, and then alternate form of polynomial interpolation and that is in the Newton form.

And then also we followed it and discussed the optimal way of computing polynomial that is Horner's algorithm that is based on minimizing the floating-point operations while evaluating the polynomial at any value of the independent variable.

Then we discussed what is the importance of sampling points; that is if we are not careful in choosing sampling points or nodes of interpolation, then even if the function values of the interpolation function may exactly agree with the specified function values at the nodes, but in between nodes there can be very wild fluctuations.

And with suitable roots of Chebyshev polynomials we saw that they provide a suitable or very nearly optimal location of nodes and the error, or the oscillation between the nodes of interpolating polynomial is very much under control

(Refer Slide Time: 02:02)



Interpolation

Numerical quadrature

Cubic spline interpolation

Cubic spline interpolation-1

Divide and conquer with smooth operation!

- The primary source of problems with constructing high degree interpolating polynomial to cover a large range of data is the global nature of approximation.

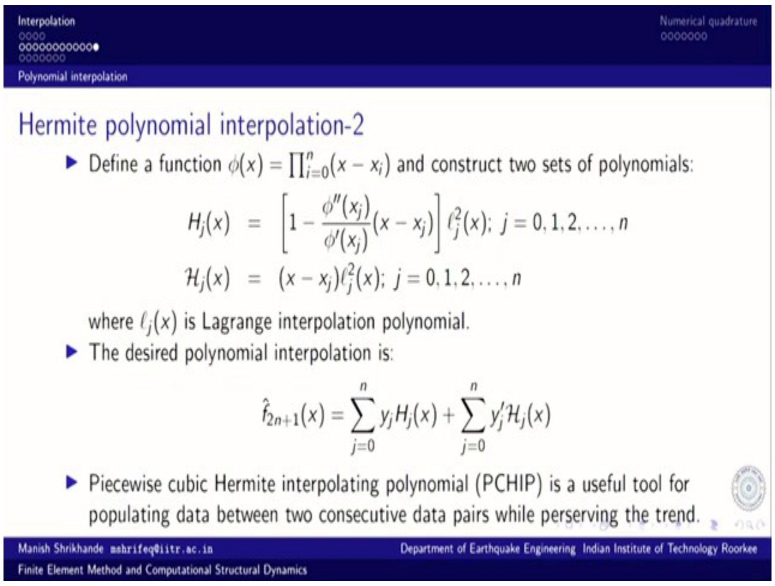
Manish Shrikhande mshrifeq@iitr.ac.in Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

And we can consider constructing a very good quality of approximation by using roots of Chebyshev polynomials as the nodes of interpolation. Then another way to constrain the errors or constrain the behaviour of interpolating polynomial is by adding more information during interpolation and that is by considering not only function values, but also the derivative.

And that leads to what we call as Hermite polynomial interpolation. And with just two nodes we can actually pass a cubic interpolation polynomial based on two conditions at each nodes.

(Refer Slide Time: 02:47)



Interpolation | Numerical quadrature

Polynomial interpolation

Hermite polynomial interpolation-2

- Define a function $\phi(x) = \prod_{i=0}^{n}(x - x_i)$ and construct two sets of polynomials:

$$H_j(x) = \left[1 - \frac{\phi''(x_j)}{\phi'(x_j)}(x - x_j)\right] \ell_j^2(x);\ j = 0, 1, 2, \dots, n$$

$$\mathcal{H}_j(x) = (x - x_j)\ell_j^2(x);\ j = 0, 1, 2, \dots, n$$

where $\ell_j(x)$ is Lagrange interpolation polynomial.

- The desired polynomial interpolation is:

$$\hat{f}_{2n+1}(x) = \sum_{j=0}^{n} y_j H_j(x) + \sum_{j=0}^{n} y_j' \mathcal{H}_j(x)$$

- Piecewise cubic Hermite interpolating polynomial (PCHIP) is a useful tool for populating data between two consecutive data pairs while perserving the trend.

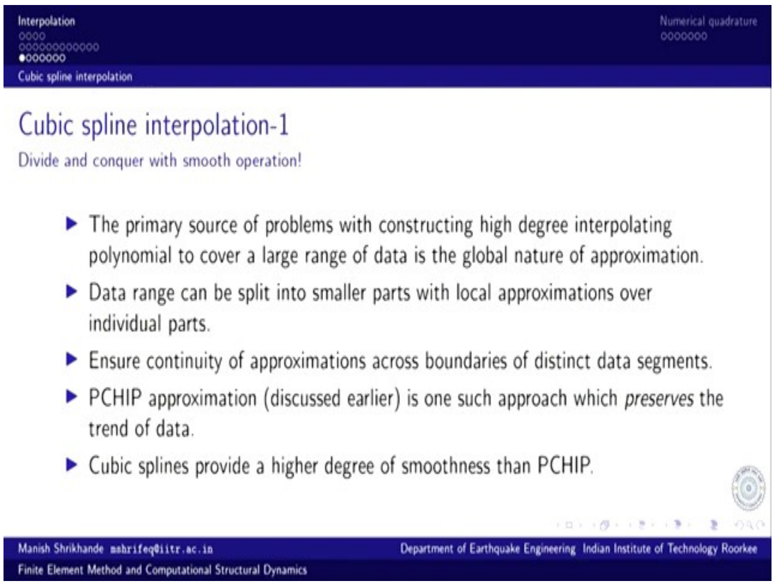
Manish Shrikhande mshrifeq@iitr.ac.in | Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

So, total four conditions are available for evaluating unknown coefficients. So, a cubic polynomial can be interpolated, and that is what we call as piecewise cubic Hermite interpolation polynomial.

And this is a very useful tool; piecewise cubic Hermite interpolating polynomial PCHIP that is used for interpolating data between two points, and with the guarantee that the trend - whatever may be the trend of data - ascending or descending, will be preserved. So, we are not disturbing the trend of data when we interpolate using PCHIP.

(Refer Slide Time: 03:37)



Interpolation | Numerical quadrature

Cubic spline interpolation

Cubic spline interpolation-1

Divide and conquer with smooth operation!

- The primary source of problems with constructing high degree interpolating polynomial to cover a large range of data is the global nature of approximation.
- Data range can be split into smaller parts with local approximations over individual parts.
- Ensure continuity of approximations across boundaries of distinct data segments.
- PCHIP approximation (discussed earlier) is one such approach which *preserves* the trend of data.
- Cubic splines provide a higher degree of smoothness than PCHIP.

Manish Shrikhande mshrifeq@iitr.ac.in | Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

Now, another kind of interpolation that is often used is spline interpolations. And as we saw during the approximation of Runge’s function by polynomial interpolation. The primary source of problem was that we were trying to develop the approximation over the complete range at one go.

So, the kind of global approximation that we were trying to control and develop was a very ambitious goal and we are not in control of all the points in the domain of the range with the same degree of control, and it might be better to divide the range and then construct approximation over smaller ranges

So, that is what we call as divide and conquer. So, PCHIP – Piecewise Cubic Hermite Interpolating Polynomial -, just two nodes at a time and that will allow us to fit a cubic interpolating polynomial.

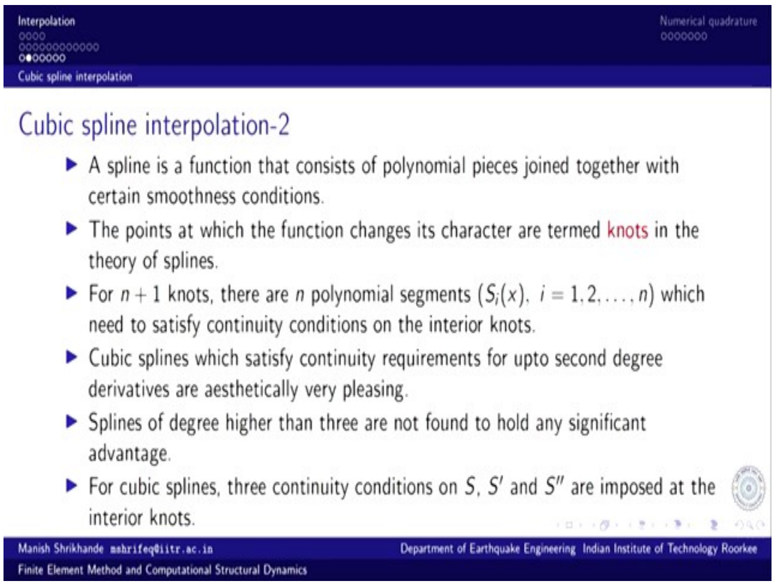
That can also be considered as part of this strategy - divide and conquer, but cubic spline interpolation splines have higher degree of smoothness than PCHIP interpolation. In PCHIP we only considered the first order derivatives to be included in the interpolation information.

In spline, we use higher order derivatives also. So, the function that we generate is much smoother than the PCHIP interpolation. So, as we were saying, this is based on the primary source of errors in approximation; which is the attempt to construct a single approximation over the whole range of the variables. If we can divide the data range into smaller parts and construct approximation over individual parts, we can actually get a much better approximation.

And that also needs to adhere to the continuity between adjacent data ranges. So, when we divide the entire domain into separate smaller ranges, we also need to take care how the variation happens or how the information exchange happens between the boundaries across these smaller ranges. So, continuity has to be ensured to make sure that the function looks continuous and there is no jump in the approximation across the boundaries.

So, as I said PCHIP approximation is one such approach which preserves the trend of data and cubic spline we can actually generate higher degree of smoothness than PCHIP, but at the cost of losing control over the trend. So, while the function that we approximate using cube splines would be much smoother much more pleasing to visual appeal, but there is no guarantee that trend of the data will be preserved.

(Refer Slide Time: 07:52)


Interpolation

Numerical quadrature

Cubic spline interpolation

Cubic spline interpolation-2

- A spline is a function that consists of polynomial pieces joined together with certain smoothness conditions.
- The points at which the function changes its character are termed knots in the theory of splines.
- For $n+1$ knots, there are n polynomial segments ($S_i(x)$, $i = 1, 2, \ldots, n$) which need to satisfy continuity conditions on the interior knots.
- Cubic splines which satisfy continuity requirements for upto second degree derivatives are aesthetically very pleasing.
- Splines of degree higher than three are not found to hold any significant advantage.
- For cubic splines, three continuity conditions on S, S' and S'' are imposed at the interior knots.

Manish Shrikhande mshrifeq@iitr.ac.in Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

So, theoretically a spline is a function that consists of polynomial pieces joined together with a certain smoothness conditions. The points at which the function changes its character are termed knots in the theory of splines. So, instead of nodes we call those points between which we are defining one particular segment as a pair of knots.

So, for (n + 1) knots there are n number of polynomial segments. So, between two points we will have one polynomial. So, if there are (n + 1) knots then there are going to be n number of segments and these we call as spline segments. So, denoting them by S_i as a function of X_i ranging from 1 to n. So, these are n number of spline segments for which we need to satisfy the continuity conditions on the interior knots.

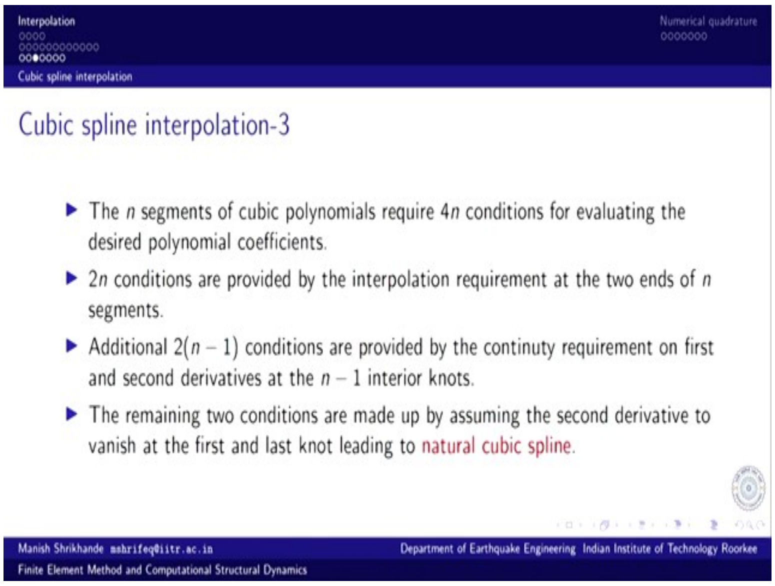
There are total (n + 1) knots - two extreme knots denoting the boundaries and (n – 1) number of knots which have to satisfy the continuity conditions. So, we can have splines of various degrees, but cubic splines are most often used. They are the most popular spline functions that are used. So, cubic splines which satisfy continuity requirements for up to second degree derivative are aesthetically very pleasing.

So, that explains why cubic splines are most popular spline functions that are used in function approximation or even graphs. I mean if we just rely on some plotting software's and give the data and leave it at the mercy of the software to plot a best fit curve, then the chances are it will be a cubic spline fit.

For splines of degree higher than 3, in addition to having more computation we do not generally find any significant advantage in it as far as the interpolation or information regeneration goal is concerned.

So, for cubic splines we impose three continuity conditions on the spline segments - the function value itself, the first derivative and the second derivative. These are imposed at the interior knots and that gives us sufficient constraints and continuity to ensure the smoothness requirement.

(Refer Slide Time: 11:03)



Interpolation | Numerical quadrature

Cubic spline interpolation

Cubic spline interpolation-3

- The n segments of cubic polynomials require $4n$ conditions for evaluating the desired polynomial coefficients.
- $2n$ conditions are provided by the interpolation requirement at the two ends of n segments.
- Additional $2(n-1)$ conditions are provided by the continuty requirement on first and second derivatives at the $n-1$ interior knots.
- The remaining two conditions are made up by assuming the second derivative to vanish at the first and last knot leading to natural cubic spline.

Manish Shrikhande mshrifeq@iitr.ac.in | Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

Total 4n conditions are required to uniquely define n segments of cubic polynomials. So, where do we get these 4n conditions? Let us count our conditions that we have - 2n conditions are provided by the interpolation requirements at the two ends of n segments.

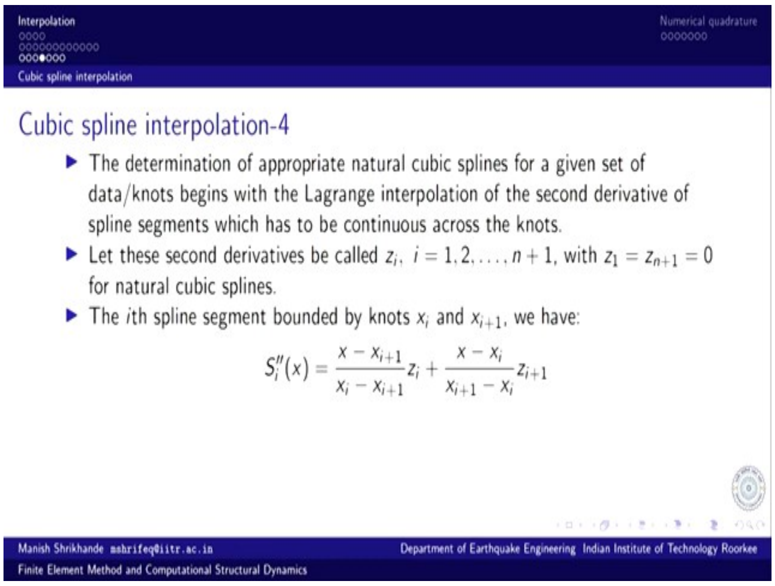
Additional two constraints are available at each of the interior node derivatives that is at the interior knots.

So, 2(n - 1) conditions are provided by ensuring the continuity of first and second derivatives at the (n - 1) interior knots. So, the first derivative of the spline functions across the boundary has to agree, the second derivative of these spline function segments have to agree across the boundaries.

So, adjacent spline segments have to be consistent. Not just with respect to the function values, they also have to ensure that the continuity of first and second derivatives are maintained across the knots. That provides us (4n – 2) conditions. We are still left with two conditions to be provided and the remaining two conditions are made up by assumptions.

We assume second derivative to vanish at the first and last knot. So, once we make this assumption then the spline approximation that we get is referred to as natural cubic spline. But if we have additional information about the second derivative at the first and last knot, we can specify those values and that would be specific to the particular case. So, it will still be cubic spline, but it will not be a natural cubic spline.

(Refer Slide Time: 14:03)



Interpolation

Numerical quadrature

Cubic spline interpolation

Cubic spline interpolation-4

- The determination of appropriate natural cubic splines for a given set of data/knots begins with the Lagrange interpolation of the second derivative of spline segments which has to be continuous across the knots.
- Let these second derivatives be called z_i, $i = 1, 2, \ldots, n+1$, with $z_1 = z_{n+1} = 0$ for natural cubic splines.
- The ith spline segment bounded by knots x_i and x_{i+1}, we have:

$$S_i''(x) = \frac{x - x_{i+1}}{x_i - x_{i+1}} z_i + \frac{x - x_i}{x_{i+1} - x_i} z_{i+1}$$

Manish Shrikhande mshrifeq@iitr.ac.in

Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

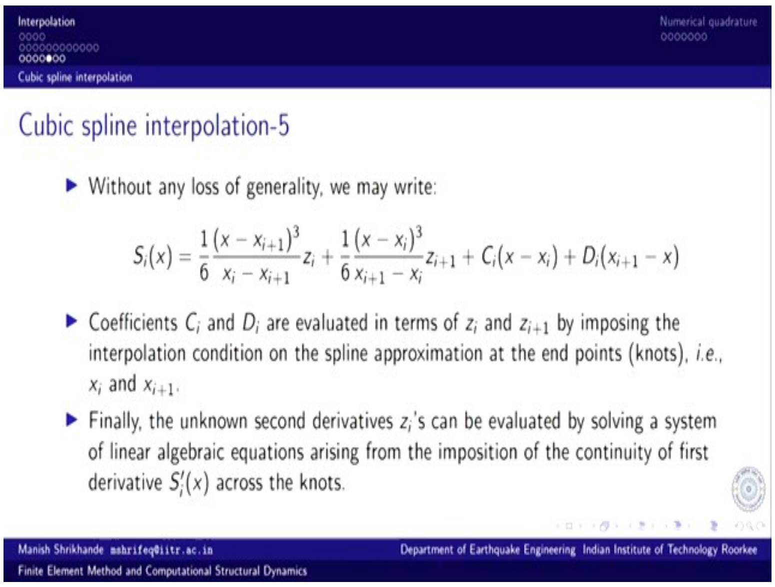
So, determination of appropriate natural cubic spline for a given set of data or knots it begins with Lagrange interpolation of the second derivative. So, second derivative we want second derivative to be continuous. So, obviously, then we define Lagrange interpolation that will ensure that continuity is maintained.

So, Lagrange interpolation of the second derivative of the spline function is defined. If we call these second derivatives whatever the values may be as some variable unknown Z. Then there are (n + 1) number of Z. So, the those are the second derivatives with the condition constrained that first and last second derivatives are 0, so that is the condition for natural cubic spline.

Or if we do not want if we have additional information of second derivative about at the starting and end points, then those values can be substituted at these boundary conditions. So, for each segment we can construct the interpolation between second derivative as a Lagrange interpolation and this is what I mean by “any segment will be bounded by two values of the second derivative and we can construct piece wise Lagrange interpolation over each segment”.

So, once we have this second derivative as a linear function then we can integrate it and derive the spline functions with some unknown coefficient C and D those are the unknown coefficient. So, we have this cubic function. So, if we integrate it then we will get a cubic function and that is what we arrive at by integrating that Lagrange interpolation.

(Refer Slide Time: 16:20)



Interpolation — Cubic spline interpolation — Numerical quadrature

Cubic spline interpolation-5

- Without any loss of generality, we may write:

$$S_i(x) = \frac{1}{6}\frac{(x - x_{i+1})^3}{x_i - x_{i+1}} z_i + \frac{1}{6}\frac{(x - x_i)^3}{x_{i+1} - x_i} z_{i+1} + C_i(x - x_i) + D_i(x_{i+1} - x)$$

- Coefficients C_i and D_i are evaluated in terms of z_i and z_{i+1} by imposing the interpolation condition on the spline approximation at the end points (knots), *i.e.*, x_i and x_{i+1}.
- Finally, the unknown second derivatives z_i's can be evaluated by solving a system of linear algebraic equations arising from the imposition of the continuity of first derivative $S_i'(x)$ across the knots.

Manish Shrikhande mshrifeq@iitr.ac.in — Department of Earthquake Engineering Indian Institute of Technology Roorkee — Finite Element Method and Computational Structural Dynamics

Now this coefficient C_i and D_i for each segment each of these i segments can be derived by imposing the constraints. So, that is just a rearrangement of terms for ease of computation. So, these C_i and D_i are evaluated in terms of second derivatives and then we impose the condition on interpolation condition on this spline approximation at the end points of this spline and that gives us a set of simultaneous equations to solve.

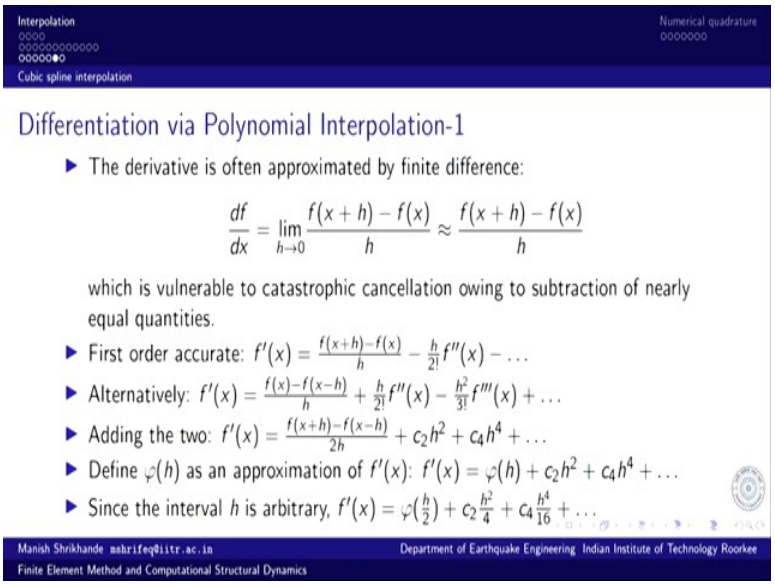
And then once we solve the simultaneous equation that yields a set of values which are Z - the second derivatives. And once we have the second derivatives available with us then we can substitute it back in this approximation and all splines can be derived over each segment.

So, you can have a look at this spline interpolation, but the whole idea of this spline interpolation that I wanted to discuss is, you can get the derivation and you can look at the numbers and work out the numbers and plot the spline, the Runge's function is a good example, to try out spline interpolation and see what difference it makes to the approximation. And its almost indistinguishable at least graphically.

And for that matter even PCHIP will do a fairly decent job of approximating Runge's function, and the entire process brings out the emphasis on the basic philosophy of approximating a function which is to divide and conquer. So, instead of trying to approximate over the whole range at one go it is always better to divide the domain into smaller sub domains and then construct approximation over each of these smaller sub domains with certain constraints that will ensure certain degree of continuity across the adjacent sub domains, and as a whole we can develop a very good approximation that would suffice for all practical purposes, for engineering analysis.

And while we are at the topic of interpolation, we will also try to wrap it up by using by discussing one of the interesting applications of interpolation theory in a very different context.

(Refer Slide Time: 19:20)



Interpolation

Numerical quadrature

Cubic spline interpolation

Differentiation via Polynomial Interpolation-1

- The derivative is often approximated by finite difference:

$$\frac{df}{dx} = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \approx \frac{f(x+h)-f(x)}{h}$$

  which is vulnerable to catastrophic cancellation owing to subtraction of nearly equal quantities.
- First order accurate: $f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{h}{2!}f''(x) - \ldots$
- Alternatively: $f'(x) = \frac{f(x)-f(x-h)}{h} + \frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x) + \ldots$
- Adding the two: $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + c_2h^2 + c_4h^4 + \ldots$
- Define $\varphi(h)$ as an approximation of $f'(x)$: $f'(x) = \varphi(h) + c_2h^2 + c_4h^4 + \ldots$
- Since the interval h is arbitrary, $f'(x) = \varphi(\frac{h}{2}) + c_2\frac{h^2}{4} + c_4\frac{h^4}{16} + \ldots$

Manish Shrikhande mshrifeq@iitr.ac.in

Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

So, we can actually find a very good approximation for function derivative by using polynomial interpolation. And many times we need differentiation. For example, if we are using any optimization process then gradient is a very often requirement. We need to find out gradient of the optimization function and that is a very often required process.

And the convergence of many processes depends on the quality of derivative computation or how accurately we can compute the gradient, because that is what defines the search path for optimal solution and several other instances where gradient information may be required.

So, typically a gradient of any function is often approximated by finite difference approximation. The theoretical definition of the gradient is based on the limiting operation. And in finite difference, throughout the limiting operation we just take the finite difference. So, the function values at two different points divided by the difference between those two points.

The separation between two points is “h” as seen here. The points that we are sampling the function are separated by some distance “h” and we are taking the limiting operation as “h” tends to 0. So, in practice it has to be a small value and then we can consider this finite difference approximation to be a good approximation of the derivative.

Now, while we were discussing about floating point operation, we also saw that the difference between two numbers which are nearly equal, always leads to a catastrophic cancellation and severe loss of precision and this is what happens in finite difference approximation for computation of derivatives.

So, f(x+h) is not going to be very different from f(x). So, these two numbers are likely to be very close to each other, if “h” is not very large (which should be small). And “h” should be small if we have to approximate the limiting operation to any reasonable degree.

So, now we have devil's alternative. If I use too large value of “h” just to ensure that function values are different then I am not probably getting a good idea about the local derivative. And if I reduce “h” to a small enough number then the chances are that I might incur very heavy error in computation of derivative because of catastrophic cancellation and loss of precision.

So, what to do? This entire process is a first order accurate process and that is based on the Taylor series approximation. If you can look at this finite difference and expand function in the neighborhood of (x+h), then from the Taylor series expansion we can get the order of error. So, the error term is second order. This h^2 error term is first order, because this h^2 term gets divided by “h” and that is how it is proportional to first degree of “h”.

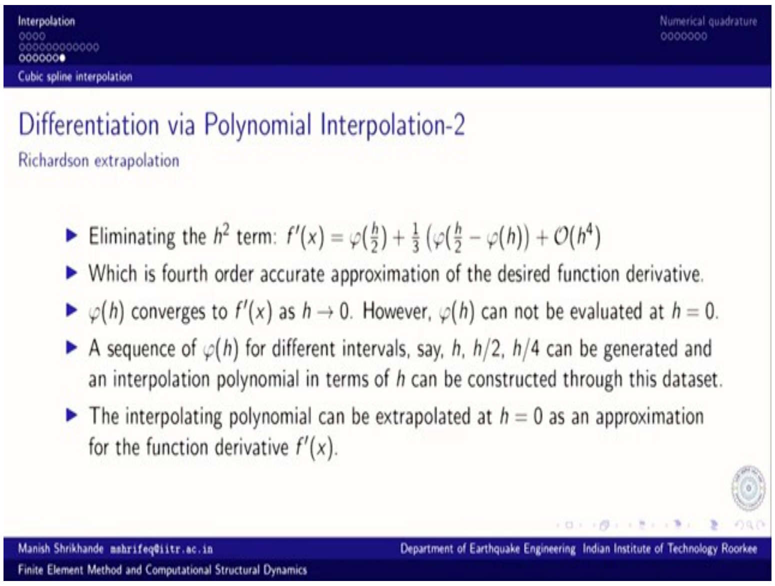
So, that is a first order accurate, the error term is proportional to the first power of the separation “h”, and we can actually work it out by backward difference; one is forward difference and another expression – the second expression - is using backward difference. No change at all but this allows us is to use this to cancel this error term. And if we add these two approximations of finite difference then we can get the difference between double twice the interval and interestingly the error now becomes second order.

So, if I change the finite difference approximation to twice the interval, then the error in the approximation of finite derivative is of second order. So, from the first order accuracy I go to second order accuracy. So, I can define a function which is an approximation. So, this particular difference operator I use is the basic function that is approximating the finite gradient and then there are error terms.

So, quadratic error and then there is fourth degree error. All odd terms will get canceled out during the process of these additions. Odd powers of error term will get canceled out. So, since “h” is an arbitrary interval I can choose anything. So, if I choose “h”, to be half the interval then this error term is also going to reduce. The second term becomes $h^2/4$, third term becomes $h^4/16$.

I can again eliminate the h^2 term, adding the two terms and looking at this approximation.

(Refer Slide Time: 26:49)



Interpolation

Numerical quadrature

Cubic spline interpolation

Differentiation via Polynomial Interpolation-2

Richardson extrapolation

- Eliminating the h^2 term: $f'(x) = \varphi(\frac{h}{2}) + \frac{1}{3}(\varphi(\frac{h}{2} - \varphi(h)) + \mathcal{O}(h^4)$
- Which is fourth order accurate approximation of the desired function derivative.
- $\varphi(h)$ converges to $f'(x)$ as $h \to 0$. However, $\varphi(h)$ can not be evaluated at $h = 0$.
- A sequence of $\varphi(h)$ for different intervals, say, h, $h/2$, $h/4$ can be generated and an interpolation polynomial in terms of h can be constructed through this dataset.
- The interpolating polynomial can be extrapolated at $h = 0$ as an approximation for the function derivative $f'(x)$.

Manish Shrikhande mshrifeq@iitr.ac.in

Department of Earthquake Engineering Indian Institute of Technology Roorkee

Finite Element Method and Computational Structural Dynamics

So, I can eliminate the terms from these two expressions and find out what is the error term. So, the function value that I have, one for the sampling of “h” and another approximation for the sampling of h/2, and then the function derivative has an accuracy of fourth order.

So, considering that initially we were having first order accuracy finite difference. Now we are looking at fourth order accuracy of the gradient function and without incurring catastrophic cancellation. And this function ϕ (h) converges to the desired gradient as “h” tends to 0.

However, we cannot really compute f(h) at h=0. What we can do is, we can construct interpolation function, as an approximation for ϕ (h) and that can be evaluated for h=0. So, a sequence of ϕ (h) for different intervals can be generated and their interpolation polynomial in terms of “h” can be constructed through this data set.

Once we have this interpolation polynomial then we can find the appropriate function derivative at h=0 as an approximation and that would be a very good approximation for the function derivative for further use which will hopefully not suffer from catastrophic cancellation and loss of precision and the computations would be more stable and more robust.

This ends our discussion on polynomial interpolation and provides the basic background for construction of finite element approximation. I will come back to this basic thing that we discussed today - divide and conquer - that forms the basis of finite element approximation.

We will keep going back on this thing again and again, divide the domain into sub domains and construct approximation over sub domains while ensuring continuity across the boundaries of sub domains. Another issue that we need to discuss that will be useful and extensively used in finite element approximation or construction or implementation of finite element method is numerical integration.

Finite element method would not have been as successful as it is today, had it not been for digital computers. All computer computations have to be cast on appropriately for application on a digital computer. And integration is an integral part of that and that is what we will discuss in our next lecture, numerical integration

Thank you.