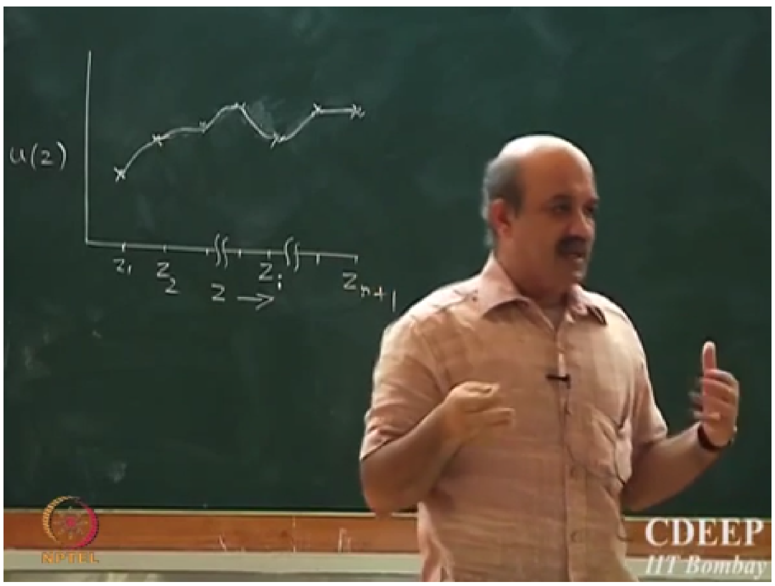
Advanced Numerical Analysis
Prof. Sachin Patwardhan
Department of Chemical Engineering
Indian Institute of Technology – Bombay

Lecture – 15
Polynomial and Function Interpolations, Orthogonal Collocations Method for Solving ODE-BVPs

So in our last lecture, we were looking at interpolation polynomial. We also looked at cubic spline interpolation. So, I talked about piecewise polynomial interpolation towards the end, but let me explain the few things before I move on about piecewise polynomial interpolation.

(Refer Slide Time: 00:40)



We have this function U, Z and we know values of this function at different points and then we wanted to fit a polynomial that passes through each of these points, okay. So, one possibility was to develop a very high order polynomial which passes through each of these points. So, that would be a continuous function passing through each of these points. There are other options. As I said finding out coefficients of a very high order polynomial higher than 3 or 4 becomes ill-conditioned problem, it is difficult to find the coefficients,

So, we resort to what is called as piecewise polynomial approximation and one particular type of approximation I talked yesterday. So, if I want to develop a piecewise polynomial approximation, what is the simplest piecewise polynomial approximation, well not cubic. The simplest one will be linear line, you know. The simplest one would be just a line. I could develop a line that connects these 2 points, then another line that connects these 2 point.

What is the problem with this? This is fine, in some cases this works, this is useful. I am not saying this is not useful, but this piecewise linear approximation is not differentiable at the boundary points. It is not differentiable. So, it is continuous but it is not differentiable, okay. That is we want to have higher order approximation. We could then think approximating this using second order that is quadratic equation.

So, each one of them instead of a line can be a quadratic equation, okay. So, this might be and so on, okay. Now, if you fitting in general, you could fit (()) (03:36) order polynomial, say quadratic cubic 4th order. We normally stop at cubic and among the piecewise polynomial approximations, the approximation for which we match derivatives up to K-1. Suppose you are fitting a K third order polynomial between each segment and you match the derivatives up to K-1 order, then that is called the spline approximation, okay.

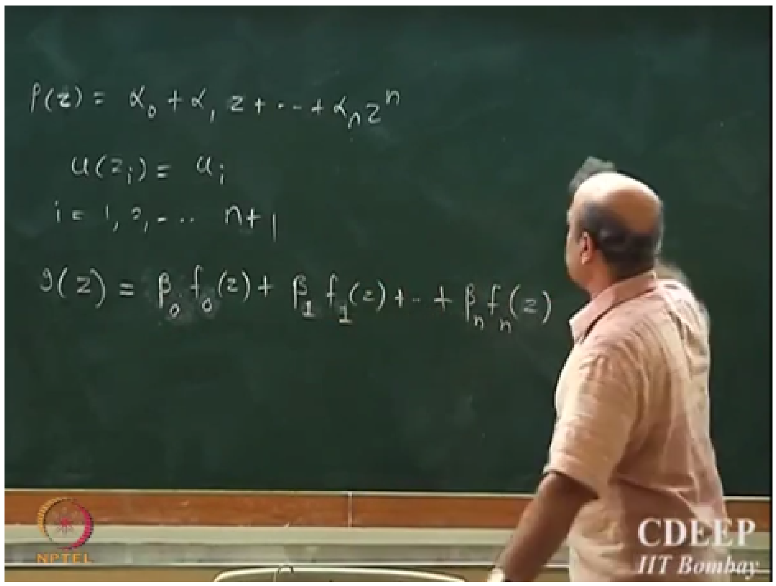
So, cubic spline, when I say cubic spline that is because they are matching first order and second order derivatives. Polynomial order is 3, first and second order derivatives are matched for smoothness purpose, so that is why it is cubic spline. Normally, we use cubic spline because beyond that it is not worth fitting high order polynomial beyond cubic. Cubic many times suffices for most of the applications.

So, you can very well develop piecewise linear approximation, piecewise quadratic approximation, piecewise cubic approximation, okay. So, depending upon what level of smoothness you need in your application. In some cases, differentiability is not so important. You can have piecewise linear, okay. In some cases, differentiability is important. So, you should have high order polynomial and match derivatives and so on, so this is a small point.

The next point that I want to make here is that when I am developing this approximation is it necessary that I only do a polynomial approximation. It is not necessary that I will only do polynomial approximation. I can do functional approximation, okay. So, in general, we are not going to use when we will do boundary value problem discretization or when I develop this orthogonal collocation method, I am not going to use function approximation.

This is just a side note that I am using the word approximation. Function interpolation, we will be approximation in different context. Just like I can do polynomial interpolation, I can do function interpolation.

(Refer Slide Time: 06:01)



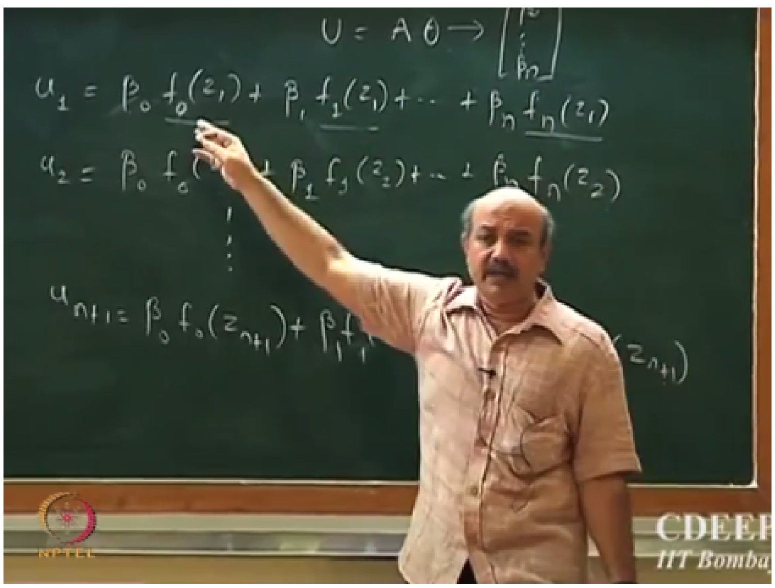
So, in general, I have this you know, I will just slightly change the notation. You have this U, Z here and till now we were looking at polynomial approximation which means we said that Pz is alpha 0+alpha 1, okay and then was the initial Lagrange interpolation and then we know these values of UZi=Ui, we know these values at i=1, 2, up to n+1. We know values of ui and then we wrote n+1 equations and n+1 (()) (06:58) and then we had simple way of finding out alpha 0, alpha 1 by matrix inversion, this we have discussed earlier.

It is not necessary that I approximate only using polynomials, no approximate. Approximation will come when you approximate boundary value problem. Right now, interpolation is something you try to construct a function that passes through every point, okay. So, this is a polynomial function that passes every point here, okay. Now, I could also develop say another function which is interpolating function and I would call this as some beta 1 f1z+beta 2 f2z. Let us keep the notation similar.

So, let us call this beta 0 f0, beta 1, f1,…up to beta n. In general, I need not represent polynomial using simple polynomial functions. I could have more complex functions appearing here. So, this is called function interpolation. Each one of them is a function of z, okay. For example, I could use some of the standard polynomial say Legendre polynomial here okay or I could use shifted Legendre polynomial, depending upon the context, you could use different kinds of polynomial.

I could use here sin and cos, okay. So, it is not necessary that interpolation should be only through polynomial. Interpolation can be through functions. This will be called interpolating function, okay.

(Refer Slide Time: 09:24)



How do you estimate beta 0 to beta n. The same thing, you know you write this equation U1=beta0 f1 Z1+beta1 and then U2=beta 0 F0, Z2+beta1 F1Z2 and likewise you can write this for all the points and I have U n+1=beta 0 F0Zn+1+beta1 F1+.

So, I have n+1 equations in n+1 unknowns. What are the unknowns here? beta0, beta1 up to beta n, these are the unknowns. These functions here F0, F1 interpolating functions, I have chosen them. I know these functions. Like sin, cos or whatever you chose. I have chosen these functions. So, I can substitute values of z and find out the value of that particular function. So, here this would be know, this would be know because I have chosen the function, I can evaluate the function at a particular point.

Then, write this into the standard form because this will give rise to U=A matrix*theta where theta is nothing but beta 0 to beta n. This vector theta is beta0 to beta n. What will A matrix have? all these f0z1, f1z1 all values of these. This A matrix is known and then you can find out. You can find the interpolation function or this is called function interpolation. So, this is a special case of function interpolation, okay.

You have chosen a special function which is simple polynomial but that is not the only way to construct interpolating function, okay. You can do in general a function interpolation. Well, interpolation is if you look at textbooks on applied mathematics, you will find much more material interpolation. Right now, my interest is in discretizing a problem. So, I am going to restrict myself to whatever I need for problem discretization.

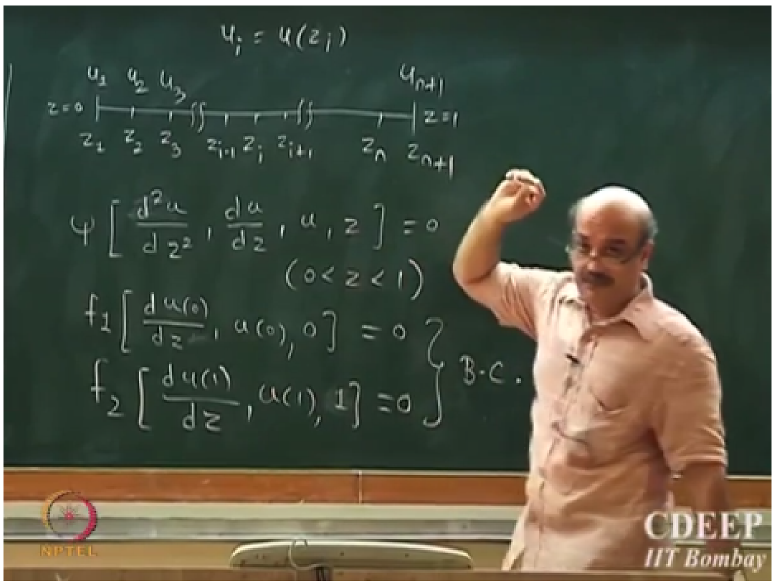
So, my aim is now to convert a boundary value problem or a partial differential equation into a set of either algebraic equations which have to be solved simultaneously or in some cases it could be differential equation that need to be solved simultaneously. We have seen several cases, right. What is going to be different here is how we approximate the derivatives. What was the key in the earlier case? Where we have used Taylor series method?

The key was approximation of first order and second order derivatives, okay. So, interpolating function or interpolating polynomial is going to be used for approximation of local derivatives, okay. So, the word approximation comes when you start discretizing not when you construct the interpolating polynomial. So, the trick is going to be the same. What did we do? when we developed finite difference method?

We started with Taylor series, we looked at a point, right and we said local derivative of this particular function can be approximated using Taylor series expansion and then we had this forward different, backward difference formula for approximating the local derivative at ith grid point. We divided the entire domain into number of grid points and then at a particular grid point, ith grid point, we had a way of constructing local approximation of the first derivative and second derivative, okay. Same thing is going to be done here.

I am going to use this interpolating polynomial to construct approximation of local derivatives at the grid points, okay. Now, just look carefully at the development because this is something which is new for most of you and somewhat different from finite difference. Finite difference in some sense is covered when you undergraduate in many cases. Many of you would have visited finite difference but orthogonal collocation probably is not part of typical undergraduate program.

(Refer Slide Time: 15:27)



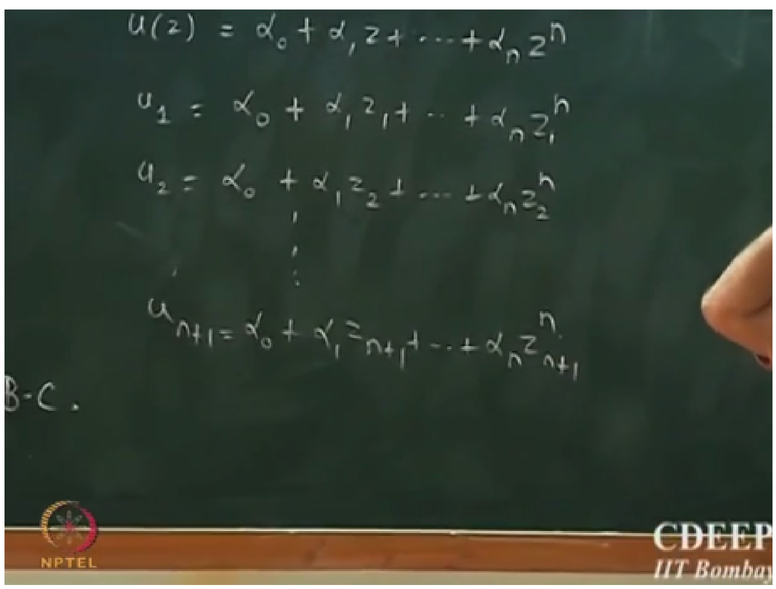
Now, let us start looking at how we develop this orthogonal collocation. So, I have this domain, so this is Z=0 and then this is my Z=1 and I have this similar notation. So, this is my Z1, this is my Z2, this is my Z3, this is Zi, I okay. These are my grid points in a domain 0 to 1 where this domain appears in the context of let us say a boundary value problem. So, let us rewrite our boundary value problem again. So, the general boundary value problem.

So, in the domain 0 to 1, I want this differential equation to hold actually at every point but when we do discretization, we will end up forcing this equation only at certain grid points, the same way that we have done for finite difference; and then, I have these boundary conditions f1. So, I have 2 boundary conditions and I have this second order differential equation. I need to develop approximations for d2U/dZ square dU/dZ and then at a particular point, let us say ith point, I want to develop local approximation for the derivatives, okay.

Now, the way I am going to proceed is let us denote the value of the solution at these points as U1, U2. So, basically the same idea where we had said that Ui=U of Zi where U is the dependent variable here. The solution of the problem, it could be temperature distribution, it could be concentration distribution whatever is the problem at hand. So, this is my solution. Now, the trouble is when you are solving the partial differential equation and when you are trying to develop an interpolation polynomial, you do not know these values, U1, U2, U3 up to Un+1.

You do not know these values, okay. So, the trick is use interpolation polynomial together with this differential equation to find U1, U2, U3, up to Un+1, okay. So, when I begin with the development, okay. I am going to develop everything in terms of U1 to Un, okay as unknowns. Now, let us see how we do this.

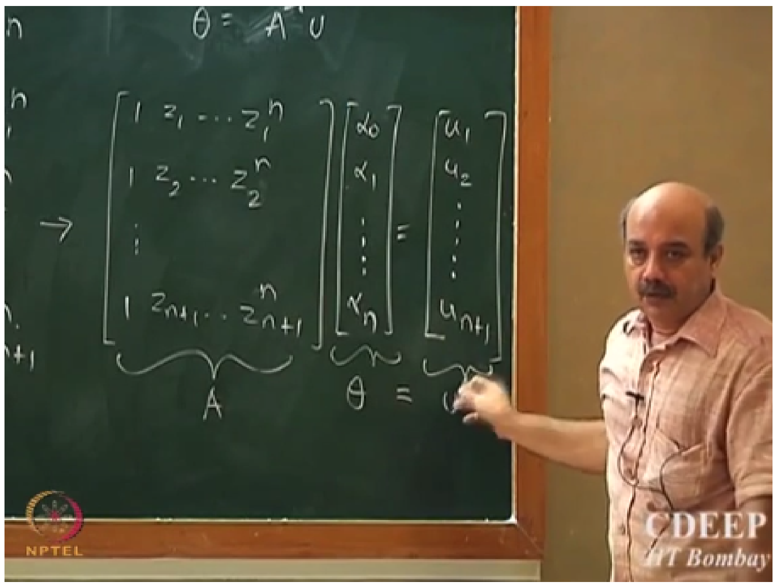
(Refer Slide Time: 19:30)



So, now what I am going to is my approximate solution for this problem, okay, I am going to represent as an interpolation polynomial. So, let me call this as UZ=alpha 0+ alpha 1 Z, okay. This is my proposed solution, approximate solution for this particular problem, okay. Now, if you want to develop interpolation polynomial, what will you do. You will write this equation U1=alpha 0+alpha 1 Z1…up to alpha n Zn, right.

Then U2=alpha n Zn. So, you would write all these equations, okay. U1, U2 up to Un, these are the values of this approximate solution at the grid points, okay. Right now, I have not talked about how to decide this grid points. In this context often called knots or collocation points. They are not really called as grid points, just a matter of terminology. They are same as grid points; they are often called as collocation points. So, these points here will be called as collocation points, okay. So, now I can write this.

(Refer Slide Time: 21:54)

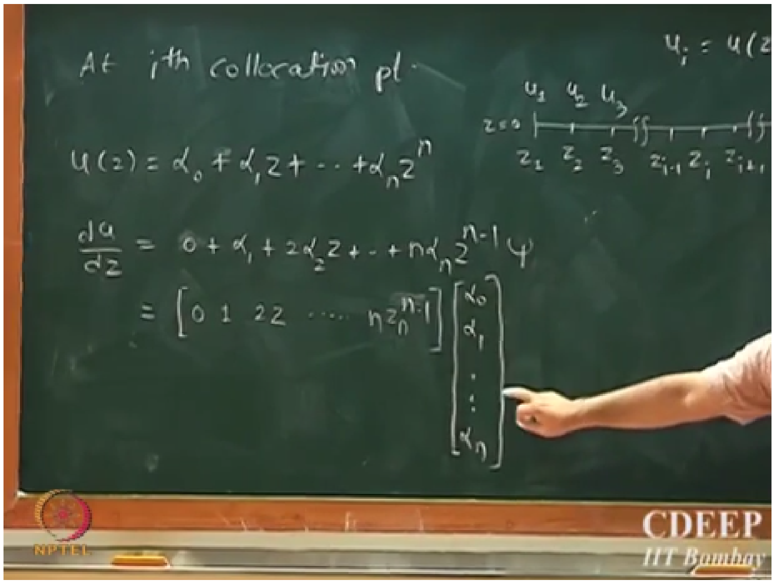


Then, I can put this into the standard matrix form, okay. So, these equations n+1 equations*n +1 unknowns, I have transformed into a standard matrix equation. Now, there is a trouble here, okay. In normal interpolation when it is not connected with a differential equation, you are given some values of the function at the grid points. So, in normal polynomial interpolation, you know these values.

Now, I do not know these values right now, okay. It does not matter. I have going to play some tricks. First of all, in this equation I do not know 2 things, I do not know alpha 0, alpha 1 to alpha n. I do not know U1 to Un+1. So, both I do not know. So, I cannot really solve this problem and find the solution, that is not possible. The solution is tied up with the differential equation, okay. So, we have to go to differential equation to get these values.

But what I am going to do here is let us call this matrix as A matrix, this vector as theta vector and this vector here as capital U, A theta=U. So, my equation is A theta=U, okay. So, I am going to transform this equation and write theta=A inverse*U, capital U is this vector Un to Un+1, okay. So, what I have do is that this set of unknowns can be represented in terms of this set of unknowns, okay. Let us leave here and then let us continue with the next part.

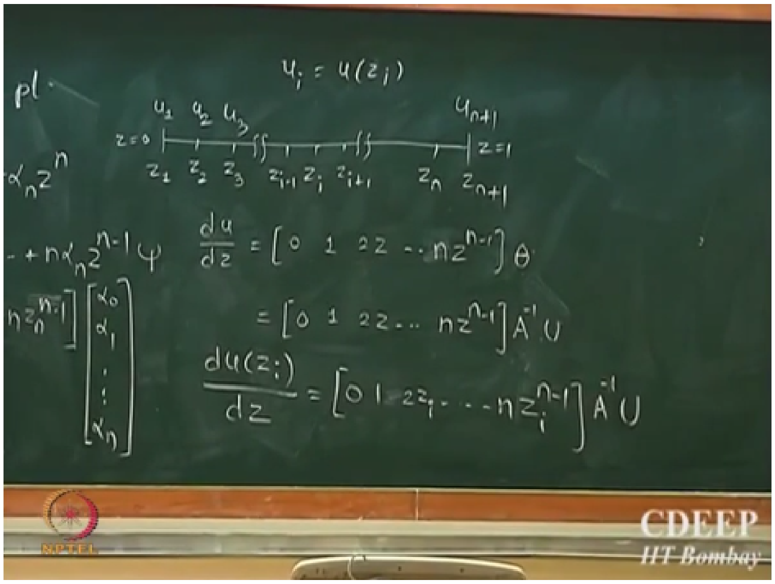
(Refer Slide Time: 24:36)



Now, let us come to ith collocation point, okay. So, at ith collocation point what is the approximate solution. Before I go to the ith collocation point, what is the general solution UZ=alpha 0+alpha 1 Z up to alpha n Zn, right. This is the approximate solution. What is the first derivative? What is dU/dZ. DU/dZ is 0+alpha 1+2 alpha 2 z up to n alpha n Z n-1. Have you all with me on this? okay.

I am going to write this in slightly different way. I am going to say that this is 0, 1, 2… I am going to write this as an inner product of 2 vectors. One vector is 0, 1, 2Z, 3Z square, 4Z cube and so on, okay. So, this has been written as inner product of 2 vectors. Dot product of 2 vectors, okay. So far so good.

(Refer Slide Time: 27:07)

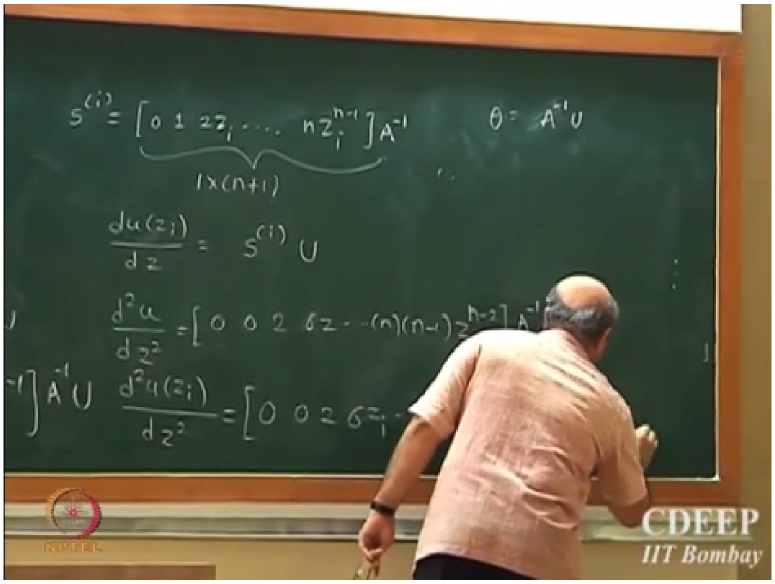


So, here now this first derivative is actually. So, this dU/dZ=0, 1, 2Z, …. nZ raise to n-1*theta. Do you agree with me? This is my theta vector. So, this into theta vector, okay which is same as 0, 1, 2Z*A inverse U. I am just replacing theta. Unknown theta is not, so convenient for me to work with. I am going to work with unknowns U1 to Un+1, okay. So, this is a known matrix. A is the known matrix. I have chosen the collocation points, so I can compute A matrix, okay.

I can compute A inverse, so this is the known matrix to me, okay. Now, this multiplication of this vector into this matrix, I am going to denote it. See, this is a vector, this is a row vector, this is a n+1 cross n+1 matrix, and this is a vector which is 1 cross n+1, okay. So, multiplication of these 2, this row vector into this matrix, what will it give you another row vector. I am going to call that row vector as Si. Si is the ith row vector or the row vector associated with the ith collocation point, okay.

There is one more step in between. So, this is the general expression for the derivative. Now, I want to find derivative at the ith collocation point. So, how will you find out. So, here we will have to put Zi, okay. So, dU Zi/dZ will be 0, 1, 2 Zi, … nZi is to n-1*A inverse U. When I want to compute the derivative at the ith collocation point? I am just going to substitute for Z=Zi I will get this expression.

(Refer Slide Time: 30:23)



Now, this quantity I am going to denote as Si=0, 1, 2Zi *A inverse, okay. This is my row vector. What you get here is a row vector. So, this will be 1 cross n+1 row vector, okay. Then, what I get a very simple expression dU Zi/dZ=Si*U. Is everyone with me on this, right. So, the derivative approximation at ith collocation point is this vector times U. What is U, U1 to Un+1, okay. Likewise, I can develop my second order derivative. Can you do that?

So what is my d2U/dZ square. So, this would be 0 0, second derivative will give you only 2, okay. Then, what will we get 6Z n*n-1 Z raise to n-2, am I correct. If I write it in terms of into theta but what is theta A inverse U, so A inverse U, right. I am just skipping in between steps. They are very simple. Same derivation as previous derivation for the first order derivative. So, I am just directly writing the final form here, okay.

You will get this vector and you will get this A inverse*U, okay. When I want to compute the derivative at ith collocation point, so I would just substitute d2U.