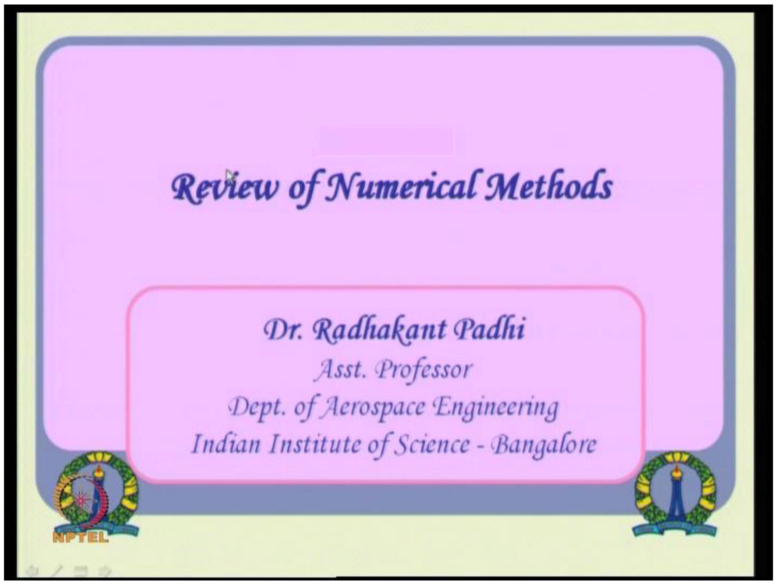
Advanced Control System Design

Prof. Radhakant Padhi
Department of Aerospace Engineering
Indian Institute of Science, Bangalore

Lecture No. # 15
Review of Numerical Methods

We will continue further in our lecture series.

(Refer Slide Time: 00:24)

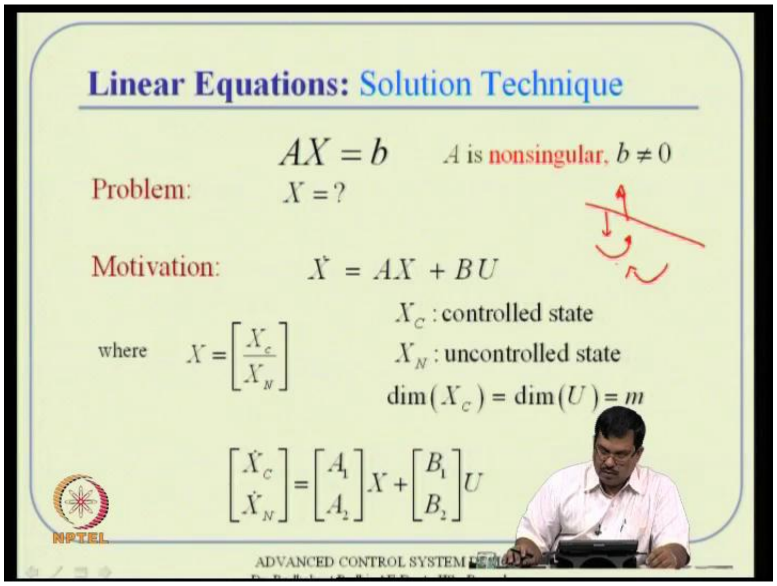


Review of Numerical Methods

Dr. Radhakant Padhi
Asst. Professor
Dept. of Aerospace Engineering
Indian Institute of Science - Bangalore

And, many times in our control theory as well, we will need a lot of a numerical methods on the way. Especially, with respect to non-linear systems theory, the things are not available in closed form. We have to go either to that. And, in linear systems also, remember, that we need some operating point first. Before we talk about a deviation dynamics, which we linearize and then interpret that as linear systems, we will talk about linearization in one of those lectures later – about the linearization method in a CS pool. And then, we will see some of these other concepts also; like root finding and other things are useful in lot of applications. So, let us quickly review some of these numerical methods that are frequently kind of required in our analysis and synthesis.

(Refer Slide Time: 01:17)



Linear Equations: Solution Technique

Problem: $AX = b$ $\quad A$ is nonsingular, $b \neq 0$

$X = ?$

Motivation: $\dot{X} = AX + BU$

where $X = \left[\frac{X_c}{X_N}\right]$ $\quad X_C$: controlled state; X_N : uncontrolled state; $\dim(X_C) = \dim(U) = m$

$$\begin{bmatrix} \dot{X}_C \\ \dot{X}_N \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} X + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} U$$

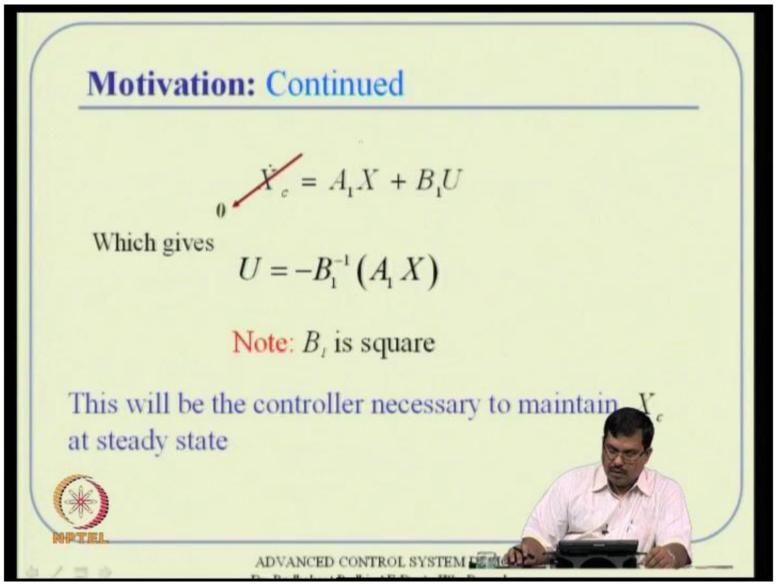
The first concept that comes to mind is – first problem is linear equation solution. Suppose you have AX equal to b. And, remember we know for sure that if A is some rectangular and things like that, A is a square matrix for which a determinant of A is not 0; that means A is nonsingular and b is not equal to 0; then, X equal to A inverse B; we know for sure. But, A inverse is adjoint of it divided by determinant of A. And, that requires a lot of computation. We do not want to go to that definition all the time whenever that is required.

Now, the question is where do we require this kind of solution in our control theory? And, one of that is you can see some of these motivation part of it. Suppose we want to find out something like this. This is (Refer Slide Time: 02:12) a typical equation X dot equal to AX plus BU. And, let us say we divide this X into like controlled state and uncontrolled state; like X c is controlled state, X n is uncontrolled state. And, we divide it in such a way that it is like dimension of X c is equal to dimension of U. So, if I just consider the upper portion of the equation; that means X c dot is equal to A 1 X plus B 1 U; then, I will tell this B 1… Remember the dimension of X c is equal to dimension of U; that means B 1 is the square matrix now.

Then, I will probably try to find out what is my equilibrium condition; that means this X c dot is 0 (Refer Slide Time: 02:49); if I take about the controlled state equilibrium and I want to interpret that as a forced equilibrium point; that means the controlled accessibility is not 0 for that. So I want to maintain that particular equilibrium point. And then, what is my control required for that? It is also like what is called as trim control in aircraft. Like if your aircraft flies steady and level, you really require some sort of a trim moment, because remember, c g and c p are not at the same point; that means continuously there is this aerodynamic lift-related moment, which will act on the vehicle actually.

If I just tell that in a little small diagram. This is (Refer Slide Time: 03:32) something like there; and, c g will be somewhere here and c p is somewhere there. So, that means this particular thing will give us some sort of a moment about that. There is a turning moment sort of a thing. So, you want to cancel that by using something like an equivalent moment; so that means you really require some sort of a small elevator deflection. So, the trim condition is actually a forced equilibrium point in an aircraft plane. And, those of you, who cannot relate that, we will see that a little later; that is very obvious also; it is not too much. You can see this problem from other example problems as well.

(Refer Slide Time: 04:15)



Motivation: Continued

$$\underset{0}{\cancel{\dot{X}_c}} = A_1 X + B_1 U$$

Which gives

$$U = -B_1^{-1}(A_1 X)$$

Note: B_1 is square

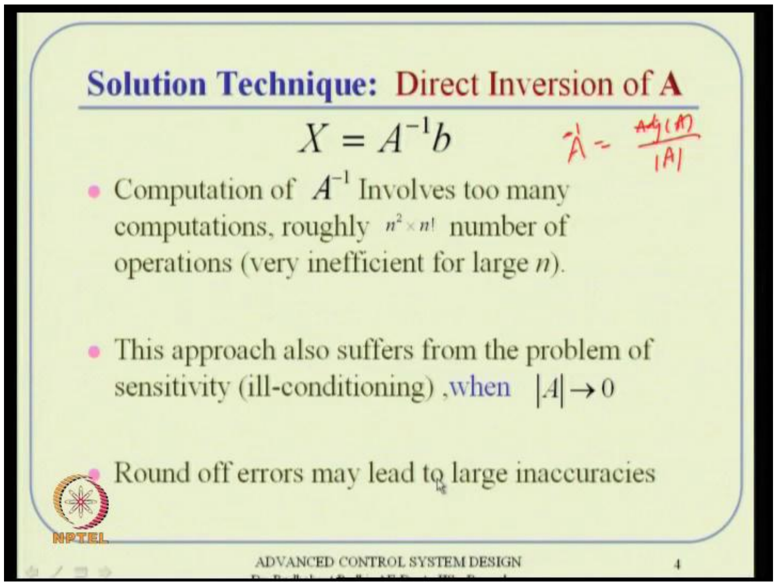
This will be the controller necessary to maintain X_c at steady state

ADVANCED CONTROL SYSTEM DESIGN

So, what I am looking at? I am looking at some solution for something like X c dot equal to 0; that means the control states are operating on the steady state. If I really want to find out what is the control necessary for that particular situation, then I can find a control action that way, because remember, B 1 is actually nonsingular; and, square are nonsingular. So, I can take that way. So, if I apply this control action – speed by control, then I will assure that this happens to be 0, so that the aircraft continues to fly at steady and level basically. That particular control is actually necessary for maintaining X c dot – 0. If X c is X dot for example, and U is delta e – elevated deflection, then this elevated deflection I have to compute it that way. So, obviously, we require this type of solution.

Now, this is (Refer Slide Time: 05:03) A 1 of X is nothing but B. If you are going back to that, that (Refer Slide Time: 05:06) is the B and this A y is nothing but B 1 in this case. So, we require this kind of solution (( )) efficient ways of getting solution and things like that. Remember, this control has to be applied online also; that means we cannot afford to have lot of computational time getting wasted for computing this B 1 inverse; we do not want that actually. We want to have an official way of controlling, finding out this B 1 inverse.

(Refer Slide Time: 05:31)



Solution Technique: Direct Inversion of A

$$X = A^{-1}b \qquad A^{-1} = \frac{Adj(A)}{|A|}$$

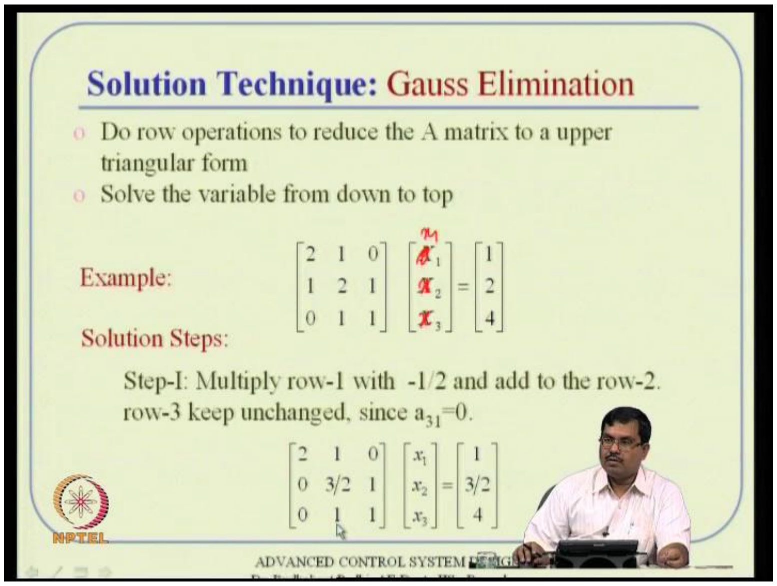
- Computation of A^{-1} Involves too many computations, roughly $n^2 \times n!$ number of operations (very inefficient for large n).
- This approach also suffers from the problem of sensitivity (ill-conditioning) ,when $|A| \to 0$
- Round off errors may lead to large inaccuracies

ADVANCED CONTROL SYSTEM DESIGN 4

Coming back to that, this is the equation that we are asking. And then, we are observing that A inverse involves too many computations. And roughly, the computational complexity sense – it requires n square into n factorial computations; that means if A is large, larger and larger, you are getting trapped in this computational complexity. So, you want to do it in an efficient way. And, one of that efficient way happens to be like Gauss elimination, which we will see in this class actually.

Now, this approach not just suffers from this computational headache; it also suffers from these other problems like what is one thing, is called ill-conditioning. Now, remember, A inverse is nothing but adjoint of A divided by determinant of A; A inverse is equal to adjoint of A divided by determinant of A. So, if determinant of A goes 0; that means if you have a singularity, is approaching – you are approaching singularity, then this is a serious problem. You cannot just talk about division by that and things like that. And, also remember, there will be like so many computations and lot of computations – all of these computations will have a small round of errors. If there are too many computations, then round off errors can be large. So, you do not want to do too many computations in any situation basically. The computers are always like finite length you can take; I mean the number and all that – you will have finite digits only. So, there will be round off errors all the time actually. So, given a choice, you do not want to do too many computations.

(Refer Slide Time: 07:11)



Solution Technique: Gauss Elimination

- Do row operations to reduce the A matrix to a upper triangular form
- Solve the variable from down to top

Example:

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \overset{x}{\cancel{x}}_1 \\ \cancel{x}_2 \\ \cancel{x}_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$$

Solution Steps:

Step-I: Multiply row-1 with -1/2 and add to the row-2. row-3 keep unchanged, since $a_{31}=0$.

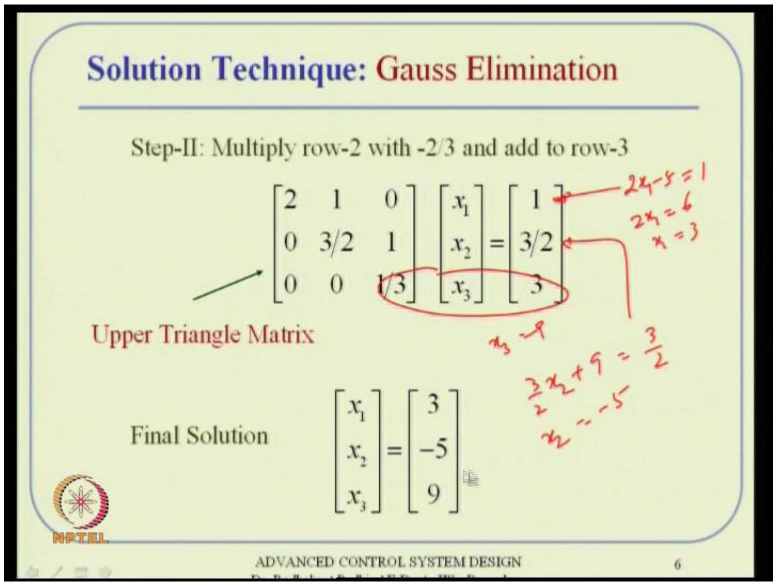
$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 3/2 & 1 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/2 \\ 4 \end{bmatrix}$$

ADVANCED CONTROL SYSTEM DESIGN

Now, Gaussian elimination is a substitute to that. And, instead of general things, we will just take it as a small example and try to see what is going on here. We have this kind of an equation let us say – 3 by 3. And then, all these… I think there is a small mistake again – all these are small x 1; this is small x 2; this is small x 3. They are all elements. So, this equation – what we want to do? We want to reduce this equation – whatever equation you have. In this A matrix, to be a upper triangular matrix. And, we take advantage of the fact that if we multiply any row by any constant, then take add and subtract to other row; then, the equation have not changed; equation remains same. So, then we take advantage of that thing and then tell we want to reduce this matrix to an upper triangular form by taking advantage of this row transform – I mean row properties like row multiplication, addition, things like that. So, what you do?

We will try to see these elements – whatever you see here – 1 1 0. This has to be all zeros (Refer Slide Time: 08:29). Now, 1 0 is already there. So I want to make sure that this 0 also happens here or this 0 happens here, either way. So, now, I see how we can do. I multiply this particular thing by minus half – one half and then add it up; that means I want to make sure that this becomes 0. If I do that, then 2 by 2 is 1; then, 1 minus 1 is 0 basically. So, that will pop up there. And, corresponding to that like say 2 minus 1 half; 2 minus 1 half is 3 by 2 like that. And, 0 will not change this. So, that is the equation that I live with. Again, 2 minus one half is 3 by 2 here. Now, you see that this is already 0 0, but I have to make sure that is also 0. So, I will multiply this element, this row by 2 by 3 this time; and then, subtract it.

(Refer Slide Time: 09:20)



Solution Technique: Gauss Elimination

Step-II: Multiply row-2 with -2/3 and add to row-3

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 3/2 & 1 \\ 0 & 0 & 1/3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/2 \\ 3 \end{bmatrix}$$

$2x_1 - 5 = 1$

$2x_1 = 6$

$x = 3$

$x_3 = 9$

$\frac{3}{2}x_2 + 9 = \frac{3}{2}$

$x_2 = -5$

Upper Triangle Matrix

Final Solution $$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -5 \\ 9 \end{bmatrix}$$

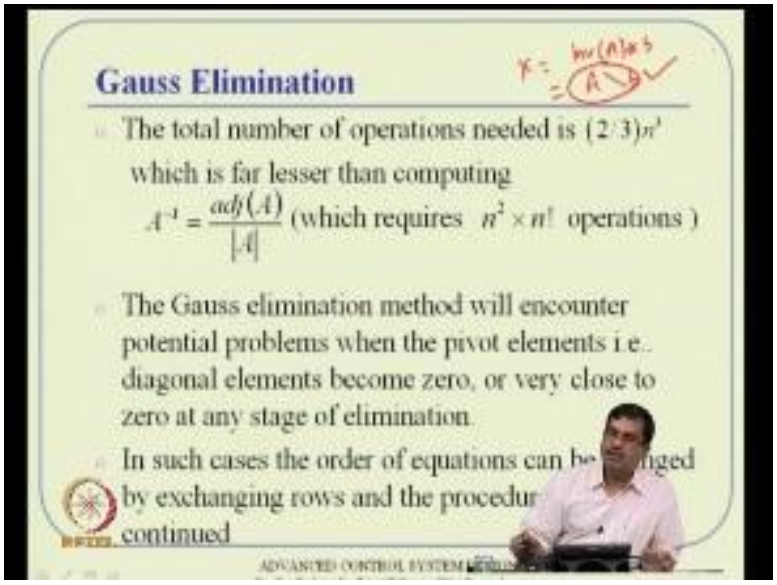
ADVANCED CONTROL SYSTEM DESIGN 6

If I do that, then I will end up with something like this. Now, this is an upper triangular matrix. So, I start looking at the equation from down to top. So, then it turns out that minus one-third is nothing but… Minus one-third of x 3 is equal to 3. And hence, x 3 is 9. Once I get x 3 value, then I put that this particular thing as a function of x 3; and, this particular… Remember this – let us do that – this particular thing has given me x 3 equal to 9. Now, if I go this equation rather, the second thing, then I get 3 by 2 x 2 plus 9, because x 3 is 9 is equal to 3 by 2. So, I try to solve this. And, if I try to solve this, then x 2 happens to be minus 5.

Now, if I go back to the first one equation now – this equation (Refer Slide Time: 10:17) now and tell this is 2 x 1 plus x 2; x 2 is already minus 5. So, this is minus 5 equal to 1. Then, 2 x 1 turns out to be 6 and x 1 equal to 3. That is why you get x 1 equal to 3,

x 2 equal to minus 5, and x 3 equal to 9; that is easy now. So, what we are you doing here? We are somehow trying to eliminate this so-called pivotal elements and make it 0 0. And then, once you get an upper triangular matrix sort of thing here, you try to look the equations from bottom to top, because then they try to solve these equations in straight forward manner. So, that is Gaussian elimination.

(Refer Slide Time: 11:07)



Gauss Elimination

$x = inv(A)*b$ ✓
$= A\backslash b$

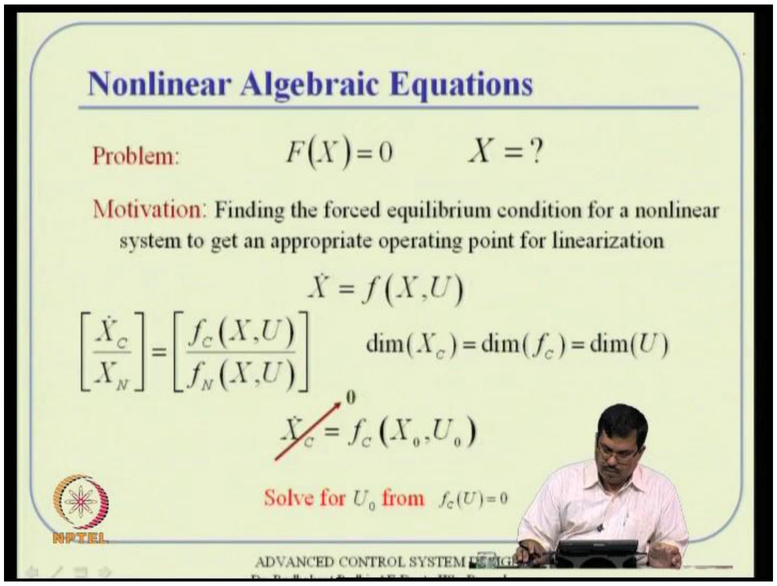
- The total number of operations needed is $(2/3)n^3$ which is far lesser than computing $A^{-1} = \frac{adj(A)}{|A|}$ (which requires $n^2 \times n!$ operations)
- The Gauss elimination method will encounter potential problems when the pivot elements i.e. diagonal elements become zero, or very close to zero at any stage of elimination
- In such cases the order of equations can be [illegible]nged by exchanging rows and the procedur[illegible] continued

ADVANCED CONTROL SYSTEM [illegible]

Now, gauss elimination suddenly makes this n square into n factorial operations. This reduces to just this much operation. Remember – this is actually something like a very high computational thing to a very low computational thing. Remember – n cube is nothing but simply n not n factorial here. And, that will get multiplied with n square, which is a lot of computation really. So, obviously, it leads to far lesser computation; and then, hence, it is kind of preferred. Especially in matlab command window, some of you use that, then you have a choice of using inv function – inv into b, which is like x; or, the same thing you can do that using A – these slashes – A slash b; that will give you Gaussian elimination. So, instead of using the first one, I will suggest that you use this one – A slash b. All is well here. But, see some problems.

What first problem is Gauss elimination method will also encounter problems if the pivotal element happens to be 0; one of the diagonal becomes… Suppose if you do this exercise and this happens to be 0 (Refer Slide Time: 12:35). Now, no matter how much you multiply and hence try to add and subtract, nothing will happen to this one; you will not be able to do that. So, in those situations, one easy way to that is just to exchange the rows. Suppose if this one happens to be 0 here, then you take the entire equation; you put a third equation. And, this one you substitute for second equation. You exchange the equations; then, proceed further. That is the idea there. So, this Gaussian elimination is a very standard practice now and lot of people make use of that. Again, in matlab window – matlab command, matlab environment, I suggest that you use this one.

(Refer Slide Time: 13:27)



Nonlinear Algebraic Equations

Problem: $F(X)=0 \qquad X=?$

Motivation: Finding the forced equilibrium condition for a nonlinear system to get an appropriate operating point for linearization

$$\dot{X}=f(X,U)$$

$$\begin{bmatrix} \dot{X}_C \\ X_N \end{bmatrix} = \begin{bmatrix} f_C(X,U) \\ f_N(X,U) \end{bmatrix} \qquad \dim(X_C)=\dim(f_C)=\dim(U)$$

$$\cancelto{0}{\dot{X}_C} = f_C(X_0,U_0)$$

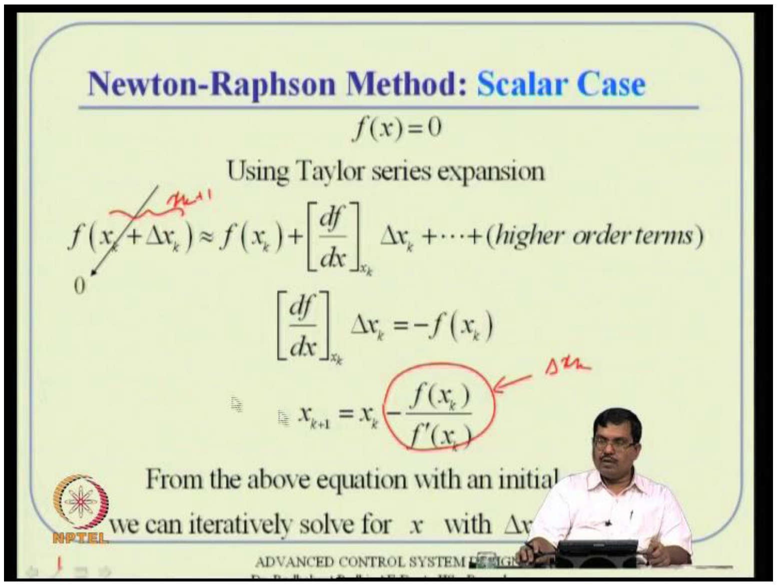
Solve for U_0 from $f_C(U)=0$

ADVANCED CONTROL SYSTEM DESIGN

Now, going back to the next concept; what we saw there is an algebraic equation solution for linear systems; that means we are talking about this is a linear sort of equation; A X equal to b, what you have (Refer Slide Time: 13:41) is actually a linear equation. Now, what if we have a non-linear equation? Remember – the numerical methods are more powerful for non-linear cases and all that.

Now, let us see something like this – F of X equal to 0 (Refer Slide Time: 13:58). And, you want to find out what is X equal to. And again, the similar motivation there – like I can partition the states, whatever X dot equal to f of X, U into two parts. I still maintain that condition – dimension of X is equal to dimension of U. Then, in this case, you will end up with like X c dot equal to 0 if I want to attach that question. Then, I have this non-linear equation now. So, f c of X 0, U 0 is equal to 0. That is the condition that I want to find out; that means can I find out some U 0, which will satisfy this equation; U 0 can be a function of X 0; that is ok; that means if I know a particular number for X, that is, X 0, what is the particular number for U, which will give me that? That is why this 0 0 notations are there. So, I will go back to this equation now – f c of U equal to 0 and try to find out what is U for that. That is the motivation why we want to address this particular problem. This is just one of the examples. Why you need that? You may need it for many different cases also.

(Refer Slide Time: 15:17)



Newton-Raphson Method: Scalar Case

$$f(x)=0$$

Using Taylor series expansion

$$\underset{0}{f\left(\overset{x_{k+1}}{\cancel{x_k+\Delta x_k}}\right)} \approx f(x_k)+\left[\frac{df}{dx}\right]_{x_k}\Delta x_k+\cdots+(\textit{higher order terms})$$

$$\left[\frac{df}{dx}\right]_{x_k}\Delta x_k=-f(x_k)$$

$$x_{k+1}=x_k\boxed{-\frac{f(x_k)}{f'(x_k)}}\xleftarrow{\Delta x_k}$$

From the above equation with an initial

we can iteratively solve for x with Δx

ADVANCED CONTROL SYSTEM DESIGN

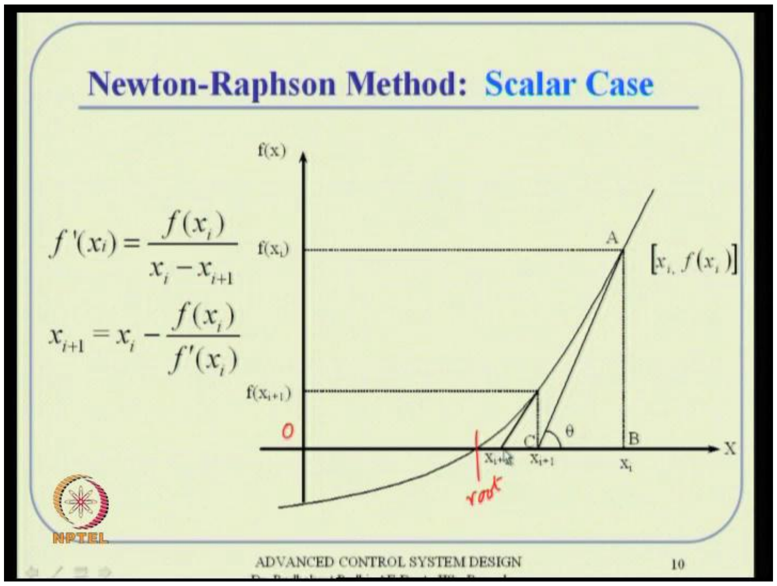
Now, we will start with a simple solution sort of idea. First, we will see what is a scalar case. By the way, this is many numerical methods available. One of that, which is popularly known is bisection method; that means if you talk about a scalar expression here for example, and you tell I have two guess values now; then, there is something called a bisection method. And, there are very other ideas as well. So, this is not… What I am talking here is not an exact review of methods available. So, what I am really doing here is rather talking about what is called as Newton-Raphson method, which is very neat. It has its own beauty basically. And, that is what is mostly used in practice. That is why we want to review this particular method of getting the solution.

What you really want to find out in this case? Remember – non-linear equation solution will require some sort of iterative solution; that means in one go you will not be able to find out. But, you may need some couple of iterations before you arrive at the final solution. So, what is the idea here? Now, let us say that we are having some x k value. Some value for x k is already available and k can be 0 also; that means you can start with some initial guess. Then, if k is 1, 2, 3, 4, then you have already some iterated values available to you. Obviously, that is not the solution. That is why you want to do iteration; that means f of x k is not really 0. However, you want to find out a delta x k such that if I make x k plus delta x k, then f of x k plus delta x k is equal to 0. That is what that mean; that means if my delta x k is proper, then x k plus delta x k will be a root of this equation; that is the whole idea there.

Now, I'll interpret this (Refer Slide Time: 17:11) f of x k plus delta x k in Taylor series expansion and tell these are all higher order terms and all that; I am not be interested in that; I will just suppress that and I will just take the first order term – up to first order term. And, this is what the equation. Remember this entire expression is 0, because that is supposed to be a root. That is what you want to find the delta x k such that f of x k plus delta x k is equal to 0 basically. So, that is the aim. That is how you want to find the delta x k; that means this is equal to 0 up to that – up to this term. Then, I can relate that delta x k – this particular term is 0. Hence, this d f by d x evaluated at x k times delta x k – this term – whatever term is there, is nothing but minus f of x k.

And, assuming that this is (Refer Slide Time: 18:10) f dash of x, remember that evaluated at x k. So, assuming that is nonzero, I can do that. Delta x k is nothing but this term. Whatever term you see here, that is nothing but delta x k. This particular term is nothing but delta x k. So, if this is delta x k, then what I really wanted is x k plus delta x k; that is x k plus 1. So, this particular term, what you see here is x k plus 1. That is why f of x k plus 1 is 0 basically; that is what you wanted. So, this is why you will get x k plus 1 is x k plus delta x k. And, delta x k is nothing but that. So, x k plus 1 is equal to x k minus f of x k divided by f dash of x k. That is the algorithm. And, you keep on doing that until you get some convergence; that means you do not get too much of improvement afterwards. Up to that you have to do that and stop.

(Refer Slide Time: 19:31)



Newton-Raphson Method: Scalar Case

$$f'(x_i) = \frac{f(x_i)}{x_i - x_{i+1}}$$

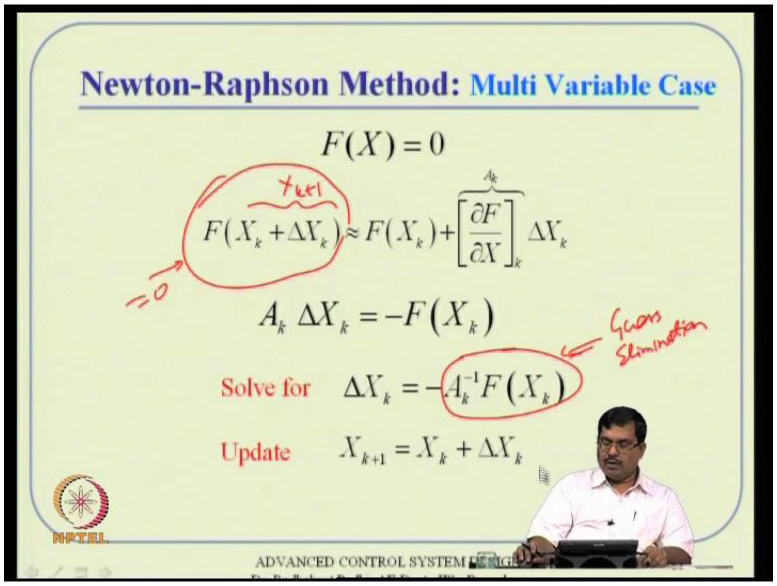
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

ADVANCED CONTROL SYSTEM DESIGN

Now, pictorially, let us see what is going on here. Here you have some f of x. Some f of x – arbitrarily, it is plotted something like that. f of x equal to 0. And, assuming that is the 0 axis; starts with 0, 0 probably; and then, obviously, what you want? You want this particular value. This is the root. This is what you really want ultimately; what you do not know the value to start with. So, you start with some sort of a guess value. So, this is the guess value; I will start with some guess value let us say. And then, f of x is that much; that is f of x. Then, f dash of x; f dash of x is nothing but the slope evaluated at x i. So, that is the slope, what is given. So, if I compute this delta x i sort of thing and if you see this is actually a linear equation, what you are doing here? x i plus 1 is x i minus this term.

And, if you just carefully look at it, it is a kind of a… If you start with a this (Refer Slide Time: 20:42) equation – straight line equation and try to find out the interpretation – the meaning of that, all that it tells me that I have a guess value; I go to that particular point, where it intersects this particular function; then, evaluate the slope; and, with that extension of the slope, I will cut this x-axis somewhere. And, that is going to be my next iteration value. So, again, I evaluate the function there; I again take the local derivative, extend that. And, wherever it cuts, it is my x i plus 2. Again, I go there and then I will proceed further like that. Then, ultimately, I will converge there actually. So, that is the interpretation of that.

(Refer Slide Time: 21:24)



Newton-Raphson Method: Multi Variable Case

$$F(X)=0$$

$$\underbrace{F(X_k+\Delta X_k)}_{X_{k+1}} \approx F(X_k)+\overbrace{\left[\frac{\partial F}{\partial X}\right]_k}^{A_k}\Delta X_k \quad =0$$

$$A_k\,\Delta X_k=-F(X_k)$$

Solve for $\Delta X_k=-\boxed{A_k^{-1}F(X_k)}$ ⟸ Gauss Elimination

Update $X_{k+1}=X_k+\Delta X_k$

NPTEL
ADVANCED CONTROL SYSTEM DESIGN

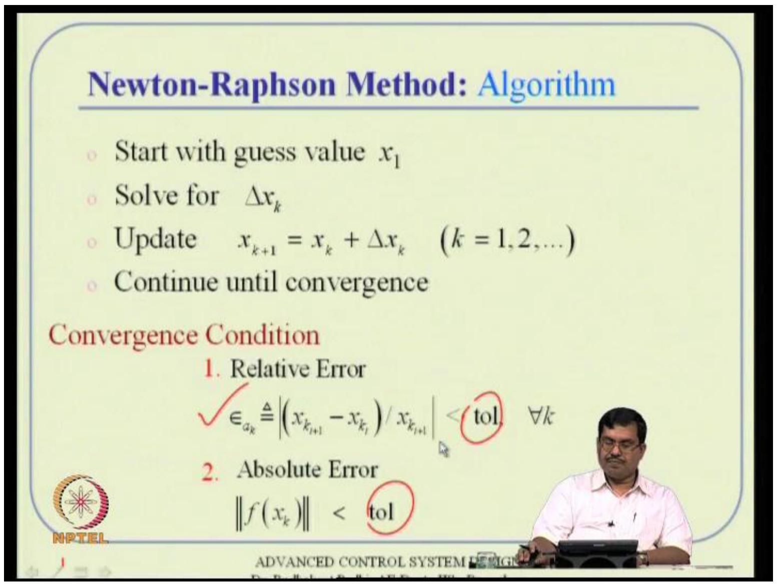
Multi variable case – there is a similar extension you can do. Now, if you want this F of X is equal to 0 you tell, I will do the same exercise; that means I will take multi variable Taylor series expansion. And then, higher order terms I will neglect. And, I want to find out delta X k in such a way that X k plus 1… This is X k plus 1 anyway; this is X k plus 1. That is going to be a root; that means F of X k plus 1 is going to be 0. So, this entire thing what I see here, I interpret that I will find delta X k in such a way that this is equal to 0. Now, if that happens, then I define this fellow as X k plus 1. I define this del F by del X, evaluated at k is nothing but A k, something like that.

And then, the equation tells me that A k times delta X k is nothing but negative of this particular thing (Refer Slide Time: 22:26) F of X k. And hence, delta X k is nothing but A k inverse F k. And remember, whenever you see this kind of a thing, you can use Gaussian elimination. You can use Gaussian elimination to compute that in a faster way. Then, you can update. Then, you get X k plus 1 is X k plus delta X k. So, very similar to what you have here (Refer Slide Time: 23:02). Instead of divided by 1 by f dash of x k in the matrix theory, you cannot talk that 1 over matrix; that is not defined. So, I will tell, it is a negative of A k inverse multiplied by that; that is the only difference; otherwise, it will continue. And, all numerical methods – most of them, will always rely on Taylor series expansion; whether it is root finding, whether it is Euler equation or whether it is many things, we will all rely on

Taylor series expansion. So, all these numerical methods derive the strength from Taylor series expansion.

The algorithm is again same. You start with some guess value and then compute delta X k like that and then proceed the algorithm like that (Refer Slide Time: 23:52). And remember, in matlab, those of you will be using that; for scalar thing, you do not have to do all these yourself. If you want, you can do it; otherwise, in matlab, there is a function called f 0. If you use this f 0 function, it will try to find out a solution around the guess value that you have to give. This function of demand – a function, which will return; and then, there is a guess value it will demand. So, it will find out a solution around this guess value. Similarly, this multi dimensional root I think to my knowledge is not available directly. But, from optimization tool box, something is available; otherwise, you can write your own function also. This I think that you need to write anyway.

(Refer Slide Time: 24:40)



Newton-Raphson Method: Algorithm

- Start with guess value x_1
- Solve for Δx_k
- Update $x_{k+1} = x_k + \Delta x_k \quad (k = 1, 2, \ldots)$
- Continue until convergence

Convergence Condition

1. Relative Error

$$\in_{a_k} \triangleq \left| \left(x_{k_{i+1}} - x_{k_i} \right) / x_{k_{i+1}} \right| < \text{tol} \quad \forall k$$

2. Absolute Error

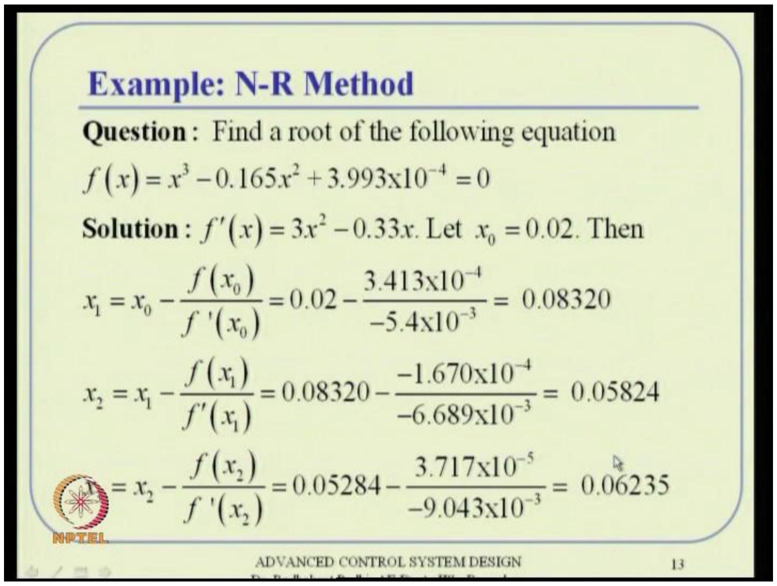
$$\| f(x_k) \| < \text{tol}$$

ADVANCED CONTROL SYSTEM

The Newton-Raphson method algorithm sense – you start with some guess value – either you take that as x 0 or x 1, whatever it is; and the matlab will start with x 1; index 0 is not defined in matlab anyway. So, you start with some guess value x 1 and then solve for delta x k; and then, update x k plus 1, is like this. And then, you continue until convergence. And, convergence – typically, it is given either in terms of relative error or in terms of absolute error. Given a choice, relative error is my first preference – first recommendation, because the problem relate that… the tolerance value that you select whether here or there – it should not be problem dependent; it should be fairly independent.

If you talk about 1 percent error, 1 percent error – whether you have a nano science problem or have a rocket science problem, either way. So, in other words, the units can be very different, but still the relative sense is still 1 percent early. So, that is why most of the time, I will recommend relative error. And, absolute errors are sometimes necessary, because suppose you have this particular x k plus 1, what you see (Refer Slide Time: 25:55) here is 0, then there is a problem of division. So, then tell, I will continue with absolute error, because I already have a physical idea of what these values are. That will help me in selecting a proper value for this absolute error. That will be always my second recommendation. First recommendation is this one.

(Refer Slide Time: 26:15)



Example: N-R Method

Question: Find a root of the following equation

$f(x) = x^3 - 0.165x^2 + 3.993\text{x}10^{-4} = 0$

Solution: $f'(x) = 3x^2 - 0.33x$. Let $x_0 = 0.02$. Then

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0.02 - \frac{3.413\text{x}10^{-4}}{-5.4\text{x}10^{-3}} = 0.08320$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.08320 - \frac{-1.670\text{x}10^{-4}}{-6.689\text{x}10^{-3}} = 0.05824$$

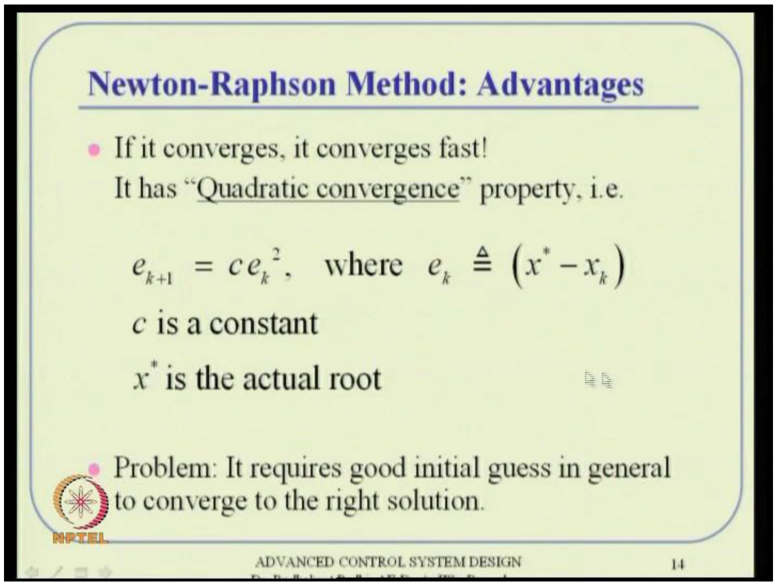
$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 0.05284 - \frac{3.717\text{x}10^{-5}}{-9.043\text{x}10^{-3}} = 0.06235$$

ADVANCED CONTROL SYSTEM DESIGN 13

Now, let us see in Newton-Raphson method, how do you find that – find the root of the following let us say. Then, you will start with some x 0 value as 0.02; that you can start with that. And then, you tell this is x 1; x 1 nothing but x 0 minus this. And then, x 2 is that and x 3 is that; you can continue. And, you can see that there is some improvement here – 0.08205; 08205 is 0.03 sort of improvement here. And then, suddenly, it is 0.01

improvement. And, you do one or two more iterations; it will converge. That is the method; that is the thing.

(Refer Slide Time: 26:53)



Newton-Raphson Method: Advantages

- If it converges, it converges fast!
  It has "Quadratic convergence" property, i.e.

  $$e_{k+1} = c e_k^{2}, \quad \text{where} \quad e_k \triangleq \left(x^* - x_k\right)$$

  c is a constant

  x^* is the actual root

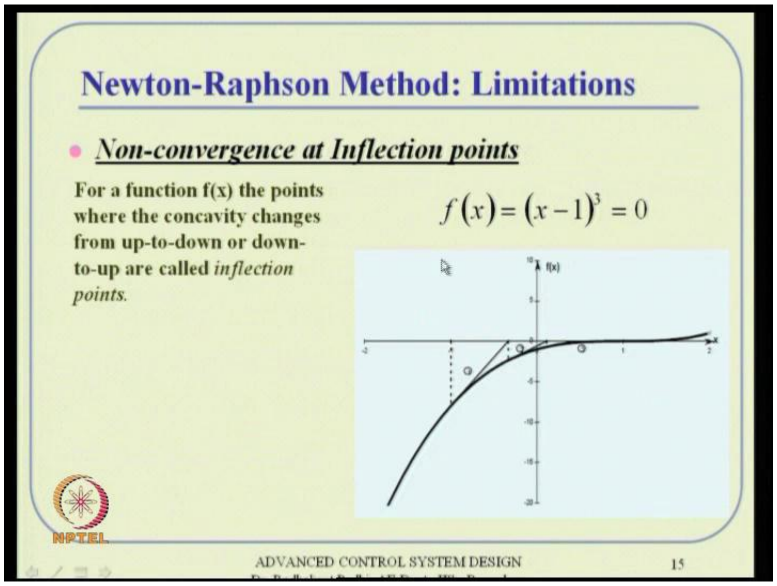
- Problem: It requires good initial guess in general to converge to the right solution.

ADVANCED CONTROL SYSTEM DESIGN 14

What is the beauty about this particular method? Why do you emphasize so much on this? Is primarily because of this property. What it tells is this particular method has something called quadratic convergence property; it means that actually; that means e k plus 1, whatever e k I get, at every instant, these are all iteration values. But, correspondingly, I can calculate e 1. And, e 1 is nothing but x 1 minus x 0 here; e 2 is x 2 minus x 1, like that; e 3 is x 3 minus x 2 like that. So, if I calculate that way, and then it will turn out that e k plus 1 is something like c times e k square.

And normally, this (Refer Slide Time: 27:34) e k – if it is relative error typically, these are all less than 1. e k s are typically less than 1. And then, if it less than 1, e k square is still further less than 1; I mean it is very small value. And, e k plus 1 is proportional to e k square; that means it will converge very fast. That is called quadratic convergence property. However, there are several problems here. And, one of the problems is, it requires a good initial guess value in general. And, that too if you give a wrong guess value, it will converge to a wrong value. There is a possibility of doing that.

(Refer Slide Time: 28:16)



Newton-Raphson Method: Limitations

- ***Non-convergence at Inflection points***

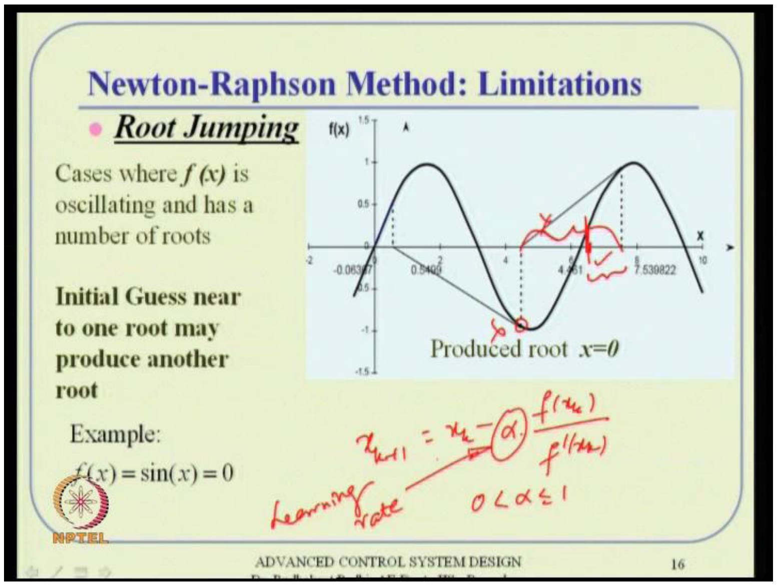
For a function f(x) the points where the concavity changes from up-to-down or down-to-up are called *inflection points*.

$$f(x) = (x-1)^3 = 0$$

ADVANCED CONTROL SYSTEM DESIGN 15

What are the problems of this method? First of all, we will see couple of issues here. And, first thing is this Newton-Raphson method does not converge at inflection points in general. What is inflection point? All powers if you have something like that. If you have x minus 1 whole cube for example, it goes through. Remember it is a solution x equal to 1. You are looking for the solution. But, it will never converge at this point of time; if you just do iteration starting from some guess value, you will see that this has some convergences. And, it turns out that if it is point of inflection, something like that, if it cuts nicely and go, it is fine. Point of inflection – normally, it has some convergences as you see. It will go close to that, but around that, it will not converge.

(Refer Slide Time: 29:08)



Newton-Raphson Method: Limitations

- ***Root Jumping***

Cases where *f (x)* is oscillating and has a number of roots

Initial Guess near to one root may produce another root

Example:

$f(x) = \sin(x) = 0$

Produced root *x=0*

$$x_{k+1} = x_k - \alpha \frac{f(x_k)}{f'(x_k)}$$

Learning rate

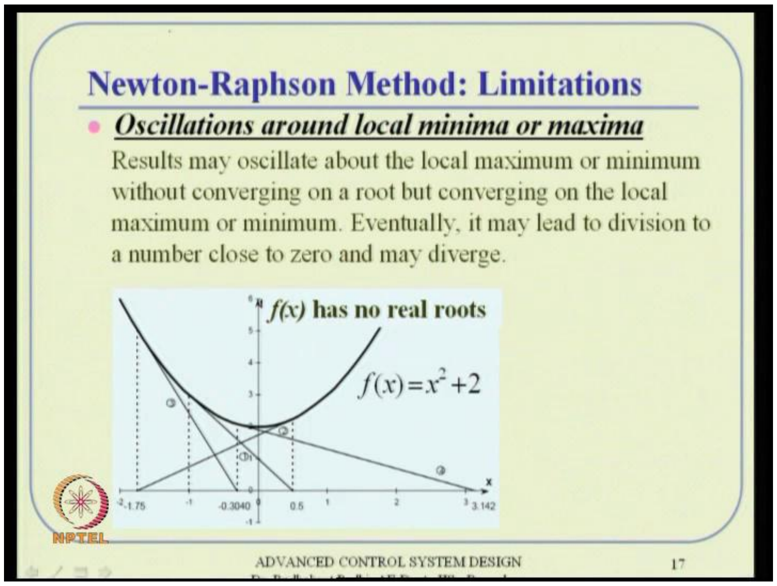
$0 < \alpha \leq 1$

ADVANCED CONTROL SYSTEM DESIGN 16

Root jumping can also happen. For example, if you take sin x; you know several solutions are there, whatever solutions are there. Now, suppose you start with this guess value somewhere, then obviously, what you intended? You intended to find out some solution here; you did not intend to find somewhere else. But, remember if you just take the local slope, it will push somewhere here; again, you take the local slope, it will push somewhere here; again, you take the local slope, it will push you somewhere here. And ultimately, you will be able to locate sin x is equal to 0; that is also a solution. That is not the right solution, because you started with this guess value with the hope that you will find out the solution.

And, one of these issues can be avoided by taking this something like what is called a learning rate. For example, x k plus 1 (Refer Slide Time: 30:02) is equal to x k minus f of x k divided by f dash of x k; that is what we discussed. Instead of doing that, you tell, there is alpha; I will multiply, where alpha is a number, which is less than 1. If I select a small value, obviously, I will not jump here; I will not take this much step. I will take another small step in this direction. So, I will be going here. So instead of going over there, I will not go here, but I will go here rather, because I will not take this particular jump. This entire thing – I will not take. But, this alpha will give me this much only. Many of these numerical algorithms will have this alpha as a term (( )) with that including optimization routines and all. This is called learning rate. And, we can probably select a particular appropriate value, which will help us eliminating some of these problems. So, first thing is, it (Refer Slide Time: 31:05) does not converge around inflection point; there is a (Refer Slide Time: 31:08) root jumping problem.

(Refer Slide Time: 31:10)



Newton-Raphson Method: Limitations

- ***Oscillations around local minima or maxima***

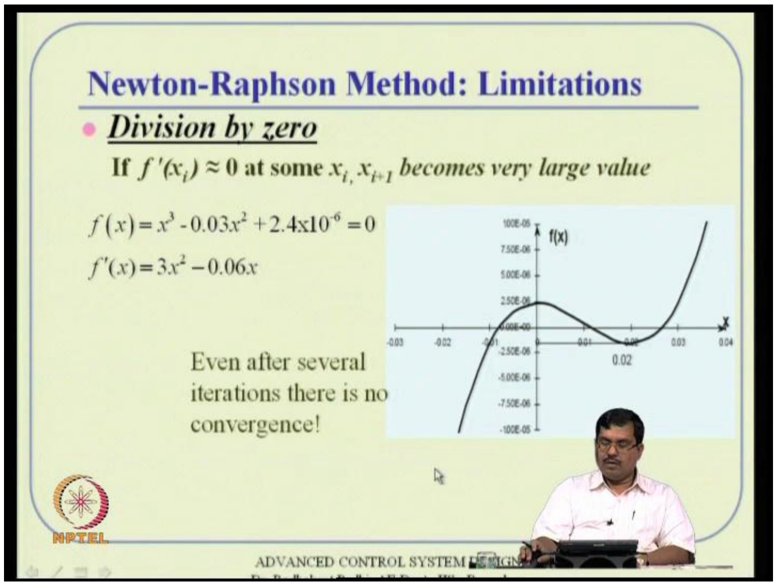
  Results may oscillate about the local maximum or minimum without converging on a root but converging on the local maximum or minimum. Eventually, it may lead to division to a number close to zero and may diverge.

$f(x)$ has no real roots
$f(x)=x^2+2$
-1.75
-0.3040
0.5
3.142
x

ADVANCED CONTROL SYSTEM DESIGN 17

Third is there is oscillation problem. Remember this particular f of x has no real roots, because it does not touch the x-axis. It just goes away from 2. There is no solution. But, you are blind to that and you still wanted a solution. If you start with some guess value, it will try to go there and then it will try to come somewhere in this… This entire thing will go somewhere here. You started with let us say something like 1 here; I do not know whatever this one. And then, it will try to kind of oscillate. It will just keep on doing; one will lead to other and other will lead to there and things like that. And obviously, that is a very clear example here, because in such situation, you see that there is no root. So, you are posing a wrong problem to the system. Anyway, it is also like a local minimum or maximum.

(Refer Slide Time: 32:15)



Newton-Raphson Method: Limitations

- ***Division by zero***

  If $f'(x_i) \approx 0$ at some x_i, x_{i+1} *becomes very large value*

$f(x) = x^3 - 0.03x^2 + 2.4\text{x}10^{-6} = 0$

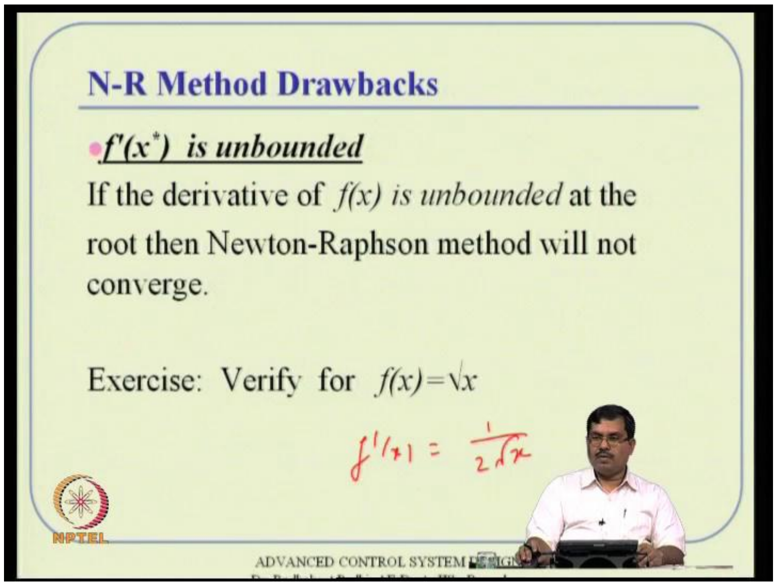
$f'(x) = 3x^2 - 0.06x$

Even after several iterations there is no convergence!

ADVANCED CONTROL SYSTEM DESIGN

There is another issue that even if it really touches the axis let us say; then, there is a problem also. Why? Because the derivative turns out to be 0 at some times. If you have this function; rather, let us say touch this axis somewhere around 0; that means this plus 2 is not there. In that situation also, what will happen is as you approach closer and closer and closer, it will approach to that solution anyway; or, once you approach closer to the solution, the slope becomes 0, so that f dash of x – this particular thing that you see here (Refer Slide Time: 32:44) – that will turn out to be 0. And hence, you will jump in (( )). It will come very close and then go back. That is where again this alpha term will help. If you have an alpha term, it will not force you to go there. So, sometimes, this adaptible learning rates are also nice. You start with a high learning rate; after couple of iterations, you reduce this value to a smaller value, smaller value, like that. So, that way, initially, you will not merge slow; you will merge rather relatively first. But, towards convergence, you will also not diverge; you will stay there. Now again, division by zero is very apparent, because whenever there is a slope zero, then there is a division by zero problem, which will not be nice either.

(Refer Slide Time: 33:33)



N-R Method Drawbacks

- ***f'(x*) is unbounded***

If the derivative of *f(x) is unbounded* at the root then Newton-Raphson method will not converge.

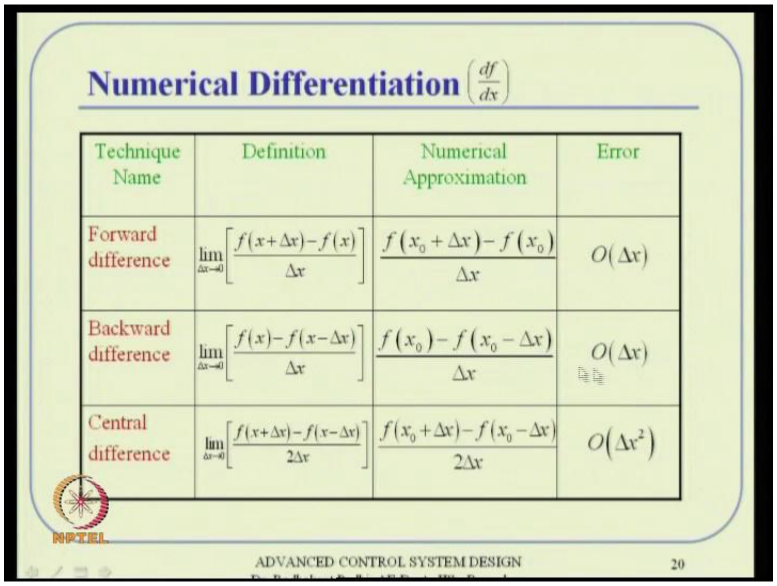
Exercise: Verify for $f(x)=\sqrt{x}$

$f'(x) = \frac{1}{2\sqrt{x}}$

ADVANCED CONTROL SYSTEM DESIGN

There is another problem that at the solution point is f dash of x is not 0, but it is unbounded; that means infinity. Then also, there is a problem. It is very counter intuitive. But, actually there is a problem. You can verify that with this example. Square root of x – if you see that, then this f dash of x star… f dash of x – remember what is f dash of x? f dash of x is actually 1 by 2 square root of x. So, if the solution turns out to be 0, that around 0 is actually infinity. Then, there is a problem there. There are many issues there for Newton-Raphson method. But, if none of these issues are actually there or you find some solution like Loranger fixing and things like that, then actually converges very fast. That is the beauty part of it.

(Refer Slide Time: 34:33)



Numerical Differentiation $\left(\frac{df}{dx}\right)$

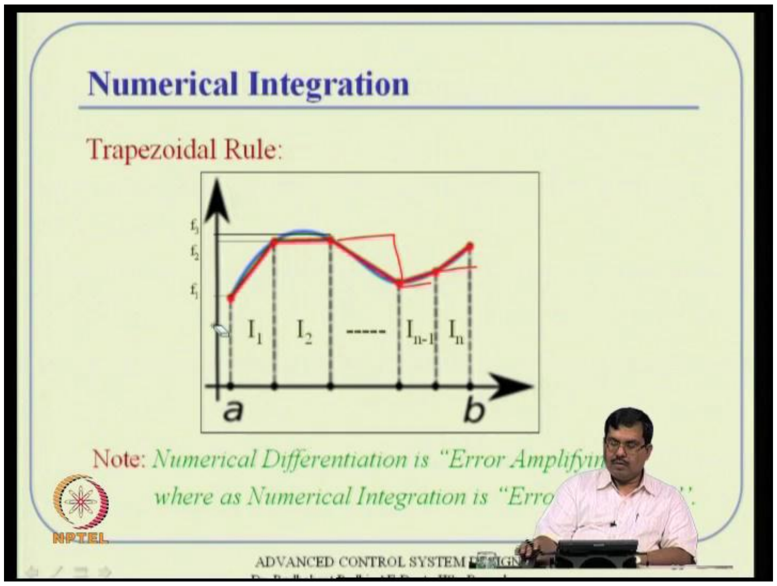
Technique Name	Definition	Numerical Approximation	Error
Forward difference	$\lim_{\Delta x \to 0}\left[\frac{f(x+\Delta x)-f(x)}{\Delta x}\right]$	$\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}$	$O(\Delta x)$
Backward difference	$\lim_{\Delta x \to 0}\left[\frac{f(x)-f(x-\Delta x)}{\Delta x}\right]$	$\frac{f(x_0)-f(x_0-\Delta x)}{\Delta x}$	$O(\Delta x)$
Central difference	$\lim_{\Delta x \to 0}\left[\frac{f(x+\Delta x)-f(x-\Delta x)}{2\Delta x}\right]$	$\frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x}$	$O(\Delta x^2)$

ADVANCED CONTROL SYSTEM DESIGN 20

We will go to the next concept. Many times we need a numerical differentiation. Numerical differentiation we know. This is the definition. I can either define it that way. Forward difference way – that limit delta x tends to 0 f of x plus delta x minus f of x divided by delta x; or, in a backward difference, that means f of x minus f of x minus delta x divided by delta x; or, you can find it in a central difference way. You can take positive; you can evaluate the function little positive and little negative; take the difference; divide by 2 delta x rather. So, numerically, if you want to really approximate that, you have an x 0 value. So, you can evaluate the function around x 0 either in a positive side or in negative side or both ways. Then, it simply comes from definition. As long as you take delta x as small values, if I eliminate this limit condition; that is all you are doing here – numerical approximation.

And, the limiting sense – delta x has to be very close to 0. But, instead of that, you will put a small finite value and still evaluate that derivative. That is called numerical approximation. So, forward difference is like that; backward difference is like that; and central difference is like that. And, it turns out that if you use either (Refer Slide Time: 35:51) forward difference or backward difference, the error is of order delta x. But, if you use central difference, the error will be of this order of delta square. So, obviously, central difference is better in accuracy sense point of view. But, sometimes, you will be required to use forward difference or backward difference. Like for example, if you start with grid 0.1, then the numbers are available only at grid 0.2; then, grid point minus 1 is not available, then you have to take forward difference anyway like that. But, wherever it is possible to take the central difference, my suggestion is to kind of opt for that, because the error is lesser in that.

(Refer Slide Time: 36:31)


Numerical Integration

Trapezoidal Rule:

Note: *Numerical Differentiation is "Error Amplifyin… where as Numerical Integration is "Erro… ".*

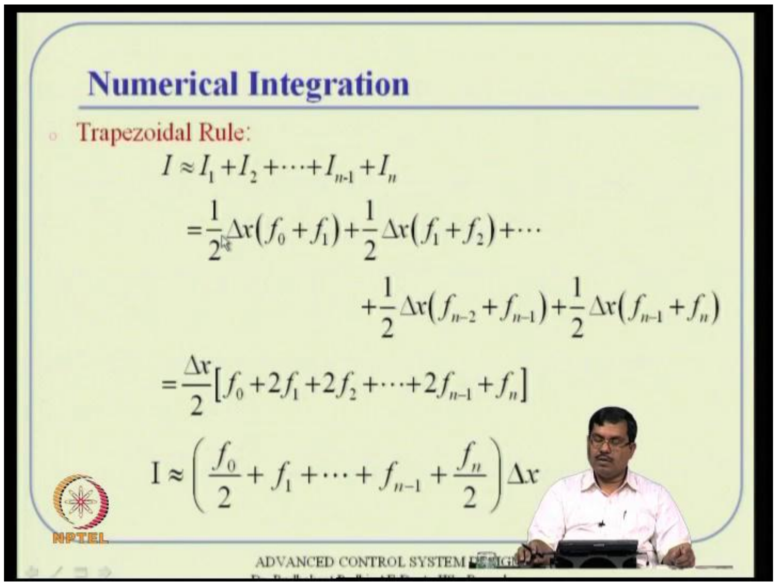
ADVANCED CONTROL SYSTEM DESIGN

Next concept is obviously numerical integration. This is all about numerical differentiation. Now, numerical integration sense – suppose you have a curve like this; whatever curve is there, I want to evaluate this integral; that means area under the curve. And, one popular way of doing that is using trapezoidal rule. One more way of doing that is also there. For example, if somebody wants to do that, they can evaluate this integral – this area and this area like that; then, this area like that. So, that means you can simply hold the values of the grid point and then try to find out whatever f 1 value at point x 1, you can just simply hold that and find out this I 1 under that. Forget this inaccuracy here, whatever happens here.

And similarly, you can try to evaluate this entire area even if you add some little more area. That is one way of doing that. But, little better way of doing that is I have these two points. This (Refer Slide Time: 37:27) function value is anyway available to me. These grid point values are also available to me. So, why do not I evaluate using this trapezoid formula, which is there. Instead of doing that, I am interpreting as something like… I will not do this;

I will just take out and I evaluate this area rather – whatever I 1. Similarly, I will carry on with I 2 and things like that. So, that happens to be better obviously; and, that is doable.

(Refer Slide Time: 37:57)



Numerical Integration

- Trapezoidal Rule:

$$I \approx I_1 + I_2 + \cdots + I_{n-1} + I_n$$

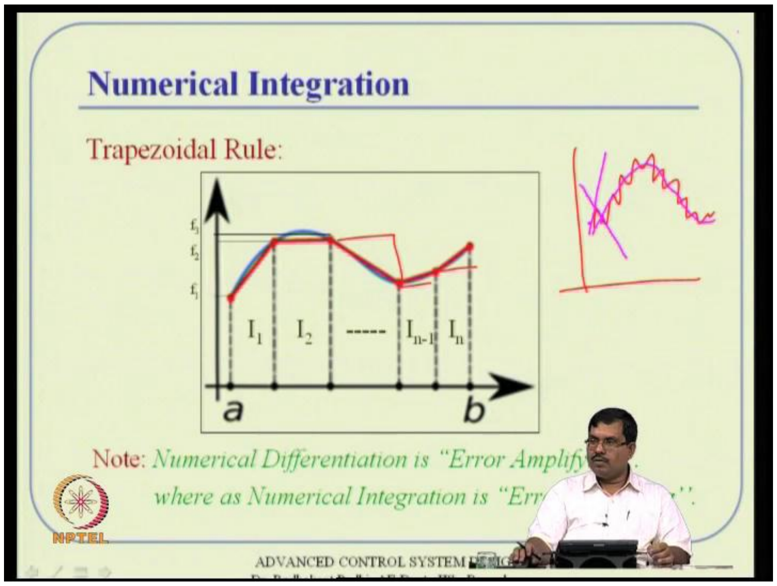
$$= \frac{1}{2}\Delta x(f_0 + f_1) + \frac{1}{2}\Delta x(f_1 + f_2) + \cdots + \frac{1}{2}\Delta x(f_{n-2} + f_{n-1}) + \frac{1}{2}\Delta x(f_{n-1} + f_n)$$

$$= \frac{\Delta x}{2}[f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n]$$

$$I \approx \left(\frac{f_0}{2} + f_1 + \cdots + f_{n-1} + \frac{f_n}{2}\right)\Delta x$$

The formula sense – I want to have this entire integration. So, I have to take summation of all these – I 1, I 2 up to I n minus 1 and I n. All these areas, whatever stripes I get (Refer Slide Time: 38:10) – little stripes I get – those stripes I have to evaluate and then take summation of that. So, that is what I am doing. Here I am assuming a uniform grid; that means delta x remains same. You can probably do that. Delta x I 1 is nothing but f 0 plus f 1. This is the trapezoidal formula for I 1. And similarly, for I 2, that is the trapezoidal formula. And, you carry on with that. Then, it interestingly turns out that if I take delta x by 2 common, then it is f 0 by 2 at the beginning and f n by 2 at the end. And, all other things are simply summations. So, if I take this 2 inside and then take delta x common here, then area by trapezoidal rule is given like this. And also, remember, there is a small comment out here that if you do numerical differentiation, it is always error amplifying; whereas, numerical integration is always error smoothing, because see if you do this numerical differentiation – these formulas (Refer Slide Time: 39:17), suppose you have a little…

(Refer Slide Time: 39:27)



Numerical Integration

Trapezoidal Rule:

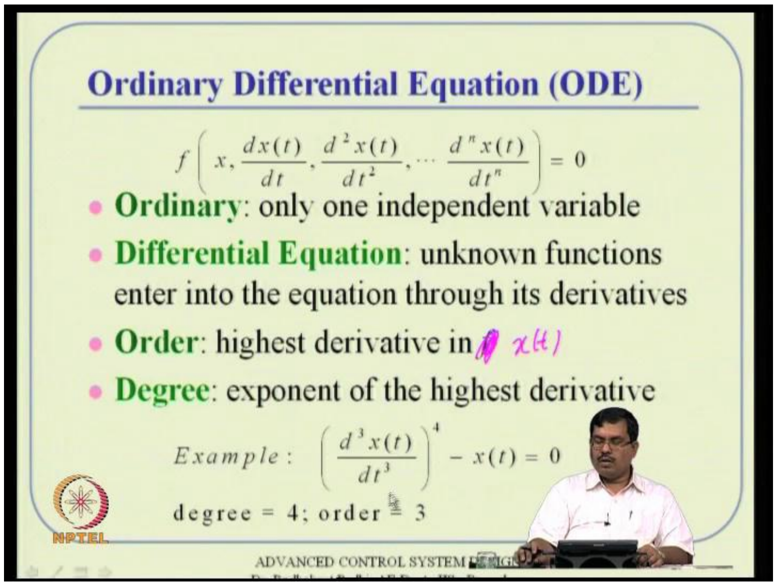
Note: *Numerical Differentiation is "Error Amplify*
where as Numerical Integration is "Err ".

ADVANCED CONTROL SYSTEM DESIG

Let us say the function has not nicely this way; but, the function has something like small error; that means the function is something like not this way, but something like this. So, now, what we will be telling? If I evaluate a slope here… I will evaluate a slope here; it will be like this. And, I will evaluate the very next time; it will be like this. Remember – the values are quite different. One is almost like plus infinity; other is minus infinity. So, that way, even the values of the functions are very small. The value at the slope at here, what I am evaluating and value of the slope here are very different; that means the differentiation operator is highly error amplifying. So, if you really have a noisy data or directly experimental data, things like that, never ever do numerical differentiation directly on those data. That is a strong recommendation rather.

However, if you do numerical integration with respect to (Refer Slide Time: 40:24) this and with respect to an average curve – let us say that is an average curve, there will be some area, which is getting added and some area which is getting subtracted. So, you do not have to really know what is the exact area value under those noisy data. But, if you do the integration, which it is rather ok; instead, it will try to smooth out the error. So, if you generate a raw data set using your experiment and things like that; if you do either differentiation or integration; do not do numerical differentiation. You can probably fit a polynomial and then take differentiation on that polynomial. That is actually a recommendation instead of directly taking numerical derivatives like this. So, these are numerical integrations.

(Refer Slide Time: 41:09)



Ordinary Differential Equation (ODE)

$$f\left(x, \frac{dx(t)}{dt}, \frac{d^2x(t)}{dt^2}, \cdots \frac{d^nx(t)}{dt^n}\right) = 0$$

- **Ordinary**: only one independent variable
- **Differential Equation**: unknown functions enter into the equation through its derivatives
- **Order**: highest derivative in $x(t)$
- **Degree**: exponent of the highest derivative

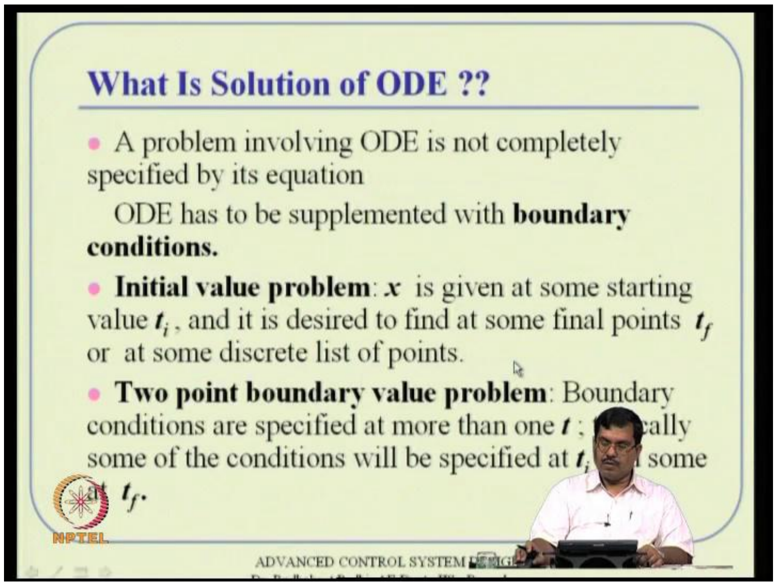
$$Example: \left(\frac{d^3x(t)}{dt^3}\right)^4 - x(t) = 0$$

degree = 4; order = 3

Then, the next concept that is needed for our kind of thing is ordinary differential equation solution. You may remember, many times we will do x dot equal to f of x 0. And, if you want to integrate those equations, and then find out what is the solution given a U basically. Now, some of these concepts were before you do that. The ordinary differential equations in general can be written in some equation like that. It is called ordinary, because it has only one independent variable, namely time here. If it is more than that, you can talk about PDEs – partial differential equations. We will not be worried about those things here; we are worried about ordinary differential equations here.

And, these are concepts called order and degree. Order of the differential equation (Refer Slide Time: 42:01) is the highest order derivative in x – highest order derivative of x of t. What you see here is the order of that. And, the degree is the exponent of the highest order derivative; that means if you have d square x square by d t square to the power cube, then it is the degree 3 basically. So, this in general is a non-linear equation. So, the non-linearity can come in higher order derivatives as well. So, you take the highest order derivative and then see what is the power of that. And, most of our problem, the highest order derivative will contain degree 1. So, that way, most of the time, we will work with degree 1. But, order can be arbitrary; that means you can talk of second order systems, third order systems, nth order system like that. So, for a small example, you can see this kind of thing. The degree is 4, because this the… First order – order is the highest of derivative, which is 3. And, this particular term has exponent 4. So, that is degree 4. So, we have a simple concept here.

(Refer Slide Time: 43:31)



What Is Solution of ODE ??

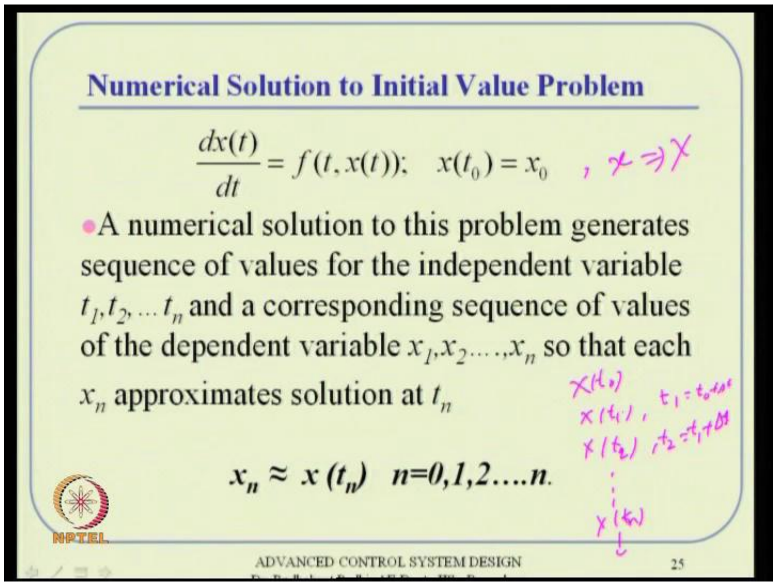
- A problem involving ODE is not completely specified by its equation
  ODE has to be supplemented with **boundary conditions.**
- **Initial value problem**: x is given at some starting value t_i, and it is desired to find at some final points t_f or at some discrete list of points.
- **Two point boundary value problem**: Boundary conditions are specified at more than one t; ally some of the conditions will be specified at t_i some t t_f.

ADVANCED CONTROL SYSTEM

What is the solution of ODE then? Problem involving ODE is not completely specified by the equation alone. You have to talk about some boundary condition, namely the initial condition. And typically, you do not call that as initial condition, because the condition can be given at any point of time; you can still integrate either forward or backward. So, it need not be only given at t 0; it can be given at t f also. But, it can be given at any point of time. It can integrate the equations both forward as well as backward using negative delta t; that is also ok.

If you talk about an initial value problem especially, then the boundary conditions are given at initial time t i; otherwise, and if all the conditions are given at any point of time either including t f, that is still called as initial value problem only; otherwise, it is something like split boundary conditions like part of the boundary conditions given at one point of time and another part of the conditions given at some other point of time. And, these problems are still possible to solve, but these are complex to solve; that means it will require some algorithms to solve two point boundary value problems. And, optimal control theory will rely some of these things. In optimal control, you will invariably be required to solve two point boundary value problems. But, in this particular class, we will talk about initial value problems only; that means all the conditions are given at one point of time. Then, how do you get that?

(Refer Slide Time: 45:09)



Numerical Solution to Initial Value Problem

$$\frac{dx(t)}{dt} = f(t, x(t)); \quad x(t_0) = x_0 \quad , x \Rightarrow X$$

- A numerical solution to this problem generates sequence of values for the independent variable $t_1, t_2, \ldots t_n$ and a corresponding sequence of values of the dependent variable $x_1, x_2, \ldots, x_n$ so that each x_n approximates solution at t_n

$$x_n \approx x(t_n) \quad n=0,1,2\ldots.n.$$

$x(t_0)$

$x(t_1)$, $t_1 = t_0 + \Delta t$

$x(t_2)$, $t_2 = t_1 + \Delta t$

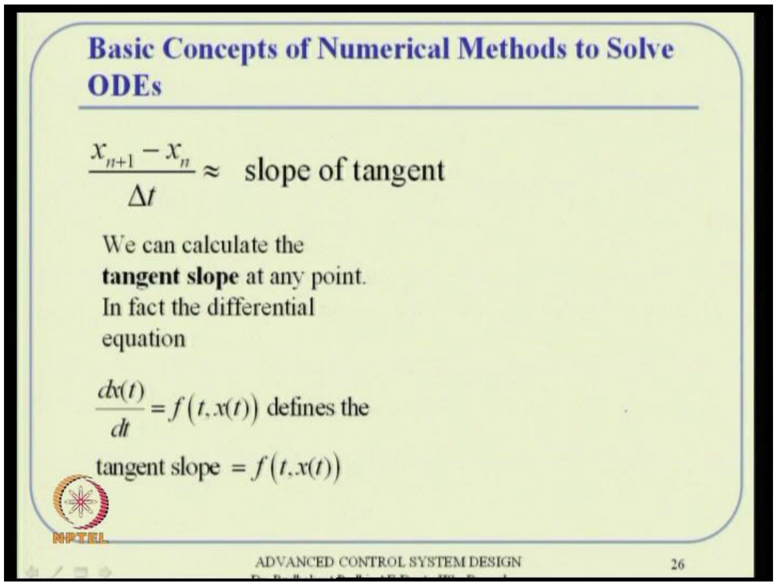
$\vdots$

$x(t_n)$

ADVANCED CONTROL SYSTEM DESIGN 25

Numerical solution to initial value problem – this is the problem eventually. dx by dt is t x of t; x of t 0 is x 0. And, remember this can be in general; this x is actually a vector x; that means it is actually x 1, x 2, x 3 – all these components are there for (( )) So, this is not necessarily a scalar equation; this is a vector equation. All these n equations are there – x 1 dot, x 2 dot up to x n dot – with all initial conditions available. Then, how do you get a solution? That means what do you mean by a solution? I want to find out what is… I know x of t 0 – I know that. I want to find out x of t 1, x of t 2 and things like that, whatever time – x of t n – I can still continue further. And, remember t 1 is nothing but t 0 plus delta t; and, t 2 is equal to t 1 plus delta t, like that. So, if I… Let us do that… delta t like that. And, these delta t's may not be same; it can be different also – delta t 1, delta t 2 – like that you can also do that. Most of the time, delta t is taken as same.

(Refer Slide Time: 46:34)



Basic Concepts of Numerical Methods to Solve ODEs

$$\frac{x_{n+1} - x_n}{\Delta t} \approx \text{ slope of tangent}$$

We can calculate the **tangent slope** at any point. In fact the differential equation

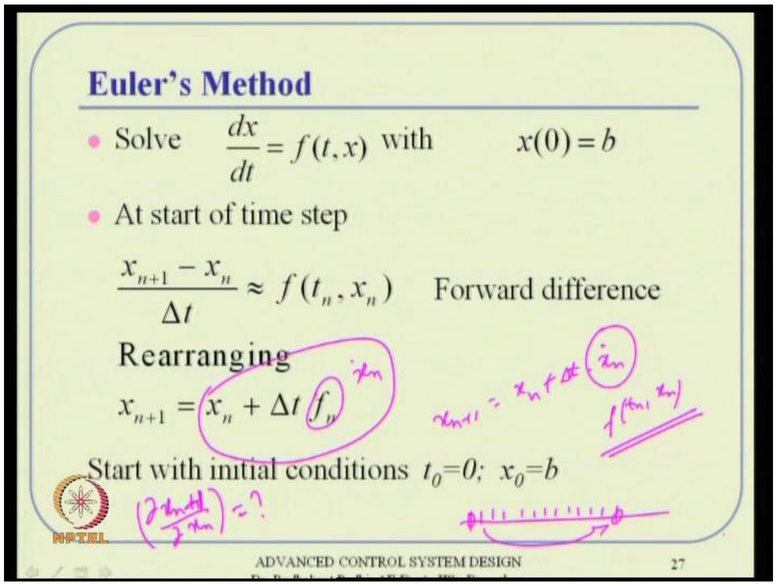
$$\frac{dx(t)}{dt} = f\left(t, x(t)\right) \text{ defines the}$$

$$\text{tangent slope } = f\left(t, x(t)\right)$$

ADVANCED CONTROL SYSTEM DESIGN 26

The first approach, what is very simple rather, is called a Euler integration in fact. What do we do here? Quickly, we will go through that. All those numerical integration methods will rely on what is called as tangent slope method. Normally, it will try to approximate the curve. Remember x dot is equal to f of t x. So, that means this is actually a function for which you can evaluate the slope at any point of that x 0 value. Suppose you know x 0, x 1 value, whatever it is and corresponding t's are known to you, then the function is completely known to you, is a function of t and x. So, you can eventually take… If you can take derivatives of that function and then take the slope of that line and things like that, and you have to exploit that. Using that the equation of the line, slope of the line and things like that, you use that information to predict the next value of the solution. So, that is what you will do.

(Refer Slide Time: 47:36)



Euler's Method

- Solve $\frac{dx}{dt} = f(t,x)$ with $x(0) = b$
- At start of time step

$\frac{x_{n+1} - x_n}{\Delta t} \approx f(t_n, x_n)$ Forward difference

Rearranging

$x_{n+1} = x_n + \Delta t f_n$ $\dot{x}_n$

$x_{n+1} = x_n + \Delta t \dot{x}_n$

$f(t_n, x_n)$

Start with initial conditions $t_0=0;\ x_0=b$

$\left(\frac{\partial x_{n+1}}{\partial x_n}\right) = ?$

ADVANCED CONTROL SYSTEM DESIGN 27

And, very quickly you see that Euler method. If you have dx by dt equal to f of t, x; again this x is vector here; that means with x of 0 equal to b, then at any start point of time, whatever point it says… Suppose I know the solution for t n already; I want to find out for t n plus 1. So, t equal to t n d x by d t, I can go back to the differentiation formula and use this (Refer Slide Time: 48:08) forward difference approximation; that is d x by d t. So, I can use that. x n plus 1 minus x n by delta t – that is forward difference. And, this is nothing but f of t n, x n. So, using forward difference formula, you can do that.

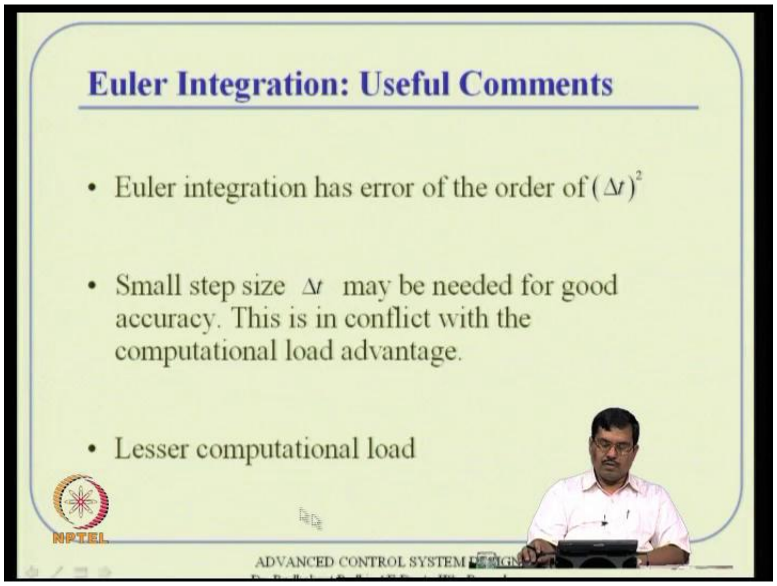
And then, you simply solve this x n plus 1 (Refer Slide Time: 48:26). This x plus n plus 1 turn out to be like that. And, remember f n plus 1, what you have here is nothing but x n dot. So, that means x n plus 1 is equal to x n plus delta t into x n dot also you can say. Sometimes people write that way. With that thing, that x n dot is nothing but f of t n, x n. So, if I know t n, x n value, I can calculate this f of x n; that is because this f is known to me; I can calculate that. And then, plug-in in this formula to get what is x n plus 1. So, it is very easy rather. Euler method is extremely easy compared to all of the methods. And, we used that in systems theories many times. But, then with the conscious information, that Euler integration is not very accurate. For getting good accuracy, you have to use delta t very small.

If you do Taylor series expansion, that is, that (Refer Slide Time: 49:27) tangent slope approach and things like that; in Taylor series expansion of this particular function (Refer Slide Time: 49:33) f of t, x – you are only retaining first order derivatives; all other terms you are throwing out. And, because of that, the accuracy is more and more better and better provided delta t is very small. And, if delta t is very small, you are going to take very baby steps – very small steps, which is typically not good in control theory either, because sometimes you may be caught with the trivial delta t. For example, delta t is 10 to the power minus 6, is 1 microsecond. And, within that 1 microsecond control of the time is extremely small. And, in my knowledge, I have never seen a control update frequency of that small. Delta t being 1 microsecond, I have never seen; it is all like millisecond level I have seen. So, you cannot do that if that problem demands that.

Now, second issue is suppose you want to do it in a faster way, tell that you can argue that all other numerical method; we are going to see one more by the way. It will demand little more computation for every iteration. This is the (Refer Slide Time: 50:35) smallest amount of computation by the way. So, computational advantage I will have. However, suppose the other method gives me relatively larger delta t, I can use that, because accuracy will be high. And, here the accuracy is being less, I have to use lot of these grid points. So, I have to repeat this computation so many times before I go from here to (Refer Slide Time: 50:55) here; whereas, in other method, I can go directly there; that means even if I am using little more computation for every step, I have to work with less number of steps. So, these are all kind of advantage, drawbacks, things like that.

And, another advantage of Euler method is because the formula turns out to be rather easy; you can do further algebra rather easily. For example, if you discretize this equation this way – this is the discretized form of this (Refer Slide Time: 51:25) continuous equation, then you can talk various derivatives of this. For example, you can talk about let us say del… Sometimes we require to take derivatives like del x n plus 1 divided by del x n. Suppose you want to take that, then using this formula, it is rather easy, because f n is this way (Refer Slide Time: 51:49) and you can take out all the derivatives in a long-end formula – close form formula. That will not be difficult. So, there are advantages, drawbacks with this Euler method. My suggestion is just use it cautiously; do not jump into that unnecessarily.

(Refer Slide Time: 52:06)



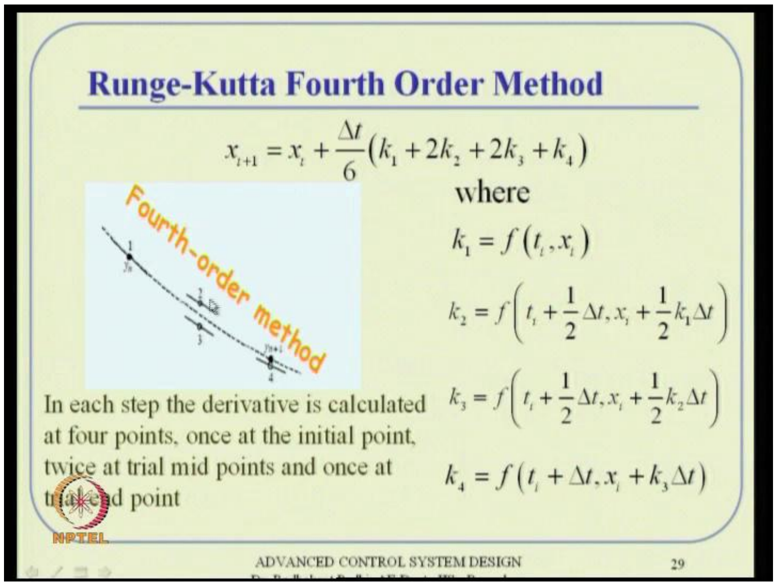
Euler Integration: Useful Comments

- Euler integration has error of the order of $(\Delta t)^2$
- Small step size Δt may be needed for good accuracy. This is in conflict with the computational load advantage.
- Lesser computational load

ADVANCED CONTROL SYSTEM DESIGN

Some useful comments – I have already told some of those. Euler integration has this error is order of delta t square, is not very highly accurate. So, small step size is necessary. And, this can come in conflict with computational load advantage. Every grid point you are doing less computation; but, you have to take several grid points, because delta t – you will be required to select much lesser. So, that way it is not that advantageous. But, in general, it has computational load; it has also like close form; further algebra is possible in close form like that.

(Refer Slide Time: 52:44)



Runge-Kutta Fourth Order Method

$$x_{i+1} = x_i + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

where

$$k_1 = f\left(t_i, x_i\right)$$

$$k_2 = f\left(t_i + \frac{1}{2}\Delta t, x_i + \frac{1}{2}k_1\Delta t\right)$$

$$k_3 = f\left(t_i + \frac{1}{2}\Delta t, x_i + \frac{1}{2}k_2\Delta t\right)$$

$$k_4 = f\left(t_i + \Delta t, x_i + k_3\Delta t\right)$$

Fourth-order method
1
2
3
4

In each step the derivative is calculated at four points, once at the initial point, twice at trial mid points and once at trial end point

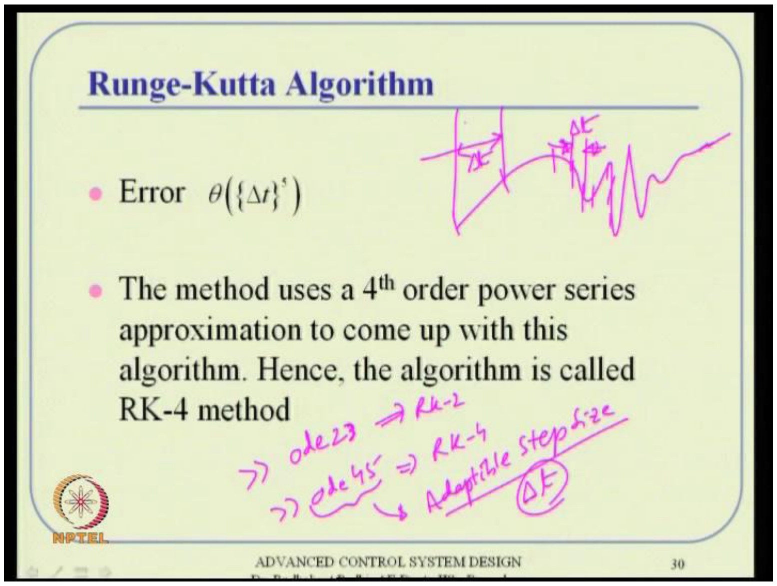
ADVANCED CONTROL SYSTEM DESIGN 29

Now, another thing that comes to method is Runge-Kutta method. And then, Runge-Kutta methods are of various orders. It can talk about second order Runge-Kutta, third order, fourth order, like that, which is very popular in systems theory. And, most of the differential equation numerical solution is fourth order. And, fourth order solution essentially the concept is like that; derivation I will not give here. I will talk about some function that f of t n, x n, is something like this let us say. And, you have some value like that here. Then, it attempts to find some approximate slopes here. Point number 1 – you already have the slope; it predicts what is the slope at point number 2, point number 3, point number 4 and then tries to kind of fit a polynomial in between that. And then, evaluates that (( )) using this polynomial, you will be able to kind of march ahead with the solution here with much more accuracy. That is the philosophy.

Algebra part and all – you can see numerical methods book. So, ultimately, what it tells me is I start with this (Refer Slide Time: 53:50) series of computation; I start with k 1 rather, which is simply the function evaluation there, whatever function I have. Then, using that function value, whatever function value k 1, I will be able to calculate k 2. And then, using this k 2, I will calculate this k 3; using this k 3, I will calculate this k 4. And, using all these k 1 to k 4, I will just try to kind of get delta t over 6 and then average out all these – k 1 plus

2 k 2 plus 2 k 3 plus k 4. So, 1 plus 2 plus 2 plus 1, that is, 6 divided by 6 sort of thing. So, this entire formula – the final formula – x i plus 1 is given something like this.

(Refer Slide Time: 54:36)



Runge-Kutta Algorithm

- Error $\theta\left(\{\Delta t\}^5\right)$
- The method uses a 4th order power series approximation to come up with this algorithm. Hence, the algorithm is called RK-4 method

Δt
Δt
>> ode23 ⇒ RK-2
>> ode45 ⇒ RK-4
Adaptible step size
Δt
ADVANCED CONTROL SYSTEM DESIGN
30

And, essentially turns out that fourth order Runge-Kutta algorithm has error – delta t to the power fifth; that means it is highly accurate compared to that. Remember Euler integration had accuracy of the order of delta t square only. Now, that square has gone to power fifth. That is why it is much more accurate. And, the second comment is this method essentially uses a fourth order power series approximation. Remember that Euler integration uses only linear term. Now, this one uses up to fourth order power series approximation in Taylor series to come up with this algorithm. Again, details of that is not here; you can see some numerical methods book. And, this particular algorithm is popularly called as RK-4 method – fourth order Runge-Kutta method. This is very popular in matlab; you can use all these.

And, probably for Euler, you do not need an integration formula; it is just that, that formula is so simple that you do not need any routine for that primarily, because this integration formula (Refer Slide Time: 55:45) you can simply write it longhand. Once you evaluate this f of x, then it is easy to just plug in there. So. they do not need formula per se. Or, this algorithm (Refer Slide Time: 55:56) requires little bit of sequential computation. So, the formula is available there. And, in matlab, you can see some of these. This is something

(Refer Slide Time: 56:09) called ODE 23; that is the function. And, there is another function, which is called ODE 45. And, ODE 23 – it is this what is called as a RK-2 method – second order Runge-Kutta method. And, this will give you RK-4 method – fourth order Runge-Kutta method. Essentially, both are similar functions, but with difference that these are there.

And, by default, whatever is there in matlab, it is what is called as (Refer Slide Time: 56:37) adaptible step size; that means delta t that you are using does not necessarily remain constant. And, it is adopted provided suppose you have function something like these, and then, for here the slopes are not changing fast; here the slopes are not rather changing that fast; but, here in this segment, it is changing rapidly. So, it will try to see this – where the slope changes happen in a rapid way; and, if the change of slope is high, then correspondingly, it will adjust delta t small. Remember – this is delta t. This delta t is small here; this delta t can be large here. So, this adaptation of delta t happens in an implicit way inside the algorithm in these ODE 45 routines and all that. Most of our type of applications we do not… We are ok with constant step size. So, we can probably write (Refer Slide Time: 57:41) these functions ourselves also many times. And then, see what is the delta t that we can select. And, (( )) size is also very good for implementation concern; you do not have to monitor what is the delta t. Every control cycle update, gradient cycle update, like that – that is fixed by some number. So, you start using that particular number in a constant step size way. So, that is, some of these numerical methods are useful here.

With that, I will probably stop here this particular class. And, you see that root finding both in linear equation and non-linear equation, numerical derivative and integration as well as numerical integration or differential equation. That is all we discussed in this class. These are all useful in our control theory. And, many of these practical uses implementation will come from your experience as you work with different different problems. I hope this will be useful in your further exercise. Thank you.