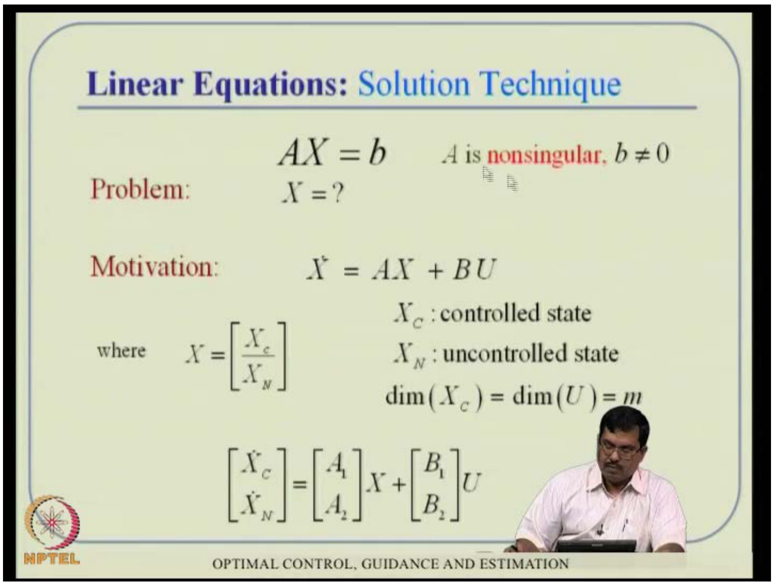
Optimal Control, Guidance and Estimation

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

Lecture No. # 03
Review of Numerical Methods

We will continue further in our lecture series and many times in our control theory as well, we will need lot of numerical methods on the way, especially with respect to non-linear systems theory the things are not available in close form (( )) to that and in linear systems. Also, remember, that we need some operating point first before we talk about deviation dynamics, which we linearize and then interrupt, that is, linear system. We will talk about linearization in one of those lectures later, I mean, about the linearization method in its useful. And then we will see some of these concepts, other concepts also like root finding and other things are useful in lot of application actually.

(Refer Slide Time: 01:17)



So, let us quickly review some of these numerical methods that are frequently, I mean, kind of required in our analysis and synthesis actually. So, we will, first concept, that comes to my mind is, first problem is linear equation solution. Suppose, you have A X equal to b and remember, we do not want, we know for sure, that if A is some rectangular and things like that, I mean, A, A is square matrix for which determinant of A is not 0. That means, A is non-singular and b is not equal to 0, then X equal to A inverse b, we know for sure, but A inverse is (( )) 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 actually.

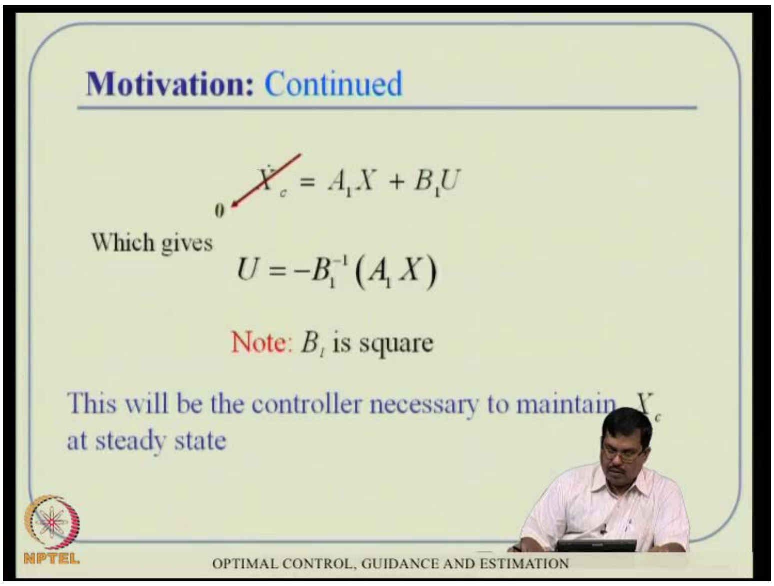
Now, the question is, why do we require this kind of solution in our control theory? And one of that is you can see some of these, I mean, some of these motivation part of it. Suppose, we want to find out something like, like this, this is a typical equation, X dot equal to AX plus BU and we, let us say we divide this X into 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, that upper portion of the equation, that means, X c dot is equal to A 1 X plus B 1 U. Then, I tell this B 1, remember the dimension of X c is equal to dimension of U; that means, B 1 is a square matrix now. Then, I will probably try to find out what is (( )) equilibrium condition? That means, this X c dot is 0, right. If I take about the controlled state equilibrium and I want to interrupt that as a forced equilibrium point, that means, my control (( )) is not 0, so that 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 actually, like if your aircraft flies (( )) level, you really require some sort of a trim movement because remember, your (( )) and C p are not at the same point. That means, there is continuously, there is a gravity related. I mean, this aerodynamic lift related movement, which will act on the vehicle actually. If I, if I just turn that in a little small diagram, see something like there and your (( )) is somewhere here and C p is somewhere there. So, that means, this, this particular thing will give us some sort of a movement about that actually. So, this is turning movement sort of thing.

So, you want to cancel that by using something like an equivalent movement actually. So, that means, you really require some sort of a small elevator deflection actually. So, those control trim condition is actually a post equilibrium point in aircraft plane and those (( )) cannot relate to that, we will see that later. And that is very obvious, it is not too much, you can see that this problem from other example problems as well.

(Refer Slide Time: 04:16)

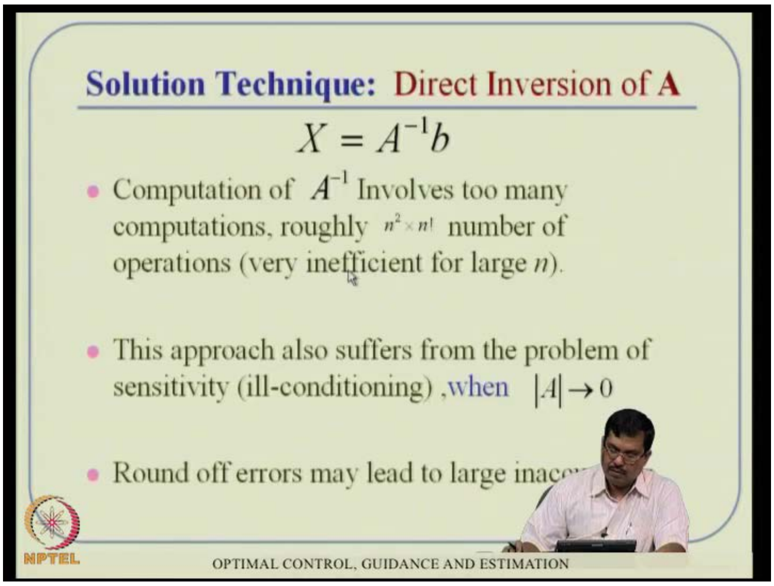


So, what I am looking at? I am looking at some solution for a, for something like X c dot is equal to 0. That means, the control states are operating under steady state, actually. If I really want to find out what is my control necessary for that particular situation, then I can find a control (( )). That way, if I remember, B is actually, B 1 is actually non-singular, square and non-singular. So, I can take that way.

If I apply this control equation, (( )) control, then I will ensure, that this happens to be 0, so that my, my aircraft continues to fly at (( )) level, that particular control is actually necessary for maintaining X e dot 0 (( )), X e is X dot for example, and BU is delta e, elevated deflection. Then, this elevated deflection I have to compute it that way, so obviously, we require this tabulation.

Now, this is A 1 of X, this is nothing, but our B, if we are going back to that, this is our B and this A (( )) is nothing, but B 1 actually, in this case. So, we require this kind of solution, I mean, efficient way of getting solutions 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 efficient way of controlling, I mean, finding out this B 1 inverse, actually.

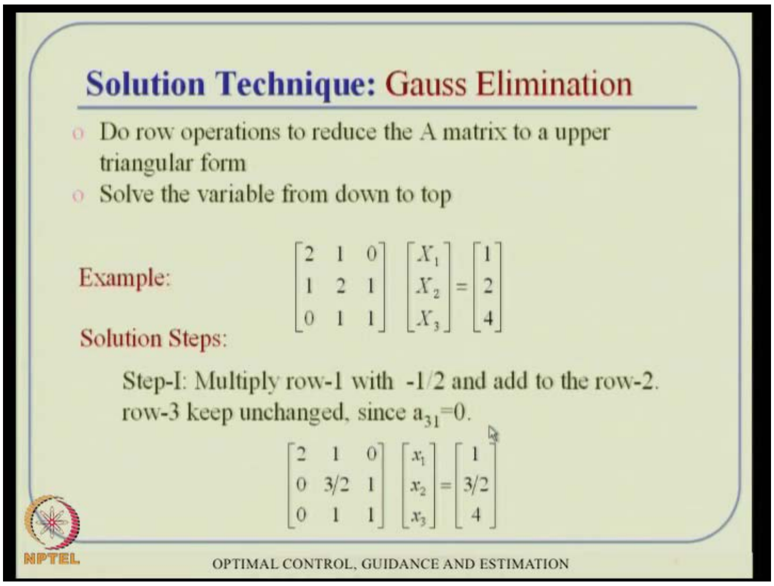
(Refer Slide Time: 05:30)



So, coming back to that, can we... So, this is the equation, that you are asking and then we observing, that A inverse involves too many computations actually and (( )) computational complexity sense, it requires n square into n factorial computations. That means, if A is larger and larger and larger, you are getting trapped in this computational actually. So, you want to do it in an efficient way and one of that efficient way happens to be like (( )) elimination, which we will see in this class, actually. Now, this also, this approach is not just suffers from the computational headache, I mean, it also suffers from these, these other problems, like what is one thing is called ill-conditioning, remember A inverse is nothing, but, but adjoint of A divided by determinant of A; A inverse is equal to adjoint of A divided by determinant of A. So, determinant of A goes to 0.

That means, if you have a singularity is approaching, you are approaching singularity, then this is a serious problem, actually. You cannot just talk about division (( )) actually. And also remember, there will be like so many computations and lot of computations, all these computations will have small round off errors, the too many computations, then round off errors can be large actually. 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, that number, you know, that finite digits only, then round of errors all the time, actually, so (( )), as you do not want to do too many computations, actually.

(Refer Slide Time: 07:11)

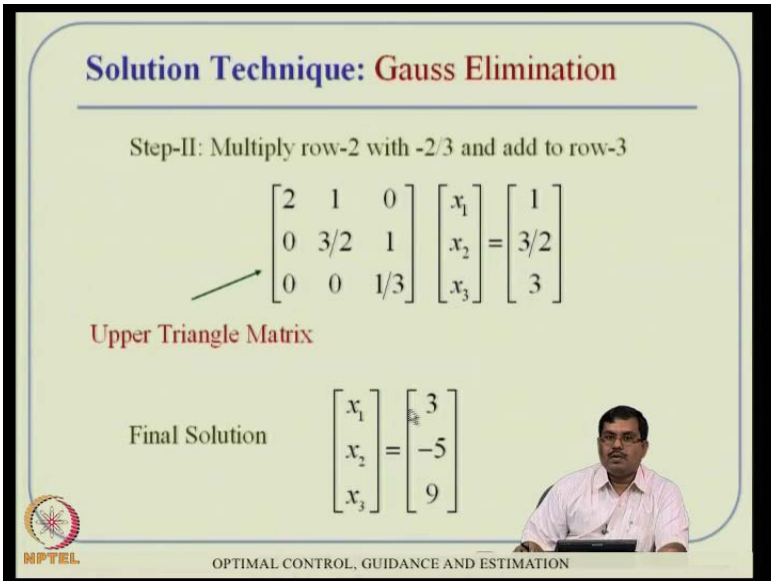


Now, Gaussian elimination is a substitute to that instead of general things. We will just take it as a small example and try to see what is going on here. So, we have this kind of an equation, let us say 3 by 3 and then, I mean, skip all these again, again small mistake here, all these are small x, x 1, this is small x 2, small x 3, these all are elements, actually. Anyway, 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, actually. And we take advantage of the fact, that you can multiply by any row with, by any constant and then take addition of (( )) of other row, then the equation will not change actually, equation remains same. So, then, we will take advantage of that thing and then, tell we want to reduce this matrix to a upper triangular form actually by taking advantage of this row transform, I mean, row properties, actually, like row multiplication, addition, things like that actually. So, what you do actually? We will try to see the, this, these elements, whatever you see here, 1, 1, 0, this has to be all zeroes actually. Now, 1, 0, already there, so I want to make sure of this 0 also happens here or this 0 happens here, either way.

So, now I see, how do I multiply this particular thing by minus half, one-half and then add it off, actually. That means, I want to make sure, that it becomes 0 itself. If I do that, then 2 by 2 is 1, then 1 minus 1 is 0, basically. So, that will pop out there corresponding to that. Now, you say 2 minus one-half actually, again 2 minus one-half is 3 by 2 like that actually and 0. We will not change this, that is, the equation that I will leave it actually. Again, 2 minus one-half is 3 by 2 actually here. Now, you see, that this is already 0, 0, but, 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 actually.

(Refer Slide Time: 09:20)



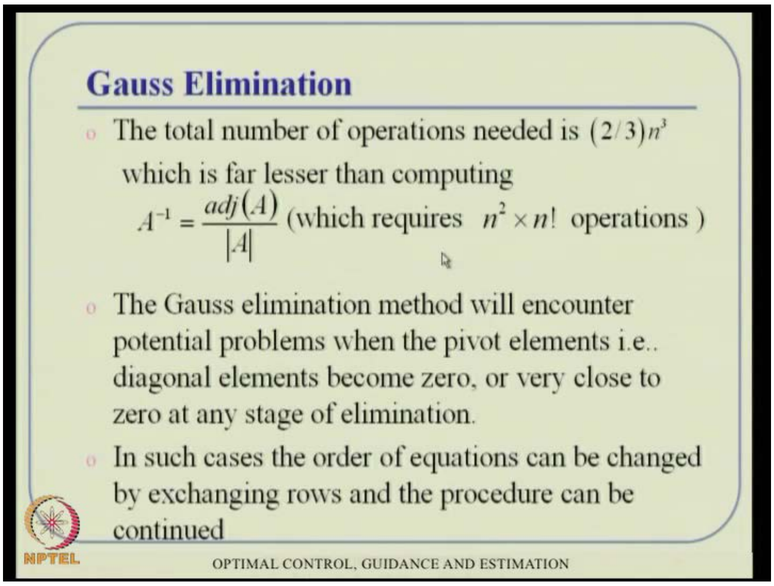
If I do that, then it will, I will end up with something like this actually. Now, this is an upper triangular matrix, so we start looking at the equation from down to top, actually.

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 is nowhere and I mean is a function of x 3, this particular, remember this let us do that, this, this particular thing has given me x 3 is equal to 9. Now, if I do this equation, rather second, 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, I tried to solve this and if I try to solve, then x 2 happens to be minus 5, actually.

Now, I go back to the first one equation. Now, first this, this equation now and tell, this is 2x 1 plus x 2, x 2 is already minus 5, so this minus 5 equal to 1, right. So, this, then 2x 1 turns out to be 6 and x 1 is equal to 3, that is why, you get x 1 equal to, 3x 1 is equal to 3x 2 equal to minus 5 and x 3 is equal to 9, that is easy now. So, what is, what are you doing here? You are somehow trying to eliminate these, these are 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 at the equations from bottom to top because then they try to solve, solve this equations straight forward actually. So, that is Gauss elimination.

(Refer Slide Time: 11:07)



And now, Gauss eliminations suddenly make this n square into n factorial operations, this reduces to just this much operation actually. Remember, this, 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 computationally.

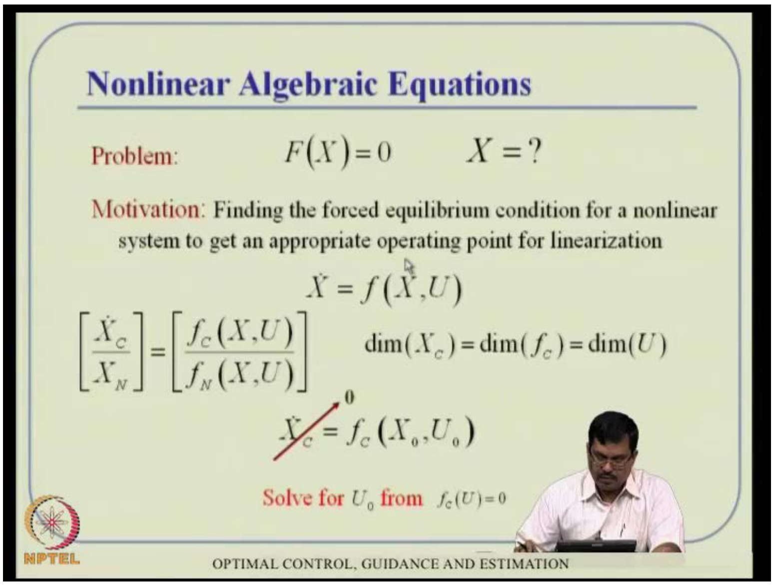
So, obviously, it leads to far lesser computation and then, hence it is kind of referred actually, especially, especially in metal of command window, if you, some of you use that, then you have a choice of using INV function, INV into B, which is like X or the same thing you can do that using this slashes, a slash, that will give you Gauss elimination, actually. So, instead of using the first one, I will suggest, that you use this one actually, A slash B, alright.

All is, all is well here, but, but you see some problems actually. So, first problem is, so this, this Gauss elimination method will also encounter problems if the pivotal elements happen to be zero, actually. I mean, that one of the diagonal element becomes, suppose this is zero, now, sorry I took it there, suppose if you do this exercise and this happens to be zero, now no matter how much you multiply and try to add and subtract, nothing will happen to this one actually, not be able to do that.

So, in this, those situations, one easy way to do that A is just to exchange the rows actually. Suppose, if it, this one happens to be 0 here, then you take you take the entire equation, you put in a 3rd equation and this one is substitute for 2nd equation, you exchange the equations actually, then proceed further. That is the idea there, actually.

So, this Gauss elimination is a very standard practice now and lot of people make use of that again, a metal of window, I mean, metal of command, metal of environment. I suggest that you use this, this one. Anyway, now going back to the next concept.

(Refer Slide Time: 13:27)

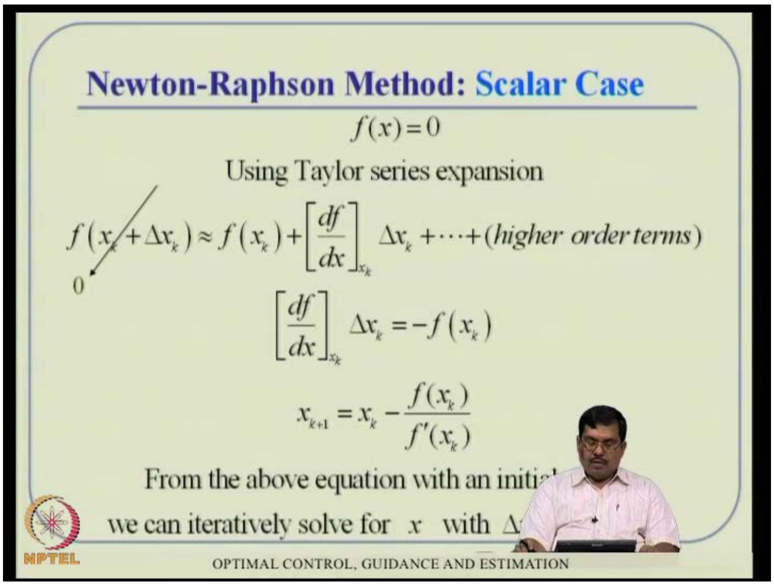


What we saw there is algebraic equation solution for linear systems. That means, we are talking about, this is a linear sort of equation actually, right. AX is equal to B, what you, what you have is actually a linear equation. Now, what if you have a non-linear equation? Remember, numerical methods are more powerful in linear cases and all that, alright.

Now, let us see something like this, F of X equal to 0, and you want to find out what is X equal to? And again, the similar motivation there, like you can, I can partition this state's, whatever X dot equal to f of X, U into two parts, I still maintain that condition, dimension of X c is equal to dimension of U. Then, in this case you land up with, like X c dot equal to 0. If I want to entrust that question, then I have this non-linear, 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.

Now, that means, now can I find out some U 0, which will satisfy this equation, this equation, this can be a function of X 0, that is, 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 and that is why this 0, 0 notations are there, actually. So, I will go back to this equation now, f c is of 0, f c of U equal to 0, and try to find out what is use for that. That is the motivation why want to address this, why we want to address this particular problem actually. This is just one of the examples why you need that. You may need for many different cases also actually.

(Refer Slide Time: 15:17)



Now, we will start with the simple solution sort of idea. First, we will see what is a scalar case? By the way, this has many, many numerical methods available, one of that, which is popularly known is bisection method. That means, if you, if you talk about the scalar expression here, for example, and you tell I have two guess values now, then there is something called bisection method and there are very, very other ideas as well, actually. So, this is not what I am talking here, is not an exhaustive review of methods available. So, what I am doing really, doing here, is rather talking about what is called as Newton-Raphson method, which is, which is very neat, it has its own beauty physically and that is what is mostly used in practice. That is why we want to review this particular method of getting the solution actually. So, what you, what you really want to find out in this case? Remember, non-linear equation solution will require some sort of iterative solutions actually, that means, in one go we will not be able to find out, but, but we may need some couple of iterations before arrive at the final solutions, actually. So, what is the idea here?

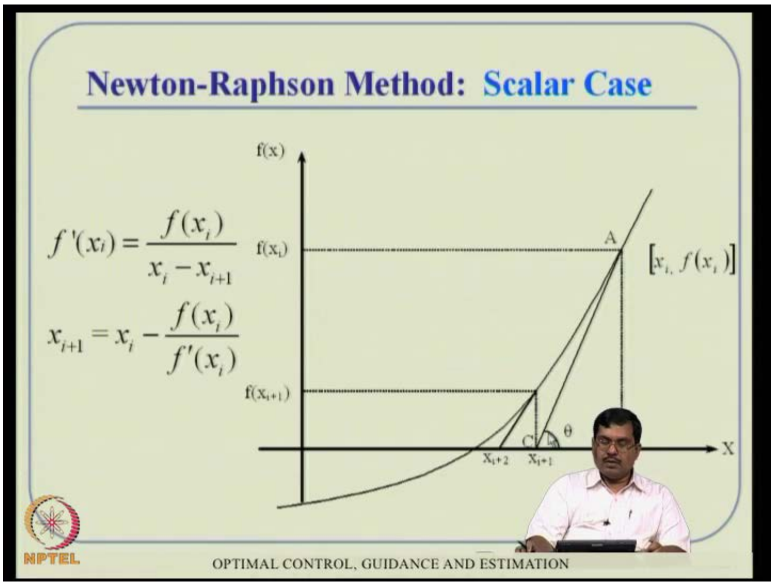
Now, let us say that we are having some x k value, some value for x k is already available and k can be 0 also, that means, you can start with some initial guess, then if k is 1, 2, 3, 4 than the already iterated values available. Obviously, that is not the solution; that is why we want to do iteration actually. That means, f of x k is not really 0. However, you want to find out delta x k such that if I make x k plus delta x k, then f of x k 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 actually and that is the whole idea there.

Now, I will interpret this, f k, f of x k plus delta x k in Taylor series expansion and tell these are all higher order terms and all that, I will not be interested in that, I will suppress that and I will just take the 1st order term, up to 1st order term and this is what my equation actually. Remember, this entire expression is zero because that is supposed to be a root actually. That is what you want to find a delta x k, such that f of this x k plus delta x k equal to zero basically. So, that is your aim actually, that is, how we want to find a delta x k. That means, this is equal to 0 up to that actually, up to this term, then I can, I can relate, that delta x, this particular term is zero. Hence, this del f by, I mean, 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 and assuming that is f dash of x, remember that, evaluated at x k.

So, assuming that, that is non-zero, I can do that actually. Delta x k is nothing, but this term, this whatever term you see here, that is nothing, but, but delta x k actually. This particular term is nothing, but, but delta x k. So, if this is delta x k, then this is my, what I really wanted is x k plus delta x k, so that is my x k plus 1. So, this is, this is this particular term. What you see here is x k plus 1; that is why if f of x k plus 1 is 0 basically right, that is what you want it. 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 actually.

So, x k, x k plus 1 is equal to x k minus f of x k divided by inverse of x k, that is the, that is the algorithm and you keep on doing that until you get some, some convergence. That means, that you do not get too much, I mean, very much of improvement afterwards actually, that is, up to that you do that and stop.

(Refer Slide Time: 19:31)

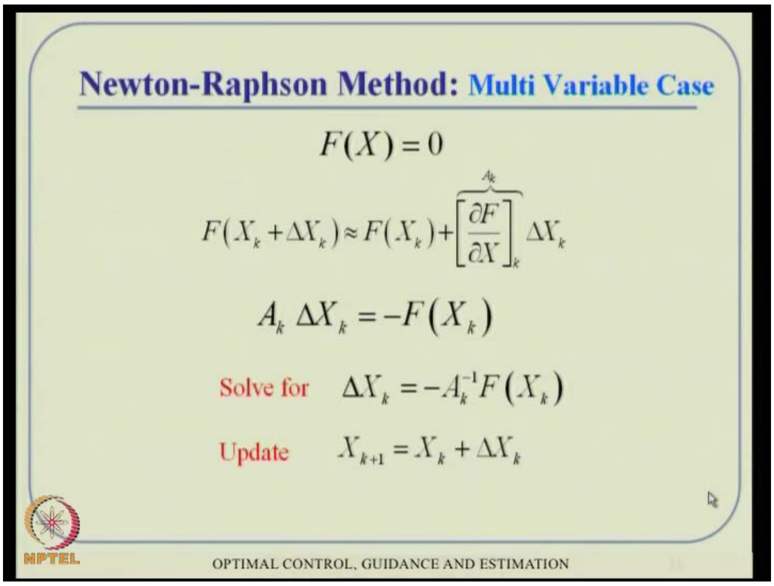


Now, picture, really, let us see what is going on here. You have sum f of, sum of x, sum of f x, arbitrarily it is plotted something like that, f of x is equal to 0 and assuming that, that is the zero-axis of course, of course it starts with 0, 0 probably, and then obviously, what you want? You want this particular value, this particular value, this is your root actually, this is what you really want. Ultimately, what you do not know? The value to start with, actually. So, you start with some sort of a guess value, this is my guess value, I will start with some guess value, let us say, then f of x is that most, that is my f of x, then f (( )) of x. f (( )) of x is nothing, but the slope actually, evaluated at x. So, that is the slope what is giving me actually.

So, if I compute this delta x i sort of thing, and if you see this is actually a linear equation. So, what you are doing here? x i plus 1 is x i minus this term actually and if you just carefully look at it, it is kind of a, to start with this equation line, I mean straight line equation and try to find out the interpretation, the meaning of that. That means, it all that tells me, that I have a guess value, I go to that particular point where it intersects this particular function, then evaluate this slope and with that extension of this slope I will cut this x, I mean, this x-axis somewhere and that is going to my next iteration value.

So, again, I evaluate the function there, I again take their local derivative, extend that and wherever it cuts, it is my i plus 2. Then, again I put, go there and I will proceed further itself like that and ultimately I will converge there actually. So, that is the interpretation of that.

(Refer Slide Time: 21:24)



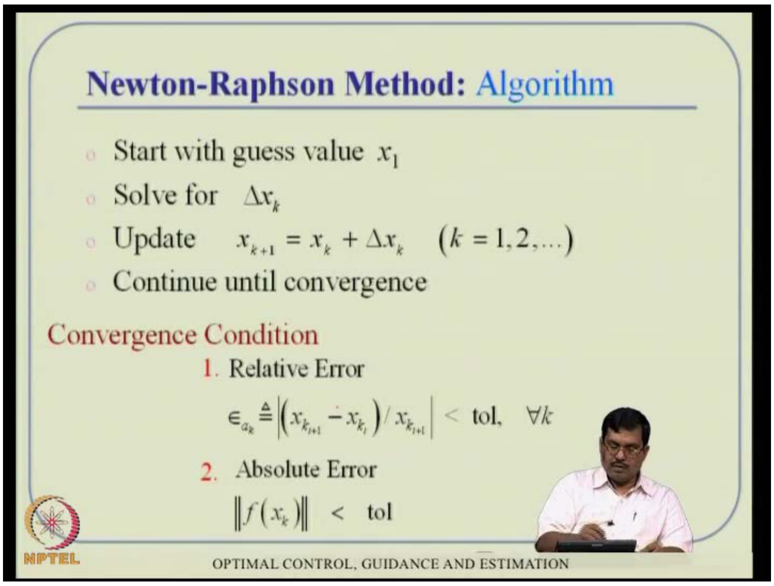
Multi variable case, this is similar extension we can do. Now, if you want this f of x is equal to 0, you tell, while I will do the same exercise. That means, I will take multi variable Taylor series in expansion and then I will tell, I mean, higher order terms I will neglect and I found I want to find output delta X k in such a way, that, that X k plus 1, this is my X k plus 1, anyway, this is my X k plus 1, that is going to be a root. That means f of X k plus 1 is going to be zero, actually.

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 A k plus 1, I define this del F by del X evaluated at k is nothing, but, but A k is something like that and then, the equation tells me, that A k times delta x k is nothing, but, but negative of this particular thing, F of X k. And hence, delta X k is nothing, but A inverse, A k inverse of F k. And remember, when you see, whenever you see this kind of a thing, you can use, you can use Gaussian elimination. You can use Gauss elimination actually to compute that in a faster way, then you can update and then you get X k plus 1 is X k plus delta X k.

So, very similar to what you have here, instead of divided by, if you had 1 by, like (( )) X k. In the matrix you cannot talk, that one, one over matrix. So, that is not defined. They will tell it is, it is negative of A k inverse multiplied by that, that is the only difference, otherwise it will continue. And all these are the backbone of, I mean, every, all numerical method, most of them will always rely on Taylor series expansion, whether it is root finding, whether it is (( )) integration, or many, many things, will all reply on Taylor series expansion. So, all these numerical methods derive the strength from Taylor series expansion theory, alright. So, the algorithm is again same, you start with some guess value and then compute delta X, that like delta X k like that and then proceed algorithm like that actually.

And remember, in (( )), those of you who are using that, I mean, we will be using that for scalar thing, you do not have to do yourself. If you want you can do it or otherwise in (( )) there is a function called f 0. If you, if you use this f 0 function it will try to find out a solution around the guess value, that you have to give actually. By this functional demand, a function, which is written and then there is a guess value, it will demand actually. So, it will find out a solution around this guess value actually. Similarly, you can write, I mean, this multi dimensional root, I think, to my knowledge is not available directly, but from optimization tool you will have something is available, otherwise you can write own function, as also, actually this thing you need to write anyway.

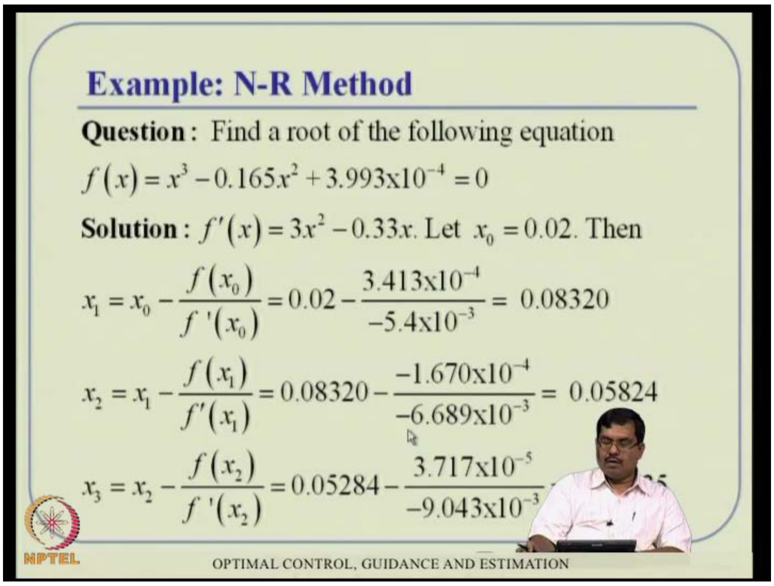
(Refer Slide Time: 24:40)



So, 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 (( )) we will start with X 1, the index 0 is not defined (( )). So, you start with some value X 1 and then solve for delta X k and then update your X k plus 1 is like this actually. And then you continue until convergence and convergence typically, it is given either in term of relative error or in terms of absolute error, given a choice relative error is, is my 1st preference actually, I mean first recommendation because your problem relate, that the tolerance value, that you select here or there, it is not problem dependent actually, it will be fairly independent.

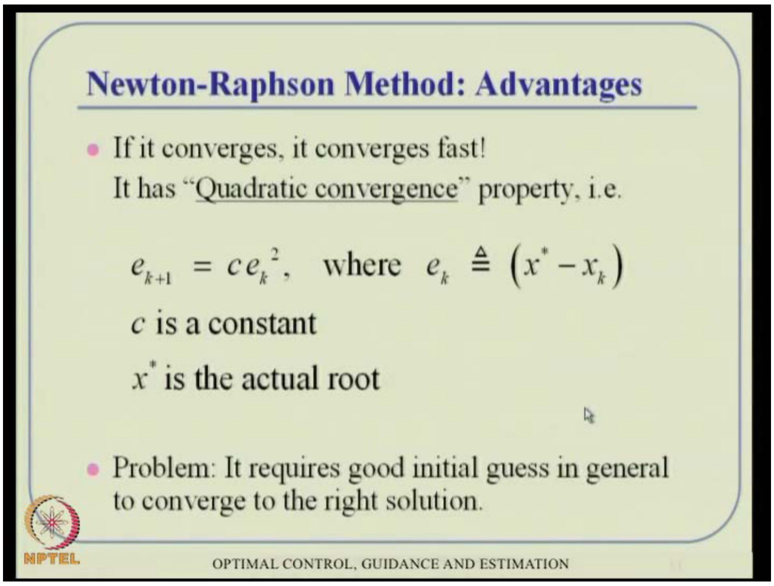
And if you talk about 1 percent error, the 1 percent error, whether you have a number sense problem or you have a rocket science problem, either way it actually. So, in other words, the units can be very different, still the relative sense is still 1 percent error. So, that is why, most of the time I will recommend relative error actually. And absolute error are sometimes necessary because suppose, if you have this particular x k plus 1, what you see here is 0, but then there is a problem of division. So, internal, ok I will continue with absolute error because I already have a physical idea of what these values are. So, that will help me in selecting a proper value for this absolute error actually. That will be always my second recommendation; first recommendation is this one actually.

(Refer Slide Time: 25:59)



Now, let us see Newton-Raphson method, how do you find that, find a root of the following equation let us say. Then you start with some, some x 0 value as 0.02, that you can start with that and then you tell, this is my x 1 and (( )) is nothing, but x 0 minus this and then x 2 is that, and x 3, that you can continue. And you can see, that there is some improvement here, 0.08 to 05, three digit, I mean, 08 to 05, this 0.03 is sort of improvement here and then suddenly, it is 0.01 improvement. And you do one or two iteration, it will converges itself actually. That is the method; that is the thing.

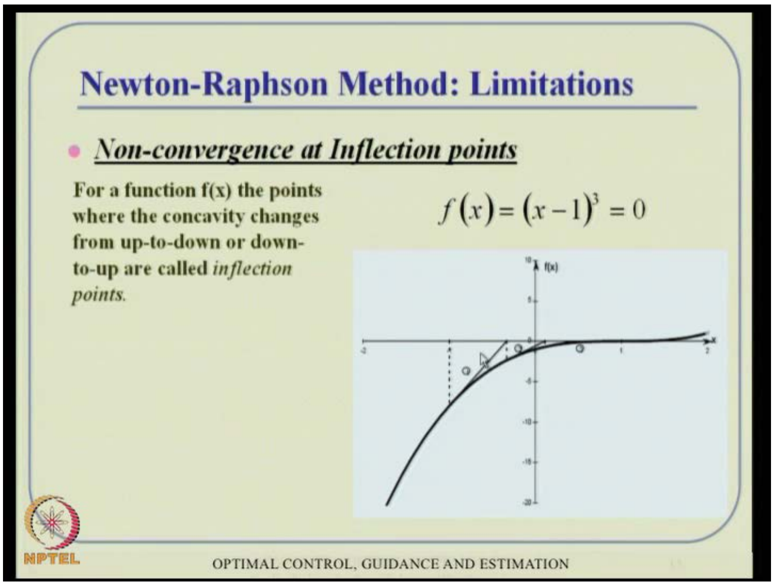
(Refer Slide Time: 26:54)



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

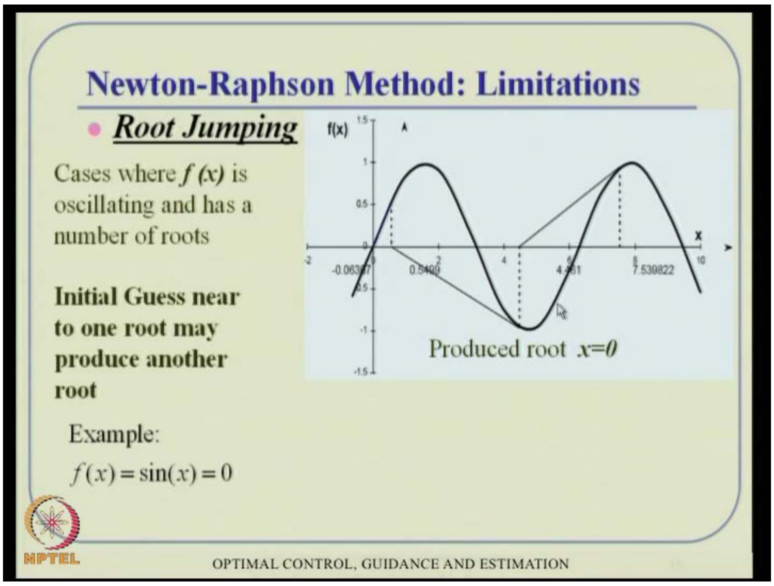
Typically, these are all less than 1, e k 1 and then, if it less than 1, e k square is still further less than 1, I mean, is very small value. And e k plus 1 will be a function of, I mean, it is proportional to e k square actually; e k plus 1 is proportional to e k square. That means it will converge very fast actually. That is the square convergence, quadratic convergence property actually. However, there are several problems here and one of the problems is, it really 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 itself actually; there is a possibility of doing that.

(Refer Slide Time: 28:17)



So, what are the problems of this method? First of all, we will see couple of misuse here and first thing, first thing is this particular fellow, I mean, Newton-Raphson method does not converge at inflection points actually. In general what is the inflection point? All powers actually, 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, we are looking for this solution, but, but it will never converge actually. At this point of time it will, if you just do iteration starting from guess value, you will see, that this is some convergence (( )) actually. It turns out, that it is point of inflection, something like that. If it is (( )) it is fine. Point of inflection, normally it is some convergence, it will go close to that, but will not converge actually.

(Refer Slide Time: 29:08)

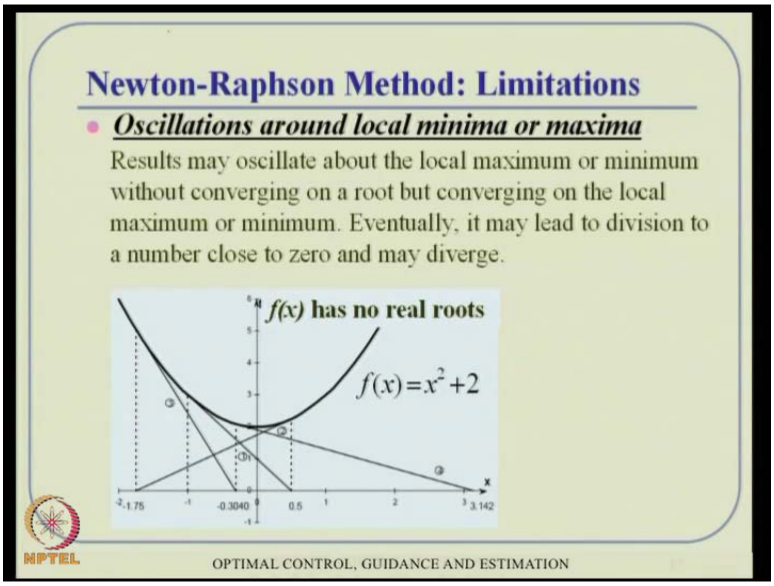


Root jumping can also happen, for example, if you take a sine x, you know, several solutions are there, whatever solutions are there. Now, suppose, if you start with case value somewhere, then I mean, you can, you obviously, what you intended? You intended to find out some solution here and then you did not intend to find somewhere else actually, but, but remember, that if you just take local slope, it will push you somewhere here and again, you take local slope it will push you somewhere here, again you take a local slope it will push you somewhere here and ultimately, you will be able to locate zero, sine x is equal to 0, that is also a solution.

That is not the right solution. You started with this case value with the hope, that you will find out this solution and one of these ratios can be avoided by taking this something like a, what is called learning rate, actually. For example, x k plus 1 is equal to x k minus f of x k divided by f dash of x k that is what we discussed actually, right. 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 just, I will not take this much step, I will take another small step in this direction. So, I will (( )) going here.

So, instead of going over there, this I will not go here, but I will go here rather. I will not take this, this particular zone, this entire thing I will not take, but this alpha will give me this much only. Many of these numerical algorithms will have this where this alpha (( )), that including optimize routines and all and this is called learning rate actually. And we can probably select a particular appropriate value, which will help us eliminating the some of these problems, actually.

(Refer Slide Time: 31:09)



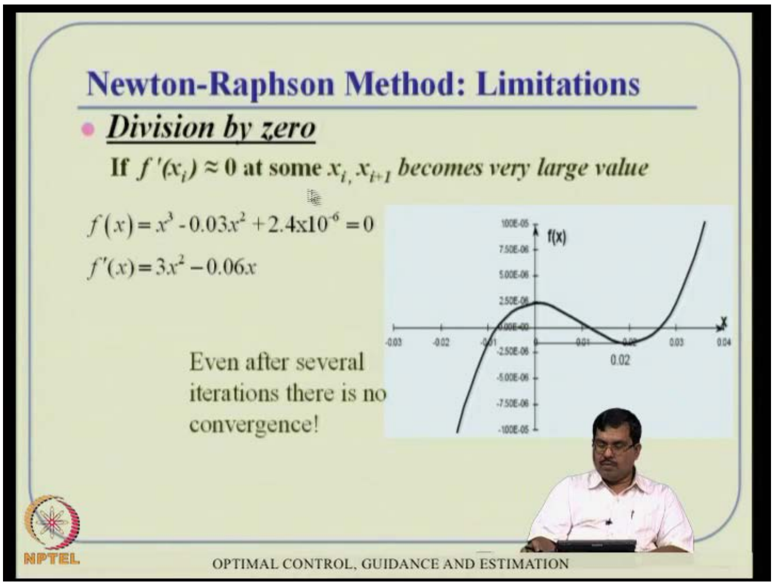
So, one first thing is, it does not converge around inflection point, it does, it is a root jumping problem, third is there is an oscillation problem. Remember, this particular f of x has no real roots really because it does not touch the x-axis, it just goes away from tube actually, whether it knows solution really, but you are not, you are planned to that and you still wanted a solution. I mean, if you start with some, some guess value, still I mean, if you, if you start with some guess value, it will try to go there and then it will try to come somehow in this, this entire thing will go somewhere here, actually.

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 kind of oscillate actually. It will just keep on doing, one will lead to other, other will lead to there, and things like that actually. And obviously, that is very clear example here because in the, in such situation you see, that it is there is no root actually. So, we are just finding out (( )), posing around problem to the system actually. Anyway, it is also like a local minimum and maximum, there is another issue, that even if it really touches the axis, let us say, then there is a problem also actually, why? Because the derivative turns out to be zero; at sometimes if, if you have this, this function rather, let us say, touch this axis, I mean, this somewhere here it is around zero. That means, this plus two is not there in that situation also. What will happen is, as you approach closer and closer and closer, we will approach to their solution.

Anyway, once you approach, close, closer to that solution, the slope becomes zero actually, so that f dash of x here, this particular thing, what you see here, that will turn out to be zero and hence, you will jump in it, actually. You will come very close and then go away, that where again this alpha term will help. If you have alpha term it will not force you to go there actually.

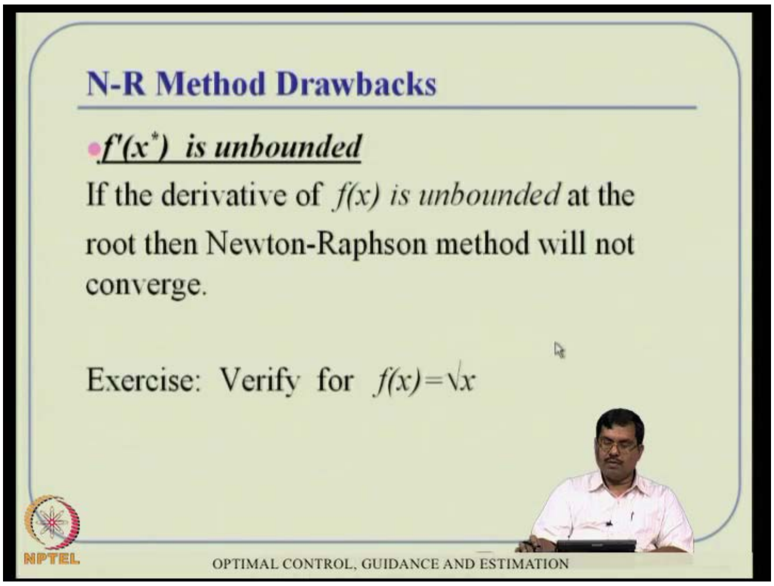
So, sometimes these adoptive learning rates are also nice start with high learning rates. After couple of iterations you reduce this value to a smaller value, smaller value like that actually. So, that way, that way you are, initially you will not mark slow, you will mark slow rather relatively fast, but towards convergence you will also not diverge actually. You will, you will stay there actually.

(Refer Slide Time: 33:25)



Now, again division by zero is very apparent. Whenever there is a slope zero, then there is a division by zero problem, which will not be a nice (( )), actually.

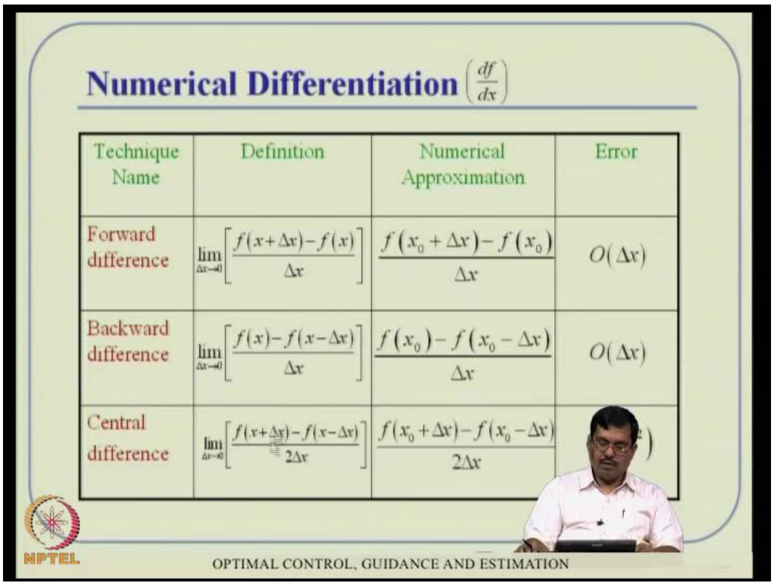
(Refer Slide Time: 33:32)



There is another problem, that if you at, at the solution point, f dash of x is not zero, but it is unbounded, that means, it is infinity. Then, also there is a problem actually. It is a very (( )), but actually it is a problem. We can verify that by 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, so this, that may around zero, it is actually infinity. So, then, there is a problem actually, there are many.

So, that means, there are many, many, many issues there for Newton-Raphson method, but if it, none of these issues are actually there, or if you find some, some of the solutions like (( )) fixing and things like that, then actually converges very fast. That is the way it is part of it actually.

(Refer Slide Time: 33:59)

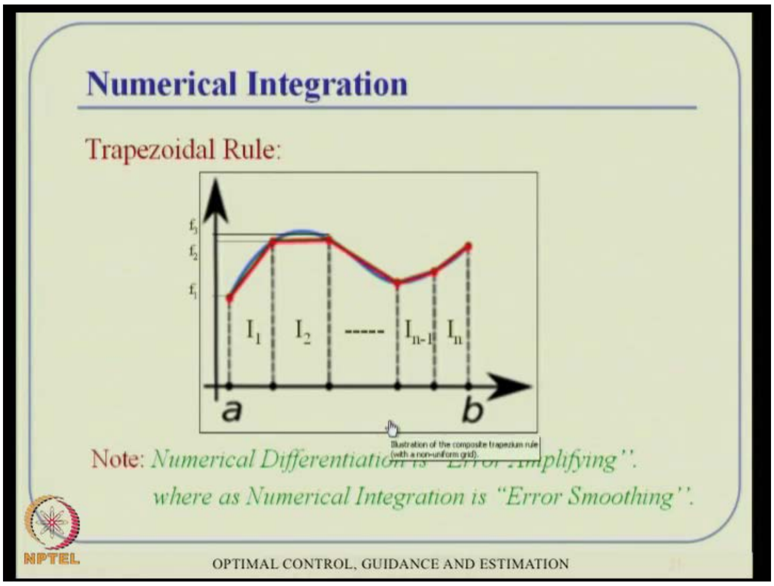


Go to the next concept, many times we need numerical differentiation, that numerical differentiation we know, this is the definition and either define it that way for a difference way, that may limit delta extends to zero, f of x plus delta x minus f of x divided by delta x, or in a backward difference. 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, then little negative, take the difference divide it by 2 delta x rather.

So, numerically, we want to really approximate, that you have a x 0 value. So, you take, evaluate the function around x 0, either in a positive side or in a negative side or both ways. Then, it simply comes from definition, as long as, you take delta x small value. If I eliminate this, a limit condition, that is all we are doing here. Numerical approximation and a limiting sense delta x has to be very close to zero, but instead of that we will put a small finite value and still evaluate the derivative actually, that is called numerical approximation, actually.

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 forward difference or backward difference, the error is of order delta x, but if you use central difference, the error will be of, I mean, this order, delta x square actually. Obviously, central difference is better in accuracy sense point of view, but sometimes you will be required to use forward difference and backward difference, like for example, if you start with grid point 1, then the numbers are available only at grid point 2, then grid point minus 1, it is not available, then you have to take for a difference, anyway. But wherever it is possible to take central difference, my suggestion is to kind of (( )) that actually, because the error is lesser.

(Refer Slide Time: 36:31)

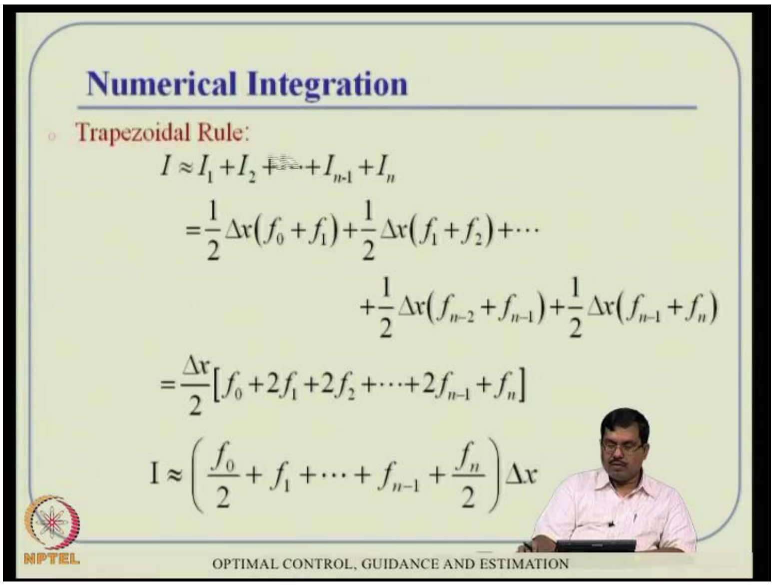


Next concept is obviously, numerical integration. This is all about numerical differences. Now, numerical integration sense, suppose you have a curve like this, whatever curve is there I want to evaluate this, this integral. That means area under the curve actually and one popular way of doing that is using trapezoidal rule. One, 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 actually and then this area like that. So, that means, you can simply hold the values of the grid point and then try to find out, you can, whatever f 1 value at point x 1, you can just simply hold that and just find out this I 1 actually (( )). Forget this inaccuracy here, whatever happens here and similarly, you can try to evaluate this entire area, if you add some little more area actually.

So, that is one way of doing that, but little bit better way of doing that is I have these two points, these function values, anyway available to me. These grid point values are also available to me. So, why do not have in, in evaluate using this trapezoidal rule and this trapezoid formula actually, which is there. Instead of doing that, I interpreting as something like, I will not do these, I will just take out and take, I will evaluate this, this area rather, actually. Whatever I 1, similarly I will carry on with I 2 and things like that actually. So, that happens to be better obviously, and that is doable.

So, the formula sense, I want to have this entire integration. So, what is, I have to take summation of all these, I 1, I 2, upto I n minus 1 and I 1 actually. All these areas, whatever stripes I get, little, little stripes, I get those stripes I have to integrate, I mean, I have to evaluate and then take summation of that actually.

(Refer Slide Time: 37:58)



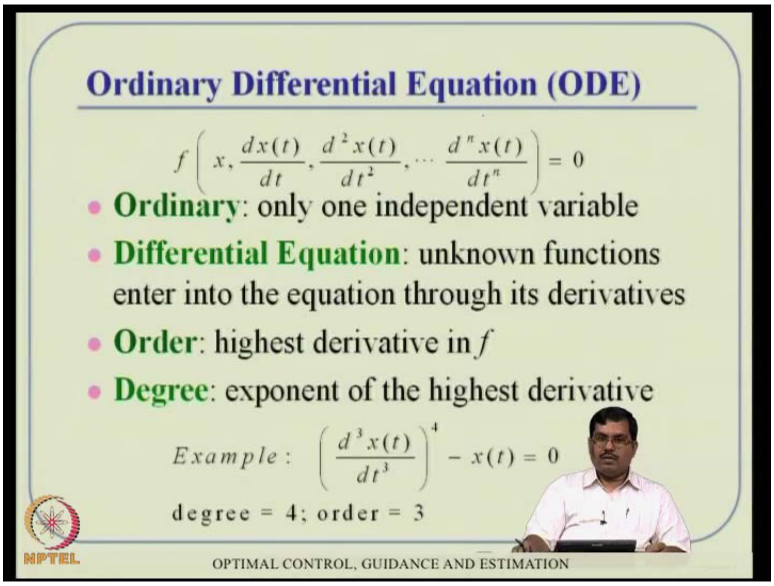
So, that is what I am doing here, assuming a uniform grid. That means, delta x remains same, you can probably do that actually, delta x, I 1 is nothing, but f 0 plus f 1, this is the trapezoidal formula for I 1. And similarly, for I 2, that is the trapezoidal formula and you carry on with all that then it interestingly turns out, that if I take delta x by 2 common, then it is f 0 by 2 in the beginning and f n by 2 at the end, and all other things are simply summations actually.

So, so if I, if I take this two inside and then take delta x common here, then area, it is by trapezoidal rule is given like this actually. And also remember, there is a small comment out here, that if you do numerical differentiation, there is always error amplifying, whereas numerical integration is always error soothing actually. Smoothing because see, you do this numerical differentiation, this formula is, suppose you have a little, let us say, the function has not, nicely this way, but the function is something like small error actually. That means, the function is something like not this way, but something like this.

So, now, what you will be telling? If I, if I evaluate a slope here, I evaluate a, alright, I will evaluate a slope here it will be like this and I evaluate, the very next time it will be like this actually. The values are quite different actually; one is almost like plus infinity, otherwise minus infinity. So, that way even the values of the functions are very small, the value of the slope at here, what I am evaluating and value of the slope here are very, very different actually. 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, that is a strong recommendation rather.

However, if you do numerical integration with respect to this and with respect to an average curve, let us say this is an average curve, so there will be some area, which is getting added, some area which is getting subtracted. You do not have to really know what is the exact area value under those noisy data, but if you do the integration with, it is, it is rather, instead it will try to smooth out the rawness actually. So, if you have to, if you generate a raw dataset using your experimental things and like that, as the differentiation and the 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. Alright, so this is the numerical integration.

(Refer Slide Time: 41:09)



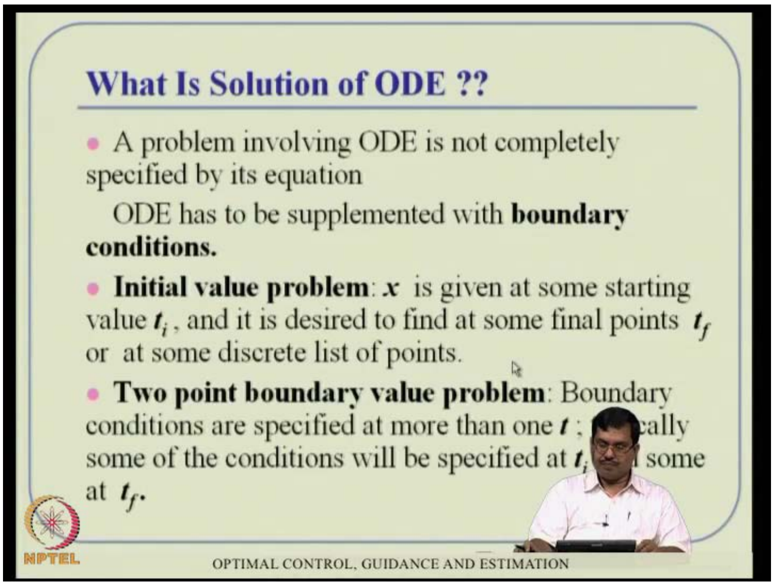
Then, the next concept that is needed for our kind of thing is actually ordinary differential equation solution, you know. Remember, many, many times we will do d A X dot equal to f of x u. And we want to integrate those equations and then find out what is solution given a (( )).

So, now, some of these concepts, before we 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, normally time here. If you have more than that you can take about PDEs, partial differential equations actually. We will not be worried about those things here, we will worry about ordinary differential equation here. And there is a something called, concept called order and degree. Order of the differential equation is the highest order derivative in f and well, not really f actually, I think it is x, highest derivative of x of t. So, that is what you see here is the order of that and the degree, degree is the exponent of the highest order derivative, actually. That means, if you have, d square by, d square x by d t square to the power q, then it is the degree 3, basically.

So, this, in general, is, is a non-linear equation. So, it can happen, nonlinearity can come in higher derivatives as well, actually. So, you take the highest order derivative and then see, what is the power of that, actually. And most of our problem, the highest order derivative will contain degree one. So, that way we are, so we will always work, most of the time we will work with degree one actually, but, but out of (( )) arbitrary, that may, you can talk second order system, third order system, (( )) system, like that actually.

So, for this small example you see, this kind of thing, the degree is 4 because this first order, order is the highest of the derivative, which is 3 and this particular term has degree, I mean, this exponent 4. So, that is degree 4 actually. These are simple, the simple concepts here.

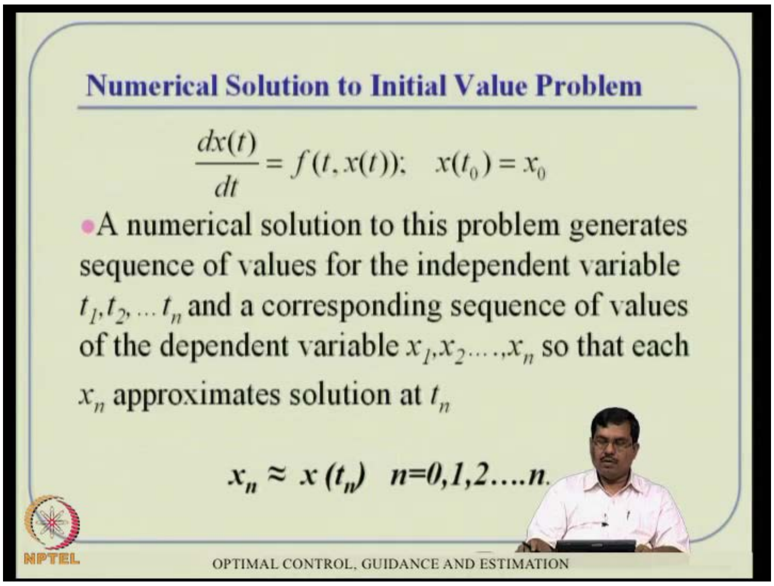
(Refer Slide Time: 43:31)



Alright, what is the solution of ODE, then a problem of, problem involving ODE is not completely specified by the equation alone, actually. You have to talk about some boundary condition, namely the initial conditions. 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, need not be only given at t 0, it can be given a t f also, but it can be given any point of time, really. It can, can integrate the equation both forward, as well as, backward, but using negative delta t, that is all.

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

(Refer Slide Time: 45:09)

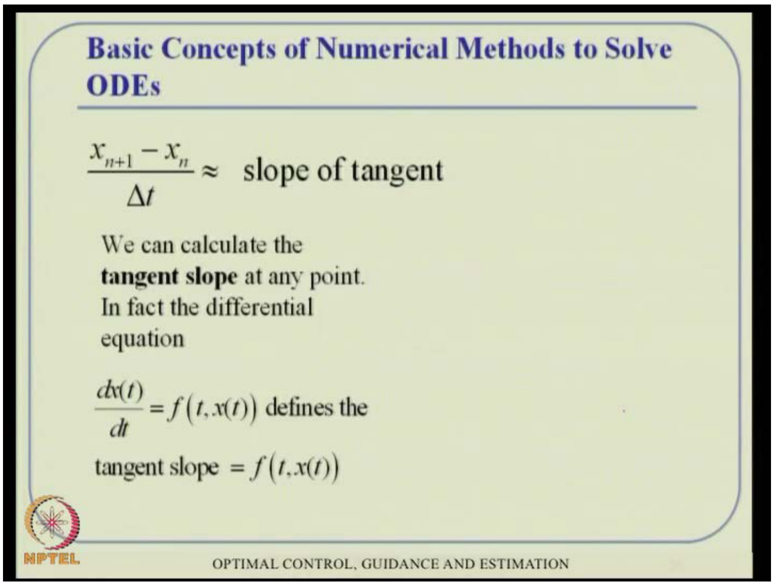


So, numerical solution to initial value problems, this is the problem. Eventually, dx by dt is t, x of t; x of t 0 is x 0 and anyway, this can be, in general, this x is actually a vector x. That means, it is actually x 1, x 2, x 3, all these components are there for you actually. So, this not necessary a scalar equation, this is a vector equation.

All these n equations are there, x 1 dot, x 2 (( )) dot with all initial conditions available, then how do you get a solution? That means, what you mean by solution, I want to find out what is my, I know, 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 actually. So, t 1, I still continue from there, remember t 1 is nothing, but t 0 plus delta t and t 2 equal to t 1 plus delta t, like that actually.

So, if I, alright, let us do that delta t like that actually and this, this delta t is, may not be same, it can be different also, delta t 1, delta t 2, like that you can also do that actually. Most of the time delta t is taken as same actually.

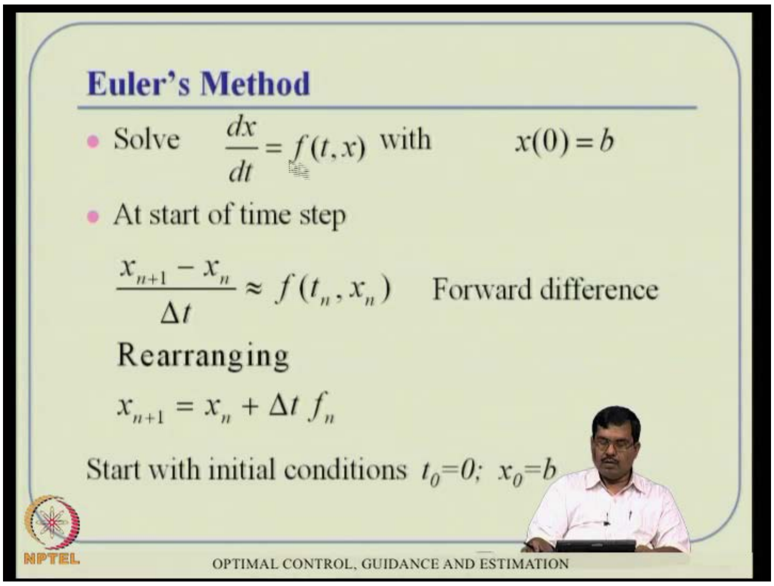
(Refer Slide Time: 46:34)



So, the first, first approach, what is very simple rather, is called Euler integration in fact. So, what you will do here is, quickly we will go through that. All the numerical integration methods will rely on what is called as tangent slope method actually. Normally, it will try to approximate the, the curve. Remember, x dot equal to f of t x, that means, is actually function for each. We can evaluate slope part to any point of, any point of that x 0 value. Suppose, you know, x 0, x 1 is very, whatever it is, the corresponding t 0 known to you and then, the function is completely known t, you a function of t and x actually.

So, you can eventually take, I mean, you can, it take derivatives of function and then you can take this, the slope and the slope of line and things like that, and it explain that, that we are using that equation of lines, the slope of line, things like that. You, you use that information to predict the next value solution actually and that is what we will do.

(Refer Slide Time: 47:37)



And very quickly we see, that Euler method, if you have dx by dt equal f of t x and again, this x is vector here. That means, with x of 0 equal to b, then that any, any start point of time, whatever point (( )). Suppose, I know the solution of t n, already I want to find out for t n plus 1 actually. So, at t equal to t n dx by dt, I can go back to my, my differentiation formula and use this for my, difference method, I mean, approximation and that is my dx by dt.

So, I can use, that x n plus 1 minus x n by delta t, that is the difference and this is nothing, but f of t n and x n. So, using for our differential formula I can do that and then, you simply solve this x n plus 1, this x 1 plus 1 (( )) turn out to be like that and remember x dot, I mean, f n plus 1, what you have is nothing, but x n dot, actually. So, that means, if x n plus 1 is equal to x n plus delta t into x n dot. Also, sometimes people write that way, with that thing 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 and that is because this f, I can calculate that and then follow this formula to get what my x n flow actually.

It is very easy, rather Euler method is extremely easy compared to all other methods and we use that in systems theory many times. But then, with the, I mean, conscious information, that Euler integration is not very accurate actually. For, for getting a good accuracy you have to use delta t, very small. If you do Taylor series expansion, that is, the tangent slope approach thing like that. In Tailor series expansion of this particular function f of t x you are only retaining first order derivates actually, 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 actually. And if delta t is very small, you are going to take very baby steps, very small steps actually, which is typically not good in control theory either actually, because sometimes you may be caught with a trivial delta t. For example, delta t is 10 to the power minus 6 and 1 microsecond, within that 1 microsecond control update time is extremely small and in my knowledge I have never seen control update frequency of that small, delta t being my 1 microsecond I have never seen, it is all like millisecond level I have seen actually. So, you cannot do that, that problem demands that actually.

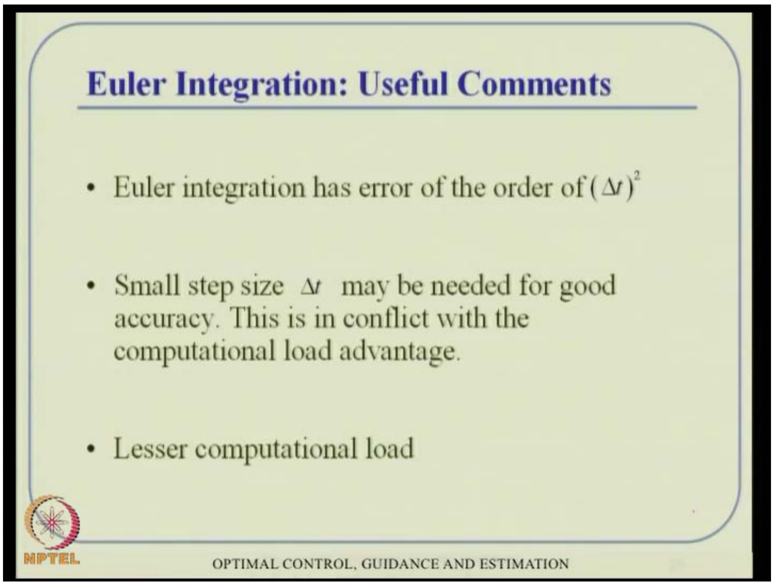
The 2nd issue is, suppose you want to do it in a faster way, tell the, you can argue, that all other numerical methods we are going to see one more by the way is a little demand little more computation for every iteration. This is the smallest amount of computation by the way. So, computational advantage (( )). However, suppose, the other method gives me relatively larger delta t, I can use that, accuracy will be high and here accuracy being less I have to use lot of these grid points actually.

So, I have to repeat this computations so many times before I go from here to here, whereas in other method I can go directly there actually. That means, even if I am using a little more computation for every step, I have to work with less number of steps actually.

So, these are all, I mean, kind of advantage drawback things actually 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 (( )) this equation this way and this is a (( )) form of this continuous equation, then you can take various derivatives of this. For example, you can talk about, let us say del, some of, sometimes we are required to take derivatives like del x 1 plus 1 divided by del x. We can close one divided by del x n. Suppose, we want to take that, then we, using this formula it is rather easy actually because f n is this way and you can take out all derivatives, I mean, in a (( )) formula, basically close (( )) formula, that will not be difficult.

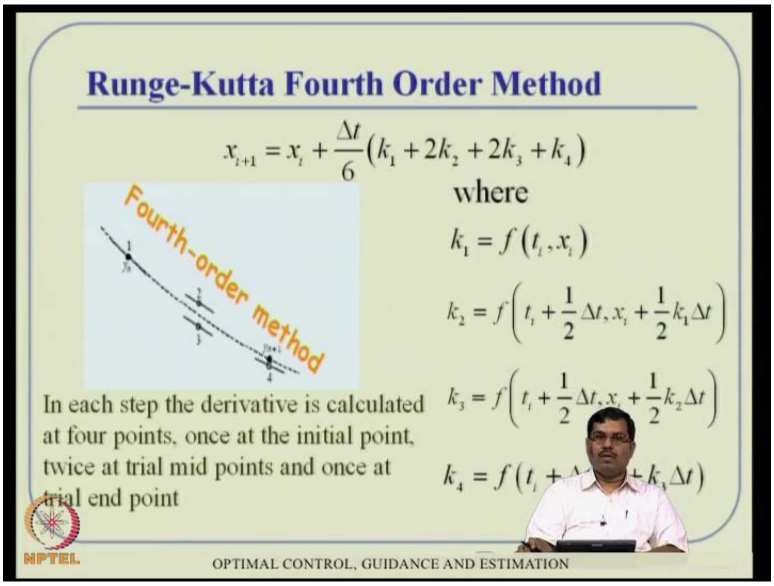
So, there are advantages, drawbacks with this Euler method. You, my suggestion is just use it cautiously actually, do not jump into that.

(Refer Slide Time: 52:06)



Some useful comments, I have already told some of those Euler integration, this error is of 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 actually, but every grid point you are doing less computation, but you have to take several grid points because delta t you have to, here we are required to select lesser actually. So, that is why it is not that advantage actually, but in general it is come computational load. It is also close form; further algebra is possibly in close form like that actually.

(Refer Slide Time: 52:44)

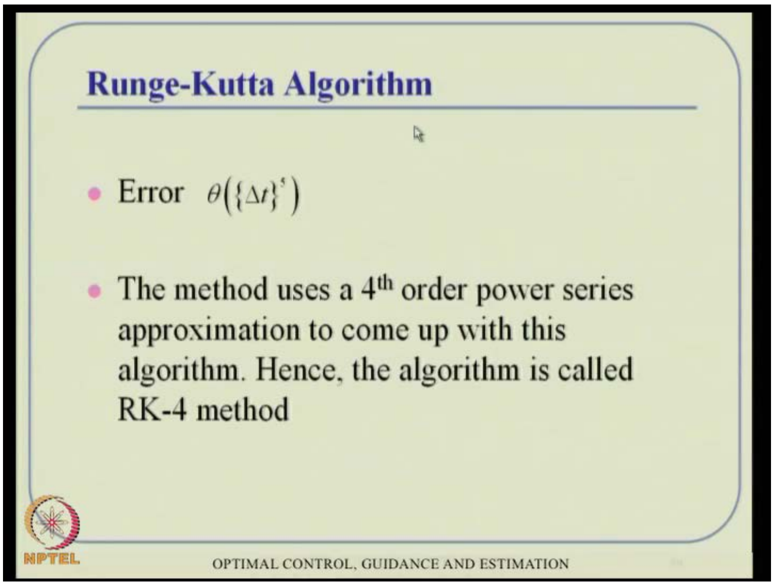


And there another thing comes to method in Runge-Kutta method and then, Runge-Kutta methods various order. We can talk about 2nd order Runge-Kutta, 3rd order, 4th order like that actually, which is very popular in system's theory. And most of differential equation solution, numerical, numerical solution is 4th order actually. And 4th order solution essentially, the concept is like that, derivation I will not give here, we will talk about like some function, that f of t n, x n is something like, let us say and you, some value like that here.

And then, it attempts to find some approximate slopes here actually, point number 1 you already have a 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 actually, and then evaluate that. Even using this polynomial it will be able to kind of march ahead with solution here with much more accuracy actually, that is the philosophy algebra part and all you can see a numerical method book actually.

Ultimately, what it tells me is, I start with the series of this 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 able to calculate k 2 actually, 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 this k 1 to k 4 I will just try to kind of, delta t over 6 and average out all this, 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 actually. So, this entire formula, final formula x i plus 1 is given something like actually.

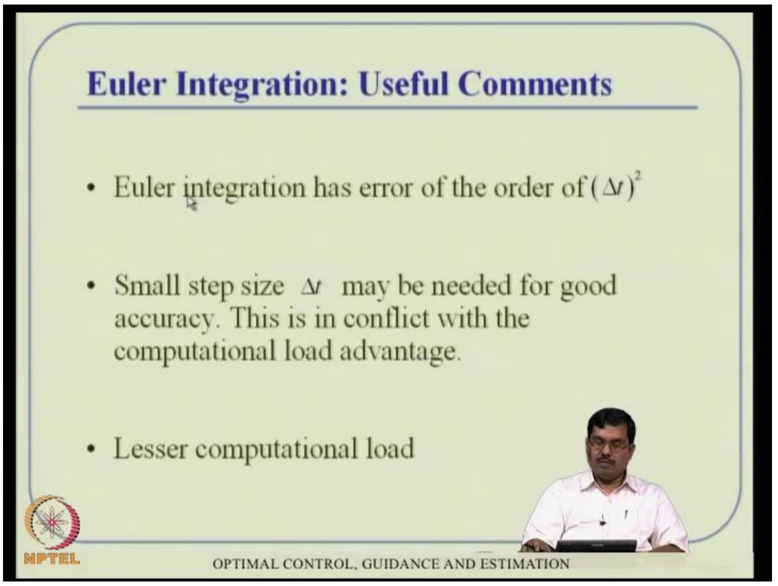
(Refer Slide Time: 54:37)



And essentially, turns out, that Runge-Kutta 4th order, Runge-Kutta algorithm has error, delta t 2 to the power 5^{th}. That means, it is highly accurate actually, compared to the, remember Euler integration had accuracy of the order of delta t square only. Now, that square gone to power 5th actually, that is why it is much more accurate, actually. So, and the 2nd comment is, the method essentially uses a 4th order power series approximation. Remember, Taylor, I mean, that Euler integration uses only linear. Now, this one is to 4th order power series approximation and Taylor series.

To come up with this algorithm, again details of that is not here, you can see some numerical, numerical methods book actually and this particular algorithm is popularly called as RK-4 method, 4th order on method actually. This is very popular in (( )), we can use all this and probably, for Euler you do not need integration formula, I mean, just that, that formula is so simple, that you do not need any integration formula, I mean, rooting for that actually, primarily.

(Refer Slide Time: 54:47)



Because this integration formula, we can simple write it longer actually and once evaluate this f of x actually, then it is easy to just (( )) there. So, do not need formula per say, but this algorithm requires little bit of evaluation, of computation, I mean little bit of sequential computation. So, the formula is available there and in (( )) we can see some of this is something called ODE23, that is a function. And there is another function, which is called ODE45 and ODE23, this is what is called as RK-2 method, 2nd order Runge-Kutta method and this is, this will give RK-4 method. So, 4th order Runge-Kutta method actually.

Essentially, both are similar function, but with difference, that these are there and by default whatever is there (( )), it is what is adaptable step size. That means, your delta t, that you are using does not the necessary remain constant actually and it is adapted provided.

Suppose, you have a function something like this and then, you have, you have, this is therefore, 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 actually. So, it will try to see, this where the slope changes happen in a rapid way and if the slope changes, changes of slope is high, then correspondingly it will adjust. Delta t is small actually, this is your delta t, the delta t is small here, delta t is large here. So, this adaptation of delta t happens in implicit way.

Inside the algorithm in RK-4, in this ODE45 routine and all that, for most our type of applications we do not, we are ok with constant step size. So, we can probably write this function ourselves also. So, many times and then see what is the delta t is, that you can select actually and constant step size is also very good. For implementation concern you do not have to monitor what is the delta t. Every control cycle update, guidance cycle update like that, that is fixed by some, some number, so you start using that particular number in constant step size way actually.

So, that is some of the, this algorithm numerical methods are useful here. So, with that I will probably stop this particular class and you will see, that root finding, both linear equation, non-linear equation, numerical derivative and integration, as well as, numerical integration are differential equation, that is all we discussed in this class. These all are useful in our, in our control theory and many of this practical usage implementation will come from your experience as you work with different, different problems each, actually. I hope this will be useful in your further exercise. Thank you.