

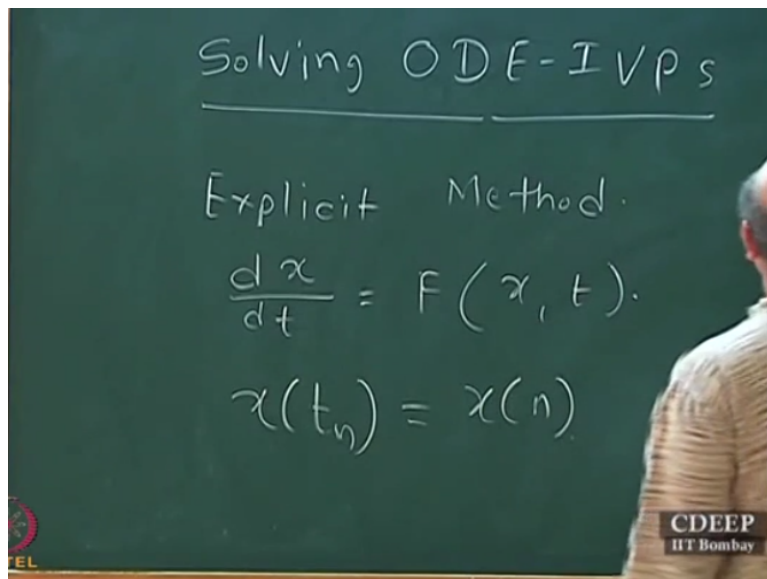
Advanced Numerical Analysis
Prof. Sachin Patwardhan
Department of Chemical Engineering
Indian Institute of Technology - Bombay

Lecture - 41

**Solving Ordinary Differential Equations - Initial Value Problems (ODE-IVPs) : Runge
Kutta Methods**

ODE initial value problems, methods for solving ordinary differential equations with initial condition specified and then I talked about 2 categories of algorithm.

(Refer Slide Time: 00:32)



We talked about 2 categories, 1 was explicit method and the other 1 was implicit method. Explicit method given this differential equation $\frac{dx}{dt} = F(x, t)$ and initial condition $x(t_n) = x(n)$.

(Refer Slide Time: 01:24)

Explicit Euler.

$$x(n+1) = x(n) + h F(n)$$

Implicit Method.

$$x(n+1) = x(n) + \frac{h}{2} [F(n) + F(n+1)]$$

$$h = t_{n+1} - t_n$$

We derived explicit Euler method that was $x_{n+1} = x_n + h \text{ times } F_n$. This was explicit Euler method and other class that we talked about was so this is explicit Euler and other class that we talked about yesterday was implicit method. So what we yesterday looked at was trapezoidal rule. So trapezoidal rule is given as $x_{n+1} = x_n + h/2 (F_n + F_{n+1})$. This is implicit method.

This is trapezoidal rule, h here is constant integration interval so h is $t_{n+1} - t_n$, difference between 2 successive time steps. So this particular method does require iterative calculations. I am just going to write down the algorithm for doing the iterative calculations here. How do I solve this? Typically, F here is a non-linear function. Now right hand side here F_{n+1} is nothing but function vector evaluated at x_{n+1} .

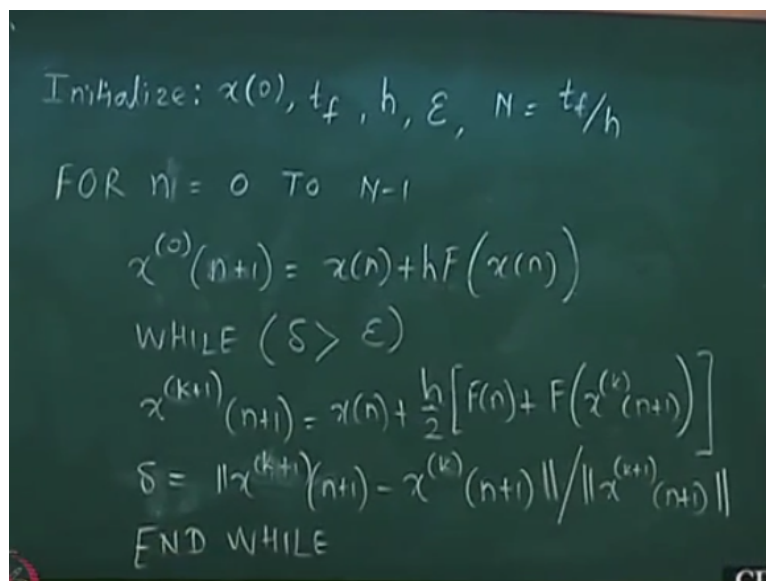
So this is an implicit equation where left hand side is x_{n+1} and the right hand side also depends upon x_{n+1} . So this is implicit equation, which you have to solve iteratively. So what is the method? How will you do this? The algorithm would be something like this. Well to solve this equation, I can use complex method like Newton-Raphson, but I do not have to.

As I told you that there are some simple derivative free methods, which can be used without requiring to compute Jacobian okay, so I can employ those methods here. If I give a very good initial guess, then it is possible to use those methods. So what we are going to do is to generate a good guess here okay for iterations I am going to use explicit Euler method. This will give me a good guess.

And then I will use it to kick off my iterations and my iterations then you know will be just successive substitutions. I am not going to use Newton-Raphson or any other more complex method. So simple successive of substitution works here if you give a good initial guess. So as I said which method you use, we talked about multiple methods for solving non-linear algebraic equations.

And which method you use when is context dependent. In this context, we most of the times sufficiently use simple successive substitutions. So what is the algorithm?

(Refer Slide Time: 05:18)



So I initialize x_0 this is the initial value from which you start your integration okay, t_f is my final time, h is my integration interval, epsilon is my tolerance and capital N is t_f/h number of steps total I want to go from time 0 to time t_f I want to integrate differential equation. Integration step size is this small h and then I am going to use this. So in my program first I initialize x_0 , t_f , h , epsilon and so on.

Then I have to go on marching in time so I am writing a pseudo code for n going from 0 to $N-1$. I want to use the implicit method okay, but every time I need a good initial guess for the implicit method. So what I do is my initial guess now just try to understand the notation. Here superscript 0 is iteration index okay sorry this should be n so $n+1 = x_n + F(x_n)$. So I initialize my iterations using explicit Euler okay.

Then I write a WHILE loop, there is h times $F(x_n)$. Then I write a WHILE loop here and my delta is nothing but okay now let us understand this algorithm. The first guess in the iterations

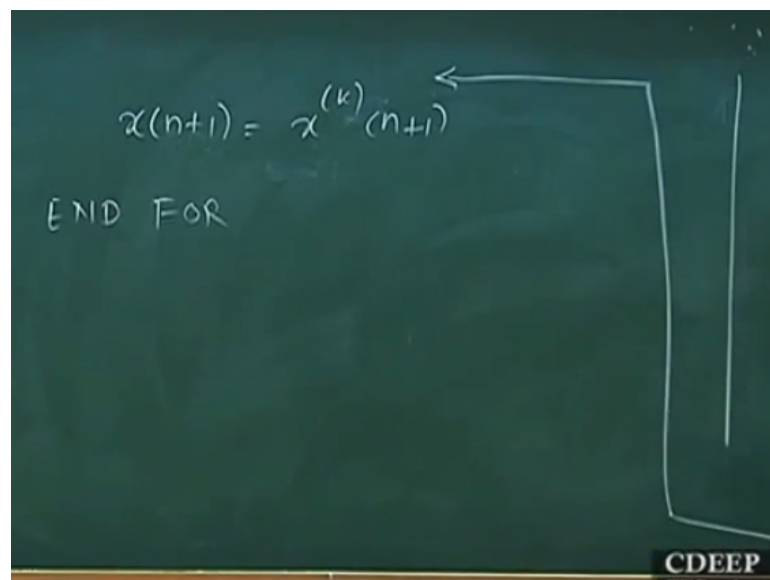
it generated using explicit Euler. This does not require any iterative calculations. Now I want to compute x_{n+1} iteratively. So what I do here is x_{n+1} is generated using old guess for $n+1$.

First time when you enter this WHILE loop okay, you have this k_0 . k_0 is calculated here that will give you k_1 , using k_1 you will get k_2 okay. Time index remain same because you are trying to get $n+1$ okay. You keep checking whether this iteration is converged or not. That is done using this delta which is relative change between 2 successive iterations should become very, very small okay.

Once this become small okay this relative change between 2 successive iterations become small, I exit this WHILE loop and whatever result I get I accept it as x_{n+1} okay. So at every time step, there are iterations in implicit algorithm. In explicit algorithm, there are no iterations. It just gives you, you just go marching in time. Here while you march in time, you keep doing iterations and these are the iterations okay.

So to close the loop, I have to put so at the end of this I am just continuing it here.

(Refer Slide Time: 11:49)



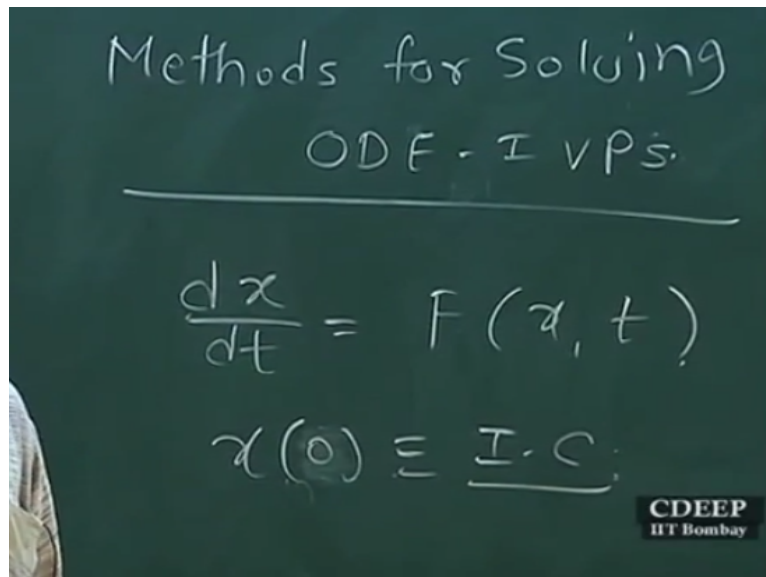
So this is continued here so once the iterations converge then I accept x_{n+1} =whatever was x_k okay. Of course, I am skipping things like you save those values and so on. Basically, in the implicit algorithm at every time step, you have to do iterations. The way I am going to do iterations is not using very complex algorithm. I am just using derivative free algorithm. If you just go back here, these iterations are nothing but successive substitutions okay.

To generate an initial guess x_0 substitute x_0 here you will get x_1 , substitute x_1 here you will get x_2 , keep doing this till you get successive values very, very close and terminate when you get closer values or sufficiently close values. So you keep doing this till this Δ is $>\epsilon$, ϵ is the tolerance that you are given. ϵ seems typically a very small number say 10 to the power -8 or something.

So relative tolerance is very, very small, 2 values are almost converged. Now if your time step is small, h is small then this will be a good guess and these iterations will converge very fast within 4 or 5 iterations this will converge okay. Now in the program which I have asked you to do iterative calculations, you will see that the inside loop will converge very fast step.

he problem might come if you have h is which is very large and then this approximation is not good then you may have problem otherwise convergence will be good; you will get solution every time very quickly okay. So now let us start developing the numerical algorithms okay.

(Refer Slide Time: 14:02)



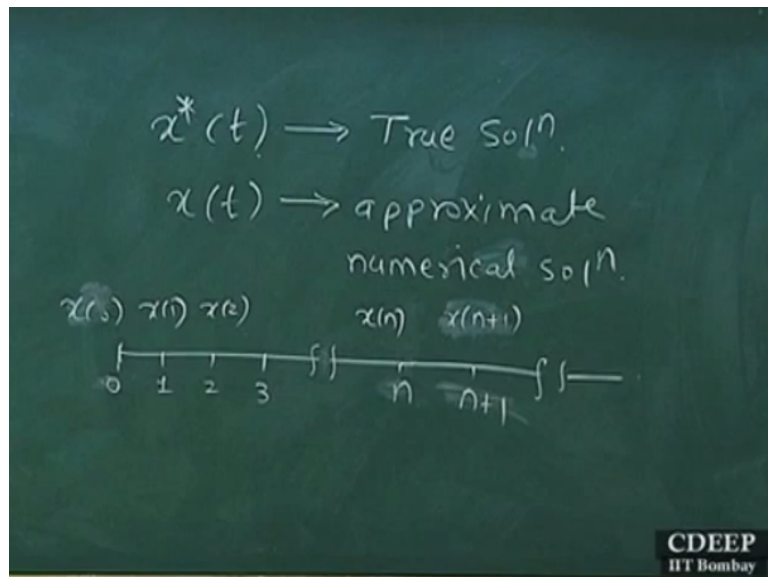
Methods for Solving
ODE - IVPs

$$\frac{dx}{dt} = F(x, t)$$
$$x(0) = \underline{I.C.}$$

CDEEP
IIT Bombay

Now there are 2 classes or I would classify them according to the approximation strategy that is used. The approximation theory is again required because you cannot solve these problems exactly okay. I have this differential equation and then the true solution of this differential equation.

(Refer Slide Time: 14:59)



Let us denote by $x^*(t)$ is the true solution. Often times, it is not possible to compute the true solution. You have to live with only an approximate solution okay. So we will keep calling this approximate solution as $x(t)$ okay, $x(t)$ will be approximate solution. Well there are some cases where it is possible to solve it analytically but if you look at the entire set of differential equations that we encounter in engineering that fraction of problems for which analytical solution exist is very, very small, it is not a large set.

So we have to live with approximate solutions that are constructed numerically okay. Actually, the true solution of a differential equation, let us see well if you take a scalar differential equation, the true solution will be a function in time or space whatever is the independent variable okay. From time 0 to whatever time t_f that you want to or if t_f is infinity from 0 to infinity it is a continuous function okay.

What you do here is that we often only construct you know discrete approximation. So if you see here I am hopping in time x_n to x_{n+1} to x_{n+2} . What about solution between $n+1$ and $n+2$? I am not saying anything about it okay. All these numerical methods see what I did was you know what I did was I started from 0, time 1, time 2, time 3 whatever and this is my time n , $n+1$.

So I am calculating solution in jumps so I have this x_0 here approximate solution. Then, I have x_1 here, I have x_2 here, I have x_n here, I have x_{n+1} here so I am trying to compute this x at discrete time points okay hopping in time. I am not saying anything about what happens between okay.

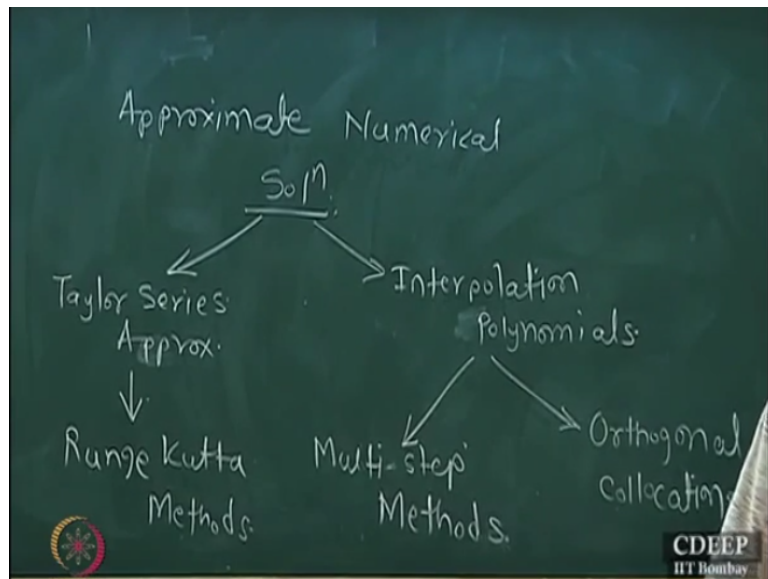
All that I can do with let us say Euler method is I can reduce the distance and come closer and closer even then it is not same thing that true solution is not in pieces like this. The true solution is a continuous function. So you are approximating okay. If it is a vector differential equation, it is a vector which is consisting of functions which are continuous over 0 to t_f that is the true solution.

What you are doing is you know you are replacing that vector function defined over 0 to t_f by number of vectors, 1000 vectors, 10,000 vectors depending upon how you choose a integration interval you will end up approximating that using a set of you know vectors at discrete time points okay so it is an approximation.

When you draw using MATLAB, you tend to connect and draw but that does not mean that you know in between solution is just connecting the 2 points. That is only for our visual. Actually, in MATLAB when you get solution only at these discrete points you should only plot the points because you do not know what is in between actually but when you plot you will not plot discrete points, you will plot continuous curve.

Actually in that some kind of linear interpolation can be done in between. So the way we erroneously represent as a continuous graph the solution which is actually discrete should not be taken as the correct true solution. It is only that you are going closer to okay. So what are the classes of methods? So I would say that there are 2 ways of developing approximate solutions.

(Refer Slide Time: 20:11)



So one class is based on Taylor series approximation and this actually leads to so called Runge Kutta methods. Taylor series approximation actually involves computing derivatives. It is very difficult to compute derivatives. So very nice substitute was developed in which you only evaluate functions at certain points without requiring derivatives but what you do in Runge Kutta methods is equivalent to do in the Taylor series approximation okay.

I will derive it and we will see how it is done. So you want to first approximately solve it using Taylor series but Taylor series itself can be difficult for a multi-variable function. So you further approximate and you develop Runge Kutta methods okay. So the class of methods that you get here. The other one I would say is apply Weierstrass theorem and do interpolation.

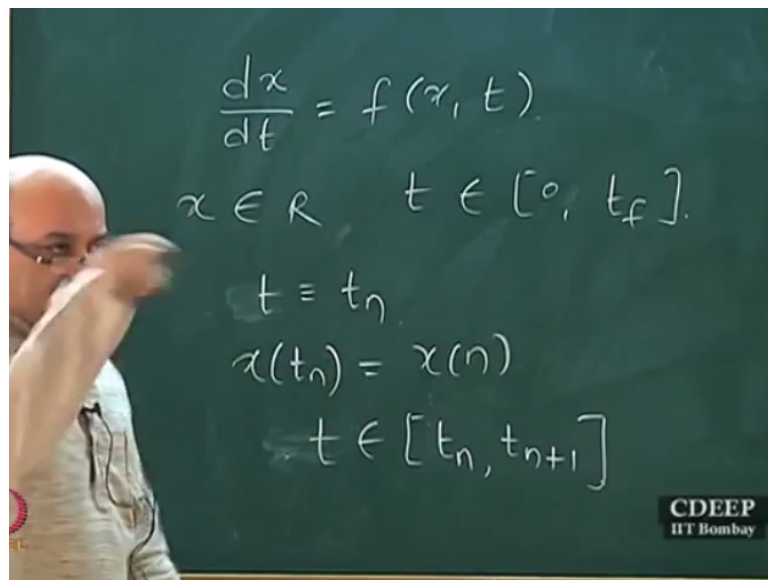
Well Weierstrass theorem is applied in both the cases so instead of just saying Weierstrass theorem you can say interpolation polynomials. The other approach is interpolation polynomials. Interpolation polynomials gives rise to again 2 sub methods. So interpolation polynomial you will get multi-step methods. I do not know whether you heard about Gear's predictor-corrector or predictor-corrector methods.

They are also known as predictor-corrector methods. So multi-step methods is follow out of using polynomial interpolations and other one is orthogonal collocation. You can use orthogonal collocations and then solve the problem converting everything into set of algebraic equations and so we are going to look at all of them. So what I want to do now is systematically derive algorithms of each class okay rather than just stating the algorithms.

So we will start from scratch, derive the algorithms and see what is the philosophy behind these methods? Okay for now when I do this development even though I want to finally work with multi-dimensional differential equations. I am going to actually restrict myself to 1 variable case because derivations become simple okay and 1 variable methods are simply extended to the multi-variable methods by just we do not do derivations separately.

Whatever coefficients or whatever you get for 1 variable method are just extended to deal with the multi-variable methods okay. So development that follows for all the classes of methods I am going to restrict myself to 1-dimensional differential equation and also mention later how it is extended to multi-variable case.

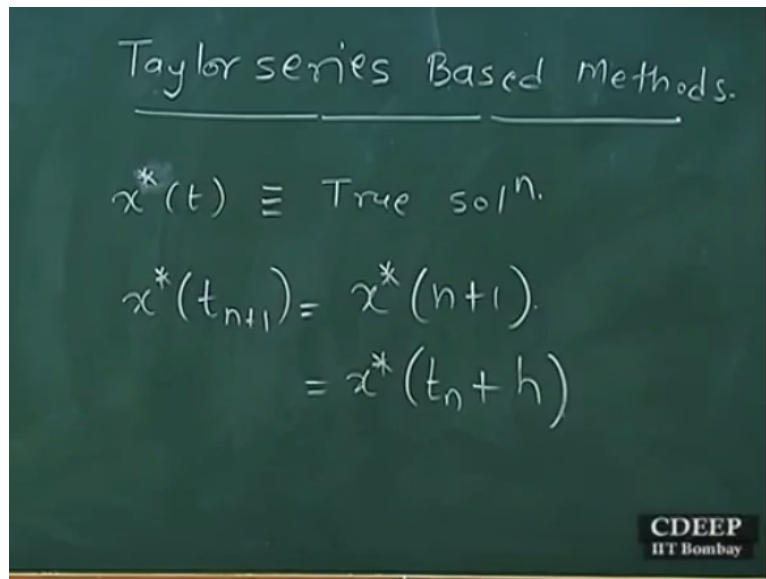
(Refer Slide Time: 24:40)



But the development we are just going to do for simple equation $dx/dt=f$ of x, t where x belongs to \mathbb{R} and you have t which is from some 0 to t_f and a specific problem at hand is we are at time point t corresponding to t_n so we have $x_{t_n}=x_n$ okay and I want to go from my problem is I want to integrate this differential equation over t that belongs to t_n to t_{n+1} .

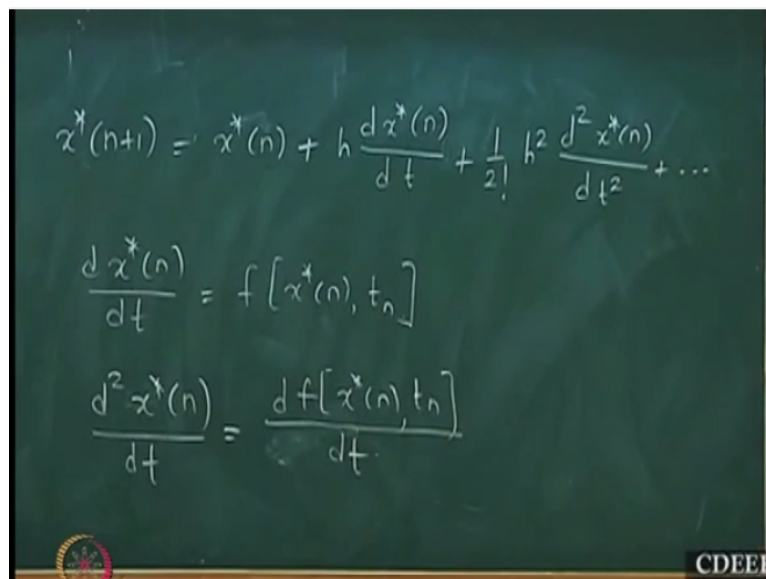
I have divided my interval 0 to t_f into smaller sub intervals and my specific problem is to go from t_n to t_{n+1} okay.

(Refer Slide Time: 25:52)



So Taylor series based methods you start by saying that let $x^* t$ represent true solution okay let x^* represent true solution. Then $x^* t_{n+1}$ which is in our notation, this is same as $x^* n+1$ right. The notation that we have adapted is $x^* n+1$ okay. This is same as $x^* t_{n+h}$ right. This is $x^* t_{n+h}$ okay so I am going to do a Taylor series expansion of x^* in the neighborhood of t_n okay.

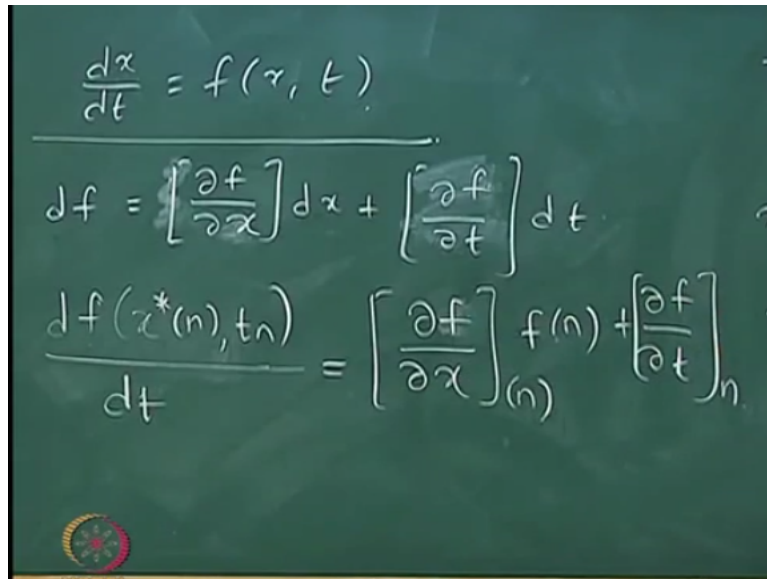
(Refer Slide Time: 27:18)



So $x^* n+1$ is $x^* n+h$ times right I have just then Taylor series expansion of the true solution in the neighborhood of this is exact equality if I take infinite series right so exact equality. Now what is this $dx^* n/dt$? x^* , I know x^* in terms of f so this will be $f x^* n, t_n$ right this should be exactly equal. Now what should be $d^2 x^*$? Is everyone with me on this?

Okay what is df? Now we have to go back and check that okay.

(Refer Slide Time: 29:20)

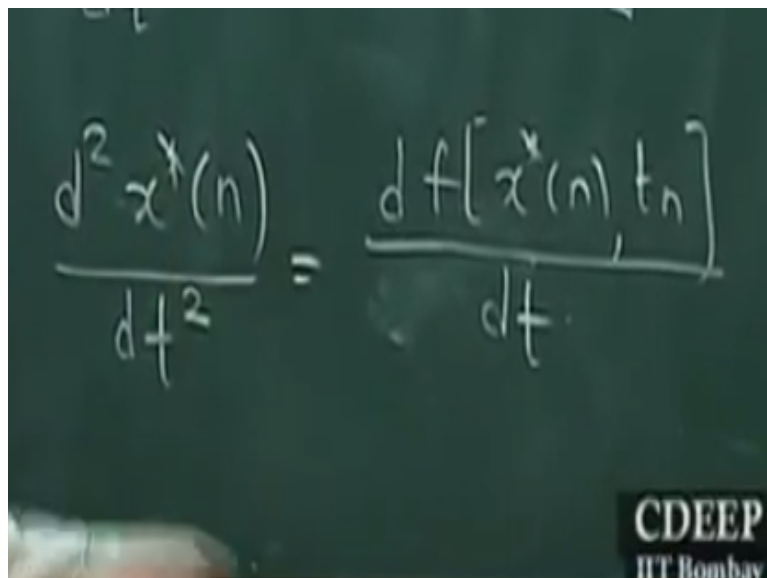


The image shows a chalkboard with three equations written in white chalk. The first equation is $\frac{dx}{dt} = f(x, t)$. The second equation is $df = \left[\frac{\partial f}{\partial x} \right] dx + \left[\frac{\partial f}{\partial t} \right] dt$. The third equation is $\frac{df(x^*(n), t_n)}{dt} = \left[\frac{\partial f}{\partial x} \right]_{(n)} f(n) + \left[\frac{\partial f}{\partial t} \right]_n$. There is a small logo in the bottom left corner of the chalkboard image.

Well I will just leave this 1 equation that is $dx/dt=f$ of x, t okay. Now df is $df/dx dx + df/dt dt$ right. I am just taking exact differential of df so df/dt at x star n . See this term that df/dt at x star n this term I can replace this by df/dx at n calculated at n okay then I will get a term here. See if I take dx/dt , I will get dx/dt right but what is dx/dt ? f okay so this will be f_n okay because dx/dt is replaced by f okay $+df/dt$ at n okay.

So this first derivative of df that is df/dt is replaced by Jacobian of f , f itself and derivative of f with respect to time okay. So this is what it is replaced by and so on. So see this is the second term I can similarly find d^2x/dt^2 .

(Refer Slide Time: 32:07)

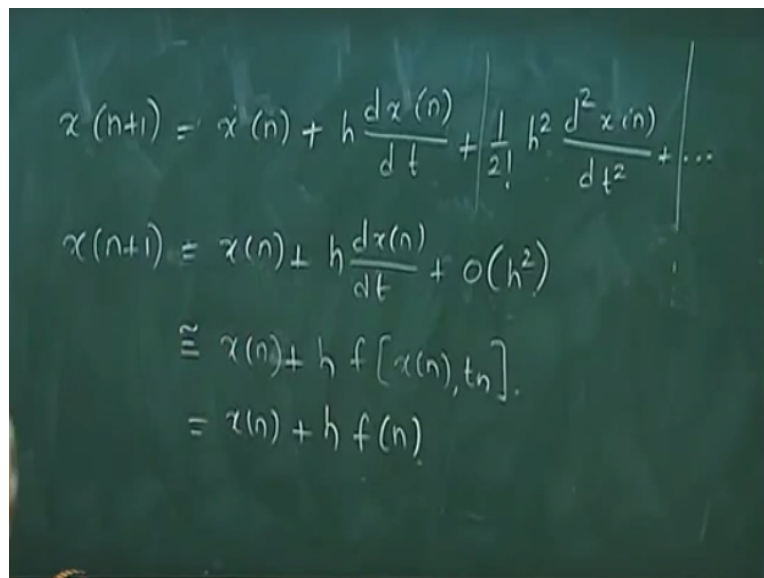


The image shows a chalkboard with the equation $\frac{d^2 x^*(n)}{dt^2} = \frac{df[x^*(n), t_n]}{dt}$. There is a logo in the bottom right corner of the chalkboard image that reads "CDEEP IIT Bombay".

Let us go back here this should be square here, so 2 here dt square. Just like I approximated the second derivative, I can approximate the third derivative, fourth derivative, fifth derivative and so on okay. I can express it in terms of f , Jacobian of f , second Jacobian of f and so on. I can just go on doing this and for the scalar case there is no problem. The terms that you are getting here for the scalar case, these are not matrices, these are just scalars okay.

So it is not so difficult to compute these scalar derivatives and principle okay. Now what I am going to argue is that when in reality we do not have the true solution with us okay. We only have an approximate solution and we are going to continue approximating using the approximate solution okay. The true solution x^* will never be known in most of the cases even for a scalar problem.

(Refer Slide Time: 33:30)



$$x(n+1) = x(n) + h \frac{dx(n)}{dt} + \frac{1}{2!} h^2 \frac{d^2x(n)}{dt^2} + \dots$$

$$x(n+1) = x(n) + h \frac{dx(n)}{dt} + O(h^2)$$

$$\approx x(n) + h f[x(n), t_n]$$

$$= x(n) + h f(n)$$

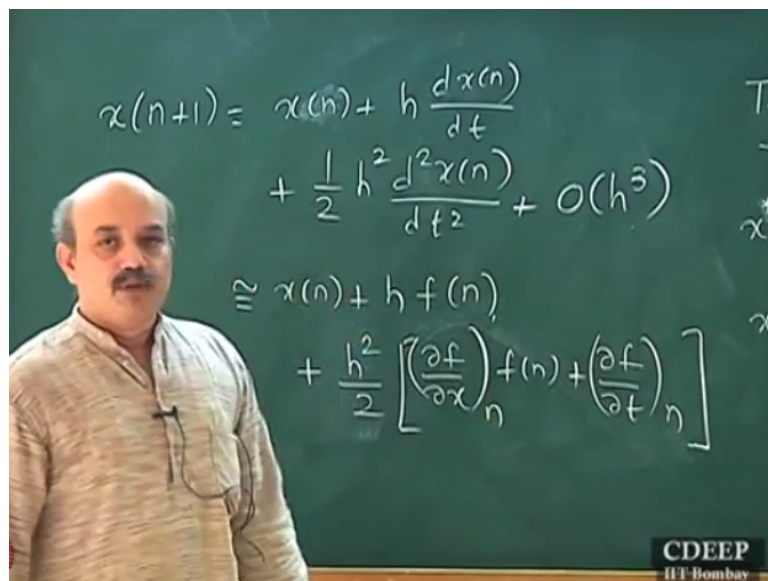
So I am going to develop this approximation not for the true case but for the approximate solution so the star will disappear and then the next thing, which I am going to do here is I cannot do summation up to infinity okay. So I am going to truncate the series after some terms okay.

If I happen to truncate the series after first term okay see I can decide to truncate the series here, I can decide to truncate the series here, I can decide to truncate the series after the third order term okay. If I decide to truncate after first order you know everything higher in terms of x square and higher is neglected. So I can write this as $x_{n+1} = x_n + h \frac{dx_n}{dt} + \text{order of } h^2$, all the terms which are of the order of x square and higher.

When you write like this it means all the terms okay. If I decide to neglect this, if my h is very, very small okay and if I decide to neglect this high order terms then I can come up with the so called explicit Euler algorithm. This is x_{n+h} . Now what is dx_n/dt ? I am going to replace it by $f(x_n)$. This is same as x_{n+h} times $f(x_n)$ in our notation okay. This is my first order Taylor series approximation okay.

What if I decide to do approximation after first 2 terms so which means somebody might say well this is too naïve you are just taking you know first order term h should be very, very small well I would like to include the second order terms.

(Refer Slide Time: 36:00)



So you will say that I will take $x_{n+1} = x_n + h \frac{dx_n}{dt} + \frac{1}{2} h^2 \frac{d^2x_n}{dt^2} + \text{order of } h^3$ so I am neglecting so I am writing it like this and when I do approximation I neglect terms which are higher than h^3 okay. So this then will be approximately equal to x_{n+h} . Now dx_n/dt I can substitute f_n here right, dx_n/dt I am substituting f_n okay $+ h^2/2$ and using just the derivation that we did we can write this as $\frac{\partial f}{\partial x} f_n + \frac{\partial f}{\partial t}$ at n .

So I can approximate the right hand side okay, the second derivative I just showed you how to approximate the second derivative I am substituting that here okay. This if I start doing calculations using this approximation, this will be a second order Taylor series method okay. Likewise, I can have a third order Taylor series method, fourth order Taylor series method okay.

So I am just using Taylor series expansion idea to construct a local solution to go marching in time okay. The way it is written here these are all explicit algorithms. You are going from n to $n+1$ using the derivatives at n local derivatives at n , first derivative, second derivative, third derivative depending upon where you want to stop okay. If h is very, very small even the first order approximation is okay, okay.

But if you want to have h to be relatively larger you cannot stop with first or second order you probably have to go to okay. Now doing these calculations particularly if you have many equations to be solved together can become very, very cumbersome. So you have to compute derivatives even for a single variable case if you are taking fourth order approximation, you have to compute derivatives every time fourth order derivatives and it is not so easy.

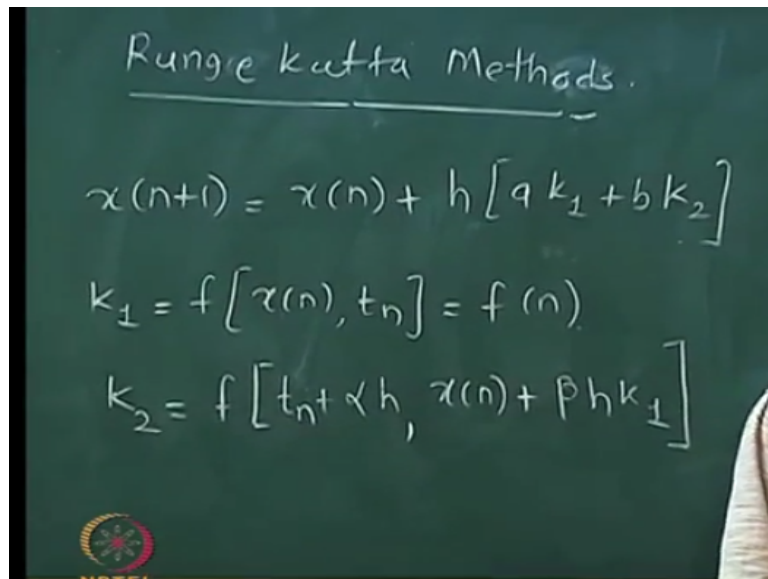
So what was done by Runge and Kutta, they came up with a very nice scheme by which you can do calculations of this type without explicitly having to compute the derivatives okay. You want to get an accuracy which is similar to that of a second order Taylor series method without actually having to compute the derivatives okay. That is the modification which Runge Kutta methods bring in.

So I am going to start with this second order method and show how will I come up with a Runge Kutta second order okay. Now If I do Taylor series expansion up to second order and if I do Runge Kutta second order, I will not get identical results but I will get equivalent results okay. That is what is important.

So what we are going to do in Runge Kutta methods? I mean if you understand the basic philosophy then you know how Runge Kutta third order, fourth order, fifth order are derived will become clear okay. Runge Kutta methods were developed in that era when we did not have computers like you have so doing computations, computing a trajectory, doing it by hand or by doing you know some logarithmic tables or whatever was not an easy job.

It was quite cumbersome. Nowadays, you can do those calculations probably within a second, in fraction of a second not just a second.

(Refer Slide Time: 40:25)



So the idea was I want to compute x_{n+1} as $x_n + h$ times $a k_1 + b k_2$ okay, h is same as your integration interval. I am going to develop a second order method now okay. I will tell you what is this k_1 and k_2 . k_1 is $f(x_n, t_n)$ that is $f(n)$ okay and k_2 is $f(t_n + \alpha h, x_n + \beta h k_1)$ okay, k_1 and k_2 are 2 function evaluations okay. k_1 is the function evaluation at the initial point okay; k_2 is the function evaluation at an intermediate point between 0 and h .

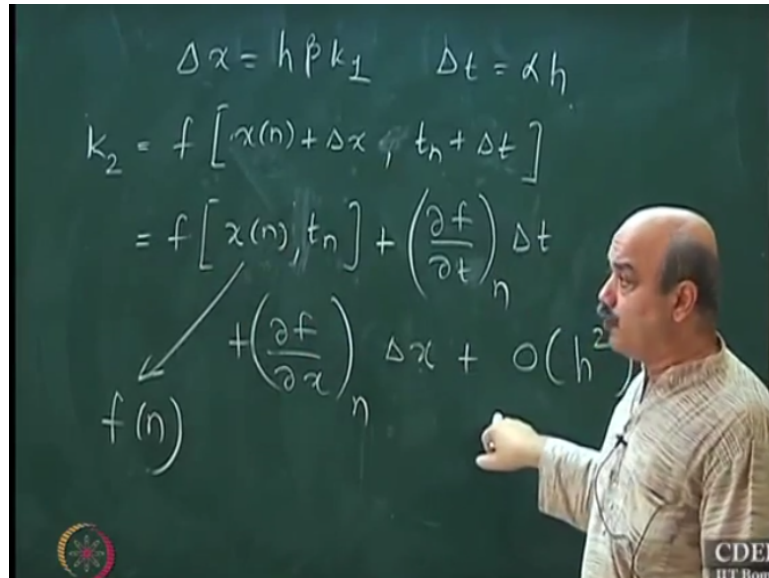
See you are going over interval, you are going from t_n to t_{n+1} , let say integration interval is h okay. So I am going to do a function evaluation at an in between point okay so I want to do only 2 function evaluations and what I want to achieve through this 2 function evaluations? I want to achieve something like this okay. So my aim is to choose α β in such a way a and b.

I have 4 unknowns here α , β , a and b . I will choose α , β , a and b in such a way that doing these 2 function evaluations and calculating this okay is equivalent to doing a second order Runge Kutta method okay. I want to do these calculations in such a way that doing these types is same as is equivalent to doing a second order Runge Kutta step. So what I am going to do is I am going to expand this using Taylor series okay.

And match the right hand side of this with right hand side of this that will help me to find out coefficients α , β , a and b okay, which will make them equivalent methods okay. What is the advantage of doing this? I am just doing function evaluations; no derivative is required. What is the disadvantage of Runge Kutta? You have to compute $\frac{df}{dx}$, you have to compute $\frac{df}{dt}$ derivative of derivatives.

I do not need that when I come here, I just need to evaluate f of x_t okay. At 2 different points, I am going to choose those points in such a way that doing these calculations is same as second order Runge Kutta method. Let us see how it is done okay.

(Refer Slide Time: 44:10)



Now just for the sake of convenience let us call $\Delta x = h \beta k_1$ and $\Delta t = \alpha h$ okay. I am just writing this for the sake of convenience. So my k_2 is function evaluation at $x + \Delta x$, $t + \Delta t$. My k_2 is function evaluation. This is same as function evaluated at well one minute let us put this little more. This is $x_n + \Delta x$ and $t_n + \Delta t$, Δt is some intermediate point, Δh is some intermediate point okay.

So this is same as $f(x_n)$, I am just doing a Taylor series approximation $f(x_n, t_n) + \frac{df}{dx} \Delta x + \frac{df}{dt} \Delta t + \frac{d^2f}{dx^2} \frac{\Delta x^2}{2} + \frac{d^2f}{dt^2} \frac{\Delta t^2}{2} + \dots$. I am just doing a Taylor series approximation of f at $x_n + \Delta x$ and $t_n + \Delta t$ around x_n, t_n okay. I am going to do approximation around x_n, t_n so this is I am just approximating this as $f(x_n, t_n)$. First derivative with respect to t and first derivative with respect to x $\Delta x, \Delta t$.

I am working with scalar; x is the scalar okay. So now if I substitute this k_2 see what I can do now see this k_1 is nothing but just look here k_1 is f_1 okay, k_2 now I have written in terms of what is this? This is f_n this is nothing but f_n okay. So this approximation of k_2 I am going to substitute here okay.

(Refer Slide Time: 47:52)

$$\begin{aligned}
 x(n+1) &= x(n) + h[a k_1 + b k_2] \\
 &= x(n) + a h f(n) \\
 &\quad + b h \left[f(n) + \left(\frac{\partial f}{\partial t} \right)_n (\alpha h) + \right. \\
 &\quad \left. \left(\frac{\partial f}{\partial x} \right)_n (\beta h f(n)) \right] + o(h^3)
 \end{aligned}$$

If I actually substitute for Taylor series approximation of k_2 then take into fact that Δx is h βk_1 and Δt is αh and all that you know if I do all the substitutions, I get this $x_{n+1} = x_n$. I am just rearranging and writing what I get okay by doing substitutions. This is $a h f_n + b h \left[f_n + \left(\frac{\partial f}{\partial t} \right)_n (\alpha h) + \left(\frac{\partial f}{\partial x} \right)_n (\beta h f_n) \right] + o(h^3)$. All that I have done is just substituted okay expansion of k_2 in the neighborhood of t_n, x_n here okay.

And then you know our Δt was αh , our Δx was $\beta h f_n$, all that I have just substituted and rewritten on this okay. Now if you compare this expression, let us compare this expression and expression here on this side okay. Look at the terms that you have $\frac{\partial f}{\partial x} x_n$ okay $\frac{\partial f}{\partial t} t_n$ and f_n okay. Here I have f_n coming in okay, I have $\frac{\partial f}{\partial t} t$ coming in okay.

And at n and $\frac{\partial f}{\partial x} x_n$ coming in. What I am going to do is simply you know rearrange this; I am going simply rearrange this and compare the coefficients of this and this okay. I am just going to rearrange this, compare the coefficients. So if I rearrange this equation I will get something like this, I will write by rearrange form right here.

(Refer Slide Time: 50:37)

$$\begin{aligned}
 x(n+1) &= x(n) + h[a k_1 + b k_2] \\
 &= x(n) + (a+b)h f(n) \\
 &\quad + \left(\frac{\partial f}{\partial t}\right)_n (\alpha b h^2) \\
 &\quad + \left(\frac{\partial f}{\partial x}\right)_n (\beta b h^2 + (n)) + o(h^3)
 \end{aligned}$$

So rearrange form comes out to be $x_{n+1} = x_n + a h f_n + b h f_{n+1} + \alpha b h^2 \frac{df}{dt} + \beta b h^2 \frac{df}{dx} + o(h^3)$. I have just rearranged this and rewritten. Now I want this to be equivalent to second order Taylor series expansion here okay. All that I want to do now is to get this a, b, α, β I am going to compare the coefficients and equate them okay.

So that if you choose a, b, α, β exactly equal to $h^2/2$ and so on then you have an equivalent calculation. Advantage is no derivatives are required okay. This is not the computational form; this is the inner form. You are never going to do these $\frac{df}{dt}$ or $\frac{df}{dx}$. To derive this α, β , we are just doing this okay. So when I equate it I get following equations. See how many terms I have here 1, 2, 3 terms.

There are also 3 terms, I have 4 unknowns. I will get 3 equations in 4 unknowns okay.

(Refer Slide Time: 52:47)

$$\begin{aligned}
 a + b &= 1 && \text{--- (1)} \\
 \alpha b &= \frac{1}{2} && \text{--- (2)} \\
 \beta b &= \frac{1}{2} && \text{--- (3)} \\
 a = 1 - b, \quad \alpha &= \frac{1}{2b} = \beta
 \end{aligned}$$

My first equation is $a + b = 1$ just look at this the multiplying coefficient is h is just 1 so $a + b = 1$ okay. Then $\alpha b = 1/2$ and $\beta b = 1/2$. So I get this 1, 2, and 3 equations okay. I get these 3 equations. I have 4 unknowns and I have 3 equations. How to solve this? Well what I am going to do is I am going to fix one value arbitrarily okay.

Then the remaining 3 will get fixed. I have 1 degree of freedom okay so if I use this 1 degree of freedom okay I will get for 1 particular value of that parameter, the remaining 3 will get fixed okay. Once I use this degree of freedom, I get different algorithms of the class of second order Runge Kutta methods okay. So let us rewrite this as $a = 1 - b$ then $\alpha = 1/2b$ and which is also β .


So if I rearrange these equations as $a = 1 - b$ and $\alpha = \beta = 1/2b$ then I have to choose only b , moment I choose b I get one particular algorithm okay.

(Refer Slide Time: 54:33)

$$b = \frac{1}{2}, \quad a = \frac{1}{2}, \quad \alpha = \beta = 1.$$

Heun's modified Method.



$$x(n+1) = x(n) + h \left[\frac{1}{2} f(n) + \frac{1}{2} f \left(t_n + h, x(n) + h f(n) \right) \right]$$

$$x(n+1) = x(n) + h f(n) + \frac{h^2}{2} \left[\left(\frac{\partial f}{\partial x} \right)_n f(n) + \left(\frac{\partial f}{\partial t} \right)_n \right]$$


So I will just move on. Now if I choose b to be 1/2 then what is a? Then a=1/2 and what are the other 2 parameters? 1 right then alpha=beta=1. What you get after that is called as Heun's modified algorithm. So this is called as Heun's modified method so this will be $x_{n+1} = x_n + h * (1/2 * f_n + 1/2 * f(t_{n+h}, x_n + h f_n))$ okay. So this is what I get. There is one more the different ways you can choose b.

It is a free parameter okay and you can choose it 1/3 and put your name if you want but all of them will be second order Runge Kutta methods okay.

(Refer Slide Time: 57:00)

$$x(n+1) = x(n) + h \left\{ (1-b) f(n) + b f \left[t_n + \frac{h}{2b}, x(n) + \frac{h}{2b} f(n) \right] \right\}$$



So I will write a generic form and so basically my general algorithm is $x_{n+1} = x_n + h$ times $(1-b) f_n + b f$. This is the generic form for different values of b okay. We will get different algorithms, b=1/2 you will get Heun's modified rule, b=1 will give you Euler Cauchy method

and so on okay. All of them are second order Runge Kutta methods okay and they will not give you identical solutions.

But implementing Heun's rule or implementing Euler Cauchy method is equivalent to doing the second order Taylor series approximation okay. Without having to compute derivatives of f with respect to x or with respect to t , you are just doing these calculations okay. That is how all the Runge Kutta methods fourth order, fifth order, sixth order whatever you normally use up to fourth and fifth order, which are derived.

They are derived using Taylor series approximation up to fifth order and matching the coefficients like this okay. So will have a look at yeah **“Professor - student conversation starts.”** Yeah no, it is equivalent to using derivatives up to second order that is what I worried means local derivatives up to second order. Solution will change with b .

So if you do exact Taylor series calculations of second order and you do calculations by this approach with some b you will not get identical solution, they have similar order of inaccuracy okay. You are neglecting terms of $O(h^2)$ and then it is equivalent to doing those derivative calculations. You will not get identical solutions. Each one of them will give a slightly different solution okay **“Professor - student conversation ends.”**