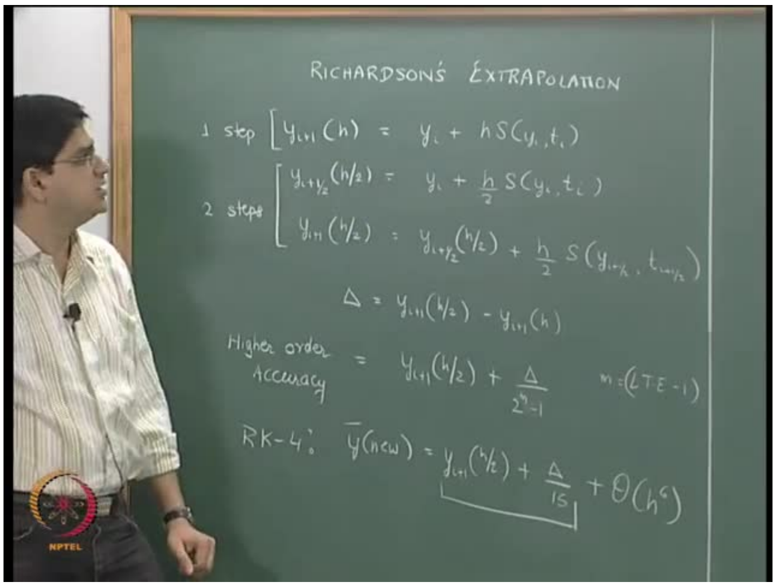
Computational Techniques

Prof. Dr. Niket Kaisare.
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. # 07
Lecture No. # 08
Ordinary Differential Equations
(Initial Value Problems)

Hello and welcome to the module - 7, lecture – 8. What we have been doing so far is, ordinary differential equations the initial value problem. In the previous lecture, we have considered some of the advanced techniques for solving the ordinary differential equations, specifically we started off with the Runge-kutta method and in the previous lecture, we talked about the Richardson's extrapolation and the adaptive step sizing techniques and finally, we discussed very briefly about what is known as stiff system of ODE's.

(Refer Slide Time: 00:45)



RICHARDSON'S EXTRAPOLATION

1 step $\left[\, y_{i+1}(h) = y_i + hS(y_i, t_i) \right.$

2 steps $\left[\begin{array}{l} y_{i+1/2}(h/2) = y_i + \frac{h}{2} S(y_i, t_i) \\ y_{i+1}(h/2) = y_{i+1/2}(h/2) + \frac{h}{2} S(y_{i+1/2}, t_{i+1/2}) \end{array}\right.$

$$\Delta = y_{i+1}(h/2) - y_{i+1}(h)$$

Higher order Accuracy $= y_{i+1}(h/2) + \frac{\Delta}{2^n - 1}$ $\quad n = (LTE - 1)$

RK-4: $\bar{y}(new) = \underbrace{y_{i+1}(h/2) + \frac{\Delta}{15}} + O(h^6)$

So, what we did in Richardson's extrapolation is starting with time t i, we used one iteration of Runge-kutta method or any method for that matter of our choice, and from t i we go to t i plus 1; so within a single step. So, we get y i plus 1 computed at computed using a single step method that we will get it as equal to y i plus h times S of (y i , t i), where this is going to be any method of our choice. So, in the Runge-kutta fourth order method, we will have S as weighted sum of slopes computed at four different points, that means, w 1 k 1 plus w 2 k 2 plus w 3 k 3 plus w 4 k 4 that is what we get for r k r k 4 method.

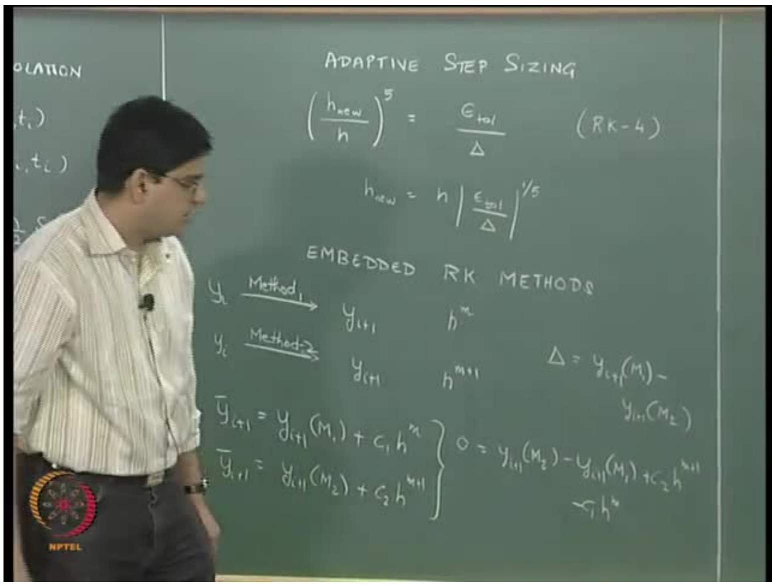
We repeat the exact same procedure again for a different step size - for half the step size. So, we will when we do this for half the step size, we will get y i plus 1 h by 2 equal to y i plus h times S of (y i, t i) or I will write this as y i plus half h by 2 as y i plus h by 2 multiplied by S of (y i , t i). And y i plus 1 computed with slope h by 2 is going to be equal to y i plus half h by 2 plus h by 2 multiplied by S of (y i plus half, t i plus half). So this is using one step; this is using two steps.

Next, we defined our delta as equal to y i plus 1 h by 2 minus y i plus 1 h; this is how we defined our delta. In Richardson's extrapolation, what we do is that the a higher order accurate solution the higher order accurate solution can be obtained as y i plus 1 h by 2 plus del actually not delta by 15 the value of 15 was for r k 4 method 2 to the power m minus 1, where m is local truncation error minus 1.

So, for r k 4 method, the local truncation error is h to the power 5; so m becomes 5 minus 1 that is 4; so 2 to the power 4 minus 1. So, for r k 4, y new is y i plus 1 h by 2 plus delta by 15, and the true value y bar is going to be equal to this guy plus the error and this error is order of or the order of accuracy for this particular method is h to the power 6.

So, we have gone from h to the power 5 accurate method to h to the power 6 accurate method by essentially by using a one-step solution and then, a two-step solution and then, using the difference between these two solutions in order to compute a higher order accurate solution. So, this is what we covered in the Richardson's extrapolation. And then, we said that we can use this value of delta in order to get adaptive step sizing, basically it will give an idea of what the order of accuracy at that particular implementation is going to be.

(Refer Slide Time: 05:30)


ADAPTIVE STEP SIZING
$\left(\frac{h_{new}}{h}\right)^5 = \frac{\epsilon_{tol}}{\Delta}$ (RK-4)
$h_{new} = h\left|\frac{\epsilon_{tol}}{\Delta}\right|^{1/5}$
EMBEDDED RK METHODS
$y_i \xrightarrow{\text{Method}_1} y_{i+1}$ h^m
$y_i \xrightarrow{\text{Method-2}} y_{i+1}$ h^{m+1}
$\Delta = y_{i+1}(M_1) - y_{i+1}(M_2)$
$\bar{y}_{i+1} = y_{i+1}(M_1) + c_1 h^m$
$\bar{y}_{i+1} = y_{i+1}(M_2) + c_2 h^{m+1}$
$0 = y_{i+1}(M_2) - y_{i+1}(M_1) + c_2 h^{m+1} - c_1 h^m$

So, in adaptive step sizing in adaptive step sizing what we said was, h new divided by h to the power 5 is going to be equal to epsilon tol divided by delta. And in order to choose our h, the desired h new we are going to use h new is going to be equal to h multiplied by epsilon tol by delta to the power 1 by 5. So, this is what we obtained when we wanted to use adaptive step sizing for the Runge-kutta fourth order method. This is valid for r k 4 method.

For r k 3, this this is going to be 1 by 4; for r k 2 it is going to be 1 y 3 and so on. This particular exponent basically depends on the order of accuracy of these methods. So, this is essentially what we had covered in the previous lecture what I am going to do do now is talk about what is known as an embed Runge-kutta method. And this is one of the major accomplishments when it comes to the Runge-kutta method is reduction what embedded Runge-kutta method allows us to do is it allows us to reduce the total amount of effort required in computing the adaptive step sizing through this particular technique.

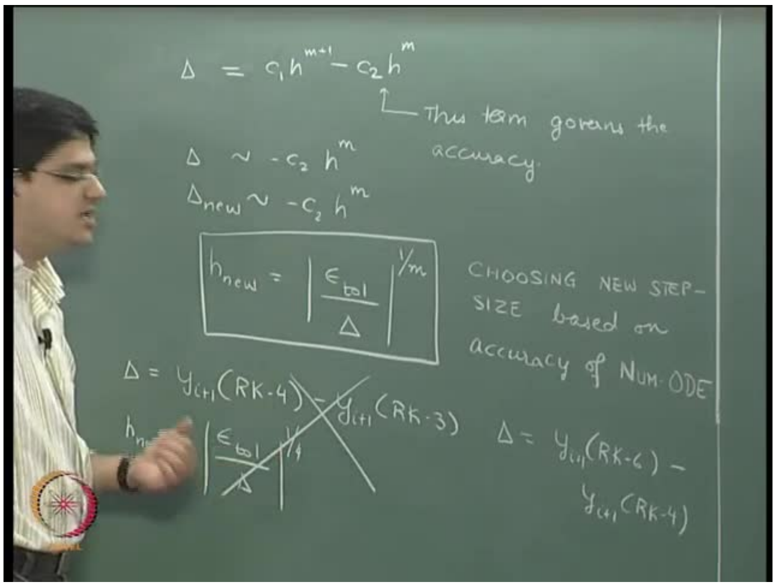
So, I will show you what what we mean by this. So, what we are going to do is, we will start with y 1 and try to use method one to go to y i plus 1 and this particular method is would be of order h to the power m accurate. We will use another method; let us call this as method two to go again from y i to y i plus 1; we are going to use the same same steps size.

So, here what we have done is we went from y i to y i plus 1 using one step and y i to y i plus 1 using two steps. Here we are not going to do that; here we are going to choose 1 r k method in order to go from y i to y i plus 1; we will choose another r k method again to go from y i to y i plus 1. The second r k method has to be more accurate than the first r k method of our choice. So, this particular method let us say the order of accuracy is h to the power m m plus 1.

Now, let us call difference between y i plus 1 using method one, and y i plus 1 using method two as delta. Keep in mind that when we are going to this step size adaptation at that time, we have the absolute values are what we are taking. So, it really does not matter, whether we take the difference m two y i plus 1 using m two minus y i plus 1; using m 1 or the other way round. So, as long as we know what this difference is going to be, we are going to be good.

So, now what happens is that when you take the difference between the two… Now, let us write down what the true value is going to be; the true value y i plus 1 bar is going to be equal to y i plus 1 m 1 plus say C 1 h to the power m, and y i plus 1 the true value over here is going to be y i plus 1 m 2 plus some other constants C 2 h to the power m plus 1. Now, when we subtract these two equations, let let us subtract the first equation from the second equation; when we subtract the first equation from the second equation, we will get 0 equal to y i plus 1 m 2 minus y i plus 1 m 1 plus C 2 h to the power m plus 1 minus C 1 h to the power m.

(Refer Slide Time: 10:53)



$\Delta = c_1 h^{m+1} - c_2 h^m$

This term governs the accuracy.

$\Delta \sim -c_2 h^m$

$\Delta_{new} \sim -c_2 h^m$

$$\boxed{h_{new} = \left| \frac{\epsilon_{tol}}{\Delta} \right|^{1/m}}$$

CHOOSING NEW STEP-SIZE based on accuracy of NUM. ODE

$\Delta = y_{i+1}(RK\text{-}4) - \cancel{y_{i+1}(RK\text{-}3)}$

$h_{n} \cancel{\left| \frac{\epsilon_{tol}}{\Delta} \right|^{1/4}}$

$\Delta = y_{i+1}(RK\text{-}6) - y_{i+1}(RK\text{-}4)$

We rearrange this; this is nothing but negative delta and so we can rearrange this and write 0 equal to minus delta plus c 1 h to the power m plus 1 minus c two h to the power m or we take delta on to the other side and we will be able to write this equation as delta equal to C 1 h to the m plus 1 minus C 2 h to the power m.

Recall what we have been doing so far all the time. What we have been doing so far all the time is the term which has the least accuracy, is the term that is going to govern how good or how bad our solution is going to be. h to the power m plus 1 term is more accurate than h to the power m term for very small values of for small values of h. As a result of this, this is the leading term that governs the accuracy.

So, this term as a result, if we are going to have another delta new equation of this form - delta new is also going to be some constant C 2 multiplied by h to the power m accurate as well as delta is going to be C 2 or rather negative C 2 h to the power m accurate. That is because the most of the error comes in because of neglecting this term, the errors because of neglecting this term are typically going to be an order of magnitude lower than the errors by neglecting this term. That is that is kind of the reasoning that we have been using for so long. And then, when we take the ratio between the two and we use the same manipulations as we have done over here, what we are going to get is that the h new that we have to choose is going to be equal to epsilon tolerance divided by delta absolute value of that raise to the power of 1 by m. So, this is the result for choosing new step size based on the accuracy of results.

So, this is the result we are going to use to choose the new step size based on what we estimate is the accuracy of the numerical method for solving the ODE equation. So for example, we could use r k 4 method and r k 3 method; r k 3 method is order h to the power 4 accurate; r k 4 method is order of h to the power 5 accurate.

So, if delta we we write this as y i plus 1 r k 4 minus y i plus 1 r k 3, then h new is going to be equal to epsilon tol divided by delta absolute value of that to the power 1 by 4 why 1 by 4? The leading error term from this difference is the error term because of the less accurate method r k 3 method. r k 3 method as a local truncation or error of h to the power 4, that means, m over here is equal to 4. So, the h new is going to be equal to epsilon tolerance divided by delta.

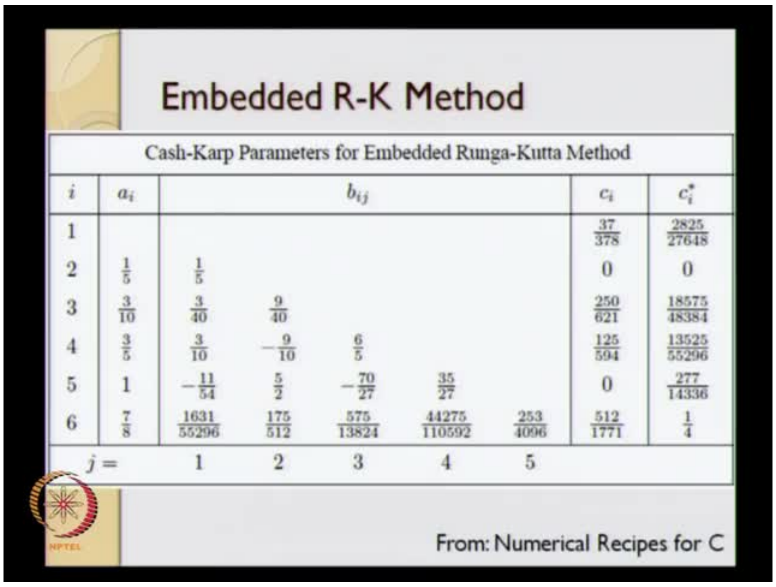
So, this is one way of doing adaptive step sizing the another way, essentially of doing adaptive step sizing. This is not how the embedded r k work, how the embedded r k work is quite a bit bit different, what we try to do is delta we will get as equal to y i plus 1 from an r k 6 method; minus y i plus 1 from an r k 4 method, remember why we are doing r k 6 minus r k 4 and not r k 5 minus r k 4 is for one very simple reason, because r k 6 method is h to the power 6 accurate, whereas r k 5 method has the same order of accuracy as the r k 4 method.

As a result, we are going to take r k 6 minus r k 4 as the the difference delta. Now, when we take this particular difference delta, we will have in in this term one epsilon divided by delta to the power 1 by m, this m is going to be equal to 5 that is because the leading error is governed by the r k 4 method.

But the catch is this; r k 3 and r k 4 methods the the various slopes are actually evaluated at different points. The whole idea behind an embedded Runge-kutta it is all over is essentially this - that r k 6 method and r k 4 method are going to share the data points at which various slope calculations are made and one of the first embedded r k method was what was known as the r k r k r k field burg method.

And after that now days in early 1990's I believe, there was a new new method introduced known as the Cash-Karp modification to the embedded r k r k method. And I will just go over to the slides and show you what what we actually mean by the embedded r k kutta r k methods.

(Refer Slide Time: 16:56)



Embedded R-K Method

Cash-Karp Parameters for Embedded Runga-Kutta Method

i	a_i	b_{ij}					c_i	c_i^*
1							$\frac{37}{378}$	$\frac{2825}{27648}$
2	$\frac{1}{5}$	$\frac{1}{5}$					0	0
3	$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$				$\frac{250}{621}$	$\frac{18575}{48384}$
4	$\frac{3}{5}$	$\frac{3}{10}$	$-\frac{9}{10}$	$\frac{6}{5}$			$\frac{125}{594}$	$\frac{13525}{55296}$
5	1	$-\frac{11}{54}$	$\frac{5}{2}$	$-\frac{70}{27}$	$\frac{35}{27}$		0	$\frac{277}{14336}$
6	$\frac{7}{8}$	$\frac{1631}{55296}$	$\frac{175}{512}$	$\frac{575}{13824}$	$\frac{44275}{110592}$	$\frac{253}{4096}$	$\frac{512}{1771}$	$\frac{1}{4}$
	$j =$	1	2	3	4	5		

From: Numerical Recipes for C

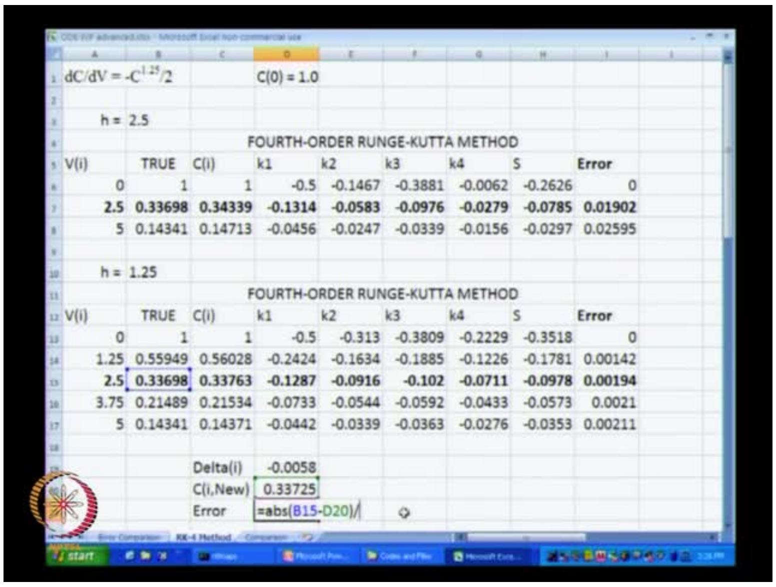
So, these are the parameter values for the Cash-Karp method of embedded r k r k. As as you can see, this is the typical structure that that we usually have, where C's are the weights that we will use; age over here essentially resemble the parameters p, and the p is over here resemble the parameter q that we have been discussing so far for the Runge-kutta methods.

So, what we see over here is that the first fourth points will actually be used by the forth order r k method, whereas all the six points are essentially what will be used by the sixth order r k method. The accuracy of this method will be h to the power 6 with C i star; the accuracy of this method is going to be equal to h to the power 5.

The difference between the two, the delta is going to be governed by the leading error term in in the two equations and the leading error term is going to be h to the power 5. So, what we do is essentially we we calculate the slope at f I; calculate the slope at f i plus 1 by i plus 1 by 5; slope at i plus 3 by 10; i plus 3 by 5; i plus 1, and i plus 7 by 8. And based on these weights, we are going to finally calculate what the slope is going to be. This particular slope is going to give us h to the power 5 accurate results and the next slope is going to give us h to the power 6 accurate results, and difference between the two is going to determine what our next step size is going to be.

So, this is essentially what I want to discussing about the adaptive step sizing, about improving the accuracy using the Richardson's extrapolation and finally, about stiff ODE solvers. What we will do next is, we will take up one simple example and try to solve that particular the same example, in fact that we have been talking about so far. We will take that particular example once again; we will try to solve these equations using the r k 4 method and then, apply Richardson's extrapolation to r k 4 method and see how the r k 4 method results are actually improved by applying the Richardson's extrapolation. That is what we will do one; in the second thing we will do is also then use the same results in order to get the adaptive step sizing and determine what the next step size is going to be.

(Refer Slide Time: 19:38)



$dC/dV = -C^{1.25}/2$ C(0) = 1.0

h = 2.5

FOURTH-ORDER RUNGE-KUTTA METHOD

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**
5	0.14341	0.14713	-0.0456	-0.0247	-0.0339	-0.0156	-0.0297	0.02595

h = 1.25

FOURTH-ORDER RUNGE-KUTTA METHOD

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**
3.75	0.21489	0.21534	-0.0733	-0.0544	-0.0592	-0.0433	-0.0573	0.0021
5	0.14341	0.14371	-0.0442	-0.0339	-0.0363	-0.0276	-0.0353	0.00211

Delta(i)	-0.0058
C(i,New)	0.33725
Error	=abs(B15-D20)/

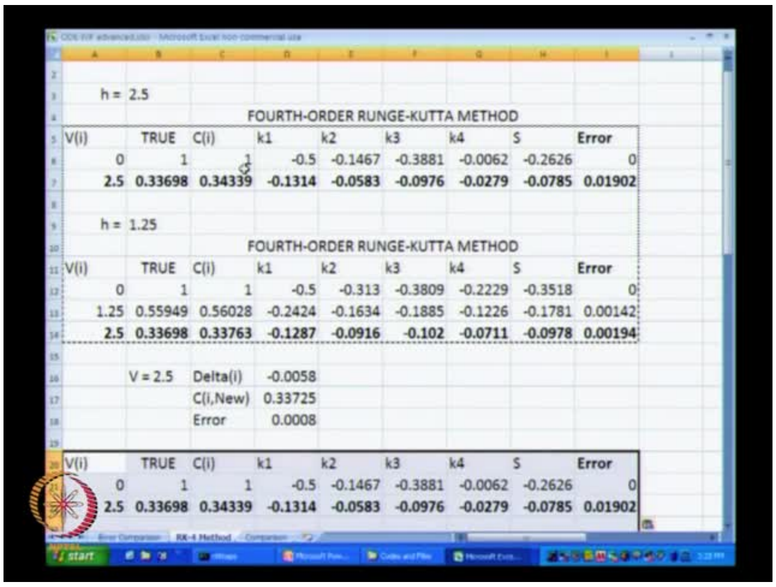
So, let us now go to Microsoft excel what I will do over here… So, this is the I believe, this is the sheet that we use in the previous lecture in order to compare the global and local truncation errors. So, what we want to do over here excuse me. So, what we want do over here is compute the concentration at volume V equal to 5 starting with the concentration C 0 equal to 1 . 0. What we will do is, we will may be, we will take a step of h equal to 5 and then, well actually we cannot take a step h equal to 5. Let us take h equal to 2. 5 and then, repeat this with a step of h equal to 1. 25. I am doing this so that the difference between the two methods becomes very evident; of course this has not changed, because we have included dollar signs over here.

So, I will just drag this and that is really the beauty of microsoft excel excel is its very easy in order f we have to really solve this particular relatively easier problems . And k 1 is f computed at C i; k 2 requires h so i will just go over here and i will drag it down. Same thing we will do for k 3, I will take this and drag it over here; same I will do for k 4, I will drag this over here, f s is nothing but i weighted average of these. So, it is not does not use h over here at all and I will delete these values. Now, C i plus 1 is going to be equal to C i plus h times S.

So, I will drag to this new h value. Now, we are set; all we need to do is really take this value and just drag it below and then, we take all these values and drag them and this is going to be the error. So, now, what we have done is, we have computed V at 2.5 and V at 5 using steps of h equal to 2.5 and step of h equal to 1.2 5. So, if we were to look at the values over here, itself these are the two values that we are going to get. So, the delta is going to be the difference between the C i computed from two step procedure and the C i computed from a single step procedure.

So, delta at location i is going to be equal to C i using h 2 minus C i using h 1 sorry h by 2 and C i using h so that is going to be our delta. And C i C i new is going to be equal to C i using h by 2 plus delta divided by 15 and this is our this is the value that we are going to get. Error, we have as we have computed over here, the error is the absolute value of the difference between true and C i divided by the true value over here. So, the error that we will compute is going to be equal to a b s of true minus the newly computed value divided by the true value.

(Refer Slide Time: 24:29)



h = 2.5

FOURTH-ORDER RUNGE-KUTTA METHOD

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

FOURTH-ORDER RUNGE-KUTTA METHOD

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

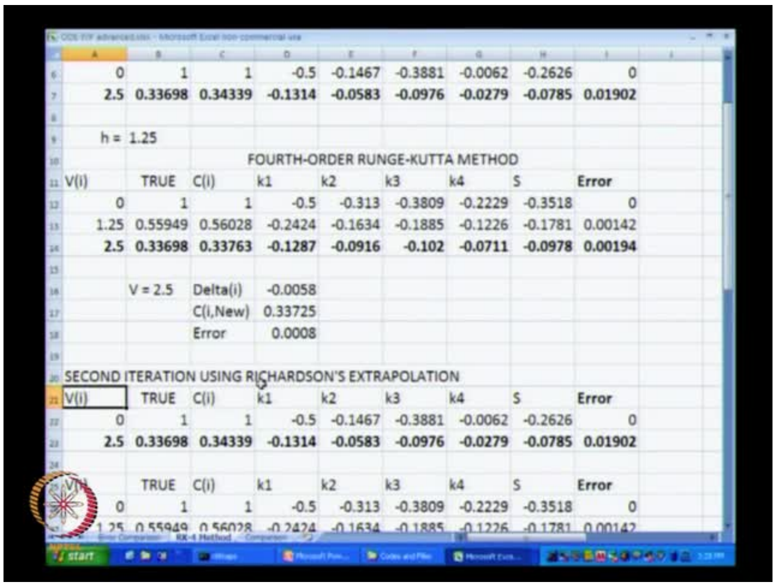
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

So, as you can see what happened was our error was 2 into 10 to the power minus 2 when when we used sorry 1 into 10 yes, in fact it was 2 into 10 to the power minus 2 when we used h equal to 2. 5; when we decrease the h from 2. 5 to 1. 2 5 the h, the error decreased from 2 e minus 2 to 2 e minus 3.

And when we took the difference between the two and we added that difference to to the the more accurate value over here; at that time the error then reduced further to 8 into 10 to the power minus 4. So, as you can see this is the result that you get for i e at at volume V equal to 2.5 using the Richardson's extrapolation method.

Next, what we do? In order to continue this further is we we do not use this particular C i we are actually going to use the more accurate C i in in its place in this and then, we are going to continue this further. as i said As I had mentioned before, its actually much more tedious to calculate this using Microsoft excel and that is the reason why we have not done it. Let us give it a shot; let us see whether we are able to do do this calculation or not. I am deleting those values really because those values are not going to be needed. Now, I am going to copy this and pasted below.

(Refer Slide Time: 26:16)



V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

FOURTH-ORDER RUNGE-KUTTA METHOD

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

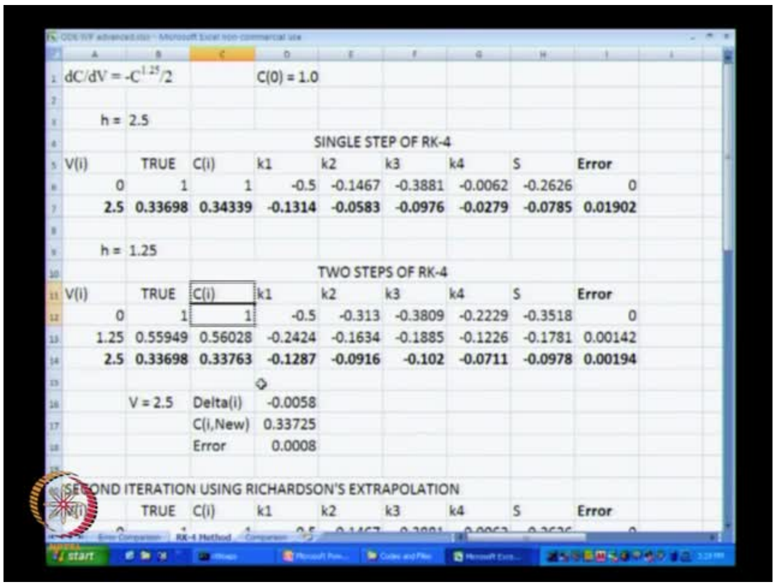
SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142

And we are going to start with value of V i and again, this is also not something you need; so we will delete this again and I will write the second iteration using Richardson's extrapolation.

(Refer Slide Time: 26:54)



$dC/dV = -C^{1.25}/2$ $\quad$ $C(0) = 1.0$

h = 2.5

SINGLE STEP OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

TWO STEPS OF RK-4

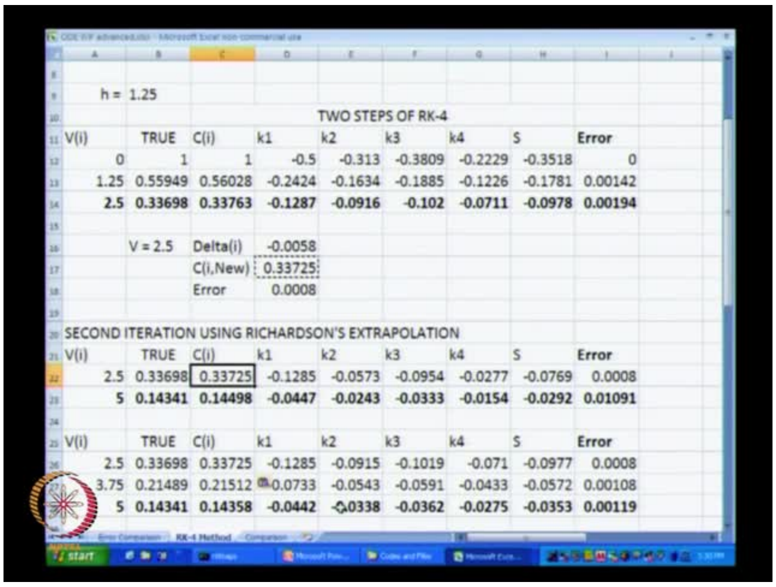
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error

(Refer Slide Time: 27:22)


h = 1.25

TWO STEPS OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

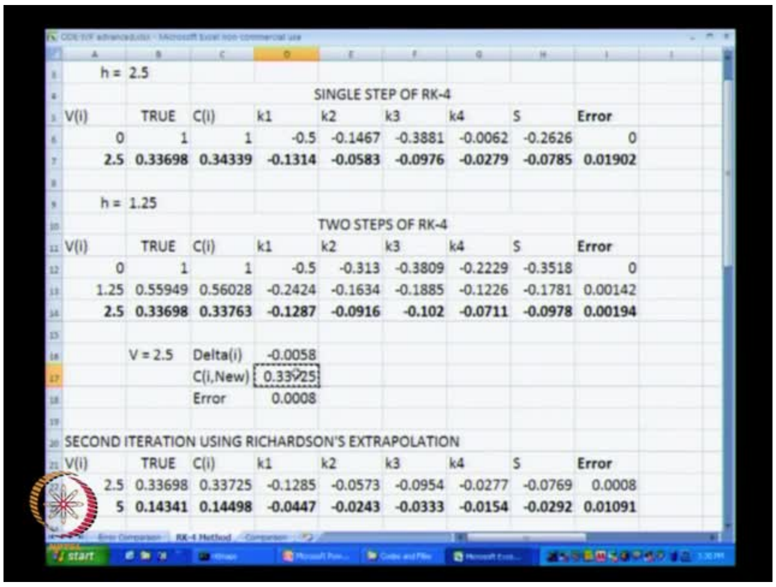
SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0573	-0.0954	-0.0277	-0.0769	0.0008
5	**0.14341**	**0.14498**	**-0.0447**	**-0.0243**	**-0.0333**	**-0.0154**	**-0.0292**	**0.01091**

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0915	-0.1019	-0.071	-0.0977	0.0008
3.75	0.21489	0.21512	-0.0733	-0.0543	-0.0591	-0.0433	-0.0572	0.00108
5	**0.14341**	**0.14358**	**-0.0442**	**-0.0338**	**-0.0362**	**-0.0275**	**-0.0353**	**0.00119**

So, the first iteration we went from 0 to 2. 5 using the two methods; one was a one-step method and the other was a two-step method. I will write those things down also; single step of r k 4 and two steps of r k 4. So, V is going to be now equal to 2.5 and the new V is going to be 5. And likewise, here the V is also going to be 2. 5. What I will do next is copy this value of C i new and then, paste it over here; paste special and values I do not want the formula to be pasted, I only want the value to be pasted paste special and values.

(Refer Slide Time: 28:13)


h = 2.5

SINGLE STEP OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

TWO STEPS OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

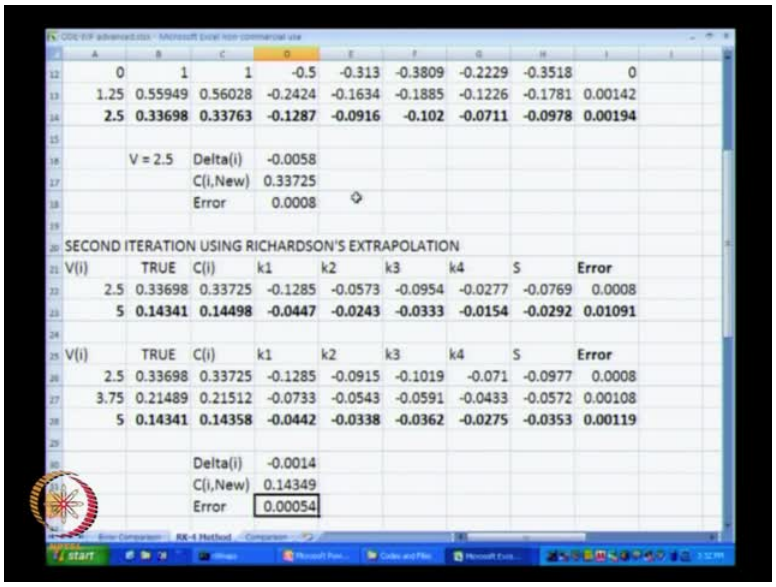
V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0573	-0.0954	-0.0277	-0.0769	0.0008
5	**0.14341**	**0.14498**	**-0.0447**	**-0.0243**	**-0.0333**	**-0.0154**	**-0.0292**	**0.01091**

So, in the second iteration using the Richardson's extrapolation, extrapolation keep in mind what we are not doing over here; we are not running the two r k 4 methods independent of each other, we run the two r k 4 methods once. We get the solution of C i numerically computed C i here and here. We use these two numerical solutions in order to get the new C i; this new C i is more accurate than any of these two C i's.

(Refer Slide Time: 28:47)



0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**
	V = 2.5	Delta(i)	-0.0058					
		C(i,New)	0.33725					
		Error	0.0008					

SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0573	-0.0954	-0.0277	-0.0769	0.0008
5	**0.14341**	**0.14498**	**-0.0447**	**-0.0243**	**-0.0333**	**-0.0154**	**-0.0292**	**0.01091**

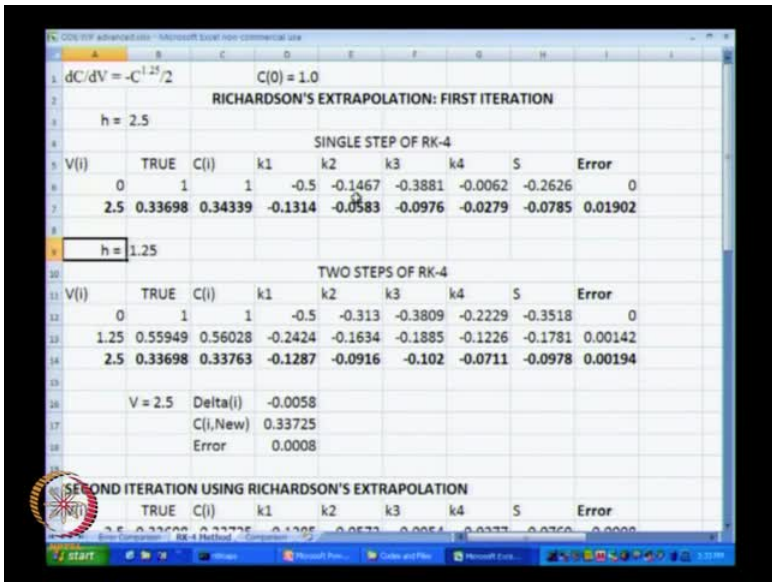
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0915	-0.1019	-0.071	-0.0977	0.0008
3.75	0.21489	0.21512	-0.0733	-0.0543	-0.0591	-0.0433	-0.0572	0.00108
5	**0.14341**	**0.14358**	**-0.0442**	**-0.0338**	**-0.0362**	**-0.0275**	**-0.0353**	**0.00119**
		Delta(i)	-0.0014					
		C(i,New)	0.14349					
		Error	0.00054					

Now, this is the best estimate of C i that we have at volume 2.5; this is the estimate of C i that we are going to use again with the r k 4 method using single step h and r k 4 the same r k 4 method using two steps h. So, we have copied this particular C i over here as well as over here; now we run this again.

So, we have run this from 2.5 to 5 and we have run from to again from 2.5 to 5, first using one step, next using two step method. Now, what we will need is, we will need the same to copy the same thing down over here and then, edit the results, the delta is nothing but this minus this. So, delta is the difference between the concentration computed using the two step method minus the concentration computed using one step method

C i new is going to be the C i computed using two step method plus delta divided by 15 and the error is going to be the C i new minus the true value absolute value of that divided by the true value and that is going to be our error. As you can see again, the error for the Richardson's extrapolation method is 5 into 10 to the power minus 4, whereas the error using the two steps of r k 2 r k 4 method was 1 e minus 3, whereas using single step of r k 4 method was 1 e minus 2. So, we have improved the accuracy from 10 to the power minus 3 to 10 to the power minus 4, but this allows us to do essentially is this. It allows us to take larger step sizes when we are try to do this computations using the r k method.

(Refer Slide Time: 30:39)



$dC/dV = -C^{1.25}/2$ C(0) = 1.0

RICHARDSON'S EXTRAPOLATION: FIRST ITERATION

h = 2.5

SINGLE STEP OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

TWO STEPS OF RK-4

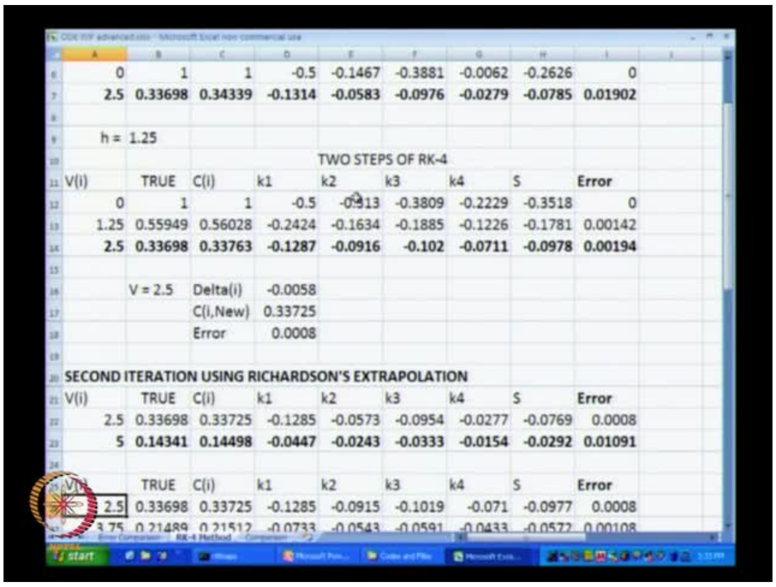
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error

(Refer Slide Time: 31:07)



0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**

h = 1.25

TWO STEPS OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.0058
	C(i,New)	0.33725
	Error	0.0008

SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION

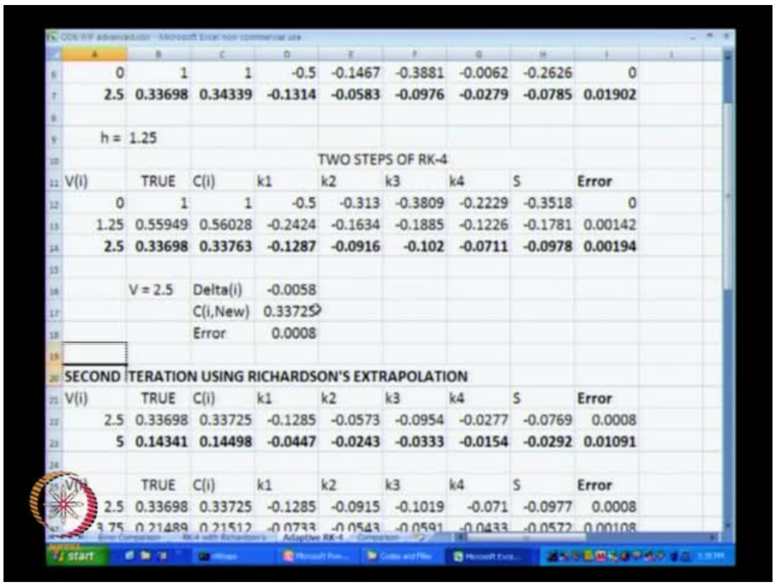
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0573	-0.0954	-0.0277	-0.0769	0.0008
5	**0.14341**	**0.14498**	**-0.0447**	**-0.0243**	**-0.0333**	**-0.0154**	**-0.0292**	**0.01091**

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0915	-0.1019	-0.071	-0.0977	0.0008
3.75	0.21489	0.21512	-0.0733	-0.0543	-0.0591	-0.0433	-0.0572	0.00108

So, this is really what we have to discuss we I have to discuss about the Richardson's extrapolation. Richardson's extrapolation first iteration and this is the result. So, what we have seen now is this; you start off with two different ways of computing any particular ODE solution- one is has a higher accuracy than the other. You take the difference between the two; now this difference between the two says something about the accuracy of the overall numerical technique.

This can this particular estimate of accuracy can be used in two ways: one it can be used to improve the accuracy of the method that we are trying to use and that is using the Richardson's extrapolation. An alternative is to go to adaptive step sizing in order to improve the step size based on what we want or what we desire the accuracy of the overall solution technique to be. So, the first part, where we use Richardson's extrapolation to get a more accurate result is what I have just covered.

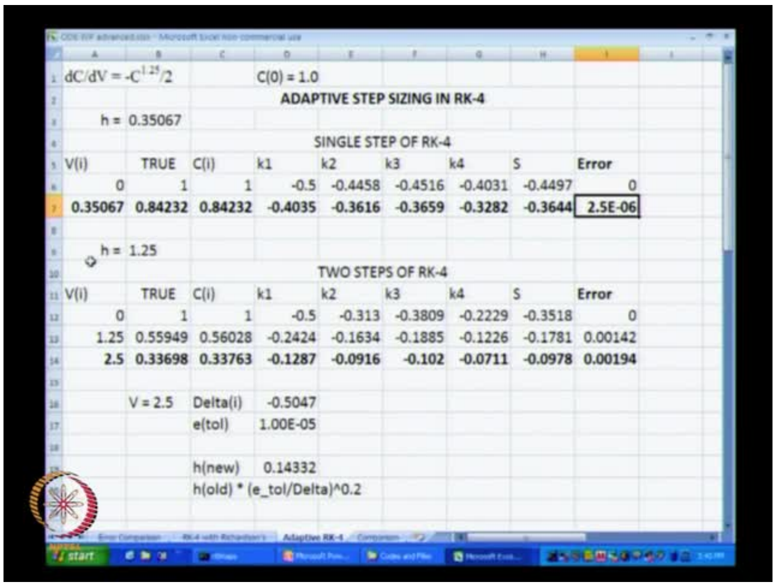
(Refer Slide Time: 32:19)



0	1	1	-0.5	-0.1467	-0.3881	-0.0062	-0.2626	0
2.5	**0.33698**	**0.34339**	**-0.1314**	**-0.0583**	**-0.0976**	**-0.0279**	**-0.0785**	**0.01902**
h =	1.25							
				TWO STEPS OF RK-4				
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**
	V = 2.5	Delta(i)	-0.0058					
		C(i,New)	0.33725					
		Error	0.0008					
SECOND ITERATION USING RICHARDSON'S EXTRAPOLATION								
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0573	-0.0954	-0.0277	-0.0769	0.0008
5	**0.14341**	**0.14498**	**-0.0447**	**-0.0243**	**-0.0333**	**-0.0154**	**-0.0292**	**0.01091**
V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
2.5	0.33698	0.33725	-0.1285	-0.0915	-0.1019	-0.071	-0.0977	0.0008
3.75	0.21489	0.21512	-0.0733	-0.0543	-0.0591	-0.0433	-0.0572	0.00108

Now, what I will do is again we will go back to excel and cover the adaptive step sizing. The good thing now is I do not have to do a whole lot; I have to just I will just copy this.

(Refer Slide Time: 33:05)



$dC/dV = -C^{1.25}/2$ C(0) = 1.0

ADAPTIVE STEP SIZING IN RK-4

h = 0.35067

SINGLE STEP OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.4458	-0.4516	-0.4031	-0.4497	0
0.35067	**0.84232**	**0.84232**	**-0.4035**	**-0.3616**	**-0.3659**	**-0.3282**	**-0.3644**	**2.5E-06**

h = 1.25

TWO STEPS OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.5047
	e(tol)	1.00E-05
	h(new)	0.14332
	h(old) * (e_tol/Delta)^0.2	

So, copy this, create a copy and this has now created a copy and I will rename this as adaptive r k 4 and the previous copy I will rename this as r k 4 with Richardson's. Now, let us go to the adaptive step sizing, we do not need any of this; I will I will show you the adaptive step sizing for one iteration only. So, now, we have adaptive step sizing in r k 4 and we have computed the r k 4 solution using h equal to 2.5 and h equal to 1.2 5.

Let us say that the desired accuracy is going to be equal to 10 to the power minus 5. I have chosen the desired accuracy of 10 to the power minus 5 just to demonstrate how this particular method is going to work. Its two orders of magnitude higher than the accuracy is two order of magnitude lower sorry than the accuracy that our method is giving.

So, desired or rather epsilon tol it is going to be 1 e minus 5 so that is the desired accuracy. I will just take this; pull this over here; we do not we do not need this C i new and error actually. So, our desired tolerance value is 10 to the power minus 5, whereas the delta that we are getting, the difference between the C i and computed using h by 2 and the C i computed using h is 5 into 10 to the power minus 3. So, h new is what we will compute and I will just increase the font size over here; recall h new is nothing but h old multiplied h old by epsilon tol divided by delta raise to the power 0.25.

So that is what I am going to compute. So, h new is going to be equal to h old, which is 1. 25 multiplied by a b s - absolute value of epsilon tolerance - divided by delta to the power 0. 2. So, the h new that we get is going to be 0. 35. So, what this says is that, in order to get the desired accuracy, we are using very large step sizes over here.

We are to be using much smaller step size than the step sizes that we have used. So, we need to decrease the step size to 0. 35. So, what I will do over here is I will go back at this step and give h equal to 0. 35067 and when I give that, I now get the desire the accuracy that now I get is higher in fact than the accuracy that I have asked my particular method to give.

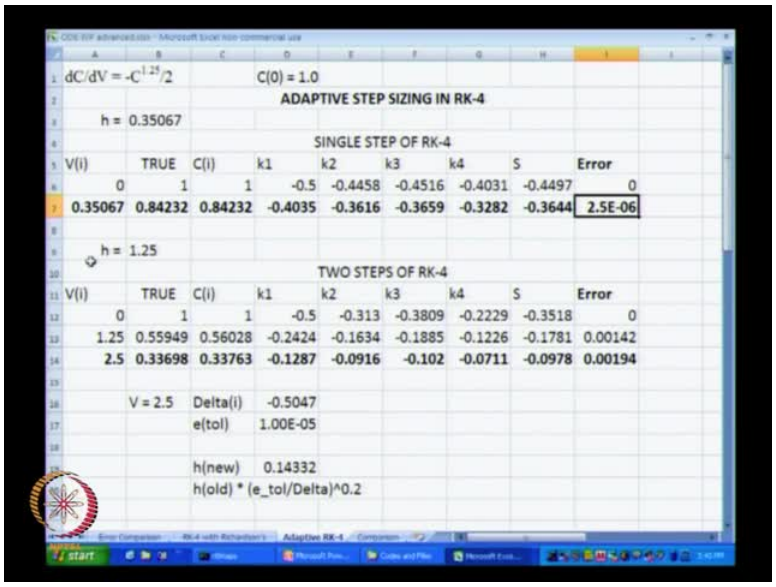
So, with this particular h value now so what we have we did is this; we used initially the h value of 1 .2 5 and h value of 2.5. What we then realize that the accuracy is not very good; the accuracy that this particular method provides is much less than what the accuracy that we have asked our method to give and when we do that what what we actually get is that, we get an error which is fairly low value.

So, this is what this is how we are going to proceed with our numerical method. So, once we could use this h equal to 0 .3 5, we go on to h i plus 1; we will use the h equal to 0.35 once again and h equal to 0.7 once again. So, from 0. 3 5 volume will go to volume of 0. 3 5 to 0 .7, and from 0 .7 to 1. 05.

So, we are going from 0. 3 5 to 1. 0 5 in two steps, then we will go from 0.35 to 1.05 in a single step. Find delta value by taking the difference between the concentration values that we obtained over here and based on this delta value and based on this particular formula, we will compute the new h.

And we will keep repeating this over and over again until we reach the volume V equal to 5. What I have shown you hopefully were that you have gathered Richardson’s extrapolation method is that by using this method, you can we can get a higher order accurate solution by using two low order accuracy solutions that is what the aim of Richardson’s extrapolation is.

(Refer Slide Time: 39:30)



$dC/dV = -C^{1.25}/2$ C(0) = 1.0

ADAPTIVE STEP SIZING IN RK-4

h = 0.35067

SINGLE STEP OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.4458	-0.4516	-0.4031	-0.4497	0
0.35067	**0.84232**	**0.84232**	**-0.4035**	**-0.3616**	**-0.3659**	**-0.3282**	**-0.3644**	**2.5E-06**

h = 1.25

TWO STEPS OF RK-4

V(i)	TRUE	C(i)	k1	k2	k3	k4	S	Error
0	1	1	-0.5	-0.313	-0.3809	-0.2229	-0.3518	0
1.25	0.55949	0.56028	-0.2424	-0.1634	-0.1885	-0.1226	-0.1781	0.00142
2.5	**0.33698**	**0.33763**	**-0.1287**	**-0.0916**	**-0.102**	**-0.0711**	**-0.0978**	**0.00194**

V = 2.5	Delta(i)	-0.5047
	e(tol)	1.00E-05
	h(new)	0.14332
	h(old) * (e_tol/Delta)^0.2	

And I hope what I have conveyed to you using the adaptive step sizing example is that, if we choose a step size too large for example, a step size of 1. 2 5 or the plug flow reactor example is indeed too large at under those conditions, if we use a very large step size, we can use this adaptive step size to reduce the next step size that we are going to use.

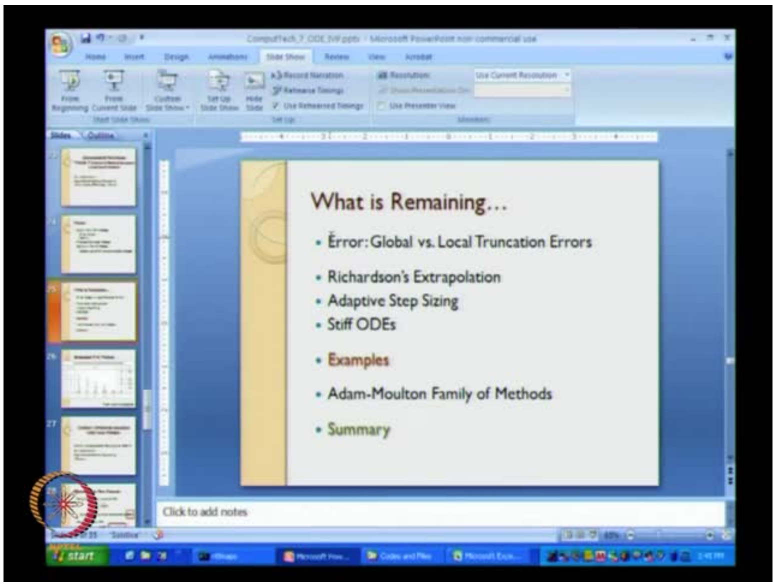
Likewise, if you were to use a very small step size, let us say we were to use step size of 0. 1, we would have been able to increase that step size from 0.1 to a higher value. In the next step, in fact I can we can actually quite easily, perhaps demonstrate that if we start with an h of 0.1 and an h of 0. 05, then the delta accuracy that we are going to get or sorry from h of 0.1, let us try 0 .2 and then, we will try h of 0. 1 so the delta accuracy that we are getting is 10 to the power minus 7 and the the tolerance that we have asked is 10 to the power minus 5, and the h that we are going to get is going to be equal to 0. 1 multiplied by this particular value and the steps the the result that we will get over here is going to be 0 .2 4.

So, as you see, instead of choosing a step size of 0. 1, our adaptive step size sizing technique has said that it would have been ok if we had chosen the step size of 0.24 as well. So that is essentially what how the adaptive step sizing actually works. if you If we look at some of the more theoretical results from the numerical recipes text books, they will recommend, that when you are increasing the step size, we increase it in a more conservative way for increasing the step size. We are going to have this to the power 0 .

2 5 and when we we are decreasing the step size sorry when we are increasing the step size, we can it is going to be h to the power 1 by 4; and when we are decreasing the step size, it is going to be h to the power 1 by 5.

There are theoretical reasons as well as practical reasons why these recommendations are given, i would not of course I do not would not go into details because already adaptive step sizing is indeed a fairly advanced topic for a typical under graduate course.

(Refer Slide Time: 41:35)



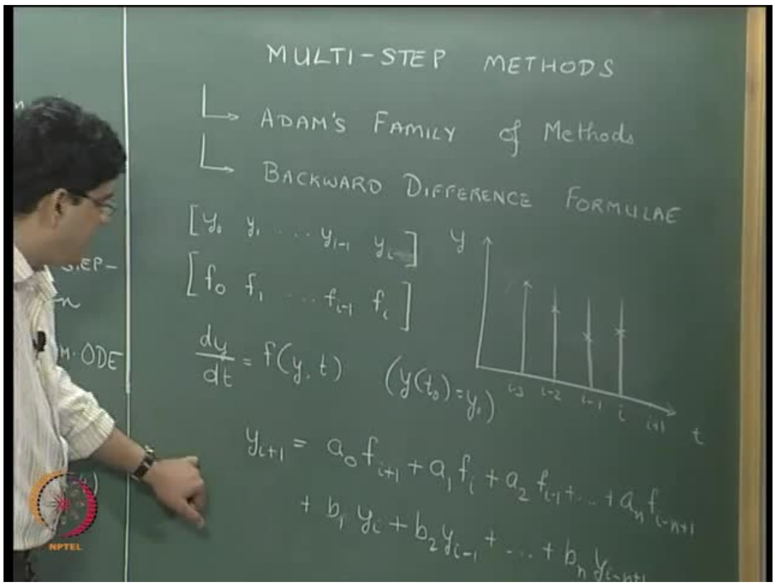
What is Remaining…

- Error: Global vs. Local Truncation Errors
- Richardson's Extrapolation
- Adaptive Step Sizing
- Stiff ODEs
- Examples
- Adam-Moulton Family of Methods
- Summary

Click to add notes

So that is what I had to cover on adaptive step sizing. Now, we will go on to new set of techniques as I had said before… we will about Let us recapping what we had. So, what we have covered is in the previous lecture are these methods. In today's lecture, we have talked about Richardson's extrapolation adaptive step sizing and taken up examples. Now, what is left in this particular module is talk about the Adam-Moulton's family of methods and that is what I am going to do in the rest of today's lecture and in the next lecture.

(Refer Slide Time: 42:16)



MULTI-STEP METHODS

↳ Adam's Family of Methods

↳ Backward Difference Formulae

$[y_0 \; y_1 \; \dots \; y_{i-1} \; y_i]$

$[f_0 \; f_1 \; \dots \; f_{i-1} \; f_i]$

$\frac{dy}{dt} = f(y, t) \quad (y(t_0) = y_0)$

$$y_{i+1} = a_0 f_{i+1} + a_1 f_i + a_2 f_{i-1} + \dots + a_n f_{i-n+1} + b_1 y_i + b_2 y_{i-1} + \dots + b_n y_{i-n+1}$$

So, the methods that we have considered so far come under the the umbrella of the Runge-kutta family of methods as well as after that, we covered the predictor-corrector methods, the third class of methods is essentially what is known as multi-step methods. So, the multi-step methods they are two types: one is would be the Adam's family of methods and the other one is backward difference formula.

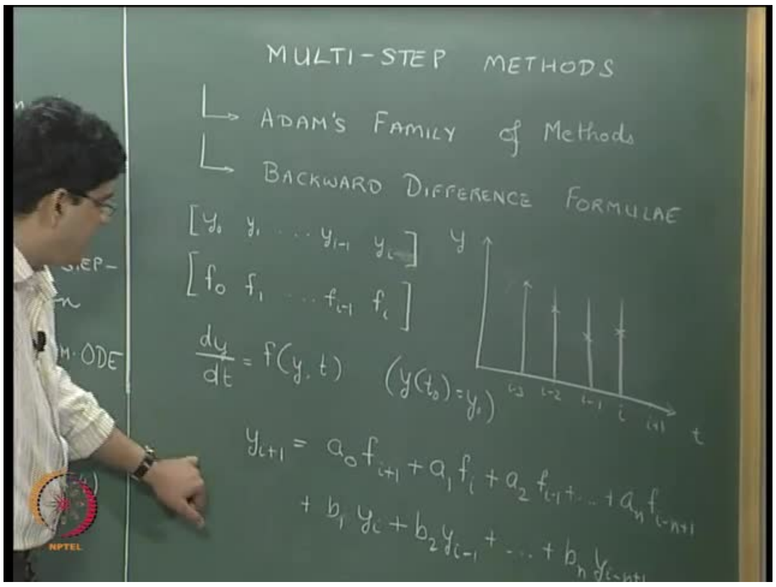
A general way of writing the multi-step method is as follows, I will first talk about a little about the geometric interpretation of this and then, we will talk about how exactly we are going to get. Let us say we are currently at location i, the next one is location i plus 1; likewise, the previous locations are i minus 1, i minus 2 so on and so forth. So, i minus 1, i minus 2, i minus 3 bla bla bla.

So, let us say we are going to plot y versus t; so y versus t that that we we will get, let say we have reached y versus t in this; we actually would not reached like this. So, what we have is i i minus 1 i minus 2 and i minus 3 and let us say we have the y i computed at i minus 3 i minus 2 i minus 1 and i. Now, what we can do is, we can use this information. So, what information is available to us now is y 0 y 1 so on up to y i minus 1 and y i minus sorry y i. So, this is one set of information available to us. We also know t 0 t 1 t 2 up to t i this information using this information what is also available to us is, f 0 f 1 f i minus 1 and f i, both these information are information are available to us. Recall that the equation that we are trying to solve is d y by d t is equal to f of (y, t) given y at t 0 equal to y 0. So, this is the equation that we are we are trying to solve. Now, given y 0, we can compute all these values y 1 y two y 3 up to y i and having these values known to us, we can get f 0 f 1 f f 2 and so on up to f i.

The multi-step methods what we try to do is this y i plus 1, we will write this as some kind of a weighted sum of the past wise as well the past f i. So, we will write this as a 0 f i plus 1 plus a 1 f i plus a 2 f i minus 1 plus so on up to a n f i minus n plus 1. oh sorry i minus yeah i minus n minus 1 no i minus n plus 1

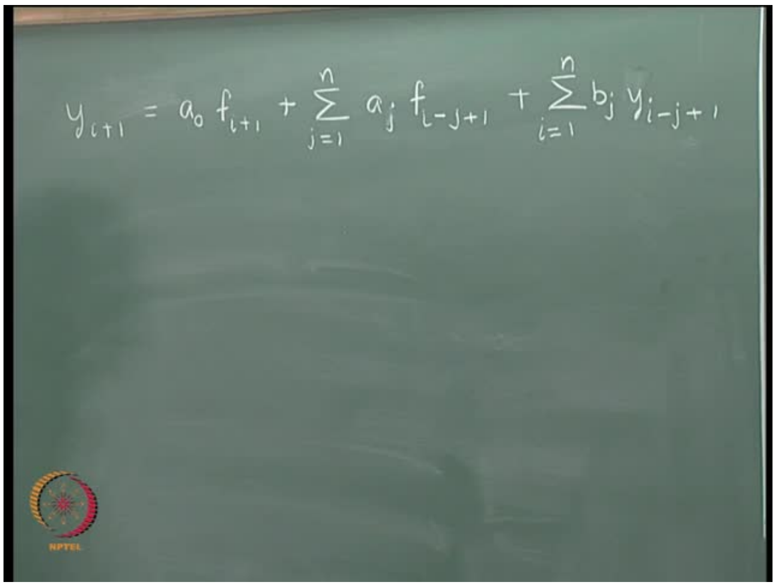
So, this is what we will we will get over here, that is the weighted average of the past function values plus b 0 y i or sorry we will write this as plus b 1 y i plus b 2 y i minus 1 plus so on up to b n y i minus n plus 1.

(Refer Slide Time: 47:01)



MULTI-STEP METHODS
ADAM'S FAMILY of Methods
BACKWARD DIFFERENCE FORMULAE
$[y_0\ y_1\ \dots\ y_{i-1}\ y_i]$
$[f_0\ f_1\ \dots\ f_{i-1}\ f_i]$
$\frac{dy}{dt} = f(y,t)\quad (y(t_0)=y_0)$
y
t
$y_{i+1} = a_0 f_{i+1} + a_1 f_i + a_2 f_{i-1} + \dots + a_n f_{i-n+1}$
$+ b_1 y_i + b_2 y_{i-1} + \dots + b_n y_{i-n+1}$

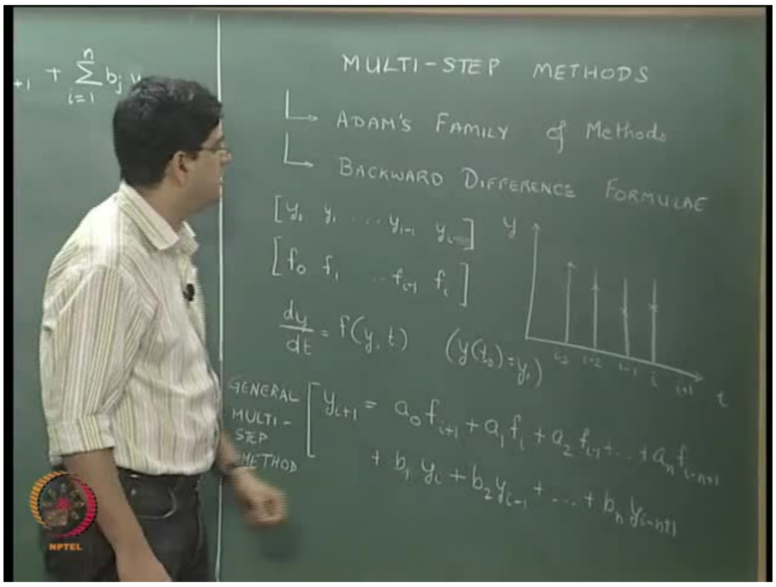
(Refer Slide Time: 47:21)



$$y_{i+1} = a_0 f_{i+1} + \sum_{j=1}^{n} a_j f_{i-j+1} + \sum_{i=1}^{n} b_j y_{i-j+1}$$

So, we will write as sum of the past function values and the sum of the past y values and this is the overall general equation for any multi-step method. So, this particular expression that we had written over here, I am writing it using the summation symbols this like this.

(Refer Slide Time: 48:20)



MULTI-STEP METHODS

$\hookrightarrow$ ADAM'S FAMILY of Methods

$\hookrightarrow$ BACKWARD DIFFERENCE FORMULAE

$[y_0 \; y_1 \; \ldots \; y_{i-1} \; y_i]$

$[f_0 \; f_1 \; \ldots \; f_{i-1} \; f_i]$

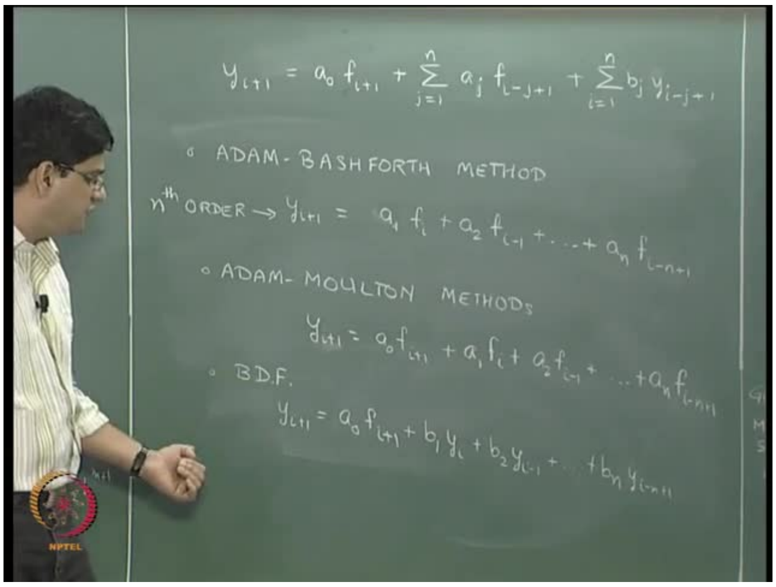
$\frac{dy}{dt} = f(y, t) \quad (y(t_0) = y_0)$

GENERAL MULTI-STEP METHOD

$$y_{i+1} = a_0 f_{i+1} + a_1 f_i + a_2 f_{i-1} + \ldots + a_n f_{i-n+1} + b_1 y_i + b_2 y_{i-1} + \ldots + b_n y_{i-n+1}$$

I have written down this particular equation over here; now I will just go on and explain how this Adams family of methods and the backward difference formulae of formulae methods are going to differ from each other and what that means really.

(Refer Slide Time: 48:28)



$$y_{i+1} = a_0 f_{i+1} + \sum_{j=1}^{n} a_j f_{i-j+1} + \sum_{j=1}^{n} b_j y_{i-j+1}$$

∘ ADAM-BASHFORTH METHOD

n^{th} ORDER $\rightarrow y_{i+1} = a_1 f_i + a_2 f_{i-1} + \dots + a_n f_{i-n+1}$

∘ ADAM-MOULTON METHODS

$$y_{i+1} = a_0 f_{i+1} + a_1 f_i + a_2 f_{i-1} + \dots + a_n f_{i-n+1}$$

∘ B.D.F.

$$y_{i+1} = a_0 f_{i+1} + b_1 y_i + b_2 y_{i-1} + \dots + b_n y_{i-n+1}$$

So, first we will talk about Adam-Bashforth method. In Adam-Bashforth method what we do is, we say that the a 0's are a 0 is 0 and b j's are 0. So, y i plus 1 is going to be computed based on the past f values or past slopes at the points that we have already computed. So, this term disappears in an Adam-Bashforth method; this term also disappears in the Adam-Bashforth method and we will get y i plus 1 equal to sorry equal to a 1 f a a 1 f i plus a 2 f i minus 1 plus so on up to a n f i minus n plus 1 and this is going to be an nth order Adam-Bashforth method.

Now, this method is an explicit method, because y i plus 1 depends only on the past points and we are assuming over here that the past points are known to us. So, this is going to be an explicit method. If a 0 is not equal to 0 what we get are implicit methods and that is the next family of methods as known as Adam-Moulton's methods. And the nth order Adam-Moulton's method is going to be y i plus 1 equal to a 0 f i plus 1 plus a 1 f i plus a 2 f i minus 1 and so on up to a n f i minus n plus 1.

So, these two are the Adam's family of methods and finally, we have the backward difference formula, and the idea behind the backward difference formula is we do not worry about the function values in the past, we only worry about the y i's in the past. So, the backward difference formula is going to be y i plus 1 is going to be a 0 f i plus 1 plus b 1 y i plus b 2 y i minus 1 and so on up to b n y i minus n plus 1.

And so, we will an nth order backward difference formula. The way of computing the backward difference formula is fairly straight forward. What we do is, we describe d f by d t equal to nothing but ah d y by d t as nothing but the numerical difference between the past points and we will use this in order to derive the backward difference formula. Adam-Bashforth method and Adam-Moulton's method we will be using the Newton's backward difference interpolating polynomials; we will fit them to the past function values f i f i minus 1 up to f i minus n plus 1 and we will use those results and integrate them in order to get y i plus 1 using the Adam-Bashforth or using the Adam-Moulton's method. The derivation of these methods is something that i will consider I will cover in next lecture.

Thank you.