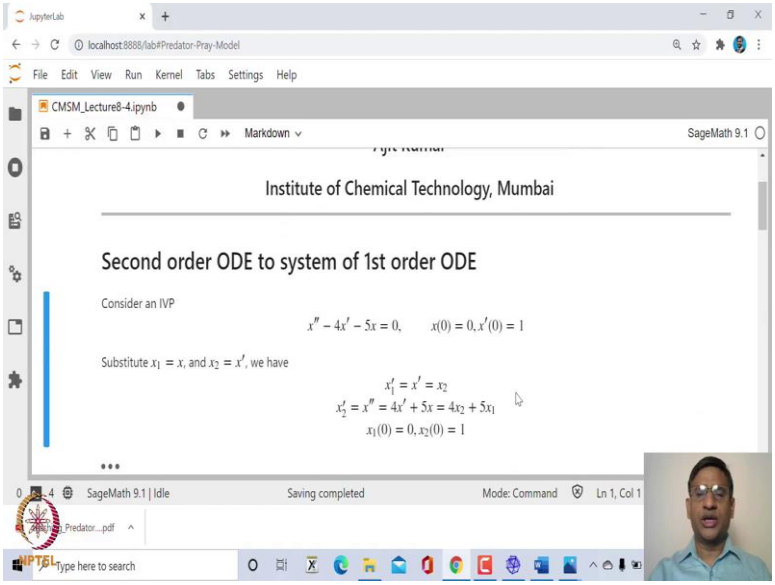
Computational Mathematics with SageMath
Prof. Ajit Kumar
Department of Mathematics
Institute of Chemical Technology, Mumbai

Lecture - 52
Fourth Order Runge-Kutta Method

Welcome to the 52nd lecture on Computational Mathematics with SageMath. In this lecture we will look at two things, one in case you have higher order differential equation with initial conditions, how to convert this into a system of first order differential equations. And the second we will look at algorithm which is known as Fourth Order Runge-Kutta Method for solving initial value problems.

(Refer Slide Time: 00:53)



So, let us start with an example. So, suppose you have a differential equation x double dash minus 4x dash minus 5x equal to 0, that is a homogeneous second order linear differential equation and where x0 is equal to 0, x dash at 0 is 1. We want to convert this into first order linear differential equations. That is, a system of first order linear differential equations.

So, how do we do that? Let us assume x1 to be x and x2 to be x dash. In that case, what we have? The first equation is, if you look at, x1 dash that is x2. So, that is the first order differential equation and second let us look at x2 dash. What is x2 dash? x2 dash is x double dash and which is equal to 4x dash plus 5x from this given equation. Therefore,

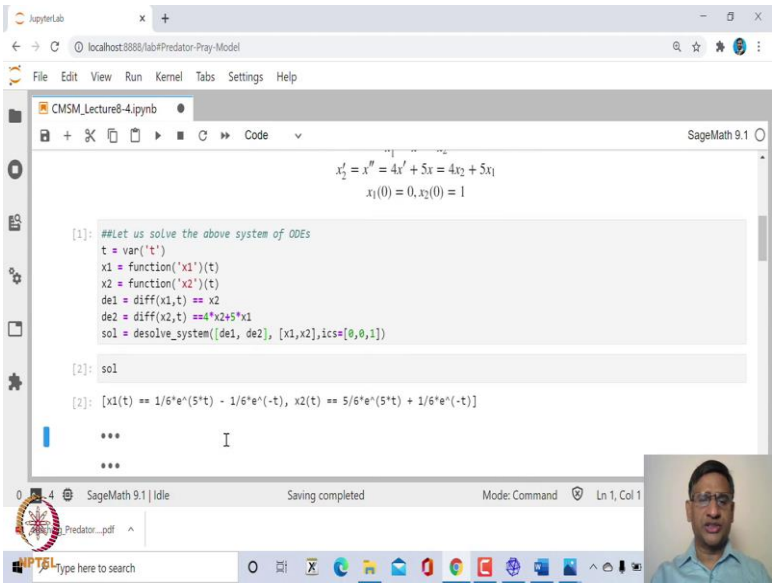
we have 2 equations namely first is x1 dash is equal to x2 and second one is x2 dash is equal to 4 x2 plus x1.

This is a system of first order linear differential equations. Now let us look at the initial conditions. Initial condition x(0) is equal to 0, but x is x1, therefore, x1(0) is equal to 0 and x dash 0 is equal to 1, therefore, x2(0) is equal to 1. So, this is a system of first order linear differential equation which is obtained from the second order linear differential equation with these initial conditions.

If you have third order you will get 3 equations in 3 variables and so on. In case you have nth order linear differential equation then you can convert this into system of n first order linear differential equations.

So, let us look at how we can solve this system? We will see that when we solve this system or we solve this system of first order equations they are the same.

(Refer Slide Time: 03:11)



So, let us look at. Suppose we want to solve this system using desolve_system. We have already seen that we can solve such system linear ordinary differential equations using de_system or we can solve this using method of diagonalizations. We looked at a few examples as an application to eigenvectors and eigenvalues.

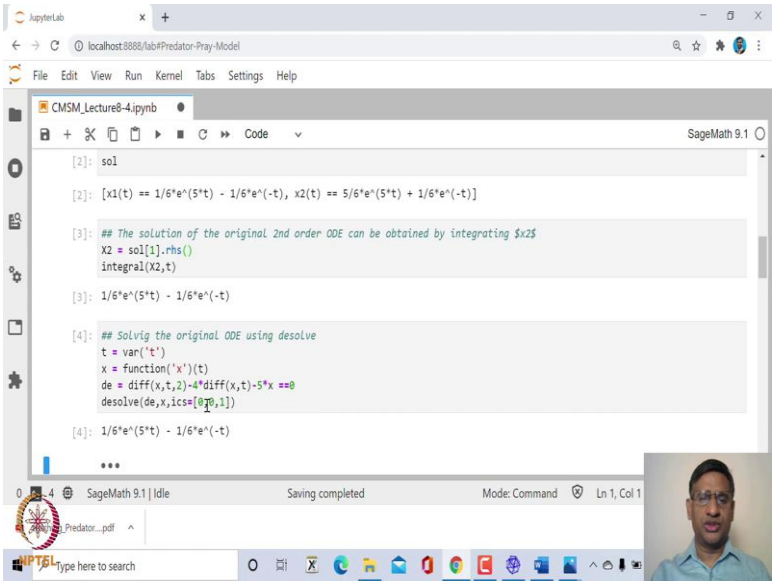
What we are doing, we are taking t to be variable, x1 and x2 as a function of t. Define the first differential equation, that is, x1 dash is equal to x2, second differential equation,

which is x2 dash is equal to 4 x2 plus 5 x1 and then solve this. When we solve this, you can see here the x is nothing, but x1. So, if we try to print what the solution, we have obtained, just a second, it is taking little bit of time, yeah, it has come.

So, x1 t is equal to 1 by 6 e to the power 5 t minus 1 by 6 e to the power minus t. So, this is the solution of this system of first order linear differential equation.

Now in case you try to get solution from this, x2 is equal to this, since x2 is x dash, the integral of x dash will give me x. Therefore, if we integrate x2, we will get the solution.

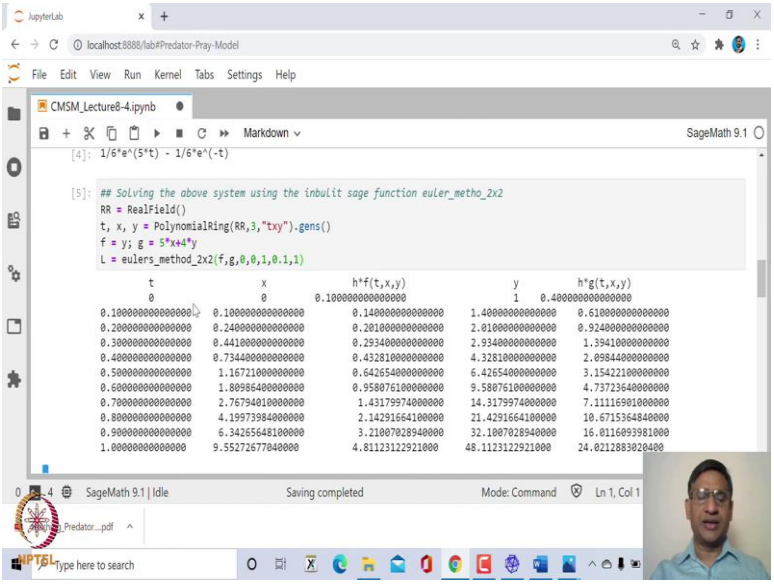
(Refer Slide Time: 04:48)



So, this is again same as x1(t), which is x(t). Now the same differential equation, suppose we try to solve using inbuilt function desolve now. This time we are solving the second order initial value problem. Again declare x, t as a variable, and x as a function of t, declare the differential equation, that is, second derivative of x minus 4 times first derivative of x minus 5 x is equal to 0.

And when we solve this with initial condition x(0)=0, x dash 0 is equal to 1, this is what we get. This is same as what we obtained using solving system of first order linear differential equations.

(Refer Slide Time: 05:39)



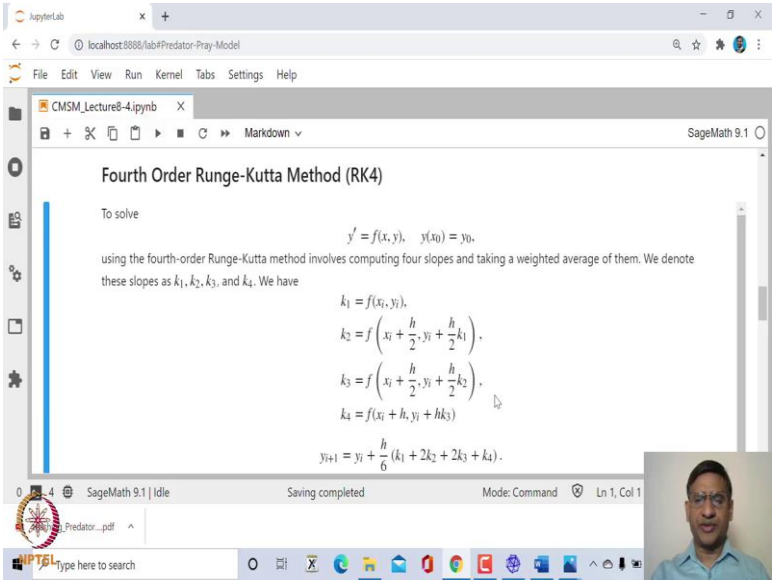
You can also use inbuilt function eulers_method_2x2. Sage has inbuilt method or inbuilt function to solve a system of 2 linear equations in 2 variables, initial value problems using Euler's method.

This is what is called eulers_method_2x2. How do we do that? First you declare, this field, a RealField and then declare x(t), x, y as variables. So these are the polynomial over PolynomialRings and the generators will give you t, x, y and then define the function f and then let us apply this method eulers_method_2x2. When we apply this, this is what we get.

So, in this case it will report the value of t. Here we are taking t initial value as 0, final value as 1 and the step length is 0.1. So, you can see here 0.1, 0.2 and at this value, it is giving value of x and value of y. It is also reporting h times f (t, x, y) and h times g(t, x, y). So, this is again very similar to Euler's method. In this case of course, you can ask how these things have been obtained? This is quite simple, actually one has to apply this Euler's method to each of this coordinate equation, that is, x dash t and y dash t and then you just put this in a matrix form.

You can also try to plot the points which we have obtained. There is a inbuilt function eulers_system_2x2_plot, that will plot also this set of points. I will leave that as an exercise for you to explore.

(Refer Slide Time: 07:38)



Now, let us look at this fourth order Runge-Kutta method or popularly it is known as RK4 method. This is an algorithm which is very efficient and quite popular. If you look at, last time we looked at the error, we compared error in Euler's method and Euler's modified Euler's method and what we observed was that the error is somewhat linear. So, it is not very efficient method. Whereas in RK4 method, error is much smaller and it decreases quite fast as you decrease the step size.

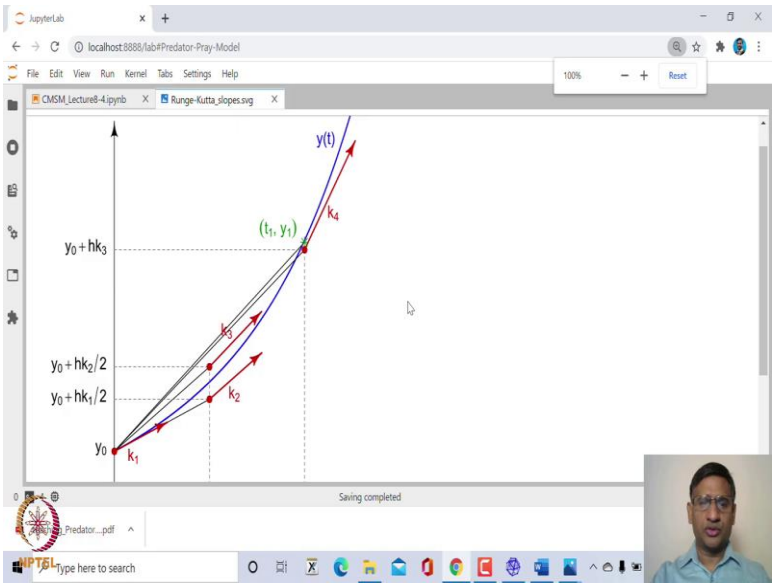
Let us look at what is this method. In this method what we want to do? We want to solve this dy dx is equal to f(x, y) and initial condition y at 0 is y0, using the fourth- order Runge-Kutta method.

This actually involves computing slopes at four different points. So, four slopes and taking weighted average of these slopes. We saw in case of Euler's improved Euler's method. We will took average slope at xi, yi and xi+1, yi+1. In this case, we take 4 points and we take the slopes and find the weighted average.

These 4 slopes, let us call this as k1, k2, k3, k4 and then what is this k1? k1 is the slope of y at xi, yi, k2 is slope at xi plus h by 2, half of this step length and yi plus h by 2 times k1, k1 is this. Similarly, k3 is f of xi plus h by 2 again at half of the step length comma yi plus h by 2 times k2 and k4 is slope at f xi plus 1, that is xi plus h and yi plus h, which will be yi plus h times k3.

And then we take yi plus 1, define the next iterate, the next approximation as yi plus h by 6 times k1 plus 2 k2 plus 2 k3 plus k4. One can also obtain this using Simpson's rule, but this is just a weighted average of this k1, k2, k3, k4.

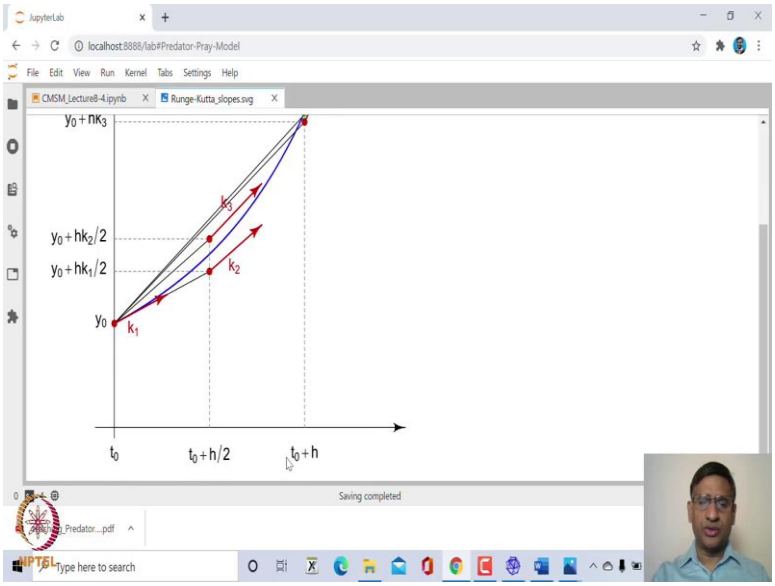
(Refer Slide Time: 10:23)



Now if you look at geometrically, let me just show you, what is geometric meaning of this. So, here let me make it small. So that it fits in the window.

So, what is it here? This blue curve is graph of y(t), and this is the initial point t0, this is t0+h by 2, t0+h.

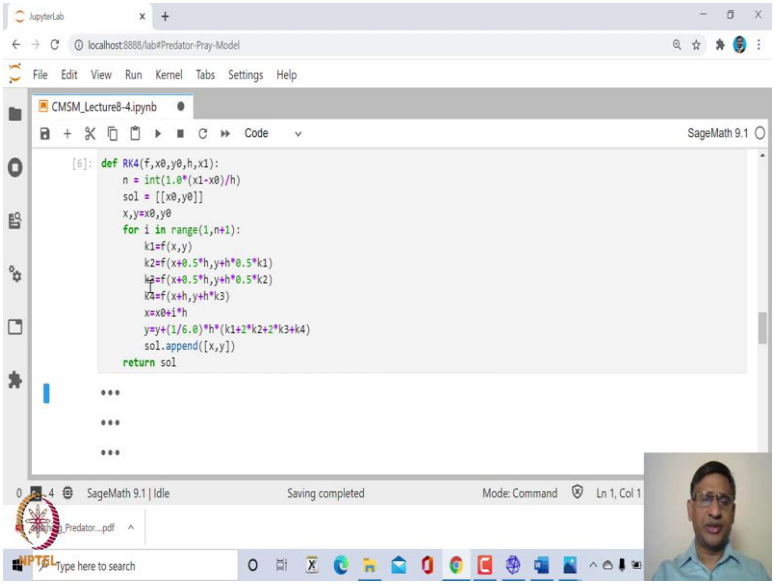
(Refer Slide Time: 10:38)



What is at t0? You have y0, and k1 is the slope at (x0, y0), and this is, t0+ h/2, that is going half of the step length, and then this point here is y0 plus h times k1 by 2, and this is the slope k2, this is how you define k2. Similarly, you take y0+ h k2/2 and at this point you slope define the slope as k3. Then you have this point which is y0+ hk3, at t0+h, you define the slope k4, and then take weighted average of these k1, k2, k3, k4. That is how geometrically it is defined. This particular graph is taken from Wikipedia. So, I must admit that. One can draw this very easily in SageMath as well.

Let me close this and get back to the worksheet. This is how we defined this iteration scheme of RK4 method.

(Refer Slide Time: 11:58)



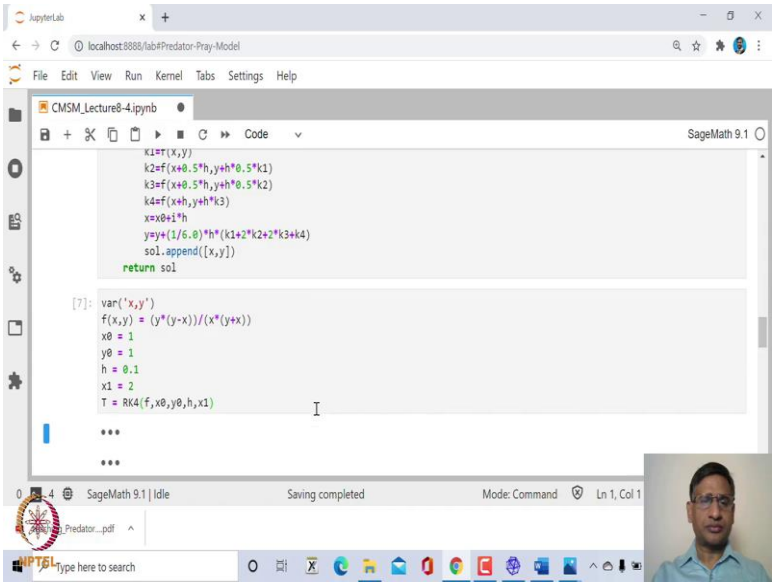
Now let us implement this in Python. It is again very similar to what we have done in case of Euler's method and improved Euler's method. Here we are giving the name RK4, inputs are f, x0, y0 and then the step size and then the end point at which you want to evaluate, that is x1. Fnd find out what is the n, number of interval in which we are dividing, x0 to x1. You start with initially as x0, y0, x, y, you initialize as x0, y0. In stead of calling xi, yi, we will simply look at x and y. So, k1 is f(x, y), run the loop i going from 1 to n plus 1 because the last one is xn. So, this range will give you 1 to n.

And then define k2 which is f at x at in plus half of h plus y plus h into half h n by 2 into Then define k2, which is f at x plus half of h, y plus h into half of h into k1, similarly k3

and k4 and then define y equals to y plus 1 by 6 into h times k1 plus 2 k2 plus 2 k3 plus k4, and then increment x by x naught plus i h, that will will become the next node point.

And then you append to x, y which you have obtained in the solution list and then return solution. So this is a simple Python program.

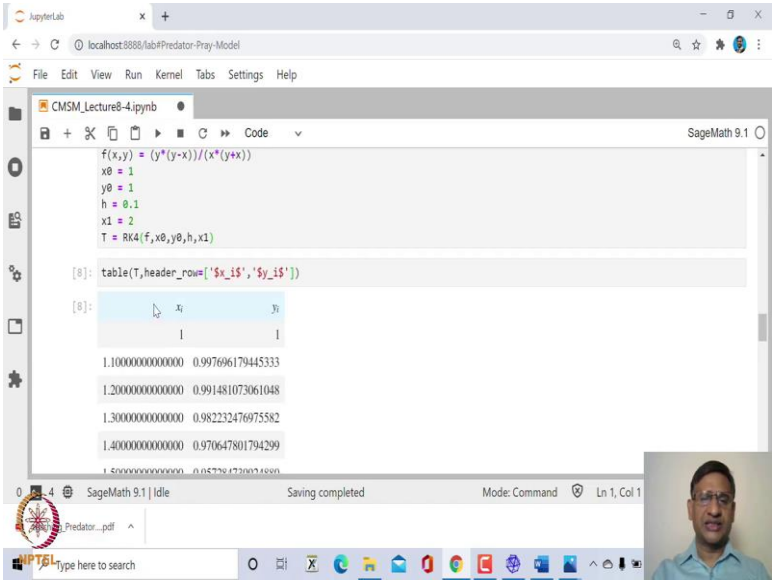
(Refer Slide Time: 13:44)



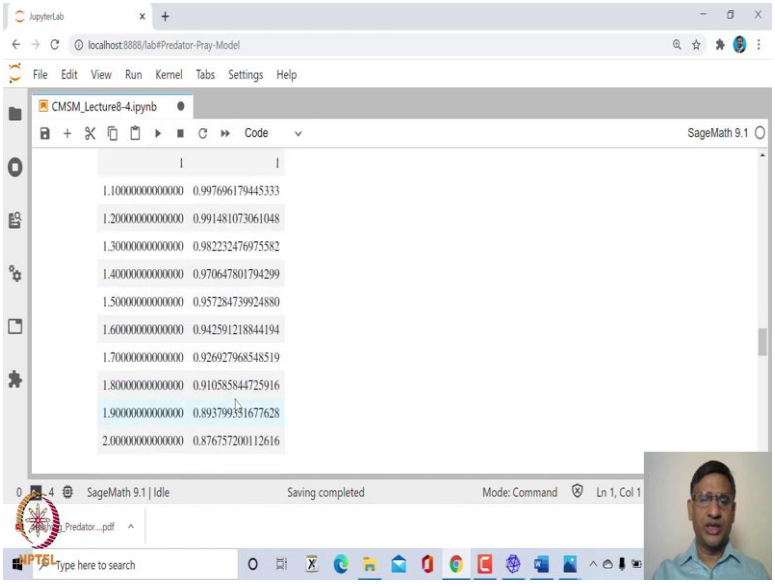
And now let us use this to solve a differential equation dy by dx is equal to y into y minus x divided by x into x plus y. This we have already solved in last lecture.

Let us look at the same problem. We are starting with x0=1 and y0=1. So, y at 0 is 1 and h, the step size is 0.1 and x1 is 2. So, between 1 and 2 you want to find the value of this yi, approximate value of yi using RK4 method. So, let us call this method and store this set of points in capital T.

(Refer Slide Time: 14:32)

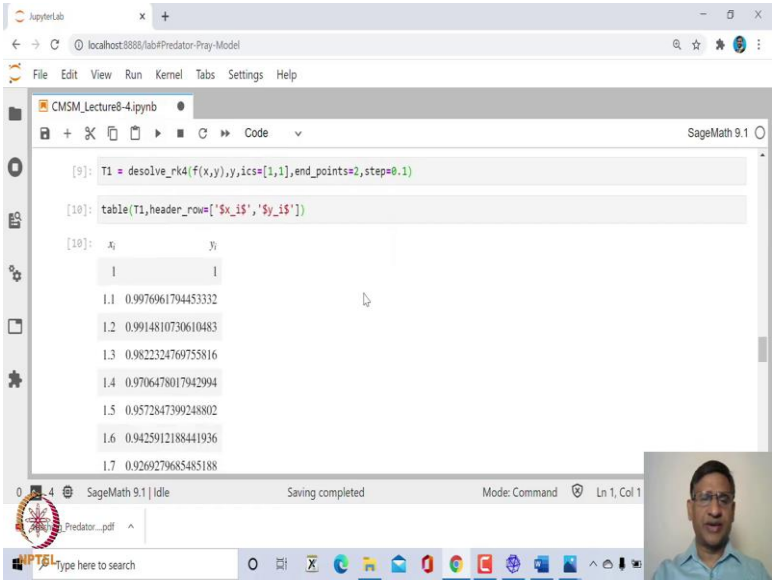


(Refer Slide Time: 14:41)



We can tabulate this xi and yi, these are the value approximate value of yi, the last value is 0.8767.

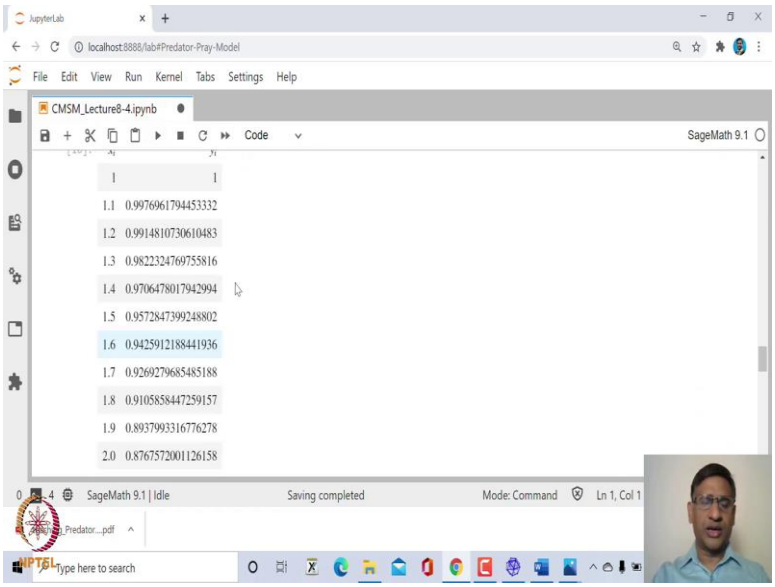
(Refer Slide Time: 14:48)



Let us use the inbuilt function. We saw that sage has inbuilt function, called desolve_RK4.

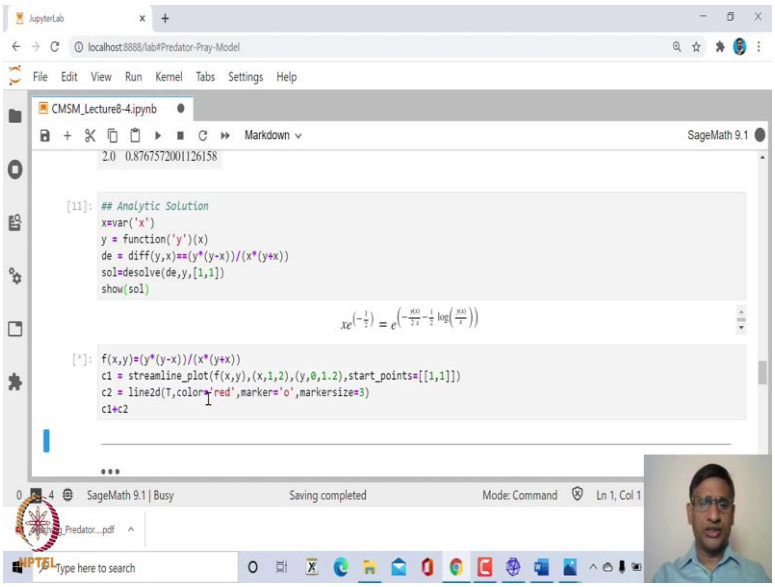
Let us call this method, so f(x, y), you want to solve with respect to y, initial conditions are 1, 1 that is, y at 1 is 1, end points is 2, step size is 0.1 and when you again tabulate these values, we should obtain, just a second it takes time. Now it has done.

(Refer Slide Time: 15:21)



So, when we tabulate this value and if you try to compare this with what we have obtained using our own user defined function it is pretty similar. This is how you can make use of inbuilt function to solve using RK4 method.

(Refer Slide Time: 15:36)

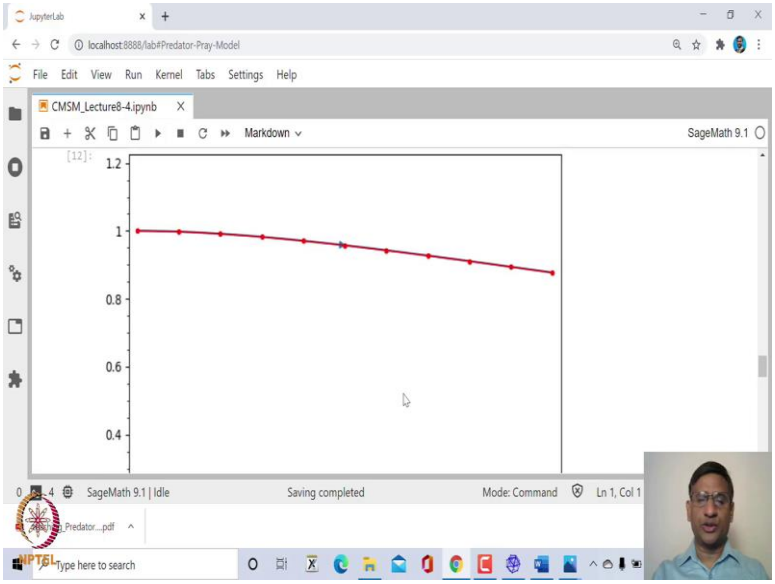


In case you have higher order differential equation, you need to convert it to the system of first order linear differential equation and then you can solve, ok. Let us try to solve this analytically.

When we solve this analytically, then the solution which we obtained is given by this implicitly defined function which is x into e to the power minus half is equal to e to the power y x by 2x minus half into log of y x by x. So, if you try to plot graph of the solution curve which is the exact solution using streamline plot.

We have again seen this how we can plot the solution curve or integral curve passing through a particular point. In this case, the point we are taking 1 comma 1, which is y at 1 is equal to 1 and then plot this T, the set of points which you obtain using our own user defined function. You could also use the T, one which is solution obtained using inbuilt RK4 method.

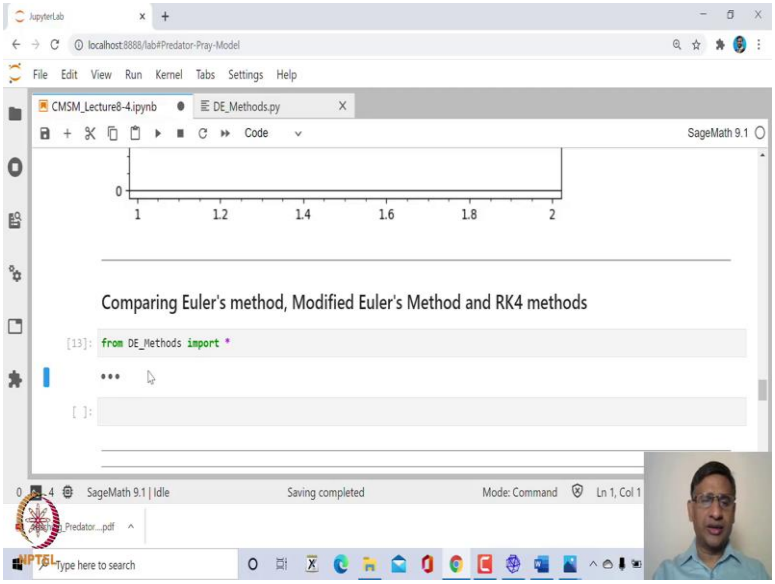
(Refer Slide Time: 16:49)



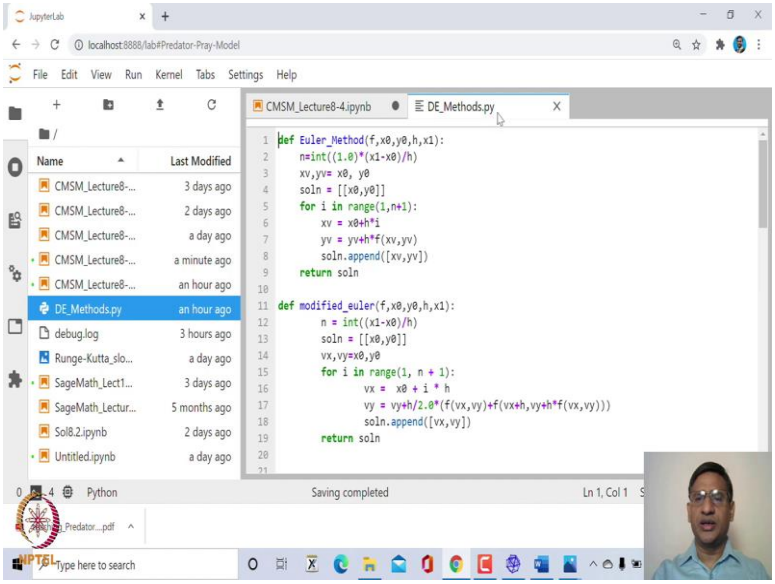
When we plot this plot this, you can see here, the blue one is exact solution and the red one is the approximate solution. And approximate solution and blue solution curve they are very close to each other. This in this case they are almost matching.

If you decrease the step size further then you will get much better approximation. Now, let us compare all these 3 methods which we have looked, at namely Euler's method and modified Euler's method and RK4 method. Let us try to plot the graph of a solution curve using these 3 methods with different step size and then see how they compare.

(Refer Slide Time: 17:44)

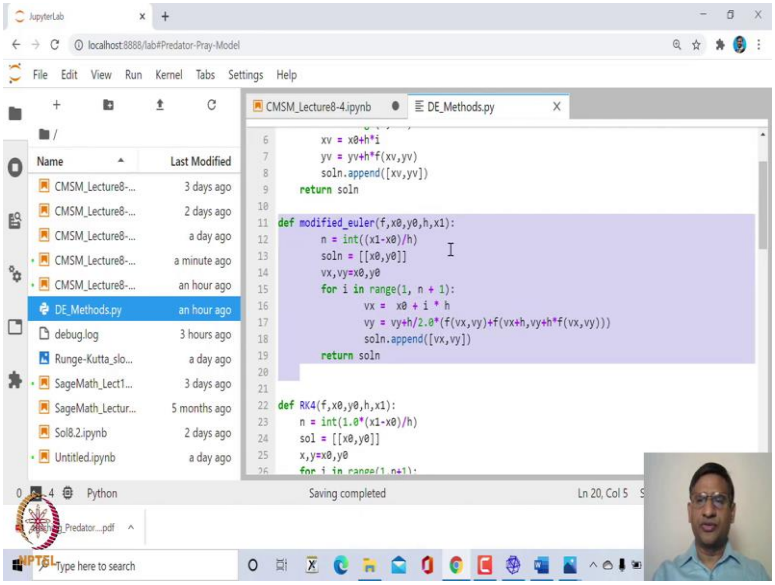


(Refer Slide Time: 17:59)



First of all, what we will do, we will store these user defined functions, we have defined namely RK4 method and earlier 2 method in a function or in a file which we have called as DE_Method.py. These functions I have stored it here. We have we saw that how to create a Python module.

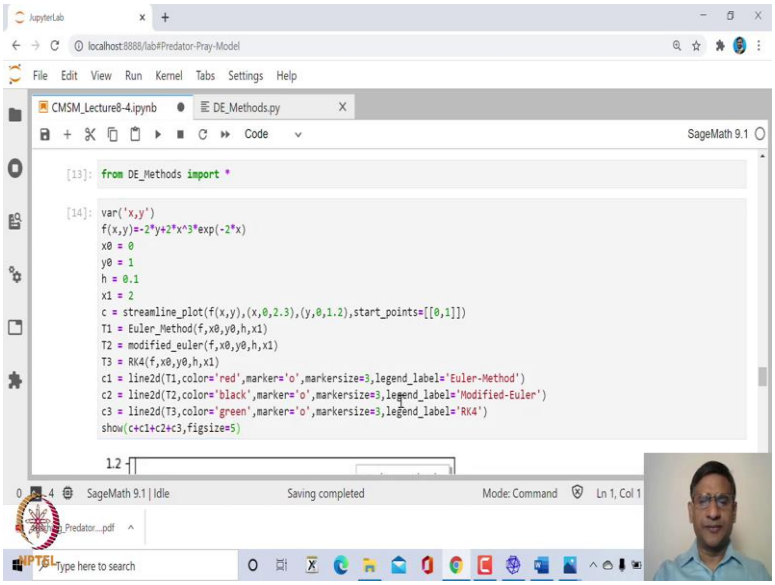
(Refer Slide Time: 18:12)



This is what is done, these 3 functions are stored in this. Let me let me close this file. All I have done is, I have copied these functions which we have created in DE_ Method.py,

dot py is extension. And then let us call this. First let us import DE_Methods, all the functions inside this, and then what we what do we do?

(Refer Slide Time: 18:45)

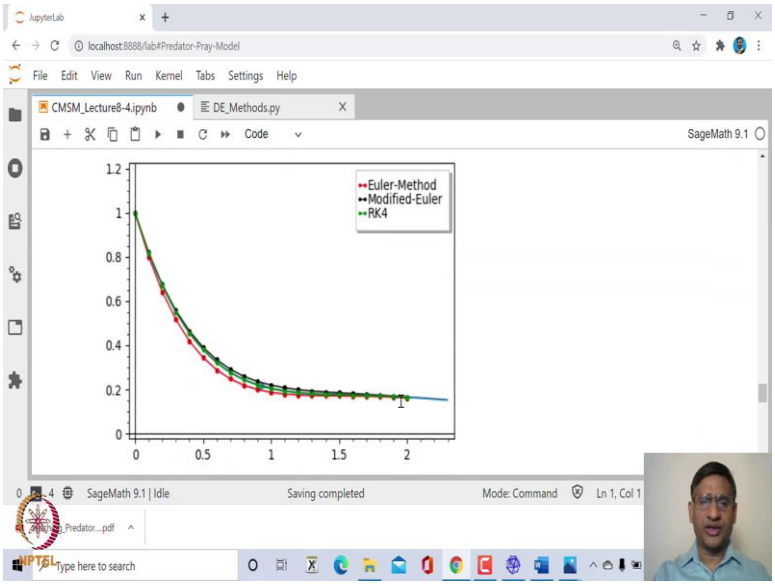


Let us look at a differential equation dy by dx is equal to f(x, y) which is minus 2 y plus 2 x cube into e to the power minus 2 x. We solved this problem in last lecture. We start with initial condition x0 is 0 and y0 is 1 that is y at 0 is 1 and h the step length is 0.05, x1 first let us start with step length 0.1 and the end point is 2.

Then let us plot the first the streamline plot, that is the exact solution obtained or the solution curve passing through 0 comma 1 and then T1 is obtained using Euler's Method, T2 is set of approximate solutions obtained using a modified Euler's Method and T3 is using RK4 method. Then let us plot these set of points which we have obtained for Euler's Method in red color, Modified Euler's in black color and RK4 method in green color.

Let us ask it to show. Let me reduce the figure size, let me say figsize is equal to 5 and plot this.

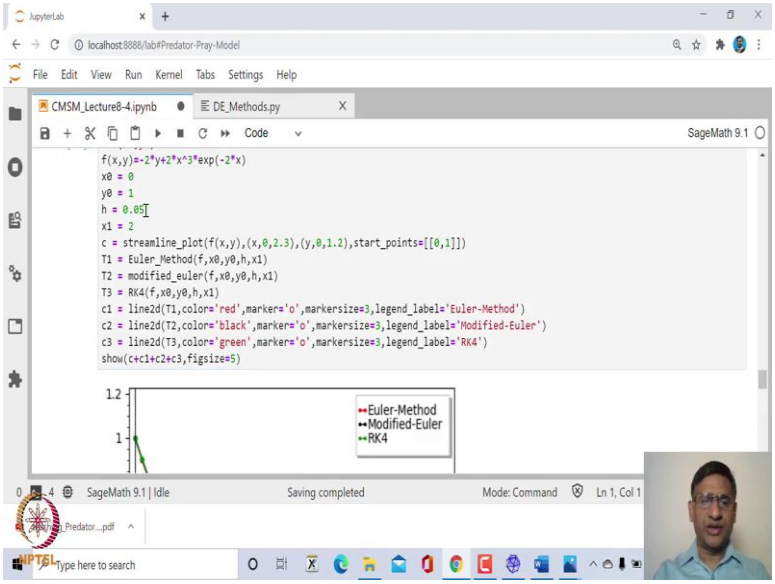
(Refer Slide Time: 20:09)



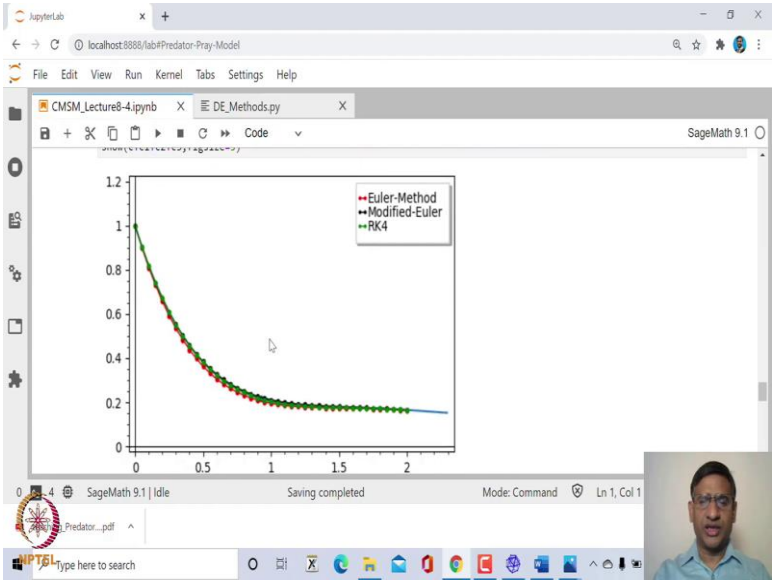
When you plot this, as you can see here, this blue one is the exact solution and this the green one is using R K4 method, which is much closer to other 2 the red one is Euler's method and the black one is Modified Euler's Method.

So, again you can see here Modified Euler's method is much better than Euler's Method whereas, RK4 method is better than these 2. If you try to reduce the step size for example, let us make it 0.05 the half of the 0.1 and then when you plot you can see here, now it has become much better the approximation, right!.

(Refer Slide Time: 20:43)



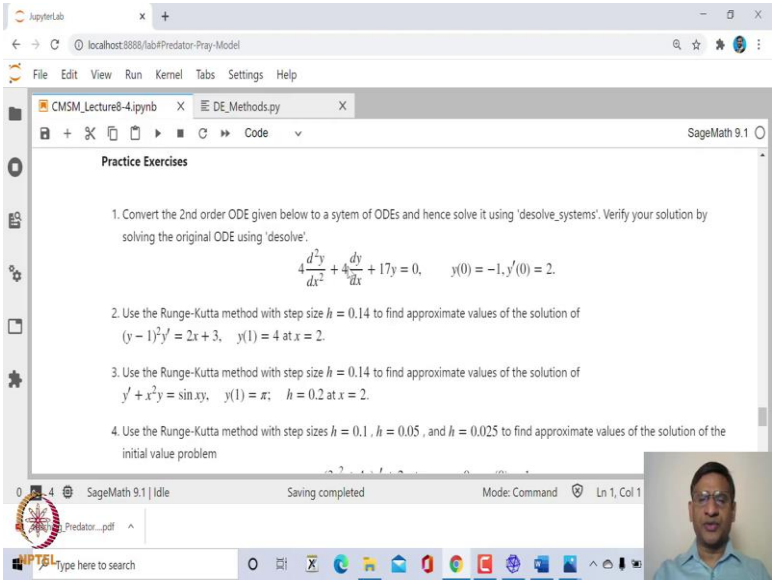
(Refer Slide Time: 20:47)



So, this is how these 3 methods compare with each other. I also expect or rather I suggest you to tabulate the error term with Euler's Method, Modified Euler's method and RK4 method. We already did tabulating errors in a tabular form using Euler's Method and Modified Euler's method.

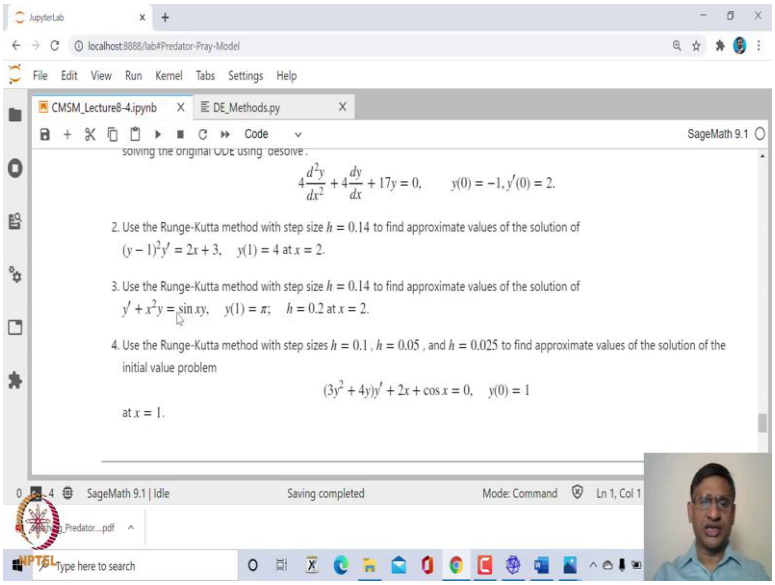
In that, you also add in the next column which is RK4 method, right.

(Refer Slide Time: 21:30)



Let me leave you with some simple exercises. The first exercise is convert this 2nd order linear ordinary differential equations with this initial condition to a system of 1st order ODE and then solved it. Solve it using desolve_systems and solve this system using RK4 method with step size h is equal to 0.14. Here the the initial condition is y at 1 is equal to 4. You need to find or need to approximate this y between 1 and 2 with step length 0.14.

(Refer Slide Time: 21:51)



Next one is again solve this using RK4 method with this step size 0.2, this unction is slightly different and the next one is again solve this using RK4 methods.

So, these are pretty simple exercises, once we have developed this user defined function.

Of course in all these things you should solve this using inbuilt function and also compare it with your user defined function.

So, let me stop here. In the next class we will look at how to solve system of non-linear ODE. We can use inbuilt method using RK4 method and we will also try to create our own function and then we will look at also some applications.

So, thank you very much let me stop here.