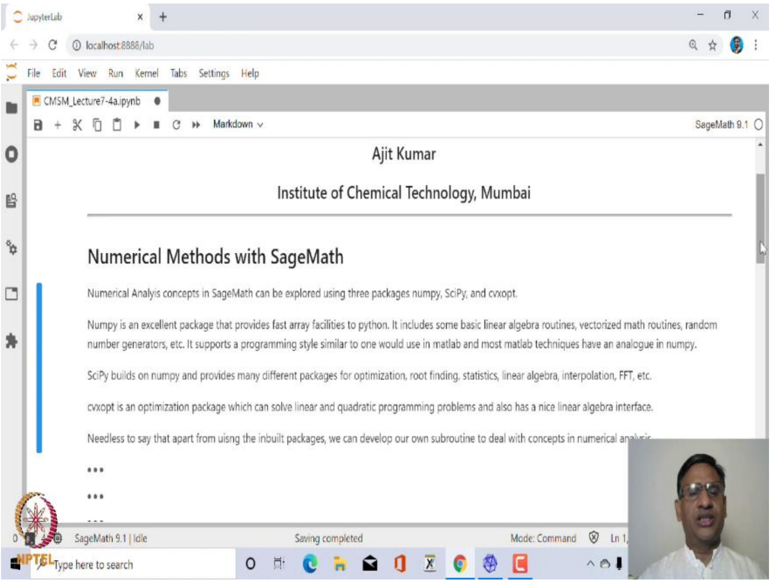
Computational Mathematics with SageMath

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

Lecture - 45
Finding Roots of algebraic and transcendental equations in SageMath

(Refer Slide Time: 00:15)



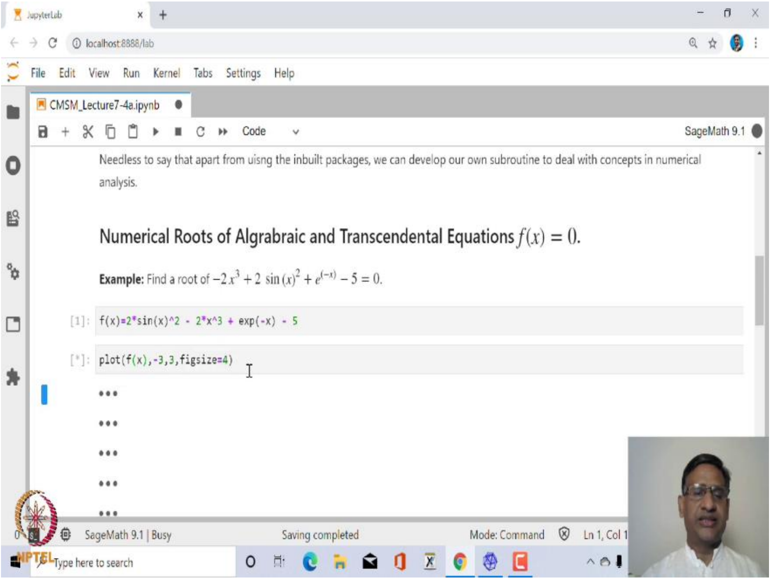
Welcome to the 45th lecture on Computational Mathematics with SageMath. In this lecture, we shall look at finding numerical roots of algebraic and transcendental equations. So, henceforth we shall look at some Numerical Methods in SageMath. So, let us get started. So, numerical analysis concepts in SageMath can be explored using, basically three packages; NumPy, SciPy, and cvxopt.

We have already seen NumPy and SciPy. NumPy is an excellent package in python that has facilities of fast array computation. It also includes linear algebra package, vectorized math routines, and random number generators. It has programming styles very similar to MATLAB.

On the other hand, SciPy builds on NumPy and provides many different packages for optimization, root finding, statistics, linear algebra, fast Fourier transforms etcetera. Cvxopt is an optimization package, which can solve linear and quadratic programming problems, and it also has nice linear algebra interface.

Of course, you can write your own programs in SageMath in order to solve problems in numerical analysis. And we will be doing for some of the methods.

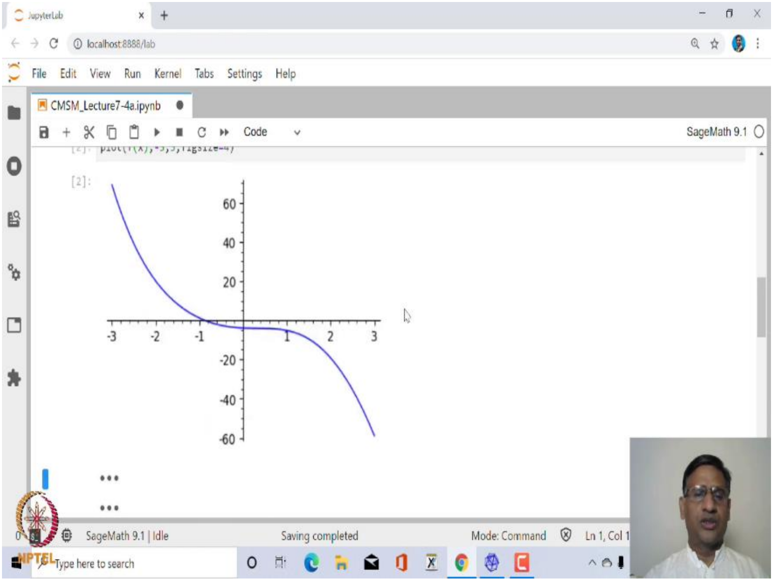
(Refer Slide Time: 02:05)



So, we shall look at finding roots of algebraic and transcendental equations in one variable, namely f(x) equal to 0. So, when I say algebraic equation, basically it means that f(x) is a polynomial. Transcendental equations, on the other hand, will be combination of polynomials, exponential functions, algebraic, trigonometric functions, etcetera.

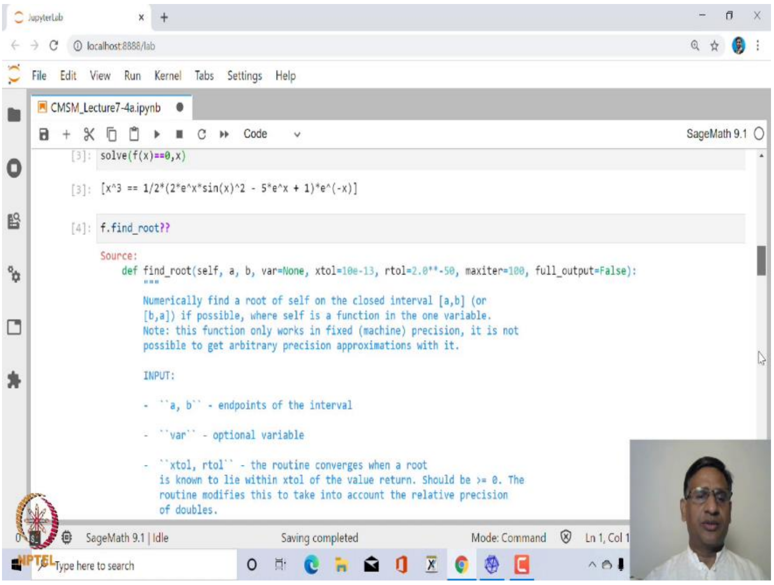
So, let us start with a problem. We want to find a root of minus 2x cube plus 2 sin x square plus e to the power x minus 5 equal to 0. We want to find x for which this left-hand side is equal to 0. So, before we find a root, let us plot graph of this function. So, first, let us declare this function.

(Refer Slide Time: 03:15)



When we, after declaring this function, let us plot its graph between, let us say, minus 3 and 3, and figuresize is equal to 4. So, this is what we see. So, we can see that between minus 3 and 3, it has a root, actually, the root lies between minus 1 and 0. So, we want to locate that root. We have already seen this in, in SageMath.

(Refer Slide Time: 03:34)



So, one can try to use solve function. So, when, when I say solve f(x) equal to 0 for x, it may or may not give a solution. So, for example, in this case, it is not finding a solution. It is unable to find x explicitly. So, this may not work. So, in that case, we have seen that we can use what is called f dot find underscore root. So, let us look at what is find underscore root, it is an inbuilt method in SageMath.

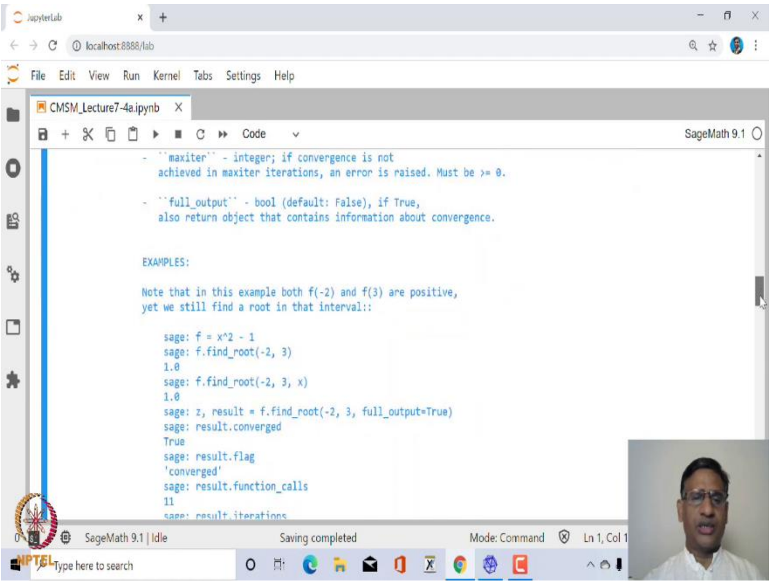
So, let us first take help on this. So, f dot find underscore root double question mark, which will give you detailed document. So, this says that find underscore root of self, that is the function, and a and b are the interval, the ,the end point of the interval.

You need to mention the variable with respect to which you want to find a root, you have to also give the tolerance limit and the number of iterates, maximum number of iterates. By default, it takes 10 iterates.

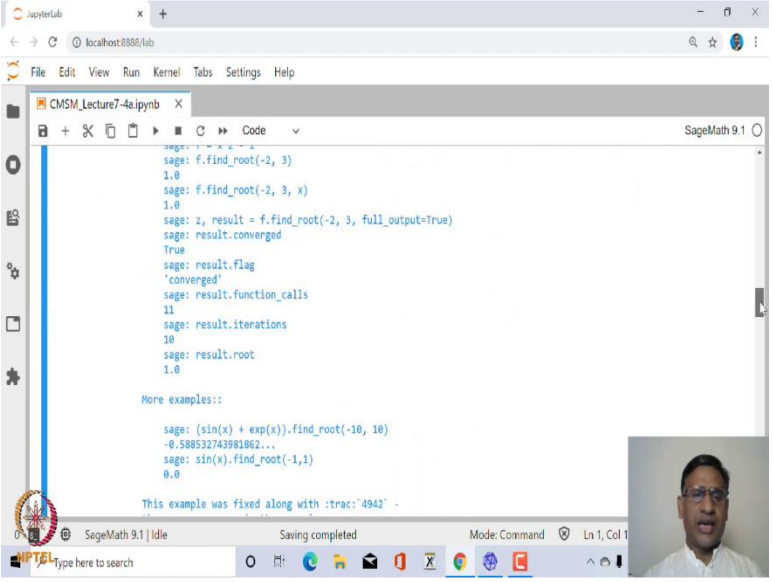
And you can mention the output true or false, that is a Boolean option, and what is it? So, it finds numerically a root of a function f in a, in a closed interval [a,b] or [b,a]. So, it does not matter, you can give [a,b] where a could be less than b, or a could be bigger than b.

And if possible, where the self-function has one variable, is of one variable. So, basically, this works with only one variable. If you have multivariable function and they want to find roots, this does not work, right? And also, here it will find solution only when there is a root between a and b.

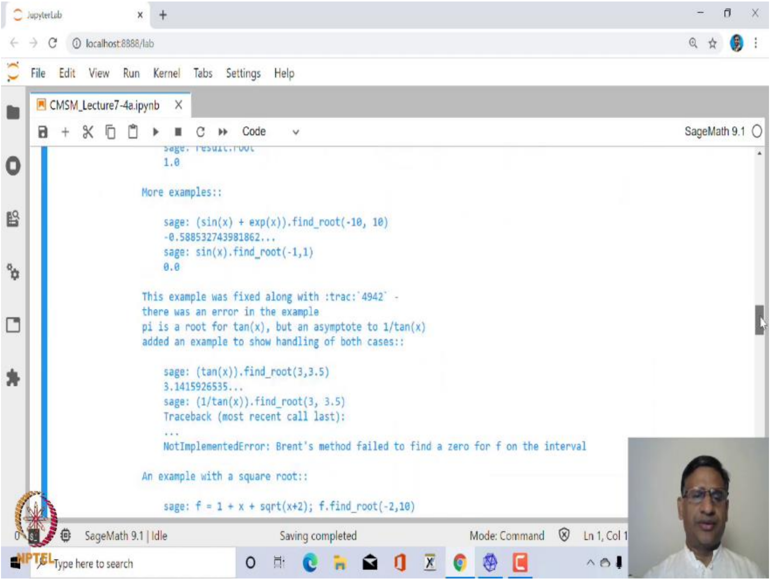
(Refer Slide Time: 05:46)



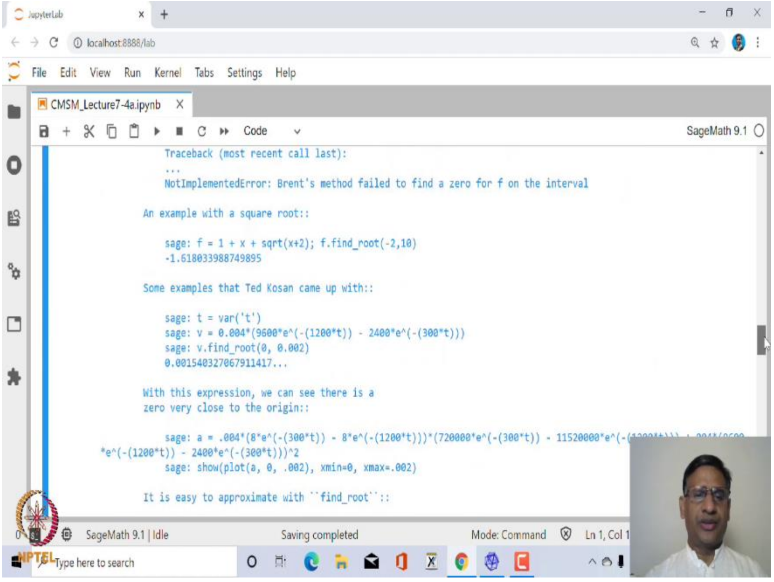
(Refer Slide Time: 05:47)



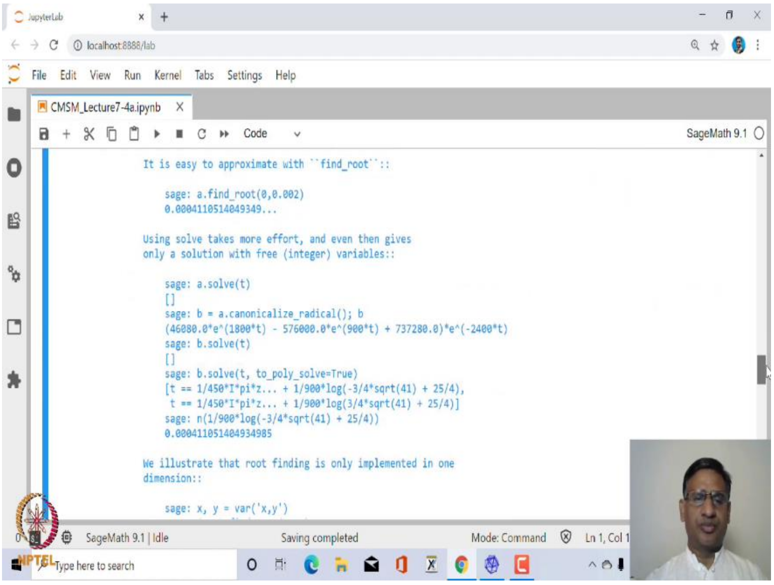
(Refer Slide Time: 05:48)



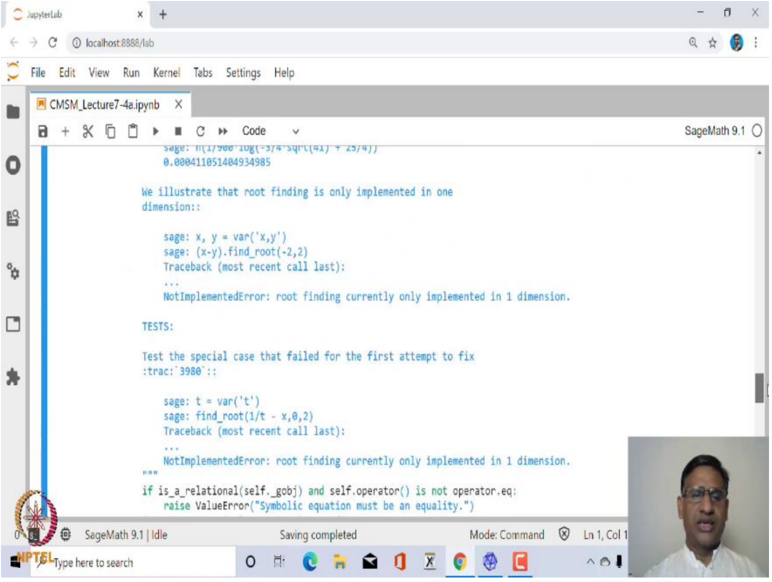
(Refer Slide Time: 05:48)



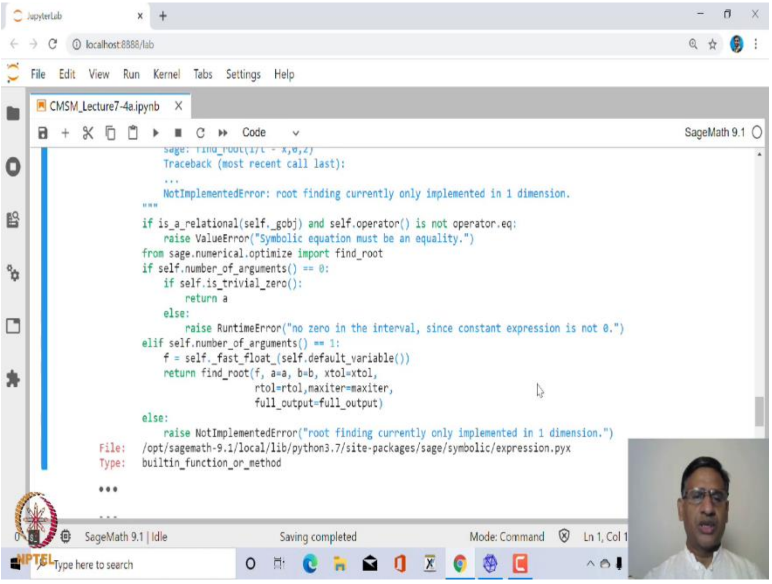
(Refer Slide Time: 05:49)



(Refer Slide Time: 05:50)

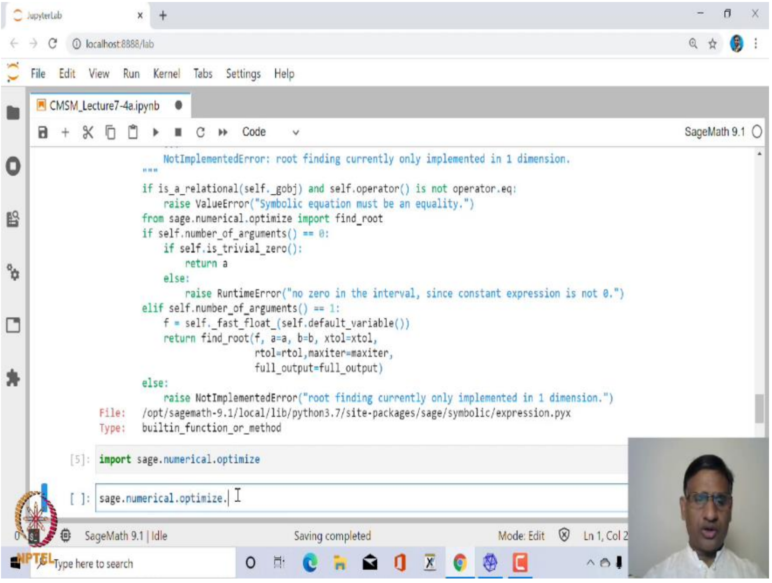


(Refer Slide Time: 05:50)

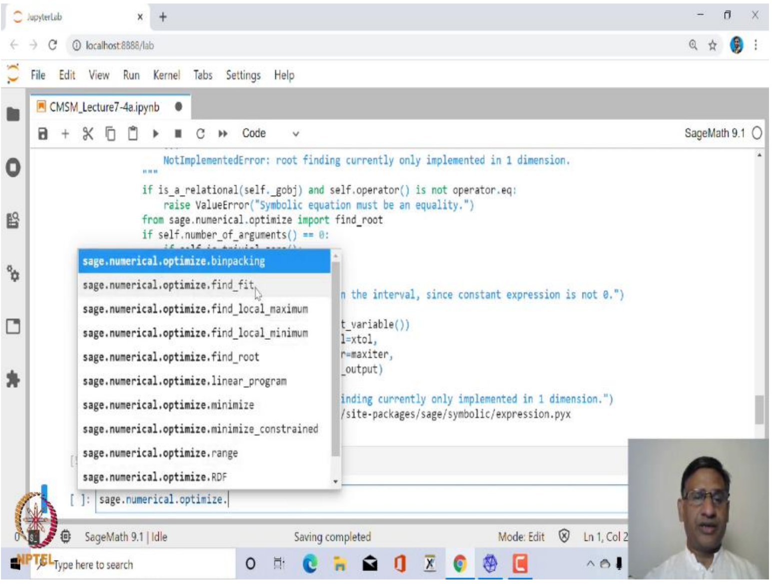


So, you can go through this document and see there are several examples. It also tells you how this function is defined. So, basically, if you look at, this actually uses a function find root, which is defined inside SageMath numerical optimize package. So, if I, if I try to take a help on this package, then let us see what we are going to get.

(Refer Slide Time: 06:17)

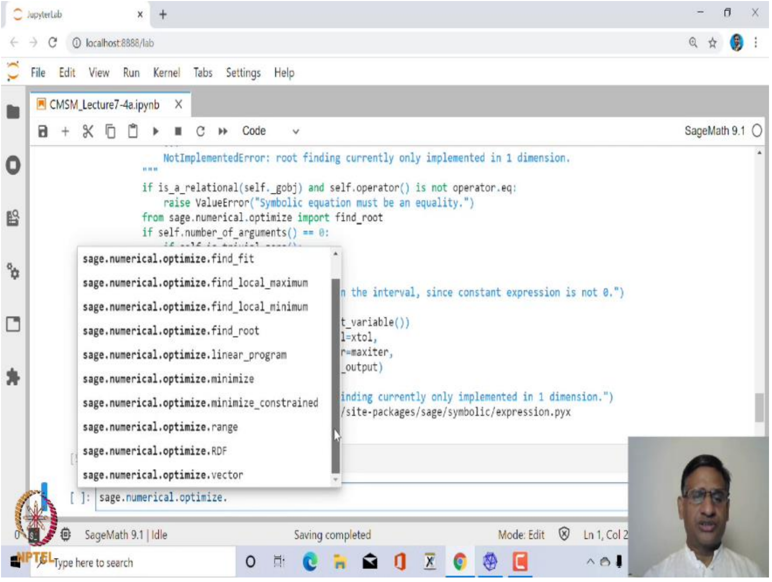


So, let us import, import sage dot numerical dot, right? Once we have imported this, then, let us see what we will get? If I say sage dot numerical dot optimize dot, and then tab. (Refer Slide Time: 06:38)



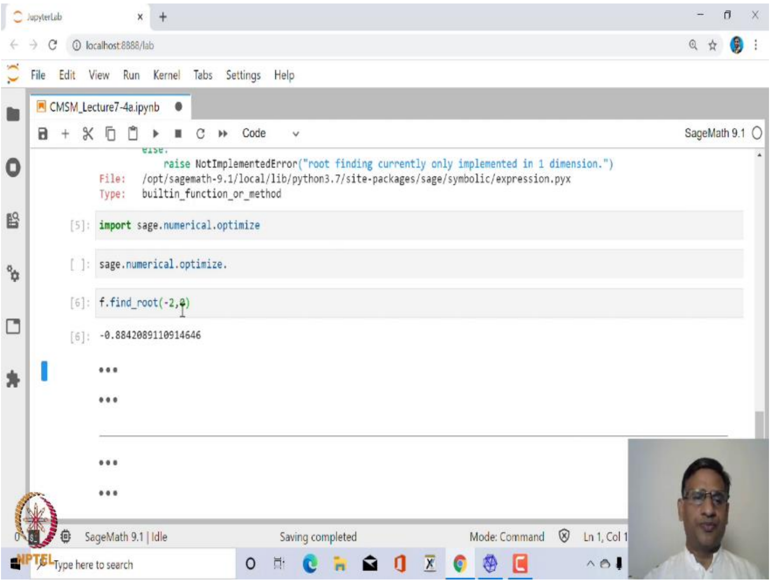
So, you can see here, this, this has many functions. One of them is find fit, which again we have made use of, when we wanted to find, find or fit best fit line or parabola, etcetera, find underscore local maximum, find local minimum, this also we have seen in, in calculus. It can also solve linear programs, that is linear programming problem, and it can find a minimizer, and similarly constrained minimization function also, right?

(Refer Slide Time: 07:13)



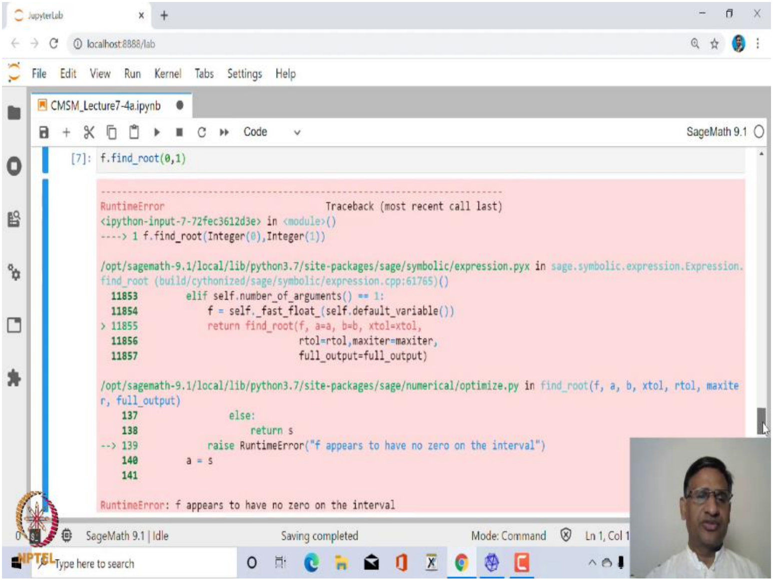
So, this is, inside optimize, numerical optimize, you have several methods, one of them is find root. So, this is what basically Sage uses in order to find root of a one variable equation f(x) equal to 0, or zero of f(x), right?

(Refer Slide Time: 07:41)



So, let us look at how we can find root. We already saw that this function f(x) has a root between minus 1 and 0. So, if we, if we try to give this f dot find underscore root minus 2 comma 0, this tells me that the root lies at minus 0.8842089, and so on. That is the, the point at which f(x) is equal to 0, right?

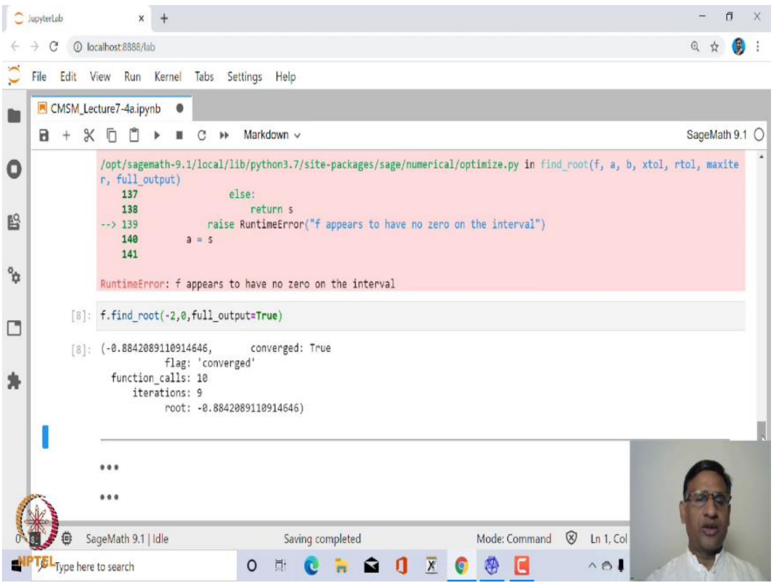
(Refer Slide Time: 08:11)



In case we want to find a root between 0 and 1, if I say, if I give the interval 0 and 1, then we, we already know from the plot that this does not have a root, and therefore, it will give you an error.

This is an error. It says runtime error, and at the end, it tells you that f appears to have no zero on this interval. So, in case you are providing an interval which does not contain a root using find root function, then it will throw an error, right?

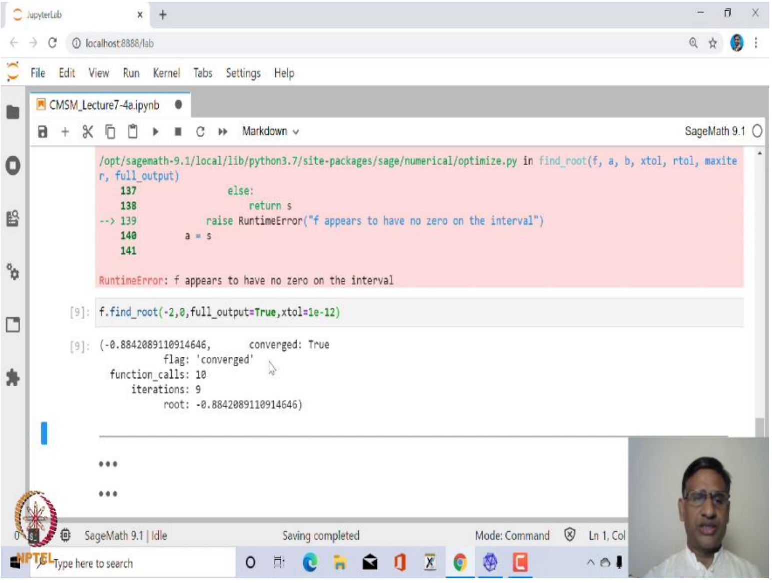
(Refer Slide Time: 08:51)



You can also give an option called full output equal to True, in that case, what does it say? It, it gives you root along with an option converged, and it says that converged equal to True.

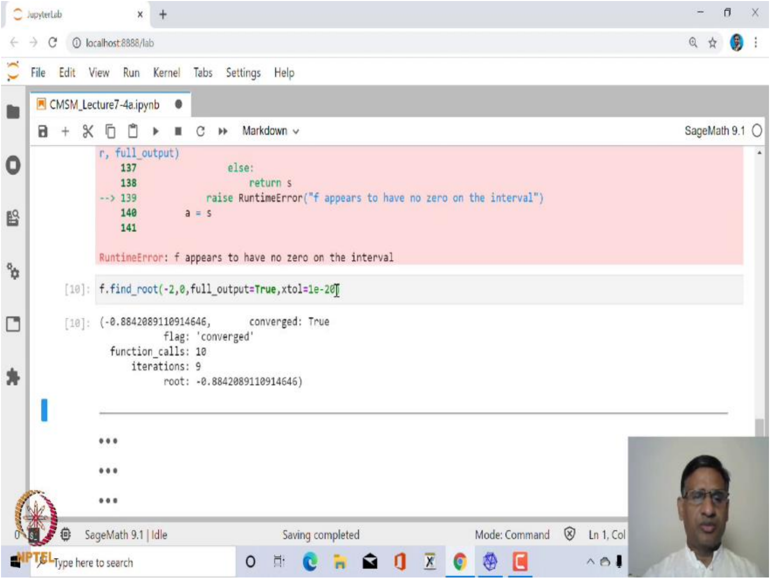
So, numerical method, when you applied any numerical methods, it may or may not converge. There are certain criteria when sequence of iterate converges to a solution. So, in case it has not converged, it will give you the answer false, and then it also tells you that how many times the function has been called. In this case, 10 times function were called, and the number of iterations which is required is, is 9, and the root lies at this. If I increase the tolerance limit?

(Refer Slide Time: 09:45)



So, for example, if I say tolerance, I think tolerance, I think it is called xtol is equal to, let us say, 10 to the power, let us say, 1 e minus 12, then it again it requires only 10 tols, right? So, let us, let us look at how the x tolerance, it is already 10 to the power 13. So, it does not, 10 power 10 is beyond that.

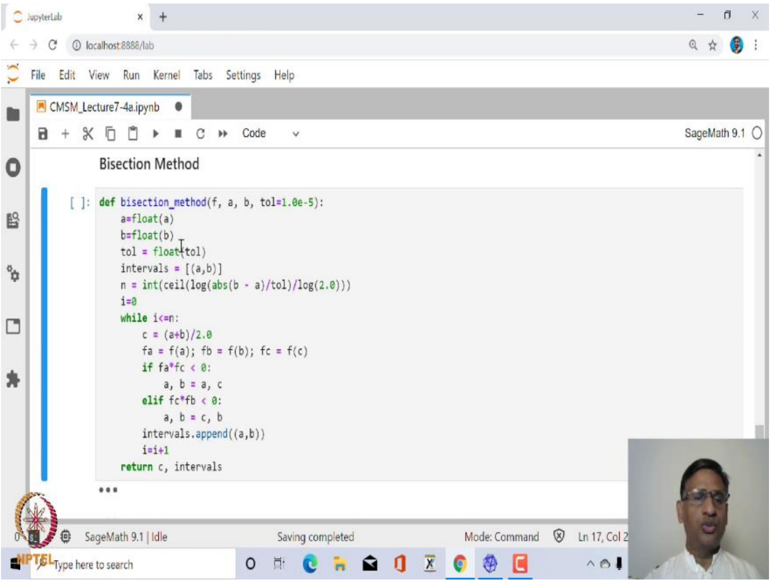
(Refer Slide Time: 10:22)



If I say 10 power 20, it may require more number of iterates. Now it, it requires only the. So, it is, in this case, the, the convergence is quite fast. Whatever method it is using, it converges quite quickly, converges quite quickly, right? So, you, this is what you can use, namely find underscore fit, to a given function in order to find a root of this function.

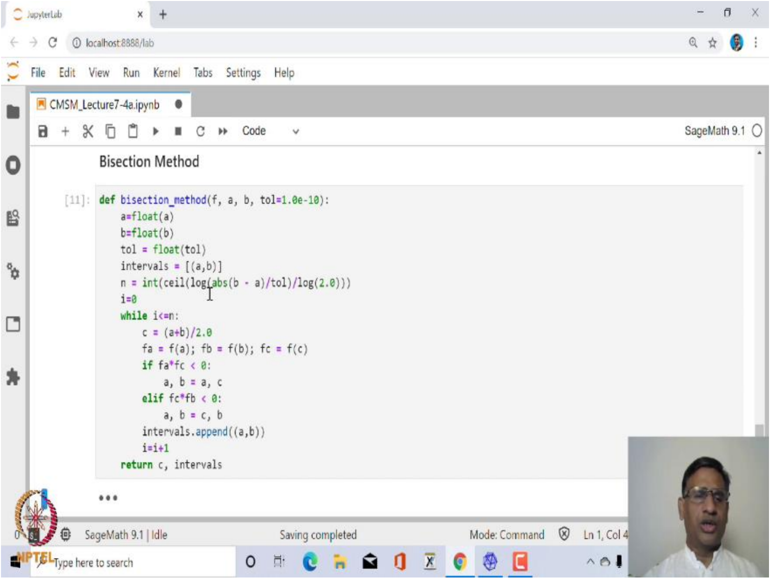
So, Sage provides you very nice way of plotting the graphs. So, first you can plot graph of the function and locate the interval in which it has a root, and then apply find underscore fit, right? That is how, that is how you can find root of an algebraic and transcendental equations. As I said, you can write your own program. You can write python program and run in this Sage worksheet.

(Refer Slide Time: 11:16)



So, for example, let us look at, suppose we want to develop our own program in order to find a root of this function using bisection method. So, bisection method is very simple, and I am sure all of you would have studied this bisection method. So, let me just show you this user-defined function for bisection methods, which requires input f, the interval a and b, and the tolerance is 10 to the power minus 5.

(Refer Slide Time: 11:54)



You can increase this tolerance to 10 to power whatever, 10 to the power, let us say, I will say 10 to power minus 10. And a and b are made float because in case you give this integer, then most often you may end up with rational number as output, which may look somewhat ugly.

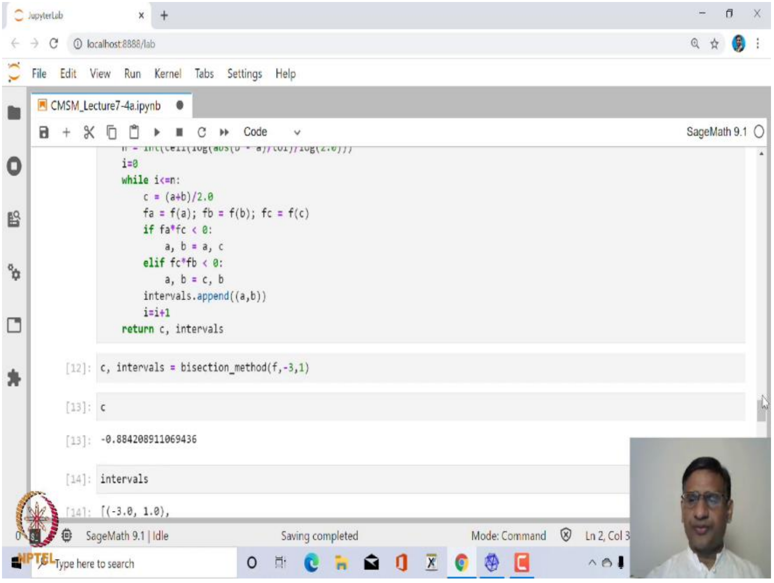
And then you, you also convert this tolerance to float, define initial interval as [a,b]. So, this will give you a tuple of intervals, and then the number of iterates which is required in order to find a root in an interval of length, let us say, a tolerance limit, in this case 10 to power minus 10, that is given by n is equal to integer part of, integer part of log of absolute value of b minus a divided by the tolerance limit divided by log of 2, right?

So, and then start with iteration 0, and then find the value of a plus b by 2, that is mid-point. Check value of f at a, f at b, and f at c. In case f at a and f at c product is less than 0, that means, root lies between a and c. So, you redefine a, b as a, c. And otherwise you redefine this as c, a, c, b.

Of course, you could also check if a, if c is a root, that is, if f(c) is equal to 0, or value of f(c) is less than the tolerance limit, then you can stop here itself. You can use break, or otherwise you keep on appending this interval a, b in the interval a, b, and then increment that the iteration by 1. And then at the end, you return the midpoint and also the set of the intervals, ok?

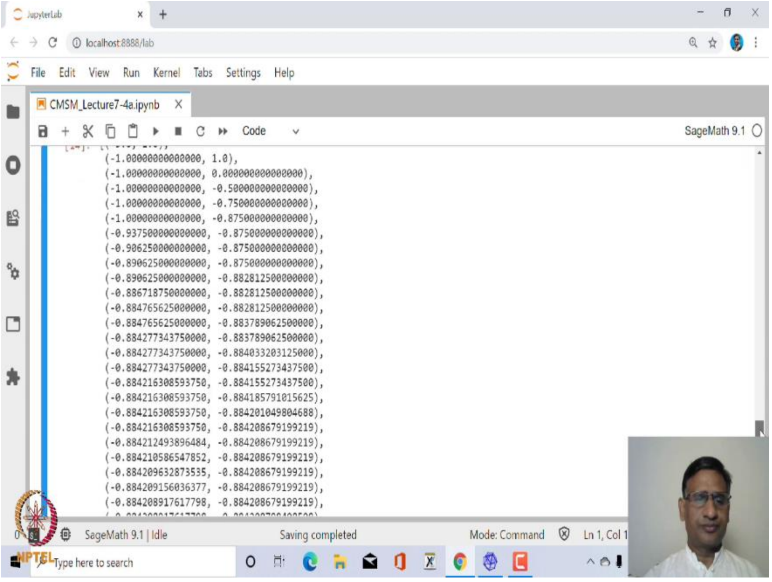
So, this is you can, of course, modify this, include more, more options here, for example, you can check whether a is less than b or not, you can also check whether a, [a,b] contains a root, all these things can be, can be included here, right?

(Refer Slide Time: 14:05)



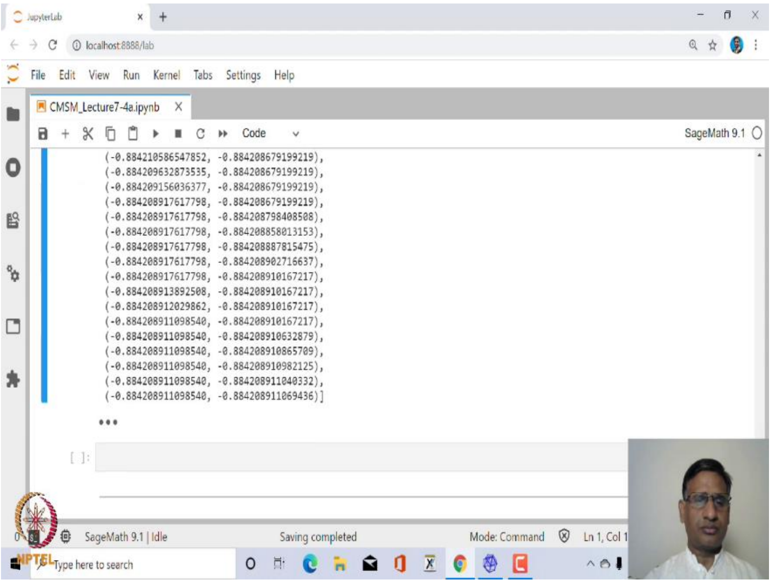
Now let us call this function bisection method for the same function f(x) which we chose earlier.So, this is, now let us see what is the value of c, c is this, and if I, if I look at what is the interval, interval, inter, interval, that intervals is what I have given.

(Refer Slide Time: 14:38)

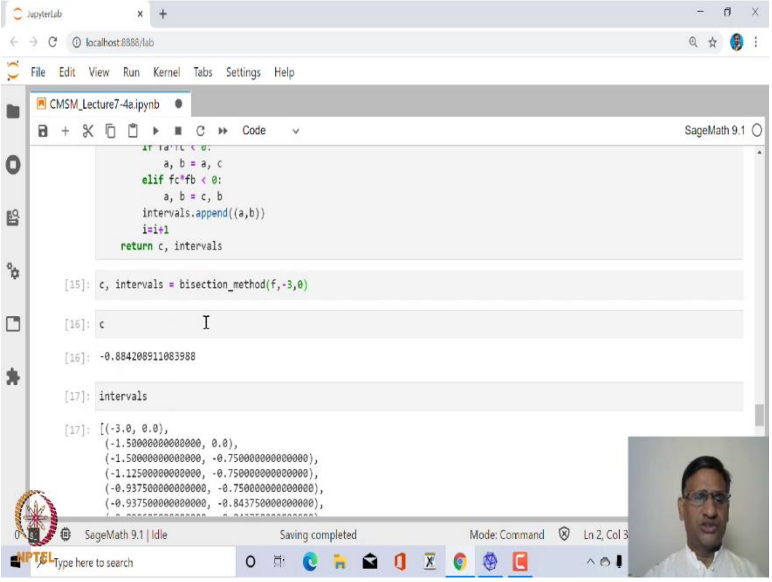


Then this is, these are the interval. This is the first interval, this this becomes, this, this, this, this, this, and so on. So, that is how the, it is approaching, right?

(Refer Slide Time: 14:46)

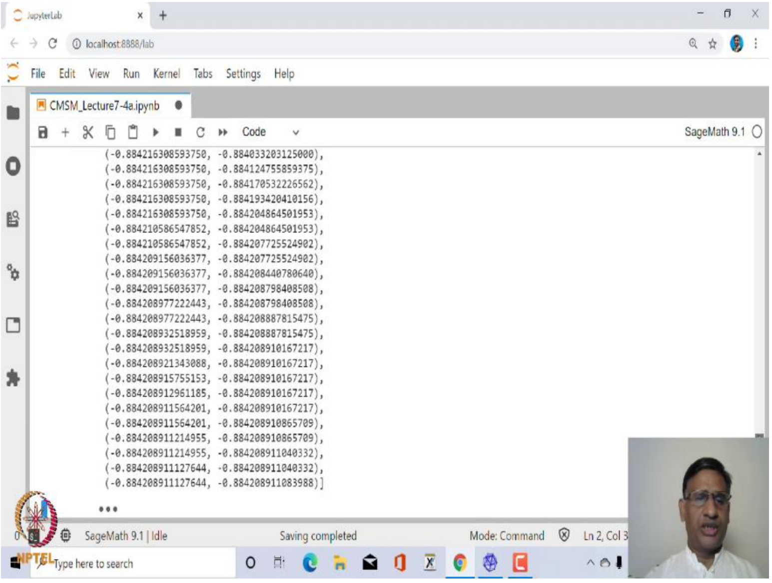


(Refer Slide Time: 14:56)



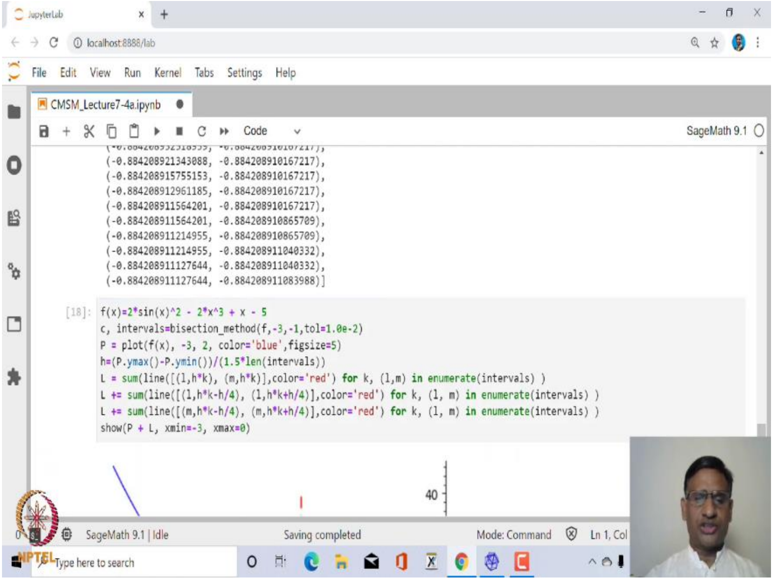
So, in this case, let me, let me, instead of minus 3 to 3, let me give you minus 3 to yeah, ok, minus 3 to 0, and this is what you get.

(Refer Slide Time: 15:07)



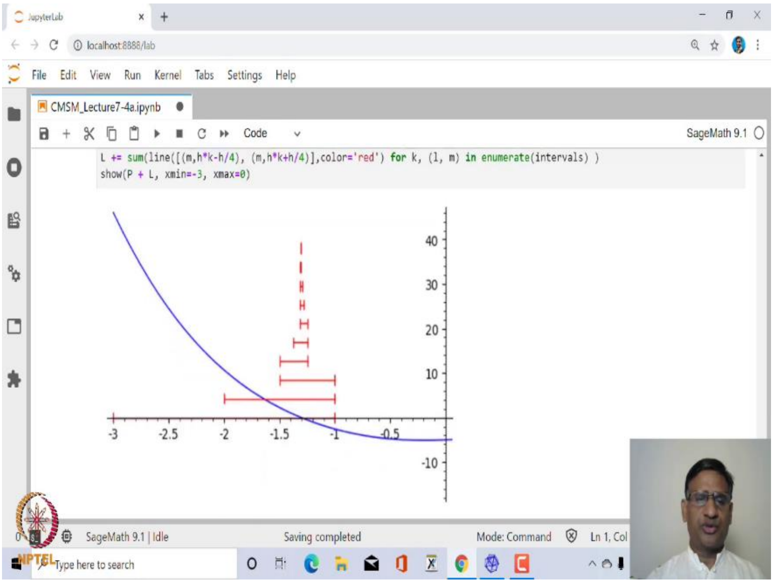
So, you can see here, there are so many iterations which is required in this case.

(Refer Slide Time: 15:10)



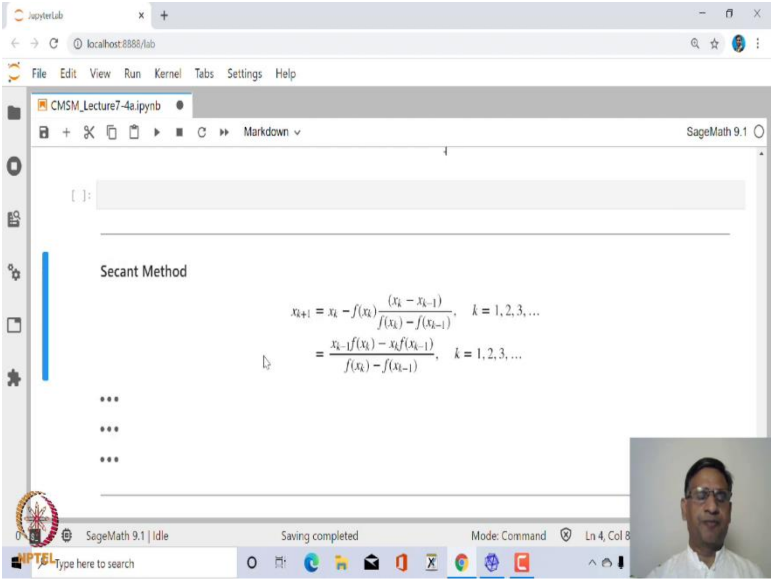
And one can even try to plot all these intervals bisecting intervals containing root, just to visualize how this iteration proceeds.

(Refer Slide Time: 15:23)



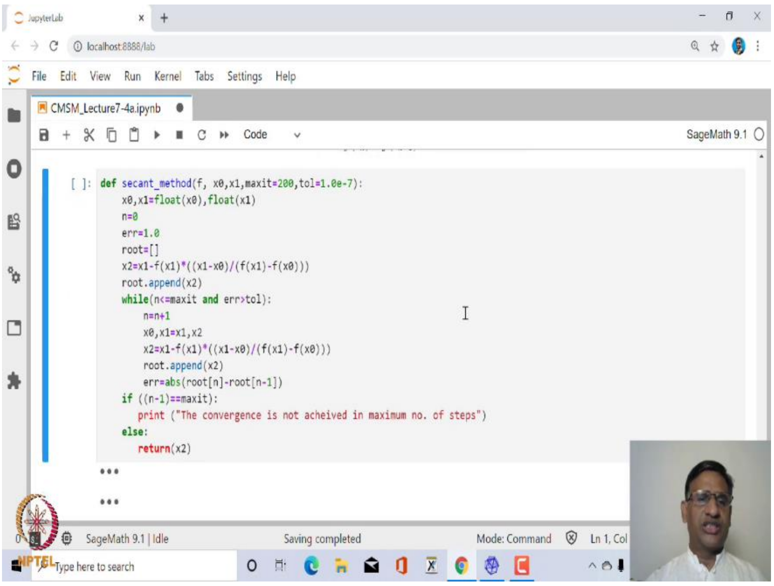
So, you can see here, the initial interval is minus 3 to minus 1, and then this is a root, and this is the 1st interval, this is the 2nd interval, 3rd interval, 4th, and so on, you can see here the length of the interval is becoming very small, right?

(Refer Slide Time: 15:44)



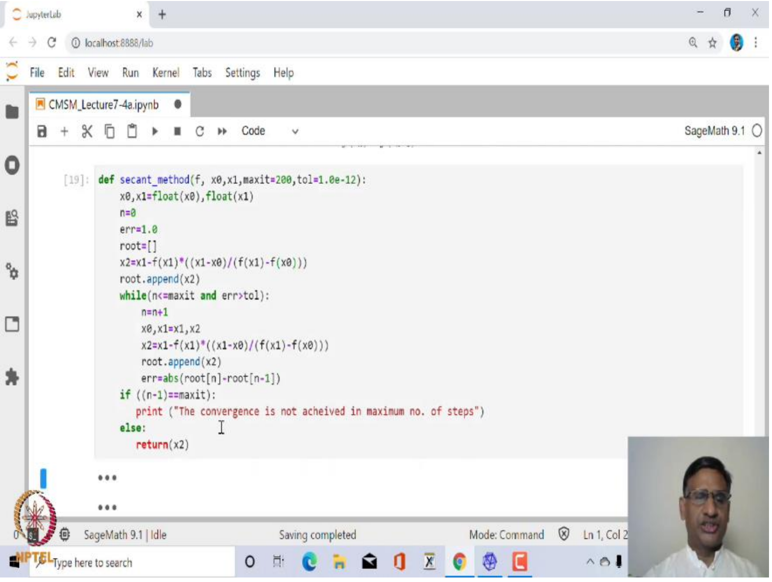
Next let us write our own program for Secant Method. The secant method this a iteration scheme is given by x k plus 1 is equal to x k minus f at x k into x k minus x k minus 1 divided by f of x k minus f of x k minus 1 again this I am sure you must have seen this.

(Refer Slide Time: 16:04)



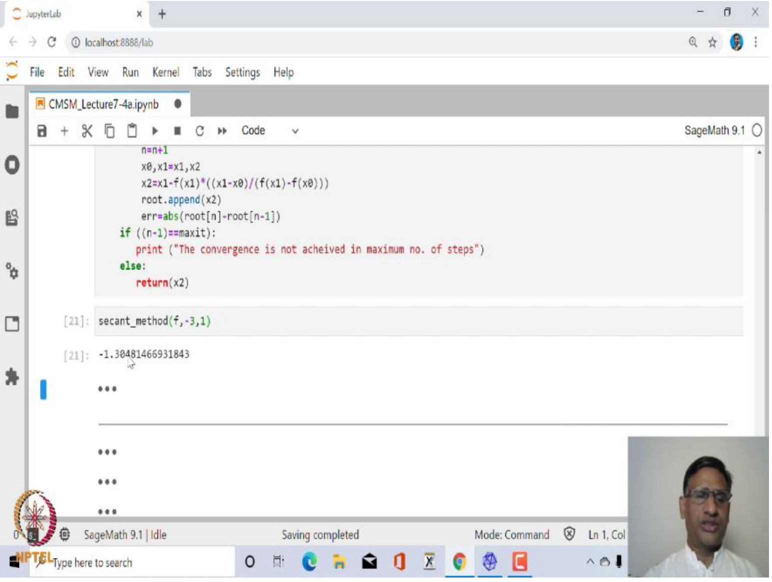
So, let us write a program for the secant method, again, very similar inputs we are giving, maximum number of iteration is 200, tolerance limit is 10 power minus 7.

(Refer Slide Time: 16:16)



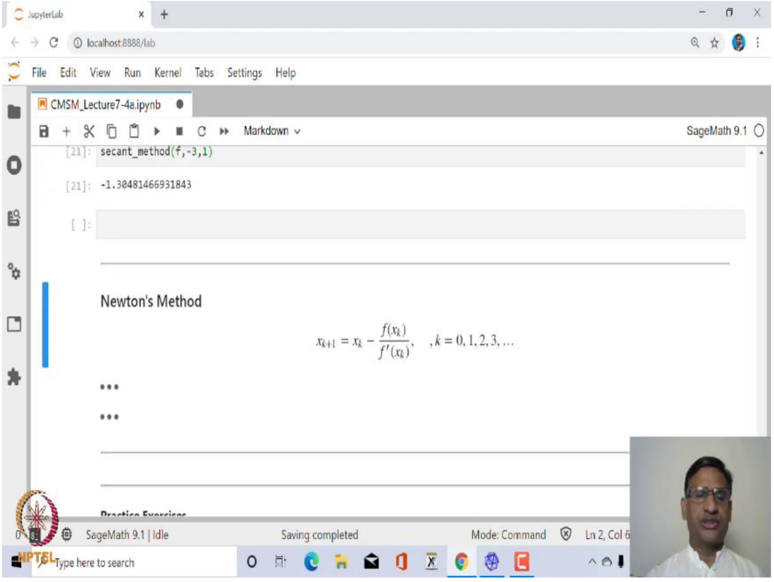
You can increase this to 10 to the power minus 12 and then let us execute this. In case you find that the number of iterate is equal to the maximum, and still you have not obtained this within the tolerance limit, you can print a message saying that the convergent is not achieved. And that is same as what you, you saw flag in case of inbuilt function find underscore root.

(Refer Slide Time: 16:44)



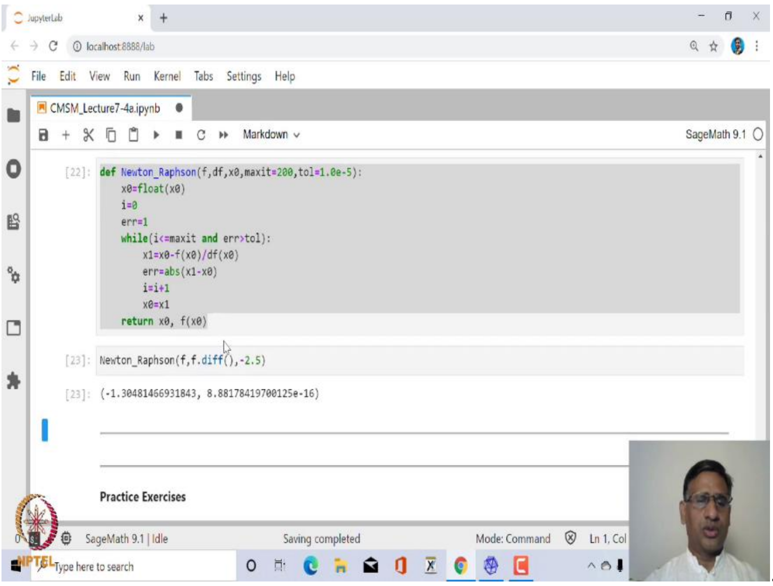
Now let us let us run this, and call this function for f(x). This is again, the root is minus 3, minus 1.30 and so on, ok.

(Refer Slide Time: 17:00)



Similarly, you can, you can now find, write your own program for Newton's method. The iteration scheme in Newton's method is given by x k plus 1 is equal to x k minus f of x k divided by f dash at x k.

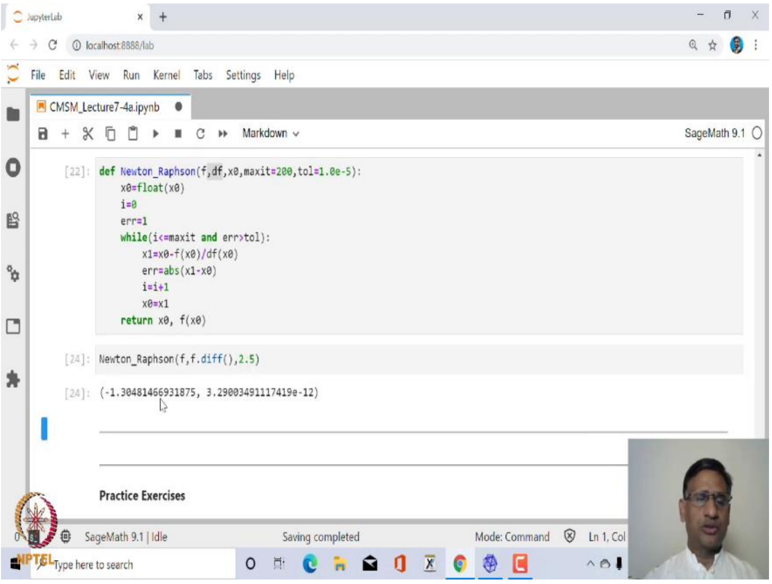
(Refer Slide Time: 17:12)



And you can create user-defined function for this. Fairly simple, very simple program, this we did in, in python also. And this, you can see here, this is exactly a python program, it has nothing to do with Sage, we are not using any Sage functionality here. We are using Sage functionality only for plotting and visualization, right?

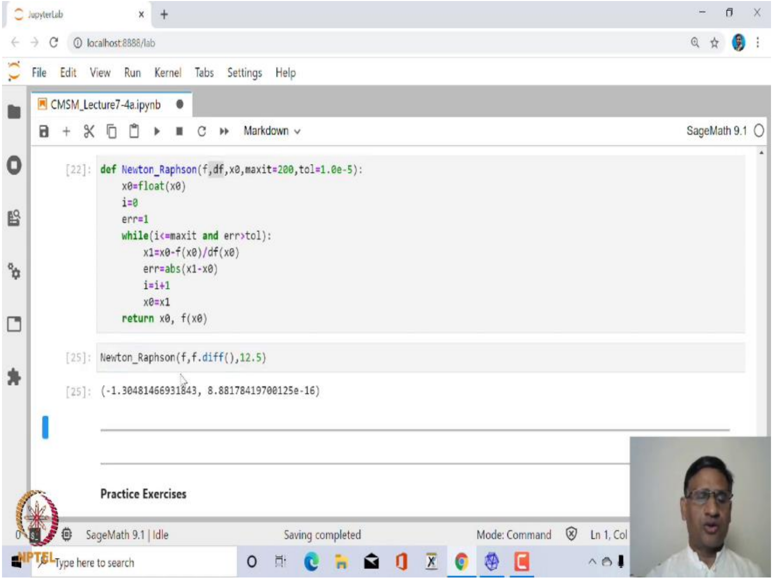
So, and then if we, if we run this, call this Newton's method, we, in this case, you need to input f and df that is derivative, right? Of course, without this also you can make use of, now, Sage functionality, and find the derivative to, define this df as f dot diff, and then, then run this, and in this case, you can see here we have passed df as f dot diff, and the initial guess is minus 2.5.

(Refer Slide Time: 18:09)

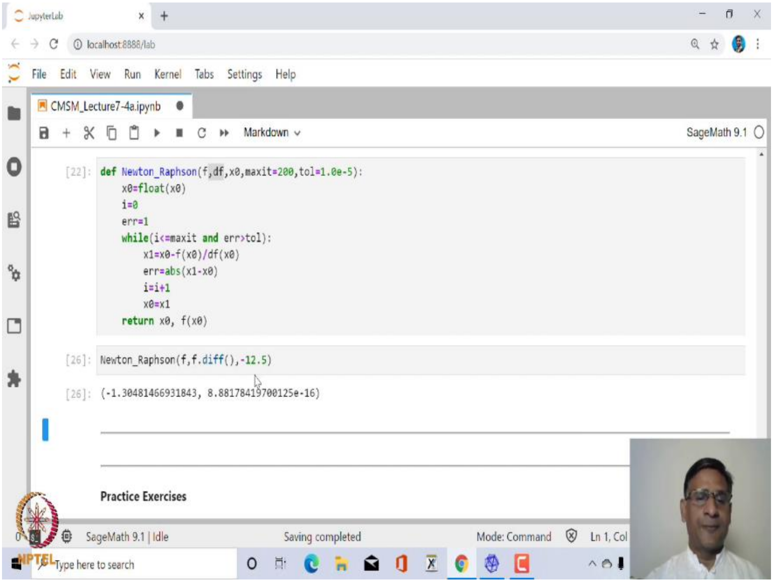


If I give initial guess, let us say 2.5, then also it is able to find.

(Refer Slide Time: 18:16)

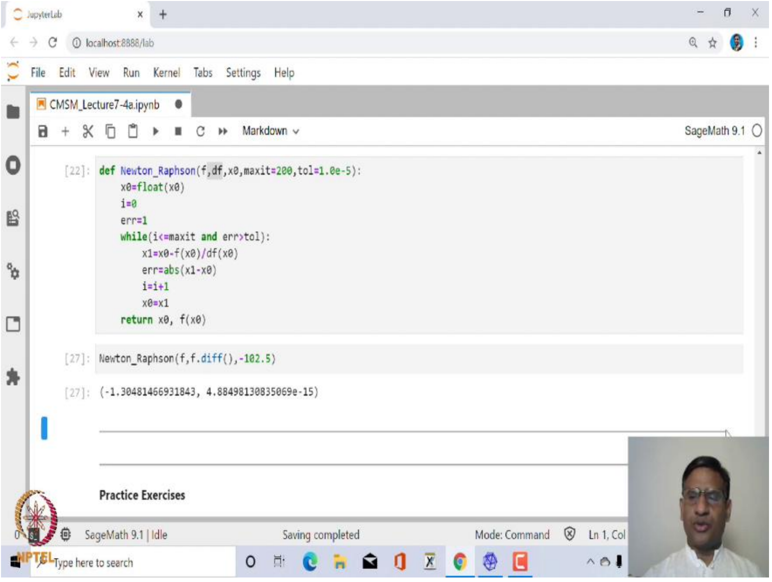


(Refer Slide Time: 18:20)

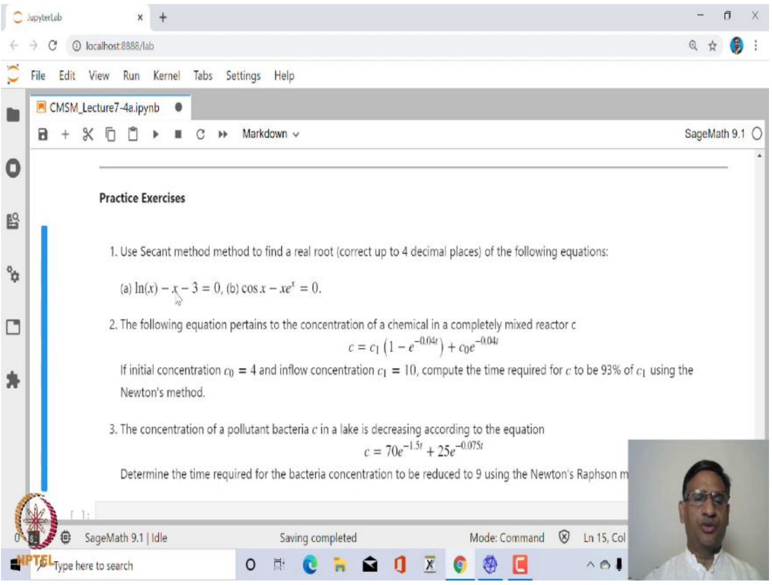


So, this is working, if I give you, let us say, 12.5, then also it is able to find, in case, if I give minus 12.5, then also it is able to find. So, this, this this function is quite nice, it seems that whatever initial guess you give, then it is able to find a root.

(Refer Slide Time: 18:34)



So, let us say, for example, if I give minus 102.5, this, then also it is able to, to, to find, ok? So, this, because it is looking at 200 iterates, ok, fine? So, let me, so what it means is that you can write your own programs for all these methods, there are several other methods to find a root of a, a quadratic, root, root of an algebraic and transcendental equation, equations. So, either you can use find root, or you can create your own user-defined functions, right? (Refer Slide Time: 19:20)



So, let me leave you with few simple exercises. So, try to solve these equations, find roots of these equations using secant method, this one using Newton's method, and this also using Newton's method. And in each of these case, you can verify this using inbuilt function find underscore root.

Of course, you should also plot graph and then try to see the interval in which it has a root, ok? So, next time we will look at how to solve set of, or system of linear equations numerically.