Scientific Computing Using Python

Professor. Vivek Aggarwal and Professor. Mani Mehra

Department of Mathematics

Indian Institute of Technology, Delhi

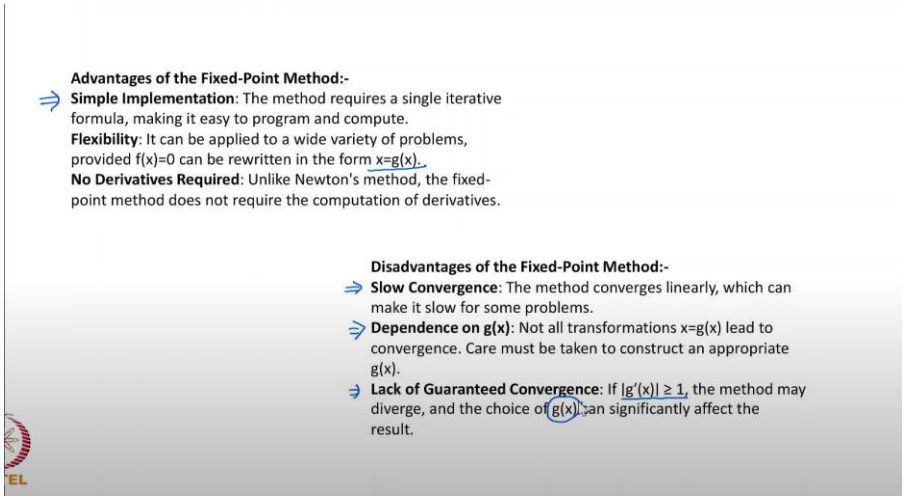
Lecture No. 09

Welcome to Scientific Computing Using Python. So today we will do the fix point method which we started with, and we will discuss the bisection method and discuss its Python code. Let's start. We have done it till here that how can we find the interval in which the root will lie. Now we do not know what the root will be and which function is the fix point function g(x). How can we define it, such that, the condition which was the sufficient condition is satisfied. So, we have explained how this convergence happens and from here we came to know that if the derivative of g(x) comes to less than one, then it will converge.

What are the advantages of fixed-point method? The best advantage of the fix point method is that it is very simple to implement. So, the method requires a single iterative formula. It is easy to program and compute. It can be flexible. It can be applied to a wide variety of problems, provided this can be written in this form. So, this is our condition that it should be written in this form, and there is no derivative anywhere in it. Whatever method we do, if we do Newton method next, then the derivative is involved in it, but no derivative is taken in it.

So, what is the disadvantage of this? Slow convergence. It will converge if that condition is satisfied, but it will do it very slowly. So, the method which converges linearly can make it slow for some problems and it depends on g(x). We have already seen that not all the transformations lead to the convergence. Care must be taken to construct an appropriate one, okay? And the lack of guarantee of convergence is that if $|g'(x)| \geq 1$, then the method may or may not converge, okay? So, the choice of g(x) is significant in this case, okay?

(Refer slide time: 2:43)

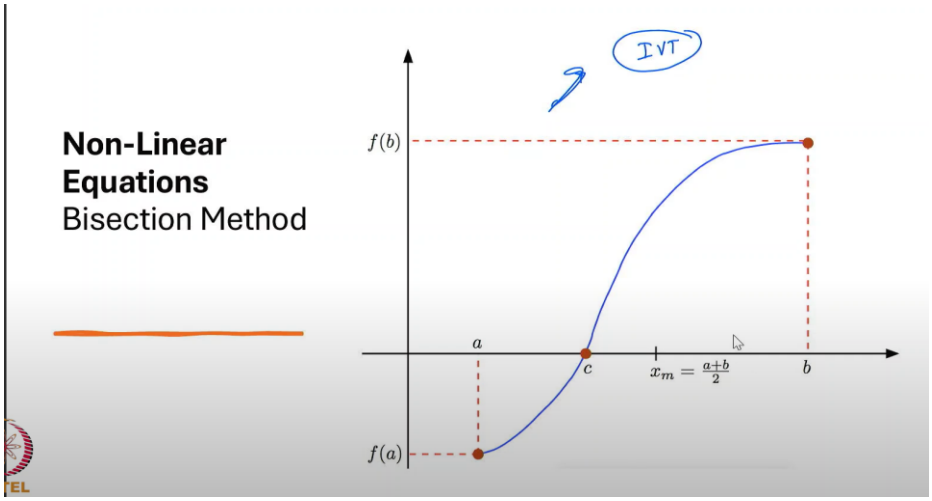


So, we have this fixed point, which we can discuss. So, after this, we will discuss another method, and that is bisection method. After that, we will discuss its Python code. So,

bisection method, which mean in this case, we will define and keep on bisecting. So, let's start it.

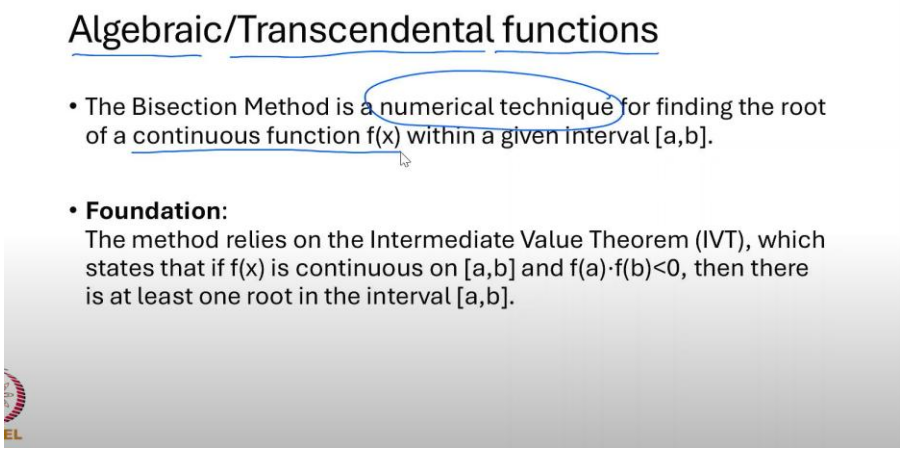
So, what do we have to do in bisection? What do we have to do in bisection? As it is showing, so now the first thing we have to do is to check where the root will lie. So, the intermediate value theorem is showing, which we had found out earlier. The intermediate value, the value of the function, which is a, and b. So here the root will come somewhere. So, what are we doing in bisection method? That we are bisecting the interval.

(Refer slide time: 3:52)



So how is this method working? As we said, it will work for both algebraic equations and transcendental equations. So, this is also a numerical technique to find out the root of the continuous function f(x) = 0, and what is its foundation? Depends completely on the IVT—initial value theorem. Initial value theorem, we had told in the previous lecture what happens. So, if the function is continuous, then we will definitely get some root or the other. If these conditions f(a) and f(b) are of opposite signs, then we will definitely get the root in this case.

(Refer slide time: 4:17)



So, what is the procedure in this bisection? Its name is also bisect—half-half we keep on doing. Now see, if this is an interval, so what do we have to do in this? First of all, we need two guesses in it, okay? So, two guesses are required. So, we start from x0 and x1. So,

suppose we have assumed x0 and this we have assumed x1. So, what will you do now? We have come to know from this that the product of f(x0) and f(x1) is negative. It means both are of opposite signs, and if they are of opposite signs, then by the initial value theorem, it will pass from here—it will be cut from the x-axis.

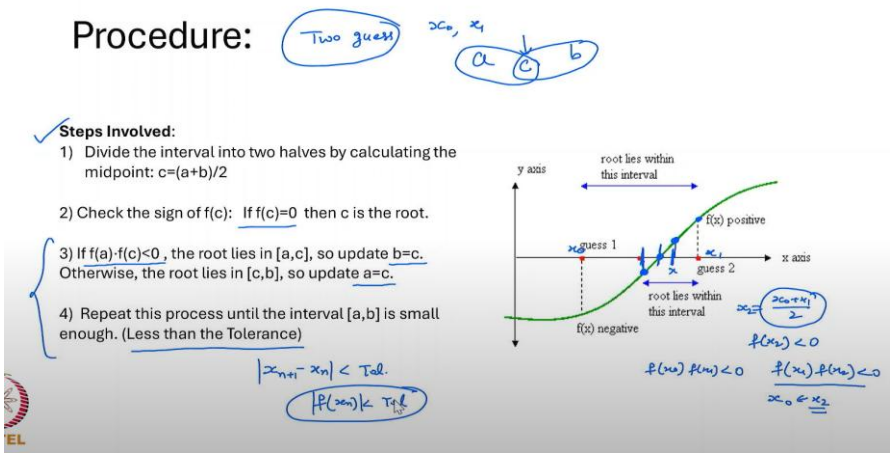
So, what did we do? We took the mean of both x1 and x0, and from there we got a new value. We named it suppose—we named it x2. So, this is the mean of this. We bisected it. Now you see, I bisected it, I got a new approximate value. Now the value of the function is this, and the other function is this. So, what will we do now? In this case, we got to know from here that in this case f(x2) is negative. So, what will we do? We will keep f(x1), we will keep f(x2), and since it is of opposite sign, so this means we will discard x0. So, what did we do? We discard x0 and filled x2 in its place.

So now we got a new value. So, this value came. After that, we took both this and this. Now after that, we bisected it. It came here. Now see, as soon as it came here, the values came here. So now what do we have to do? This x1 was in place of this. This x1 became our new one. Then after that, we will bisect both of them. So, by doing this a little bit, ultimately we will reach here and we will get our root.

So, what are the steps in this? It divide the interval into two halves by calculating the midpoint. So, we took the midpoint of both. Then we checked the sign to see if it has become the root—we have got it, okay. Then we will put the value of f(c). If f(c) comes zero, then it will be considered as the root. Otherwise, we will check its sign. So, we checked that if f(a) and f(c) are of opposite sign, then what does it mean? Our b becomes c. Or if it does not happen, then a becomes c, okay?

So out of the two, our starting was a and b. So, from there one came and we got c. So now what is happening is either b will become c or a will become c depending upon whether their values are positive or negative, and we will repeat this process. We will keep doing this for how long? We will keep doing it till our error; the iterative error becomes less than the tolerance. Or we can say that we will keep doing it till the values that we are getting here, |f(xn)|, are less than the tolerance or go towards zero. Less tolerance means almost zero. So, we will assume that whatever values we have, we have got the root.

(Refer slide time: 8:46)



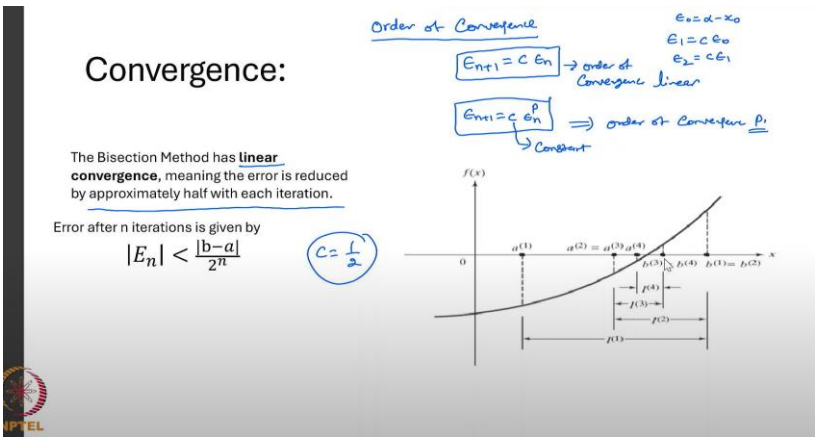
Now the convergence of bisection method is linear. So, what do we do in this? We define the convergence. So how do we define the convergence? So here I write order of convergence. By the way, this is a separate topic, and I will define that too. So, what does order of convergence mean? What did we do? We have iterative method, from there we are getting approximate values.

So, what do we do? We saw the error in some step n+1 and we got some constant multiple of the previous step. Okay, so what does it mean that we saw at any step and then saw at the next step that the error is coming after multiplying by one c, so what did we do, and we saw that e1 is some c e0 is coming, okay, e0 was suppose this was alpha which is our real root minus the initial gas that we gave, then that e0 came, then we got e1, then I saw that e1 into c, then it came e2, okay, so this is happening every time, so from here if this is happening, then we will say that its order, which is the order of convergence is linear and what will happen in reality, if we say that the order of convergence is p, then if this is happening to the some power p, if this is happening and c is a constant, okay it is independent of e, then we will say in this case that the order of convergence is p, okay, so if whatever power is coming is like one, then we will say that its order of convergence is one, similarly we have fixed point I had seen what is happening in the fix point, if you see the errors there, I had done it, you must have checked it here, it came, right?

So if I say this, then what does it mean, what is the power here, p=1, so what does it mean, this will also give linear convergence, it means the fix point method will also give us linear convergence and the bisection method will also converge linearly, so the bisection method is linear convergence meaning the error is reduced by approximately half with each iteration, so what is happening in this case, our c is getting halved again and again, so whatever error we found at any step, then after that the next step, it got halved, why because the width is getting reduced, earlier the error was somewhere in the middle of it, so there was an error at the root, next time we halved it, it became half, so the radius of the error got reduced and became half, right?

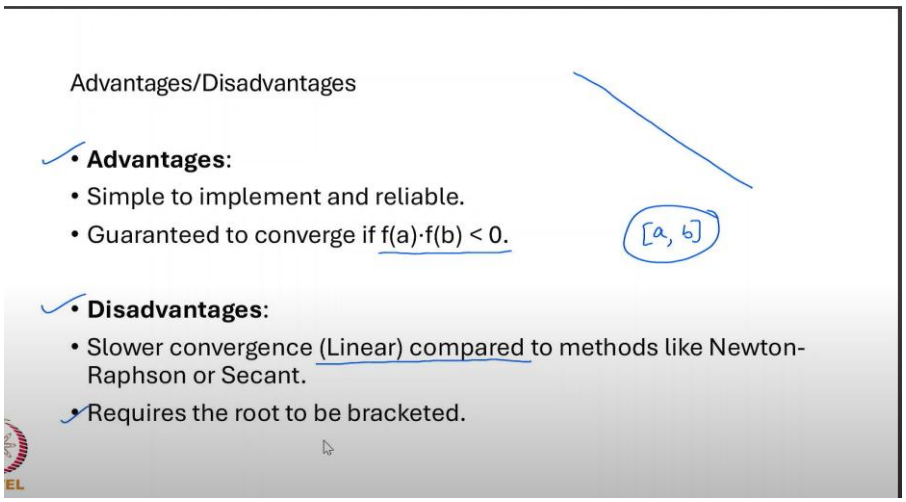
So, what happens when it is reduced to half is that its range is reduced to half so if we keep reducing it again and again then our error will keep getting halved so our method will remain the same in this way only the value of c will get halved in this way.

(Refer slide time: 13:11)



What is its advantage and disadvantage. So, the advantage is that it is simple to implement and reliable. In that we had to find g(x) in the fixed point, now we don't have to find g, even in this and convergence is also guaranteed as long as it is satisfied. What is the disadvantage? It is the disadvantage that this method converges slowly and it is linear convergent. The convergence in it is coming linear, so linear means that our convergence will come the error will keep getting reduced and require the roots to be bracket. We also call it bracket method, so we have the roots. And we first have to check where it is. So, we have to find this interval only then we will get to know that the root lies within it. So, we require the root to be bracketed. This is its disadvantage.

(Refer slide time: 14:15)

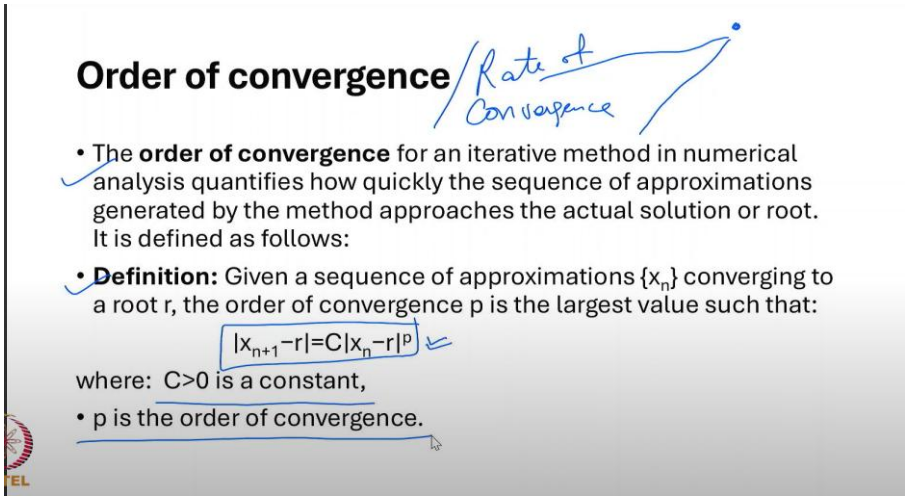


Now I had defined the order of convergence like this. So, now we have the definition of order of convergence. It iterates the method numerically — how quickly the sequence of approximants approaches the solution and the root. So, now we will do the iteration slowly, and we are reaching the roots comfortably. That is, we are reaching linearly.

So, if we are reaching fast, then we will say that its order of convergence is fast, and if we are doing it slowly, then we will say that its order of convergence is low. So, now we have to reach a point. Suppose here we have a point where we have to reach. So, if we are going slowly here and now we are going by running, then there is a difference between the two. In this, we will say that there is fast convergence in it, and in this, we will say that there is slow convergence as compared to the other one.

So, from here we have this formula. This r is the root of this formula. So, on every iteration, we will see how much error is occurring between them. So, if the error follows this, then we will say that the rate of convergence is the order of convergence. It is also called the rate of convergence. So, if I write it like this, then it can also be called the rate of convergence. We will say c is a positive constant and p is the order of convergence.

(Refer slide time: 15:56)



So, what do we have in the order of convergence? If we say that there is linear convergence, then what does it mean? The value of p is one. Okay, so in this, the value of p, as I told you, is in the fixed-point method, bisection method — both of these methods are converging linearly.

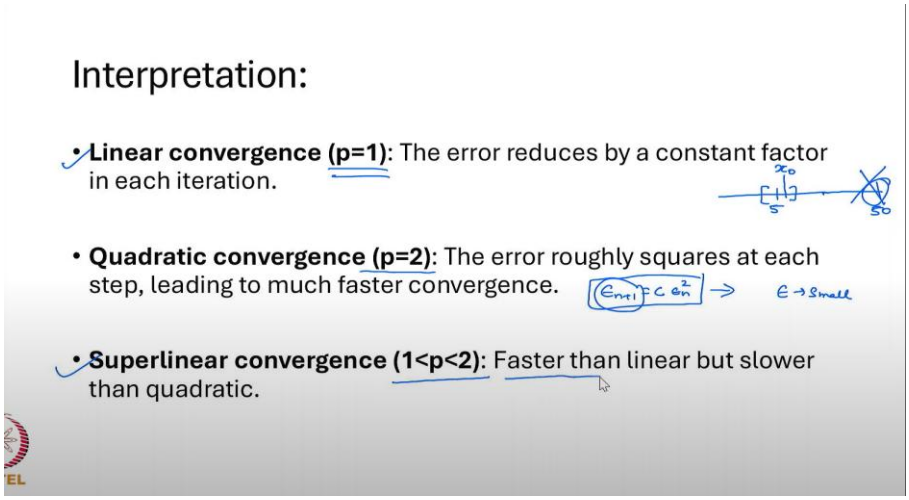
Now we have a quadratic method. What is happening in this is that the value of $e_{n+1} = c \times (e_n)^2$ is coming. What does it mean? When we square the error at n steps, it will be smaller because errors are very small. It is not the case that the error will be two, three, or four. No, that is not possible. The errors are very small.

So, if we have a root — suppose someone's root is 5 — and we started by taking the initial root, I started from 50, so the error between the two is 45. So, this will not work. So, the initial value that we have to give is very important. So, what we do is we go closer and find an interval, which we call the interval of interest, where the root can lie. We do not know what the actual root is in that interval, but we get to know that it will lie somewhere here.

So, after that, we choose a value. We call it x0. So, you see, the value of 5 minus x0 will be the quantity. It will be very small. Okay, so let's assume that it is a small quantity and less than one. If we square it, it will become smaller. Next time, we will get a smaller error. Then we square it, and then we will get a smaller error. So, ultimately, it will converge very fast. So, we call it quadratic convergence.

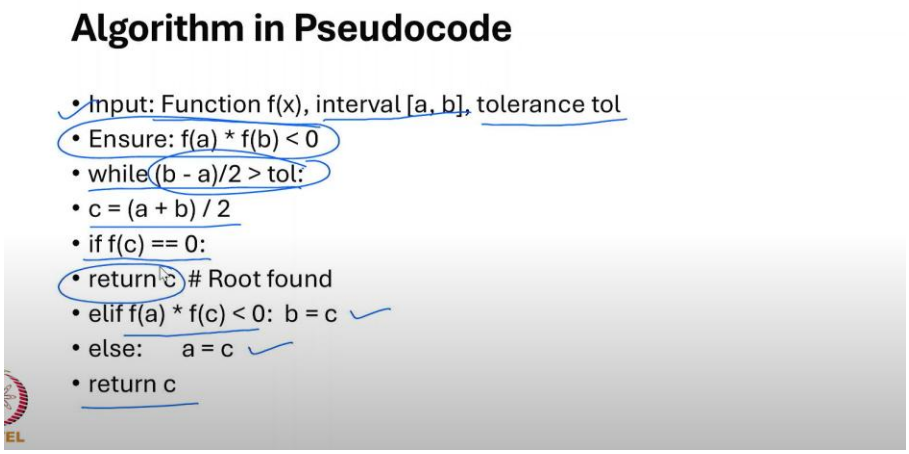
Super linear convergence happens when it is between one and two, faster than linear but slower than quadratic. So, the value of p will be greater than one and less than two. So, we will say it is super linear. So, this method that we have is the secant method. It gives super linear convergence. So, we will discuss that later.

(Refer slide time: 18:23)



Now we need an algorithm as to how we will do it in Python. So, what about the bisection method? We will have to define a function. First of all, we will have to define an interval in which the root can lie, and the tolerance will have to be defined. So, that's what we will do. Let us ensure that the multiplication of both of them will be less than zero, which means they will be of different signs. After that, we will apply a while loop, so that is (b – a)/2 > tolerance, only then it will be satisfied and enter inside. From there, we will take the mean. If we get this, then we will say, "Okay, we found the c," then we will return it. Otherwise, what will we do? We will calculate the product of both and show that if this is so, then b = c, if not, then a = c, and we will return c. This while loop will continue until we get a solution until we return here.

(Refer slide time: 19:46)



So, with the help of this algorithm, we will write the Python code. So, I have given an example. We can do any example, like for an example, I can do $x^3 - x - 1 = 0$, which we just discussed. So, here we have taken this example. I have taken the second example — I can take $x^3 - 2x - 8 = 0$. From here, we got to know that the root lies somewhere between one and two. Okay, in this we can see where the root lies.

So, I will do the same thing. If I put f(1) and see, if I put, say, zero, then the result is coming out negative. Okay, if I put a zero, then the result -8 is coming out. If I put f(1), then how much will it come out 1-2- 8, this also comes out negative. If I put f(2), then 8 - 4 - 8, so this

also comes out negative. If I put f(3), then 27 - 6 - 8 = 13, which is positive. So, it means the root lies somewhere here. So, in this case, our root lies in this interval [2,3], in which we have to take the initial guess, and from here.

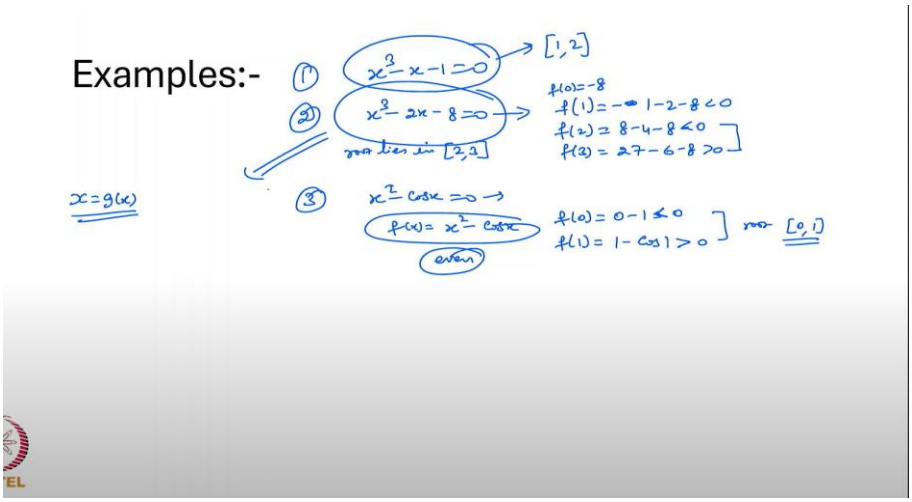
I can take a third example: $x^3 - \cos x = 0$. Now, we don't know how to solve this. What can be the root, but our function in this case is $f(x) = x^3 - \cos x$. Okay, so now if we take this as zero, then it will be 0−cos(0)=0-1, then it will be negative. Okay?

After this, Suppose I will take one, so 1- cos(1). Now we know that whatever its value will be, cos will lie between zero and one. So, it will always be positive. This means from here we can say that the root will lie between zero and one. And this our function, is an even function. Okay? So, it means minus of that root can also exist. It can have two roots: positive and negative both. Okay?

So, from here we will get roots. So, we can do this example and can find out which root will come here. With the help of this, we can find out which root we have to find out. Okay? So, by applying this condition, we can find it out. We came to know how we have to choose g(x) in this case. Also, we came to know in this case. Also, we can find out how to choose g(x).

So, in the same way, we have to choose g(x). If we want to do it from a fixed point, then we have to put x=g(x) in the value, and from there we have to find out g(x). So, we can solve both these problems with the help of Python. So now, let us see how we will use the Python code.

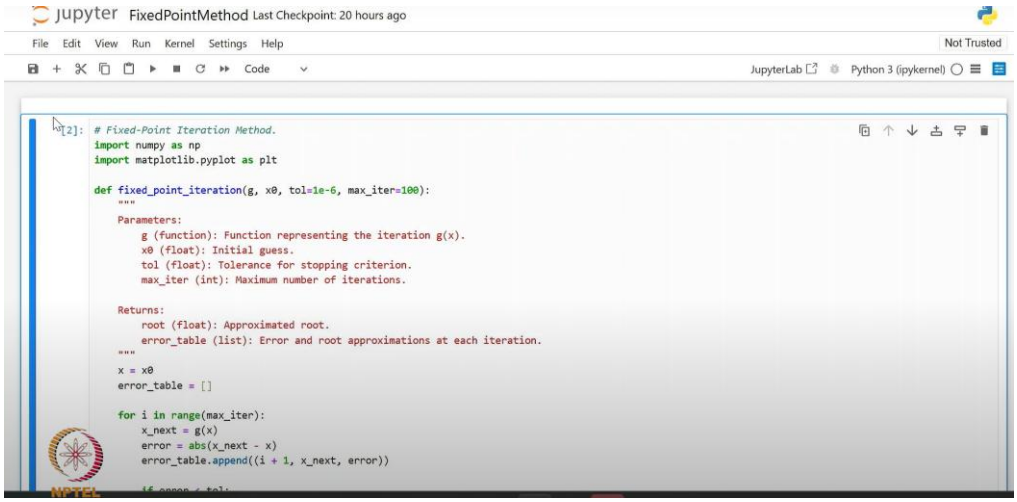
(Refer slide time: 23:16)



So, this is our Python, and I will open the file named “FixedPointMethod.” I clicked on it, I took it, and opened it. From here, I came to the desktop, and we clicked on it from here and opened it.

So, fixed-point iteration—this is the Python code that we have written. So, we will study it line by line as to how we can study it. Because now it is not like we will write it one by one, because now we have come to know how Python works—how we can write its commands.

(Refer slide time: 24:39)



So, what do we have to do in this? I named it "Fixed Point Method." So, we have to use fixed-point iteration. So, what will we do in this first of all? If we have to do all the calculations and find out, then we will use NumPy, and if we have to plot, then we will do it by matplotlib.

Then, what did I do here? I defined a function whose name I have written as fixed_point_iteration(). I will send g(x) inside it, which we know is the function we need. x is the initial guess, and the tolerance we need—so what can we do with the tolerance here? I can define it as 0.5 * 10^-6. Okay? So, it means 10^-6, or I can also write it like this: 0.5e-6. And the maximum iteration—I took 100 because we have to see how many iterations we need. So, we thought let's take 100. We took it like this.

After that, we defined the function. These are comments. These are for us, or if someone reads this, then they should know what the purpose of this code is. So, I have done it here.

I have defined the parameters of g and the function representing the iteration. x0 will be floating—initial guess. Tolerance is also floating, and maximum iteration will be an integer. We have that. What will be its value? So, we have also told its type. And what should it return? It should give us a root in return. Okay?

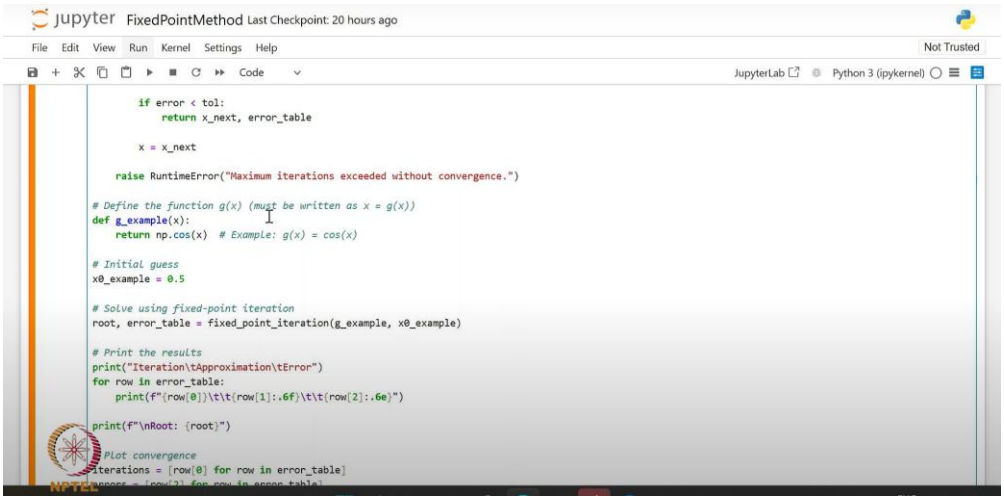
And after that, I also found out the errors in it, how much error is there in each step. Now, I took an initial guess x, error_table, I don’t know how many elements will be there in it. So, I left it open and made a table.

Now, what did I do? I put a for loop: for i in range(max_iter). So, we will go to the first iteration. So, I wrote x_next = g(x) okay? So, this x_next on the left will give the new x when we put the old x here.

Then, how to do it? What will be the error? abs(x_next - x). So, we already know—it will tell the absolute error. Okay? And this error will be saved here, error_table.append((i+1, x_next, error)). In this, the iteration, the value of x, and the error—all three will be stored in the table. So, if the error is less than the tolerance, then it will return. It will stop, and then it will return.

Otherwise, what will it do? It will go to the next point. x = x_next, and this is how we will keep finding out in the for loop. It will keep going. And at run time, we will get to know: “Maximum iteration exceeded without convergence.” If it does not converge, then we will get to know, and a message will come that it has exceeded.

(Refer slide time: 28:16)



Now, our job is to define the g(x). We know how to define it. So, we can define g(x). So, suppose we have taken the equation: x=cos(x). I have taken g(x)=cos(x). So, it means that what we are defining here is—I can write it here that we are solving x=cos(x). This is our function: x−cos(x)=0.

So, I have made it in the form of fixed-point iteration. So, we took cos(x), and then it is g(x)= cos(x), and we know that if we take the derivative of cos(x), then it is -sin(x), which is less than one in magnitude. So, it means that this g(x) function will satisfy our sufficient condition of convergence in the fixed-point method.

So, what have we defined? We define the function:

def g_example(x):

return np.cos(x)

We have defined it. Okay? So, what are we doing here? We will input x, and in return, we will get cos(x) because we have to calculate it every time at every value.

Now, we took the initial guess we took 0.5. Why? Because x−cos(x), so the value of x on 0, we know that it is negative, and its value on one is positive. So, the root will lie between zero and one, and we have just written—I had found out that its root will lie between zero and one. So here we can write in the initial guess. Why did I do this? Because the root lies somewhere between zero and one. Okay?

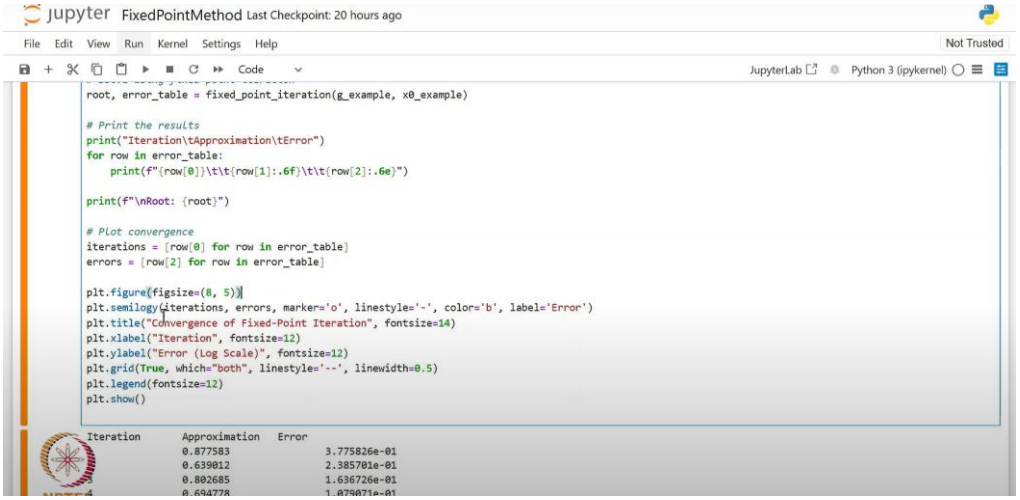
So now I have solved using fixed-point iteration. Now what will we do? We have to call this function that we had defined. So, what did we do here? We took the initial guess, and from here, we called it. I sent the g_example, and sent the x0_example. Okay?

And from here, whatever value we get, it will return the root and the error table that we had done—it is returning the root and the error table. From here, we get this value. Then, we will keep printing. We will keep printing in print—the iteration, approximations. And we print the rows in the error table.

So, this is one way of printing that how can we print it so that we can see it clearly. Okay, row zero – first element, row one – second element, row two – third element, because we

were writing only three elements in it, this one, then this one. So, we will have three columns in it basically. Okay, and we will print it.

(Refer slide time: 32:13)



After that, the root. Now what we have to do is that the iteration will keep coming in the roots, errors will also keep coming. After that, we will calculate everything and finally we will plot it. So, we have defined a plot: plt.figure and given its size 8 and 5. Okay.

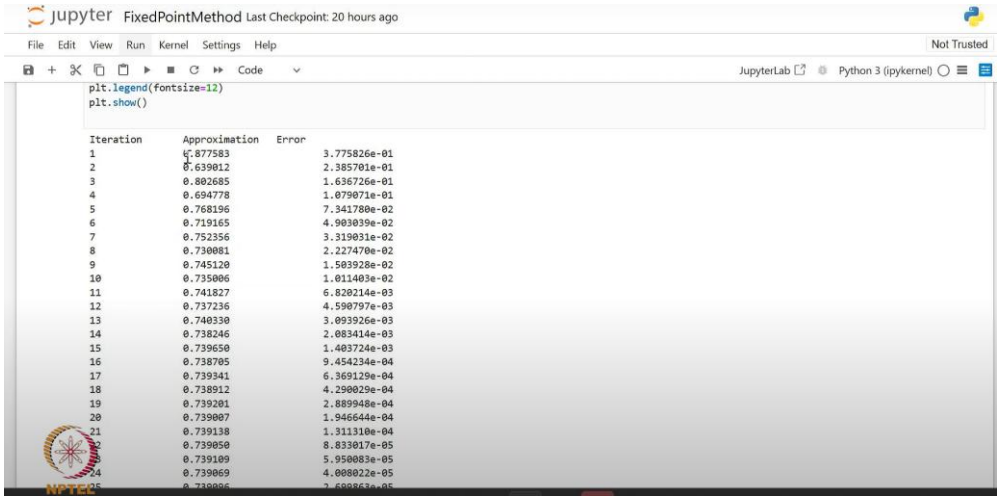
Iteration semilog – so we have defined these: iteration, errors and markers, which we had just defined today. That means we had defined the style, which has to be given like this: keep the color ‘b’ (blue), the label is the error like this. I have given it a title. What do we get from here? It will give this title. The name of this title is "Convergence of Fixed Point Iteration", and we have given its font size 14.

At the x-label, we have given "Iterations", and at the y-label, I have given "Error (Log Scale)", because basically we keep the log scale since the error will be 10 to the power of something (minus something). So, if we take log, then basically its powers will appear in front of it: -1, -2, -3. In this way, we can take the log value.

Then we have turned on the grid. So, we will get a grid. We will apply legend. Legend is a curve, which is a legend. And then we will show this plot.

So, we have followed all this procedure. Now what will we do? We will run it and see what answer we got after running it.

(Refer slide time: 33:37)

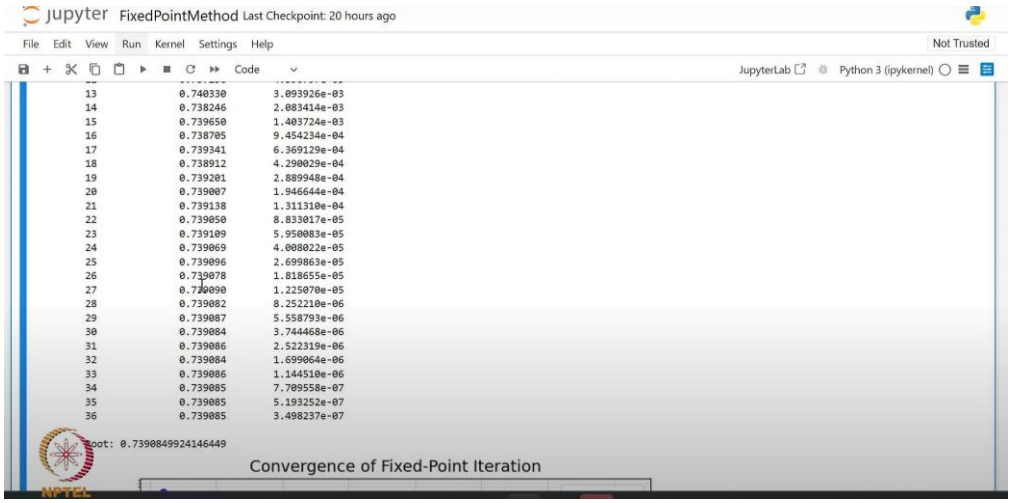


So, as soon as I ran it, we saw that the first iteration came, We got the approximate root and error.

So first see, first approximate error: 3.77 × 10^-1. Now see, this error that has come is not in the normalized form. It is in the scientific form because we had shown in the scientific form that if the value of p is that comes in between 1 and 10, then the values are called scientific. If they come between 0.1 and 1, then it is called normalization.

So, the errors coming in this are in scientific form. Then we took the next iteration, and our approximation improved, and it became 0.63, and the error reduced a little. Then we went to the next iteration, and the error reduced further. Okay. Then when we went to the next iteration, the error reduced further. So, it means the error is reducing again and again.

(Refer slide time: 35:38)

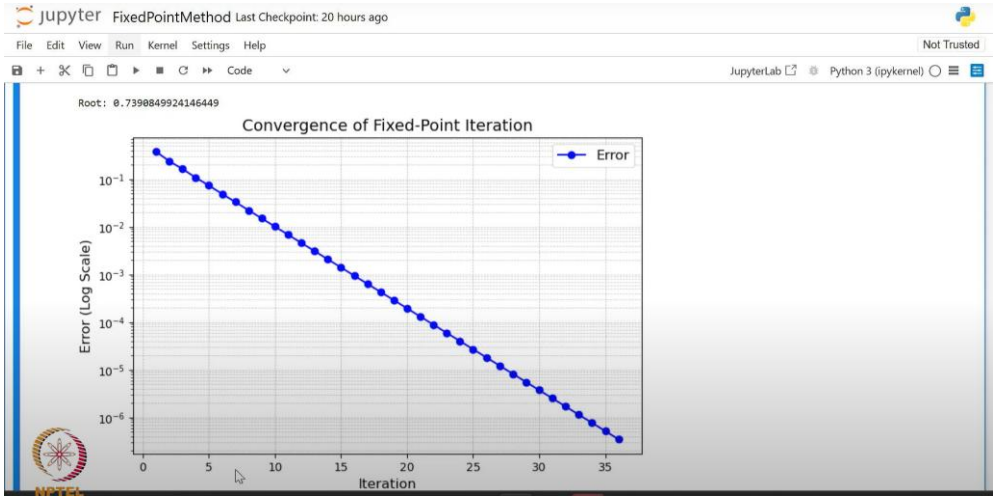


Now it has become 10^-2. Here, if you see, in the fifth iteration our error got reduced and became this. Now like this, our iteration continued and we saw that after 36 iterations or 20 or 22 iterations, see, almost the same was coming, and in the last, I saw that after 36 iterations—now you see the 34th iteration, 35th iteration, 36th iteration—see, the same error is happening.

So, the error that we have is here. So, what happened? The error has come to e^-7. So, what will we do with it now? From this error, we got to know that it is converging. So, this method is converging, and its root is close to 0.73.

Now what I did was we plotted it. So, we can see the error from the plot also. So, as I saw the error from the plot, it has come to us—see—it is linear, isn't it? It is linear convergence. So, see what is happening: the error is happening on the log scale. The value is coming written as -3, -4, -5,-6 .

(Refer slide time: 36:50)



And this is the iteration. You see, we started from the first iteration, so our error was this. It got reduced and reached here. On the 36th, the error came to 3.4 × 10^-7. So, what does it mean? That it has given us the convergence. Okay. So, from here, we got the convergence. So, this is our method. This code has worked.

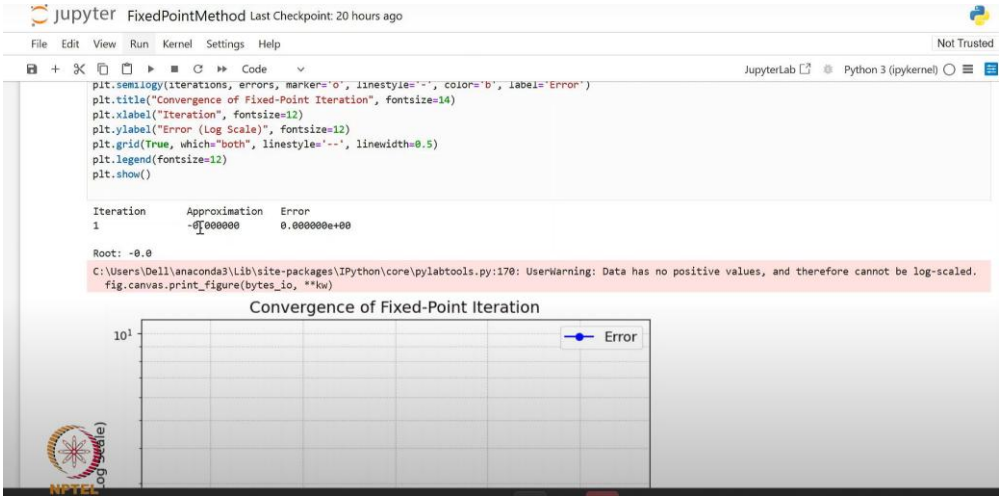
So, in this way we can solve it. I can write sinx in place of cosx. So, I gave the values suppose I define it. Where is this? I will not do this as 2 because this can increase the quantity. So, I write, suppose, sin here.

So, what did I do? Let's try to plot x -sin(x) . Let's see what happens. But x= sin(x), If you see, both the values will merge at zero only, because what will happen at x = 0? Both the values will be g(0) – 0 that is 0. So, zero will become its root. So, what does it mean? That in this, we have to take that iteration.

We can take the guess from -0.3. So, what do I do here? I take it 0.3. Let's try to take it. Okay. Now let's see what happens. And I took the sine. Then an error came. So, what happened in this? The maximum iterations exceeded without convergence, so its not converging in this because we know that the root of sine is zero. So, in this case, our value will exceed, then from here we will know that the convergence could not happen because the maximum iteration that I have kept 100.

It is possible that I close its values a little more or take it in minus—say, -0.2. Now let's see what will happen. Then the maximum iteration has come. Okay, so this maximum iteration has exceed the runtime. Now it is possible that by working in it, it is happening in 200 iterations. If we check it, it has come more than 200 iterations. So, it depends on how we are behaving with it.

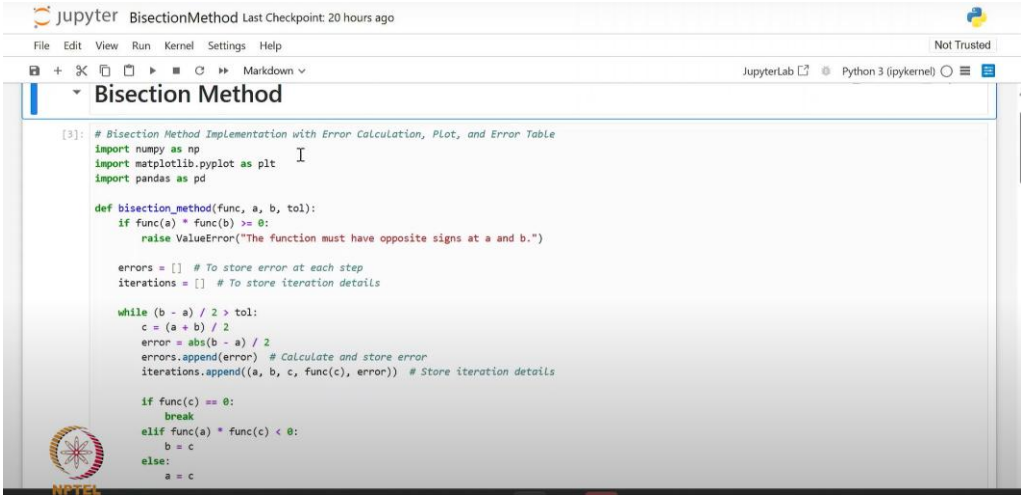
(Refer slide time: 40:02)



If I take it as zero, then it becomes zero because we know that its solution was zero. So, it happened in one go. So, what happened in this? We gave it its root. So, from here, we got the value. So, it depends on what our function does. This is how we are behaving—we are taking the sine, and we are taking the cosine, and what are we taking? So, we can define such functions.

Now what I do is I open it and do the bisection. So, what have I done in the bisection? We have taken this function: x^3 - x - 2. So, we have taken this value because in this, we do not have to define any g(x). In the fixed point, we still had to define it. But in this, we do not have to define any g(x).

(Refer slide time: 41:21)



So how can we do the bisection method? And this is it. So, what do we have to do in the bisection? We have to calculate the same error, plot it, and make an error table. So, what did we do in it? We imported NumPy, Matplotlib, and Pandas. Pandas is also a library that is useful for us related to data. So, we defined it.

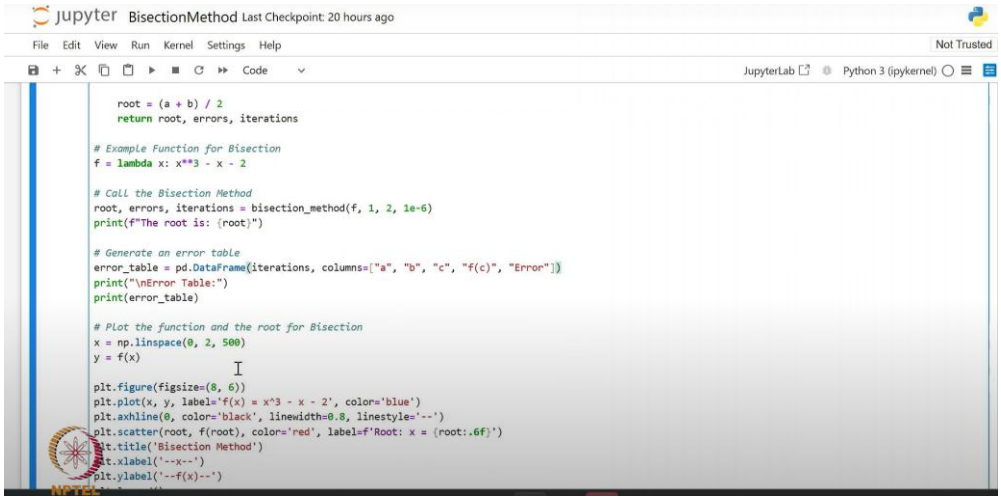
And what did we do? We gave a name to the function: bisection_method. And here I have defined this. Now what do we have to do for both of them? We have to check the values. If it

is positive, then what should we do? The function must have opposite signs. If it is not, then let's make an error table. Let's make an iteration table. And then apply this while loop.

If (b – a)/2 is greater than the tolerance, then it will go inside this, because it will be true.

Then we will get c. And as soon as we find c, we will find out the error. And after that, we will save it and append it in error and in iteration. So, we will save the value of a, b, c, the value of function and error of the function, and we will store it here. So, if c is its root, then break, and we will get the solution. We will come out of this if loop. Okay, we will come out of the while loop. Otherwise, we will calculate and convert b to c or we will convert a to c.

(Refer slide time: 44:01)



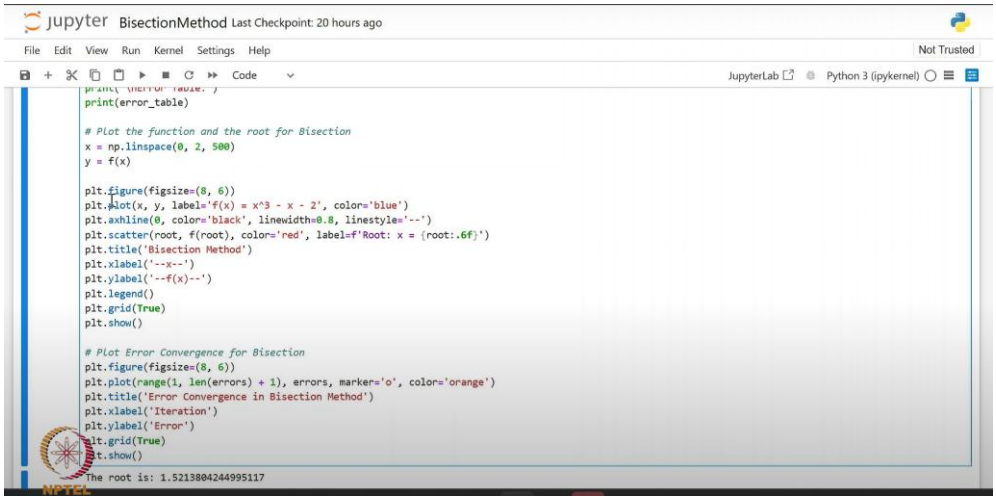
Okay, so now we have written its root like this. And from here, we have returned root, errors, and iteration.

Now see, in this, I have defined a function. Okay, so this function is lambda x, so we have defined it as x^3 - x - 2. So now if you look at this, x is negative on 0, it is negative on 1, it becomes positive on 2. So, its root will lie between 1 and 2. So, I called the bisection method. In it, I gave the initial values—a was given 1, and b was given 2, and passed the f function. I called it and printed its root.

So now what will it do? It will keep calling like this and will keep iterating again and again, and its values will keep getting saved in our error table. So, the values are getting saved inside the error table. And after that, what will we have? Our error table is created here, and we printed it and printed the error table which came.

Okay, so now what do we have to do? We have to plot it also. So, we plotted the function. The linspace, which is it, creates a mesh between 0 to 2 with 500 points. Okay, and then we have defined the function y= f(x). Now what will we do? We will plot it.

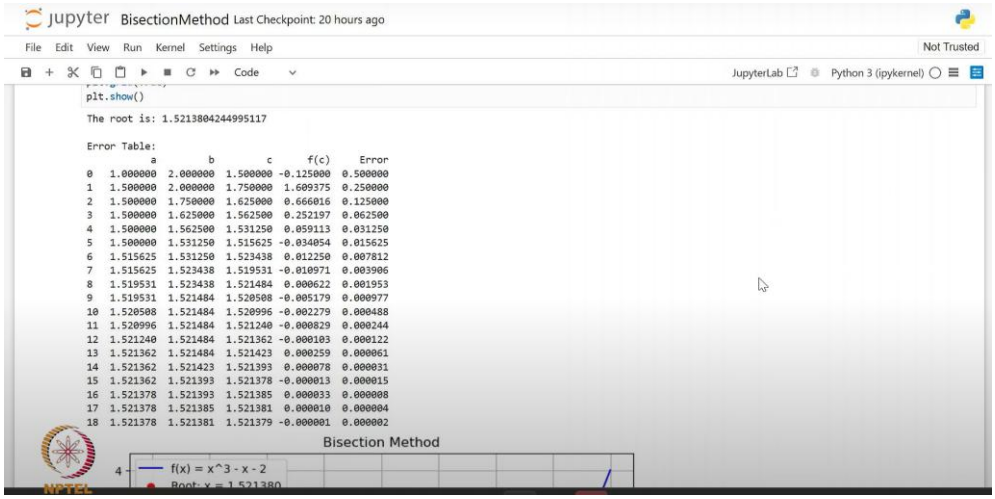
(Refer slide time: 44:23)



So, in plotting, we have the same plt.figure(). Okay, I have defined this function here. I have defined its x line. We can do scatter, we can make scatter in it. I have written "Bisection Method", I have written xlabel, I have written ylabel, I have written legend, I have turned on grid and shown it. Okay.

So, these things we ran. Now I have given the values a and b, 1 and 2, and I have run it. So, as we run, we get this value. And the root is—we know that it will come between 1 and 2. So the root that has come from here is this. This is the root: 1.521378... This root has come to us.

(Refer slide time: 45:21)



Okay, so if I do this—see, first I gave a as 1, then b as 2. Now what will be the mean of these two? 1.5. So, this came. I saw that fc is negative and fa was also negative at 1, and it is also negative at 1.5. It means that the value of a will replace c, c will replace a.

So, what does it mean? Now what will our a become 1.5? It was 1.5, b is 2 only. Then we took the mean, and it came out to be 1.75. So as soon as 1.75 came, we checked, and we got a positive value. So, it means that b will replace it. So, look, here b replaced 1.75. My a remained the same. So, from here, again c came, then it came out to be positive, then I replaced b, then again replaced b. Now as soon as a negative came here, what does it mean? That now we will replace a. So, I replaced a.

So, we kept going like this, and we saw that the errors that we had in the end became this. So, it means that till the fourth or fifth digit, the solution will be exactly the same. There will be no error. There is an error of 2 on the sixth digit, right? The difference here will be the difference of 2. So, we can see from here that our value of a is 1.521378, and the value of the second one is 1.521381. So, see, here if we take the difference of both, we will get 10^-6, right?

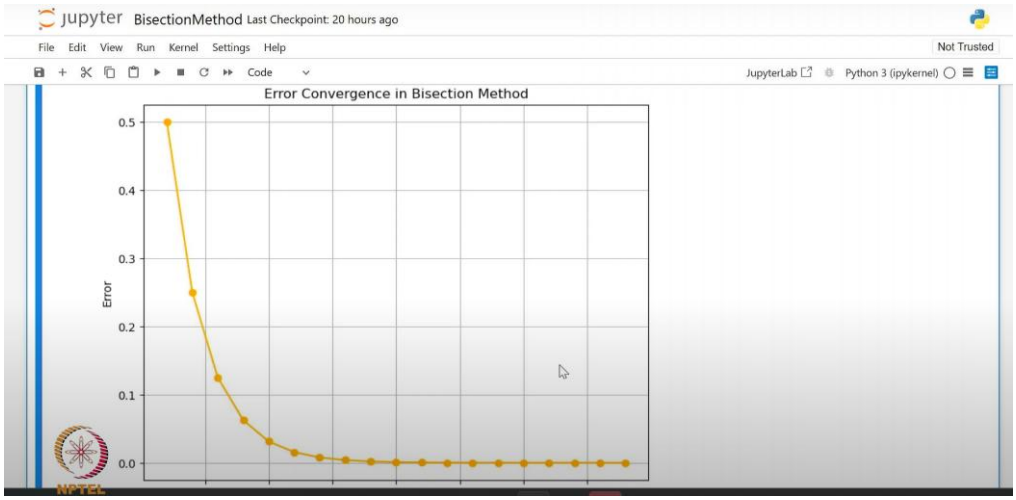
So, we kept doing this error like this because the error we had given was 10^-6, right? So we had to see till six digits. So, till five digits, it will give us the correct values—same, right?

(Refer slide time: 47:28)

So, we plotted it from here. Now we plotted it through bisection. So, this was our function, and this is x. So, we will see that if we plot this function, then our root came, and the value of the root is around 1.5. So, our value came out to be 1.521378, okay?

And the convergence given to us keeps reducing like this. See, the error is getting reduced with iteration. Okay, so what is happening in this? The error is coming absolutely linear. Now as we had seen in the fixed point, it was coming absolutely linear. In this, we are not getting linear. Here the values are decreasing completely, and here once, after reduction, we are getting to know that there is not much change.

(Refer slide time: 48:10)

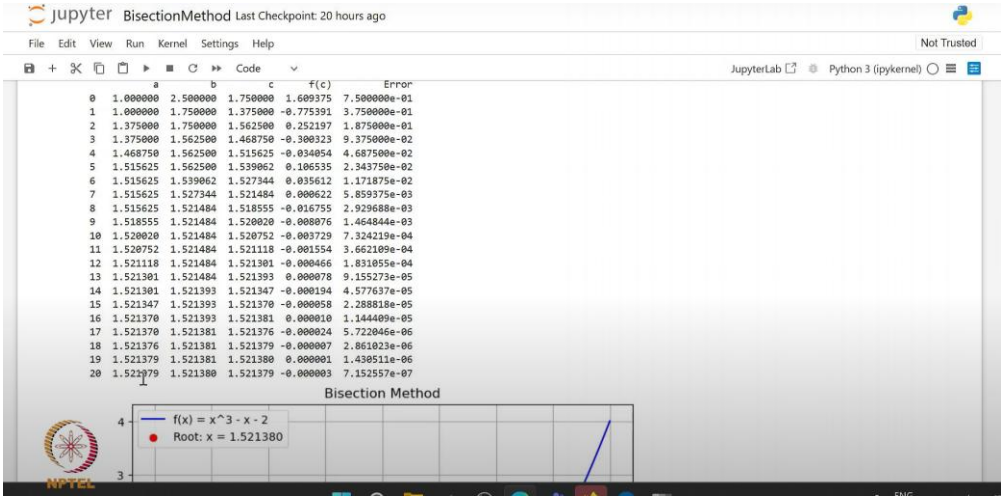


So, it means that in the beginning, it is reducing the error very fast, but after that, the error reduction has reduced and slowed down. Okay, so now we have the methods, so we will keep changing the equation like this.

Now see, instead of 1, I will make it 2.5. Suppose I did it like this. Okay. And I will let it remain 1. The convergence is—I will write it as 0.5. Now let's see what happens. It has come. Okay.

So, there we did that. If at n number, we have to reduce the error exactly. So now see, we took a as 1 and took b as 2.5, so the error that came here, which was 7.5 in 10^-1, got reduced and came here. Okay. So, the error reached from 10^-1 and reach to 10^-7. If it became smaller than ε, then it stopped. And here it was big till here, it became smaller here.

(Refer slide time: 49:47)



So, like this, if you see the error that we have, what happened in this: 1.521379 and 1.521380. So, see the difference between both the points, then it came out to be 0.000001, okay?

So, if we calculated our error like this, then our error will come in this form. So, the value of fc also came out to be zero, because if c is the root, then fc should come out to be zero. So, it will come out to be zero there.

So, in this case, when I plotted it, then we got this root. The root will remain the same. Our root is around 1.5. It will remain the same. The convergence is also almost the same because we just changed the initial guess. Okay?

So, if we change the initial guess, then this will come out. Now suppose I change it a little bit. As I said—so if I change its values to x^3 - 2x - 8, let me see what happens then. Let's see. In this, we had seen that at 1, f(x) is negative.

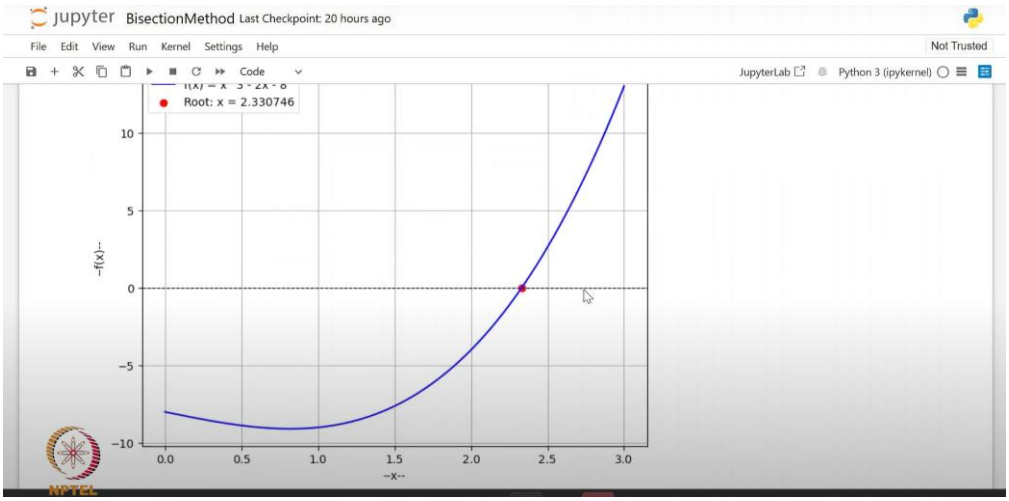
On 1 and 0 it is also negative. The root will come between 2 and 3. So what will we do here? I will send 2 here and send 3 here and write it here as well, because this will be a label. Let's write it down. So here it comes: x^3 - 2x - 8. I have defined it. So, let's see what happens.

So, what is its root? It comes to 2.33. And the error reached here at the end. So, the roots that we got are exactly the same up to five digits. The difference is of two at the sixth digit. Because we had defined it here 0.5 in 10^-6. Okay, so the values are here.

Now, let's see. We have plotted this nice function. We have not changed its axis. So, okay, that function will be plotted. This is the bisection method, and here we have done it. Okay, so this—the root which is 2.33, and if we plot this function between 0 to 3, then here we have this function.

From where did we say that? So instead of 0 to 2, I will make it 0 to 3, because it is in between them. So now I will see what I do with it. Now it has come. See—so the root which we have is this, and this is 2.33. The root has come here.

(Refer slide time: 53:10)



And the convergence is how our error is getting reduced. So, this table is showing that the error will reduce very fast in the beginning and after a point, it will become almost the same. So, this is the convergence, because this is the bisection method. In this, we have seen that the error is halved. So, every time, one error is getting reduced by half. So, every time it will be reduced by half.

So, our function, which is the error, is reducing in this way. It is becoming half. So, from here, we can apply the bisection method. So, once we get the solution, we can change any value here. We can give any function here: we can give x=cos(x), we can give x= sin(x), we can define any function here.

So, like this, we will write the code of the bisection method. And after that, we will apply different-different functions. And from there, we will get all these solutions. So, from this, we will come to know that convergence is happening and how our tables are being made. So, this is very beneficial for us. From this, we will also come to know what will be its root.

So today, we discussed the fixed-point method and bisection method. And we also saw their Python code. And from the Python code, we came to know how we can make the error table and how we will come to know from the plot that our method is converging towards the root of the given function.

So, I hope you liked this lecture. Thank you for watching this lecture.