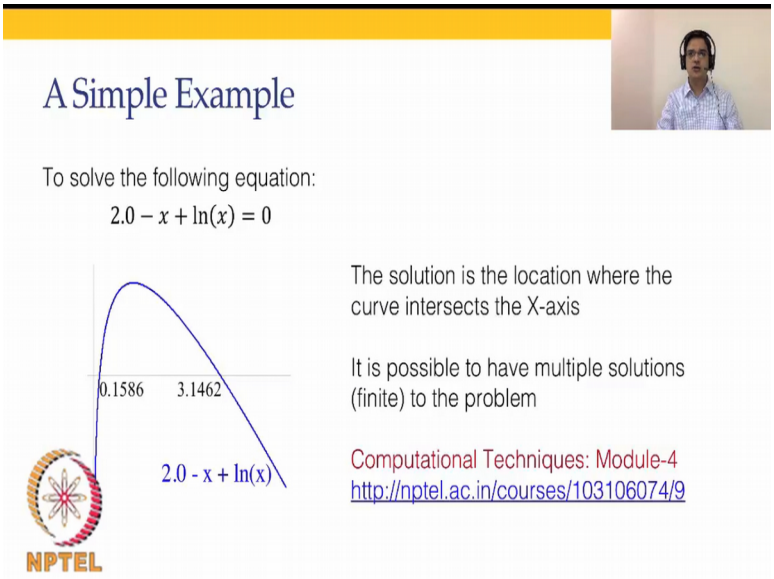
MATLAB Programming for Numerical Computation
Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #05
Lecture No. #5.1
Non-Linear Algebraic Equations - Nonlinear equation in single variable

Hello and welcome to MATLAB programming for numerical computations. We are now, in week number 5. We are about the halfway mark, in this online course. In this week, we are going to cover nonlinear algebraic equations. In the first couple of lectures, we are going to cover nonlinear algebraic equations in a single variable, and finally we will cover nonlinear algebraic equations in multiple variables.

The first lecture, lecture 5.1, is going to be about, introduction to ourselves for nonlinear equations. And we will use, one simple method known as the Bisection method, in order to solve one, an example of nonlinear equation in a single variable.

(Refer Slide Time: 00:55)



So, the simple example that we are going to talk about today, is to solve the equation 2-x+ln(x) =0. Now if I am going to plot this function on the positive x axis, this is what the function is going to look like, the blue color line represents 2-x+ln(x). Graphically, the solution to this

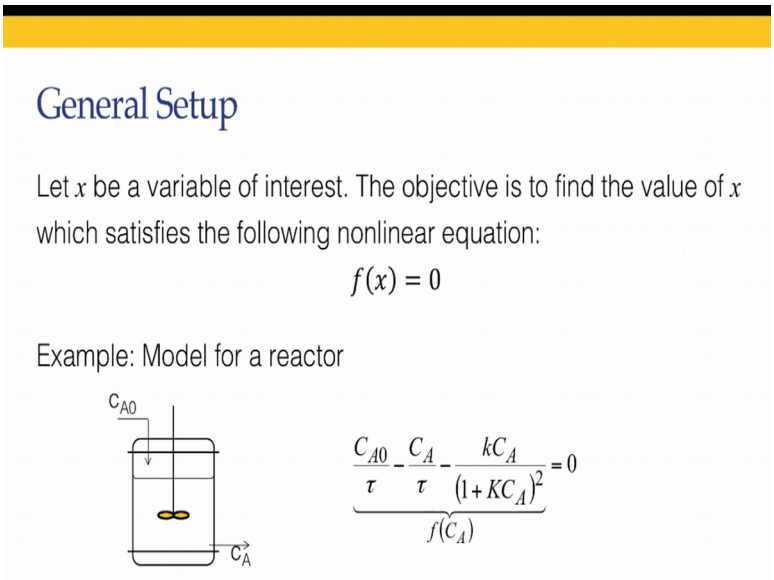
particular curve, is going to be nothing but, all the points that the curve intersects the x axis. Keep in mind that, while the solution that is going to be the solutions where the curve intersects the x axis; it is also possible to have multiple solutions as seen over here.

Unlike, the linear equations case, in linear equations, if the matrix A had is full rank matrix, then we can have only unique solution. However, in nonlinear equations case, we can have finite number of multiple solutions. If you recall from linear equations, whenever we had multiple solutions, we had infinite number of multiple solutions okay. So, that is going to be one difference between the linear algebra linear equations case and the nonlinear equations case.

The related set of lectures, in the computational techniques course is in module number 4 of computational techniques, the link for the first lecture is shown over here. So, you can go to that link, in order to find out some of the more theoretical stuff, related to solving nonlinear equations. What we are going to primarily do in this lecture series is, really take examples of solving nonlinear equations and solve them using MATLAB.

So, we are not going to derive any theory, but our primary focus is going to be on how to use MATLAB, in order to solve this nonlinear equation.

(Refer Slide Time: 02:45)

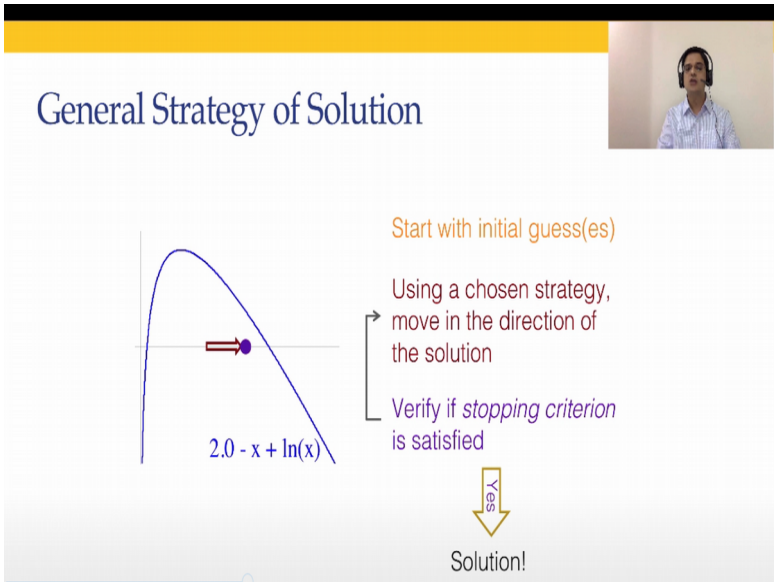


So, the general setup, is going to be, let x be the variable of interest. And the objective is for to find the value of x, that satisfies the nonlinear equation f(x) =0. An example for this is a model of a reactor and the model of the reactor follows an equation that is given below. In this example that x gets replaced by Ca and we are going to solve to obtain the concentration Ca, at the outlet of this reactor okay.

There are several other problems of interest, from various engineering discipline, as well as from various sciences. In this particular, set of lecture series, we are interested in solving, any generic equation of the sort f(x) = 0. When we talk about, a variable in a single variable, or a function in a single variable, we basically mean that x is a scalar valued variable, and f is a single scalar valued function.

In case of multivariable case, x is going to be an n dimensional vector, and our function f is actually not going to be a single function, but a vector of n functions, f1, f2 and so on up to fn okay. So, let us now concentrate in the next couple of lectures, on solving the equations in single variable.

(Refer Slide Time: 04:11)



And the methodology that we are going to adopt, the general strategy that we are going to adopt is as follows okay. So when we have this particular curve, now keep in mind that, this curve how the curve looks like and so on, is typically not known to us a prior. This is a reasonably simple example, that we can actually plot the curve, and we can know how the curve looks like, but lot of times we may not be able to know how the curve looks like.

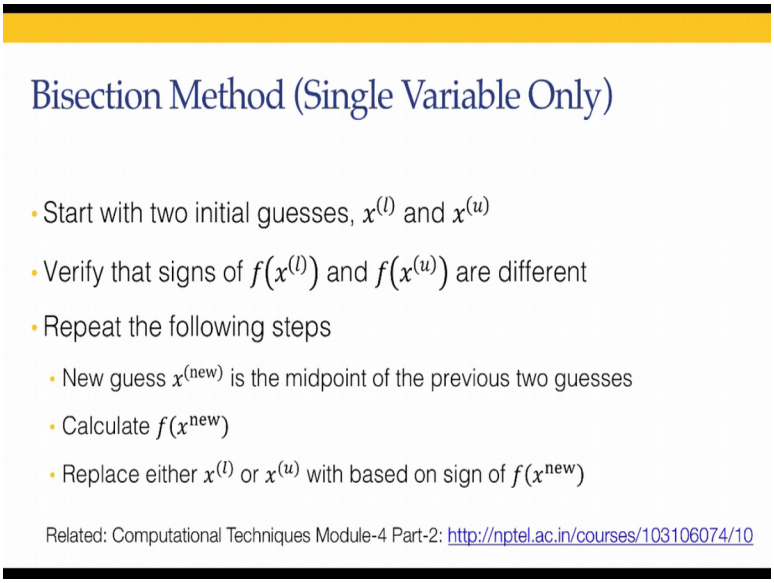
In that case, we are going to start with certain initial guess which is shown by this orange color dot over here okay. The choice of an initial guess is actually very important in nonlinear equations, something that has been covered in the computational techniques lecture series, which I mentioned a couple of slides earlier. You can refer to any standard textbook, in mathematics or in numerical methods. And you will be able, to know, what, how the initial guess has a profound impact on, how the solution evolves.

So we start with an initial guess, and this is going to be an iterative solution technique, just like the Gauss Siedel or the Jacobi iterations where, in module 3, excuse me, in module 4 okay. So we start with certain initial guess and we use a chosen method, chosen strategy, to move in the direction of the solution. So that direction of the solution, is shown by the brown arrow and the orange dot moves from its initial position to a new position okay.

Now, we actually check whether some kind of stopping criteria is satisfied or not. What start stopping criteria could be, is whether or not that change in x, is sufficiently large and that is the stopping criteria based on x, or whether f (x) is sufficiently close to 0. So the stopping criteria, is either going to be from a graphical view point, whether this arrow, this brown color arrow is short enough, so that we can say, that we have reached close enough the solution or if this vertical distance is, short enough, so that we can say that, we are reasonably accurate, when it comes to solving the equation $f(x) = 0$.

Clearly in this particular geometry case we have not reached our desired solution. And, therefore we need to repeatedly solve these 2 steps over and over again. We use the same method to choose, to move further in the direction of the solution. Again, verify whether the stopping criteria is met. If the stopping criteria is met, we have our solution, if not we keep repeating these 2 steps over and over again okay. So, this is the methodology that we are going to implement in an example that I am going to cover next.

(Refer Slide Time: 07:01)



And that example, is going to be bisection method, in a single variable okay. So, this bisection method, is what is known as a, bracketing method, that means we do not start with a single solution. Unlike this figure, we are not going to start with a single solution, but we are going to start with 2 solutions. One of them is going to be, on one side of our actual solution, and the other guess is going to be on the other side of the actual solution.

Which means that, our f(guess 1) and f(guess 2) are going to have opposite signs okay. So, start with 2 initial guesses xl and xu, x lower bound and x upper bound. Verify that signs of xl and xu are different. So, in this case, let us say our fl was going to be equal to 1, and our xl was going to be equal to 1 and xu was going to be equal to 4 okay. Then, first we verify, whether f (1) is of a different sign than f (4). If it is of a different sign, then those are permissible initial guesses okay.

And then, we choose the next guess, as nothing but the midpoint of the previous two guesses. So our xnew is going to be nothing but x (l) +x (u) / 2 okay. Then we calculate our f (xnew), and depending on where the sign i is, we are going to either retain xl or xu. So, for example, our guesses are over here and over here are the initial guesses. The next is going to be, (4+1) / 2 which is 2.5. So, let us say that 2.5 is over here which is on the same side as xl. So we drop xl and xnew becomes our new xl okay.

We continue that, and let us say our x, xnew again comes on the other side of this dot, which means on the same side of x(u). In that case, we will drop xu instead of xl. And we keep doing that over and over again, in an iterative fashion, and that becomes our bisection method okay. The link, to the computational techniques module4, part2, is given over here. You can go to that link, to get and to understand, some of the theoretical aspects behind the bisection method okay.

So, let us head on over to MATLAB and try to solve the equation 2-x+ln(x) = 0, using bisection rule. Our initial guesses are going to be, x (l) = 1 and x (u) = 4. So, let us go on to MATLAB to solve this okay. (Video Starts: 09:51) edit bisecRule okay. Yes, open a new function, script file to use bisection method to solve 2-x+log (x) = 0 okay.

Initial guesses, xl =1, xu=4, fl is going to be equal to 2-xl+log(xl) and fu is going to be equal to 2-xu+log(xu). Check if signs, of fl and fu are different okay. So how do we check, whether the signs are different or not, we will just multiply fl and fu, if the signs are same the product fl multiplied by fu is going to be positive? So, if both fl and fu are positive, the sign is, of the product is going to be positive.

If both fl and fu are negative, again the sign is going to be positive. So therefore, we are going to multiply fl and fu, and check whether it is less than 0. If it is less than 0, we are good. If it is greater than 0, that means fl multiplied by fu is positive. Then we are not going to be, we know that fl multiplied by fu is positive, that means fl and fu have the same sign, so okay.

So, if fl * fu greater than 0, then error, initial guesses should have different signs. And fl multiplied by fu is actually going to be less than 0. Then we are fine. So we do not need an else condition over here. Iterative solution using bisection okay. So, let us look at, so let us say error, is going to be equal to nothing but abs (xl-xu). That is going to be our error, and let us say, we are now going to have our next solution. xnew is going to be (xl + xu) / 2 right. So, (xl + xu) / 2. So that is going to be our xnew okay.

What we want to check is, whether f (xnew) that is fnew whether that has the same sign or not. So fnew, is going to be 2-xnew+log (xnew) okay. So, let us check, the sign of fnew multiplied by fl. If fl multiplied by fnew is going to be greater than 0, which means fl and fnew have the same sign, then xl = xnew and fl = fnew, else xu is going to be equal to xnew and fu = fnew and so this is one single iteration.

We need to iterate over this, but let us go on and run one single iteration and see whether that works or not okay. So let us go to MATLAB and say bisecRule. Let us hope, that we do not get an error. Let us press enter okay. So, it works for now okay. So now our xu is, so let us check what our xl is. xl is 2.5, xu is 4.0, our fl now is 0.4163 and our fu remains the same as negative 0.6137 okay. So, what has happened in this particular case is, we have run one single iteration okay of our bisection rule and in that single iteration of the bisection rule. (Video Ends: 15:02)

Now let us go on to power and see what has actually happened. So, we have moved from, this particular x=1 to a mid which is located somewhere over here okay. (Video Starts: 15:15) So now if you check the value of, f (l) or f (u), they are still sufficiently far away from 0. Or if you check the distance between xu and xl, we still have a very large distance between xu and xl. So therefore we conclude that our iteration has not converged.

So, let us go on and run this for one more iteration. So, I will just go on into this section, right click and I will say, evaluate current section. We can do this sectioning, because we have used a double percentage sign over here okay. If we do not use, that sectioning, it is not possible we can then instead, highlight this entire passage. Right click on this, and say evaluate current, sorry evaluate selection. So, the selected part, will be evaluated again, or we can just go over here and say evaluate this section.

So, evaluate current section, which means only these commands are going to be reevaluated, and let us see what our xl and xu are. Now xl remains 2.5, our xu has changed from 4 to 3.25, now 3.25 is the midpoint between 4 and 2.5 okay. And our fl, if we check, will remains the same as before 0.4163 and if we check fu, now fu would have changed and if fu has become -0.0713.

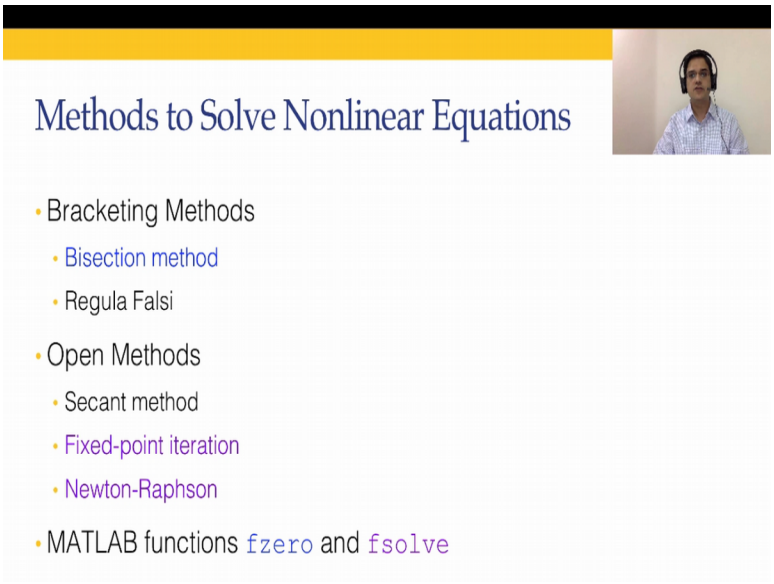
So, let us go to MATLAB and do this in a loop. So, for now, I will just do this in a for loop okay. So, for i=1: maxIter, where maxIter basically means max iteration. So, let us say maxIter equal to say 50. Let us run this for 50 iterations okay. Let us select this, right click and say smart indent. Smart indent means. The entire set of code will get beautified and we will go at the end and press end okay. So, we now have created this in a loop and in a for loop and we want to check the errors okay. So save this and error, keep in mind is nothing but abs value of xl-xu okay. And let us run this. Let us see if we get an errors okay.

There are no errors okay. And what do we have over here is, err, which is the difference between the xl and xu is 10 to the power of -15. So that is pretty small number. So that means, we should be extremely satisfied with our solution, and if we check our fnew, our fnew is also 1*10 to the power-15. So, that means we are now very close, to the solution fx = 0 as well okay. So we actually did not need 50 iterations. So let us go back and reduce the number of iterations to 25 and see how this works. So save this and run it.

Let us go over here, and check our error. err is nothing but difference between xl and xu which will give us, a tolerance in our solution, that we want in the solution x that we want and that error is 10 to the power-7 and fnew which is trying to see how close our solution is to 0, is 10 to the power -8. So, I think with this also it is a sufficient amount of convergence. (Video Ends: 19:07) So let us go back to our PowerPoint and see what we have done. So we have solved this using bisection method right. Now we have kind of solved it in a naïve way, not in a very polished way. We will change this in the next lecture okay.

However, what we have done is, we demonstrate the core principle of any non-linear equation solving, that we start with an initial guess or a set of initial guesses. And we use a chosen strategy, to the move in the direction of the solution. In this case, that the case of bisection method, the movement in the direction of the solution, was taking the average between the 2 guesses xl and xu. And checking whether the stopping criteria is satisfied. The last part, we have not yet done, that is something that we would do in the next lecture okay.

(Refer Slide Time: 19:59)



Methods that to solve nonlinear equations, that were covered in computational techniques module 4. Where, Bisection and Regula Falsi methods were bracketing methods, Secant, Fixed Iteration and Newton Raphson were the open methods. We are not going to cover, all the methods. We do not intend to be exhaustive in covering all the methods. But we want to cover the most important of the methods. The most important bracketing methods in my opinion is the bisection method that is also the method that is used partly that is used in the MATLAB function fzero, and then we have open methods which were fixed iteration and Newton Raphson method.

Fixed iteration, is kind of similar to the Gauss Seidel method that we saw in the previous module. Newton Raphson method is one of the most popular methods to solve nonlinear equations, and a related MATLAB method is known as fsolve, which uses a very different sort of algorithm, in order to solve the equations. We will not go into the algorithm for fsolve, but we are going to primarily use MATLAB functions fzero and fsolve as well. So that is going to be our strategy in the next 5 or 6 lectures of this module.

We are going to talk about Fixed Point Iteration and Newton Raphson method in single variable. We are going to talk about how to use fzero method in order to get solution. We are then going to talk about multivariate Newton Raphson method and multivariate Fixed Iteration method and finally end with using fsolve. In order to solve a sequence of nonlinear equations, n nonlinear equations and n unknowns and that is going to be the strategy for module 5. So with that I come to the end of the lecture 5.1. I will see you in the next lecture.