MATLAB Programming for Numerical Computation

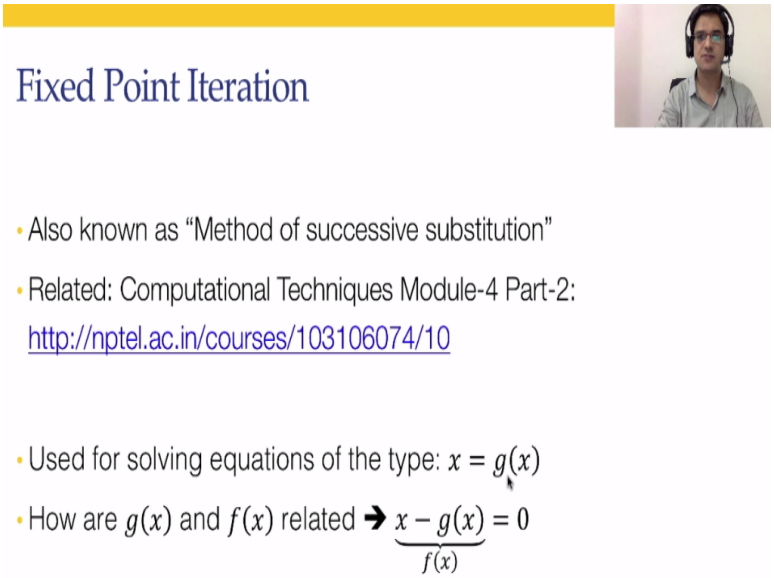
Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #05
Lecture No. #5.3
Non Linear Algebraic Equations – Fixed point Iteration (Single Variable)

Hello and welcome to MATLAB programming for numerical computations. We are in week number 5 in this week. We are covering methods for solving nonlinear algebraic equations of the form f(x) = 0. In lecture 5.1 we covered introduction to nonlinear algebraic equation solving techniques as well as covered 1 technique known as bisection method.

In lecture 5.2 we covered MATLAB function called fzero that can be used to solve the equation in single variable. In today's lecture we are only going to cover single variable fixed point iteration method.

(Refer Slide Time: 00:49)

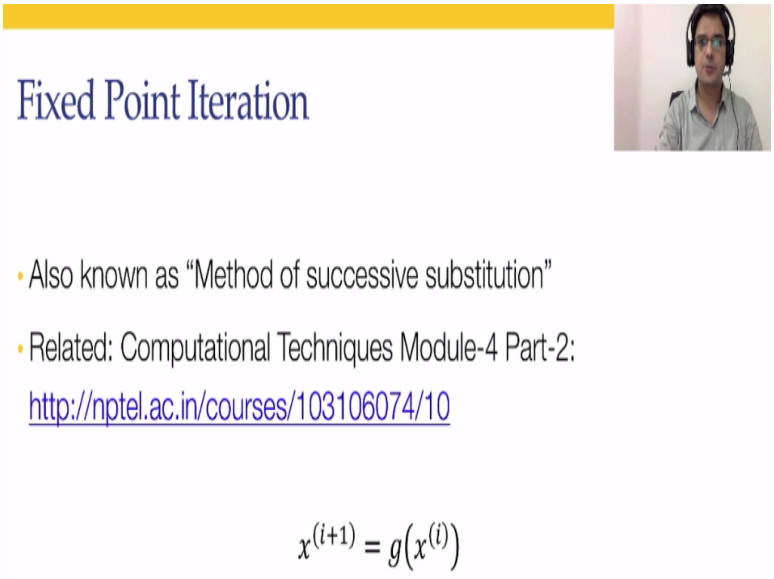


So the fixed point iteration is also known as method of successive substitution the reason for this will be clear in a couple of minutes from now. Related module from computational techniques course is module number 4, lecture 2 the link for which is given below. That is the 10th lecture in the computational techniques course.

The fixed point iteration instead of solving the equation of the form f(x) = 0. It instead solves an equation of the form x = g(x). Now question is how g (x) and f(x) are related okay. So if we were rewrite this equation as x - g(x) = 0 or g(x) - x = 0. We would have recast this equation as the equation f(x) = 0.

We should note that g(x) is not a unique representation of the overall function okay. So there are multiple ways in which we can convert f(x) into g(x) okay.

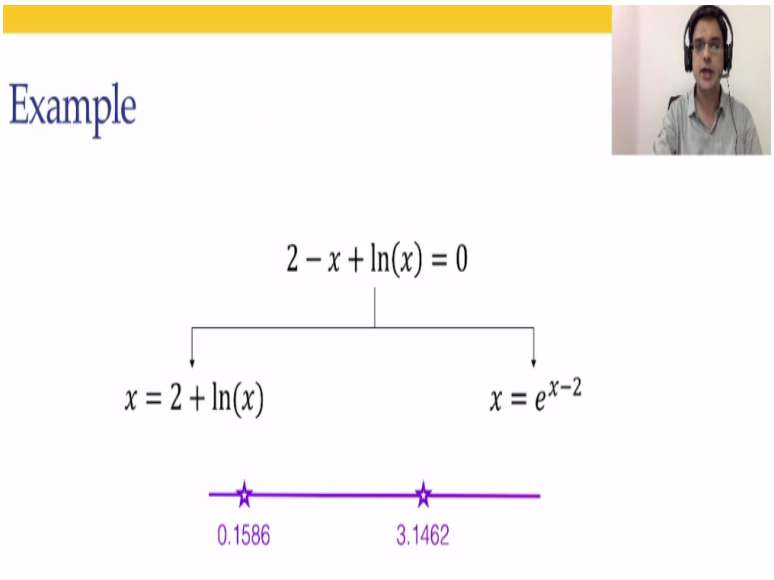
(Refer Slide Time: 01:59)



So looking back at the equation that we want to solve. The equation that we want to solve is $x^{i+1} = g(x^i)$. So what it means is that we start with initial guess x0 okay. And in this equation and in this original equation we just substitute the value of x0. The result that we get we assigned that as x1 okay. Next again x1, we substitute in our function g(x) the result that we get, we assign as x2 so on so forth.

What we are actually doing is, we are successively substituting x(i) into the equation and getting x(i+ 1) and therefore it is also known as method of successive substitution. Now this method is not guaranteed to converge under all conditions and when it converges and when it does not converge has been also studied quite extensively.

(Refer Slide Time: 03:00)



What we will do in today's lecture is this, we will take up the example that we have been doing for the last couple of times 2-x+ ln(x) convert it into the form x = g(x) and see what solutions we are going to get when we solve this in MATLAB. The 2 ways I have shown over here are you just take this x onto the right hand side and you will get x = 2+ ln(x).

Other option is take these guys to the right hand side and get an exponent that will result in ln (x)= (x-2) or x = e ^ x-2. So in case 1, we have g(x) = 2+ ln (x), in case 2, we have g(x) = e ^ x-2. What we will do is, try to find out which equation for g(x) leads to which solution.

We have seen previously that there are 2 solutions to this problem. One solution is 0.1586 and the other solution is 3.1462 okay. Depending on whether we start to the left of the solution, whether we start between these 2 solutions or whether we start to the right of the solution. We are going to get different results using the fixed point Iteration and that is something that we are going to study in today's lecture okay.

So when it comes to the fixed point iteration, what we are going to do next is, to look at solving x = g(x) for this formulation as well as for this formulation for different initial conditions okay. Let us head over to MATLAB and write our codes to do the same. (Video Starts: 04:47) Edit fixPtIter. So that is a new file that I will create to solve nonlinear equations using fixed point iteration.

Case 1: g(x) is 2+ ln(x), Case 2 g(x) is e ^ x-2 okay. So Let us say initial conditions and initial condition, let us say will start between the 2 solutions. Initial condition is x not equal to 1. We are going to do for several iterations. So Let us say maximum number of iterations let's just put it as 50 okay. And Let us now start of i = 1 to maxIter okay.

So okay, what our fixed point iteration was is, just going to be x = g(x), g(x) was 2+ log(x) okay. And that should be fine. Let us go on to MATLAB and just do help log to just ensure that log is indeed natural logarithm and not log to the base of 10 okay. Yes that indeed what it is so log is for natural logarithm, if you want logarithm to the base 10 which is not what we want in this problem but in another problem.

Let us say if we wanted logarithm to the base 10. For that the MATLAB command is log10. And log is the MATLAB command for natural logarithm. Let us clean this screen and let us go on back to our code okay. So we also may want to compute our error. So I will just comment this for now. Let us say err = abs(x-xold). And xold, we will comment this also, xold = x.

So whenever we run this stimulation once we have computed the error, then the current value of x we have just now we just code in xold. And then we go back to the iterations okay. And Let us say computation using fixed point iteration okay. Now what I expect? I expect there to be an error. Let us see whether you can look at the code and figure out what that error is going to be.

Anyway I will held on to MATLAB and try to solve this problem. And we will see that we are going to get an error over here. And I will go through, why we got that error and how to fix it. So fixPtIter, enter yeah. So we are getting an error which is undefined function or variable x error in line number 11.

So let us click on this line number 11. What does the error say? The error says that when MATLAB reaches line number 11, it does not understand the variable x. What does that mean is that the variable x is not in its workspace. So let us go to line number 11 and see what is happening okay. So that does not seem to be any problem with 11.

So let us see what the problem is, we have defined x0 we have defined maxIter but at this point we have not defined our variable x that appears on the right hand side over here okay. So what do we mean by an initial guess? Initial guess basically means that before the first iteration starts, our variable x should get the value of x0. So actually I should be putting it at the computation part x = x0.

So x in bracket 0 is x0, x1 is g (x0) which is what happens when i = 1, x2 is g (x1) which is what happen when i = 2 so on and so forth. So this is the change that we need to make. Save this and run it and we should be able to get the solution. So fixed point iteration when we click that, we get the solution x this 3.1462.

Let us now calculate what the errors and old values are. So let us uncomment this okay. Again the same error we are likely to get over here. If we do not change something xold at this point is not going to be defined. So what I will do over here is, I will also define xold as x0 okay. And save this, run this and when it is run, the error has actually gone to 0.

We may not want to keep solving using fixed point iteration till error goes below the machine precision. We need to decide a priori what our error tolerances are. So the solution is 3.1462. We want the solution only up to let us say the fourth decimal place. So therefore we can say that the error tolerance should be 1*10 ^ -4. And that should be sufficient for us.

So what we will do over here is, say tolX tolerance in x = 1e-4 and we compare whether the error is less than or greater than the tolerance. So if err < tolX then come out of the loop using the command break, end okay. This is exactly what we have done in our earlier lecture as well. And now let us save this. Let us add a command that will clear all the variable.

Clear all and then let us run this and see what happens okay. And we have run this with an initial condition of x0=1. Our solution is 3.1462, our tolerance was 1e-4 and the error at which we stopped was 3.6e-5. The 11 iterations in order to reach the error below our error threshold. So

that is all we have at this point of time. So that is a fair amount of information that we are actually able to generate through this 1 code okay.

So now let us change from x0 = 1 to x0 = 0.1 okay. And when we change to x0 = 0.1, what is going to happen is log (0.1), will result in x as a negative number and we will eventually get our x value as an imaginary or rather as a complex number. So let us run it and see what we actually get okay. So as we had said that we get our x value as a complex number.

Let us type out x. x is a complex number and the reason why that happens is when we start of with x0, so let us say we were to just copy this and paste it in the MATLAB screen and replace it with x0. Our x value is -0.3026. Now if we were to take log(x) that is going to result in an imaginary or rather in a complex number. And therefore at this stage we should not be continuing this overall method because we are not guaranteed to reach the solution that we desire.

If we were to desire if we were desiring a real number solution okay. So again in this case it is we are we were lucky that we were able to reach the solution 3.1462 and our imaginary part of that complex number was 0. That may not always happen. So if you were expecting a real valued solution and you get a complex number that is also a sign that you should break the loop that is solving it.

However we are not going to change this code for now. Let us now try for a different initial condition. Let us take an initial condition of x0= 4 and run this simulation. When we run this with x0= 4 within 9 iterations we have again converge to 3.1462. Let us take another x0 like say x0= 40 and see what happens. If we have x0= 40, this takes 11 iterations in order to reach again 3.1462. (Video Ends: 14:22)

So what we see over here is that if we were to use this particular g(x), we are going to reach no matter where we start on this line. We are going to reach the second solution if we start of too close to 0, we will actually get a complex valued result because we end up with logarithm of a negative number during our computation okay.

So that is the first part of our solution technique. The second part is if we were to solve this for x=e ^ x-2, so let us go back (Video Starts: 15:00) to MATLAB and change this to e ^ x-2 okay. And for us to be able to do that the only place this remains same our initial conditions maxIter remains same, tolX remains same, this part also remain same.

The only thing that changes is this particular line and this will change to exp x-2, e ^ x-2 okay. And rest everything is also going to remain the same. Let us start with low enough initial condition. Let us start with an initial condition of 0.1, which is to the left of the first solution. Let us save this and run this code. When we run this code, the code runs fixed point iteration has converged within 5 iterations okay.

And the solution is x = 0.1586. Let us now start between the first and the second solution. Let us say x0=1. Let us solve this run and it is converge in 7 iterations. And we have reached a converts converge solution of 0.1586 again okay. Let us say we were to start at 3. 1 which is very close to the second solution. Let us run this and see what happens.

We have taken 11 iterations, starting with an initial condition of 3 .1. Again we have reached the solution x = 0.1586. (Video Ends: 16:42) So as long as we are starting to the left of the solution we are converging to the first solution. Let us see what happens if we start to the right of the solution. (Video Starts: 16:52) If the initial guess was to the right of 0.315, let us say what we get.

So let us start with an initial solution of 4, save and run this okay. And when we run this, our x is going to be infinity okay. Even in 50 iterations our fixPtIter has not converged it has indeed diverged so okay. So Let us again look at what this is x = e ^ x-2. So Let us put it over here and if x was e ^ x0 - 2. Let us call this as x1 okay. x1 was 7.39.

x2 we will now calculates x2 = g (x1). So we have put that press enter and from 7.34 we have gone to 219 okay. Let us say x3 was x2 e ^ x2-2 and we have got 1 .74 *10 ^ 94. So as you can see the solution has very quickly diverged. (Video Ends: 18:09) So what we have observed over

here is, if you start with g, if you take g(x) = 2+ ln(x), no matter where on this line you start you reach solution number 2 that is 3. 1462.

If you take g(x) = e ^ x-2. If you start between 0 and the second solution you will converge to the first solution. If you start beyond the second solution our fixPtIter scheme is diverging okay. So that is all I wanted to cover in this lecture today okay. With that we come to the end of lecture 5. 3.

In lecture 5.4, we going to cover a new method known as Newton Raphson method which is probably the one of the most popular methods for solving nonlinear algebraic equations. So thank you for listening to lecture 5.3 and I will see you in the next lecture bye.