MATLAB Programming for Numerical Computation

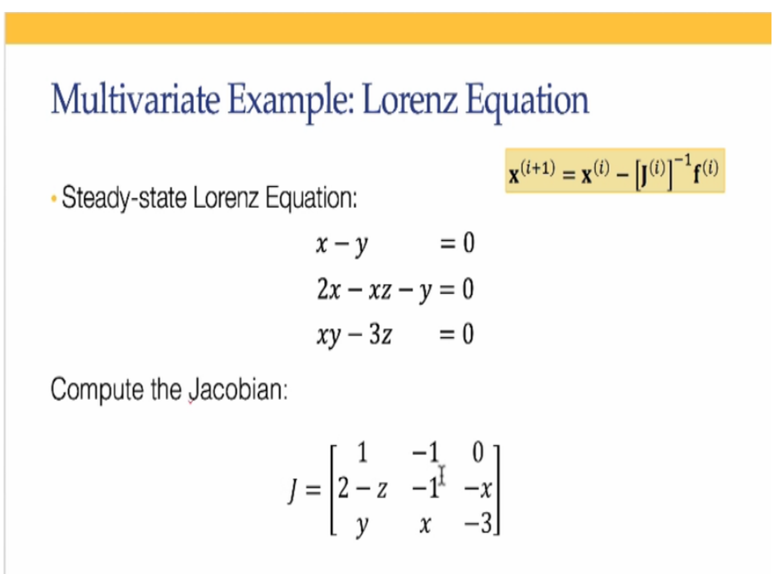
Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #05
Lecture No. #5.6
Non-Linear Algebraic Equations - Multivariable Newton Raphson

Hello and welcome to MATLAB programming for numerical computations. We are in module 5 where, we are covering methods to solve nonlinear algebraic equations. This is the last lecture of this module lecture 5.6. In this lecture, we are going to consider multivariable Newton Raphson method. In previous lecture 5.5, we have covered fsolve, which is a MATLAB function in order to solve, multiple nonlinear equations in multiple unknowns.

So, when we have a system of n nonlinear algebraic equations in, n unknowns, we can use fsolve in order to solve that. In lecture 5.4, we had covered a single variable Newton Raphson method. In today's lecture, we are going to extend the single variable Newton Raphson method covered in lecture 5.4 to the multivariable case and we are going to solve the Lorenz equation problem that we saw in the previous lecture, lecture 5.5 okay.

(Refer Slide Time: 01:18)



So, let us look at the steady state Lorenz equation. The steady state Lorenz equation, is of the form $x-y = 0$, $2x-xz-y = 0$ and $xy-3z = 0$ okay. Now in computation, of Newton Raphson's method, for a multivariable case, we need to compute a Jacobian. The Jacobian as you might be aware is computed in the following form okay.

The 1, 1 element of the Jacobian is going to be partial f1 by partial x, df1 / dx. So df1 is f1 is this guy okay, so when we take a partial derivate, with respect to x, we get that value equal to 1 okay. The next guy, is going to be partial of f1 with respect to y. So, that is going to be -1 and third is going to be, df1 / dx. Now let us look at the second row.

The second row is going to be, the first element of the second row is going to be, df2 by dx in that is going to be 2-z. The second guy, is going to be df2 / dy which is -1, the third guy is going to be, df2 / dz, which is going to be -x.

So, d / dz, of this term and this term is going to disappear, because there is no z dependence over there. And only this term is going to remain and because we are taking partial derivatives with respect to z, x is held constant and we get -x * 1, which is -x as the third guy. Likewise, if we look at the last equation df3 / dx is going to be y, df3 / dy is going to be x and df3 by dz is going to be -3 okay.

Recall that Newton Raphson's method in a single variable. Let us insert equation over here, Newton Raphson method in single variable, was written as xi+ 1 okay. So, the Newton Raphson method in single variable was xi+ 1 = xi - fi / ji okay. Where, j in that case was nothing but f dash. In case of multivariable case, we are going to just convert all of these into vectors okay. And because we cannot divide by j, what we need to do instead is take an inverse okay.

So, this is going to be, ji inverse of that multiplied by fi. And this is our multivariable, Newton Raphson's okay. So this is for multivariable Newton Raphson. We want to solve where, fxyz is nothing but this guy and Jacobi j is nothing but this guy okay. So let us head on over to MATLAB and try to solve this problem, using multivariable Newton Raphson okay.

(Video Starts: 05:59) So, let us look at, what we had written earlier, a function called lorenzSystem which was used with fsolve .Let us save that as a different file name. Let us call that lorenzsysnr. nr standing for Newton Raphson okay. And we need to use the same function name, as the file name. So, we will just replace the, function name with the file name. This particular file needs to calculate both fvalue as well as the Jacobian. So, let us call the Jacobian, as jac and let us define Jacobian okay.

And let us look at what we had over here. 1, -1 and 0. So j is going to be 1,-1, 0, the second row is 2-z, -1, -x, and third guy is y, x, -3 okay. And this is all that we need at this stage okay. So keep in mind that, we were able to write it in exactly same fashion, as we had returned it over here, mainly because we were we had written the variables x, y and z = x1, x2 and x3 right in the beginning.

Sometimes, we will not be able to do that, in which case we will have to write this as 2-x3-1. And then this is going to be -x1. Likewise, y is nothing but x2 and we can write this as x2 and this is going to be x1 okay.

Either ways is going to be exactly fine, they are both exactly the same thing okay. You need to be careful, about that the variables that you have used. For simplicity, I will just undo my edits that I have done a few minutes back okay. And we should be good to go okay. One thing you will notice, I will just actually try to do this right. Now if I call this, and are using our variable some x okay.

So, let us call this, Lorenz. Let us call f, j = lorenzSysnr say 1, 1, 1. We are going to get some problem, and the problem, is the output arguments jac is not assigned. Now let us see, why we get that particular error. We get this error because, instead of calling the Jacobian jac, I have called it capital j over here. So I just need to change this and there will not be an error okay.

Other thing to keep in mind is that MATLAB. I will just undo this and show you now. MATLAB, is fairly helpful when it comes to pointing out some errors or warnings. So if you see over here, what was the green button has changed now to orange button over here. And it says that, now if you click over here, it will go to the location where it is giving you a warning. If you take your mouse and hover at that point, you will be able to see the warning.

The warning says the function return value jac might be unset. What that means, is that MATLAB is trying to tell you that, in this function lorenzSysnr. I am expecting jac as my output variable. However, in the entire function, I have not specified what that jac is. So I should go here and change my j to jac. When I do that you notice that the orange bullet has turned into green which says that no warnings are found okay. So, let us save this okay.

Now let us say, we want to go to Newton Raphson's and we want to do a multivariable Newton Raphson. So let us go over here and save as multivariablenr for Newton Raphson okay. So let us say, solve not yes, solve a sequence of nonlinear equations using nr, Newton Raphson okay.

Now, our initial conditions were capital X = 1, 1, 1 okay. Keep in mind, what I am doing over here is, I have taken a previous solve problem and I am editing that, rather than writing this entire multivariable. Newton Raphson from scratch, maximum iteration was 50 and tolx to keep the same as, what we saw in fsolve in the previous lecture. We will take the tolx as 1e-6 okay.

Computation using, not fixed point iteration, Newton Raphson okay. Our initial X, we should call this X0, our initial X is X0, and our Xold is also X0, so okay. Now over here f, j is = lorenzSysnr (x) okay. So previously what we had was, we calculated f using 2- x+ log(x) instead of that instead of our fx equal being equal 2-x+ log(x). Our f(x) is x-y and so on and so forth. Our f(x) was x-y, 2x-xz-y and xy-3z. So, we replaced this particular guy with this lorenzSysnr and lorenzSysnr in addition to giving f also gives Jacobian okay.

Our new value of X is going to be, our old value of X-jacobian inverse. So inv j multiplied by f okay. Error is going to be X-Xold okay and error, we want all the rules ith column should be this okay. And our capital xold = x and if error is less than, if all the errors, are less than tolerance x, then end this computation okay. So, that is what we have, let us clear all and okay. Let us solve this multivariable nr okay.

Solve multivariable nr and we get the solution x as 1 .7321, 1. 7321 and 1.0 and we have got this x in only 5 iterations okay. Now let us change, that initial guess to -1, -1 and 1 and let see what happens when we run this. So, when we run this, we again got the solution and solution again we reached in 5 iterations okay. And the solution is -root 3, -root 3 and 1.0.

Let us change this to -1 -1 and 0. And let us see with that what we get our solution and we run this okay. We get our x as 10 ^ -21. So that is really close to 0. This is the overall code that we have used in order to solve the system of 3 nonlinear equations and 3 unknowns using multivariable Newton Raphson's.

We started off with the single variable, with Newton Raphson code and the changes that we made were primarily at 2 locations. The first change, that we made were, instead of calculating x= x sorry, 2-x+ ln(x) we used our function lorenzSysnr in order to calculate our f and our Jacobian. And instead of using x = x - f / df instead of that, inverted Jacobian and pre- multiplied our function f, with that Jacobian okay. That was the other change that we made.

The third change that we made was that error in any ith iteration is no longer a single value, but it is a column vector okay. So, we found that column vector, as the difference between x and xold and checked whether all the 3 elements of that column vector, whether or not they are less than that x tolerance. And if they were, then we stopped the iteration, if not we kept continuing the iteration, till the stopping criteria was met okay. (Video Ends: 16:34)

So, with that we basically come to the end of, not only this lecture but to end of module number 5 okay. And thank you, for listening. In the next module, number 6, we are going to cover regression, interpolation and curve fitting. Thank you and see you next week.