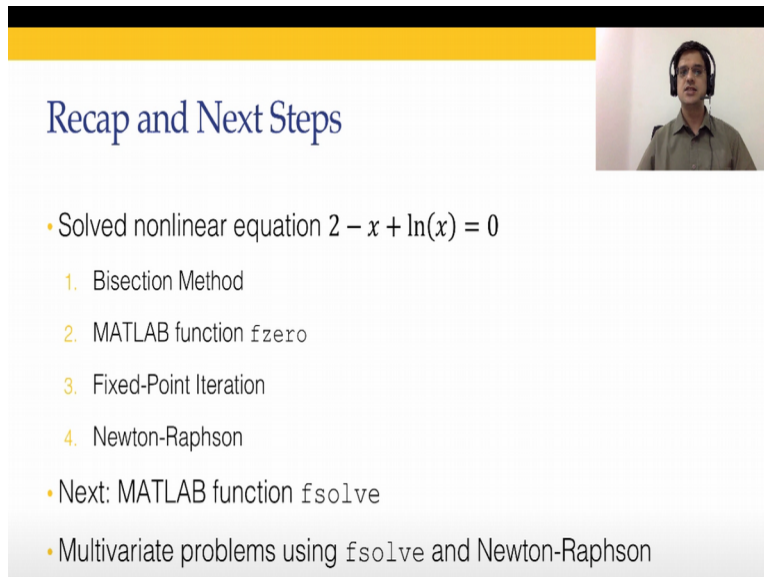


MATLAB Programming for Numerical Computation
Dr Niket Kaisare
Department of Chemical Engineering
Indian Institute of Technology, Madras

Module No. #05
Lecture No. #5.5
Non-Linear Algebraic Equations - Using MATLAB function fsolve

Hello and welcome to this course on MATLAB programming for numerical computations. We are in module 5, covering methods to solve nonlinear algebraic equations. In this lecture, that is lecture 5.5. I am primarily going to cover, how we can use a MATLAB function fsolve. Previously, we have used another MATLAB function fzero. fzero was able to solve an equation, in a single variable. fsolve, on the other hand, is capable of solving, multiple nonlinear algebraic equations simultaneously.

(Refer Slide Time: 00:49)

A presentation slide titled "Recap and Next Steps" with a yellow header bar. It contains a bulleted list of topics covered and upcoming. A small video inset of the lecturer is in the top right corner.

Recap and Next Steps

- Solved nonlinear equation $2 - x + \ln(x) = 0$
 1. Bisection Method
 2. MATLAB function `fzero`
 3. Fixed-Point Iteration
 4. Newton-Raphson
- Next: MATLAB function `fsolve`
- Multivariate problems using `fsolve` and Newton-Raphson

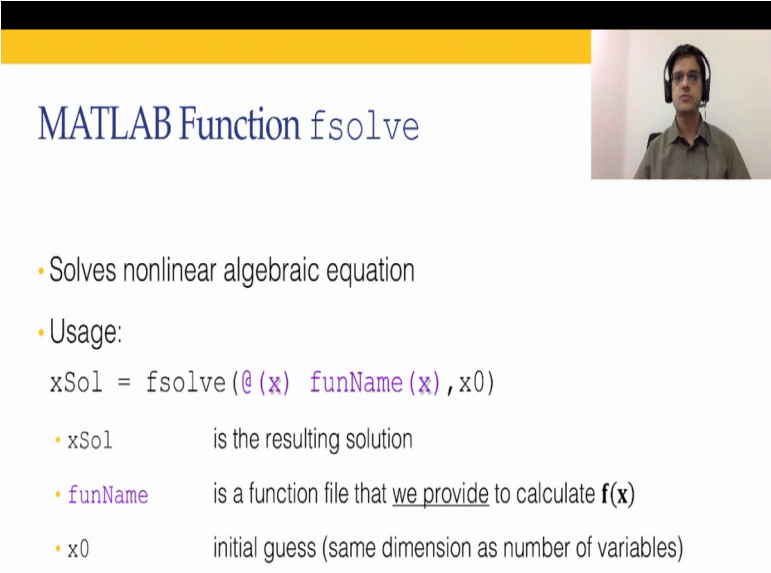
Before we move on to the contents of today's lecture, let us just recap what we have done so far in module 5. What we have done is, if we have taken an example of $f(x) = 2 - x + \ln(x) = 0$ and solved this single variable equation, to obtain the value of x , that gives the solution. In the first lecture, we covered a method known as bisection method, we wrote our own code in order to solve this equation and find a solution.

In the second lecture, we showed how we can use the MATLAB function `fzero`. In the next 2 lectures, we covered Fixed Point Iteration and Newton Raphson method. Both of these, can be extended to multiple variables. However, in lecture 3 and lecture 4, we covered Fixed Point iteration and Newton Raphson method for the same example $2-x+\ln(x)$. We used this example to also demonstrate, some of the properties of the numerical methods that we have covered in the 4 lectures.

What comes next, is the following; in this lecture, we are going to look at, how we can use MATLAB function `fsolve`. First, we are going to use `fsolve`, to solve the same equation $2-x+\ln(x) = 0$. In the next lecture, we are going to look at multivariate Newton Raphson method to solve n equations and n unknowns.

So, let us get started with a content of today's lecture where, we are going to use the MATLAB function `fsolve` to solve the single equation, in single variable $2-x+\ln(x)$ and then we are going to extend it, to multiple variable example okay.

(Refer Slide Time: 02:36)



The slide features a title 'MATLAB Function fsolve' in blue text. Below the title, there is a list of bullet points: 'Solves nonlinear algebraic equation', 'Usage:', and a code snippet 'xSol = fsolve(@(x) funName(x), x0)'. Below the code snippet, there are three more bullet points explaining the variables: 'xSol' is the resulting solution, 'funName' is a function file that we provide to calculate $f(x)$, and 'x0' is the initial guess (same dimension as number of variables). In the top right corner, there is a small video inset showing a man wearing a headset.

MATLAB Function `fsolve`

- Solves nonlinear algebraic equation
- Usage:
`xSol = fsolve(@(x) funName(x), x0)`
- `xSol` is the resulting solution
- `funName` is a function file that we provide to calculate $f(x)$
- `x0` initial guess (same dimension as number of variables)

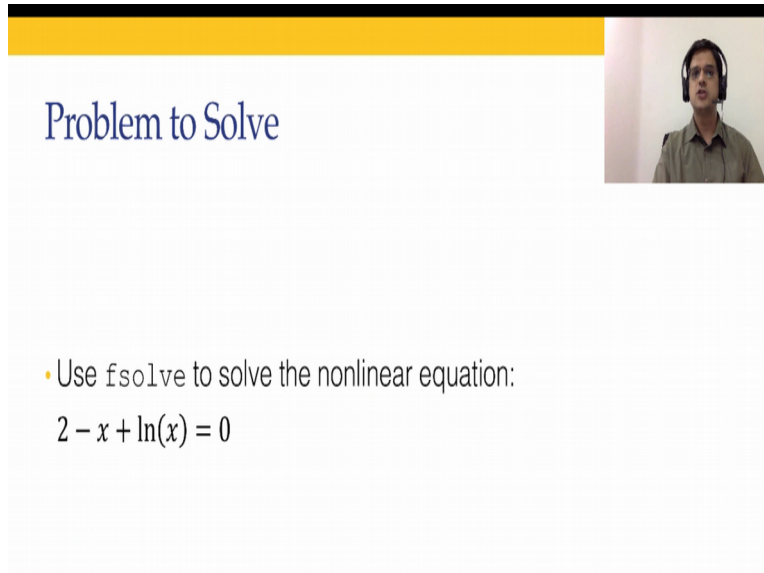
So how to use the function `fsolve`? The syntax of function `fsolve` is very similar to the function `fzero`. So, in this case, our variable x is going to be a vector okay. As we have discussed earlier x is going to be a column vector. So it has n rows and a single column. So, it is an $n/1$ vector. The solution is going to be, `xSol` okay. And that is also going to be $n/1$ vector and the function

funName is the function file that will calculate a vector valued function f as the function of the vector x .

Again, because we are interested in solving n equations in n unknowns, f is going to return or the function fun Name is going to return n values and it will be returned as a column vector okay.

And we need to provide initial guess, and the initial guess has to be of the same dimension as number of variables. Unlike `fzero`, we do not have to provide initial guesses that bracket the solution. We only have to provide, the initial guess, that we feel is the best guess, for the solution that we are expecting okay.

(Refer Slide Time: 03:59)



The video player shows a slide with a yellow header bar. The title 'Problem to Solve' is in blue. Below it, a bullet point states: 'Use `fsolve` to solve the nonlinear equation: $2 - x + \ln(x) = 0$ '. A small video inset on the right shows a man with a headset.

So, let us look at an example, we will use, `fsolve` in order to solve the nonlinear equation $2 - x + \ln(x) = 0$. (Video Starts: 04:11) Let me go to MATLAB and so let us look at the function that we have already created. We had already created the function called `fun4bisec` okay. Where, the function value, it calculated the function value, $2 - x + \ln(x)$ okay. And that is what we want to solve so. (Video Ends: 04:39)

As I had said here, our funName is a function that calculates $f(x)$ okay. `fsolve` is the MATLAB function that we will solve $f(x) = 0$. It needs, somehow a function that calculates for it the value $f(x)$ okay. (Video Starts: 05:02) And that is what we are going to provide over here. And we are

not going to modify that function at all okay. And we are just going to use the syntax, as shown over here.

`xSol` = or the solution or the = `fsolve` okay @ `x` that is the variable, that we want to solve for, name of function `4bisec(x)` `fsolve` @ `x` space `fun4bisec` in brackets `x`. That is how we call this particular function and pass it onto `fsolve` and we need an initial guess, so let us say the initial guess in this case, let us take the initial guess as 1 okay. And let us just press enter.

When we give an initial guess as $x = 1$ okay, we land into a problem. (Video Ends: 06:00). The reason why, we get this problem is, let us look at this particular function. Now this function has a maxima at $x = 1$. So, for example, if you differentiate this, as we had seen in the earlier lecture, we get this as $-1 + 1$ by x okay. $-1 + 1$ by x at $x = 1$ is going to be equal to 0.

Which means that particular guy is either going to be a maxima or a minima okay. So, if now, if we take, d^2 square we are going to get -1 by x square which is a negative value with at $x = 1$ which means that it is a maxima. So, this function, as we had seen earlier, goes through maxima at value of 1 okay. (Video Starts: 06:55) So, unlike `fzero`, where we needed to give solutions, that give an initial guess, that brackets the solution here, we need to provide a reasonable value of the initial guess.


The problem with providing the value of initial guess is equal to 1 is that the method such as Newton Raphson or `fsolve`, cannot proceed because the slope calculated at $x = 1$ is equal to 0 okay. So that is one of the problems, that both `fsolve` as well as Newton Raphson method faces. You cannot solve, with start, with an initial guess that is a local minima or a local maxima. What we are going to see next is if we start with an initial guess which is to the left of that maximum we are going to reach the first solution, if we start with an initial guess to the right of the maxima, we are going to reach the second solution okay.

So let us take an initial guess to the left and that initial guess for example is 0.5 okay. And let us solve this. When we solve this, we get the result of MATLAB tells, that the equation is solved and the solution `xSol` = 0.1586. Recall, that that was the solution, the left solution that we obtained in the previous lecture okay. Now let us take an initial guess of $x = 2$ and let us see what happens. When we take an initial guess of $x = 2$, the `xSol` ends up becoming 3.1462. So we get now to the second solution.


Finally, a third set of initial guesses, are going to be to the right of the second solution. So, let us take that value as 5 and see whether the fsolve converges or not. When we take the method 5, again sorry, we take the initial guess as 5 again, we get the result that the equation is solved and the solution reached is the second solution 3.1462 okay. So, with that we come to the end of the first example, which was a single variable example okay. (Video Ends: 09:07)

The single variable example, with that we used in fsolve. We saw that, we were able to use the exact same subroutine or sorry, the exact same function, as we use in f0 in order to return f(x). Once, that particular function was passed onto fsolve, fsolve was able to solve the nonlinear algebraic equation, for most of the initial guesses, but it was not always able to solve that equation okay. One of the cases, when it is unable to solve this equation is, if we start at or if we start close to a local minima or a local maxima as we saw, when we started with $x_0 = 1$ okay.

(Refer Slide Time: 09:55)



Multivariate Example: Lorenz Equation



- Wikipedia: https://en.wikipedia.org/wiki/Lorenz_system
- First example to demonstrate “Chaos”
- Observed by Edward Lorenz for atmospheric convection
- Example problem in this Module: Find steady-state solution:

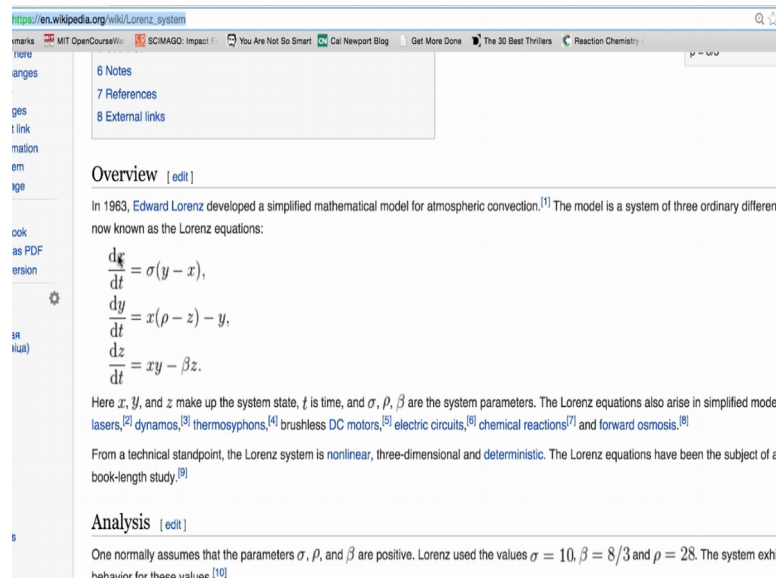
$$\begin{aligned}x - y &= 0 \\2x - xz - y &= 0 \\xy - 3z &= 0\end{aligned}$$

Next, we are going to cover multivariable examples okay. And the example that I am going to cover is a 3-dimensional system. So, we have our solution variable x in the f(x). Our x is going to be a 3-dimensional variable and f is going to be a 3-dimensional vector valued function. For this, we are going, to take an example of, what is known as the Lorenz system.

The Lorenz system, was a first example that demonstrated chaos and it is an extremely interesting example, for people to learn and if you want to look at or try to get some idea about

what Lorenz equation is, you can go to this Wikipedia article over here. And it has a pretty nice description of Lorenz equation is? You can go to this Wikipedia article over here, we are going to get these equations, $\dot{x} - y = 0$; $2\dot{x} - xz - y = 0$; $\dot{xy} - 3z = 0$. So, let's go and try to solve this using MATLAB.

(Refer Slide Time: 10:46)



LorenzSystem okay. I will create a new file function fval= lorenzSystem X okay.

So, we have used, capital X as the inlet input variable over here. Let us define our, define three variables $x=x1$, $y=x2$, $z=x3$, define $f(x)$ okay. So $fval(1, 1)$ equal to, $fval(2, 1)$ equal to, $fval(3, 1)$ equal to again. We had said multiple times earlier, that we are interested in ensuring that, $fval$ is going to be a column vector. That means n number of rows and a single column okay. So we have $x-y$ as the first equation. $2x-xz-y$, $2 * x - x * z$, $-y$ that is the second f and third is $xy-3z$ is at $x * y$, $-3 * z$, okay. Let us save this okay. And let us go to our command prompt okay.

So, now we need an initial guess. So, let us take the initial guess as, 1, 1, 1 okay. And we will use $xsol = fsolve @$ let us say x lorenzSystem x , x not that is the initial guess, and that should give us the solution in $xsol$. I will press enter, equation is solved okay. And the solution is 1.7321, 1.7321 and 1. If you take these equations and try to solve these, you will find that Lorenz equation or

Lorenz system has 3 solutions. 1 solution is $\sqrt{3}, \sqrt{3}, 1$, another solution is $-\sqrt{3}, -\sqrt{3}, 1$ and the third solution is $0, 0, 0$ or the origin okay.

So, let us try, to get the other solution also. x_0 instead of saying is $1, 1, 1$. Let us say x_0 was $-1, -1, 1$ and let us now try to find x_{sol} . And when we do that, we get the other solution $-\sqrt{3}, -\sqrt{3}, 1$ okay. So let us try another initial guess. And in this case, let us say, the initial guess was say $-1, -1, 0$. Let us try with that okay. And x_{sol} . When we try okay, we get the solution, which is approximately equal to 0.

Keep in mind, that we are not getting the solution exactly 0. Why because, we have a stopping criteria, and the stopping criteria in `fsolve` is, that the tolerance value for x is 10^{-6} okay. And that you can, let us say help `fsolve` okay.

There is a command called `optimoptions`. And `optimoptions` is the command that you can use, in order to change the optimization properties, in which case you will need to give the command for solving `fsolve` as shown over here. Where option comes from the vector, options come from, this `optimoptions` okay.

So, let us look at `optimoptions`. Let us say `fsolve` okay. When we type this okay, we get, what are the default options that `fsolve` uses. The default option is, that the function tolerance. That we specify in order to say $f(x) = 0$ is 10^{-6} . The tolerance on x that is our solution variable is also 10^{-6} .

So as long as our solution, differs from the true solution, by less than 10^{-6} , our method is going to converge, going to be set to have converge. In this case, the true solution was 0 okay. And when the solution, was give comes out to be 10^{-7} multiplied by .3, .3, .5. That means, we are closer to 0, than the variable tolerance `tolx` value okay. (Video Ends: 17:45)

So, with that we come to the end of this lecture. What we covered in this lecture were 2 things, single variable `fsolve` and a multivariable `fsolve`. We took the example of Lorenz equations, for the multivariable `fsolve` okay. With that I come to the end of this lecture, and I will see you in the next lecture. Thank you.