Lab #6b -- Some Practice with Nonlinear Equations

Overview

Well, we have almost made it to the end of the semester!!! In this last lab, we will focus on some techniques for solving nonlinear equations. In general, nonlinear problems are much more difficult to solve than linear systems and, in many cases, without a good starting guess, most solutions methods will simply fail to find a solution. Thus, it is important for the user to supply a reasonable starting solution to improve his or her chances of success.

For this lab we will focus on a single exercise involving three nonlinear equations. The system selected for study here is a particularly challenging problem, so the first task will be to attempt to locate some regions of interest where a root may occur. Then, with a set of reasonable guesses, we will use both Matlab's ***fsolve*** function and **Newton's method** to find all the roots of the given nonlinear function.

The lab instructor will walk you through most of the exercise but, along the way, you will have plenty of opportunity to stay engaged by graphically searching for some reasonable starting guesses for the formal nonlinear solution algorithms, by using ***fsolve*** to find a root (once a starting guess has been identified), by computing the Jacobian needed for Newton's method, and finally, by modifying the **nldemo2_lesson6.m** file to use Newton's method to solve the current problem. By the end of the lab, you should have a much better appreciation for the challenge associated with solving nonlinear equations, and have a new set of tools in your Matlab toolbox for addressing this type of problem. With success here, you should have no problem in completing your HW assignment for next week -- since the HW exercise asks you to go through a set of similar steps to solve a much easier 2-equation nonlinear system…

Problem 1 -- A Challenging 3-Equation Nonlinear System

Our goal with this problem is to gain some experience solving nonlinear equations. In particular, consider the following 3×3 system of nonlinear equations:

$$x_1^3 - e^{x_2} + \sinh x_3 = 3.6288188$$

$$x_1^2 x_3 + \left(x_2^2 - x_3\right)^2 = 4.0$$

$$x_1 x_2 x_3 - x_3 + x_1 x_2 = 5.0$$

We will try to solve this system using both Matlab's ***fsolve*** function and by implementing Newton's method.

The first step in finding a solution to a nonlinear system is to identify an appropriate initial guess for the solution algorithm. However, since this 3×3 system is purely arbitrary (does not represent a physical system), we have no idea how many solutions there are (if any) and where they are located. Thus, our first task is to learn a little about the potential solution space.

In particular, for a 3×3 system, one way to help identify the number and location of the solutions is to reduce the vector equations, $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, to a scalar system, $F(x_1,x_2) = 0$, that is only a function of two variables, and then plot $F(x_1,x_2)$ vs. x_1 and x_2 and visualize the locations where $F(x_1,x_2)$ approach zero. We can do this for the above system, as follows:

1. Solve the 3rd equation for x_3 in terms of x_1 and x_2.
2. Define a vector of x_1 and x_2 values and use Matlab's ***meshgrid*** command to create a matrix of $\mathbf{X}_1$ and $\mathbf{X}_2$ values.
3. With the $\mathbf{X}_1$ and $\mathbf{X}_2$ matrices, compute $\mathbf{X}_3$ from Step 1 and obtain a measure of the imbalance in the original equations at each x_1,x_2 grid location by computing

   $$\mathbf{F} = \mathbf{F}_1^2 + \mathbf{F}_2^2$$

   where $\mathbf{F}_1$ and $\mathbf{F}_2$ are the matrix versions of $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ [that is, evaluated with the $\mathbf{X}_1$, $\mathbf{X}_2$, and $\mathbf{X}_3$ matrices], with

   $$f_1(\mathbf{x}) = x_1^3 - e^{x_2} + \sinh x_3 - 3.6288188 \qquad \text{and} \qquad f_2(\mathbf{x}) = x_1^2 x_3 + \left(x_2^2 - x_3\right)^2 - 4.0$$

4. Plot $\mathbf{F}$ vs. $\mathbf{x}_1$ and $\mathbf{x}_2$ with Matlab's ***plot3*** command to try to "see" where the roots are located. Note that a ***view(2)*** perspective to give a top view might help and other views, such as ***view(0,0)*** may also be useful. Also, Matlab's ***contour*** function could prove to be useful here. Be creative…

   Note that, since we are interested in only small values of F (the scalar function F should be zero at the solution to the original nonlinear equations), we can focus the plot on the regions of interest by setting all the large values within matrix **F** to nan (not a number). For example, ***F(F > 1.0) = nan*** sets all values of F > 1.0 to nan -- and Matlab ignores nan values when plotting.

With the above background, perform the following analyses/computations:

a. Use the graphical technique described above to find a complete set of reasonable starting guesses within the domain defined by $-5 \le x_1 \le 5$ and $-4 \le x_2 \le 4$. Use a fine grid (i.e. lots of points) here so that you don't miss any potential regions of interest.

b. With appropriate guesses from the graphical analysis in Part a, use Matlab's ***fsolve*** command to find all the solutions to the original three coupled nonlinear equations within the domain defined in Part a. Here you will need a separate function file or an anonymous function to define the original three nonlinear equations, $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, for use by the ***fsolve*** routine.

c. Now, based on the description and the **nldemo2_lesson6.m** file from the Lecture Notes, implement Newton's method into Matlab for solution of this problem. Here you will need to evaluate the vector function and the Jacobian matrix at each guess for the root. Using the same initial guesses as in Part b, compare your solutions using Newton's method with those obtained using Matlab's ***fsolve*** command.