Mech 221: Computer Lab 3

Hand in the solutions to the three questions in the lab at the *end* of the lab.

Question 1: Function files

Consider the function $f(x) = x\ln(\sin x + 1.1)$. We wish to write an .m file function in MATLAB to evaluate this for a vector of values, `x`. We would do this by typing the following in an .m file:

```
function f = myfunc(x)

f = x.*log(sin(x) + 1.1);
```

and saving it as "myfunc.m".

- Create the myfunc.m file as we have defined it above.
- Create a vector `x` of 100 values from 0 to 10 using the `linspace` command and generate a plot of the function $f(x) = x\ln(\sin x + 1.1)$ from $x = 0$ to $x = 10$. Label the axes and add a title to your plot using the `xlabel`, `ylabel` and `title` commands described in the pre-lab. Print out your plot.
- Write an .m file that will evaluate $f(x) = e^x \cos(x^2)$ and plot the function for $0 \leq x \leq 3$. Make sure that the m. file function can accept x as a vector of values. Label the axes and add a title to your plot. Print out your plot and your function file.
- *Hand in both plots above and the code for your function file in the previous point.*

Question 2: Debugging Newton's Method

An important skill to have when using any programming language is the ability to debug your code. Programming errors in MATLAB are very common and MATLAB is usually good at telling you where the errors are occurring. If you are running an .m file and an error occurs, the message will usually tell you what type of error is occurring and on what line in your .m file it is occurring at.

Recall the example of Newton's Method given in your pre-lab. We want to find the positive, real solution x for the equation $x^6 - \pi = 0$ and we used

Newton's method to find it. Your friend, Bob, wanted to get a head start on this lab and decided to write a program in MATLAB for this example. Unfortunately his code has errors and does not work. His code is available for download from the vista site in a file entitled **NewtonsMethod.m**

- Download the code NewtonsMethod.m and save it in the directory that MATLAB is working in.
- Run the code in MATLAB, by typing "`NewtonsMethod`" in the command window. You will get an error.
- According to the message that appears when you run the code, what error is occurring? Where is the error occurring? (i.e. which file?, which line?). Without changing the code on the line where the error is occurring, how would you fix this specific error?
- There are more errors in Bob's code. Using MATLAB, figure out what these errors are and how to fix them. Apply your solutions to the code. Keep track of the errors you find and the changes you made by writing out a short list, which will be handed in. Note that there may be more than one error per line. Note also that even though a MATLAB code runs without MATLAB errors, it may have mathematical errors in it. Before your changes to the code become effective, you will need to save the file.
- Make sure you understand the meaning of the condition of the while loop in NewtonsMethod.m.
- What is the positive 6th root of π? (Use the debugged code with $x_0 = 3$ as your initial guess to solve this).
- *Hand in the list of errors in the code you found and the changes you made. Also hand in the sequence of Newton's iterations starting with $x_0 = 3$ generated above.*

Question 3: Plotting the solution

In the pre-lab, we had you solve a differential equation that had an implicitly defined solution. In this question we will go about plotting that solution for different values of x. Take our implicitly defined solution and write it in the form

$$G(y) - F(x) - C = 0.$$

Use Newton's method to solve for y given a value of x. Note that when taking the derivative of $G(y) - F(x) - C$ with respect to y, the derivative of $F(x)$ with respect to y is zero (x and so $F(x)$ are given constants for this problem).

- Write two .m file functions called `func` and `funcprime` as would be required to solve for y using Newton's Method. (Note: These functions will be required to accept two arguments: x and y.) Be sure to save them in the same directory as NewtonsMethod.m

- Modify NewtonsMethod.m to use `func` and `funcprime` to solve for y at a given x. See that it is working correctly by solving for y when $x = 0.5$. You should get $y \approx 1.1498$.

- This question may take a fair amount of time to program. Modify NewtonsMethod.m so that it will find the solution y for 30 values of x evenly spaced from $x = 0$ to $x = 3$. The vector `x` can be filled with the `linspace` command and the vector `y` of values to be determined can be initialized in size using the `zeros` command. The value $y(0)$ is known from the initial conditions. For each other value of x, find the solution for y using Newton's Method. You will want to use a `for` loop for this task. The value of y at the previous x can be used as an initial guess for the value of y at the next x. Print out your code for `NewtonsMethod.m, func.m` and `funcprime.m`.

- Plot the solution, $y(x)$ and add axes labels and a title.

- *Hand in the print outs of your code and your plot above.*