EAS 596, Fall 2019, Homework 1
Due Wednesday 9/4, **4 PM**, Box outside Jarvis 326

Work all problems. Show all work, including any M-files you have written or adapted. Make sure your work is clear and readable - if the TA cannot read what you've written, that work will not be graded. All electronic work (m-files, etc.) **must** be submitted through UBLearns by the due time shown above. Electronic files must obey the following naming convention: `ubitname_hw1_p1.m`, replacing ubitname with your ubitname. Any handwritten work may be submitted in class. Each problem will be graded according to the following scheme:

- 4 Points: Solutions are complete and correct. Code runs with no need for modification.
- 3 Points: One mistake in the code and it is easily found. Code runs after the modification.
- 2 Points: Two to three minor mistakes in the code, which are easily found. Code runs after the modification.
- 1 Points: Many mistakes in the code. No attempt will be made to modify it to run.
- 0 Points: Code has major conceptual issues.

Problem 1:
In this problem you will be constructing a simple root-finding function using the bisection method. Let $f(x)$ be a continuous function defined over the range $[a, b]$ such that $f(a) \times f(b) < 0$. The fact that $f(a) \times f(b) < 0$ indicates that the sign of $f(a)$ and $f(b)$ differs and therefore $f(y) = 0$ for some value $y \in [a, b]$. The bisection method is an iterative method:

1. Compute the mid-point between $[a, b]$: $c = (a + b)/2$.
2. If $f(a) \times f(c) < 0$ then replace the value b by c. Thus the new range is $[a, c]$. If $f(b) \times f(c) < 0$ replace a by c. Thus the new range is $[c, b]$.
3. Repeat until $b - a < \epsilon$, where ϵ is the tolerance.
4. Return a root of $x_0 = (b + a)/2$.

Write a MATLAB function with a call of `[root, nIters] = ubitname_hw1_p1(@(x)(f), [a b])`, where `f` is the function to find the root of and `[a b]` provides the initial range. The code `@(x)(f)` creates an anonymous function of a single variable. For example, to find the root of $\sin(4x)$ you would use `@(x)(sin(4*x))`. When completed this function returns both the root and the number of iterations needed.

This function should do the following:

- Check if $f(a) \times f(b) < 0$. If it does not, return an error using the `error` function.
- Iterate until $b - a < \epsilon$. Use $\epsilon = 10^{-6}$.
- If the number of iterations exceeds 1000, return an error using the `error` function.
- If successful return both the root and the number of iterations needed for convergence.

1. Test your function by computing the root of $\sin(x)$ using an initial range of $[2, 4]$.

2. In the range $[2, 10]$ there are three roots of $\sin(x)$. Using that range, use your function to compute the root. Will your function ever be able to find the other two roots? Explain your answer.

Problem 2:
This problem explores the trade-off between the truncation error in the finite difference approximation of the derivative and floating point truncation error incurred when using finite precision arithmetic. Using a simple Taylor series, it is easy to construct a *forward difference* approximation of the derivative:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h)$$

where h is some defined step-size.

1. Write a MATLAB script that computes the forward finite difference approximation of the derivative of $\sin(x)$ for $x = \pi/3$. Use step sizes, h, ranging from 10^{-1} to 10^{-16}. You may find the `logspace` command useful. Plot the error, using a log-log scale, between your approximation and the exact derivative for each of your step sizes. Make sure to upload your script to UBLearns and name it `ubitname_hw1_p2.m` and properly label the plot.

2. Based on your plot, what is the best step size to use for the finite difference step size?

3. Explain the behavior you see in the graph.