EAS 596, Fall 2019, Homework 7
Due Wednesday 12/4, **3:00 PM**, Box outside Jarvis 326 or in class

Work all problems. Show all work, including any M-files you have written or adapted. All electronic work (m-files, etc.) **must** be submitted through UBlearns and obey the following naming convention: `ubitname_hw7_pN.m`, replacing ubitname with your ubitname and `N` with the problem number.

All two point problems will be graded according to the following scheme:

- 2 Points: Solution is complete and correct.
- 1 Points: Solution is incomplete or incorrect, but was using correct ideas and concepts.
- 0 Points: Using incorrect ideas and concepts.

All four point problems 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.

1. (2 pts) Write a MATLAB function `[x, e] = ubitname_hw7_p1(f, a, b)` that accepts an anonymous function `f` of one variable and initial range `a` and `b` and uses the *Regula Falsi* method to return the root `x` such that `f(x)=0`. The function should also return the error for each iteration in `e`. Use a convergence criteria of $|f(y)| < 10^{-6}$ with a maximum number of iterations of 1000. If the method does not converge return `NaN` for the "root". Please upload your code to UBlearns. This problem does not require a hardcopy submission. Your function should not produce any output other than `x` and `e` (e.g. nothing should be printed to the command window).

2. (4 pts) Write a MATLAB function `[x, e] = ubitname_hw7_p2(f, x0)` that accepts an anonymous function `f` of one variable and initial guess `x0` and uses Newton's method to return the root `x` such that `f(x)=0`. The function should also return the error for each iteration in `e`. When computing the derivative use a second-order finite difference approximation: $f'(y) \approx \frac{f(y+h)-f(y-h)}{2h}$ with $h = 10^{-6}$. Use a convergence criteria of $|f(y)| < 10^{-6}$ with a maximum number of iterations of 1000. If the method does not converge return `NaN` for the "root". Please upload your code to UBlearns. This problem does not require a hardcopy submission. Your function should not produce any output other than `x` and `e` (e.g. nothing should be printed to the command window).

3. (2 pts) Write a MATLAB script which uses your functions from P1 and P2 to find the non-trivial solutions to $sin(x) = x^3$ in the range $[0.5, 1.0]$. Use the function from P1 with the initial range $[0.5, 1.0]$, the function from P2 with an initial guess of $x = 0.5$, and the function from P2 with an initial guess of 0.75. On a single plot use `semilogy` to plot the error versus iteration number for each result. On the figure use the `legend` command to label each line

with the method, initial range or guess, and final result. Use can use `sprintf('P1, [0.5, 1.0], root = %.4f', x)` and `sprintf('P2, x0, root = %.4f', x)` to create a formatted string for each method, replacing `x0` as appropriate for the Newton method. Please upload your code to UBlearns. This problem does not require a hardcopy submission.

4. (4 pts) Write a MATLAB function `[t,y]=ubitname_hw7_p4(f, tspan, y0, order, dt)` which solves an initial value problem `y'(t)=f(t,y)` where `tspan=[t0 tf]` is the time to solve over, `y0` is the value `y(t0)=y0`, `order` is the order, and `dt` is the time step to use. On return `t` contains the times for the solution vector `y`. On input `order` indicates the explicit method to use. The value can be 1 (Forward Euler), 2 (Improved Euler), or 4 (4^{th}-order Runge Kutta). Your code needs to check if an invalid option is entered and return an error. Please upload your code to UBlearns. This problem does not require a hardcopy submission.

5. (2 pts) The solution to the ODE $y'(t) = t + y(t)$ with $y(0) = 1$ is $y(t) = 2e^t - t - 1$. Using your code from P4 to create a MATLAB script called `ubitname_hw7_p5` which produces a log-log plot of the error at $t = 2$ as a function of time step for the three methods. Include on the plot a properly scaled line showing the expected order of convergence for each method. Be sure to label the axes and provide a proper legend. Please upload your code to UBlearns. This problem does not require a hardcopy submission.