Homework 04

Out of problem 1 and 2 choose only one to implement. Problems 3, 4, and 5 are to be done unconditionally.

With your email submission attach listings of each function which you implemented except maybe for problem 6.

General requirements:

1. All root finding functions must have optional outputs with the function value at solution point, and number of iterations. So the general root finding function definition should look like
   `function [x_sol, f_at_x_sol, N_iterations] = find_root_method(f_handle,...)`

2. **Name your function and use its parameters ordered exactly as prescribed in the problem assignment**. The TA will run tests assuming the prescribed naming scheme. If your code is working but does not follow the specification points will be reduced.

3. Check for the possible user **misuse** of the algorithms, think what could go wrong. All relevant input parameters should be validated against possible user errors. If user submits wrong input parameters, your function must exit with an error message immediately without attempts to correct the wrong behavior. See `error` documentation about how to do it.

4. Test your implementation with at least $f(x) = \exp(x) - 5$ and the initial bracket [0,3], but do not limit yourself to only this case.

5. If the initial bracket is not applicable (for example, in the Newton-Raphson algorithm) use the right end of the test bracket as the starting point of the algorithm.

6. All methods should be tested for the following parameters `eps_f=1e-8` and `eps_x=1e-10`.

Problem 1 optional (5 points):

Write proper implementation of the false position algorithm. Define your function as

```
function [x_sol, f_at_x_sol, N_iterations] = regula_falsi(f, xn, xp
, eps_f, eps_x)
```

Problem 2 optional (5 points):

Write a proper implementation of the secant algorithm. Define your function as

```
function [x_sol, f_at_x_sol, N_iterations] = secant(f, x1, x2, eps_f
, eps_x)
```

Problem 3 (5 points):

Write a proper implementation of Newton-Raphson algorithm. Define your function as `function [x_sol, f_at_x_sol, N_iterations] = NewtonRaphson(f, xstart, eps_f, eps_x, df_handle)`. Note that `df_handle` is a handle to calculate derivative of the function `f` it could be either analytical representation of $f'(x)$ or its numerical estimate via the central difference formula.

Problem 4 (5 points):

Write a proper implementation of Ridders' algorithm. Define your function as `function [x_sol, f_at_x_sol, N_iterations] = Ridders(f, x1, x2, eps_f, eps_x)`

Problem 5 (5 points):

For each of the root finding implementation of your homework find roots of the following two functions

(a) $f1(x) = \cos(x) - x$ with the 'x' initial bracket [0,1]

(b) $f2(x) = \tanh(x - \pi)$ with the 'x' initial bracket [-10,10]

Make a comparison table for the above algorithms with following rows

(a) Method name

(b) root of $f1(x)$

(c) initial bracket or starting value used for $f1$

(d) Number of iterations to solve $f1$

(e) root of $f2(x)$

(f) initial bracket or starting value used for $f2$

(g) Number of iterations to solve $f2$

make columns corresponding to 3 algorithms which you have chosen to implement.

If an algorithm diverges with the suggested initial bracket: indicate so, appropriately modify the bracket, and show the modified bracket in the above table as well. Make your conclusions about speed and robustness of the methods.

Problem 6 Bonus (2 points):

Plot the $\log_{10}$ of the absolute error of the $\sin(x)$ derivative at $x = \pi/4$ calculated with forward and central difference methods vs. the $\log_{10}$ of the step size h value. See `loglog` help for plotting with the logarithmic axes. The values of h should cover the range $10^{-16} \cdots 10^{-1}$ (read about Matlab's `logspace` function designed for such cases).

Why the error decreases as h goes down and then starts to increase?

Note: the location of the minimum of the absolute error indicates the optimal value of h for this particular case.