COMPUTER SCIENCE 349A, FALL 2018
ASSIGNMENT #3 - 20 MARKS

DUE FRIDAY OCTOBER 12, 2018 (11:30 p.m. PST)

This is a really large class and the logistics of grading assignments are challenging. Me and the markers require your help in making this process go smoothly. Please ensure that your assignments conform to the following requirements - any violation will result in getting a zero for the particular assignment.

- All assignments should be submitted electronically through the ConneX course website and shoud be **SINGLE PDF FILES**. No other formats will be accepted.. Handwritten answers are ok but they will need to be scanned and merged into a single pdf file together with any code examples and associated plots.
- The assignment number, student name and student number should be clearly visible on the top of every page of your assignment submission.
- **PLEASE DO NOT COPY THE ASSIGNMENT DESCRIPTION IN YOUR SUBMISSION**
- The asnwers to the questions should be in the same order as in the assignment specification.
- Some of the questions of the assignments are recycled from previous years but typically with small changes in either the description or the numbers. Any submission that contains numbers from previous years in any questions will be immediately graded with zero.
- Any assignment related email questions should have a subject line of the form CSC349A Assignment X, where X is the number of the corresponding assignment.
- The total number of points for this assignment is 20.

Question #1 - 10 marks.
Consider the function

$$f(x) = \frac{1 + \cos x}{(x - \pi)^2}$$

where $x \neq \pi$ is in radians.

(a) Using $b = 10$, $k = 4$, idealized, floating-point arithmetic with rounding, compute $fl(f(x))$ at $x = 3.154$. Note that $fl(\pi) = 3.142$.
Furthermore, The correct value of $f(3.154)$ to 8 correct significant digits is 0.49999359. Your floating-point approximation should have a relative error greater than 30%.

(b) Determine the fourth order Taylor polynomial approximation for $\cos(x)$ expanded about $x = \pi$ (Do not include the remainder term.) Leave this polynomial in terms of expressions involving $(x - \pi)^k$, for integer values of k.

(c) Substitute the polynomial approximation from (b) into the formula for $f(x)$, and simplify in order to obtain a polynomial approximation for $f(x)$. Note: do not multiply out the remaining factor $(x - \pi)^2$; leave it in this form.

(d) Show that the problem of computing $f(3.145)$ is well-conditioned. Use the definition of condition and the notation in Handout 6 to show this.

Hint: Use the polynomial approximation for $f(x)$ from (c) to show that the exact value of $f(3.154 + \varepsilon)$ is approximately equal to 0.49999 (that is, approximately equal to the exact value of $f(3.154)$) whenever $\left|\frac{\varepsilon}{3.154}\right|$ is small.

(e) Show that the computation of $fl(f(3.154))$ in (a) is unstable. Use the definition of stability and the notation in Handout 7 to show this.

Hint: Use the result in (c) to show that when $\left|\frac{\varepsilon}{3.154}\right|$ is small, the exact value of $f(3.154 + \varepsilon)$, done in (d), cannot be close to the floating-point calculation $fl(f(3.154))$ in (a), for any ε.

Question #2 - 10 Marks

(a) Write a MATLAB function M-file with header

```
function  root = Bisect ( xl , xu , eps , imax, f, enablePlot )
```

corresponding to the pseudocode given in Handout #8 for the Bisection method (in `xl` it is an "ell" not a "one"). Note, in the pseudocode "exit" is used to break out of the function. In Matlab, this will cause the program to close. You want to use "return" in implementation.

The only differences from that given algorithm are the following:

- print a caption for your computed approximations by inserting the following statement just before the <u>while</u> statement:

```
fprintf ( ' iteration     approximation \n')
```

- print each successive computed approximation by inserting the following statement after the computation of x_r at the beginning of the while loop:

  ```
  fprintf ( ' %6.0f %18.8f \n', i, xr )
  ```

- print a message to indicate that the algorithm has failed to converge in `imax` steps by replacing the last statement in the pseudocode by the following:

  ```
  fprintf ( ' failed to converge in %g iterations\n', imax )
  ```

- The extra argument *enablePlot* is used to select an optional plot of showing each iteration of the bisection method when *enablePlot* is set to 1. When *enablePlot* is set to 0 no figure will be generated. The command *hold on* can be used to in the same figure using the *plot* command. For example try the following to understand how this works:

  ```
  hold on;
  x = [0:0.1:1];
  plot(x, exp(x));
  plot(x, log(x));
  ```

  For each iteration you should plot the function between *xl* and *xu* as well as stars indicating on the graph the values of $f(xl)$, $f(xu)$ and $f(xr)$. The following example MATLAB code (continuation of the previous example) can help you understand the syntax to accomplish this.

  ```
  z = [0.2, 0.4, 0.8];
  fz = exp(z);
  plot(x, exp(x), z, fz, '*g');
  hold off;
  ```

  DO NOT INCLUDE ALL THE ENDPOINTS IN YOUR ANSWER. DO INCLUDE THE ENDPOINTS FOR ITERATIONS 1, 2, 4, 6.

DELIVARABLES: A copy of your MATLAB M-file.

(b) Water is flowing in a trapezoidal channel at a rate of $Q = 20m^3/s$. The critical depth y for such a channel must satisfy the equation:

$$0 = 1 - \frac{Q^2}{gA_c^3}B$$

where $g = 9.81m/s^2$, $A_c =$ the cross-sectional area (m^2), and $B =$ the width of the channel at the surface (m). For this case, the width and the cross-sectional area can be related to depth y by

$$B = 3 + y \quad \text{and} \quad A_c = 3y + \frac{y^2}{2}$$

Express this problem as a root finding problem for an appropriately defined function of the critical depth y.

DELIVARABLES: Show all your work in deriving your formula.

(c) Use the function M-file Bisect to solve the above problem (b) with initial guesses of $x_l = 0.5$ and $x_u = 2.5$, and itereate until the approximate error falls below 1% or the number of iterations exceeds 10. You will need to write an additional MATLAB function M-file. If

```
function  y = your_function(x)
```

corresponds to the function of which you are computing a zero call this with

```
Bisect ( xl  , xu  , eps  , imax, @your_function, enable_plot)
```

with the appropriate parameter values.

DELIVARABLES:

- the additional function M-file.
- a copy of the MATLAB statement(s) you used to call `Bisect`.
- your output from Bisect including the figure with the endpoints for iterations (1,2,4,6).