MACM 316 – Computing Assignment #3

Due Date: Friday October 23 at 11:00pm

Submission Instructions: You must upload one .pdf file to **Crowdmark** that consists of 2 pages ONLY: page 1 is your report which should fit all of your results, discussion, data and figures into a single page; and page 2 is a listing of your code. The deadline is **11:00pm** on the due date. The actual due time is set to 11:05pm – if Crowdmark indicates that you submitted late then you will be assigned a grade of 0 on this assignment. Your TA has emailed you a Crowdmark link that you will use for the entire semester to upload your assignment solutions.

- Review the **Guidelines for Computing Assignments** carefully.
- Acknowledge any collaborations or assistance from colleagues/TAs/instructor.
- If you have questions about this assignment or Matlab programming, you can obtain help by posting your questions to the "Computing Assignment" discussion board in Canvas. This discussion board will be checked regularly, and also monitored continuously during computational workshop hours.

Computing Assignment – Hybrid Newton-Bisection Algorithm

In this computing assignment, you will extend Newton's method to use a hybrid approach that combines it with the bisection method. The single initial guess is replaced with an interval that brackets the root, and the bracket is updated as needed using bisecton steps. Consider the following equation

$$3.06 = \frac{(1-x)(3+x)^{1/3}}{x(4-x)^{1/2}} \tag{1}$$

where Newton's method alone can easily fail, but a hybrid Newton–bisection algorithm is more reliable – in other words, the hybrid method is *more robust*.

Your report should address the following:

(a) Express eq. (1) as a nonlinear root-finding problem of the form $f(x) = 0$. Plot your function $f(x)$ on the interval $[-2, 3]$ and describe its overall behaviour. Clearly identify the number of roots and their approximate locations.

(b) Attempt to approximate the positive root of $f(x)$ using Newton's method with initial guess $x_0 = 1$, and then repeat using bisection method with initial bracket $[0.1, 1.0]$. You can use the `bisect2.m` and `newton.m` codes from lectures. Discuss your results.

(c) Create a modified version of the Newton code (say, `newtonb.m`) that takes a bracket interval as input instead of a single initial guess. Using the given initial bracket, take a single bisection step to determine x_0, which is then used as the initial guess for the Newton iteration. Test your `newtonb.m` code using the initial bracket $[0.1, 1.0]$.

Note: *A single initial bisection step is often not sufficient to get Newton's method to converge, and so some form of bisection must be incorporated inside the Newton iteration as well. The remaining two parts add this feature to your code.*

(d) Write a utility function called `newtBrack.m` that takes a single step of Newton's method and checks whether the new guess for the root lies within the current bracket. The function should have a definition resembling

```
function [ok, xnewt] = newtBrack(a, b, x, fx, fpx)
```

where a and b are the bracket endpoints, x is the previous Newton iterate, and `fx` and `fpx` are the values of $f(x)$ and $f'(x)$ (not the functions themselves!). The return value `xnewt` is the new guess for the root using a Newton step. The value of `ok` is a logical variable (`0` = false, `1` = true) that identifies whether `xnewt` lies within the interval $a \leq x \leq b$. Modify your `newtonb` code from part (c) to replace the Newton step with a call to `newtBrack`. The modified `newtonb` function should continue with the normal Newton iterations unchanged, but print a warning message if `ok` is "false".

(e) Next, update the iteration in your `newtonb` function so that it takes a Newton step only if the new approximation `xnewt` from `newtBrack` lies inside the current bracket (that is, if `ok == 1`). If `xnewt` falls outside the bracket, then print a warning message and take a bisection step using the midpoint from bisection as your next guess instead ... and don't forget to update your current bracket $[a, b]$! Use an absolute error tolerance of 10^{-10} for your stopping criterion. In your report, give a list of your iterations that you obtain from the updated `newtonb` code, including all warning messages. Finally, add the approximate root to your plot from part (a).