MACM 316 – Computing Assignment #2

Due Date: Friday October 6 at 11:00pm.

Instructions: Upload to Crowdmark a single PDF file that consists of only two pages: page #1 is your report (containing all discussions, data and figures), and page #2 is a listing of your Matlab code. The assignment is due at **11:00pm sharp!** If Crowdmark indicates that your submission was late, then you will be assigned a grade of zero – with no exceptions – so don't leave submitting until the last minute. When this CA is released, you should receive an e-mail containing a link for uploading your assignment, so make sure you save this message.

- Carefully review the **"Guidelines for Computing Assignments"** posted on Canvas.
- If you have any questions about this assignment or Matlab, then you can obtain help in the computational workshops, tutorials, or the "Computing Assignment" discussion group on Canvas.
- Acknowledge any collaborations or assistance from fellow students, TAs or instructor.

CA2 – Explorations in Root-Finding

In this assignment, you will apply a combination of analytical techniques and root-finding algorithms to study the roots of the nonlinear equation

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

(a) Start by plotting $f(x)$ on the interval $x \in [-6\pi, 6\pi]$ and describe the overall behaviour of the function, as well as the number and rough location of its roots. Use the "zoom" feature of Matlab's plotting window (or change the axis limits with `set` or `axis` commands) to make sure that you identify **all roots** – you may have to increase your plotting point density in order to view f in sufficient detail!

(b) Investigate analytically what happens to the function for large $|x|$ by computing the following two limits of the exponential term in $f(x)$:

$$\lim_{x\to-\infty}\left(\frac{1}{1+e^{-2x}}\right) \qquad \text{and} \qquad \lim_{x\to+\infty}\left(\frac{1}{1+e^{-2x}}\right).$$

Use these results to determine two simpler "limit functions" that approximate the negative and positive "halves" of $f(x)$:

- $f_-(x)$ for $x < 0$, which approximates f for large negative values of x,
- $f_+(x)$ for $x > 0$, which approximates f for large positive values of x.

Plot $f_\pm(x)$ on their respective half-intervals along with the original function $f(x)$. Then derive analytically the exact values for all roots of $f_-(x)$ (for $x < 0$) and $f_+(x)$ (for $x > 0$). Add these roots to your plot and comment on how well they seem to approximate the actual zeros of $f(x)$.

(c) Next, apply the bisection method to compute to within an absolute error tolerance of 10^{-6} the smallest[†] <u>negative</u> root of $f(x)$ (call it x^*). Use the `bisect2.m` code from lectures and choose an initial bracket that is motivated by your plot from part (b). Using an appropriate error measure, compare your computed root x^* to the smallest root of $f_-(x)$ (call it x_-) that you determined analytically in part (b). How well does x_- approximate your bisection result x^*?[‡]

Use bisection to compute the next two negative roots of $f(x)$ and compare them with the corresponding roots of $f_-(x)$. What do you notice about the accuracy of your approximations?

[†]By "smallest" I mean the smallest in magnitude or closest to zero.

[‡]Here I want you to think of the bisection result as your "exact" root, and the analytical result as the "approximate" root.

(d) Repeat part (c) to compute the four smallest <u>positive</u> roots of $f(x)$ and compare them with roots of $f_+(x)$. Explain how you choose an appropriate initial bracket for each root, and compare your computed roots with the corresponding estimates x_+ from $f_+(x)$.

(e) Consider the following fixed point iteration[§]:

$$x_{k+1} = g(x_k) = \arccos\left(\frac{-1}{1+e^{-2x_k}}\right). \qquad (*)$$

Show that finding a fixed point of $g(x)$ is equivalent to finding a root of $f(x) = 0$. Use the code `fixedpt.m` from lectures to approximate two roots:

- the smallest negative root from part (c) using an initial guess of $x_0 = -1.5$, and
- the smallest positive root from part (d) using an initial guess of $x_0 = 3.0$.

Compare with the results you obtained using the bisection method, making sure that you apply the same stopping criterion for x – *note that this may require modifying the* `fixedpt` *code!*

Describe any unexpected behaviour you observe. Can you explain the convergence of your fixed point iterations (perhaps with the help of a sketch)? What is the real problem with using the fixed point function in $(*)$ to try to compute <u>all roots</u> of $f(x)$?

[§] The Matlab built-in function for the "arccos" or inverse cosine function is called `acos`.