A few notes:

- The first 2 homework problems are not really related to the discussion of computational math we have had in class, but are designed to get you comfortable (or uncomfortable) with more advance programming features in MATLAB.

- The completed homework assignment should be turned in to your shared google drive folder in a subfolder labeled “HW2”. All work and code should be self-contained in one PDF file with the name “HW2_<YourMainFolderName>.pdf”. All code (mlx and m files) for the problems should also be put in the folder so that I may run it to produce the results you present.

1. (Generating random numbers, 15pts) Using the MATLAB rand command, generate an array of \( N = 10^5 \) random numbers \( x_i, \ i = 1, 2, \ldots, N \) in the interval \([0, 1]\). Then, use the cumulative sum function cumsum to construct an array containing the “partial averages” \( s_1, s_2, \ldots, s_N \) defined as

\[
s_n = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

(a) Plot \( s_n \) as a function of \( n \) and show that as \( n \) gets large, the average value of the first \( n \) random numbers in your array approaches 1/2. To show this, add to your plot the line \( y = 1/2 \). Your plot should be similar to that shown in Figure 1, but not exactly the same.

(b) Add a title and axis labels to your plot.

![Plot of cumulative averages of a randomly distributed variable.](image)

Figure 1: Problem 1 : Plot of cumulative averages of a randomly distributed variable.

For this problem, you will need the functions rand, cumsum, plot and the colon (:) operator.
2. **(Continued fractions, 25 pts)** Continued fractions take the following form

\[
x = a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4 + \ldots + \frac{1}{a_n}}}},
\]

where \(a_1\) is an integer and \(a_2, a_3, \ldots, a_n\) are positive integers. Many of the most famous irrational numbers can be expressed as infinite continued fraction expansions. Truncating these expansions after a relatively few terms often gives an excellent approximation to the irrational number.

(a) Write a function that computes the continued fraction expansion for a given array (or vector) of numbers \(a_j, j = 1, 2, \ldots, n\). Your code should take as input an array containing the continued fraction coefficients \(a_j\) and return the value \(x\) using formula in (1).

(b) The continued fraction coefficients for \(\sqrt{2}\) are \(a_1 = 1\) and \(a_j = 2\), for \(j = 2, 3, \ldots\). Create an array containing these numbers up to \(n = 20\) terms (i.e. \(a = [1; 2*ones(19,1)]\)) and use your function from part (a) to approximate \(\sqrt{2}\). Report the approximation you obtain to 16 digits and report the absolute error in the approximation.

(c) Use the Online Encyclopedia of Integer Sequences to find the integer sequence A001203 for the continued fraction coefficients that can be used to compute \(\pi\). Use these coefficients with your function from part (a) to approximate \(\pi\) to full machine precision (16 digits). Report your result and report the minimum \(n\) you need to use to obtain the approximation to full machine precision.

3. **(Lambert W function, 20pts)** Consider the following problem: Given a value of \(c \geq -1/e\), find the value of \(x\) such that

\[
x e^x = c.
\]

This is related to the Lambert W function, which has many applications ([https://en.wikipedia.org/wiki/Lambert_W_function](https://en.wikipedia.org/wiki/Lambert_W_function)).

Use Newton’s method to solve equation (2) for values of \(c \geq -1/e\) (Hint: solve the problem \(c - xe^x = 0\)). Produce three tables showing convergence for your method for \(c = 0.5, c = 1,\) and \(c = 10^{-14}\). You may have to experiment around with making initial guesses.

4. **(Computing square roots, 15pts)** The almost universally used algorithm to compute \(\sqrt{a}\), where \(a > 0\), is the recursion

\[
x_{n+1} = \frac{1}{2} \left( x_n + \frac{a}{x_n} \right),
\]

easily obtained by means of Newton’s method for the function \(f(x) = x^2 - a\). One potential problem with this method is that it requires a floating point division, which not all computer processors support, or which may too expensive for a particular application. For these reasons, it is advantageous to devise a method for computing the square root that only uses addition, subtraction, multiplication, and division by 2 (which can be easily done by shifting the binary representation one bit to the right). The trick for doing this is to use Newton’s method to compute \(\frac{1}{\sqrt{a}}\), and then obtain \(\sqrt{a}\) by multiplying by \(a\). Write down your recursion formula for computing \(\frac{1}{\sqrt{a}}\) in a manner similar to (3). This formula should only involve addition/subtraction, multiplication and division by 2.

Try you algorithm on the problem of computing \(\sqrt{5}\). As an initial guess use \(x_0 = 0.5\). Report the values of \(x_0, x_1, \ldots, x_5\) in a nice table and verify that your algorithm is working by comparing these numbers to the true value of \(\sqrt{5}\).

To see where this sort of software assisted acceleration is used in gaming, see the course webpage for a link to the article: *Origin of Quake3’s Fast InvSqrt(*).
5. (Using `fzero`, 15pts)

(a) Consider the Colebrook equation for the friction factor in a fully developed pipe flow

\[
\frac{1}{\sqrt{f}} = -2 \log_{10} \left( \frac{\epsilon/D}{3.7} + \frac{2.51}{ReD\sqrt{f}} \right)
\]

where $f$ is the Darcy friction factor, $\epsilon/D$ is the relative roughness of the pipe material, and $Re$ is the Reynolds number based on a pipe diameter $D$. Use `fzero` to find the friction factor $f$ corresponding to parameter values $\epsilon/D = 10^{-4}$ and $Re_D = 3.3 \times 10^5$. Use a tolerance $10^{-8}$ with `fzero`. In your homework write-up give the value of $f$ as well as the code you used to solve the problem. Do not include the `fzero` code just how you called it.

**Hint**: Use the function `optimset` to set up the tolerance $TolX$.

(b) David Peters (SIAM Review, 1997) obtains the following equation for the optimum damping ratio of a spring-mass-damper system designed to minimize the transmitted force when an impact is applied to the mass:

\[
\cos \left[ 4\zeta \sqrt{1-\zeta^2} \right] = -1 + 8\zeta^2 - 8\zeta^4
\]

Use `fzero` to find the $\zeta \in [0,0.5]$ that satisfies this equation with a tolerance $10^{-12}$. In your homework write-up give the value of $\zeta$ as well as the code you used to solve the problem. Do not include the `fzero` code just how you called it.

6. (Freezing water mains\(^1\), 10pts) Water utility companies would like to avoid freezing water mains, which is why they bury them in the ground. The question is how deep should they be buried? Newton’s law of cooling can be used to provide a good estimate. If uniform soil conditions are assumed, then according to this law, the temperature $T(x,t)$ at a distance $x$ below the surface of the ground and time $t$ is given approximately by

\[
\frac{T(x,t) - T_s}{T_i - T_s} = \text{erf} \left( \frac{x}{2\sqrt{\alpha t}} \right).
\]

(4)

Here $T_s$ is the temperature at the ground surface, $T_i$ is the initial temperature of the soil and $\alpha$ is the thermal conductivity of the soil. Suppose that $T_i = 18.3^\circ$ C (the average temperature in Boise in October), $T_s = -4.5^\circ$ C (the average temperature over the longest period of freezing temperatures in Boise), and $\alpha = 0.138 \cdot 10^{-6}$ m$^2$/s (an estimate of the average thermal conductivity properties of soil). Using these values in (4), determine how deep a water main should be buried so that it will not freeze until at least 30 days’ exposure. Report your value with the correct units.

Note: you can use `fzero` to solve this problem. Also, the function `erf` in (4) is called the *error function* and is available in MATLAB using `erf`.

---

\(^1\)Adapted from Numerical Computing in MATLAB problem 4.16