MA 3457 / CS 4033
HW #1
Due: Tuesday 10/27 by 11 pm

This assignment is due on **Tuesday 10/27 by 11 pm on Canvas**. Your assignment submission should contain a few files. You should submit all of your Matlab code and it should be properly commented to explain what the code is doing. You can submit as separate m-files saved as HW#Q# OR you can submit a single word or text file with all of the code pasted in (specifying or delineating code for each problem). For additional written work and discussion of problems, this should be a single pdf that is well-organized and either typed or neatly written. (If hand-written, use an app to scan and save as a single pdf). This file should be saved as HW#. To receive full credit on a problem, the code must run with no errors and the written work/discussion of the problem must also be complete. Matlab output should be discussed in the write-up.

1. (8 points) *Taylor Series*: Taylor polynomials are truncated Taylor Series and are often a good local approximation of a function close to the point the series is expanded about. In this problem, you are exploring the truncation error (remainder) when using the Taylor polynomial $P_2(x)$ for the function $f(x) = e^x \cos(x)$ about $x_o = 0$.

   (a) (4 points) Use $P_2(0.5)$ to approximate $f(0.5)$. Find an upper bound for absolute error $|f(0.5) - P_2(0.5)|$ using the truncation error (remainder) formula. Compare the error bound to the absolute error.

   (b) (4 points) Approximate $\int_0^1 f(x)dx$ using $\int_0^1 P_2(x)dx$. Find an upper bound for the error using $\int_0^1 |R_2(x)|dx$ and compare the bound to the absolute error.

   Note: You may do this problem with or without code. Please include a detailed write-up as well as commented m-files if you choose to utilize code for the error calculations.

2. (8 points) *Machine Epsilon*: Machine Epsilon is the maximum relative error for a specified rounding procedure. Commonly, this is determined as the smallest floating point number that when added to 1, results in a floating point number greater than 1. In this problem you are exploring what machine epsilon is in Matlab.

   (a) (5 points) Write a program (m-file in MATLAB) that will find machine epsilon. One way to do this is set an initial value, e.g. `x=1`, and repeatedly change it by making it smaller, e.g. `x=x/2`, and check whether Matlab can still recognize that it is something greater than 0, e.g. checking to see if `x+1>x`. Your code should utilize loops and/or conditional statements.

   (b) (3 points) Comment on your machine epsilon value from your code and compare it with the built in Matlab command that determines machine epsilon, which can be found by typing in `eps` in the command window.

3. (8 points) *Single and Double Precision*: We briefly discussed that there are different floating point representations of numbers. Single precision format uses 32 bits whereas double precision uses 64 bits. As one might guess, single precision is often used when precision matters less and double precision is used in scientific calculations. By default, everything in Matlab is double precision unless the user specifies that it should be single precision.

   Consider the following linear system $A\vec{x} = \vec{b}$:

   $$\begin{pmatrix} 1 & 1-\alpha \\ 1+\alpha & 1 \end{pmatrix} \vec{x} = \begin{pmatrix} 2-\alpha \\ 2+\alpha \end{pmatrix}$$

   (a) (2 points) Verify by hand that the solution is $\vec{x} = (1,1)^T$.

   (b) (4 points) In Matlab, determine the numerical solution when $\alpha =$`2e-3` by doing the direct calculation (recall there is a formula for the inverse of a 2×2 matrix and we can write the solution as $\vec{x} = A^{-1}\vec{b}$. Next, repeat this calculation in Matlab, specifying that α should be single precision by adding the line $\alpha =$`single(`α`)`. (If you are getting the same answer, you might need to clear previous variables by typing `clear all` at the top of the m-file or in the command window.

(c) (2 points) Comment on the exact solution versus the different solutions in single and double precision.

4. (6 points) *Loss of Significance:* Subtractive Cancellation can increase error or reduce precision of a final computed answer.

   Consider the following functions:

$$f(x) = x(\sqrt{x+1} - \sqrt{x})$$
$$g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

   (a) (3 points) Verify by hand that $g(x)$ is equivalent to $f(x)$.

   (b) (3 points) Compare the results of calculating $f(500)$ and $g(500)$ by hand using six digit rounding.

   Note: An example of six-digit rounding is given here: 0.123456# would be replaced/approximated by 0.123457 if $\# \geq 5$ or 0.123456 if $\# < 5$.