Homework 1, Due: Thursday, 11/1

This assignment is due on **Thursday, November 1**, by 11:59 PM. Your assignment should be well-organized, typed (or neatly written and scanned) and saved as a .pdf for submission on Canvas. You must show all of your work to receive full credit. For problems requiring the use of MATLAB code, remember to also submit your .m-files on Canvas as a part of your completed assignment. Your code should be appropriately commented to receive full credit.

Problems

$\boxed{1}$ Find the second Taylor polynomial $P_2(x)$ for the function $f(x) = e^x \cos(x)$ about $x_0 = 0$.

(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.

$\boxed{2}$ Consider the real number $p = e^{10}$.

(a) (3 points) Write the normalized floating point decimal form of p using both 6-digit chopping and 6-digit rounding.

(b) (3 points) Compute the actual error, absolute error, and relative error in approximations of p by $p^* = 22000$.

$\boxed{3}$ Suppose that $0 < q < p$ and that $\alpha_n = \alpha + O(n^{-p})$.

(a) (4 points) Show that $\alpha_n = \alpha + O(n^{-q})$.

(b) (4 points) Make a table listing $1/n$, $1/n^2$, $1/n^3$, and $1/n^4$ for $n = 5$, 10, 100, and 1000 and discuss the varying rates of convergence of these sequences as n becomes large.

$\boxed{4}$ This problem is designed to give you practice entering and interpreting some basic MATLAB commands.

(a) (6 points) Consider the following output from the MATLAB command window:

```
>> A = [3 -1 2 0 5; 1 0 -4 2 7; 2 1 4 3 9]

A =

     3    -1     2     0     5
     1     0    -4     2     7
     2     1     4     3     9

>> size(A)

ans =

     3     5

>> A'

ans =

     3     1     2
    -1     0     1
     2    -4     4
     0     2     3
     5     7     9
```

Describe what each of the three commands (next to the `>>` symbols) above do and interpret their output.

For the remaining parts of the problem, you'll be entering commands into the MATLAB command window. Create a script .m-file in MATLAB to save your commands and a diary file to save your command window output. See "sample_hw1prob4_script.m" posted on Canvas for an example of how to do this. You can download and edit this sample file, or start from scratch by clicking "New Script" on the Home tab of the MATLAB toolbar.

(b) (6 points) Enter the following matrices into MATLAB (for help, see the example in the "Intro to MATLAB" .pdf on Canvas):

$$\mathsf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathsf{B} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 3 & 1 \end{bmatrix}$$

Enter `A*B` and display the output. What does this command produce? Now enter `inv(A)` and display the output. What does this command produce? What happens if you enter `inv(B)` and why? (For the full experience, turn up the volume on your computer before entering `inv(B)` to hear the sound that MATLAB makes! `beep`.)

(c) (4 points) To test the formatting options, enter `y = [4/3 1.2345e-6];` then enter each of the following commands, followed by the variable `y` each time to display the changes to the output: `format short`, `format long`, `format short e`, `format rat`. Comment on how each command changes the formatting of `y`. Before moving on, enter `format short` to get back to the default setting.

(d) (4 points) Enter the vector `x = 1:0.5:6` and the command `length(x)`. What does the `length` command tell you? What is `x(3)`? Now enter the following code:

```
for i = 1:length(x)
    z(i) = log(x(i));
end
```

This is called a "`for` loop". Enter `help for` in the command window to learn more about `for` statements. What does this particular `for` loop compute? Display the output vector `z`. (Note: This particular `for` loop is for demonstration purposes only. One of the tricks of coding in MATLAB is to figure out ways to get around using `for` loops when possible to speed up computations. Explore `help log` and see if you can figure out a way to compute `z` without using a loop.)

$\boxed{5}$ (6 points) In a floating point number system, *machine epsilon* is defined as the smallest floating point number which, when added to 1, results in a floating point number greater than 1. To explore machine epsilon in MATLAB, create a MATLAB code to implement and run the following algorithm for approximating machine epsilon:

```
macheps = 1;
while ( 1 + macheps > 1 )
    macheps = macheps / 2;
end
macheps = 2*macheps;
```

Compare the results of your algorithm to the MATLAB command `eps`.

Note: For any of the above problems for which you use MATLAB to help you solve, you must submit your code/.m-files as part of your work. Your code must run in order to receive full credit.