Homework #3

Math 2010
Due Midnight, October 15

- Create a script file that performs calls the appropriate .m files for the following problems. Publish the report as an html file. Zip all the .m files, figures and html file together and upload zip file to the dropbox in D2L.

1. (8 points) Write an m-file *funcavgvel.m* which determines the average bungee jumper velocity over a range from $t = 0$ to $t = 12$. Given mass (m) and constant drag velocity (c_d) and gravity (9.8 m/s^2), the velocity of a bungee jumper is given as

$$v(t) = \sqrt{\frac{gm}{c_d}} \tanh\left(\sqrt{\frac{gc_d}{m}}t\right).$$

   Use this function given $m = 68.1$ and $c_d = 0.25$. The first line of your m-file should look something like

   function favg = funcavgvel(a,b,n)

   Here a is the lower bound of the range, b is the upperbound of the range, and n is the number of intervals. Helpful hints: You will need to define a vector of n points going from $t = 0$ to $t = 12$ (the built-in Matlab *linspace* command may be helpful here). Then find the vector consisting of the corresponding velocities as well (you do not need a for loop to do this). Then, to compute the average, you can use the built-in Matlab function *mean*.

2. (8 points) Revise the above m-file *funcavgf.m* to now find the **average** of a nonspecific function name f that is passed in as an argument (use a function handle). The first line of the file should be similar to this:

   function favg = funcavgf(f,a,b,n)

   Try this out by defining the function *vel* **using a function handle** in the workspace (see the notes on function handles) to be the velocity of the bungee jumper given a specific time t. You should have something like

   vel = @(t) ....(function definition)

   in the workspace, and should call your average function using the command similar to

   funcavgf(vel,0,12,60)

   for instance.

3. (14 points) For computers, the machine epsilon ϵ can also be thought of as the smallest number that when added to one gives a number greater than 1. An algorithm based on this idea can be developed as

Step 1: Set $\epsilon = \mathbf{1}$

Step 2: If $1 + \epsilon$ is less than **or** equal to 1, then go to Step 5. Otherwise go to Step 3.

Step 3: $\epsilon = \frac{\epsilon}{2}$

Step 4: Return to Step 2.

Step 5: $\epsilon = 2 * \epsilon$

   Write your own M-file based on this algorithm tp determine the machine epsilon. Validate the result be comparing it with the value computed with the built-in function *eps*.

4. (14 points) Number 1.39 in your textbook, *Numerical Computing with MATLAB* by Cleve Moler.
5. (14 points) Number 1.34 in your textbook, *Numerical Computing with MATLAB* by Cleve Moler.
6. (14 points) Number 3.3 (a) and (b) in your textbook, *Numerical Computing with MATLAB* by Cleve Moler.
7. (14 points) Number 3.4 in your textbook, *Numerical Computing with MATLAB* by Cleve Moler.
8. (14 points) Number 3.5 in your textbook, *Numerical Computing with MATLAB* by Cleve Moler.

- **Graduate Students:** The graduate students will be working towards a presentation (end of the semester) on applying numerical analysis methods in research. Graduate students should look for a specific project posted tomorrow on D2L. It will have a different deadline but the time allotted for this extra project will overlap with this current assignment.