COMPUTER SCIENCE 349A, FALL 2021
ASSIGNMENT #1

DUE FRIDAY SEPTEMBER 24, 2021 (11:30 p.m. PST)

This is a really large class and the logistics of grading assignments are challenging. Me and the markers require your help in making this process go smoothly. Please ensure that your assignments conform to the following requirements - any violation will result in getting a zero for the particular assignment.

- All assignments should be submitted electronically through the Brightspace course website and shoud be a **SINGLE PDF FILE** with all the material plus files for any functions (.m or .py). No other formats will be accepted.. Handwritten answers are ok but they will need to be scanned and merged into a single pdf file together with any code examples and associated plots.
- The assignment number, student name and student number should be clearly visible on the top of every page of your assignment submission.
- **PLEASE DO NOT COPY THE ASSIGNMENT DESCRIPTION IN YOUR SUBMISSION**
- The asnwers to the questions should be in the same order as in the assignment specification.
- Some of the questions of the assignments are recycled from previous years but typically with small changes in either the description or the numbers. Any submission that contains numbers from previous years in any questions will be immediately graded with zero.
- Any general assignment related questions can be posted in the Discusscion Forum in Brightspace for the assignment.
- Any specific assignment related questions (details about your solution) should be e-mailed (rlittle@uvic.ca) to me and have a subject line of the form CSC349A Assignment X, where X is the number of the corresponding assignment.
- You may use Python instead of MATLAB anywhere the instructions say MATLAB. In that case replace M-FILE or .m file with .py file. Again, I recommend the Spyder IDE for this.

A MATLAB function M-file for Euler's method (as described in class and on pages 17-19 of the textbook) for solving the differential equation

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

(this is (1.9) on page 14) is given below. As discussed in class, this numerical method is obtained by approximating $\frac{dv}{dt}$ at time t_i by $\frac{v(t_{i+1})-v(t_i)}{t_{i+1}-t_i}$, which results in the computed approximation

$$v(t_{t+1}) = v(t_i) + \left[g - \frac{c}{m}v(t_i)\right](t_{i+1} - t_i).$$

To create a MATLAB function M-file, either type **edit** in the Command Window or select **HOME** → **New** → **Script** or select **HOME** → **New** → **Function** (the latter gives you a template for creating a function).

Each of these options will open a new window (an Editor window) in which to type in the MATLAB statements for Euler's method. Enter the following, (Note - don't copy and paste as the quotes might not work). Statements starting with % are comments, documenting the MATLAB code.

```
function Euler(m,c,g,t0,v0,tn,n)
%  print headings and initial conditions
fprintf('values of t    approximations v(t)\n')
fprintf('%8.3f',t0),fprintf('%19.4f\n',v0)
%  compute step size h
h=(tn-t0)/n;
%  set t,v to the initial values
t=t0;
v=v0;
%  compute v(t) over n time steps using Euler's method
for i=1:n
     v=v+(g-c/m*v)*h;
     t=t+h;
     fprintf('%8.3f',t),fprintf('%19.4f\n',v)
end
```

To save your M-file, select

EDITOR → **Save** → **Save As...**

At the top of this window, it should say

File name: Euler.m

Save as type: MATLAB files (*.m)

Select

Save

to save your file, and close the Editor window.

In order to use the above function, you must specify values for the 7 local parameters in the function Euler:

`m` is the mass of the falling object
`c` is the drag coefficient
`g` is the gravity constant
`t0` is the initial time, `v0` is the initial velocity
`tn` is the final time at which the velocity is to be computed
`n` is the number of time steps into which $[t_0, t_n]$ is divided

Thus, in the function `Euler`, the step size $h = (t_n - t_0)/n$ is computed, and Euler's method is used to compute an approximation to the solution $v(t)$ of the differential equation at the n points (values of time)

$$t_0 + h,\ t_0 + 2h,\ t_0 + 3h,\ t_0 + 4h,\ \ldots,\ t_0 + nh\ =\ t_n.$$

For example, in order to use Euler to solve the problem given in Example 1.2 on page 17 and to save your results in a file for printing, you could enter the following (in the MATLAB Command Window):

diary filename
Euler (68.1 , 12.5 , 9.81, 0 , 0 , 12 , 6)

{*the desired results should appear here*}

diary off

Question #1 - 8 marks

(a) Create a working copy of the MATLAB function Euler in your installation of MATLAB.

DELIVERABLES: A copy of the M-FILE in your pdf plus the M-FILE itself.

(b) Use Euler to solve the differential equation using $m = 82.6$, $c = 12.5$ and initial conditions $v(0) = 0$ on the time interval $[0, 12]$ using 20 time steps and $g = 9.81$.

DELIVERABLES: Your answer should include the function call to `Euler` and the resulting output.

(c) Use Euler for a falling parachutist with the same parameters as Q1b but with a gravitational constant of 8.83 (as would be the case if the parachutist was falling on Venus).

DELIVERABLES: Your answer should include the function call to `Euler` and the resulting output.

(d) Use MATLAB to compute the relative error $|\varepsilon_t|$ in the computed approximation at time $t = 12$, using the constants from Q1b. To do this, use the true (exact) solution:

$$v(t) = \frac{gm}{c}(1 - e^{-\frac{ct}{m}})$$

DELIVERABLES: A copy of the commands and output.

Note that in MATLAB e^x is computed as `exp(x)` and $|x|$ as `abs(x)`.

Question #2 - 6 Marks.

In our mathematical model of a falling parachutist, instead of assuming that air resistance is linearly proportional to velocity (that is, $F_U = -cv$), you might choose to model the upward force on the parachutist as a second-order relationship,

$$F_U = -kv^2$$

where k is a second-order drag coefficient. This leads to the following differential equation

$$\frac{dv}{dt} = g - \frac{k}{m}v^2$$

(a) Modify the MATLAB function Euler in Question 1 so that it will use Euler's method to solve this differential equation. Use the function header

Euler2(m , k , g, t0 , v0 , tn , n)

DELIVERABLES: A copy of the M-FILE in your pdf plus the M-FILE itself.

(b) Use Euler2 to compute a numerical approximation to the above differential equation using $m = 82.6$, $k = 0.234$ and initial condition $v(0) = 0$ on the time interval $[0, 12]$ using 20 time steps.

DELIVERABLES: The function call to Euler2 and the resulting output.

(c) Use the fact the exact analytic solution of this problem is

$$v(t) = \sqrt{\frac{gm}{k}} \tanh\left(\sqrt{\frac{gk}{m}}t\right)$$

to compute (either in MATLAB or using your calculator) the relative error in the computed solution at $t = 12$.

Question #3 - 6 Marks

The function e^{-x} can be approximated by its McLaurin series expansion as follows (note the alternating + and −):

$$e^{-x} \approx 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots \pm \frac{x^n}{n!}$$

Alternatively, note that $e^{-x} = \frac{1}{e^x}$. Thus, e^{-x} can also be apporximated by 1 over the McLaurin series expansion of e^x. That is,

$$e^{-x} \approx \frac{1}{1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!}}$$

Approximate $e^{-2.75}$ using both approaches above for $n = 1, 2, 3, 4, 5$ and 6. Note, n is the degree of the polynomial not the number of terms. So here you use 2 terms, then 3 terms, ..., and finally 7 terms. Compare each approximation to the true value of $e^{-2.75} = 0.06392786...$, using the true relative error. What conclusions can you make about the two approaches? You may do this by hand with a calculator or program it using either MATLAB or Python.

DELIVERABLES: All your work. If programmed, copies of functions and results plus the corresponding .m or .py files.