MAE3456 – MEC3456 LAB 04

Due: 11:00PM (Sharp), Wednesday 13 May 2020 (Mid of Week 8)

This lab should be completed INDIVIDUALLY. Plagiarism will result in a mark of zero. Plagiarism includes letting others copy your work and using code you did not write yourself. Collaborating with others to discuss algorithms and details of MATLAB syntax and structures is acceptable (indeed encouraged), however you *MUST* write your own MATLAB code. All assignments will be checked using plagiarism-detecting software and similarities in submitted code will result in a human making a decision on whether the similarity constitutes plagiarism.

INSTRUCTIONS

Download **`template.zip`** from Moodle and update the M-Files named **`Lab04`**`_Q1a.m`**,** **`Lab04`**`_Q1b.m`**,** etc… with your Lab code. DO NOT rename the M-Files in the template and ONLY modify **`run_all.m to add in running scripts that answer any bonus questions I might ask`**. Once you have coded, check your solutions to the questions by running **`run_all.m`** and ensuring all questions are answered as required.

SUBMITTING YOUR ASSIGNMENT

Submit your assignment online using Moodle. You must include the following attachments:

A **ZIP** file (**NOT .rar or any other format**) named in the following way:

Surname_StudentID_Lab_04.zip (e.g. Rudman_23456789_Lab_04.zip)

The zip file should contain the following:

a. All MATLAB m-files for lab tasks: **run_all.m**, **LabNN.m, `Q1a.m`**, **`Q1b.m`**, etc…

b. Any additional MATLAB function files required by your code

c. All data files needed to run the code, **including any input data provided to you**

d. Any hand calculations or written responses asked for - scanned in as a ***SINGLE*** PDF file

YOUR ZIP FILE WILL BE DOWNLOADED FROM MOODLE AND ONLY THOSE FILES INCLUDED IN YOUR SUBMISSION WILL BE MARKED

We will extract (unzip) your ZIP file and mark the lab based on the output of **run_all.m** and any hand calculations or written responses. It is your responsibility to ensure that everything needed to run your solution is included in your ZIP file.

STRONG RECOMMENDATION: After uploading to Moodle, download your Lab to a NEW folder, extract it and ensure it runs as expected. ***CHECK YOUR PDF IS INCLUDED***. You can upload your submission as many times as you like ***BEFORE*** the submission deadline. After this time, it is not possible to resubmit your lab without incurring a late penalty. If you forget to include some of your code or your PDF you cannot include it later without penalty.

MARKING SCHEME

This lab is marked out of 80 and full marks is worth 8% of your total Unit mark for the semester. Code will be graded using the following criteria:

1) **`run_all.m`** produces results **automatically** (no additional user interaction needed except where asked explicitly – NOTE, I have included pause commands in run_all so that intermediate answers can easily be viewed by the demonstrators – please don't remove them)
2) Your code produces correct results (printed values, plots, etc…) and is well written.
3) Programming style, efficiency of algorithm and quality of output (figures, tables, written text ...)

ASSIGNMENT HELP

1) You can ask questions in the Discussion Forum on Moodle
2) Hints and additional instructions are provided as comments in the assignment template M-Files
3) Hints may also be provided during lectures
4) The questions have been split into sub-questions. It is important to understand how each sub-question contributes to the whole, but each sub-question is effectively a stand-alone task that does part of the problem. Each can be tackled individually.
5) I recommend you break down each sub-question into smaller parts too, and figure out what needs to be done step-by-step. Then you can begin to put things together again to complete the whole.
6) To make it clear what must be provided as part of the solution, I have used bold italics and a statement that (usually) starts with a verb (e.g. ***Write a function ..., Print the value...***, etc.)

QUESTION 1 [5 MARKS TOTAL]

Background

Many problems in engineering can be described using ODEs. For example, the drainage of a tapered funnel, such as shown in Figure 1 below, can be described using the ODE:

$$\frac{dy}{dt} = -\frac{\tan^2\theta}{y^{1.5}}\frac{d^2}{4}\sqrt{2g} \qquad \text{Eqn (1)}$$

where y is the height of the water level from outlet of the funnel, θ is the taper angle, g is gravitational acceleration and d is the diameter of the funnel outlet. At t = 0, the water is at a level of y_0 above the outlet (see Figure 1).

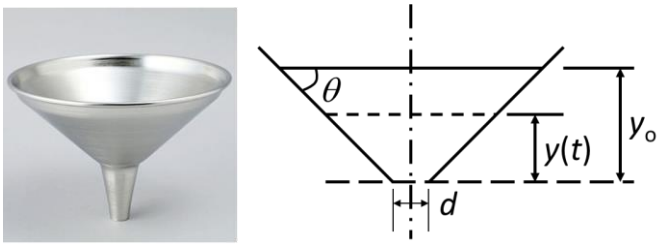


Figure 1: A typical funnel (left) and the schematic showing parameters defining the drainage flow (right).

Q1a) Using the Taylor series approach discussed in Workshop 13, derive a numerical scheme for this ODE (i.e. Eqn (1)) that has a leading order error term of h^3.

QUESTION 2 [10 MARKS TOTAL]

Q2a)

Write a MATLAB function that undertakes "n" time step of the RK4 method. Your function header MUST be given by:

```
[tnew,unew] = RK4(told,uold,RHS,dt,n)
```

Where the input parameters are:

- told – the time from the previous step
- uold – the value of the solution, u, from the previous step
- RHS – the right-hand-side function of the ODE
- dt – the stepsize
- n – the number of steps to undertake (the function MUST work for n=1)

And the output parameters are

- tnew – a vector containing all the time after completing n steps
- unew – a vector containing the value of the solution, u, after completing n steps

Q2b)

Let y_0 = 0.15 m, θ = 45°, d = 0.005 m and g = 9.81 ms^{-2}.

Modify the template **Lab_04_Q2b.m** to integrate the ODE in Eqn 1 from t = 0 to t = 120 s using stepsizes Δt = 40, 20, 10 and 5 s using the RK4 method.

Plot the solution $y(t)$ against time for each value of Δt; each represented using a different color line.

Also ***plot the exact solution*** of Eqn I in the same figure. The exact solution is given by Eqn 2:

$$y(t) = \left[y_0^{2.5} - \frac{5d^2 \tan^2\theta}{8} \left(\sqrt{2g}\right) t \right]^{0.4} \qquad \text{Eqn (2)}$$

(**Note:** Use a solid curve to represent the exact solution. Use sufficient points to ensure a smooth curve is obtained for the exact solution.)

Write a few sentences to the command window describing what you observe when you change the time step AND why you think that is the case.

Q2c)

Continue to modify the template **Lab_04_Q2b.m** and use your RK4 function with whichever timestep you like and estimate the time it takes for the water level inside the funnel to reach 0.001 m, i.e. at what value of time is $y(t)$ = 0.001?

Print to the command window your estimate of the time taken for the water level inside the funnel to reach 0.001 m and write a few sentences comparing the corresponding value you can calculate using the exact solution (Eq 2).

QUESTION 3 [20 MARKS TOTAL]

Background

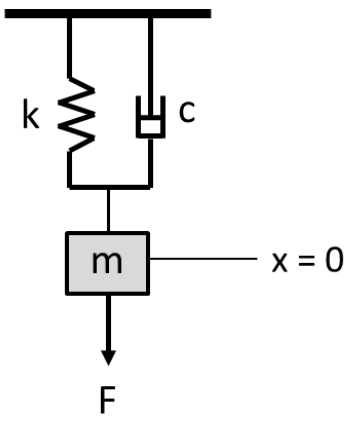


Figure 2: The mass-spring-damper system

Figure 2 shows the mass-spring-damper system, where a block of mass m is connected to a spring with spring constant k and a damper with damping constant c. F represents the force acting on the mass. The mass is pulled downwards with a sufficient force such that at time = 0, $x = x_0$ and $dx/dt = 0$. The force is then removed and the mass released. In the absence of force F, the mass-spring-damper system becomes a damped-harmonic oscillator and the position of the mass with respect to time can be described using the following second order ODE:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0 \qquad \text{Eqn (3)}$$

Q3a)

Reduce Eqn (3) to a system of first order ODEs and write down the initial conditions corresponding to the system of first order ODEs. (**Note**: Your solution to **Q3a** must be included as part of the PDF document from **Q1**.)

Q3b)

Solve Eqn (3) by hand using the RK4 method for $m = 1$, $k = 0.5$, $c = 1.5$ and $x_0 = 1.5$ using $\Delta t = 0.1$ s from $t = 0$ to $t = 0.1$ s. Show clearly all your working (include this answer in your PDF).

Q3c)

Modify your RK4 function from **Q2**a so that it will now solve a system of 1st order ODEs for n time steps. The function header must be:

```
[tnew,unew] = RK4_sys(told,uold,RHS,dt,n)
```

Where the input parameters are:

- told – the time from the previous step

- uold – a column vector of the solution, u, from the previous step
- RHS – a column vector that represents the RHS functions of the ODE system (in general, it will be a function of t and **u**)
- dt – the step size
- n – the number of steps to undertake (the function MUST work for $n=1$)

And the output parameters are

- tnew – a vector containing all the time after completing *n* steps
- unew – a matrix with 2 rows and "M" columns. Each column corresponds to the solution of the dependent variables at the time specified in the vector t

Q3d)

Modify the template **Lab_04_Q3d.m** and ***solve Eqn (3)*** for $x(t)$ from t = 0 to t = 30 s using the RK4 method for the two cases below:

Case 1: m = 1, k = 0.5, c = 1.5 and x_o = 1.5.

Case 2: m = 1, k = 1.5, c = 0.5 and x_o = 1.5.

For each case, perform the simulations using Δt = 1, 0.5 and 0.1 s.

Plot the values of $x(t)$ against time for Cases 1 and 2 for each of the value of Δt considered. You should have three figures, each representing the results obtained using Δt = 1, 0.5 and 0.1 s. Use blue to represent Case 1 and red to represent Case 2.

Write a few sentences to the command window describing the movement of the mass with time for Case 1 and Case 2, **AND** explain why that is the case. Compare also the plots for the different Δt **AND** explain why you think you observe what you do.

QUESTION 4 [20 MARKS TOTAL]

Suppose that there is an unsteady external force acting on the mass in **Figure 2**. The ODE in Eq (2) then becomes

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = F(t) \qquad \text{Eqn (4)}$$

One of the more common type of external force is a periodic force of the form $F(t) = F_o\cos(\omega t)$, which induces forced vibrations in the mass-spring-damper system.

Q4a)

Referring to the discussion on adaptive step size control for the RK4 method in Workshop 15 (Lecture 16), ***write an adaptive step size controlled RK4 function*** to solve a system of 1st order ordinary differential equations. HERE you will need to integrate from the beginning to the end of the time range, not just "n" steps. (In fact, the number of steps is not known before calling this function.) (Make sure you include the Richardson extrapolation step as well). The technique should have the following header file:

```
function [t,u,nstep,hstep] = RK4A(trange,uinit,f,tol)
```

where the input parameters are:

- f – a column vector function that describes the right-hand-side of the 1st order system of ODE's
- trange – a vector with the start and end times [t0, tend]
- uinit – the column vector of initial values for each dependent variable
- tol – the maximum allowable local error (for the solution variable, not its derivative)

The outputs from the function are:

- t – a vector of the times at which a solution estimate was accepted (it will have length "M" that is unknown until the code is run)
- u – a matrix with 2 rows and "M" columns. Each column corresponds to the solution of the dependent variables at the time specified in the vector t.
- nstep – the total number of (accepted) steps taken by the adaptive RK4A solver
- hstep – the vector of step sizes taken by the adaptive RK4A solver

Note: You must use your RK4_sys function from **Q3a** to perform the individual RK4 steps inside RK4A. As you will have two solutions from your variable u, you should take the solution for x when determining the optimum stepsize.

Q4b)

Modify the template **Lab_04_Q4b.**m to integrate Eqn (4) using the adaptive RK4 scheme that you wrote in **Q4a**. Set $F_o = 1$ and $\omega = 2.5$ and let all the values of m, k, c and x_0 be the same as that in Case 1 in **Q3d**. ***Integrate from*** t = 0 to t = 30 for tol values of 10^{-4}, 10^{-5}, 10^{-6} and 10^{-7}.

Print out to the command window the number of time steps taken by the solver.

Using the **plotyy** command (not recommended by MATLAB, but recommended by your lecturer), **plot *x*(*t*)** against time on the left-hand-side axis and the stepsize against time on the right-hand-side axis.

Write a few sentences to the command window describing what you observe from the plot.

QUESTION 5 [25 MARKS TOTAL]

Consider the ODE in Eqn (5) below:

$$\frac{dy}{dt} = -3y + 6e^{-t} \qquad y(0) = 1.0 \qquad \text{Eqn (5)}$$

Note: The solutions to **Q5a** and **Q5b** must be presented as part of the PDF document.

Q5a)

Using the ODE in Eq (5), ***derive the stability criteria*** of the explicit Euler method. Show clearly all your workings.

Q5b) ***Repeat*** **Q5a** for the implicit Euler method.

Q5c)

Write a MATLAB function that implements the explicit Euler method. Your function header must be:

```
[t,u] = ExpEuler(tspan,uinit,rhs,dt)
```

Where the input parameters are

- tspan is a vector of two containing the starting and ending time ([t0, tend])
- uinit is the initial value of the solution (u0)
- RHS is the right-hand-side function of the ODE
- dt is the stepsize

And the outputs returned by the function are

- t is a vector containing all the time levels at which the solution u was calculated
- u is a vector containing the solution for u evaluated at the times specified in t

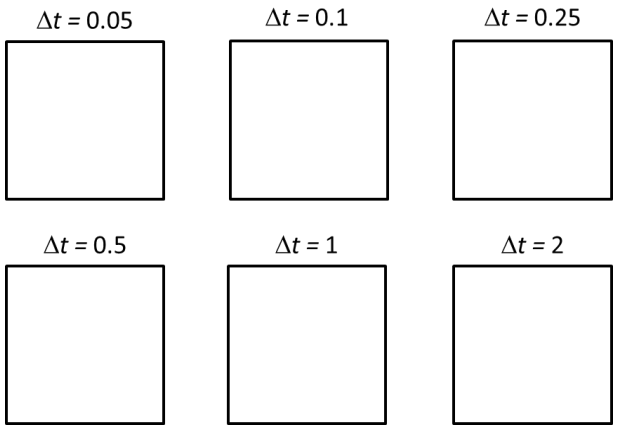
Q5d)

Modify the template **Lab_04_Q5d.m** and ***solve the ODE*** in Eqn (5) from t = 0 to t = 20 for Δt = 0.05, 0.1, 0.25, 0.5, 1 and 2.

The exact solution of the ODE above is given by:

$$y(t) = -2e^{-3t} + 3e^{-t} \qquad \text{Eqn (6)}$$

Using the **subplot** command**, *plot the solutions*** obtained using the explicit Euler method and the exact solution (Eqn (6)) against time in a 2 x 3 grid, such as shown below. Each subfigure represents a different Δt value considered. Use blue symbols to represent the Euler solution and a red line to represent the exact solution.



Write a few sentences to the command window describing what you observe. Explain how the selection of Δ*t* affect the solution. HINT: refer to the stability criteria that you have derived in **Q5a.**

Q5e)

Write a MATLAB function that implements the implicit Euler method. Your function header must be:

```
[t,u] = ImpEuler(tspan,uinit,dt)
```

The input parameters and return values have the same meaning as in **Q5c**.

Note: that the right hand side function is not passed into the ImpEuler function and must be hardcoded into your function – one of the downsides of the method.

Q5f)

Repeat Q5d using the implicit Euler method by ***modifying*** the template **Lab_04_Q5e.m**.

Poor Programming Practices [-10 Marks]

(Includes, but is not limited to, poor coding style or insufficient comments or unlabeled figures, etc.)

(END OF LAB)