EN40: Dynamics and Vibrations

Homework 1:
Mupad and MATLAB practice
Due 12:00 noon Friday February 3 (online submission)

School of Engineering
Brown University

- Your solution to this homework should consist of two files:
    1. A commented MUPAD .mw file
    2. A commented MATLAB .m file
- Please submit the assignment electronically on the EN40 canvas website. You can log into canvas at http://brown.edu/it/canvas/ (the login link is near the top right of the page). Instructions for uploading are at https://sites.google.com/a/brown.edu/teaching-with-technology/canvas/student-help-site/assignments

Part 1: Use Mupad to solve the following math problems. Be sure to save your work frequently

1. Use Mupad to find all the solutions to the simultaneous equations

$$x^2 + 2y^2 = 18 \qquad x + y - xy = 1$$

2. Plot the function $P(x,s) = \dfrac{1}{x(1-x)}\exp\left(\dfrac{-[\ \log(\ x/(1-x)\ )\ ]^2}{s^2}\right)$ in the range $0.01 < x < 0.99$, for $s = 0.5, s = 1, s = 2, s = 2.5$ (on the same plot). (*P(t)* is the 'logistic-normal' distribution function). Problem 2 from HW1 2016 has an example you can follow.

3. Evaluate the integral

$$\int_0^1 P(x,s)dx$$

To simplify the answer, insert the line

```
assume(s>0):
```

just before you do the integral

4. Find the maximum value of the function $x^2/\left(x^3 + a^3\right)$ (see Mupad tutorial for an example!)

5. The variation of pressure p with altitude z above the earth's surface is governed by the differential equation

$$\frac{dp}{dz} = -\frac{g}{R}\frac{p}{T(z)}$$

where g is the gravitational acceleration, R is the individual gas constant for air, and *T(z)* is the temperature (in Kelvin). In the troposphere (below about 30000 ft), the temperature decreases approximately linearly with altitude, so

$$T(z) = T_S - \lambda z$$

where T_S is the temperature at the surface, and $\lambda \approx$ 6^0C/km is the 'lapse rate.'

Use MUPAD to solve the differential equation with boundary condition $p = p_s$ at *z=0*, (Mupad gives a funny looking answer – can you simplify it by hand?).

Federal Aviation Regulations (see Table I) require altimeters to be accurate to within +/- 80 feet at 10000 feet. Use Mupad to calculate the difference in pressure between 10080 and 9920 feet, and hence determine the maximum allowable error (in %) of the pressure sensor. Use the following data:

- Gravitational acceleration g= 9.81 m/s^2
- Pressure at sea level 101325 N/m^2
- Temperature at sea level $T_S = 298K$
- Gas constant for air $R = 287$ J kg^{-1}K^{-1}
- Lapse rate $\lambda \approx$ 6^{0}C/km is the 'lapse rate.'

You can substitute values for the variables using , eg:

```
 subs(expr,{R=287,Ts=298,ps=101325}) )
```

Part 2: Please solve the remaining problems using MATLAB (write your code in a matlab .m file). You should make your MATLAB (.m) file a function, so that when the file is executed, it will solve all the homework problems. For example:

```
   function EN40_Homework1_2016
          code that will solve problems 6-11
   end
   function edges = detect_edges(image)
   … Code
   End
   … Other functions for the ODE problems go here… .
```

(You might find the solutions to homework 1, 2016 helpful, if you get stuck)

6. Create a vector t of 500 equally spaced points between 0 and 2π

7. Using the solution to problem 6, create two vectors x and y that contain values of the function

$$x = -2\sin t + \sin 2t$$
$$y = 2\cos t - \cos 2t$$

Hence, plot a graph of y-v-x. (You get some credit for making your plot look nice!)

8. By using two nested loops and a conditional statement (you can use a more efficient method if you prefer), construct a 300x300 matrix z(i,j) with

$$z(i,j) = \begin{cases} 1 & \sin(4\pi i/300)\sin(4\pi j/300) > 0 \\ 0 & \sin(4\pi i/300)\sin(4\pi j/300) < 0 \end{cases}$$

To visualize the matrix, use the matlab statements

```
figure
imshow(z)
```

To understand what you just did, note that:

- A grayscale image in MATLAB is stored as a matrix z(i,j). The value of z(i,j) specifies the color of the pixel that is i rows below, and j columns to the right, of the top left hand corner of the image. z=1 corresponds to a white pixel, z=0 corresponds to a black one. Values in between are gray.
- The 'imshow' function displays an image.
- In your matrix, z is either 0 or 1, so you see a checkerboard pattern.

9. In this problem you will write a MATLAB function to detect edges in an image. The function should have the following format

```
function edges = detect_edges(image)
… Code
end
```

Here, image is a matrix storing a gray-scale image, and edges is a new matrix that displays (as curves) all the edges (boundaries between black and white regions) in the original image. You can use the following method to detect the edges:

For *i=2:299* and *j=2:299* (use two nested loops) compute

$$\text{edges(i,j)}=\begin{cases}0, & \left(\text{image(i+1,j)-image(i-1,j)}\right)^2+\left(\text{image(i,j+1)-image(i,j-1)}\right)^2>0.5\\ 1, & \left(\text{image(i+1,j)-image(i-1,j)}\right)^2+\left(\text{image(i,j+1)-image(i,j-1)}\right)^2\le 0.5\end{cases}$$

(you can improve this code – eg make it work for an image of arbitrary size). Test your function by using it to find the edges in the image in the checkerboard created in problem 8 (eg

```
edges_of_z = detect_edges(z)
figure
imshow(edges_of_z)
```

Optional: not for credit: you could explore MATLAB's built-in edge detection functions . You could also test your code on Matlab's test image, which you can load and display with

```
I = imread('circuit.tif');
imshow(I)
```

Your code won't work well on this image, but you could try to think of ways to improve it.

10. The 'Ermentrout-Koppel model' SIAM-J.-Appl.-Math **46** (1986), 233. is a simple differential equation that describes neuron bursting. The simplest form of the differential equation is

$$\frac{d\theta}{dt}=1-\cos\theta+(1+\cos\theta)A\sin\omega t$$

Here θ is a variable that characterizes the internal state of the neuron (a concentration or membrane potential), and $A\sin\omega t$ describes some form external stimulus (eg from a second neuron). The neuron 'fires' whenever $\cos\theta=-1$.

Write a MATLAB function that will use this formula to calculate a value for $d\theta/dt$ (the output variable of the function) given values for t,θ,A (the input variables). Hence, use the MATLAB 'ode45' function to solve (numerically) the equation. Use the following values for parameters: $A=0.01$ $\omega=0.01$ rad/s and use $\theta=0.5$ at time *t=0*. Obtain a solution for $0<t<5000$. Display your solution by plotting a graph of cos(θ) (**NB: not** θ **!)** as a function of time.

You will find that the solution looks sensible for about 100 sec or so but after than the predictions are very strange. If you see computer predictions like this in professional life investigate them carefully – the chances are they are garbage!

For this example the solution can be improved by setting a higher accuracy in the ode45 function. To do this, change the 'relative tolerance' parameter in the ode solver, by inserting the line

```
options = odeset('RelTol',0.000001);
```

just before the line that uses ode45. You can then use ode45 with the 'options' argument, eg:

```
[times,sols] = ode45(@(t,q) neuron(t,q,omega,A),time_range,init_q,options);
```

(or something equivalent – the exact form will depend on how you wrote your script)

As a solution to this problem make your MATLAB script plot both the incorrect and correct graphs (i.e. use ode45 twice).

11. Cells have a mysterious internal 'chemical clock' that governs their repeated cycles of growth and mitosis (division). The famous 'Goldbeter model' Proc. Natl. Acad. Sci. USA, Vol. 88, p 9107, (1991) was one of the earliest explanations for how the chemical reactions involved in cell divisions can lead to regular periodic cycles. The goal of this problem is to solve a simplified version the differential equations governing the relevant chemical reactions in MATLAB.

Three chemicals play a role in the process: 'Cyclin' proteins (cell division is triggered when the concentration of cyclin is large); a 'kinase' which forms a complex with cyclin, and a 'protease,' which degrades proteins. Their fractional concentrations (which vary between 0 and 1) can be denoted by C (for cyclin), K (for kinase) and P (for protease). They are governed by

$$\begin{aligned}
\frac{dC}{dt} &= k_1(1-C) - k_2 P \frac{C}{b+C} \\
\frac{dK}{dt} &= k_3 \frac{C}{(1+C)}\left(\frac{1-K}{b+(1-K)}\right) - k_4 \frac{K}{b+K} \\
\frac{dP}{dt} &= k_4 K \frac{1-P}{b+(1-P)} - k_5 \frac{P}{b+P}
\end{aligned}$$

Write a function that computes the vector of time derivatives $d\mathbf{w}/dt = [dC/dt; dK/dt; dP/dt]$ given a value of t and the current values of $\mathbf{w} = [C; K; P]$ and the values of the parameters $b,\ k_1, k_2, k_3, k_4, k_5$.

Hence, use the ode45 function in MATLAB to integrate (approximately) the system of equations with respect to time, and plot a graph of C, K, P as a function of time. Use the following values for parameters: $b = 0.005$ (b has no units), $k_1 = 0.025,\ k_2 = 0.25,\ k_3 = 3,\ k_4 = 1,\ k_5 = 0.5$ (all in hrs). Run the simulation for 100 hours, with initial conditions $C = 0, K = P = 0.5$.

If your code runs correctly, you should see steady oscillations in the concentrations. The nature of the solutions to the ODE are rather sensitive to the values of $b,\ k_1, k_2, k_3, k_4, k_5$, however - the oscillations may not occur if other values are used.