School of Engineering
Brown University

EN40: Dynamics and Vibrations

Homework 1:
MATLAB practice
Due 12:00 noon Friday May 28 (online submission)

- Your solution to this homework should consist of two files:
  1. A commented MATLAB Live Script stored as a .mlx file
  2. A commented MATLAB function stored as a .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). You can find some instructions on Canvas use (from CIT) at https://ithelp.brown.edu/kb/articles/canvas-for-students
- (You might find the solutions to homework 1, 2009-2020 helpful, if you get stuck – you could use the 2020 HW as a template, if you like.)

Part 1: Write a Matlab 'Live Script' to solve the following problems.

1. Find all the solutions to the simultaneous equations

$$\exp\{(x/2)^2-(y/4)^2+(z/6)^2)\}=9 \qquad \log(x+y-zx)=0 \qquad x+y+z=1$$

(here 'exp' denotes the exponential function, i.e. $\exp(x)=e^x$ and 'log' denotes the natural log. You can type these into MATLAB as exp(x) and log(x), respectively)

2. Plot the function

$$P(x,\alpha,\beta,\lambda)=\left[\alpha e^{\beta x}+\lambda\right]\exp\left\{-\lambda x-\frac{\alpha}{\beta}\left(e^{\beta x}-1\right)\right\}$$

in the range $0<x<100$, for $(\alpha=5\times10^{-5},\ \lambda=0)$, $(\alpha=5\times10^{-4},\ \lambda=0)$, $(\alpha=5\times10^{-5},\ \lambda=0.001)$ and $\beta=0.085$ (on the same figure). (P is the 'Gompertz-Makeham' distribution, which is used to describe human mortality – x represents age, and $dN=NP(x,\alpha,\beta,\lambda)dx$ is the number of people in a population of N people who perish between ages x and $x+dx$). A few points of credit are awarded for making the plot look nice.

3. Evaluate the integral

$$S(x,\alpha,\beta,\lambda)=1-\int_0^x P(\xi,\alpha,\beta,\lambda)d\xi$$

(S is the fraction of a population who will survive to age x). Hence, plot $S(x,\alpha,\beta,\lambda)$ in the range $0<x<100$, for $(\alpha=5\times10^{-5},\ \lambda=0)$, $(\alpha=5\times10^{-4},\ \lambda=0)$, $(\alpha=5\times10^{-5},\ \lambda=0.001)$ and $\beta=0.085$ (on the same figure). You can compare the plots with data in the CDC life tables, if you are curious. A few points of credit are awarded for making the plot look nice.

4. Find a formula for the value of x that maximizes $P(x,\alpha,\beta,\lambda)$ (you can find potential maxima by differentiating P with respect to x; then solving $dP/dx = 0$. Depending on your version of MATLAB, you may get a warning message saying the solution is valid only under certain conditions. MATLAB will give you more than one solution – you can work out which one is a maximum by looking at the results for the special case $\lambda = 0$.

5. Find a formula (i.e. an equation) for the maximum value of P

6. The differential equation

$$\frac{dI}{dt} = (\lambda - \mu)I - \lambda I^2$$

predicts the fraction of a population who are infected by a contagious disease such as strep throat, for which infection which does not confer immunity. The constants λ and μ quantify the rates of infection and recovery, respectively.

6.1 Use the 'dsolve' function to solve the differential equation for I as a function of time. Use the initial condition $I(t) = I_0 \quad t = 0$.

6.2 Plot a graph showing $I(t)$ for an initial infected fraction $I_0 = 0.05$ and $\lambda = 1 \ day^{-1}$, for $0 < t < 30$ days, for the following values of $\mu = 0.1, 0.5, 0.85$

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_2020
        code that will solve problems 7-11
end
function [time_vals,sol_vals] = mynewode45(f,time_int,initial_w)
   Code for function you write in problem 10…
end
Code for function you write for problem 11….
```

7. Using a loop, or dot notation, or the 'linspace' function, create a vector t of 501 equally spaced points between 0 and 2π

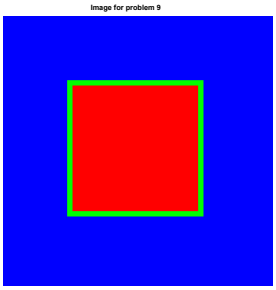
8. Using the solution to problem 7, along with a loop, create vectors x and y that contain values of the function

$$x = \cos(t)\left|\cos(t)\right|^{a-1} \qquad y = \sin(t)\left|\sin(t)\right|^{a-1}$$

for $0 < t < 2\pi$ and $a=5$. In MATLAB you can take the absolute value of a variable using, eg , answer = abs(variable). Hence, plot a graph of y (on the vertical axis of the plot) against x (on the horizontal axis).

Repeat the calculation and plot curves for ,$a=2$,$a=1$. Show the curves for all three values of a on the same plot. A few points of credit are awarded for making the plot look nice.

9. A color image in MATLAB is stored as a (3 dimensional) matrix M. Each triplet M(i,j,1), M(i,j,2), M(i,j,3) in the matrix specifies the color of a pixel in the image., using the 'rgb' convention. The indices *i,j* specify the coordinates of the pixel, with the *i=j=1* at the top left corner of the image, *i* (the row of the matrix) vertically downwards, and *j* (the column) horizontally. The three numbers specify the red, blue and green content of the color. Confusingly, M can be either an integer valued matrix or a floating point number. If M is an integer, then each number varies from 0 (the minimum color content) to 255 (the max color content), while if M is a floating point matrix, then each number varies between 0 and 1, with 0 the min color, and 1 the max. So, for example, if M(i,j,1)=1.0, M(i,j,2)=0.0, M(i,j,3)=0.0, the pixel at coordinaes (i,j) is colored red. This is the same as M(i,j,1)=255, M(i,j,2)=0, M(i,j,3)=0



You can display a matrix using the 'imshow' command, e.g.

```
imshow(M,'Initialmagnification',200)
```

(The 'Initialmagnification' is optional, it just makes the figure larger)

9.1 Create a 101x101x3 matrix that will produce the image shown in the picture, and display it. The green border is 2 pixels wide; the top left corner is at (25,25) and bottom right is at (75,75). Some commands you could use include:

(a) M = zeros(100,100,3) creates a 100x100x3 matrix with 0 in every entry of the matrix (this shows up as black)

(b) M(1:10,1:10,1) = 1 (eg) will set the top 10x10 block of a matrix to red (try it and see what happens; also try changing the 1:10 to get a feel for how the indexing works)

9.2 Using the procedure outlined below, convert the 'blueno' picture shown into 'pinkno' (does this change the gender or political affiliation of blueno?)

(i) Download the file called 'blueno.jpg' from the HW page of the course website. Make sure the file is in the same directory as your matlab script.

(ii) Load the file into your MATLAB script with

```
blueno = imread('blueno.jpg');
```

(iii) Display the original image with 'imshow'

(iv) Convert the image to the 'hsv' color scheme using

```
bluenohsv = rgb2hsv(blueno);
```

(v) Find the number of rows and columns in the image (see the tutorial for how to do this)

(vi) Using a pair of nested loops and a conditional statement, check the 'hue' value `bluenohsv(i,j,1)` of each pixel in the image. If the hue value lies between 0.44 and 0.55, set the 'r' value of `pinkno(i,j,1)` to 242

(vii) Use 'imshow' to display the modified image.

Optional: create a composite image with 'blueno/pinkno' next to each other as shown below

Blueno and pinkno

10. In this problem you will write your own version of the MATLAB 'ode45' differential equation solver. Before starting the problem you might find it helpful to read through Sect 11.8 of the written matlab tutorial, which contains a similar example code.

The goal of the solver is to estimate the solution to a differential equation of the form

$$\frac{dw}{dt} = f(t, w)$$

over a time interval $t_1 < t < t_2$, given the initial condition $w = w_1$ at time t_1. The following algorithm (which is more stable than the method described in the tutorial) will be used:

(1) Start with the known solution $w = w_1$ at time t_1

(2) Let N be the number of points in the solution, and define $\Delta t = (t_2 - t_1)/(N-1)$

(3) For each $i = 2...N$, calculate, and store in a vector (see the example in the tutorial)

$$t_i = t_{i-1} + \Delta t$$

$$w_i = w_{i-1} + \Delta t f\left(t_{i-1} + \Delta t/2, w_{i-1} + \Delta t f[t_{i-1}, w_{i-1}]\right)$$

Implement this algorithm by writing a function with the following properties:

Given: (i) A function handle (see the tutorial); (ii) a time interval in a vector $[t_0, t_1]$ and (iii) The initial value w_1

Calculate (i) a vector of time values t_i and (ii) a vector of solution values w_i

You will find that you only need to make a few small modifications to the example in the tutorial.

Test your function by using it to solve problem 6 (try *N=500*). Plot the solution along with that returned by ode45.

We suggest solving this problem by breaking it up into small steps, so you can check each step works before moving on:

(1) Solve Problem 6, but instead of using 'dsolve' in a Live Script, solve it using the built-in MATLAB ode45. This should be straightforward – just follow the examples in the video tutorial.
(2) Now go to Section 11.8 of the written matlab tutorial, and cut and paste the example code for the simple ode solver example called 'my_ode45' into your homework. Then replace the call to 'ode45' in the code you wrote for part (1) with 'my_ode45.' Run the code – it should produce the same solution as in part (1) above.
(3) Finally, modify the 'my_ode45' function to implement the more stable algorithm given in the problem. If you figure out how the loop works in the written matlab tutorial, this will just need 1 additional line of code, and a small modification to one line of existing code. Alternatively, you could re-write the code to start the loop at 2 and end at *N,* following the recipe given in the problem. You'd have to change some lines of code before the loop as well as the loop itself to do this.

Optional: write your code so that it will solve simultaneous differential equations (such as those in problem 11). In this case the solution should return a vector of time values, and a matrix of solution values, in which each column stores the value of one of the unknowns at the times listed in the vector.

11. This publication uses following differential equations to predict the world's population p and GDP g as a function of time

$$\frac{dp}{dt} = \alpha_1 p + \alpha_2 gp + \alpha_3 g$$
$$\frac{dg}{dt} = \beta_1 g + \beta_2 pg$$

where $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2$ are constants.

11.1 Write a function that computes the vector of time derivatives $d\mathbf{w}/dt = [dp/dt; dg/dt]$ given a value of t and the current values of $\mathbf{w} = [p; g]$ and the values of $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2$.

11.2 Hence, use the ode45 function in MATLAB to integrate (approximately) the system of equations with respect to time, and plot a graph of p, g as a function of time (put the curves on the same plot, but use a separate y axis for p and g. See the MATLAB 'help' for axisyy to learn how to create a second axis).

Use the following values for parameters:

$\alpha_1 = 0.5\%/\,year$ $\alpha_2 = -4\times10^{-4}\,/\,(Trillion US\$\ year)$ $\alpha_3 = 5.2\times10^{-3}\,Billion\ \ people\,/\,(Trillion US\$\ year)$

$\beta_1 = 3.1\%/\,year$ $\beta_2 = -10^{-13}\,/\,(Billion\ people\ \ year)$

(Of course the values given in % should be entered as, eg, $\alpha_1 = 0.005/\,year$, eg)

Run the simulation from $1850 < t < 2150$ (in years), with initial conditions $p = 1.15\ \ Billion\ \ people$ $g = 0.21\ Trillion US\$$ in 1850. Keep the units of p and g as billions of people and trillion us$ to be consistent with the units for the constants…..

Optional: If you solved the optional part of problem 10, repeat the calculation with your own ODE solver.