School of Engineering
Brown University

EN40: Dynamics and Vibrations

Homework 1:
MATLAB practice
Due 12:00 noon Friday January 31 (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, 2009-2019 helpful, if you get stuck – you could use the 2019 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

$$\exp(x^2+y^2-z^2)=9 \qquad x+xy-z=1 \qquad x+y+z=1$$

2. Plot the function

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

in the range $0.01<x<3$, for $\sigma=1$ and $\alpha=1.1$, $\alpha=2$, $\alpha=5$ (on the same plot). (P is one of the 'Pareto' distribution functions, which has various applications. For example, with $\alpha=1.161$ it describes income distributions satisfying the 80%-20% law) A few points of credit are awarded for making the plot look nice.

3. Evaluate the integral

$$\int_0^x P(\xi,\sigma,\alpha)d\xi$$

Assume that $\sigma>0$ and σ is real, and that $\alpha>1$.

4. Find the value of x that maximizes $Q(x)=(x^2-1)/(x^4+7)$ (you can find potential maxima by differentiating Q with respect to x; then solving $dQ/dx=0$. MATLAB will give you three solutions; two of them will be maxima. You can plot Q if it's not obvious which of the solutions are maxima.

5. Find the maximum value of Q

6. Thiele's 'Life insurance equation'

$$\frac{dV}{dt} = P + \eta V - \mu(S - V)$$

predicts the amount of capital $V(t)$ (in $ per client) that an insurer must keep in reserve in order to be able to pay out all the claims that are likely to result from a group of their clients. Here, *P* is the $/year collected in premiums, η is the interest rate earned on the reserve, *S* is the benefit that will be paid out in the event of the clients death, and μ is the 'force of mortality' – a measure of the probability that the client will die (μ is actually a function of the client's age, but we take it to be constant to keep things simple).

6.1 Use the 'dsolve' function to solve the differential equation for *V* as a function of time. Use the initial condition $V(t) = V_0 \quad t = 0$, where V_0 is a constant to be determined.

6.2 The insurance policy has a term t_0 (if the client dies after time t_0 they will not collect a benefit). This means that $V(t_0) = 0$. Use this condition to solve for the unknown constant V_0 , and substitute the result into your formula for *V(t)* from 6.1. (The result should be a formula for *V* in terms of t_0)

6.2 Plot a graph showing $V(t)$ for a life insurance policy for $1000000 with a $t_0 = 10$ year term, an interest rate $\eta = 4\%$, premium $P = 800\$ / year$ and $\mu = 0.002, 0.02, 0.034$ (typical force of mortality at ages 30, 40 and 60, respectively). Show the 3 cases on the same plot, with a legend identifying the curves. Use a time range of 0<*t*<10 years for the 'x' axis.

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 package = hide_payload(cover,payload)
   Code…
end
function payload = recover_payload(package,cover)
end
```

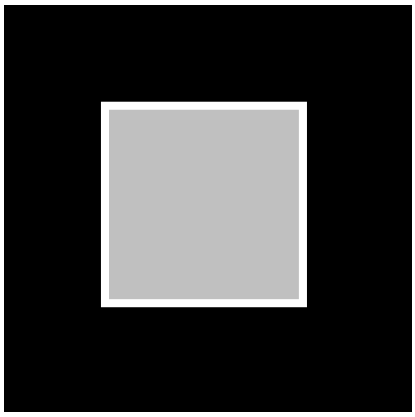
7. Using a loop, or dot notation, or the 'linspace' function, create a vector t of 501 equally spaced points between 0 and 2π

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

$$x = \begin{cases} \log(t+2)\sin(t) & 0 < t < \pi \\ \log(t+2-\pi)\sin(t) & \pi < t < 2\pi \end{cases} \qquad y = \begin{cases} \log(t+2)\cos(t) & 0 < t < \pi \\ \log(t+2-\pi)\cos(t-\pi) & \pi < t < 2\pi \end{cases}$$

(log denotes the natural log; there are of course other better ways to construct the vectors but the point of the problem is to figure out how to do loops/conditionals so please use those…). Hence, plot a graph of *y* (vertical axis) against *x* (horizontal axis). A few points of credit are awarded for making the plots look nice.

9. A grayscale image in MATLAB is stored as a matrix M. Each entry in the matrix specifies the color of a pixel in the image. By default, if M(i,j)=1 the pixel is white; if M(i,j)=0 the pixel is black, and if 0<M(i,j)<1 the pixel is gray. The indices i,j identify the position of the pixel, so for example M(1,1) is the pixel at the top left corner of the image. M(10,1) is the pixel 10 rows down on the left hand side of the image; and M(1,10) is the pixel 10 columns to the right at the top of the image.



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)

If you have a matrix with a larger range of values (eg 0<M(i,j)<100) you can display the matrix using

```
imshow(img,'DisplayRange',[0, 100],'InitialMagnification',400)
```

Create a 101x101 matrix that will produce the image shown in the picture, and display it. The white border is 2 pixels wide; the top left corner is at (25,25) and bottom right is at (75,75). The gray square at the center has intensity 0.75. Some commands you could use include:

(a) M = zeros(100) creates a 100x100 matrix with 0 in every entry of the matrix

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

You should be able to create your image with just 3 lines of script.

10. Images often conceal a watermark identifying their source, or can hide concealed messages as a form of steganography. The MATLAB default image is an example. You can display the default image using

```
ii = image();
imshow(ii.CData,'DisplayRange', ...
[min(ii.CData(:)),max(ii.CData(:))], ...
       ,'InitialMagnification',400)
```

To see the data hidden in the image, cut and paste the code below into an m file and run it (you don't need to submit this with your homework)

```
defImage = pow2(get(0,'DefaultImageCData'),47);
for shift = 10:63
    img = bitshift(defImage,shift);
    imshow(img,'DisplayRange',[min(img(:))...
                max(img(:))],'InitialMagnification',400)
    title(num2str(shift));
    pause(1)
end
```

Write two MATLAB functions of your own to hide and reveal a message or image inside another image. You can use any method you like to hide the image or message. Try to come up with your own idea, but if you get stuck we can suggest a method in recitation section or office hrs.

The first function should have the form

```
function package = hide_payload(cover,payload)
% Function to hide a message or image inside another image
% cover: the raw cover image (stored as a matrix, see problem 9)
% payload: the message or image to be hidden
% package: the modified image that includes the payload
% Write your code below
end
```

The second function should have the form

```
function payload = recover_payload(package,cover)
% Function to recover a message or image embedded inside another image
% cover: the raw cover image
% package: the modified cover image that includes the payload
% payload: the recovered message or image
% Write your code below
end
```

Test your functions with the code below (in your main HW function)

```
% You need to download the files from the HW website
cover = imread('cover_image.jpg'); % you need to download the file
payload = imread('payload_image.jpg');
package = hide_image(cover,payload);
figure
imshow(package)
title('Package image')
payload = recover_image(package,cover);
figure
imshow(payload)
title('Recovered payload image')
```

These MATLAB functions might be useful:

- `img = imread('filename.jpg');` will read an image from a file;
- new_img = double(img); will convert an image that is stored as matrix of integers to one that is stored as a matrix of floating point numbers. This might be helpful for the sample images we provided, which are stored as integers.
- new_img = img/max(max(img)); will convert a matrix that has values 0<img<some number to a new one that has values 0<new_img<1. This will be helpful for the images provided, which have values between 0 and 255. See MATLAB 'help' to understand how the 'max()' function works on matrices – it's a bit strange.
- `imggray = rgb2gray(img);` will convert a color image to grayscale (not needed for the sample images we provided – they are already greyscale. This is only for fancy stuff)
- `new_img = imresize(img,scale_factor);` will change the size of an image (not needed for the images we provided – they are already the same size)

The graders will test your code with two 'default' image files cover_image.jpg and payload_image.jpg from the HW website (the images happen to have exactly the same number of rows and columns, and are both grayscale images, to keep things simple) so make sure your code will work with these when you submit your HW. But you can test your code with other images or messages too (and if you think this

homework is a bit too elementary you can (a) find a way to handle cover and payload images with different sizes; (b) handle both grayscale and color images).

11. The 'Brusselator' is a pair of ordinary differential equations that describe a cyclic autocatalitic chemical reaction of four species A,B,C,D in solution. Two of them (A and B) have constant concentrations n_A, n_B, while the remaining two (C and D) have concentrations n_C, n_D that vary with time, according to

$$\frac{dn_C}{dt} = k_1 n_A + k_2 n_C^2 n_D - k_3 n_B n_C - k_4 n_C$$

$$\frac{dn_D}{dt} = k_3 n_B n_C - k_2 n_C^2 n_D$$

where k_i with i=1,2,3,4 are constants.

11.1 Write a function that computes the vector of time derivatives $d\mathbf{w}/dt = [dn_C/dt; dn_D/dt]$ given a value of t and the current values of $\mathbf{w} = [n_C; n_D]$ and the values of the concentrations n_A, n_B .

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 n_C, n_D as a function of time (on the same plot). Use the following values for parameters:

$$k_1 = 1 hr^{-1} \qquad k_2 = 1 l^2 hr^{-1} \; k_3 = 1 l hr^{-1}, \; k_4 = 1 hr^{-1}, \; n_A = 1 mol/l \quad n_B = 3 mol/l$$

Run the simulation for 50 hours, with initial conditions $n_C = n_D = 1 mol/l$.

If your code runs correctly, you should see steady oscillations in the concentrations.