**BE2080 Week#5**

**Submission:**

**1. You can still copy/paste your programs or expression in a word document for submission.**

**2. Since the programs may become longer, you can also submit your Matlab programs (.m files). If you choose to do so, please name your files clearly, for example, Wk4\_Q1.m.**

**3. Please check and make sure all your files are submitted.**

1) (10 pts) Write a function PlotData.m that will receive one input argument: which is a handle to a plot function (such as plot(), bar(), etc.). Inside the function, a vector of 10 random integers between [-10, 10] will first generated. Then these random data will be plotted using the type of plot function specified in the input argument. Display the type of plot function as the title of the plot.

For example, a call to the function PlotData(@bar) will plot the ten random data points using the bar chart.

function [] = PlotData(plot) \*\*avoid using plot as variable

%{

This function plots ten random data points

using any plot you have as input argument.

function [] = PlotData(type of plot handle)

%}

y = randi([-10,10], 1, 10);

plot(y);

title(func2str(plot));

end

2) (10 pts) Write a function that will receive three input arguments: first argument is the data in the form of a x vector; the second argument is a handle to a function; and the third argument is the handle to a plot function. The function will calculate the function values of the x vector, then it will produce a plot using the third input argument. For example, a call to the function HW51(x, @sin, @bar) will plot x vs. sin(x) using bar chart.

function [] = HW51(x, handle, phandle)

%{

This function calculates function values of x

vector and produces plot using third input

argument.

%}

y = handle(x);

phandle(x, y);

3) (10 pts) Write a function that will return a random number (float number). If no arguments are passed to the function, it will return one random number from 0 to 1. If one argument is passed, it is the max and the returned one random number will be from 0 to max. If two arguments are passed, they represent the min and max and the function will return one random number within the range of [min, max].

function [ran\_num] = Question3(varargin)

%Function returns a random number (float number).

if nargin == 0

ran\_num = rand;

elseif nargin == 1

max = varargin{1};

ran\_num = rand\*(max);

elseif nargin == 2

if varargin{1} < varargin{2}

min = varargin{1};

max = varargin{2};

else

min = varargin{2};

max = varargin{1};

end

ran\_num = rand\*(max - min) + min;

end

end

4) (10 pts) Write a function that will receive the radius *r* of a sphere. It will calculate and return the surface area of the sphere (4 π r2). If the function call gets two output arguments, the function will also return the volume of the sphere (4/3 π r3) as the second output argument in addition to the surface area as the first argument.

function [s\_a, varargout] = Question4(r)

%{

Calculates and returns surface area of sphere. If two output

arguments, function also returns volume of sphere.

[surface\_area, volume (if 2nd output arg exists)] = Question4(radius)

%}

s\_a = 4 \* pi \* r^2;

if nargout == 2

varargout{1} = (4/3) \* pi \* r^3;

end

end

5) (10 pts) Write a function “plotexvar” that will plot data points represented by the x and y vectors which are passed as the first two input arguments. In other words, this function takes **at least two input arguments**. If a third argument is passed, it is a color, and if a fourth argument is also passed, it is a line width for the plot. Add a title to the plot to display the total number of arguments passed to the function.

hints: check Matlab help on plot() regarding changing the plot properties (color, linewidth). Here is an example of calling the function and the resulting plot:

>> x=-pi:pi/50:2\*pi;

>> y = sin(x);

>> plotexvar(x,y, [0.5,0.5,0.5],3)



function [] = plotexvar(x, y, varargin)

%{

This function plots data poins represented by x and y vectors. If

third argument is passed, it's the color and if fourth argument is

passed, it's the line width for the plot.

%}

if nargin == 2

plot(x, y);

title('2 arguments')

elseif nargin == 3

plot(x, y, 'Color', varargin{1});

title('3 arguments')

elseif nargin == 4

plot(x, y, 'Color', varargin{1}, 'LineWidth', varargin{2});

title('4 arguments')

end

end

6) (10 points) Write a function that will receive a variable number of function handles (up to 4) as input arguments. It will display plots of these functions in subplots in one single Figure window, with the function names in the titles. The function will first create an x vector that has 100 random numbers from 1 to 100. All plots will be organized in the following way:

(a) one input function handle 🡪display in a single plot

(b) two input function handles 🡪display in a 1 (row) x 2 (cols) subplot format

(c) three input function handles 🡪display in a 1 (row) x 3 (cols) subplot format

(d) four input function handles 🡪display in a 2 (rows) x 2 (cols) subplot format

function [] = Question6(varargin)

%{

Function receives variable number of function handles as input args and

then displays plots of functions in subplots in one single window.

%}

x = randi([1, 100], 1, 100);

for i = 1:nargin

if nargin == 1

subplot(1, 1, i)

elseif nargin == 2

subplot(1, 2, i)

elseif nargin == 3

subplot(1, 3, i)

elseif nargin == 4

subplot(2, 2, i)

end

varargin{i}(x);

title(func2str(varargin{i}));

end

7) (10 points) The Wind Chill Factor (WCF) measures how cold it feels with a given air temperature (T, in degrees Fahrenheit) and wind speed (V, in miles per hour). One formula for the WCF is: WCF = 35.7 + 0.6 T – 35.7 (V 0.16) + 0.43 T (V 0.16)

Write a script to plot the WCF for varying wind speeds from 0 mph to 50 mph at 4 different temperatures 30°F, 50°F, 70°F. Plot all 3 curves in a single plot window, using different colors. Please include appropriate axis labels, title, and legends in your plot (see below for an example plot).



%{

This script plots Wind Chill Factor for varying wind speeds from 0 mph to

50 mph at 3 different temperatures (F).

%}

V = 0:50;

T = [30 50 70];

for i = 1:3

WCF = 35.7 + 0.6\*T(i) - 35.7\*(V.^0.16) + 0.43\*T(i)\*(V.^0.16);

plot(V, WCF)

hold on

end

xlabel('Wind speed')

ylabel('Temperature')

legend('30 deg', '50 deg', '70 deg')

title('Wind chill factors')

(8) (10 points) The enclosed file ‘population.txt’ contains historical data of world population of human. Read in the data to Matlab and plot the world population vs. year.

**Use the “plot tool” in Matlab to customize the plot** using the following specific format (see an example plot below).

1. use yellow “figure” background;
2. use log scale for the y-axis (population);
3. use linewidth 3, and “cyan” color [0, 1, 1] for the line plot;
4. use font size 14, bold font weigh for all axes labels and title.

Use the “code generation” option in Matlab to generate a function to plot this figure.

**Submit the automatically generated .m file.**

**\*\*\*\*\* File is called createfigure.m**



(9) (10 points) Bode plots: the data file bode.txt stores three columns of data that represent the frequency response of an electric circuit. The first column is the frequency in Hz, the 2nd column data is the amplitude of the frequency response, the 3rd column is the ‘phase’ of the response in radian.

Write a script to read in this data file, and plot the amplitude and phase responses in one figure using two subplots (see example below). Convert the amplitude data to decibel (dB) before plotting: Data(db) = 20\*log10(Data). Please use log scale in x-axis, and use linewidth of 3.



%{

This script plots amplitude and phase responses in

one figure using two subplots.

%}

d = importdata('bode.txt');

frequency = d.data(:, 1);

amplitude = d.data(:, 2);

for i = 1:length(amplitude)

amplitude(i) = 20 \* log10(amplitude(i));

end

phase = d.data(:, 3);

figure(1)

subplot(2, 1, 1)

semilogx(frequency, amplitude, 'LineWidth', 3)

ylabel('Amplitude (dB)')

title('Bode Diagram')

subplot(2, 1, 2)

semilogx(frequency, phase, 'LineWidth', 3)

ylabel('Phase (rad)')

xlabel('Frequency (Hz)')

\*need to use grid on