LAB 3: Introduction to Plotting in MATLAB

1. OBJECTIVES:

Following are the objectives of lab session#3:

- To analyze information using plots in MATLAB.
- To be able to plot a graph from given data points.
- To know about the different types of plots in MATLAB.
- To format a plot in MATLAB.

2. INTRODUCTION:

Plots are a very useful tool for presenting and analyzing information. This is true in any field, but especially in the fields of science and engineering, where MATLAB is mostly used. MATLAB has a huge collection of commands for creating different types of plots. These include standard plots, area plots, bar plots, stairs plots, polar plots, three-dimensional contour surface and mesh plots, and many more. The plots can be formatted to have a desired appearance. The line type (solid, dashed, etc.), color, and thickness can be prescribed. Line markers, grid lines, titles, labels and text comments can also be added. Several graphs can be created in the same plot, and several plots can be placed on the same page. All of this detail will be covered in this lab session.

3. 2-D PLOTS:

3.1. plot() command:

2D Plots in MATLAB is a feature that enables a user to create the two-dimensional graph for any dependent variable as a function of a depending variable. A plot can present the data in continuous, discrete, surface or volume form. The default standard function for 2D graph plotting is plot() function. It creates a line plot for data 'Y' with respect to its corresponding data in the 'X' axis. The syntax of plot command is given by:

$$plot(x, y)$$

3.2. Line Specifiers in MATLAB:

The line drawn by the plot command can be modified by line specifiers. It can be used to determine the style, color and markers on a line. They are typed as a string within the plot command. Following is the syntax and the detail of line specifiers:

$$plot(x, y, 'line\ specifiers')$$

Line Style	Specifier	Line Color	Specifier	Marker Type	Specifier
Solid	-	red	r	plus sign	+
dotted	:	green	g	circle	o
dashed	--	blue	b	asterisk	*
dash-dot	-.	Cyan	c	point	.
		magenta	m	square	s
		yellow	y	diamond	d
		black	k		

Figure 1: Line Specifiers

3.3. Multiple graphs on single plot:

Multiple graphs can also be added to a single plot. This can be done by any of the two ways:

1. Using multiple input arguments in a single plot command. The syntax would then be:

$$plot(x, y, u, v, t, h)$$

This will then give three graphs namely, the y versus x graph, the v versus u graph and the h versus t graphs on a single plot. These will all be of different colors by default. Their styles can also be changed by adding line specifiers inside the plot command as indicated below:

$$plot(x, y, 'line\ spec\ 1', u, v, 'line\ spec\ 2', t, h, 'line\ spec\ 3')$$

2. Using hold on and hold off commands. Hold on holds the current plot and all axis properties so that subsequent plot commands add to the existing plot whereas hold off returns to the default mode whereby plot commands erase the previous plots and reset all axis properties before drawing new plots.

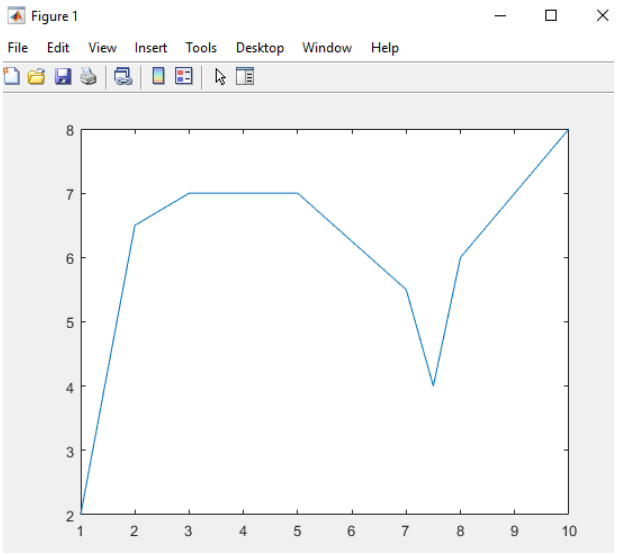
Quick Hands-on Practice:

1) Plot the given data using MATLAB:

x	1	2	3	5	7	7.5	8	10
y	2	6.5	7	7	5.5	4	6	8

```
x=[1 2 3 5 7 7.5 8 10];
y=[2 6.5 7 7 5.5 4 6 8];
plot(x,y)
```

Output:

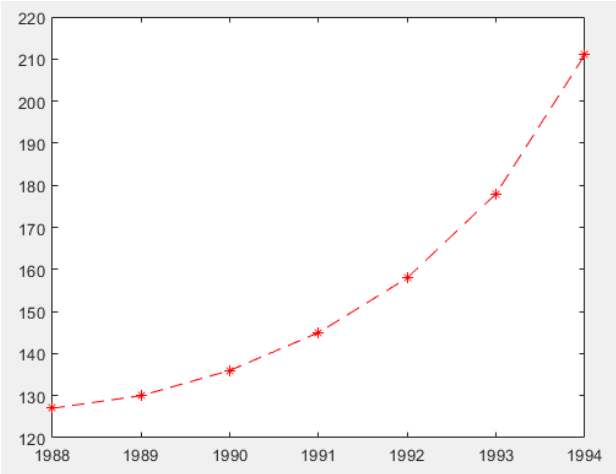


2) Plot the following dataset in MATLAB such that the plot is a dashed red line with asterisk (*) as a marker.

Year	1988	1989	1990	1991	1992	1993	1994
Sales (M)	127	130	136	145	158	178	211

```
year=1988:1:1994;
sales=[127, 130, 136, 145, 158, 178, 211];
plot(year,sales,'--r*')
```

Output:

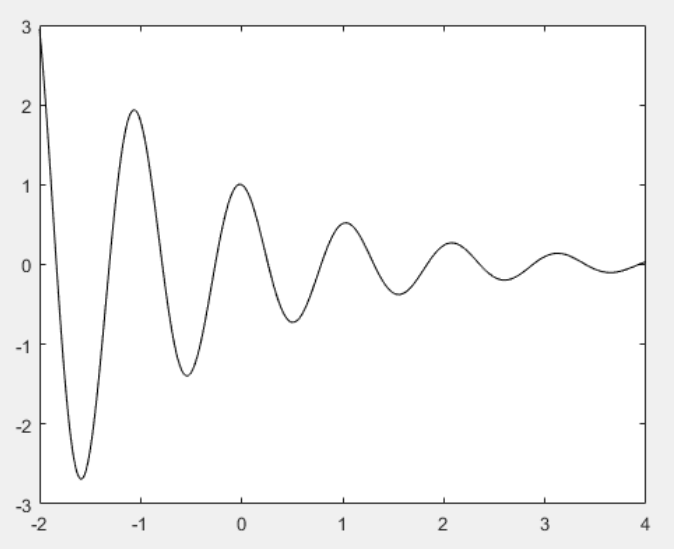


3) Plot the following function in MATLAB. Choose an appropriate increment in x values so as to obtain a continuous curve:

$$y = 3.5^{-0.5x}\cos(6x)\ for - 2 \leq x \leq 4$$

```
x=-2:0.01:4;
y=3.5.^(-0.5*x).*cos(6*x);
plot(x,y,'k')
```

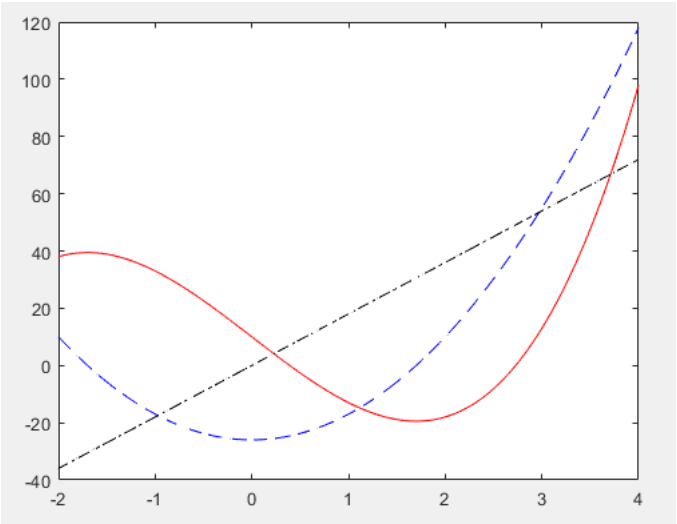
Output:



4) Plot the function, $y = 3x^3 - 26x + 10$ and its first and second derivatives, for $-2 \leq x \leq 4$ all in the same plot.

```
x=[-2:0.01:4];
y=3*x.^3-26*x+10;
yd=9*x.^2-26;
ydd=18*x;
plot(x,y,'r',x,yd,'--b',x,ydd,'-.k')
```

Output:



3.4. fplot() command:

The fplot command can be used to plot a function with the form: y=f(x). The benefit of using the fplot command is that it is easy to implement and the function can be typed as a string. The syntax of fplot() is given by:

$$\boldsymbol{fplot('function', limits)}$$

Here, the limits are a vector dictating the domain/the limit of x and optionally the limit of y values as well.

$$limits = [\text{xmin}, \text{xmax}] \text{ or } [\text{xmin}, \text{xmax}, \text{ymin}, \text{ymax}]$$

4. FORMATTING OF PLOTS

Formatting commands are used to change/enhance our plot by placing additional information on it. With formatting methods, you can:

- Add title to the plot.
- Add labels to axes.
- Change range of the axes.
- Add legend.
- Add text blocks.
- Add grid.

4.1. Title:

The syntax for adding a title is:

$$title('String')$$

By typing this command, the 'String' will be added as the title of the plot.

4.2. X-label:

The syntax for adding a label on x axis is:

$$xlabel('String')$$

By typing this command, the 'String' will be added as the label of the x-axis.

4.3. ylabel command:

The syntax for adding a label on y axis is:

$$ylabel('String')$$

By typing this command, the 'String' will be added as the label of the y-axis.

4.4. Changing Axes:

The syntax for changing the axes limit is:

$$axis([xmin, xmax, ymin, ymax])$$

This command will set the maximum and minimum limit of both x and y axes.

4.5. Adding Text:

The syntax for adding text to a plot is:

$$text(x, y, 'String')$$

This command will place the text on the plot at the specified coordinate (x,y) relative to the plot axes.

4.6. Adding Legend:

The syntax for adding legend to a plot is:

$$legend('String1','String2','String3',\dots)$$

This command will place a legend on the plot which is basically a text for differentiating multiple curves. In the syntax above, string1 will be assigned to first curve, string2 to second curve and so on.

5. MULTIPLE PLOTS USING SUBPLOT:

Several plots on one page can be created with the subplot command. The syntax of subplot command is given below.

$$subplot(m,n,p)$$

This command creates mxn plots in the Figure Window. The plots are arranged in m rows and n columns. The variable p defines which plot is active. The plots (p in the syntax above) are numbered from 1 to m x n. The upper left plot is 1 and the lower right plot is m x n The numbers increase from left to right within a row, from the first row to the last

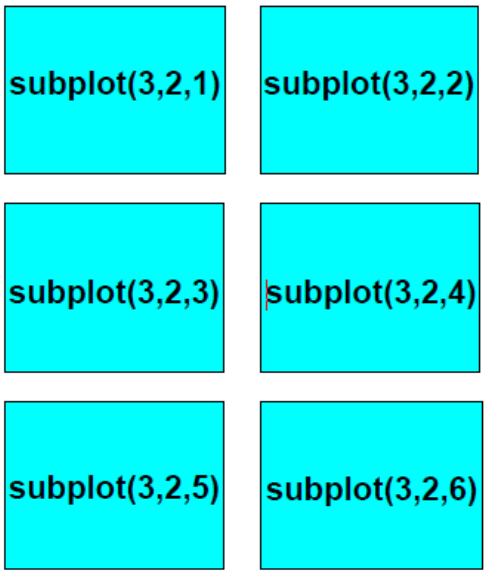


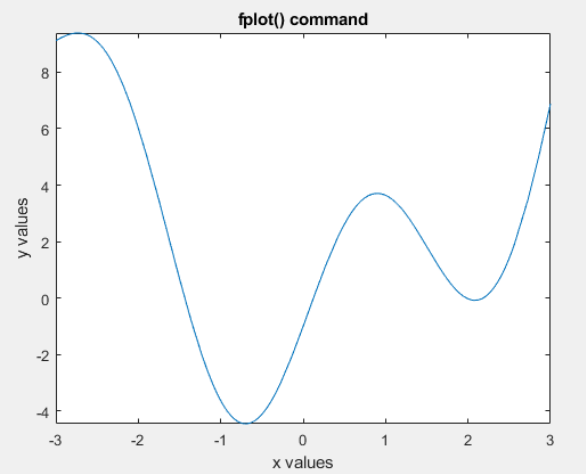
Figure 2: Subplot command

Quick Hands-on Practice:

1) Plot the function $y = x^2 + 4\sin(2x) - 1$ for $-3 \leq x \leq 3$ using the fplot() command. Also add x and y-axis labels as well as a title.

```
clc;
fplot('x^2+4*sin(2*x)-1',[-3,3])
xlabel('x values')
ylabel('y values')
title('fplot() command')
```

Output:

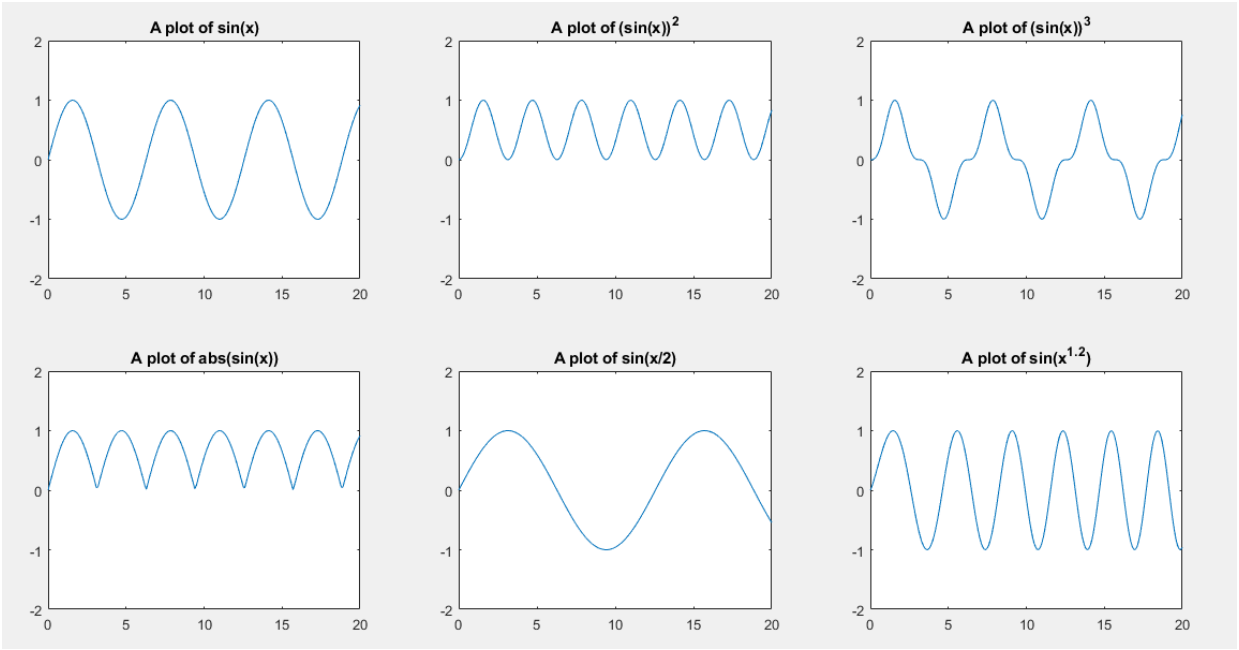


2) Plot the graphs of sin(x), $\sin^2$(x), $\sin^3$(x), abs(sin(x)), sin(x/2) and sin($x^{1.2}$) using subplot command. Also add appropriate titles.

```
x=[0:0.1:20];
y1=sin(x);
y2=(sin(x)).^2;
y3=(sin(x)).^3;
y4=abs(sin(x));
y5=sin(x/2);
y6=sin(x.^1.2);
subplot(2,3,1);
plot(x,y1);
title('A plot of sin(x)');
axis([0 20 -2 2]);
subplot(2,3,2);
plot(x,y2);
title('A plot of (sin(x))^2');
axis([0 20 -2 2]);
subplot(2,3,3);
plot(x,y3);
title('A plot of (sin(x))^3');
axis([0 20 -2 2]);
```

```
subplot(2,3,4);
plot(x,y4);
title('A plot of abs(sin(x))');
axis([0 20 -2 2]);
subplot(2,3,5);
plot(x,y5);
title('A plot of sin(x/2)');
axis([0 20 -2 2]);
subplot(2,3,6);
plot(x,y6);
title('A plot of sin(x^1^.^2)');
axis([0 20 -2 2]);
```

Output:



6. OTHER 2-D PLOTS:

6.1. Area Plot:

Area plots are a type of 2-D plotting in which the area under the curve for a function y=f(x) is filled. The basic syntax of area plot is given by:

$$area(y)$$

Where y is a function of any independent variable.

6.2. Stem Plot:

Stem plots are a type of 2-D plotting in which the graph grows vertically like a stem for a particular x value. The basic syntax of stem plot is given by:

$$stem(x, y)$$

Where y is a dependent variable and a function of the independent variable x.

6.3. Bar Plot:

Bar plots are a type of 2-D plotting that represent the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The basic syntax of bar plot is given by:

$$bar(x, y)$$

Where y is a dependent variable and a function of the independent variable x.

6.4. Barh Plot:

Barh plots are a type of 2-D plotting that represent the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. Barh plots are plotted in a horizontal manner instead of vertical (as is done in simple bar plots). The basic syntax of barh plot is given by:

$$barh(x, y)$$

Where y is a dependent variable and a function of the independent variable x.

6.5. Stairs Plot:

Stair plots are a type of 2-D plotting in which the generated graph looks like a staircase. The basic syntax of stairs plot is given by:

$$stairs(x, y)$$

Where y is a dependent variable and a function of the independent variable x.

6.6. Pie Plot:

Pie plots are a type of 2-D plotting in which a pie chart is obtained. A pie chart/plot is used to indicate data in percentage form. The basic syntax of pie plot is given by:

$$pie(x)$$

Where x is the input dataset.

6.7. Scatter Plot:

Scatter plots are a type of 2-D plotting that use dots to represent values for two different numeric variables. The position of each dot on the horizontal and vertical axis indicates values for an individual data point. Scatter plots are used to observe relationships between variables. The basic syntax of scatter plot is given by:

$$scatter(x, y)$$

Where y is a dependent variable and a function of the independent variable x.

6.8. Polar Plot:

Polar plots are a type of 2-D plotting that create the graph using polar coordinates i.e., with the values of theta (the angle in radian) and the values of rho (the radius for each data point). The basic syntax of polar plot is given by:

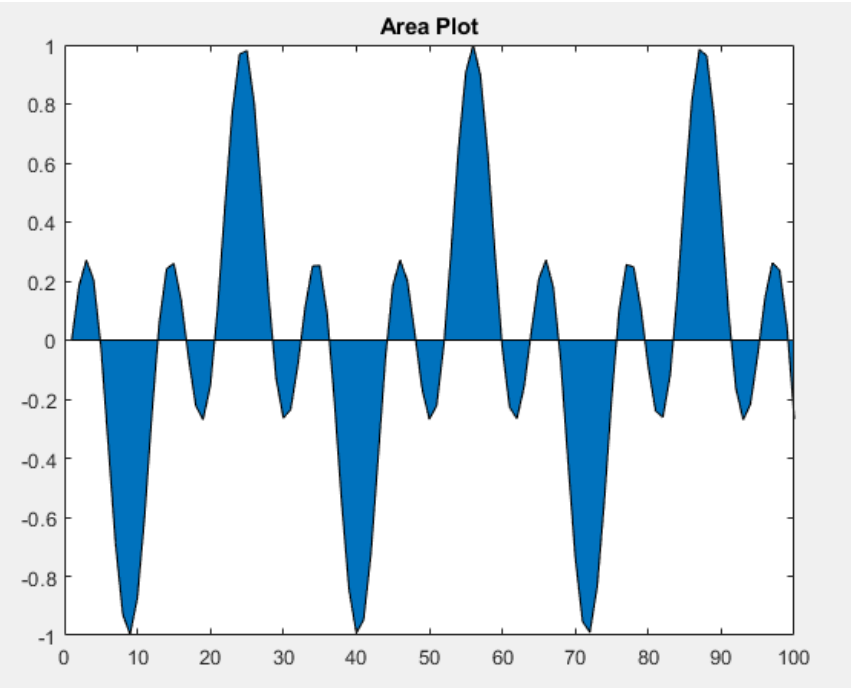
$$polar(theta, rho)$$

Quick Hands-on Practice:

1) Plot the function $y = sin(t)cos(2t)$ for $0 \leq t \leq 20$ using the area plot command.

```
t=0:0.2:20;
y=sin(t).*cos(2*t);
area(y);
title('Area Plot');
axis([0 100 -1 1]);
```

Output:

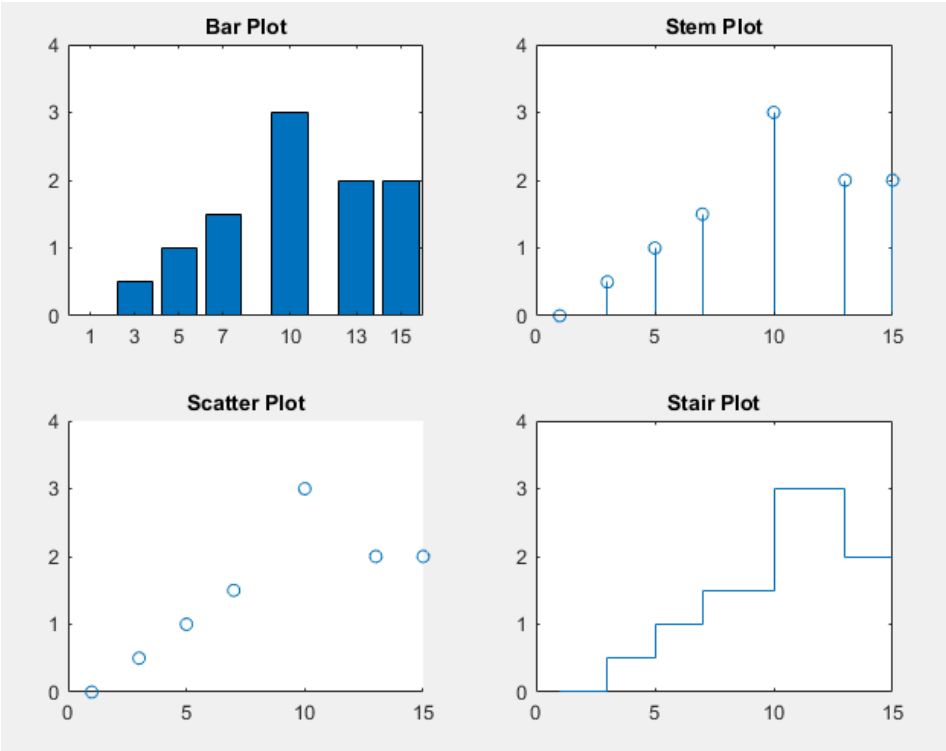


2) Keeping in mind the following dataset, plot a bar plot, stem plot, scatter plot and stairs plot using a subplot command. Add titles so that plots can be differentiated.

X	1	3	5	7	10	13	15
Y	0	0.5	1	1.5	3	2	2

```
x=[1 3 5 7 10 13 15];
y=[0 0.5 1 1.5 3 2 2];
subplot(2,2,1);
bar(x,y);
axis([0 16 0 4]);
title('Bar Plot');
subplot(2,2,2);
stem(x,y);
axis([0 15 0 4]);
title('Stem Plot');
subplot(2,2,3);
scatter(x,y);
axis([0 15 0 4]);
title('Scatter Plot');
subplot(2,2,4);
stairs(x,y);
axis([0 15 0 4]);
title('Stair Plot');
```

Output:



Assigned Tasks

TASK 1:

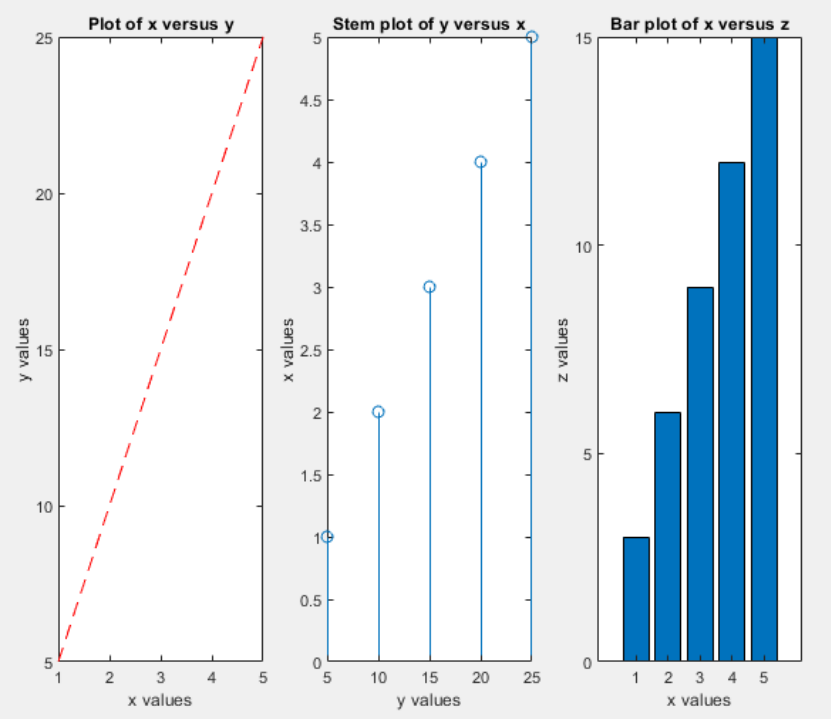
Write a script file in which the user should be able to enter the array of five random numbers (between 1 and 10), and then the modeler should store these numbers in the variable "x". After it, perform the following functions:

a) Plot a line graph for the equation: $Y=mx^2$ where m represents the number of entities (i.e., 5).
b) Replace the independent and dependent variables of part (a) and now draw the stem plot between Y (on the horizontal axis) and X (On the vertical axis).
c) If the average of the entered five numbers is less than 20 then make another variable "Z" which will be comprised of five entities and the value of each entity should be three times the value of entered entities (i.e., Z=3x). Conversely, if the average is greater than 50 then $Z=x^4$. Draw a bar chart of the above-explained supposition.

```
clc
x=[];
fprintf("You have to enter an array of 5 random numbers between 1 and
10...\n\n");
for i=1:5
        text=sprintf("Enter number # %d >> ",i);
        x(end+1)=input(text);
    while x(end)<1 || x(end)>10
        fprintf('Number must be between 1 and 10!!\n');
        text=sprintf("Enter number # %d >> ",i);
        x(end)=input(text);
    end
end
fprintf("The entered array is:");
x
y=length(x)*x;
subplot(1,3,1);
plot(x,y,'--r');
xlabel('x values');
ylabel('y values');
title('Plot of x versus y');
subplot(1,3,2);
stem(y,x);
xlabel('y values');
ylabel('x values');
title('Stem plot of y versus x');
if mean(x)<20
    Z=3*x;
elseif mean(x)>50
    Z=x.^4;
end
subplot(1,3,3);
bar(x,Z);
xlabel('x values');
ylabel('z values');
title('Bar plot of x versus z');
```

Output:

```
You have to enter an array of 5 random numbers between 1 and 10...

Enter number # 1 >> 1
Enter number # 2 >> 2
Enter number # 3 >> 44
Number must be between 1 and 10!!
Enter number # 3 >> 3
Enter number # 4 >> 4
Enter number # 5 >> 5
The entered array is:
x =

     1     2     3     4     5
```

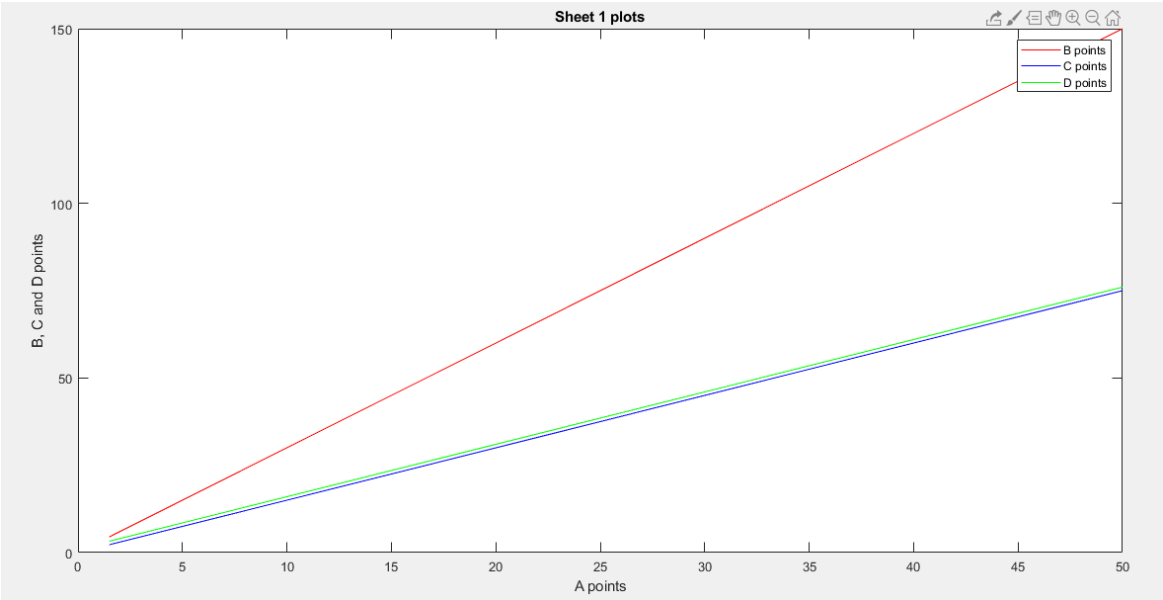


TASK 2:

Write a script file that will read the data from the provided excel file (Datapoints.xlsx) and draw a single line plot in which the first column of the excel file (X) will be considered as an independent variable (On the x-axis) and all other columns shall be considered as data points of the dependent variable (Let's say Output). To put it in a nutshell, the end graph shall be consisting of three lines (As per the data points A, B, and C). You can include plot formatting techniques for better visualization of all three lines.

```
A=xlsread('Datapoints.xlsx',1);
x=A(:,1);
y1=A(:,2);
y2=A(:,3);
y3=A(:,4);
plot(x,y1,'r',x,y2,'b',x,y3,'g');
legend('B points','C points','D points');
xlabel('A points')
ylabel('B, C and D points')
title('Sheet 1 plots')
```

Output:



TASK 3:

In this simplest problem, you are required to graphically interpret newton's second law of motion. Most importantly, the Script file should be written in such a way that it should ask the user three different things:

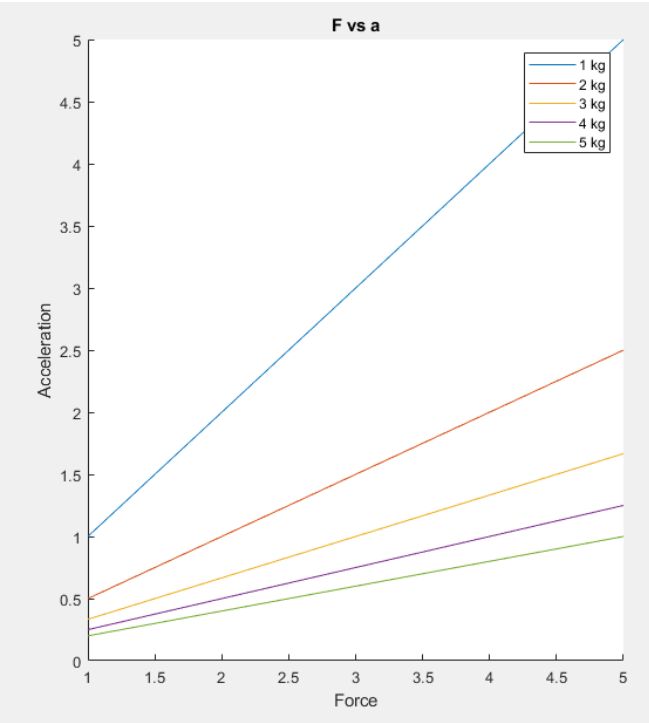
a) The first input should be step size which would be utilized as the incremented value during the given range of input parameters.
b) Range of the applied force (For example, I want to analyze the acceleration due to applied force ranging from 1 N to 20 N and during which you can incorporate the entered step size of part (a) i.e., 1:step size:20).
c) The third thing should be a statement "Whether you want to keep the mass constant?". If yes, then it should further ask the user for a single value of mass. On the contrary, if the user opts for no, then MATLAB should ask the user five different values of mass.
d) Graph of multivalued mass (Five lines): MATLAB code should be in such a way that it should now automatically extract the maximum values of acceleration @ mass and respective force. Those values of acceleration with respective unique mass should be stored sequentially in arrays. Finally, the code should give another graph (This time a stem plot) between mass and acceleration (Of course the force will be constant in this case as per newton's hypothesis).

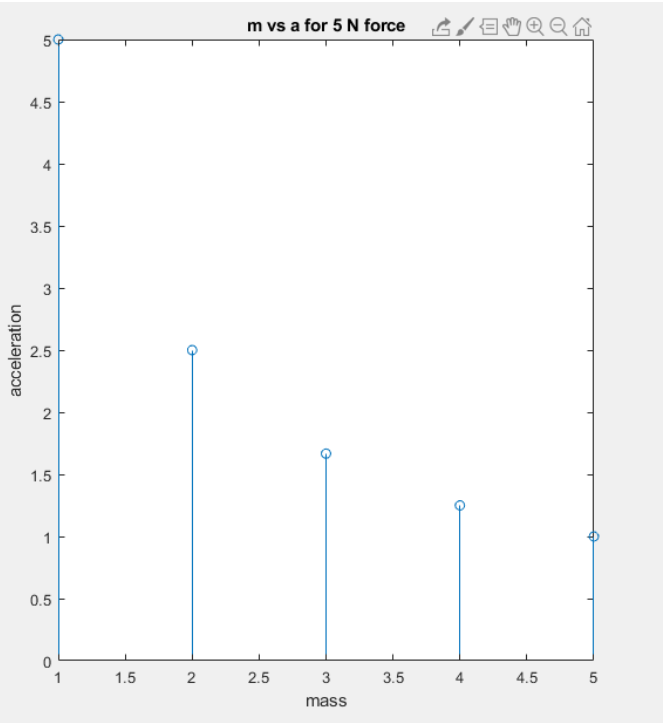
```
clc
step=input('Please enter a step size for force >> ');
rangeF=input('Please enter a range of force in the format [min,max] >> ');
fprintf('The force values entered are:');
F=rangeF(1):step:rangeF(2)
query=input('Whether you want to keep the mass constant? [Y/N] >> ','s');
while query~='Y' && query~='N'
    fprintf('Invalid\n');
    query=input('Whether you want to keep the mass constant? [Y/N] >>
','s');
end

if query=='Y'
    mass=input('\nPlease enter a value of mass >> ');
    fprintf('The mass value entered is:');
    mass
    fprintf('\nThe corresponding acceleration values are:');
    acc=F./mass
    plot(F,acc,'--r');
    xlabel('Force');
    ylabel('Acceleration');
    text=sprintf('F vs a for %d kg mass',mass);
    title(text);
elseif query=='N'
    mass=input('\nPlease enter five values of mass (enclosed in square
brackets) >> ');
    fprintf('The mass values entered are:');
    mass
    acc=[];
    subplot(1,2,1);
    for i=1:5
        a=F./mass(i);
        hold on
        plot(F,a);
        acc=[acc;a];
    end
    fprintf('\nThe corresponding acceleration values are:');
    acc
    fprintf('Here each row is for a unique mass and varying force.\n');
    xlabel('Force');
    ylabel('Acceleration');
    title('F vs a');
    legend(sprintf('%d kg',mass(1)) , sprintf('%d kg',mass(2)) , sprintf('%d
kg',mass(3)) , sprintf('%d kg',mass(4)) , sprintf('%d kg',mass(5)));
end
```

```
Size=size(acc);
if Size(1)>1
    [val,index]=max(F);  % since maximum acceleration occurs at maximum
applied force for a given mass
    maxAcc=acc(:,index)';
    fprintf('\nThe maximum acceleration values occur at %dN force and for
each entered mass is given below:',val);
    maxAcc
    subplot(1,2,2);
    stem(mass,maxAcc);
    xlabel('mass');
    ylabel('acceleration');
    title(sprintf('m vs a for %d N force',val));
end
```

Output:

```
Please enter a step size for force >> 1
Please enter a range of force in the format [min,max] >> [1,5]
The force values entered are:
F =

     1     2     3     4     5

Whether you want to keep the mass constant? [Y/N] >> N

Please enter five values of mass (enclosed in square brackets) >> [1 2 3 4 5]
The mass values entered are:
mass =

     1     2     3     4     5


The corresponding acceleration values are:
acc =

    1.0000    2.0000    3.0000    4.0000    5.0000
    0.5000    1.0000    1.5000    2.0000    2.5000
    0.3333    0.6667    1.0000    1.3333    1.6667
    0.2500    0.5000    0.7500    1.0000    1.2500
    0.2000    0.4000    0.6000    0.8000    1.0000

Here each row is for a unique mass and varying force.

The maximum acceleration values occur at 5N force and for each entered mass is given below:
maxAcc =

    5.0000    2.5000    1.6667    1.2500    1.0000
```


F vs a
1 kg
2 kg
3 kg
4 kg
5 kg
Acceleration
Force


m vs a for 5 N force
acceleration
mass