**Introduction to Matlab Graphics**

**LAB # 04**

****

**Spring 2021**

**CSE301L-Signal $ System**

Submitted by: **Ashfaq Ahmad**

Registration No: **19PWCSE1795**

Class Section: **B**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Engr Durr-e-Nayab**

May 30, 2021

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**OBJECTIVES OF THE LAB**

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

In this lab, we will cover the following topics:

* *Discrete Signal representation in Matlab*
* *Matlab Graphics*
* *Two Dimensional Plots*
* *Plot and subplot*
* *Different Plotting Functions Used in Matla*b

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

**-------------------------TASK 01--------------------------**

Given the signals:

x1[n] = [2 5 8 4 3] x2[n] = [4 3 2]

a) Write a MatLab program that adds these two signals. Use vector addition and multiplication.

b) Instead of using vector addition and multiplication, use for loop to add and multiply the signals.

c) Design a function **SigPlot** that takes the original signals and their sum and product as input and plots them as:

i) Separate Figures,

ii) Single Figure overlapping all the signals, and

iiI) Single Figure with separate signal plots using subplots.

**Source code:**

function[sum product]= sigplot

disp('\*\*\*\*\*\* Task no 01\*\*\*\*\*\*\*\*');

x1=[2 5 8 4 3]

x2=[4 3 2 1 7]

disp('sum and product of two vectors using vector addition and multiplication');

sum= x1+x2

product=x1.\*x2

disp('Now sum and product using for loop ');

for i=1:5;

sum(i)=x1(i)+x2(i);

product(i)=x1(i).\*x2(i);

end

figure (1)

plot(x1,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('first vector');

grid on

figure (2)

plot(x2,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('2nd vector');

grid on

figure (3)

plot(sum,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Sum vector');

grid on

figure (4)

plot(product,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('product');

grid on

figure (5)

plot(x1,'r --','Linewidth',2);

hold on

plot(x2,'g --','Linewidth',2);

hold on

plot(sum,'b --','Linewidth',2);

hold on

plot(product,'y --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('overlapping');

grid on

figure (6)

subplot(4,1,1)

plot(x1,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('first vector');

grid on

subplot(4,1,2)

plot(x2,'b --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('2nd vector');

grid on

subplot(4,1,3)

plot(sum,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Sum vector');

grid on

subplot(4,1,4)

plot(product,'r --','Linewidth',2);

xlabel('x-axis');

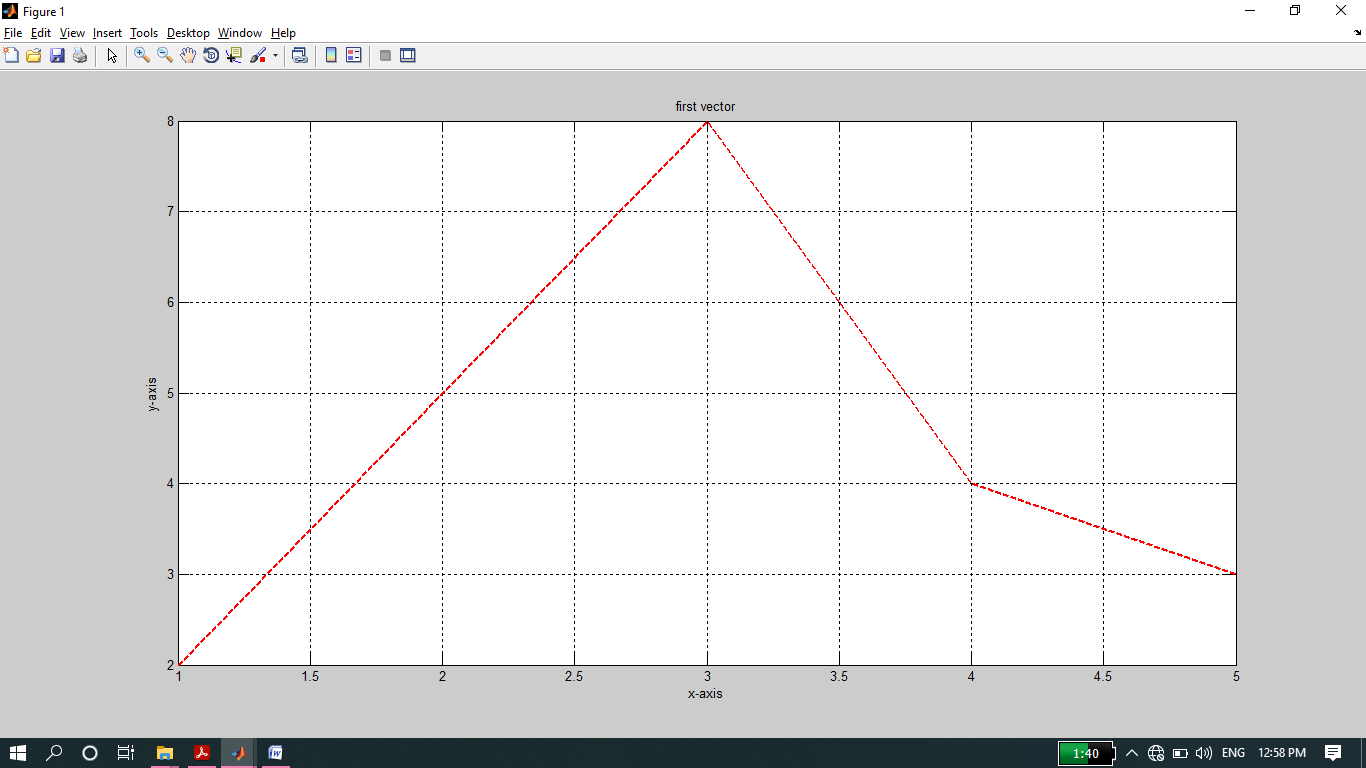
ylabel('y-axis');

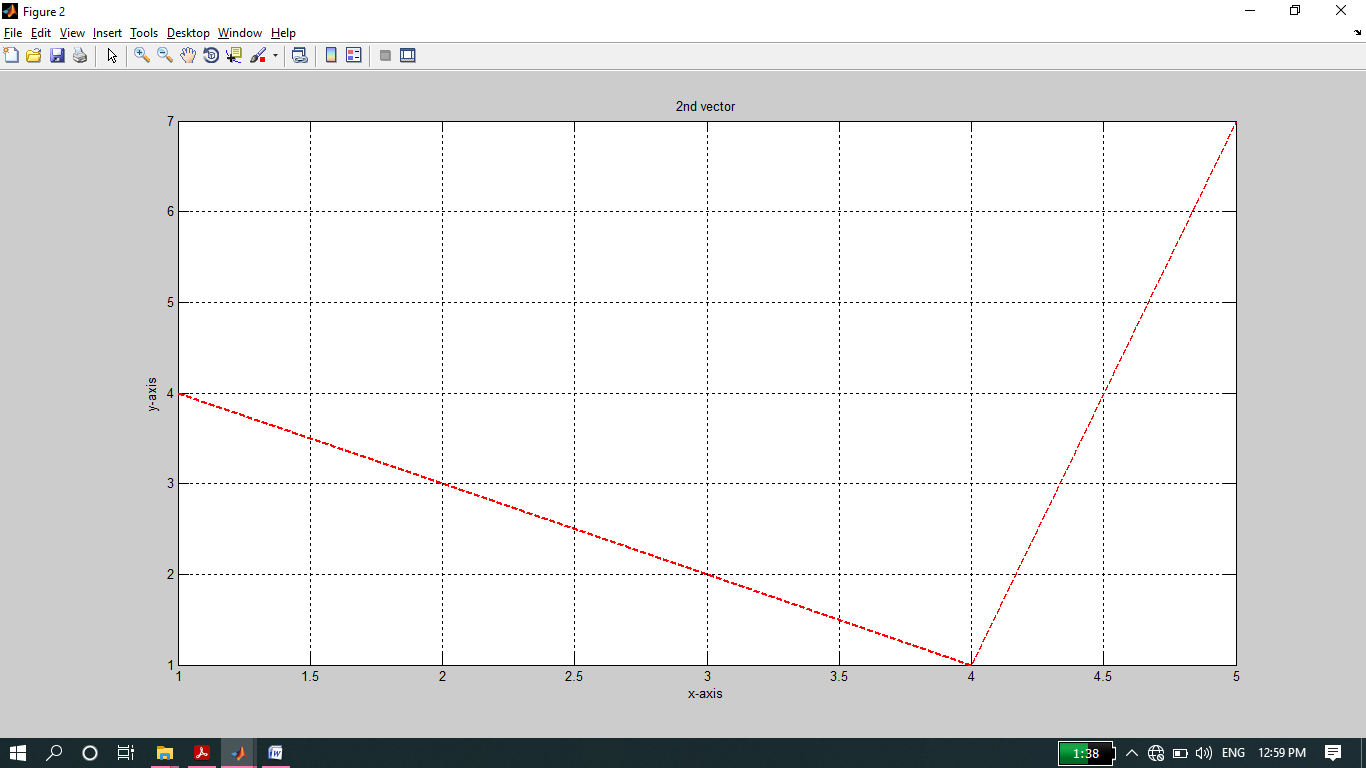
title('product');

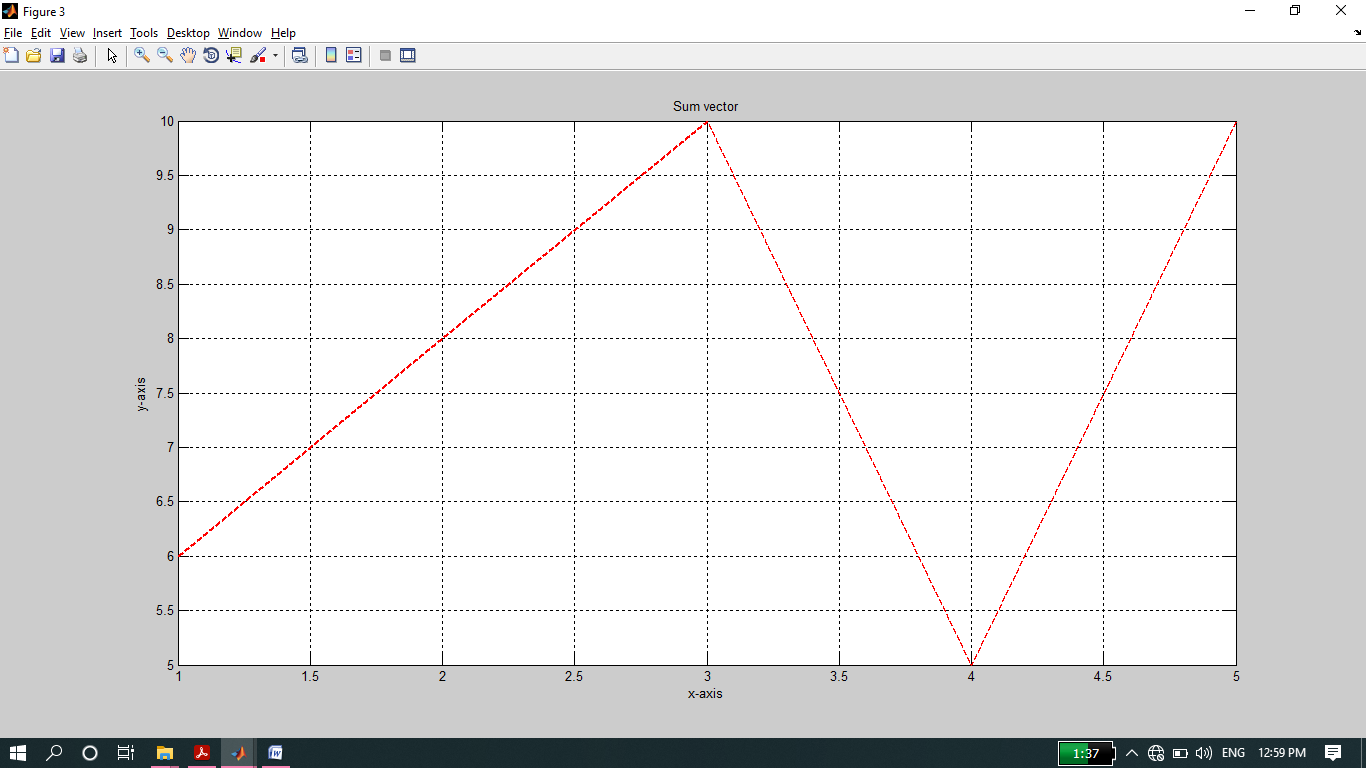
grid on

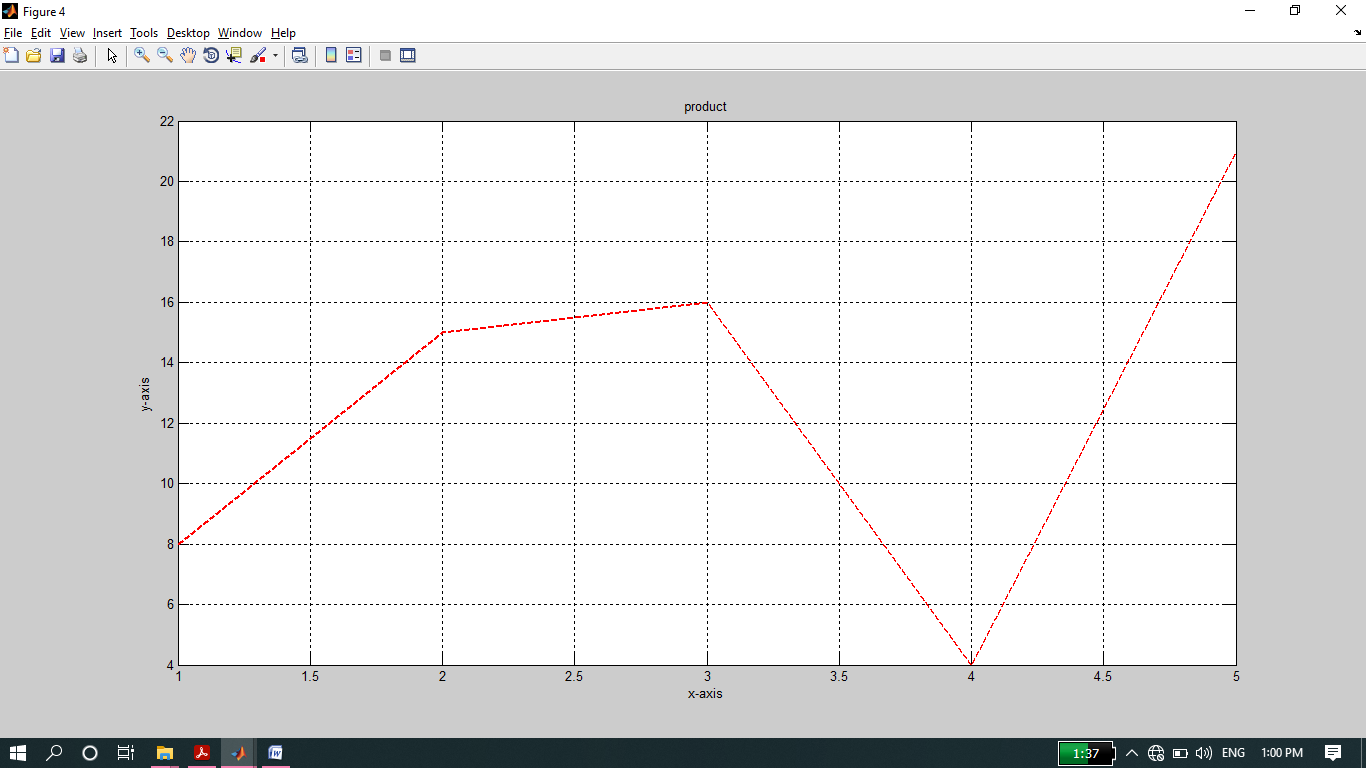
**OUTPUT:**

1. **Separate figures**

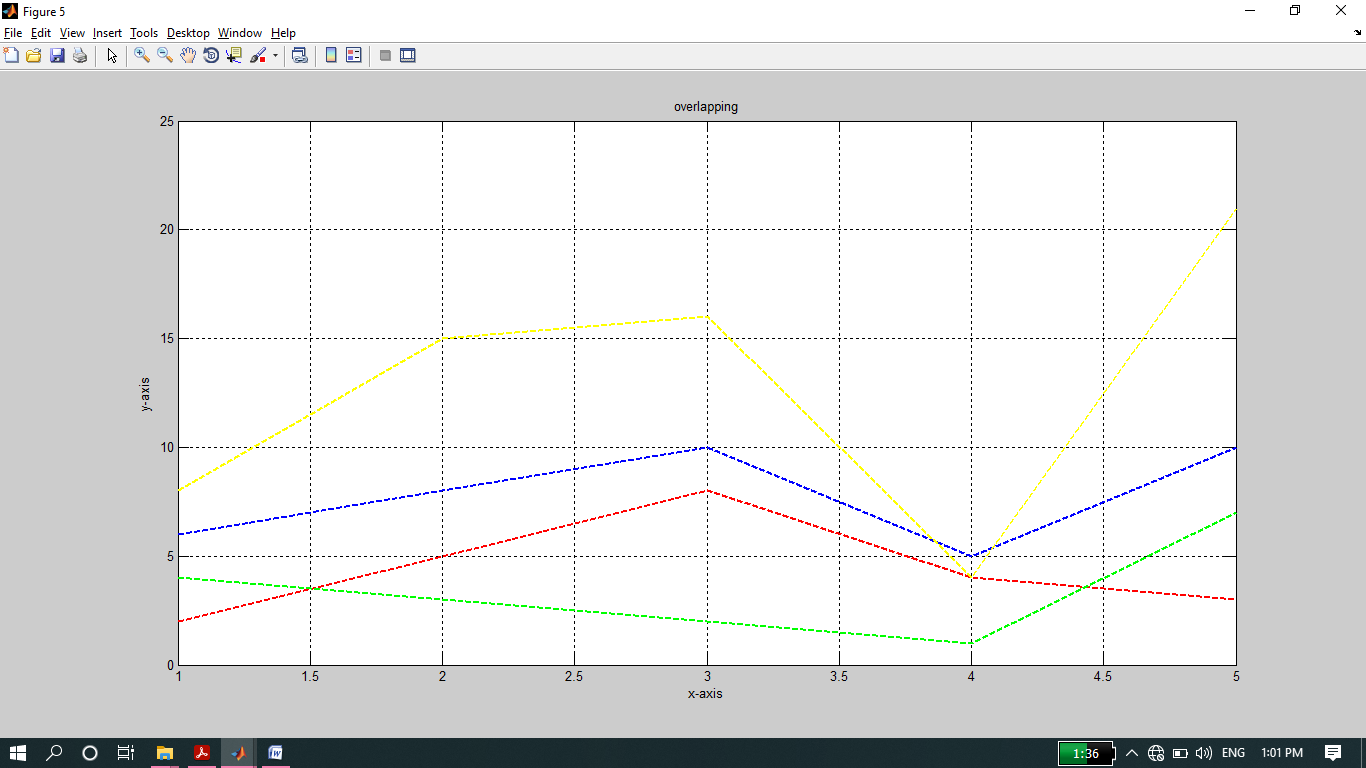
****



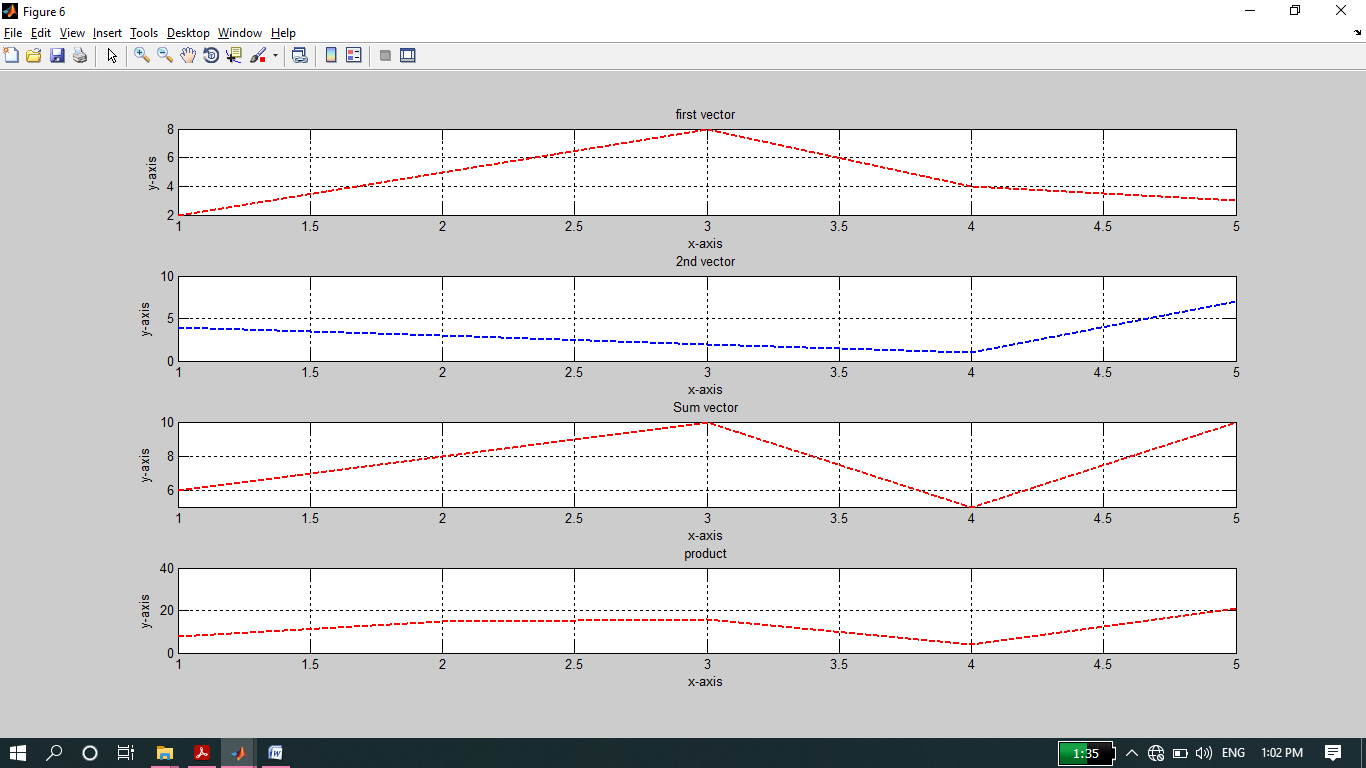




1. Overlapping:



1. All graph in single command:



**---------------TASK 02--------------------------**

Amplitude scaling by a factor β causes each sample to get multiplied by β. Write a user‐defined function **ScaleSig** that has two input arguments: (i) a signal to be scaled and (ii) scaling factor β. The function should return the scaled output to the calling program. In the calling program, get the discrete time signal as well as the scaling factor from user and then call the above‐mentioned function.

Design a function **SigPlot** that takes the original signals and their scaled versions as input from the main calling program and plots them as:

i) Separate Figures,

ii) Single Figure overlapping all the signals, and

iiI) Single Figure with separate signal plots using subplot.

**Source code:**

function[signal,B,product]= sigplot3

disp('\*\*\*\*\*\*task 02\*\*\*\*\*\*\*\*');

disp('please enter signal s elements: ');

for i=1:5;

signal(i)=input('enter element: ');

end

B=input('please enter the value of beta: ');

for i=1:5;

product(i)=signal(i)\*B;

end

figure (1)

stem(signal,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Original signal');

grid on

figure (2)

stem(product,'g --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('scaled version');

grid on

figure (3)

stem(signal,'r --','Linewidth',2);

hold on

stem(product,'g --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('overlapping');

hold on

grid on

figure (4)

subplot(1,2,1)

stem(signal,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Original signal');

grid on

subplot(1,2,2)

stem(product,'g --','Linewidth',2);

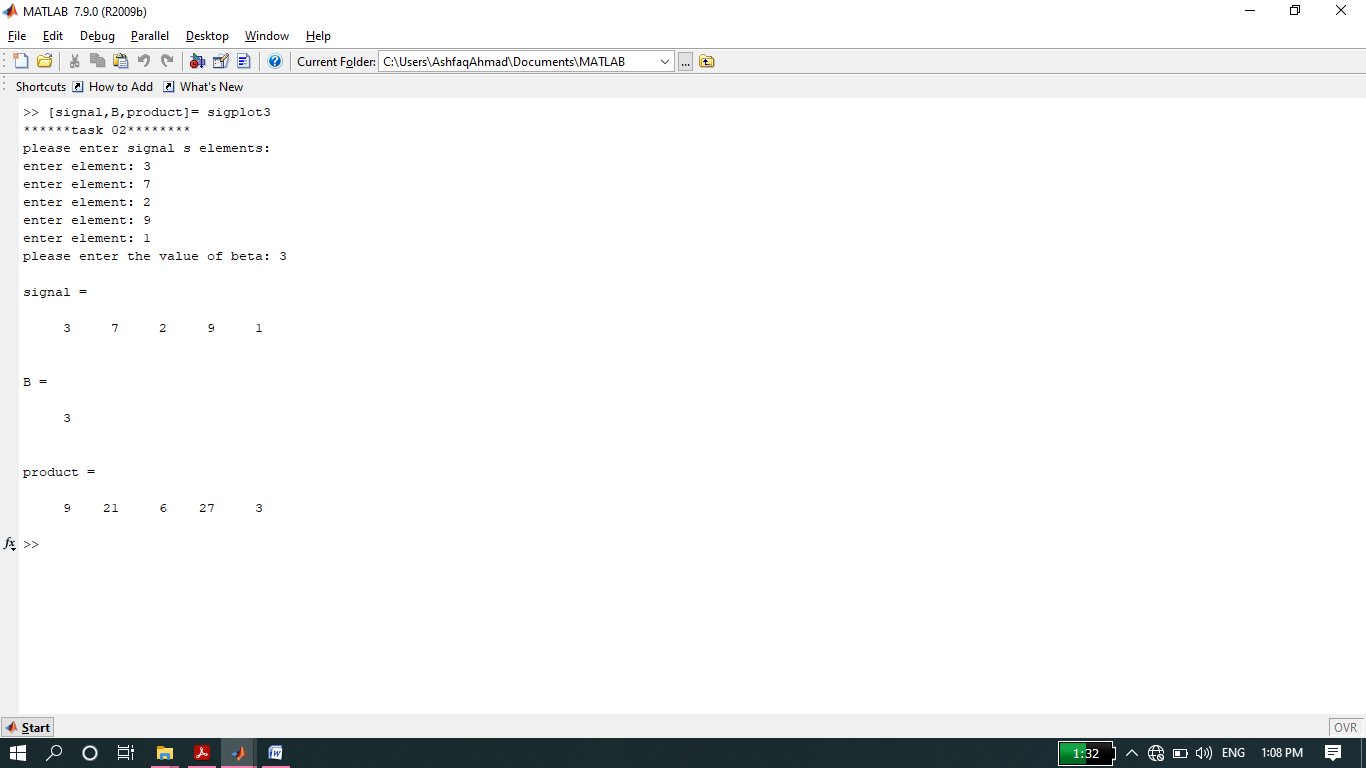
xlabel('x-axis');

ylabel('y-axis');

title('product');

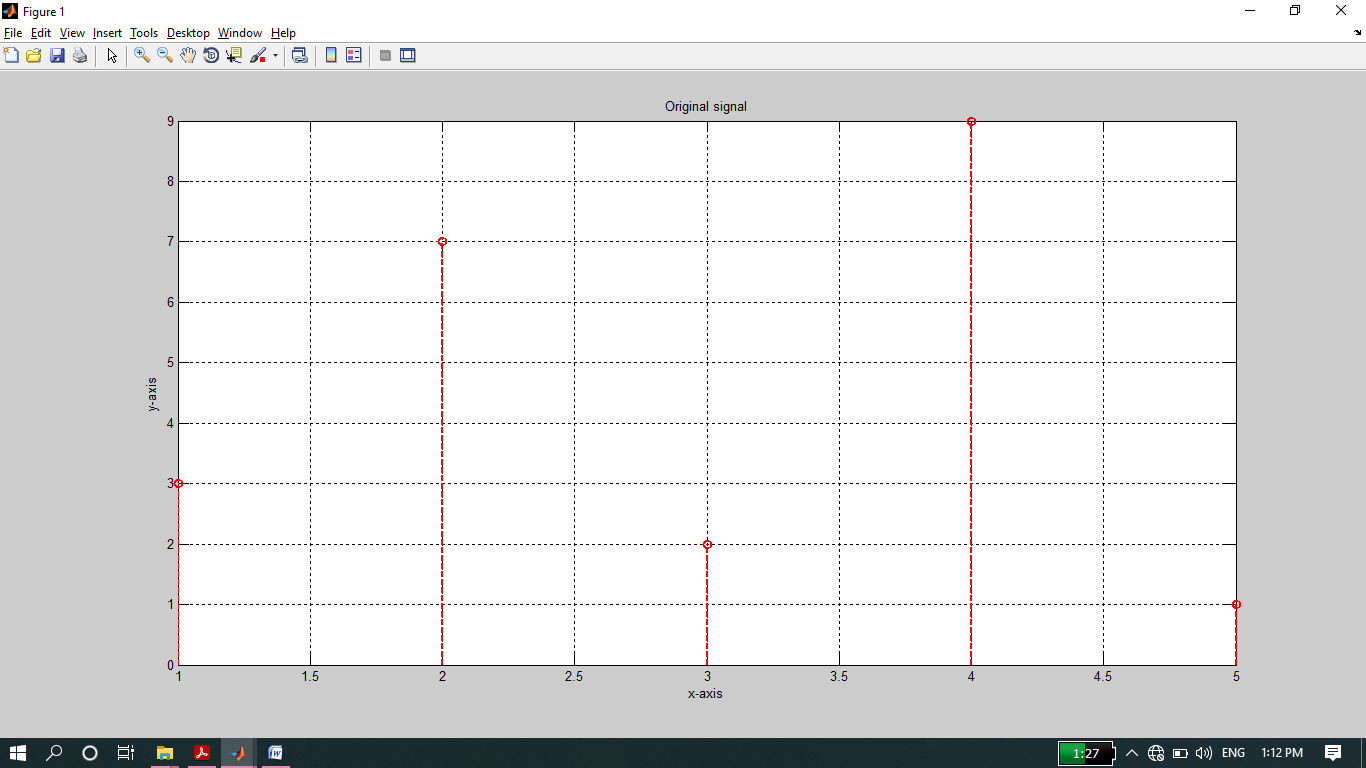
grid on

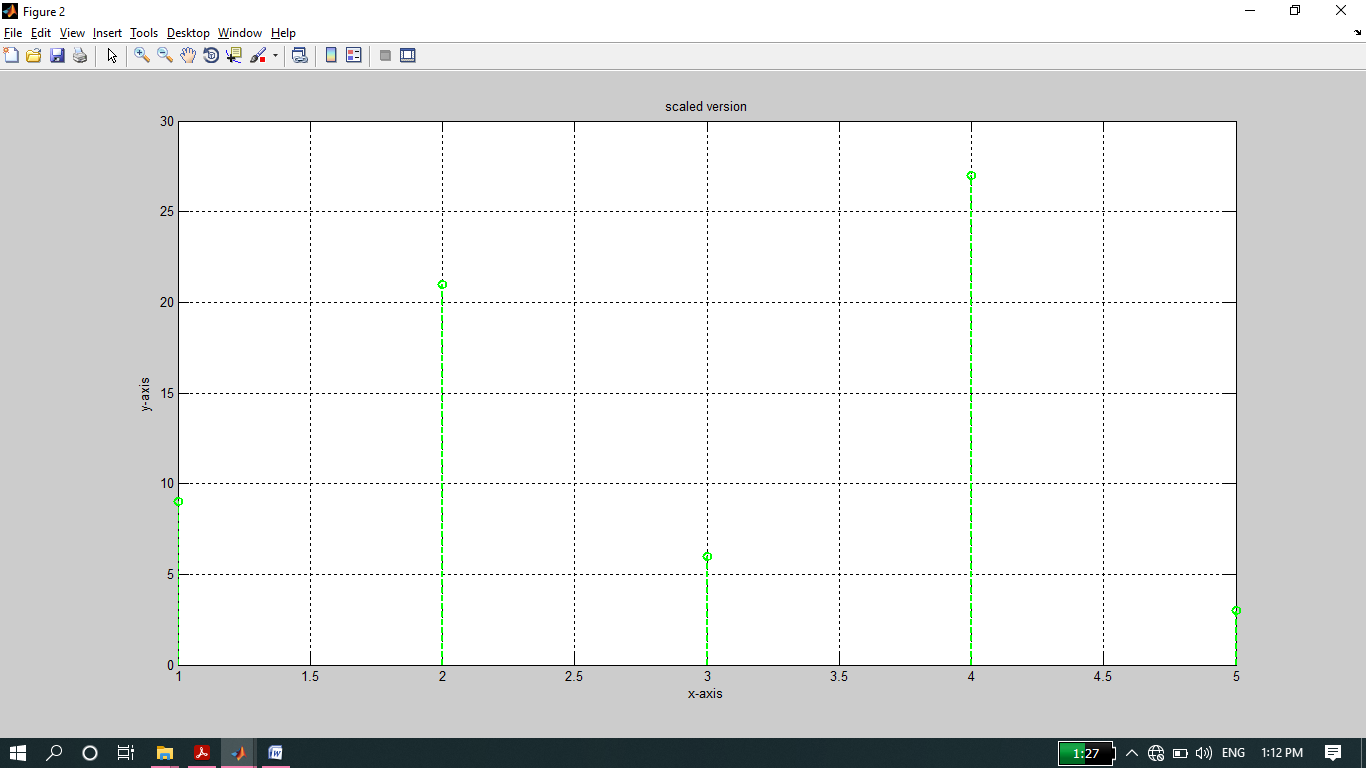
**Output:**



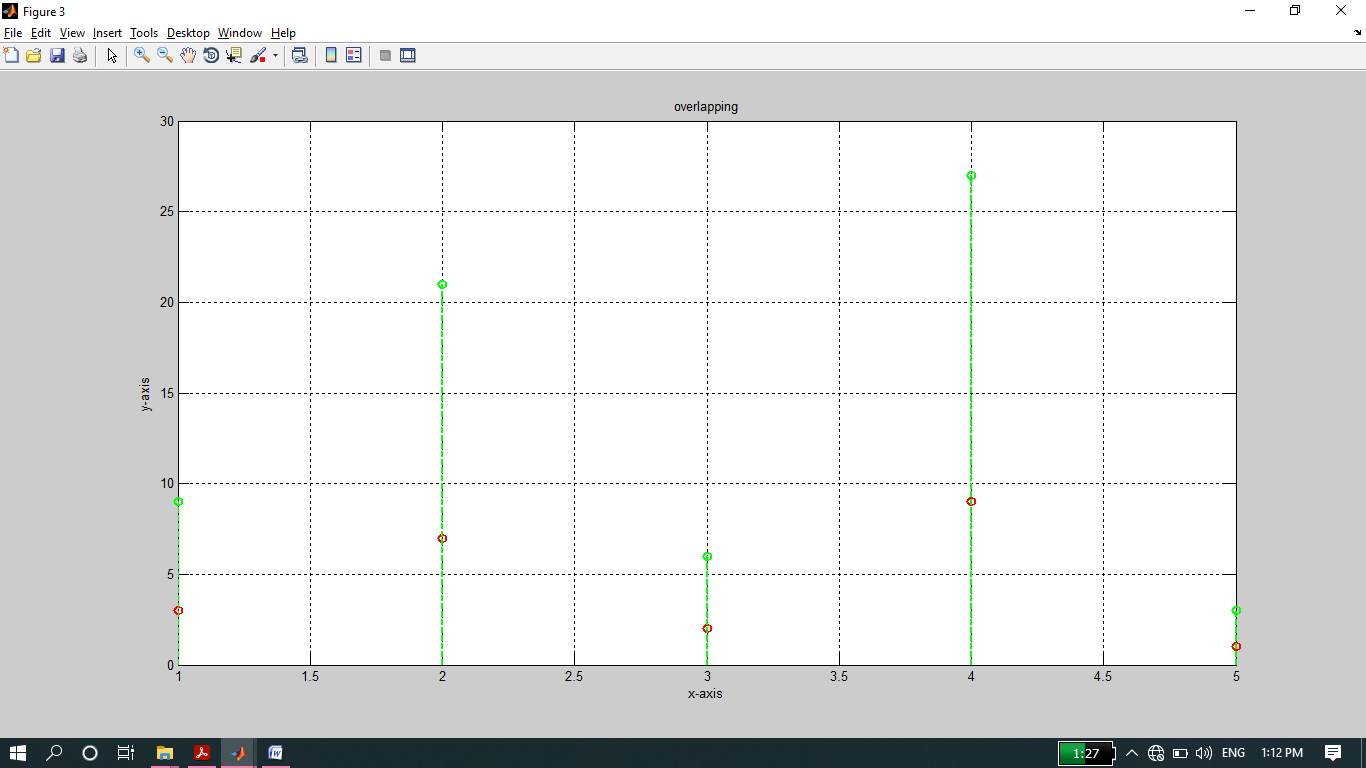
**Graphs:**

1. Separate graph:

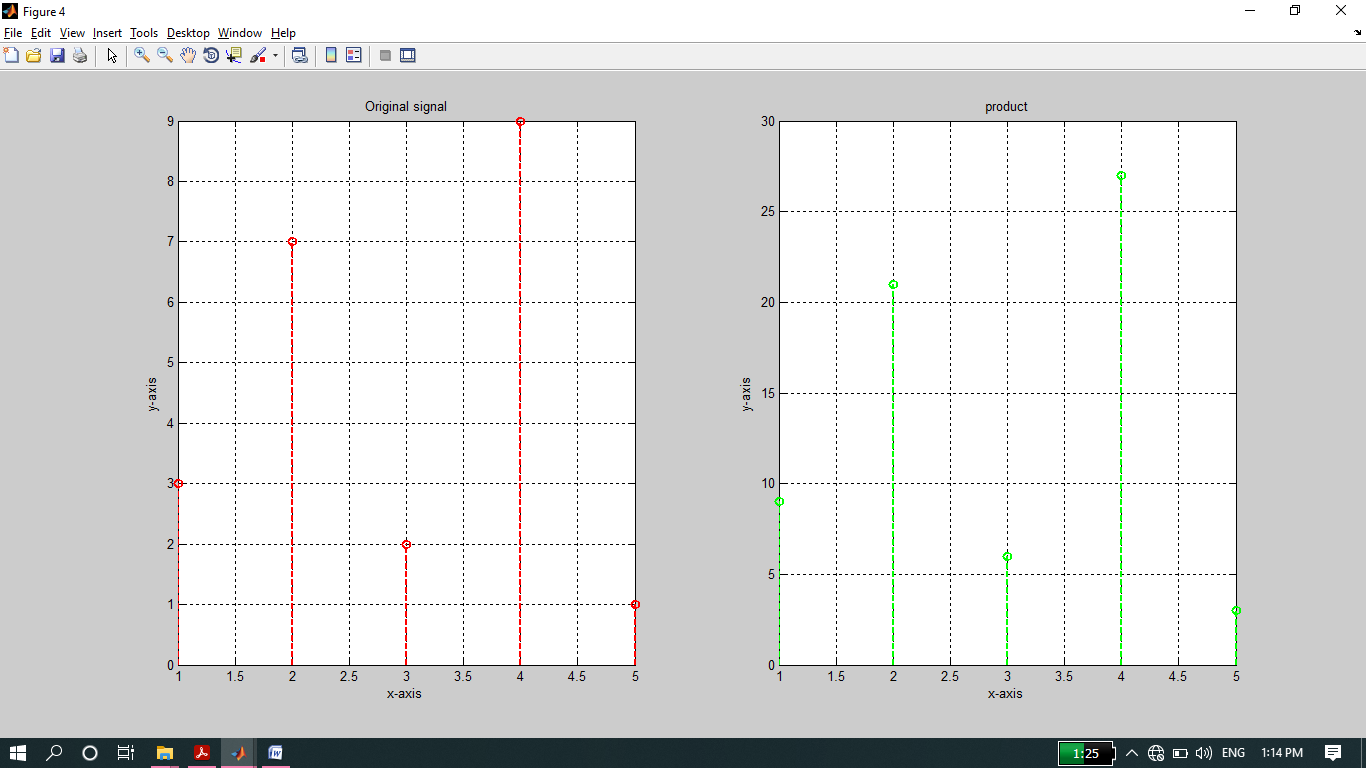




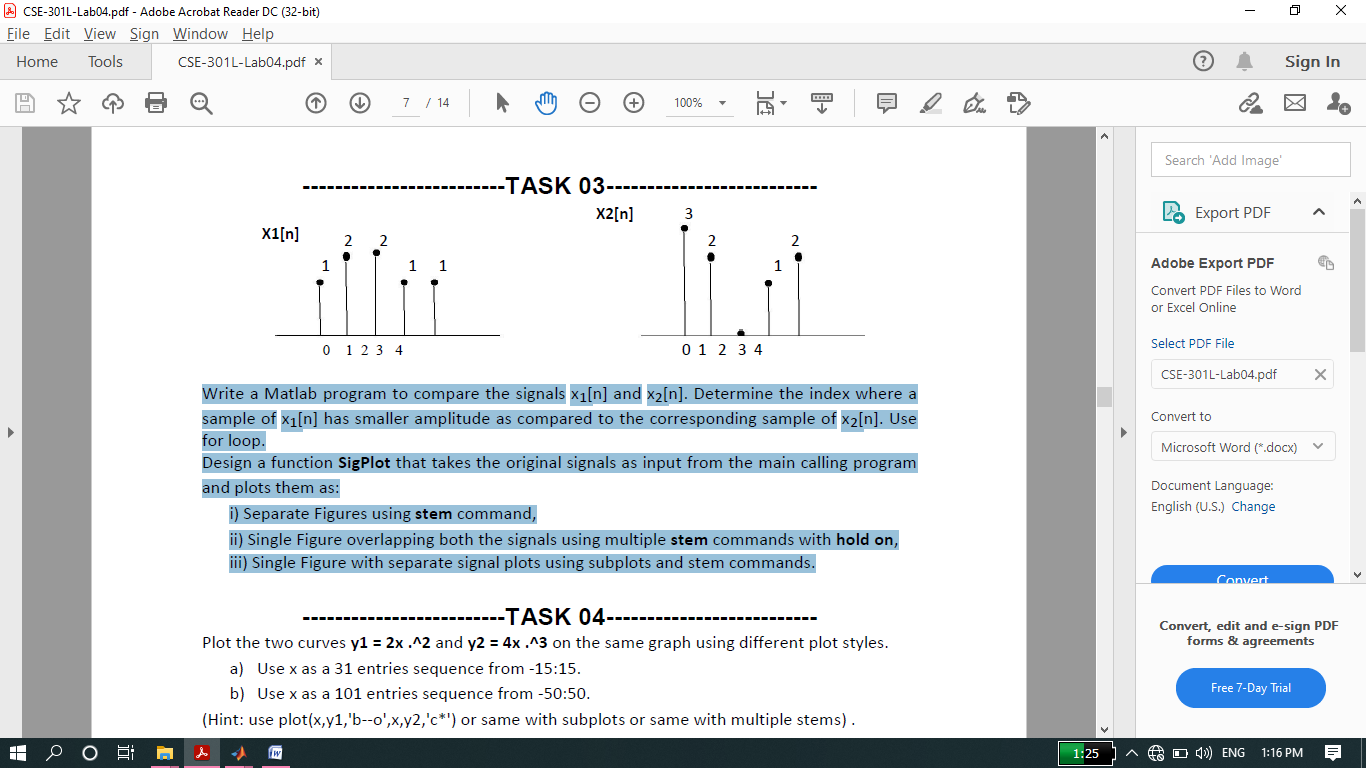
1. Overlapping:



1. All on single window:



**-------------------------TASK 03--------------------------**



Write a Matlab program to compare the signals x1[n] and x2[n]. Determine the index where a sample of x1[n] has smaller amplitude as compared to the corresponding sample of x2[n]. Use for loop.

Design a function **SigPlot** that takes the original signals as input from the main calling program and plots them as:

i) Separate Figures using **stem** command,

ii) Single Figure overlapping both the signals using multiple **stem** commands with **hold on**,

iii) Single Figure with separate signal plots using subplots and stem commands.

**Source code:**

function sigplot2

disp('\*\*\*\*\*\*\*\*task 03\*\*\*\*\*\*\*\*');

for i=1:5;

x1(i)=input('please enter the element of x1 ');

x2(i)=input('please enter the element of x2 ');

end

disp('vector x1= ');

disp(x1);

disp('vector x2= ');

disp(x2);

disp('the elements at the following indexs in x1 are smaller than the element at same position in x2')

for i=1:5;

if x1(i)<x2(i)

disp(i);

end

end

figure (1)

stem(x1,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('first vector');

grid on

figure (2)

stem(x2,'g --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('2nd vector');

grid on

figure (3)

stem(x1,'r --','Linewidth',2);

hold on

stem(x2,'g --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('overlapping');

hold on

grid on

figure (4)

subplot(2,1,1)

stem(x1,'r --','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('first vector');

grid on

subplot(2,1,2)

stem(x2,'g --','Linewidth',2);

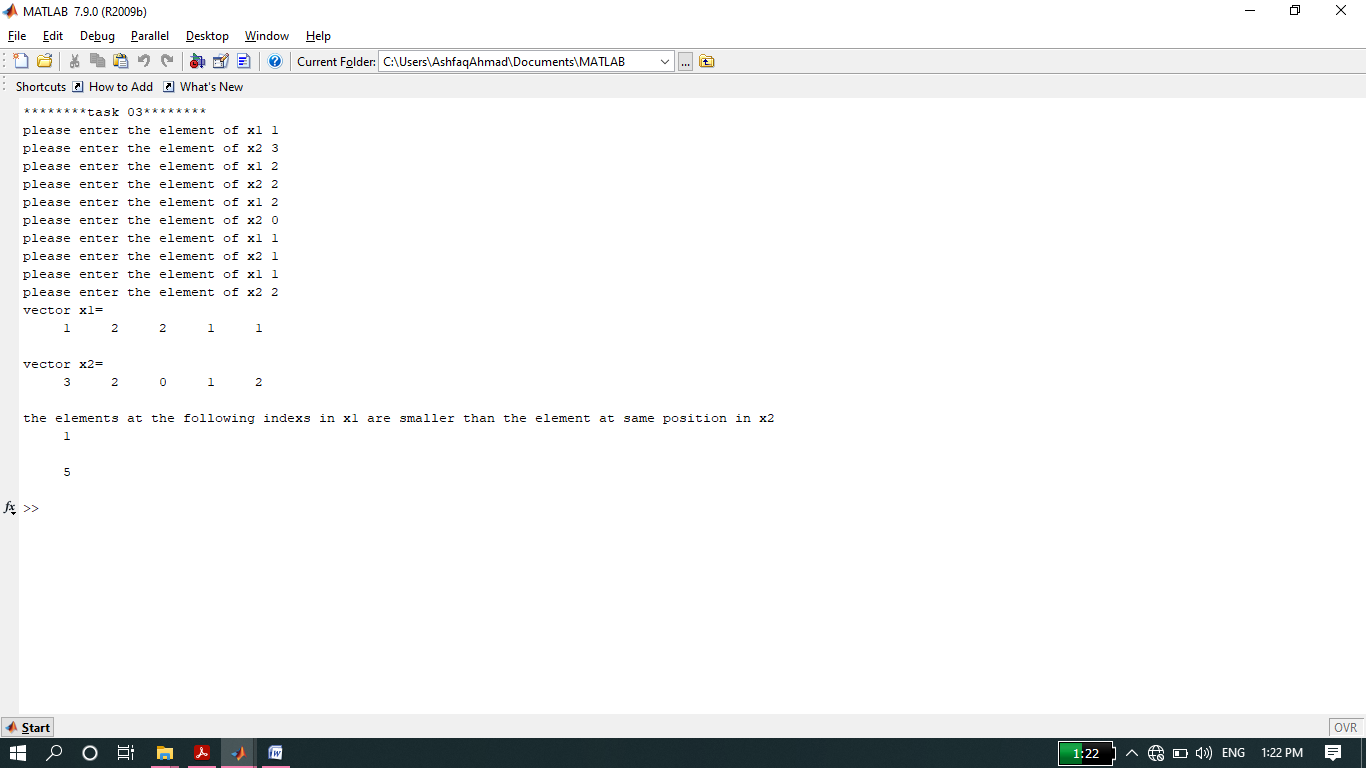
xlabel('x-axis');

ylabel('y-axis');

title('2nd vector');

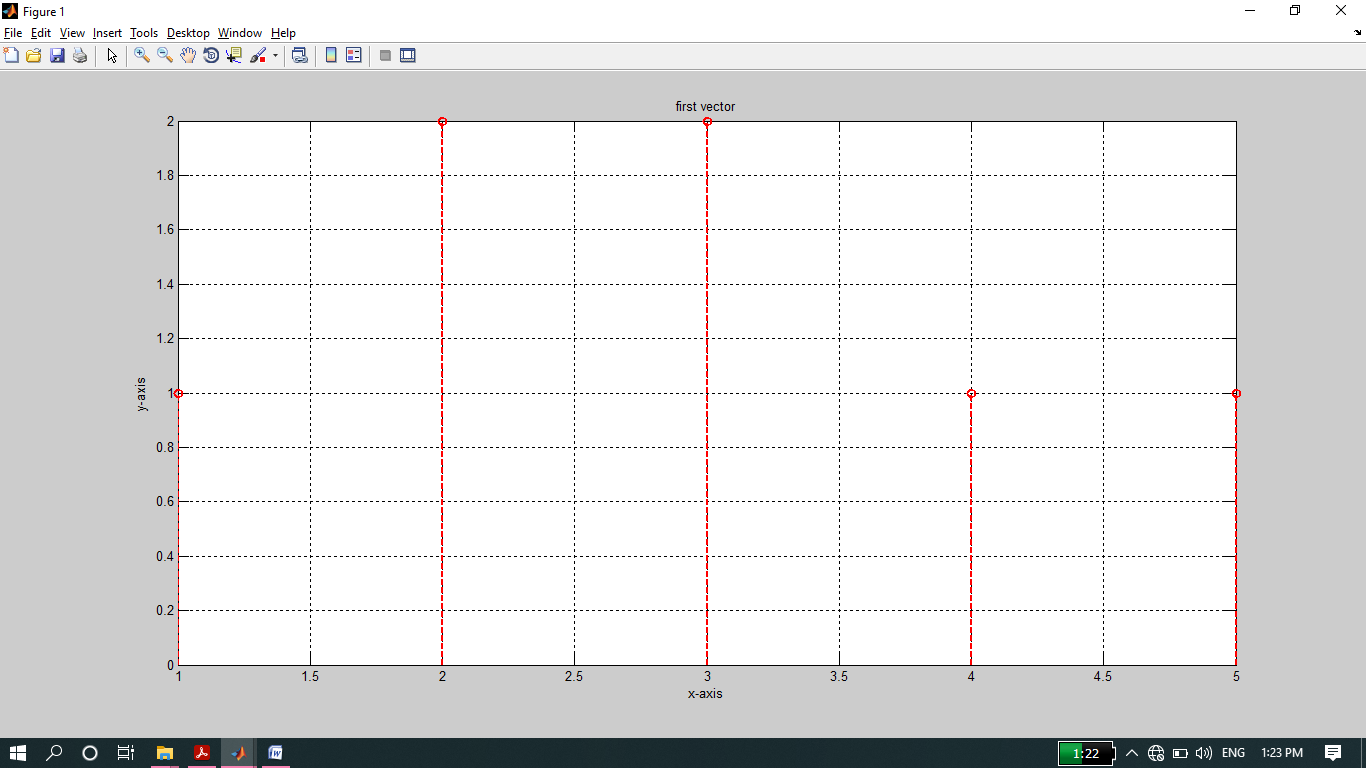
grid on

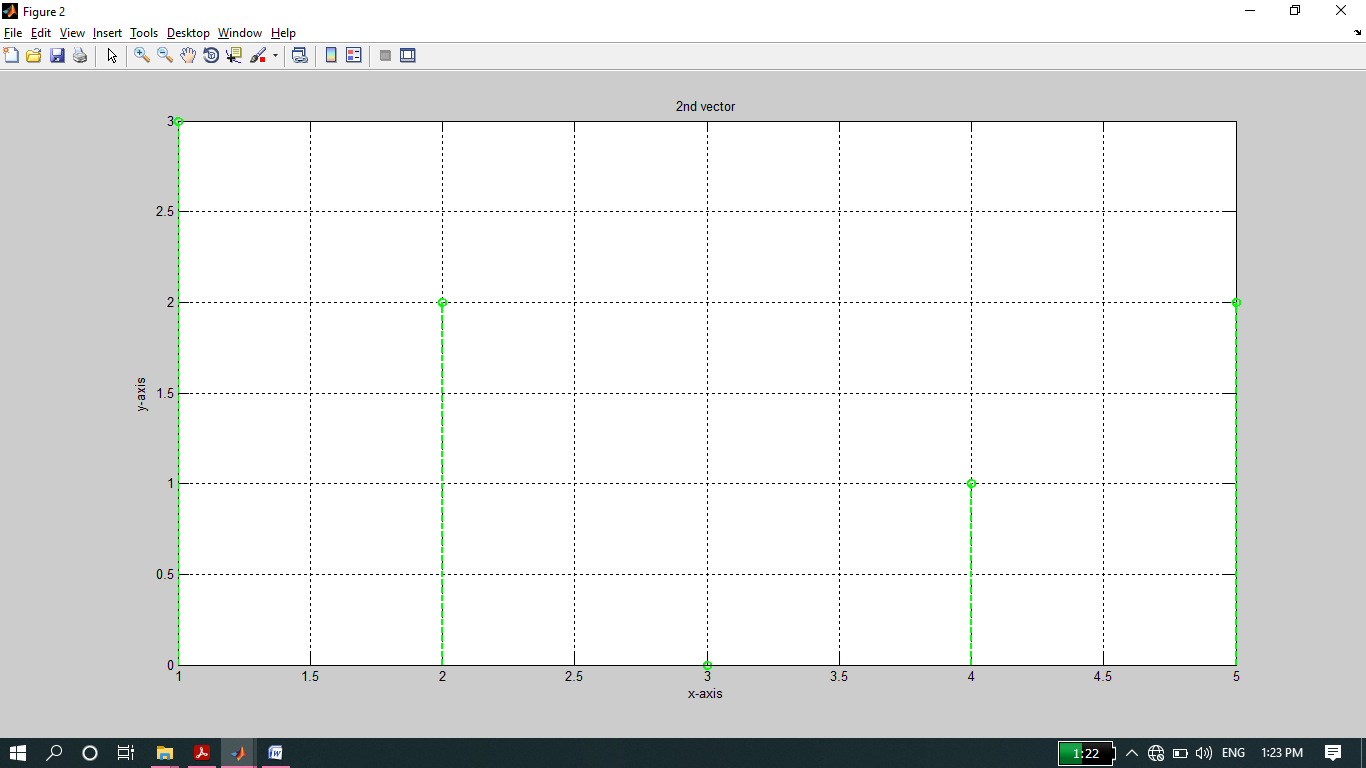
**Output:**



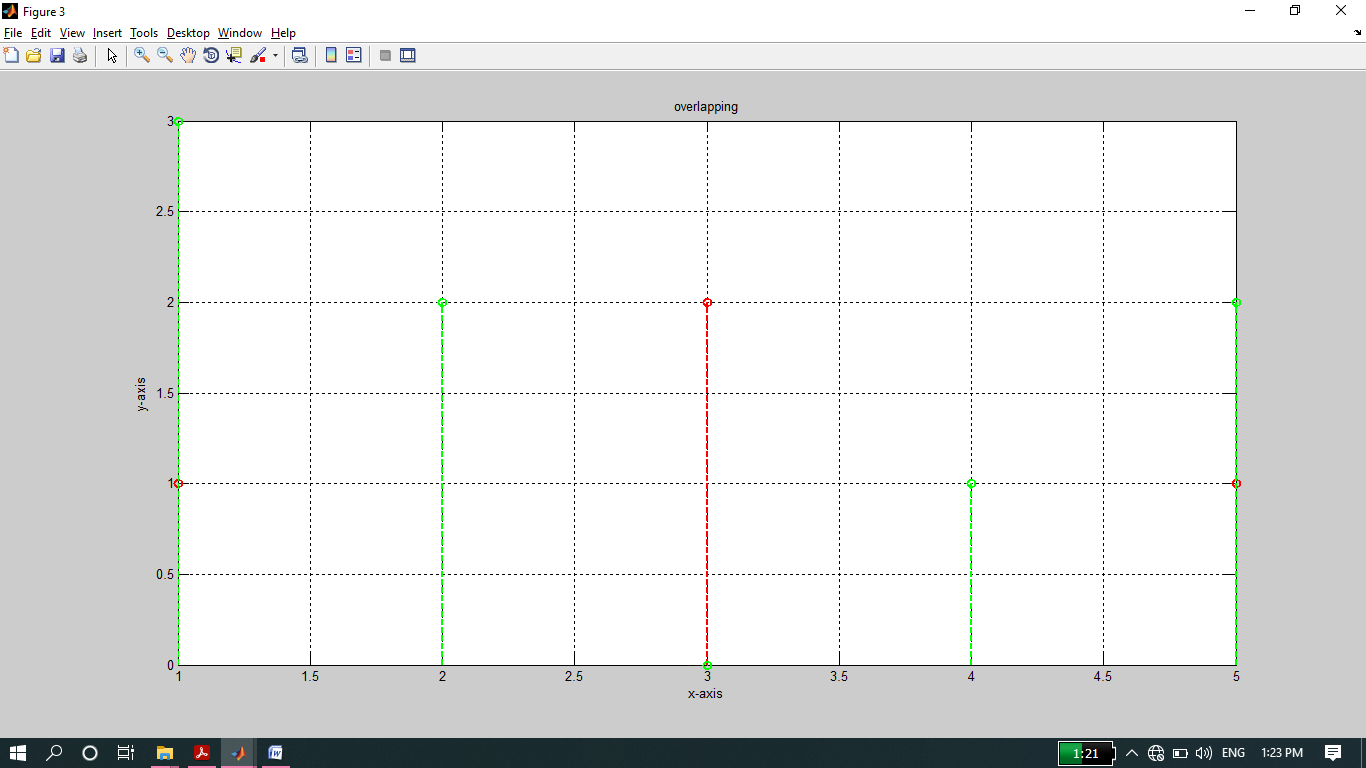
**Graphs:**

1. Separate graphs:

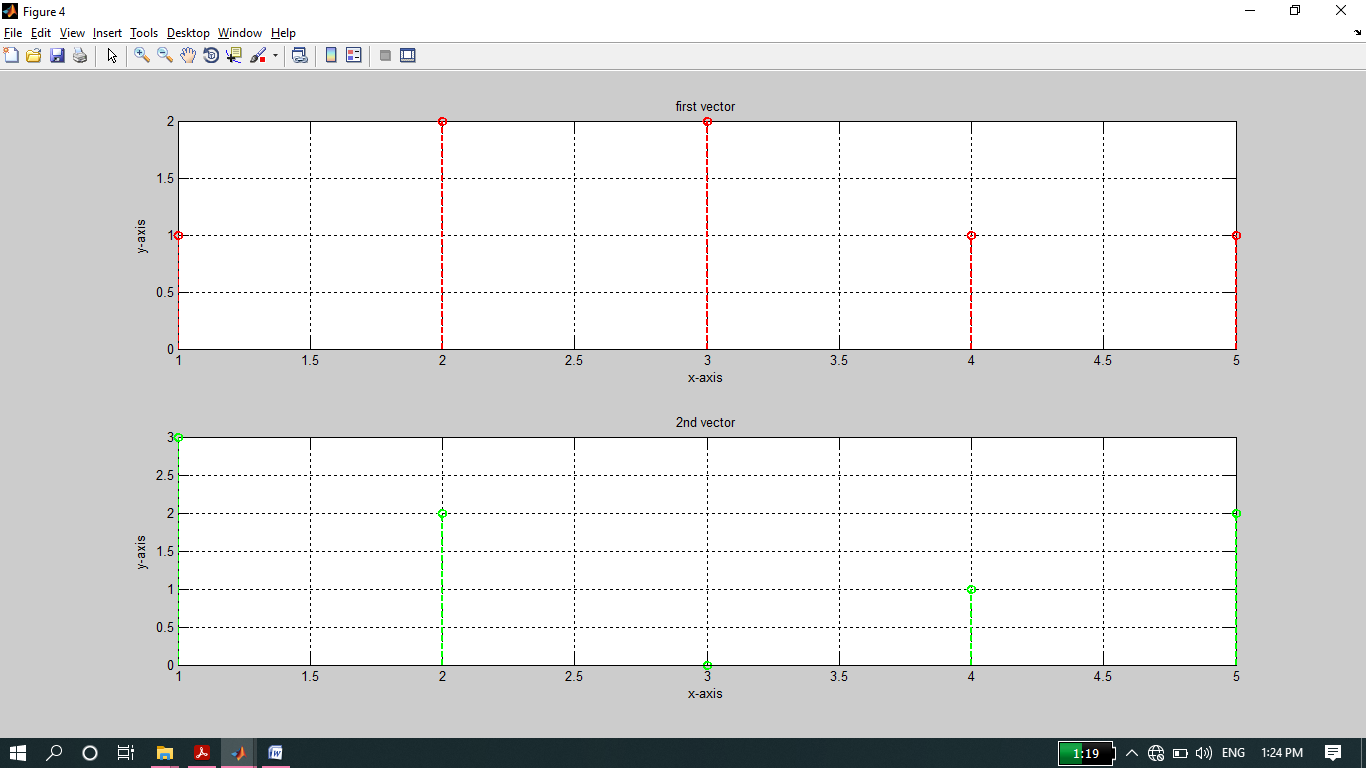




1. Overlapping:



1. All graph on Signal Window:



**-------------------------TASK 04--------------------------**

Plot the two curves **y1 = 2x .^2** and **y2 = 4x .^3** on the same graph using different plot styles.

a) Use x as a 31 entries sequence from -15:15.

b) Use x as a 101 entries sequence from -50:50.

(Hint: use plot(x,y1,'b--o',x,y2,'c\*') or same with subplots or same with multiple stems) .

**Source Code:**

function task04

disp('\*\*\*\*\*\*\*\*\* Task 04\*\*\*\*\*\*\*\*\*');

x=-15:15;

y1 = (2\*x.^2);

y2 = (4\*x.^3);

figure (1)

plot(x,y1,'b--o',x,y2,'c\*')

title('plot range -15:15')

grid on

x=-50:50;

y1 = (2\*x.^2);

y2 = (4\*x.^3);

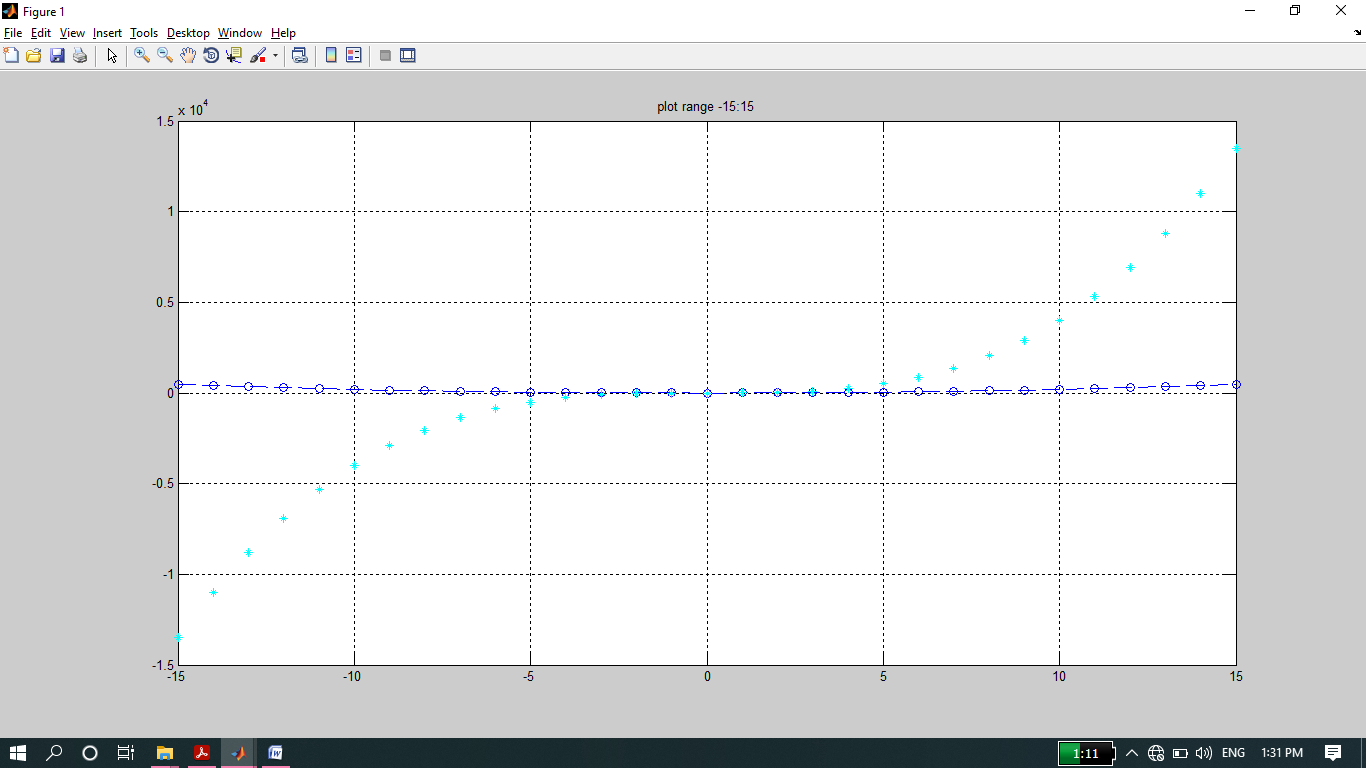
figure(2)

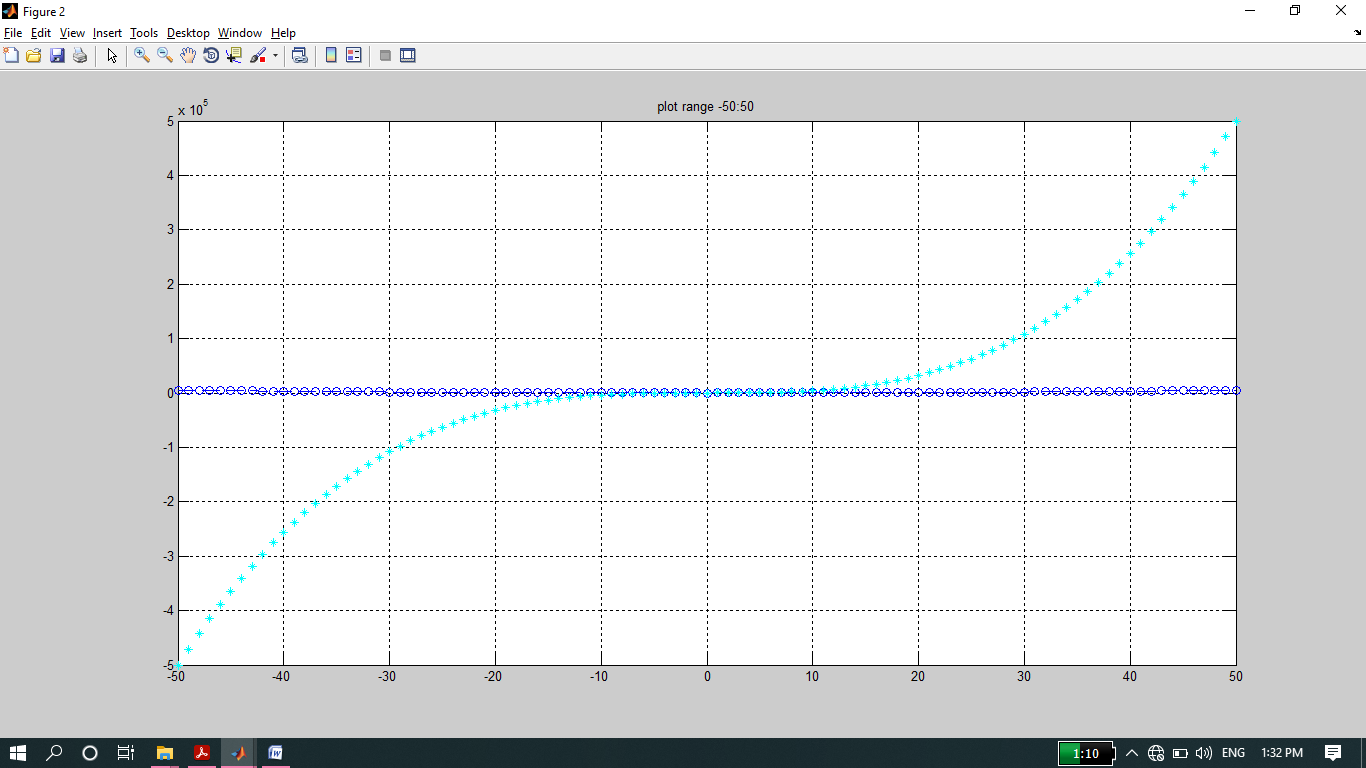
plot(x,y1,'b--o',x,y2,'c\*')

title('plot range -50:50')

grid on

**Output:**





**-------------------------TASK 05--------------------------**

Create a function **PlotCircle** that takes points x, y and radius r from user as inputs and generates a graph of circle centered at point (x,y) with a radius equal to r. Use **axis equal** to use equal data units along each coordinate direction and use **axis square** to view square axis.

(Hint: use circle equation: x-axis = r\*cos(theta)+x; y-axis=r\*sin(theta)+y; where theta=0:1/100:2\*pi and plot x-axis versus y-axis)

**Source code:**

function plotcircle

disp('\*\*\*\*\*\*\*\*\*task no 05\*\*\*\*\*\*\*');

x=input('please enter point x ');

y=input('please enter point y ');

r=input('please enter redius r');

theta=0:1/100:2\*pi;

xaxis=(r\*cos(theta))+x;

yaxis=r\*sin(theta)+y;

plot(xaxis,yaxis,'red','Linewidth',2)

xlabel('x-axis');

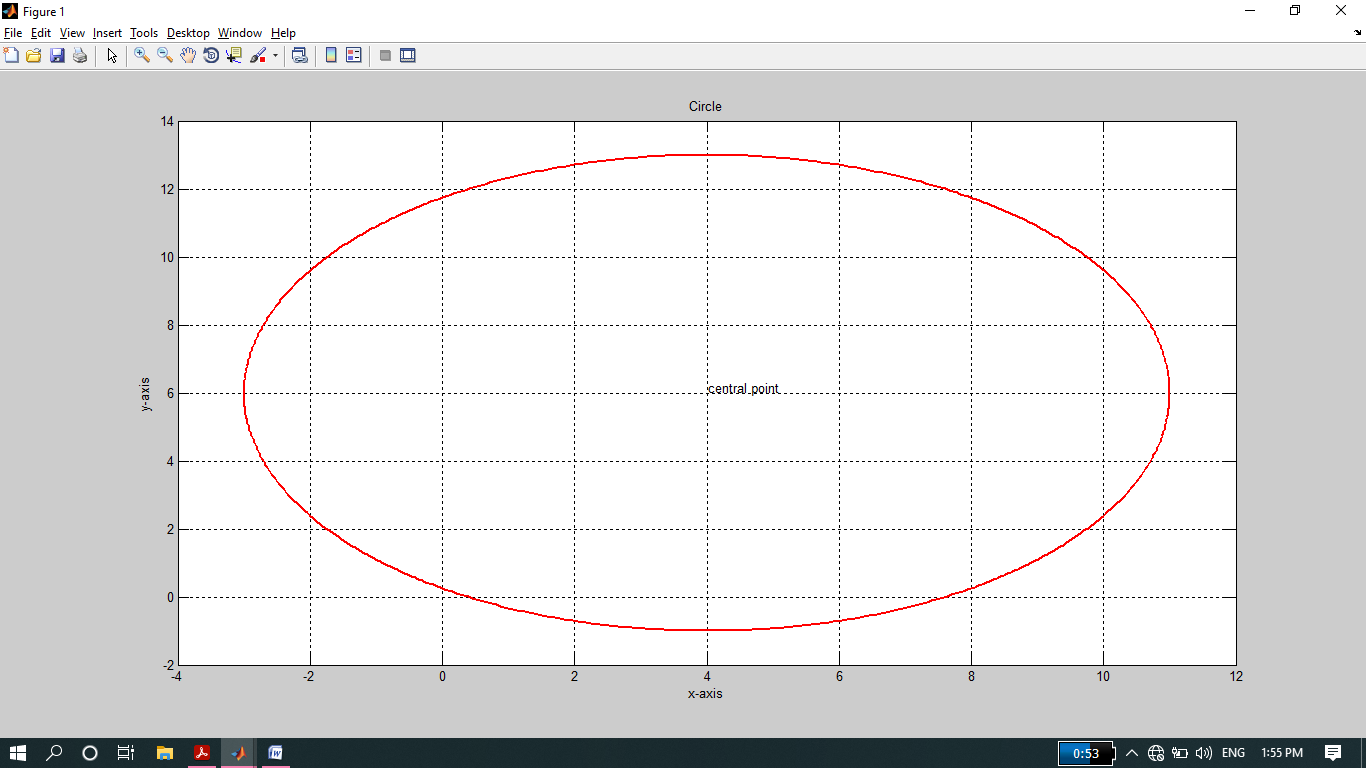
ylabel('y-axis');

title('Circle')

gtext('central point');

grid on

**Output:**



**-------------------------TASK 06--------------------------**

Given the signals:

X1[n] = 2δ[n] + 5δ[n‐1] + 8δ[n‐2] + 4δ[n‐3] + 3δ[n‐4]

X2[n] = δ[n‐4] + 4δ[n‐5] +3δ[n‐6] + 2δ[n‐7]

Write a Matlab program that adds these two signals. Plot the original signals as well as the final results using different plotting designs.

**Source code:**

clc

clear all

close all

disp('\*\*\*\*\*\*task 06\*\*\*\*\*\*\*\*\*');

x1=[2 0 0 0 0 0 0 0 ];

x2=[0 5 0 0 0 0 0 0 ];

x3=[0 0 8 0 0 0 0 0 ];

x4=[0 0 0 4 0 0 0 0 ];

x5=[0 0 0 0 3 0 0 0];

y1=[0 0 0 0 1 0 0 0];

y2=[0 0 0 0 0 4 0 0];

y3=[0 0 0 0 0 0 3 0];

y4=[0 0 0 0 0 0 0 2];

first\_sig=x1+x2+x3+x4+x5;

disp('first signal= ');

disp(first\_sig);

second\_sig=y1+y2+y3+y4;

disp('second signal= ');

disp(second\_sig);

sum\_sig=(first\_sig+second\_sig);

disp('sum signal= ');

disp(sum\_sig);

subplot(3,1,1)

plot(first\_sig,'g--+','Linewidth',2)

xlabel('x-axis');

ylabel('y-axis');

title('first signal');

grid on

subplot(3,1,2);

plot(second\_sig,'b--\*','Linewidth',2)

xlabel('x-axis');

ylabel('y-axis');

title('second signal');

grid on

subplot(3,1,3);

plot(sum\_sig,'r--\*','Linewidth',2)

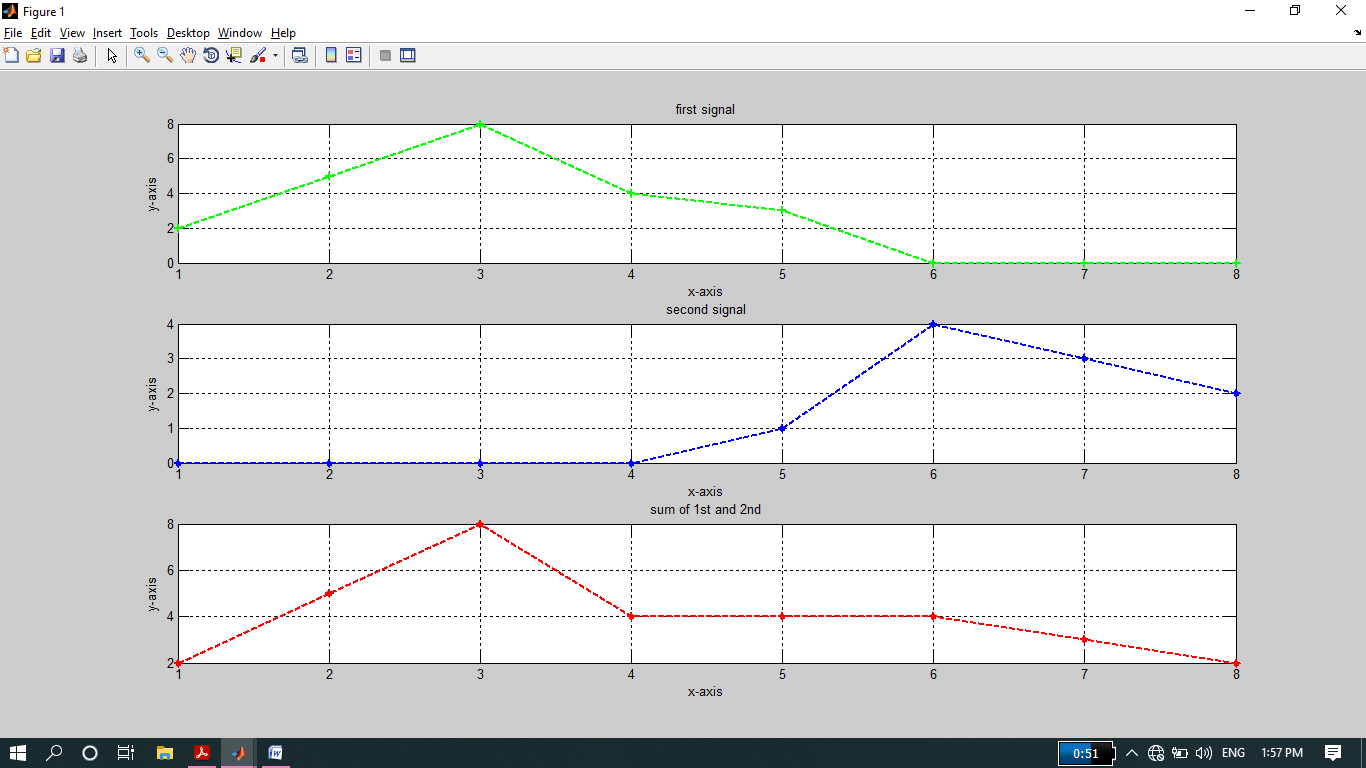
xlabel('x-axis');

ylabel('y-axis');

title('sum of 1st and 2nd');

grid on

**Output:**

****

**-------------------------TASK 07--------------------------**

Create a function **AmpScale** that takes a discrete‐time signal **S** and a threshold **T** from user and scales the amplitude of the input signal. The function saves and counts the number of samples with amplitude greater than **T** and less than **-T** and plots the amplitude scaled signal and gives the number of sample within the thresholds as output.

**Source code:**

function ampscale(signal,threshold)

disp('\*\*\*\*\*\*\*task no 7\*\*\*\*\*\*\*');

g=0;

l=0;

disp('the descrite time signal is :');

disp(signal);

for i=1:length(signal);

if signal(i)>threshold

signal1(i)=signal(i)-threshold;

g=g+1;

end

if signal(i)<(-threshold)

signal1(i)=signal(i)+threshold;

l=l+1;

end

if signal(i)>=(-threshold) && signal(i)<threshold;

signal1(i)=signal(i);

end

end

disp('the amplitude scaled signal is :');

disp(signal1);

disp('total no of amplitude samples greater than -ive of threshold are: ');

disp(g);

disp('total no of amplitude samples less than threshold are: ');

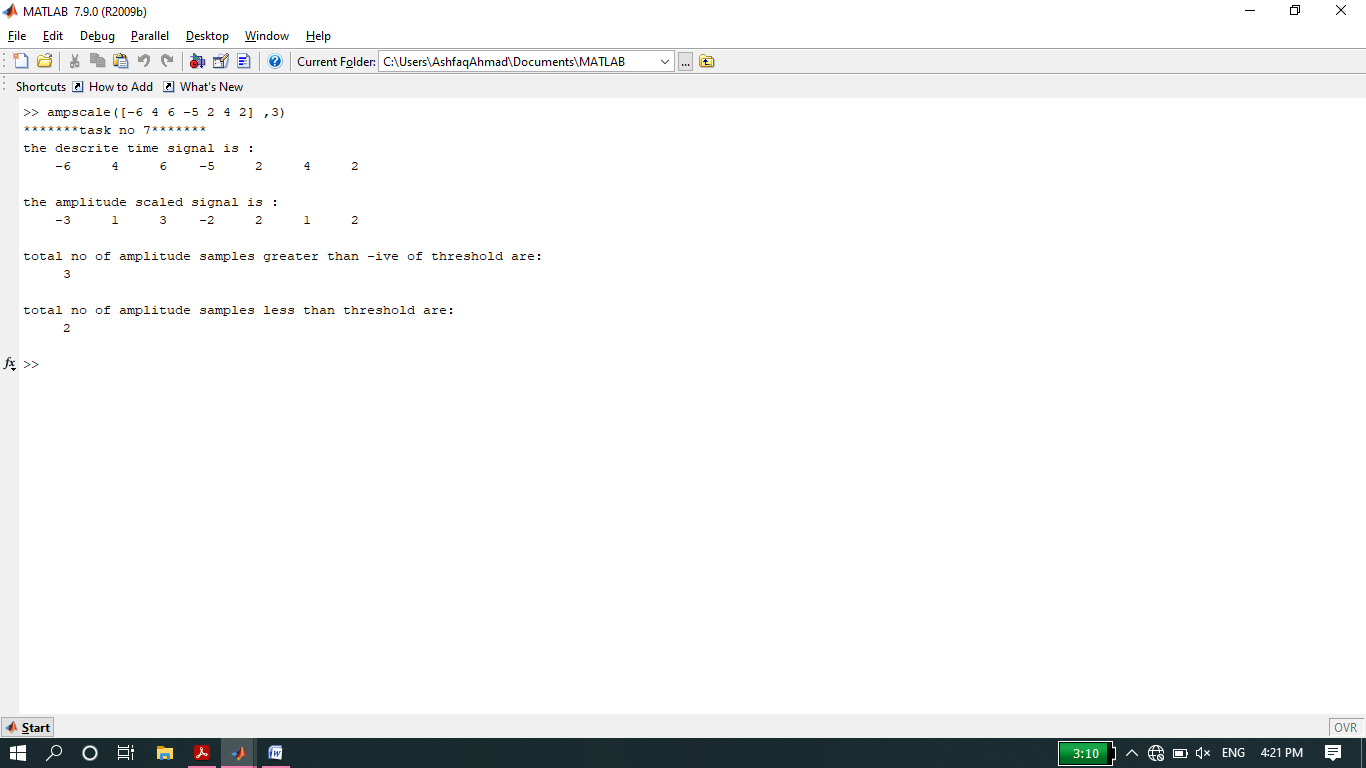
disp(l);

stem(signal1,'r', 'Linewidth',2)

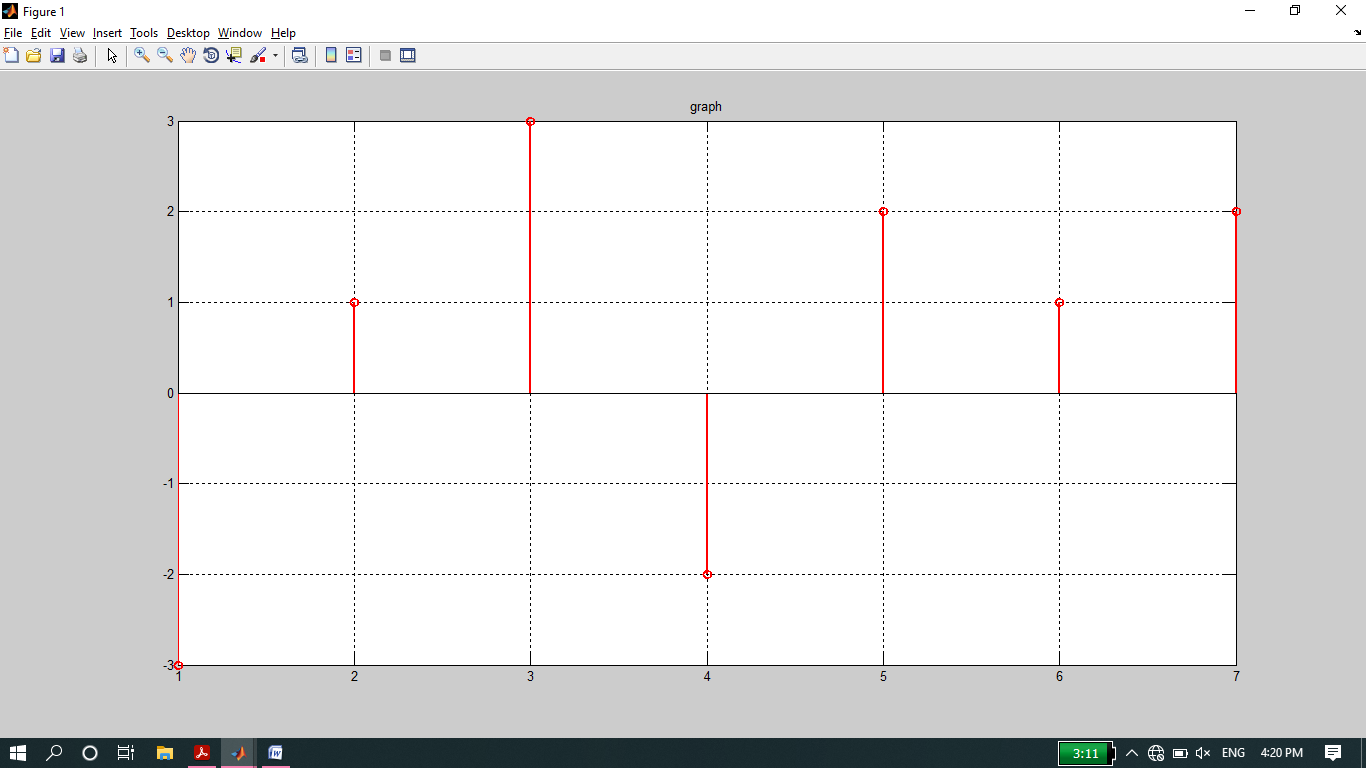
title('graph');

grid on

**Output:**

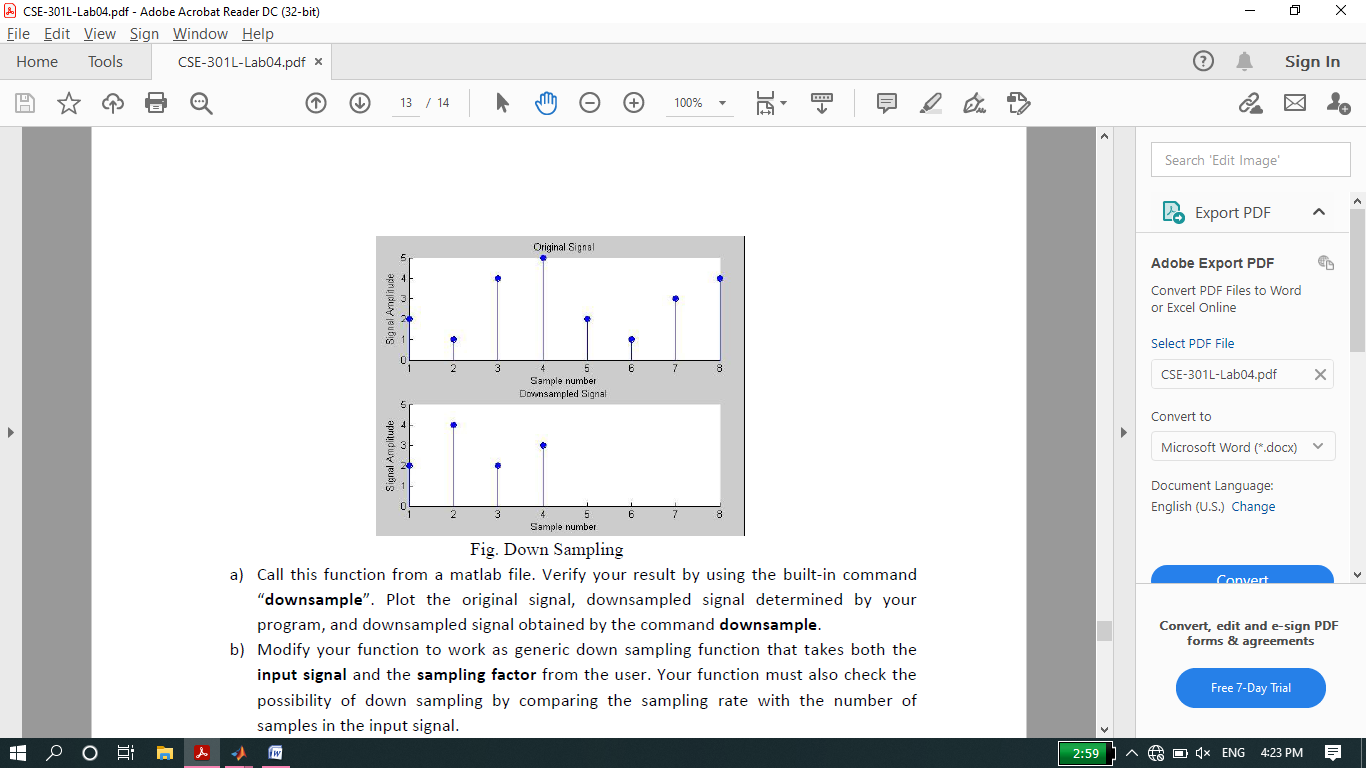
****

**Graph:**

****

**-------------------------TASK 08--------------------------**

Write your own function **downsamp** that takes a signal as input, retain odd numbered samples of the original signal and discard the even‐numbered (down sampling by 2). The function must return the down sampled version of that signal as output. See Fig for example.

****

**Source code:**

function downsampling(original\_signal)

disp('\*\*\*\*\*\*task no 8\*\*\*\*\*\*\*\*\*');

n=input('please input the length of signal: ');

for i=1:n;

original\_signal(i)=input(['please enter sample no' num2str(i) ' = ']);

end

disp('the orignal signal is:')

disp(original\_signal)

for i=1:n;

if mod(original\_signal(i),2)~=0;

Down\_sample(i)=original\_signal(i);

end

end

disp('the orignal signal after down-sampling is:');

disp(Down\_sample);

subplot(2,1,1)

stem(original\_signal,'g','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Original signal');

grid on

subplot(2,1,2)

stem(Down\_sample,'r','Linewidth',2);

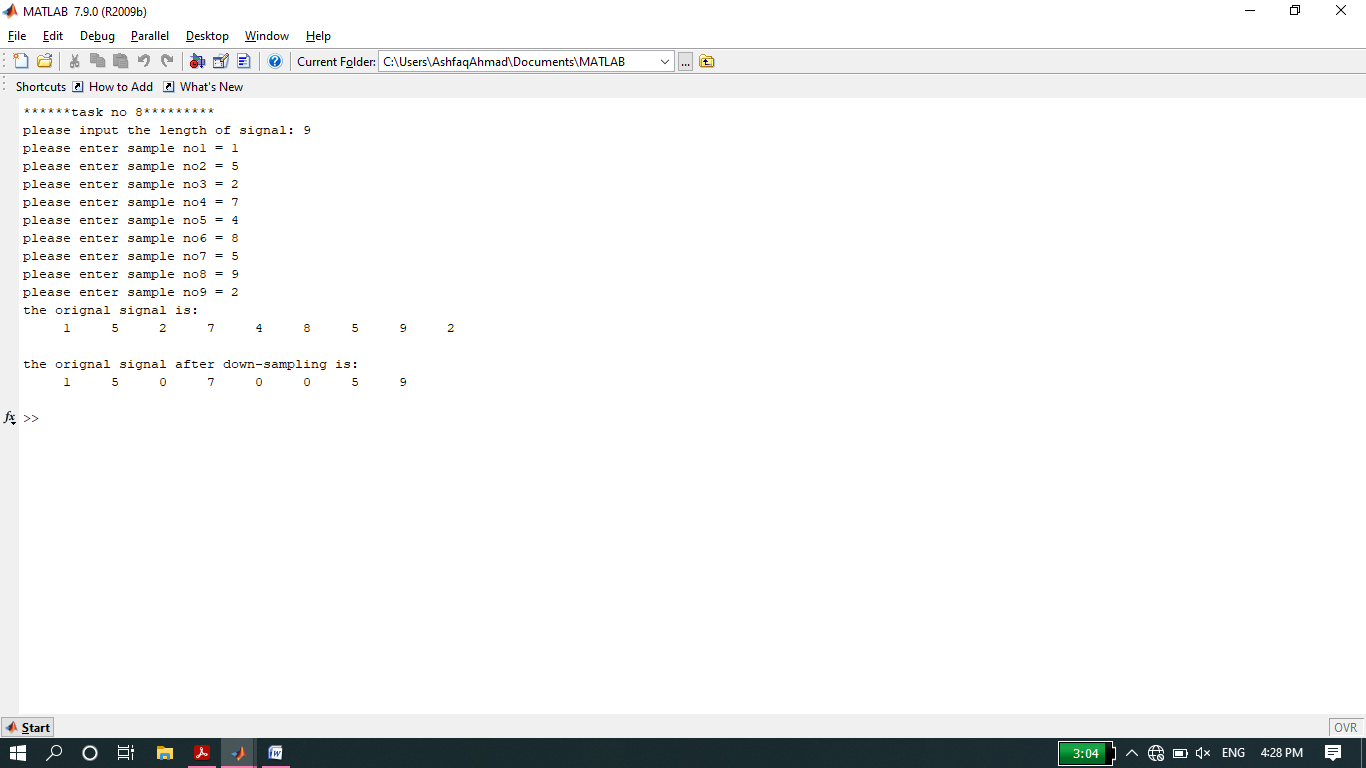
xlabel('x-axis');

ylabel('y-axis');

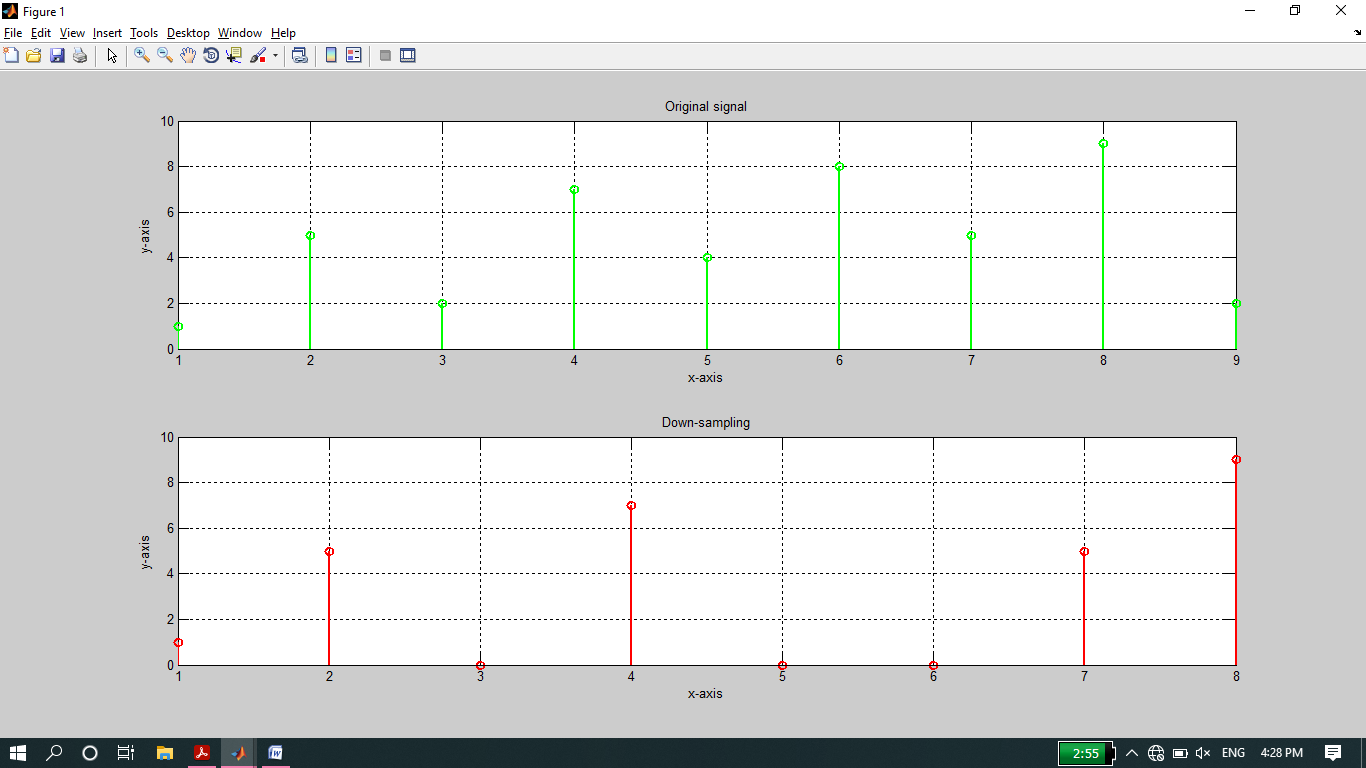
title('Down-sampling');

grid on

**Output:**

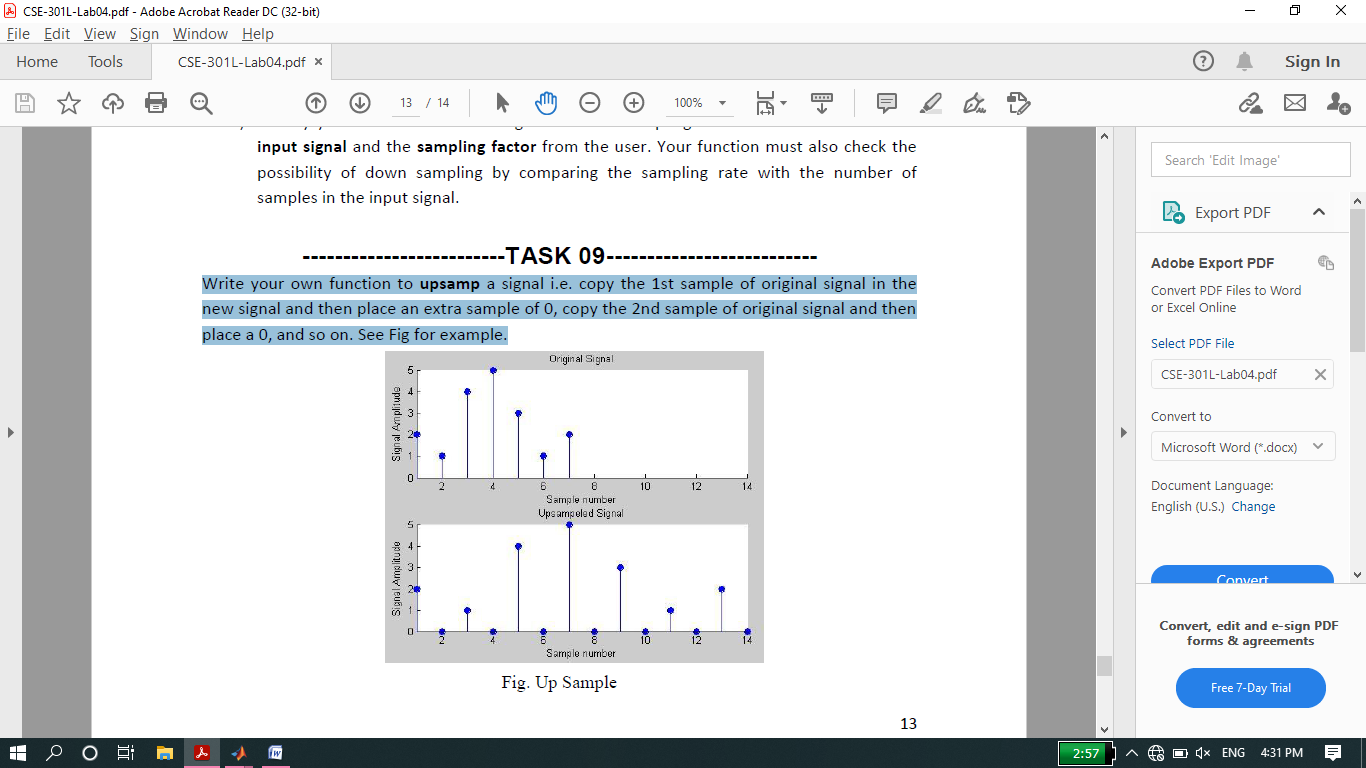


**Graph:**



**-------------------------TASK 09--------------------------**

Write your own function to **upsamp** a signal i.e. copy the 1st sample of original signal in the new signal and then place an extra sample of 0, copy the 2nd sample of original signal and then place a 0, and soon. See Fig for example.



**source code:**

function Upsampling(original\_signal)

disp('\*\*\*\*\*\*task no 09\*\*\*\*\*\*\*\*\*');

n=input('please input the length of signal: ');

for i=1:n;

original\_signal(i)=input(['please enter sample no' num2str(i) ' = ']);

end

disp('the orignal signal is:')

disp(original\_signal)

i=1;

for j=1:2\*n;

if mod(j,2)==0;

up\_sample(j)=0;

else

up\_sample(j)=original\_signal(i);

i=i+1;

end

end

disp('the orignal signal after down-sampling is:');

disp(up\_sample);

subplot(2,1,1)

stem(original\_signal,'g','Linewidth',2);

xlabel('x-axis');

ylabel('y-axis');

title('Original signal');

grid on

subplot(2,1,2)

stem(up\_sample,'r','Linewidth',2);

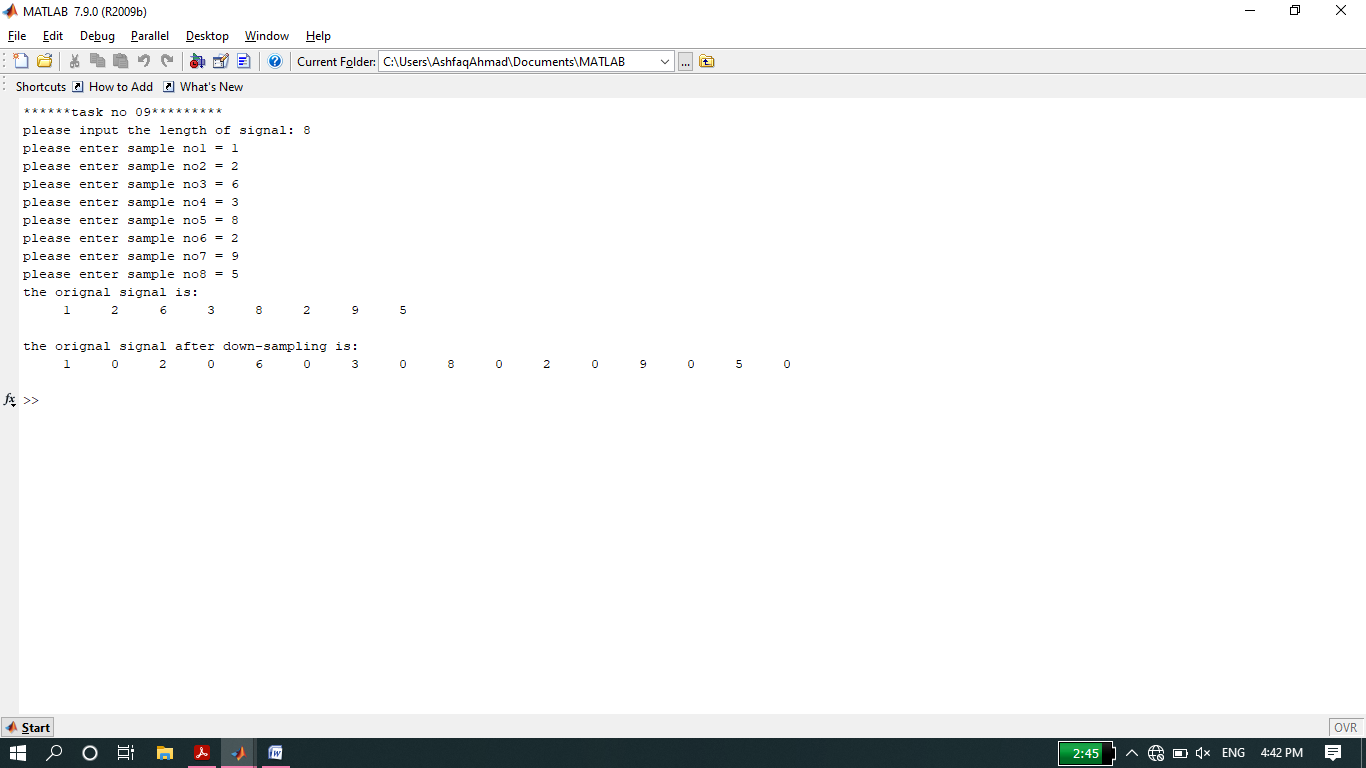
xlabel('x-axis');

ylabel('y-axis');

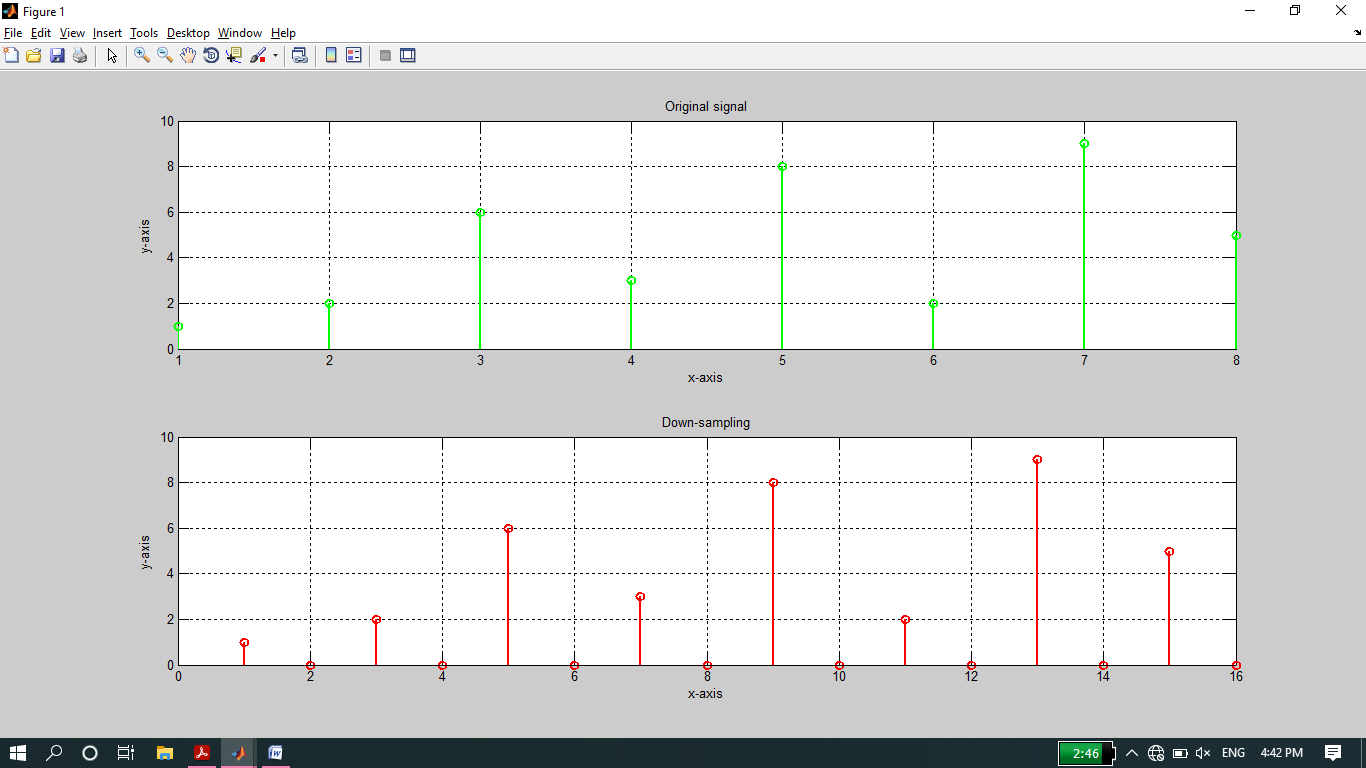
title('Down-sampling');

grid on

**Output:**



**Graph:**



**-------------------------TASK 10--------------------------**

Plotting **3-D graphics** with MatLab. This is a complementary task for practicing 3d graphs in MatLab. **Surf** command is used in Matlab for plotting 3D graphs, the **meshgrid** command is used for setting up 2D plane

**Source code:**

clc

clear all

close all

disp('\*\*\*\*\*\*\*task 10\*\*\*\*\*\*');

[x,y] = meshgrid([-2:.2:2]);

Z = x.\*exp(-x.^2-y.^2);

figure

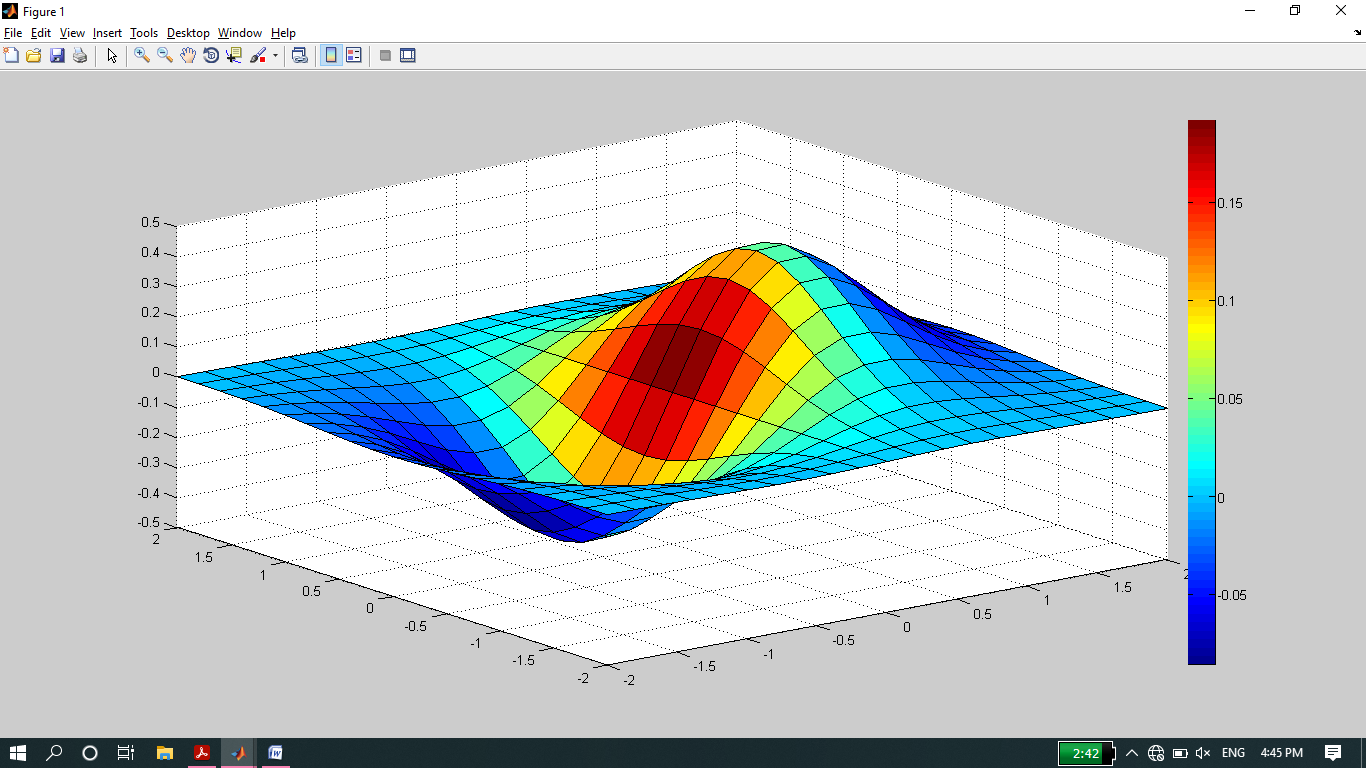
% surface plot, with gradient(Z) determining color distribution

surf(x,y,Z,gradient(Z))

% display color scale, can adjust location similarly to legend

colorbar

**Output:**



**THE END**