**INTRODUCTION TO SIGNALS IN MATLAB**

**LAB # 04**



**Spring 2022**

**CSE301L Signals & Systems Lab**

Submitted by: **Ebtihaj Abdullah**

Registration No. : **20PWCSE1885**

Class Section: **A**

“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**

Friday, May 27, 2022

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

## Lab Objective(s):

Objectives of this Lab are;

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

## 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.

**Problem Analysis:**

Working on signals is very easy in MATLAB. Here we see how we operate on signals in MATLAB

**Algorithm:**

* Write signals
* Add them
* Multiply them
* Display as separate and combined graphs

**Code:**

*x=[2 5 8 4 3];*

*y=[4 3 2 0 0];*

*%a)*

*disp('Vector Sum:')*

*addition=x+y;*

*disp(addition)*

*disp('Vector Product:')*

*product=x.\*y;*

*disp(product)*

*%b)*

*disp('Using for loop:')*

*for j=1:5*

*S(j)=x(j)+y(j);*

*end*

*disp('Sum:')*

*disp(S)*

*for j=1:5*

*P(j)=x(j)\*y(j);*

*end*

*disp('Product:')*

*disp(P)*

*figure(1)*

*plot(addition,'-m')*

*title('Sum')*

*xlabel('Sample No.')*

*ylabel('Signal Amp')*

*hold on*

*figure(2)*

*plot(product,'-m')*

*title('Product')*

*xlabel('Sample No.')*

*ylabel('Signal Amp')*

*figure(3)*

*subplot(1,2,1)*

*plot(addition,'-m')*

*title('Sum')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*subplot(1,2,2)*

*plot(product,'-m')*

*title('Product')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*figure(4)*

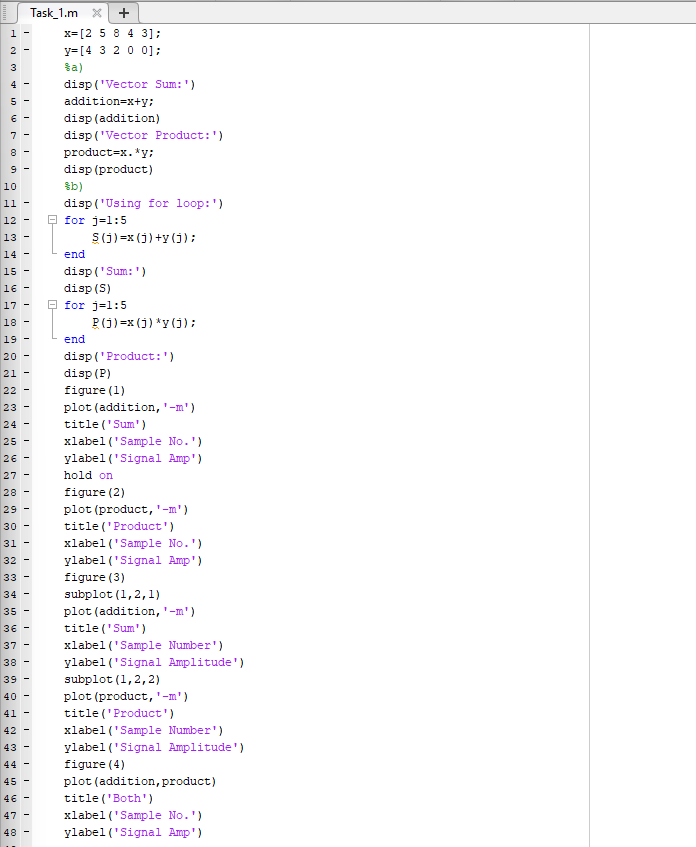
*plot(addition,product)*

*title('Both')*

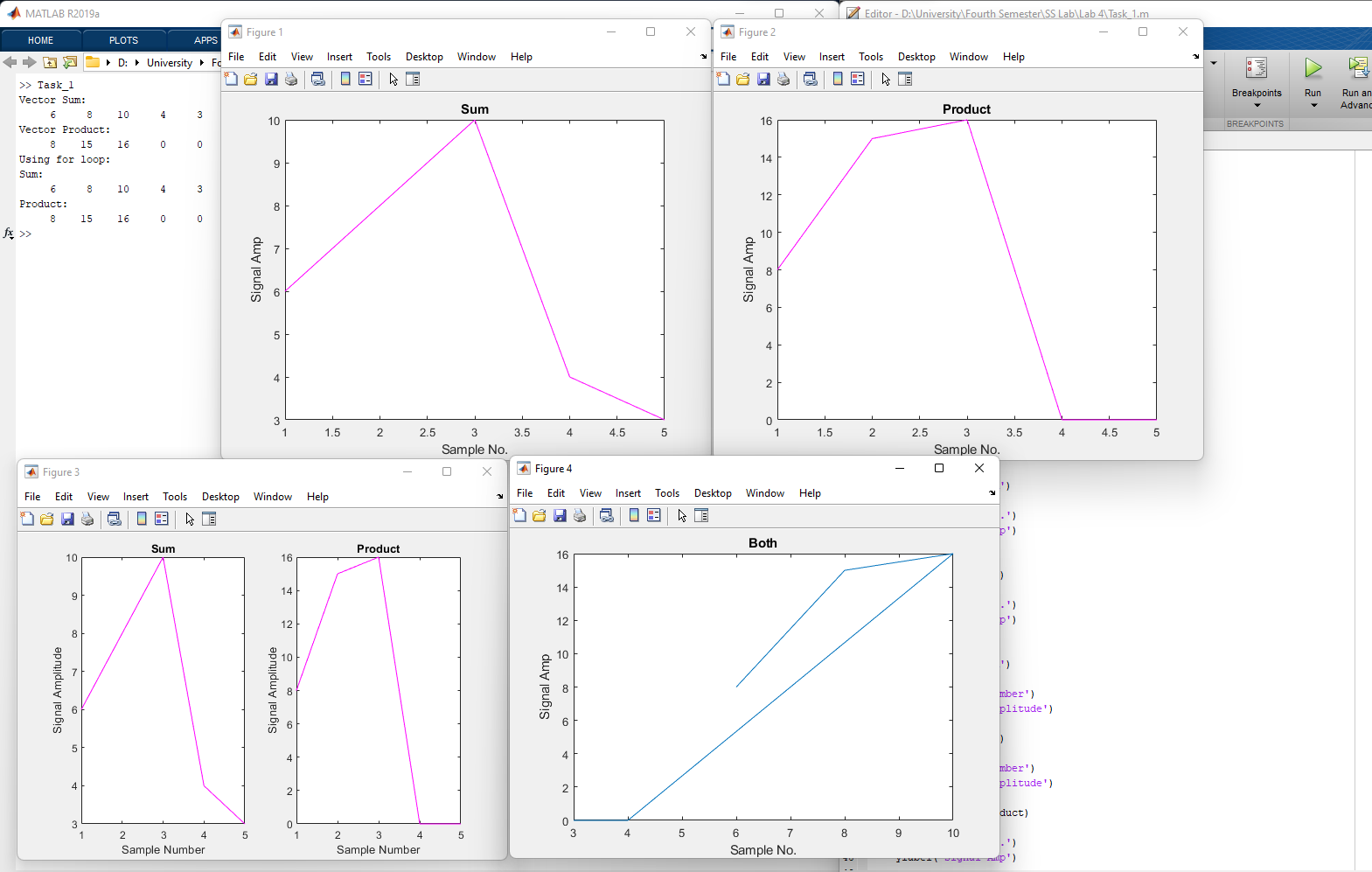
*xlabel('Sample No.')*

*ylabel('Signal Amp')*

**Code SS:**

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

Hence studying signals is very easy in MATLAB.

## 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.

**Problem Analysis:**

We see how we can perform amplification operations on signals in MATLAB.

**Algorithm:**

* Write the signal
* Scale it in scalesig function
* Display the resultant signal

**Code:**

*%Main*

*n=input("Input n");*

*x=input("Input x");*

*figure(1)*

*stem(n,x);*

*title('Unscaled');*

*b=input("Input scaling factor");*

*[n2]=Scalesigfunc(n,b);*

*figure(2)*

*stem(n2,x);*

*title('Scaled');*

*figure(3)*

*hold on;*

*stem(n,x);*

*stem(n2,x);*

*hold off*

*title('Both');*

*xlabel('Sample No.');*

*ylabel('Signal Amp');*

*figure(4)*

*subplot(1,2,1);*

*stem(n,x);*

*title('Unscaled');*

*subplot(1,2,2);*

*stem(n2,x);*

*title('Scaled');*

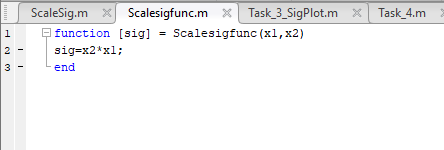
*%ScaleSigFunction*

function [sig] = Scalesigfunc(x1,x2)

sig=x2\*x1;

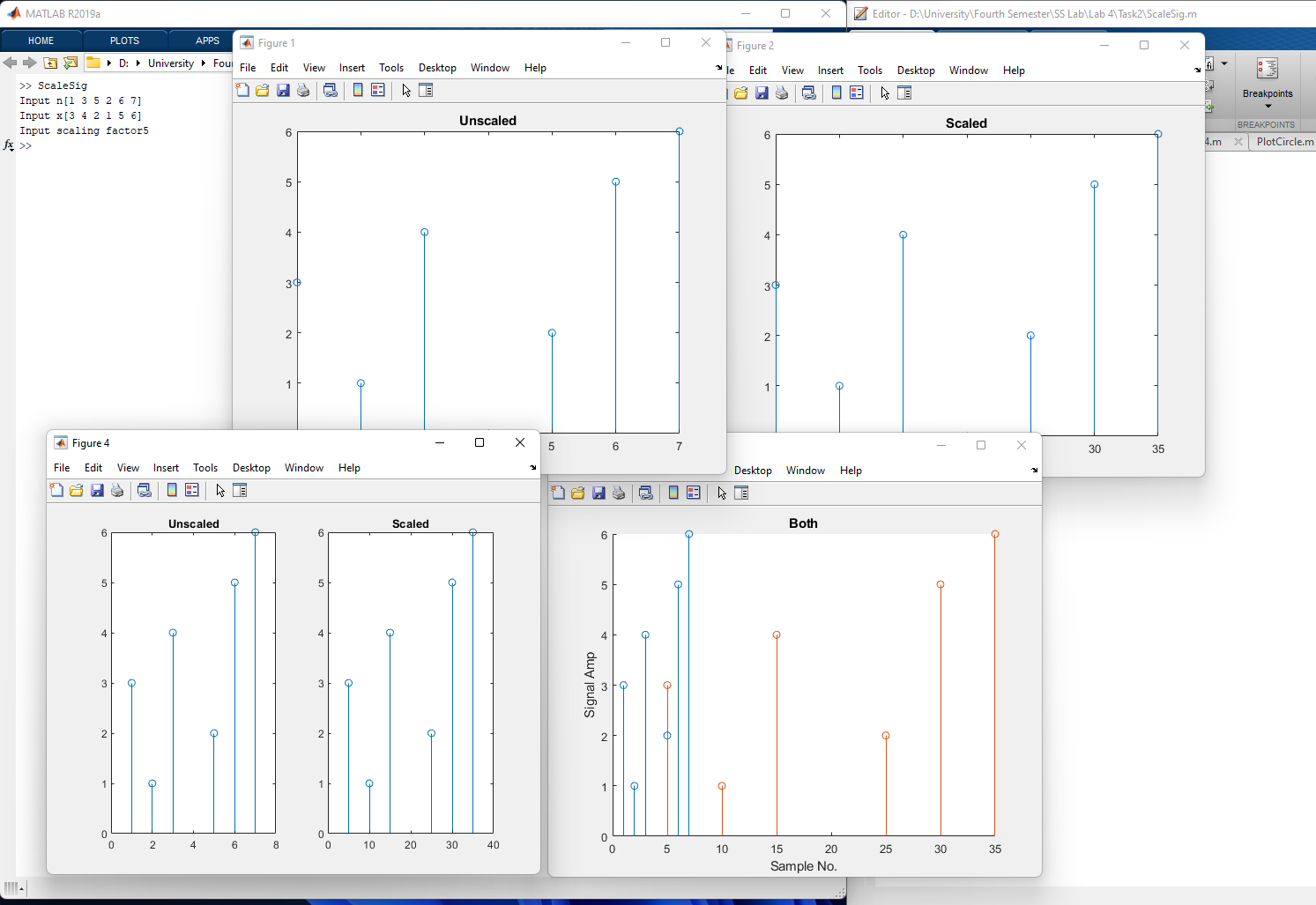
end

**Code SS:**

****

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

Amplification of signals can be done in matlab.

## 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.

**Problem Analysis:**

We See How To Perform Signal Comparison In MATLAB

**Algorithm:**

* Write two signals
* Compare them in the sigplot function

**Code:**

*function AmpScale*

*Si=input('Enter a signal:');*

*th=input('Enter threshold:');*

*neg\_th=-th;*

*count=0;*

*for i=1:length(Si)*

*if(Si(i)>th&&Si(i)<neg\_th)*

*count=count+1;*

*end*

*end*

*disp('The number of samples greater than thredshold or less than threshold:')*

*disp(count);*

*figure(1)*

*plot(Si,'-m')*

*title('Plot 1')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*figure(2)*

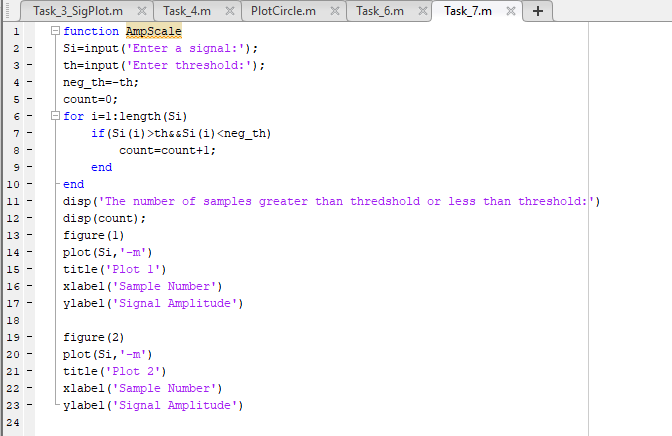
*plot(Si,'-m')*

*title('Plot 2')*

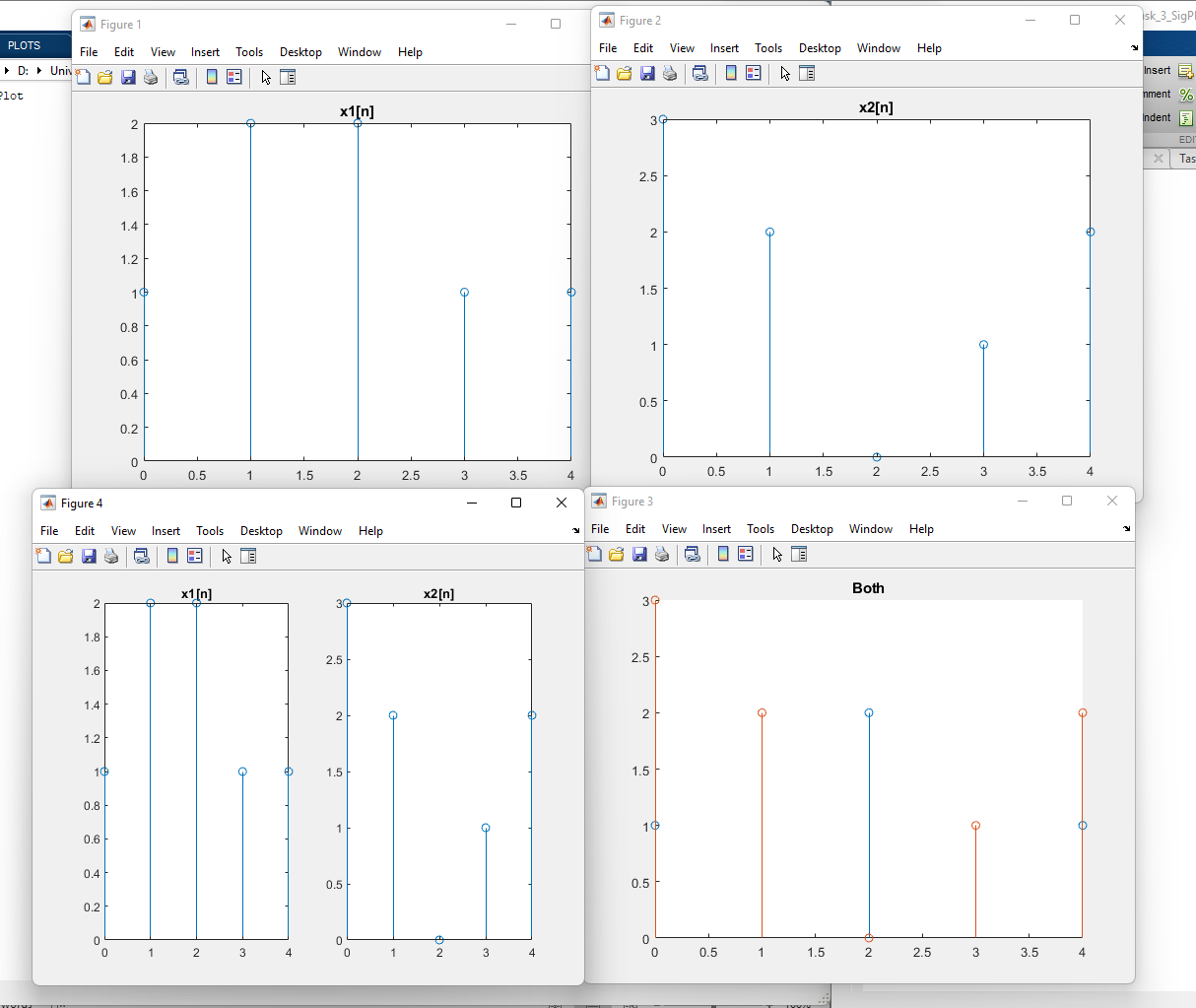
*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

**Code SS:**



**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

We successfully compared two signals in MATLAB

## 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) .

**Problem Analysis:**

We can use MATLAB to plot signal curves in different plotting styles .

**Algorithm:**

* Enter the signals.
* Plot them using different types of plotting methods

**Code:**

*%Part 1*

*x=-15:1:15;*

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

*y2=4\*x+3;*

*figure(1)*

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

*title('Part 1')*

*xlabel('Sample No.')*

*ylabel('Signal Amp')*

*%Part 2*

*x=-50:1:50;*

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

*y2=4\*x+3;*

*figure(2)*

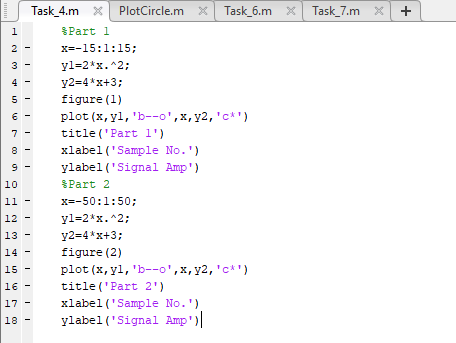
*plot(x,y1,'b--o',x,y2,'c\*')*

*title('Part 2')*

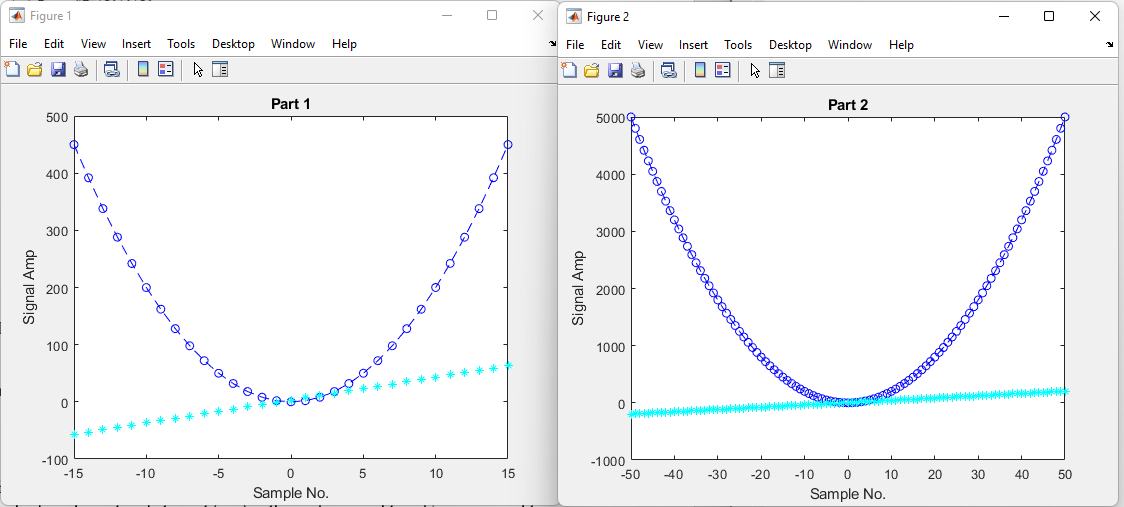
*xlabel('Sample No.')*

*ylabel('Signal Amp')*

**Code SS:**



**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

MATLAB can plot signals in various methods.

## 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)

**Problem Analysis:**

We use MATLAB and Signals to plot a cricle

**Algorithm:**

* Enter the parameters of a circle
* Plot the circle

**Code:**

*function PlotCircle*

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

*x=input('Input x: ');*

*y=input('Input y: ');*

*r=input('Input r: ');*

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

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

*plot(xax,yax);*

*%Using alternate method*

*x=input('Input x');*

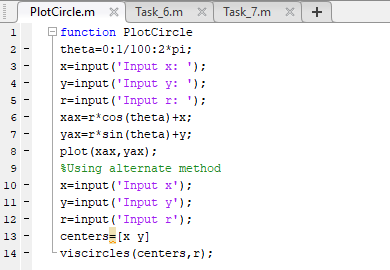
*y=input('Input y');*

*r=input('Input r');*

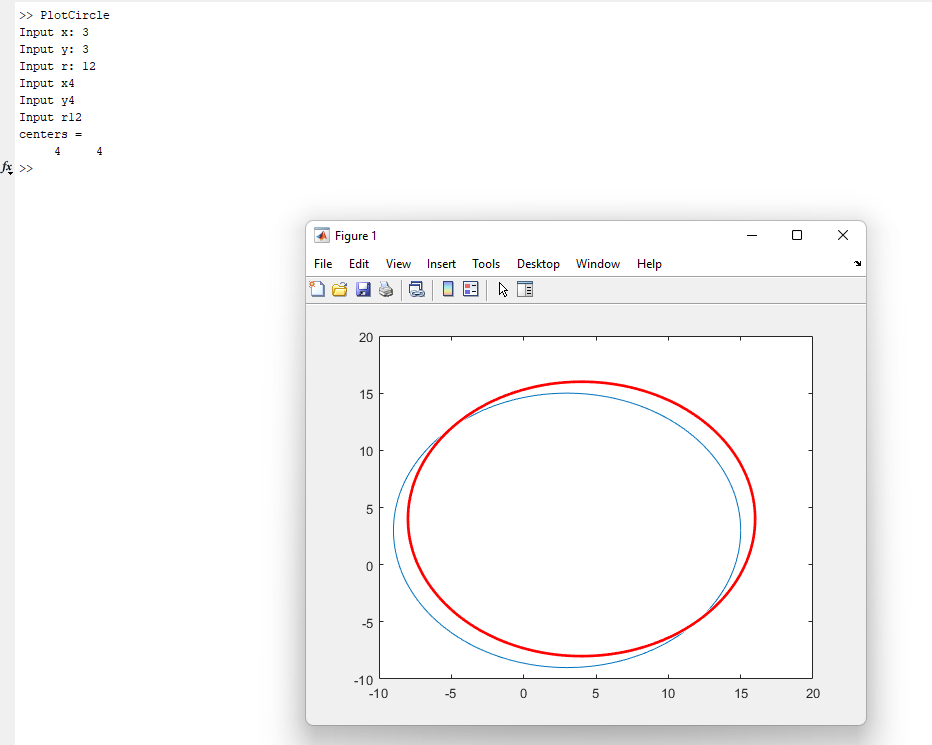
*centers=[x y]*

*viscircles(centers,r);*

**Code SS:**

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

MATLAB can be used to plot figures

## 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.

**Problem Analysis:**

We use MATLAB to add signals.

**Algorithm:**

* Write the two signals
* Add the signals using a for loop
* Plot the signals

**Code:**

*function AmpScale*

*Si=input('Enter a signal:');*

*th=input('Enter threshold:');*

*neg\_th=-th;*

*count=0;*

*for i=1:length(Si)*

*if(Si(i)>th&&Si(i)<neg\_th)*

*count=count+1;*

*end*

*end*

*disp('The number of samples greater than thredshold or less than threshold:')*

*disp(count);*

*figure(1)*

*plot(Si,'-m')*

*title('Plot 1')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*figure(2)*

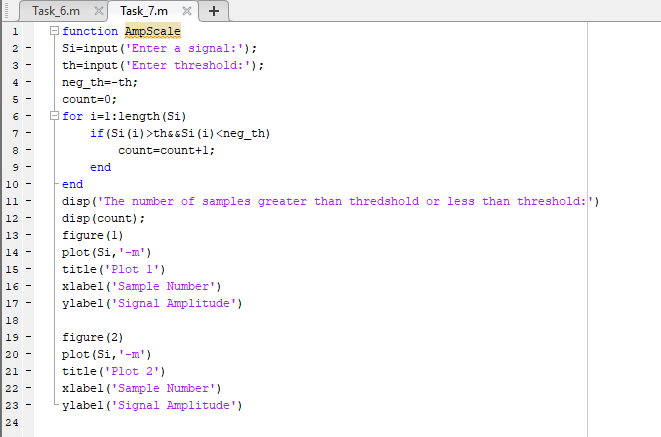
*plot(Si,'-m')*

*title('Plot 2')*

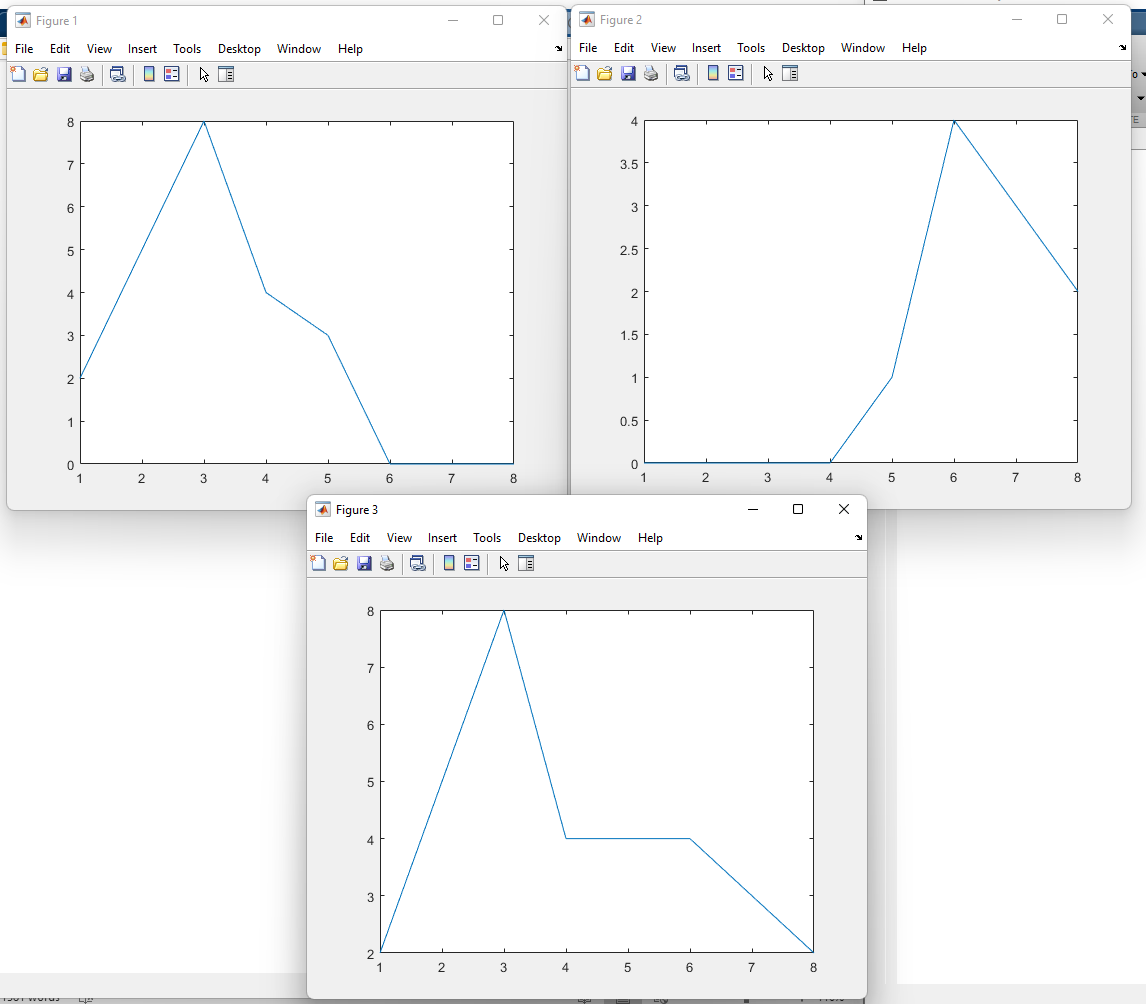
*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

**Code SS:**

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

## 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.

**Problem Analysis:**

We often need to measure the threshold samples of a signal. So we perform that using MATLAB.

**Algorithm:**

* Enter the signal
* Enter threshold
* Pass the signal to the AmpScale Function
* Plot

**Code:**

*function AmpScale*

*Si=input('Enter a signal:');*

*th=input('Enter threshold:');*

*neg\_th=-th;*

*count=0;*

*for i=1:length(Si)*

*if(Si(i)>th&&Si(i)<neg\_th)*

*count=count+1;*

*end*

*end*

*disp('The number of samples greater than thredshold or less than threshold:')*

*disp(count);*

*figure(1)*

*plot(Si,'-m')*

*title('Plot 1')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*figure(2)*

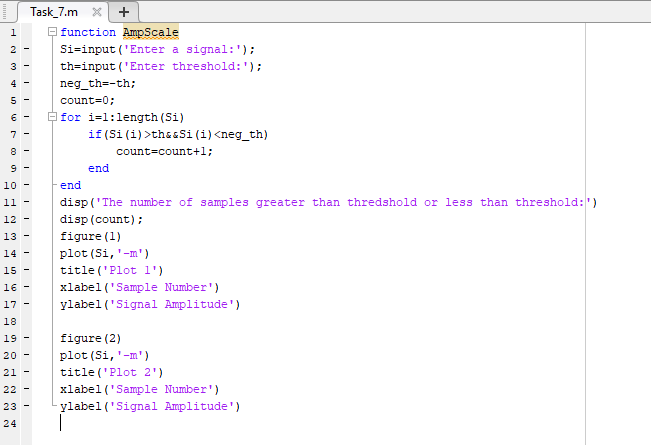
*plot(Si,'-m')*

*title('Plot 2')*

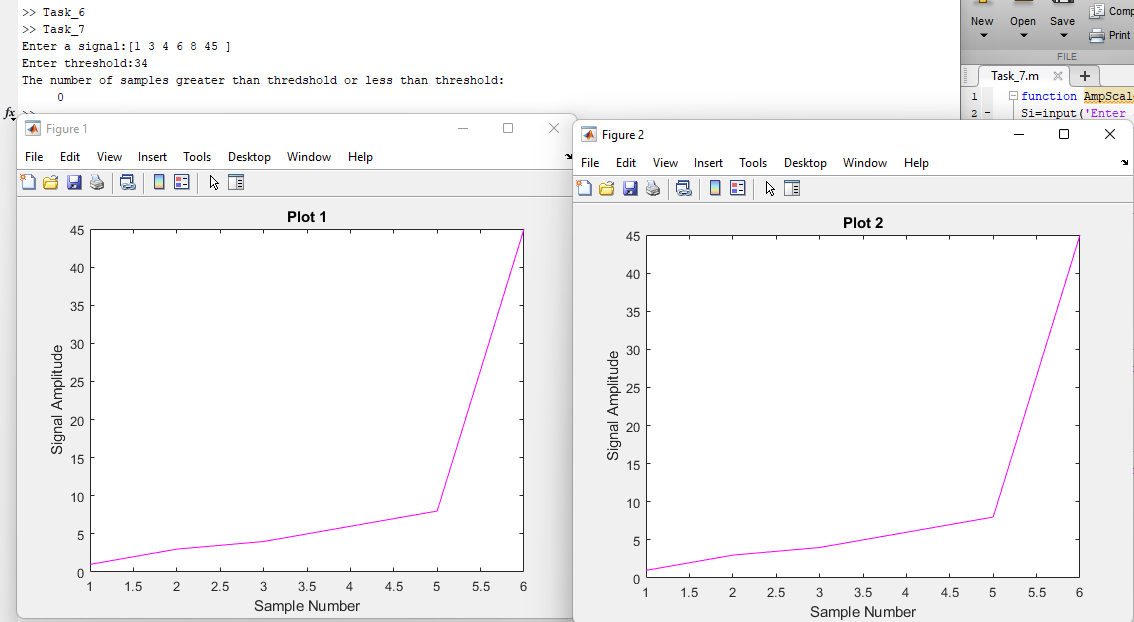
*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

**Code SS:**



**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

MATLAB compares the threshold with a function

## 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.

a) Call this function from a matlab file. Verify your result by using the built-in command

“downsample”. Plot the original signal, downsampled signal determined by your

program, and downsampled signal obtained by the command downsample.

b) Modify your function to work as generic down sampling function that takes both the

input signal and the sampling factor from the user. Your function must also check the

possibility of down sampling by comparing the sampling rate with the number of

samples in the input signal.

**Problem Analysis:**

Downsampling is an important operation on signals. We see how we can perform that in MATLAB.

**Algorithm:**

* Enter signals
* Pass the signals to the user made downsample function
* Plot resultant signal

**Code:**

*%main*

*n=input('Input n: ');*

*x=input('Input x: ');*

*Y=step(n,x);*

*figure(1)*

*stem(n,x);*

*ds=DownsampleFunc(Y);*

*for i=1:length(x)*

*x(i)=0;*

*end*

*Y1=step(n,x);*

*ds=ds+Y;*

*figure(2)*

*stem(n,ds);*

*%DownSample Function*

*function ds=DownsampleFunc(s)*

*j=1;*

*for i=1:length(s)*

*if(mod(i,2)~=0)*

*ds(j)=s(i);*

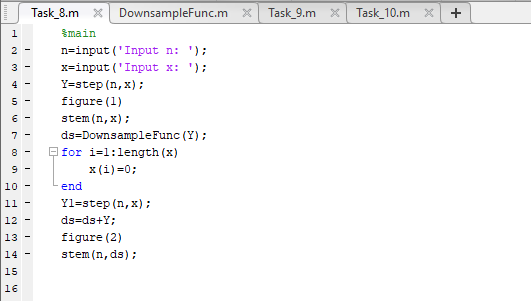
*j=j+1;*

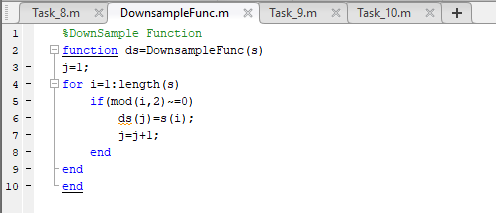
*end*

*end*

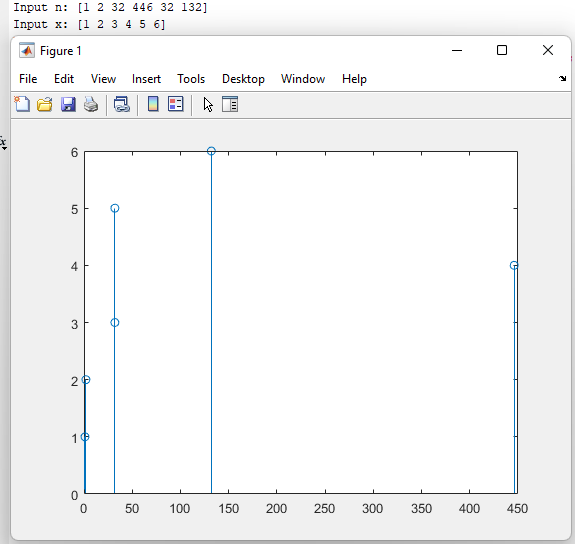
*end*

**Code SS:**

****

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

MATLAB can be used to downsample a function

## 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 so on.

a) Call this function from a matlab file. Verify your result by using the built-in command

“upsample”. Plot the original signal, upsampled signal determined by your function

upsamp, and upsampled signal obtained by the command upsample.

b) Modify your function to work as generic up sampling function that takes both the input

signal and the sampling factor from the user.

c) Your function must also perform other up sampling methods such as instead of 0, the

new sample is the copy of preceding or succeeding sample of the original signal or the

new sample is the average of both. Check for possibility new up sampling methods by

comparing the samples in the input signal.

**Problem Analysis:**

Just life downsampling, we perform upsampling also

**Algorithm:**

* Enter the signals
* Pass them to the user made UpSampling Function
* Plot the resultant Signal

**Code:**

*S=input('Enter a signal:');*

*us=upsample(S);*

*disp(us);*

*% plotting the original signal*

*subplot(2,1,1)*

*stem(S)*

*title('Original Signal')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

*% plotting the upsampled signal*

*subplot(2,1,2)*

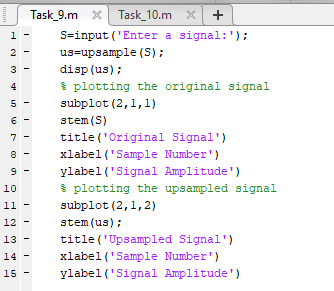
*stem(us);*

*title('Upsampled Signal')*

*xlabel('Sample Number')*

*ylabel('Signal Amplitude')*

**Code SS:**

****

**Output / Graphs / Plots / Results:**

**Discussion and Conclusion:**

MATLAB can be used to Upsample a signal.

## 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

**Problem Analysis:**

We will see how to plot 3D graphs in MATLAB.

**Algorithm:**

* Write the code
* Execute code
* Note Observations

**Code:**

*clear all*

*close all*

*to all rows of x size(-2:.2:2) times and vice versa*

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

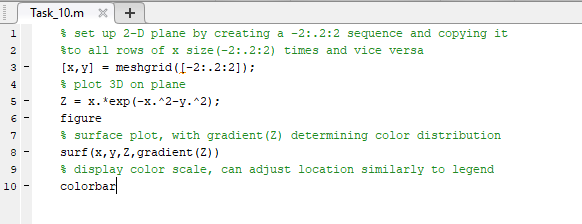
*Z = x.\*exp(-x.^2-y.^2);*

*figure*

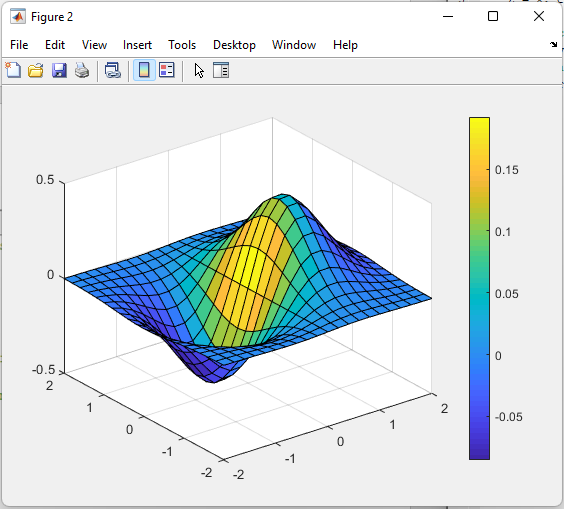
*surf(x,y,Z,gradient(Z))*

*colorbar*

**Code SS:**

****

**Output / Graphs / Plots / Results:**

****

**Discussion and Conclusion:**

MATLAB can be used for 3D plotting.