Signals & Systems Laboratory

CSE- 301L

Lab # 04

OBJECTIVES OF THE LAB

This lab will help you grasp the following concepts:

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

4.1 DISCRETE-TIME SIGNAL REPRESENTATION IN MATLAB

In MATLAB, finite-duration sequence (or discrete time signal) is represented by row matrix/vector of appropriate values. Such representation does not have any information about sample position n. Therefore, for correct representation, two vectors are required, one for x and other for n. Consider the following finite duration sequence & its implementation:

$$x(n) = \{ 1\ -1\ 0\ \underset{\uparrow}{2}\ 1\ 4\ 6 \}$$

```
>> n = [-3:1:3]

n =

    -3  -2  -1    0    1    2   3

>> x = [1 -1 0 2 1 4 6]

x =

     1    -1    0    2  1   4   6
```

NOTE # 01: When the sequence begins at n=0, x-vector representation alone is enough.

NOTE # 02: An arbitrary infinite-sequence can’t be represented in MATLAB due to limited memory.

-------------------------TASK 01-------------------------

Given the signals:

$x_1[n]$ = [2 5 8 4 3]

$x_2[n]$ = [4 3 2]

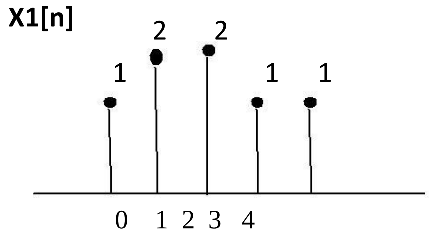
a) Write a Matlab program that adds these two signals. Use vector addition and multiplication.

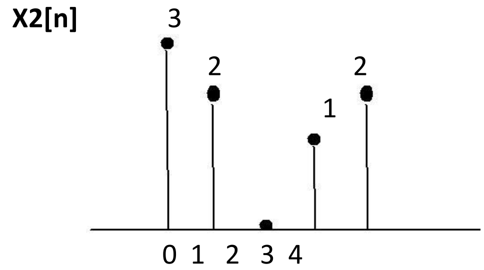
b) Instead of using vector addition and multiplication, use for loop to add and multiply the signals.

-------------------------TASK 02-------------------------

Amplitude scaling by a factor β causes each sample to get multiplied by β. Write a user-defined function 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.

-----------------------TASK 03------------------------





Write a Matlab program to compare the signals $x_1[n]$ and $x_2[n]$. Determine the index where a sample of $x_1[n]$ has smaller amplitude as compared to the corresponding sample of $x_2[n]$. Use for loop.

4.2 GRAPHICS

Two- and three-dimensional MATLAB graphs can be given titles, have their axes labeled, and have text placed within the graph. The basic functions are:

```
Function Description
============================================================================
plot(x,y)                plots y vs x
plot(x,y1,x,y2,x,y3)     plots y1, y2 and y3 vs x on the same graph
stem(x)                  plots x and draws a vertical line at each
                         datapoint to the horizontal axis
xlabel('x axis label')   labels x axis
ylabel('y axis label')   labels y axis
title ('title of plot')  puts a title on the plot
gtext('text')            activates the use of the mouse to position a
                         crosshair on the graph, at which point the
                         'text' will be placed when any key is pressed.
zoom                     allows zoom IN/OUT using the mouse cursor
grid                     draws a grid on the graph area
print filename.ps        saves the plot as a black and white postscript
                         file
Shg                      brings the current figure window forward.
CLF                      clears current figure.
============================================================================
```

4.2.1 Two-dimensional plots

The plot command creates linear x-y plots; if x and y are vectors of the same length, the command plot(x,y) opens a graphics window and draws an x-y plot of the elements of x versus the elements of y.

Example: Let's draw the graph of the sine function over the interval -4 to 4 with the following commands:

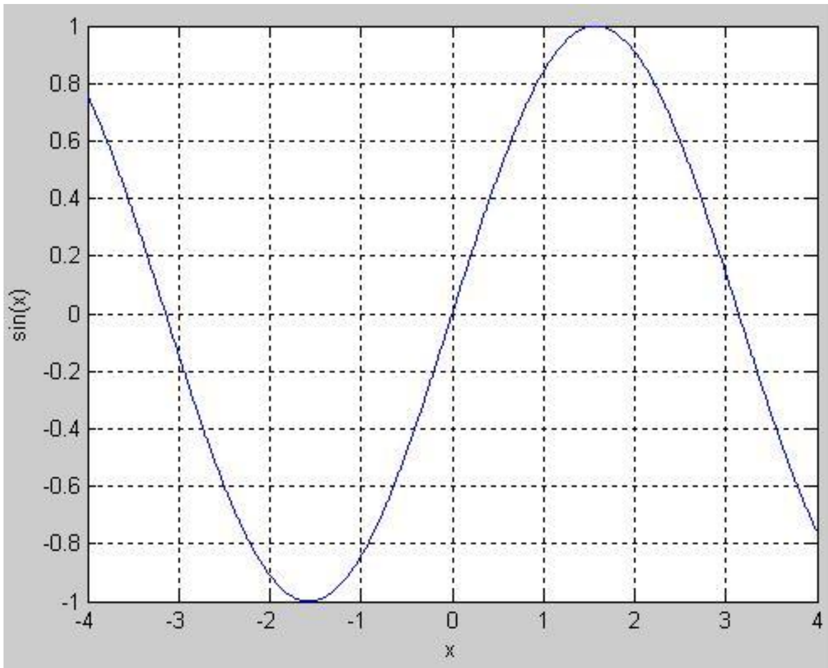
>> x = -4:.01:4; y = sin(x); plot(x,y)

>> grid

>> xlabel('x')

>> ylabel('sin(x)')

The vector x is a partition of the domain with meshsize 0.01 while y is a vector giving the values of sine at the nodes of this partition (recall that sin operates entrywise). Following figure shows the result.



4.2.1.1 MULTIPLE PLOTS ON SAME FIGURE WINDOW

Two ways to make multiple plots on a single graph are:

i. **Single plot command**

```
x = 0:.01:2*pi;
y1=sin(x);
y2=sin(2*x);
y3 = sin(4*x);
plot(x,y1,x,y2,x,y3)
```

ii. **Multiple plot commands**

Another way is with **hold**. The command hold freezes the current graphics screen so that subsequent plots are superimposed on it. Entering hold again releases the ``hold.''

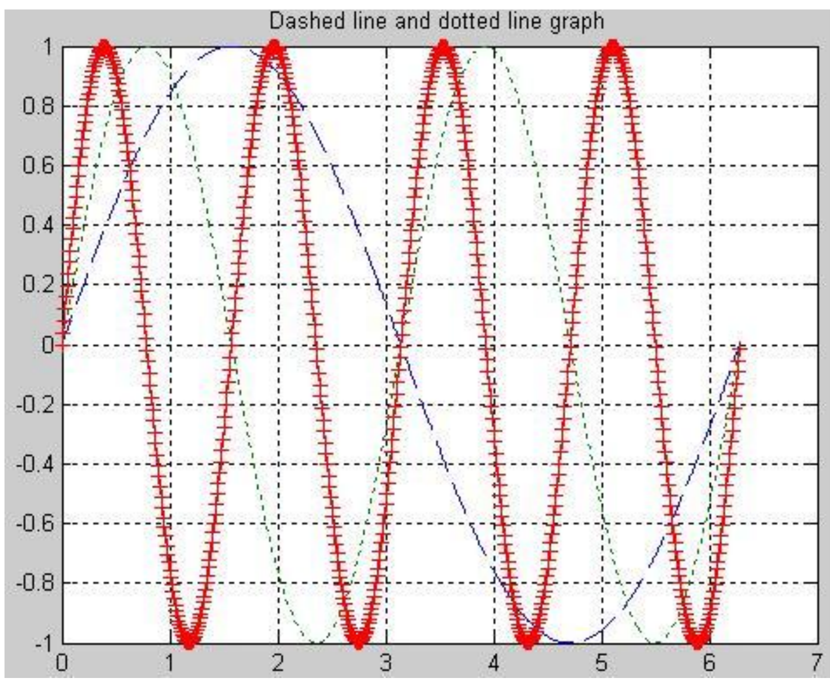
```
x = 0:.01:2*pi;
```

```
y1=sin(x);
y2=sin(2*x);
y3 = sin(4*x);
plot(x,y1);
hold on;
plot(x, y2);
plot(x,y3);
```

4.2.1.2 OVERRIDING THE DEFAULT PLOT SETTINGS

One can override the default linetypes and pointtypes. For example, the command sequence

```
x = 0:.01:2*pi;
y1=sin(x);
y2=sin(2*x);
y3=sin(4*x);
plot(x,y1,'--',x,y2,':',x,y3,'+')
grid
title ('Dashed line and dotted line graph')
```



The line-type and mark-type are

```
=============================================================
Linetypes : solid (-), dashed (-). dotted (:), dashdot (-
.) Marktypes : point (.), plus (+), star (*, circle (o),
x-mark (x)
=============================================================
```

-------------------------TASK 04-------------------------

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

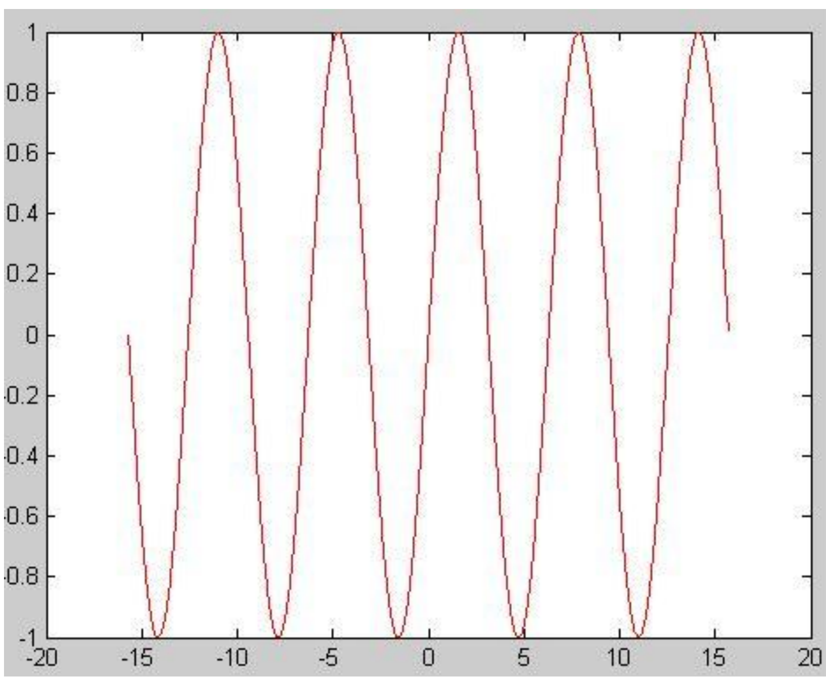
4.2.1.3 AXES COMMANDS (MANUAL ZOOMING)

MATLAB automatically adjusts the scale on a graph to accommodate the coordinates of the points being plotted. The axis scaling can be manually enforced by using the command **axis([xmin xmax ymin ymax])**. A signal can be zoomed out by specifying the axis coordinates by user himself.

Example:

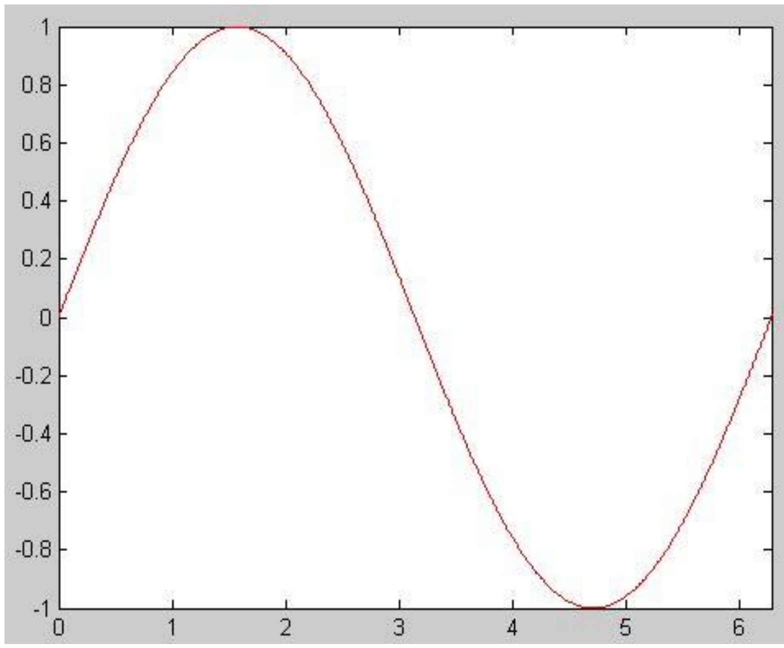
```
x = -5*pi:.01:5*pi;
y1=sin(x);
plot(x,y1, 'r')
```

The plot is shown in the figure below.



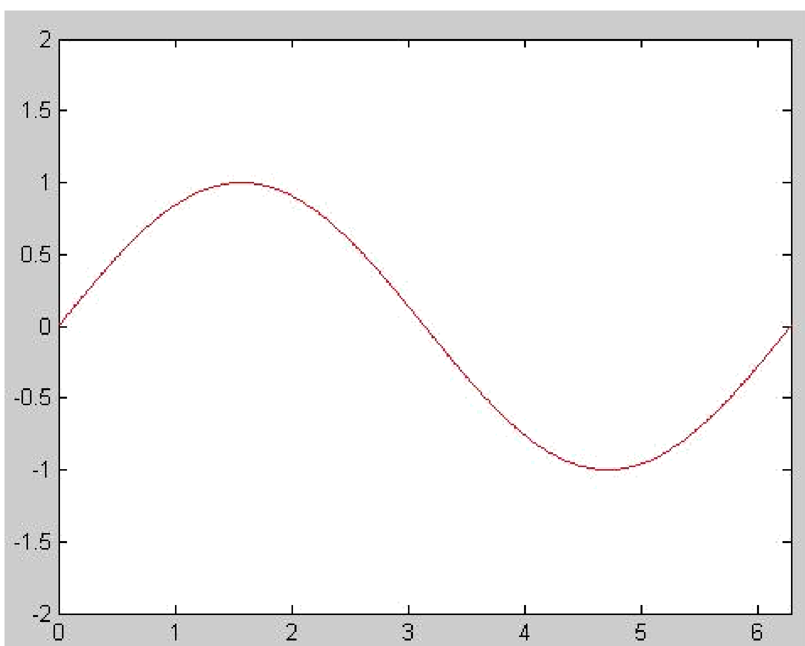
In order to see only one cycle of this signal from 0 to 2π, the signal is zoomed using axis command. Here we have specified xmin and xmax as 0 and 2π respectively.

The magnified plot is shown in the figure below.



Similarly the y-axis can be adjusted according to requirements.

```
x = -5*pi:0.01:5*pi;
y1=sin(x);
plot(x,y1, 'r')
axis([0 2*pi -2 2])
```



4.2.1.4 LABELING A GRAPH

To add labels to your graph, the functions xlabel, ylabel, and title can be used as follows:

```
xlabel('x-axis')
ylabel('y-axis')
title('points in a plane')
```

4.2.1.5 SUBPLOT

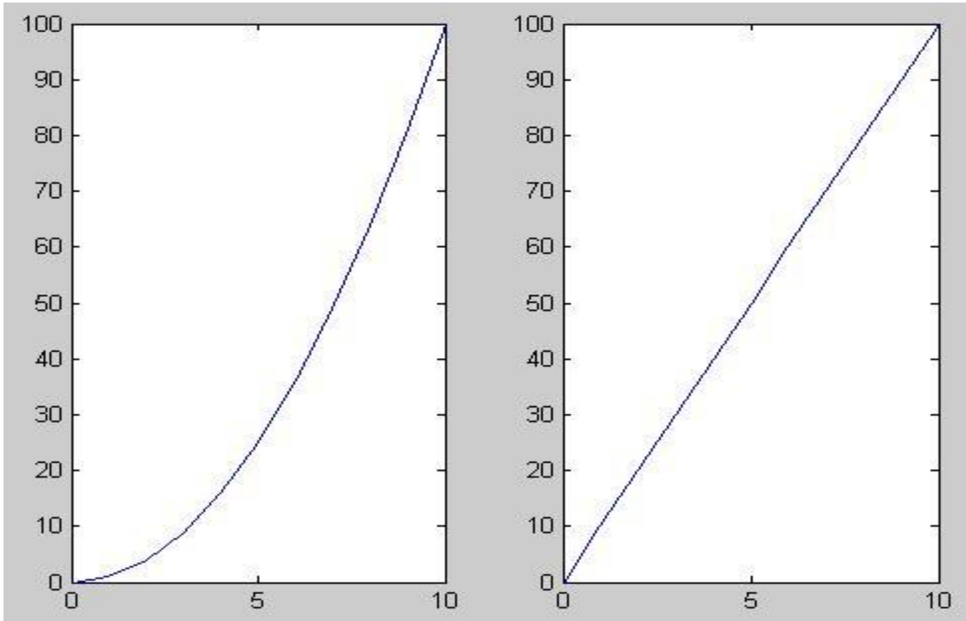
SUBPLOT Create axes in tiled positions.

MATLAB graphics windows will contain one plot by default. The command subplot can be used to partition the screen so that up to four plots can be viewed simultaneously. A single figure can be divided into a number of plotting areas where different graphs can be plotted. This can be accomplished by using the command subplot(m, n, p) where m, n specifies the total number of rows and columns respectively in the figure window and p specifies the specific cell to plot into.

```
x=0:1:10;
y=x.^2;
z=10*x;
```

Now type the following code

```
figure
subplot (1,2,1)
plot(x,y)
subplot (1,2,2)
plot(x,z)
```



In the above case subplot(m,n,p) command was used, in our case subplot (1,2,1) and subplot (1,2,2). Here m=1 means that divide the figure into 1 row, n=2 means to divide the figure into 2 columns. This gives us a total of 2 subplots in one figure. Where p=1 means the window on the left (starting from row 1 and counting p=1 subplots to the right) and p=2 means the subplot on the right (starting from row 1 and counting p=2 subplots to the right).

Example: Performing operations on signals entered by user

```
clear
x=input('Enter the first discrete time signal\n');
len_x=length(x);
y=input('Enter the second discrete time signal\n');
len_y=length(y);

while(len_y~=len_x)
    disp('Error: Length of signals must match. Enter the 2nd
    signal again')
    y=input('');
    len_y=length(y);
end
```

```
z=x+y;
subplot(3,1,1)
stem(x,'filled')
title('Signal 1')
xlabel('Sample number')
ylabel('Signal Amplitude')

subplot(3,1,2)
stem(y,'filled')
title('Signal 2')
xlabel('Sample number')
ylabel('Signal Amplitude')

subplot(3,1,3)
stem(z,'filled')
title('Resultant Signal')
xlabel('Sample number')
ylabel('Signal Amplitude')
```

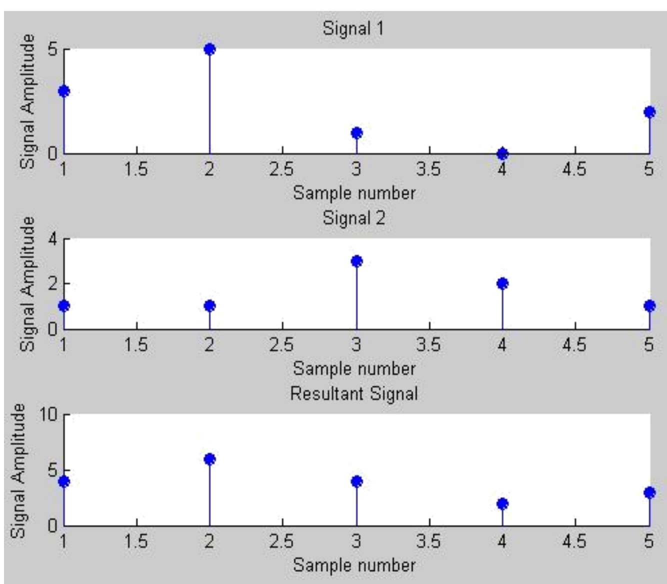
output:

Enter the first discrete time signal

[3 5 1 0 2]

Enter the second discrete time signal

[1 1 3 2



-------------------------TASK 05-------------------------

Make two separate functions for signal addition and multiplication. The functions should take the signals as input arguments and return the resultant signal. In the main program, get the signals from user, call the functions for signal addition and multiplication, and plot the original signals as well as the resultant signals.

-------------------------TASK 06-------------------------

Given the signals:

$$X1[n] = 2\delta[n] + 5\delta[n-1] + 8\delta[n-2] + 4\delta[n-3] + 3\delta[n-4]$$
$$X2[n] = \delta[n-4] + 4\delta[n-5] + 3\delta[n-6] + 2\delta[n-7]$$

Write a Matlab program that adds these two signals. Plot the original signals as well as the final result.

-------------------------TASK 07-------------------------

Take a discrete-time signal from user. Count the number of samples with amplitude greater than a threshold of 3 and less than a threshold of -3 (use for loop).

-------------------------TASK 08-------------------------

Write your own function to downsample a signal i.e. retain odd numbered samples of the original signal and discard the even-numbered (downsampling by 2). The function must take a signal as input and return the downsampeled version of that signal. See Fig for example. Call this function from a matlab file. Verify your result by using the command “**downsample**”. Plot the original signal, downsampled signal determined by your program, and downsampled signal obtained by the command **downsample**.

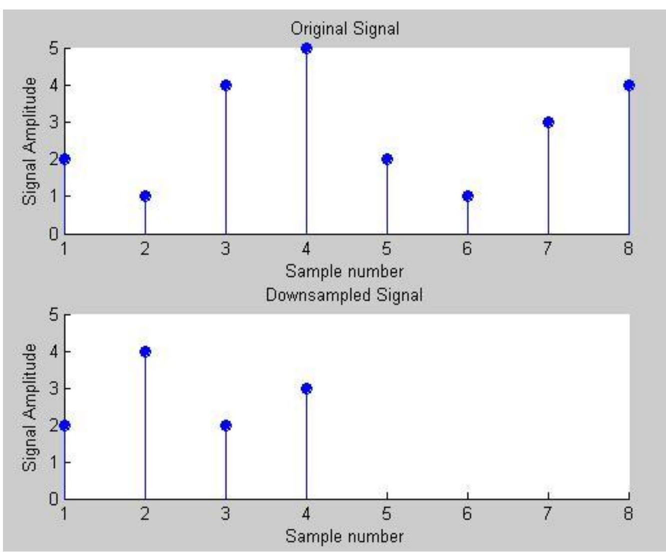


Fig. DownSampling

-------------------------TASK 09-------------------------

Write your own function to upsample 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. See Fig for example. Call this function from a matlab file. Verify your result by using the command "**upsample**". Plot the original signal, upsampled signal determined by your program, and upsampled signal obtained by the command **upsample**.

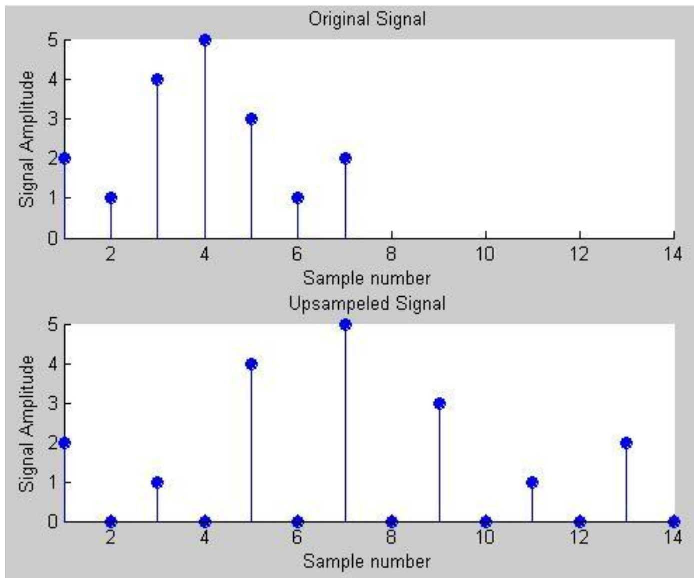


Fig. Upsampling
