4. SS Lab 4

The fourth lab of Signals and Systems would focus on representing signals in MATLAB, plotting signals in elucidatory manner, and functions related to plotting signals in the MATLAB environment.

Suggestions for improvement or correction of the manuscript would be appreciated.

4.1 Lab Objectives

In this lab, the following topics would be covered:

- Discrete signal representation in MATLAB
- MATLAB graphics
- Two dimensional plots
- Plot and subplot
- Different plotting functions used in MATLAB

4.2 Discrete-Time Signal Representation in MATLAB

In MATLAB, a finite-duration sequence (or discrete time signal) `x` 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 and its implementation:

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

The proper representation in MATLAB would be as follows:

```
>> 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
```

Please note that:

1. If the sequence begins at `n=0`, `x`-vector representation alone is enough.
2. An arbitrary infinite-sequence cannot be represented in MATLAB due to limited memory.

4.3 Graphics in MATLAB

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 as follows:

```
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.
================================================================================
```

4.3.1 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 us 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) as in Figure 4.1:

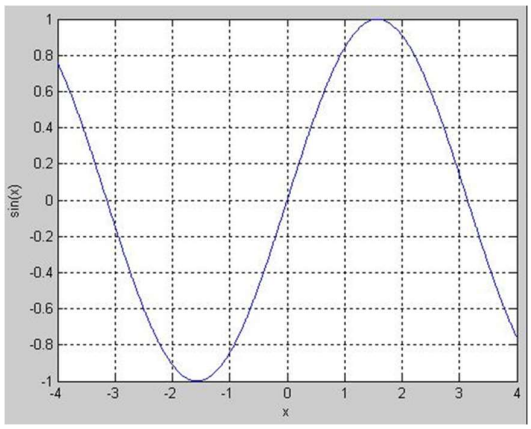

Figure 4.1: Sine plot

4.3.2 Multiple Plots on the Same Figure Window

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

i. **Single plot command**: The first method for multiple plots in the same figure can be executed as follows:

```
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 command**: Another way for multiple plots in the same figure is using the `hold` command. The `hold` command freezes the current graphics screen so that subsequent plots are superimposed on it. Entering `hold` again releases the "hold".

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

4.3.3 Overriding the Default Plot Settings

It is possible to override the default line-types and mark-types. For example, check 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')
```

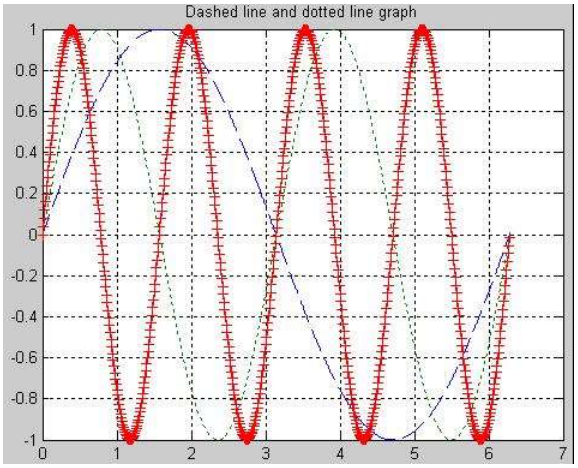

Figure 4.2: Overriding plot setting

The line-types and mark-types that one can use include:

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

4.3.4 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 the user.

Example:

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

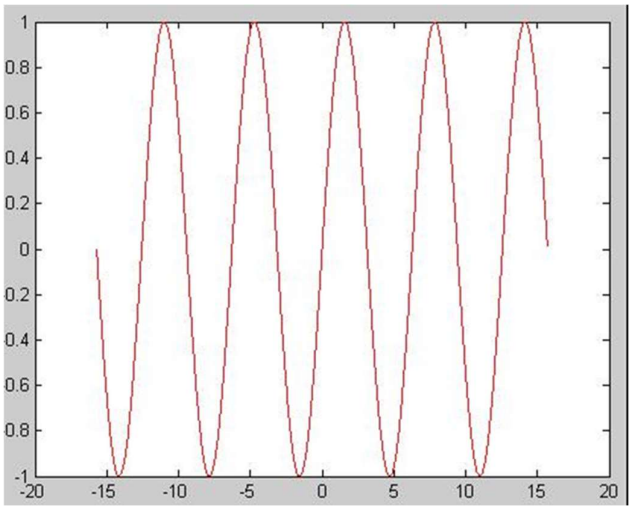

Figure 4.3: Sine plot colored red

If it is intended to see only one cycle of this signal from 0 to 2π, the signal can be zoomed in using the `axis` command. Here, the minimum and the maximum values for x-axis can be specified as 0 and 2π respectively.

```
x = -5*pi:0.01:5*pi;
y1=sin(x);
plot(x,y1, 'r')
axis([0 2*pi -1 1])
```

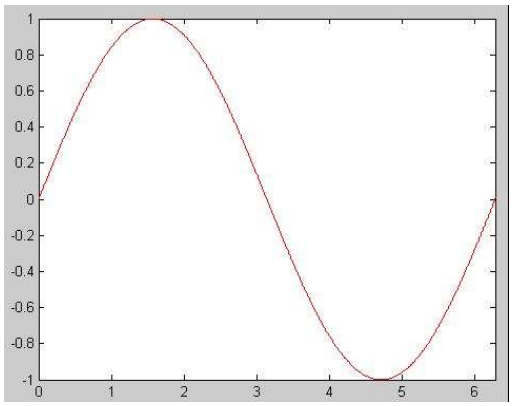

Figure 4.4: Sine plot colored red from 0 to 2π

Similarly, the y-axis can also 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])
```

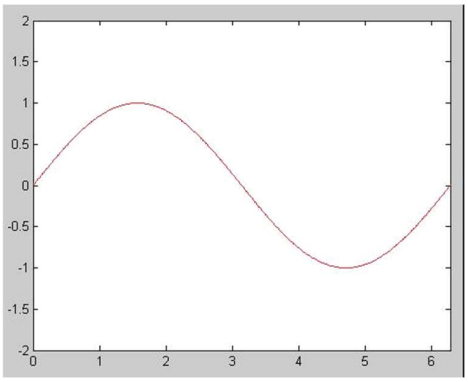

Figure 4.5: Sine plot colored red from 0 to 2π and scaled amplitude

4.3.5 Labelling 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')
```

4.3.6 Subplot

Subplot is used to create axes in tiled positions. The MATLAB graphics windows contains 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` specify the total number of rows and columns respectively in the figure window and `p` specifies the specific cell to plot.

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

Check the following code:

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

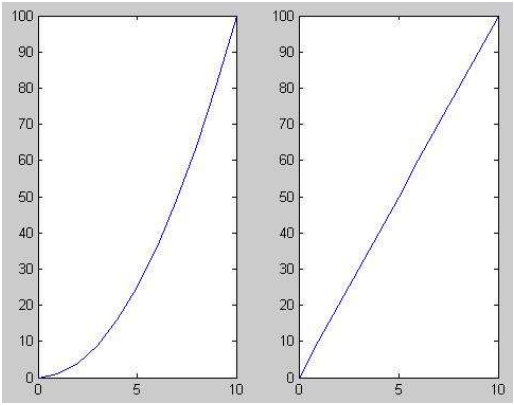

Figure 4.6: Subplot example

In this case, `subplot(m,n,p)` command was used, in our case `subplot(1,2,1)` and `subplot(1,2,2)`. Here `m=1` means to 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: Signal length not matching. Re-enter 2nd signal’)
   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 1]
```

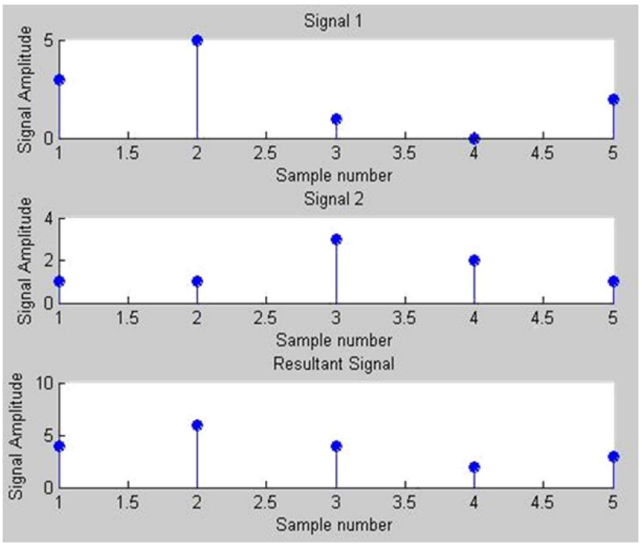

Figure 4.7: Another subplot example

4.4 Tasks

Perform the following tasks:

4.4.1 Task 01

Given the signals:

$x_1[n] = \{2, 5, 8, 4, 3\}$ and $x_2[n] = \{4, 3, 2\}$

a) Write a MATLAB program that adds and multiply these two signals. Use vector addition and multiplication respectively.
b) Instead of using vector addition and multiplication, use `for` loop to add and multiply the corresponding elements of the two signals.

4.4.2 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.

4.4.3 Task 03

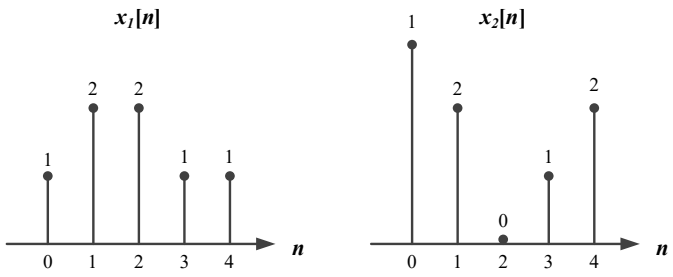

Figure 4.8: Signals x1 and x2

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.

4.4.4 Task 04

Plot the two curves $y_1 = 2x + 3$ and $y_2 = 4x + 3$ on the same graph using different plot styles.

4.4.5 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.

4.4.6 Task 06

Given the signals:

$$X_1[n] = 2\delta[n] + 5\delta[n-1] + 8\delta[n-2] + 4\delta[n-3] + 3\delta[n-4]$$
$$X_2[n] = \delta[n-4] + 4\delta[n-5] + 3\delta[n-6] + 2\delta[n-7]$$

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

4.4.7 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).

4.4.8 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 downsampled version of that signal. See Figure 4.9 for reference. 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`.

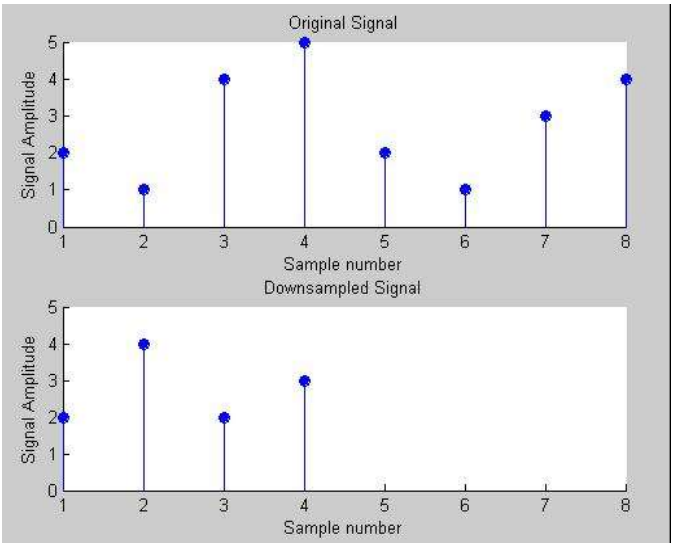

Figure 4.9: File downsampling

4.4.9 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 Figure 4.10 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`.

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.

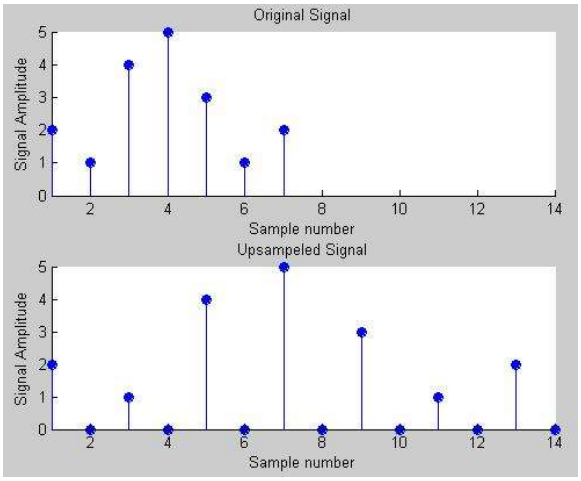

Figure 4.10: Upsampling

4.4.10 Task 10

Plotting 3-D graphics with MATLAB is highlighted in this problem. 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.

```
clear all
close all

% set up 2-D plane by creating a -2:.2:2 sequence and copying it
to all rows of x size(-2:.2:2) times and vice versa
[x,y] = meshgrid([-2:.2:2]);

% plot 3D on plane
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
```