NED University of Engineering & Technology

Department of Electrical Engineering

LAB MANUAL

For the course

Digital Signal Processing
(EE-394) For T.E.(EE)

Instructor name: Muhammad Omar

Student name: Abdul Moeed Alam

Roll no: EE-19170 **Batch:** 2019-20

Semester: Fall 2021 **Year:** T.E-2021

LAB MANUAL

For the course

Digital Signal Processing
(EE-394) For T.E.(EE)

Content Revision Team:

Mr. Muhammad Omar & Mr. Nabeel Fayyaz

Last Revision Date: 29-12-2020

Approved By

The Board of Studies of Department of Electrical Engineering

_____ _____

_____ _____

_____ _____

CONTENTS

Psychomotor Level: 4
CLO: Experimentally verify the analytical and design techniques developed for discrete time systems and signals. **[PLO (5)]**

S.No.	Date	Title of Experiment	Total Marks	Signature
1	7-12-21	To study the effects of Sampling in Discrete Time Signals.		
2	21-12-21	To study the effects of Quantization in Discrete Time Discrete Valued Signals.		
3	28-12-21	To study and verify Discrete-Time convolution and its properties.		
4	4-1-22	To study Discrete-Time correlation.		
5	4-1-22	The Discrete Fourier Transform as a Linear Transformation		
6	11-1-22	Studying Discrete Fourier Transform using an audio signal example		
7	11-1-22	To become familiar with the practical constraint in calculating the Fourier transform of any real-world signal. i.e., DFT or FFT leakage and its solution.		
8	18-1-22	Relationship between Laplace and CTFT.		
9	18-1-22	Relationship between Z transform and DTFT.		
10	25-1-22	Designing FIR filters with windowing		
11	25-1-22	Designing IIR filters using FDA tool		
12	21-12-21	**Open Ended Lab1 (due after lab 2)**		
13	11-1-22	**Open Ended Lab2(due after lab 6)**		

LAB SESSION 01

OBJECTIVE:

To study the relationship between discrete-time and continuous time signals by examining sampling and aliasing.

THEORY:

Signals are physical quantities that carry information in their patterns of variation. Continuous-time signals are continuous functions of time, while discrete-time signals are sequences of numbers. If the values of a sequence are chosen from a finite set of numbers, the sequence is known as a digital signal. Continuous-time, continuous-amplitude signals are also known as analog signals.

Analog phenomenon is continuous – like a human speech of speaker, or a continuously rotating disc attached to the shaft of motor etc. With analog phenomena, there is no clear separation between one point and the next; in fact, between any two points, an infinite number of other points exist. **Discrete phenomenon**, on the other hand, is clearly separated. There's a point (in time or space), and then there's a neighboring point, and there's nothing between the two.

Signal processing is concerned with the acquisition, representation, manipulation, transformation, and extraction of information from signals. In analog signal processing these operations are implemented using analog electronic circuits. Converting the continuous phenomena of images, sound, and motion into a discrete representation that can be handled by a computer is called analog-to-digital conversion. Digital signal processing involves the conversion of analog signals into digital, processing the obtained sequence of finite precision numbers using a digital signal processor or general-purpose computer, and, if necessary, converting the resulting sequence back into analog form. When stored in a digital computer, the numbers are held in memory locations, so they would be indexed by memory address. Regardless of the medium (either sound or an image), analog-to-digital conversion requires the same two steps: **Sampling and Quantization.**

Sampling: This operation chooses discrete (finite) points at which to measure a continuous phenomenon (which we will also call a **signal**). In the case of sound, the sample points are evenly separated in time. In the case of images, the sample points are evenly separated in space.

Sampling Rate: The number of samples taken per unit time or unit space is called the sampling rate. The frequency of sampled/discrete phenomenon (signal) can be calculated as

$$\boldsymbol{fd = F/Fs\ (cycles/sec)/(samples/sec) = cycles/samples}$$

Where, F = Frequency of analog or continuous phenomenon (signal). [Unit: cycles/sec]
Fs = Sampling frequency or sampling rate [Unit: samples/sec]
f = Frequency of Discrete phenomenon (signal). [Unit: cycles/sample]

Sampling Theorem: A continuous time phenomenon or signal like x(t) can be reconstructed exactly from its samples x(n) = x(nTs), if the samples are taken at a rate Fs = 1/Ts that is greater than twice the frequency of the signal being sampled i.e. $Fs \geq 2 * F$.

Mathematically,

For a single frequency component	For a composite signal of various frequency components
$\frac{-F_s}{2} \leq F \leq \frac{F_s}{2}$	$\frac{-F_s}{2} \leq F_{max} \leq \frac{F_s}{2}$
$\frac{-1}{2} \leq \frac{F}{F_s} \leq \frac{1}{2}$	$\frac{-1}{2} \leq \frac{F_{max}}{F_s} \leq \frac{1}{2}$
$\frac{-1}{2} \leq f_d \leq \frac{1}{2}$	$\frac{-1}{2} \leq f_{d_{max}} \leq \frac{1}{2}$

Aliasing: A common problem that arises when sampling a continuous signal is aliasing, where a sampled signal has replications of its sinusoidal components which can interfere with other components. It is an effect that causes two discrete time signals to become indistinct due to improper sampling (fd>1/2 cycles/sample).

PROCEDURE:

1. Simulate and plot two CT signals of 10 Hz and 110 Hz for 0 < t < 0.2 secs.
2. Sample at Fs = 100 Hz and plot them in discrete form.
3. Observe and note the *aliasing* effects.
4. Explore and learn.

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Open M-file editor and type the following code:

```
clear all;close all;clc;

F1 = 10;
F2 = 110;
Fs = 100;
Ts = 1/Fs;
t = [0 : 0.0005 : 0.2];
x1t = cos(2*pi*F1*t);
x2t = cos(2*pi*F2*t);

figure,
plot(t,x1t,t,x2t, 'LineWidth',2);
xlabel('cont time (sec)');
ylabel('Amp');
xlim([0 0.1]);
grid on;
legend('10Hz','110Hz');
title('Two CTCV sinusoids plotted');
```

3. Save the file as P011.m in your current directory and 'run' it, either using F5 key or writing
the file name at the command window.
(Check for the correctness of the time periods of both sinusoids.)

Now add the following bit of code at the bottom of your P011.m file and save.

```
nTs = [0 :Ts : 0.2];
n = [1 : length(nTs)-1 ];

x1n = cos(2*pi*F1*nTs);
x2n = cos(2*pi*F2*nTs);

figure,
subplot(2,1,1),
stem(nTs,x1n,'LineWidth',2);
grid on;
xlabel('discrete time (sec)');
ylabel('Amp');
xlim([0 0.1]);

subplot(2,1,2)
stem(nTs,x2n,'LineWidth',2);
grid on;
title('110Hz sampled')
xlabel('discrete time(sec)');
ylabel('Amp'); xlim([0 0.1]);
```

1. Before hitting the 'run', just try to understand what the code is doing and try to link it with what we have studied in classes regarding concepts of frequency for DT signals.
2. Now 'run' the file and observe both plots.

To see what is really happening, type the following code at the bottom of your existing P011.m file and run again.

```
figure,
plot(t,x1t,t,x2t);
hold;
stem(nTs,x1n,'r','LineWidth',2);
xlabel('time (sec)');
ylabel('Amp');
xlim([0 0.05]);

 legend('10Hz','110Hz');
```

3. Observe the plots.

Outputs:

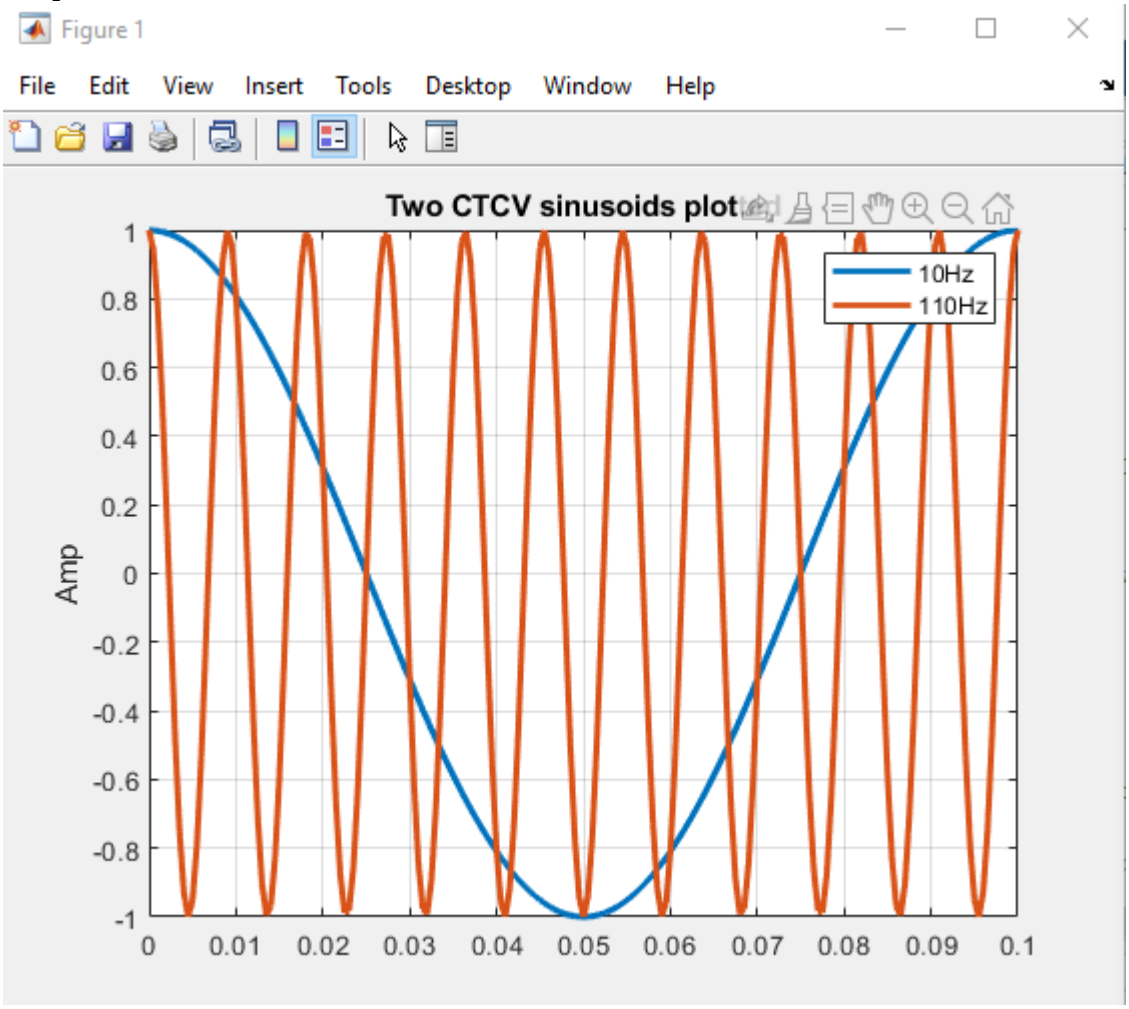


Figure 1
File Edit View Insert Tools Desktop Window Help
Two CTCV sinusoids plot
10Hz
110Hz
Amp
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1

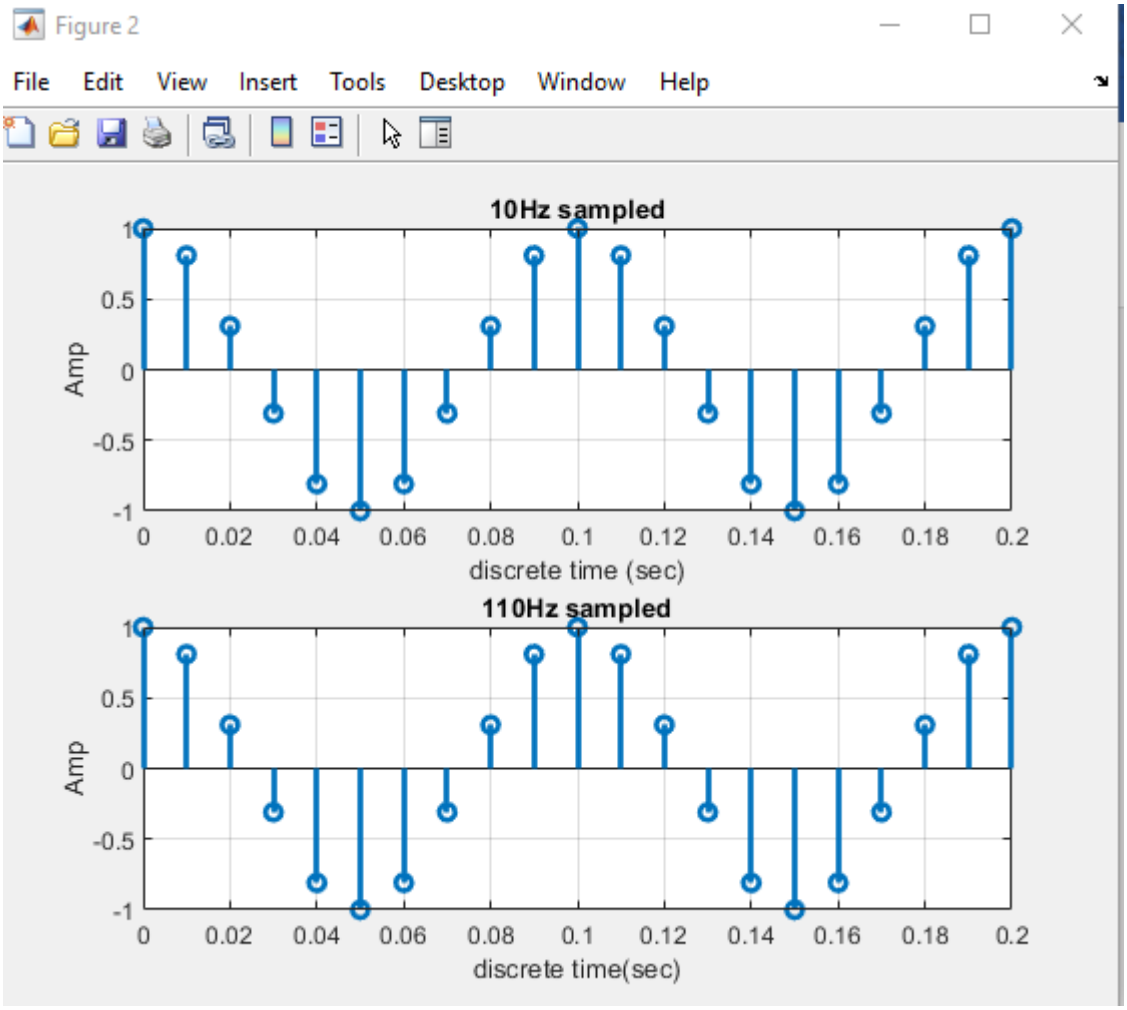
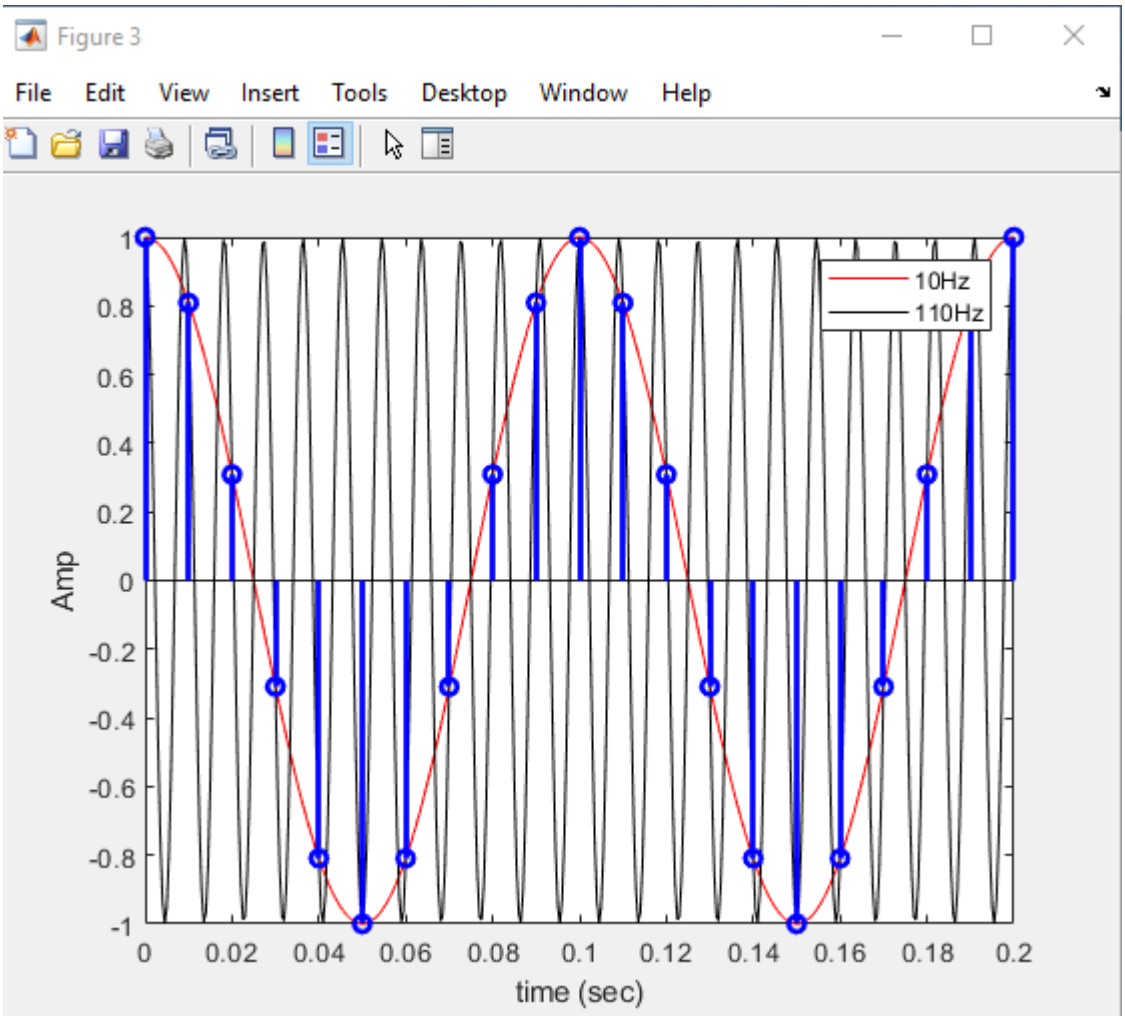


Figure 2
File Edit View Insert Tools Desktop Window Help
10Hz sampled
Amp
discrete time (sec)
110Hz sampled
Amp
discrete time(sec)



RESULT:

Aliasing occurred in 110 Hz signal because its sampling rate simply didn't follow Nyquist criteria hence giving it identity in digital domain as if it were a 10 Hz signal in continuous domain

LAB TASKS:

1. Consider the following CT signal:

x(t) = sin (2 pi F t). The sampled version will be:

x(n) = sin (2 pi F/Fs n),

where n is a set of integers and sampling interval Ts=1/Fs.

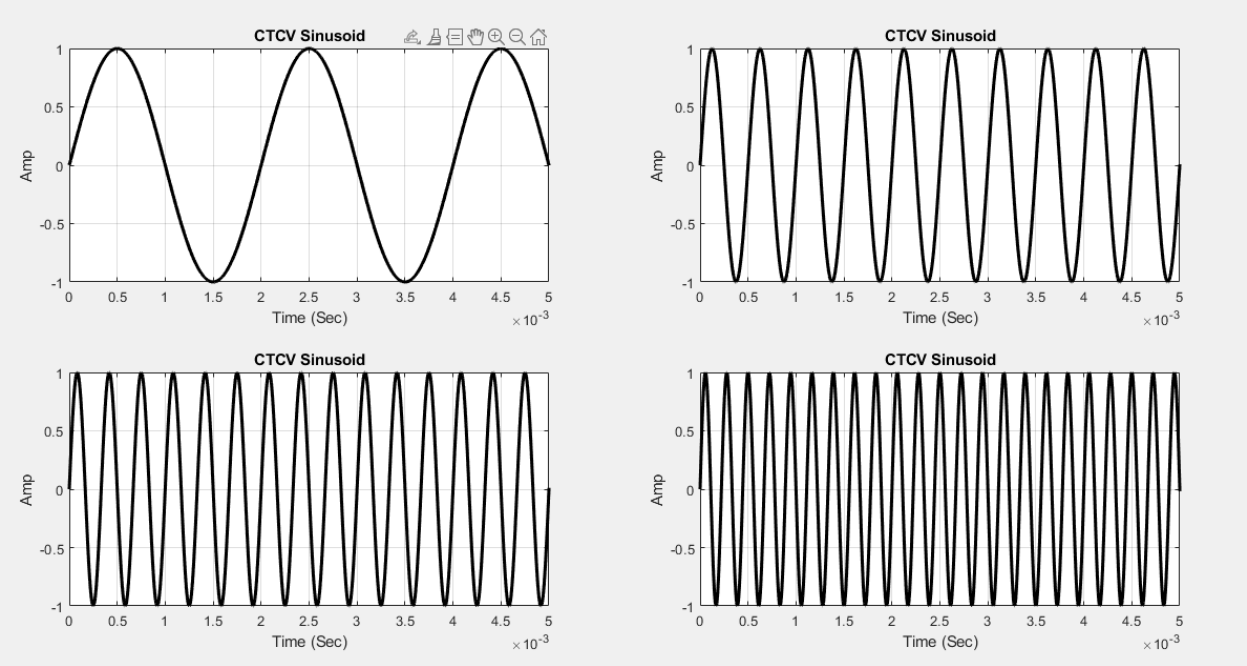
Plot the signal x(n) for n = 0 to 99

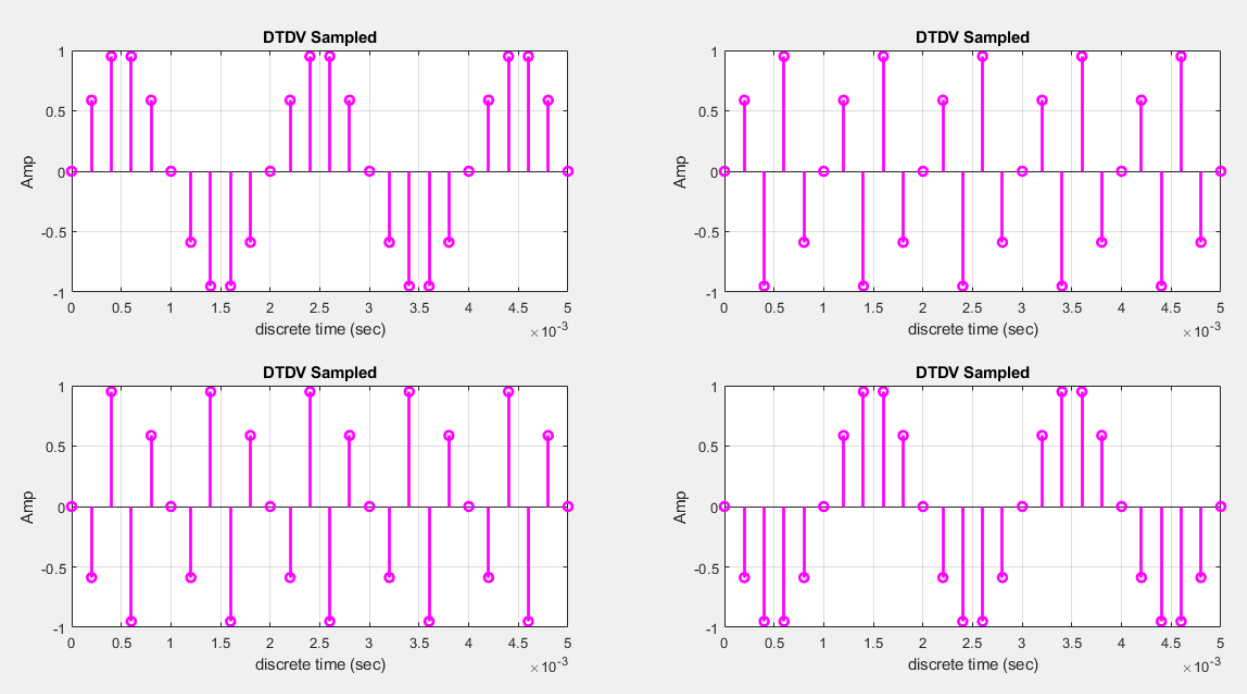
for Fs = 5 kHz and F = 0.5, 2, 3 and 4.5 kHz. Explain the similarities and differences among various plots.

Code:

```
clc,clear,close all
f=[500,2000,3000,4500];
Fs= 5000;
Ts = 1/Fs;
figure,
for i=1:length(f)
   F = f(i);
   t = [ 0 : 0.000005 : 0.02 ];
   x1t = sin(2*pi*F*t);
   hold on
   subplot (2,2,i);
   plot(t,x1t,'LineWidth',2);
   ylabel('Amp');
   xlabel('Time (Sec)')
   xlim([0 0.005]);
   grid on;
   title('CTCV Sinusoid');
end
hold off
figure,
for i=1:length(f)
   F=f(i);
   nTs = [ 0 : Ts : 0.02 ];
   n = [1 : length(nTs)-1 ];
   x1n = sin( 2*pi*F*nTs );
   hold on
   subplot ( 2,2,i );
   stem( nTs,x1n,'LineWidth',2 );
   grid on;
   title( 'DTDV Sampled' )
   xlabel( 'discrete time (sec)' );
   xlim([0 0.005]);
   ylabel( 'Amp' );
end
hold off
```

Output:


CTCV Sinusoid
Amp
Time (Sec)
×10^-3
CTCV Sinusoid
Amp
Time (Sec)
×10^-3
CTCV Sinusoid
Amp
Time (Sec)
×10^-3
CTCV Sinusoid
Amp
Time (Sec)
×10^-3


DTDV Sampled
Amp
discrete time (sec)
×10^-3
DTDV Sampled
Amp
discrete time (sec)
×10^-3
DTDV Sampled
Amp
discrete time (sec)
×10^-3
DTDV Sampled
Amp
discrete time (sec)
×10^-3

2. Generate a tone in MATLAB with varying frequency f = 1000,2000,3000,4000, 5000, 6000, 8000, 9000, 2500, -1000, -2000, -3000 Hz with Fs = 8000 samples/sec.
Listen to the tones, and observe at Sounds like what frequency?
Also Specify whether Aliasing is happening or not.

Code:

```
clear all;close all;clc;
F=1000;
Fs=8000;
Ts=1/Fs;
nTs = [ 0 :Ts : 1 ];
x1=cos(2*pi*F*nTs);
sound(x1)
```

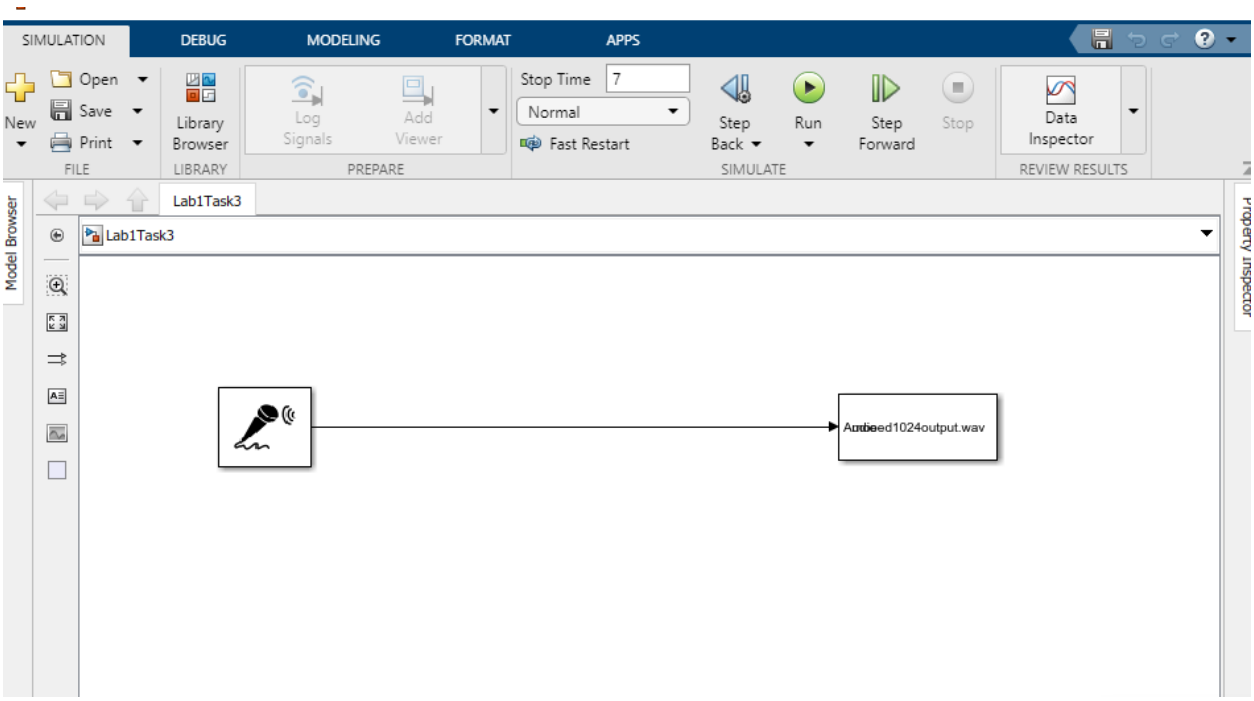
Observations:
The tones from 1 to 4 kHz were unique cos wave sounds but after that aliasing started till 9kHz. Other than that, the negative frequency sinusoids are still unique because of -Fs/2.

3. Record a sentence in your voice. (You may use Simulink /audacity to record). Change Fs =44100, 22050, 11025, 8192, 4096, 2048, 1024
and observe

a) Voice quality during playback [Excellent/Good/OK/Bad]
b) File size in kilobytes
c) Aliasing happening or not?

Observations:

a) Voice quality got decreased as sampling rate was slowly decreasing.
b) Seen in output below.
c) Aliasing didn't occur.



moeed1024output	27-Dec-21 11:25 PM	WAV File	17 KB	16kbps	00:00:08
moeed2048output	27-Dec-21 11:24 PM	WAV File	31 KB	32kbps	00:00:07
moeed4096output	27-Dec-21 11:23 PM	WAV File	59 KB	65kbps	00:00:07
moeed8192output	27-Dec-21 11:22 PM	WAV File	115 KB	131kbps	00:00:07
moeed11025output	27-Dec-21 11:21 PM	WAV File	153 KB	176kbps	00:00:07
moeed22050output	27-Dec-21 11:20 PM	WAV File	303 KB	352kbps	00:00:07
moeed44100output	27-Dec-21 11:19 PM	WAV File	605 KB	705kbps	00:00:07

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 02

OBJECTIVE:

To observe the quantization effects on sampled signals and to understand how quantization leads to quantization error. In this lab, we will investigate the influence of the number of quantization levels on the quality of digitized signal. Method of selection of ADC is also a part of this lab session.

THEORY:

Everything stored on a computer is discrete time discrete valued signal. Because computer has finite number of registers and each register is a finite length register. We take too many samples to give the 'effect' of continuous time signals. But actually, they are discrete time. We also take very fine resolution of amplitude axis to give the effect of continuous valued signal but due to finite word length of the computer register, the stored variables are already quantized. This lab aims to explain the quantization effects in a computer.

Regardless of the medium (audio or image), the digitization of real-world analog signal usually involves two stages: **sampling**, i.e., the measurement of signal at discretely spaced time intervals, and **quantization**, i.e., the transformation of the measurements (amplitudes) into finite-precision numbers (allowed discrete levels), such that they can be represented in computer memory. **Quantization** is a matter of representing the amplitude of individual samples as integers expressed in binary. The fact that integers are used, forces the samples to be measured in a finite number of bits (discrete levels). The range of the integers possible is determined by the **bit depth**, the number of bits used per sample. The **bit depth** limits the precision with which each sample can be represented.

Bit Depth:

Within digital hardware, numbers are represented by binary digits known as bits—in fact, the term bit originated from the words Binary digit. A single bit can be in only one of two possible states: either a one or a zero. When samples are taken, the amplitude at that moment in time must be converted to integers in binary representation. The number of bits used to represent each sample, called **the bit depth (bits/sample) or sample size,** determines the precision with which the sample amplitudes can be represented. Each bit in a binary number holds either a 1 or a 0. In digital sound, bit depth affects how much you have to round off the amplitude of the wave when it is sampled at various points in time

The number of different values that can be represented with b-bit is 2^b. The largest decimal number that can be represented with an b-bit binary number is $2^b - 1$. For

example, the decimal values that can be represented with an 8-bit binary number range from 0 to 255, so there are 256 different values (levels of ADC). A bit depth of 8 allows 2^8=256 different discrete levels at which samples can be approximated or recorded. Eight bits together constitute one byte. A bit depth of 16 allows 2^{16} = 65,536 discrete levels, which in turn provides much higher precision than a bit depth of 8.

The number of bits in a data word is a key consideration. The more bits used in the word, the better the resolution of the number, and the larger the maximum value that can be represented. Some computers use 64-bit words. Now, 2^{64} is approximately equal to 1.8 x 10^{19}—that's a pretty large number. So large, in fact, that if we started incrementing a 64-bit counter once per second at the beginning of the universe (≈20 billion years ago), the most significant four bits of this counter would still be all zeros today.

To simplify the explanation, take an example of ADC with a bit depth of 3, 2^3 = 8 quantization levels ranging from -4 to 3 are possible in signed magnitude representation. For bipolar ADCs (or signed magnitude representation), by convention, half of the quantization levels are below the horizontal axis (that is $\mathbf{2^1}$, of the quantization levels). One level is the horizontal axis itself (level 0), and $\mathbf{2^{b-1} - 1}$ level are above the horizontal axis. Note that since one bit is used for the signed bit (in 2-complementformat), the largest magnitude corresponds to 2^ (b -1). (Not $\mathbf{2^b}$). When a sound is sampled, each sample must be scaled to one of the 8 discrete levels. However, the samples in reality might not fall neatly onto these levels. They have to be rounded up or down by some consistent convention.

QUANTIZATION ERROR:

The samples, which are taken at evenly-spaced points in time, can take on the values only at the discrete quantization levels to store on our computer. Therefore, quantization leads to a loss in the signal quality, because it introduces a **"Quantization error"**. Quantization error is sometimes referred to as '"**Quantization noise**". **Noise** can be broadly defined as part of an audio signal that isn't supposed to be there. However, some sources would argue that a better term for quantization error is "**distortion**", defining distortion as an unwanted part of an audio signal that is related to the true signal.

The difference between the quantized samples and the original samples constitutes quantization error or rounding error (if round-off method is used)

$$\boldsymbol{Xe(n) = Xq(n) - x(n)}$$

The lower the bit depth, the more values potentially must be approximated (rounded), resulting in greater quantization error

To calculate the required bit depth of ADC i.e., bits/sample, there are two important points which we must have to consider:

a) How much noise is already present in the analog signal?
b) How much more noise can be tolerated in the digital signal? Signal-to -noise-ratio- SNR (of analog signal)

Before looking at SNR specifically in the context of digital imaging and sound, let's consider the general definition. Signal-to-noise ratio can generally be defined as the ratio of the meaningful content of a signal versus the associated background noise.

$$\boldsymbol{SNR = 10\log_{10}(Px/Pe)}$$

Where, $\mathbf{P_x}$ and $\mathbf{P_e}$ are average power of the analog signal and noise signal respectively.

A signal is any type of communication – something a person says to you, a digital signal sending an image file across a network, a message posted on an electronic bulletin board, a piece of audio being played from a cassette, etc. The noise is the part of the message that is not meaningful; in fact, it gets in the way of the message intended in the communication. You could use the term signal-to-noise ratio to describe communications in your everyday life. If you know someone who talks a lot but doesn't really convey a lot of meaning, you could say that he or she has a low signal-to-noise ratio. Web-based bulletin board and chat groups are sometimes described as having a low SNR – there may be quite a few postings, but very much meaningful content. In these first two examples, the noise consists of all the empty "filler" words. In the case of a digital signal sent across a network, the noise is the electronic degradation of the signal. On a piece of audio played from cassette, the noise could be caused by damage to the tape or mechanical imperfections in the cassette player.

In analog data communication (analog signals), the signal-to-noise ratio is defined as the ratio of the average power in the signal versus the power in the noise level. In this context, think of a signal being sent over a network connection compared to the extent to which the signal is corrupted. This is related to the general usage of the term described above. This usage of the term SNR applies to analog signals.

SIGNAL-TO-QUANTIZATION-NOISE-RATIO- SQNR (OF ADC):

Using finite word lengths prevents us from representing values with infinite precision, increases the background noise in our spectral estimation techniques etc. The amount of error implicit in a chosen bit depth can be measured in terms of the signal-to-noise ratio

(SNR).

For a digitized image or sound, the signal-to-noise ratio is defined as the ratio of the maximum sample value versus the maximum quantization error. In this usage of the term, the ratio depends on the bit depth chosen for the signal. Any signal encoded with a given bit depth will have the same ratio. This can also be called signal-quantization-noise ratio (SQNR), but you should be aware that in many sources the term signal-to-noise ratio is used with this meaning as well. (Henceforth, we'll use the term SQNR to distinguish this measurement from SNR.)

Practical A/D converters are constrained to have binary output words of finite length. Commercial A/D converters are categorized by their output word lengths, which are normally in the range from 8 to 16 bits. There is no infinite bit ADC. So, whenever we will digitize our signal, we will always have a quantization error. Quantization error represents the quality of quantization process but the total error may also turn out to be zero, so signal-quantization-noise-ratio (SQNR) is used to describe the quality of quantization process and it can be defined as

$$\boldsymbol{SQNR}_{A/D} = \mathbf{10}\log_{\mathbf{10}}(\boldsymbol{Px}/\boldsymbol{Pe})$$

Where, **Px** and $\mathbf{P_e}$ are average power of the DTCV (sampled) signal and quantization error signal respectively.

$$P_x = \frac{1}{N}\sum_{n=0}^{N-1} \left| x(n) \right|^2$$

$$P_e = \frac{1}{N}\sum_{n=0}^{N-1} \left| x_e(n) \right|^2 => P_e = \frac{1}{N}\sum_{n=0}^{N-1} \left| x_q(n) - x(n) \right|^2$$

PROCEDURE:

1. Simulate a DTCV sinusoid of 1/50 cycles/sample with length of the signal be 500.
2. Choose the no. of significant digits for round-off and apply to the signal generated above.
3. Compute the error signals and SQNR
4. Explore and observe.

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.

2. Open M-file editor and write the following code:

```
clear all;
close all;
clc;
fd1 = 1/50;
n = [0 : 499 ];


q=input('No. of Digits after decimal points to be retained (0-9): ');
x1 = cos(2*pi*fd1*n);
Px1 = sum(abs(x1).^2)/length(x1);
x1q = round(x1*10^q)/10^q;
x1e = x1 -x1q;
Pe1 = sum(abs(x1e).^2)/length(x1e);



SQNR = 10*log10(Px1/Pe1);
disp(['The Signal to Quantization Noise Ratio is: ' num2str(SQNR) ' dB.' ]);
figure,
subplot(2,1,1);
plot(n,x1,n,x1q);
xlabel('indices');
ylabel('Amp');
xlim([0 49]);
ylim([-1.1 1.1]);
legend('DTCV','DTDV');


subplot(2,1,2);
plot(n,x1e);
xlabel('indices');
ylabel('Error');
xlim([0 49]);
```

3. Save the file as **P021.m** in your current directory and run it.

Explore and take notes.
Now modify the above code as follows and save as another file P022.m.

```
clear all;
close all;
clc;
fd1 = 1/50;
n = [0 : 499 ];
q = [0 : 10];

% No. of Digits after decimal points to be retained for num = 1 : length(q)
```

```
x1 = cos(2*pi*fd1*n);
Px1 = sum(abs(x1).^2)/length(x1);
x1q = round(x1*10^q(num))/10^q(num);
x1e = x1 -x1q;
Pe1 = sum(abs(x1e).^2)/length(x1e);
SQNR(num) = 10*log10(Px1/Pe1);
end

figure,
plot(q,SQNR);
xlabel('Significant Digits');
ylabel('SQNR (dB)');
xlim([q(1) q(end)]);
```

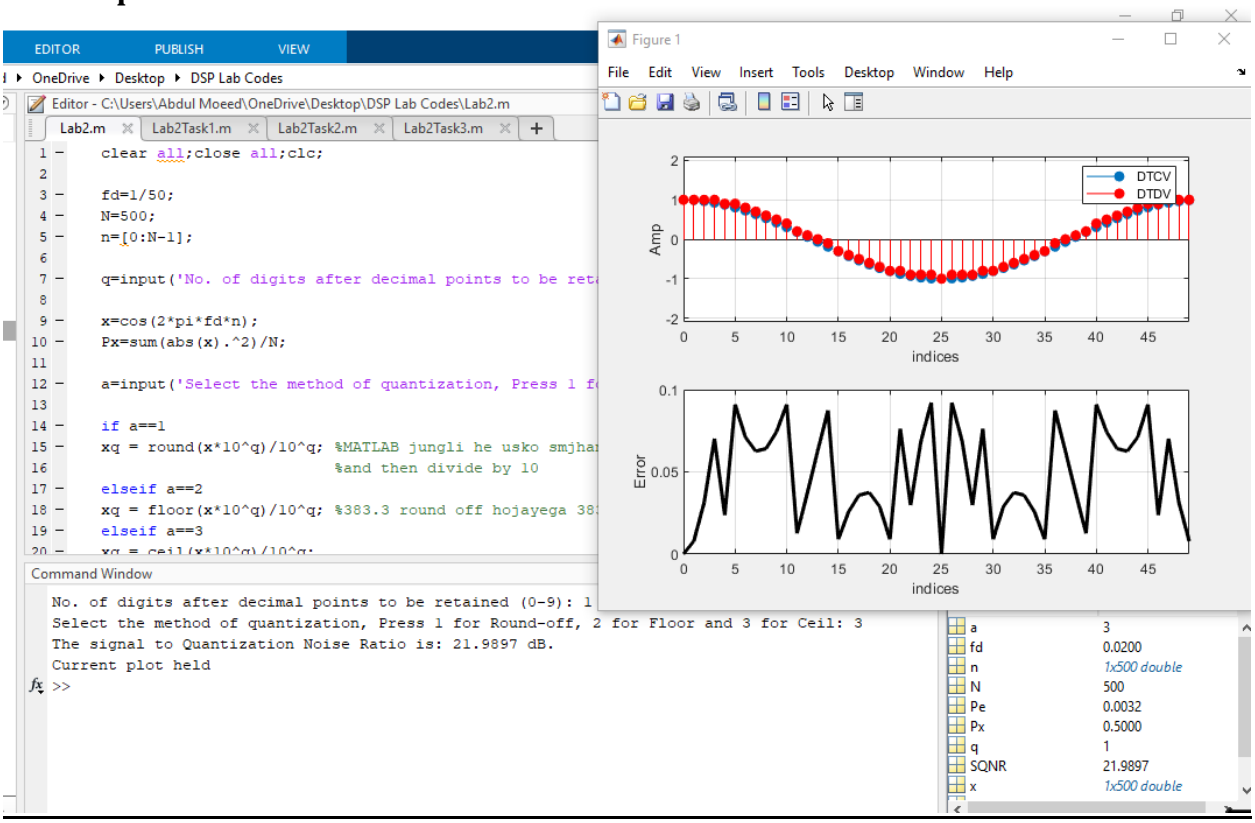
1. Before hitting the 'run', just try to understand what the code is doing and try to link it with the previous code.
2. Now 'run' the file and observe the results.

Modified Combined Code:

```
clear all;close all;clc;

fd=1/50;
N=500;
n=[0:N-1];

q=input('No. of digits after decimal points to be retained (0-9): ');

x=cos(2*pi*fd*n);
Px=sum(abs(x).^2)/N;

a=input('Select the method of quantization, Press 1 for Round-off, 2 for Floor and 3 for Ceil: ');

if a==1
xq = round(x*10^q)/10^q; %MATLAB jungli he usko smjhane k liye k itne round off krke do to
multiply by 10
                %and then divide by 10
elseif a==2
xq = floor(x*10^q)/10^q; %383.3 round off hojayega 383 to agr 383.3 chaiye to
(383.3*10)/10=383.3
elseif a==3
xq = ceil(x*10^q)/10^q;
end

xe=xq-x;
Pe=sum(abs(xe).^2)/N;
SQNR=10*log10(Px/Pe);
disp(['The signal to Quantization Noise Ratio is: ' num2str(SQNR) ' dB.' ]);

figure,
```

```
subplot(2,1,1);
stem(n,x,'filled');
hold;
stem(n,xq,'r','filled');
grid;
xlabel('indices');
ylabel('Amp');
xlim([0 49]);
ylim([-2.1 2.1]);
legend('DTCV','DTDV');

subplot(2,1,2);
plot(n,xe, 'k','Linewidth',2);
grid;
xlabel('indices');
ylabel('Error');
xlim([0 49]);
```

Output:



RESULT:

LAB TASKS:

1. Effects of Quantization with variable precision levels

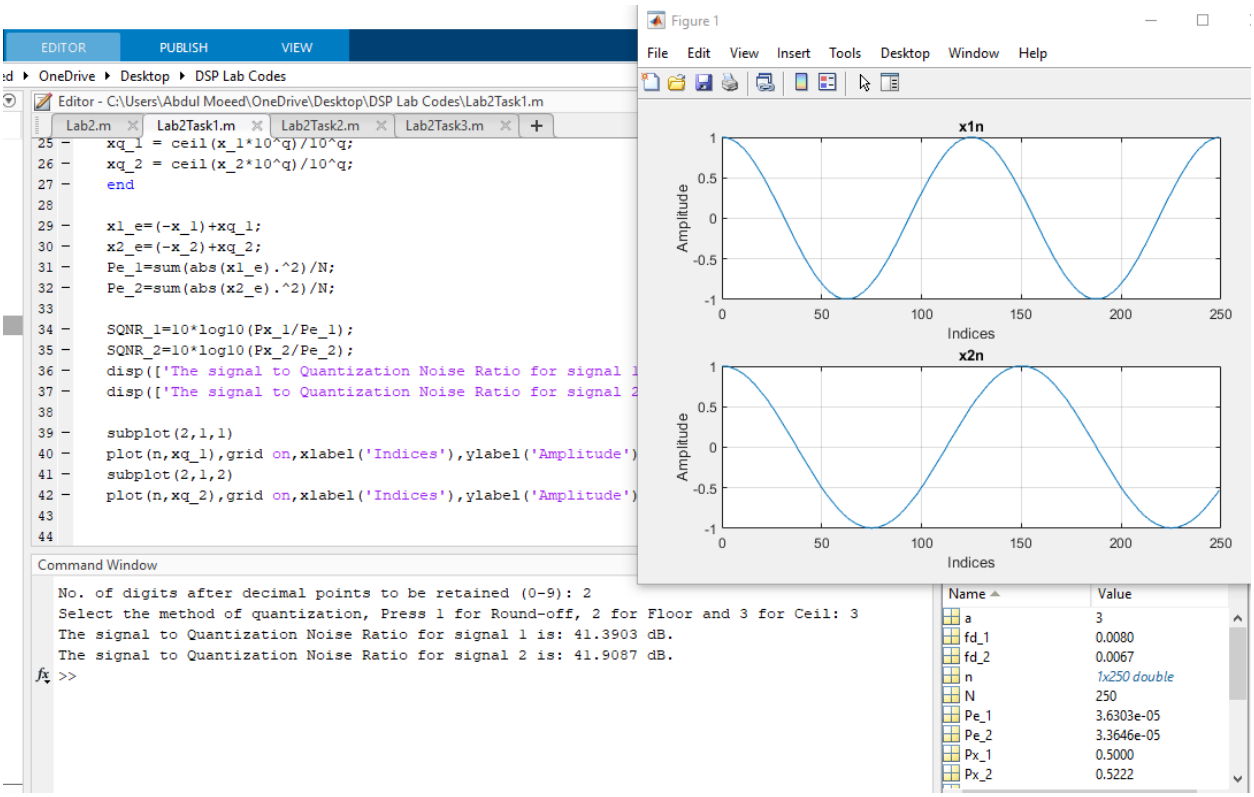
Simulate a DTCV sampled composite signal of $fd1$=125 samples/sec and $fd2$=150 samples/sec with length of the signal be 250 samples.
Take the desired number of significant digits from user as an input. Then choose the method of Quantization (round-off, floor & ceil) and apply to the signal generated above. Compute the quantization error signals and SQNR.

Code:

```
clear all;close all;clc;

fd_1=1/125;
fd_2=1/150;
N=250;
n=[0:N-1];

q=input('No. of digits after decimal points to be retained (0-9): ');

x_1=cos(2*pi*fd_1*n);
x_2=cos(2*pi*fd_2*n);

Px_1=sum(abs(x_1).^2)/N;
Px_2=sum(abs(x_2).^2)/N;

a=input('Select the method of quantization, Press 1 for Round-off, 2 for Floor and 3 for Ceil: ');

if a==1
xq_1 = round(x_1*10^q)/10^q; %MATLAB jungli he usko smjhane k liye k itne round off krke do to
multiply by 10
xq_2 = round(x_2*10^q)/10^q; %and then divide by 10
elseif a==2
xq_1 = floor(x_1*10^q)/10^q;
xq_2 = floor(x_2*10^q)/10^q;  %383.3 round off hojayega 383 to agr 383.3 chaiye to
(383.3*10)/10=383.3
elseif a==3
xq_1 = ceil(x_1*10^q)/10^q;
xq_2 = ceil(x_2*10^q)/10^q;
end

x1_e=(-x_1)+xq_1;
x2_e=(-x_2)+xq_2;
Pe_1=sum(abs(x1_e).^2)/N;
Pe_2=sum(abs(x2_e).^2)/N;
SQNR_1=10*log10(Px_1/Pe_1);
SQNR_2=10*log10(Px_2/Pe_2);
disp(['The signal to Quantization Noise Ratio for signal 1 is: ' num2str(SQNR_1) ' dB.' ]);
```

```
disp(['The signal to Quantization Noise Ratio for signal 2 is: ' num2str(SQNR_2) ' dB.' ]);

subplot(2,1,1)
plot(n,xq_1),grid on,xlabel('Indices'),ylabel('Amplitude');
subplot(2,1,2)
plot(n,xq_2),grid on,xlabel('Indices'),ylabel('Amplitude');
```

Output:



2. Simple sinusoid quantized to various bits per sample

Generate a 100 Hz sinusoid sampled at 10000 samples/sec and quantized at 1_bit/sample. Now increase the bit depth for various numbers of bits per sample (2, 3, 4, 5, 6, 7, 8) and attach plots. You can use two column formats for plotting (but the diagrams should be visible).

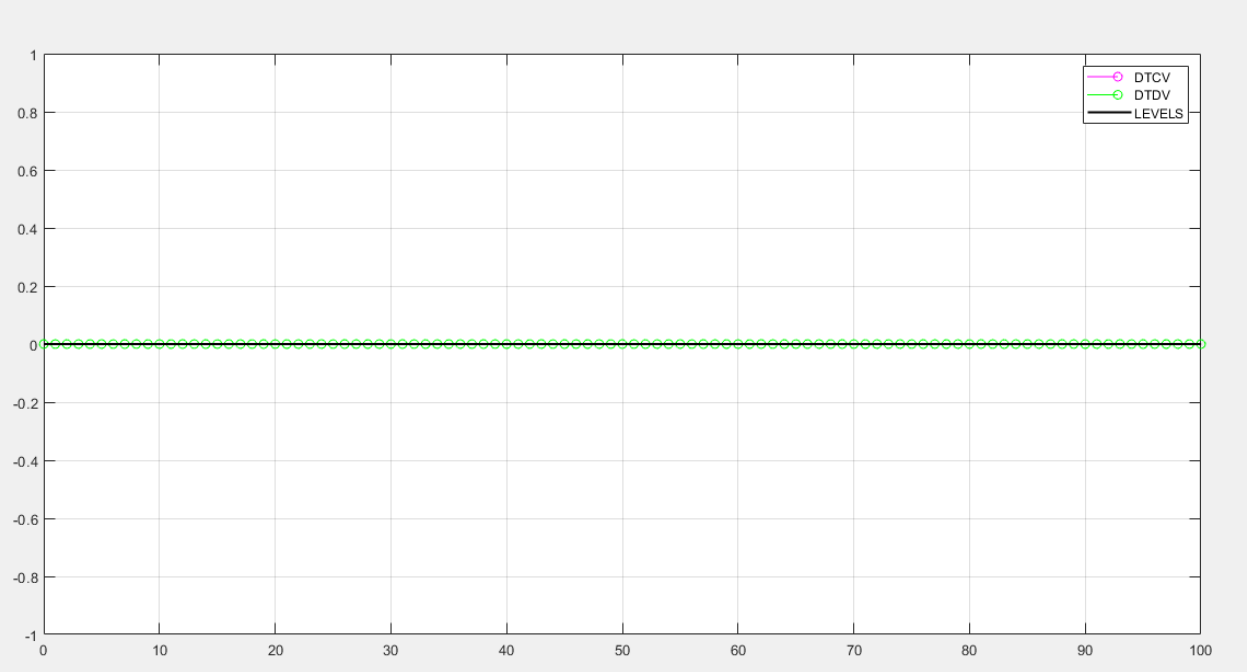
Code:

```
clear all;close all;clc;
F=100;
Fs=10000;
Ts=1/Fs;
N=500;
n=[0:N-1];
```

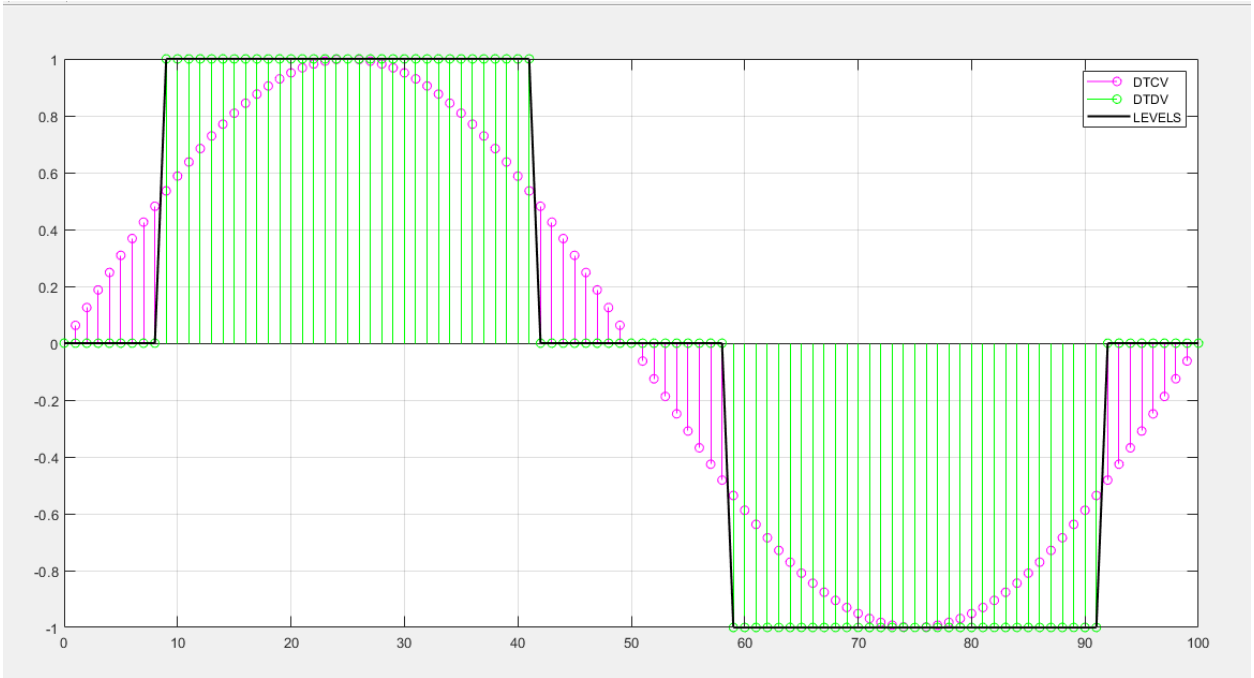
```
t=[0:Ts:N*Ts];
x=sin(2*pi*F*t);
q=input('Enter No. of bits (1-8): ');
fd=F/Fs;
xn=sin(2*pi*fd*n);
qs=input('Select the method of quantization, Press 1 for Round-off, 2 for Floor and 3 for Ceil: ');
max=2^(q-1)-1;
xn=max*xn;

if qs==1
xq = round(xn);
elseif qs==2
xq = floor(xn);
elseif qs==3
xq = ceil(xn);
end

figure;
stem(n,xn,'m');xlim([0 100]),grid on;hold on;
%figure;
stem(n,xq,'g');xlim([0 100]);hold on;
%figure;
plot(n,xq,'k','linewidth',1.5);xlim([0 100]);
legend('DTCV','DTDV','LEVELS');
```

Outputs:

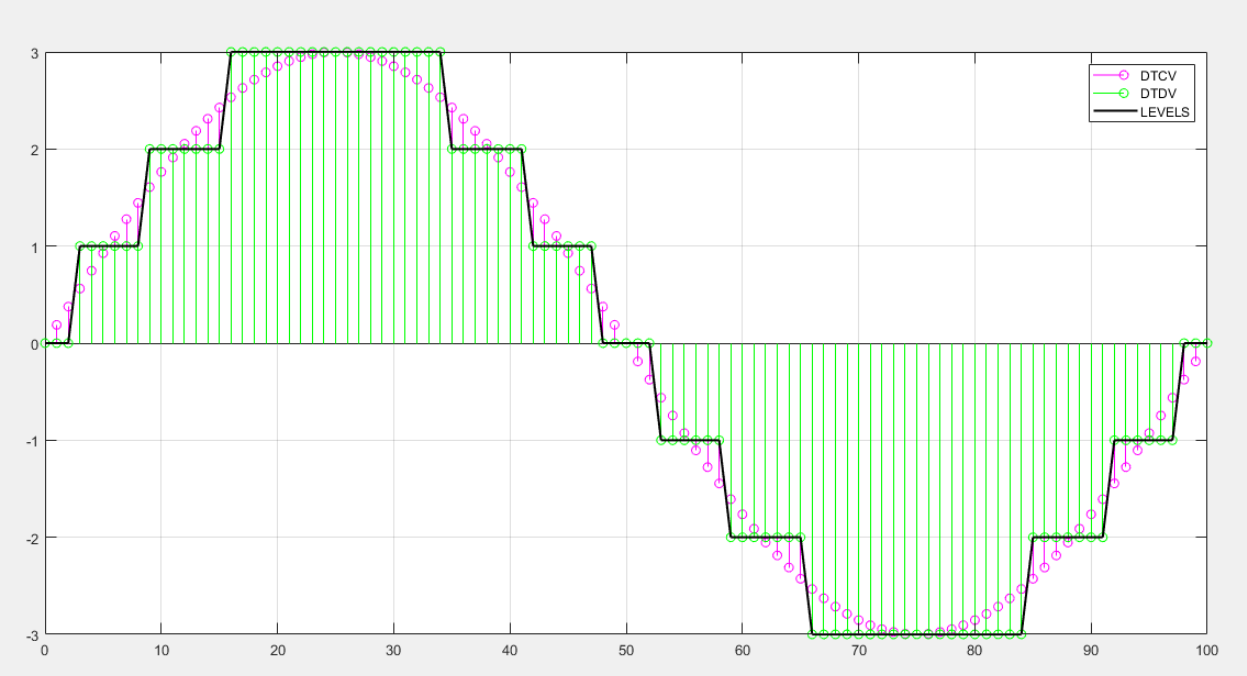
At 1 bits/sample:



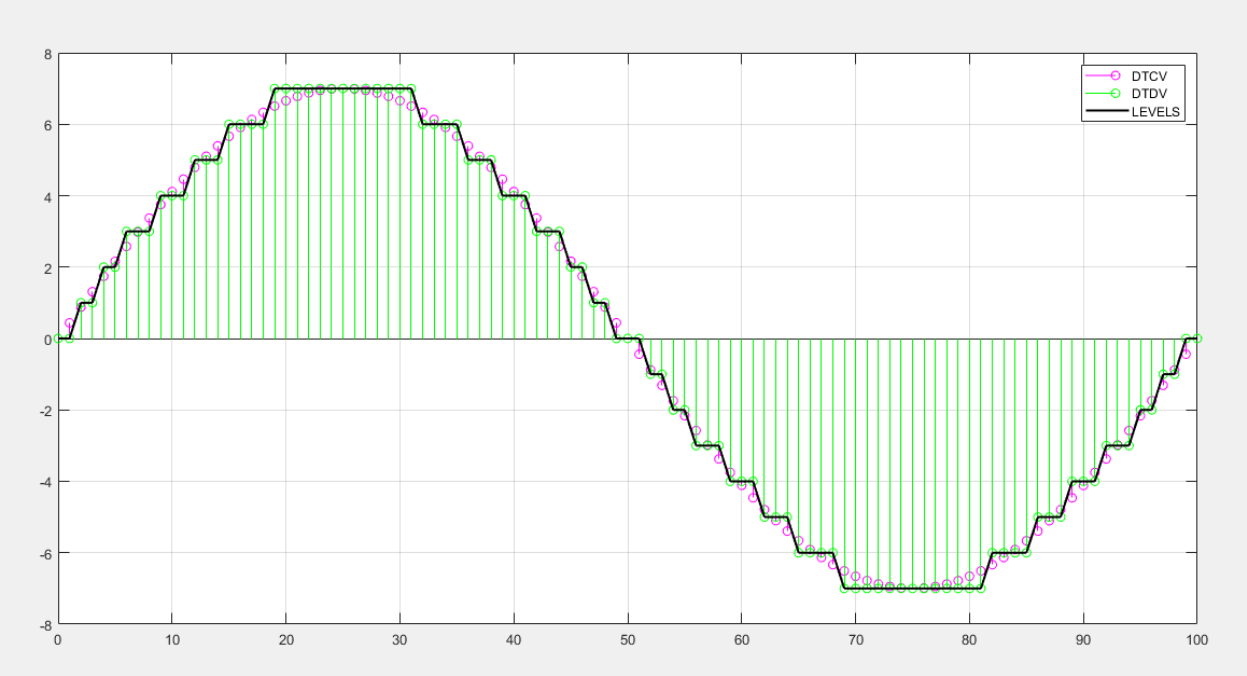
At 2 bits/sample:



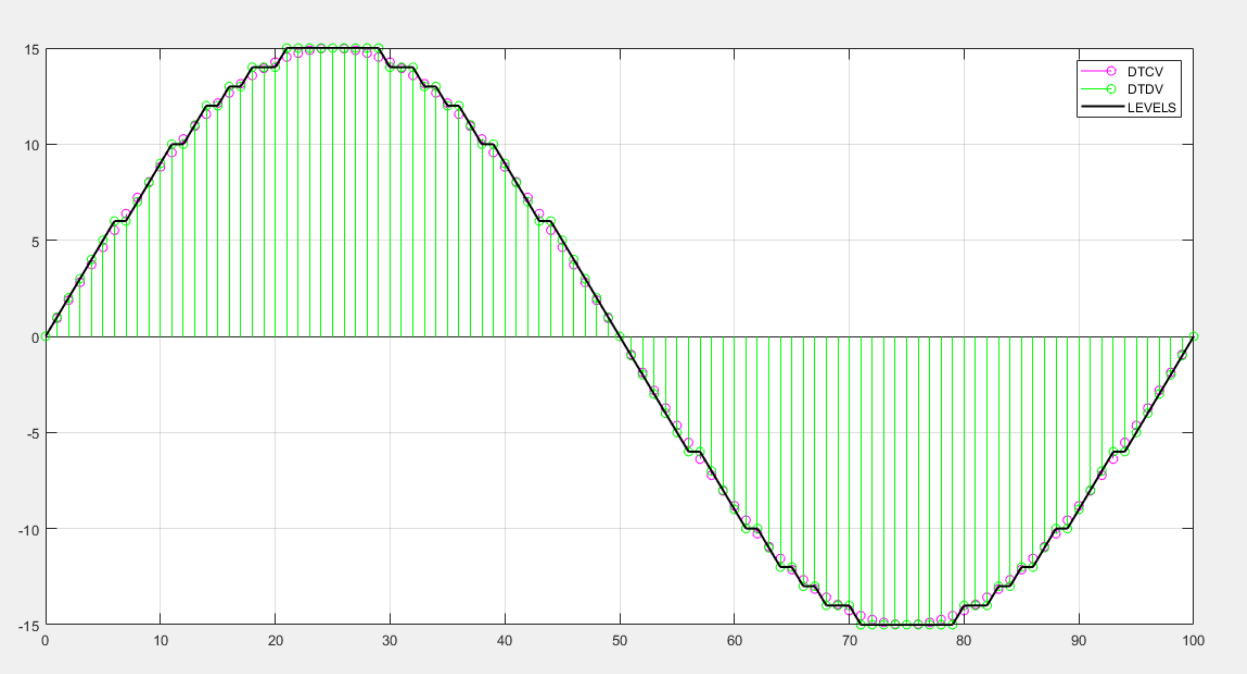
At 3 bits/sample:



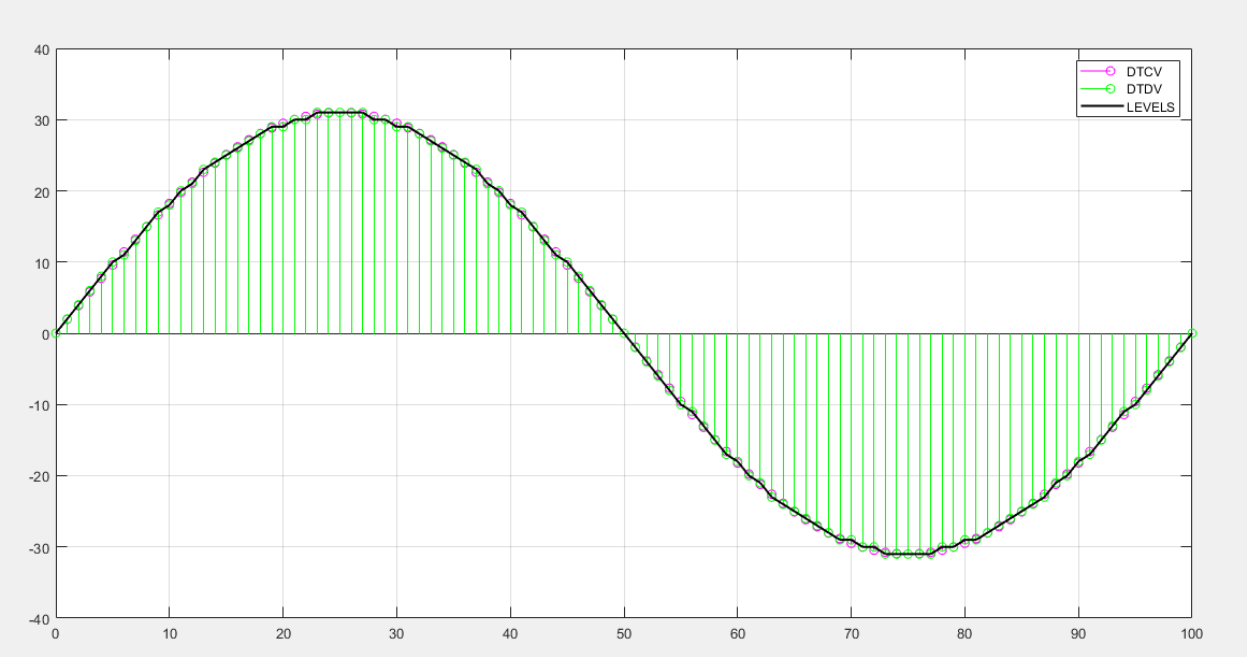
At 4 bits/sample:



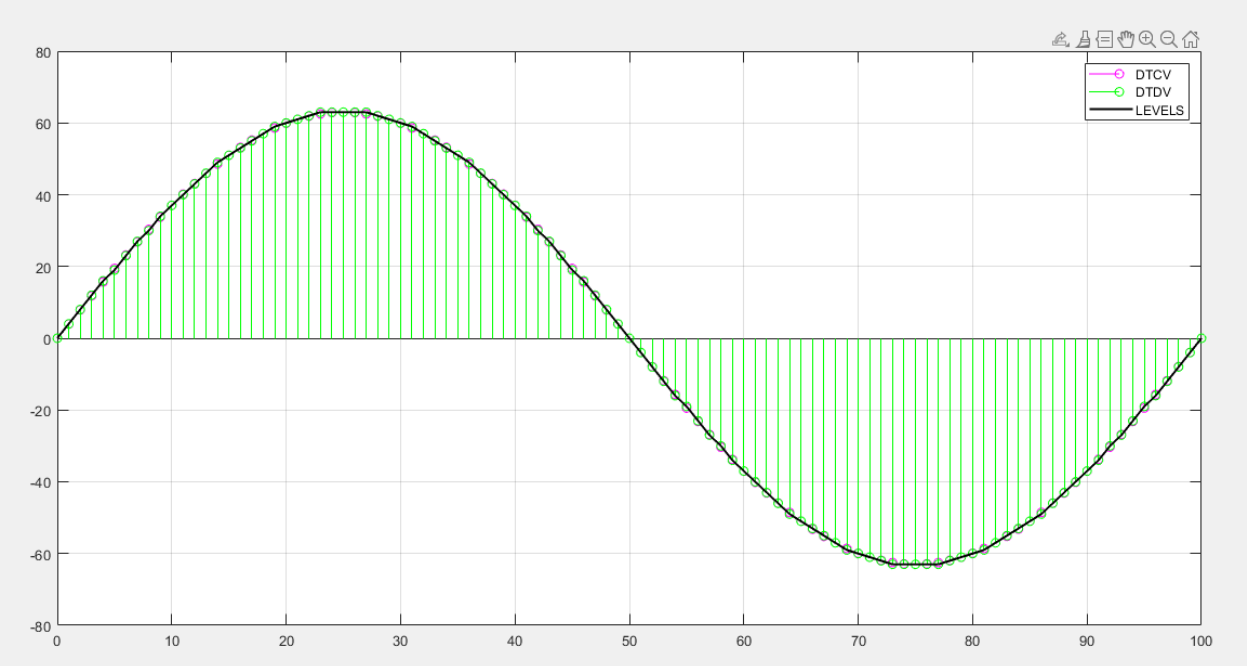
At 5 bits/sample:



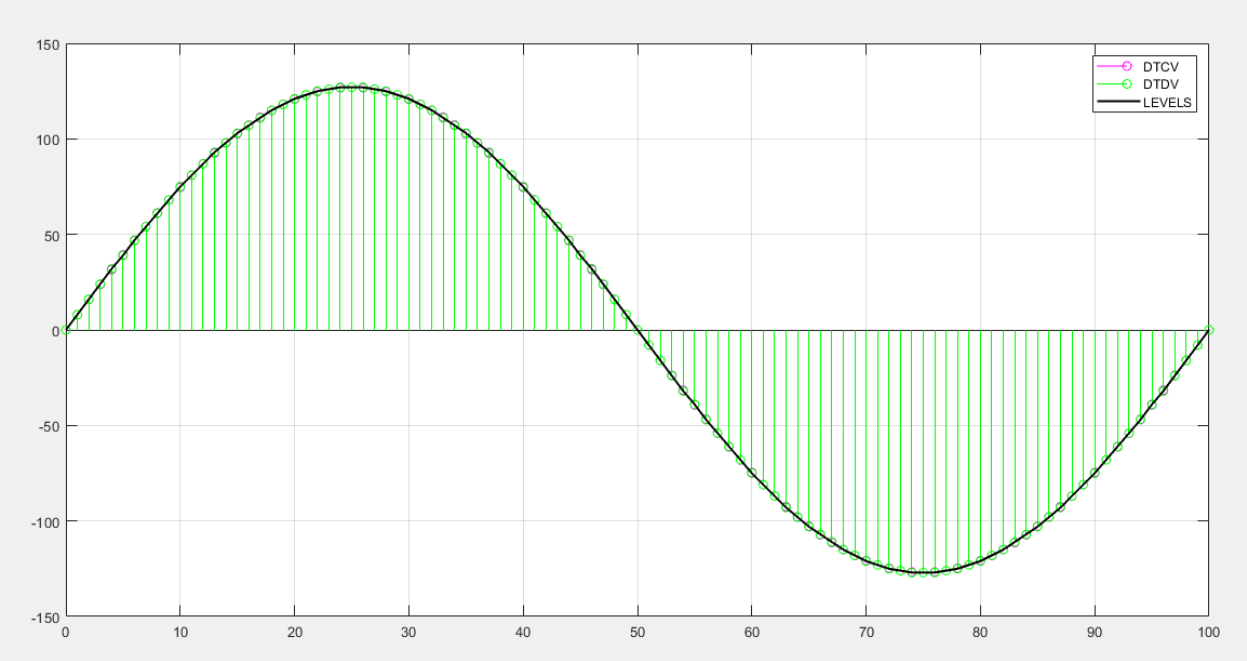
At 6 bits/sample:



At 7 bits/sample:



At 8 bits/sample:



3. Audio signal quantization to various bits per sample

Use your recorded voice in last session and quantize it at 1 bit /sample.

Change bit depth to 2,3,4 and then listen and take notes of your observations.

Decide no. of bits for audio until quality stops improving.

Code:

```
clear all;close all;clc;
[y,Fs] = audioread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab
Codes\moeed44100output.wav');
bits = input("No. of bits : ");
method = input ("1)Round"+newline+'2)Ceil'+newline+'3)Floor'+newline+"Method of quantization:
");
max=(2^(bits-1))-1;
y=max*y;
if method == 1
   xq = round(y)/max;
elseif method == 2
   xq = ceil(y)/max;
elseif method == 3
   xq = floor(y)/max;
end
y=y/max;
restoredefaultpath
 sound(xq,Fs);
```

Observations: Increasing bit depth improved sound quality and the best no of bits were 4; at 4 bits we observed the best results.

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 03

OBJECTIVE:

To study impulse response, observe convolution technique in signal processing, and verify different properties like causality, commutative, distributive and associative properties.

THEORY:

1. Convolution is given as: y(n) = x(n)*h(n) =

   $$: \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)\,.$$

   i.e.one can compute the output y(n) to a certain input x(n) when impulse response h(n) of that system is known. Convolution holds commutative property.

2. The length of the resulting convolution sequence is N+M-1, where N and M are the lengths of two convolved signals respectively.

3. In causal system, the outputs only depend on the past and/or present values of inputs and NOT on future values. This means that the impulse response h(n) of a causal system will always exist only for n≥ 0.

PROCEDURE:

1. We have the impulse response of a system as h(n) = {3,2,1, -2,1,0, -4,0,3}
   ↑
2. For x(n)= {1, -2,3, -4,3,2,1}
   ↑

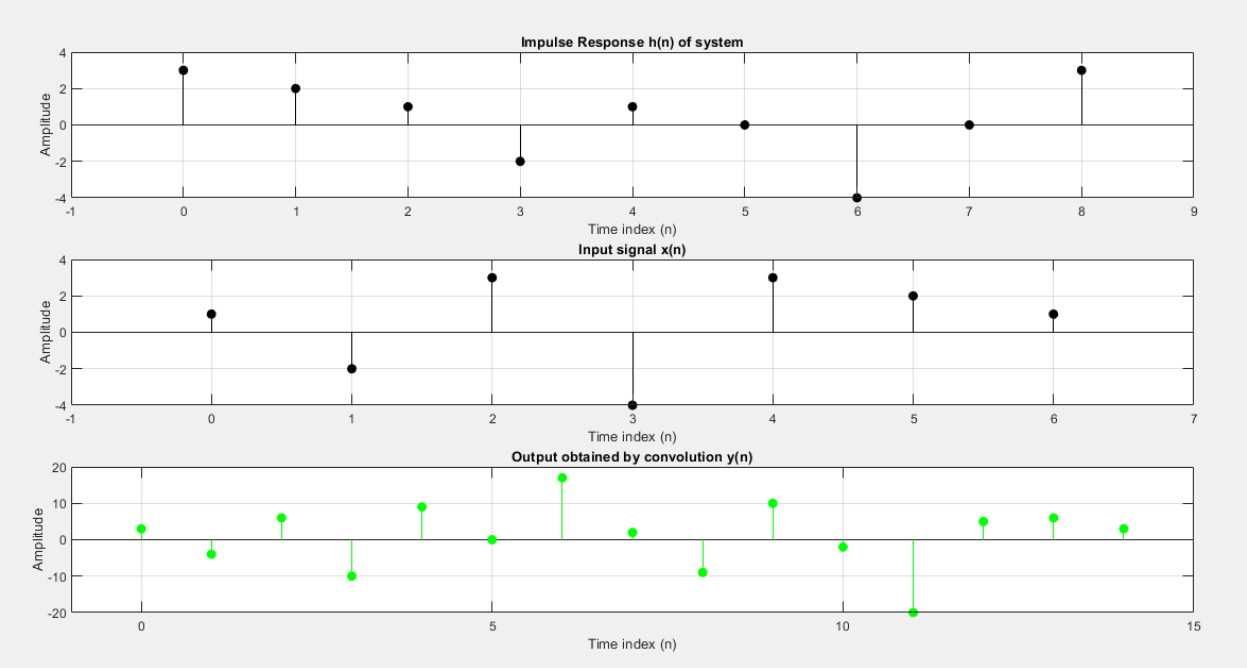
STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Open M-file editor and write the following code:

```
clear all;close all;clc;

h = [3 2 1 -2 1 0 -4 0 3]; %impulse response
org_h = 1;              %sample location at which the origin of h(n) exists
nh = [1:length(h)]-org_h;  %index vector for h(n)

x = [1 -2 3 -4 3 2 1];    %input sequence
org_x = 1;              %sample location at which the origin of x(n) exists
nx = [1:length(x)]-org_x;  %index vector for x(n)

y = conv(h,x);
ny = [nh(1)+nx(1): nh(end)+nx(end)]; %index vector for convolved y(n)

figure,
subplot(3,1,1),
stem(nh,h,'filled','k');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nh(1)-1 nh(end)+1]);
title('Impulse Response h(n) of system');
grid;

subplot(3,1,2),
stem(nx,x,'filled','k');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nx(1)-1 nx(end)+1]);
title('Input signal x(n)');
grid;

subplot(3,1,3),
stem(ny,y,'g','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny(1)-1 ny(end)+1]);
title('Output obtained by convolution y(n)');
grid;
```

1. Save the file as **P031.m** in your current directory and 'run' it.
2. Calculate the length of input signal (N) and impulse response (M) used in above task?
3. Calculate the length of the output sequence and verify the result with N+M-1
4. Try to learn, explore the code and make notes.
5. Now modify the above code such that h(n)= {3,2, 1, -2,1,0, -4,0,3} (origin is shifted) and check for causality. ↑

1.) Output:



2.) & 3.) :

Editor - C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\Lab3.m

Lab3.m | Lab3Task1.m | Lab3Task2_3.m | Lab3Task4.m

```
clear all;close all;clc;

h = [3 2 1 -2 1 0 -4 0 3]; %impulse response
org_h = 1;                 %sample location at which the origin of h(n) exists
nh = [1:length(h)]-org_h;  %index vector for h(n)

x = [1 -2 3 -4 3 2 1];     %input sequence
org_x = 1;                 %sample location at which the origin of x(n) exists
nx = [1:length(x)]-org_x;  %index vector for x(n)

y = conv(h,x);
ny = [nh(1)+nx(1): nh(end)+nx(end)]; %index vector for convolved y(n)

figure,
subplot(3,1,1),
stem(nh,h,'filled','k');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nh(1)-1 nh(end)+1]);
title('Impulse Response h(n) of system');
```

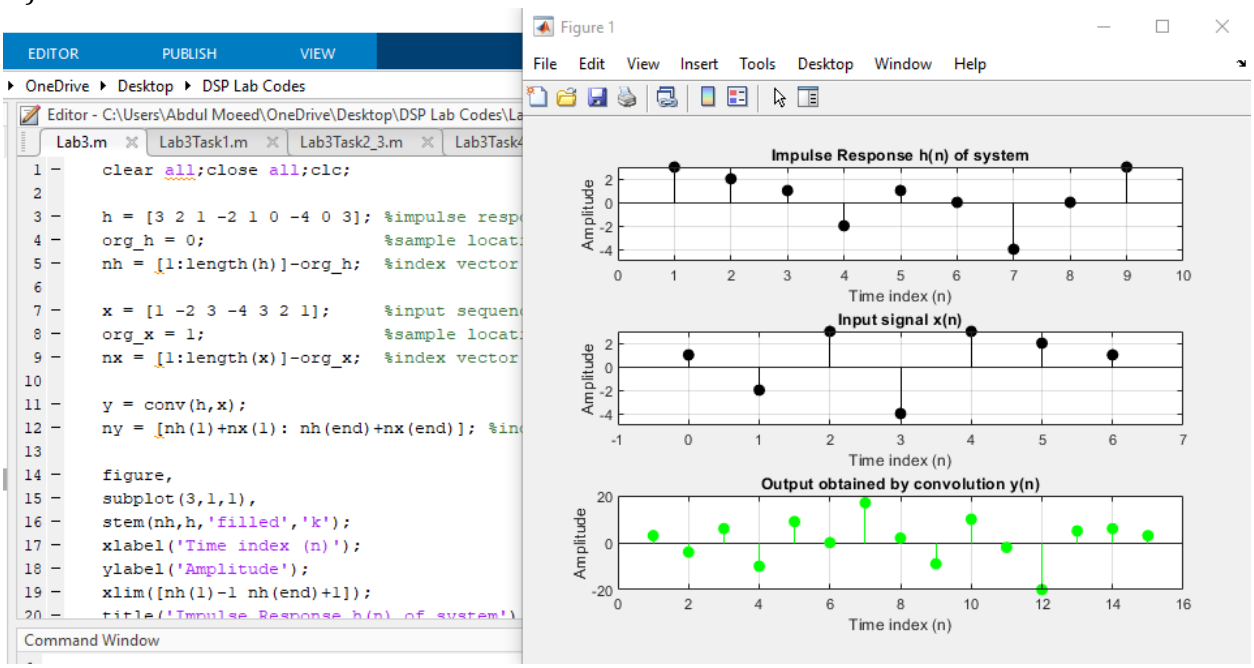
Command Window

```
>> length(x)

ans =

     7

>> length(h)

ans =

     9
```

Workspace

Name	Value
ans	9
h	[3,2,1,-2,1,0,-4,0,3]
nh	[0,1,2,3,4,5,6,7,8]
nx	[0,1,2,3,4,5,6]
ny	1x15 double
org_h	1
org_x	1
x	[1,-2,3,-4,3,2,1]
y	1x15 double

5.):



EXERCISE:

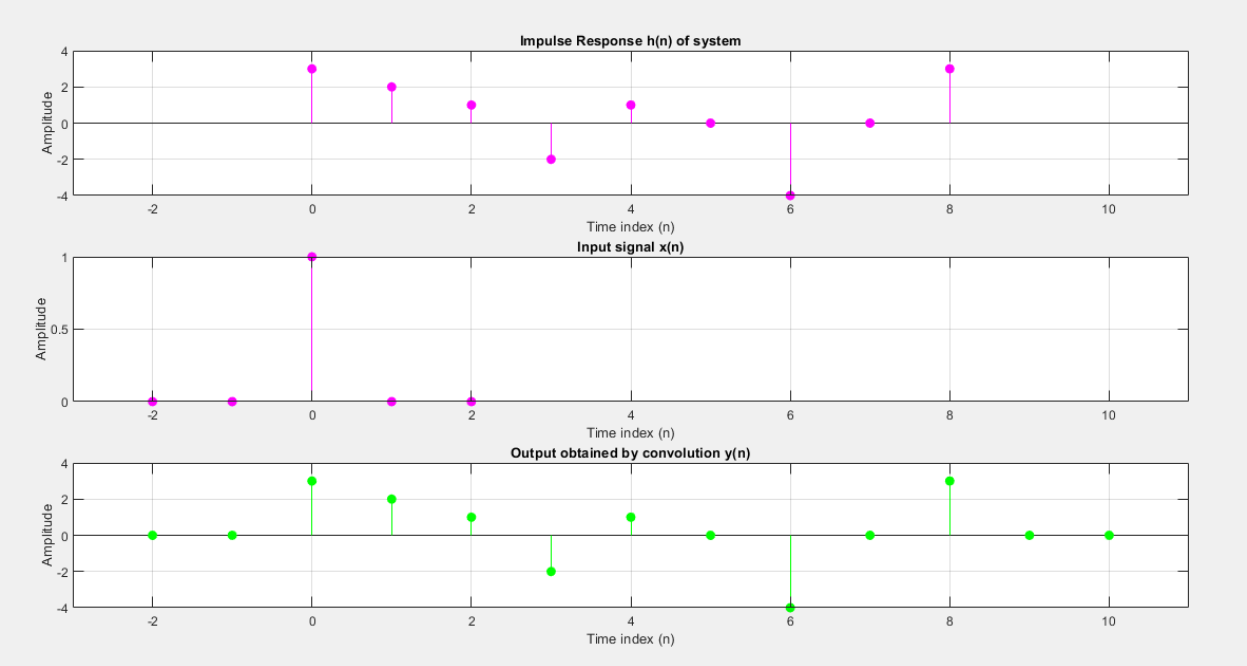
1. What will happen if we input x(n)= {0,0,1,0,0} into the above system.
↑

Code:

```
%Lab3Task1
clear all;close all;clc;

h = [3 2 1 -2 1 0 -4 0 3];
org_h = 1;
nh = [1:length(h)]-org_h;

x = [0,0,1,0,0];
org_x = 3;
nx = [1:length(x)]-org_x;

y1 = conv(h,x);
ny1 = [nh(1)+nx(1): nh(end)+nx(end)]; %index vector for convolved y(n)

figure,
subplot(3,1,1),
stem(nh,h,'filled','m');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny1(1)-1 ny1(end)+1]);
title('Impulse Response h(n) of system');
```

```
grid;

subplot(3,1,2),
stem(nx,x,'filled','m');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny1(1)-1 ny1(end)+1]);
title('Input signal x(n)');
grid;

subplot(3,1,3),
stem(ny1,y1,'g','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny1(1)-1 ny1(end)+1]);
title('Output obtained by convolution y(n)');
grid;
```

Output:



2. Can you prove the commutative property of the convolution?

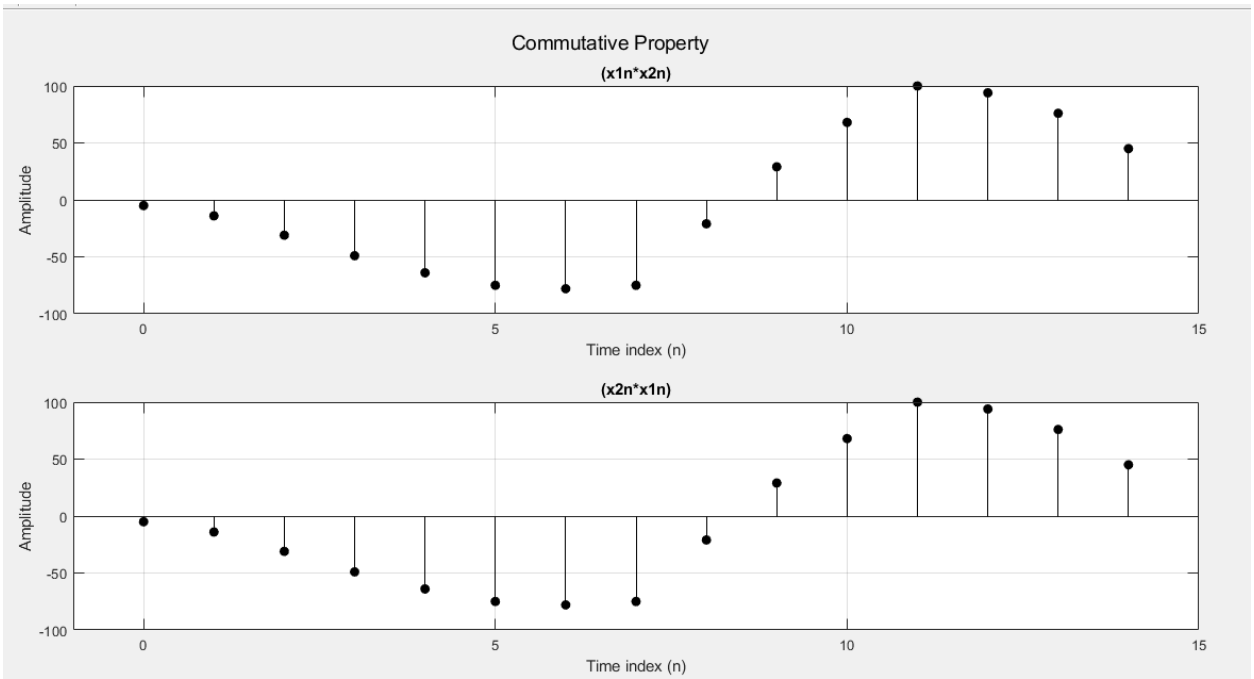
3. Modify the code to prove Associative and Distributed properties of the convolution.

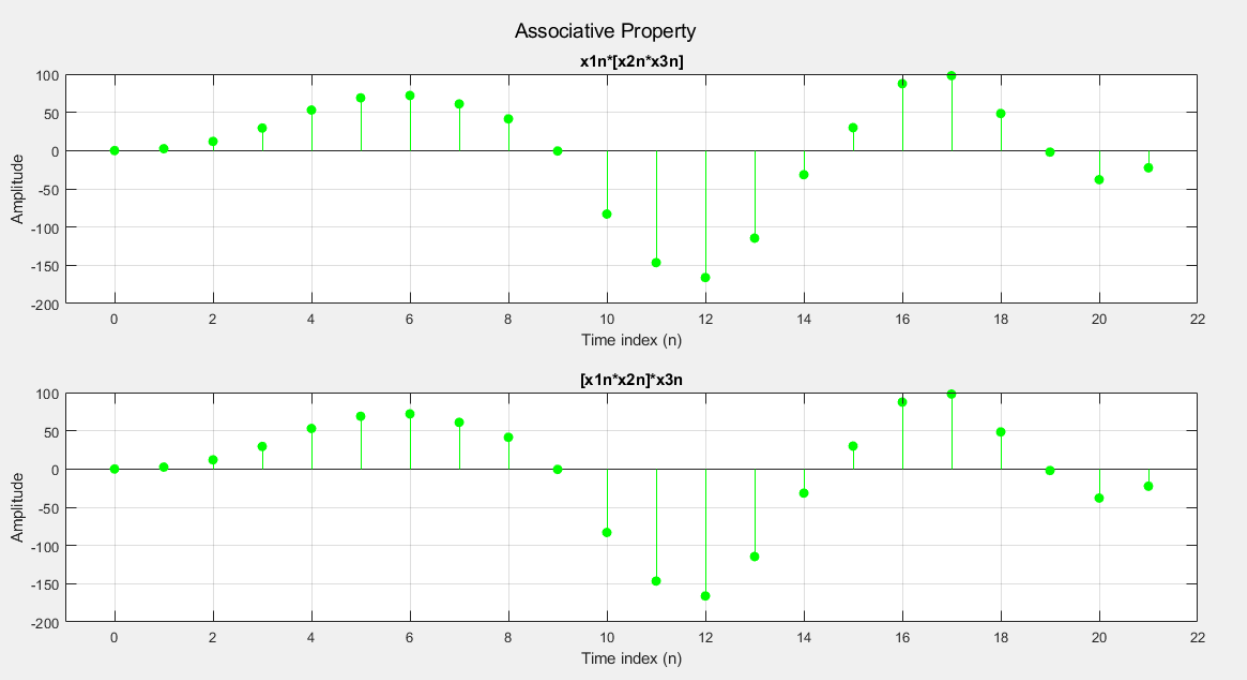
Code:

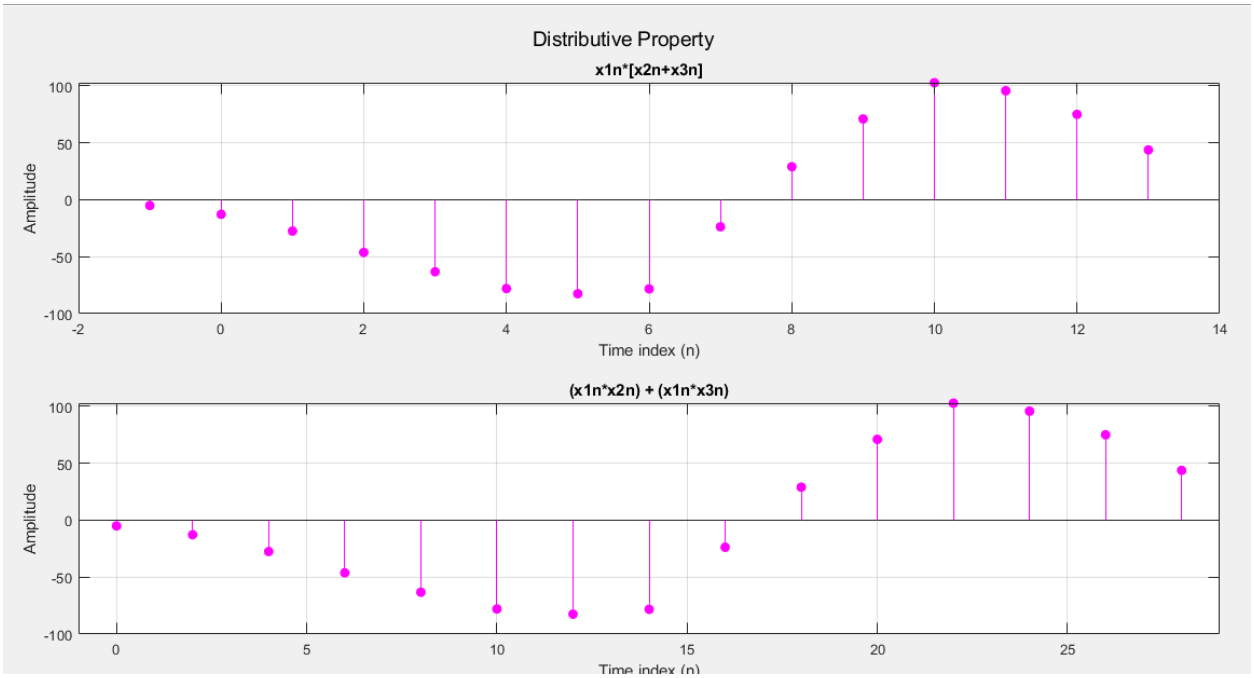
```
clear all;close all;clc;

x1 = [-2.5 -2 -1.5 -1 1 1.5 2 2.5];
x2 = [2 4 8 10 12 14 16 18];
x3 = [0 -0.5 -1 0 0.5 1 0 -0.5 ];

orgin_x1 = 1;
nx1 = [1:length(x1)]-orgin_x1;
orgin_x2 = 1;
nx2 = [1:length(x2)]-orgin_x2;
orgin_x3 = 1;
nx3 = [1:length(x3)]-orgin_x3;

%commutative
%L.H. S:
y1 = conv(x1,x2);
%R.H.S:
y2 = conv(x2,x1);
ny1 = [nx1(1)+nx2(1): nx1(end)+nx2(end)];
ny2 = [nx2(1)+nx1(1): nx2(end)+nx1(end)];

figure;
subplot(2,1,1),
stem(ny1,y1,'k','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny1(1)-1 ny1(end)+1]);
title('(x1n*x2n)');
grid on;
subplot(2,1,2),
stem(ny2,y2,'k','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([ny2(1)-1 ny2(end)+1]);
title('(x2n*x1n)');
sgtitle('Commutative Property')
grid on;

%associative x1n*[x2n*x3n]=[x1n*x2n]*x3n
%L.H.S:
a1 = conv(x2,x3);
a2 = conv(x1,a1);
na1 = [nx1(1)+nx2(1)+nx3(1): nx1(end)+nx2(end)+nx3(end)];
%R.H.S:
a3 = conv(x1,x2);
a4 = conv(a3,x3);
na2 = [nx1(1)+nx2(1)+nx3(1): nx1(end)+nx2(end)+nx3(end)];
```

```
figure;
subplot(2,1,1),
stem(na1,a2,'g','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([na1(1)-1 na1(end)+1]);
title('x1n*[x2n*x3n]');
grid on;
subplot(2,1,2),
stem(na2,a4,'g','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([na2(1)-1 na2(end)+1]);
title('[x1n*x2n]*x3n');
sgtitle('Associative Property')
grid on;

%distributive x1n*[x2n+x3n] = x1n*x2n + x1n*x3n
%L.H.S:
new_signal_1 = x2+x3;
orgin_new_signal_1 = orgin_x2 + orgin_x3 ;
ns1 = [1:length(new_signal_1)]-orgin_new_signal_1;
d1 = conv(x1,new_signal_1);
nd1=[nx1(1)+ns1(1): nx1(end)+ns1(end)];
%R.H.S:
conv_d1 = conv(x1,x2);
conv_d2 = conv(x1,x3);
nconv_d1 = [nx1(1)+nx2(1): nx1(end)+nx2(end)];
nconv_d2 = [nx1(1)+nx3(1): nx1(end)+nx3(end)];
nd2  = nconv_d1+nconv_d2 ;
d2 = conv_d1 + conv_d2  ;
figure;
subplot(2,1,1),
stem(nd1,d1,'m','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nd1(1)-1 nd1(end)+1]);
title('x1n*[x2n+x3n]');
grid on;
subplot(2,1,2),
stem(nd2,d2,'m','filled');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nd2(1)-1 nd2(end)+1]);
title('(x1n*x2n) + (x1n*x3n)');
sgtitle('Distributive Property')
grid on;
```

Outputs:


Commutative Property
(x1n*x2n)
Amplitude
Time index (n)
(x2n*x1n)
Amplitude
Time index (n)


Associative Property
x1n*[x2n*x3n]
Amplitude
Time index (n)
[x1n*x2n]*x3n
Amplitude
Time index (n)



4. Convolve your recorded sound with drumloop.wav. Note your observation

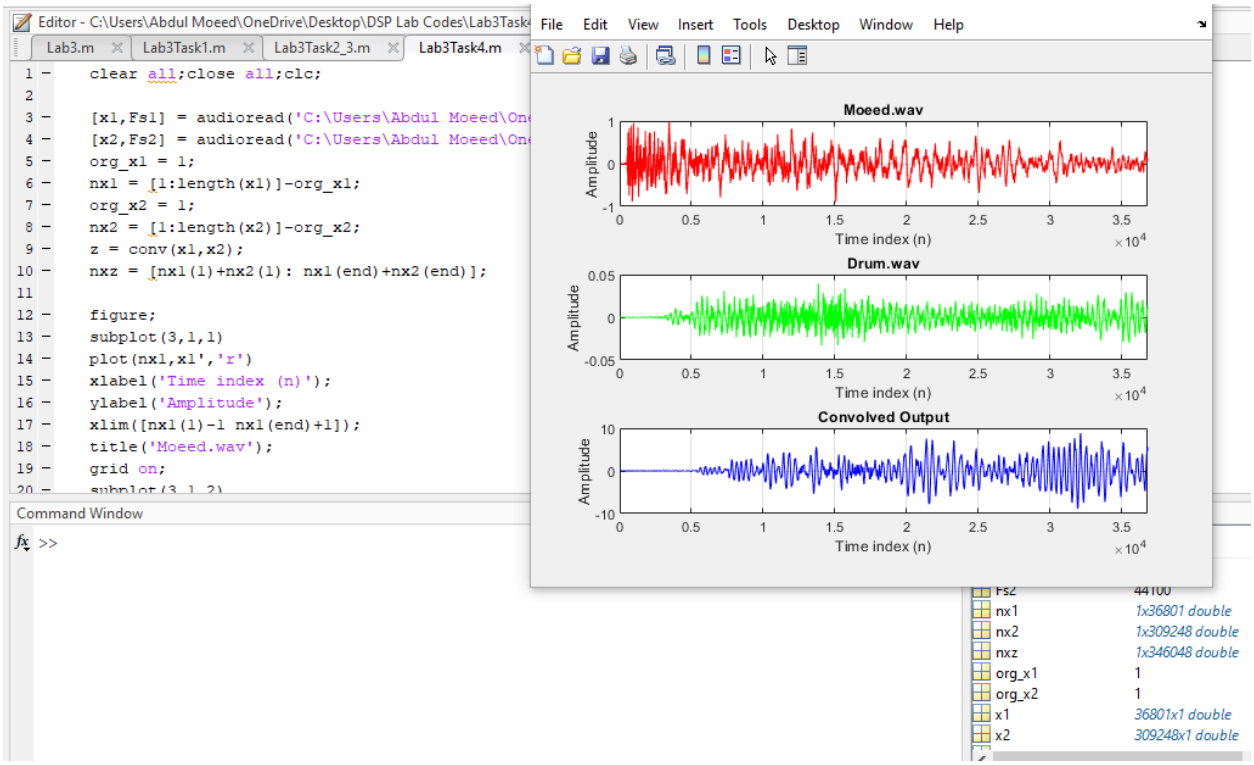
   a) Plot the output.
   b) Listen the output

Code:

```
clear all;close all;clc;

[x1,Fs1] = audioread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\drum.wav');
[x2,Fs2] = audioread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab
Codes\moeed44100output.wav');
org_x1 = 1;
nx1 = [1:length(x1)]-org_x1;
org_x2 = 1;
nx2 = [1:length(x2)]-org_x2;
z = conv(x1,x2);
nxz = [nx1(1)+nx2(1): nx1(end)+nx2(end)];

figure;
subplot(3,1,1)
plot(nx1,x1','r')
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nx1(1)-1 nx1(end)+1]);
title('Moeed.wav');
```

```
grid on;
subplot(3,1,2)
plot(nx2,x2','g')
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nx1(1)-1 nx1(end)+1]);
title('Drum.wav');
grid on;
subplot(3,1,3)
plot(nxz,z','b');
xlabel('Time index (n)');
ylabel('Amplitude');
xlim([nx1(1)-1 nx1(end)+1]);
title('Convolved Output');
grid on;

sound(z,Fs2)
z1 = rescale(z);
audiowrite('C:\Users\Abdul Moeed\OneDrive\Desktop\MoeedConvolved.wav', z1 , Fs2 );
```

Output:



OUTCOME BASED EDUCATION

LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 04

OBJECTIVE:

To study discrete time correlation and apply it to real data to observe the correlation between two signals.

THEORY:

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n)$$

1. **Correlation** is given as where '*l*' is the lag. This is called cross-correlation and it gives the magnitude and location of similarity between two signals. The correlation between x(n) and y(n). It is given as:

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} y(n)x(n-l) = \sum_{n=-\infty}^{\infty} y(n+l)x(n).$$

2. Generally, $rxy(l) = ryx(l)$. These two are the same when x(n) and y(n) are the same signals or when x(n) and y(n) are even symmetric signals.

3. The length of the resulting correlation sequence is N+M-1, where N and M are the lengths of the two signals.

4. Correlation may also be computed using convolution algorithm with a modification that we need to fold one of the signals before applying convolution**.**
   Mathematically, $rxy(n) = x(n) * y(-n)$

STEPS:

1. Generate two sinusoids of length 10 and fd = 0.1 with variable phase.
2. Apply correlation and check for certain properties such as magnitude and location of maximum correlation with varying phases.

PROCEDURE:

1.Make a folder at desktop and name it as your current directory within MATLAB. -
2. Open M-file editor and write the following code:

```
clear all;close all;clc;

n = [0:49];
phi_1 = 0;
phi_2 = 0;

x = sin(2*pi*0.1*n + phi_1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = sin(2*pi*0.1*n + phi_2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

[rxy , l]=xcorr(x,y); %xcorr = cross correlation
[maxR , indR] = max(rxy);
disp(['The correlation at lag zero is: ' num2str(rxy(l==0)) '.']);
disp(['The maximum correlation is at lag ' num2str(l(indR)) '.']);

energy_x=sum(abs(x).^2);
energy_y=sum(abs(y).^2);
disp(['The energy of signal x at lag equals to 0 is ' num2str(energy_x)]);
disp(['The energy of signal y at lag equals to 0 is ' num2str(energy_y)]);

norm_corr=rxy/max(abs(rxy));
perct_corr=norm_corr*100;

figure;
subplot(4,1,1)
stem(nx,x,'filled','b')
xlabel('Time index (n)');
ylabel('Amplitude')
xlim([nx(1)-1 nx(end)+1])
title('Signal x(n)')
grid on;

subplot(4,1,2),
stem(ny,y,'filled','r')
xlabel('Time index (n)'),
ylabel('Amplitude')
xlim([ny(1)-1 ny(end)+1])
title('Signal y(n)')
grid on;

subplot(4,1,3)
stem(l,rxy,'filled','k')
xlabel('Lag index (l)');
ylabel('Correlated Output')
title('Correlation')
grid on;

subplot(4,1,4)
```

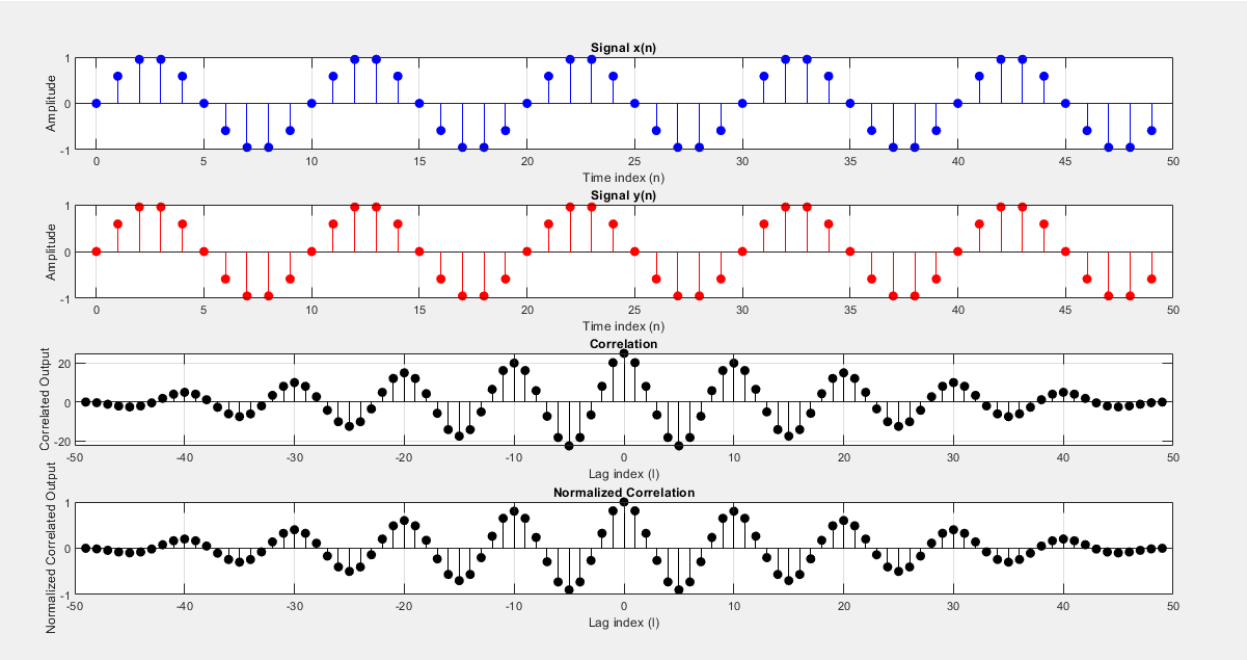
```
stem(l,norm_corr,'filled','k')
xlabel('Lag index (l)'),
ylabel('Normalized Correlated Output')
title('Normalized Correlation')
grid on;
```

Save the file as **P041.m** in your current directory and 'run' it.

Learn the specific logical bits of the code and make notes

Now modify the phase of the second signal to pi/2 (it will make it cosine) and observe the correlation at lag zero. Modify the phase again to 'pi' and observe.

When phi_1 == phi_2 == 0:



Editor - C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\Lab4.m

Lab4.m | Lab4_CommutativeDoesnotHold.m | Lab4Task1.m | Lab4Task2.m

```
clear all;close all;clc;

n = [0:49];
phi_1 = 0;
phi_2 = 0;

x = sin(2*pi*0.1*n + phi_1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = sin(2*pi*0.1*n + phi_2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

[rxy , l]=xcorr(x,y); %xcorr = cross correlation
[maxR , indR] = max(rxy);
disp(['The correlation at lag zero is: ' num2str(rxy(l==0)) '.']);
disp(['The maximum correlation is at lag ' num2str(l(indR)) '.']);

energy_x=sum(abs(x).^2);
```

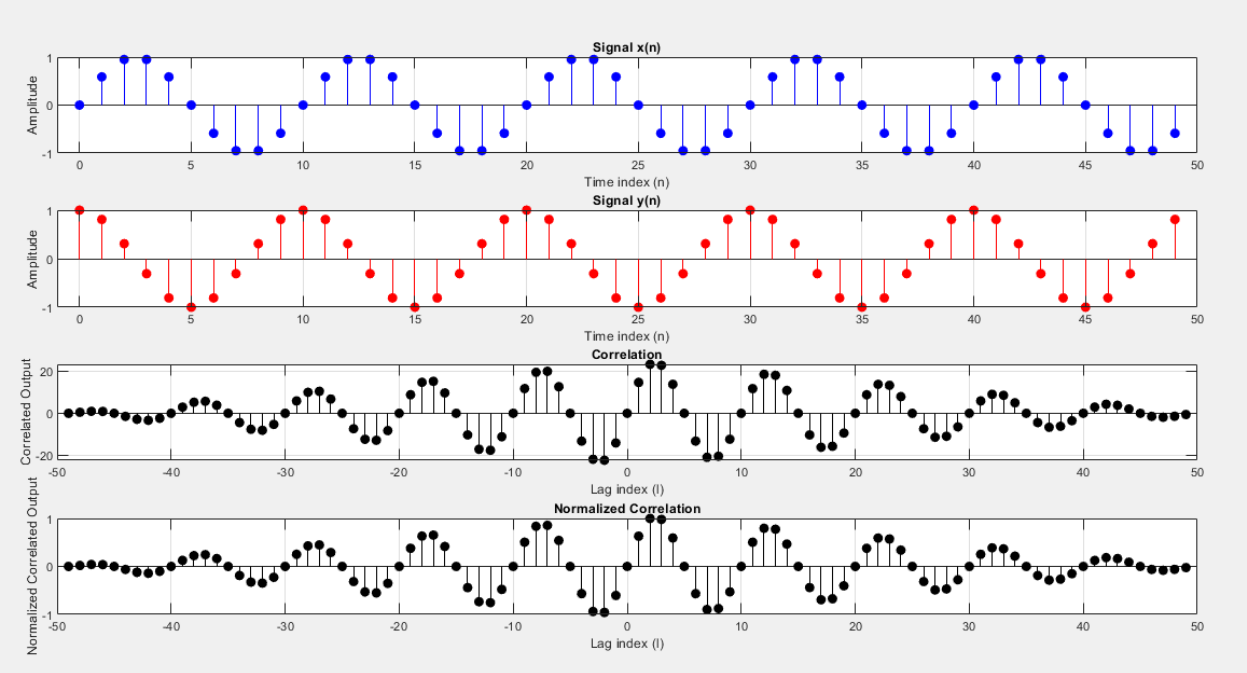
Command Window

```
The correlation at lag zero is: 25.
The maximum correlation is at lag 0.
The energy of signal x at lag equals to 0 is 25
The energy of signal y at lag equals to 0 is 25
>>
```

Workspace

Name	Value
energy_x	25.0000
energy_y	25.0000
indR	50
l	1x99 double
maxR	25
n	1x50 double
norm_corr	1x99 double
nx	1x50 double
ny	1x50 double

When phi_2 =pi/2:



Editor - C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\Lab4.m

Lab4.m | Lab4_CommutativeDoesnotHold.m | Lab4Task1.m | Lab4Task2.m

```
clear all;close all;clc;

n = [0:49];
phi_1 = 0;
phi_2 = pi/2;

x = sin(2*pi*0.1*n + phi_1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = sin(2*pi*0.1*n + phi_2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

[rxy , l]=xcorr(x,y); %xcorr = cross correlation
[maxR , indR] = max(rxy);
disp(['The correlation at lag zero is: ' num2str(rxy(l==0)) '.']);
disp(['The maximum correlation is at lag ' num2str(l(indR)) '.']);

energy_x=sum(abs(x).^2);
```

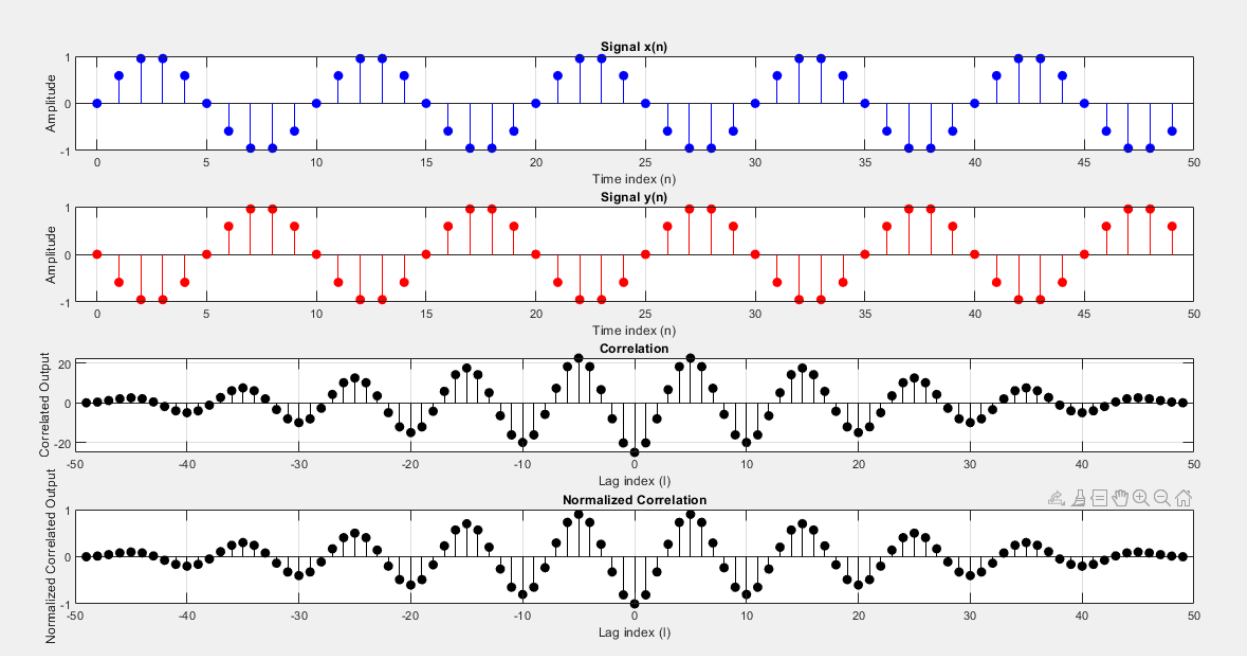
Command Window

```
The correlation at lag zero is: 4.3898e-15.
The maximum correlation is at lag 2.
The energy of signal x at lag equals to 0 is 25
The energy of signal y at lag equals to 0 is 25
>>
```

Workspace

Name	Value
energy_x	25.0000
energy_y	25
indR	52
l	1x99 double
maxR	23.3009
n	1x50 double
norm_corr	1x99 double
nx	1x50 double
ny	1x50 double

When phi_2 =pi:

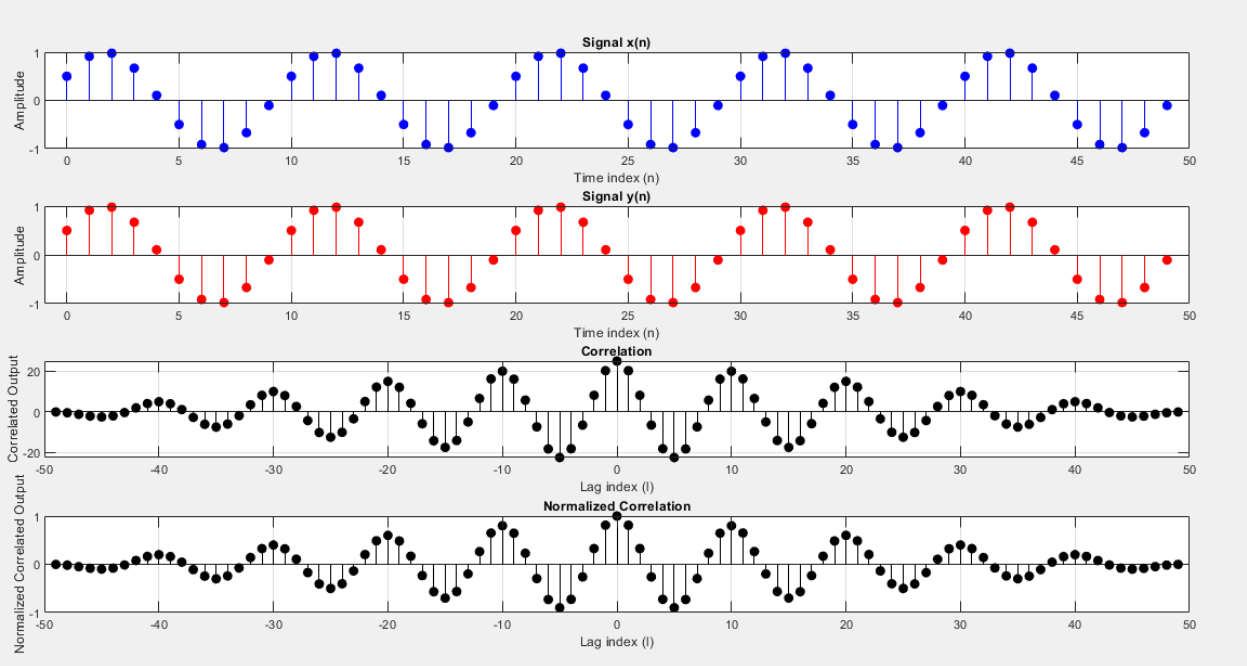


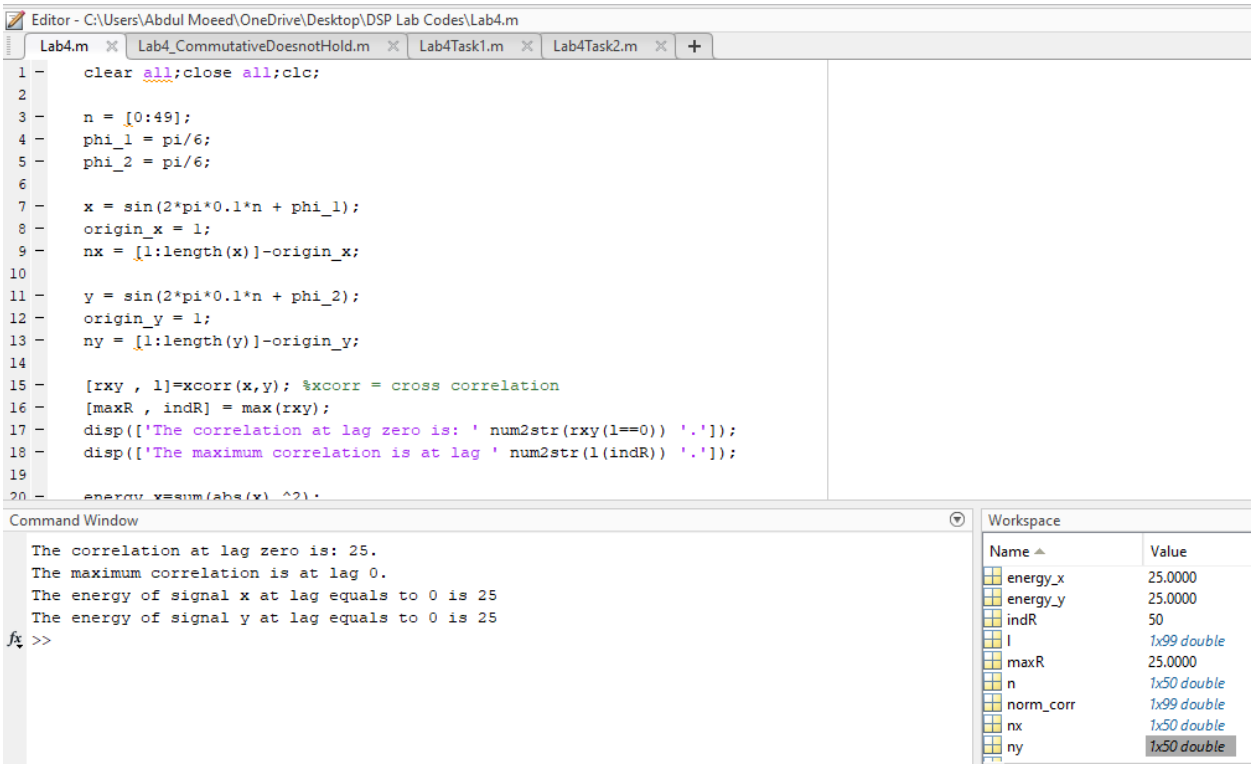
```
clear all;close all;clc;

n = [0:49];
phi_1 = 0;
phi_2 = pi;

x = sin(2*pi*0.1*n + phi_1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = sin(2*pi*0.1*n + phi_2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

[rxy , l]=xcorr(x,y); %xcorr = cross correlation
[maxR , indR] = max(rxy);
disp(['The correlation at lag zero is: ' num2str(rxy(l==0)) '.']);
disp(['The maximum correlation is at lag ' num2str(l(indR)) '.']);
```

```
The correlation at lag zero is: -25.
The maximum correlation is at lag -5.
The energy of signal x at lag equals to 0 is 25
The energy of signal y at lag equals to 0 is 25
>>
```

Name	Value
energy_x	25.0000
energy_y	25.0000
indR	45
l	1x99 double
maxR	22.5000
n	1x50 double
norm_corr	1x99 double
nx	1x50 double
ny	1x50 double

1. Check for auto-correlation (ph1 = ph2) that the lag zero value gives the energy of the signal.





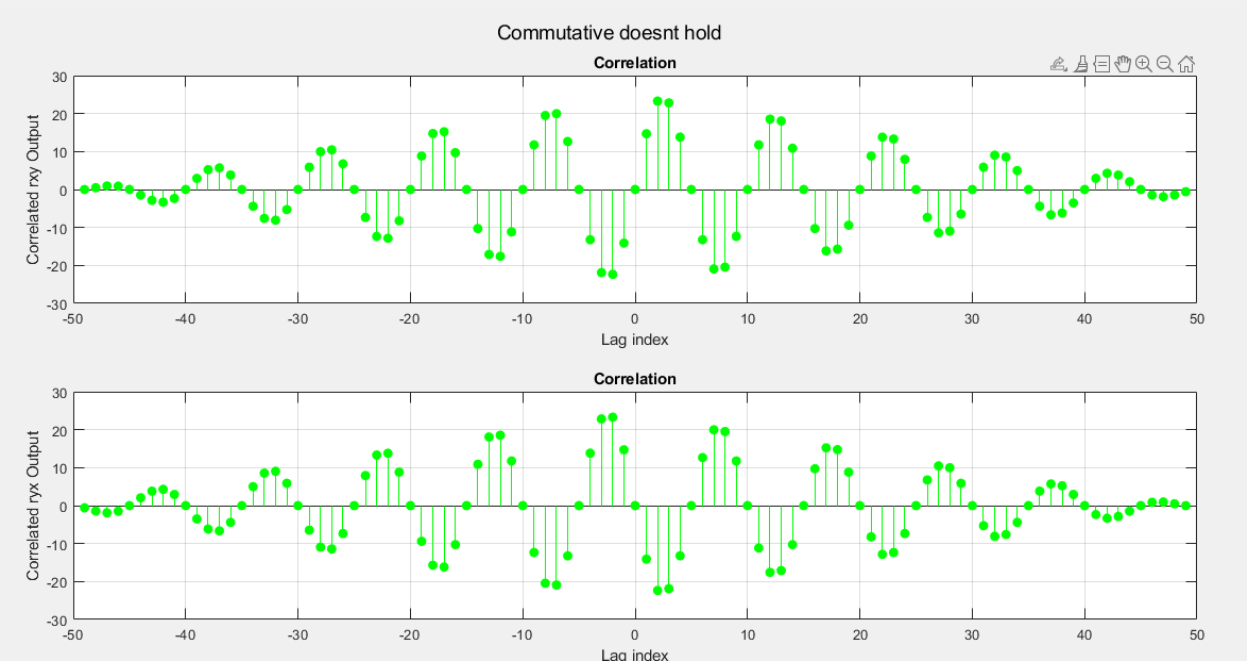
2. Observe that the commutative property does not hold.

Code:

```
%rxy!=ryx
clear all;close all;clc;

n = [0:49];
phi_1 = 0;
phi_2 = 0;

x = sin(2*pi*0.1*n + phi_1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = cos(2*pi*0.1*n + phi_2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

[rxy , c1]=xcorr(x,y);
[ryx , c2]=xcorr(y,x);

figure,
subplot(2,1,1)
stem(c1,rxy,'filled','g')
xlabel('Lag index');
```

```
ylabel('Correlated rxy Output')
title('Correlation')
grid on;
subplot(2,1,2)
stem(c2,ryx,'filled','g')
xlabel('Lag index');,
ylabel('Correlated ryx Output')
title('Correlation')
sgtitle('Commutative doesnt hold')
grid on;
```

Output:



RESULT: Verified through code that correlation is in fact not commutative.

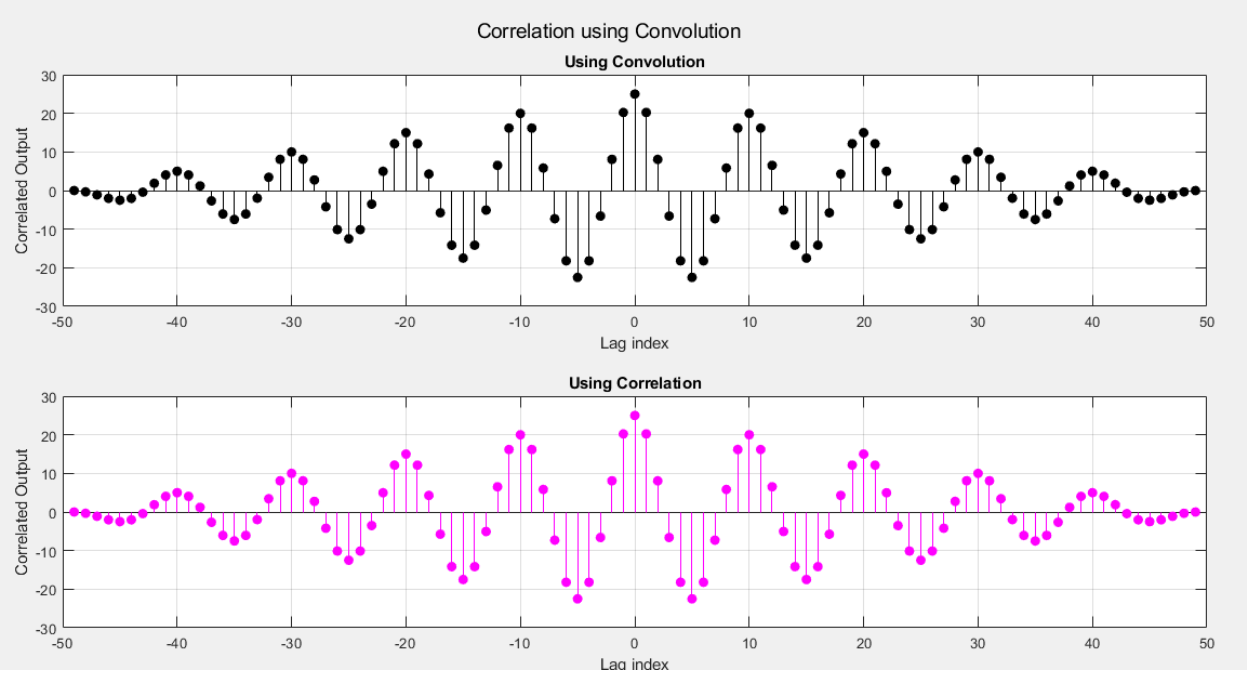
EXERCISE:

1. Modify the code, such that the correlation is obtained using convolution command.

Code:

```
% rxy(n)=x(n)*y(-n)
clc;clear all;close all;

n = [0:49];
ph1 = 0;
ph2 = 0;

x = sin(2*pi*0.1*n + ph1);
origin_x = 1;
nx = [1:length(x)]-origin_x;

y = sin(2*pi*0.1*n + ph2);
origin_y = 1;
ny = [1:length(y)]-origin_y;

z=fliplr(y); %180 ghuma dega array ko hence x(n)==x(-n) bn jayega
rxy_1=conv(x,z);
[rxy_2 l]=xcorr(x,y);

subplot(2,1,1)
stem(l,rxy_1,'filled','k')
xlabel('Lag index');
ylabel('Correlated Output')
title('Using Convolution')
grid on;
subplot(2,1,2)
stem(l,rxy_2,'filled','m')
xlabel('Lag index');
ylabel('Correlated Output')
title('Using Correlation')
sgtitle('Correlation using Convolution')
grid on;
```

Output:



2. Calculate correlation between voltages of any two phases of a 10HP motor using the data given below. First use MS Excel to copy data and then calculate correlation.

Voltage A Min	Voltage B Min	Voltage C Min
189.358	153.917	195.735
189.175	159.719	201.877
188.783	161.575	186.718
188.757	172.186	187.659
176.995	173.206	205.876
180.472	176.865	204.831
180.524	176.917	192.494
180.262	189.28	199.839
181.778	189.828	211.887
179.975	189.462	211.94
178.642	189.253	212.462
180.315	188.94	193.749

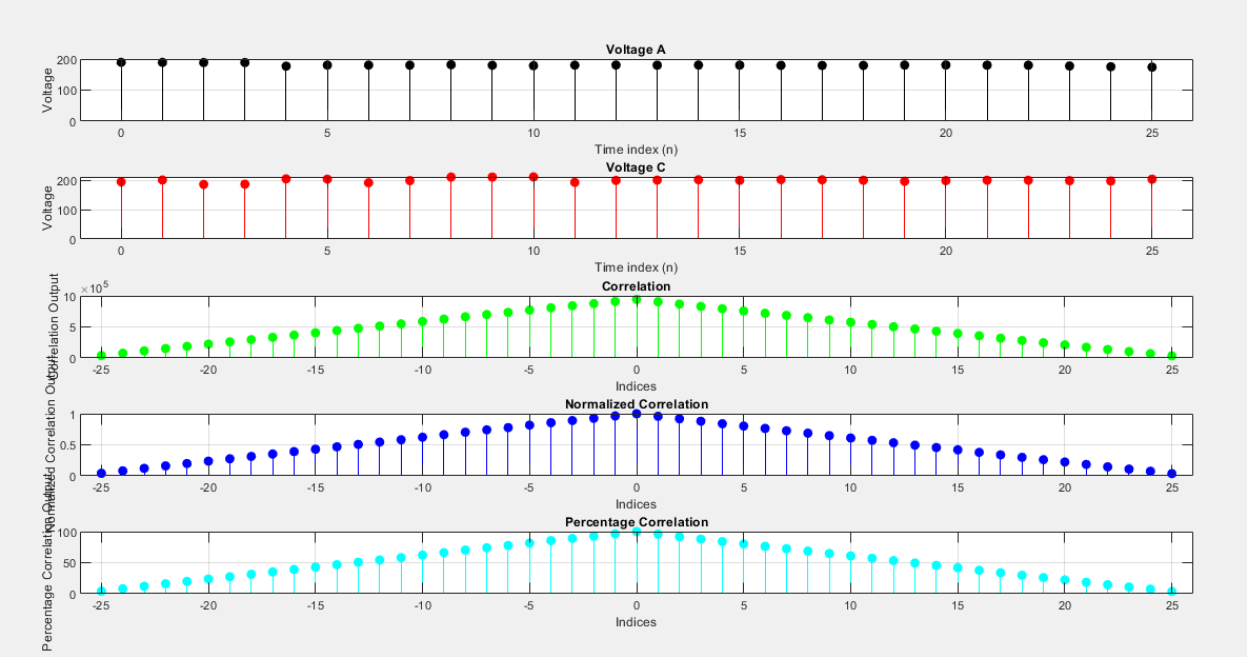
180.707	190.377	200.492
180.262	190.194	201.433
180.628	190.064	202.635
180.315	189.907	200.701
179.635	189.541	203.289
179.243	189.567	202.635
179.4	189.619	200.989
180.576	189.044	197.591
180.837	189.123	199.865
180.184	189.332	201.093
180.08	189.097	201.041
177.675	189.044	199.656
175.297	189.018	198.558
173.99	189.123	204.595

Code:

```
clc;clear all;close all;

[numbers, strings, raw] = xlsread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\Lab4Task3');
numbers_1 = numbers(:,1);
numbers_3 = numbers(:,3);
org_num1 = 1;
num1_x = [ 1:length(numbers_1) ] - org_num1 ;
orgin_num3 = 1;
num3_x = [ 1:length(numbers_3) ] - orgin_num3 ;
[values,indices] = xcorr(numbers_1,numbers_3);
[maxVal,position] = max(values);

disp(['The correlation at lag zero is: ' num2str(values(indices==0)) '.']);
disp(['The maximum correlation is at lag ' num2str(indices(position)) '.']);

normalized_correlation=values/max(abs(values));
percent_correlation=normalized_correlation*100;

subplot(5,1,1)
stem(num1_x,numbers_1,'filled','k')
xlabel('Time index (n)');
ylabel('Voltage')
xlim([num1_x(1)-1 num1_x(end)+1])
```

```
title('Voltage A')
grid on;
subplot(5,1,2)
stem(num3_x,numbers_3,'filled','r')
xlabel('Time index (n)');
ylabel('Voltage')
xlim([num3_x(1)-1 num3_x(end)+1])
title('Voltage C')
grid on;
subplot(5,1,3)
stem(indices,values,'filled','g')
xlabel('Indices');
ylabel('Correlation Output')
xlim([indices(1)-1 indices(end)+1])
title('Correlation')
grid on;
subplot(5,1,4)
stem(indices,normalized_correlation,'filled','b');
xlabel('Indices');
ylabel('Normalized Correlation Output')
xlim([indices(1)-1 indices(end)+1])
title('Normalized Correlation')
grid on;
subplot(5,1,5)
stem(indices,percent_correlation,'filled','Color',[0 1 1]);
xlabel('Indices');
ylabel('Percentage Correlation Output')
xlim([indices(1)-1 indices(end)+1])
title('Percentage Correlation')
grid on;
```

Output:



OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 05

OBJECTIVE:

To study the computer implementation of Discrete Fourier Transform (DFT) and Inverse Fourier Transform (IDFT) using Twiddle factor.

THEORY:

The formulas for the DFT and IDFT are given as

X(K) = $\sum_{n=0}^{N-1} x(n)\, W_N^{kn}$; k=0, 1,...N-1

X(n) = $\frac{1}{N} \sum_{n=0}^{N-1} X(k)\, W_N^{-kn}$; k=0, 1,...N-1

Where by definition $W_N = e^{\frac{-j2\Pi}{N}}$

Which is an Nth root of unity. Where W_N is called Twiddle Factor , also

$[W_N] = [e^{-j2\Pi/N}]^{kn}$

$W_N^{kn} = e^{-j2\Pi kn/N}$

DFT analysis equation in matrix form is

$X_N = [W_N^{kn}]x_N$

DFT synthesis equation in matrix form is

$X_N = [W_N^{kn}]^{-1}X_N$

PROCEDURE:

TASK

Compute 4-point DFT of x(n)= (1,2,3,0).

STEPS

1.Generate given sequence in Matlab.
2.Take N=4 to calculate 4-point DFT.
3.Define 0: N-1 point vector for time and frequency samples.
4.Define W matrix and then use DFT analysis equation to compute DFT.

```
close all,clear all;clc;
x=[1 ,2 ,3 ,0];
N=4;
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N)
nk=n'*k;
WNnk=WN.^nk;
Xk=x*WNnk
```

LAB TASK

Prove DFT synthesis equation using DFT output generated from lab task.

EXERCISE:

1. Forward DFT using matrices

Develop a MATLAB code to find the Forward DFT output of the following time domain sequence by using DFT equation in matrix form. Also plot the magnitude and phase spectrum. Take $F_s = 1000\ samples/sec$

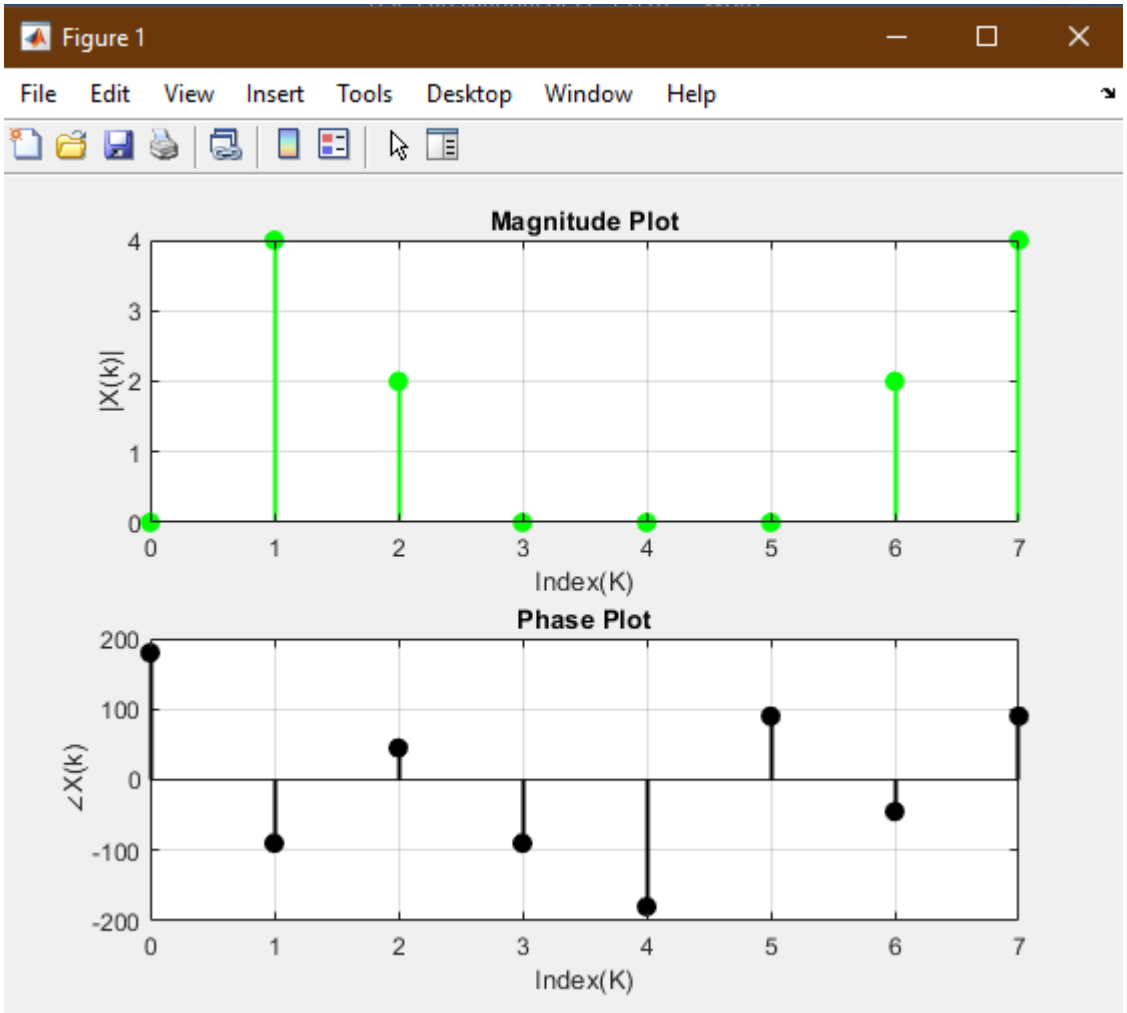
$$x(n) = \{0.3535,\quad 0.3535,\quad 0.6464,\quad 1.0607,\quad 0.3535,\quad -1.0607,\quad -1.3535,\quad -0.3535\}$$

Code:

```
close all,clear all;clc;
x=[0.3535, 0.3535,   0.6464,    1.0607,   0.3535,  -1.0607,  -1.3535 ,  -0.3535];
N=8;
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
 nk=n'*k;
 WNnk=WN.^nk;
```

```
Xk=x*WNnk
mag_Xk=abs(Xk);
phase_Xk=angle(Xk);
phase_degrees=rad2deg(phase_Xk);
figure;
subplot(2,1,1)
stem(n,mag_Xk,'filled','g','LineWidth',1.5)
xlabel('Index(K)'),ylabel('|X(k)|'),title('Magnitude Plot'),grid;
subplot(2,1,2)
stem(n,phase_degrees,'filled','k','LineWidth',1.5)
xlabel('Index(K)'),ylabel('?X(k)'),title('Phase Plot'),grid;
B=conj(WNnk);
Xk_inv=Xk.';
Xn=(1/N)*(Xk)*B
figure;
stem(n,Xn,'filled','g','LineWidth',1.5)
xlabel('Index(n)'),ylabel('x(n)'),title('Signal in time domain x(n)'),grid;
```

Output:



2. Inverse DFT using Matrix inversion

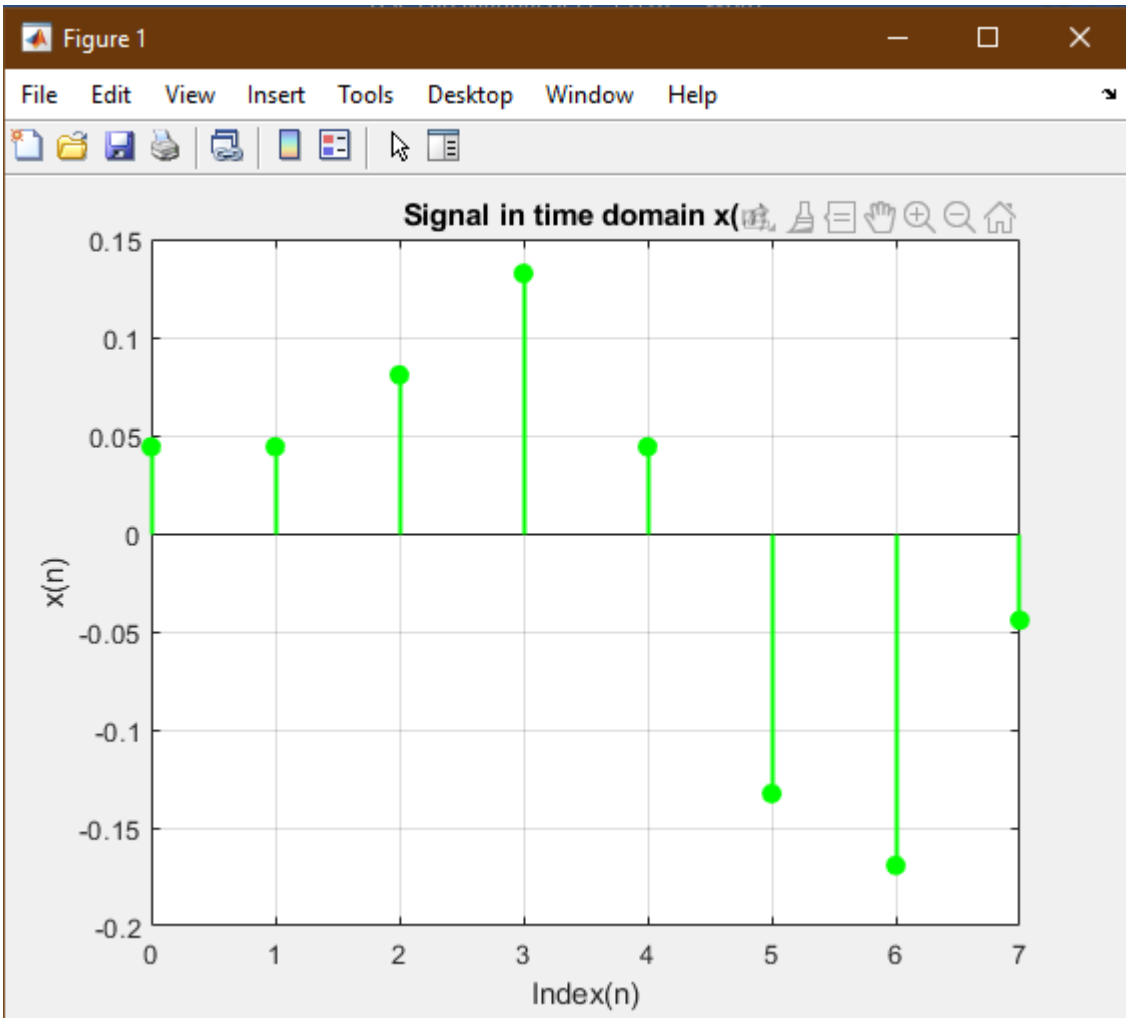
Develop a MATLAB code to find the inverse DFT output of the following frequency domain sequence by using IDFT equation in matrix form (use matrix inversion).

$$X(k) = \{0, \quad 4\angle -90^\circ, \quad 2\angle 45^\circ, \quad 0, \quad 0, \quad 0, \quad 2\angle -45^\circ, \quad 4\angle 90^\circ\}$$

Code:

```
close all,clear all;clc;
Xk=[0,-4i ,1.414+1.414i, 0 , 0, 0, 1.414-1.414i ,4i];
N=8;
n=[0:1:N-1];
k=[0:1:N-1];
WN=exp(-j*2*pi/N);
nk=n'*k;
WNnk=WN.^nk;
B=inv(WNnk);
Xk_inv=Xk.';
Xn=(1/N)*(Xk)*B
figure;
stem(n,Xn,'filled','g','LineWidth',1.5)
xlabel('Index(n)'),ylabel('x(n)'),title('Signal in time domain x(n)'),grid;
```

Output:



3. Inverse DFT using Conjugate method

Develop a MATLAB code to find the inverse DFT output of the following frequency domain sequence by using IDFT equation in matrix form (use conjugate method).

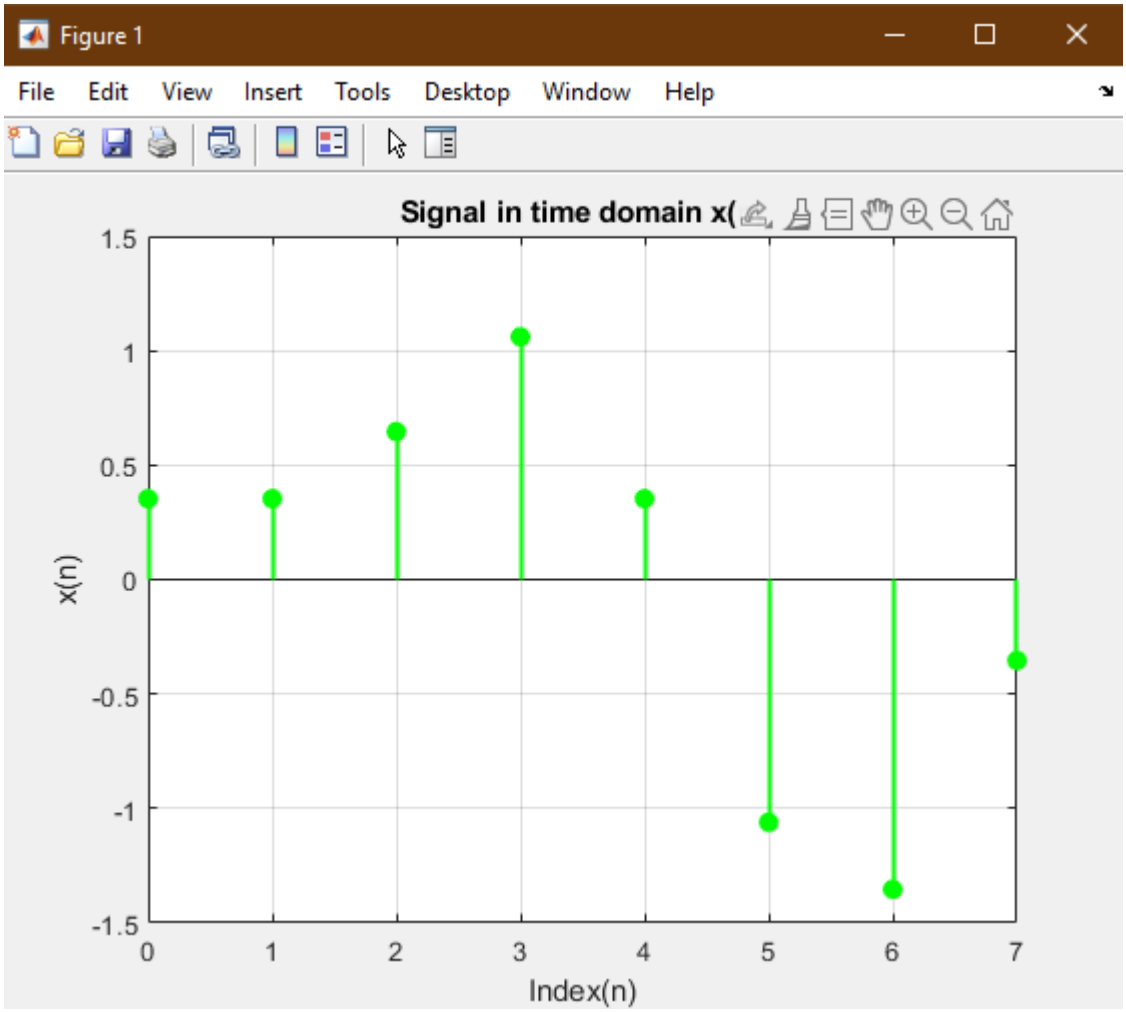
$$X(k) = \{0, \quad 4\angle -90\,^\circ, \quad 2\angle 45\,^\circ, \quad 0\,, \quad 0, \quad 0, \quad 2\angle -45\,^\circ\,, \quad 4\angle 90\,^\circ\}$$

Code:

```
close all,clear all;clc;
Xk=[0, -4i, 1.414+1.414i, 0, 0, 0, 1.414-1.414i, 4i];
N=8;
n=[0:1:N-1];
k=[0:1:N-1];
```

```
WN=exp(-j*2*pi/N);

nk=n'*k;

WNnk=WN.^nk;
B=conj(WNnk);
Xk_inv=Xk.';
Xn=(1/N)*(Xk)*B
figure;
stem(n,Xn,'filled','g','LineWidth',1.5)
xlabel('Index(n)'),ylabel('x(n)'),title('Signal in time domain x(n)'),grid;
```

Output:



OUTCOME BASED EDUCATION

LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 06

OBJECTIVE:

To observe/find different frequency components in an audio signal and plot it with different x-axes.

THEORY:

- DF analysis equation: $X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N}, \quad k = 0,1,\dots,N-1$
- DFT synthesis eq: $x(n) = \frac{1}{N} \sum_{k=0}^{N} \frac{1}{} X(k) e^{j2\pi kn/N}, \quad n = 0,1,\dots,N-1$
- Frequency Resolution is given by $\Delta F = \frac{F_s}{N}$.
- Sample frequencies are given by $F_k = \Delta F \times k, \; k = 0,1,\dots,N-1.$

PROCEDURE:

1. Load an audio file 'noisy.wav' into Matlab.
2. There is a tone added to the speech in this file. The objective is to find the frequency of this tone.
3. Compute the DFT of this signal;
4. Generate frequency vector in Hz.
5. Display the DFT and observe the frequency of the added tone.

STEPS

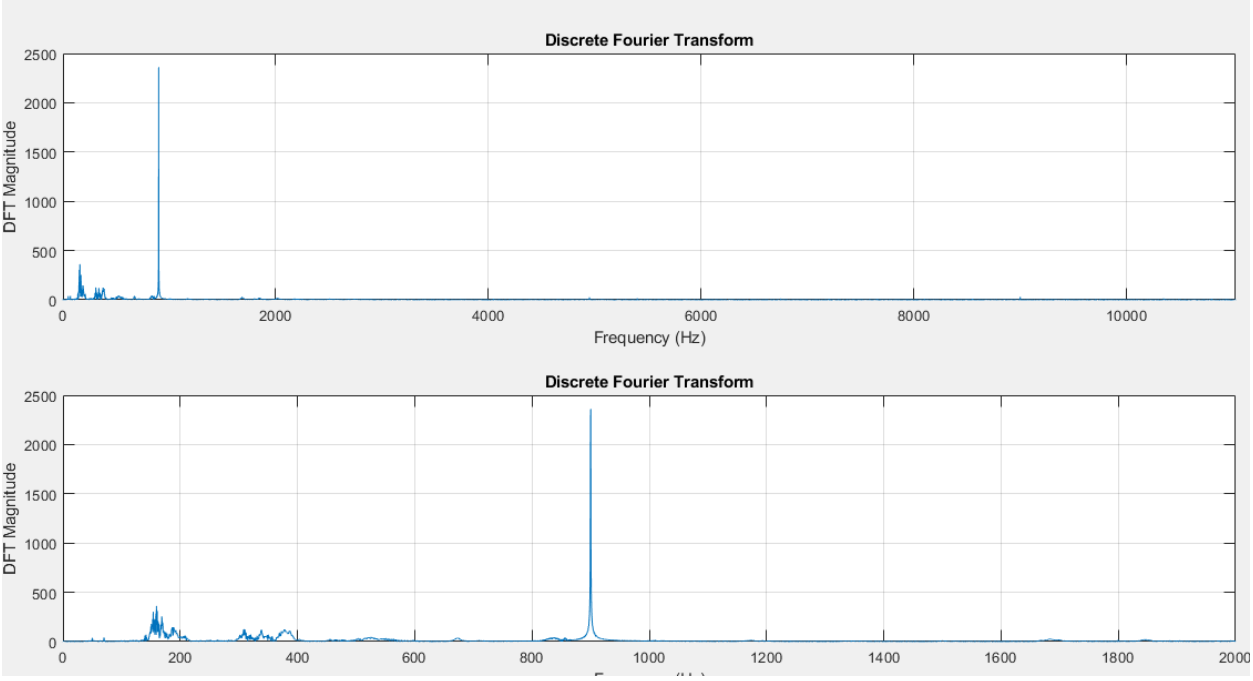
1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Copy the audio file 'noisy.wav' into your current directory.
3. Open M file editor and write the following code:

```
clear all; clc; close all;
[y,Fs] = audioread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\noisy.wav');
Ts = 1/Fs;
n = [0:length(y)-1];
t = n.*Ts;
k = n;
Df = Fs./length(y);
F = k.*Df;
Y = fft(y);
magY = abs(Y);
```

```
sound(y,Fs);
figure,

subplot(2,1,1);
plot(F,magY);
grid on;
xlim([0 Fs/2]);
xlabel('Frequency (Hz)');
ylabel('DFT Magnitude');
title('Discrete Fourier Transform');

subplot(2,1,2);
plot(F,magY);
grid on;

xlim([0 2000]);
xlabel('Frequency (Hz)');
ylabel('DFT Magnitude');
title('Discrete Fourier Transform');
```

4. Save the file as **P081.m** in your current directory and run it.

RESULT:



EXERCISE:

Use recorded data,

1. Plot different frequencies present in it with

a) x-axis as time

b) x-axis as frequency. (Take FFT and plot).

2. Calculate the amount of energy present in fundamental frequency.

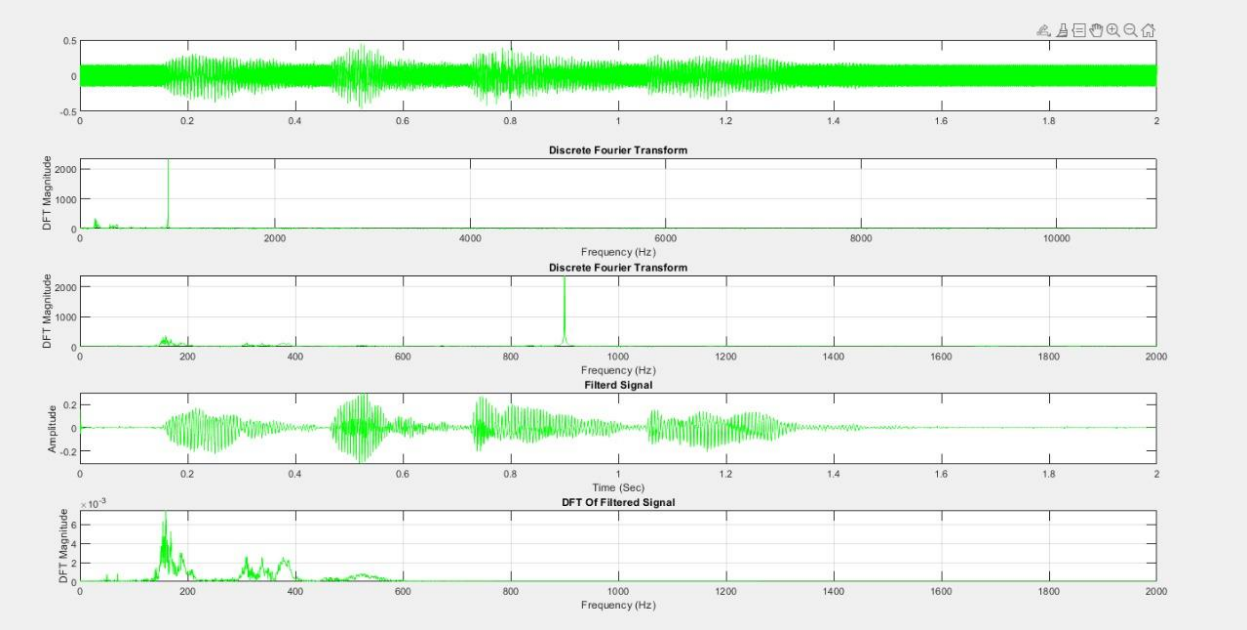
3. Calculate the amount of energy present in different harmonics.

Code:

```
clc,clear,close all
[data,fs]=audioread('C:\Users\Abdul Moeed\OneDrive\Desktop\DSP Lab Codes\noisy.wav');
Ts = 1/fs;
n = (0:length(data)-1);
t = n.*Ts; k = n;
Df = fs/length(data);
F = k.*Df;
Y = fft(data);
magY = abs(Y);
ang = angle(Y);
sound(data,fs);
subplot(5,1,1);
plot(t,data'),xlim([0 2])
subplot(5,1,2);
plot(F,magY);
grid on;
xlim([0 fs/2]);
xlabel('Frequency (Hz)');
ylabel('DFT Magnitude');
title('Discrete Fourier Transform');
subplot(5,1,3);
plot(F,magY);
grid on;
xlim([0 2000]);
xlabel('Frequency (Hz)');
ylabel('DFT Magnitude');
title('Discrete Fourier Transform');
cuttoff_frequency = 600;
orderOfFilter = 11;
```

```
normalizedFrequency=cuttoff_frequency/(fs/2);
[y,x] = butter(orderOfFilter,normalizedFrequency);
filteredSignal = filtfilt(y,x,data);
player = audioplayer(filteredSignal, fs);
pause(2)
play(player);
normalizedFFTOfFilteredSignal = abs(fft(filteredSignal))/length(filteredSignal);
subplot(5,1,4)
plot(t,filteredSignal),xlim([0 2])
xlabel('Time (Sec)');
ylabel('Amplitude');
title('Filterd Signal');
subplot(5,1,5)
plot(F,normalizedFFTOfFilteredSignal),xlim([0 2000]),grid on
xlabel('Frequency (Hz)');
ylabel('DFT Magnitude');
title('DFT Of Filtered Signal');
```

Output:



OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 07

OBJECTIVE:

The purpose of this lab is to become familiar with the practical constraint in calculating the Fourier transform of any real-world signal. i.e., DFT or FFT leakage and its solution.

THEORY:

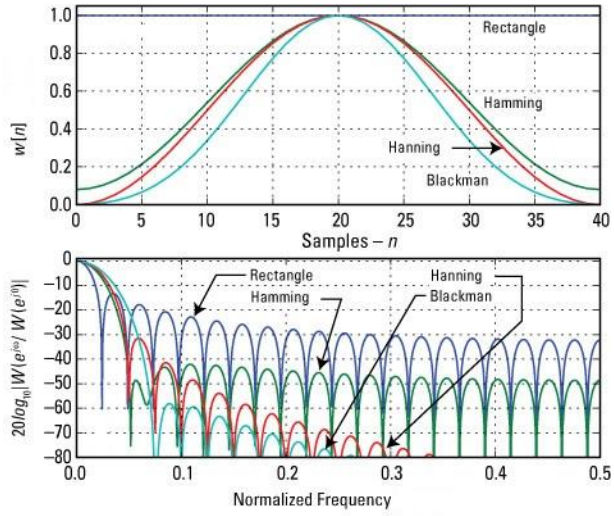
When DFT Leakage will happen?

- If the input signal contains frequency which is ***integer*** multiple of the DFT frequency resolution (Fs/N) or analysis frequencies (k*Fs/N), it will show up only at the correct DFT output bin or frequency. All other DFT output bins will be zero. It means spectrum is correctly estimated/calculated and no DFT leakage occurs in this case. But this is an ideal case to dictate the ingredient components of real-world signal to become integral multiple of DFT frequency resolution or analysis frequencies.
- If the input signal contains any frequency which is ***NOT*** integer multiple of analysis frequencies (which is the most probably happening case), it will show up in all DFT output bins or *leaks out to all DFT output bins*. This effect is simply known as **DFT Leakage/ Spectral Leakage**. We typically say that input signal energy shows up in all of the DFT's output bins. Think of bins as buckets to which you pour energy from infinite amount of possible frequency components. If the signal's frequency is not matched with exact frequency bin, then it smears leaks over rest of the bins.

DFT Leakage and Use of Window functions

Another way to think about DFT leakage is, when the input signal has frequencies which do not complete integer number of cycles in the sample interval ***N* (Length of DFT)**, the input to DFT seems like **abruptly starting** and **abruptly ending** giving rise to side lobes in the DFT output. A selection of finite input samples means that we have multiplied the input sequence by a rectangular window. The DFT output will follow the Sinc shape which is the Fourier Transform of the rectangular function. For real life signals, DFT leakage can never be entirely eliminated because no matter what point DFT you take and no matter what your sampling rate is, there is a very high probability that the input would contain a frequency which is not an integer multiple of analysis frequencies and DFT leakage *would* happen. We can minimize DFT leakage by multiplying the input with a smooth window which will make the input go from zero to maximum and back to zero in a very smooth and slow fashion. This reduces the abruptness of the input which minimizes the higher sides lobes in the DFT output.

Some Window functions and their spectrums



LAB TASKS

1. Condition of DFT/FFT Leakage

Fill the following table specifying whether DFT leakage would happen or not?

Sr. No.	Fs [samples/sec]	N [samples]	Input contains these frequencies in Hz	DFT Leakage Yes, or no?
1	8000	8	1000	No
2	8000	9	1000, 2000	Yes
3	16000	32	250, 500, 1000, 2250	250,2250 yes,500,100 no
4	22050	128	11025	Yes, but theoretically is shouldn't be here
5	44100	4096	1000, 2000, 2500, 11025	Yes
6	44100	44100	1101, 2202, 3303	No
7	96000	48000	1200, 1202, 2002, 2003	2003 yes

2. Calculation of cycles in *N* sample interval (length of DFT/FFT)

   Calculate exactly how many cycles of the 440 Hz wave sampled at 1000 Hz are contained in the 1024-sample window. Explain why this number indicates leads to spectral leakage.

Cycles present will be integral multiple of ΔF till Fs/2 after which alias frequency would start to be plotted and since 440 is not an integral multiple of ΔF this would lead to spectral leakage in frequency plot.

3. DFT or FFT Leakage

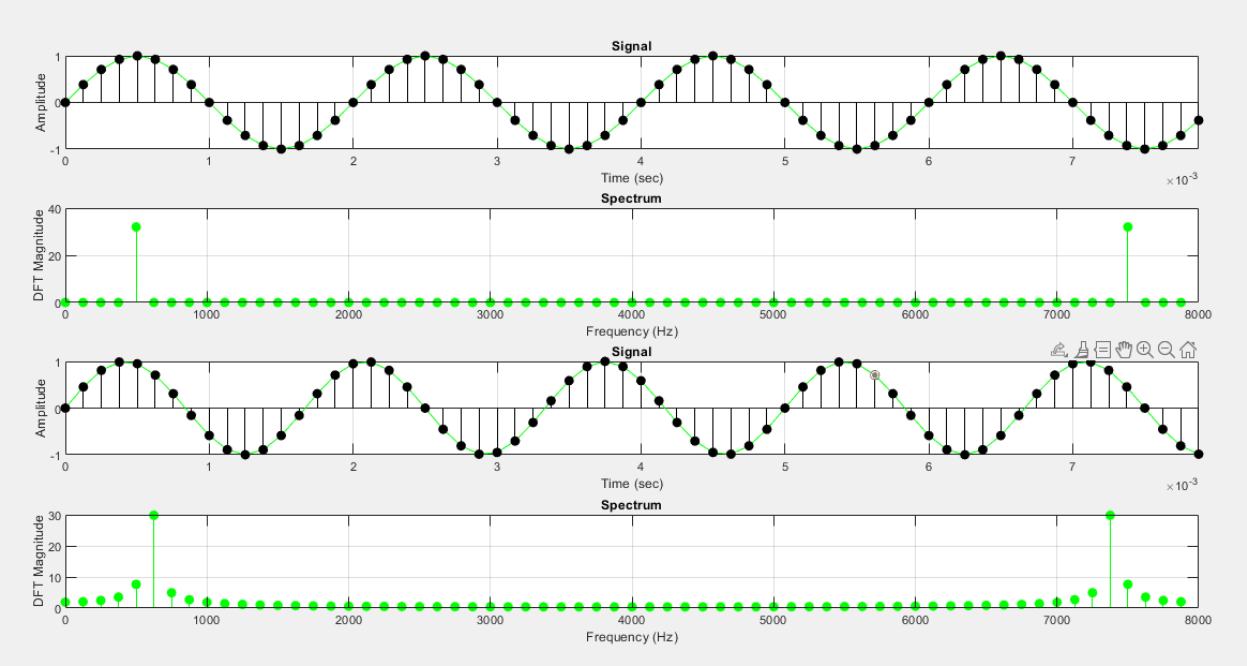
   a) Generate two sinusoids of frequencies 500 Hz and 600 Hz. Both signals should be sampled at 8000 samples/sec.
   b) Take 64-point DFT of both signals and observe their spectrums.
   c) Comment on the DFT Leakage for both cases.

PROCEDURE:

```
clear all; close all; clc;

F1 = 500;
F2 = 600;
Fs = 8000;
Ts = 1/Fs;
N = 64;
n =[0:N-1];
k = n;
Df = Fs/N;
Fk = k.*Df;

x1 = sin(2*pi*F1*n*Ts);
X1 = abs(fft(x1,N));
x2 = sin(2*pi*F2*n*Ts);
X2 = abs(fft(x2,N));

subplot(4,1,1)
plot(n*Ts,x1,'r')
hold
stem(n*Ts,x1,'filled')
xlabel('Time (sec)')
ylabel('Amplitude')
title('Signal')
grid
axis tight

subplot(4,1,2)
```

```
stem(Fk,X1,'r','filled')
xlabel('Frequency (Hz)')
ylabel('DFT Magnitude')
title('Spectrum')
xlim([0 Fs])
grid

subplot(4,1,3)
plot(n*Ts,x2,'r')
hold
stem(n*Ts,x2,'filled')
xlabel('Time (sec)')
ylabel('Amplitude')
title('Signal')
grid
axis tight

subplot(4,1,4)
stem(Fk,X2,'r','filled')
xlabel('Frequency (Hz)')
ylabel('DFT Magnitude')
title('Spectrum')
xlim([0 Fs])
grid
```

Output:



Here ΔF is equal to 8000/64 = 125Hz. Since 500 is an integral multiple of 125 therefore leakage is nowhere to be found in this spectrum but 600 is not an integral multiple of 125 therefore leakage is present in its spectrum.

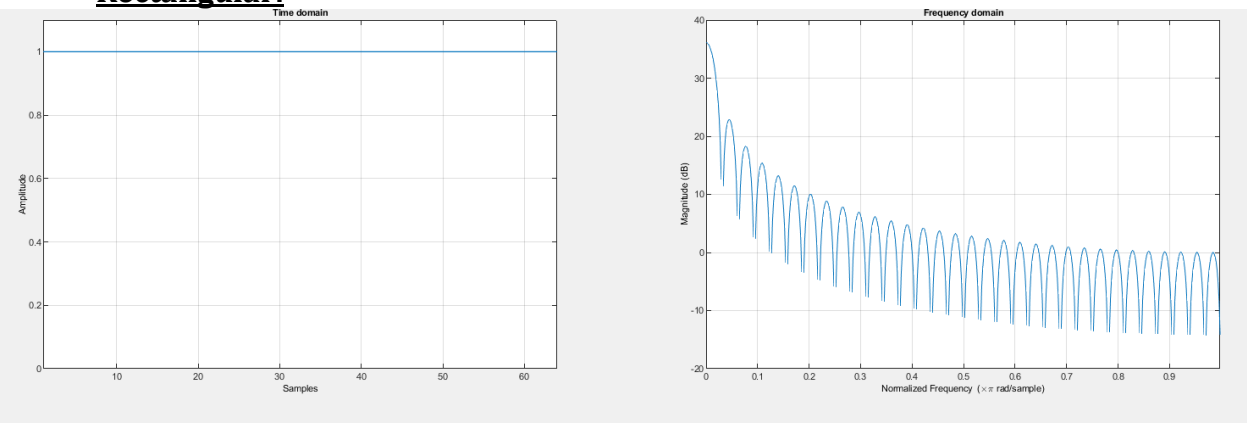
4. Evaluation of Various Window functions using wintool of MATLAB or Window Analysis & Designing

   Type wintool (Window Design & Analysis Tool) in the command window of MATLAB. Observe different window functions like **Rectangular (no window), Triangular, Hamming, Hanning, Blackman, Flat Top, Gaussian** etc. Attach the time domain and frequency domain representations of various window functions and make a comparison between their performances on the basis of following attributes:

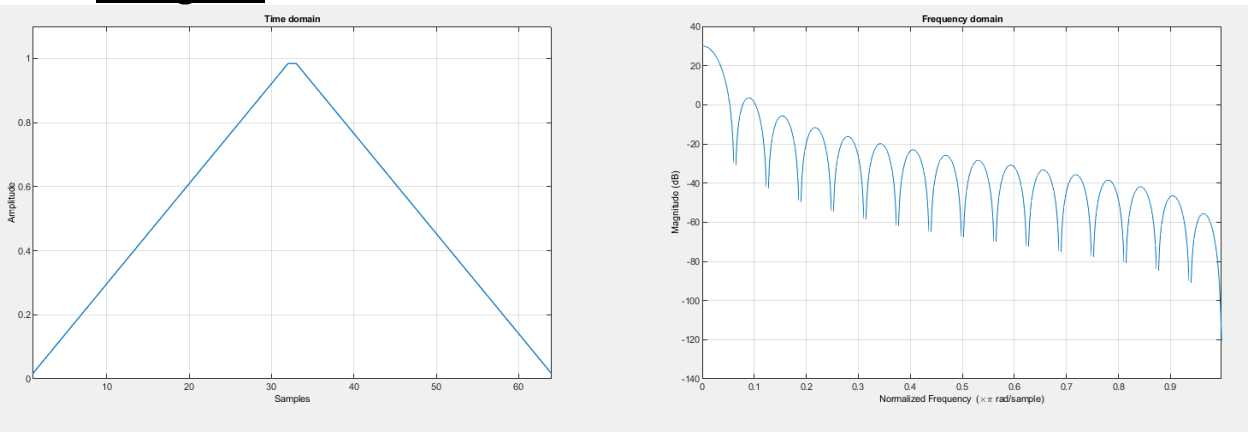
   a) Leakage Factor in %
   b) Width of the main lobe in fraction of π ($rad/sample$) or $F_s/2$ ($samples/\sec or\ Hz$)
   c) The first (highest)side lobe magnitude in dB
   d) The rate of side lobes falloff **in** dB/octave

S.no.	**Window name**	**Leakage Factor**	**Main Lobe Width**	**First Side Lobe Magnitude**	**Rate of side lobes falloff**
1	Rectangular	9.14%	0.027344	-22.86713	-13.3
2	Triangular	0.28%	0.039063	-3.546566	-26.6
3	Hamming	0.03%	0.039063	-16.34576	-42.5
4	Hann	0.05%	0.042969	-1.529982	-31.5
5	Blackman	0%	0.050781	-29.68592	-58.1
6	Flat Top	0%	0.11719	-67.28888	-88
7	Gaussian	0.01%	0.042969	-1.56708	-44

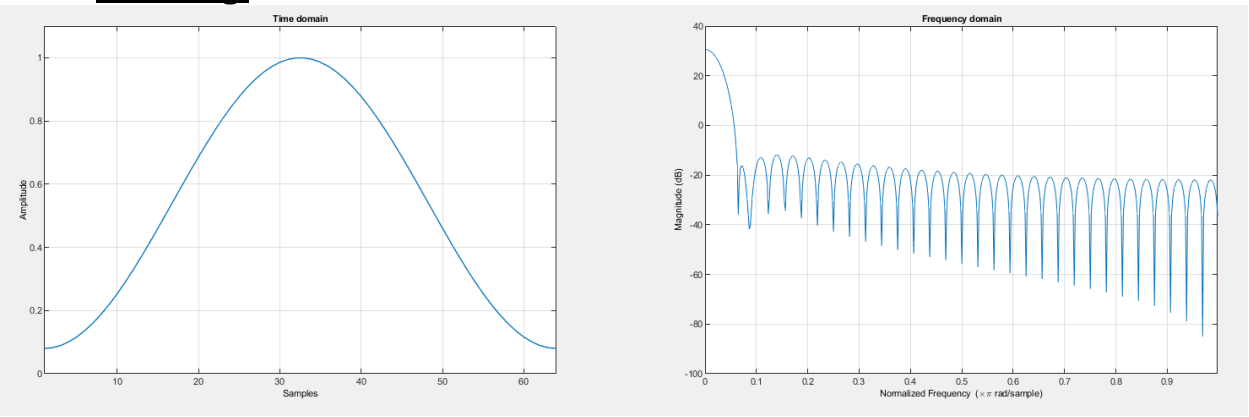
Rectangular:



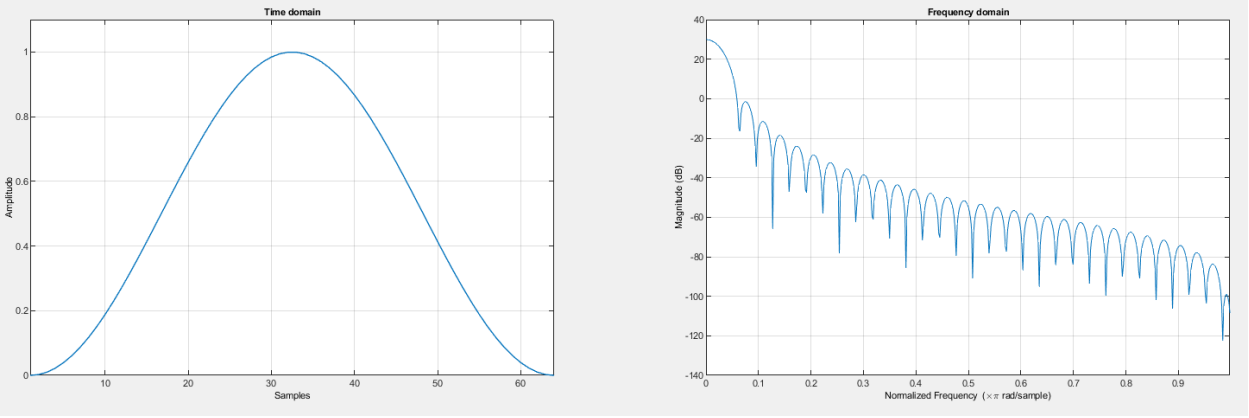
Triangular:



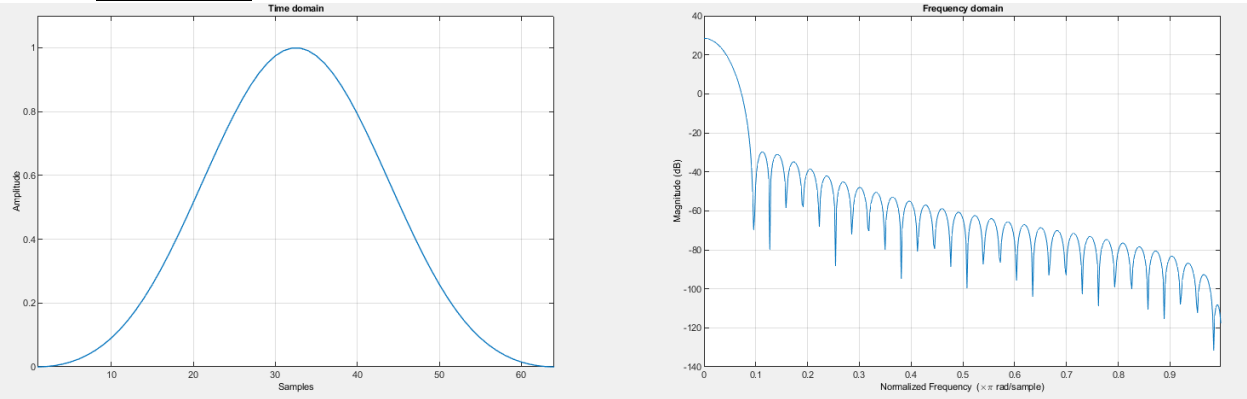
Hamming:



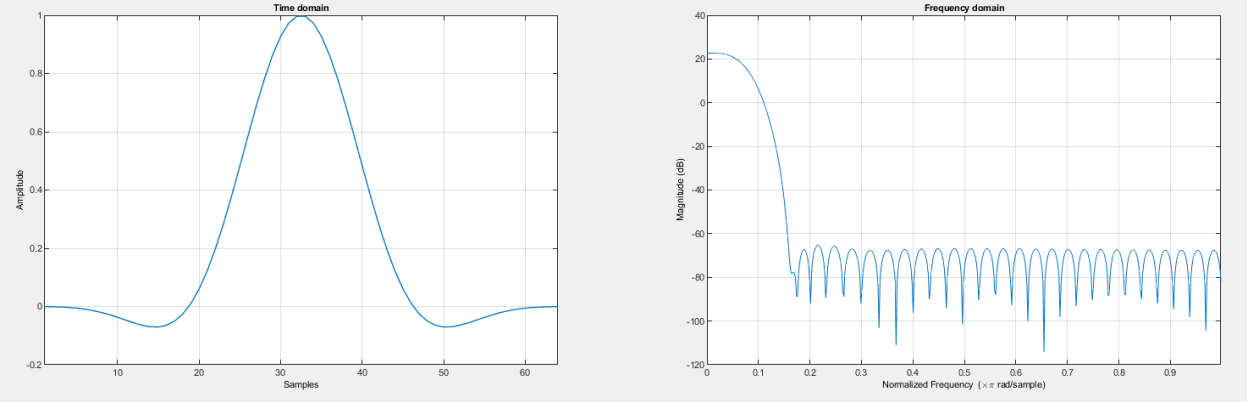
Hann:



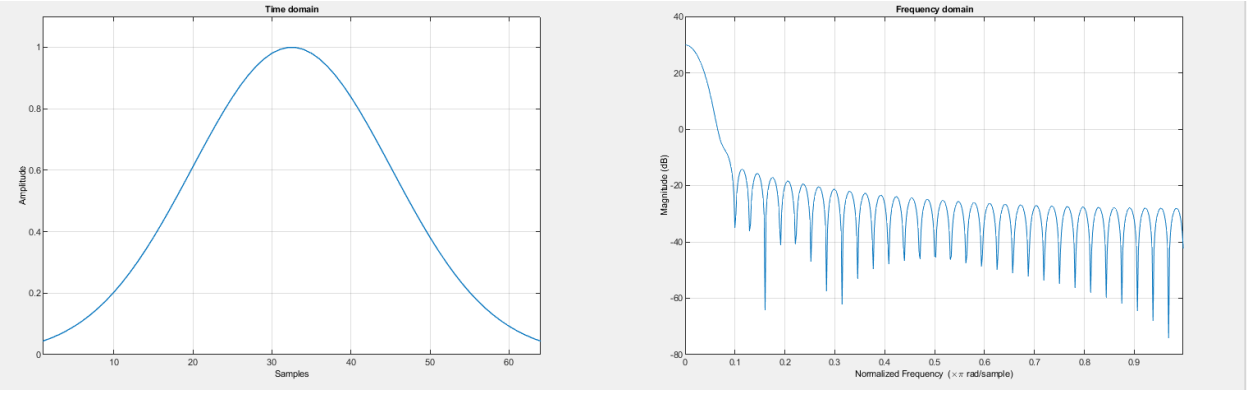
Blackman:



Flat Top:



Guassian:



5. Spectrum of windowed signal

   a) Generate a sinusoid of frequency 1050 Hz sampled at 8000 samples/sec. Initial phase is $\pi/4$.
   b) Take 64-point DFT of the generated signal and observe the spectrum.
   c) Comment on the DFT Leakage phenomenon for the above case.
   d) Generate a triangular window function of 64 points.
   e) Now apply a triangular window to the signal and then observe the spectrum of windowed signal.
   f) Change the window function from **Triangular** to **Hamming, Hanning, Blackman, Flat Top** and **Gaussian**. Compare the results of these window functions on a signal's spectrum.

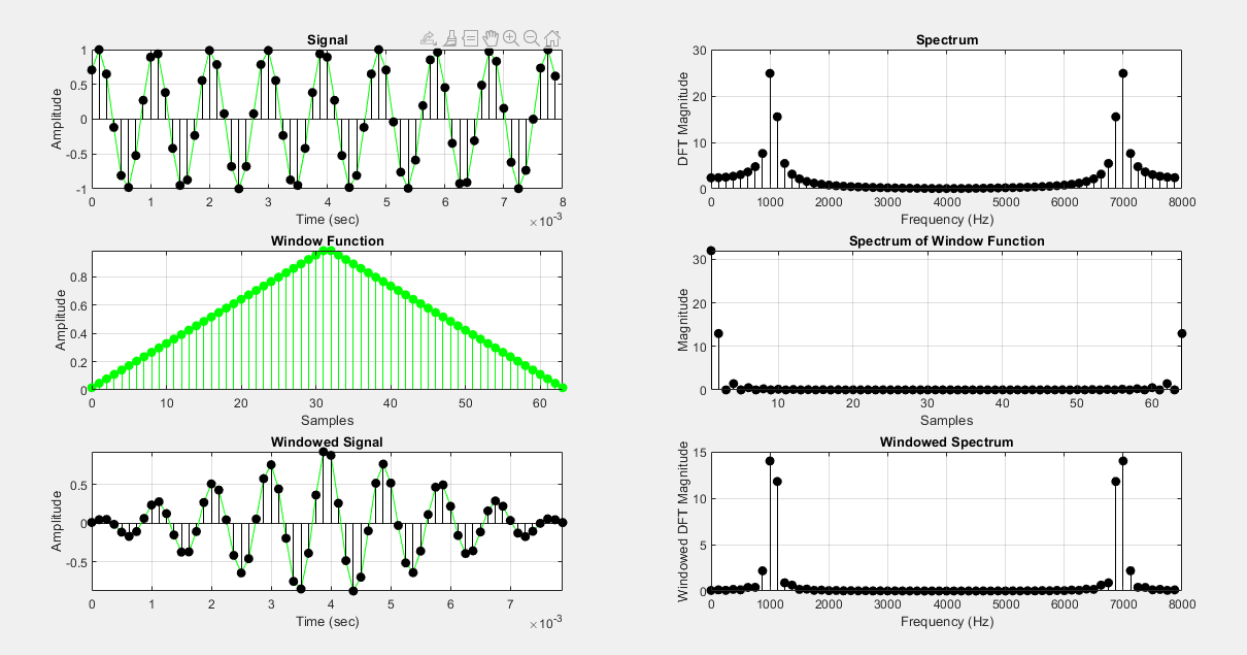
PROCEDURE:

```
clear all;close all;clc;
F = 1050;
Fs = 8000;
Ts = 1/Fs;
N = 64;
n = [0:N-1];
k = n;
Df = Fs/N;
Fk = k.*Df;
x = sin(2*pi*F*n*Ts+pi/4);
w = window(@triang,N);
xw = x.*w';  % Windowed signal
X  = abs(fft(x));
Xw = abs(fft(xw));

subplot(321)
plot(n*Ts,x,'g')
hold
stem(n*Ts,x,'filled','k')
xlabel('Time (sec)')
ylabel('Amplitude')
title('Signal')
grid

subplot(322)
stem(Fk,X,'filled','k')
xlabel('Frequency (Hz)')
ylabel('DFT Magnitude')
title('Spectrum')
grid
subplot(323)
stem(n,w,'g','filled')
```

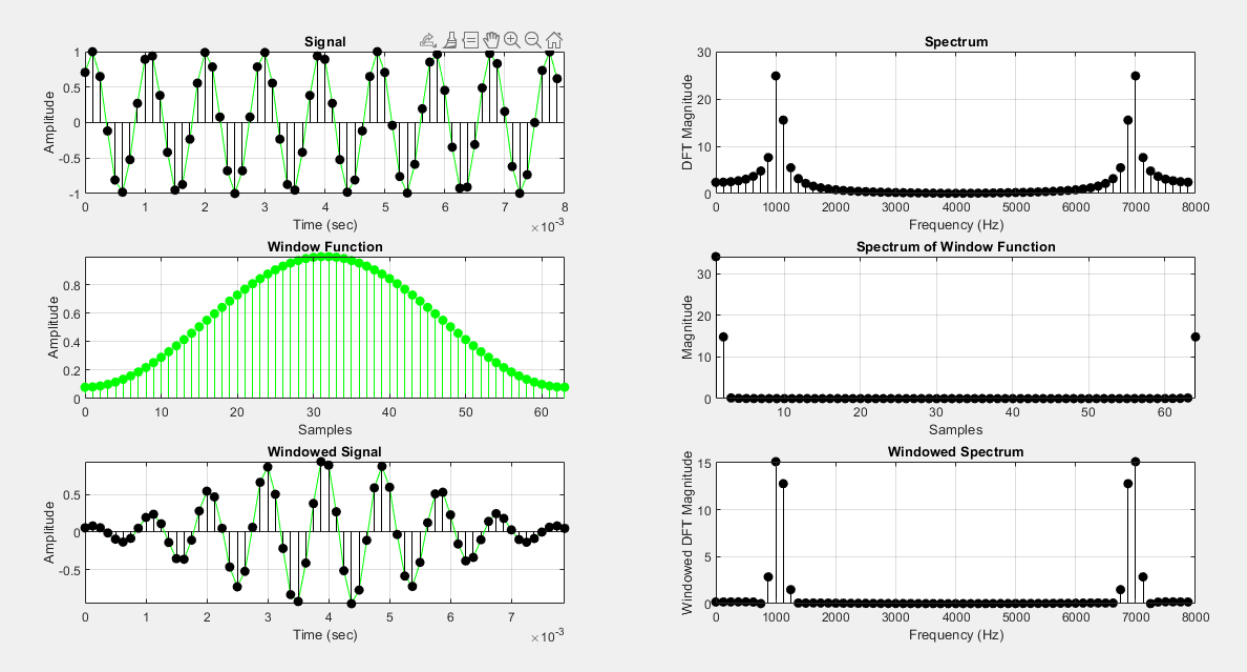
```
xlabel('Samples')
ylabel('Amplitude')
title('Window Function')
grid
axis tight
subplot(324)
stem(abs(fft(w)),'k','filled')
xlabel('Samples')
ylabel('Magnitude')
title('Spectrum of Window Function')
grid
axis tight
subplot(325)
plot(n*Ts,xw,'g')
hold
stem(n*Ts,xw,'filled','k')
xlabel('Time (sec)')
ylabel('Amplitude')
title('Windowed Signal')
grid
axis tight
subplot(326)
stem(Fk,Xw,'filled','k')
xlabel('Frequency (Hz)')
ylabel('Windowed DFT Magnitude')
title('Windowed Spectrum')
grid
```

Output:

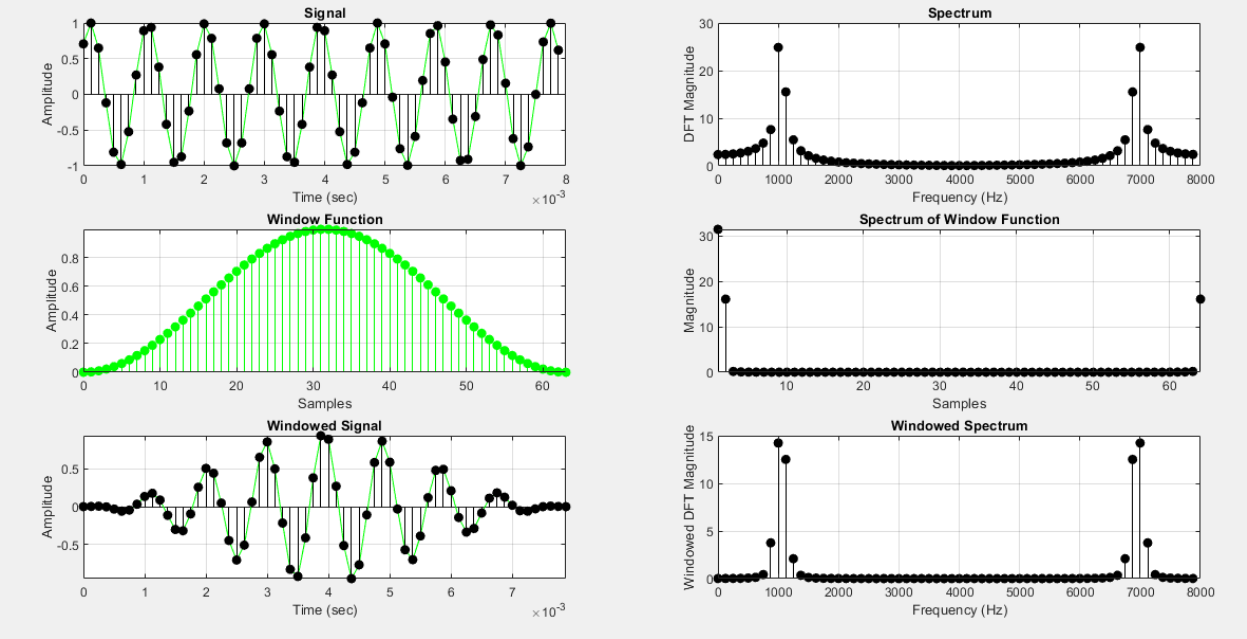
1) Triangular



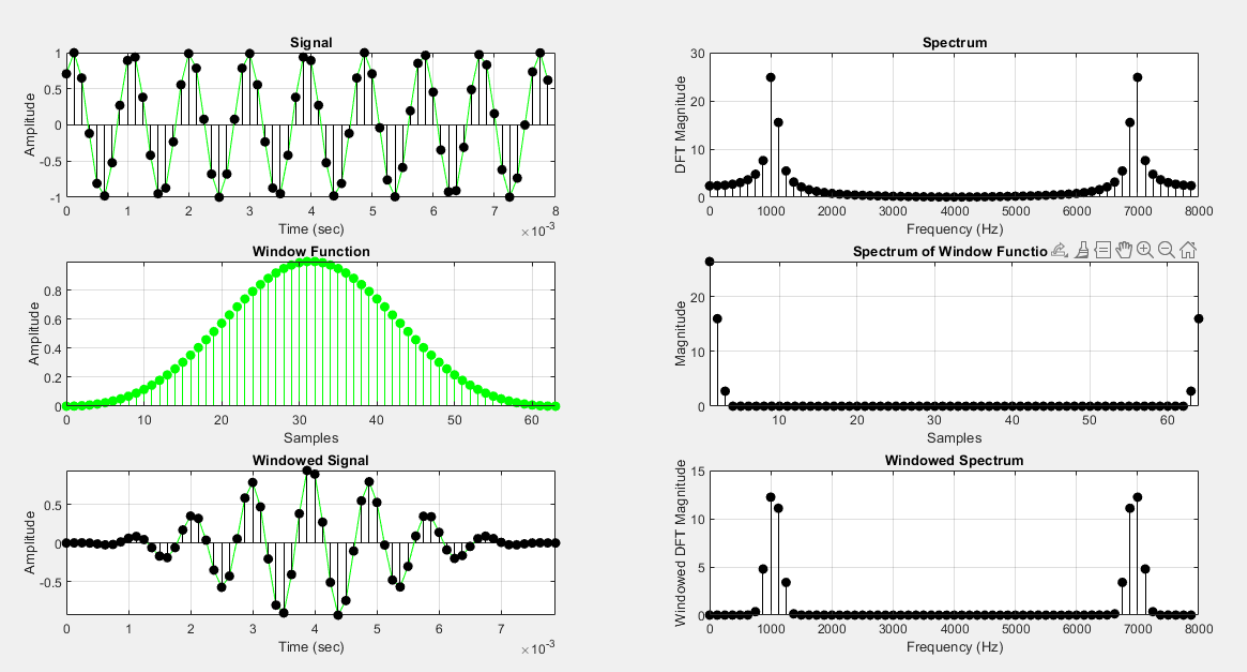
2) Hanning



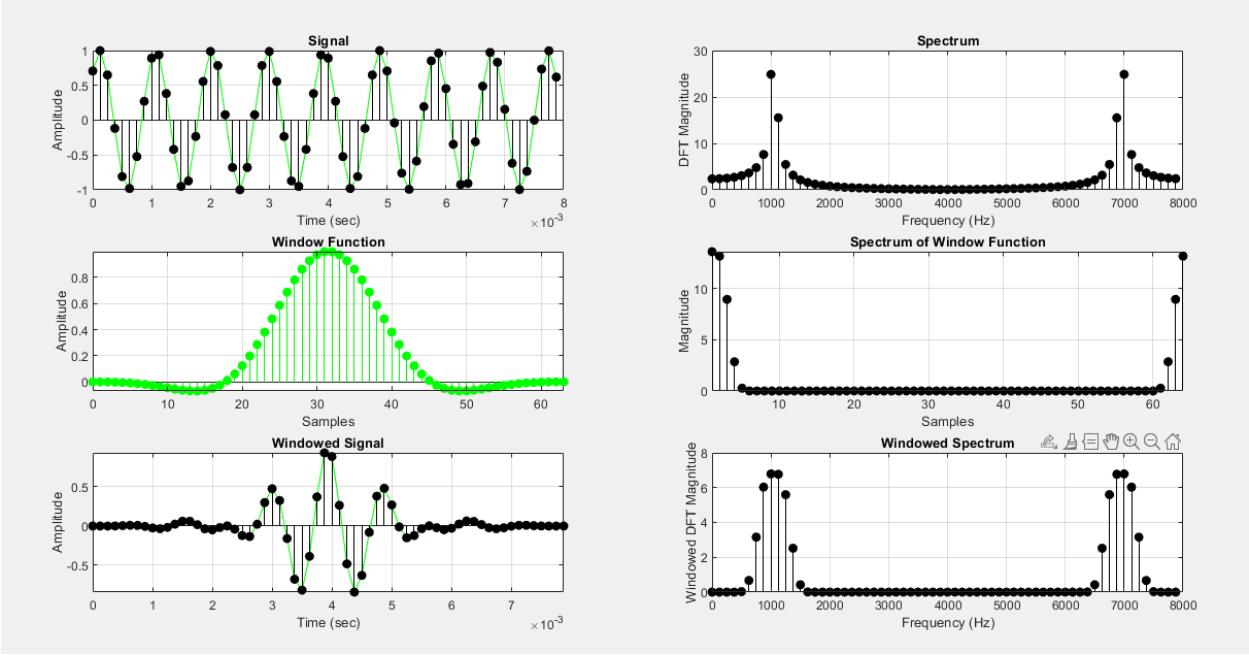
3) Hann



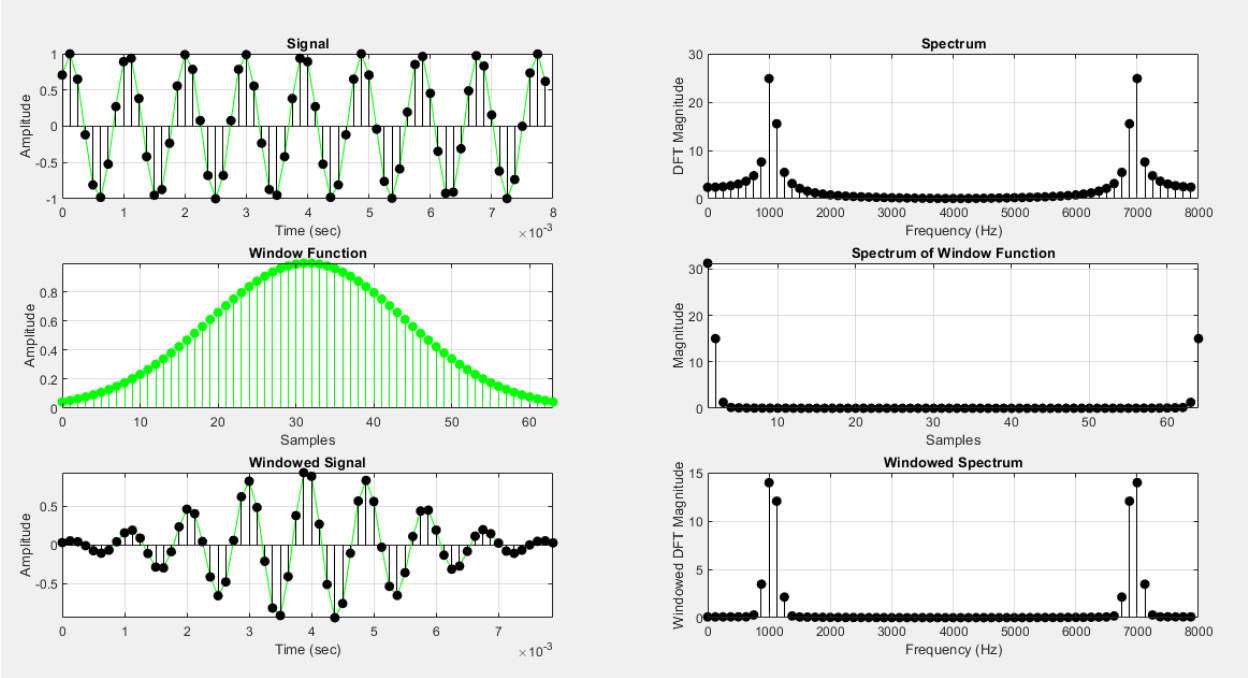
4) Blackmann



5) Flat Top



6) Gaussian


Signal
Amplitude
Time (sec)
×10^-3
Spectrum
DFT Magnitude
Frequency (Hz)
Window Function
Amplitude
Samples
Spectrum of Window Function
Magnitude
Samples
Windowed Signal
Amplitude
Time (sec)
×10^-3
Windowed Spectrum
Windowed DFT Magnitude
Frequency (Hz)

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 08

OBJECTIVE:

To study s-plane and plot impulse and frequency response for different pole zero location in s-plane. Also, to determine weather system is FIR or IIR.

THEORY:

The Laplace Transform of a general continuous time signal x (t) is defined as;

$$\boldsymbol{X(S) = \int x(t)\, e^{-st}\, dt}$$

Where the complex variable s=δ+ j w, with δ and w the real and imaginary parts. CTFT is a subset of Laplace when δ =0. Since 'δ' information is not present in CTFT, therefore information about stability can only be obtained from Laplace. If pole lies on L.H.S of s-plane, system is stable. If pole lies on R.H.S of s-plane, system is unstable. If pole lies on y(jw)-axis, system is marginally stable or oscillatory. If system has FIR, it is stable. If system is IIR, it can be stable or unstable.

PROCEDURE:

Generate pole zero constellation in s plane.

1. Plot corresponding Frequency (Bode magnitude) response.
2. Plot impulse response and determine that the system is FIR or IIR.
3. Modify location of poles in s plane to observe the corresponding change in frequency and impulse response.

STEPS.

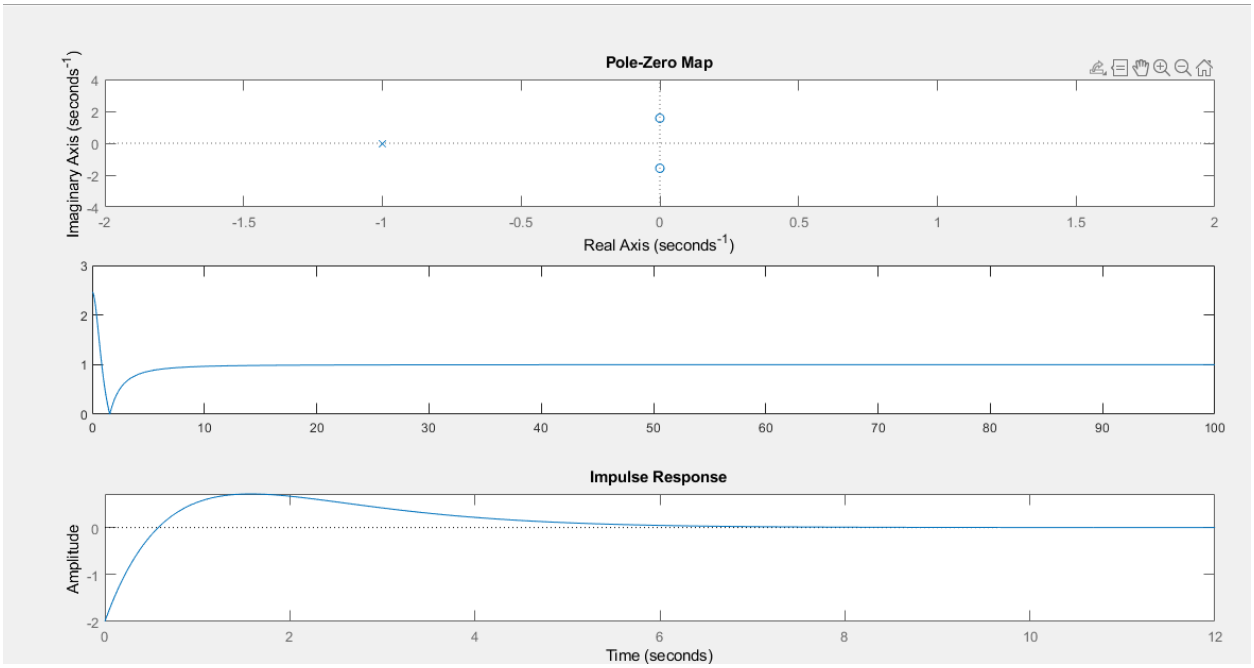
1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Open M-file editor and write the following code:

```
clc,clear,close all

Num = poly([(0-(1i*(pi/2))),(0+(1i*(pi/2)))]);

Zeros=roots(Num);

Den = poly([-1,-1]);

poles=roots(Den);

sys=tf(Num,Den);

figure;

subplot(3,1,1);
```

```
pzmap(sys);
xlim([-2 2]);
ylim([-4 4]);
subplot(3,1,2);
[mag,phase,w]=bode(sys);
mag=squeeze(mag);
plot(w,mag);
subplot(3,1,3);
impulse(sys);
H=dfilt.df1(Num,Den);
A=isfir(H);
disp(A)3. Save the file as P091.m in your current directory and 'run' it.
```

RESULT:

1. Learn the specific logical bits of the code and make notes.
2. Observe the plots.
3. Now, explain (write) in your own words the cause and effects of what you just saw.



System is IIR

EXERCISE:

1. Change the location of poles from L.H.S of s-plane to y axis first, and then to R.H.S of s-plane and observe the effects.

2. Analysis of 1st order Analog system [one pole at origin]

   a) Generate pole zero constellation of an analog system in s-plane having only one pole at s = 0.

   b) Write the transfer function of a system.

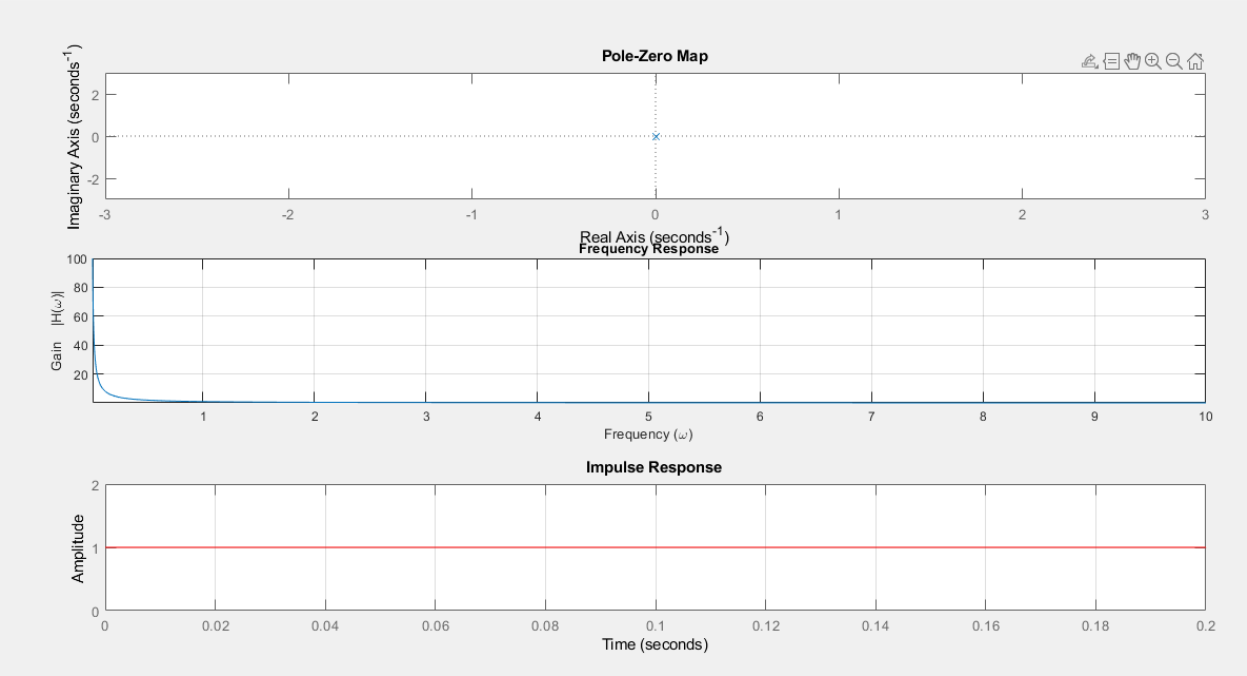
   c) Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.

PROCEDURE:

a) Make a folder at desktop and name it as your current directory within MATLAB.
b) Open M-file editor and write the following code:

```
clear all; close all; clc;

Num = [1];

Den = poly([0]); % At origin
```

```
sys=tf(Num,Den)

zeros=roots(Num)

poles=roots(Den)

figure;

subplot(3,1,1)

pzmap(sys)

xlim([-3 3])

ylim([-3 3])
subplot(3,1,2)

[H w]=freqs(Num,Den);

mag=abs(H);

plot(w,mag)

xlabel('Frequency (\omega)')

ylabel('Gain    |H(\omega)|')

title('Frequency Response')

axis tight

grid

subplot(3,1,3)

impulse(sys,'r')

grid
```

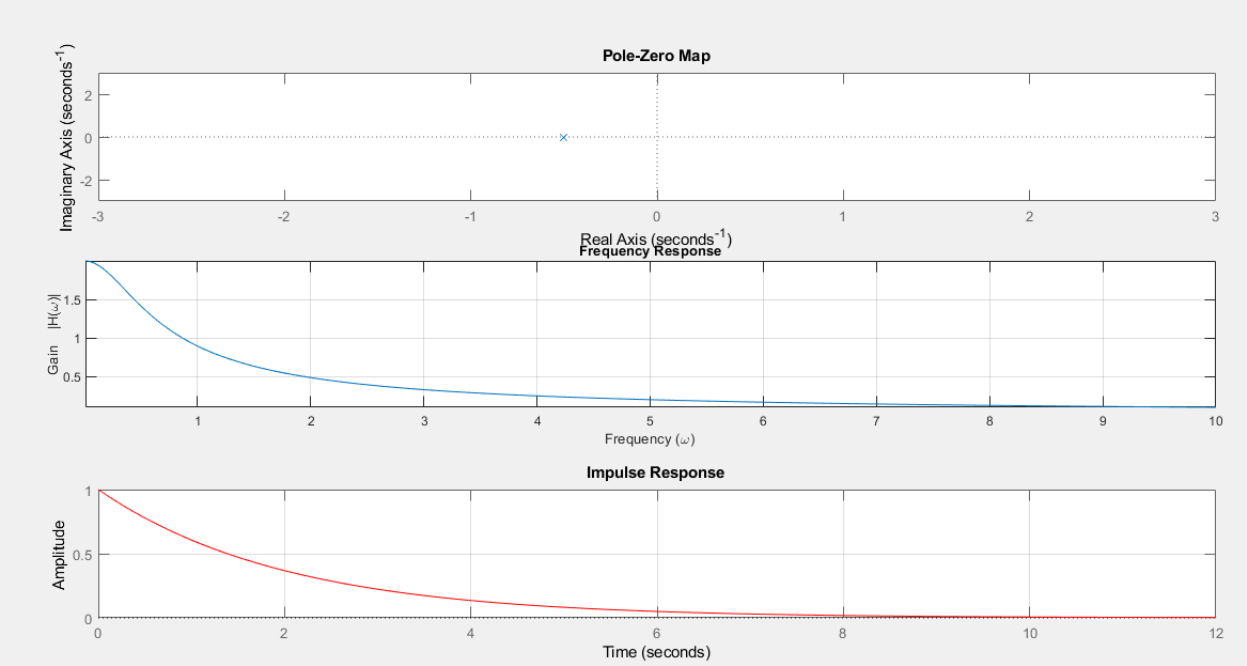


Transfer Function: $H(S)=\frac{1}{S}$

The system is marginally stable.

3. Analysis of 1st order Analog system [one pole at LHP]

   a) Generate pole zero constellation of an analog system in s-plane having only one pole at: s = -0.5
   b) Write the transfer function of a system
   c) Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.

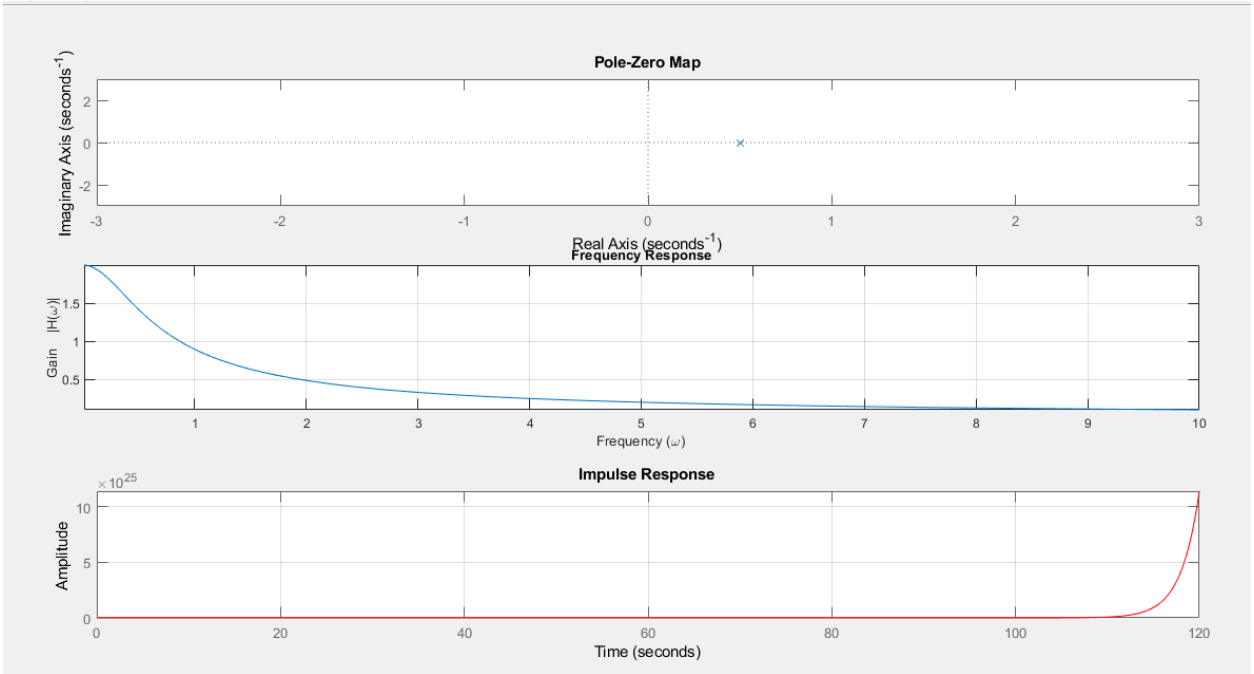


Transfer Function: $H(S)=\frac{1}{s+0.5}$

The system is Stable.

4. Analysis of 1st order Analog system [one pole at RHP]

a) Generate pole zero constellation of an analog system in s-plane having only one pole at: s =0.5
b) Write the transfer function of a system
c) Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.

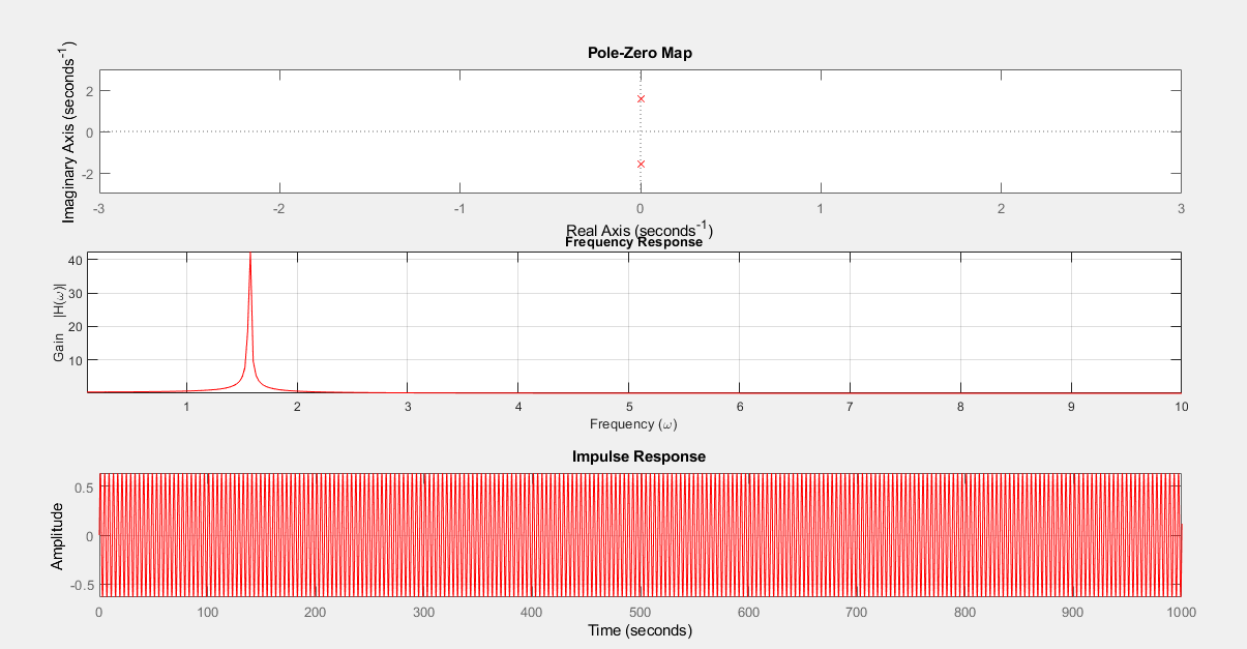


Transfer Function: $H(S)=\frac{1}{S-0.5}$

The system is Stable.

Analysis of 2nd order Analog system [poles at jω axis]

a) Generate pole zero constellation of an analog system in s-plane having pure imaginary poles at: s= $j\omega = \pm j\,\pi/2$
a. Write the transfer function of a system
b. Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.



Transfer Function: H(S)=$\frac{1}{s^2+2.467}$

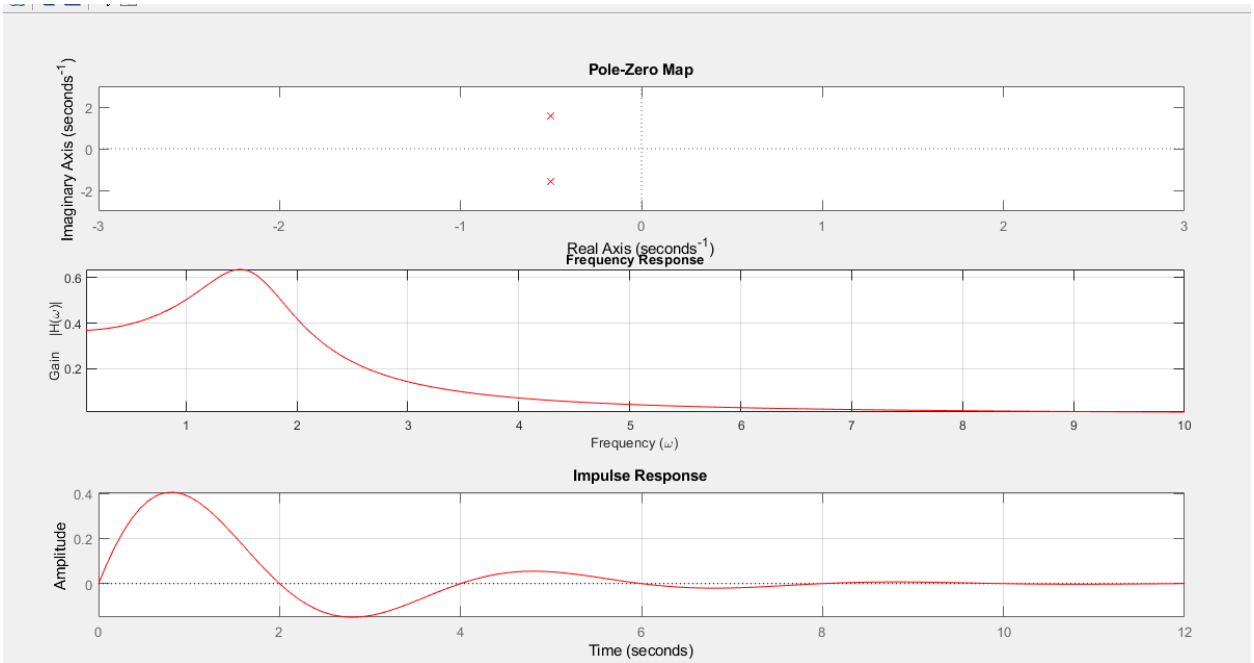
The system is oscillating and is neither decaying nor rising.

5. Analysis of 2nd order Analog system [complex poles at LHP

a) Generate pole zero constellation of an analog system in s-plane having complex conjugate poles at: $s = -0.5 \pm j\,\pi/2$

b) Write the transfer function of a system

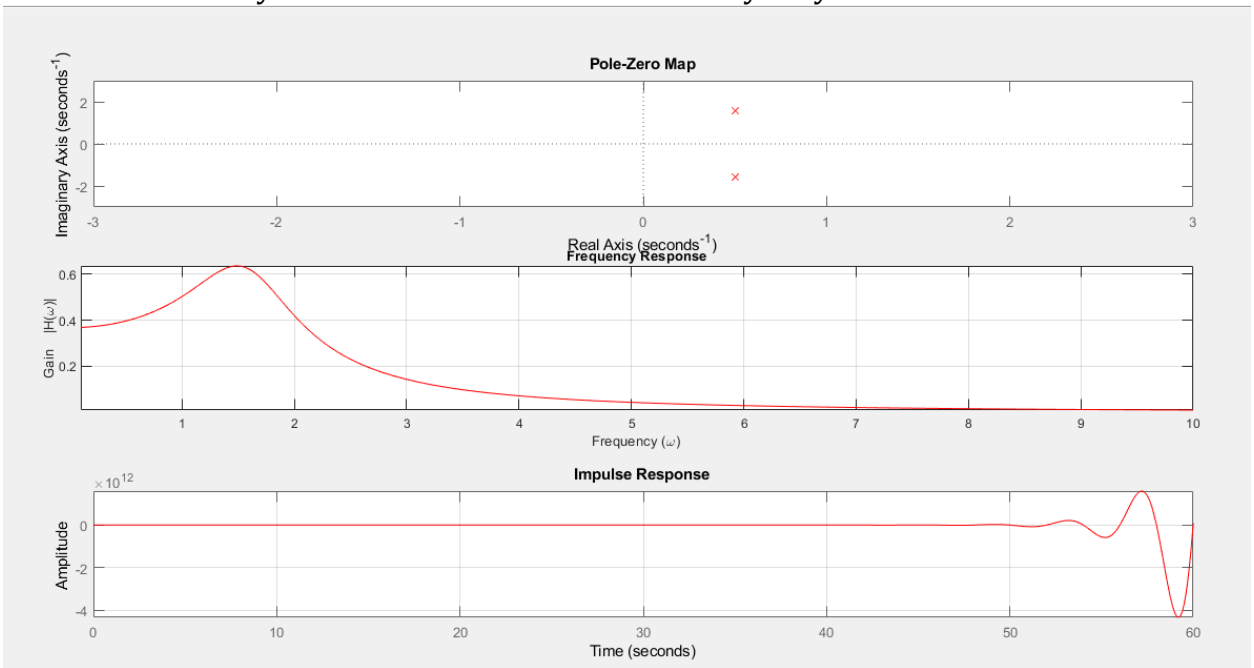
c) Plot the corresponding frequency response and impulse response of a system Also comment on the stability of system.



Transfer Function: H(S)= $\frac{1}{s^2+s+2.717}$

The system is oscillating and decaying

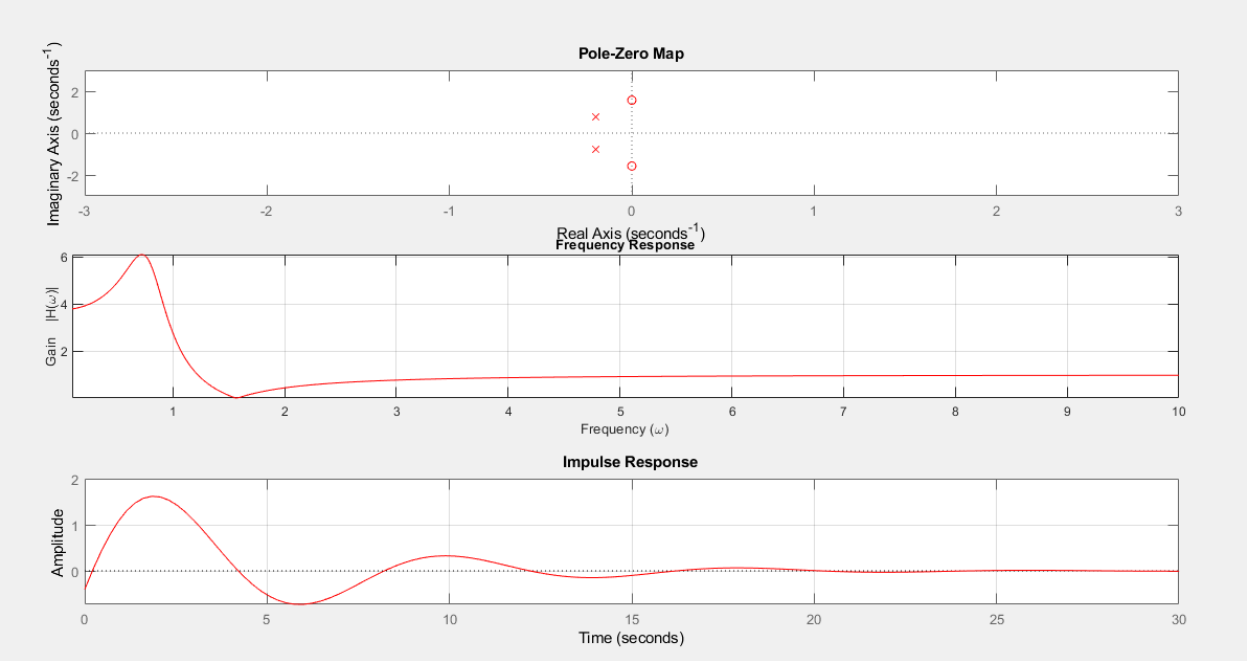
6. Analysis of 2nd order Analog system [complex poles at RHP]
   a) Generate pole zero constellation of an analog system in s-plane having complex conjugate poles at: $s = +0.5 \pm j\,\pi/2$
   b) Write the transfer function of a system
   c) Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.



Transfer Function: H(S)=$\frac{1}{s^2-s+2.717}$

The system is rising and oscillatory.

7. Analysis of 2nd order Analog system [complex zeros and poles at LHP]
   a) Generate pole zero constellation of an analog system in s-plane for given roots. [zeros: $s = \pm j(\pi/2)$ & poles: $s = -0.2 \pm j(\pi/4)$]
   b) Write the transfer function of a system
   c) Plot the corresponding frequency response and impulse response of a system. Also comment on the stability of system.

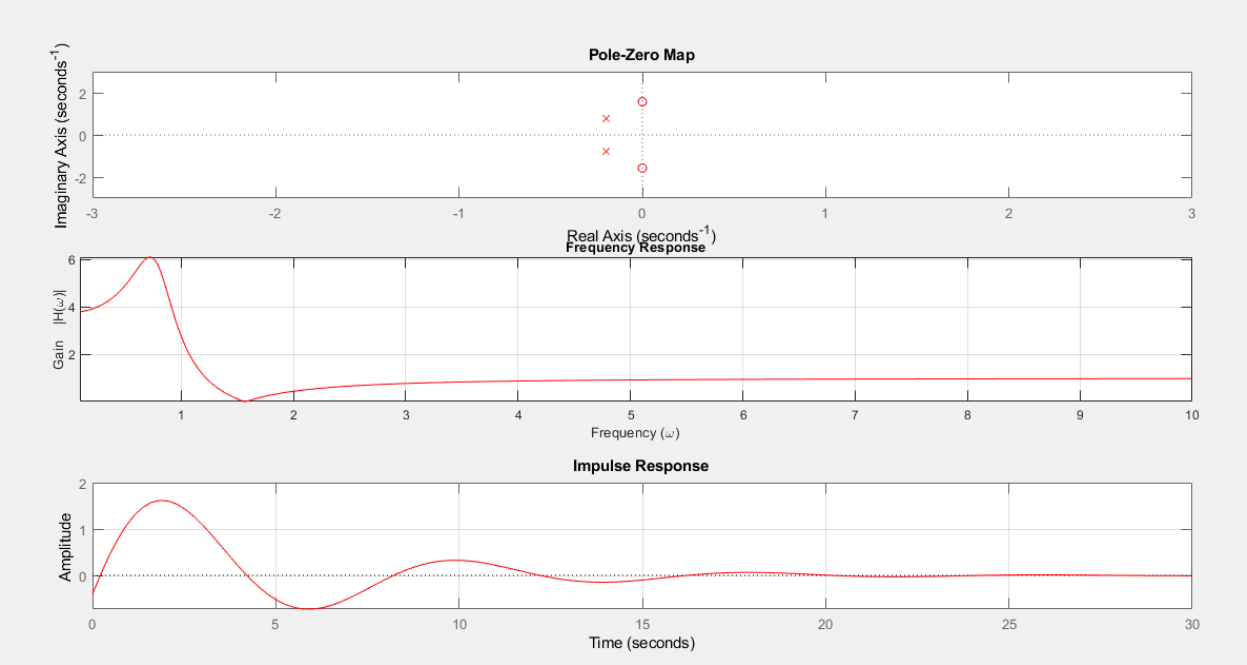


Transfer Function: H(S)=$\frac{1}{s^2-s+2.717}$

The system is decaying or oscillatory.

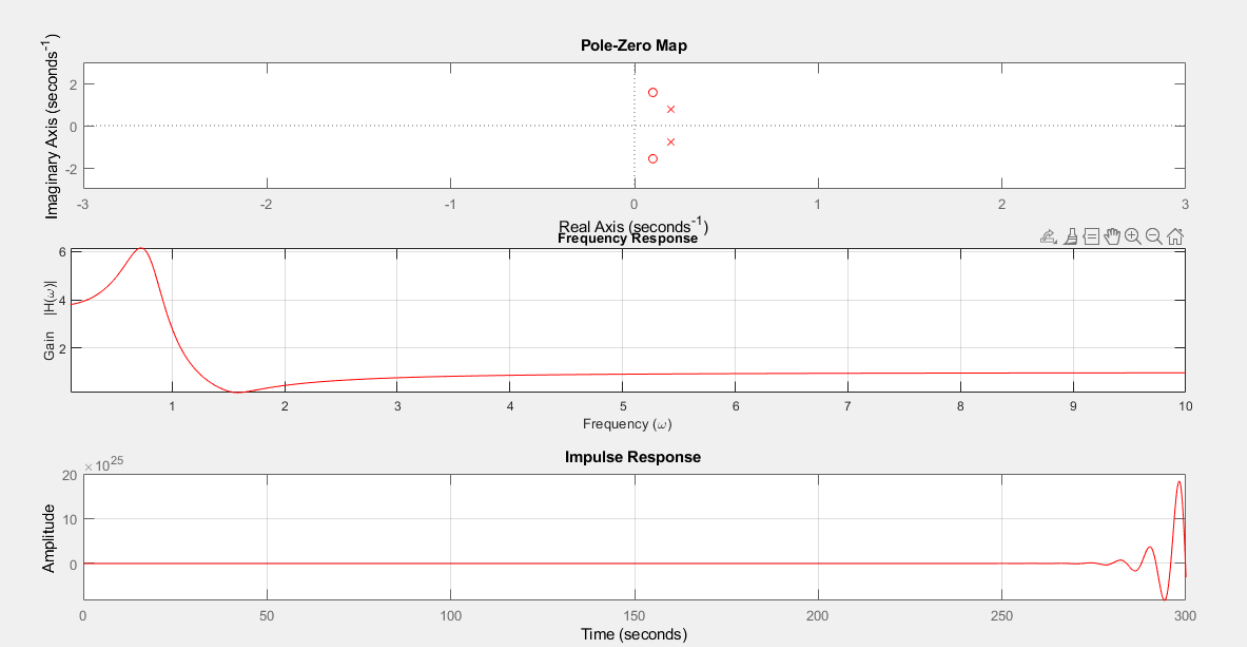
8. Effect of system's poles on system stability

   Change the location of poles of a system defined in Task-7 from L.H.S of s-plane to $j\omega$ axis first, and then to R.H.S of s-plane and observe the effects on impulse response and frequency response of a system.



Transfer Function: H(S)=$\frac{s^2 + 2.467}{s^2 + 0.4\,s + 0.6569}$

9. Effect of system's zeros on system stability
Modify the location of zeros of a system defined in Task-7 from $s = \pm j(\pi/2)$ to $s = -0.1 \pm j(\pi/2)$, and then to $s = 0.1 \pm j(\pi/2)$. Do not change the location of poles. Does the impulse response change? Can we place analog system's zeros on the right-hand plane [RHP]?



Transfer Function: H(S)=$\frac{s^2 + 2.467}{s^2 - 0.4\,s + 0.6569}$

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 09

OBJECTIVE:

To study z-plane and plot impulse and frequency response for different pole zero location in z-plane. Also, to determine weather system is FIR or IIR.

THEORY:

The z - Transform of a general discrete time signal x(n) is defined as;

$$X(z) = \sum_{n=o}^{\infty} x(n)\ z^{-n}$$

Where the complex variable z=r $\angle w$, with r the radius and w the angle. DTFT is a subset of z transform when r =1. Since 'r' information is not present in DTFT, therefore information about stability in discrete time can only be obtained from z transform. If pole lies inside the unit circle, system is stable. If pole lies outside the unit circle, system is unstable. If pole lies at the unit circle, system is marginally stable or oscillatory. If system has FIR, it is stable. If system is IIR, it can be stable or unstable.

PROCEDURE:

1. Generate pole zero constellation in z plane.
2. Plot corresponding Frequency (Bode magnitude) response.
3. Plot impulse response and determine that the system is FIR or IIR.
4. Modify location of poles in z plane to observe the corresponding change in frequency and
   impulse response.

STEPS:

1. Make a folder at desktop and name it as your current directory within MATLAB.
2. Open M-file editor and write the following code:

```
clear all;
close all;
clc;
Num = poly([(0-(i*(pi/2))),(0+(i*(pi/2)))]);
```

```
Den = poly([-1,-1]);
Num1 = poly([j,-j]);
Den1 = poly([exp(-1),exp(-1)]);
sys1=tf(Num1,Den1,1)

figure;
subplot(3,1,1);
pzmap(sys1);
xlim([-2 2]);
ylim([-4 4]);
subplot(3,1,2);
[mag phase w]=bode(sys1);

mag=squeeze(mag);
plot(w,mag);
xlim([0 100])
subplot(3,1,3);
impulse(sys1);
H=dfilt.df1(Num,Den);
A=isfir(H)
figure;
pzmap(sys1)
grid on;
```

3. Save the file as **P010.m** in your current directory and 'run' it.

RESULT:

1 Learn the specific logical bits of the code and make notes.
2 Observe the plots.
3 Now, explain (write) in your own words the cause and effects of what you just saw.

EXERCISE:

1. Change the location of poles from inside the unit circle to outside and at the unit circle and observe and note the changes.

2. Analysis of 1st order Digital system [one pole inside the Unit circle at DC]

   a) Generate pole zero constellation in z-plane for the following roots of a digital system:

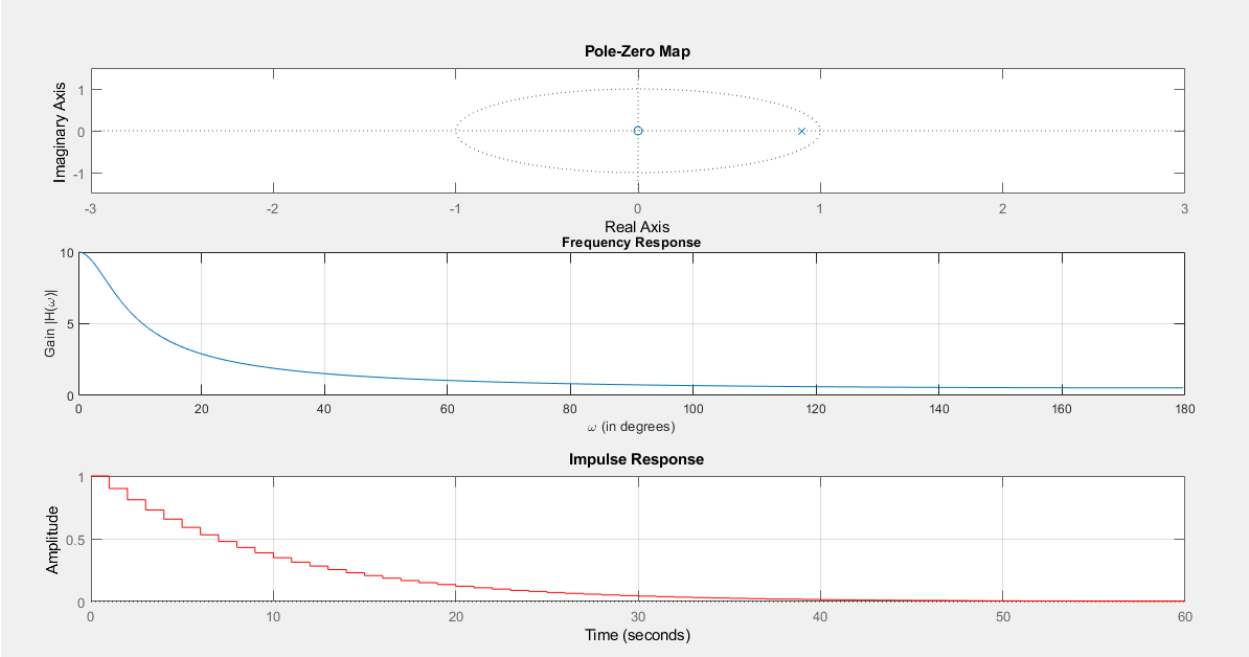
$$[\text{zero: } z = 0 \ \& \ \text{pole: } z = 0.9\angle 0]$$

b) Write the transfer function of a system [see command window]
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system

PROCEDURE:

a) Make a folder at desktop and name it as your current directory within MATLAB.
b) Open M-file editor and write the following code:

```
clear all;close all;clc;

Num = poly([0]);

Den = poly([0.9*exp(j*0)]);

sys=tf(Num,Den,1)

%tf(num,den,Ts) for DT system where Ts is sampling time

figure

subplot(3,1,1)

pzmap(sys)

xlim([-3 3]), ylim([-1.5 1.5])

subplot(3,1,2)

[H w]=freqz(Num,Den);

mag=abs(H);

plot((w*180)/pi,mag)

xlabel('\omega (in degrees)')

ylabel('Gain |H(\omega)|')

title('Frequency Response')

xlim([0 180])

grid

subplot(3,1,3)

impulse(sys,'r')

grid

H=dfilt.df1(Num,Den) %Digital Filter with Direct Form-I approach

A=isfir(H)
```



Transfer Function: $H(S)=\frac{z}{z-0.9}$

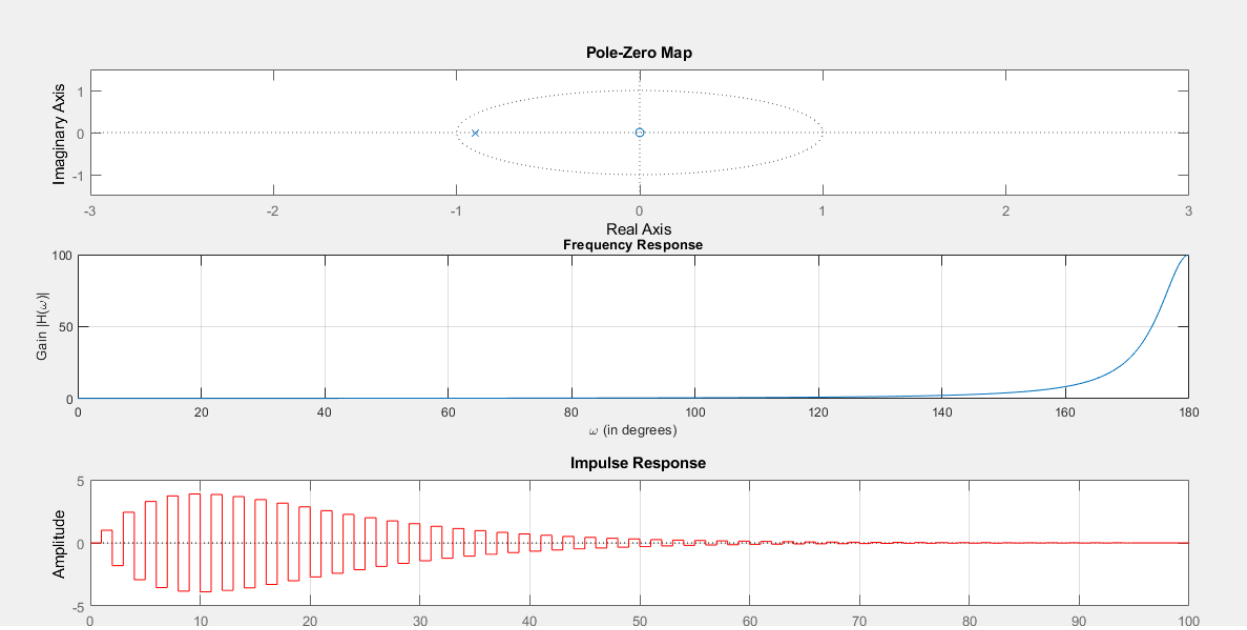
The system is IIR

3. Analysis of 1st order Digital system [pole inside the Unit circle at Fs/2]

   a) Generate pole zero constellation in z-plane for the following roots of a digital system:

   $$[\text{zero: } z = 0 \ \& \ \text{pole: } z = 0.9\angle \pm \pi]$$

   b) Write the transfer function of a system
   c) Determine that whether the digital system is FIR or IIR
   d) Plot corresponding frequency response and impulse response of a digital system



Transfer Function: $H(S)=\frac{z}{z^2+1.8\,z+0.81}$

The system is IIR

4. Analysis of 1st order Digital system [one pole at the DC location of Unit circle]

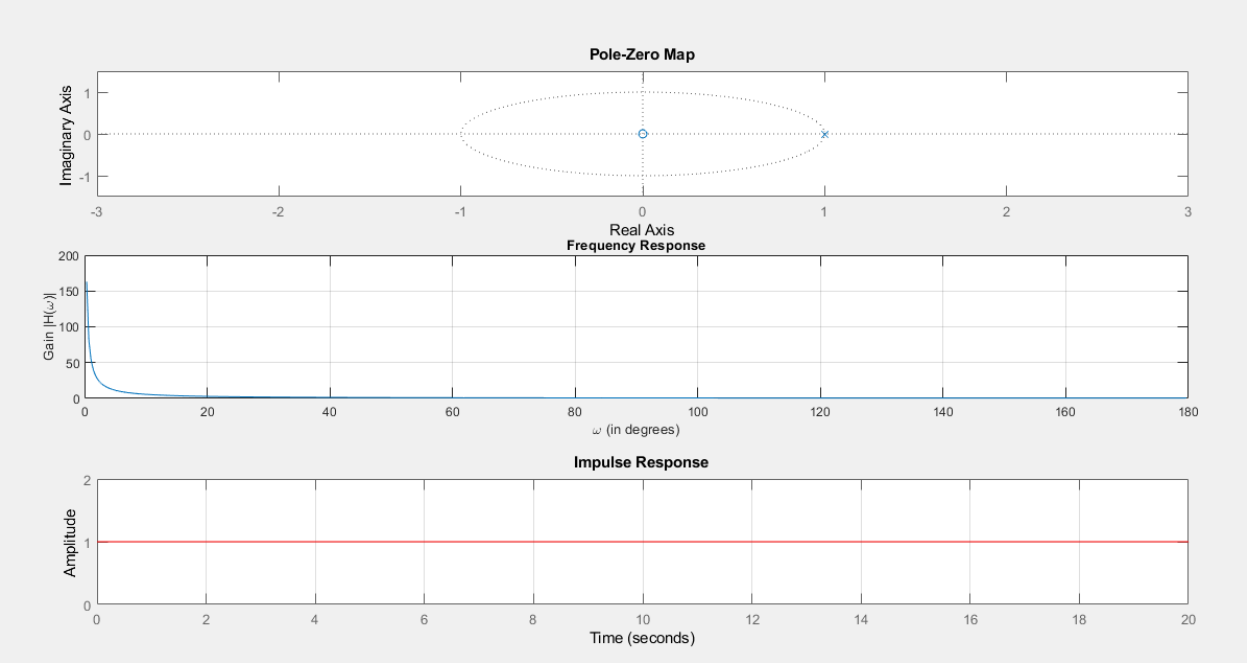
a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zero: } z=0 \ \& \ \text{pole: } z=1\angle 0]$$

b) Write the transfer function of a system

c) Determine that whether the digital system is FIR or IIR

d) Plot corresponding frequency response and impulse response of a digital system



Transfer Function: $H(S)=\frac{z}{z-1}$

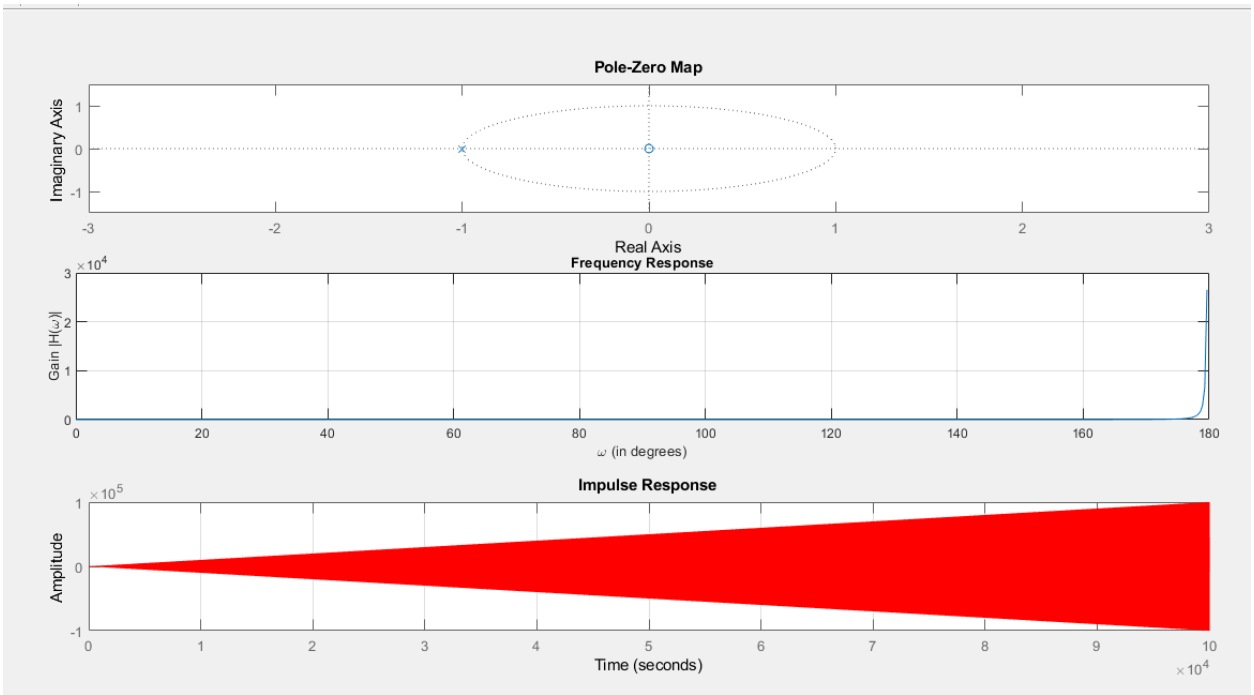
The system is IIR.

e) Analysis of 1st order Digital system [pole at the Fs/2 location of Unit circle]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zero: } z = 0 \ \& \ \text{pole: } z = 1\angle \pm \pi]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of digital system



Transfer Function: $H(S)=\frac{z}{z^2+2z+1}$

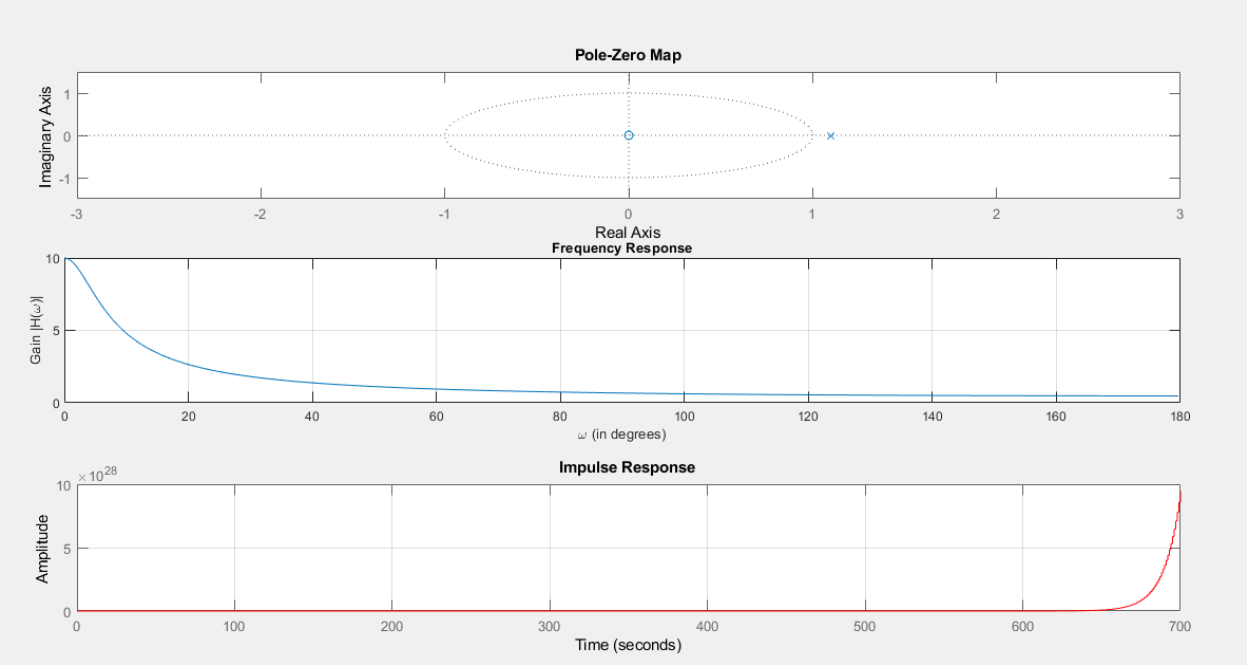
The system is IIR.

f) Analysis of 1st order Digital system [one pole outside the Unit circle at DC]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zero: } z = 0 \ \& \ \text{pole: } z = 1.1\angle 0]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system



Transfer function: $H(S)=\frac{z}{z-1.1}$

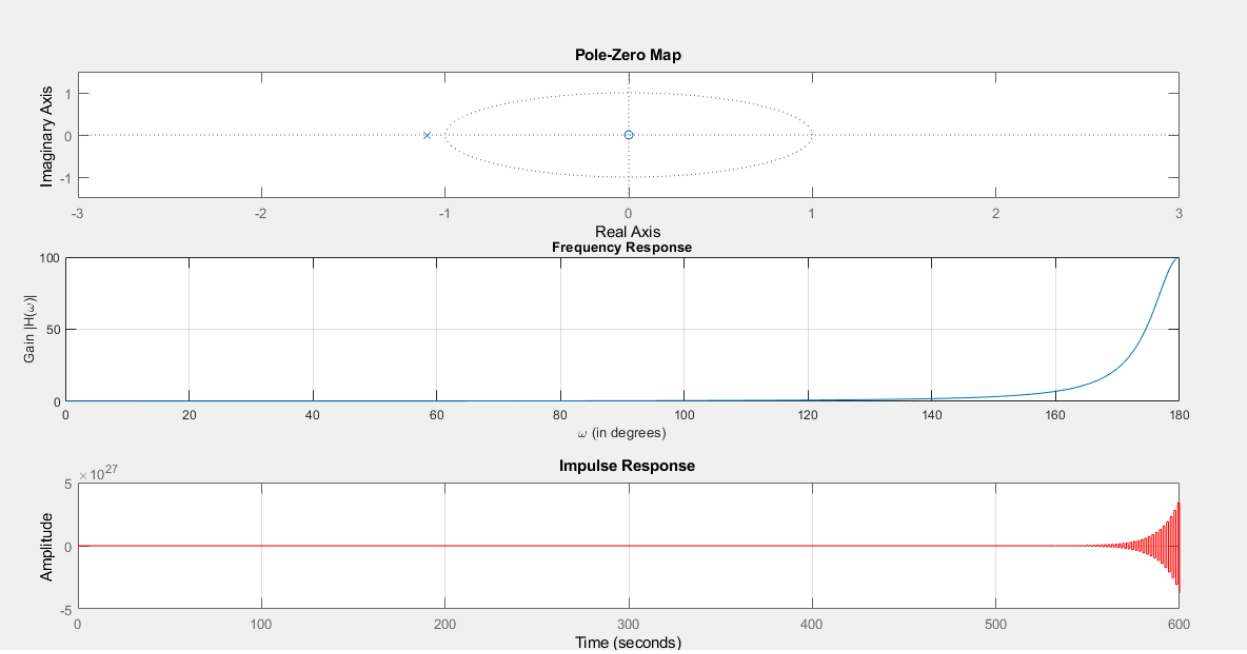
The system is IIR.

g) Analysis of 1st order Digital system [pole outside the Unit circle at Fs/2]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zero: } z = 0 \ \& \ \text{pole: } z = 1.1\angle \pm \pi]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system.



Transfer Function: $H(S)=\frac{z}{z^2+2.2\,z+1.21}$

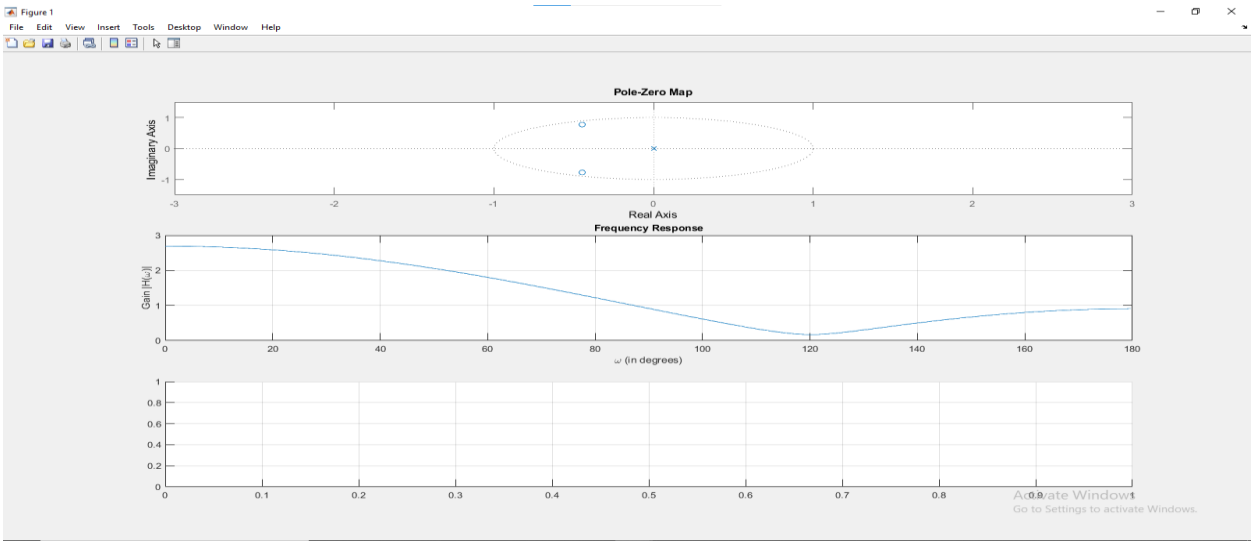
The system is IIR.

h) Analysis of 2nd order Digital system [poles at origin]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 0.8944\angle \pm 2\pi/3) \ \& \ \text{poles: } z = 0]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system



The impulse cannot be calculated on MATLAB for noncausal signals.

Transfer Function: $H(S)=\frac{z\text{^}2 + 0.8944\ z + 0.8}{z}$

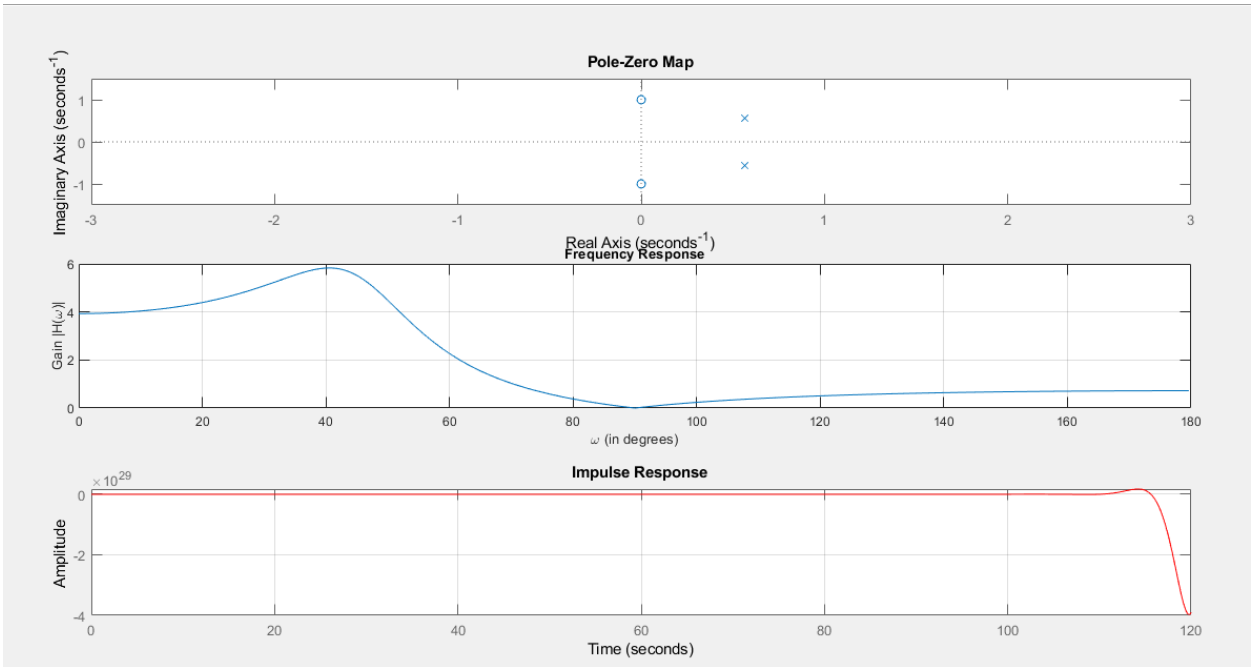
The system is FIR.

i) Analysis of 2nd order Digital system [complex poles inside the Unit circle]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 1\angle \pm \pi/2) \ \& \ \text{poles: } z = 0.8\angle \pm \pi/4]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system



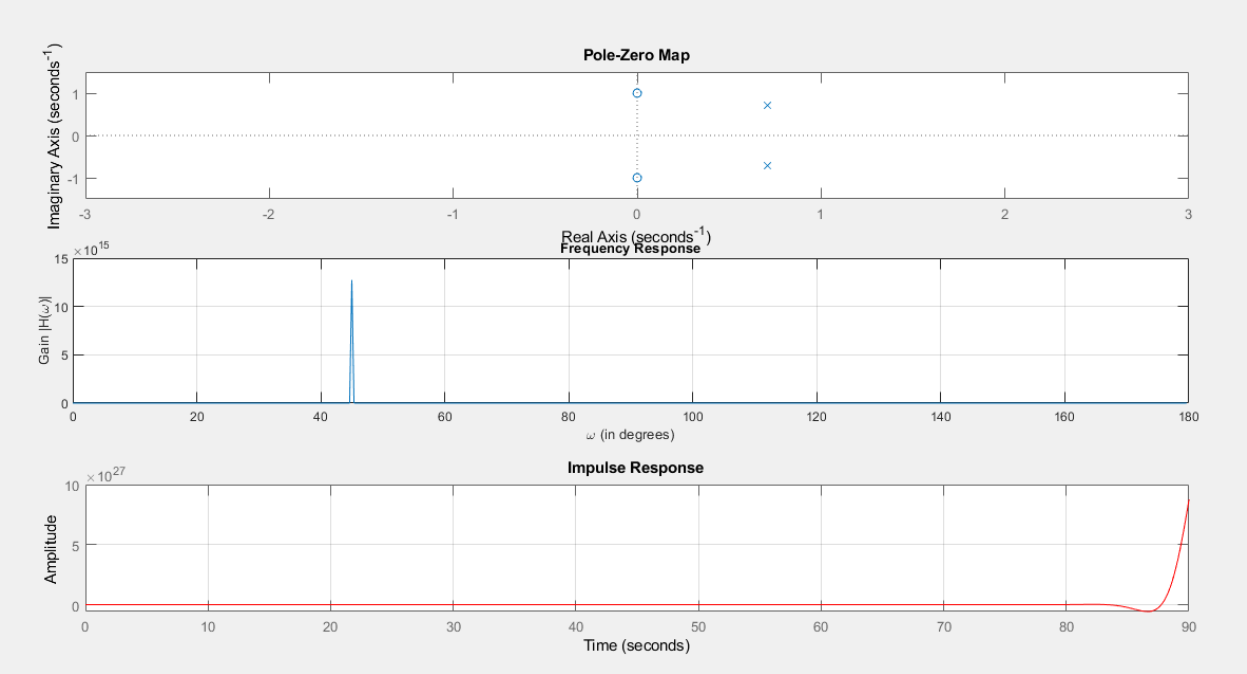
Transfer Function:
The system is IIR.

j) Analysis of 2nd order Digital system [complex poles at the Unit circle]

a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 1\angle \pm \pi/2) \ \& \ \text{poles: } z = 1\angle \pm \pi/4]$$

b) Write the transfer function of a system
c) Determine that whether the digital system is FIR or IIR
d) Plot corresponding frequency response and impulse response of a digital system

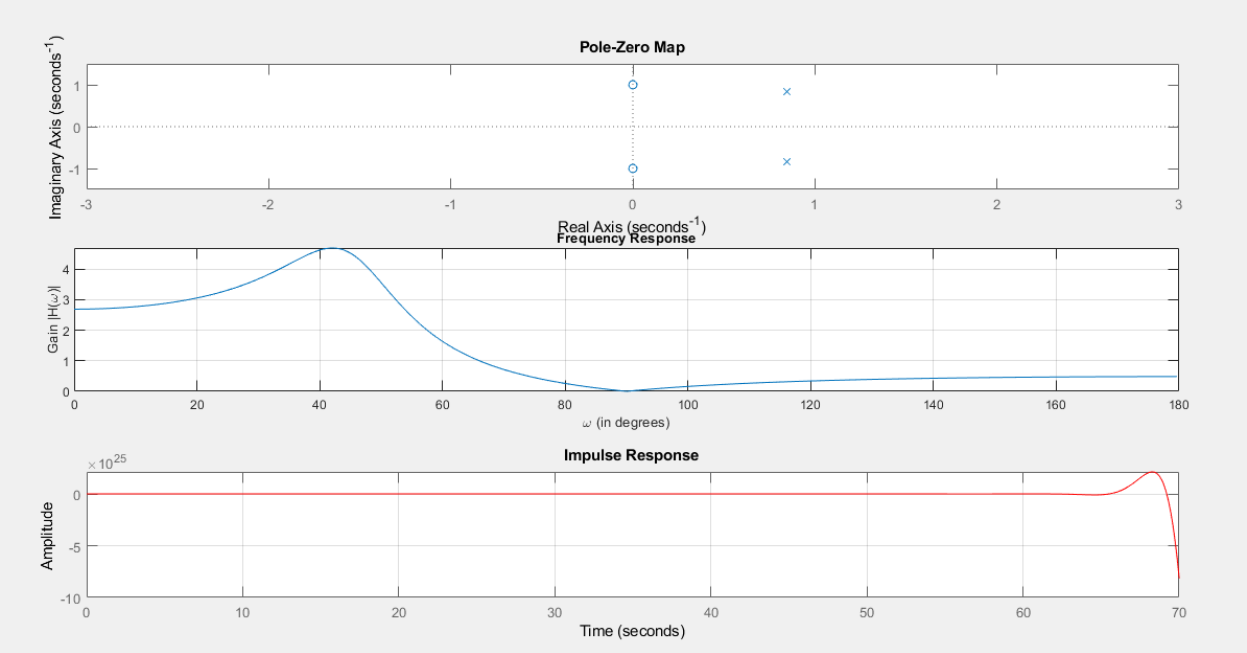


Transfer Function:
The system is IIR.

k) Analysis of 2nd order Digital system [complex poles outside the Unit circle]
   a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 1\angle \pm \pi/2) \ \& \ \text{poles: } z = 1.2\angle \pm \pi/4]$$

   b) Write the transfer function of a system
   c) Determine that whether the digital system is FIR or IIR
   d) Plot corresponding frequency response and impulse response of a digital system.

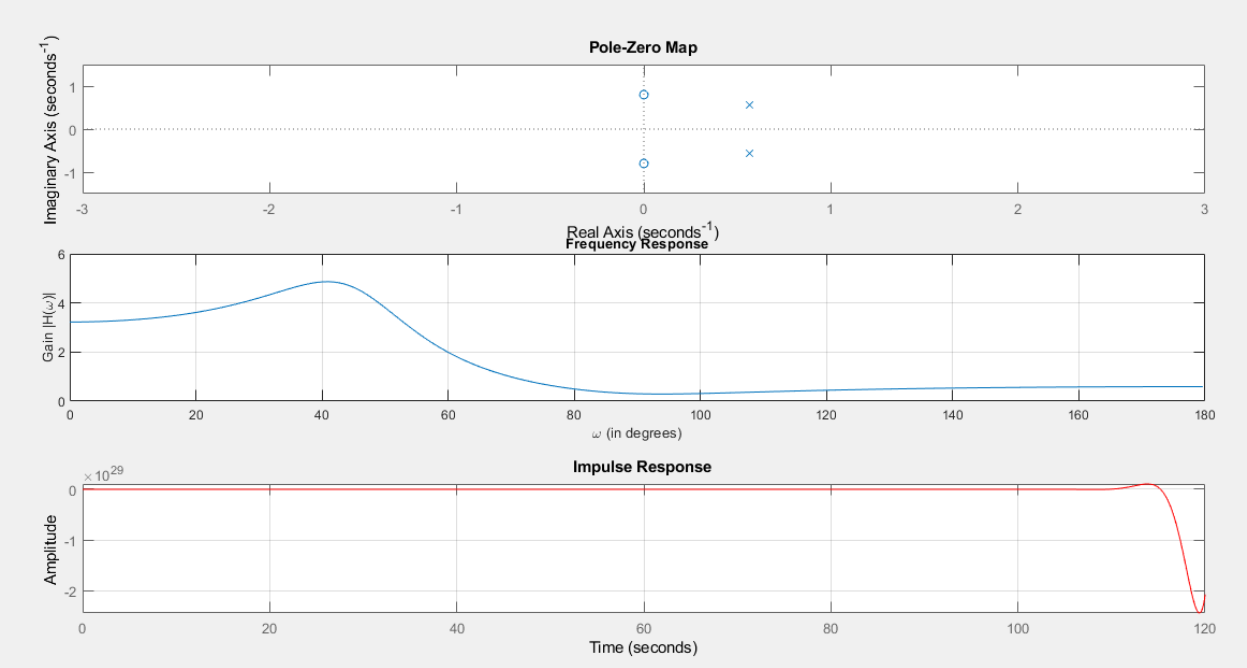


Transfer Function:
The system is IIR.

l) Analysis of 2nd order Digital system [complex zeros inside the Unit circle]
   a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 0.8\angle \pm \pi/2) \ \& \ \text{poles: } z = 0.8\angle \pm \pi/4]$$

   b) Write the transfer function of a system
   c) Determine that whether the digital system is FIR or IIR
   d) Plot corresponding frequency response and impulse response of a digital system

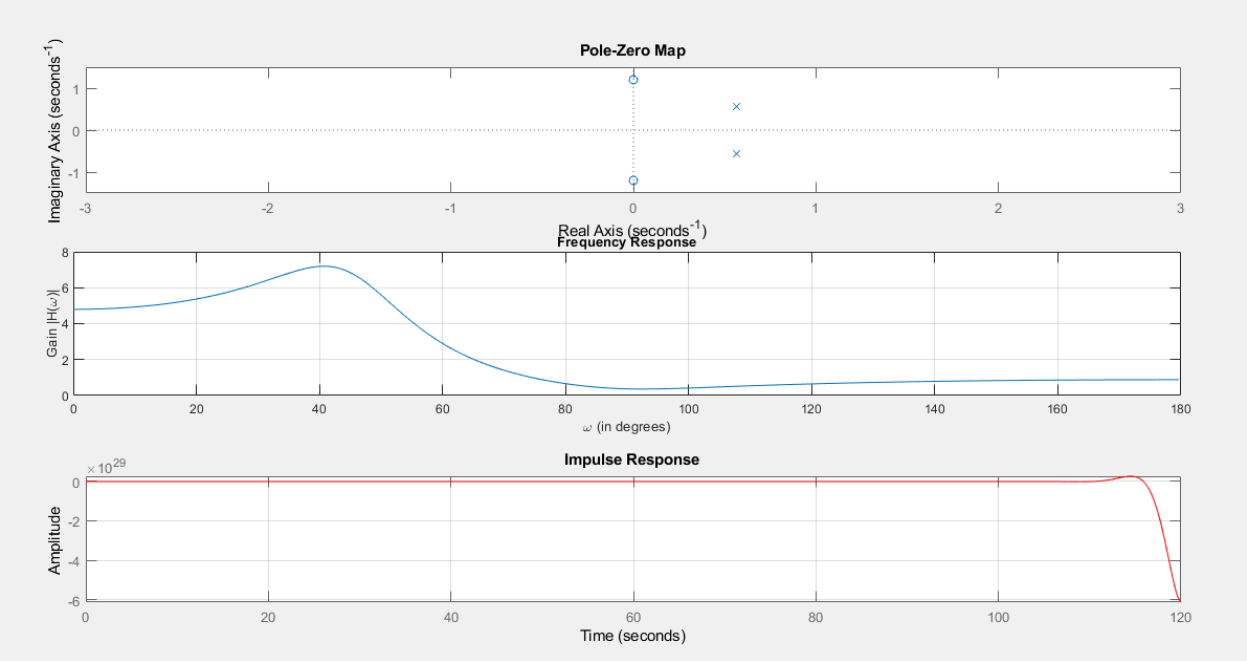


Transfer Function:
The system is IIR.

m) Analysis of 2nd order Digital system [complex zeros outside the Unit circle]
   a) Generate pole zero constellation in z-plane for the following roots of a digital system:

$$[\text{zeros: } z = 1.2\angle \pm \pi/2) \ \& \ \text{poles: } z = 0.8\angle \pm \pi/4]$$

   b) Write the transfer function of a system
   c) Determine that whether the digital system is FIR or IIR
   d) Plot corresponding frequency response and impulse response of a digital system



Transfer Function:
The system is IIR.

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 10

OBJECTIVE:

Objective of this lab is introduction to digital filters and its types, design FIR filter and study how it performs filtering on a signal. Further truncate different types of FIR filter like Low Pass, High Pass, Band Pass using different windows like rectangular, Kaiser Etc. and compare the results obtained from different windows.

THEORY:

The process of deriving a realizable transfer function of a digital filter by considering given frequency response specifications is known as digital filter design. The digital filter can be classified as:

1. Finite –duration impulse response (FIR) filter
2. Infinite –duration impulse response (IIR) filter

In MATLAB, there are built in functions which can be used to design digital filter like IIR and FIR.

The different types of FIR filters are listed as follows:

- Window techniques-based FIR filter.
  1. Rectangular windows.
  2. Hamming window.
  3. Hanning window.
  4. Blackman window.
  5. Barlett window.
  6. Kaiser window.
- Equiripple linear phase FIR filter.
- Least square error FIR filters.

The different types of IIR filters are listed as follows:

- Butterworth filter
- Chebyshev Type I filter
- Chebyshev Type II filter
- Elliptic filter

FIR digital filter operates on digital sample values. It uses current and past input samples to produce a current output sample. It does not use previous output samples. There are various types of FIR filter based on need viz. low pass, high pass, band pass and band stop,

Low pass filter.
Following points are usually considered to design FIR filter other the window type.

INPUT:

- Window Type
- Passband and stopband ripples
- passband and stopband edge frequencies
- sampling frequency
- order of the filter
- window coefficients

OUTPUT:

- magnitude and phase responses

COMPARISON OF FIR AND IIR FILTERS

1. FIR filters are Finite Impulse Response filters with no feedback, whereas IIR contains feedback.
2. Transfer function of FIR filter does not contain any non-trivial poles. Their frequency response is solely dependent on zero locations. IIR filters contain poles as well as zeros.
3. As there are no poles, FIR filters cannot become unstable; however, IIR filters can become unstable if any pole lies outside the unit circle in z-plane.
4. More number of coefficients is needed to design the desired filter in FIR than IIR.

PROCEDURE:

TASK-1

1. Create a signal vector containing two frequencies as:
   i) 100 Hz. and ii) 150 Hz. with Fs = 1000 Hz.
2. Design two band pass FIR filters with 64 coefficients and with pass bands as i) 125 to 175 Hz. and ii) 75 to 125 Hz.
3. Use both filters on the created signal and observe their outputs.
4. Plot frequency responses and pole-zero constellations of both filters and note observations.

```
close all; clear all; clc; % Frequencies in Hz.

F1 = 100;  F2 = 150;

Fs = 1000;  % Sampling Frequency in samples/sec.

t = [0 : 1/Fs : 1]; % Time Vector
F = Fs*[0:length(t)-1]/length(t); % Frequency Vector

x = exp(j*2*pi*F1*t)+2*exp(j*2*pi*F2*t); % Signal Vector

bh = fir1( 64 , [125  175]/500); % filter coeffs.
bl = fir1( 64 , [75  125]/500); % filter coeffs. [hh,wh]=freqz(bh,1,length(t),'whole');
% Frequency response for filter 1 [hl,wl]=freqz(bl,1,length(t),'whole'); %
Frequency response for filter 2
% Filter operation - see filtfilt in help to learn what it does

yh = filtfilt(bh,1,x);
yl = filtfilt(bl,1,x);

% Plotting

figure, subplot(5,1,1),

plot(F,abs(fft(x)));

xlim([0 Fs/2]);
title('FFT of original signal');

subplot(5,1,2),
plot(F,abs(hh));
xlim([0 Fs/2]);
title('Frequency response of Filter One');

subplot(5,1,3),
   plot(F,abs(fft(yh)));
   xlim([0 Fs/2]);
   title('FFT of filtered signal from filter one');

subplot(5,1,4),
plot(F,abs(hl));
xlim([0 Fs/2]);
title('Frequency response of Filter Two');

subplot(5,1,5),
plot(F,abs(fft(yl)));
```

```
xlim([0 Fs/2]);
title('FFT of filtered signal from filter two');
xlabel('Hz.')



% Pole Zero Constellations

[bh,ah] = eqtflength(bh,1);
[zh,ph,kh] = tf2zp(bh,ah);
[bl,al] = eqtflength(bl,1);
[zl,pl,kl] = tf2zp(bl,al);

figure,
subplot(1,2,1),
pzplane(bh,ah);
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
title('Filter_One');
subplot(1,2,2),
pzplane(bl,al);
xlim([-1.5 1.5]);
ylim([-1.5 1.5]);
title('Filter Two');
```

TASK -2

Write a program to design a FIR filter using Hanning windows,take inputs from user for design values of filter.

```
close all;
clear all;
clc;

fp=input('Enter the pass band frequency');
fs=input('Enter the stop band frequency');
rp=input(' Enter the pass band attenuation');
rs=input('Enter the stop band attenuation');
f=input(' Enter the sampling frequency');

% Calculating filter order

num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
```

```
n=abs(n);

% Normalizing the frequencies

wp=2*fp/f;
ws=2*fs/f;
wn=(ws+wp)/2;

%Adjusting the filter order. The order of window must be an odd number
%and the order of filter must be one less than that of the window

if (rem(n,2)==0)
   m=n+1;
else
      m=n;
      n=n-1;
end

%Window sequence calculation

w=hann(m);

%Calculation of filter coefficients

b=fir1(n,wn,'low',w);

%Plotting the filter response

freqz(b,1,500,3000);
TITLE('Magnitude and Phase response');
```

TASK-3

Write a program for FIR (Finite Impulse Response) filter like Low pass FIR filter, High pass FIR filter, Band pass FIR filter and Band stop FIR filter using rectangular window using MATLAB.

ALGORITHM:

LOW PASS FILTER:

Step 1: Read the input sequence

Step 2: Perform low pass filter calculations

Step 3: Plot the output sequences

HIGH PASS FILTER:
Step 1: Read the input sequence
Step 2: Perform high pass filter calculations
Step 3: Plot the output sequences

BAND PASS FILTER:
Step 1: Read the input sequence
Step 2: Perform band pass filter calculations
Step 3: Plot the output sequences

BAND STOP FILTER:
Step 1: Read the input sequence
Step 2: Perform band stop filter calculations
Step 3: Plot the output sequences

PROGRAM:

```
clc;
clear all;
close all;
rp=input('Enter the passband ripple(rp):');
rs=input('Enter the stopband ripple(rs):');
fp=input('Enter the passband frequency(fp):');
fs=input('Enter the stopband frequency(fs):');
f=input('Enter the sampling frequency(f):');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2)~=0)
   n1=n;
   n=n-1;
end
y=boxcar(n1);
%Low pass filter
b=fir1(n,wp,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(m);
ylabel('Gain(db)->');
```

```
xlabel('(a)Normalised frequency->');
%High pass filter
b=fir1(n,wp,'high',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));

subplot(2,2,2);
plot(m);
ylabel('Gain(db)');
xlabel('(b)Normalised frequency');
%Band pass filter
wn=[wp*ws];
b=fir1(n,wn,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);
plot(m);
ylabel('Gain(db)');
xlabel('(c)Normalised frequency');
%Band stop filter==============
wn=[wp*ws];
b=fir1(n,wn,'stop',y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(m);
ylabel('Gain(db)');
xlabel('(d)Normalised frequency-');
```

EXERCISE:

Q1. Perform Q3. using Hamming and Kaiser Window.
Compare results of designed filters using three different windows on a single plot.

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

LAB SESSION 11

OBJECTIVE:

Object of this lab is to design different IIR filter using FDA tool.

THEORY:

Filter Design and Analysis Tool (FDA Tool) is Graphic User Interface for designing and analyzing filters. It is used to design FIR and IIR filters by entering the desired filter specifications, or by importing filter from MATLAB workspace or by adding, moving or deleting poles and zeros. After designing a filter, the response can be viewed and analyses in other Graphic User Interface tool named Filter Visualization Tool (FV Tool) linked with FDA Tool. The different types of responses that can be viewed are listed below:

- Magnitude response
- Phase response
- Group delay
- Phase delay
- Impulse response
- Step response
- Pole-zero plot
- Zero-phase plot

OPENING FDA TOOL WINDOW:

FDA Tool can be opened using command:

fdatool

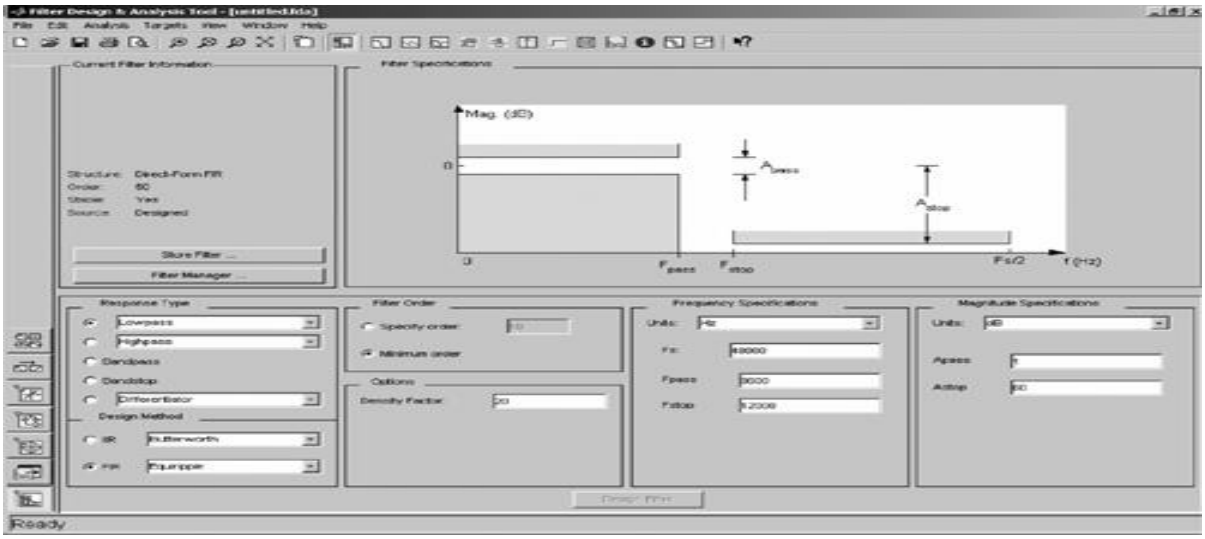


Figure A

The different steps involved in designing a filter using FDA Tool can be listed as:

1. Selection of type of response required
2. Selection of type of filter design
3. Specifying the filter order
4. Entering the filter specifications
5. Entering the magnitude specifications

After providing the information listed above, filter can be designed and its response can be viewed and analyzed.

The complete description of the FDA Tool window and different steps required to design a filter are elaborated below:

1. **Selecting response type**: The desired response type is selected from the list of available
   options, i.e., lowpass, high pass, bandpass, band stop, differentiation, multiband, peaking etc.

2. **Type of design method**: The design can be of FIR or IIR filter. Depending upon whether
FIR Or IIR filter design is selected, further options are available in the dropdown menu.
In IIR filter design, the different options available in dropdown menu are as given below:

- Butterworth
- Chebyshev type I
- Chebyshev type II
- Elliptic
- Maximally flat
- Least Pth-norm
- Const least Pth-norm

In FIR filter design the options available are listed as follows:

- Equirriple
- Least square
- Window
- Const least squares
- Complex equiripple
- Least Pth norm
- Constrained equiripple
- Generalized equiripple
- Constrained band equirriple
- Interpolated FIR

The options available depend upon the selection of response type.

3. **Filter order**: Under this, two options available are as
   - User-defined order: Under this option user has to enter the order of the filter.
   - Minimum order: The other option available for selecting the filter order is minimum order. It is calculated by system itself.

4. **Filter specifications**: Depending upon the response type and design method selected, the graphical representation of generalized filter specifications appear in the display region of FDA Tool. These specifications are 'Frequency Specifications' and 'Magnitude Specification'.
These specifications are provided by the user, as per filter design requirement, in the appropriate blocks.

5. **Designing filter**: After all the requisite information is entered, a filter can be designed by
clicking the 'Design Filter' button available at the bottom of the window. Filter | coefficients are calculated and magnitude response appears in the display region.

(Note: 'Design Filter' button will be disabled once the filter coefficients are computed. This button will be enabled again in case any changes are made in the filter specifications.)

6. **Displaying filter responses:** Once the filter coefficients are calculated as per the specifications provided by the user, the display region will show
magnitude response of the designed filter. The other filter response characteristics can be viewed in the display region or FV Tool. The response to be viewed can be selected from the different icons displayed on the toolbar shown in Figure below.

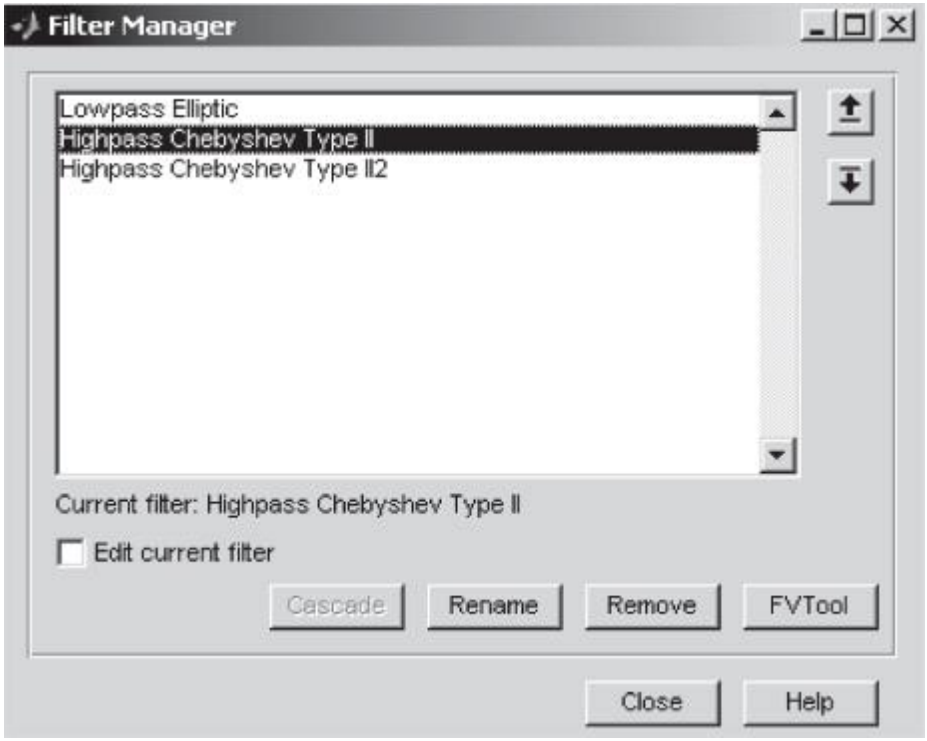
Figure: Different Response Icons on the Toolbar

(NOTE: The different responses for display can also be selected from the 'Analysis' menu on menu bar.)

7. **Current filter information**: The information about the designed filter is given in the 'Current Filter Information' region of FDA Tool window as shown in Figure A The information provided is about the 'structure', 'order', 'stability' and 'source'

- Storing a filter
  The designed filter is stored by clicking 'Store Filter' button in the 'Current Filter Information' region.
- Filter manager
  The 'Filter Manager' button opens up a new Filter Manager window (Figure B) showing the list of filters stored. This window also has options as: Edit current filter, Cascade, Rename, Remove and FV Tool.

  Figure B Filter Manager Window



To cascade two or more filters, highlight the designed filters and press 'Cascade' button. A new cascaded filter is added to the 'Filter Manager'.

8. Saving and opening filter design session:
The filter design session is saved as MAT-file and can be used later on. It can be saved by clicking save icon or selecting save session
option in File menu and giving desired session name. Similarly, the saved session can be opened by clicking open icon or by selecting open option in file menu and selecting the previously saved filter design session.

FILTER VISUALIZATION TOOL:

The response characteristics can be viewed in a separate window by selecting the 'Filter Visualization Tool' (FV Tool) from 'view' menu or clicking the 'Full View Analysis' button on the toolbar. The FV Tool window is shown in Figure C

FV Tool has most of the menus on the menu bar and icons on the toolbar similar to that FDA Tool with some additional icons which are mainly used to work with

representation of the responses.

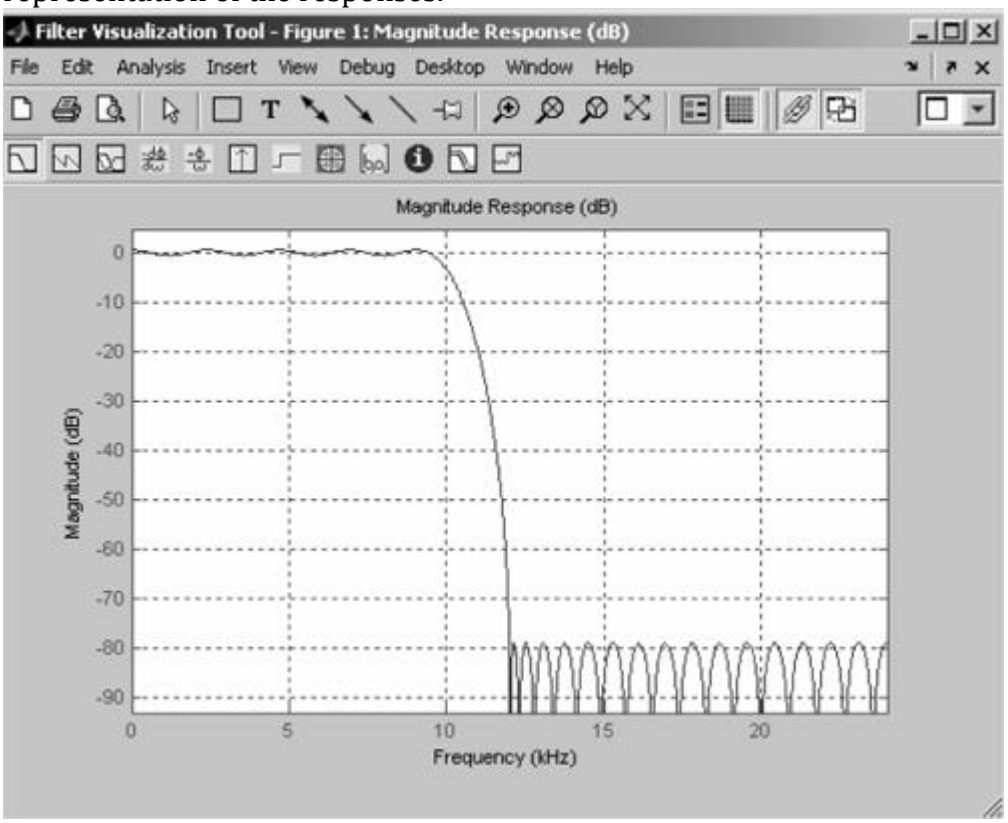


Figure C Filter Visualization Tool (FV Tool) window

IIR FILTER DESIGN USING FDA TOOL

TASK-1

Design an IIR Butterworth band pass filter with the following specifications:

Normalized pass band edges at 0.40 and 0.65

Normalized stop band edges at 0.3 and 0.75

Pass band ripple 1 dB

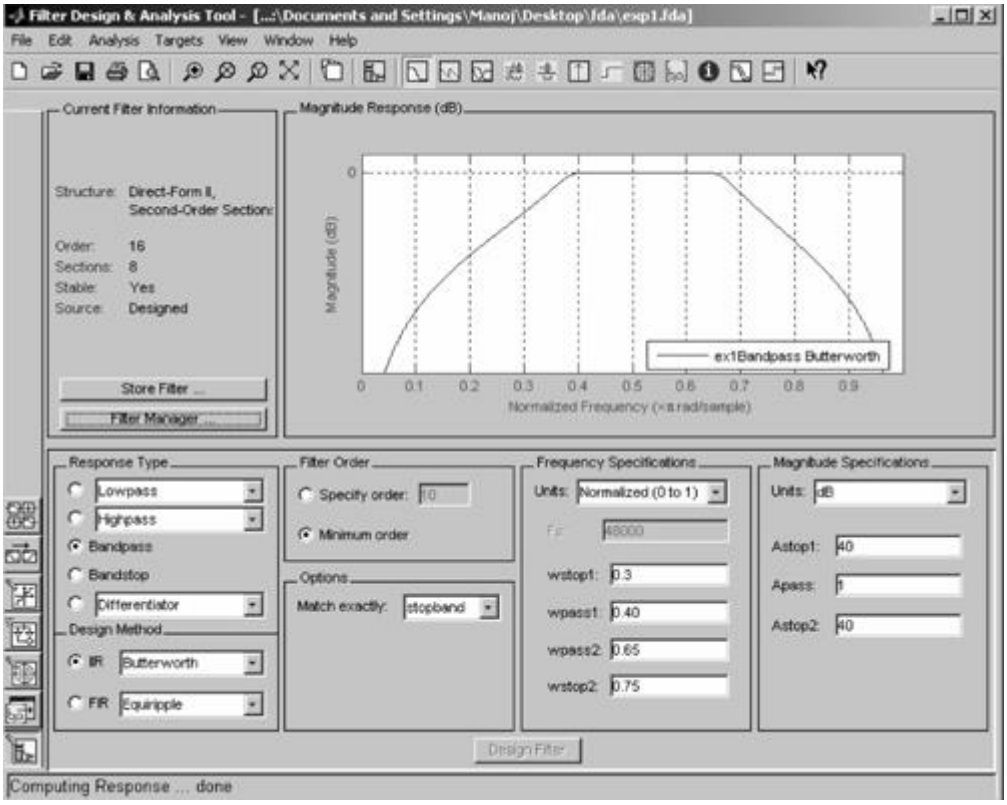
Minimum stop band attenuation 40 dB

Show (i) Magnitude response (ii) Phase response (iii) Group delay (iv) Phase delay response.

Solution:

As per the given specifications, the requisite data is entered in new FDA Tool window as shown in Figure

Figure. FDA Tool Window Showing Specification Entered and Magnitude Response for Task-1.



The filter is designed for minimum order so as to reduce the complexity of the design. In case, it has to be designed for user defined order, then the order of the filter has to be calculated first by user using appropriate formulas or MATLAB function.

The other responses can be viewed by clicking on the appropriate icon on the toolbar and responses obtained are shown in Figures below

Figure Magnitude Response in dB

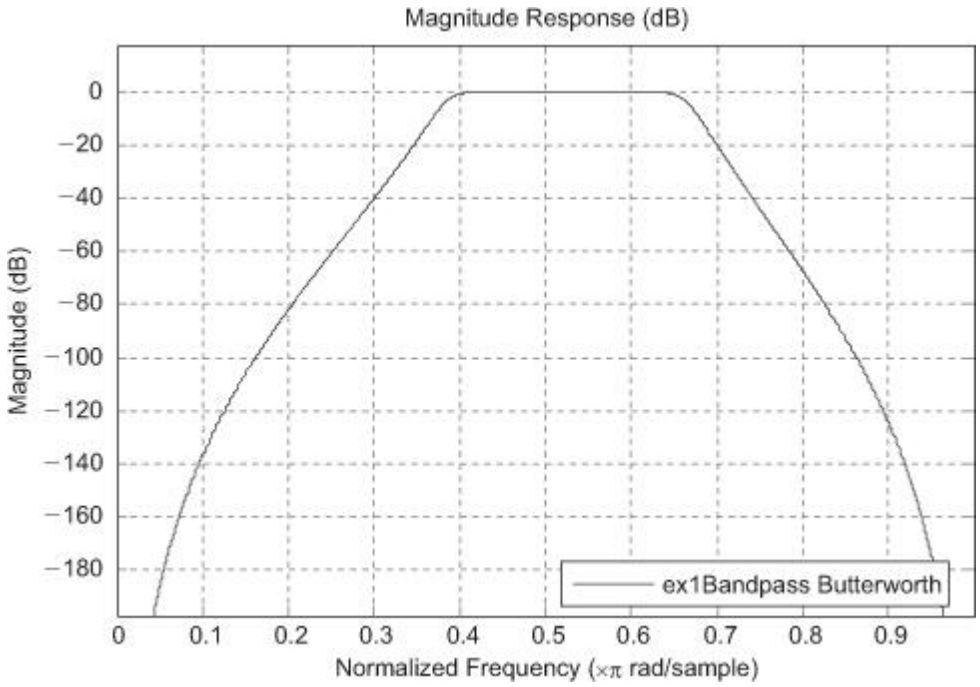


Figure Phase Response

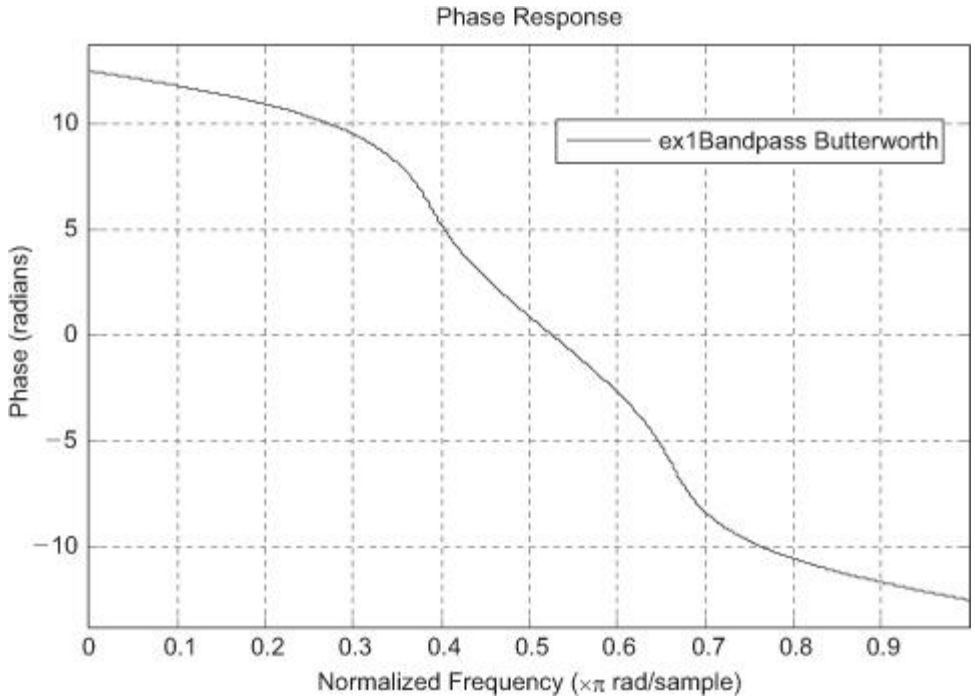


Figure Group Delay Response

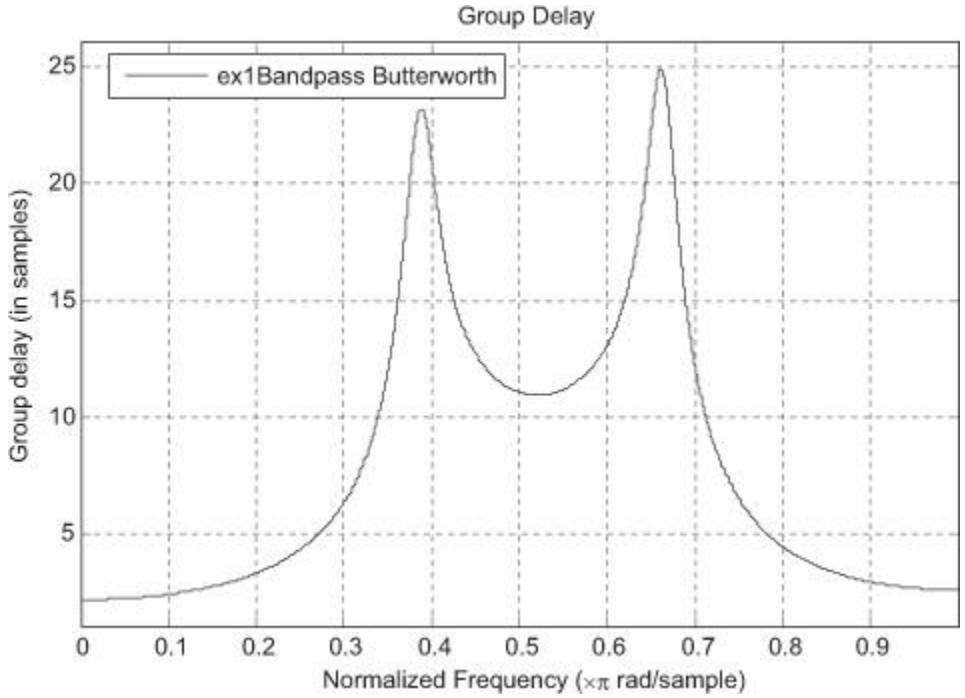
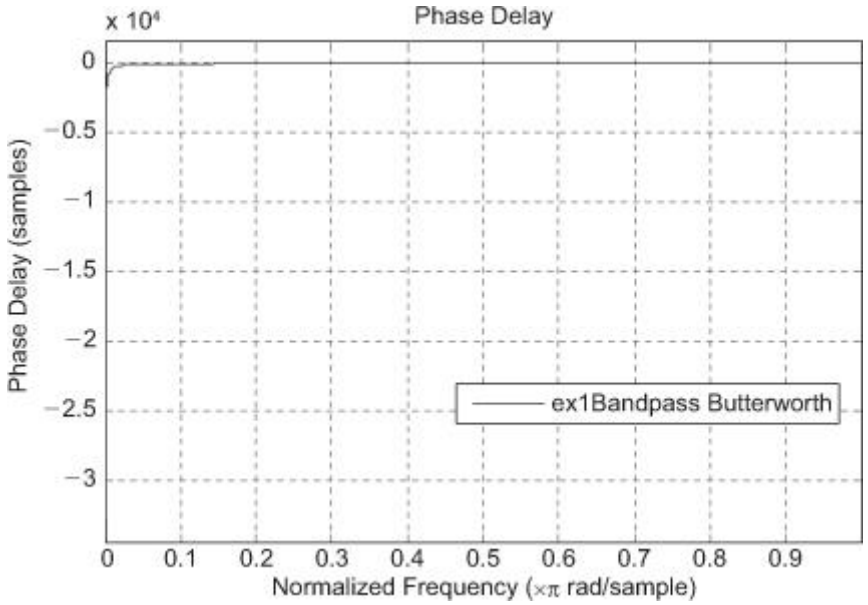


Figure. Phase Delay



These responses can also be viewed in FV Tool.

TASK-2

Design a Type II Chebyshev IIR lowpass filter with the following specifications:

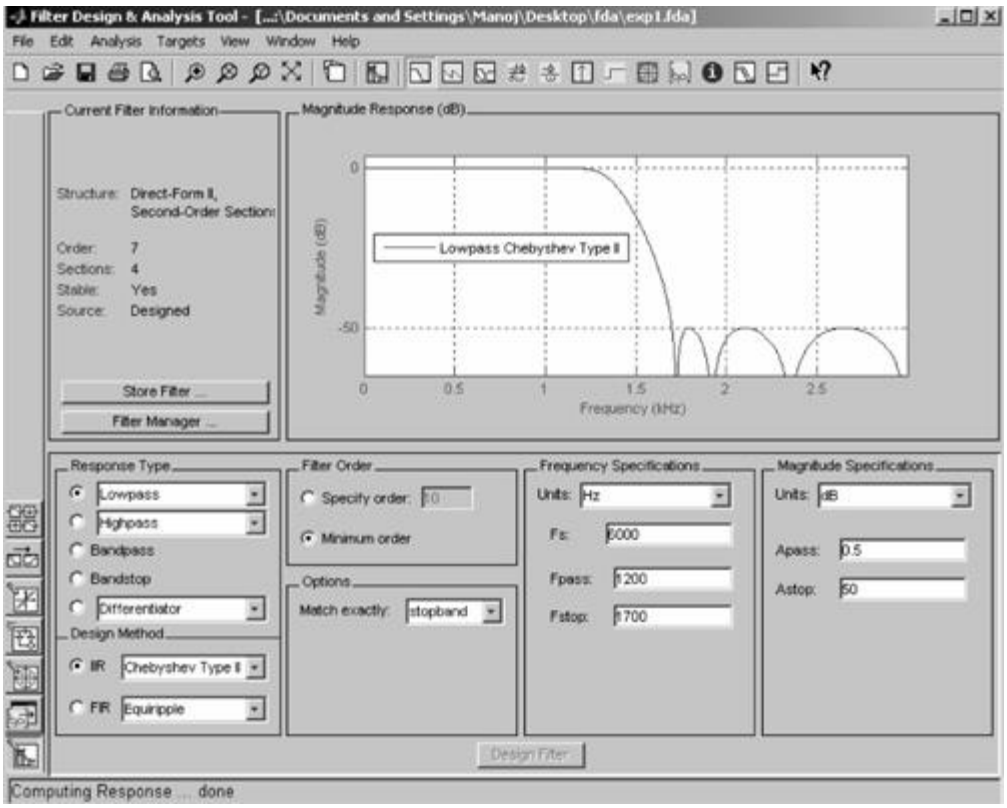
Passband frequency	1,200 Hz
Stopband frequency	1,700 Hz
Sampling frequency	6,000 Hz
Passband ripple	0.50 dB

1. Show magnitude response.

2. Show pole/zero plot.
3. Impulse Response
4. Pole Zero plot

Solution: FDA Tool Window showing given specifications duly entered and magnitude response in response display region is shown in Figure.

Figure. FDA Tool Window for Example



By using the FV Tool, the magnitude response and pole/zero plot is obtained as separate figures and is shown in Figures.

Figure. Magnitude Response for Task-2.

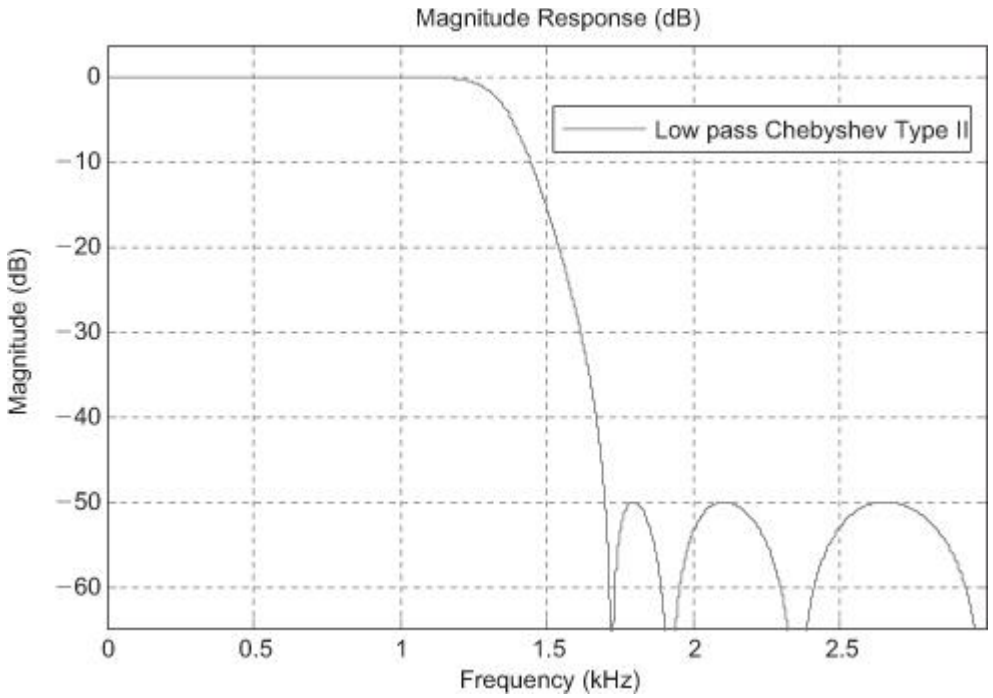
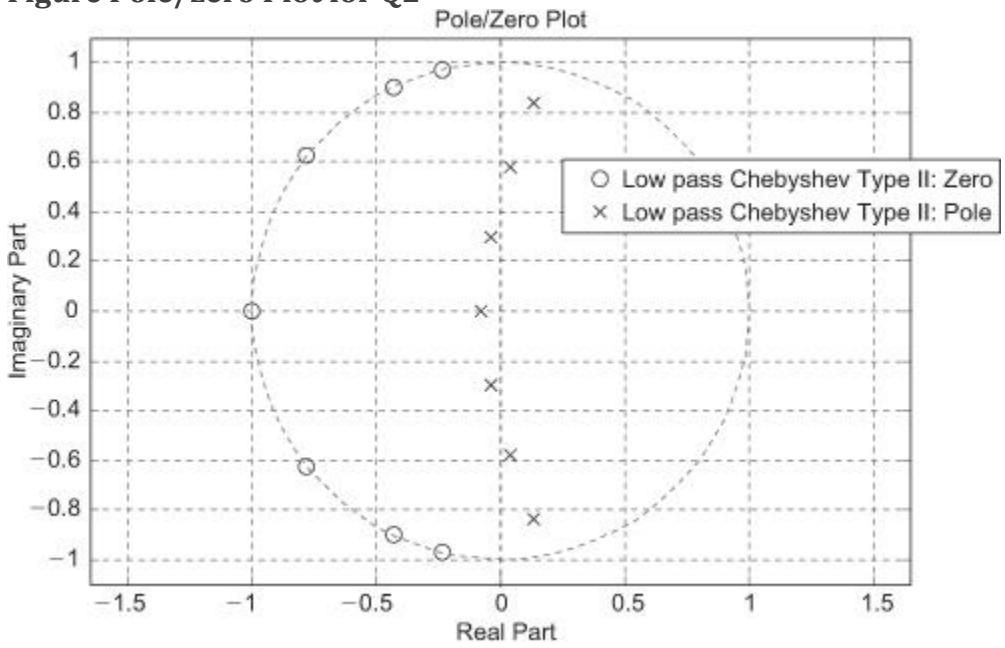


Figure Pole/zero Plot for Q2



TASK-3:

Design an elliptic IIR low pass filter with following specifications:

Pass band ripple	0.5 dB
Stop band attenuation	40 dB
Pass band frequency	800 Hz
Stop band frequency	1,000 Hz
Sampling frequency	4,000 Hz

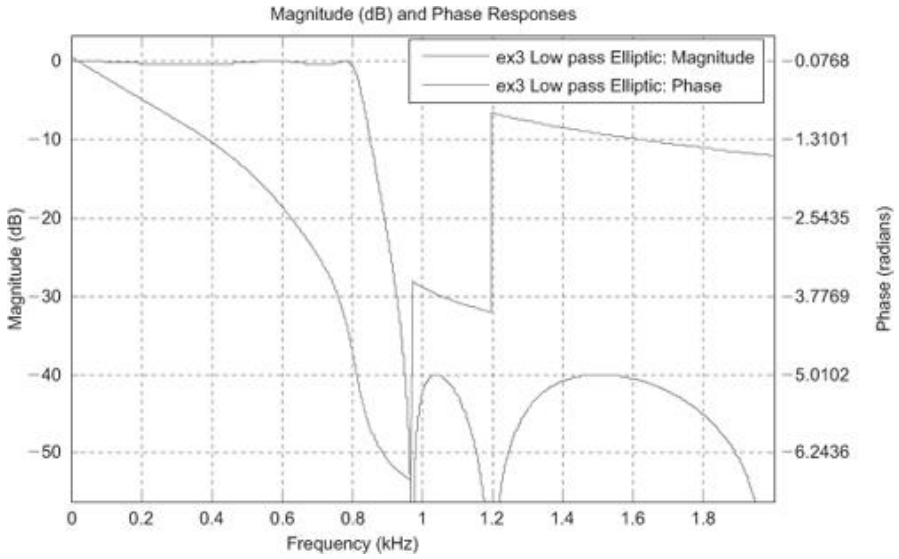
1. Show magnitude and phase response in the same window.
2. Attach impulse response
3. Attach pole zero plot
4. Obtain filter information

Solution:

As per given specifications, the requisite data is entered in FDA Tool window. By clicking appropriate icon on toolbar, the magnitude and phase responses are obtained in the same window.

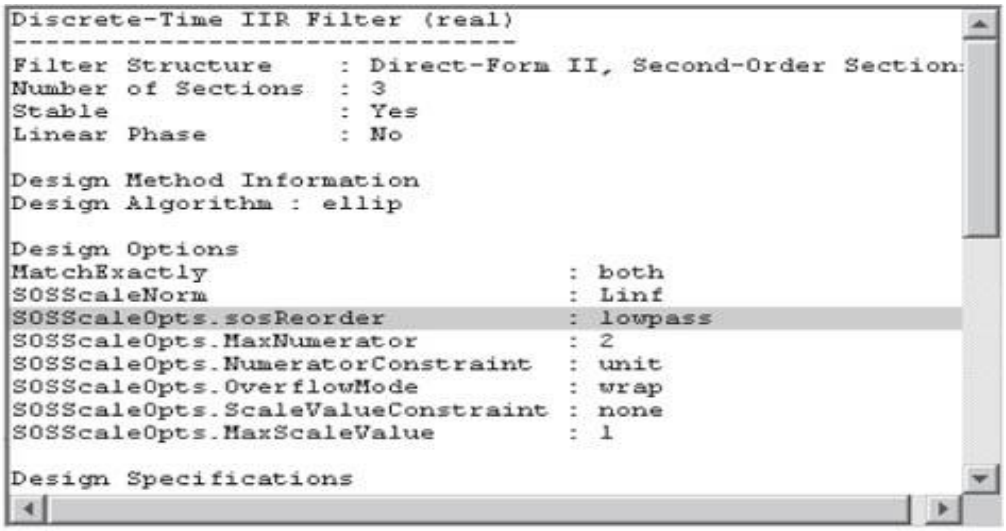
1. These magnitude and phase responses obtained are viewed in FV Tool window also and are shown in Figure.

Figure. Magnitude and Phase Response for Task-3.

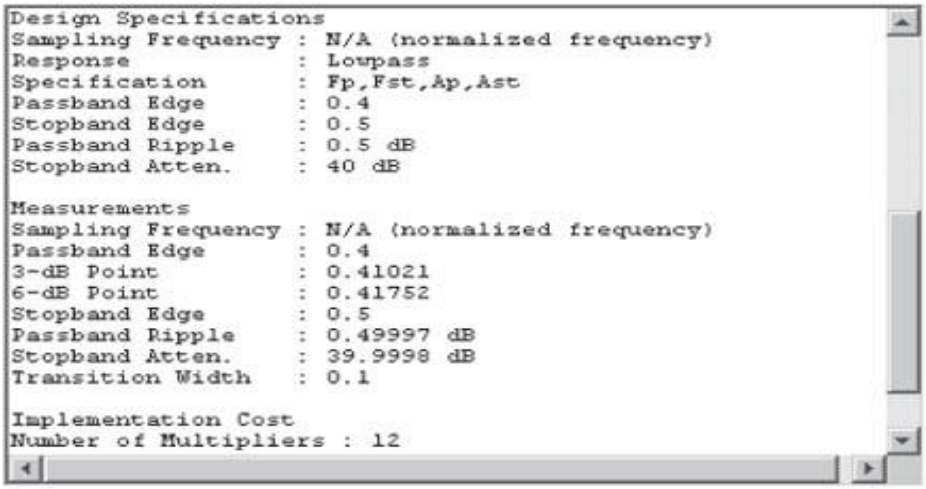


2. To obtain the information about the filter 'Filter Information' icon on Toolbar of FDA Tool Window is clicked or 'Filter Information' option is selected from 'Analysis' menu. The detailed filter information appears in the display region as shown in Figure a, b and c.

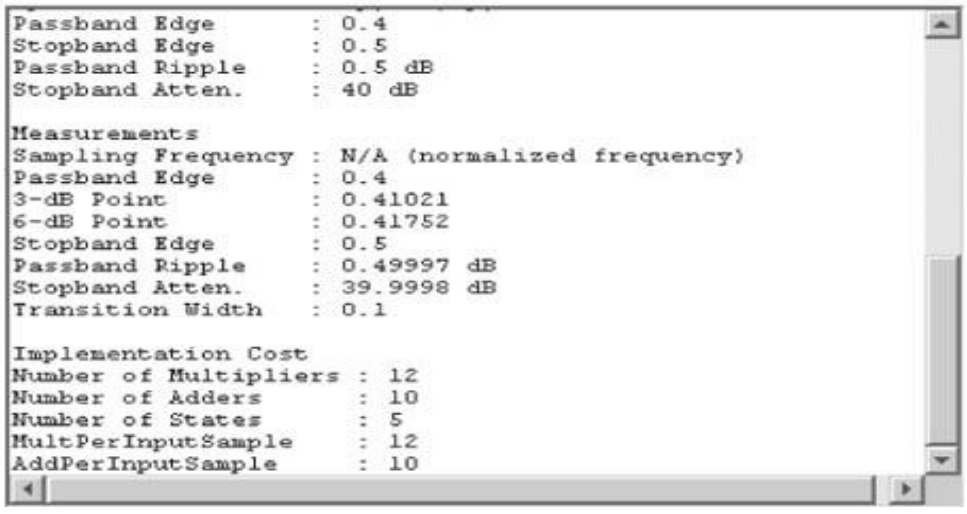
(a)


Discrete-Time IIR Filter (real)
Filter Structure : Direct-Form II, Second-Order Section
Number of Sections : 3
Stable : Yes
Linear Phase : No
Design Method Information
Design Algorithm : ellip
Design Options
MatchExactly : both
SOSScaleNorm : Linf
SOSScaleOpts.sosReorder : lowpass
SOSScaleOpts.MaxNumerator : 2
SOSScaleOpts.NumeratorConstraint : unit
SOSScaleOpts.OverflowMode : wrap
SOSScaleOpts.ScaleValueConstraint : none
SOSScaleOpts.MaxScaleValue : 1
Design Specifications

(b)


Design Specifications
Sampling Frequency : N/A (normalized frequency)
Response : Lowpass
Specification : Fp,Fst,Ap,Ast
Passband Edge : 0.4
Stopband Edge : 0.5
Passband Ripple : 0.5 dB
Stopband Atten. : 40 dB
Measurements
Sampling Frequency : N/A (normalized frequency)
Passband Edge : 0.4
3-dB Point : 0.41021
6-dB Point : 0.41752
Stopband Edge : 0.5
Passband Ripple : 0.49997 dB
Stopband Atten. : 39.9998 dB
Transition Width : 0.1
Implementation Cost
Number of Multipliers : 12

(c)


Passband Edge : 0.4
Stopband Edge : 0.5
Passband Ripple : 0.5 dB
Stopband Atten. : 40 dB
Measurements
Sampling Frequency : N/A (normalized frequency)
Passband Edge : 0.4
3-dB Point : 0.41021
6-dB Point : 0.41752
Stopband Edge : 0.5
Passband Ripple : 0.49997 dB
Stopband Atten. : 39.9998 dB
Transition Width : 0.1
Implementation Cost
Number of Multipliers : 12
Number of Adders : 10
Number of States : 5
MultPerInputSample : 12
AddPerInputSample : 10

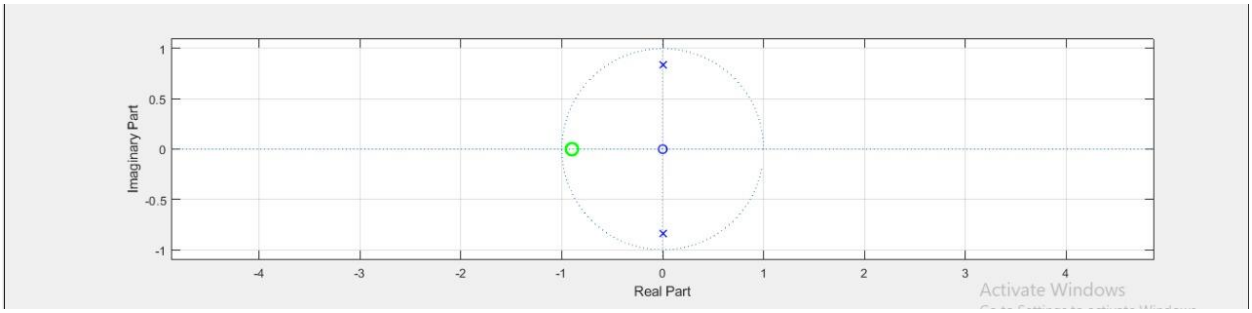
The filter information is obtained by scrolling down the text in the window shown in Figure b.

Task-4:
IIR filter designing by Pole-Zero placement method

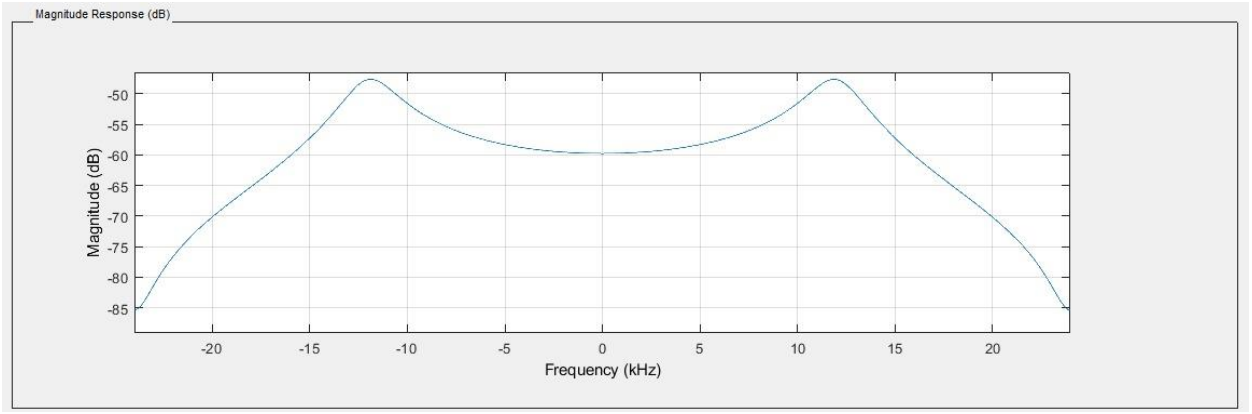
Design an IIR filter with the following specifications:

Pass band at pi/2 and stop bands at 0 and pi. Take $F_s = 8000\ Hz$.

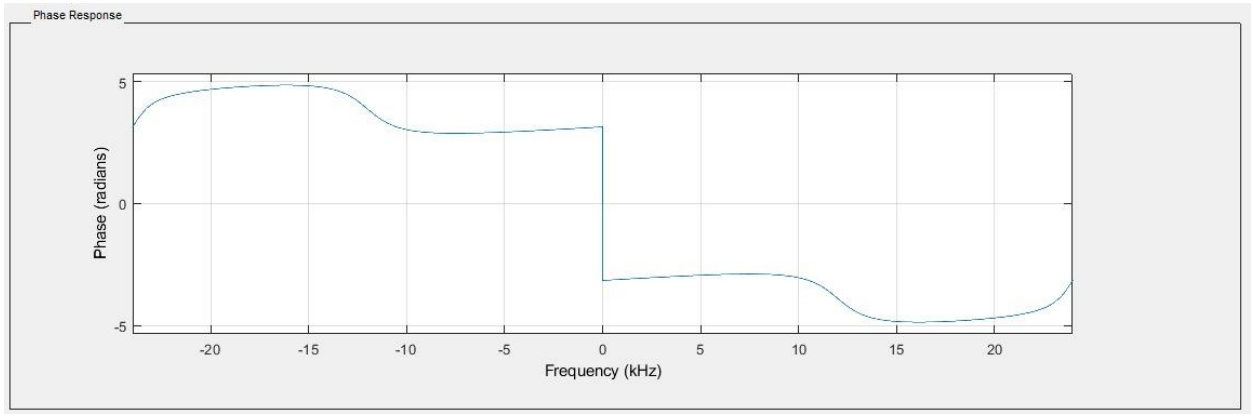
a) Attach pole zero plot



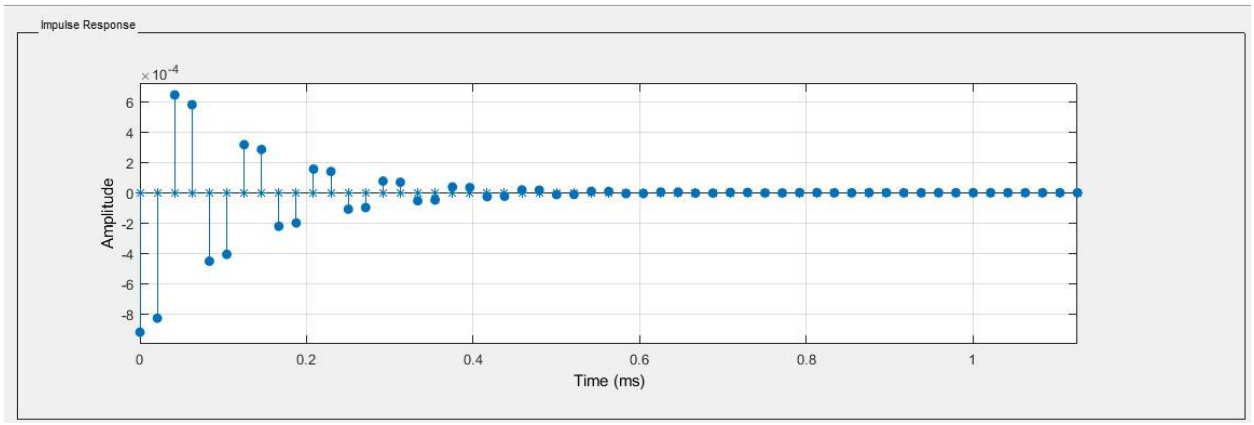
b) Attach magnitude response



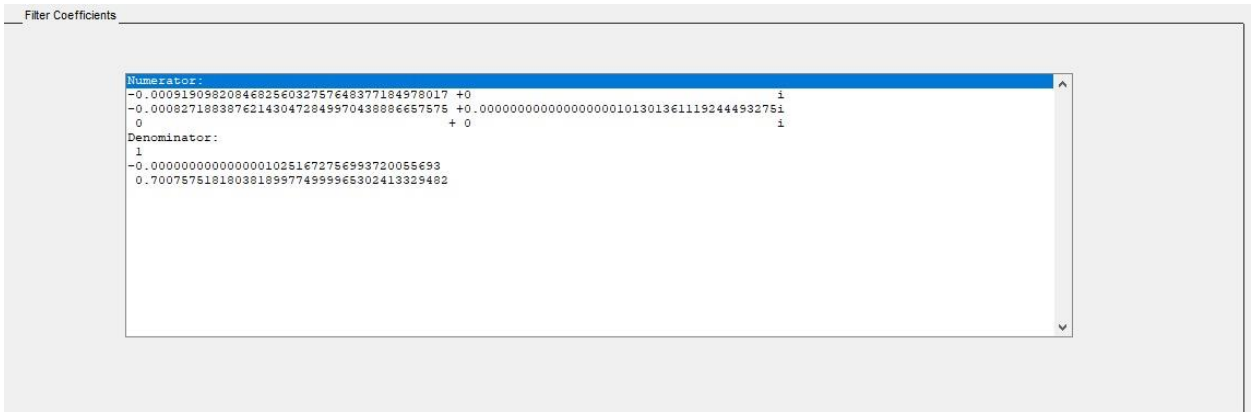
c) Attach phase response



d) Attach impulse response



e) Attach filter coefficients



EXERCISE:

Record Your Voice at home while turn any motor of your house ON.
Design a filter using FDA Tool.

Remove sound of motor from recorded signal.

Listen the output signal.

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

(Open Ended Lab 01)

Objective:

To convert an analog (voltage & current) signal into digital signal using ADC (audio card) and display it on MATLAB Simulink environment.

Required Components:

1. Audio Card
2. Transformer (220V/12V)
3. Resistors (for VDR)
4. Veroboard
5. Audio jack
6. PC with MATLAB environment

Procedure:

- Using Transformer convert 220VAC from mains into 12VAC.
- Using VDR convert 12VAC to a voltage compatible to audio card (show all the calculations of resistances with their power ratings).
- Set the sampling frequency of the audio card ADC in MATLAB Simulink environment with proper justification
- Plot the acquired voltage waveform to Simulink scope.
- Mention the safe operating range of your equipment.

Project Summary: (Not more than one sheet)

Project Specification:

Calculations:

- VDR and Resistance power calculation
- Resolution of ADC
- Sampling Frequency
- Gain Factor

Attachments:

- Project Block Diagram
- Real Project Image
- Image of current and voltage plot (with proper labelling)

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed Alam **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		

(Open Ended Lab 02)

Objective:

To convert analog (Voltage and current) signal into digital signal using ADC (audio card). Display it on MATLAB Simulink environment and perform Spectral Analysis of the resulting current signal.

Required Components:

1. Audio Card
2. Current Sensor (current sensing resistor / hall effect sensor / CT)
3. Vero board
4. Audio jack
5. Harmonic producing Load (Electronic devices)
6. PC with MATLAB environment

Procedure:

- Using current sensor, convert the current flowing through load into an equivalent voltage.
- If required, using VDR to convert the voltage as obtained from current sensor to a voltage compatible to audio card (show all the calculations of resistances with their power ratings).
- Set the sampling frequency of the audio card ADC in MATLAB Simulink environment with proper justification
- Plot the acquired current waveform to Simulink scope.
- Mention the safe operating range of your equipment.
- Plot the frequency spectrum of the obtained current and Voltage waveform. Use windowing function to reduce DFT leakage if required.
- Also, plot the frequency spectrum of the line voltage as obtained in open ended lab 01.

Project Summary:

Project Specification:

Attachments:

- Project Block Diagram
- Real Project Image

- Image of current and voltage plot and Spectrum (with proper labelling)

Results:

OUTCOME BASED EDUCATION
LAB RUBRICS
Digital Signal Processing (EE-394)

Student Name: Abdul Moeed **Roll No:** EE-19170 **Section:** D

	Excellent 100%	**Good 75%**	**Average 50%**	**Poor 0%**	**Comments**
Clarity of Concepts	The background theory is fully grasped.	The concepts are not completely cleared.	The theoretical knowledge or concepts are inappropriate.	There is no knowledge of the background theory/concepts.	
Obtaining plot/results	MATLAB program/Simulink model is properly simulated to obtain the desired response.	The logic of the program/model is correct but there is some syntax error to obtain the results	The program/model has some major errors.	The program/model is not correct.	
Submission	Lab task submitted in time.		Late submission of Lab Task		