School of Engineering and Applied Science (SEAS)
Ahmedabad University

BTech(ICT) Semester VI:Digital Signal Processing

Laboratory Assignment-4

Enrollment No:AU1841131 **Name:Mansi Dobariya**

AIM :: LAB4 helps to revise the concept of signal and systems. Solving the properties of signal like Linear Convolution, Circular Convolution, DFT, IDFT, IFFT signal in MATLAB . that's clearly shown in plots to understand better.

1. Solution Problem-1

(a) Matlab Script:

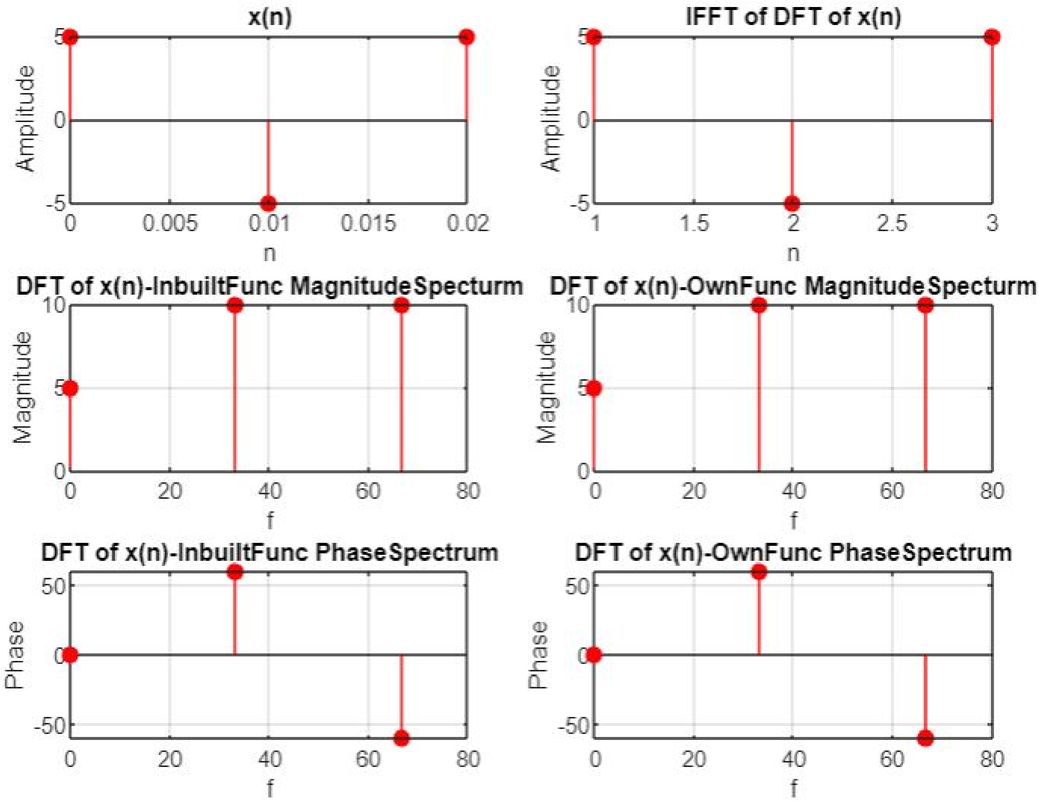
```
clc;
close all;
A=input( 'Enter amplitude:: ' );                    %amplitude
f=input( 'Enter frequency:: ' );                    %frequency
P=input( 'Enter initial phase:: ' );                %phase
fs=input( 'Enter sampling frequency:: ' );          %sampling frequency
cycle=input( 'Enter the number of cycle:: ' );      %no. periods

lower_limit=0;
upper_limit=cycle/f;
t=lower_limit:1/fs:upper_limit;

x=A*cos((2*pi*f).*t + P);                          %input x(n) signal
y_ownFunc=DFT(x);                                  %DFT on x own function
y_inbuiltFunc=fft(x);                              %DFT on x inbuilt function fft

% Magnitude
magnitude_ownFunc = abs(y_ownFunc);
magnitude_inbuiltFunc = abs(y_inbuiltFunc);

% Phase
phase_ownFunc = unwrap(angle(y_ownFunc));
phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));

% vector of freq
freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);

% IFFT
y_ifft = ifft(y_ownFunc);

subplot(3,2,1);
stem(t, x,'fill','r');
hold on;
grid on;
title('x(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,2);
stem(1:length(y_ifft), y_ifft,'fill','r');
hold on;
grid on;
title('IFFT of DFT of x(n)');
xlabel('n');
ylabel('Amplitude');
```

```
subplot(3,2,3);
stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc MagnitudeSpecturm');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,4);
stem(freq_ownFunc, magnitude_ownFunc,'fill','r');
hold on;
grid on;
title('DFT of x(n)-OwnFunc MagnitudeSpecturm');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,5);
stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
xlabel('f');
ylabel('Phase');

subplot(3,2,6);
stem(freq_ownFunc, phase_ownFunc*180/pi,'fill','r');
hold on;
grid on;
title('DFT of x(n)-OwnFunc PhaseSpectrum');
xlabel('f');
ylabel('Phase');

%function for DFT of x(n) for N points
function y=DFT(x, N)
    if nargin < 2
        N=length(x);
    end
    y=zeros(1, max(N, length(x)));
    %k,n is from 1 to k,N, instead of k,n it be k-1,n-1
    for k=1:N
        arg=-2*pi*(k-1)/N;
        for n=1:min(N, length(x))
            ejtheta=exp(arg*(n-1)*1j);
            y(k)=y(k)+(x(n)*ejtheta);
        end
    end
end
```

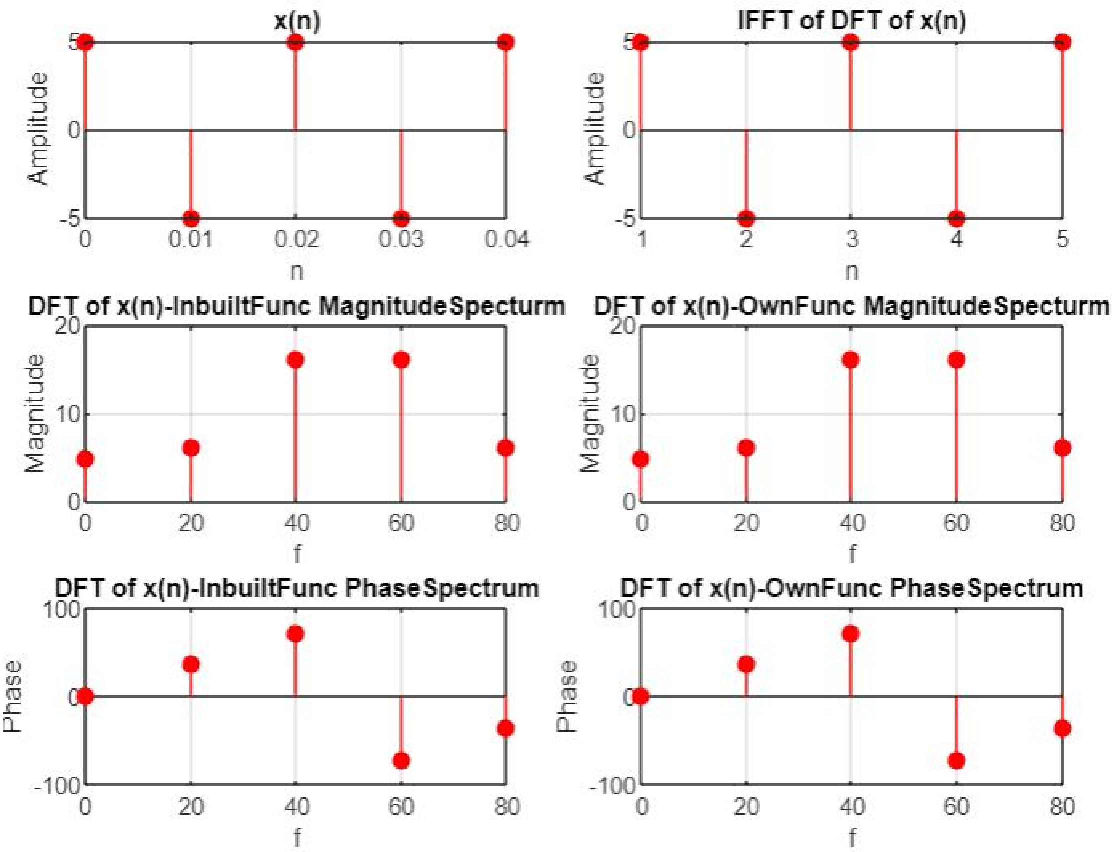
(b) Approach:
DFT possible for ,N length(x), summing from n = 0 to N-1 [x(n)*exp(arg*n*j)] which is equal to y(k); where arg = -2*pi*k/N all k from 0 to N-1,length(y)=N.after Looping from k = 1 to N to get y(k). k-1 instead of k; arg=-2*pi*(k-1)/N; n from 1 to min(N, length(x)). finally,y(k)=y(k)+(x(n)*exp(arg*(n-1)*1j)).

(c) Simulation Output:
Inputs for : A=5; f=50; P=0; fs=2*f=100;

cycle=1


x(n)
IFFT of DFT of x(n)
Amplitude
n
DFT of x(n)-InbuiltFunc MagnitudeSpecturm
DFT of x(n)-OwnFunc MagnitudeSpecturm
Magnitude
f
DFT of x(n)-InbuiltFunc PhaseSpectrum
DFT of x(n)-OwnFunc PhaseSpectrum
Phase

cycle=2


x(n)
IFFT of DFT of x(n)
Amplitude
n
DFT of x(n)-InbuiltFunc MagnitudeSpecturm
DFT of x(n)-OwnFunc MagnitudeSpecturm
Magnitude
f
DFT of x(n)-InbuiltFunc PhaseSpectrum
DFT of x(n)-OwnFunc PhaseSpectrum
Phase

2. **Solution Problem-2**

   (a) Matlab Script for 2(a):

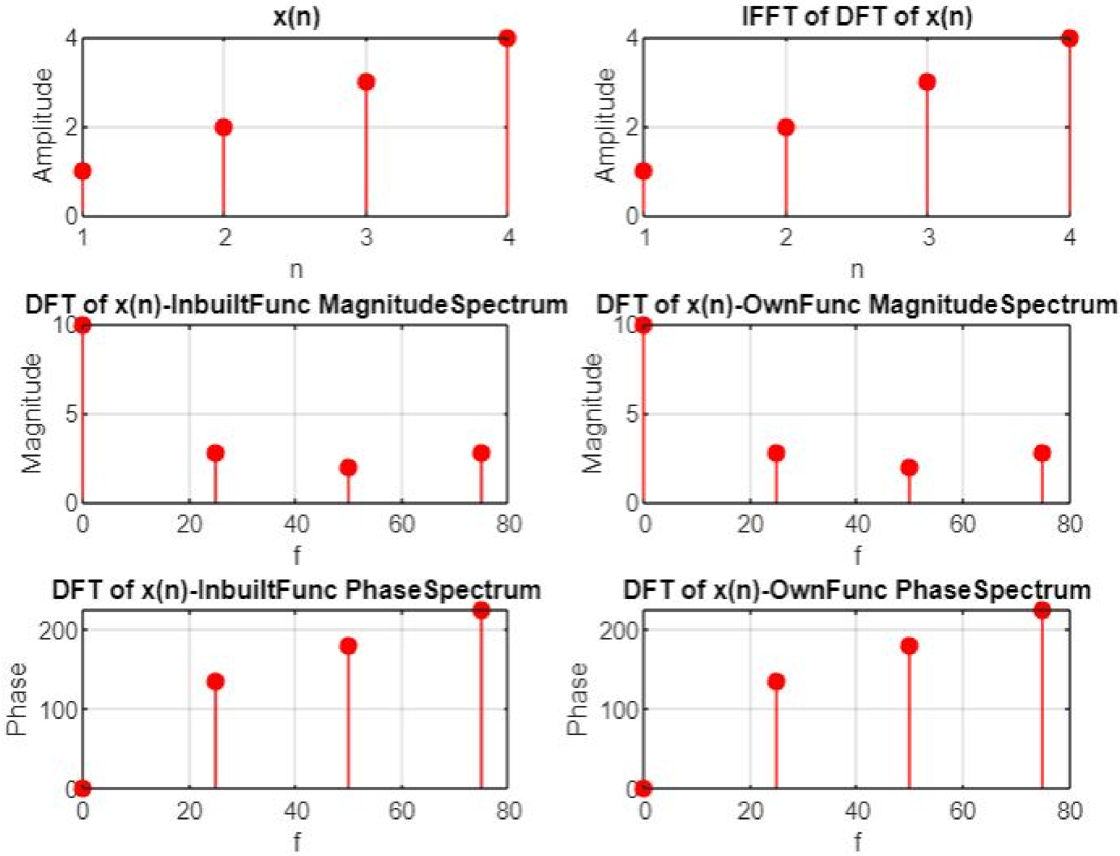
```
clc;
close all;
x=[1 2 3 4];                          %input signal x(n)
N=4;                                  %length of DFT
%N=8;                                 %length of DFT

y_ownFunc=DFT(x,N);                   %DFT on x own  function
y_inbuiltFunc=fft(x,N);               %DFT on x own  function fft

%Magnitude
magnitude_ownFunc = abs(y_ownFunc);
magnitude_inbuiltFunc = abs(y_inbuiltFunc);

% Phase
phase_ownFunc = unwrap(angle(y_ownFunc));
phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));

% vector of freq
freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);

% IFFT
y_ifft = ifft(y_ownFunc,N);

subplot(3,2,1);
stem(1:length(x), x,'fill','r');
hold on;
grid on;
title('x(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,2);
stem(1:length(y_ifft), y_ifft,'fill','r');
hold on;
grid on;
title('IFFT of DFT of x(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,3);
stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc MagnitudeSpectrum');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,4);
stem(freq_ownFunc, magnitude_ownFunc,'fill','r');
hold on;
grid on;
title('DFT of x(n)-OwnFunc MagnitudeSpectrum');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,5);
stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
xlabel('f');
ylabel('Phase');

subplot(3,2,6);
stem(freq_ownFunc, phase_ownFunc*180/pi,'fill','r');
```

```
hold on;
grid on;
title('DFT of x(n)-OwnFunc PhaseSpectrum');
xlabel('f');
ylabel('Phase');

%function for DFT of x(n) for N points
function y=DFT(x, N)
    if nargin < 2
        N=length(x);
    end
    y=zeros(1, max(N, length(x)));
    %k,n is from 1 to k,N, instead  of k,n it be k-1,n-1
    for k=1:N
       arg=-2*pi*(k-1)/N;
        for n=1:min(N, length(x))
            ejtheta=exp(arg*(n-1)*1j);
            y(k)=y(k)+(x(n)*ejtheta);
        end
    end
end

```

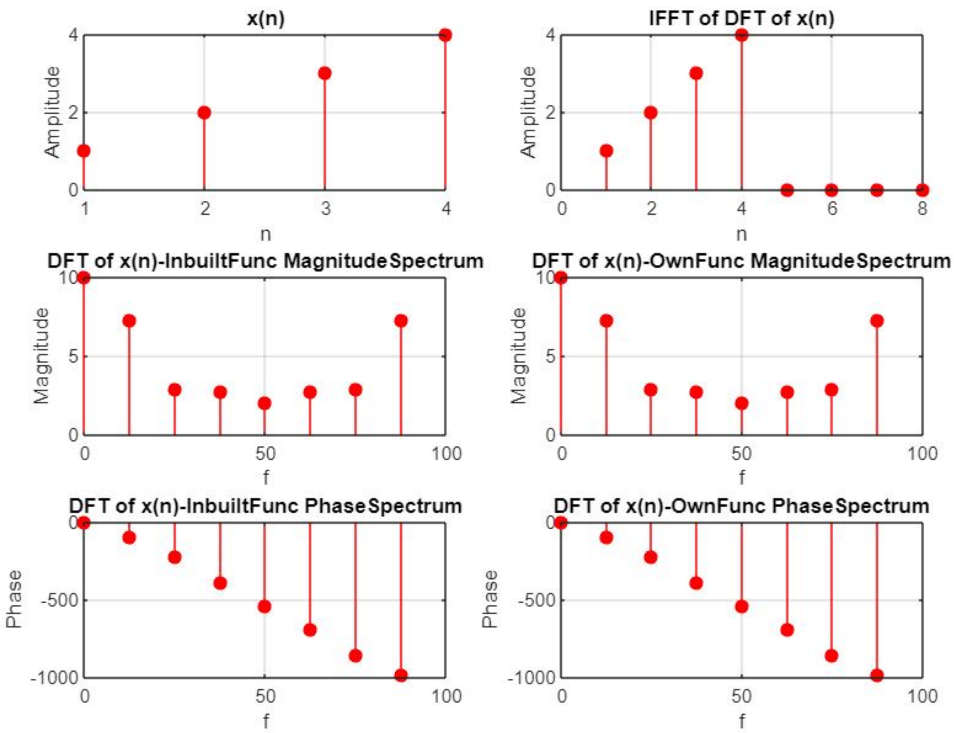
(b) Approach:
own function first argument x and second argument (N) a number of points for the transform,which is DFT length.DFT posssible for N length(x), summing from n = 0 to N-1 [x(n)*exp(arg*n*j)] which is equal to y(k); where arg = -2*pi*k/N all k from 0 to N-1,length(y)=N.after Looping from k = 1 to N to get y(k). k-1 instead of k; arg=-2*pi*(k-1)/N; n from 1 to min(N, length(x)). finally,y(k)=y(k)+(x(n)*exp(arg*(n-1)*1j)).

(c) Simulation Output:

N=4



N=8



(d) Matlab Script for 2(b):

```
clc;
```

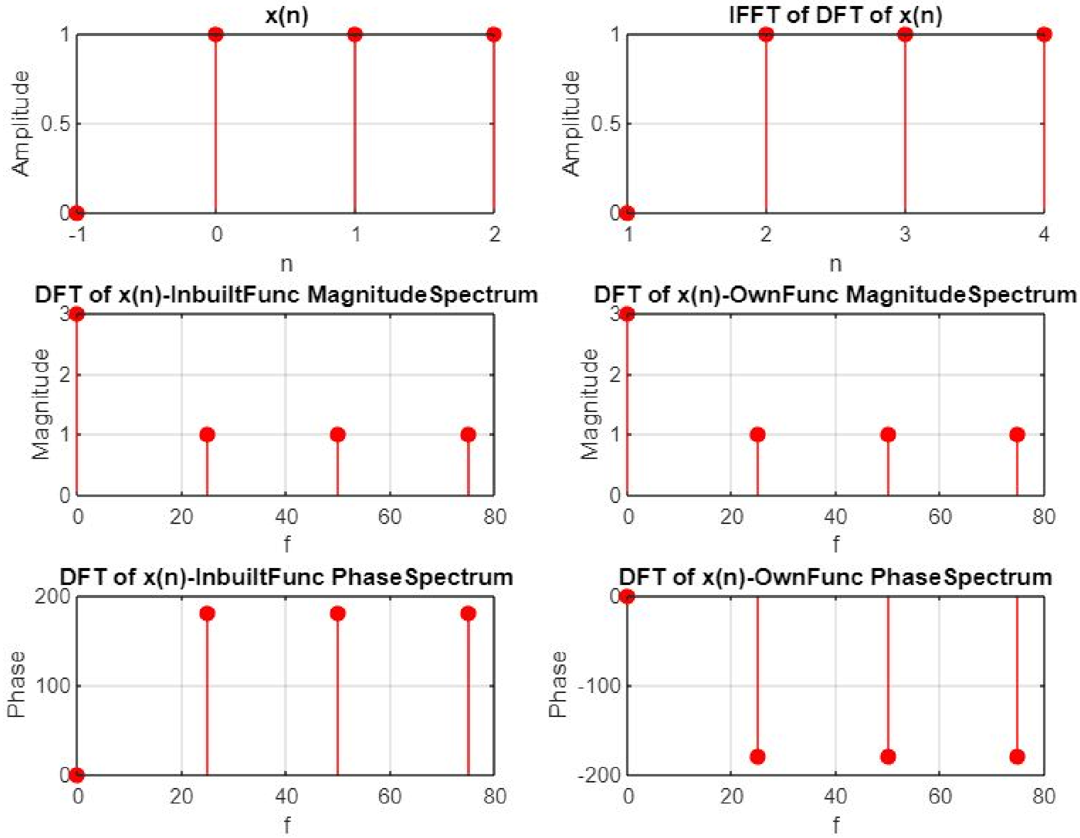
```
close all;
n=-1:1:2;
x=u(n)-u(n-3);                          %input signal

N=4;                                    %length of DFT
%N=8;                                   %length of DFT

y_ownFunc=DFT(x,N);                     %DFT on x own  function
y_inbuiltFunc=fft(x,N);                 %DFT on x own  function fft

%Magnitude
magnitude_ownFunc = abs(y_ownFunc);
magnitude_inbuiltFunc = abs(y_inbuiltFunc);

% Phase
phase_ownFunc = unwrap(angle(y_ownFunc));
phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));

% vector of freq
freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);

% IFFT
y_ifft = ifft(y_ownFunc,N);

subplot(3,2,1);
stem(n, x,'fill','r');
hold on;
grid on;
title('x(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,2);
stem(1:length(y_ifft), y_ifft,'fill','r');
hold on;
grid on;
title('IFFT of DFT of x(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,3);
stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc MagnitudeSpectrum');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,4);
stem(freq_ownFunc, magnitude_ownFunc,'fill','r');
hold on;
grid on;
title('DFT of x(n)-OwnFunc MagnitudeSpectrum');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,5);
stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill','r');
hold on;
grid on;
title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
xlabel('f');
ylabel('Phase');

subplot(3,2,6);
stem(freq_ownFunc, phase_ownFunc*180/pi,'fill','r');
hold on;
grid on;
title('DFT of x(n)-OwnFunc PhaseSpectrum');
```

```
xlabel('f');
ylabel('Phase');

%function for DFT of x(n) for N points
function y=DFT(x, N)
    if nargin < 2
        N=length(x);
    end
    y=zeros(1, max(N, length(x)));
    %k,n is from 1 to k,N, instead  of k,n it be k-1,n-1
    for k=1:N
       arg=-2*pi*(k-1)/N;
        for n=1:min(N, length(x))
            ejtheta=exp(arg*(n-1)*1j);
            y(k)=y(k)+(x(n)*ejtheta);
        end
    end
end
%function for unit step signal ,If current value is greater than 0,output value=1 elsewhere 0
function y = u(n)
    y=zeros(1, length(n));
    for i=1:length(n)
        if n(i) >= 0
            y(i)=1;
        else
            y(i)=0;
        end
    end
end

```

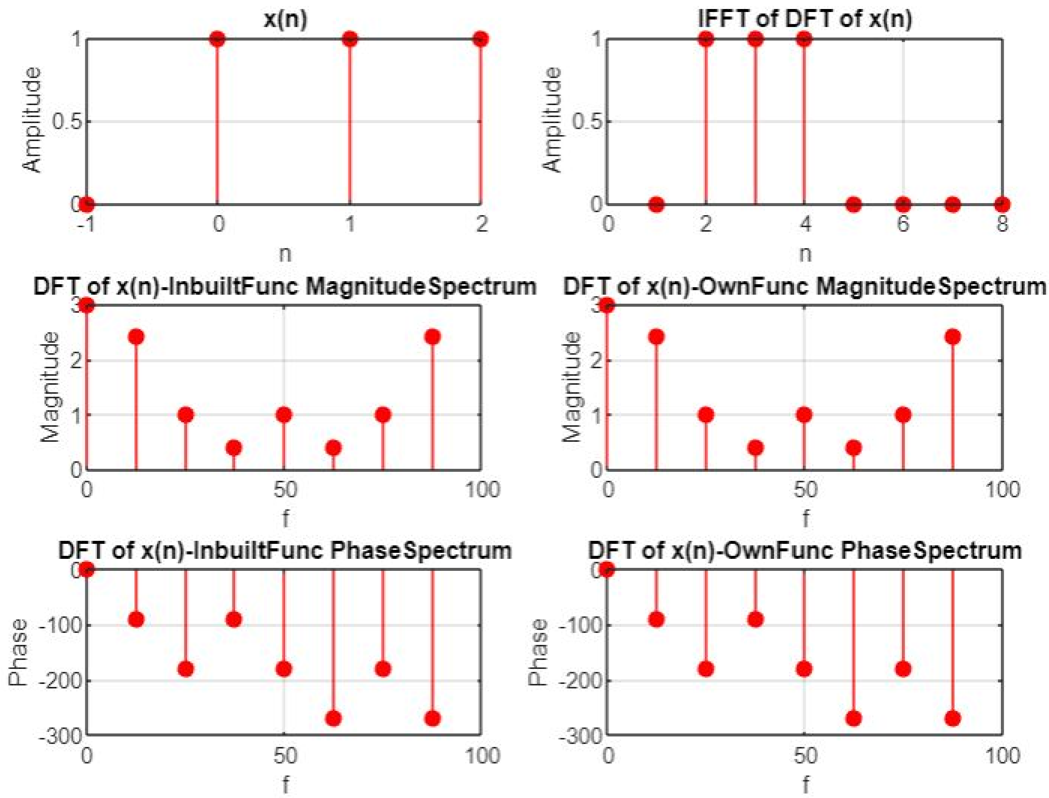
(e) Approach:
own function first argument x and second argument (N) a number of points for the transform,which is DFT length.Calculating Unit step function for input sequence and DFT posssible for N length(x), summing from n = 0 to N-1 [x(n)*exp(arg*n*j)] which is equal to y(k); where arg = -2*pi*k/N all k from 0 to N-1,length(y)=N.after Looping from k = 1 to N to get y(k). k-1 instead of k; arg=-2*pi*(k-1)/N; n from 1 to min(N, length(x)). finally,y(k)=y(k)+(x(n)*exp(arg*(n-1)*1j)).

(f) Simulation Output:

N=4


x(n)
IFFT of DFT of x(n)
DFT of x(n)-InbuiltFunc MagnitudeSpectrum
DFT of x(n)-OwnFunc MagnitudeSpectrum
DFT of x(n)-InbuiltFunc PhaseSpectrum
DFT of x(n)-OwnFunc PhaseSpectrum
Amplitude
Magnitude
Phase
n
f

N=8


x(n)
IFFT of DFT of x(n)
DFT of x(n)-InbuiltFunc MagnitudeSpectrum
DFT of x(n)-OwnFunc MagnitudeSpectrum
DFT of x(n)-InbuiltFunc PhaseSpectrum
DFT of x(n)-OwnFunc PhaseSpectrum
Amplitude
Magnitude
Phase
n
f

3. **Solution Problem-3**

(a) Matlab Script:

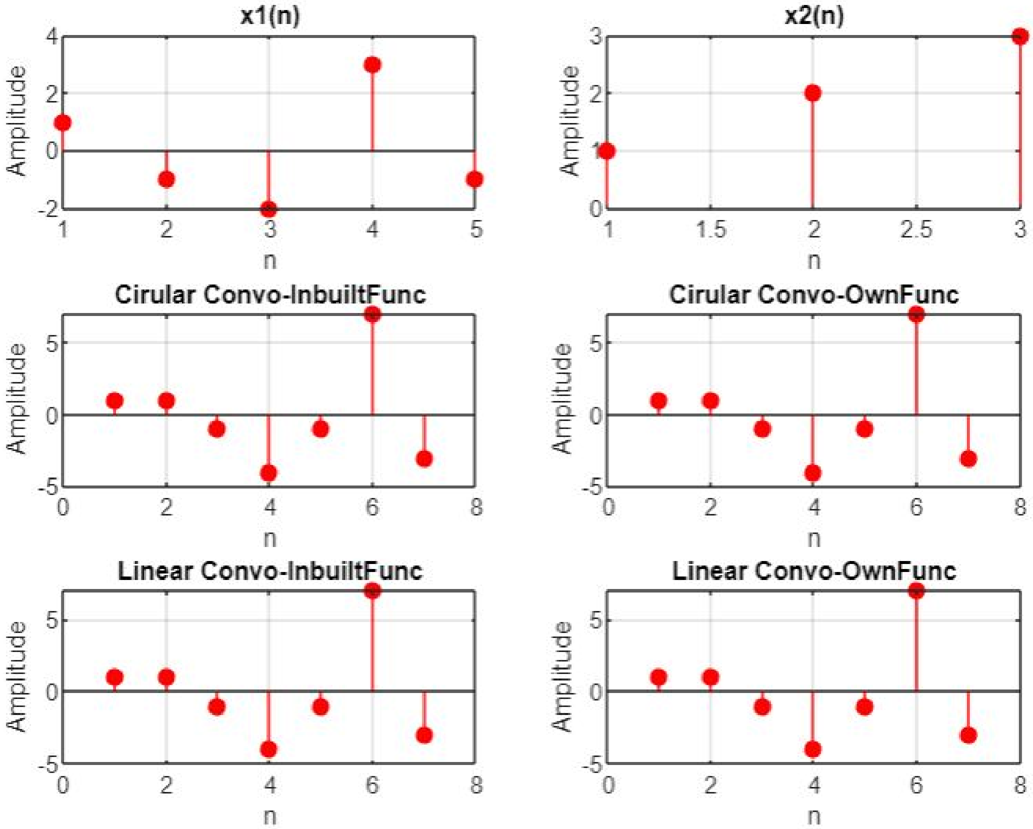
```
% Case 1
x1=[1 -1 -2 3 -1];
x2=[1 2 3];
%N=length(x1)+length(x2)-1;
% Case 2
% x1=[1 2 1 2];
% x2=[3 2 1 4];
%N=length(x1)+length(x2)-1;
% Case 3
% x1=[1 2 3 4];
% x2=u(n)-u(n-3);
%N=-1:1:length(x1);
clc;
close all;

%circular convo
N=length(x1)+length(x2)-1;

cirConvoOwnFunc=cir_convo_ownFunc(x1, x2, N); %CirculaurConvo of x own function
cirConvoInbuiltFunc=cconv([x1 zeros(1, max(0,length(x1)-1))],[x2 zeros(1, max(0,
    length(x2)-1))], N); %CirculaurConvo of x inbuilt function

%linear convo
linConvoOwnFunc=lin_convo_ownFunc(x1, x2, N);%linearConvo of x own function
linConvoInbuiltFunc=conv([x1 zeros(1, max(0,length(x1)-1))],[x2 zeros(1, max(0,
    length(x2)-1))], N); %linearConvo of x inbuilt function

subplot(3,2,1);
stem(1:1:length(x1), x1,'fill','r');
hold on;
grid on;
title('x1(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,2);
stem(1:1:length(x2), x2,'fill','r');
hold on;
grid on;
title('x2(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,3);
stem(1:1:length(cirConvoInbuiltFunc), cirConvoInbuiltFunc, 'fill','r');
hold on;
grid on;
title('Cirular Convo-InbuiltFunc');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,4);
stem(1:1:length(cirConvoOwnFunc), cirConvoOwnFunc,'fill','r');
hold on;
grid on;
title('Cirular Convo-OwnFunc');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,5);
stem(1:1:length(linConvoInbuiltFunc), linConvoInbuiltFunc, 'fill','r');
hold on;
grid on;
title('Linear Convo-InbuiltFunc');
xlabel('n');
ylabel('Amplitude');
```

```
subplot(3,2,6);
stem(1:1:length(linConvoOwnFunc), linConvoOwnFunc,'fill','r');
hold on;
grid on;
title('Linear Convo-OwnFunc');
xlabel('n');
ylabel('Amplitude');

%function for DFT of x(n) for N points
function y=DFT(x, N)
    if nargin < 2
        N=length(x);
    end
    y=zeros(1, max(N, length(x)));
    %k,n is from 1 to k,N, instead  of k,n it be k-1,n-1
    for k=1:N
       arg=-2*pi*(k-1)/N;
        for n=1:min(N, length(x))
            ejtheta=exp(arg*(n-1)*1j);
            y(k)=y(k)+(x(n)*ejtheta);
        end
    end
end

%circular convolution of x(n) using own DFT func
function cirConv=cir_convo_ownFunc(x1, x2, N)
    min_n=length(x1)+length(x2)-1;
    if nargin < 2
        N=min_n;
    end
    cirConv = ifft(DFT(x1, N).*DFT(x2, N));
end

%linear convolution of x(n) using own DFT func
function linConv=lin_convo_ownFunc(x1, x2, N)
    min_n=length(x1)+length(x2)-1;
    if nargin < 2
        N=min_n;
    end
    linConv = ifft(DFT(x1, N).*DFT(x2, N));
end

%function  for  unit  step  signal  ,If  current  value  is  greater  than 0, output  value=1 elsewhere 0
function y = u(n)
    y=zeros(1, length(n));
    for i=1:length(n)
        if n(i) >= 0
            y(i)=1;
        else
            y(i)=0;
        end
    end
end
```

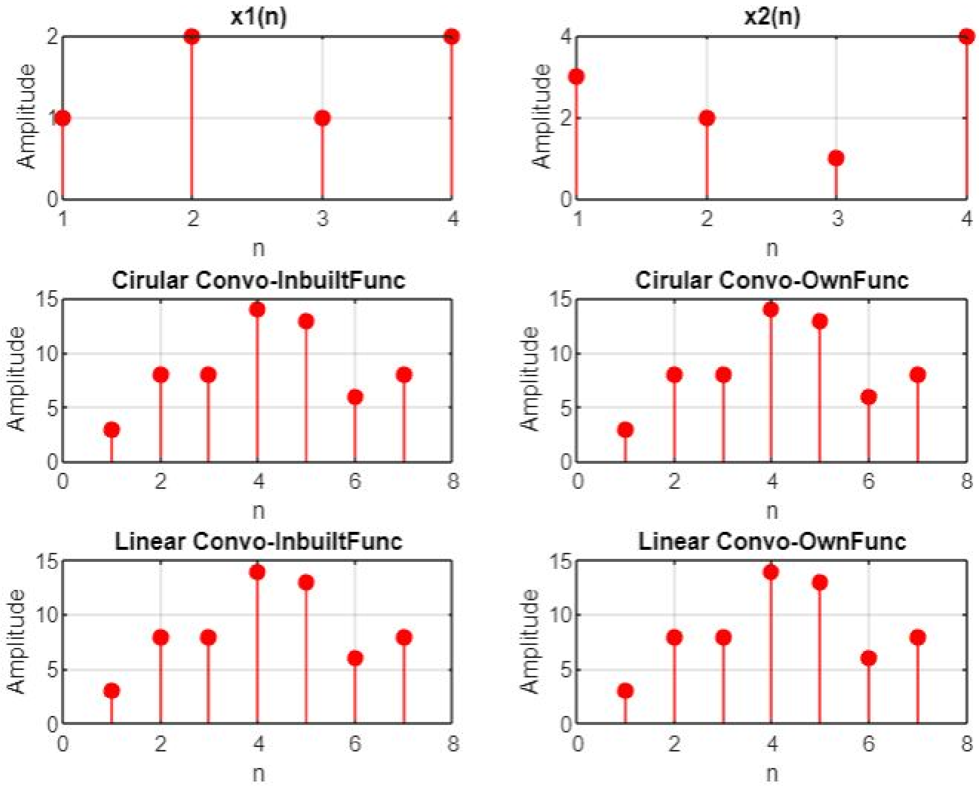
(b) Approach:
linear convolution using circular convolution of x1(n) of length L and x2(n) of length M.they should be made of equal length by appending M-1 zeros to x1(n) and L-1 zeros to x2(n).that's why sequences must be made N, Once the zeros are appended, the N point circular convolution of gives the linear convolution of the sequence.DFT possible for ,N length(x), summing from n = 0 to N-1 [x(n)*exp(arg*n*j)] which is equal to y(k); where arg = -2*pi*k/N all k from 0 to N-1,length(y)=N.after Looping from k = 1 to N to get y(k). k-1 instead of k; arg=-2*pi*(k-1)/N; n from 1 to min(N, length(x)). finally,y(k)=y(k)+(x(n)*exp(arg*(n-1)*1j)).

(c) Simulation Output:

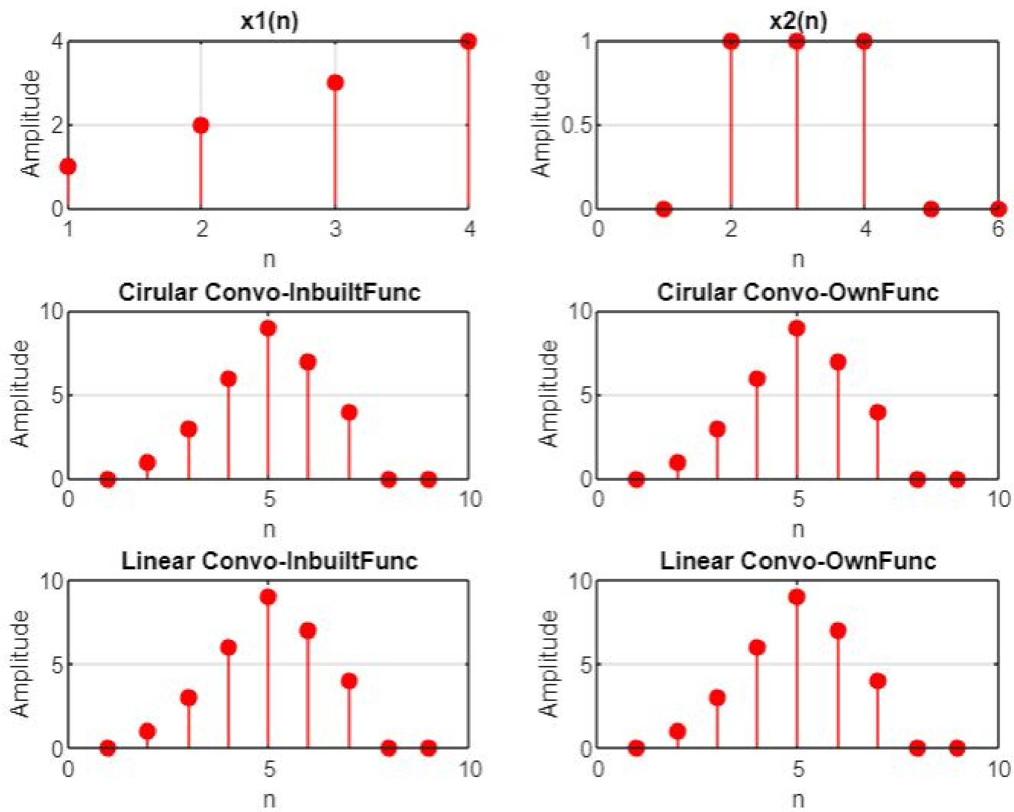
case (a) x1(n) = {1,-1,-2,3,-1} and x2(n) = {1,2,3}



case (b) x1(n) = {1, 2, 1, 2} and x2(n)= {3,2,1,4}



case (c) x1(n) = {1, 2, 3, 4} and x2(n) = u(n)-u(n-3)



4. **Solution Problem-4(a)**

1.[y,Fs] = audioread(filename) reads data from filename, and returns sampled data(y) and a sample rate for that data (Fs)
2.audiowrite(filename,y,Fs) writes a matrix of audio data(y) with sample rate (Fs) to a file called filename.
3.y = filter(b,a,x) filters input x using a rational transfer function coefficients b and a.
4.y = fftshift(x) does a Fourier transform on x and shifts the zero freq component to the center of the array.
5.[b,a] = butter(n,Wn,ftype) designs a diff types of filters lowpass, highpass, bandpass or bandsto.It depends on the value of ftype also number of elements of Wn.

Solution Problem-4(b)

(a) Matlab Script:

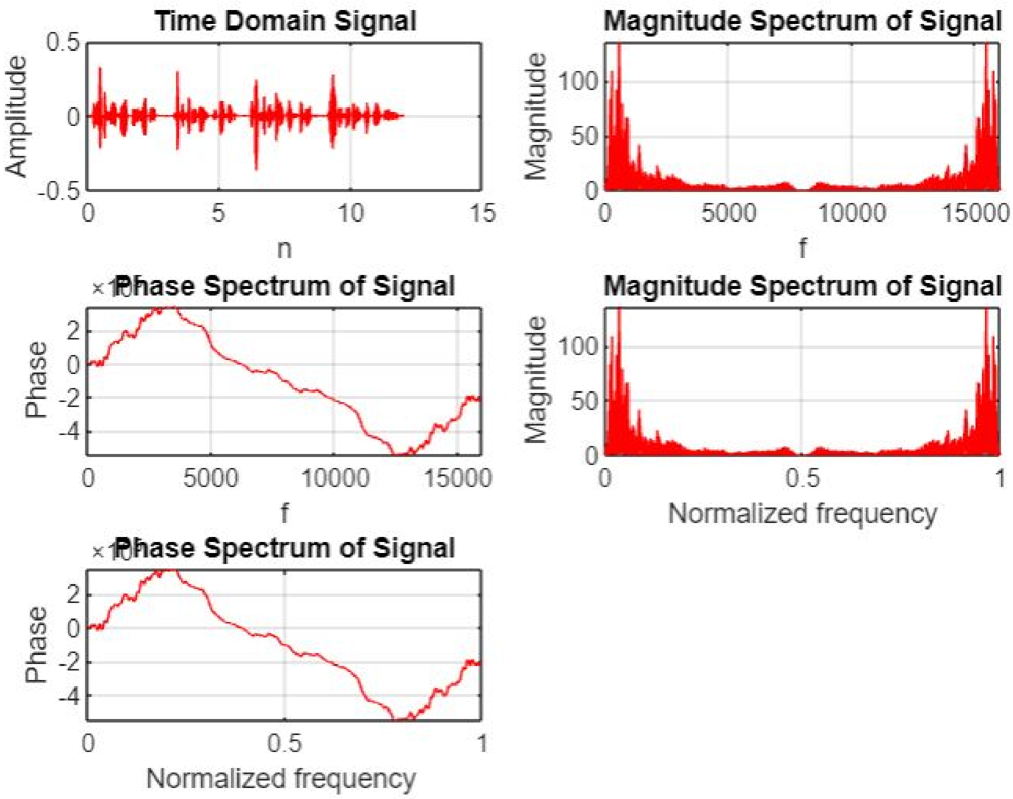
```
clc;
close all;
filename="sample_sound.wav";
[y, fs]=audioread(filename);                % read the audio file
N = length(y);                              % Length of vector y
y_fft = fft(y,N);                           % Fourier transform of y
y_magnitude = abs(y_fft);                   % Magnitude of the FFT of y
y_phase = unwrap(angle(y_fft));             % Phase of the FFT of y
freq_min=0;                                 % minimum frequency
freq_max=(1-1/N)*fs;                        % maximum frequency
df=fs/N;
f = freq_min:df:freq_max;
w = 2*pi*f;                                 % omega
freq_normalized = f/freq_max;               % Normalized frequency
t=(0:1:length(y)-1)/fs;
fprintf("\nThe sampling frequency is %d Hz\n ", fs);
fprintf("\nMinimum frequency is %f Hz and Maximum frequency is %f Hz\n", min(y),
    max(y));
fprintf("\nNormalized frequency is %f Hz\n",max(freq_normalized) );
subplot(3,2,1);
plot(t, y,'r');
hold on;
grid on;
title('Time Domain Signal');
xlabel('n');
ylabel('Amplitude');

subplot(3,2,2);
plot(f, y_magnitude,'r');
hold on;
grid on;
title('Magnitude Spectrum of Signal');
xlabel('f');
ylabel('Magnitude');

subplot(3,2,3);
plot(f, y_phase*180/pi,'r');
hold on;
grid on;
title('Phase Spectrum of Signal');
xlabel('f');
ylabel('Phase');

subplot(3,2,4);
plot(freq_normalized, y_magnitude,'r');
hold on;
grid on;
title('Magnitude Spectrum of Signal');
xlabel('Normalized frequency');
ylabel('Magnitude');

subplot(3,2,5);
plot(freq_normalized, y_phase*180/pi,'r');
```

```
hold on;
grid on;
title('Phase Spectrum of Signal');
xlabel('Normalized frequency ');
ylabel('Phase');

```

(b) Approach:
Reading the audio file using audioread function then FFT on file using fft command. Normalized the frequency like f/maxf and getting magnitude and phase spectrum using abs and angle function.Plotting this values in graph

(c) Simulation Output:

```
The sampling frequency is 16000 Hz

Minimum frequency is -0.366730 Hz and Maximum frequency is 0.328491 Hz

Normalized frequency is 1.000000 Hz
>>
```



Solution Problem-4(c)

(a) Matlab Script:

```
clc;
close all;
filename="sample_sound.wav";
[y, fs]=audioread(filename);                    % read the audio file
N = length(y);                                  % Length of vector y
y_fft = fft(y,N);                               % Fourier transform of y
```

```
y_magnitude = abs(y_fft);                           % Magnitude of the FFT of y
y_phase = unwrap(angle(y_fft));                     % Phase of the FFT of y
freq_min=0;                                         % minimum frequency
freq_max=(1-1/N)*fs;                                % maximum frequency
df=fs/N;
f = freq_min:df:freq_max;
w = 2*pi*f;                                         % omega
freq_normalized = f/freq_max;                       % Normalized frequency
t=(0:1:length(y)-1)/fs;
%Noise
noise=0.5*cos(2*pi*2*t);
yn=zeros(1, length(y));
%Addition of Noise
for idx=1:length(t)
    yn(idx)=y(idx)+noise(idx);
end
filename="sample_sound_with_noise_addition.wav";
audiowrite(filename, yn, fs);
[yn, fsn]=audioread(filename);                      % read the noisy file
[b,a] = butter(3, [0.3 0.7],'bandpass');
% nth-order bandpass digital Butterworth filter; n = 3
y_filteresignal=filter(b, a, yn); % filtering the noisy signal
audiowrite("sample_sound_filtered.wav", y_filteresignal, fsn);

fprintf("\nThe sampling frequency is %d Hz\n", fs);
fprintf("\nMinimum frequency is %f Hz and Maximum frequency is %f Hz\n", min(y),
    max(y));
subplot(2,2,1);
plot(t, y,'r');
hold on;
grid on;
title('Time Domain Signal');
xlabel('n');
ylabel('Amplitude');

subplot(2,2,2);
plot(t, noise,'r');
hold on;
grid on;
title('Noise Signal');
xlabel('n');
ylabel('Amplitude');

subplot(2,2,3);
plot(t, yn,'r');
hold on;
grid on;
title('Original + noise ');
xlabel('n');
ylabel('Amplitude');

subplot(2,2,4);
plot(t, y_filteresignal,'r');
hold on;
grid on;
title('Signal after filtering');
xlabel('n');
ylabel('Amplitude');
```

(b) Approach:
Reading the audio file using audioread function then FFT on file using fft command. Normalized the frequency like f/maxf and getting magnitude and phase spectrum using abs and angle function.Then generate Noisy signal and adding to original signal and take butter filter and filter addition signal .Plotting this values in graph

(c) Simulation Output:

```
The sampling frequency is 16000 Hz

Minimum frequency is -0.366730 Hz and Maximum frequency is 0.328491 Hz
>>
```