

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:

```
1 clc;
2 close all;
3 A=input( 'Enter amplitude:: ' );           %amplitude
4 f=input( 'Enter frequency:: ' );           %frequency
5 P=input( 'Enter initial phase:: ' );        %phase
6 fs=input( 'Enter sampling frequency:: ' ); %sampling frequency
7 cycle=input( 'Enter the number of cycle:: ' ); %no. periods
8
9 lower_limit=0;
10 upper_limit=cycle/f;
11 t=lower_limit:1/fs:upper_limit;
12
13 x=A*cos((2*pi*f).*t + P);                  %input x(n) signal
14 y_ownFunc=DFT(x);                          %DFT on x own function
15 y_inbuiltFunc=fft(x);                      %DFT on x inbuilt function fft
16
17 % Magnitude
18 magnitude_ownFunc = abs(y_ownFunc);
19 magnitude_inbuiltFunc = abs(y_inbuiltFunc);
20
21 % Phase
22 phase_ownFunc = unwrap(angle(y_ownFunc));
23 phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));
24
25 % vector of freq
26 freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
27 freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);
28
29 % IFFT
30 y_ifft = ifft(y_ownFunc);
31
32 subplot(3,2,1);
33 stem(t, x, 'fill', 'r');
34 hold on;
35 grid on;
36 title('x(n)');
37 xlabel('n');
38 ylabel('Amplitude');
39
40 subplot(3,2,2);
41 stem(1:length(y_ifft), y_ifft, 'fill', 'r');
42 hold on;
43 grid on;
44 title('IFFT of DFT of x(n)');
45 xlabel('n');
46 ylabel('Amplitude');
```

```

47
48 subplot(3,2,3);
49 stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill','r');
50 hold on;
51 grid on;
52 title('DFT of x(n)-InbuiltFunc MagnitudeSpectrum');
53 xlabel('f');
54 ylabel('Magnitude');
55
56 subplot(3,2,4);
57 stem(freq_ownFunc, magnitude_ownFunc, 'fill','r');
58 hold on;
59 grid on;
60 title('DFT of x(n)-OwnFunc MagnitudeSpectrum');
61 xlabel('f');
62 ylabel('Magnitude');
63
64 subplot(3,2,5);
65 stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill','r');
66 hold on;
67 grid on;
68 title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
69 xlabel('f');
70 ylabel('Phase');
71
72 subplot(3,2,6);
73 stem(freq_ownFunc, phase_ownFunc*180/pi, 'fill','r');
74 hold on;
75 grid on;
76 title('DFT of x(n)-OwnFunc PhaseSpectrum');
77 xlabel('f');
78 ylabel('Phase');
79
80 %function for DFT of x(n) for N points
81 function y=DFT(x, N)
82     if nargin < 2
83         N=length(x);
84     end
85     y=zeros(1, max(N, length(x)));
86     %k,n is from 1 to k,N, instead of k,n it be k-1,n-1
87     for k=1:N
88         arg=-2*pi*(k-1)/N;
89         for n=1:min(N, length(x))
90             ejtheta=exp(arg*(n-1)*1j);
91             y(k)=y(k)+(x(n)*ejtheta);
92         end
93     end
94 end
95

```

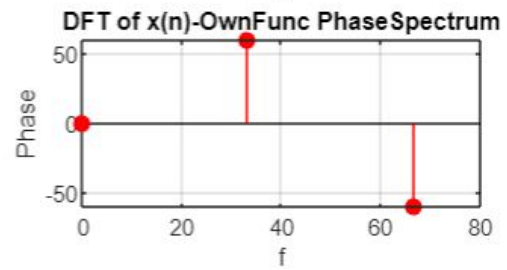
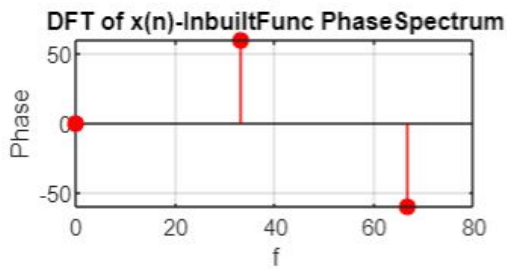
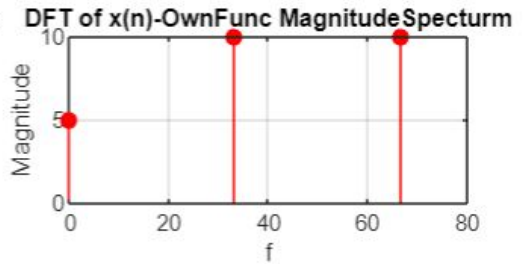
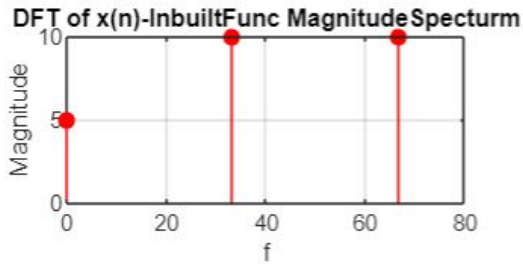
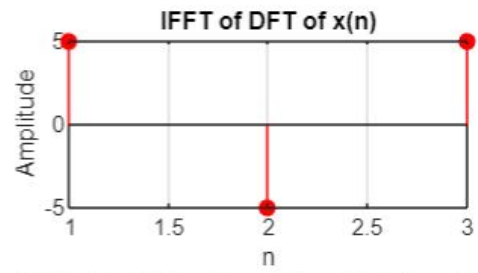
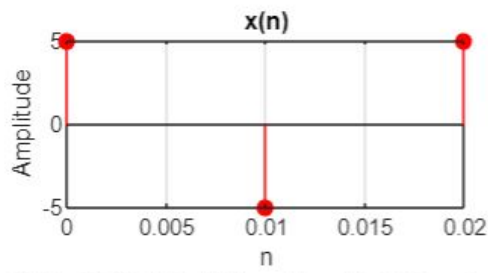
(b) Approach:

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  $N-1$ ,  $\text{length}(y)=N$ . 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  $\min(N, \text{length}(x))$ . finally,  $y(k)=y(k)+(x(n)*\exp(\arg*(n-1)*1j))$ .

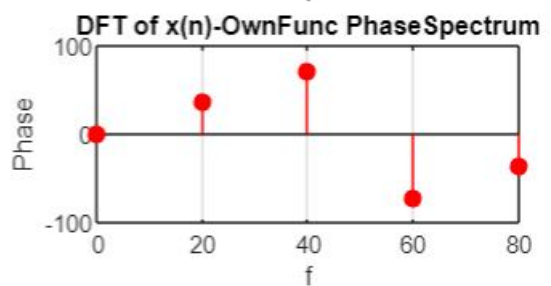
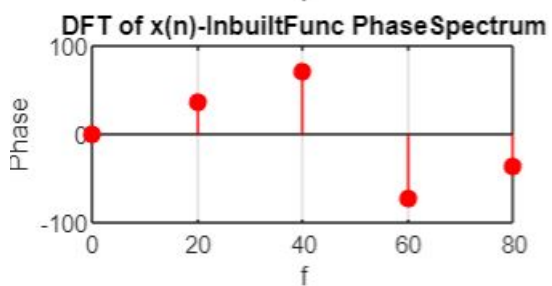
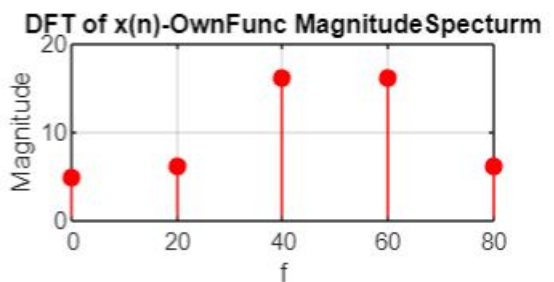
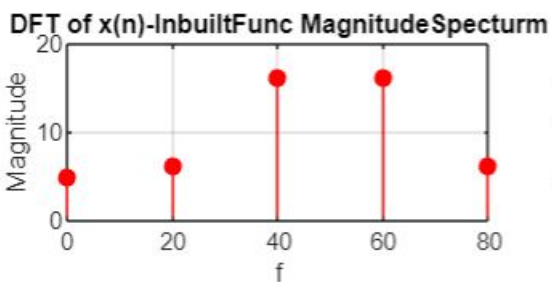
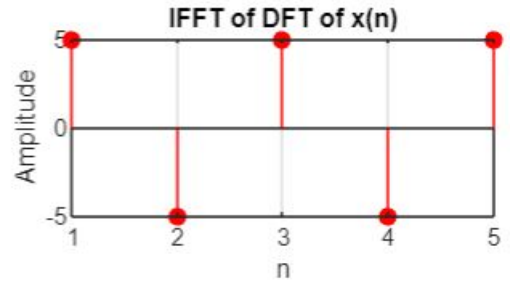
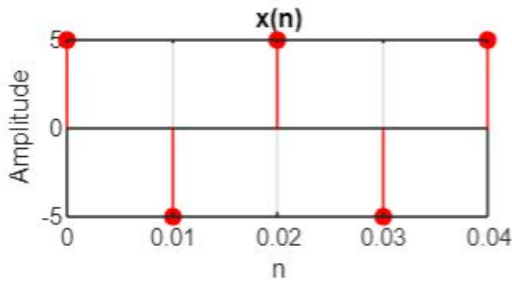
(c) Simulation Output:

Inputs for :  $A=5$ ;  $f=50$ ;  $P=0$ ;  $fs=2*f=100$ ;

cycle=1



cycle=2



## 2. Solution Problem-2

(a) Matlab Script for 2(a):

```
1  clc;
2  close all;
3  x=[1 2 3 4];           %input signal x(n)
4  N=4;                   %length of DFT
5  %N=8;                  %length of DFT
6
7  y_ownFunc=DFT(x,N);    %DFT on x own function
8  y_inbuiltFunc=fft(x,N); %DFT on x own function fft
9
10 %Magnitude
11 magnitude_ownFunc = abs(y_ownFunc);
12 magnitude_inbuiltFunc = abs(y_inbuiltFunc);
13
14 % Phase
15 phase_ownFunc = unwrap(angle(y_ownFunc));
16 phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));
17
18 % vector of freq
19 freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
20 freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);
21
22 % IFFT
23 y_ifft = ifft(y_ownFunc,N);
24
25 subplot(3,2,1);
26 stem(1:length(x), x,'fill','r');
27 hold on;
28 grid on;
29 title('x(n)');
30 xlabel('n');
31 ylabel('Amplitude');
32
33 subplot(3,2,2);
34 stem(1:length(y_ifft), y_ifft,'fill','r');
35 hold on;
36 grid on;
37 title('IFFT of DFT of x(n)');
38 xlabel('n');
39 ylabel('Amplitude');
40
41 subplot(3,2,3);
42 stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill','r');
43 hold on;
44 grid on;
45 title('DFT of x(n)-InbuiltFunc MagnitudeSpectrum');
46 xlabel('f');
47 ylabel('Magnitude');
48
49 subplot(3,2,4);
50 stem(freq_ownFunc, magnitude_ownFunc,'fill','r');
51 hold on;
52 grid on;
53 title('DFT of x(n)-OwnFunc MagnitudeSpectrum');
54 xlabel('f');
55 ylabel('Magnitude');
56
57 subplot(3,2,5);
58 stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill','r');
59 hold on;
60 grid on;
61 title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
62 xlabel('f');
63 ylabel('Phase');
64
65 subplot(3,2,6);
66 stem(freq_ownFunc, phase_ownFunc*180/pi,'fill','r');
```

```

67 hold on;
68 grid on;
69 title('DFT of x(n)-OwnFunc PhaseSpectrum');
70 xlabel('f');
71 ylabel('Phase');
72
73 %function for DFT of x(n) for N points
74 function y=DFT(x, N)
75     if nargin < 2
76         N=length(x);
77     end
78     y=zeros(1, max(N, length(x)));
79     %k,n is from 1 to k,N, instead of k,n it be k-1,n-1
80     for k=1:N
81         arg=-2*pi*(k-1)/N;
82         for n=1:min(N, length(x))
83             ejtheta=exp(arg*(n-1)*1j);
84             y(k)=y(k)+(x(n)*ejtheta);
85         end
86     end
87 end
88

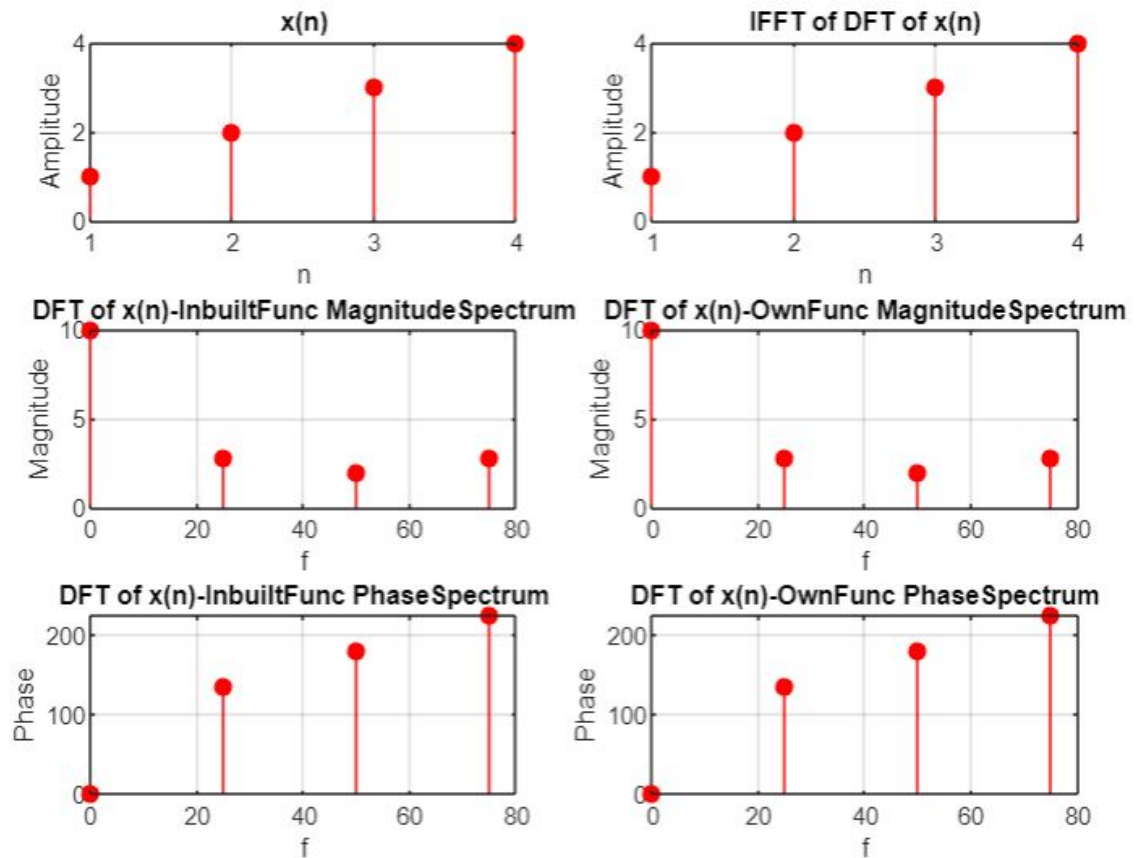
```

(b) Approach:

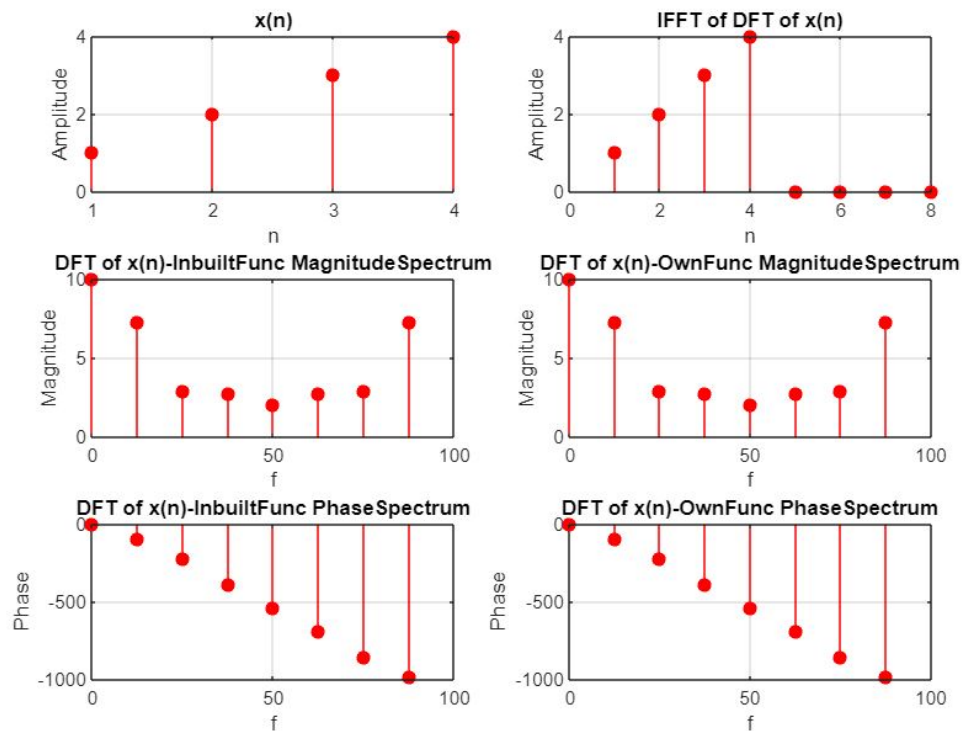
own function first argument x and second argument (N) a number of points for the transform, which is DFT length. DFT possible for N length(x), summing from  $n = 0$  to  $N-1$   $[x(n) \cdot \exp(\arg \cdot n \cdot j)]$  which is equal to  $y(k)$ ; where  $\arg = -2 \cdot \pi \cdot k / N$  all k from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

(c) Simulation Output:

N=4



$N=8$



(d) Matlab Script for 2(b):

```
1 clc;
```

```

2 close all;
3 n=-1:1:2;
4 x=u(n)-u(n-3); %input signal
5
6 N=4; %length of DFT
7 %N=8; %length of DFT
8
9 y_ownFunc=DFT(x,N); %DFT on x own function
10 y_inbuiltFunc=fft(x,N); %DFT on x own function fft
11
12 %Magnitude
13 magnitude_ownFunc = abs(y_ownFunc);
14 magnitude_inbuiltFunc = abs(y_inbuiltFunc);
15
16 % Phase
17 phase_ownFunc = unwrap(angle(y_ownFunc));
18 phase_inbuiltFunc = unwrap(angle(y_inbuiltFunc));
19
20 % vector of freq
21 freq_ownFunc = (0:length(y_ownFunc)-1)*100/length(y_ownFunc);
22 freq_inbuiltFunc = (0:length(y_inbuiltFunc)-1)*100/length(y_inbuiltFunc);
23
24 % IFFT
25 y_ifft = ifft(y_ownFunc,N);
26
27 subplot(3,2,1);
28 stem(n, x, 'fill', 'r');
29 hold on;
30 grid on;
31 title('x(n)');
32 xlabel('n');
33 ylabel('Amplitude');
34
35 subplot(3,2,2);
36 stem(1:length(y_ifft), y_ifft, 'fill', 'r');
37 hold on;
38 grid on;
39 title('IFFT of DFT of x(n)');
40 xlabel('n');
41 ylabel('Amplitude');
42
43 subplot(3,2,3);
44 stem(freq_inbuiltFunc, magnitude_inbuiltFunc, 'fill', 'r');
45 hold on;
46 grid on;
47 title('DFT of x(n)-InbuiltFunc MagnitudeSpectrum');
48 xlabel('f');
49 ylabel('Magnitude');
50
51 subplot(3,2,4);
52 stem(freq_ownFunc, magnitude_ownFunc, 'fill', 'r');
53 hold on;
54 grid on;
55 title('DFT of x(n)-OwnFunc MagnitudeSpectrum');
56 xlabel('f');
57 ylabel('Magnitude');
58
59 subplot(3,2,5);
60 stem(freq_inbuiltFunc, phase_inbuiltFunc*180/pi, 'fill', 'r');
61 hold on;
62 grid on;
63 title('DFT of x(n)-InbuiltFunc PhaseSpectrum');
64 xlabel('f');
65 ylabel('Phase');
66
67 subplot(3,2,6);
68 stem(freq_ownFunc, phase_ownFunc*180/pi, 'fill', 'r');
69 hold on;
70 grid on;
71 title('DFT of x(n)-OwnFunc PhaseSpectrum');

```

```

72 xlabel('f');
73 ylabel('Phase');
74
75 %function for DFT of x(n) for N points
76 function y=DFT(x, N)
77     if nargin < 2
78         N=length(x);
79     end
80     y=zeros(1, max(N, length(x)));
81     %k,n is from 1 to k,N, instead of k,n it be k-1,n-1
82     for k=1:N
83         arg=-2*pi*(k-1)/N;
84         for n=1:min(N, length(x))
85             ejtheta=exp(arg*(n-1)*1j);
86             y(k)=y(k)+(x(n)*ejtheta);
87         end
88     end
89 end
90 %function for unit step signal ,If current value is greater than 0,output value=1
    elsewhere 0
91 function y = u(n)
92     y=zeros(1, length(n));
93     for i=1:length(n)
94         if n(i) >= 0
95             y(i)=1;
96         else
97             y(i)=0;
98         end
99     end
100 end
101

```

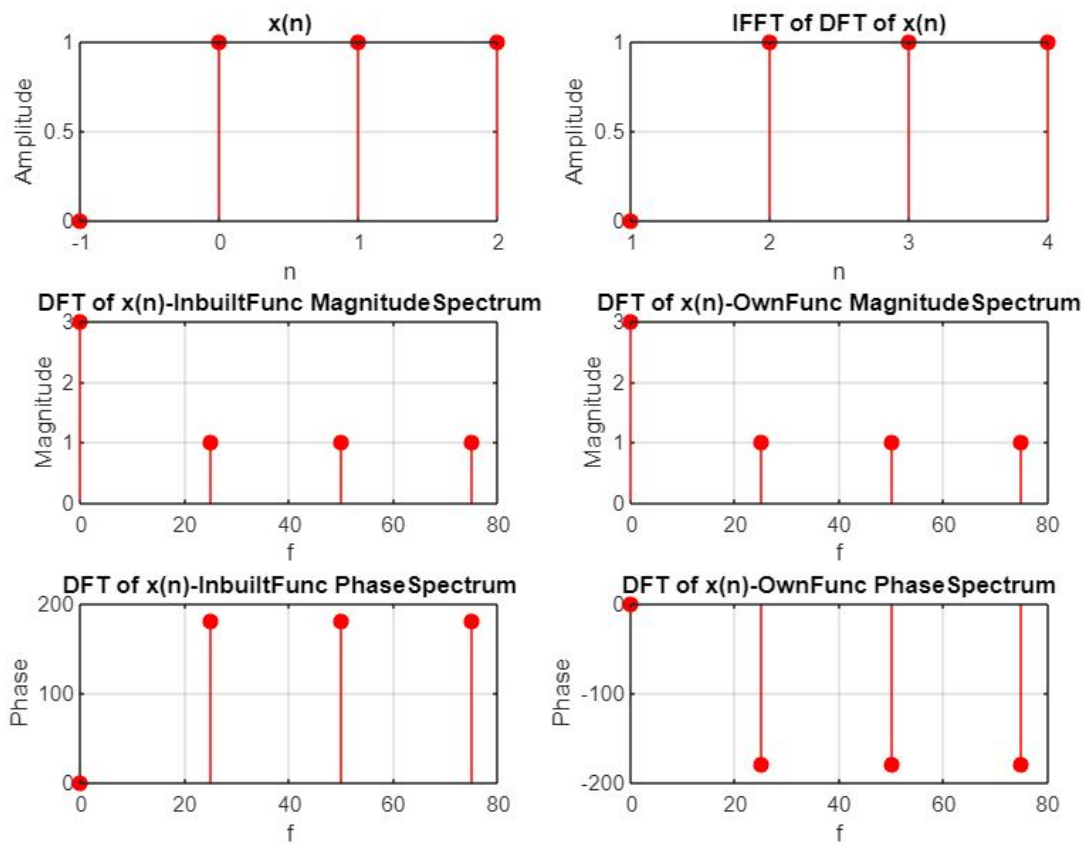
(e) Approach:

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 possible for N length(x), summing from  $n = 0$  to  $N-1$   $[x(n) \cdot \exp(\arg \cdot n \cdot j)]$  which is equal to  $y(k)$ ; where  $\arg = -2\pi \cdot k / N$  all  $k$  from 0 to  $N-1$ ,  $\text{length}(y) = N$ . after Looping from  $k = 1$  to  $N$  to get  $y(k)$ .  $k-1$  instead of  $k$ ;  $\arg = -2\pi \cdot (k-1) / N$ ;  $n$  from 1 to  $\min(N, \text{length}(x))$ . finally,  $y(k) = y(k) + (x(n) \cdot \exp(\arg \cdot (n-1) \cdot j))$ .

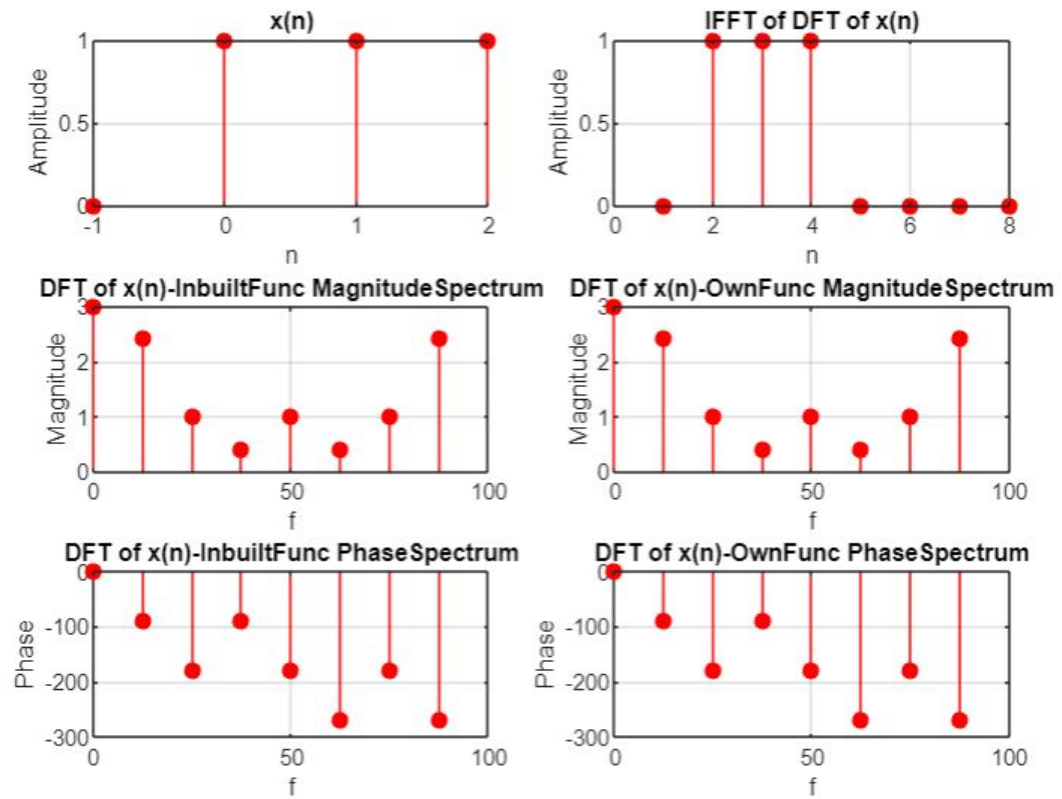
(f) Simulation Output:

$$N=4$$





N=8



### 3. Solution Problem-3

(a) Matlab Script:

```
1 % Case 1
2 x1=[1 -1 -2 3 -1];
3 x2=[1 2 3];
4 %N=length(x1)+length(x2)-1;
5 % Case 2
6 x1=[1 2 1 2];
7 x2=[3 2 1 4];
8 %N=length(x1)+length(x2)-1;
9 % Case 3
10 x1=[1 2 3 4];
11 x2=u(n)-u(n-3);
12 %N=-1:1:length(x1);
13 clc;
14 close all;
15
16 %circular convo
17 N=length(x1)+length(x2)-1;
18
19 cirConvoOwnFunc=cir_convo_ownFunc(x1, x2, N); %CircularConvo of x own function
20 cirConvoInbuiltFunc=cconv([x1 zeros(1, max(0,length(x1)-1))],[x2 zeros(1, max(0,
    length(x2)-1))], N); %CircularConvo of x inbuilt function
21
22 %linear convo
23 linConvoOwnFunc=lin_convo_ownFunc(x1, x2, N);%linearConvo of x own function
24 linConvoInbuiltFunc=conv([x1 zeros(1, max(0,length(x1)-1))],[x2 zeros(1, max(0,
    length(x2)-1))], N); %linearConvo of x inbuilt function
25
26 subplot(3,2,1);
27 stem(1:1:length(x1), x1,'fill','r');
28 hold on;
29 grid on;
30 title('x1(n)');
31 xlabel('n');
32 ylabel('Amplitude');
33
34 subplot(3,2,2);
35 stem(1:1:length(x2), x2,'fill','r');
36 hold on;
37 grid on;
38 title('x2(n)');
39 xlabel('n');
40 ylabel('Amplitude');
41
42 subplot(3,2,3);
43 stem(1:1:length(cirConvoInbuiltFunc), cirConvoInbuiltFunc, 'fill','r');
44 hold on;
45 grid on;
46 title('Circular Convo-InbuiltFunc');
47 xlabel('n');
48 ylabel('Amplitude');
49
50 subplot(3,2,4);
51 stem(1:1:length(cirConvoOwnFunc), cirConvoOwnFunc,'fill','r');
52 hold on;
53 grid on;
54 title('Circular Convo-OwnFunc');
55 xlabel('n');
56 ylabel('Amplitude');
57
58 subplot(3,2,5);
59 stem(1:1:length(linConvoInbuiltFunc), linConvoInbuiltFunc, 'fill','r');
60 hold on;
61 grid on;
62 title('Linear Convo-InbuiltFunc');
63 xlabel('n');
64 ylabel('Amplitude');
```

```

65
66 subplot(3,2,6);
67 stem(1:length(linConvoOwnFunc), linConvoOwnFunc,'fill','r');
68 hold on;
69 grid on;
70 title('Linear Convo-OwnFunc');
71 xlabel('n');
72 ylabel('Amplitude');
73
74 %function for DFT of x(n) for N points
75 function y=DFT(x, N)
76     if nargin < 2
77         N=length(x);
78     end
79     y=zeros(1, max(N, length(x)));
80     %k,n is from 1 to k,N, instead of k,n it be k-1,n-1
81     for k=1:N
82         arg=-2*pi*(k-1)/N;
83         for n=1:min(N, length(x))
84             ejtheta=exp(arg*(n-1)*1j);
85             y(k)=y(k)+(x(n)*ejtheta);
86         end
87     end
88 end
89
90 %circular convolution of x(n) using own DFT func
91 function cirConv=cir_convo_ownFunc(x1, x2, N)
92     min_n=length(x1)+length(x2)-1;
93     if nargin < 2
94         N=min_n;
95     end
96     cirConv = ifft(DFT(x1, N).*DFT(x2, N));
97 end
98
99 %linear convolution of x(n) using own DFT func
100 function linConv=lin_convo_ownFunc(x1, x2, N)
101     min_n=length(x1)+length(x2)-1;
102     if nargin < 2
103         N=min_n;
104     end
105     linConv = ifft(DFT(x1, N).*DFT(x2, N));
106 end
107
108 %function for unit step signal ,If current value is greater than 0,
109     output value=1 elsewhere 0
110 function y = u(n)
111     y=zeros(1, length(n));
112     for i=1:length(n)
113         if n(i) >= 0
114             y(i)=1;
115         else
116             y(i)=0;
117         end
118     end
119 end

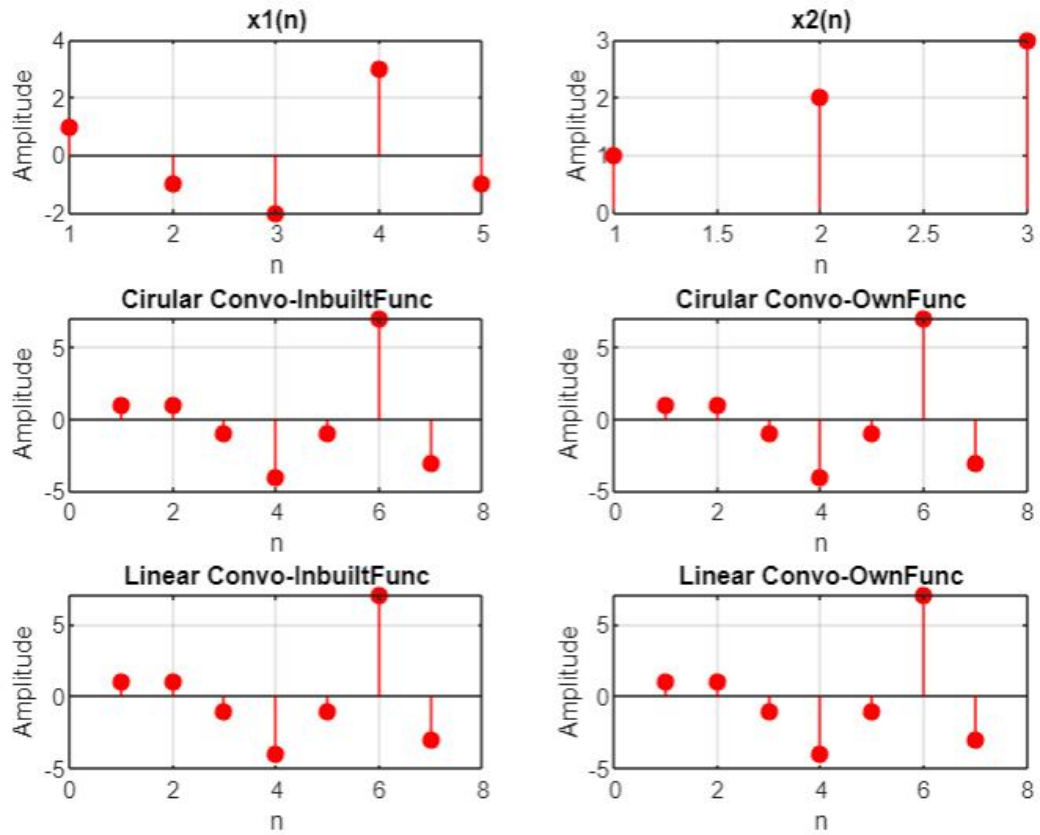
```

(b) Approach:

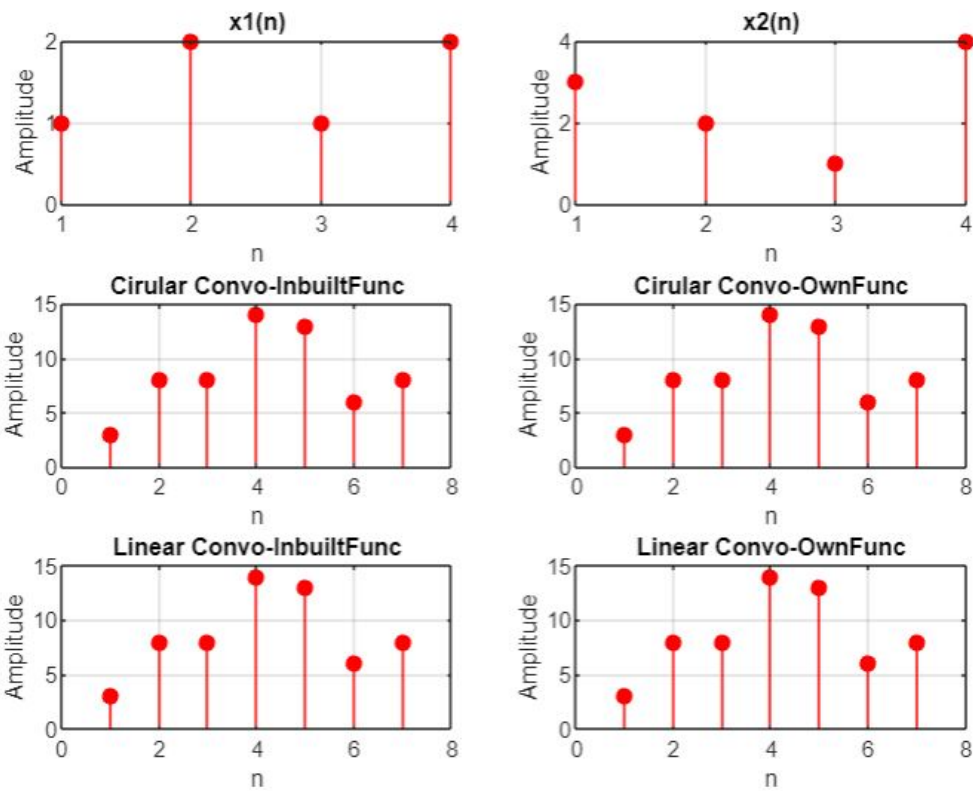
linear convolution using circular convolution of  $x_1(n)$  of length  $L$  and  $x_2(n)$  of length  $M$ . they should be made of equal length by appending  $M-1$  zeros to  $x_1(n)$  and  $L-1$  zeros to  $x_2(n)$ . that's why sequences must be made  $N$ , Once the zeros are appended, the  $N$  point circular convolution of gives the linear convolution of the sequence. DFT possible for  $N$ ,  $N = \text{length}(x)$ , summing from  $n = 0$  to  $N-1$   $[x(n) \cdot \exp(\arg \cdot n \cdot j)]$  which is equal to  $y(k)$ ; where  $\arg = -2\pi \cdot k / N$  all  $k$  from  $0$  to  $N-1$ ,  $\text{length}(y) = N$ . after Looping from  $k = 1$  to  $N$  to get  $y(k)$ .  $k-1$  instead of  $k$ ;  $\arg = -2\pi \cdot (k-1) / N$ ;  $n$  from  $1$  to  $\min(N, \text{length}(x))$ . finally,  $y(k) = y(k) + (x(n) \cdot \exp(\arg \cdot (n-1) \cdot j))$ .

(c) Simulation Output:

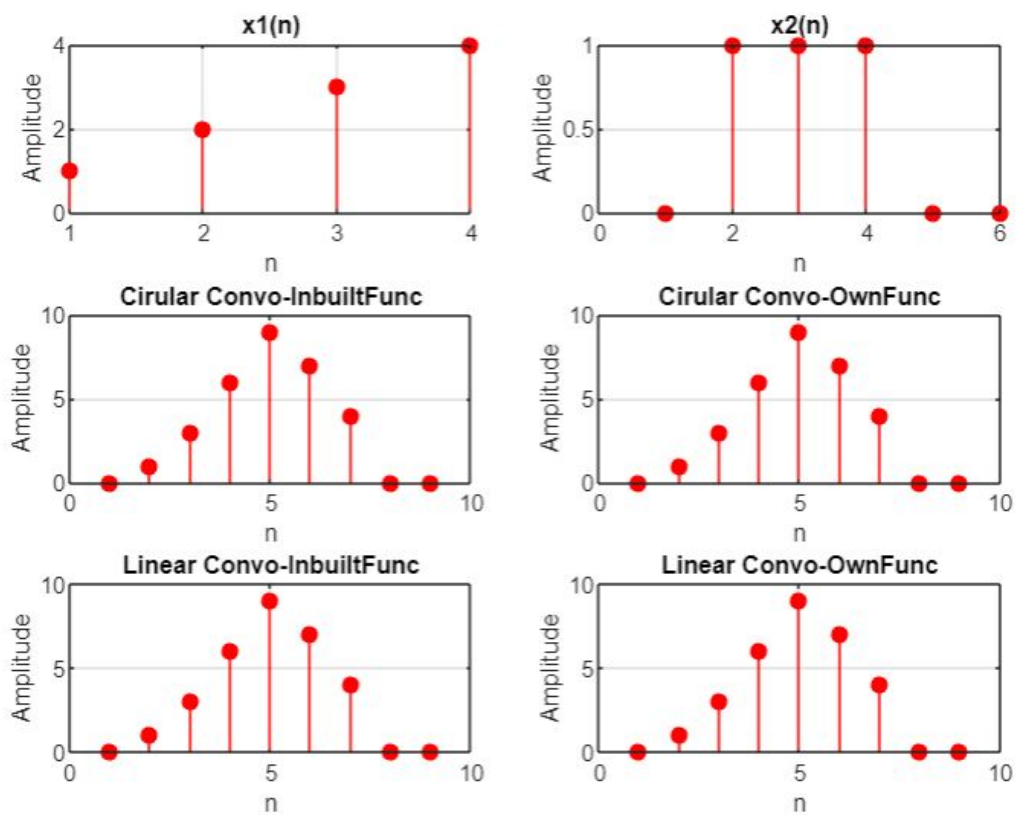
case (a)  $x_1(n) = \{1, -1, -2, 3, -1\}$  and  $x_2(n) = \{1, 2, 3\}$



case (b)  $x_1(n) = \{1, 2, 1, 2\}$  and  $x_2(n) = \{3, 2, 1, 4\}$



case (c)  $x_1(n) = \{1, 2, 3, 4\}$  and  $x_2(n) = u(n) - u(n-3)$



#### 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 bandstop. It depends on the value of ftype also number of elements of Wn.

#### Solution Problem-4(b)

(a) Matlab Script:

```

1  clc;
2  close all;
3  filename="sample_sound.wav";
4  [y, fs]=audioread(filename);           % read the audio file
5  N = length(y);                         % Length of vector y
6  y_fft = fft(y,N);                     % Fourier transform of y
7  y_magnitude = abs(y_fft);              % Magnitude of the FFT of y
8  y_phase = unwrap(angle(y_fft));        % Phase of the FFT of y
9  freq_min=0;                            % minimum frequency
10 freq_max=(1-1/N)*fs;                   % maximum frequency
11 df=fs/N;
12 f = freq_min:df:freq_max;
13 w = 2*pi*f;                            % omega
14 freq_normalized = f/freq_max;           % Normalized frequency
15 t=(0:1:length(y)-1)/fs;
16 fprintf("\nThe sampling frequency is %d Hz\n", fs);
17 fprintf("\nMinimum frequency is %f Hz and Maximum frequency is %f Hz\n", min(y),
    max(y));
18 fprintf("\nNormalized frequency is %f Hz\n",max(freq_normalized) );
19 subplot(3,2,1);
20 plot(t, y, 'r');
21 hold on;
22 grid on;
23 title('Time Domain Signal');
24 xlabel('n');
25 ylabel('Amplitude');
26
27 subplot(3,2,2);
28 plot(f, y_magnitude, 'r');
29 hold on;
30 grid on;
31 title('Magnitude Spectrum of Signal');
32 xlabel('f');
33 ylabel('Magnitude');
34
35 subplot(3,2,3);
36 plot(f, y_phase*180/pi, 'r');
37 hold on;
38 grid on;
39 title('Phase Spectrum of Signal');
40 xlabel('f');
41 ylabel('Phase');
42
43 subplot(3,2,4);
44 plot(freq_normalized, y_magnitude, 'r');
45 hold on;
46 grid on;
47 title('Magnitude Spectrum of Signal');
48 xlabel('Normalized frequency');
49 ylabel('Magnitude');
50
51 subplot(3,2,5);
52 plot(freq_normalized, y_phase*180/pi, 'r');

```

```

53 hold on;
54 grid on;
55 title('Phase Spectrum of Signal');
56 xlabel('Normalized frequency ');
57 ylabel('Phase');
58

```

(b) Approach:

Reading the audio file using audioread function then FFT on file using fft command. Normalized the frequency like  $f/\text{maxf}$  and getting magnitude and phase spectrum using abs and angle function. Plotting this values in graph

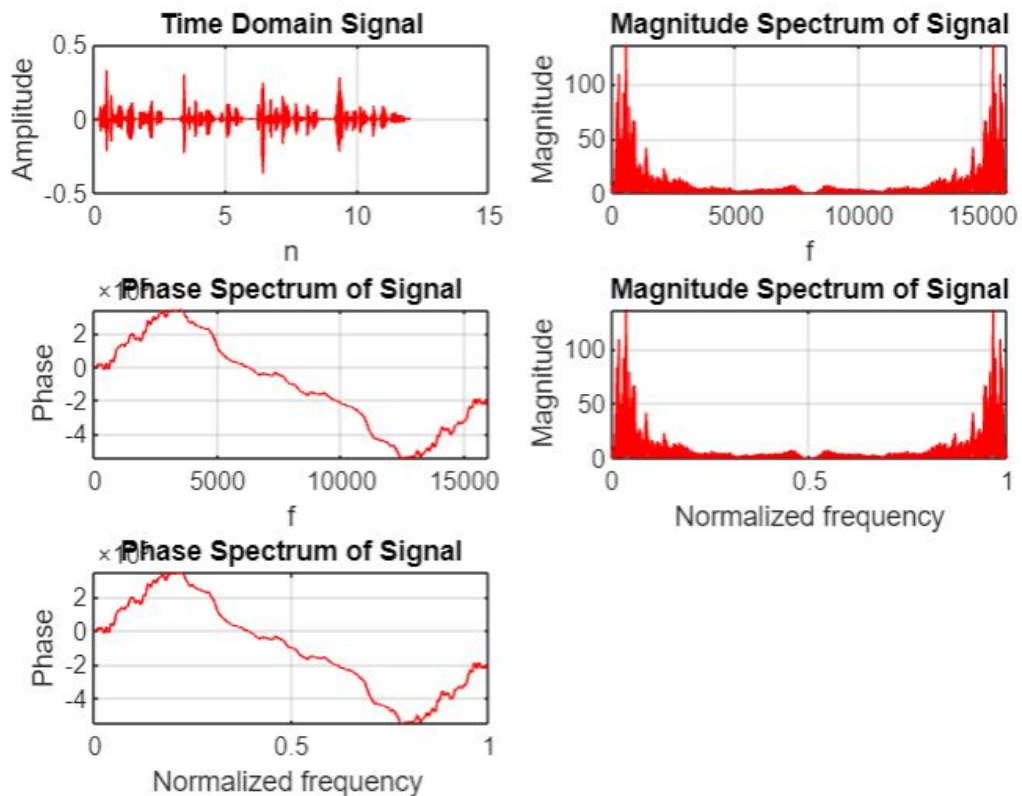
(c) Simulation Output:

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:

```

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

```

```

7  y_magnitude = abs(y_fft); % Magnitude of the FFT of y
8  y_phase = unwrap(angle(y_fft)); % Phase of the FFT of y
9  freq_min=0; % minimum frequency
10 freq_max=(1-1/N)*fs; % maximum frequency
11 df=fs/N;
12 f = freq_min:df:freq_max;
13 w = 2*pi*f; % omega
14 freq_normalized = f/freq_max; % Normalized frequency
15 t=(0:1:length(y)-1)/fs;
16 %Noise
17 noise=0.5*cos(2*pi*2*t);
18 yn=zeros(1, length(y));
19 %Addition of Noise
20 for idx=1:length(t)
21     yn(idx)=y(idx)+noise(idx);
22 end
23 filename="sample_sound_with_noise_addition.wav";
24 audiowrite(filename, yn, fs);
25 [yn, fsn]=audioread(filename); % read the noisy file
26 [b,a] = butter(3, [0.3 0.7], 'bandpass');
27 % nth-order bandpass digital Butterworth filter; n = 3
28 y_filteresignal=filter(b, a, yn); % filtering the noisy signal
29 audiowrite("sample_sound_filtered.wav", y_filteresignal, fsn);
30
31 fprintf("\nThe sampling frequency is %d Hz\n", fs);
32 fprintf("\nMinimum frequency is %f Hz and Maximum frequency is %f Hz\n", min(y),
33     max(y));
34 subplot(2,2,1);
35 plot(t, y, 'r');
36 hold on;
37 grid on;
38 title('Time Domain Signal');
39 xlabel('n');
40 ylabel('Amplitude');
41
42 subplot(2,2,2);
43 plot(t, noise, 'r');
44 hold on;
45 grid on;
46 title('Noise Signal');
47 xlabel('n');
48 ylabel('Amplitude');
49
50 subplot(2,2,3);
51 plot(t, yn, 'r');
52 hold on;
53 grid on;
54 title('Original + noise ');
55 xlabel('n');
56 ylabel('Amplitude');
57
58 subplot(2,2,4);
59 plot(t, y_filteresignal, 'r');
60 hold on;
61 grid on;
62 title('Signal after filtering');
63 xlabel('n');
64 ylabel('Amplitude');

```

(b) Approach:

Reading the audio file using audioread function then FFT on file using fft command. Normalized the frequency like  $f/\max f$  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

>>

