



VISVESVARAYA NATIONAL INSTITUTE
OF TECHNOLOGY (VNIT), NAGPUR

**Digital Signal Processing
(ECL304)**

Lab Report

Submitted by :

Bipasha Parui (BT19ECE019)

Semester 4

Submitted to :

Dr. Deep Gupta and Dr. Ankit A. Bhurane

(Course Instructors)

Department of Electronics and Communication Engineering,
VNIT Nagpur

Contents

| | | |
|-----|---|----|
| 1 | Assignment -1: A digital filter is described by $y(n) - 2.56y(n-1) + 2.22y(n-2) - 0.65y(n-3) = x(n) + x(n-3)$ where $x(n)$ is the input and $y(n)$ is the output. Assume all zero initial conditions | 3 |
| 2 | Assignment -2: | 31 |
| 2.1 | Q1: Generate a 10-kHz sinusoid sampled at 100 kHz. Plot four cycles of this signal. Also plot the spectrum of this signal in the interval $(-\pi, \pi)$ | 31 |
| 2.2 | Q2: Decimate this signal by a factor of 2. Plot the time-domain signal. Also plot the spectrum of the decimated signal in the interval $(-,)$. Comment on your results. What frequency does represent now ? | 34 |
| 2.3 | Q3: Upsample the original sinusoidal sequence by a factor of 3. Plot the time domain signal. Also plot the spectrum of the upsampled signal in the interval $(-,)$. Comment on your results. What frequency does represent now? | 36 |
| 3 | Assignment -3: | 39 |
| 3.1 | Q1):Write a program to compute the N-point DFT of a sequence. Do not use the inbuilt command. Write your own routine of the DFT. | 39 |
| 3.2 | Q2):Use your program to find the 200-point DFT of the following sequences: (i) $x(n) = 2\cos(2n/10) + \cos(2n/5)$ (ii) $x(n) = n$ Plot the magnitude response of these sequences. The horizontal axis must be scaled appropriately to represent the interval $(0,)$. Also Comment on the validity of your results. | 40 |

| | | |
|-----|--|----|
| 3.3 | Q3: Write a program to compute the N-point inverse-DFT (IDFT) of a sequence. DO not use the inbuilt command. Test your program on the DFTs of the sequences in the previous part. | 45 |
| 3.4 | Q4: Write a program to compute the circular convolution of two length-N sequences via the DFT approach. Use your DFT and IDFT programs for this purpose. Use your programs (using calling function) to find the circular convolution of the following pairs of sequences: (i) $x(n) = [1; 3; -2; 4; 7]$, $h(n) = [3; 1; 21; -3]$ (ii) $x(n) = n$; $h(n) = (0.5)n$, $0 \leq n \leq 10$ | 50 |
| 3.5 | Q5: Write a program to perform circular convolution in the time domain. Test your program on the previous sequences given in part (d). | 54 |
| 4 | Assignment -4: | 57 |
| 4.1 | Q1: Implement the equation of Analog Butterworth Low Pass Filter Approximation and observe the effect of order N on the magnitude response. Plot the curves for the Bilinear Transformation and it's inverse so as to compensate the frequency warping effect. Prepare a succinct and original report of your own based these tasks. | 57 |
| 4.2 | Q2 - With reference to the IIR LP filter example discussed in the class, generate two sinusoids one within passband and other out of passband, add them and pass through the filter as designed in the class. Plot the input and output signals. . . | 59 |
| 4.3 | Q3 - Create a generic filter without inbuilt function. | 61 |
| 4.4 | Q4- Implement a BPF given the following specifications $wls = 0.1\pi$, $wlp = 0.4\pi$, $whs = 0.9\pi$, $whp = 0.6\pi$ | 62 |
| 4.5 | Conclusion: | 63 |

Assignment -1:

A digital filter is described by

$$y(n) - 2.56y(n-1) + 2.22y(n-2) - 0.65y(n-3) = x(n) + x(n-3)$$

where $x(n)$ is the input and $y(n)$ is the output. Assume all zero initial conditions

Section a): Generate the input signal $x(n)$, which is a sinusoid of frequency 500 Hz sampled at 6 kHz.

Code:

```

1 % Code for input x[n] sinusoid creation
2 clear
3
4 F = 500; % Sinusoid frequency ( in Hz -- Cycles per second)
5 Fs = 6000; % Sampling Frequency ( in Hz --Samples per second )
6 n = 100; % no. of samples
7 % At n= Fs/F= 12(i.e no.of samples required per cycle) we get a ...
   single i/p cycle
8 t = linspace(0,n/Fs,n); % n/Fs = no. of seconds per 100 sample
9
10 x_n = sin(2 * pi * t * F); % input sinusoid
11
12 % coefficients in numerator(input signals in difference equation)
13 num = [1 0 0 1];
14 % coefficients in denominator(output signals in difference equation)
15 den = [1 -2.56 2.22 -0.65];
16
17 plot(t, x_n, 'LineWidth', 2);
18 grid on;
19 xlabel('time in seconds', 'FontSize', 15);
20 ylabel('Amplitude ', 'FontSize', 15);

```

Observations and Discussions: It can be observed from Fig. 2 and 1, that the time period of the sinusoid is approximately

$$2 \times 10^{-3} s$$

Conclusions: Since Frequency(F) is 500 Hz(cycles per second) and we are sampling(Fs) at 6 kHz(samples per second) , the output sinusoid repeats itself at rate of 12 samples/cycle i.e $6000 / 500$. Now if 6000 Samples take 1 second 12 samples take $12/6000 = 0.002$ seconds . I.e time period of input $x[n]$ is 0.002 seconds.

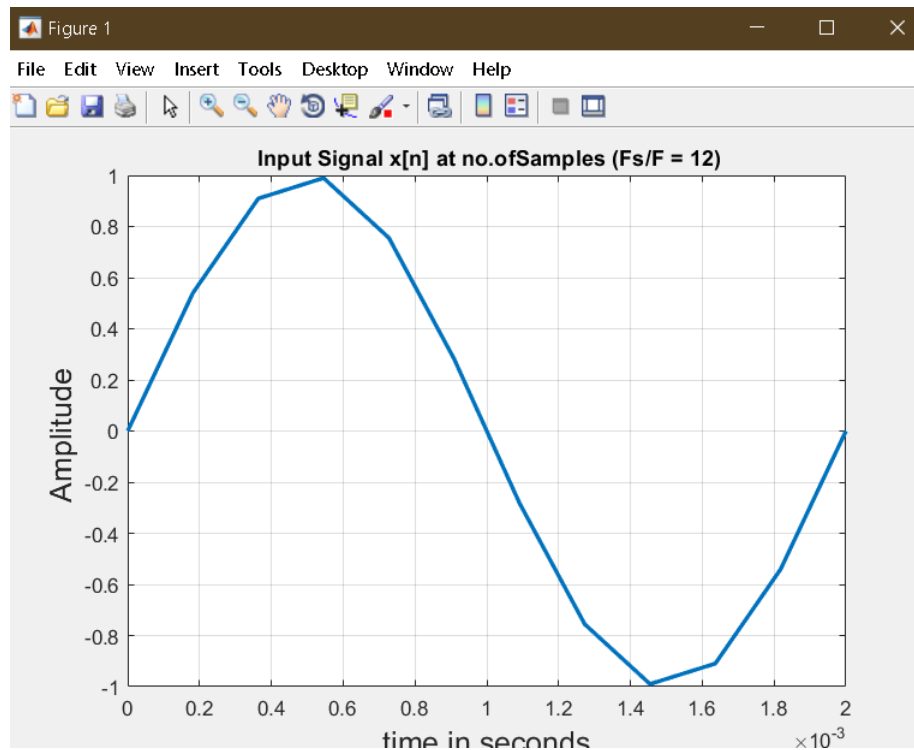


Figure 1: Sampled Sinusoid with 1 Cycle .

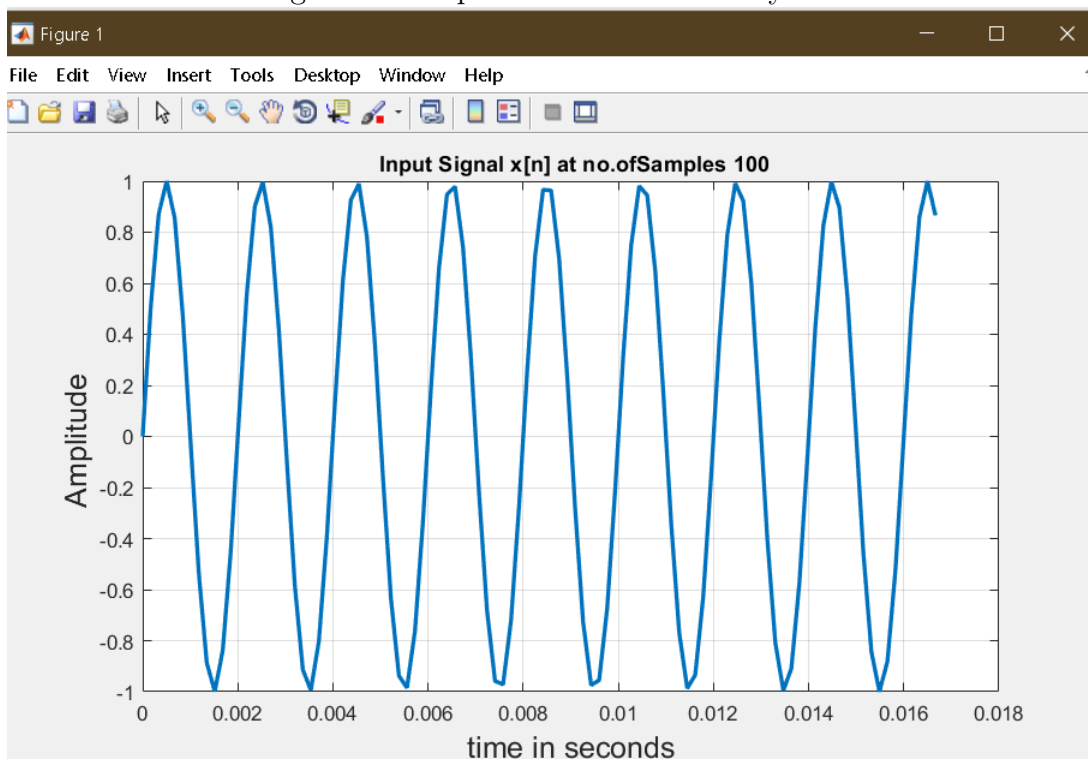


Figure 2: Sampled Sinusoid .

Section b): Compute the first four cycles of the output by directly implementing the above difference equation. Plot the input and output on the same graph.

Code:

```

1 % Code for output signal from given digital filter using filter()
2 clear
3 F = 500;           % Sinusoid frequency ( Hz - Cycles/ second)
4 Fs = 6000;         % Sampling Frequency ( Hz -Samples/ second )
5 n = 48;           % no. of samples
6 % 4 cycles = 12X4 = 48 samples
7 t = linspace(0,n/Fs,n);
8 x_n = sin(2*pi*t*F); % input sinusoid
9 y_n = zeros(1,n);    % Defining Output leny[n] = lenx[n]
10 y_n(1:3) = [0 0 0]; % Given Zero initial conditions
11
12 % using loops to get value for each y[k] k E (4,5..48)
13 for k = 4:n
14     y_n(k) = x_n(k)+ x_n(k-3) + 2.56*y_n(k-1) - 2.22*y_n(k-2) + ...
        0.65*y_n(k-3);
15 end
16
17 plot(t,y_n,'LineWidth',2)
18 grid on
19 hold on
20 plot(t,x_n,'LineWidth',2)
21 title("Input x[n],Output y[n] without 'Filter' function ...
        (no.ofsamples = 48)");
22 xlabel("time (t) in seconds");
23 ylabel("Amplitude");
24
25 legend("y[n]","x[n]");

```

Observations and Discussions: 2 points can be observed from Fig. 3

1. Though the input signal is periodic, the output signal is aperiodic.
2. The amplitude of the resultant signal is higher than input signal.

Conclusions: Since it has been established that the input sinusoid $x[n]$ is periodic at a rate of 12samples/cycle. In order to generate a signal with approximate 4 cycle we have used **12x4 = 48 samples**. However since the output is **aperiodic** we cannot define its cycle or time period.

Here the first 3 values of output signal $y[n]$ which forms the initial conditions is 0

$$y[0] = y[1] = y[3] = 0$$

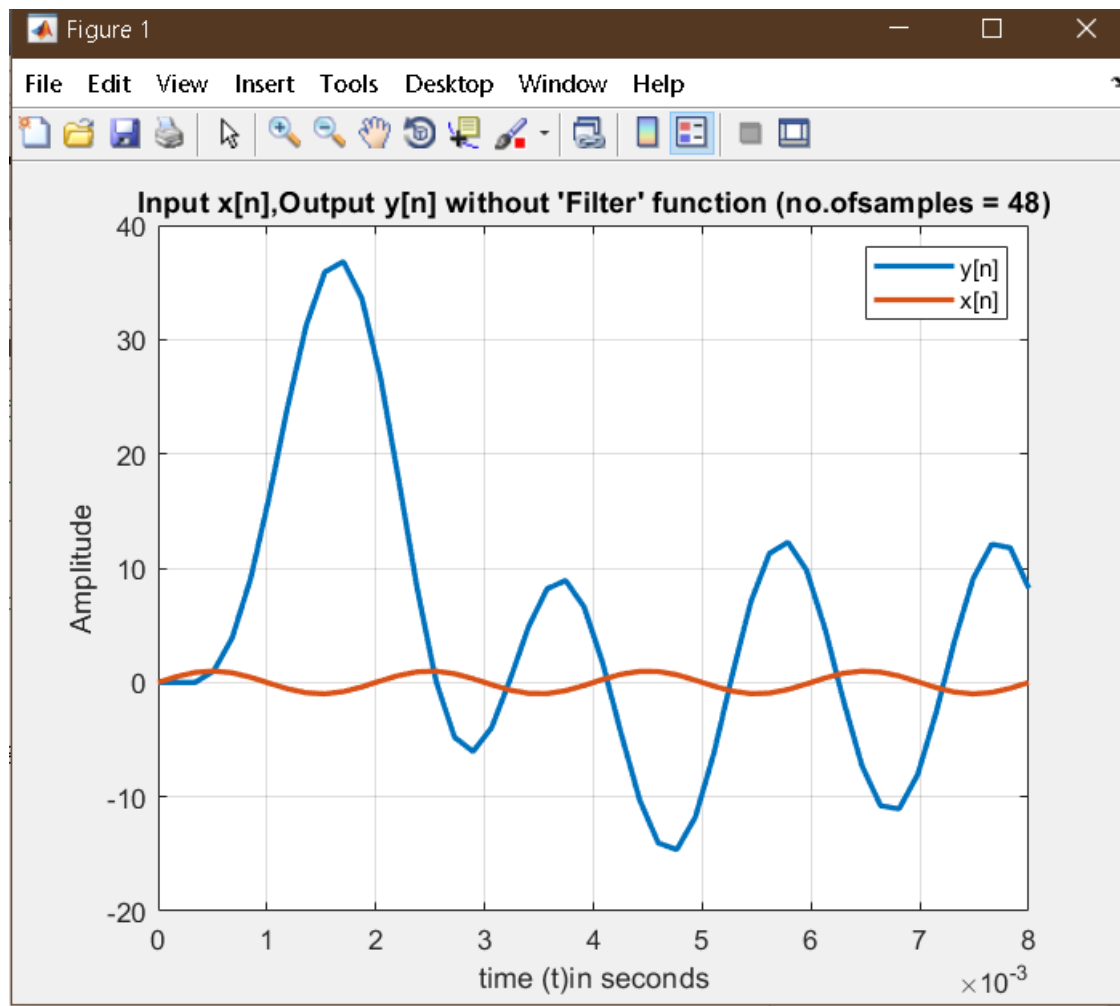


Figure 3: Output Signal from given Sinusoidal Input Signal passed through Digital filter (using loops).

. For the rest values i.e $y[k]$ $k = 4$ to 48 the code iterative implements the difference function.

Section c): Implement the above filter by using MATLAB “filter” function. Compare your results with part(b). Comment on your result.

Code:

```

27 % Code for output signal from given digital filter using filter()
28 clear
29 F = 500;           % Sinusoid frequency ( Hz - Cycles/ second)
30 Fs = 6000;        % Sampling Frequency ( Hz -Samples/ second )
31 n = 48;           % no. of samples
32 % 4 cycles = 12X4 = 48 samples
33 t = linspace(0,n/Fs,n);
34 x_n = sin(2*pi*t*F); % input sinusoid
35
36 num = [1 0 0 1]; %Coefficients of x[n-k]
37 den = [1 -2.56 2.22 -0.65]; %Coefficients of y[n-k]
38 y_n = filter(num,den,x_n); %INbuilt function to solve difference ...
    equation
39
40 plot(t, x_n, '-c', 'LineWidth', 2);
41 grid on;
42 title("x[n],y[n] with Filter function (n=48)");
43 xlabel('time(t) in seconds', 'FontSize', 15);
44 ylabel('Amplitude', 'FontSize', 15);
45 hold on;
46 plot(t, y_n, '-m', 'LineWidth', 2);
47 grid on;
48 legend("x[n]", "y[n]");

```

Observations and Discussions: It can be observed from 4 that the output signal obtained from using inbuilt function is **almost** same as observed in Section b) done manually. However Certain deviations can be seen. For example

1. The maximul amplitude in both -ve and +ve y direction is higher (approx 50,-15) than in output signal in previous section (approx 37,-6).
2. Due to initial condition in section B the signals remains at amplitude 0 till 3 points however in section C

Conclusions: Though slight differences the nature and shape of the output in Section b and c are same . Also here the coefficients num and den defined in the code correspond to ”**feed-forward**” and ”**feed-backward**” coefficients in the form

$$y[n] + \sum_{k=1}^M a_k y[n-k] = \sum_{k=0}^N b_k x[n-k]$$

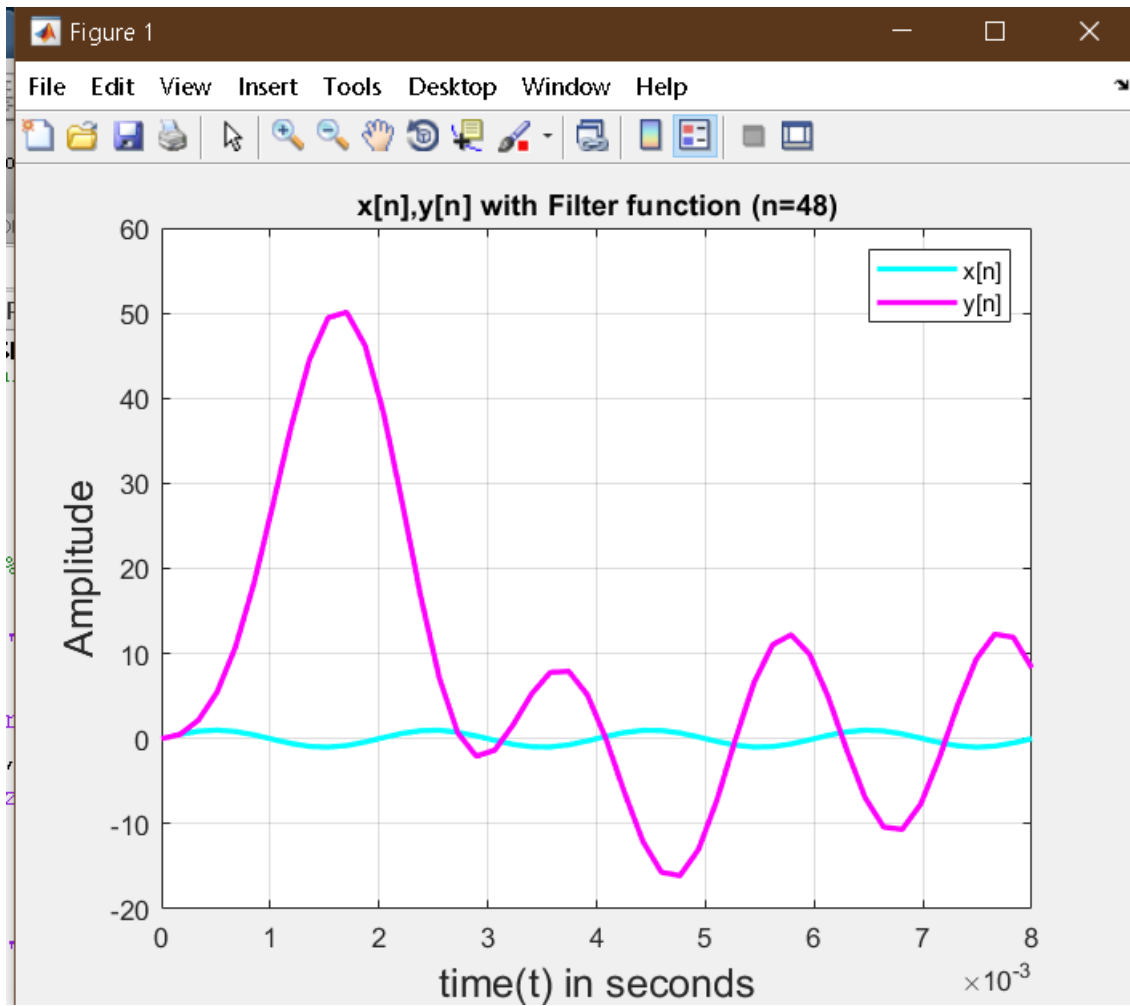


Figure 4: Output Signal from given Sinusoidal Input Signal passed through Digital filter (using MATLAB filter()).

Section d): Plot the impulse response of the filter by using MATLAB “impz” function. Comment on the results.

Code:

```
50 %Code for Impulse Response
51 num = [1 0 0 1];           %Feed forward Coefficients
52 den = [1 -2.56 2.22 -0.65]; %Feed Backward Coefficients
53
54 [h,t] = impz(num,den);
55
56 plot(h, 'LineWidth',2);
57 grid on
58 xlabel("Samples");
59 ylabel("Amplitude");
60 title("Impulse response of digital filter");
```

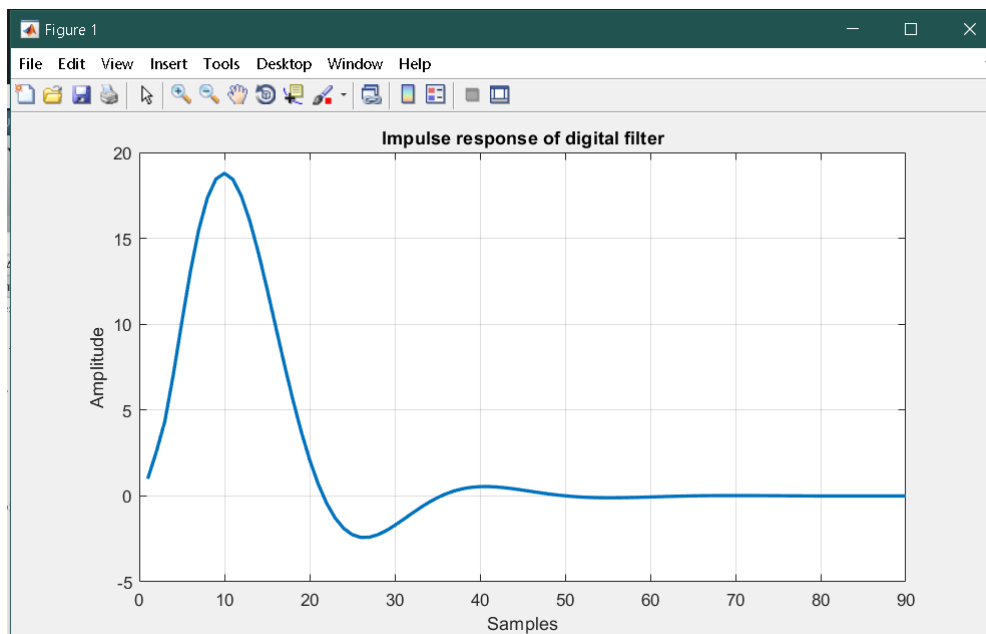


Figure 5: Impulse response of Digital filter with $n = 90$ samples

Observations and Discussions: We can observe from 5 that the impulse response **seems** its amplitude becomes zero after a certain time

Conclusions: Impulse response is a characterisation of the behaviour of the digital filter.

Even though it looks like a Finite Impulse response(FIR) . It is an Infinite Impulse response(IIR) i. it does not become zero after a point but continues infinitely. This

can be said by looking at the Digital Filter from which the response has been derived
 $y(n) - 2.56y(n-1) + 2.22y(n-2) - 0.65y(n-3) = x(n) + x(n-3)$

this is in the form :

$$\sum_{j=0}^Q a_j y[nj] = \sum_{i=0}^P b_i x[n - i]$$

and the z transform as:

$$H(z) = \frac{\sum_{i=0}^P b_i z^{-i}}{1 + \sum_{j=1}^Q a_j z^{-j}}$$

where

$$a_0 = 1.$$

This is the general form of IIRs in contrast to FIRs which have feed backward coefficients $a_1, a_2, \dots = 0$ and $a_0 = 1$. Also Clearly,

$$a_j \neq 0$$

i.e the poles are not located at the origin of the z-plane indicating its an infinite Impulse Response.

Section e): Find the output signal of the filter. If we truncate the impulse response upto 32 points, Compare your results with part (b) and (c). Comment on the results.

Code:

```

61 % Code for output with truncated Impulse response
62 clear
63 F = 500;           % Signal Frequency (Hz-cycles per second)
64 n = 48;           % samples (4 cycles)
65 Fs = 6000;        % Sampling Frequency(Hz-samples per second)
66 t = linspace(0, n/Fs, n);
67 x_n = sin(2 * pi * t * F);
68 num = [1 0 0 1];   %Feed forward Coefficients
69 den = [1 -2.56 2.22 -0.65]; %Feed Backward Coefficients
70
71 h = impz(num,den);
72 x_0 = conv(x_n,h(1:32)); %IIR*x[n]
73
74 figure
75 subplot(3,1,1)
76 plot(h(1:32),'LineWidth',2)
77 title("Truncated Impulse(32samples) ','FontSize', 15);
78 xlabel('Samples', 'FontSize', 8);
79 ylabel('Amplitude', 'FontSize', 8);
80
81 subplot(3,1,2)
82 plot(x_0,'LineWidth',2)
83 grid on
84 title("(Impulse Response)*(Input signal-4 cycle) ','FontSize', 15);
85 xlabel('Samples', 'FontSize', 8);
86 ylabel('Amplitude', 'FontSize', 8);
87
88
89 subplot(3,1,3)
90 plot(t,x_0(1:48),'LineWidth',2)
91 grid on
92 title("Same as above but with unitoftime upto 4 input cycle(48 ...
    samples) ','FontSize', 15);
93 xlabel('Time(t) in seconds', 'FontSize', 8);
94 ylabel('Amplitude', 'FontSize', 8);

```

Observations and Discussions: In the 3rd graph it can be observed that the curve is highly similar to that obtained in section b and c where filter was applied manually and functionally. Here **amplitude(max -apprx 50, min-apprx -15.**

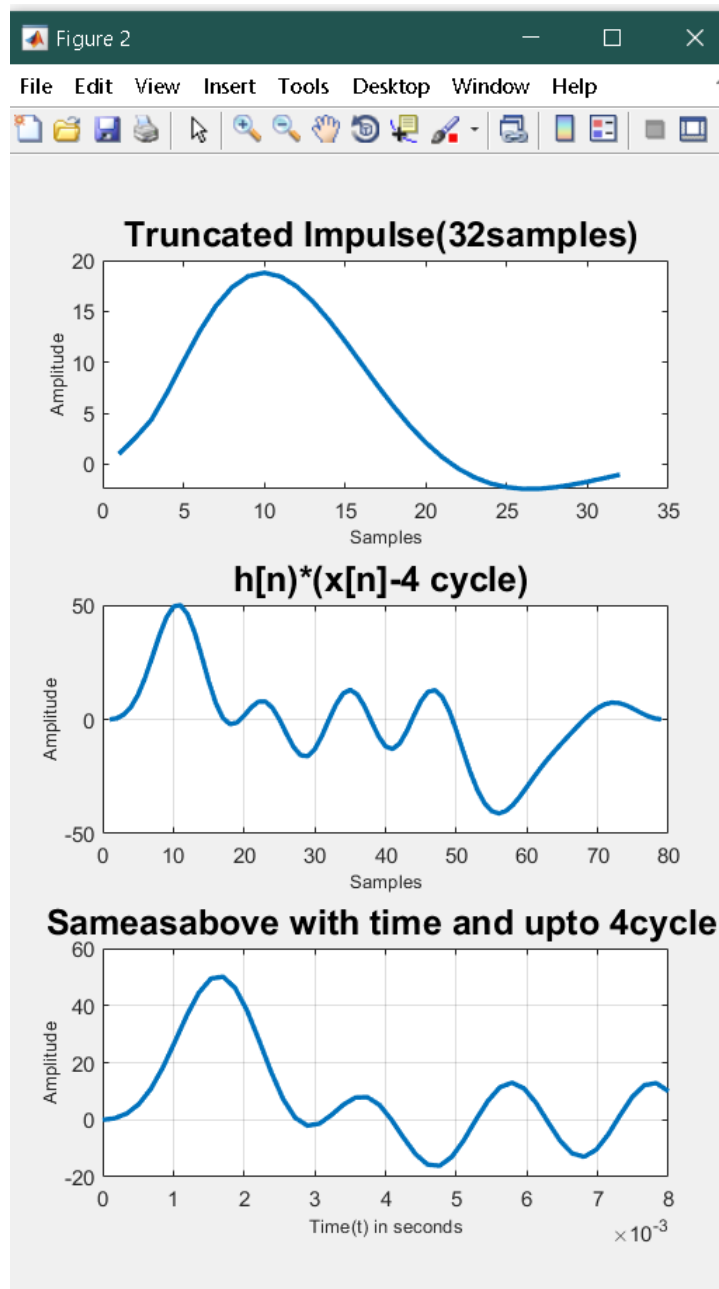


Figure 6: Output filtered with truncated Impulse response

Conclusions: The 3rd graph of 6 is just a zoom in on the first 48 points on the 2nd graph and plotted along with appropriate time units . This striking similarity between **3rd graph and graphs in section b and c** can be explained by noting that the Impulse response is a characterisation of the behaviour of the digital filter. Thus on convolving the impulse response with the input signal similar curve in terms of nature and shape has been observed.

Section f): Generate a new input signal $x(n)$, which is a summation of two sinusoids with frequency 500 Hz and 1500 Hz sampled at 6 kHz, and Repeat part (c) on generated new signal. Comment on your results.

Code:

```

96 %code
97 clear
98 F1 = 500; % Sinusoid frequency1 ( Hz))
99 F2 = 1500; % Sinusoid frequency2 ( Hz )
100 Fs = 6000; % Sampling Frequency ( Hz )
101 n = 100; % no. of samples
102 t = linspace(0,n/Fs,n);
103
104 x_n1 = sin(2 * pi * t * F1); %Sinusoid 1 at F1=500Hz Fs=6kHz
105 x_n2 = sin(2*pi*t*F2); %Sinusoid 1 at F1=500Hz Fs=6kHz
106 x_n = x_n1+x_n2; % Signal_1 + Signal_2
107
108 num = [1 0 0 1]; % Feed Forward Coeffs
109 den = [1 -2.56 2.22 -0.65]; % Feed Backward Coeffs
110 y_n = filter(num,den,x_n); %Digital filterer on New signal
111
112 figure
113 subplot(2,1,1)
114 plot(t, x_n, '-r', 'LineWidth', 2);
115 title("x[n] = x_{1}[n] + x_{2}[n]", 'FontSize', 15);
116 xlabel("Time in seconds", 'FontSize', 15);
117 ylabel('Amplitude', 'FontSize', 15);
118 grid on;
119 subplot(2,1,2)
120 plot(t, y_n, 'LineWidth', 2);
121 title("Output signal y[n]", "LineWidth", 2);
122 xlabel('Time in seconds', 'FontSize', 15);
123 ylabel('Amplituude', 'FontSize', 15);
124 grid on;

```

Observations and Discussions: It is observed that though the input signal has changed due to an addition of two different signals at frequency 500 and 1500 with same sampling frequency of 6kHz, the the output signal obtained after applying the digital filter is similar to that observed in the sections b,c and e .

Conclusions: The signal 1 with frequency 500Hz repeats at 12samples per cycle obtained from $F_s/F_1 = 6000/500 = 12$. Similarly signal 2 with frequency 1500 repeats at 4samples per cycle $F_s/F_2 = 6000/1500 = 4$. Also $4 \times 3 = 12$ i.e the time period of signal1 is an integral multiple of (3) time period of first signal. Due to this

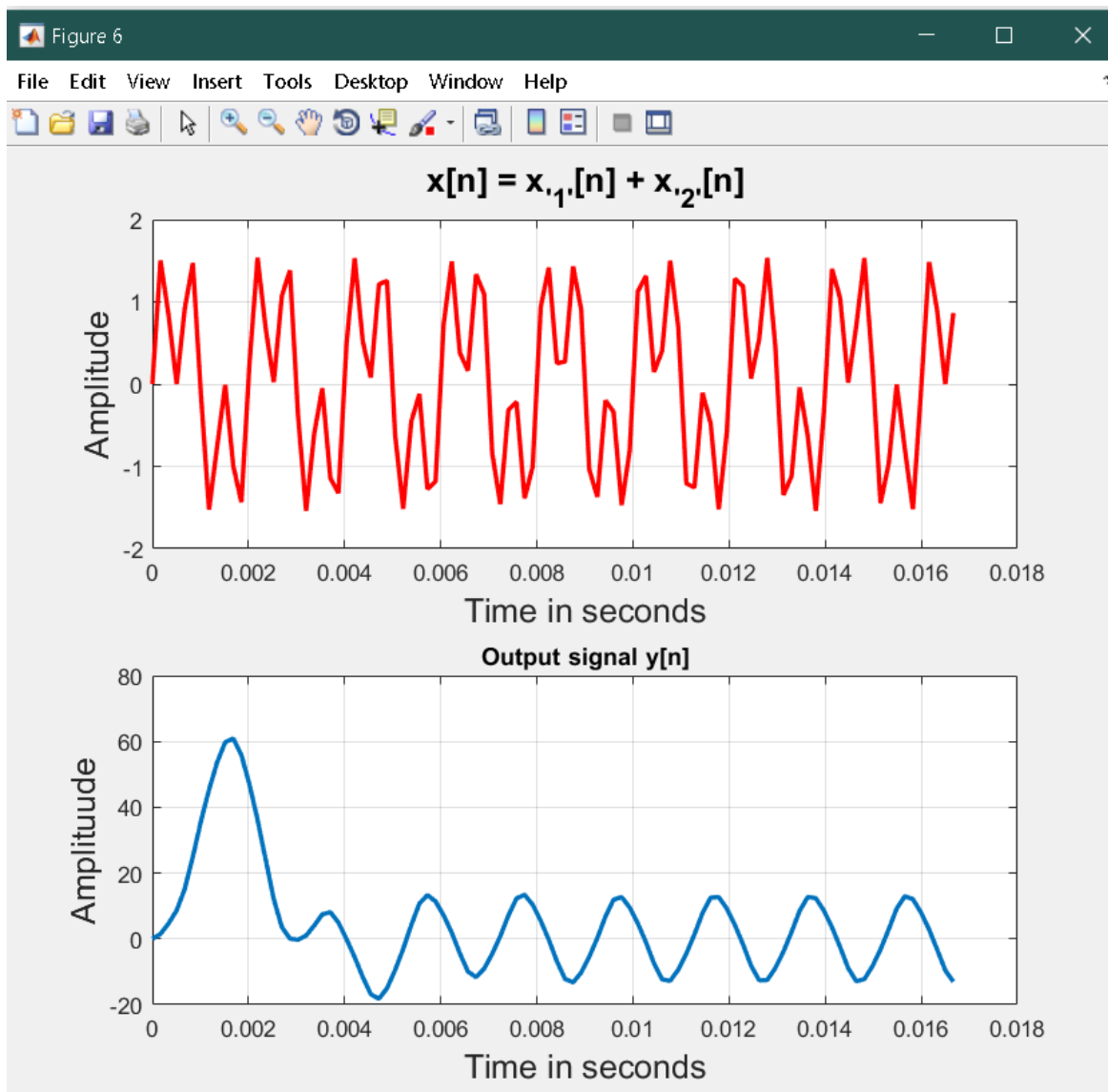


Figure 7: Output signal with Input signal

reason the resultant input signal is periodic and acts a sinusoid signal with uniform noise. Due to this the output signal is similar to previous results.

Section g): Find the frequency response of the filter and plot its magnitude and phase response of the filter.

Code:

```

126 num = [1 0 0 1];           %Feed Forward Coefficient
127 den = [1 -2.56 2.22 -0.65]; %Feed Backward Coefficient
128 [h,w] = freqz(num,den);    %Computing Frequency response
129 % h is the frequency response and w is the angular frequency (0 ...
    to pi)
130 figure
131 subplot(2,1,1)
132 plot((w/pi),abs(h),'LineWidth',2)
133 grid on
134 xlabel("Normalised Frequency (X\pi rad/sample) ','FontSize',15);
135 ylabel("Magnitude",'FontSize',15);
136 title("Magnitude response",'FontSize',15);
137
138 subplot(2,1,2)
139 plot(w/pi,angle(h),'LineWidth',2)
140 grid on
141 xlabel("Normalised Frequency (X\pi rad/sample)","FontSize',15);
142 ylabel("Phase Angle in rad",'FontSize',15);
143 title("Phase response",'FontSize',15);

```

Observations and Discussions: the magnitude frequency response of the provide digital filter highly has high gain between frequencies

$$0 - 0.1\pi \text{rad/sample}$$

and drastically falls such that it becomes nearly zero after

$$0.2\pi \text{rad/sample}$$

. In the phase response the phase angle keeps reducing from 0-0.1 and then suddenly changes its sign from negative to positive ,,. Then it keeps slowly reducing until around 0.33 after which the phase angle drops and becomes almost constant.

Important: The frequency response highly resembles frequency response of a low pass filter

Conclusions: The frequency response is a complex output ..Hence for analysis we need to plot the phase and magnitude parts separately. Also the frequency response of filter is also the fourier transform of the impulse response plotted in section d. One can spot the correlation between the magnitude response and its phase response

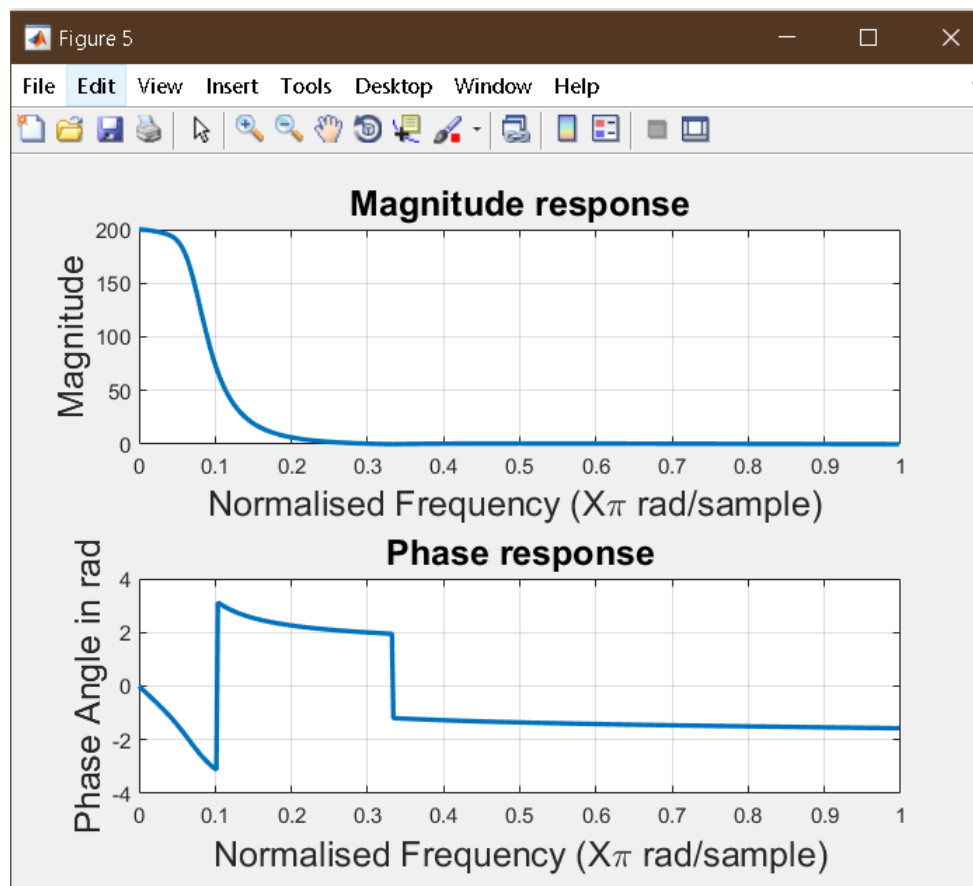


Figure 8: Frequency Response of Digital Filter

Graph a. Magnitude response . Graph b. Phase response

| | Magnitude | Phase |
|----------|-------------------------|----------------------------------|
| 0-0.1 | reduces, concave manner | linearly reduces (angle -ve) |
| 0.1-0.33 | reduces, Convex manner | exponential decrease (angle +ve) |
| 0.33 | almost constant at 0 | Almost constant (angle -ve) |

i.e when the decrease in magnitude changes from concave to convex, the phase decrease changes its polarity from negative to positive.

Section h): Generate the input signal $x(n)$, which is sinusoid with frequency 0.1 cycles/sample corrupted by a white Gaussian noise of approximately 5 dB.

Code:

```

145 %Code for White Gaussean Noise addition
146 clear
147
148 F = 100;          % (Cycles/second / Samples/Second) = Cycles/Sample
149 Fs = 1000;        % F/Fs = 100/1000 = 0.1 Cycles/Sample
150 n = 100;          % no. of samples| basically 10 cycles
151 t = linspace(0,n/Fs,n);
152 x_n = sin(2 * pi * t * F);    %sin(2pi*n*(1/Fs*F))
153
154 num = [1 0 0 1];          %Feed Forward Coefficients
155 den = [1 -2.56 2.22 -0.65]; %Feed Backward Coefficients
156 op = awgn(x_n,5, 'measured'); % Noise induced Signal at SNR = 5dB
157 SNR = snr(op);
158 impr = filter(num,den,op);
159 r_i = snr(impr);
160
161 figure
162 subplot(3,1,1)
163 plot(x_n, 'linewidth',2);
164 grid on
165 title("No Noise Input Signal ", 'FontSize',15);
166 xlabel("Time in seconds", 'FontSize',15);
167 ylabel("Amplitude", 'FontSize',15);
168 subplot(3,1,2);
169 plot(op, 'linewidth',2);
170 grid on
171 title("Input Signal with White Gaussean Noise(SNR = 5dB) ...
    ", 'FontSize',15);
172 xlabel("Time in seconds", 'FontSize',15);
173 ylabel("Amplitude", 'FontSize',15);
174 subplot(3,1,3)
175 snr(op,Fs);

```

Observations and Discussions: Graph 1: Since we had taken Frequency at 100Hz and Sampling frequency at 1000 Hz. the Cycles per ratio is $100/1000 = 0.1$. The No noise signal repeats at 10 samples/cycle or in other words it has a time period of $10(\text{samples/cycle})/1000(\text{samples/second}) = 0.01$ seconds. Graph 2: After adding a noise if 5dB using awgn function the sinusoid has become distorted but we can still roughly make out the 10 noisy cycles that correspond to the 10 noise free cycle in graph 1. Graph 3: In this graph which was obtained by calculating the snr on the obtained noisy signal in graph 2 reaffirms that the SNR remained at 5.064dB.. This

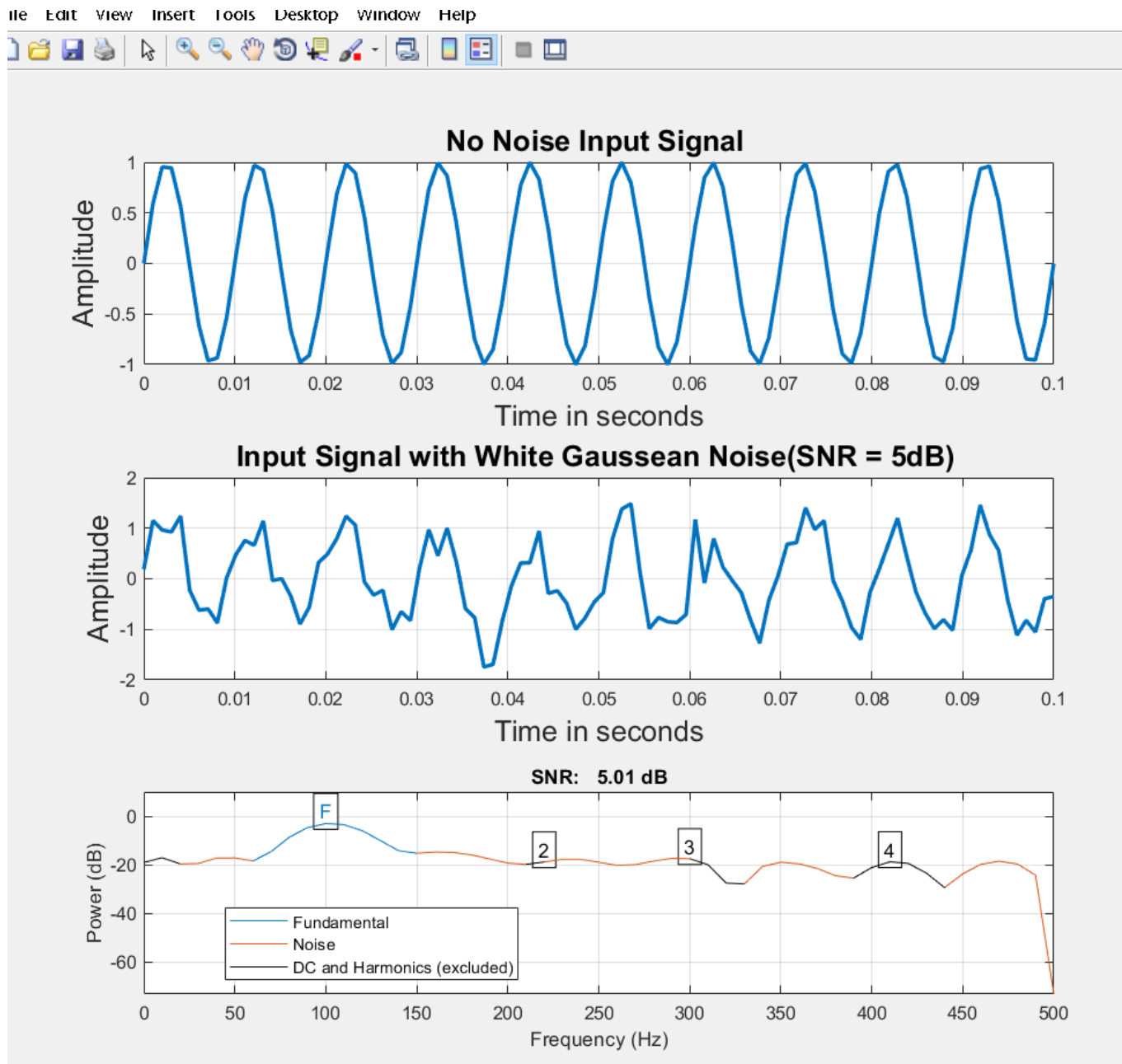


Figure 9: Graph 1. Clean Signal with 0.1 cycles/Sample
 Graph 2. White Gaussean Noise added signal at SNR = 5dB.
 Graph 3 Noise spectrum

graph is an inbuilt plot by calling `snr()` function from matlab.

Conclusions: The white gaussian noise that added is a noise which has '**White**' meaning a uniform power throughout the whole frequency band and '**Gaussean**' i.e having normal distribution with mean at 0 for each sample. This ensures that the noise added is uniform and hence though irregular we can make out the peaks and trough i.e cycles of the signal.

Section i): Filter the noisy signal generated in part (h) and compute the improvement in Signal to Noise Ratio. Also Comment on your results.

Code:

```

177 %Code for Improvement in SNR
178 clear
179
180 F = 100;           % (Cycles/second / Samples/Second) = Cycles/Sample
181 Fs = 1000;         % F/Fs = 100/1000 = 0.1 Cycles/Sample
182 n = 100;           % no. of samples| basically 10 cycles
183 t = linspace(0,n/Fs,n);
184 x_n = sin(2 * pi * t * F); %sin(2pi*n*(1/Fs*F))
185
186 num = [1 0 0 1];           %Feed Forward Coefficients
187 den = [1 -2.56 2.22 -0.65]; %Feed Backward Coefficients
188 op = awgn(x_n,5,'measured'); % Noise induced Signal at SNR = 5dB
189 SNR = snr(op);
190 improved = filter(num,den,op);
191 r_i = snr(improved);
192
193 figure
194 subplot(2,1,1)
195 plot(t,improved,'linewidth',2);
196 grid on
197 title("Filtered White Gaussean Noise signal",'FontSize',15);
198 xlabel("Time in seconds",'FontSize',15);
199 ylabel("Amplitude",'FontSize',15);
200 subplot(2,1,2)
201 snr(improved,Fs);

```

Observations and Discussions: The noise signal that was filtered had a SNR value of 4.98dB before going through the Filter Obtained in the variable SNR in code. After appying Digital Filter , the new SNR of the signal became 38.17dB . As recorded in variable ri and the noise spectrum plot obtained from its snr() function.

Conclusions: since SNR refers to signal power/Noise power , a higher SNR value would mean a relatively lower noise in the signal . Hence it can be concluded that the obtained graph is an noisy but improved signal. The improvement can be calculated as

$$\begin{aligned}
 \text{Improvement in SNR} &= \text{SNR after filter} - \text{SNR before filter} \\
 &= 38.17 - 4.98 = 33.19\text{dB}
 \end{aligned}$$

Section i): Filter the noisy signal generated in part (h) and compute the improvement in Signal to Noise Ratio. Also Comment on your results.

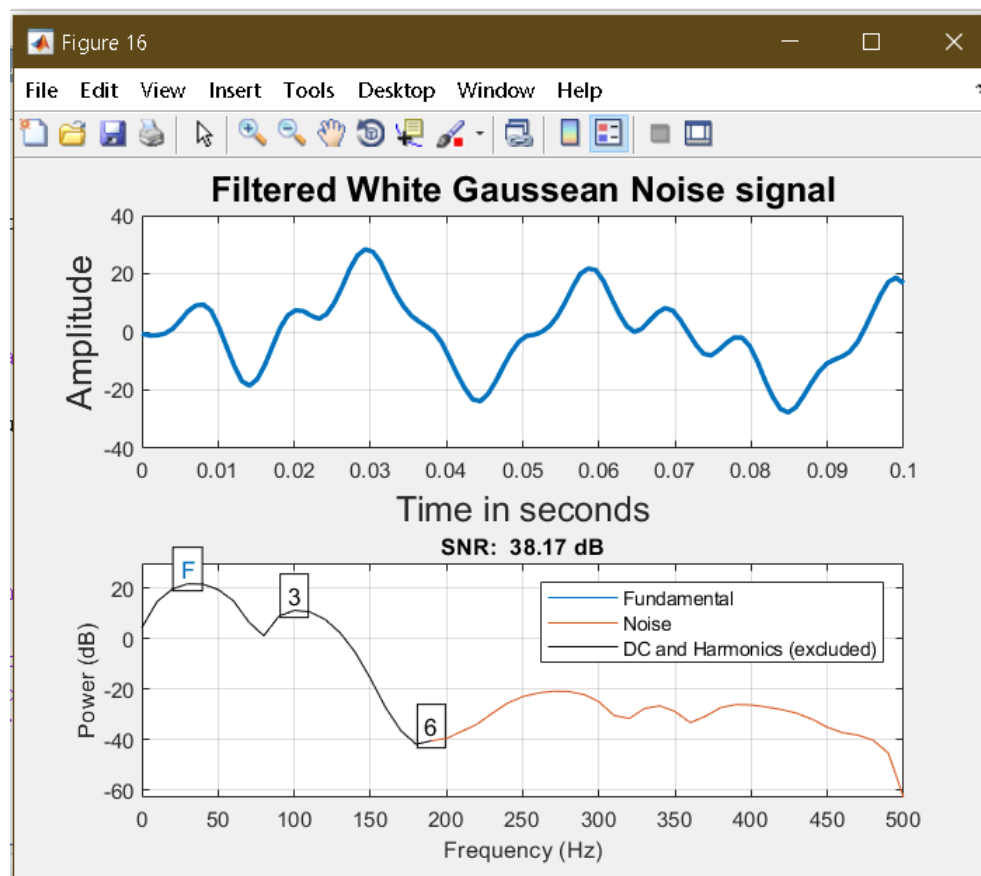


Figure 10: Graph 1. Filtered WGN signalGraph 2. Noise Spectrum

Code:

```

203 % Code for spectrum of signal (at 4 cycles , perfectly sampled
204 clear
205 F = 10000;           % Sinusoid frequency ( Hz - Cycles/ second)
206 Fs = 100000;        % Sampling Frequency ( Hz -Samples/ second )
207 n = 4*Fs/F;         % 4(no. of cycles)* (no. of samples/cycle)
208 t = 0:n-1;         %time axis , 1/Fs is the sampling period
209 x_n = sin(2*pi*t*F*1/Fs); % sampled signal
210
211 fft_x_n = fft(x_n)/length(x_n)*2; %finding the fft, normalising ...
    value for correct amplitude range
212 fft_x_n = fftshift(fft_x_n); % shifting fft to make it ...
    symmetric about 0
213 Freq = ((-n/2:(n/2)-1))/n)*2*pi; % defining frequency scale
214
215 figure
216 subplot(3,1,1)
217 stem(t/Fs,x_n,'linewidth',2)
218 title("Sampled Sinusoid - 4 cycles",'FontSize',15);
219 xlabel("Time in seconds",'FontSize',15);
220 ylabel("Amplitude",'FontSize',15);
221 hold on
222 plot(t/Fs,x_n,'-g','linewidth',1.5)
223 grid on
224 subplot(3,1,2)
225 stem(Freq,angle(fft_x_n),'linewidth',2);
226 title("Phase spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)",'FontSize',15);
227 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
228 ylabel("Phase (in rad)",'FontSize',15);
229 hold on
230 plot(Freq,angle(fft_x_n),'linewidth',1.5);
231 grid on
232 subplot(3,1,3)
233 stem(Freq,abs(fft_x_n),'linewidth',2);
234 title("Amplitude spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)",'FontSize',15);
235 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
236 ylabel("Amplitude",'FontSize',15);
237 hold on
238 plot(Freq,abs(fft_x_n),'linewidth',1.5);
239 grid on

```

Observations and Discussions: We can observe that the sinusoid has a rate of 10 samples/cycle and hence 4 cycles consist of samples at $s = 0$ to 39 . We can also see that there is no aliasing happening in the sinusoid . Also the amplitude/Magnitude spectrum of the signal has 2 peaks at around $w = -0.6$ and $+0.6$ i.e symmetrically

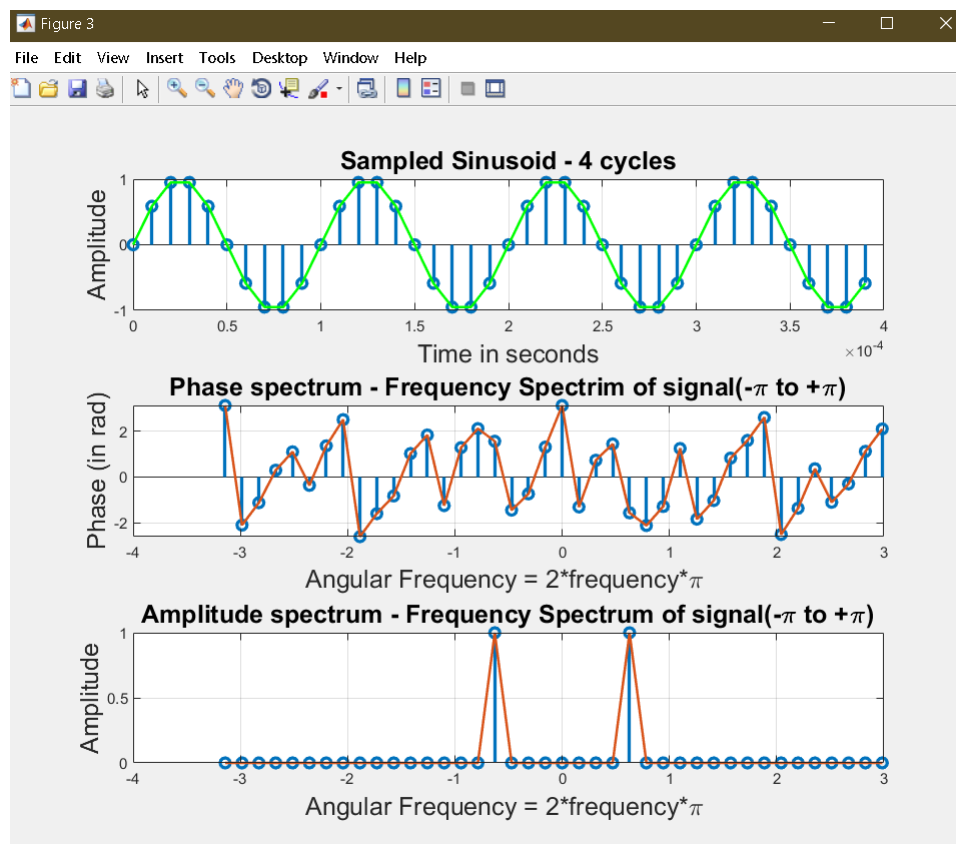


Figure 11: Signal Spectrum with Sampling

about the origin

Conclusions: Here the range $-\pi$ to $+\pi$ refers to the angular frequencies of the fourier transform of the sampled signal. Also the sampling frequency is adequate enough to represent the sinusoidal shape .

Code:

```

240 % Code for input x[n] sinusoid creation
241 clear
242 F = 10000;           % Sinusoid frequency ( Hz - Cycles/ second)
243 Fs = 100000;         % Sampling Frequency ( Hz -Samples/ second )
244 n = 4*Fs/F;         % 4(no. of cycles)* (no. of samples/cycle) =40
245
246 t2 = 0:n-1;          %time axis with samples taken at 0,1,...39 = ...
    40 samples
247 x_n2 = sin(2*pi*t2*F/Fs); % input sinusoid
248 x_n2 = downsample(x_n2,2); % Downsampling Signal by factor of 2
249 t2 = 0:n/2-1;        %redefining time axis for ...
    downsampled signal
250 fft_x_n2 = fft(x_n2)/length(x_n2)*2;
251 %finding the fft, normalising value for correct amplitude range
252 fft_x_n2 = fftshift(fft_x_n2);
253 % shifting fft to make it symmetric about 0
254 Freq = ((-n/4:(n/4-1))/(n/2))*2*pi;
255
256 figure
257 subplot(3,1,1)
258 stem(t2/(Fs/2),x_n2,'linewidth',2)
259 title("Sampled Sinusoid - 4 cycles",'FontSize',15);
260 xlabel("Time in seconds",'FontSize',15);
261 ylabel("Amplitude",'FontSize',15);
262 hold on
263 plot(t2/(Fs/2),x_n2,'-g','linewidth',1.5)
264 grid on
265 subplot(3,1,2)
266 stem(Freq,angle(fft_x_n2),'linewidth',2);
267 title("Phase spectrum - Frequency Spectrim of signal(-\pi to ...
    +\pi)", 'FontSize',15);
268 xlabel("Angular Frequency = 2*frequency*\pi ", 'FontSize',15);
269 ylabel("Phase (in rad)", 'FontSize',15);
270 hold on
271 plot(Freq,angle(fft_x_n2),'linewidth',1.5);
272 grid on
273 subplot(3,1,3)
274 stem(Freq,abs(fft_x_n2),'linewidth',2);
275 title("Amplitud spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)", 'FontSize',15);

```

```

276 xlabel("Angular Frequency = 2*frequency*\pi ", 'FontSize',15);
277 ylabel("Amplitude", 'FontSize',15);
278 hold on
279 plot(Freq,abs(fft_x_n2), 'linewidth',1.5);
280 grid on

```

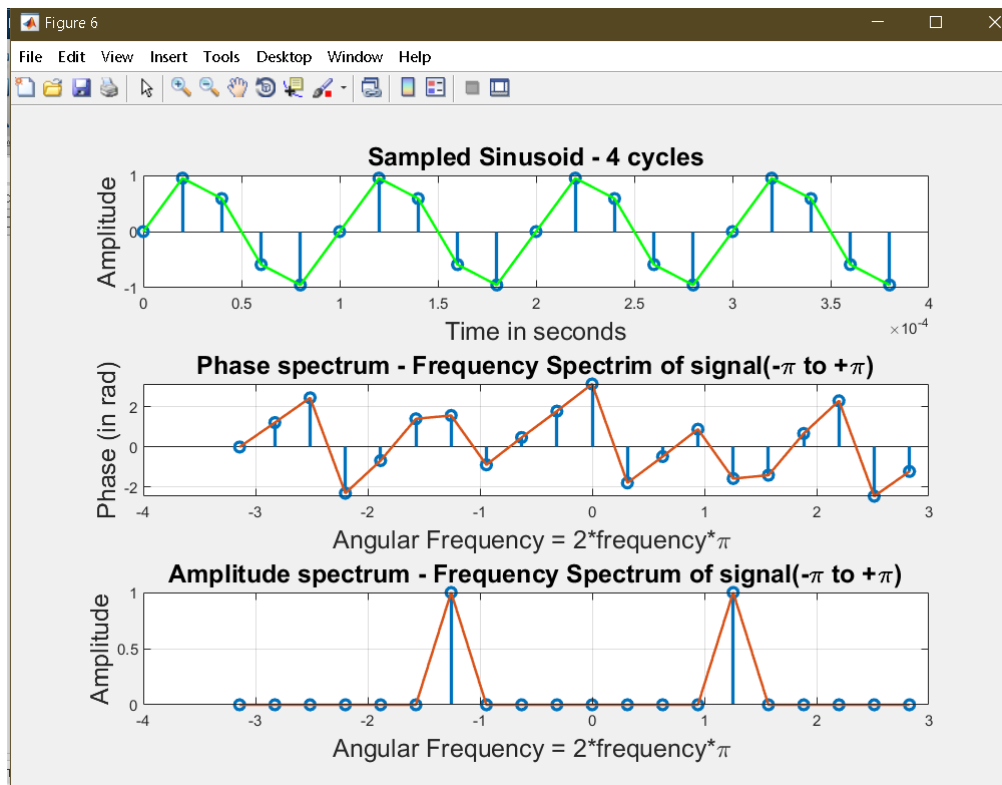


Figure 12: Signal Spectrum with downSampling

Observations and Discussions: It can be observed from Fig. 14, that the sinusoid obtained by downsampling previous sinusoid gives a graph which is less smooth. In the magnitude spectrum we again observe 2 symmetrically occurring peaks. However they are more far apart than in the first case with $F_s = 100000$.

Conclusions: We can say that the sampling frequency is not sufficient to correctly describe a smooth sinusoidal signal. Also though the magnitude/amplitude spectrum seems to have been plotted from $-\pi$ to less than $+\pi$ It can be considered as a frequency spectrum of 4 cycles of sinusoid from $-\pi$ to $+\pi$ with the value at $+\pi$ being the very next point after the last point in magnitude spectrum graph and will

have a value same as that at $w = -\pi$ i.e 0 . Here for downsampling has been done based on the formula .

Assignment -2:

2.1 Q1: Generate a 10-kHz sinusoid sampled at 100 kHz. Plot four cycles of this signal.

Code:

```

281 % Code for spectrum of signal (at 4 cycles , perfectly sampled
282 clear
283 F = 10000;           % Sinusoid frequency ( Hz - Cycles/ second)
284 Fs = 100000;         % Sampling Frequency ( Hz -Samples/ second )
285 n = 4*Fs/F;          % 4(no. of cycles)* (no. of samples/cycle)
286 t = 0:n-1;          %time axis , 1/Fs is the sampling period
287 x_n = sin(2*pi*t*F*1/Fs); % sampled signal
288
289 fft_x_n = fft(x_n)/length(x_n)*2; %finding the fft, normalising ...
    value for correct amplitude range
290 fft_x_n = fftshift(fft_x_n); % shifting fft to make it ...
    symmetric about 0
291 Freq = ((-n/2:(n/2-1))/n)*2*pi; % defining frequency scale
292
293 figure
294 subplot(3,1,1)
295 stem(t/Fs,x_n,'linewidth',2)
296 title("Sampled Sinusoid - 4 cycles",'FontSize',15);
297 xlabel("Time in seconds",'FontSize',15);
298 ylabel("Amplitude",'FontSize',15);
299 hold on
300 plot(t/Fs,x_n,'-g','linewidth',1.5)
301 grid on
302 subplot(3,1,2)
303 stem(Freq,angle(fft_x_n),'linewidth',2);
304 title("Phase spectrum - Frequency Spectrim of signal(-\pi to ...
    +\pi)", 'FontSize',15);
305 xlabel("Angular Frequency = 2*frequency*\pi ", 'FontSize',15);
306 ylabel("Phase (in rad)", 'FontSize',15);
307 hold on
308 plot(Freq,angle(fft_x_n), 'linewidth',1.5);
309 grid on
310 subplot(3,1,3)
311 stem(Freq,abs(fft_x_n), 'linewidth',2);
312 title("Amplitude spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)", 'FontSize',15);
313 xlabel("Angular Frequency = 2*frequency*\pi ", 'FontSize',15);
314 ylabel("Amplitude", 'FontSize',15);
315 hold on
316 plot(Freq,abs(fft_x_n), 'linewidth',1.5);
317 grid on

```

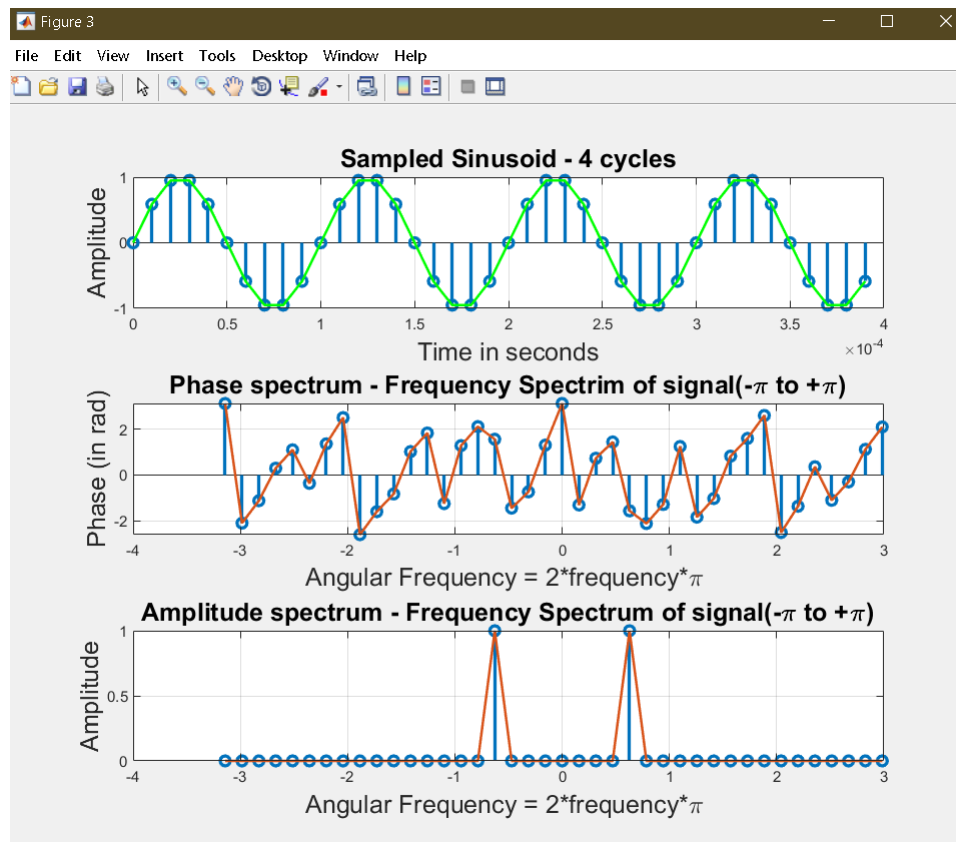



Figure 13: Signal Spectrum with Sampling

Observations and Discussions: We can observe that the sinusoid has a rate of 10 samples/cycle and hence 4 cycles consist of samples at $s = 0$ to 39 . We can also see that there is no aliasing happening in the sinusoid . Also the amplitude/Magnitude spectrum of the signal has 2 peaks at around $w = -0.6$ and $+0.6$ i.e symmetrically about the origin

Conclusions: Here the range $-\pi$ to $+\pi$ refers to the angular frequencies of the fourier transform of the sampled signal. Also the sampling frequency is adequate enough to represent the sinusoidal shape .

2.2 Q2: Decimate this signal by a factor of 2. Plot the time-domain signal. Also plot t**Code:**

```

318 % Code for input x[n] sinusoid creation
319 clear
320 F = 10000;           % Sinusoid frequency ( Hz - Cycles/ second)
321 Fs = 100000;        % Sampling Frequency ( Hz -Samples/ second )
322 n = 4*Fs/F;         % 4(no. of cycles)* (no. of samples/cycle) =40
323
324 t2 = 0:n-1;          %time axis with samples taken at 0,1,...39 = ...
    40 samples
325 x_n2 = sin(2*pi*t2*F/Fs); % input sinusoid
326 x_n2 = downsample(x_n2,2); % Downsampling Signal by factor of 2
327 t2 = 0:n/2-1;        %redefining time axis for ...
    downsamped signal
328 fft_x_n2 = fft(x_n2)/length(x_n2)*2;
329 %finding the fft, normalising value for correct amplitude range
330 fft_x_n2 = fftshift(fft_x_n2);
331 % shifting fft to make it symmetric about 0
332 Freq = ((-n/4:(n/4-1))/(n/2))*2*pi;
333
334 figure
335 subplot(3,1,1)
336 stem(t2/(Fs/2),x_n2,'linewidth',2)
337 title("Sampled Sinusoid - 4 cycles",'FontSize',15);
338 xlabel("Time in seconds",'FontSize',15);
339 ylabel("Amplitude",'FontSize',15);
340 hold on
341 plot(t2/(Fs/2),x_n2,'-g','linewidth',1.5)
342 grid on
343 subplot(3,1,2)
344 stem(Freq,angle(fft_x_n2),'linewidth',2);
345 title("Phase spectrum - Frequency Spectrim of signal(-\pi to ...
    +\pi)",'FontSize',15);
346 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
347 ylabel("Phase (in rad)",'FontSize',15);
348 hold on
349 plot(Freq,angle(fft_x_n2),'linewidth',1.5);
350 grid on
351 subplot(3,1,3)
352 stem(Freq,abs(fft_x_n2),'linewidth',2);
353 title("Amplitud spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)",'FontSize',15);
354 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
355 ylabel("Amplitude",'FontSize',15);
356 hold on
357 plot(Freq,abs(fft_x_n2),'linewidth',1.5);
358 grid on

```

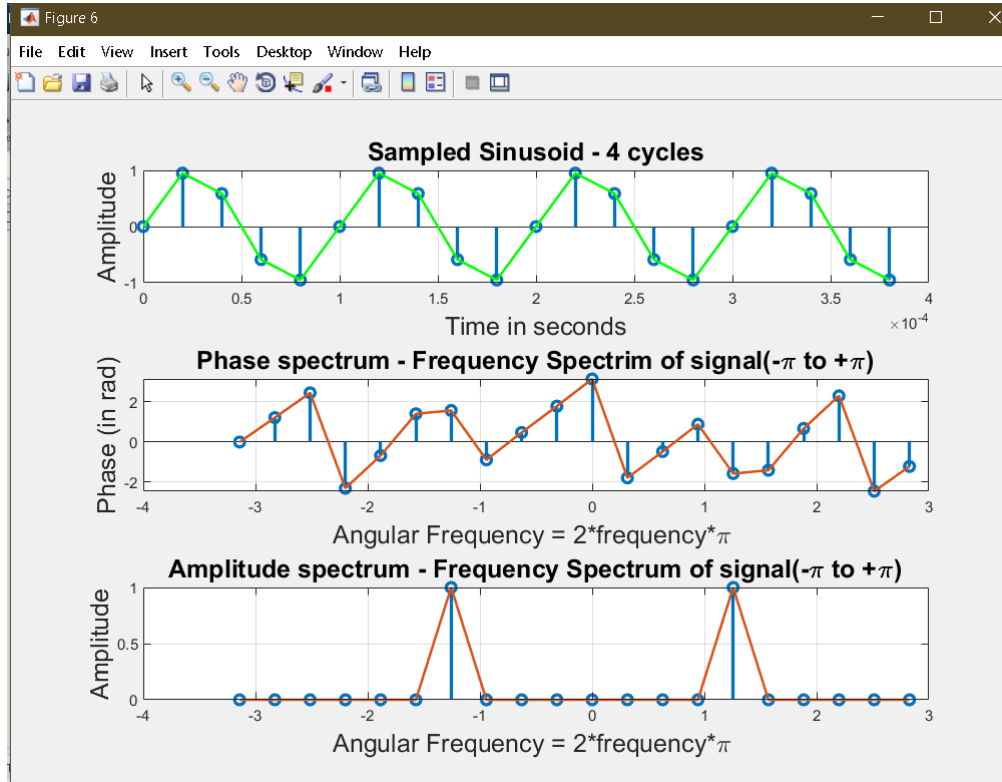


Figure 14: Signal Spectrum with downSampling

Observations and Discussions: It can be observed from Fig. 14, that the sinusoid obtained by downsampling previous sinusoid gives a graph which is less smooth. In the magnitude spectrum we again observe 2 symmetrically occurring peaks. However they are more far apart than in the first case with $F_s = 100000$.

Conclusions: We can say that the sampling frequency is not sufficient to correctly describe a smooth sinusoidal signal. Also though the magnitude/amplitude spectrum seems to have been plotted from $-\pi$ to less than $+\pi$ It can be considered as a frequency spectrum of 4 cycles of sinusoid from $-\pi$ to $+\pi$ with the value at $+\pi$ being the very next point after the last point in magnitude spectrum graph and will have a value same as that at $\omega = -\pi$ i.e 0. Here for downsampling has been done based on the formula.

2.3 Q3: Upsample the original sinusoidal sequence by a factor of 3. Plot the time domain

Code:

```

359 % Code for spectrum of upsampled signal
360 clear
361 F = 10000;           % Sinusoid frequency ( Hz - Cycles/ second)
362 Fs = 100000;        % Sampling Frequency ( Hz -Samples/ second )
363 n = 4*Fs/F;         % 4(no. of cycles)* (no. of samples/cycle) =40
364
365 t3 = 0:n-1;          %time axis with samples taken at 0,1,...119 ...
    = 120 samples
366 x_n3 = sin(2*pi*t3*F/Fs); % input sinusoid
367 x_n3 = upsample(x_n3,3); % upsampling Signal by factor of 3
368 t3 = 0:n*3-1;        %redefining time axis for upsampled signal
369 fft_x_n3 = fft(x_n3);
370 %finding the fft, normalising value for correct amplitude range
371 fft_x_n3 = fftshift(fft_x_n3);
372 % shifting fft to make it symmetric about 0
373 Freq = ((-(3*n)/2:((n*3)/2)-1))/(n*3))*2*pi;
374
375 figure
376 subplot(3,1,1)
377 stem(t3/(Fs*3),x_n3,'linewidth',2)
378 title("Sampled Sinusoid - 4 cycles",'FontSize',15);
379 xlabel("Time in seconds",'FontSize',15);
380 ylabel("Amplitude",'FontSize',15);
381 hold on
382 plot(t3/(Fs*3),x_n3,'-g','linewidth',1.5)
383 grid on
384 subplot(3,1,2)
385 stem(Freq,angle(fft_x_n3),'linewidth',2);
386 title("Phase spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)",'FontSize',15);
387 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
388 ylabel("Phase (in rad)",'FontSize',15);
389 hold on
390 plot(Freq,angle(fft_x_n3),'linewidth',1.5);
391 grid on
392 subplot(3,1,3)
393 stem(Freq,abs(fft_x_n3),'linewidth',2);
394 title("Amplitude spectrum - Frequency Spectrum of signal(-\pi to ...
    +\pi)",'FontSize',15);
395 xlabel("Angular Frequency = 2*frequency*\pi",'FontSize',15);
396 ylabel("Amplitude",'FontSize',15);
397 hold on
398 plot(Freq,abs(fft_x_n3),'linewidth',1.5);
399 grid on

```

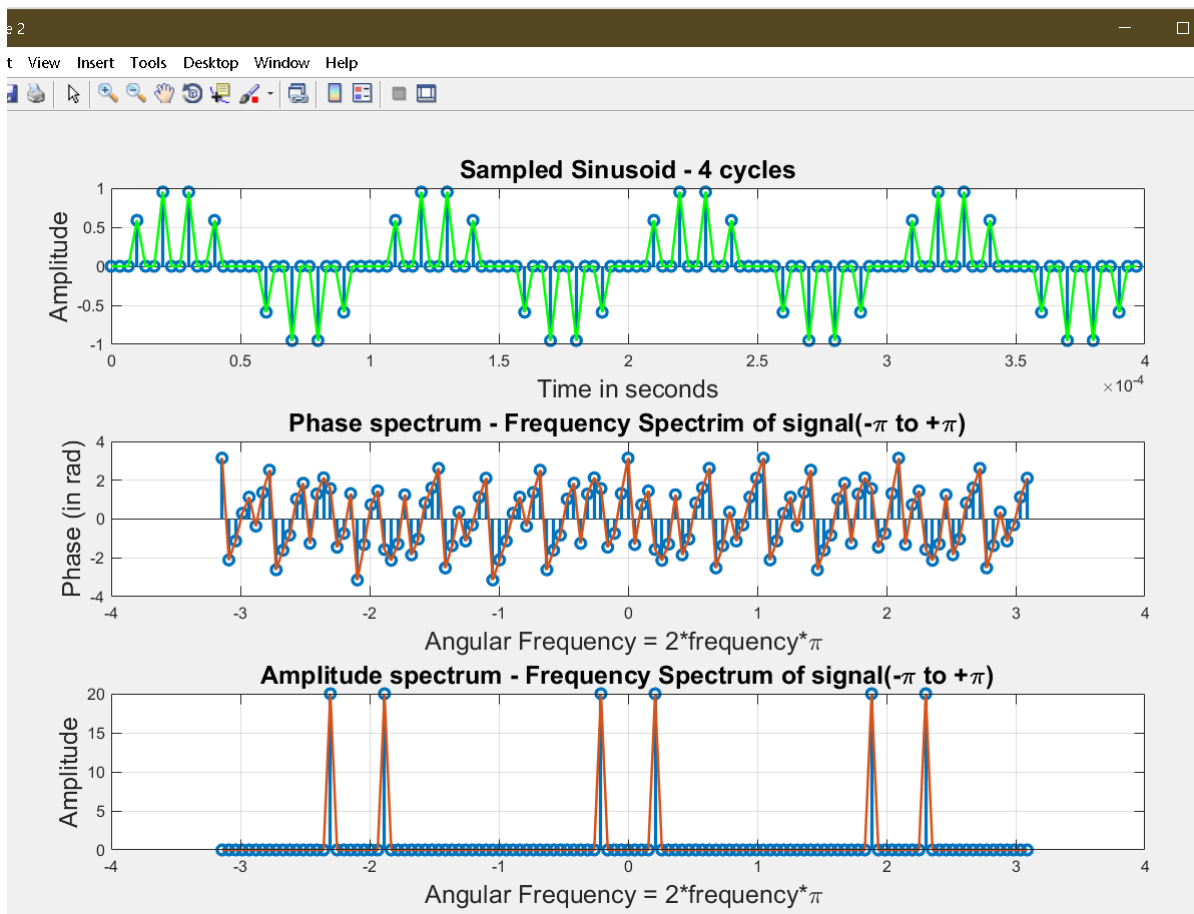


Figure 15: Signal Spectrum with upSampling

Observations and Discussions: It can be observed from Fig. 13, that though the function should represent a sinusoid, Due to the presence of 0 values samples in between the non zero samples, the overall plot of the signal is no longer sinusoid . Also this can be observed in the magnitude spectrum of the signal i.e the magnitude(fourier transform(signal)) that instead of 2 spikes as observed in the previous 2 cases it has 6 spikes place symmetrically across 0 in the range $-\pi$ to $+\pi$

Conclusions: The reason for such a sinusoid is overampling. The signal has been very highly oversampled causing unrequired samples and deviating from sinusoidal shape . Here the range $-\pi$ to $+\pi$ thus refers to the angular frequency range of the fourier transform of a highly oversampled signal.

Assignment -3:

3.1 Q1):Write a program to compute the N-point DFT of a sequence. Do not use the i

Code:

```

401 % Code for n-point DFT function
402 clc
403 clear
404
405 x_n = [1 2 3 4]; %Sample sequence for testing
406 x_n_nfft = nfft(x_n,length(x_n)); %Calling n point dft function ...
    (nfft)
407 disp(x_n_nfft);
408
409 function [x_k]=nfft(x_n,N)
410 len = length(x_n);
411 if N>len
412     x_n = [x_n zeros(1,N)];
413 elseif N<len
414     x_n = x_n(1:N);
415 end
416 for l = 1:N
417     x_k(l) = 0;
418     for n = 1:N
419         x_k(l) = x_k(l) + x_n(n).*exp((-1j).*2.*pi.*(n-1).*(l-1)/N);
420     end
421 end
422 end

```

Output: For analyzing the phenomenon,the sequence

$$x[n] = [1 \ 2 \ 3 \ 4]$$

for which the output so observed was

$$10.0000 + 0.0000i \ -2.0000 + 2.0000i \ -2.0000 - 0.0000i \ -2.0000 - 2.0000i$$

Conclusion: The function works correctly , verified by comparing result with the inbuilt function separately

3.2 Q2): Use your program to find the 200-point DFT of the following sequences: (i) $x(n)$

i) $x(n) = 2\cos(2n/10) + \cos(2n/5)$:

Code:

```

424 clc
425 clear
426
427 n = 0:199;    % No. of samples
428 N = 200;      %Length of sequence
429 xn = 2*cos(2*pi*n/10) + cos(2*pi*n/5); % defining function
430 w = pi*n/N;   % Converting 200 samples ...
      to 0-2pi
431 xk = nfft(xn,N); % Finding 200 point DFT
432
433
434 figure
435 subplot(2,1,1)
436 stem(n,xn)
437 xlabel("samples")
438 ylabel("Amplitude")
439 title("x[n] = 2cos(2\pin/10) + cos(2\pin/5)")
440 grid on
441 subplot(2,1,2)
442 stem(w,abs(xk))
443 hold on
444 plot(w,abs(xk))
445 xlabel("omega")
446 ylabel("Magnitude")
447 title("x(k) against omega")
448 grid on

```

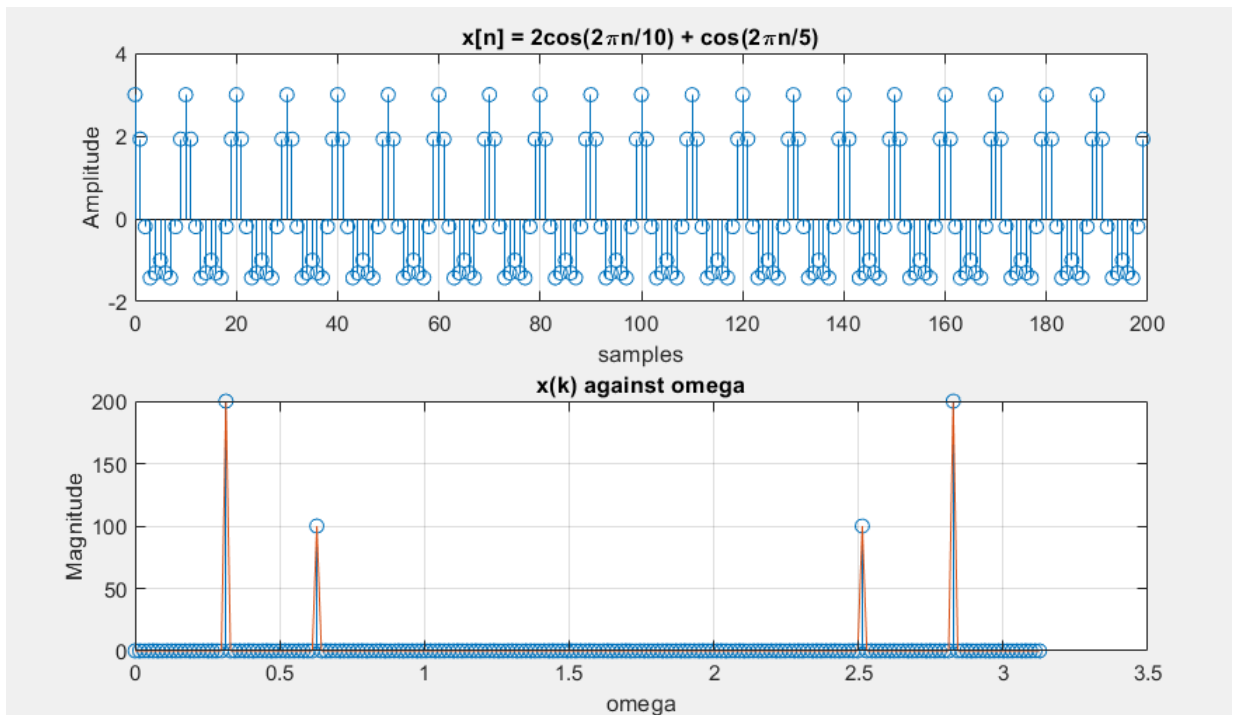


Figure 16: Magnitude response of 200-point DFT of input signal

ii) $x(n) = n$:

Code:

```

449 clc
450 clear
451
452 n = 0:199;    % No. of samples
453 N = 200;      %Length of sequence
454 xn =n;    % defining function
455 w = pi*n/N;                                     % Converting 200 samples ...
         to 0-2pi
456 xk = nfft(xn,N);                               % Finding 200 point DFT
457
458
459 figure
460 subplot(2,1,1)
461 stem(n,xn,'LineWidth',1)
462 xlabel("samples")
463 ylabel("Amplitude")
464 title("x[n] = n")
465 grid on
466 subplot(2,1,2)
467 stem(w,abs(xk),'LineWidth',1)
468 xlabel("omega")
469 ylabel("Magnititude")
470 title("x(k) against omega")
471 grid on

```

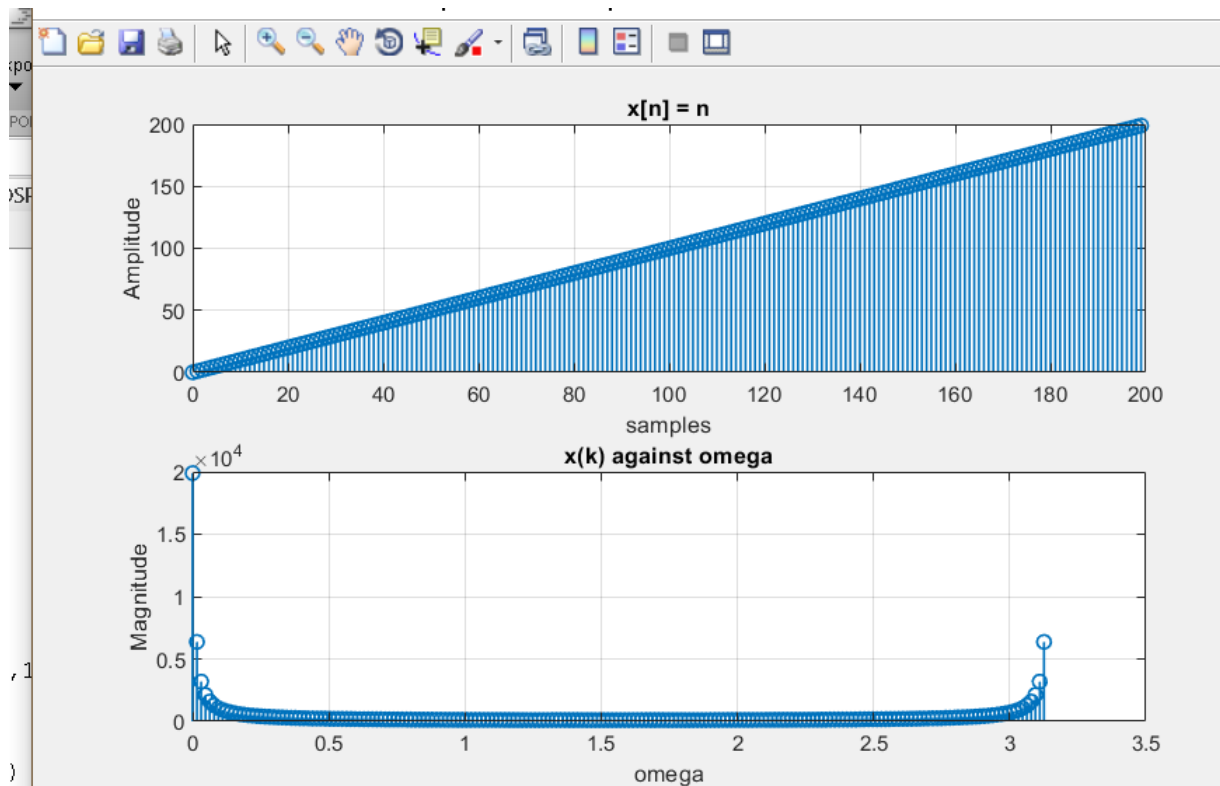


Figure 17: Magnitude response of 200-point DFT of input signal

Conclusion: The function works correctly , verified by comparing results with the result obtained from the inbuilt function separately.

i)The cos function seems to have 2 pair of symmetric peak.

ii)On the other hand the linear function first had the magnitude decreasing at the beginning upto zero and then increasing towards the end samples. However it is non identical.

3.3 Q3: Write a program to compute the N-point inverse-DFT (IDFT) of a sequence.

Inverse DFT function::

Code:

```

473 % Inverse DFT function
474 function [x_n]=infft(x_k,N)
475 len = length(x_k);
476 if N>len
477     x_k = [x_k zeros(1,N)];
478 elseif N<len
479     x_k = x_k(1:N);
480 end
481 for k = 1:N
482     x_n(k) = 0;
483     for n = 1:N
484         x_n(k) = x_n(k) + x_k(n).*exp((1j).*2.*pi.*(n-1).*(k-1)/N);
485     end
486     x_n(k) = x_n(k)/N;
487 end
488 end

```

i) $x(n) = 2\cos(2n/10) + \cos(2n/5)$:

Code:

```

489
490 n = 0:199;
491 N = 200;
492 x_n = 2*cos(2*pi*n/10) + cos(2*pi*n/5);
493 x_k = nfft(x_n,N); % Finding DFT using created function
494 x_ngen = infft(x_k,N); % Finding original sequence using made ...
    IDFT function
495 w = pi*n/N;
496
497
498 figure
499 subplot(3,1,1)
500 stem(w,abs(x_k),'LineWidth',1)
501 xlabel("Omega")
502 ylabel("Magnitude")
503 title("DFT of x[n] = 2*cos(2*\pi*n/10) + cos(2*\pi*n/5)")
504 grid on
505 subplot(3,1,2)
506 plot(n,x_ngen)
507 xlabel("Samples")

```

```

508 ylabel("Amplitude")
509 title("IDFT of x[k] , Generated from DFT")
510 grid on
511 subplot(3,1,3)
512 plot(n,x_n)
513 xlabel("Samples")
514 ylabel("Amplitude")
515 title("Original function x[n]")
516 grid on

```

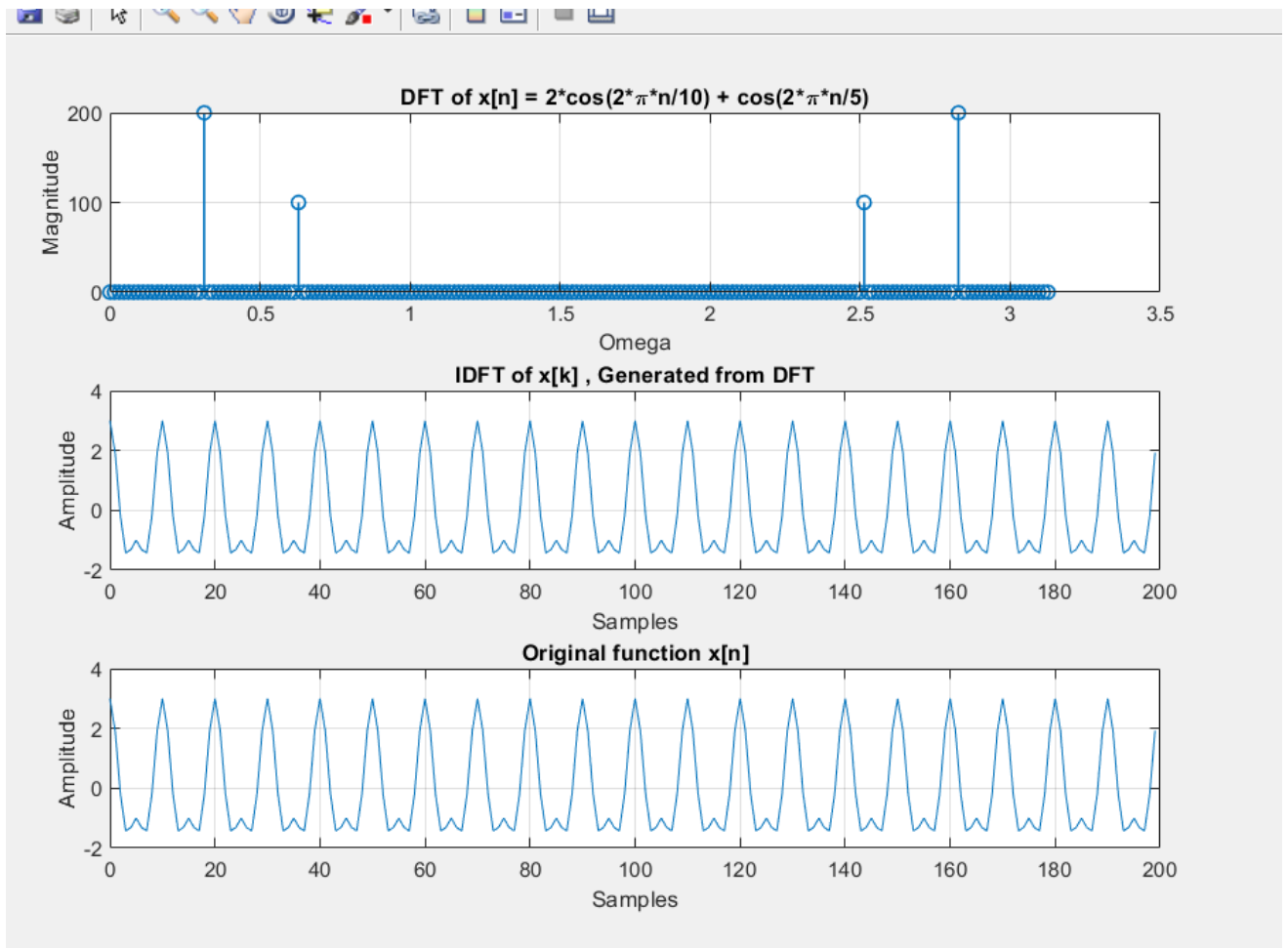


Figure 18: Inverse DFT of signal1

ii) $x(n) = n$:

Code:

```
518 clc
519 clear
520
521 n = 0:199;
522 N = 200;
523 x_n = n;
524 x_k = nfft(x_n,N); % Finding DFT using created function
525 x_ngen = ifft(x_k,N); % Finding original sequence using made ...
    IDFT function
526 w = pi*n/N;
527
528
529 figure
530 subplot(3,1,1)
531 stem(w,abs(x_k),'LineWidth',1)
532 xlabel("Omega")
533 ylabel("Magnitude")
534 title("DFT of x[n] = n")
535 grid on
536 subplot(3,1,2)
537 stem(n,x_ngen)
538 xlabel("Samples")
539 ylabel("Amplitude")
540 title("IDFT of x[k] , Generated from DFT")
541 grid on
542 subplot(3,1,3)
543 stem(n,x_n)
544 xlabel("Samples")
545 ylabel("Amplitude")
546 title("Original function x[n]")
547 grid on
```

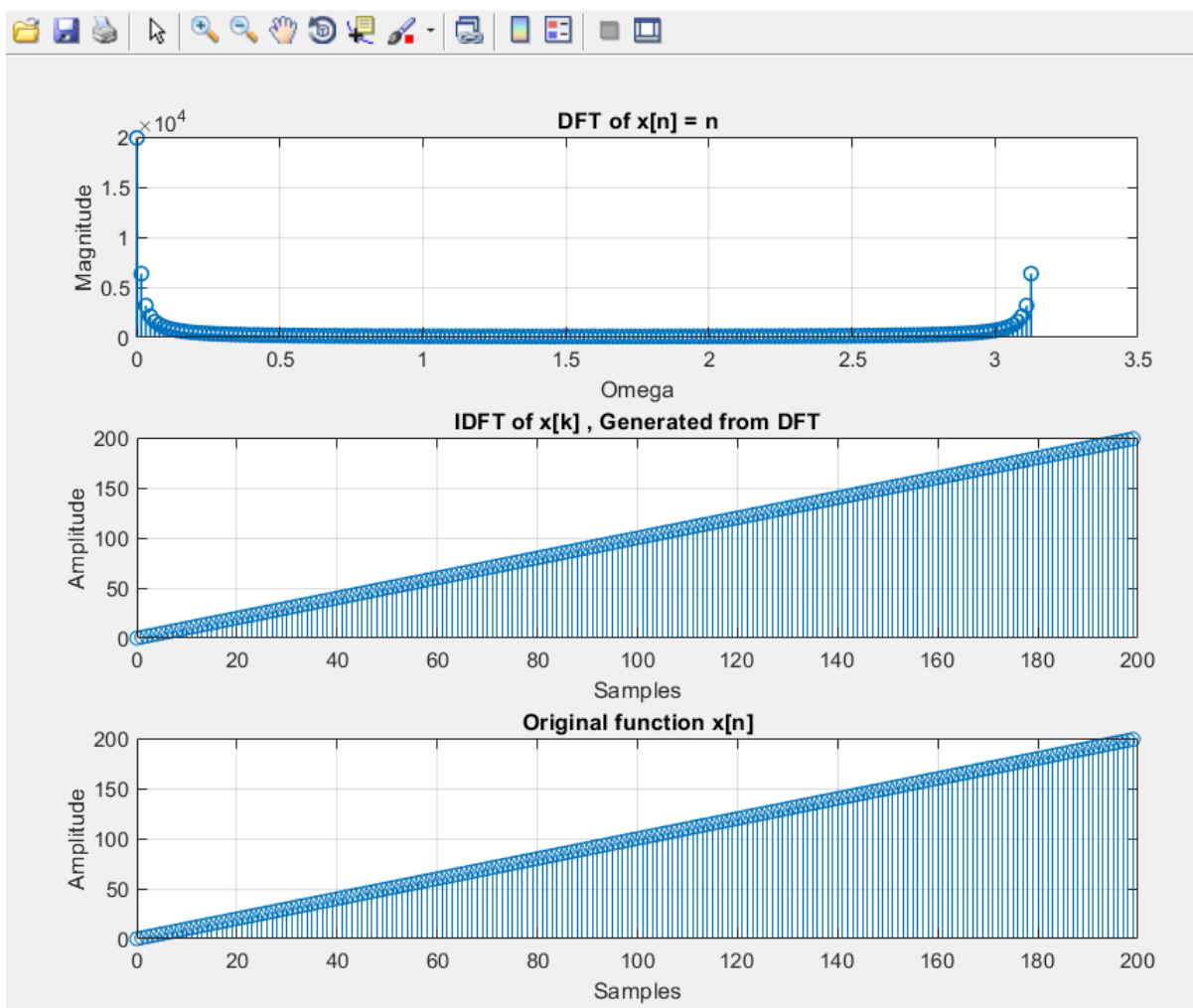



Figure 19: Inverse DFT of signal 2

Conclusion: The function works correctly ,this can be verified by looking at the original and the obtained idft function. They are identical. Also while plotting IDFT of $X[k]$, the imaginary part is ignored since we know the original signal is real. Even if manually real part is not plotted, the matlab does it automatically.

3.4 Q4: Write a program to compute the circular convolution of two length-N sequences

Circular convolution Using DFT IDFT::

$x(n) = [1; 3; -2; 4; 7]$, $h(n) = [3; 1; 21; -3]$:

Code:

```

549 clc;
550 clear;
551
552 %given sequences
553 %y[n]<-->Y[k] = X[k]XH[k]<--> x[n]*h[n] (circular)
554 %('*' Convolution,'X' element wise multiplication)
555 x_n = [1, 3, -2, 4, 7];
556 h_n = [3, 1, 21, -3];
557
558 N = max(length(x_n),length(h_n)); % Finding length of output ...
    signal y = max(len x,len h)
559 x_n = [x_n zeros(1,N-length(x_n))]; %Padding x[n] upto N ...
    samples with zero
560 h_n = [h_n zeros(1,N-length(h_n))]; %Padding h[n] upto N ...
    samples with zero
561 X = nfft(x_n,N); % DFT x[n] = X[k]
562 H = nfft(h_n,N); % DFT h[n] = H[k]
563 Y_k = X.*H; % DFT y[n] = Y[k] = X[k] x H[k]
564 y_n = ifft(Y_k,length(Y_k)); % IDFT Y[k] = y[n]
565 stem(y_n, 'LineWidth',1);
566 xlabel("Samples")
567 ylabel("Magnitude")
568 title("x[n]*h[n]")
569 hold on
570 disp(y_n);

```

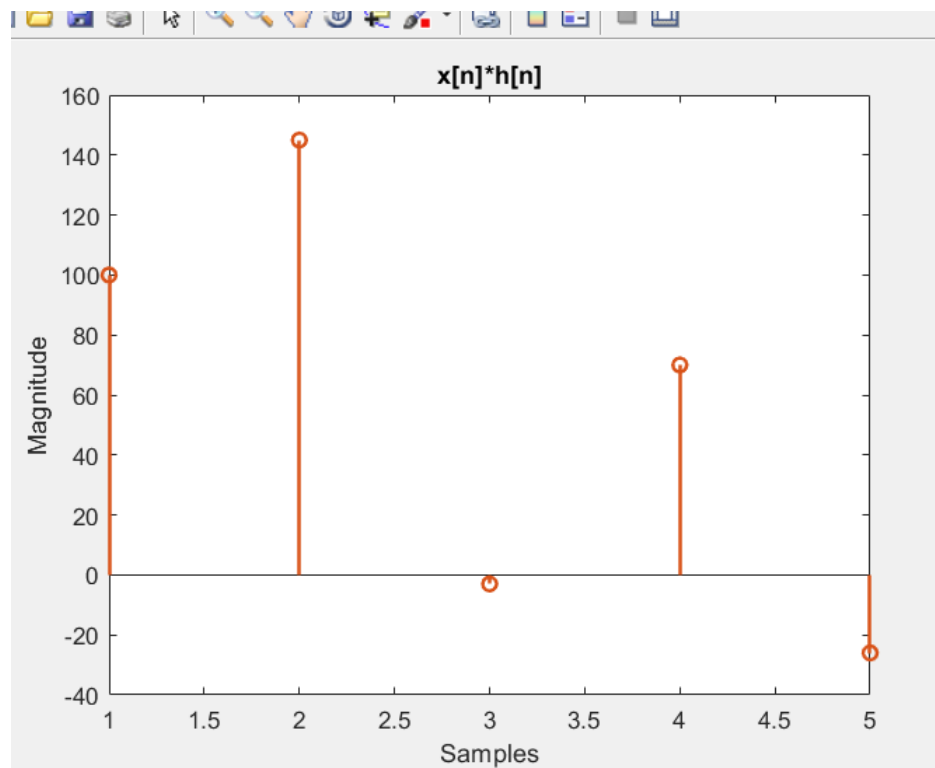


Figure 20: Circular conv using DFT iDFT, signal 1

$$x(n) = n; h(n) = (0.5)^n, 0 \leq n \leq 10:$$

Code:

```

572 clc;
573 clear;
574
575 %given sequences
576 %y[n]<-->Y[k] = X[k]XH[k]<--> x[n]*h[n] (circular)
577 %('*' Convolution,'X' element wise multiplication)
578 n = 0:10;
579 x_n = n;
580 h_n = (0.5).^n;
581
582 N = max(length(x_n),length(h_n)); % Finding length of output ...
    signal y = max(len x,len h)
583 x_n = [x_n zeros(1,N-length(x_n))]; %Padding x[n] upto N ...
    samples with zero
584 h_n = [h_n zeros(1,N-length(h_n))]; %Padding h[n] upto N ...
    samples with zero
585 X = nfft(x_n,N); % DFT x[n] = X[k]
586 H = nfft(h_n,N); % DFT h[n] = H[k]
587 Y_k = X.*H; % DFT y[n] = Y[k] = X[k] x H[k]
588 y_n = ifft(Y_k,length(Y_k)); % IDFT Y[k] = y[n]
589 stem(y_n,'LineWidth',1.5);
590 xlabel("Samples")
591 ylabel("Magnitude")
592 title("(x[n] = n)*(h[n] = 0.5^n)")
593 hold on
594 disp(y_n);

```

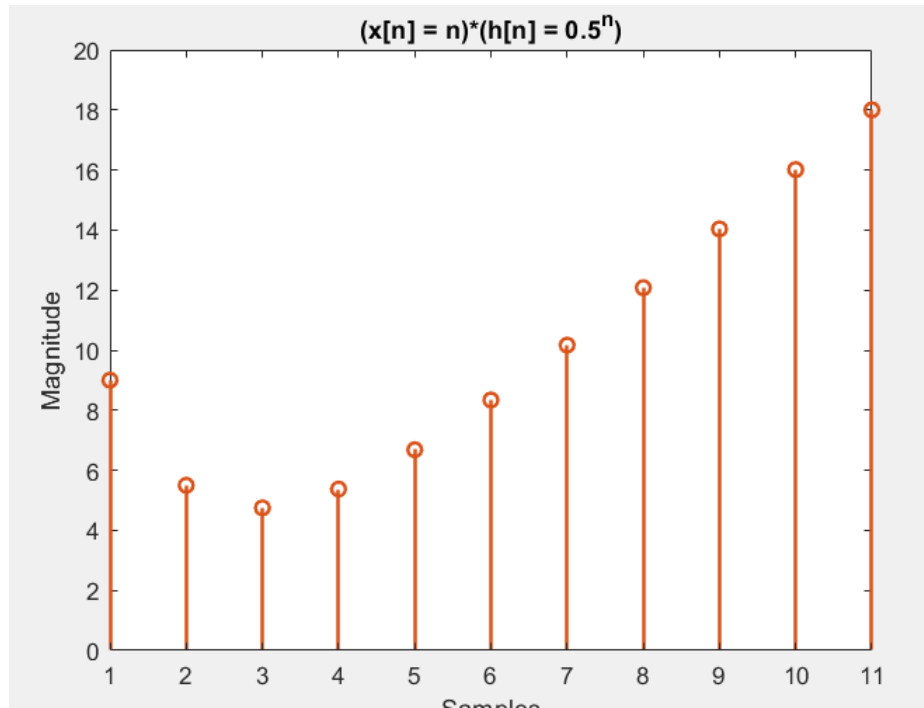


Figure 21: Circular conv using DFT iDFT, signal 2

Conclusion: The function works correctly , verified by comparing result with the inbuilt function cconv separately.

3.5 Q5: Write a program to perform circular convolution in the time domain. Test your

Circular Convolution function::

Code:

```

596 function [y_n] = circ_conv(x_n,h_n)
597 N = max(length(x_n),length(h_n)); %length of convolved signal ...
    required
598 x_n = [x_n zeros(1,N-length(x_n))];
599 h_n = [h_n zeros(1,N-length(h_n))];
600 y_n = zeros(1,N);
601     for n=1:N
602         for i=1:N
603             j=n-i+1;
604             if(j<=0)
605                 j=N+j;
606             end
607             y_n(n)=y_n(n)+(x_n(i)*h_n(j));
608         end
609     end
610 end

```

i) $x(n) = [1; 3; -2; 4; 7]$, $h(n) = [3; 1; 21; -3]$:

Code:

```

611 clc;
612 clear;
613 x_n = [1, 3, -2, 4, 7];
614 h_n = [3, 1, 21, -3];
615
616 y_n = circ_conv(x_n,h_n);
617 stem(y_n,'LineWidth',1.5)
618 xlabel("Sample")
619 ylabel("Amplitude")
620 title("y[n] = (x[n]=[1, 3, -2, 4, 7])*(h[n] = [3, 1, 21, -3])")

```

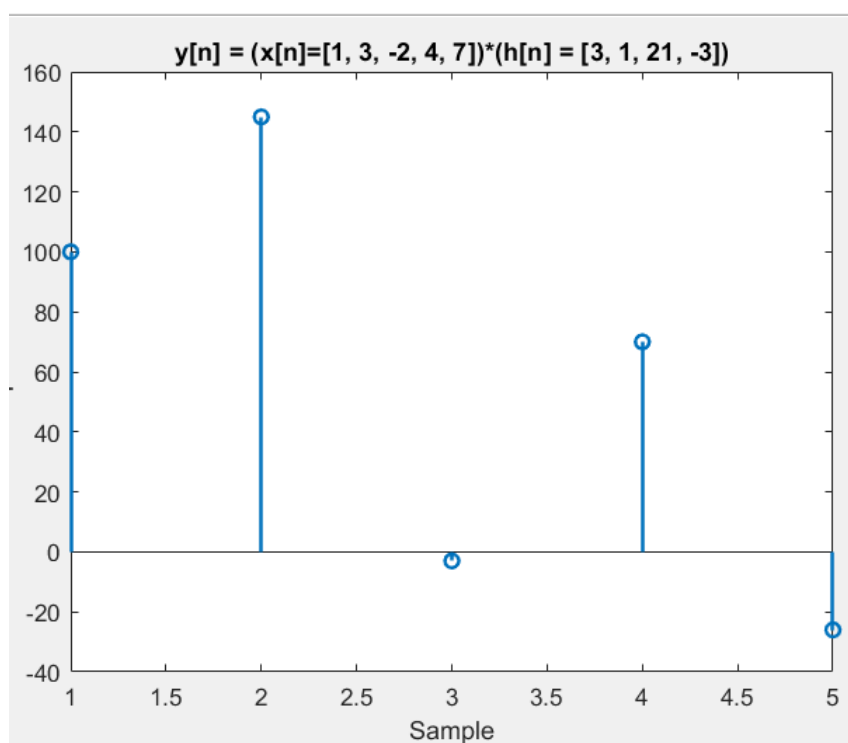


Figure 22: Circular conv signal 1

ii) $x(n) = x_n = n, h_n = (0.5)^n, 0 \leq n \leq 10$:

Code:

```
621 clc;
622 clear;
623 n = 0:10;
624 x_n = n;
625 h_n = (0.5).^n;
626 y_n = circ_conv(x_n, h_n);
627 stem(y_n, 'LineWidth', 1.5)
628 xlabel("Sample")
629 ylabel("Amplitude")
630 title("y[n] = (x_n = n)*(h_n = (0.5)^n)")
```

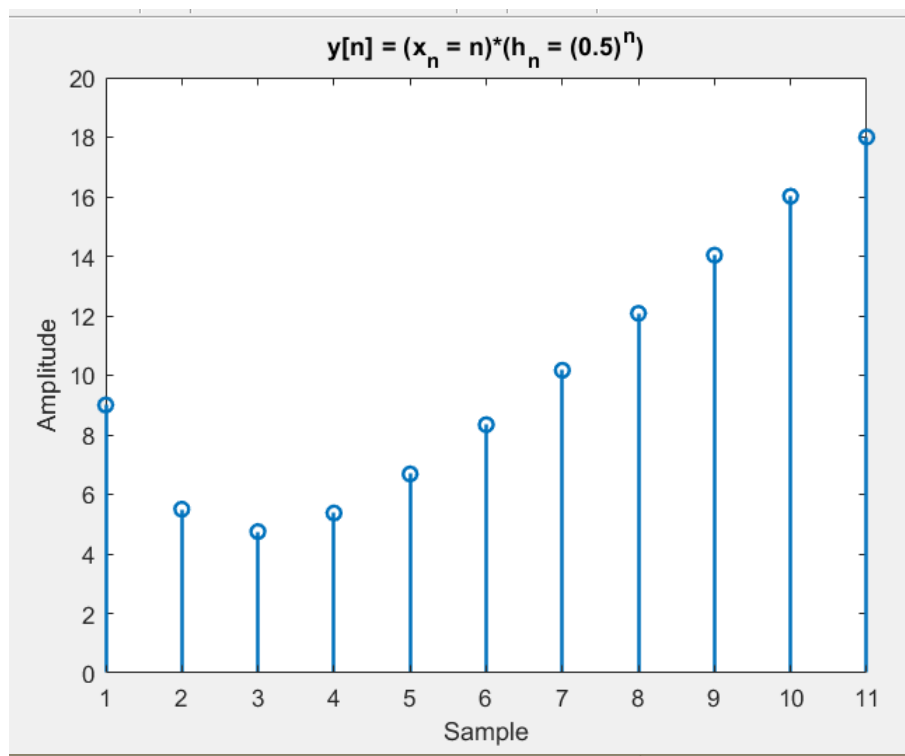


Figure 23: Circular conv signal 2

Conclusion: The function works correctly, verified by comparing result with the inbuilt function `cconv` separately. Also it is to be noted that though the process of obtaining the convolved output was different, the end result was the same in both the cases.

Assignment -4:

4.1 Q1: Implement the equation of Analog Butterworth Low Pass Filter Approximation

Code:

```

631 %Code for task 1
632 %Given specifications
633 nn=80
634 w = 1:nn;
635 w_stop = 25:60;
636 w_pass = 20;
637 As = -25;
638 Ap = -3;
639
640
641 %Order
642 a = 10^(-Gs/10)-1;
643 b = 10^(-Gp/10)-1;
644 numer = log10((10^(-As/10)-1)/(10^(-Ap/10)-1));
645 denom = 2*log10(w_stop./w_pass);
646 n = ...
        ceil((log10((10^(-As/10)-1)/(10^(-Ap/10)-1))./(2*log10(w_stop./w_pass)))
647 omega_c = w_pass./(10^(-Ap/10)-1).^(1./2*n);
648
649 for i=1:numel(n)
650     flt = (1+1i.*(w./omega_c(j)).^(2*n(i))).^-1;
651     magitude = abs(flt);
652     plot(linspace(0,pi,nn),magitude)
653     hold on;
654 end
655 grid on;
656
657 ylabel('Magnitude')
658 xlabel('Normalized frequency')

```

Observations and Discussions: We can observe that the filtered output is noisy sinusoid

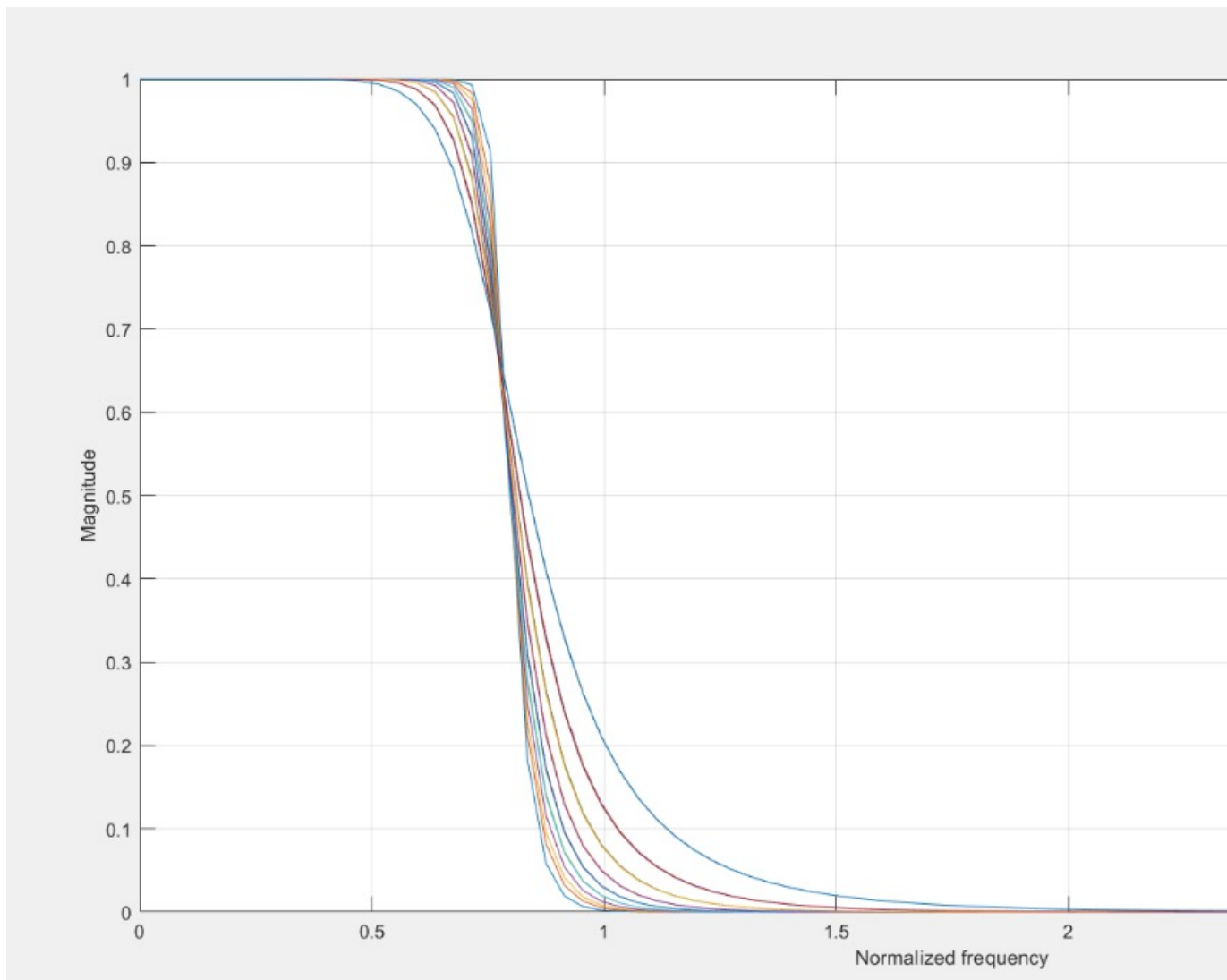


Figure 24: Results

4.2 Q2 - With reference to the IIR LP filter example discussed in the class, generate t

Code:

```

660 %Code for filter task
661 coeffy = [1, -0.919, 0.325];
662 coeffx = [0.102, 0.204, 0.102];
663
664
665 n=0:100;
666 x_1 = sin(0.1*pi*n); % Sinusoid within passband
667 x_2 = sin(0.65*pi*n); % Sinusoid out of passband
668
669 n=0:100;
670 x_add = x_1+x_2; % adding them
671 y = filter( coeffx, coeffy,x_add); % applying filter given
672
673 figure();
674 subplot(411);
675 plot(n,x_1);
676 grid on;
677 title("x1=sin(0.1\pi n");
678
679 subplot(412);
680 plot(n,x_2);
681 grid on;
682 title("x1=sin(0.65\pi n");
683
684 subplot(413);
685 plot(n,x_add);
686 grid on;
687 title("x=x1+x2");
688
689 subplot(414);
690 plot(n,y);
691 grid on;
692 title("filter output on x");

```

Observations and Discussions: We can observe that the filtered output is looking like noisy sinusoid.

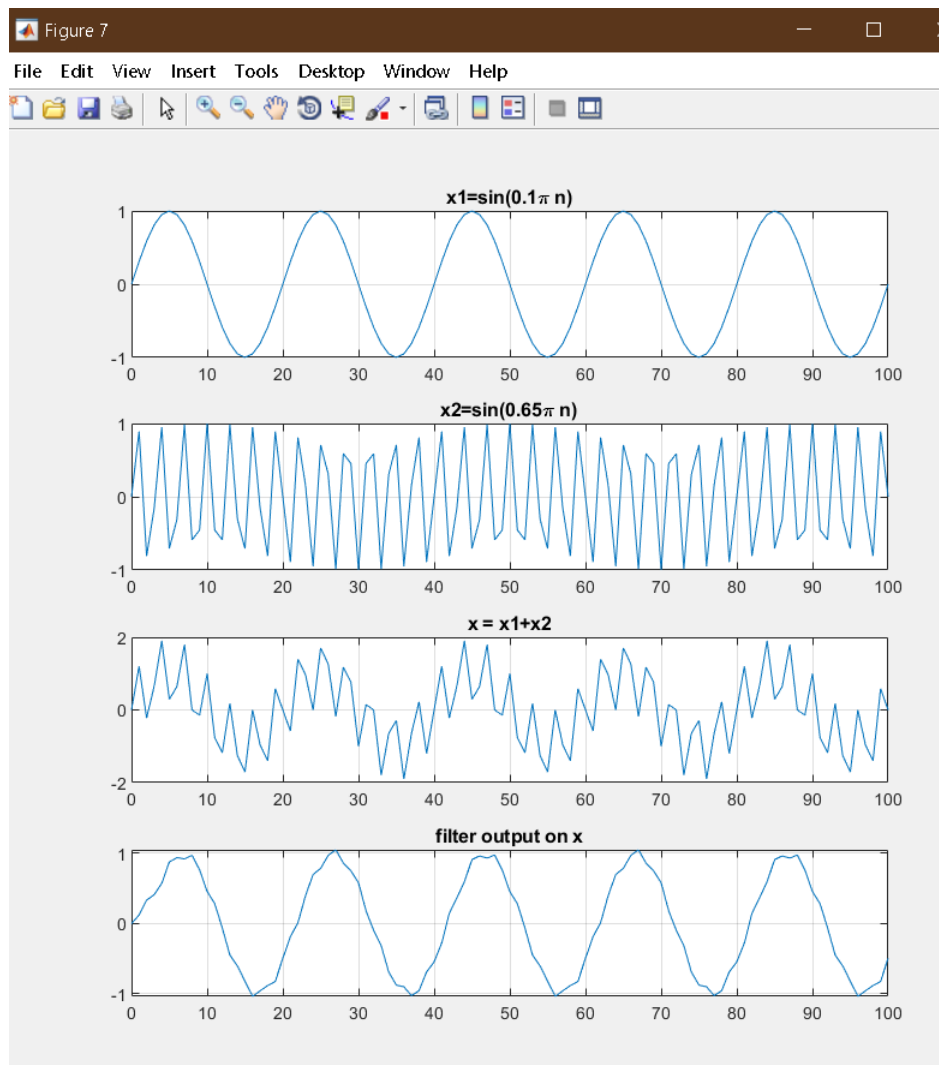


Figure 25: Results

4.3 Q3 - Create a generic filter without inbuilt function.:**Code:**

```
693 %Code for filter task
694 F = 500;
695 Fs = 6000;
696 n = 48;
697 t = linspace(0,n/Fs,n);
698 x_n = sin(2*pi*t*F);
699 y_n = zeros(1,n);
700 y_n(1:3) = [0,0,0];
701 for k = 4:n
702     y_n(k) = x_n(k) + x_n(k-3) + 2.56*y_n(k-1) - 2.2*y_n(k-2) + ...
              0.65*y_n(k-3);
703 end
```

Observations and Discussions: It is a self cited code from 2nd assignment

4.4 Q4- Implement a BPF given the following specifications $wls = 0.1\pi$, $wlp = 0.4\pi$, w

Code:

```

704 % Code for BPF function calculation
705
706 %Given
707 whs= 0.9*pi;
708 wls = 0.1*pi;
709 whp = 0.6*pi;
710 wlp = 0.4*pi;
711 As = -18;
712 Ap = -3;
713
714 %Doing prewarping to get ohmega values(analog freq)
715 osl = tan(wls/2);
716 opl = tan(wlp/2);
717 opu = tan(whp/2);
718 ou = tan(whs/2);
719
720 %For prototype LPF specs
721 o_pass= 1;
722 o_stop = (osu^2-opl*opu)/(osu*(opu-opl));
723
724 %Finding N = order
725 N = ...
       ceil((log10((10^(-As/10)-1)/(10^(-Ap/10)-1))/(2*log10(o_stop/o_pass))));
726
727 %According to this
728 s0 = -1;
729
730 %omega c
731 ohm_c = o_stop/(0^(-As/10)-1)^0.5;
732
733 %frequencies cutoff
734 B = opu - opl;
735 omega_o = (opu*opl)^0.5;
736 B = 0.5*((ohm_c*B)^2 + 4*omg_o^2)^0.5;
737 A = ohm_c*B*0.5;
738 omega_c1 = -A+B;
739 omega_c2 = A+B;
740
741 s = -1
742 %getting H_s
743 s_fin = (s^2 + omega_o^2)/ s(omega_c1 - omega_c2)
744 H_s = 1/s_fin +1
745 % Solving H_s
746 coeff_num = [0.77587, 0, -0.77587]/.(2.77587)
747 coeff_den = [2.77587,0,1.22413]/.(2.77587)

```

```

748
749 freqz(coeff_num , coeff_den)
750
751 t = [0:0.05:2*pi]
752 x = sin(t);
753 Y = filter(coeff_num,coeff_den,x)

```

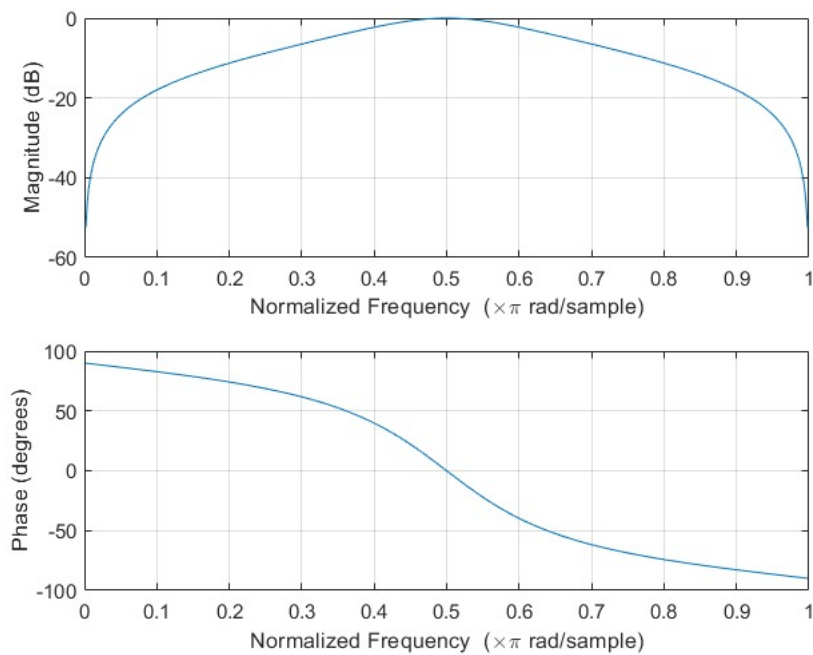


Figure 26: Results

Observations and Discussions: We observe the given frequency response of coefficients.

4.5 Conclusion: On plotting the frequency response of obtained coefficients of the filter we see that band pass filter is formed successfully