
Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

# DIGITAL SIGNAL PROCESSING (DSP)

# ENCS4310

---

MATLAB Assignment

---

Prepared by:

**Jihad Awad - 1210088**

**Mohammed Abed Alkareem - 1210708**

Instructor: **Dr.Alhareth Zyoud**

Date: **Tuesday, August 27, 2024**

## Abstract

In this assignment, multiple signals were plotted in MATLAB, showing spectrums, convolution outputs, pole-zero diagrams as well as filter effects.

## Table of Contents

Abstract ..... 1
Table of Contents ..... 2
Table of Figures ..... 3
Procedure And Discussion ..... 1
1.1 Question I: ..... 1
1.2 Question II: ..... 3
1.3 Question III: ..... 4
1.3.1 A) ..... 4
1.3.2 B) ..... 5
Conclusion ..... 7
Appendices ..... 8
1.1 Question 1 Code ..... 8
1.2 Question II Code ..... 9
1.3 Question III Code ..... 10
1.3.1 A) ..... 10
1.3.2 B) ..... 10
1.3.3 Testing the filter ..... 10

# Table of Figures

FIGURE 1.1.1: MAGNITUDE SPECTRUM OF X[N] ..... 1
FIGURE 1.1.2: PHASE SPECTRUM OF X[N] ..... 2
FIGURE 1.1.3: X[N] & RECONSTRUCTED SIGNAL ..... 2
FIGURE 1.2.1: CONVOLUTION VS MULTIPLICATION AS DFTS ..... 3
FIGURE 1.3.1: POLE ZERO DIAGRAM OF FILTER ..... 4
FIGURE 1.3.2: FILTER SPECTRUM ..... 5
FIGURE 1.3.3: FILTER EFFECT ON TEST SIGNAL ..... 6
FIGURE 1.3.4: FILTERED VS ORIGINAL SIGNAL ..... 6

## Procedure And Discussion

### 1.1 Question I:

$$\boldsymbol{x}[n] = \begin{Bmatrix} 1, n = 1 \ldots .10 \\ 0, Otherwise \end{Bmatrix}$$

The code for Question 1 can be found in the Appendix

Using FFT (which performs DFT), the signal's magnitude and phase were plotted as shown in Figure 1.1.1 and Figure 1.1.2 respectively.

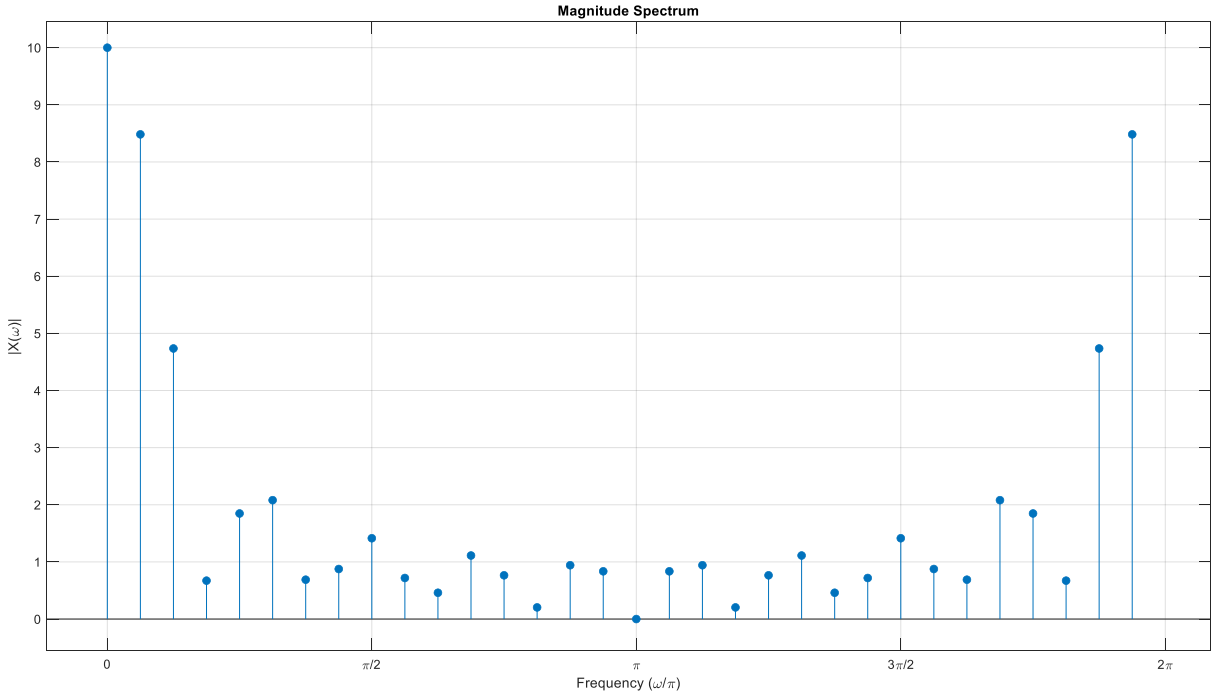


Figure 1.1.1: Magnitude Spectrum of x[n]

The magnitude spectrum of the signal, which looks like a square wave before conversion, is similar to a sinc function, however, DFT's result is discrete and periodic, so the sinc-like result will be discrete points that get repeated periodically, while the peak magnitude is the sum of all points (10 points of 1 mag each), which is 10.

The x-axis was converted to ω in π notation and labelled for clarity, to match the result of a DFT, the same was done for the rest of the DFT plots.

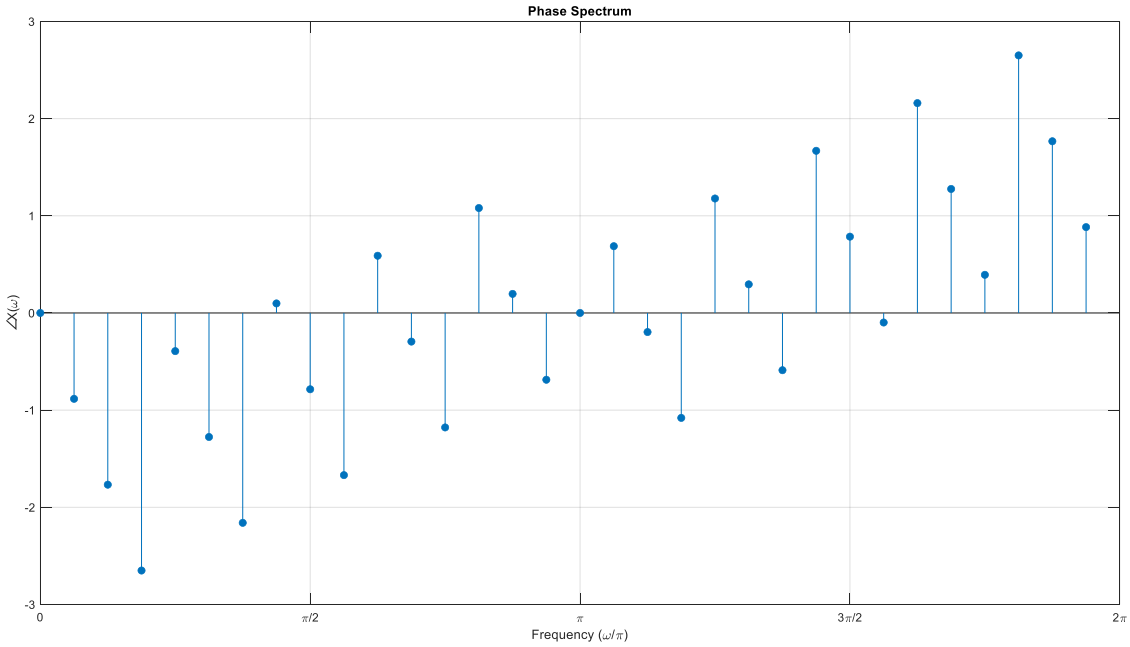


Figure 1.1.2: Phase Spectrum of x[n]

The phase spectrum repeats every 3 or M points while increasing every period, this is due to how the magnitude looks and the way it similarly repeats.

The phase in each period is linear (decreasing) which is similar to its result in the time domain if it were a continuous signal.

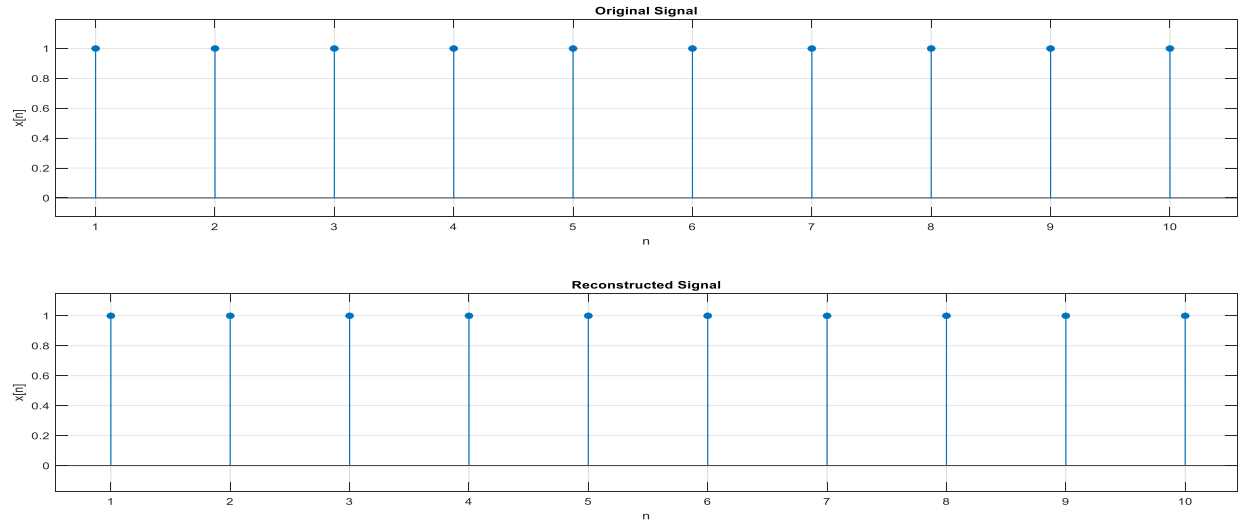


Figure 1.1.3: x[n] & Reconstructed Signal

The signal was reconstructed with IFFT in order to confirm the results.

## 1.2 Question II:

$$x[n] = [0, 0.3, 0.6, 0.8, 1]$$

$$h[n] = [0.5, 1, 0.5]$$

The code for Question 2 can be found in the Appendix

After padding the signals and performing FFT on both, they were multiplied then converted back to the time domain using IFFT. The original signals were also convoluted using the conv function, giving us the result in Figure 1.2.1:

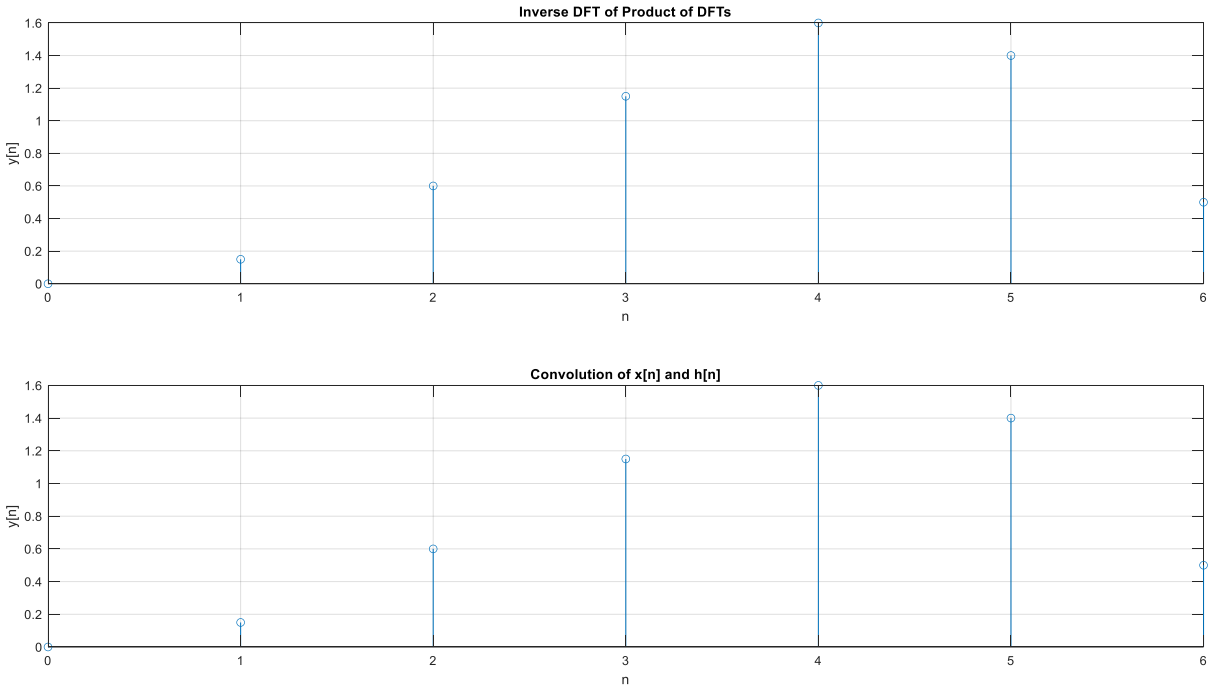


Figure 1.2.1: Convolution VS Multiplication as DFTs

The resulting plots are the exact same, meaning convolution can be performed through multiplication of the DFTs of the signals, which could be more convenient depending on the signal and its complexity.

- Result from Inverse DFT of Product of DFTs:
  - 0 0.1500 0.6000 1.1500 1.6000 1.4000 0.5000
- Result from Convolution:
  - 0 0.1500 0.6000 1.1500 1.6000 1.4000 0.5000

## 1.3 Question III:

$$h[n] = \{1, 1.5, 1\}$$

### 1.3.1 A)

The code for Question 3 a) can be found in the Appendix

The zero-pole diagram of the filter was plotted using the zplane function, where the zeroes are expected to be at $(\frac{3\pi}{4})$ and the other side $(\frac{5\pi}{4})$, which as shown in Figure 1.3.1 is indeed the case:

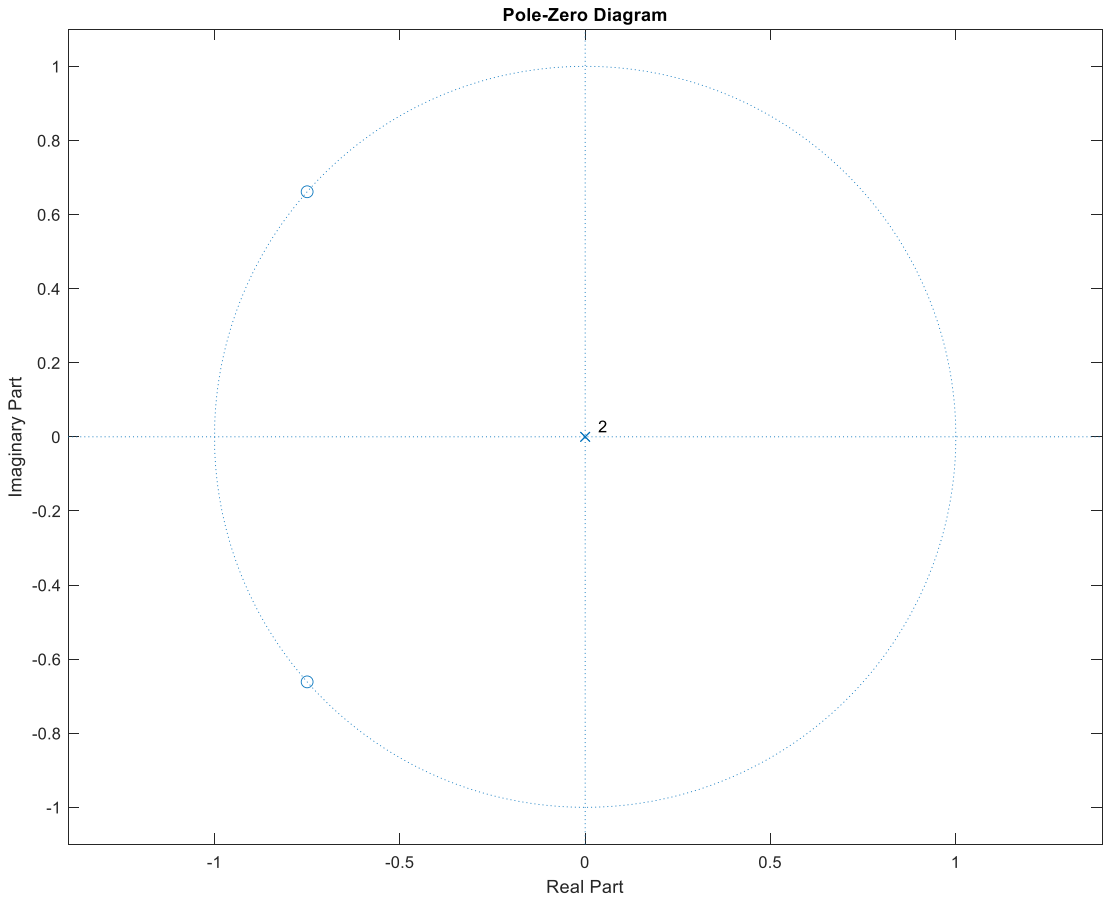


Figure 1.3.1: Pole Zero Diagram of Filter

With a pole at zero, the magnitude should be 0 at the two zeroes, while being attenuated around them, where the further they are, the less attenuated.

### 1.3.2 B)

The code for Question 3 b) can be found in the Appendix

Similar to question 1, the filter's spectrum was plotted using FFT:

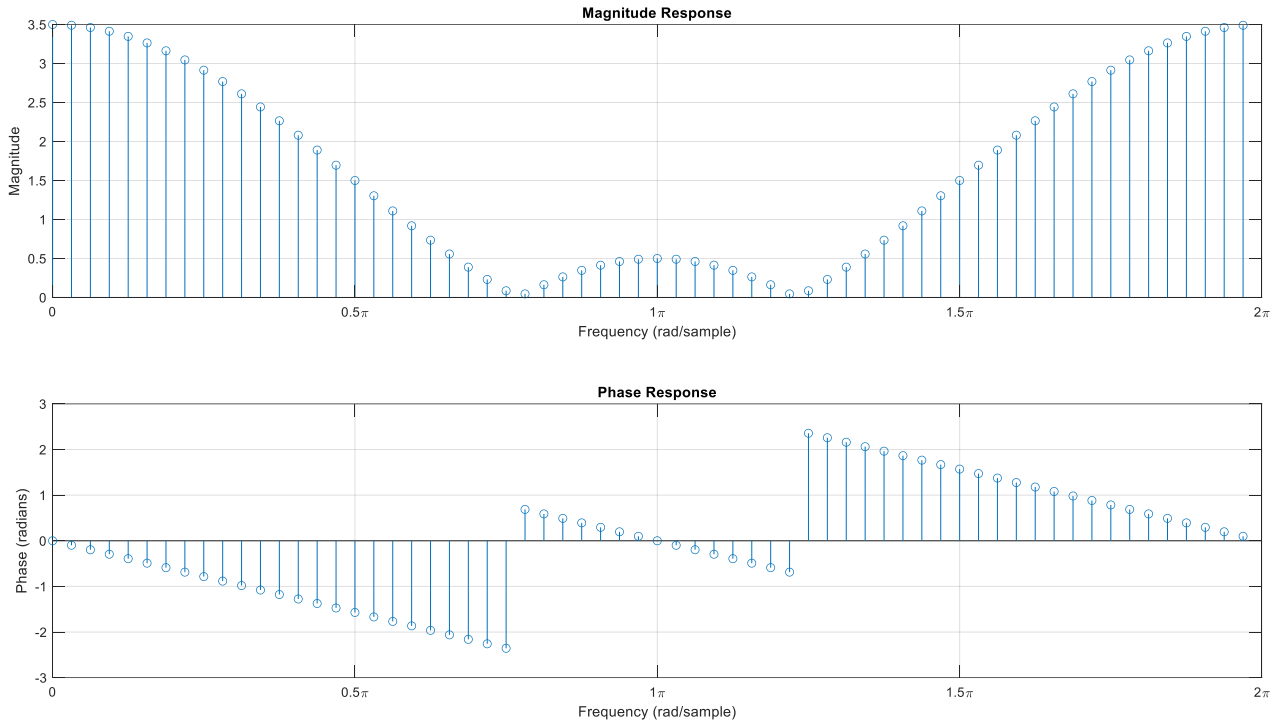


Figure 1.3.2: Filter Spectrum

The magnitude response has a visible dip at around the exact same values as in the plot-zero diagram ($\frac{3\pi}{4}$ & $\frac{5\pi}{4}$), with the phase being linearly decreasing.

The increased attenuation of the function the closer it is to the zeroes, is what you would expect of a filter.

In order to test it, a signal consisting of frequencies at 100, 300 and 700Hz of equal amplitude was convoluted with the filter as shown in Figure 1.3.3.

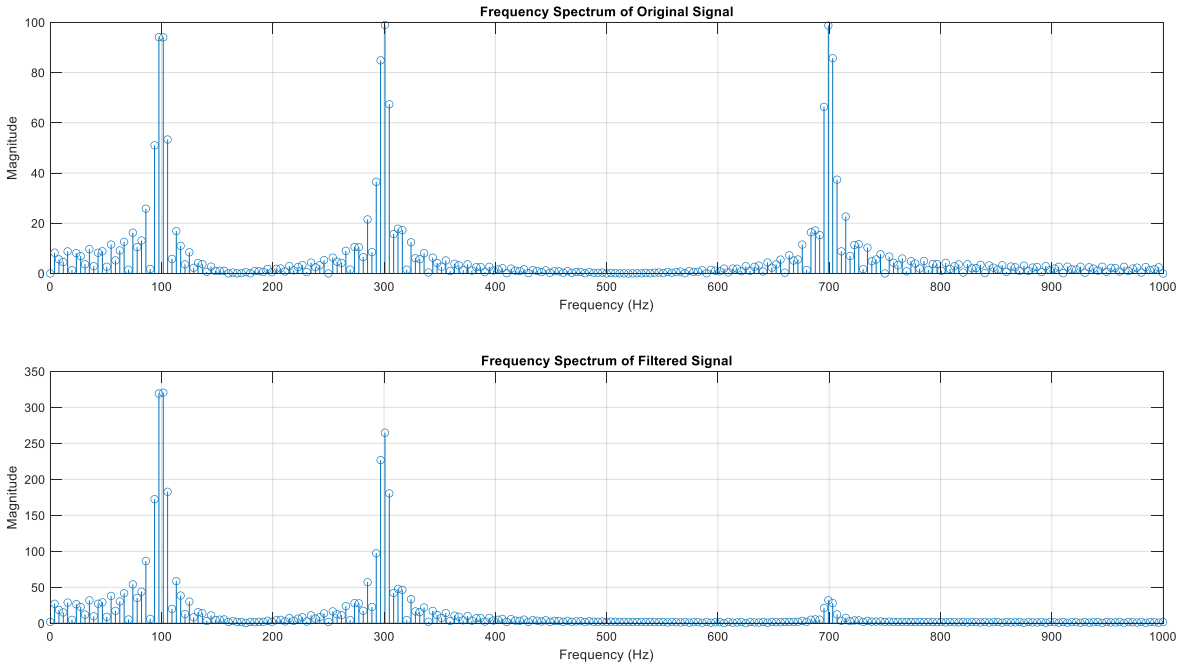


Figure 1.3.3: Filter Effect on Test Signal

The figure clearly shows the frequencies being attenuated the higher they are, where 100Hz remained the same, 300Hz was slightly attenuated, and 700Hz was almost removed, meaning the filter is low-pass.

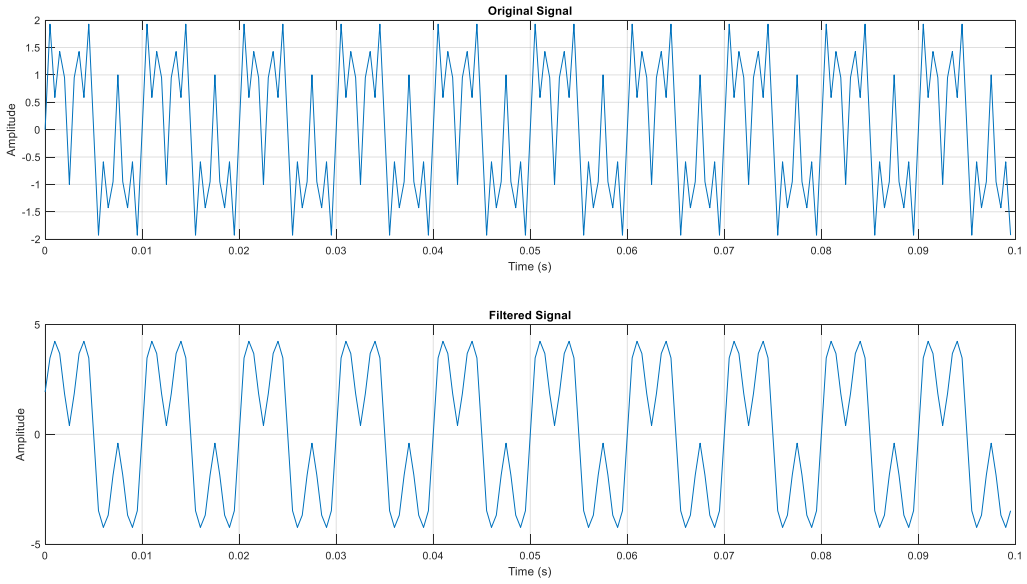


Figure 1.3.4: Filtered Vs Original Signal

## Conclusion

All the signals were successfully plotted and tested to be correct using MATLAB, the effects of DFTs, and the meaning of pole-zero diagrams as well as their effect on the spectrum were consistent with the theory learned throughout the course, confirming their usefulness in practical settings.

# Appendices

## 1.1 Question 1 Code

```
N = 32; % Number of points, bigger for clarity
x = [ones(1,10), zeros(1,N-10)];

% Calculate DFT
X = fft(x);

% Calculate frequency axis (in pi notation)
k = 0:N-1;
w = (k/N)*2;

% Plot the magnitude spectrum
figure(1);
stem(w, abs(X), 'filled');
xlabel('Frequency (\omega/\pi)');
ylabel('|X(\omega)|');
title('Magnitude Spectrum');
xlim([0 2]);
xticks(0:0.5:2);
xticklabels({'0', '\pi/2', '\pi', '3\pi/2', '2\pi'});
grid on;

% Plot the phase spectrum
figure(2);
stem(w, angle(X), 'filled');
xlabel('Frequency (\omega/\pi)');
ylabel('∠X(\omega)');
title('Phase Spectrum');
xlim([0 2]);
xticks(0:0.5:2);
xticklabels({'0', '\pi/2', '\pi', '3\pi/2', '2\pi'});
grid on;

% Reconstruct the signal using ifft (to check)
x_reconstructed = ifft(X);

% Plot original and reconstructed signals (only the first 10 points)
figure(3);
subplot(2,1,1);
stem(1:10, x(1:10), 'filled');
title('Original Signal');
xlabel('n');
ylabel('x[n]');
ylim([0 1.2]);
grid on;

subplot(2,1,2);
stem(1:10, x_reconstructed(1:10), 'filled');
title('Reconstructed Signal');
xlabel('n');
ylabel('x[n]');
```

```
ylim([0 1.2]);
grid on;
```

## 1.2 Question II Code

```
% Define the signals
x = [0, 0.3, 0.6, 0.8, 1];
h = [0.5, 1, 0.5];

% Zero-pad to the same length
N = length(x) + length(h) - 1;
x_padded = [x, zeros(1, N - length(x))];
h_padded = [h, zeros(1, N - length(h))];

% Calculate the DFTs
X = fft(x_padded);
H = fft(h_padded);

% Element-wise multiplication of DFTs (since they're discrete)
Y = X .* H;

% Inverse DFT to return to the time domain
y_dft = ifft(Y);

% Calculate the convolution of x[n] and h[n]
y_conv = conv(x, h);

% Display the results as text to check
disp('Result from Inverse DFT of Product of DFTs:');
disp(y_dft);
disp('Result from Convolution:');
disp(y_conv);

% Plot the results
figure;
subplot(2,1,1);
stem(0:length(y_dft)-1, y_dft);
title('Inverse DFT of Product of DFTs');
xlabel('n');
ylabel('y[n]');
grid on;

subplot(2,1,2);
stem(0:length(y_conv)-1, y_conv);
title('Convolution of x[n] and h[n]');
xlabel('n');
ylabel('y[n]');
grid on;
```

## 1.3 Question III Code

### 1.3.1 A)

```
% Define the impulse response
h = [1, 1.5, 1];

% Calculate H(z) using FFT for frequency response
N = 64; % Number of points for FFT
H = fft(h, N);
k = 0:N-1;
w = 2*pi*k/N;

% Plot the pole-zero diagram
figure(1);
zplane(h);
title('Pole-Zero Diagram');
```

### 1.3.2 B)

```
% Note: Continues from (A) above

% Plot the frequency response (using stem for discrete values)
figure(2);
subplot(2,1,1);
stem(w, abs(H));
title('Magnitude Response');
xlabel('Frequency (rad/sample)');
ylabel('Magnitude');
xticks([0 pi/2 pi 3*pi/2 2*pi]);
xticklabels({'0', '0.5\pi', '1\pi', '1.5\pi', '2\pi'});
grid on;
xlim([0 2*pi]);

subplot(2,1,2);
stem(w, angle(H));
title('Phase Response');
xlabel('Frequency (rad/sample)');
ylabel('Phase (radians)');
xticks([0 pi/2 pi 3*pi/2 2*pi]);
xticklabels({'0', '0.5\pi', '1\pi', '1.5\pi', '2\pi'});
grid on;
xlim([0 2*pi]);
```

### 1.3.3 Testing the filter

```
% Test the filter with a sample signal
fs = 2000; % Sample rate
t = 0:1/fs:0.1-1/fs; % Time vector (0.1 seconds)

% Create a test signal with multiple frequency components
f1 = 100;
f2 = 300;
f3 = 700;
x = sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t);
```

```
% Apply the filter using convolution
y = conv(x, h, 'same'); % 'same' returns the central part of the convolution
only

% Plot the original and filtered signals
figure(3);
subplot(2,1,1);
plot(t, x);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
plot(t, y);
title('Filtered Signal');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;


N_freq = 512;  % for more clarity when plotted

% Frequency Spectrum of Original Signal
X_freq = fft(x, N_freq);
f = fs*(0:(N_freq/2))/N_freq;

figure(4);
subplot(2,1,1);
stem(f, abs(X_freq(1:length(f))));
title('Frequency Spectrum of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Frequency Spectrum of Filtered Signal
Y_freq = fft(y, N_freq);
f = fs*(0:(N_freq/2))/N_freq;

subplot(2,1,2);
stem(f, abs(Y_freq(1:length(f))));
title('Frequency Spectrum of Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```