DSP Lab 9
name: Basil khowaja bk08432

Task 1)

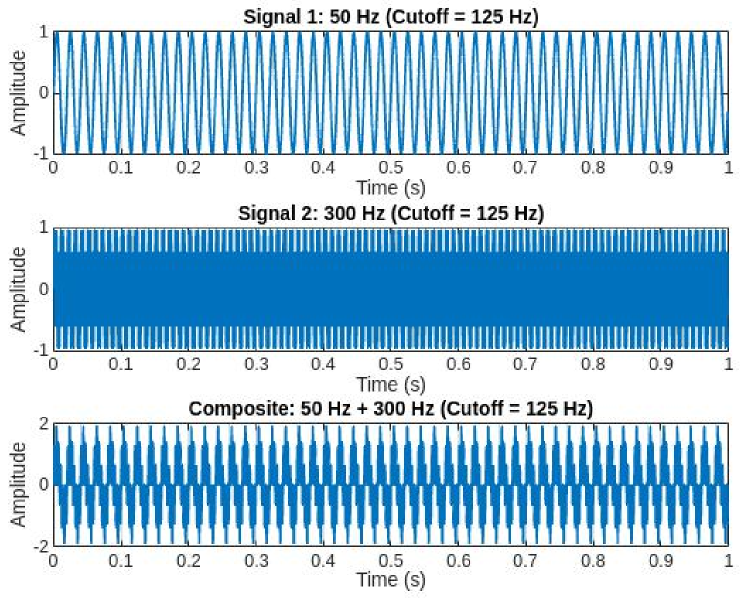
code:

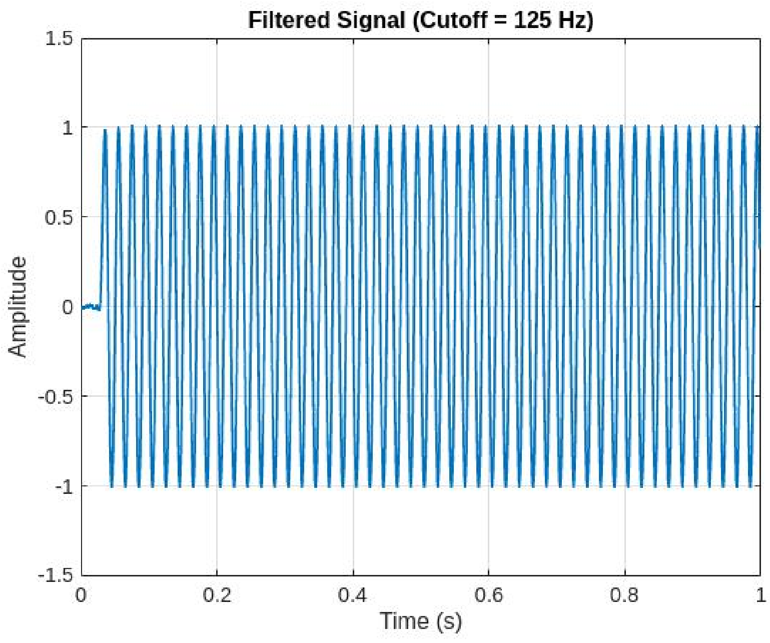
```
%task1:
clc; clear; close all;


L = 61;
M = (L - 1) / 2;
omega_c = 0.25 * pi;
fc = 125;
n = 0 : (L - 1);


h = zeros(size(n));
for k = 1:length(n)
if n(k) == M
h(k) = omega_c / pi;
else
h(k) = sin(omega_c*(n(k) - M)) / (pi*(n(k) - M));
end
end


fs = 1000;
t = 0 : 1/fs : 1 - 1/fs;


x1_freq = 50;
x2_freq = 300;


x1 = sin(2*pi*x1_freq * t);
x2 = sin(2*pi*x2_freq * t);
x = x1 + x2;


figure('Name','Original Signals');
subplot(3,1,1);
plot(t, x1, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
```

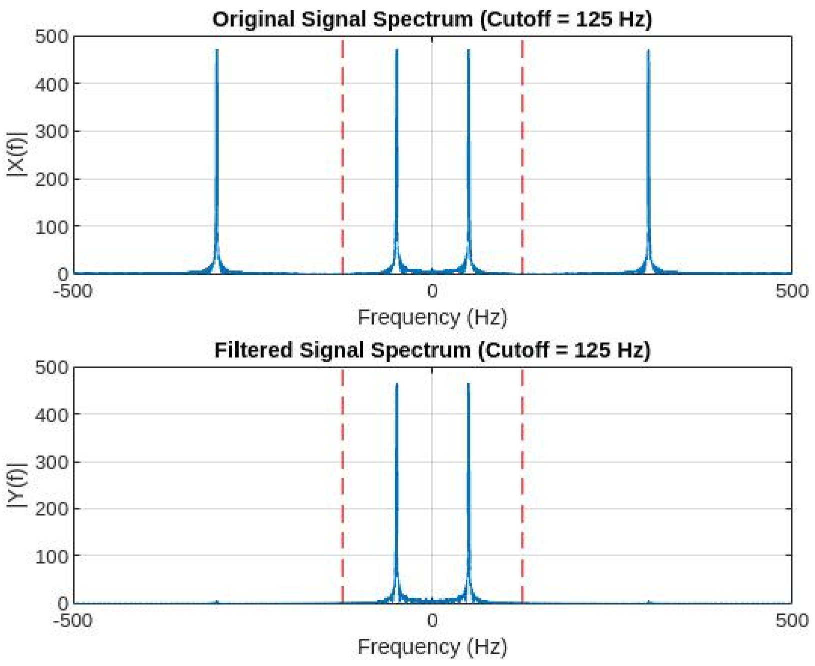
```matlab
title(sprintf('Signal 1: %d Hz (Cutoff = %d Hz)', x1_freq, fc));
grid on;


subplot(3,1,2);
plot(t, x2, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
title(sprintf('Signal 2: %d Hz (Cutoff = %d Hz)', x2_freq, fc));
grid on;


subplot(3,1,3);
plot(t, x, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
title(sprintf('Composite: %d Hz + %d Hz (Cutoff = %d Hz)', x1_freq, x2_freq, fc));
grid on;


y = filter(h, 1, x);


figure('Name','Filtered Signal');
plot(t, y, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
title(sprintf('Filtered Signal (Cutoff = %d Hz)', fc));
grid on;


N_fft = 2048;
X = fft(x, N_fft);
X_shifted = fftshift(X);


Y = fft(y, N_fft);
Y_shifted = fftshift(Y);


freq_axis = linspace(-fs/2, fs/2, N_fft);


figure('Name','Frequency Domain with fftshift');


subplot(2,1,1);
plot(freq_axis, abs(X_shifted), 'LineWidth', 1.2);
hold on;
```

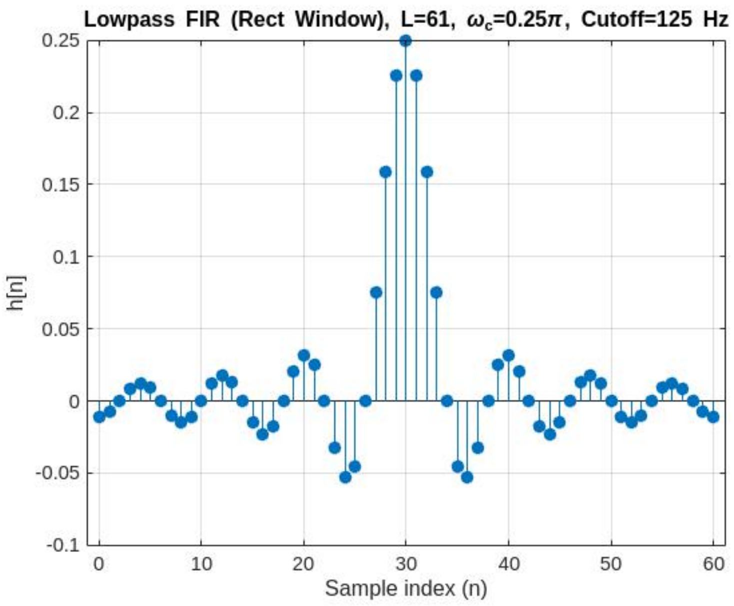
```
xline(-fc, 'r--', 'LineWidth', 1);
xline(fc, 'r--', 'LineWidth', 1);
hold off;
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title(sprintf('Original Signal Spectrum (Cutoff = %d Hz)', fc));
grid on;


% Filtered Spectrum
subplot(2,1,2);
plot(freq_axis, abs(Y_shifted), 'LineWidth', 1.2);
hold on;
xline(-fc, 'r--', 'LineWidth', 1);
xline(fc, 'r--', 'LineWidth', 1);
hold off;
xlabel('Frequency (Hz)');
ylabel('|Y(f)|');
title(sprintf('Filtered Signal Spectrum (Cutoff = %d Hz)', fc));
grid on;


%% 8) Impulse Response Plot (Optional)
figure('Name','Impulse Response');
stem(n, h, 'filled');
xlabel('Sample index (n)');
ylabel('h[n]');
title(sprintf('Lowpass FIR (Rect Window), L=%d, \\omega_c=0.25\\pi, Cutoff=%d
Hz', L, fc));
grid on;
```

output:


Signal 1: 50 Hz (Cutoff = 125 Hz)
Signal 2: 300 Hz (Cutoff = 125 Hz)
Composite: 50 Hz + 300 Hz (Cutoff = 125 Hz)
Amplitude
Time (s)


Filtered Signal (Cutoff = 125 Hz)
Amplitude
Time (s)





observation:

we start by setting up the filter parameters for our lowpass FIR filter. We choose a filter length L=61, which is an odd number. This choice is important because an odd-length filter allows us to have a symmetric impulse response, ensuring that the filter has linear phase. The midpoint of the filter is calculated as M=(L−1)/2Min this case, M equals 30. The cutoff frequency for our filter is defined as ωc=0.25π   =0.25π (in radians per sample). Given a sampling rate of fs=1000 this digital cutoff frequency corresponds to an analog cutoff of approximately 125 Hz. In practical terms, our filter is designed to allow signals below 125 Hz to pass through while attenuating signals above that frequency.

Next, the code constructs the ideal impulse response of a lowpass filter using the sinc function. In theory, an ideal lowpass filter has an impulse response that extends infinitely, following the sinc function However, because an infinite-length filter is not practical, we truncate the impulse response to a finite length of 61 samples. The code does this by looping over the index nnn from 0 to 60. At the center n=Mn, where the expression would result in a division by zero, we use the limit value ωc/π. For all other values of n, the impulse response is computed normally using the sinc formula. This truncated impulse response approximates the behavior of an ideal lowpass filter within the selected sample range.

Following the filter design, two test signals are generated. One signal has a frequency of 50 Hz, which is below the cutoff of 125 Hz, and the other has a frequency of 300 Hz, which is above the cutoff. These signals are individually generated as sine waves and then summed to form a composite signal. The rationale here is that the 50 Hz component should pass through the lowpass filter almost unchanged, while the 300 Hz component should be significantly attenuated.

The composite signal is then filtered by convolving it with the FIR filter impulse response using the filter function. In the time-domain plot of the filtered signal, you will notice that the output closely resembles a 50 Hz sine wave. This indicates that the filter has effectively removed most of the 300 Hz component, confirming that our lowpass filter works as intended.

To further validate the filter's performance, we analyze the signals in the frequency domain. The code computes the Fast Fourier Transform (FFT) of both the original and the filtered composite signals using a 2048-point FFT, which gives a high-resolution spectrum. The FFT output is then shifted using fftshift so that zero frequency is centered, and a frequency axis is created ranging from In the frequency-domain plot of the original signal, wecan see prominent peaks corresponding to both 50 Hz and 300 Hz. In contrast, the frequency-domain plot of the filtered signal shows that the peak at 300 Hz is greatly diminished while the 50 Hz peak remains strong. Additionally, red dashed lines are drawn at −125 Hz-125 to mark the cutoff boundaries, making it clear that the filter is passing frequencies below 125 Hz and attenuating those above.

Finally, the code also produces a stem plot of the filter’s impulse response, displaying the discrete values of h[n]. This plot shows a central peak at n=30 with a series of smaller lobes on either side, which is characteristic of a truncated sinc function that has been windowed by a rectangular function. The symmetry of this plot confirms the linear-phase property of the filter.

In summary, Task 1 demonstrates how to design a lowpass FIR filter using a truncated sinc (rectangular window) approach. The code clearly shows the creation of the filter’s impulse response, the generation and combination of test signals, the filtering process, and both time-domain and frequency-domain analyses. The overall observations indicate that the filter effectively passes low frequencies (like 50 Hz) while attenuating higher frequencies (like 300 Hz), which is exactly the desired behavior for a lowpass filter with a 125 Hz cutoff.

Task 2)

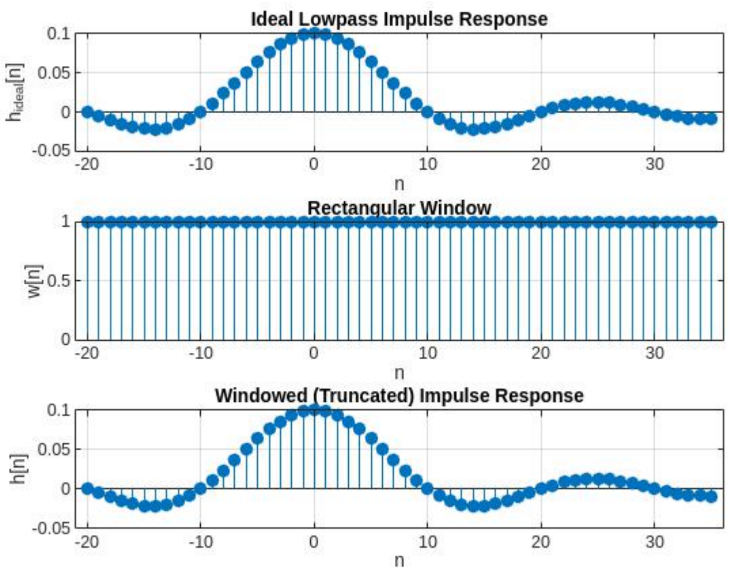
code;

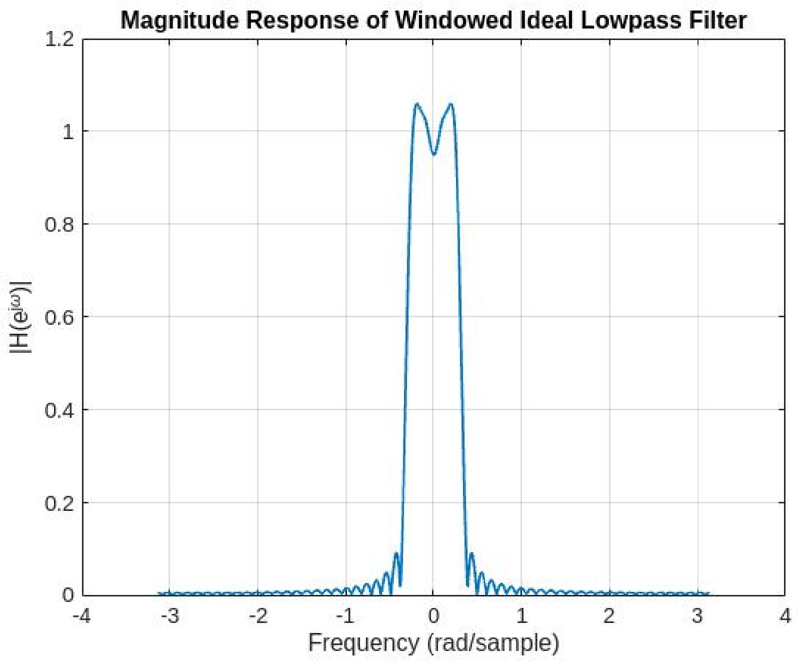
```
%task2:
clc; clear; close all;
n = -20:35;
omega_c = 0.1 * pi;
h_ideal = zeros(size(n));
for k = 1:length(n)
if n(k) == 0
h_ideal(k) = omega_c / pi;
else
h_ideal(k) = sin(omega_c * n(k)) / (pi * n(k));
end
end


w_rect = ones(size(n));
h_windowed = h_ideal .* w_rect;


figure('Name','Windowing Process');
subplot(3,1,1);
stem(n, h_ideal, 'filled');
xlabel('n'); ylabel('h_{ideal}[n]');
title('Ideal Lowpass Impulse Response');
grid on;
```

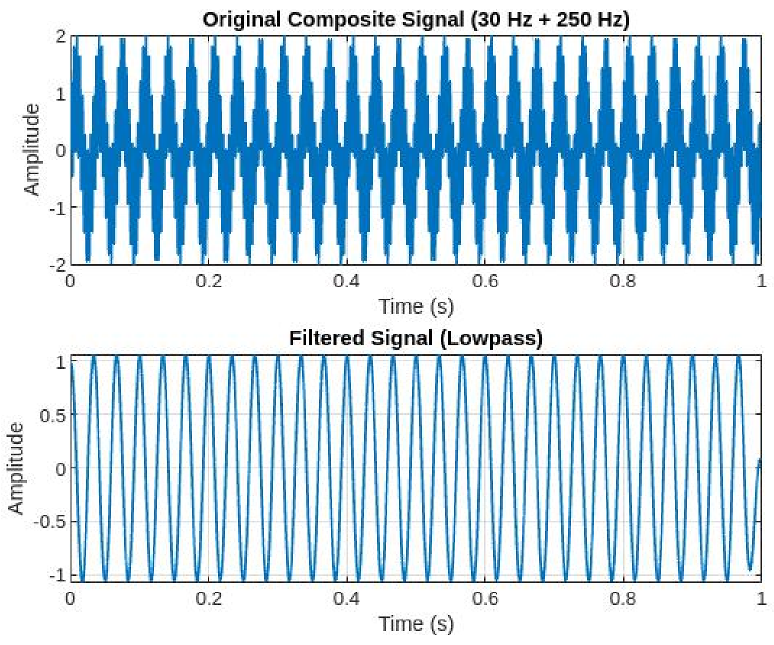
```
subplot(3,1,2);
stem(n, w_rect, 'filled');
xlabel('n'); ylabel('w[n]');
title('Rectangular Window');
grid on;


subplot(3,1,3);
stem(n, h_windowed, 'filled');
xlabel('n'); ylabel('h[n]');
title('Windowed (Truncated) Impulse Response');
grid on;


N_fft = 1024;
H = fft(h_windowed, N_fft);
H_shifted = fftshift(H);
freq_axis = linspace(-pi, pi, N_fft);


figure('Name','Frequency Response of Windowed Filter');
plot(freq_axis, abs(H_shifted), 'LineWidth', 1.2);
xlabel('Frequency (rad/sample)');
ylabel('|H(e^{j\omega})|');
title('Magnitude Response of Windowed Ideal Lowpass Filter');
grid on;


fs = 1000;
t = 0 : 1/fs : 1-1/fs;
f1 = 30;
f2 = 250;


signal1 = sin(2*pi*f1 * t);
signal2 = sin(2*pi*f2 * t);
composite_signal = signal1 + signal2;


filtered_signal = conv(composite_signal, h_windowed, 'same');


figure('Name','Time-Domain Signals');
subplot(2,1,1);
plot(t, composite_signal, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title(sprintf('Original Composite Signal (%d Hz + %d Hz)', f1, f2));
grid on;
```

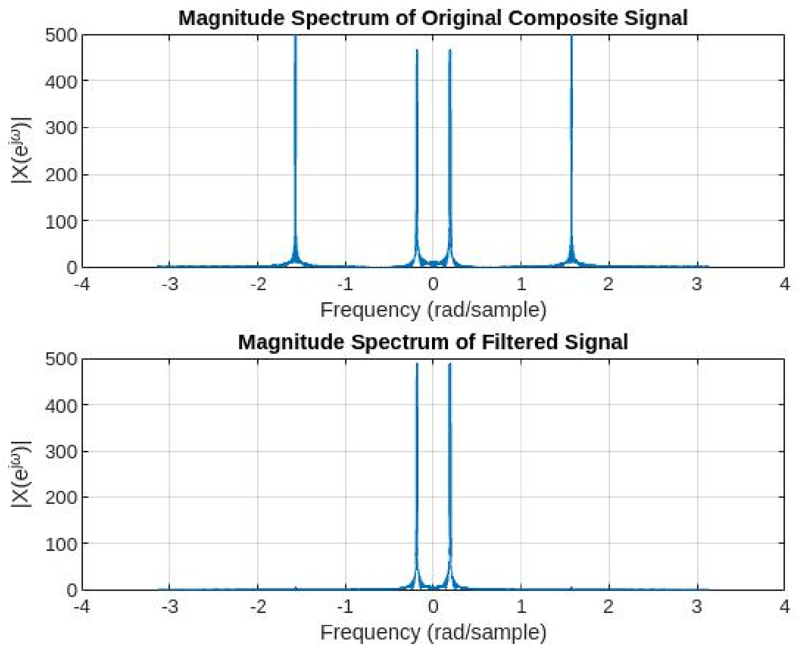
```
subplot(2,1,2);
plot(t, filtered_signal, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Filtered Signal (Lowpass)');
grid on;


N_fft_sig = 2048;
Composite_fft = fft(composite_signal, N_fft_sig);
Filtered_fft = fft(filtered_signal, N_fft_sig);
Composite_fft_shifted = fftshift(Composite_fft);
Filtered_fft_shifted = fftshift(Filtered_fft);


freq_axis_sig = linspace(-pi, pi, N_fft_sig);


figure('Name','Frequency-Domain Signals');
subplot(2,1,1);
plot(freq_axis_sig, abs(Composite_fft_shifted), 'LineWidth', 1.2);
xlabel('Frequency (rad/sample)');
ylabel('|X(e^{j\omega})|');
title('Magnitude Spectrum of Original Composite Signal');
grid on;


subplot(2,1,2);
plot(freq_axis_sig, abs(Filtered_fft_shifted), 'LineWidth', 1.2);
xlabel('Frequency (rad/sample)');
ylabel('|X(e^{j\omega})|');
title('Magnitude Spectrum of Filtered Signal');
grid on;
```

outputs:


Ideal Lowpass Impulse Response
$h_{ideal}[n]$
n
Rectangular Window
w[n]
n
Windowed (Truncated) Impulse Response
h[n]
n


Magnitude Response of Windowed Ideal Lowpass Filter
$|H(e^{j\omega})|$
Frequency (rad/sample)





Observation:

First, we define a range of sample indices n from -20 to 35. This defines the length of our impulse response, which represents how our lowpass filter will react to an impulse input. The choice of ω_c = 0.1π means that we are designing a lowpass filter, which will allow frequencies below 0.1π to pass while blocking higher frequencies.

Then, we compute the ideal impulse response using the sinc function. The sinc function is fundamental in filter design because it represents an ideal lowpass filter in the frequency domain. The sinc function extends infinitely, meaning it provides a perfect frequency cutoff, but in reality, we cannot implement an infinite-length filter. That’s why we need to truncate it to a finite length.

Since truncating a sinc function directly causes unwanted ripples in the frequency domain, we apply a window function to smoothen these ripples. Here, we use a rectangular window, which is simply a set of ones multiplied with the impulse response. This means that we are keeping the sinc function’s values without any extra weighting. The first subplot in the "Windowing Process" figure shows the ideal sinc function impulse response, and the second subplot shows the rectangular window itself.

After applying the window, we get the final truncated impulse response, which is shown in the third subplot. This response still retains the general shape of the sinc function but is now limited in length, making it practically implementable in real systems.

Now, we check the frequency response of this truncated filter. The magnitude plot shows that the lowpass filter has a peak around 0 and gradually reduces at higher frequencies, which means it is attenuating high frequencies while allowing low frequencies to pass. However, since we applied a rectangular window, we see some side lobes—small ripples in the magnitude response—which represent undesired frequencies not being completely blocked.

To test how well this filter works, we create a test signal composed of two sinusoids: one at 30 Hz (low frequency, should pass) and another at 250 Hz (high frequency, should be blocked). The "Original Composite Signal" figure shows that both frequencies are present before filtering.

After filtering the signal with our designed lowpass filter, we observe the filtered signal. The high-frequency component (250 Hz) is removed, leaving behind only the low-frequency 30 Hz sinusoid, which confirms that our filter is working correctly.

Finally, we check the frequency spectra of the original and filtered signals. The original spectrum shows two distinct peaks at 30 Hz and 250 Hz, corresponding to the two input frequencies. After filtering, the 250 Hz component is significantly reduced, leaving mostly the 30 Hz frequency, proving that our lowpass filter has effectively suppressed high frequencies.

Task 3)

code:

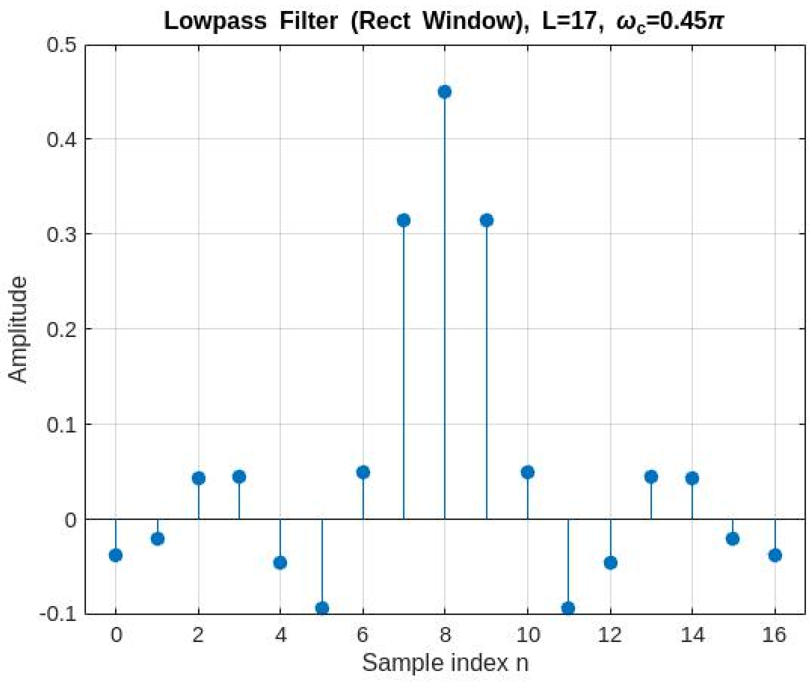
```
%task3
clc; clear; close all;
omega_p = 0.4 * pi;
omega_s = 0.5 * pi;
L = 17;
N_fft = 8192;


omega_c = (omega_p + omega_s)/2;
M = (L - 1)/2;
n = 0 : (L - 1);


h_lowpass = zeros(1, L);
for k = 1:length(n)
if abs(n(k) - M) < eps
h_lowpass(k) = omega_c / pi;
else
h_lowpass(k) = sin(omega_c * (n(k) - M) + eps) / (pi * (n(k) - M) + eps);
end
end


figure('Name','Lowpass Impulse Response');
stem(n, h_lowpass, 'filled');
xlabel('Sample index n'); ylabel('Amplitude');
title(sprintf('Lowpass Filter (Rect Window), L=%d, \\omega_c=%.2f\\pi', L,
omega_c/pi));
grid on;


H_LP = fft(h_lowpass, N_fft);
H_LP_shifted = fftshift(H_LP);


freq_axis_rad = linspace(-pi, pi, N_fft);


figure('Name','Lowpass Frequency Response (Rad/Sample)');
subplot(2,1,1);
plot(freq_axis_rad, abs(H_LP_shifted), 'LineWidth', 1.2);
xlabel('Frequency (rad/sample)');
```

```
ylabel('|H_{LP}(e^{j\\omega})|');
title('Magnitude Response (Lowpass)');
grid on;
subplot(2,1,2);
plot(freq_axis_rad, angle(H_LP_shifted), 'LineWidth', 1.2);
xlabel('Frequency (rad/sample)');
ylabel('Phase (radians)');
title('Phase Response (Lowpass)');
grid on;


fs = 1000;
t = 0 : 1/fs : 1-1/fs;


f_below = 150;
f_above = 350;
x_below = sin(2*pi*f_below * t);
x_above = sin(2*pi*f_above * t);
x_composite = x_below + x_above;


y_lowpass = filter(h_lowpass, 1, x_composite);


figure('Name','Lowpass Time-Domain');
subplot(2,1,1);
plot(t, x_composite, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title(sprintf('Original Composite (%.0f Hz + %.0f Hz)', f_below, f_above));
grid on;
subplot(2,1,2);
plot(t, y_lowpass, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Filtered Signal (Lowpass)');
grid on;


X = fft(x_composite, N_fft);
Y_LP = fft(y_lowpass, N_fft);
X_shifted = fftshift(X);
Y_LP_shifted = fftshift(Y_LP);


freq_axis_Hz = linspace(-fs/2, fs/2, N_fft);


figure('Name','Lowpass Frequency-Domain (Centered)');
```

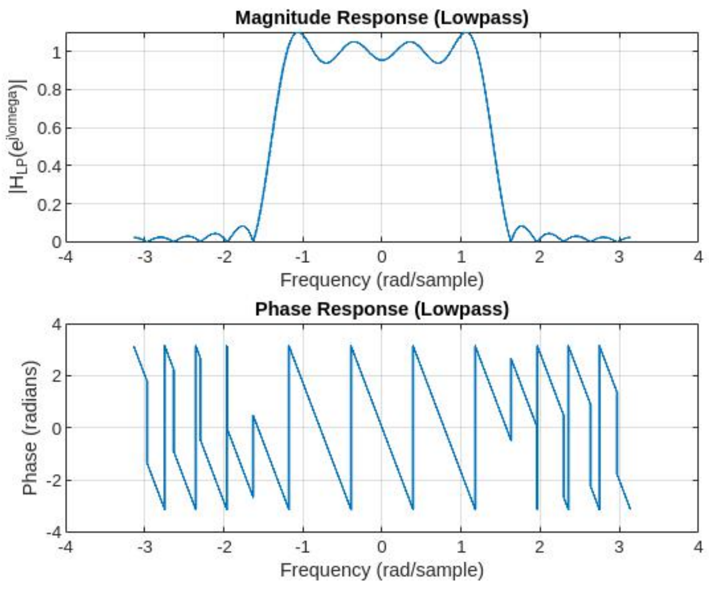
```
subplot(2,1,1);
plot(freq_axis_Hz, abs(X_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Original Composite Spectrum (Centered)');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, abs(Y_LP_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|Y_{LP}(f)|');
title('Lowpass Filtered Spectrum (Centered)');
grid on;


ap = zeros(1, L);
for k = 1:length(n)
if abs(n(k) - M) < eps
ap(k) = 1;
else
ap(k) = sin(pi*(n(k) - M + eps)) / (pi*(n(k) - M + eps));
end
end


h_highpass = ap - h_lowpass;
figure('Name','LP, AP, HP Impulse Responses');
subplot(3,1,1);
stem(n, h_lowpass, 'b', 'filled');
xlabel('n'); ylabel('Amplitude');
title('Lowpass Filter Impulse Response');
grid on;
subplot(3,1,2);
stem(n, ap, 'r', 'filled');
xlabel('n'); ylabel('Amplitude');
title('All-Pass Function');
grid on;
subplot(3,1,3);
stem(n, h_highpass, 'g', 'filled');
xlabel('n'); ylabel('Amplitude');
title('Highpass Filter (ap - Lowpass)');
grid on;


f_lowHP = 150;
f_highHP = 350;
x_low = sin(2*pi*f_lowHP * t);
x_high = sin(2*pi*f_highHP * t);
```

```
x_compHP = x_low + x_high;


y_highpass = filter(h_highpass, 1, x_compHP);


figure('Name','Highpass Time-Domain');
subplot(2,1,1);
plot(t, x_compHP, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title(sprintf('Original Composite (%.0f Hz + %.0f Hz)', f_lowHP, f_highHP));
grid on;
subplot(2,1,2);
plot(t, y_highpass, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Filtered Signal (Highpass)');
grid on;


X_HP = fft(x_compHP, N_fft);
Y_HP = fft(y_highpass, N_fft);
X_HP_shifted = fftshift(X_HP);
Y_HP_shifted = fftshift(Y_HP);


figure('Name','Highpass Frequency-Domain (Centered)');
subplot(2,1,1);
plot(freq_axis_Hz, abs(X_HP_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|X_{HP}(f)|');
title('Original Composite Spectrum (Centered)');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, abs(Y_HP_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|Y_{HP}(f)|');
title('Highpass Filtered Spectrum (Centered)');
grid on;
```

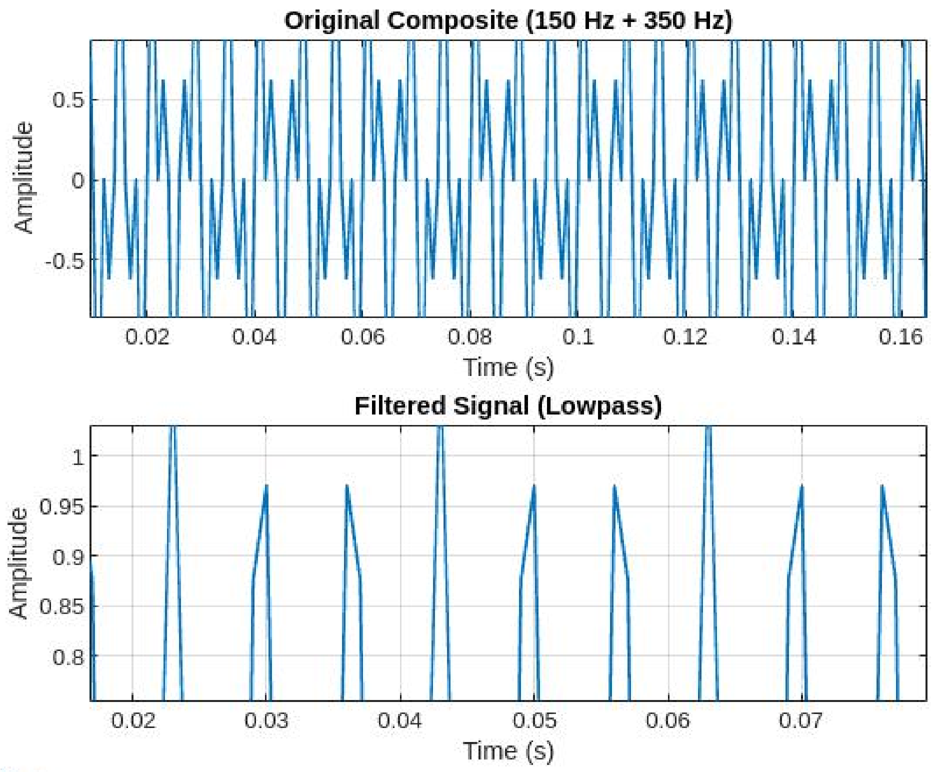
outputs:



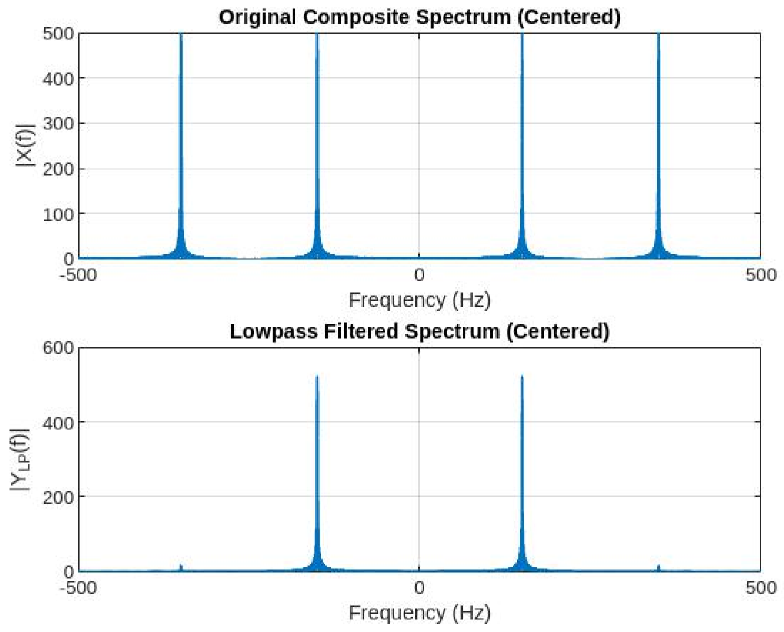
Lowpass Filter (Rect Window), L=17, ω_c=0.45π
Amplitude
Sample index n

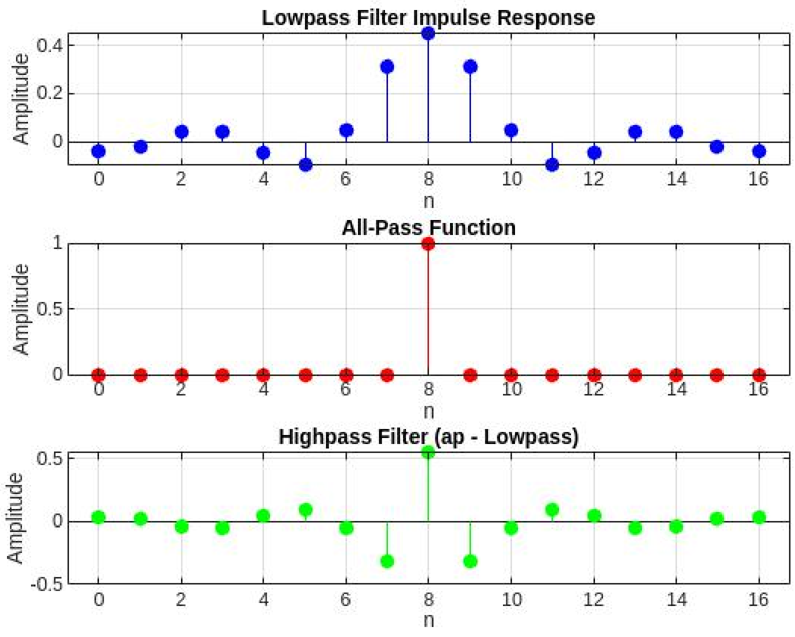


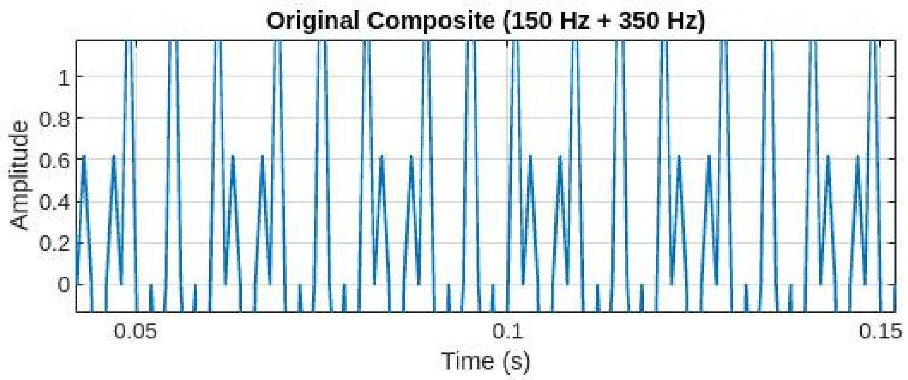
Magnitude Response (Lowpass)
$|H_{LP}(e^{jomega})|$
Frequency (rad/sample)
Phase Response (Lowpass)
Phase (radians)
Frequency (rad/sample)

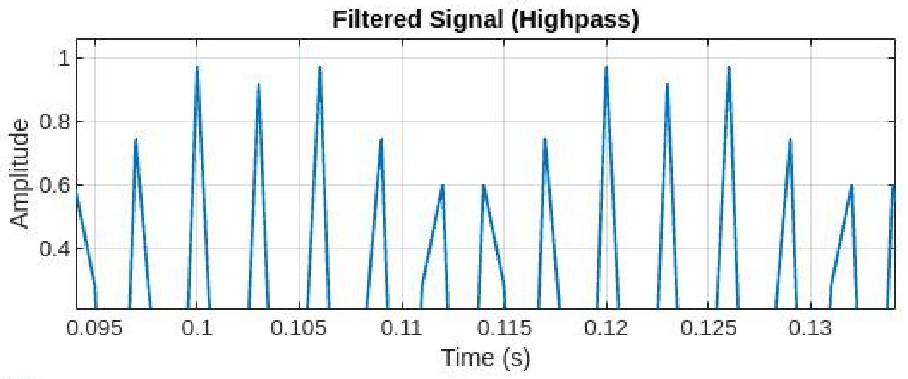

Original Composite (150 Hz + 350 Hz)
Amplitude
Time (s)
Filtered Signal (Lowpass)
Amplitude
Time (s)

Code ^

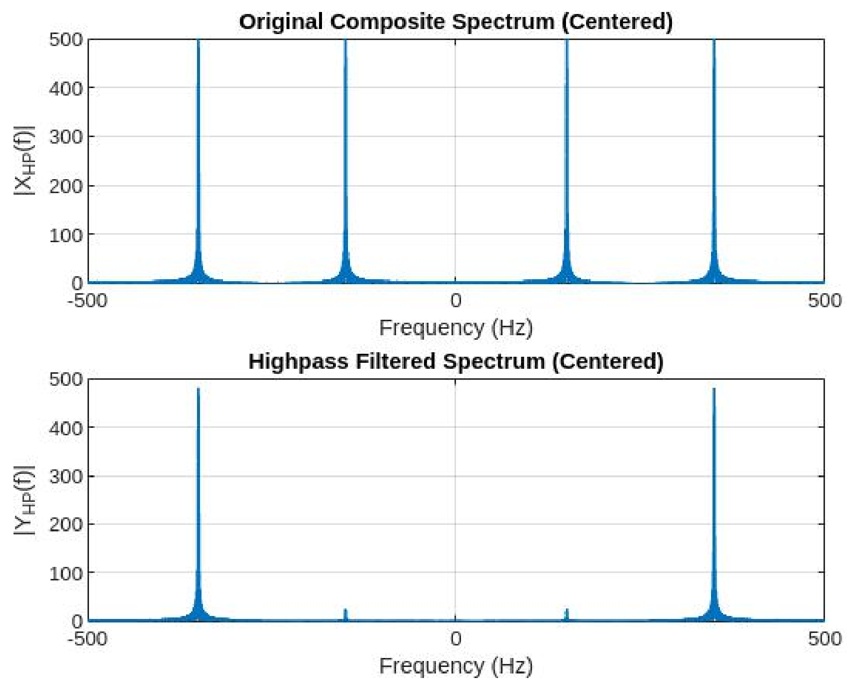

Original Composite Spectrum (Centered)
|X(f)|
Frequency (Hz)
Lowpass Filtered Spectrum (Centered)
$|Y_{LP}(f)|$
Frequency (Hz)


Lowpass Filter Impulse Response
Amplitude
n
All-Pass Function
Amplitude
n
Highpass Filter (ap - Lowpass)
Amplitude
n


Original Composite (150 Hz + 350 Hz)
Amplitude
Time (s)


Filtered Signal (Highpass)
Amplitude
Time (s)

Code ^



Observation:

First, we design a lowpass filter using a rectangular window. The impulse response of this filter is generated using a sinc function, which extends infinitely, but since we cannot implement an infinite-length filter in real life, we truncate it to a finite length. This truncation introduces small oscillations in the response, which is known as the Gibbs effect. This means that instead of a perfect cutoff, there will be small ripples near the transition between passband and stopband. This is important in real applications because it affects how well the filter removes unwanted frequencies.

Then, we plot the impulse response of the filter. The plot shows a symmetric sinc-like function, which is expected for an ideal lowpass filter. The symmetry is important because it ensures that the filter does not introduce any unwanted phase distortions. This means that if we apply this filter to a real signal, it will keep the original waveform shape intact while only removing the higher frequency components.

Next, we check the frequency response of the filter. The magnitude response confirms that the filter allows low frequencies to pass while attenuating the higher frequencies. However, we see small ripples in the stopband, which happen because of the sudden truncation of the sinc function. In real applications, this means that some unwanted high frequencies may not be completely removed, but instead slightly reduced. The phase response is also checked, and it is observed to be linear, meaning that the filter does not distort the timing of different frequency components in the signal. This is important in applications like audio processing, where phase distortions can make sounds unclear.

After this, we test the lowpass filter using a composite signal containing two frequencies: one low and one high. Before filtering, the signal in the time domain appears as a complex waveform with rapid oscillations due to the high-frequency component. This is because when two frequencies are combined, their interference creates a waveform that looks complicated. After filtering, we observe that the high-frequency oscillations are gone, and only the smooth low-frequency signal remains. This shows that the filter is working as expected, removing high-frequency components while keeping the lower ones intact.

To confirm this result, we analyze the frequency spectrum before and after filtering. Initially, the spectrum shows two peaks, each corresponding to one of the frequency components in the composite signal. After filtering, the peak corresponding to the high-frequency component disappears, which means that the filter has successfully removed it. In practical terms, this is useful for applications like noise removal in communication signals, where we want to remove high-frequency noise while keeping the important lower-frequency parts.

Then, we move on to designing a highpass filter. Instead of allowing low frequencies to pass, this filter is designed to remove low frequencies and allow high frequencies to pass. We create it by subtracting the lowpass filter from an all-pass function. This changes the impulse response, making it alternate between positive and negative values, which is expected for a highpass filter.

Next, we test this highpass filter on the same composite signal. Before filtering, the signal contains both slow and fast variations due to the mixture of low and high frequencies. After applying the highpass filter, we see that the slow variations are removed, and only the fast-changing high-frequency components remain. This confirms that the highpass filter is correctly removing low frequencies while preserving high frequencies.

Finally, we check the frequency spectrum before and after applying the highpass filter. Before filtering, both low and high-frequency components are present. After filtering, the peak corresponding to the low-frequency component is removed, and only the high-frequency part remains. This proves that the highpass filter is functioning correctly. In real-life applications, this type of filter is useful in removing low-frequency noise, such as background hum in audio recordings, or isolating high-frequency components in medical signals like ECGs.

Task 4)

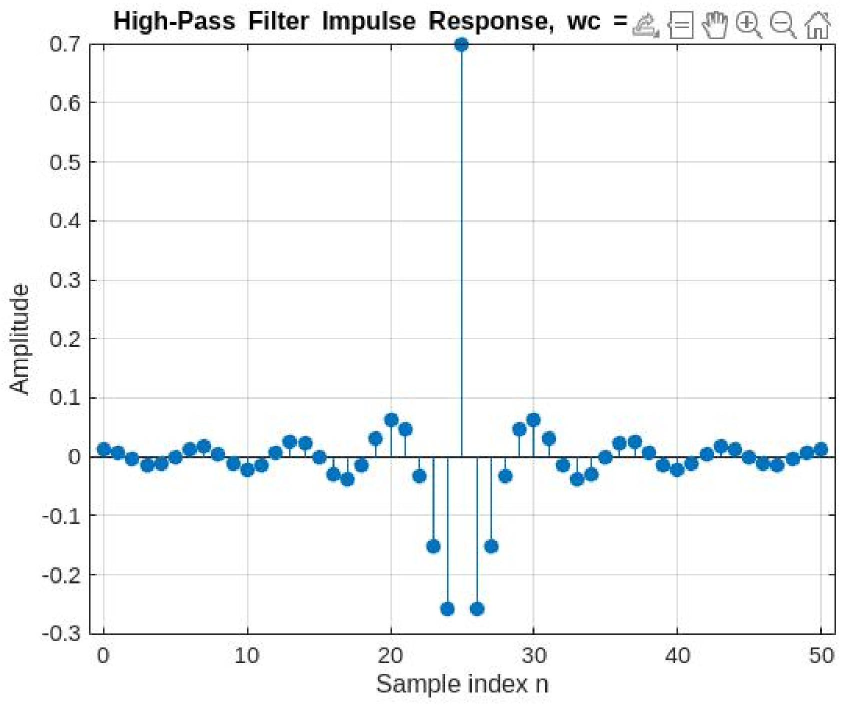
code:

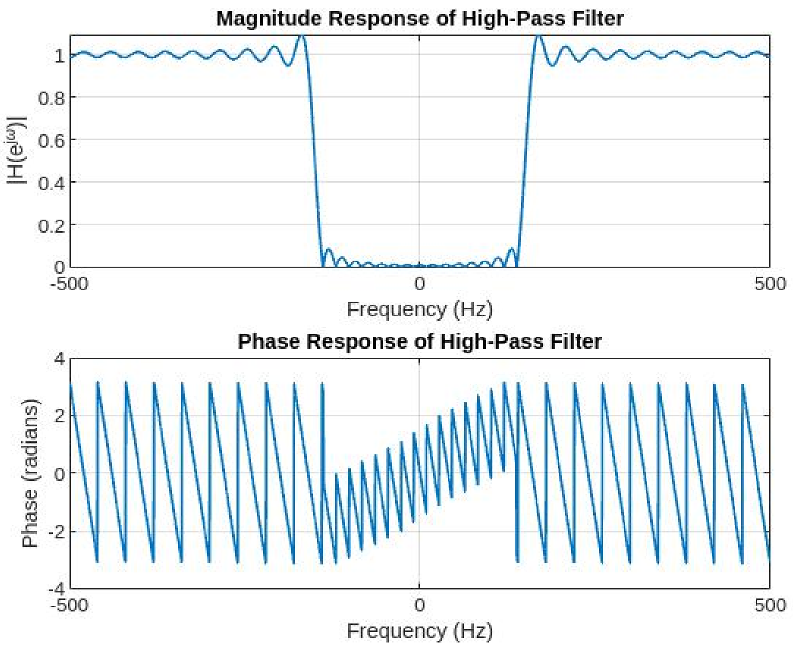
```
%task 4:
function h_hp = LVHPFViaSincLPFRectwin(wc, L)


M = (L - 1) / 2;
n = 0:(L-1);


h_lp = zeros(1, L);
for k = 1:length(n)
if n(k) == M
h_lp(k) = wc / pi;
else
h_lp(k) = sin(wc * (n(k) - M)) / (pi * (n(k) - M));
end
end


ap = zeros(1, L);
for k = 1:length(n)
if n(k) == M
ap(k) = 1;
else
ap(k) = sin(pi * (n(k) - M)) / (pi * (n(k) - M));
end
end


h_hp = ap - h_lp;


end


clc; clear; close all;
%making high pass filter below
wc = 0.3 * pi;
L = 51;
fs = 1000;
```

```
h_hp = LVHPFViaSincLPFRectwin(wc, L);


n = 0:(L-1);
figure('Name','High-Pass Filter Impulse Response');
stem(n, h_hp, 'filled');
xlabel('Sample index n');
ylabel('Amplitude');
title(sprintf('High-Pass Filter Impulse Response, wc = %.2f\\pi, L = %d', wc/pi, L));
grid on;


N_fft = 2048;
H_hp = fft(h_hp, N_fft);
H_hp_shifted = fftshift(H_hp);
freq_axis = linspace(-fs/2, fs/2, N_fft);


figure('Name','High-Pass Frequency Response');
subplot(2,1,1);
plot(freq_axis, abs(H_hp_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|H(e^{j\omega})|');
title('Magnitude Response of High-Pass Filter');
grid on;
subplot(2,1,2);
plot(freq_axis, angle(H_hp_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('Phase Response of High-Pass Filter');
grid on;


f_c = (wc/(2*pi)) * fs;
fprintf('High-pass filter cutoff frequency: %.2f Hz\n', f_c);


t = 0:1/fs:1-1/fs;


x_low = sin(2*pi*100*t);
x_high = sin(2*pi*300*t);
x_composite = x_low + x_high;


y_hp = filter(h_hp, 1, x_composite);
```

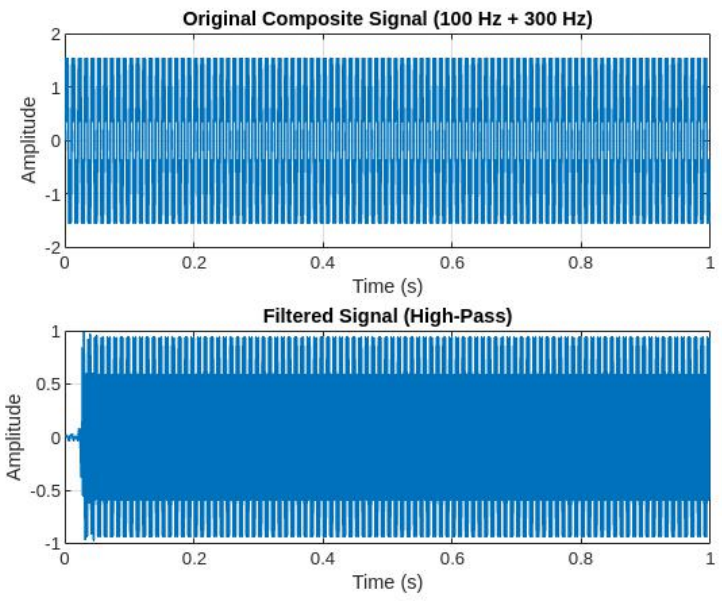
```
figure('Name','High-Pass Time-Domain Comparison');
subplot(2,1,1);
plot(t, x_composite, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
title('Original Composite Signal (100 Hz + 300 Hz)');
grid on;
subplot(2,1,2);
plot(t, y_hp, 'LineWidth', 1.2);
xlabel('Time (s)');
ylabel('Amplitude');
title('Filtered Signal (High-Pass)');
grid on;


X_orig = fft(x_composite, N_fft);
Y_filtered = fft(y_hp, N_fft);
X_orig_shifted = fftshift(X_orig);
Y_filtered_shifted = fftshift(Y_filtered);


figure('Name','High-Pass Frequency-Domain Comparison');
subplot(2,1,1);
plot(freq_axis, abs(X_orig_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|X(f)|');
title('Spectrum of Original Composite Signal');
grid on;
subplot(2,1,2);
plot(freq_axis, abs(Y_filtered_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|Y(f)|');
title('Spectrum of High-Pass Filtered Signal');
grid on;
```

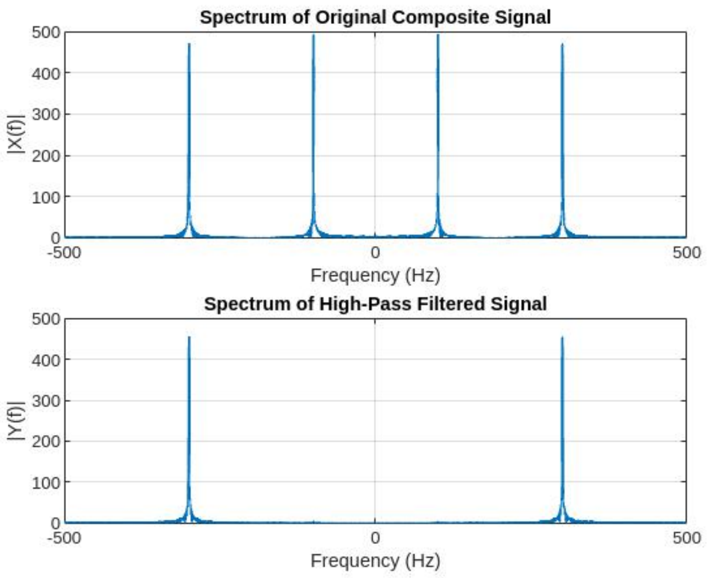
Outputs:


High-Pass Filter Impulse Response, wc =
Amplitude
Sample index n
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
0
10
20
30
40
50


Magnitude Response of High-Pass Filter
$|H(e^{j\omega})|$
Frequency (Hz)
-500
0
500
Phase Response of High-Pass Filter
Phase (radians)
Frequency (Hz)

```
High-pass filter cutoff frequency: 150.00 Hz
```





Observation:

First, we create a high-pass filter using the sinc function method. We first define the required parameters like the cutoff frequency wc, filter length L, and sampling frequency fs. Then, we generate the impulse response of an ideal low-pass filter using the sinc function. Since we need a high-pass filter, we subtract this low-pass response from an all-pass function, which allows all frequencies to pass through. This subtraction gives us the high-pass filter impulse response.

Next, we plot the impulse response of the high-pass filter. The plot shows that the center coefficient has the highest amplitude, and the values on both sides decay symmetrically. This means the filter is properly designed and follows the expected high-pass shape. The presence of oscillations around the center shows the sinc function behavior.

Then, we compute and plot the frequency response of the high-pass filter. The magnitude response shows a significant suppression of low-frequency components, and higher frequencies are allowed to pass. This confirms that the filter is functioning as expected. The phase response is also plotted, showing phase discontinuities, which are expected in high-pass filters.

After verifying the filter, we test it with an input composite signal containing two sinusoids—one at 100 Hz (low frequency) and another at 300 Hz (high frequency). We apply the high-pass filter to this composite signal. The time-domain plot of the original signal shows a mixture of these frequencies, but after filtering, the lower frequency (100 Hz) is mostly removed, and only the higher frequency (300 Hz) remains.

Finally, we analyze the frequency spectrum of the signals. The original spectrum shows peaks at both 100 Hz and 300 Hz, representing both input frequencies. After filtering, the high-pass filtered spectrum shows that the 100 Hz component is heavily reduced, while the 300 Hz component remains strong. This confirms that the filter successfully removed the lower frequency and retained the higher frequency.

Task 5)

code:

```
%task5
function h_bp = LVBPFViaSincLPFRectwin(wc1, wc2, L)


M = (L - 1) / 2;
n = 0:(L - 1);
h_bp = zeros(1, L);


for k = 1:length(n)
if abs(n(k) - M) < eps
h_bp(k) = (wc2 - wc1) / pi;
else
h_bp(k) = ( sin(wc2*(n(k)-M)) - sin(wc1*(n(k)-M)) ) / (pi*(n(k)-M));
end
end


end


clc; clear; close all;


L_bp = 79;
wc1 = 0.25 * pi;
wc2 = 0.55 * pi;
fs = 1000;
N_fft = 8192;
M_bp = (L_bp - 1)/2;
n_bp = 0:(L_bp-1);
h_bp = zeros(1, L_bp);
for k = 1:length(n_bp)
if abs(n_bp(k) - M_bp) < eps
h_bp(k) = (wc2 - wc1)/pi;
else
h_bp(k) = ( sin(wc2*(n_bp(k)-M_bp)) - sin(wc1*(n_bp(k)-M_bp)) ) / (pi*(n_bp(k)-
M_bp));
end
end


figure('Name','Task 5 - Bandpass Filter Impulse Response');
```

```
stem(n_bp, h_bp, 'filled');
xlabel('Sample index n'); ylabel('Amplitude');
title(sprintf('Bandpass Impulse Response, L = %d, wc1 = %.2f\\pi, wc2 = %.2f\\pi',
L_bp, wc1/pi, wc2/pi));
grid on;


H_bp = fft(h_bp, N_fft);
H_bp_shifted = fftshift(H_bp);
freq_axis_Hz = linspace(-fs/2, fs/2, N_fft);


figure('Name','Task 5 - Bandpass Frequency Response');
subplot(2,1,1);
plot(freq_axis_Hz, abs(H_bp_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('|H(e^{j\omega})|');
title('Magnitude Response of Bandpass Filter');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, angle(H_bp_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)');
ylabel('Phase (radians)');
title('Phase Response of Bandpass Filter');
grid on;


t = 0:1/fs:1-1/fs;
x1 = sin(2*pi*50*t);
x2 = sin(2*pi*200*t);
x3 = sin(2*pi*400*t);
x_comp_bp = x1 + x2 + x3;


y_bp = filter(h_bp, 1, x_comp_bp);


figure('Name','Task 5 - Time Domain Comparison (Bandpass)');
subplot(2,1,1);
plot(t, x_comp_bp, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Original Composite Signal (50 Hz + 200 Hz + 400 Hz)');
grid on;
subplot(2,1,2);
plot(t, y_bp, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Bandpass Filtered Signal');
grid on;
```

```
X_bp = fft(x_comp_bp, N_fft);
Y_bp_out = fft(y_bp, N_fft);
f_axis_onesided = linspace(0, fs/2, N_fft/2+1);
X_bp_onesided = abs(X_bp(1:N_fft/2+1));
Y_bp_onesided = abs(Y_bp_out(1:N_fft/2+1));


figure('Name','Task 5 - Frequency Domain (One-Sided, Bandpass)');
subplot(2,1,1);
plot(f_axis_onesided, X_bp_onesided, 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|X(f)|');
title('Original Composite Spectrum');
grid on;
subplot(2,1,2);
plot(f_axis_onesided, Y_bp_onesided, 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|Y(f)|');
title('Bandpass Filtered Spectrum');
grid on;


fprintf('For fs = %d Hz: wc1 = 0.25*pi corresponds to approx. %.1f Hz, wc2 =
0.55*pi corresponds to approx. %.1f Hz.\n', fs, (0.25/2)*fs, (0.55/2)*fs);
%band reject
L_br = 71;
M_br = (L_br - 1)/2;
wc1_br = 0.4 * pi;
wc2_br = 0.5 * pi;
n_br = 0:(L_br-1);
h_bp_br = zeros(1, L_br);
for k = 1:length(n_br)
if abs(n_br(k) - M_br) < eps
h_bp_br(k) = (wc2_br - wc1_br)/pi;
else
h_bp_br(k) = ( sin(wc2_br*(n_br(k)-M_br)) - sin(wc1_br*(n_br(k)-M_br)) ) /
(pi*(n_br(k)-M_br));
end
end


h_allpass = zeros(1, L_br);
for k = 1:length(n_br)
if abs(n_br(k) - M_br) < eps
h_allpass(k) = 1;
else
h_allpass(k) = sin(pi*(n_br(k)-M_br)) / (pi*(n_br(k)-M_br));
end
```

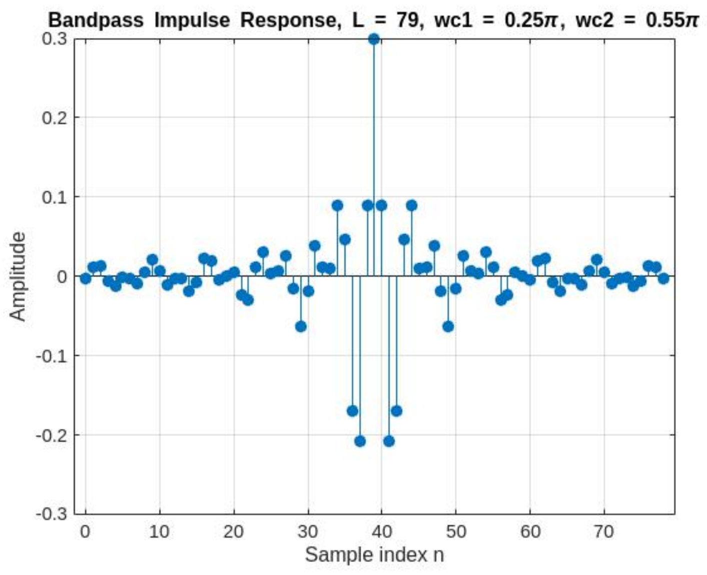
```
end

h_br = h_allpass - h_bp_br;

figure('Name','Task 12 - Band-Reject Filter Impulse Responses');
subplot(3,1,1);
stem(n_br, h_bp_br, 'b','filled');
xlabel('n'); ylabel('Amplitude');
title('Bandpass Prototype Impulse Response (to be rejected)');
grid on;
subplot(3,1,2);
stem(n_br, h_allpass, 'r','filled');
xlabel('n'); ylabel('Amplitude');
title('All-Pass Function');
grid on;
subplot(3,1,3);
stem(n_br, h_br, 'g','filled');
xlabel('n'); ylabel('Amplitude');
title('Band-Reject Filter Impulse Response (All-Pass - Bandpass)');
grid on;

H_br = fft(h_br, N_fft);
H_br_shifted = fftshift(H_br);
figure('Name','Task 12 - Band-Reject Frequency Response');
subplot(2,1,1);
plot(freq_axis_Hz, abs(H_br_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|H(e^{j\omega})|');
title('Magnitude Response (Band-Reject Filter)');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, angle(H_br_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('Phase (radians)');
title('Phase Response (Band-Reject Filter)');
grid on;

x1_br = sin(2*pi*100*t);
x2_br = sin(2*pi*250*t);
x3_br = sin(2*pi*400*t);
x_comp_br = x1_br + x2_br + x3_br;
y_br = filter(h_br, 1, x_comp_br);

figure('Name','Task 12 - Band-Reject Time Domain');
subplot(2,1,1);
```

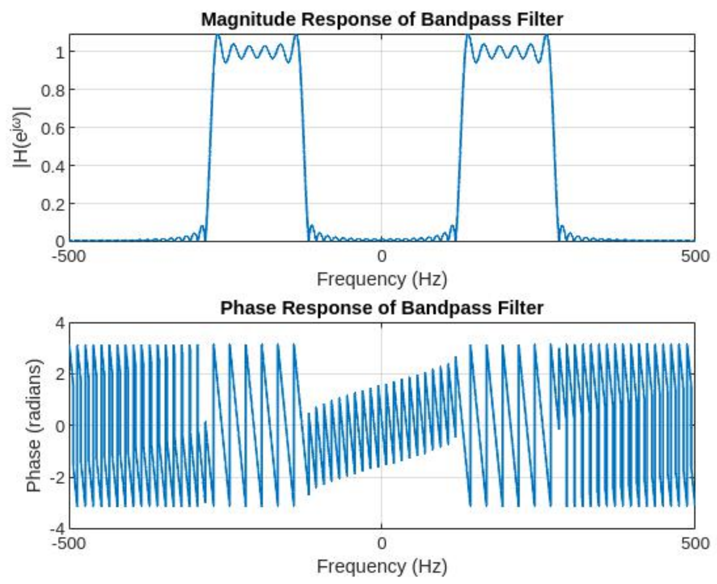
```
plot(t, x_comp_br, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Original Composite Signal (100 Hz, 250 Hz, 400 Hz)');
grid on;
subplot(2,1,2);
plot(t, y_br, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Band-Reject Filtered Signal');
grid on;


X_br = fft(x_comp_br, N_fft);
Y_br_out = fft(y_br, N_fft);
X_br_shifted = fftshift(X_br);
Y_br_shifted = fftshift(Y_br_out);
figure('Name','Task 12 - Band-Reject Frequency Domain');
subplot(2,1,1);
plot(freq_axis_Hz, abs(X_br_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|X(f)|');
title('Original Composite Spectrum');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, abs(Y_br_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|Y(f)|');
title('Band-Reject Filtered Spectrum');
grid on;


%bandstop:
L_bs = 71;
M_bs = (L_bs - 1)/2;
n_bs = 0:(L_bs-1);


wc1_bs = 0.45 * pi;
wc2_bs = 0.75 * pi;


h_bp_bs = zeros(1, L_bs);
for k = 1:length(n_bs)
if abs(n_bs(k) - M_bs) < eps
h_bp_bs(k) = (wc2_bs - wc1_bs)/pi;
else
h_bp_bs(k) = ( sin(wc2_bs*(n_bs(k)-M_bs)) - sin(wc1_bs*(n_bs(k)-M_bs)) ) /
(pi*(n_bs(k)-M_bs));
end
end
```

```
h_allpass_bs = zeros(1, L_bs);
for k = 1:length(n_bs)
if abs(n_bs(k) - M_bs) < eps
h_allpass_bs(k) = 1;
else
h_allpass_bs(k) = sin(pi*(n_bs(k)-M_bs)) / (pi*(n_bs(k)-M_bs));
end
end


h_bs = h_allpass_bs - h_bp_bs;


figure('Name','Task 13 - Bandstop Filter Impulse Response');
stem(n_bs, h_bs, 'filled');
xlabel('n'); ylabel('Amplitude');
title(sprintf('Bandstop Filter Impulse Response, L=%d, band edges =
[0.4,0.5,0.7,0.8]π', L_bs));
grid on;


H_bs = fft(h_bs, N_fft);
H_bs_shifted = fftshift(H_bs);
figure('Name','Task 13 - Bandstop Frequency Response');
subplot(2,1,1);
plot(freq_axis_Hz, abs(H_bs_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|H(e^{j\omega})|');
title('Magnitude Response of Bandstop Filter');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, angle(H_bs_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('Phase (radians)');
title('Phase Response of Bandstop Filter');
grid on;


x1_bs = sin(2*pi*100*t);
x2_bs = sin(2*pi*200*t);
x3_bs = sin(2*pi*350*t);
x_comp_bs = x1_bs + x2_bs + x3_bs;
y_bs = filter(h_bs, 1, x_comp_bs);


figure('Name','Task 13 - Bandstop Time Domain Comparison');
subplot(2,1,1);
plot(t, x_comp_bs, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Original Composite Signal (100 Hz + 200 Hz + 350 Hz)');
```

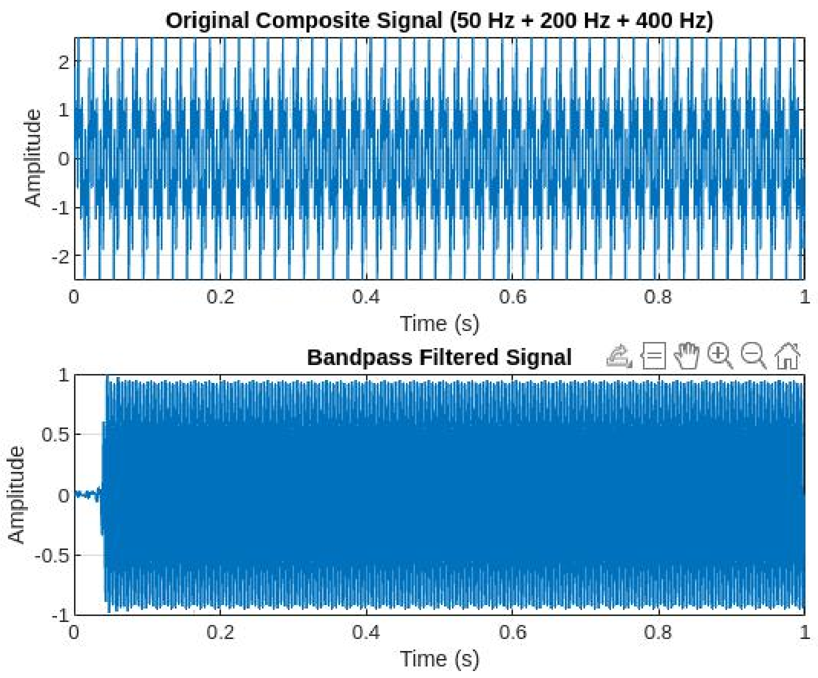
```
grid on;
subplot(2,1,2);
plot(t, y_bs, 'LineWidth', 1.2);
xlabel('Time (s)'); ylabel('Amplitude');
title('Bandstop Filtered Signal');
grid on;


X_bs = fft(x_comp_bs, N_fft);
Y_bs_out = fft(y_bs, N_fft);
X_bs_shifted = fftshift(X_bs);
Y_bs_shifted = fftshift(Y_bs_out);
figure('Name','Task 13 - Bandstop Frequency Domain Comparison');
subplot(2,1,1);
plot(freq_axis_Hz, abs(X_bs_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|X(f)|');
title('Spectrum of Original Composite Signal');
grid on;
subplot(2,1,2);
plot(freq_axis_Hz, abs(Y_bs_shifted), 'LineWidth', 1.2);
xlabel('Frequency (Hz)'); ylabel('|Y(f)|');
title('Spectrum of Bandstop Filtered Signal');
grid on;
```

outputs:

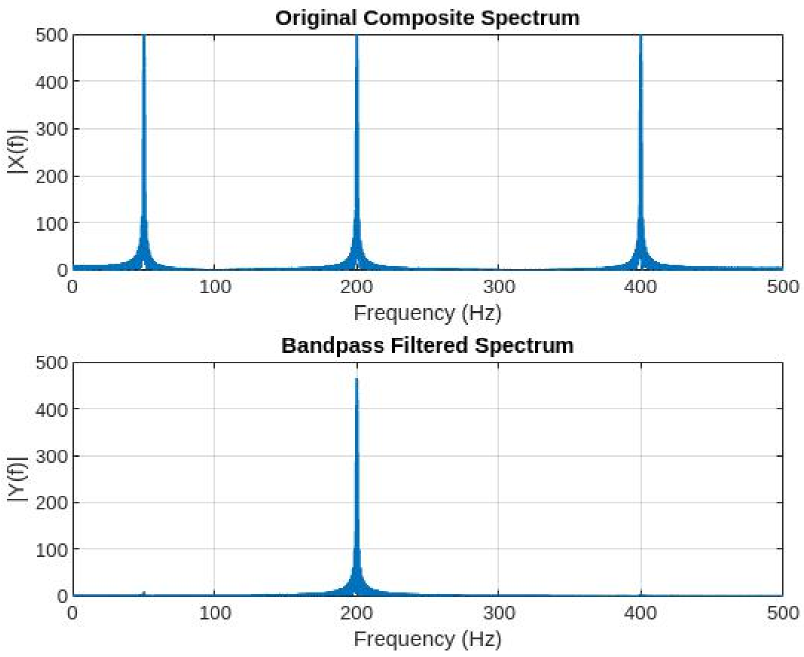




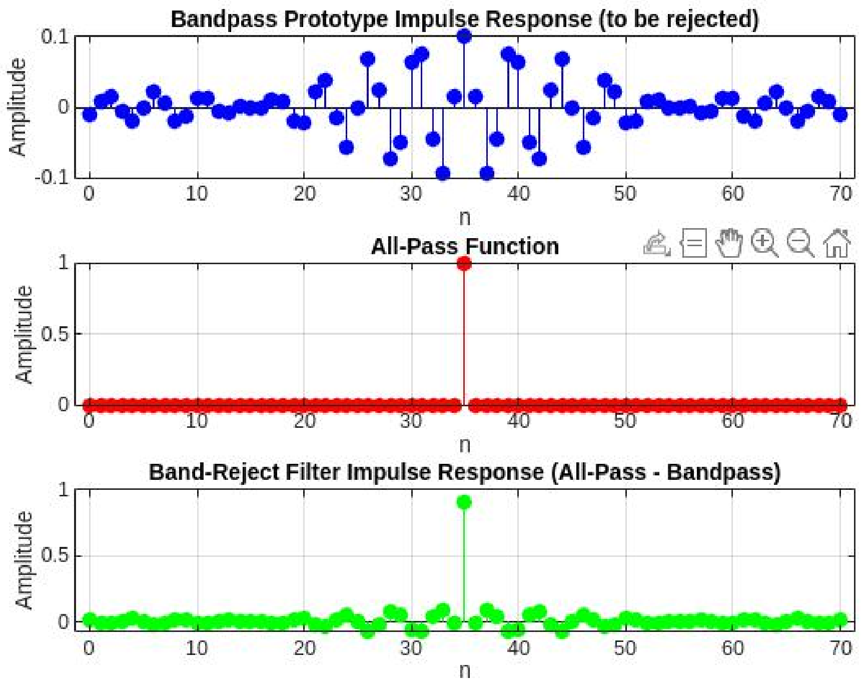
Magnitude Response of Bandpass Filter
$|H(e^{j\omega})|$
0
0.2
0.4
0.6
0.8
1
-500
0
500
Frequency (Hz)
Phase Response of Bandpass Filter
Phase (radians)
-4
-2
0
2
4
-500
0
500
Frequency (Hz)

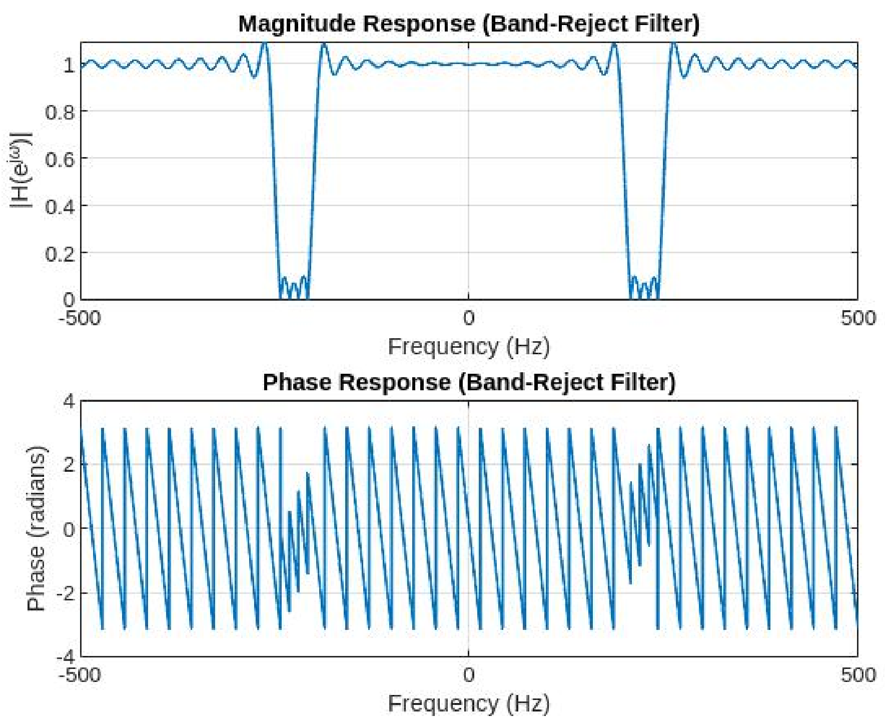


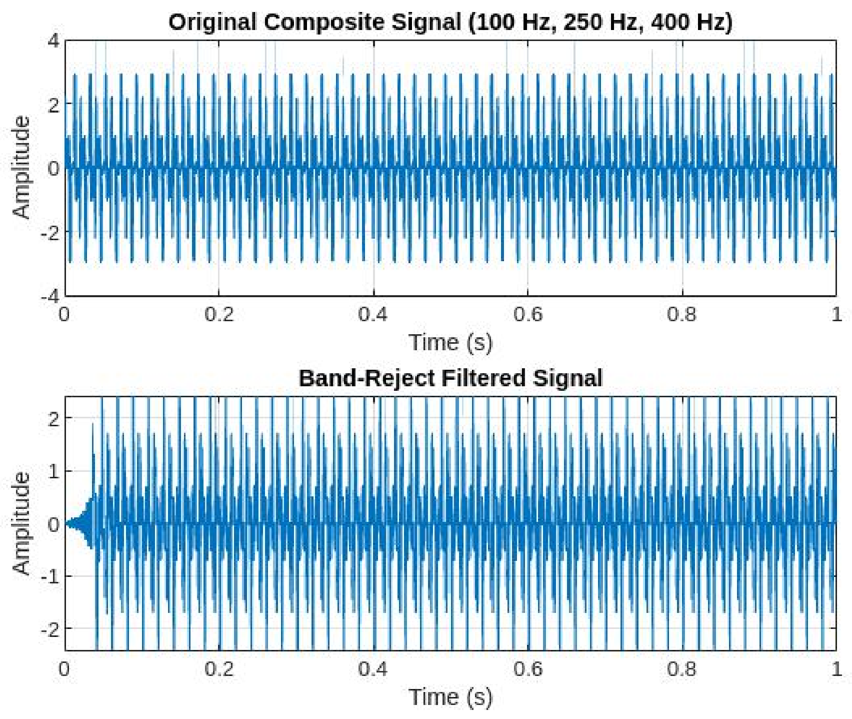
Original Composite Signal (50 Hz + 200 Hz + 400 Hz)
Amplitude
-2
-1
0
1
2
0
0.2
0.4
0.6
0.8
1
Time (s)
Bandpass Filtered Signal
Amplitude
-1
-0.5
0
0.5
1
0
0.2
0.4
0.6
0.8
1
Time (s)

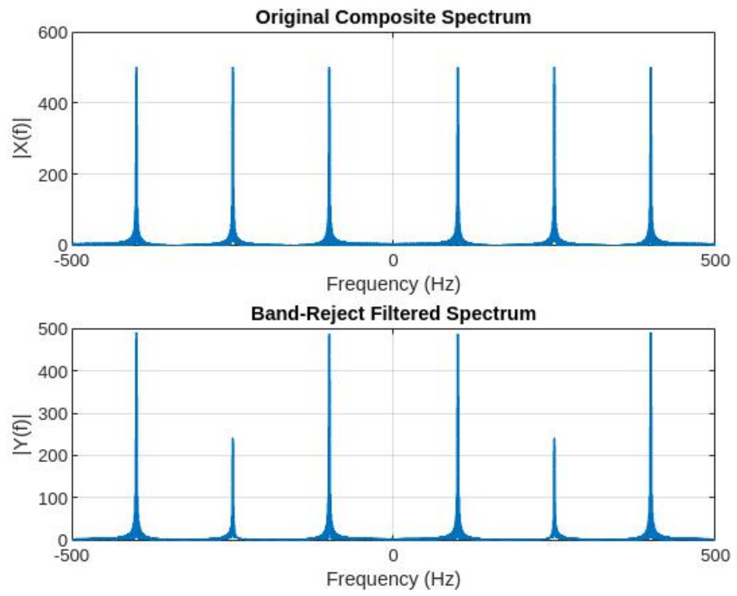


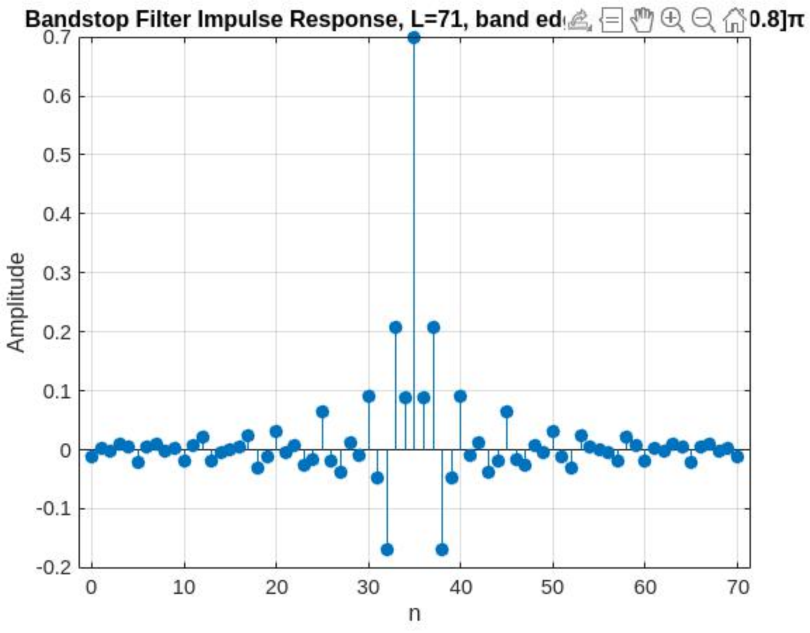
For fs = 1000 Hz: wc1 = 0.25*pi corresponds to approx. 125.0 Hz, wc2 = 0.55*pi corresponds to approx. 275.0 Hz.

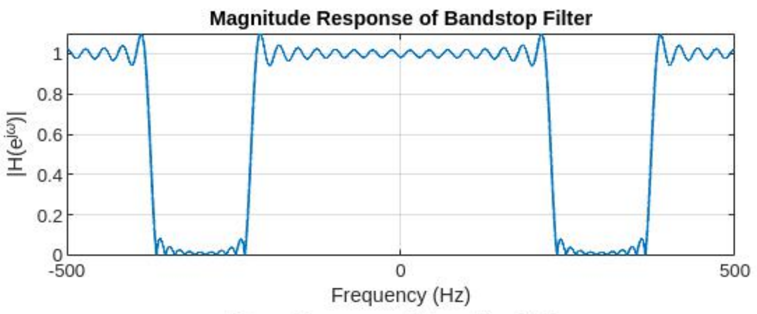


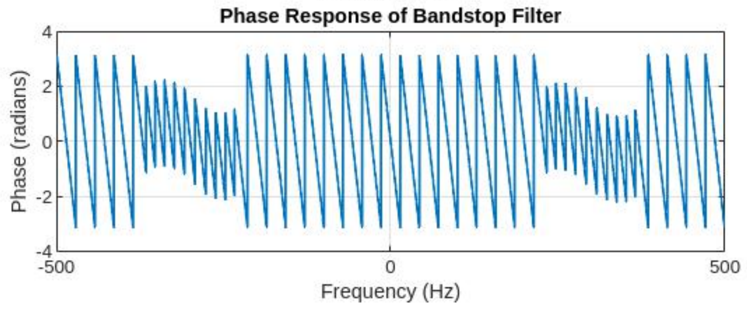

Magnitude Response (Band-Reject Filter)
$|H(e^{j\omega})|$
Frequency (Hz)
Phase Response (Band-Reject Filter)
Phase (radians)
Frequency (Hz)

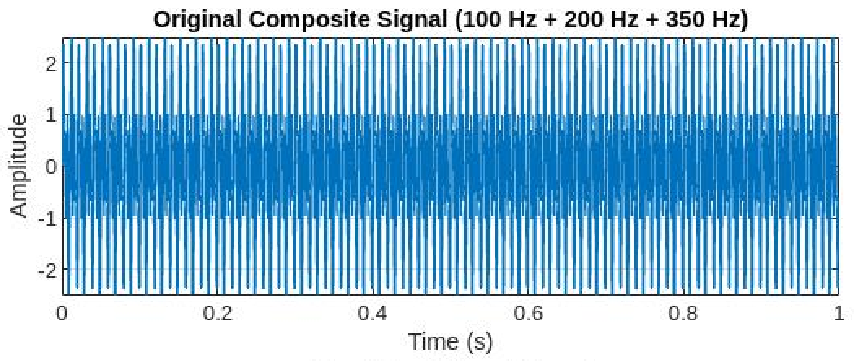

Original Composite Signal (100 Hz, 250 Hz, 400 Hz)
Amplitude
Time (s)
Band-Reject Filtered Signal
Amplitude
Time (s)

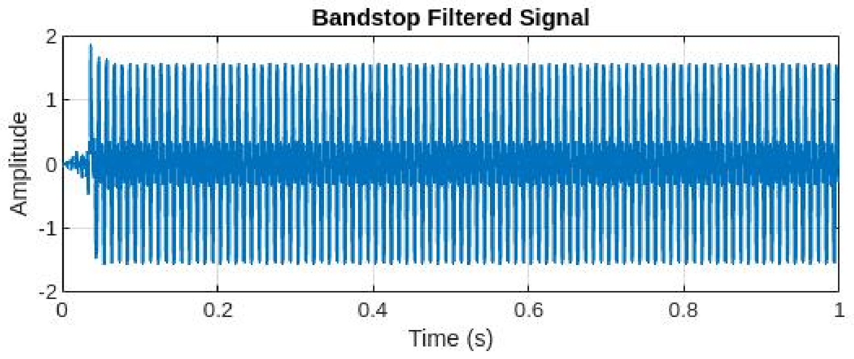

Original Composite Spectrum
|X(f)|
Frequency (Hz)
Band-Reject Filtered Spectrum
|Y(f)|
Frequency (Hz)

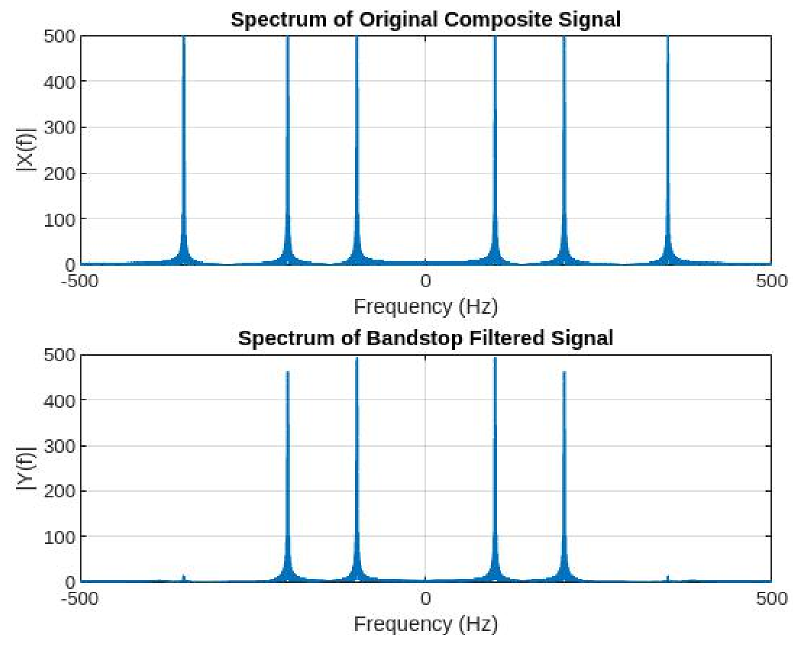

Bandstop Filter Impulse Response, L=71, band ed
0.8]π
Amplitude
n


Magnitude Response of Bandstop Filter
|H(e^{jω})|
Frequency (Hz)
0
0.2
0.4
0.6
0.8
1
-500
0
500


Phase Response of Bandstop Filter
Phase (radians)
Frequency (Hz)
4
2
0
-2
-4
-500
0
500


Original Composite Signal (100 Hz + 200 Hz + 350 Hz)
Amplitude
Time (s)
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1


Bandstop Filtered Signal
Amplitude
Time (s)
2
1
0
-1
-2
0
0.2
0.4
0.6
0.8
1



Observation:

First, we designed a bandpass filter using the sinc function with a rectangular window. This filter allows a specific range of frequencies to pass while blocking all others. The impulse response plot shows a symmetrical pattern centered at the middle sample index, indicating a linear-phase filter. The peaks around the center suggest that most of the energy is concentrated at the filter's cutoff frequencies.

Next, we computed and plotted the magnitude and phase response of the bandpass filter. The magnitude response shows two clear passbands, meaning the filter allows signals within those frequency ranges to pass while blocking frequencies outside this range. The phase response exhibits a typical linear phase characteristic with slight variations.

After designing the bandpass filter, we applied it to a composite signal consisting of three frequencies: 50 Hz, 200 Hz, and 400 Hz. The time-domain plot of the original signal shows a combination of these three frequencies, appearing as a high-frequency oscillating waveform. After filtering, the time-domain plot of the bandpass-filtered signal shows that only the 200 Hz component is preserved, meaning that the filter successfully removed the 50 Hz and 400 Hz components.

Then, we examined the frequency domain representation. The original signal spectrum shows three distinct peaks at 50 Hz, 200 Hz, and 400 Hz, corresponding to the frequencies present in the signal. After applying the bandpass filter, the filtered signal spectrum confirms that only the 200 Hz component remains, as the filter successfully removed the unwanted low and high-frequency components.

Next, we moved on to the band-reject filter, which is created by subtracting the bandpass filter from an all-pass function. The impulse response of this filter looks similar to the bandpass response but with an inverted effect, showing that it is designed to block a specific range of frequencies instead of passing them.

The magnitude response of the band-reject filter clearly shows two notches where frequencies are being attenuated, meaning that this filter removes signals within that range while allowing others to pass. The phase response follows the expected behavior of a linear-phase filter but with discontinuities around the stopband edges.

We then applied the band-reject filter to a composite signal containing 100 Hz, 250 Hz, and 400 Hz. The time-domain plot of the original signal shows a mixture of these frequencies, while the filtered signal shows that the 250 Hz component is removed, leaving behind only the 100 Hz and 400 Hz frequencies.

The frequency spectrum of the original signal shows three peaks, corresponding to the three frequencies. After filtering, the spectrum of the band-reject filtered signal confirms that the 250 Hz component has been removed while the other two remain unaffected.

Finally, we designed a bandstop filter, which is another form of a band-reject filter but with different cutoff frequencies. The impulse response follows a similar pattern to the band-reject filter but with a slightly wider stopband. The magnitude response confirms this, showing that a wider range of frequencies is attenuated compared to the band-reject filter.

After applying the bandstop filter to a composite signal of 100 Hz, 200 Hz, and 350 Hz, the time-domain plot of the filtered signal shows that the 200 Hz component has been removed, leaving behind the other two frequencies.

In the frequency domain, the spectrum of the original signal shows three distinct peaks at 100 Hz, 200 Hz, and 350 Hz. After filtering, the bandstop filtered spectrum confirms that the 200 Hz component has been successfully removed while the 100 Hz and 350 Hz components remain.