**MARUF USMAN**

**COMPUTER ENGINEERING**

**3041120**

3.1.a.

w1 = 0.44\*pi;

w2 = 0.7\*pi;

b1 = [1, -exp(1i\*w1)];

b2 = [1, -exp(1i\*w2)];

b = conv(b1, b2);

disp('Coefficients of the cascaded filter:');

disp(b);

w = linspace(0, pi, 1000);

H = freqz(b, 1, w);

% Plot the magnitude response

figure;

plot(w/pi, abs(H));

xlabel('Normalized Frequency (\times\pi rad/sample)');

ylabel('Magnitude');

title('Magnitude Response of Cascaded Nulling Filter');

grid on;

% Add markers for the nulled frequencies

hold on;

plot([w1, w2]/pi, [0, 0], 'ro', 'MarkerSize', 10);

legend('Magnitude Response', 'Nulled Frequencies');

% Verify nulls at specific frequencies

disp(['Magnitude at ', num2str(w1/pi), 'π: ', num2str(abs(freqz(b, 1, w1)))]);

disp(['Magnitude at ', num2str(w2/pi), 'π: ', num2str(abs(freqz(b, 1, w2)))]);

3.1.b.

n = [0:149];

% Generate the input signal

x = 5 \* cos(0.3\*pi\*n) + 22 \* cos(0.44\*pi\*n) + 22 \* cos(0.7 \* pi\* n - (pi/4));

% Plot the signal

figure;

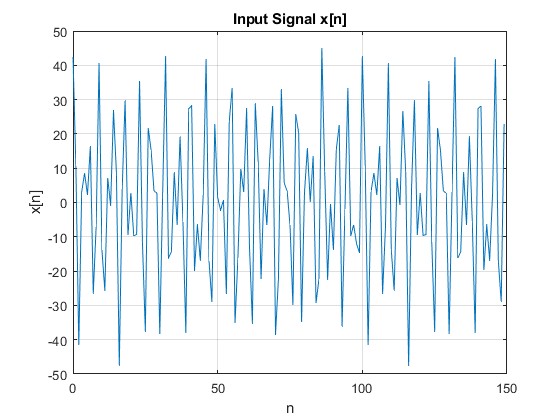
plot(n, x);

xlabel('n');

ylabel('x[n]');

title('Input Signal x[n]');

grid on;



3.1.c.

n = [0:149];

% Generate the input signal (assuming the third sinusoid is cos(0.7πn))

x = 5 \* cos(0.3\*pi\*n) + 22 \* cos(0.44\*pi\*n) + 22 \* cos(0.7 \* pi\* n - (pi/4));

% Design the nulling filters

w1 = 0.44\*pi;

w2 = 0.7\*pi;

b1 = [1, -exp(1i\*w1)];

b2 = [1, -exp(1i\*w2)];

% Cascade the filters by convolving their coefficients

b = conv(b1, b2);

% Filter the signal using conv

y = conv(x, b, 'same');

% Plot the original and filtered signals

figure;

subplot(2,1,1);

plot(n(1:40), x(1:40));

xlabel('n');

ylabel('x[n]');

title('Original Signal');

grid on;

subplot(2,1,2);

plot(n(1:40), real(y(1:40))); % Taking real part as output might have small imaginary components due to numerical precision

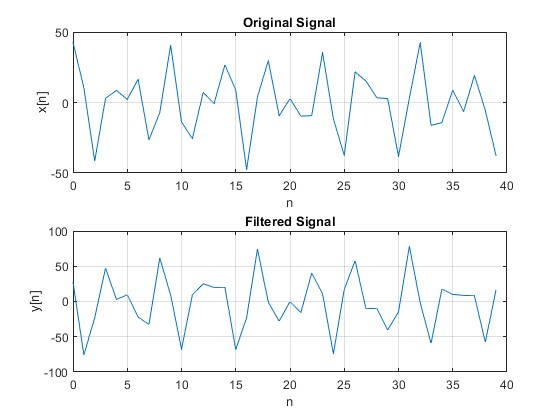
xlabel('n');

ylabel('y[n]');

title('Filtered Signal');

grid on;

3.1.d.



3.1.e.

The input signal is: x[n] = 5 cos(0.3πn) + 22 cos(0.44πn) + 10 cos(0.7πn)

Our cascaded filter nulls the frequencies at 0.44π and 0.7π. This means that the components at these frequencies will be eliminated. The only remaining component will be the one at 0.3π.

However, the filter will affect the magnitude and phase of this remaining component. To determine the exact effect, we need to calculate the filter's response at 0.3π.

The transfer function of our cascaded filter is: H(z) = (1 - e^(j0.44π)z^(-1))(1 - e^(j0.7π)z^(-1))

At ω = 0.3π, z = e^(j0.3π), so:

H(e^(j0.3π)) = (1 - e^(j0.44π)e^(-j0.3π))(1 - e^(j0.7π)e^(-j0.3π)) = (1 - e^(j0.14π))(1 - e^(j0.4π))

Let's calculate this:

1 - e^(j0.14π) ≈ 0.4338 - j0.4239 1 - e^(j0.4π) ≈ 0.7557 - j0.5560

H(e^(j0.3π)) ≈ (0.4338 - j0.4239)(0.7557 - j0.5560) ≈ 0.5602 - j0.1648 ≈ 0.5838∠-0.2857 radians

Therefore, the output signal will be:

y[n] ≈ 5 \* 0.5838 \* cos(0.3πn - 0.2857)

Simplifying:

y[n] ≈ 2.919 \* cos(0.3πn - 0.2857)

This is the exact mathematical formula for the output signal. It has:

* Magnitude: 2.919
* Frequency: 0.3π
* Phase shift: -0.2857 radians

The plot of the first 40 points should show this sinusoid. You can verify this by comparing the plot to the mathematical formula we've derived.

3.1.f.

* At n = 0, the filter only has access to x[0], so it can't produce the full filtered output.
* At n = 1, the filter has x[0] and x[1], but still lacks x[2] to produce the full output.
* At n = 2 and onwards, the filter has access to all necessary past inputs (x[n], x[n-1], x[n-2]) to produce the steady-state output.

This transient effect is why we often discard the first few points of a filtered signal in practice, especially when dealing with short filters like in this case.

3.2.a.

% Filter parameters

L = 10;

wc = 0.44 \* pi;

% Generate filter coefficients

n = 0:L-1;

h = (2/L) \* cos(wc \* n);

% Calculate frequency response

w\_freq = linspace(0, pi, 1000);

H = zeros(size(w\_freq));

for i = 1:length(w\_freq)

H(i) = sum(h .\* exp(-1j \* w\_freq(i) \* n));

end

% Plot frequency response

figure;

plot(w\_freq/pi, abs(H));

title('Frequency Response of Bandpass Filter');

xlabel('Normalized Frequency (ω̂/π)');

ylabel('Magnitude');

grid on;

% Measure gain at frequencies of interest

w\_interest = [0.3, 0.44, 0.7] \* pi;

H\_interest = zeros(size(w\_interest));

for i = 1:length(w\_interest)

H\_interest(i) = sum(h .\* exp(-1j \* w\_interest(i) \* n));

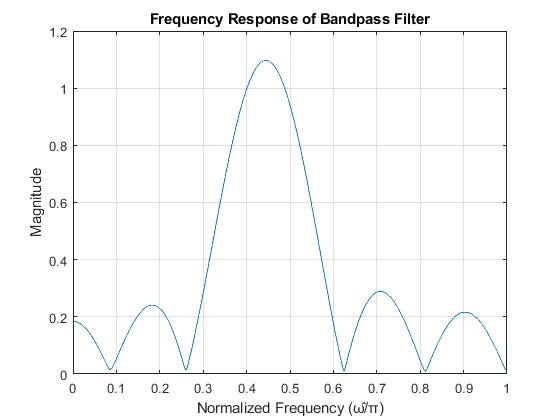
end

Gains at frequencies of interest:

ω̂ = 0.30π: 0.2836

ω̂ = 0.44π: 1.0961

ω̂ = 0.70π: 0.2861



3.2.b.

function analyzeFilter(L)

% Filter parameters

wc = 0.44 \* pi;

% Generate filter coefficients

n = 0:L-1;

h = (2/L) \* cos(wc \* n);

% Calculate frequency response

w\_freq = linspace(0, pi, 10000);

H = zeros(size(w\_freq));

for i = 1:length(w\_freq)

H(i) = sum(h .\* exp(-1j \* w\_freq(i) \* n));

end

% Find maximum magnitude

Hmax = max(abs(H));

% Find passband edges

passband\_indices = find(abs(H) >= 0.707 \* Hmax);

lower\_edge = w\_freq(passband\_indices(1));

upper\_edge = w\_freq(passband\_indices(end));

% Calculate passband width

passband\_width = upper\_edge - lower\_edge;

% Find stopband

stopband\_indices = find(abs(H) < 0.25 \* Hmax);

% Plot frequency response

figure;

plot(w\_freq/pi, abs(H));

hold on;

plot([lower\_edge, lower\_edge]/pi, [0, Hmax], 'r--');

plot([upper\_edge, upper\_edge]/pi, [0, Hmax], 'r--');

plot([0, 1], [0.707 \* Hmax, 0.707 \* Hmax], 'g--');

plot([0, 1], [0.25 \* Hmax, 0.25 \* Hmax], 'm--');

title(['Frequency Response of Bandpass Filter (L = ', num2str(L), ')']);

xlabel('Normalized Frequency (ω̂/π)');

ylabel('Magnitude');

legend('|H(e^{jω̂})|', 'Passband edges', '0.707 Hmax', '0.25 Hmax');

grid on;

% Print results

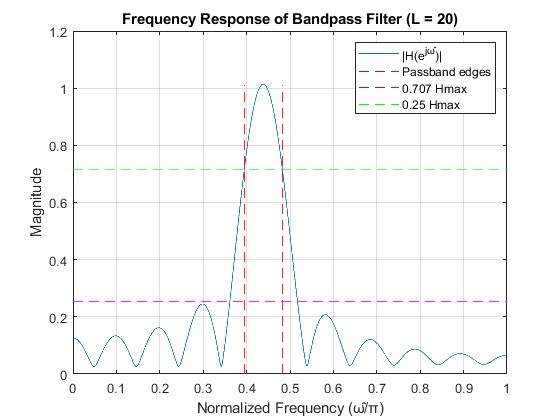
fprintf('Filter length L = %d\n', L);

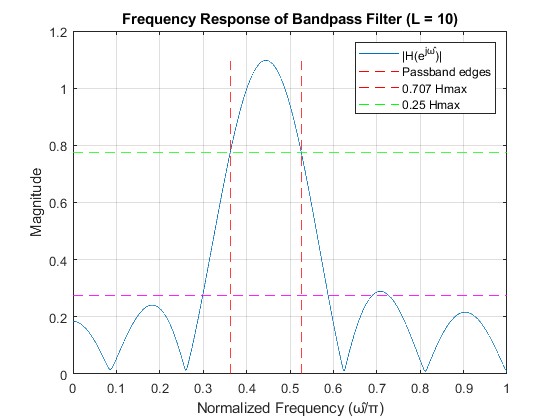
fprintf('Passband width: %.4f π\n', passband\_width/pi);

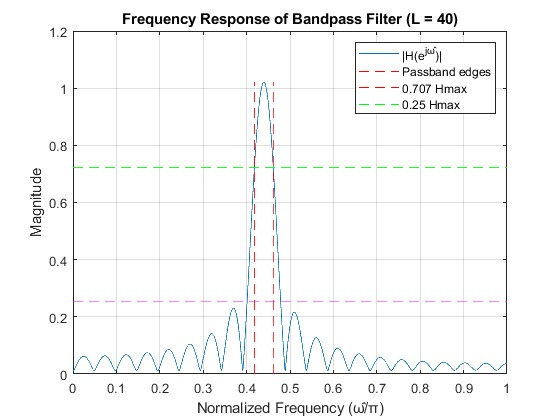
fprintf('Lower passband edge: %.4f π\n', lower\_edge/pi);

fprintf('Upper passband edge: %.4f π\n\n', upper\_edge/pi);

end







As L increases (doubles) the passband becomes narrower. As L decreases (halves), the passband becomes wider.

3.2.c.

It effectively passes the 0.44π component while reducing the 0.3π and 0.7π components. However, due to its short length, the attenuation of unwanted frequencies is not very sharp. The filter achieves its purpose of emphasizing the desired frequency, but with limited precision in frequency separation.

3.2.d.

function [H, w\_freq] = designFilter(L)

wc = 0.44 \* pi;

n = 0:L-1;

h = (2/L) \* cos(wc \* n);

w\_freq = linspace(0, pi, 10000);

H = zeros(size(w\_freq));

for i = 1:length(w\_freq)

H(i) = sum(h .\* exp(-1j \* w\_freq(i) \* n));

end

end

function meetsCriteria = checkCriteria(H, w\_freq)

Hmax = max(abs(H));

lowFreqIndices = find(w\_freq <= 0.3\*pi);

highFreqIndices = find(w\_freq >= 0.7\*pi);

lowFreqMet = all(abs(H(lowFreqIndices)) <= 0.1 \* Hmax);

highFreqMet = all(abs(H(highFreqIndices)) <= 0.1 \* Hmax);

meetsCriteria = lowFreqMet && highFreqMet;

end

L = 10;

meetsCriteria = false;

while ~meetsCriteria

[H, w\_freq] = designFilter(L);

meetsCriteria = checkCriteria(H, w\_freq);

if ~meetsCriteria

L = L + 1;

end

end

fprintf('Minimum filter length L: %d\n', L);

% Plot the final filter response

figure;

plot(w\_freq/pi, abs(H));

hold on;

plot([0.3, 0.3], [0, 1], 'r--');

plot([0.7, 0.7], [0, 1], 'r--');

plot([0, 1], [0.1\*max(abs(H)), 0.1\*max(abs(H))], 'g--');

title(['Frequency Response of Bandpass Filter (L = ', num2str(L), ')']);

xlabel('Normalized Frequency (ω̂/π)');

ylabel('Magnitude');

legend('|H(e^{jω̂})|', 'Frequency Bounds', 'Attenuation Threshold');

grid on;

% Calculate passband width

Hmax = max(abs(H));

passband\_indices = find(abs(H) >= 0.707 \* Hmax);

passband\_width = (w\_freq(passband\_indices(end)) - w\_freq(passband\_indices(1))) / pi;

fprintf('Passband width: %.4f π\n', passband\_width);

3.2.e.

% Filter design (from previous part)

L = 73; % Optimum filter length from previous part

wc = 0.44 \* pi;

n = 0:L-1;

h = (2/L) \* cos(wc \* n);

% Generate input signal (sum of 3 sinusoids)

N = 100; % Number of points to plot

n = 0:N-1;

x = 5 \* cos(0.3\*pi\*n) + 22 \* cos(0.44\*pi\*n) + 22 \* cos(0.7 \* pi\* n - (pi/4));

% Apply the filter

y = conv(x, h, 'same');

% Plot input and output signals

figure;

subplot(2,1,1);

plot(n, x);

title('Input Signal: Sum of 3 Sinusoids');

xlabel('Sample Index');

ylabel('Amplitude');

grid on;

subplot(2,1,2);

plot(n, y);

title('Filtered Output Signal');

xlabel('Sample Index');

ylabel('Amplitude');

grid on;

% Calculate and display amplitude ratios

amp\_ratios = abs(fft(y)) ./ abs(fft(x));

freq = (0:length(x)-1) \* (2\*pi/length(x));

ratio\_03pi = amp\_ratios(round(0.3 \* length(x)/2) + 1);

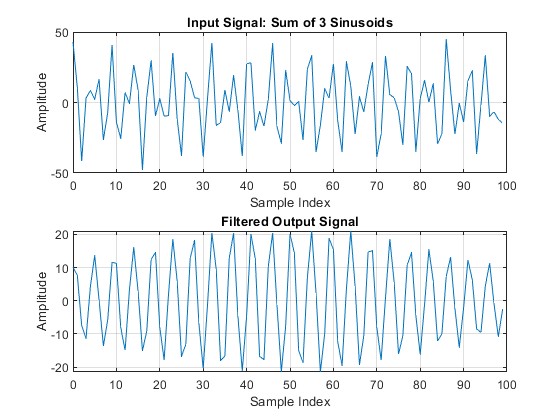
ratio\_044pi = amp\_ratios(round(0.44 \* length(x)/2) + 1);

ratio\_07pi = amp\_ratios(round(0.7 \* length(x)/2) + 1);

fprintf('Amplitude ratio at 0.3π: %.4f\n', ratio\_03pi);

fprintf('Amplitude ratio at 0.44π: %.4f\n', ratio\_044pi);

fprintf('Amplitude ratio at 0.7π: %.4f\n', ratio\_07pi);



0.44π component falls within the passband of our filter whilst 0.3π and 0.7π components fall within the stopbands of the filter.

3.2.f.

L = 10; % Start with L = 10

meetsCriteria = false;

while ~meetsCriteria

[H, w] = designFilter(L);

meetsCriteria = checkCriteria(H, w);

if ~meetsCriteria

L = L + 1;

end

end

fprintf('Smallest filter length L: %d\n', L);

% Plot the frequency response

figure;

plot(w/pi, abs(H));

title(['Frequency Response (L = ', num2str(L), ')']);

xlabel('Normalized Frequency (ω/π)');

ylabel('Magnitude');

grid on;

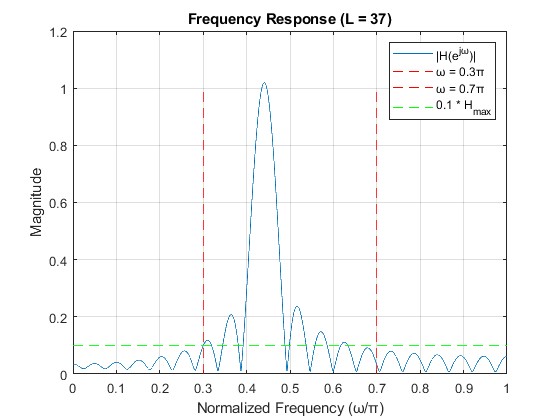
hold on;

plot([0.3 0.3], [0 1], 'r--');

plot([0.7 0.7], [0 1], 'r--');

plot([0 1], [0.1 0.1], 'g--');

legend('|H(e^{jω})|', 'ω = 0.3π', 'ω = 0.7π', '0.1 \* H\_{max}');



Smallest filter length L: 37.