DSP Lab Solved problems

Last updated: 2025-10-21 07:51

LaTeX source here

I asked Gemini to create some problems for me. And these are solution to some of them.

Contents

1 FIR Filter Design for Signal Restoration

You are given a discrete-time signal $x(n)$ which is a combination of a desired low-frequency signal and an unwanted high-frequency noise component. The signal is defined as:

$$x(n) = \cos(0.1\pi n) + 0.5sin(0.8\pi n) \quad \text{for} 0 \leq n \leq 100$$

Question is:

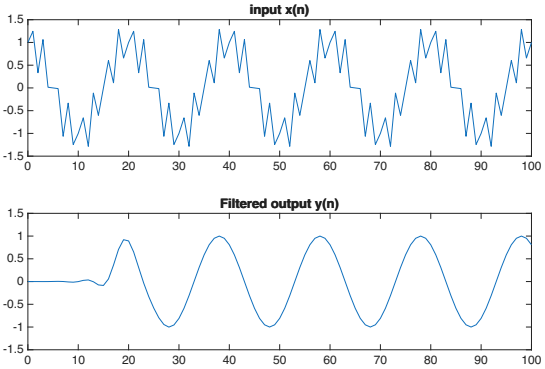
1. Design a digital FIR low-pass filter to remove the high-frequency noise component.
2. Filter the input signal using designed filter

Program

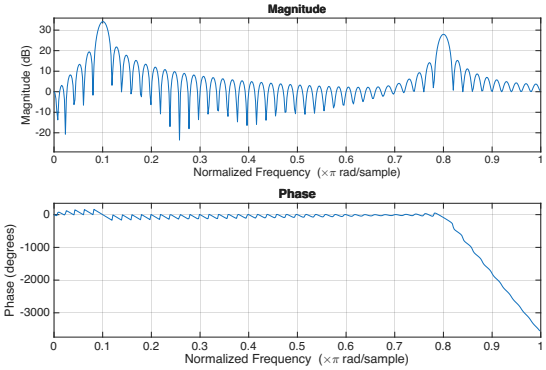
```
clearvars; close all;

% given signal
n = 0:100;
x = cos(0.1 * pi * n) + 0.5*sin(0.8 * pi * n);

%%% design LPF
wp = 0.2*pi;
ws = 0.5*pi;
TW = (ws - wp);
fc = (ws + wp)/2/pi;
```

```
% As = 60dB; so use blackman which has As = 74dB
M = ceil(11 * pi / TW);
m = 0:M-1;
m = m - ceil((M-1)/2);
w = blackman(M);

hd = fc * sinc(fc * m);
h = hd .* w'; % required filter

%%% Filter the input signal
y = filter(h, 1, x);

tiledlayout(2, 1);
nexttile; plot(n, x); title("input x(n)");
nexttile; plot(n, y); title("Filtered output y(n)");
% plot frequency responses and spectrums
figure; freqz(h);
figure; freqz(x);
figure; freqz(y);
```

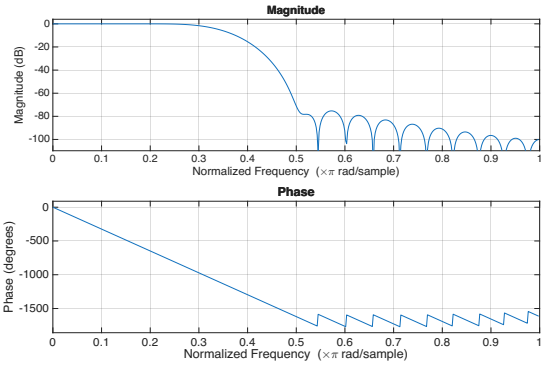
Plotted signal



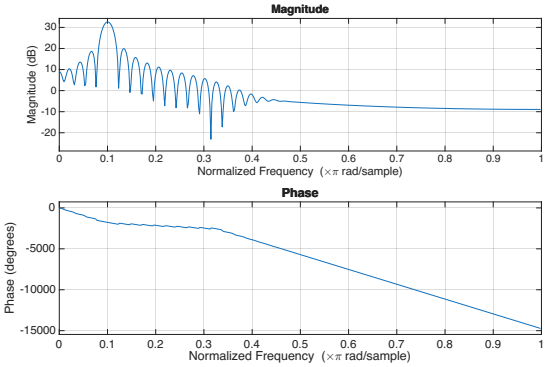
Input message spectrum



Filter's frequency response



Filtered Output spectrum



2 LTI System Stability Analysis and Modification

Consider the LTI system described by the following difference equation:

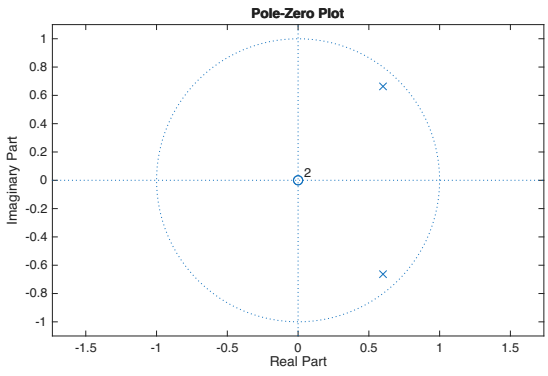
$$y(n) - 1.2y(n-1) + 0.8y(n-2) = x(n)$$

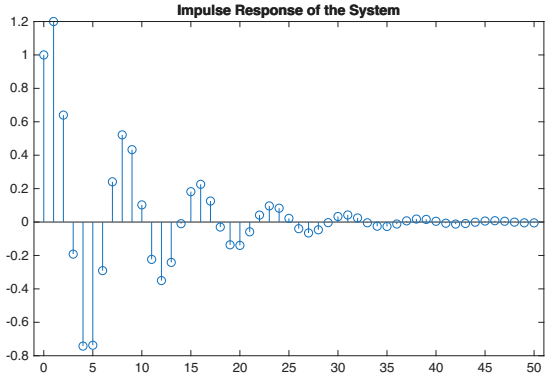
Question:

1. Determine if this system is stable or not
2. Plot poles and zeros of the system
3. Compute and plot the impulse response of the system for $0 \leq n \leq 50$
4. Modify the system by changing only the coefficient of y(n-1) (the -1.2 term) so that the poles are located at $p = 0.5 \pm 0.5j$. Plot the pole-zero diagram for new system

Program for Question 1-3:

```
close all; clearvars;
% given system: y(n)  1.2y(n1) + 0.8y(n2) = x(n)
num = 1; den = [1, -1.2, 0.8];

%%% plot zeros and poles
zplane(num, den);

%%% check if system is stable or not
% by checking if all poles lie inside unit circle
system = filt(num, den);
if (abs(pole(system)) < 1)
    disp("stable");
else
    disp("unstable");
end

%%% compute impulse response
n = 0:50;
impulse = n == 0;
response = filter(num, den, impulse);
figure;
stem(n, response);
title('Impulse Response of the System');
```





Program For Question 4

Given system has 2 zeros at $p = 0$ and 2 poles. To change the poles location to $p = 0.5 \pm 0.5j$, First write it in system function form, change denominator expressions and work backwards: (here z^2 implies double zero at $p = 0$)

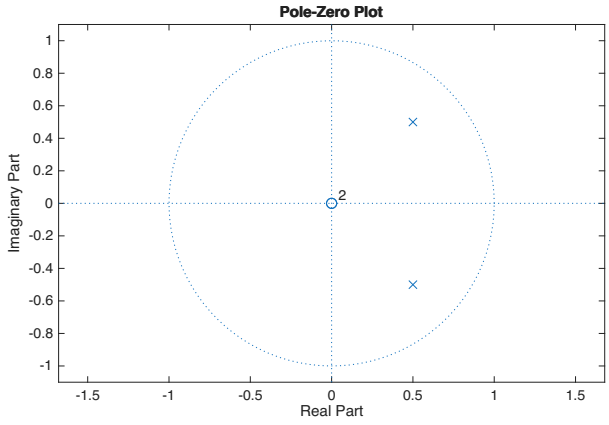
$$\begin{aligned} H'(z) &= \frac{z^2}{(z - p_1)(z - p_2)} \\ H'(z) &= \frac{z^2}{[z - (0.5 + 0.5i)]\,[z - (0.5 - 0.5i)]} \\ &= \frac{z^2}{z^2 - 1z + 0.5} \\ \frac{Y(z)}{X(z)} &= \frac{1}{1 - 1z^{-1} + 0.5z^{-2}} \\ \Rightarrow X(z) &= Y(z)\left[1 - 1z^{-1} + 0.5z^{-2}\right] \end{aligned}$$

Applying inverse Z-transform gives the required system:

$$x(n) = y(n) - y(n-1) + 0.5y(n-2)$$

Hence the implementation:

```
num = 1; den = [1, -1, 0.5];
zplane(num, den);
```



new system function

3 Simulation of Gibbs oscillation

Simulate Gibbs oscillation on a given signal $s(t) = \text{sgn}(t)$.

Background

The Fourier series of a complex-valued periodic function $s(t)$, integrable over the interval $[a, b]$on the real line, is defined as:

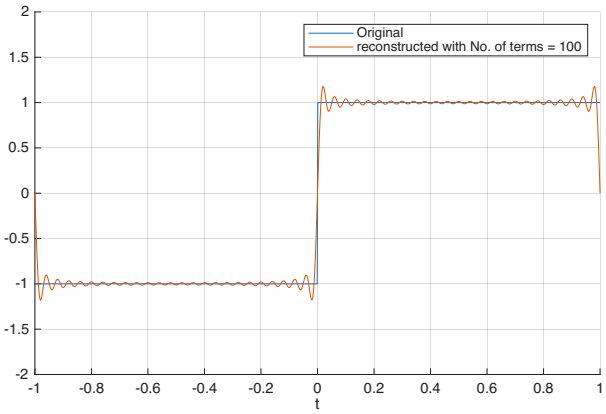
$$s(t) = \sum_{n=-\infty}^{\infty} c_n e^{i2\pi \frac{n}{T} t}$$

Where $T = b - a$ is the period of function. Fourier coefficients c_n are:

$$c_n = \frac{1}{T} \int_a^b s(t)\ e^{-i2\pi \frac{n}{T} t} dt$$

Program

```
clearvars; close all;

a = -1;
b = 1;
t = linspace(a, b, 1000);
T = b - a;

% input signal
s = @(t) sign(t);

% Reconstruct signal with N harmonics on each side
N = 50;
f_series = zeros(size(t));

% coefficients
c = @(n) 1 / T * integral(@(t) s(t) .* exp(-2j * pi * t * n / T), a, b);

% Fourier series summation
for n = -N:N
    f_series = f_series + c(n) * exp(2j * pi * n * t / T);
end

% Plot the actual and reconstructed signal
hold on;
plot(t, s(t));
plot(t, real(f_series));
xlabel('t');
legend(["Original", "reconstructed with No. of terms = " + N*2]);
grid on;
ylim([-2, 2]);
```



Notice the wide oscillation near discontinuities

4 Extraction of periodic signals from noisy signal

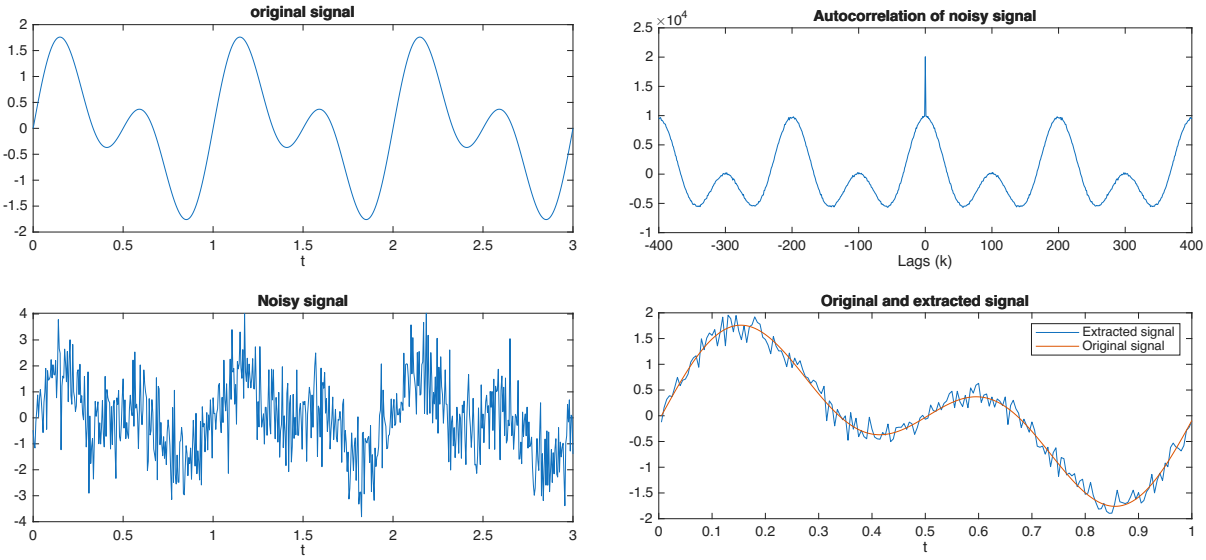
Given a input signal $s(t) = sin(2\pi t) + sin(4\pi t)$, add some white noise to it (obtaining $s_{noisy}(t)$) and extract the periodic signal

Program

To do this, evaluvate $s_{noisy}(t)$ over very large time interval and finding the period using autocorrelation. The autocorrelation of periodic signal is also periodic with same period as that of original signal Then by observing the plot, obtain a estimate of time period T_{est} and slice $s_{noisy}(t)$ into array of signals with length T_{est} (using `reshape(s, cols, rows)` function).

```
clc; clearvars; close all;

periods = 50; % number of periods
T = 200; % number of samples in a time period
t = 0:1/T:periods;

% signal
s = sin(2 * pi * t) + sin(4 * pi * t);

% add some noise
noise = randn(size(t));
s_noisy = s + noise;

% plot signal
subplot(2, 1, 1);
plot(t, s);
xlim([0, 3]); % just show 3 periods
title("original signal");
xlabel("t");

subplot(2, 1, 2);
plot(t, s_noisy);
xlim([0, 3]);
title("Noisy signal");
xlabel("t");
```

```
%%% Autocorrelation
[Rxx, lags] = xcorr(s_noisy, s_noisy);

figure;
subplot(2, 1, 1);
plot(lags, Rxx);
xlim([-T*2, T*2]); % to show few periods around central part
title("Autocorrelation of noisy signal");
xlabel("Lags (k)");

%%% notice that autocorrelation is periodic with sample period 200
%%% So we slice noisy signal into segments of length 200
%%% and average all of them

% estimated period of signal
T_est = 200;
no_of_segments = floor(length(s_noisy) / T_est);

% take (T_est * no of segments) points from signal
s_new = s_noisy(1:T_est*no_of_segments);

% reshape to a matrix: cols = T_est, rows = no of segments
s_slices = reshape(s_new, T_est, no_of_segments);

% average along row to get estimate of the signal
s_est = mean(s_slices, 2);

t_1peroid = (1:T_est) / T_est;
subplot(2, 1, 2);
plot(t_1peroid, s_est);
hold on;
plot(t_1peroid, s(1:T_est));
xlabel("t");
title("Original and extracted signal");
legend(["Extracted signal", "Original signal"])
```



Notice that autocorrelation (top right) is periodic with period 200

5 Signal Reconstruction from Aliased Components

An analog signal containing two important frequency components, $f_1 = 100$ Hz and $f_2 = 800$ Hz, was sampled at $f_s = 1000$ Hz. This sampling rate satisfies the Nyquist criterion for f_1 but violates it for f_2.

Question:

1. Determine the aliased frequency of the 800 Hz component in the discrete-time domain. The formula for the aliased frequency is $f_{alias} = |f_2 - kf_s|$, where k is an integer that brings the result into the range $[0, f_s/2]$.

2. Write a MATLAB script to generate the resulting discrete-time signal, $x(n)$, for $N = 200$ samples (sampling frequency of 200Hz).

3. Design two separate FIR filters with sufficient stop band attenuation ($\approx$ 70dB) to isolate each of the original components:

   (a) A low-pass filter to isolate the 100 Hz component.

   (b) A band-pass filter to isolate the 800 Hz component (which now appears at its aliased frequency).

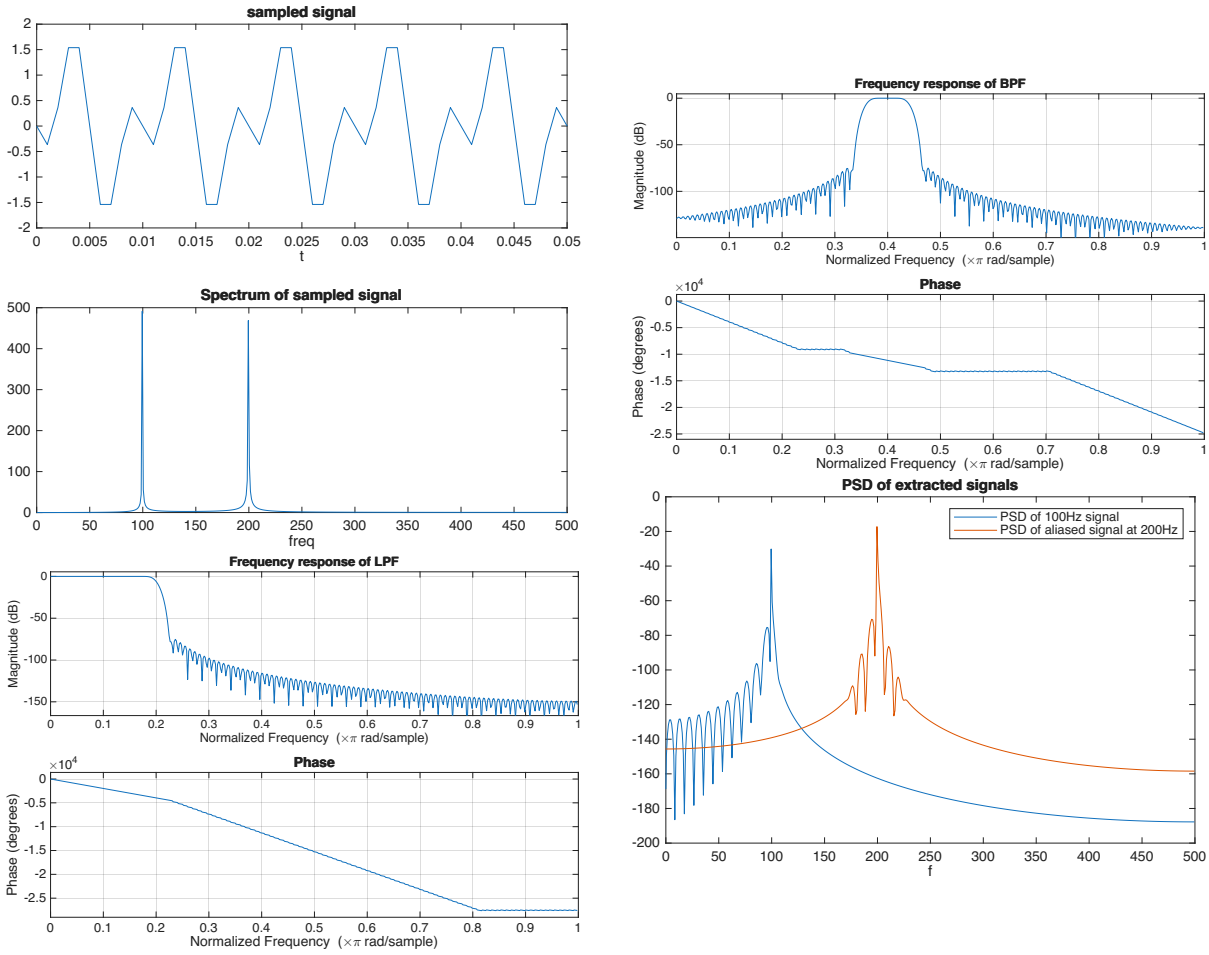
4. Apply each filter to the mixed signal $x(n)$ to get two output signals, $y_{100}(n)$ and $y_{800}(n)$.

Program

```
clc; clearvars; close all;

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

x = sin(2 * pi * 100 * t) + sin(2 * pi * 800 * t);
```

```
x_spec = abs(fftshift(fft(x)));
N = length(x);
f = (-N/2:N/2-1);
%%% plot signals
subplot(2, 1, 1);
plot(t, x); xlabel("t");
xlim([0, 5/100]);
title("sampled signal");

subplot(2, 1, 2);
plot(f, x_spec);
xlim([0, fs/2]);
xlabel("freq");
title("Spectrum of sampled signal");

%%% design filters

% LPF for signal at f = 100Hz
fcl = 100 / (fs / 2);

TW = 0.05 * pi;
M = ceil(11 * pi / TW);
n = 0:M-1;
m = n - ceil((M-1)/2);
w = blackman(M);
hd = fcl * sinc(fcl * m);
hl = hd .* w';
figure;
freqz(hl);
title("Frequency response of LPF");

% filtered output
y100 = filter(hl, 1, x);

% a narrow BPF for aliased signal at f_alias = 200Hz
% with passband bandwidth of 40Hz (to get maximum attenuation for other
% signals)
fch = 220 / (fs / 2);
fcl = 180 / (fs / 2);

hd2 = fch * sinc(fch * m) - fcl * sinc(fcl * m);
hb = hd2 .* w';
figure;
freqz(hb);
title("Frequency response of BPF");

% filtered output
y900 = filter(hb, 1, x);

% plot psd of both signal
psd_y100 = (abs(fftshift(fft(y100))) / N) .^ 2;
psd_y900 = (abs(fftshift(fft(y900))) / N) .^ 2;
```

```
f = (-N/2:N/2-1);
figure;

plot(f, 10 * log(psd_y100));
hold on;
plot(f, 10 * log(psd_y900));
xlim([0, fs/2]);
title("PSD of extracted signals");
legend(["PSD of 100Hz signal", "PSD of aliased signal at 200Hz"]);
xlabel("f");
```



6 Multi-Band Stop FIR Filter Design

You have a signal that is corrupted by noise at two specific frequency bands: a low-frequency hum and a high-frequency hiss.

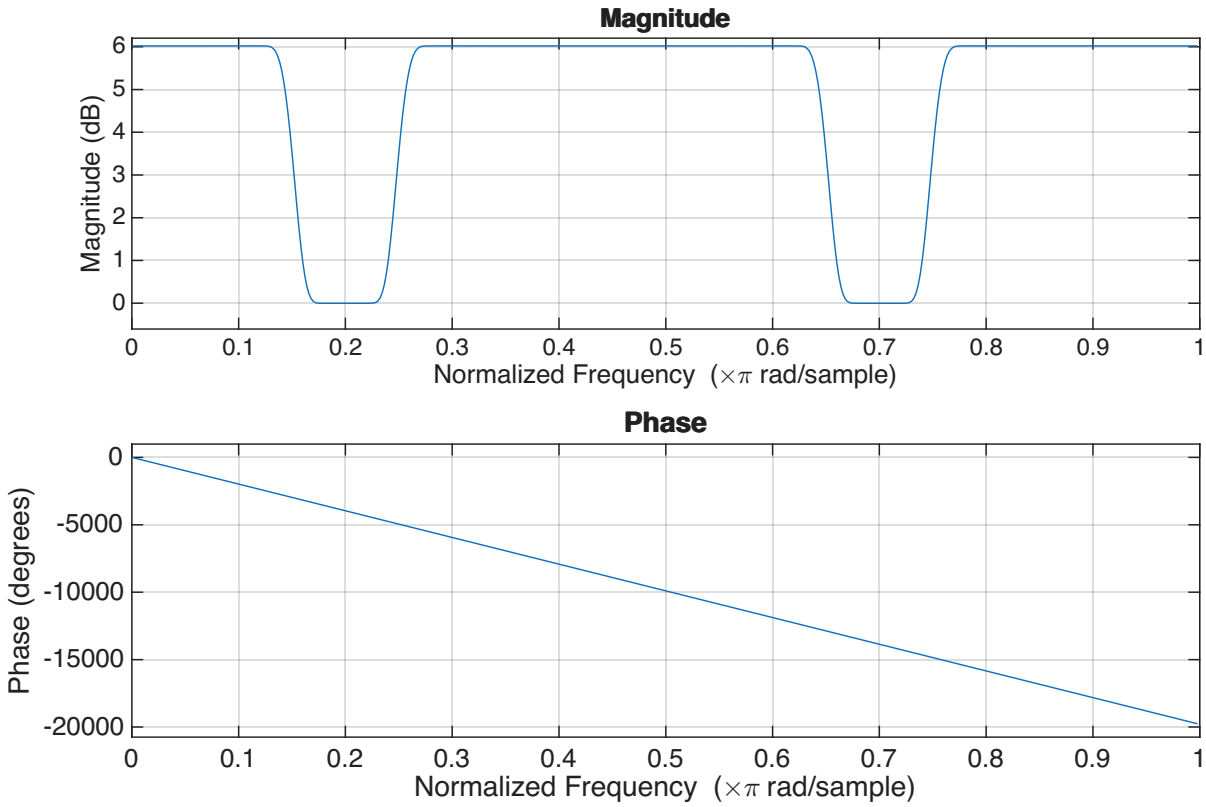
- Low-frequency hum: centered around $\omega = 0.2\pi$.
- High-frequency hiss: centered around $\omega = 0.7\pi$.

Task: Design a single FIR filter that passes all frequencies except for those in the two specified bands.
Specifications:

1. Stopband 1: $[0.15\pi, 0.25\pi]$
2. Stopband 2: $[0.65\pi, 0.75\pi]$
3. Minimum Stopband Attenuation: 55 dB.

Program

```
clc; clearvars; close all;
% pass band freqs
fp1 = 0.2;
fp2 = 0.6;
% stop bands
fs1 = [0.15 , 0.25];
fs2 = [0.65 , 0.75];

% transition width (same for both filters)
TW = fp1 - fs1(1);
% As > 55dB, choose blackmann
M = ceil(11  / TW);
n = 0:M-1;
m = n - ceil((M-1) / 2);
% construct 2 BPF filters
BPF1 = fs1(2) * sinc(fs1(2) * m) - (fs1(1) * sinc(fs1(1) * m));
BPF2 = fs2(2) * sinc(fs2(2) * m) - (fs2(1) * sinc(fs2(1) * m));

% corresponding BRF is:
BRF1 = sinc(m) - BPF1;
BRF2 = sinc(m) - BPF2;

window = blackman(M)';
BRF1 = BRF1 .* window;
BRF2 = BRF2 .* window;

% merge filters:
mergedFilter = BRF1 + BRF2;

freqz(mergedFilter);
```


Magnitude
Magnitude (dB)
6
5
4
3
2
1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase
Phase (degrees)
0
-5000
-10000
-15000
-20000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)