Name: **Group 2**

Alamay, Carl Justine (Lab Report)

Zeng, Stanley Yale (Coding)

Section: **S11B**

Laboratory Activity Title: “**Frequency Response: Bandpass and Nulling Filters**”

**Files Used:**

**Instructions:** , for each of the sections indicated, please provide your corresponding answers to each of the questions indicated in the lab activity section. Please provide relevant information, i.e. **source code, graphs, explanations/answers to the question/s.**

**3.1 Nulling Filters for Rejection**

ww = -pi:pi/100:pi;

figure; % figure for nulling filter subplots

b\_044 = [1 -2\*cos(0.44\*pi) 1]; %0.44\*pi nulling filter

H\_044 = freqz(b\_044,1,ww); %finding the frequency response

subplot(3,1,1);

plot(ww,abs(H\_044));

title('0.44pi Nulling Filter');

b\_07 = [1 -2\*cos(0.7\*pi) 1]; %0.7\*pi nulling filter

H\_07 = freqz(b\_07,1,ww); %finding the frequency response

subplot(3,1,2);

plot(ww,abs(H\_07));

title('0.7pi Nulling Filter');

b\_res = conv(b\_044,b\_07); %cascading the filters using convolution

HH = freqz(b\_res,1,ww); %finding the frequency response

subplot(3,1,3);

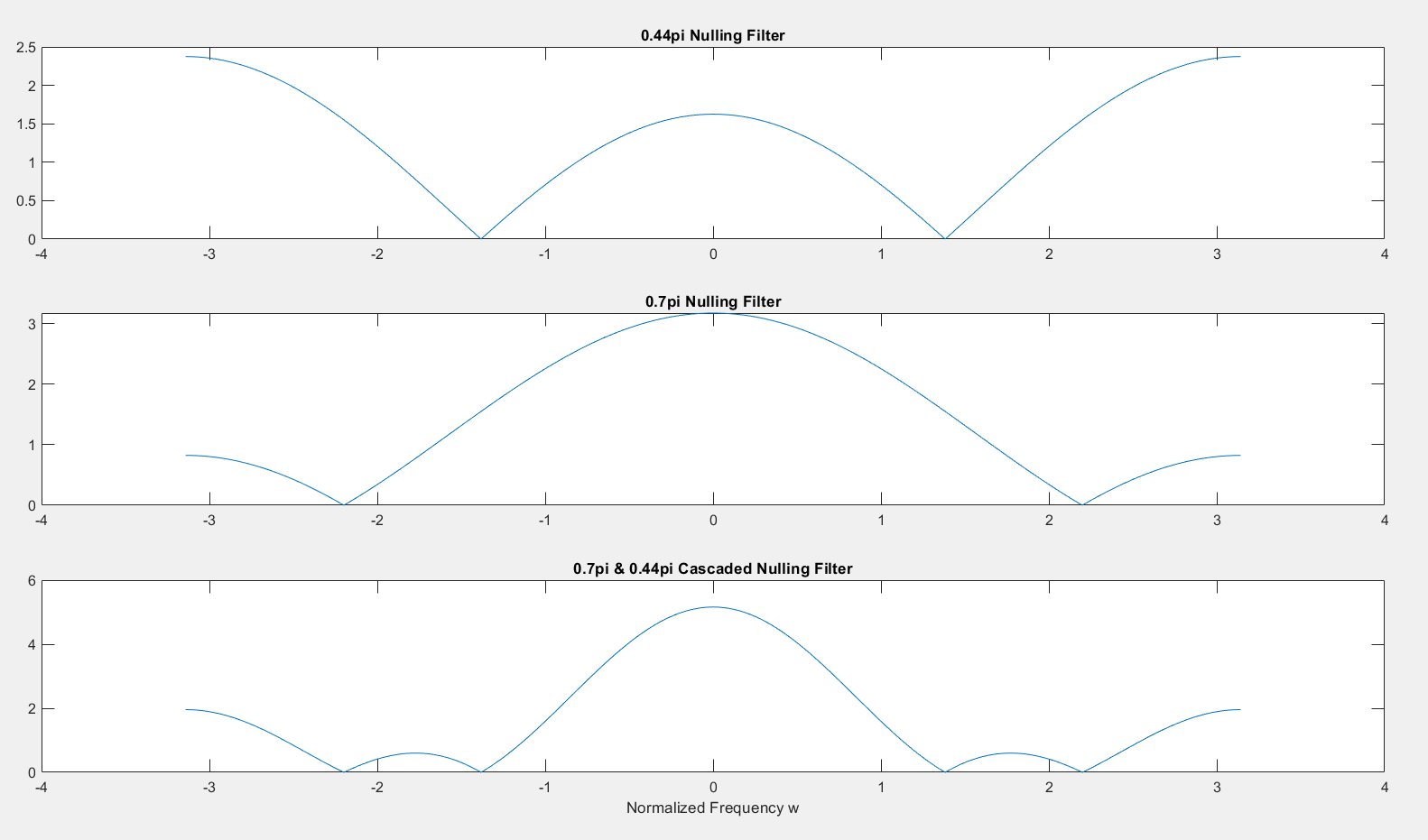
plot(ww,abs(HH));

title('0.7pi & 0.44pi Cascaded Nulling Filter');

xlabel('Normalized Frequency w');

**Code of the Filtering System that would eliminate input frequencies of 0.44π and 0.7π.**

**(a)** The code above is used to demonstrate and visualize how two distinct nulling filters are able to directly affect the overall cascading system. It first starts off by defining a range of frequencies from -π to π, which are used as the basic for evaluating the frequency response of both nulling filters. The first filter nullifies 0.44π while the other nullifies 0.7π, which are defined by their coefficients [1 -2\*cos(0.44 π) 1] and [1 -2\*cos(0.7 π) 1]. The combined effect of cascading both nulling filters result in both of their functionalities being applied simultaneously, resulting in both 0.44π and 0.7π being nullified.



**Graphs of the 0.44π and 0.7π nulling filters, along with the overall cascaded system.**

nn = 0:149; %150 samples

xx = 5\*cos(0.3\*pi\*nn) + 22\*cos(0.44\*pi\*nn-pi/3) + 22\*cos(0.7\*pi\*nn-pi/4); % a signal that is the sum of 3 sinusoids

**Code to generate the sum of three sinusoids.**

**(b)** The code above adds three sinusoids with distinct values for amplitudes, frequencies, and phases. The first sinusoid has an amplitude of 5 and a frequency of 0.3π; the second and third sinusoid both have an amplitude of 22, but the former having a frequency of 0.44π with a phase shift of -π/3 while the latter having a frequency of 0.7π with a phase shift of -π/4. The signal generated from their sum is sampled over a range of 150 samples.

% (c) signal xx through cascaded filter

b\_res = conv(b\_044,b\_07); %cascading the filters using convoution

HH = freqz(b\_res,1,ww); %finding the frequency response

figure

subplot(4,1,1);

plot(nn,xx);

grid on;

title('Original Signal');

xx\_filt = firfilt(b\_res,xx); % filtering the signal through the cascaded system

subplot(4,1,3);

plot(nn,xx\_filt(1:length(nn)));

grid on;

title('Filtered Signal');

% (d) plot first 40 samples

n = 40; % size to show the signal

subplot(4,1,2);

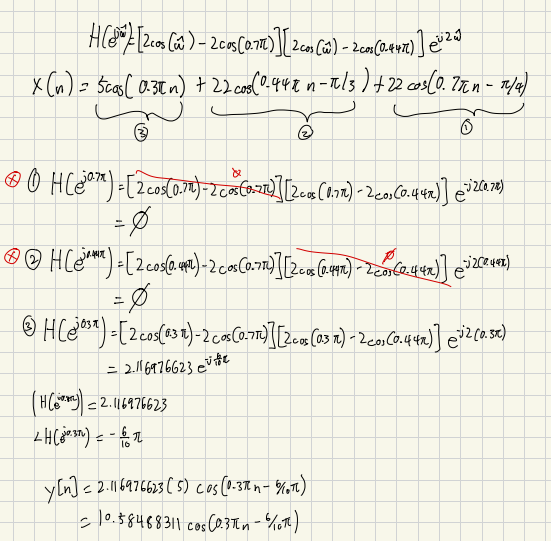
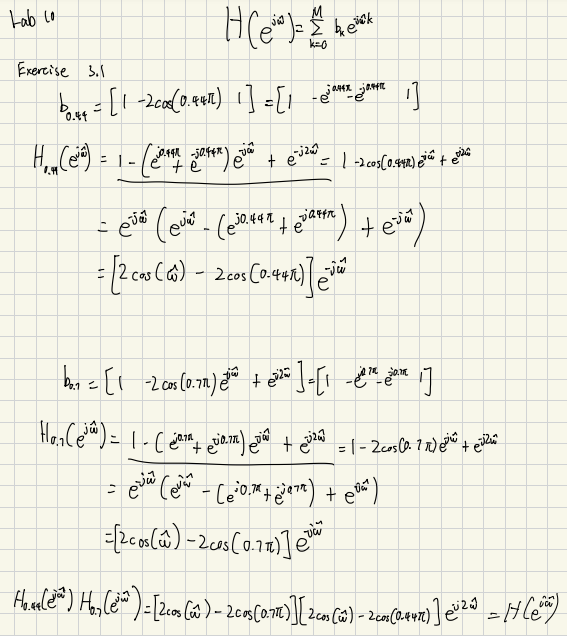
plot(nn(1:n),xx\_filt(1:n));

grid on;

title('First 40 Samples of Filtered Signal');

**Code to generate and visualize the output signal of the sum of three sinusoids.**

**(c – d)** The generated signal from adding 3 different sinusoids is passed through the cascaded system with two nulling filters who are able to nullify the frequency response at 0.44π and 0.7π, as defined in the previous sections. After the application of the cascaded system, the signal is altered in the sense that frequencies at 0.44π and 0.7π are virtually at 0, which also results in the filtered signal being minimalized.



**Mathematical proof for the formula of the output signal.**

**Graph of the of the sum of the three sinusoidal signals under different conditions.A diagram of a graph

Description automatically generated with medium confidence**

subplot(4,1,4);

yy = 10.58488311\*cos(0.3\*pi\*nn-6/10\*pi);

plot(nn,yy);

title('Mathematical Formula Plot of Output Signal')

figure

t\_i = 5; % start time of plot

n\_i = t\_i + 1; % start index of plot

n = 40; % size to show the plot

subplot(2,1,1);

plot(nn(n\_i:n),xx\_filt(n\_i:n));

grid on;

title('n[5 - 40] of Filtered Signal');

subplot(2,1,2)

plot(nn(n\_i:n),yy(n\_i:n));

grid on;

title('n[5 - 40] of Mathematical Plot of Output Signal')

**Code to plot the calculated output signal over a certain range.**

**(e)** In the code above, the calculated output signal with a mathematical formula of 10.58488311cos(0.3πn - 6/10π) is visualized as a showcase of how theoretical calculations are able to closely relate to automated systems of calculations. The filtered signal and the calculated signals are then compared over the range of 5 <= n <= 40, which showcases the effectiveness of the automated filter function by comparing it to the calculated signal.

A graph of a diagram

Description automatically generated with medium confidence

**Graph of the filtered signal and the calculated output signal.**

**(f)** When performing convolution of two systems, the resulting length of the cascaded system would be the sum of their length - 1, or length of a system plus the order of the other system. Reason for why the starting indices compared to the mathematical formula is different is because convolution at n < L or at n <= M is considering 0 values at n < 0, therefore the number of "startup points" to correspond to the mathematical formula is the order M or L - 1.

In the case of the filtered signal and calculated output signal, the FIR filters used both have a length of 3 which would mean that the amount of “start-up” points would equal to 5. This means that the first 4 points on the generated graph would still be influenced by the input signal, resulting in deviations and digressions from the calculated output signal.

**3.2 Simple Bandpass Filter Design**

wc=0.44\*pi;

L = 10;

ww = [0.3\*pi 0.44\*pi 0.7\*pi];

w = (0:pi/100:pi);

ww\_index = [];

for i = 1:3

ww\_index = [ww\_index find(abs(w - ww(i)) < 0.00000001)];

end

w\_pi = w/pi;

figure;

[HH,ww\_abs] = bandfilt(wc,L,1,ww);

HH\_abs = abs(HH); % magnitudes of HH

title('Bandpass centered at 0.44pi at L = 10')

hold on;

plot(w\_pi(ww\_index), HH\_abs(ww\_index),"r o"); % ww red o

fprintf('Magnitudes of the Normalized Frequencies:\n0.3pi: %.4f\n0.44pi: %.4f\n0.7pi: %.4f\n',ww\_abs(1),ww\_abs(2),ww\_abs(3));

% 0.3pi: 0.2836, 0.44pi: 1.0961, 0.7pi: 0.286

**Code to generate a bandpass filter with specified parameters.**

**(a)** The code above generates a bandpass filter with the frequency response centered at 0.44π with a specified length of 10. It first specifies the list of frequencies of interest, those being being 0.3π, 0.44π, and 0.7π, along with the frequency range of 0 to π with a step size of π/100. The indices of the frequency range are then identified using a threshold-based searching method with a tolerance of 10 raised to -8.

The magnitude response of the bandpass filter is then generated using the bandfilt() function, which is then plotted along the frequency range with the specified points of interest. The magnitudes of the normalized frequencies are: **0.2836 for 0.3π, 1.0961 for 0.44π, and 0.286 for 0.7π**.

A graph with a line graph

Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Graph of the generated bandpass filter.**

function [HH,ww\_abs,ww\_angle,ww\_HH]= bandfilt(wc, L, plotyn, ww)

%wc: center frequency (at normalized frequency)

%L: length

%plotyn: to plot or not (0 = no, else = mag only, 2 = angle and mag, 3 = angle only)

w=0:pi/100:pi;

bb = bpf\_bb(wc, L);

HH = freqz(bb,1,w);

%HH=((1-exp(1i.\*L.\*(wc-w)))./(1-exp(1i.\*(wc-w)))+(1-exp(-1i.\*L.\*(wc+w)))./(1-exp(-1i.\*(wc+w))))/L;

HH\_abs = abs(HH);

HH\_angle = angle(HH);

if nargin == 4

ww\_abs =[];

ww\_angle=[];

ww\_HH = [];

for i = 1:length(ww)

index\_ww = find(abs(w - ww(i)) < 0.00000001); % 0.00000001 tolerance

abss = HH\_abs(index\_ww);

angg = HH\_angle(index\_ww);

ww\_abs = [ww\_abs abss];

ww\_angle = [ww\_angle angg];

ww\_HH = [ww\_HH abss\*exp(j\*angg)];

end

else

ww\_abs = NaN;

ww\_angle = NaN;

ww\_HH = NaN;

end

w\_pi = w./pi; % plot according to normalized frequency / pi

if plotyn ~= 0

if plotyn == 2

subplot(2,1,2);

plot(w\_pi,HH\_angle);

xlabel('\omega/\pi');

ylabel('\angleH(e^{j\omega})');

grid on;

subplot(2,1,1);

end

if plotyn ~= 3

plot(w\_pi,HH\_abs);

grid on

xlabel('\omega/\pi');

ylabel('|H(e^{j\omega})|');

end

if plotyn == 3

plot(w\_pi,HH\_angle);

xlabel('\omega/\pi');

ylabel('\angleH(e^{j\omega})');

grid on;

end

end

if nargin == 4

if plotyn ~= 0 && plotyn ~= 3

ww\_index = [];

for i = 1:length(ww)

ww\_index = [ww\_index find(abs(w - ww(i)) < 0.00000001)];

end

hold on;

plot(w\_pi(ww\_index), HH\_abs(ww\_index),"r o"); % ww red o

hold off;

end

end

end

**Function definition of bandfilt.m.**

The user-defined function bandfilt() calculates the frequency response of a bandpass filter using necessary parameters in the center frequency, filter length, and optionally the specific frequency range. It utilizes the user-defined function bpf\_bb() to generate filter coefficients and uses freqz() function to evaluate the filter's response across a normalized frequency range. The function's ability to plot the frequency response depends on the plotyn parameter, which determines whether or not to plot the magnitude and phase angle.

function [bb]= bpf\_bb(wc, L)

%wc: center frequency (at normalized frequency)

%L: length

w=0:pi/100:pi;

bb = (2/L)\*cos(wc\*(0:L-1));

end

**Function definition for bpf\_bb.m.**

A graph with a line graph

Description automatically generated

**Graph of the generated bandpass filter.**

wc=0.44\*pi; % function [HH]= bpf\_bands(wc, L, plotyn)

L = 10;

w=0:pi/100:pi;

figure;

HH = bandfilt(wc,L,1);

title\_str = sprintf('Bandpass centered at $%.2f\\pi$ for $L = %d$', wc/pi, L);

title(title\_str, 'Interpreter', 'latex');

HH; % frequency response function at wc = 0.44\*pi

HH\_abs = abs(HH); % magnitudes of HH

**Code to set up the parameters of the bandpass filter, and to generate its frequency response.**

(b) The code above initializes the initial parameters of the bandpass filter, while subsequently generating its frequency response using the bandfilt() function. It’s centered at 0.44π with a length of 10, where the frequency response is then plotted and visualized across the specified range of 0 to π.

% Finding H\_max

H\_max = max(HH\_abs);

H\_max\_i = find(HH\_abs == H\_max); % finding index of H\_max

% determining passband region

passband\_region\_index = find(HH\_abs/H\_max >= 1/sqrt(2));

passband\_cutoff\_index = [passband\_region\_index(1) passband\_region\_index(end)];

passband\_cutoff\_mag = [HH\_abs(passband\_cutoff\_index(1)) HH\_abs(passband\_cutoff\_index(end))];

passband\_cutoff\_freq = [w(passband\_cutoff\_index(1))/pi w(passband\_cutoff\_index(end))/pi];

% determining stopband region

first\_stopband\_region\_index = find(HH\_abs(1:passband\_cutoff\_index(1))/H\_max < 1/4);

second\_stopband\_region\_index = passband\_cutoff\_index(end) + find(HH\_abs(passband\_cutoff\_index(end):length(HH\_abs))/H\_max < 1/4) - 1;

stopband\_cutoff\_index = [first\_stopband\_region\_index(end) second\_stopband\_region\_index(1)];

stopband\_region\_index = find(HH\_abs/H\_max < 1/4);

**Code to identify the maximum response and passband, along with the stopband region.**

The maximum magnitude of the bandpass filter’s frequency response is located and then its index is calculated at the beginning. By comparing each point in the frequency response’s magnitude to the maximum, the passband region is identified as the frequencies where the response is at least 1 over the square root of the maximum magnitude, which is calculated to be 0.707. The cutoff frequencies are then calculated, which provides an apparent boundary for the bandpass filter.

The stopband regions of the frequency response are then calculated and identified as the areas where the bandpass filter’s magnitude drops below ¼ of the maximum magnitude. By calculating both the lower and upper stopband cutoff points, it will pinpoint the specific ranges in the frequency response where the signal is significantly weakened.

w\_pi = w./pi;

hold on;

plot(w\_pi(passband\_cutoff\_index), HH\_abs(passband\_cutoff\_index),"r x"); % passband\_cutoff red x

plot(w\_pi(passband\_region\_index), HH\_abs(passband\_region\_index),"r ."); % passband\_region red .

plot(w\_pi(H\_max\_i), HH\_abs(H\_max\_i),"o b"); % H\_max blue circle

plot(w\_pi(stopband\_cutoff\_index), HH\_abs(stopband\_cutoff\_index),"x g") % plot stopband\_cutoff green x

plot(w\_pi(stopband\_region\_index), HH\_abs(stopband\_region\_index),". g") % plot stopband\_region green .

**Code to plot the passband and stopband regions.**

A graph of a function

Description automatically generated

**Graph of the generated frequency response of the specified bandpass filter.**

L = 20

bpf\_bands(0.44\*pi, 20, 1);

L = 40

bpf\_bands(0.44\*pi, 40, 1);

**Snippets of code to generate the same bandpass filter with varying lengths.**

The short code above demonstrates calling a user defined function called bpf\_bands() which uses the same fundamental logic utilized in generating the bandpass filter shown in the previous section. The aforementioned function accepts three parameters: the central frequency, the length, and a Boolean expression to determine whether or not the frequency response will be graphed or not.

But instead of a length of 10, two different lengths of 20 and 40 are passed to the function to be computed. The same process will be repeated to generate their frequency response - identifying both the stopband and passband regions with respect to the specified length, and the generated frequency response will be graphed to showcase the different relevant regions of the passband filter.

function [HH]= bpf\_bands(wc, L, plotyn)

%wc: center frequency (at normalized frequency)

%L: length

%plotyn: to plot or not (0 = no, else = yes)

w=0:pi/100:pi;

p = 0;

if plotyn ~= 0

p = 1;

end

figure;

% frequency response function at wc = 0.44\*pi

HH = bandfilt(wc,L,p);

HH\_abs = abs(HH); % magnitudes of HH

% Finding H\_max

H\_max = max(HH\_abs);

H\_max\_i = find(HH\_abs == H\_max); % finding index of H\_max

% determining passband region

passband\_region\_index = find(HH\_abs/H\_max >= 1/sqrt(2));

passband\_cutoff\_index = [passband\_region\_index(1) passband\_region\_index(end)];

passband\_cutoff\_mag = [HH\_abs(passband\_cutoff\_index(1)) HH\_abs(passband\_cutoff\_index(end))];

passband\_cutoff\_freq = [w(passband\_cutoff\_index(1))/pi w(passband\_cutoff\_index(end))/pi];

% determining stopband region

first\_stopband\_region\_index = find(HH\_abs(1:passband\_cutoff\_index(1))/H\_max < 1/4);

second\_stopband\_region\_index = passband\_cutoff\_index(end) + find(HH\_abs(passband\_cutoff\_index(end):length(HH\_abs))/H\_max < 1/4) - 1;

stopband\_cutoff\_index = [first\_stopband\_region\_index(end) second\_stopband\_region\_index(1)];

stopband\_region\_index = find(HH\_abs/H\_max < 1/4);

w\_pi = w./pi;

if plotyn ~= 0

title\_str = sprintf('Bandpass centered at $%.2f\\pi$ for $L = %d$', wc/pi, L);

title(title\_str, 'Interpreter', 'latex');

hold on;

plot(w\_pi(passband\_cutoff\_index), HH\_abs(passband\_cutoff\_index),"r x"); % passband\_cutoff red x

plot(w\_pi(passband\_region\_index), HH\_abs(passband\_region\_index),"r ."); % passband\_region red .

plot(w\_pi(H\_max\_i), HH\_abs(H\_max\_i),"o b"); % H\_max blue circle

plot(w\_pi(stopband\_cutoff\_index), HH\_abs(stopband\_cutoff\_index),"x g") % plot stopband\_cutoff green x

plot(w\_pi(stopband\_region\_index), HH\_abs(stopband\_region\_index),". g") % plot stopband\_region green .

hold off;

end

end

**Function definition of bpf\_bands.m.**

A graph of a graph

Description automatically generated

**Graph of the frequency response of the generated bandpass filter with length of 20.**

A graph of a graph

Description automatically generated

**Graph of the frequency response of the generated bandpass filter with length of 40.**

wc=0.44\*pi;

L = 10;

figure;

ww = [0.3\*pi 0.44\*pi 0.7\*pi];

bpf\_bands(wc,L,1);

hold on;

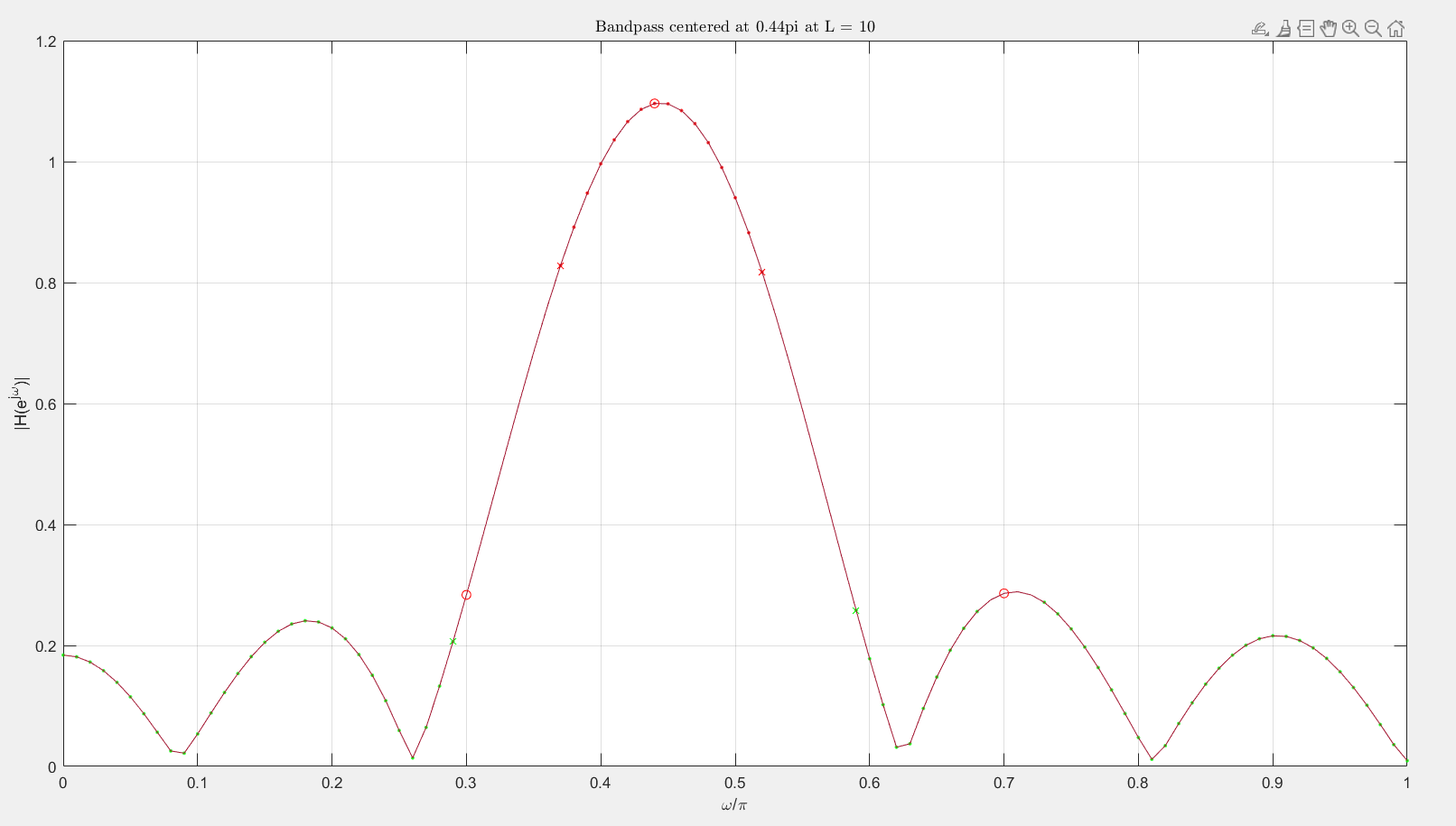
[HH,ww\_abs] = bandfilt(wc,L,1,ww);

title('Bandpass centered at 0.44pi at L = 10')

fprintf('Magnitudes of the Normalized Frequencies:\n0.3pi: %.4f\n0.44pi: %.4f\n0.7pi: %.4f\n',ww\_abs(1),ww\_abs(2),ww\_abs(3));

**Graph of the frequency response of the generated bandpass filter with length of 10.**

**(c)** The frequency that are relatively unchanged are regions within the passbands, or in the region marked red (normalized frequencies from 0.37π to 0.52π). And that frequencies outside the passband, and near (0.3π) and within the pass (0.7π) will have a significantly reduced magnitude.



**Graph of the frequency response of the generated bandpass filter with length of 10.**

The regions marked in red are designed to cater to the passband filter’s purpose of passing frequencies around 0.44π with minimal alterations, while frequencies outside the passband are reduced in magnitude due to the filter’s selectivity. In other words, it’s the filter’s way of reducing “unwanted” frequencies.

wc=0.44\*pi;

%L = 10;

w1 = 0:pi/100:0.3\*pi; % |w| <= 0.3pi

w2 = 0.7\*pi:pi/100:pi; % 0.7pi <= |w| <= pi

ww = [w1 w2];

for i = 10:1:100

L = i;

[HH,ww\_abs] = bandfilt(wc,i,0,ww);

if max(ww\_abs) <= 0.1

break;

end

end

figure;

bandfilt(wc,L,1,ww);

hold on;

title\_str = sprintf('Bandpass centered at $%.2f\\pi$ for $L = %d$', wc/pi, L);

title(title\_str, 'Interpreter', 'latex');

hold off;

fprintf('Smallest Length for less than a factor of 10 on\n[|w| <= 0.3pi] and [0.7pi <= |w| <= pi] is L = %d',L); % L = 37

**Code to identify the smallest possible length of a bandpass filter.**

**(d)** The code above searches for the smallest possible length for the bandpass filter so that the certain frequency components are greatly reduced at 0.3π and 0.7π, specifically below and above the respective frequencies. This is achieved by incrementally testing filter lengths individually from 10 to 100 using the bandfilt() function to determine which frequency responses’ maximum value does not exceed 0.1.

After the loop end, the specific length is then used to generate the frequency response of a bandpass filter using that specific length value. And after the code snippet end, the specific length value is generated. The smallest length for less than a factor of 10 on |w| <= 0.3π and 0.7π <= |w| <= π is calculated to be 37.

A graph of a graph

Description automatically generated

**Graph of the frequency response of the bandpass filter at length 37.**

nn = 0:99; %100 samples

xx = 5\*cos(0.3\*pi\*nn) + 22\*cos(0.44\*pi\*nn-pi/3) + 22\*cos(0.7\*pi\*nn-pi/4); % a signal that is the sum of 3 sinusoids

xx\_arr = [5\*cos(0.3\*pi\*nn); 22\*cos(0.44\*pi\*nn-pi/3); 22\*cos(0.7\*pi\*nn-pi/4)];

ww = [0.3\*pi 0.44\*pi 0.7\*pi];

figure;

subplot(2,1,1);

plot(nn,xx);

title('Input Signal');

% Filtering

L = 37;

wc=0.44\*pi;

[HH,ww\_abs,ww\_angle,ww\_HH] = bandfilt(wc,L,0,ww);

yy = 0;

for i = 1:length(ww\_HH)

yy = yy+ww\_HH(i)\*xx\_arr(i,:);

end

subplot(2,1,2);

plot(nn,yy(1:length(nn)));

xlabel('n');

title('Output Signal');

**Code to plot 100 points of the sum of 3 sinusoids and its filtered signal.**

**(e)** With the first sinusoid having an amplitude of 5 and a frequency of 0.3π; the second and third sinusoid both having an amplitude of 22, but the former having a frequency of 0.44π with a phase shift of -π/3 while the latter having a frequency of 0.7π with a phase shift of -π/4. The magnitudes of their normalized frequencies are: 0.2836 for 0.3π, 1.0961 for 0.44π, and 0.286 for 0.7π.

Given the following effects caused by the filter with respect to the given frequencies at 0.3π, 0.44π, and 0.7π, their corresponding resulting magnitude is as above. The involvement of frequencies 0.3pi and 0.7pi are thus significantly smaller, this phenomenon can be observed in the graph that the output signal is overall less sporadic than the input signal, this is due to the lessen involvement of frequencies other than 0.44pi.

A screenshot of a graph

Description automatically generated

**Graph of the first 100 samples of the sum of 3 sinusoids and its filtered signal.**

wc=0.44\*pi;

L = 37;

ww = [0.3\*pi 0.44\*pi 0.7\*pi];

figure;

bandfilt(wc,L,1,ww);

**Code to generate the frequency response of the same bandpass filter with only magnitude.**

**(f)** The frequency response function is a function of normalized frequency, in which when receives a corresponding input signal (which is the sum of pure form signals), extracted are which is the frequencies of the input signal, such that each frequency has a different magnitude (increased/reduced/rejected) output given the filter. The simplest explanation would be to understanding the frequency response magnitude plot. Encircled are the corresponding frequencies given in the example of problem 3.2 (e), each of the frequency have their corresponding reference on the plot, thus the result of a given signal at a certain frequency is relative to the magnitude given by the frequency response function.

A graph with lines and numbers

Description automatically generated

**Graph of the frequency response of the same bandpass filter with only magnitude.**