Memorial University of Newfoundland

ECE 8600: Design Assignment 1

Aaron Wilcox 201810330

April 1, 2023

Introduction

This assignment describes the design of low and highpass filters for the purpose of separating a signal into its high and low dsicrete time frequency components.

A filter bank using a highpass and a lowpass filter in parallel, was the main idea of the design.

1 Scheme 1

1.1 Task a

The filers to be implemented are:

$$H_0 = \frac{1}{2}(1 + Z^{-1})$$
$$H_1 = \frac{1}{2}(1 - Z^{-1})$$

MATLAB code for task a is in the section 'a3scheme1ResponsePlots.m' of Appendix A.

Frequency Response Plots:

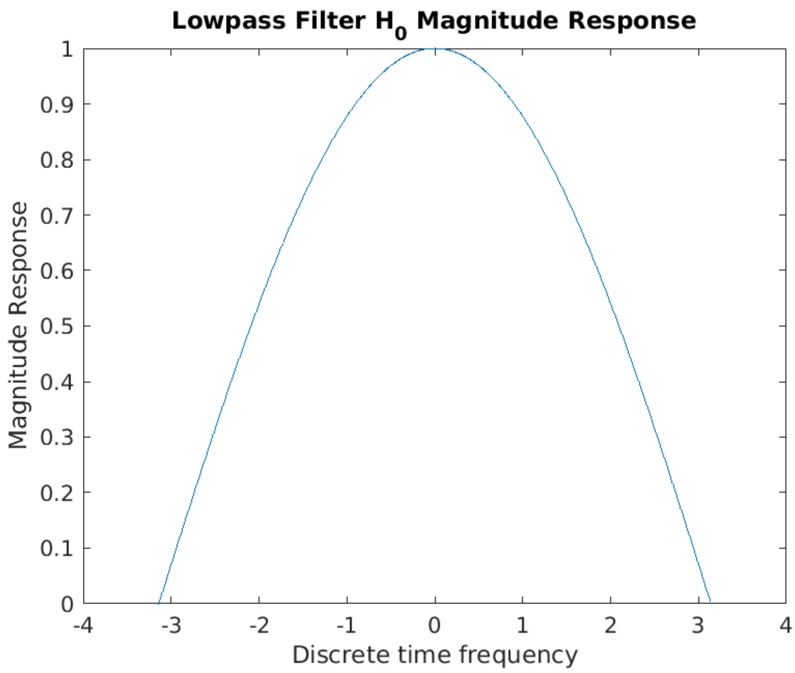

Figure 1.1: Magnitude of the frequency response for H_0

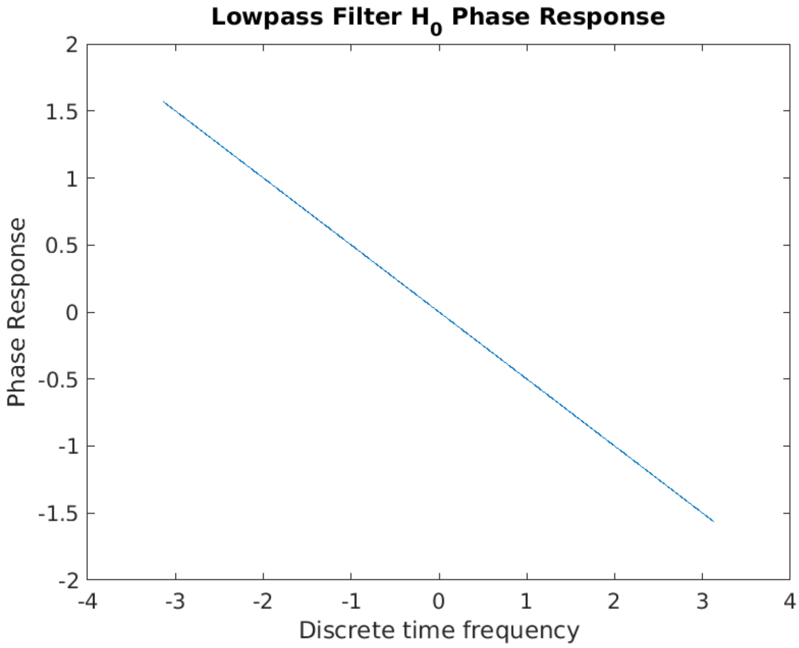

Figure 1.2: Phase of the frequency response for H_0

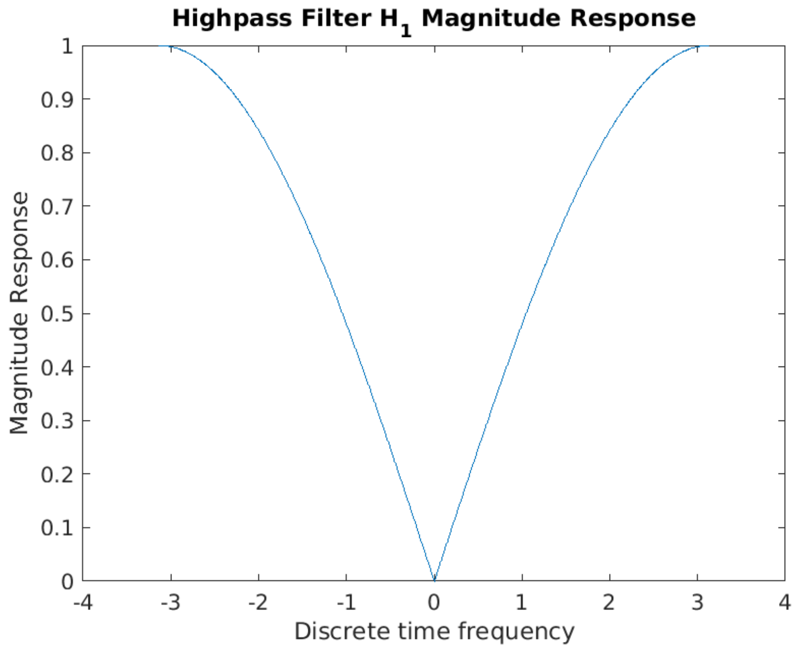

Figure 1.3: Magnitude of the frequency response for H_1

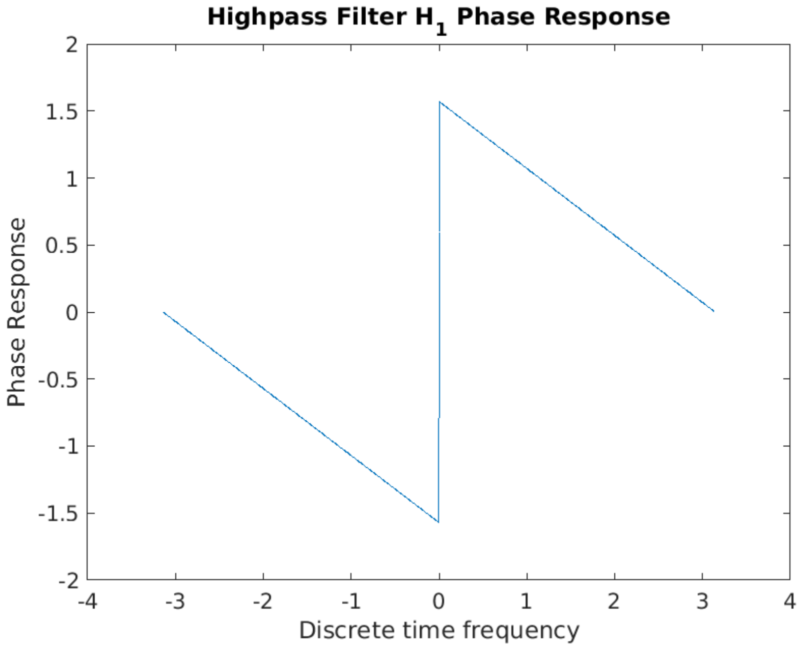

Figure 1.4: Phase of the frequency response H_1

Task b

The filers to be implemented are:

$$G_0 = /frac12 * (1 + Z^{(} - 1))$$
$$G_1 = /frac12 * (-1 + Z^{(} - 1))$$

MATLAB code for task a (part1Filter.m of Appendix A):

```
z = tf('z',1/(2*pi*10000));

G\textunderscore 0 = 1/2*(1+z^(—1));
G\textunderscore 1 = 1/2*(—1+z^(—1));

frequencies=(—pi:(2*pi/1000):pi);
frequencies=frequencies(1:1000);
frequencies2=exp(j.*frequencies);
G\textunderscore 0\textunderscore resp  = freqresp(G\textunderscore 0,
    frequencies2);
G\textunderscore 1\textunderscore resp  = freqresp(G\textunderscore 1,
    frequencies2);

fignum=1;

plotResp(1, fignum, 1, 1, abs(G\textunderscore 0\textunderscore resp),
    frequencies, [1000], ...
[Lowpass Filter G_0 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

```

```
angle\textunderscore of\textunderscore signal = angle(G\textunderscore 0\
    textunderscore resp);

plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
[Lowpass Filter G_0 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);



plotResp(1, fignum+2, 1, 1, abs(G\textunderscore 1\textunderscore resp),
    frequencies, [1000], ...
[Highpass Filter G_1 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

angle\textunderscore of\textunderscore signal = angle(G\textunderscore 1\
    textunderscore resp);

plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
[Highpass Filter G_1 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);
```

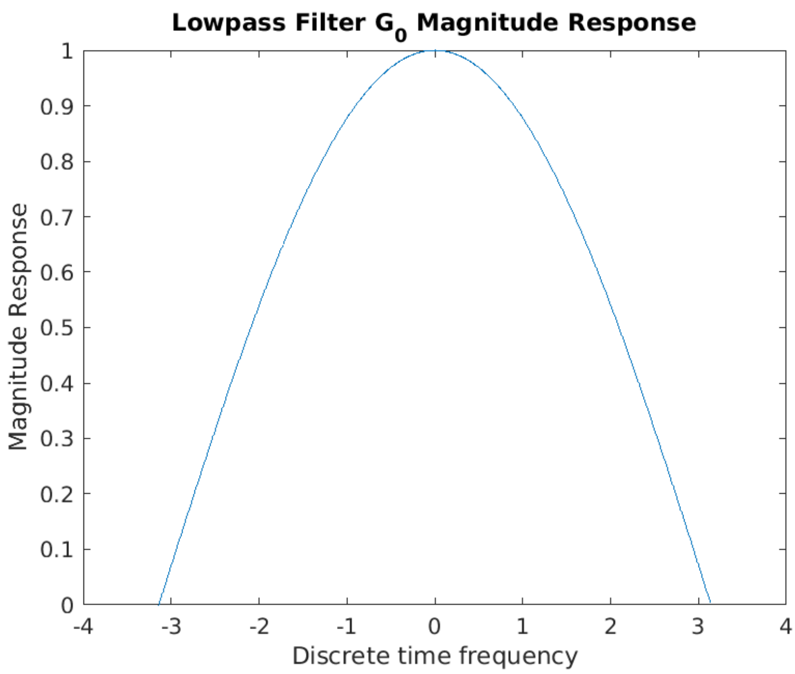

Figure 1.5: Magnitude of the frequency response for G_0

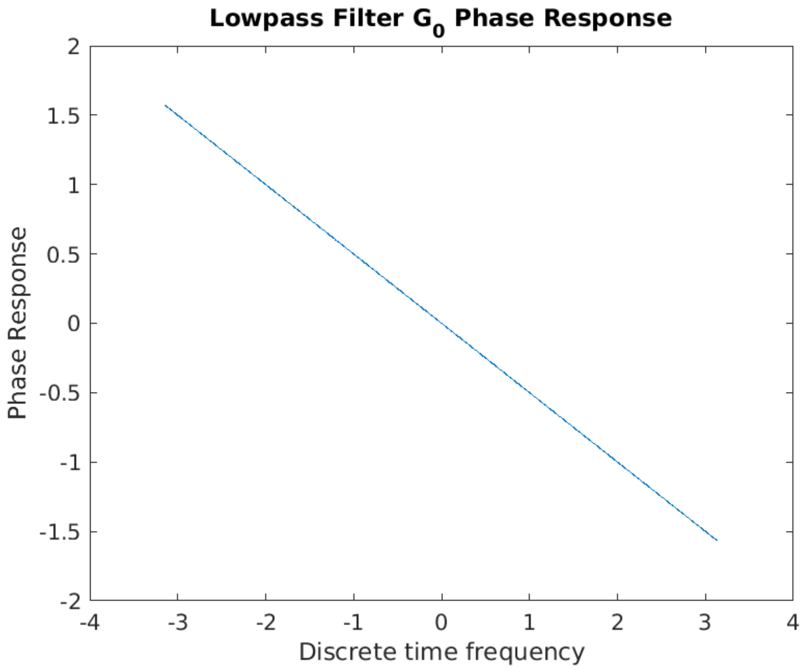

Figure 1.6: Phase of the frequency response for G_0

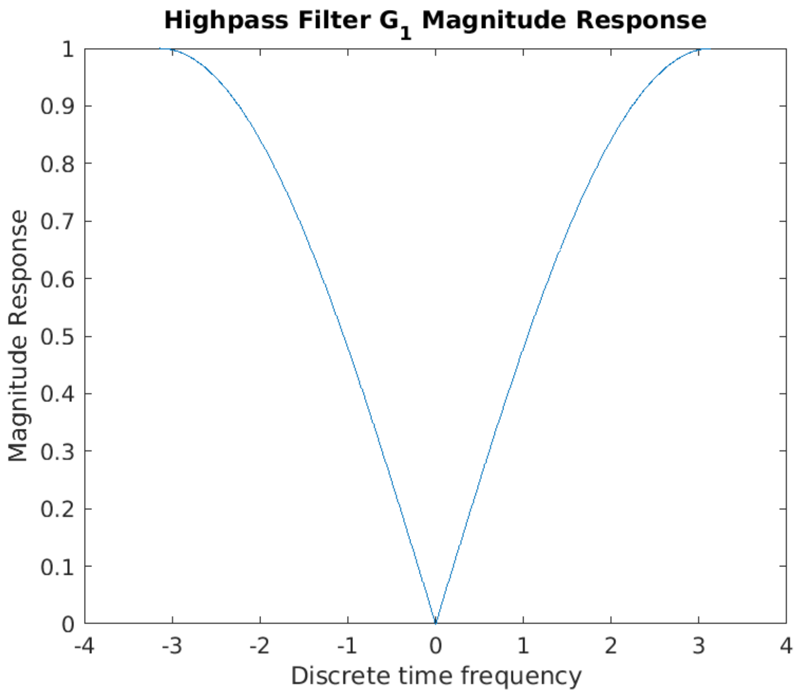

Figure 1.7: Magnitude of the frequency response for G_1

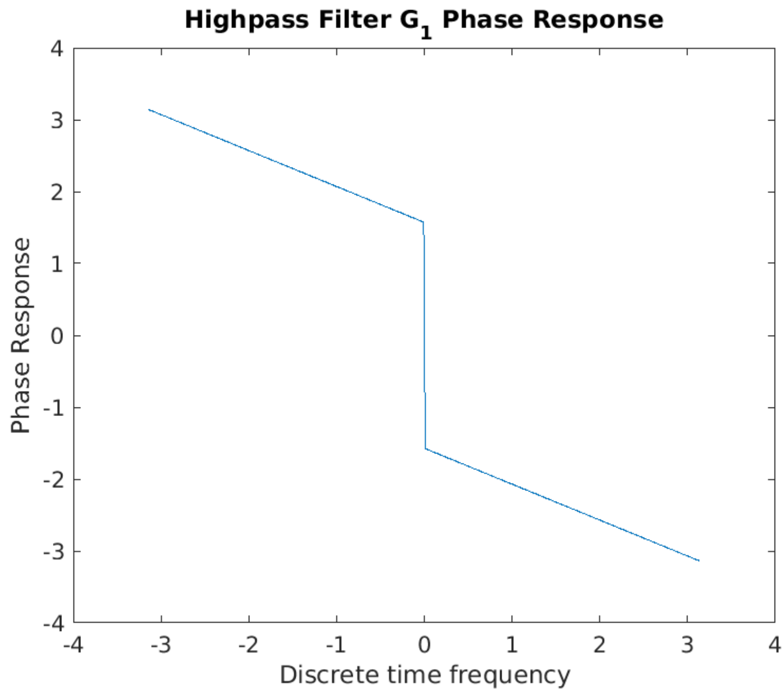

Figure 1.8: Phase of the frequency response for G_1

1.2 Task c: Create a Filter Bank

The code for the filter bank is listed in branchResponses.m of Appendix A.

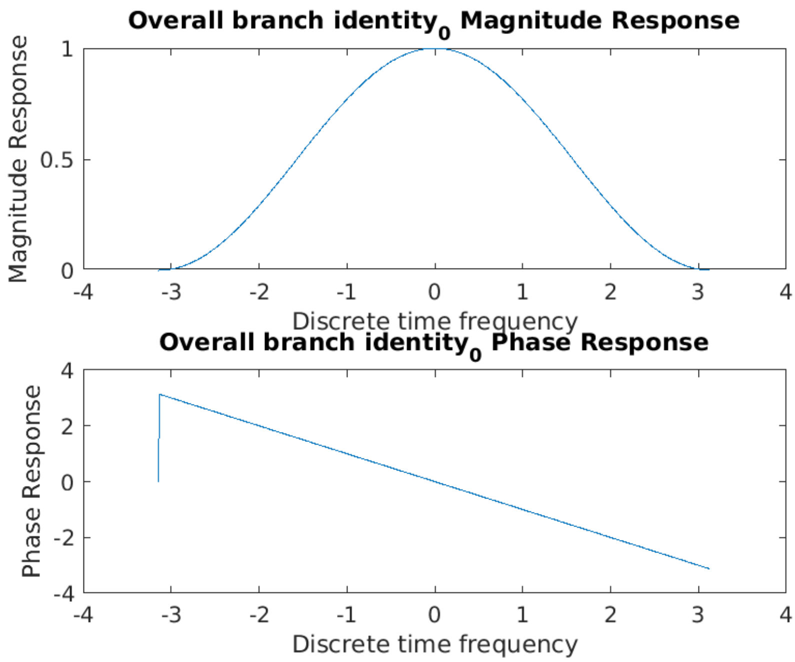

Figure 1.9: The frequency response of the first branch of the parallel filters

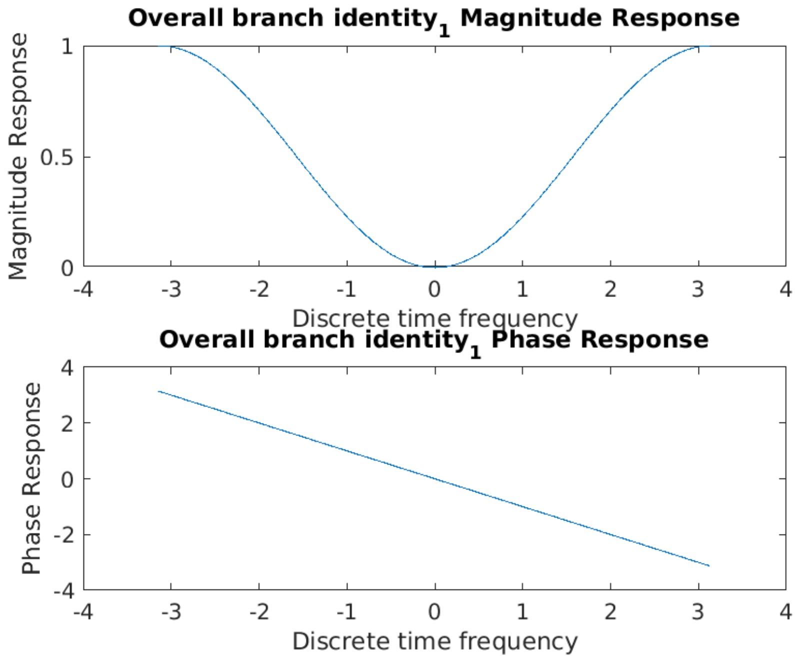

Figure 1.10: The frequency response of the second branch of the parallel filters

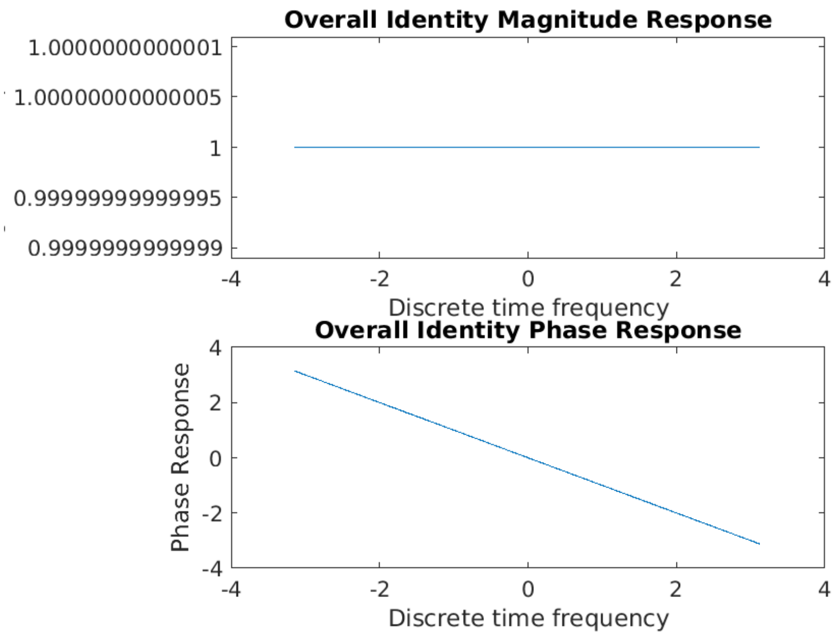

Figure 1.11: The frequency response of the overall filter. Note that the input would be recovered perfectly (with a time delay) with this implementation

1.3 Task d: Test the Filter Bank

The filter bank was tested by generating 10 sinusoidal inputs, all at different frequencies, and observing the filter output. The details of generating the output can be seen in the code, the main thing to note is that the output was generated using the difference equation of the filter.

A note about delay aligning:

There is a MATLAB function called **finddelay()** that is supposed to find the delay between the input and output signals of a system. I found that inspecting the graphs of the output was a simpler, more accurate way to find the time delay for the case of a single sinusoidal input. The phase of these filters is close to linear, so the input signal and output signal resemble each other quite well (aside from the magnitude change). It should be possible to calculate the time delay because the phase is roughly linear. However, I stuck with my simple method of inspecting the graphs visually. Delays are listed in the below table:

input signal DT frequency ω_0	lowpass filter delay	highpass filter delay
$\frac{\pi}{10}$	1	15
$\frac{3\pi}{10}$	6	1
$\frac{6\pi}{10}$	4	3
π	0	0

Table 1.1: output delay for the input signal x = $cos(k\omega_0 n)$ where $\omega_0 = \frac{\pi}{10}$ and k ranges from 1 to 10.

A note about transients:

My method for determining the length of the transients was the same as my method for finding the time delay between input and output. I inspected the graphs visually and settled on a value of 40 samples to be safe.

input signal DT freq	lowpass SNR	highpass SNR
$\frac{\pi}{10}$	36.9427	-38.2942
$\frac{3\pi}{10}$	0.0207	-68.2517
$\frac{6\pi}{10}$	-70.57	-1.9453
π	0	inf

Table 1.2: test results for the input signal x = $cos(k\omega_0 n)$ where $\omega_0 = \frac{\pi}{10}$ and k ranges from 1 to 10. The -inf result is due to the MSE value for cos(pi) being zero. This is dues to the filter having a passband gain of 1 at DT frequencies of π

The SNR of 0 that occurs for input frequencies of pi for the lowpass filter, despite perfect attenuation to 0, is due to the MSE being the same as the square of the ideal value. I chose 0.05*cos(w n) as the ideal value because there would be a 0/0 error otherwise. Instead of 10log(0/0), I end up with 10log(1/1) thus an SNR of 0.

All of the time responses were plotted. Note that for the Haar filter the highpass filter passes all frequencies above $\frac{5\pi}{10}$ and the lowpass filter passes all frequencies below to $\frac{5\pi}{10}$, and both filters attenuate the input signal to approximately 1/2 of its input amplitude at an input frequency of $\frac{5\pi}{10}$.

The Attenuation of the filters is not overly helpful, however. The lowpass filter attenuates signals of frequency $\frac{8\pi}{10}$ and greater by more that 1/2, and the highpass filter attenuates signals of frequency $\frac{2\pi}{10}$ by more than 1/2.

The lowpass and highpass branches' outputs can't simply be taken to be the high and low frequency portions of a given input signal because there would be some content from a signal that is not supposed to be present in the output, present in the output.

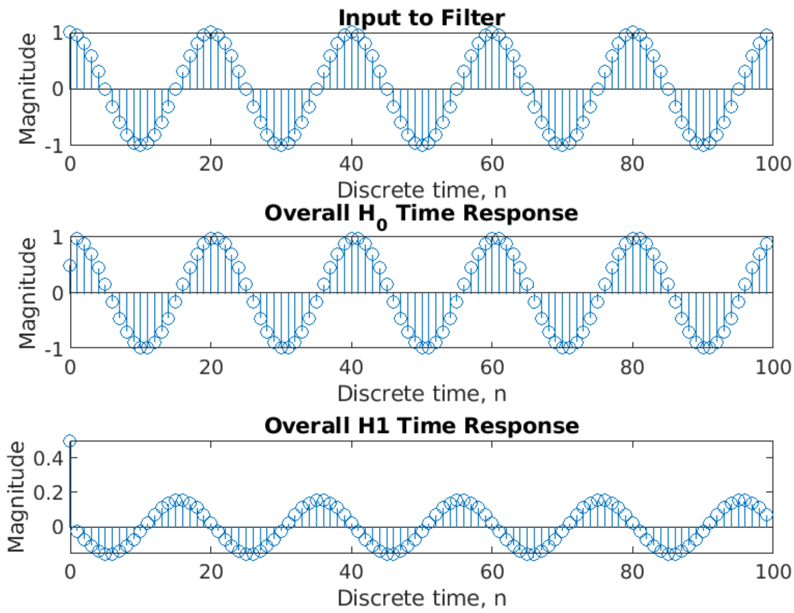

Figure 1.12: The time response of the filters , input frequency is $\pi/10$

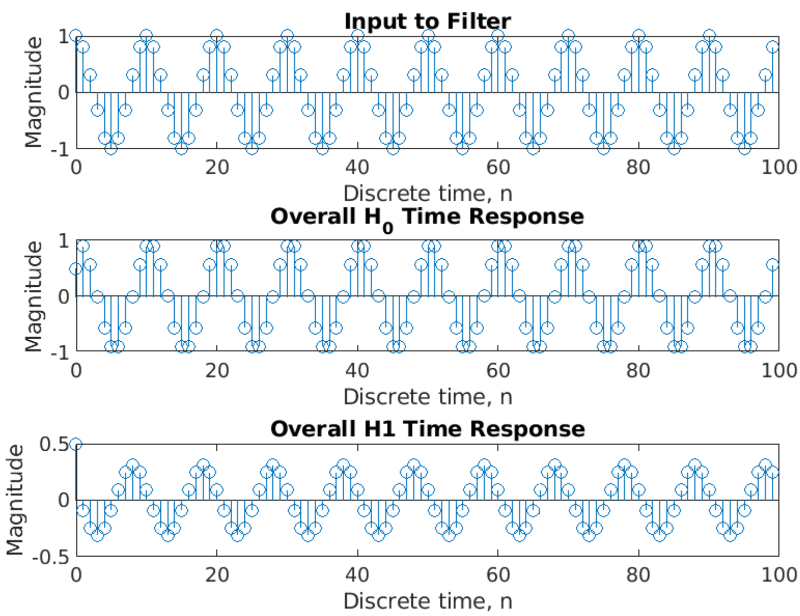

Figure 1.13: The time response of the filters, input frequency is $2\pi/10$

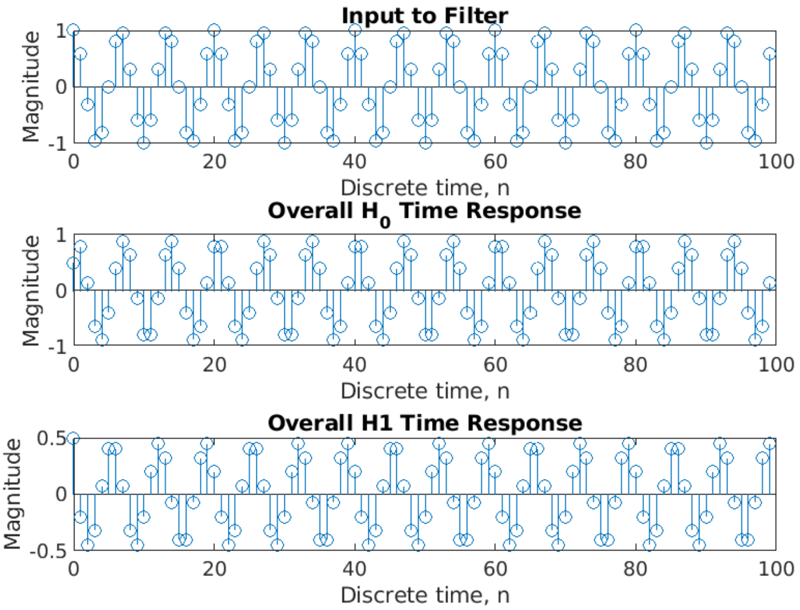

Figure 1.14: The time response of the filters, input frequency is $3\pi/10$

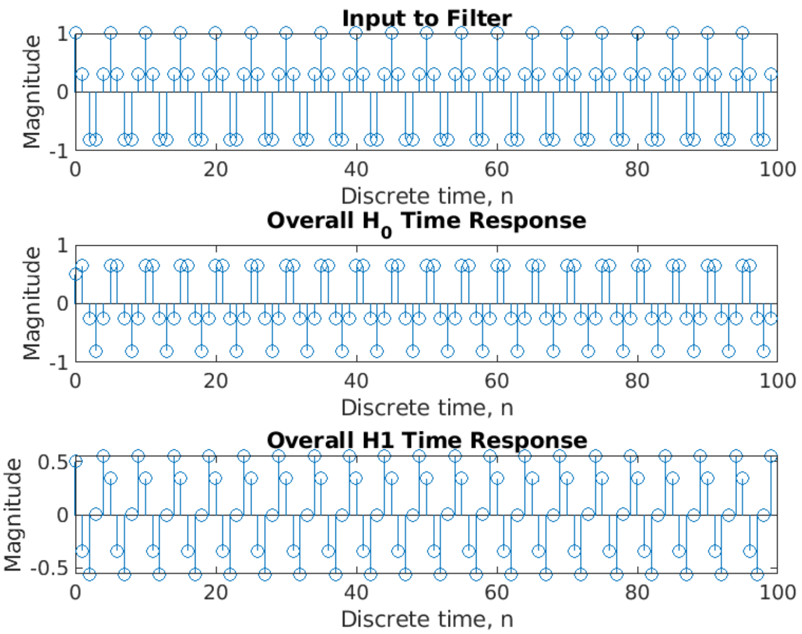

Figure 1.15: The time response of the filters, input frequency is $4\pi/10$

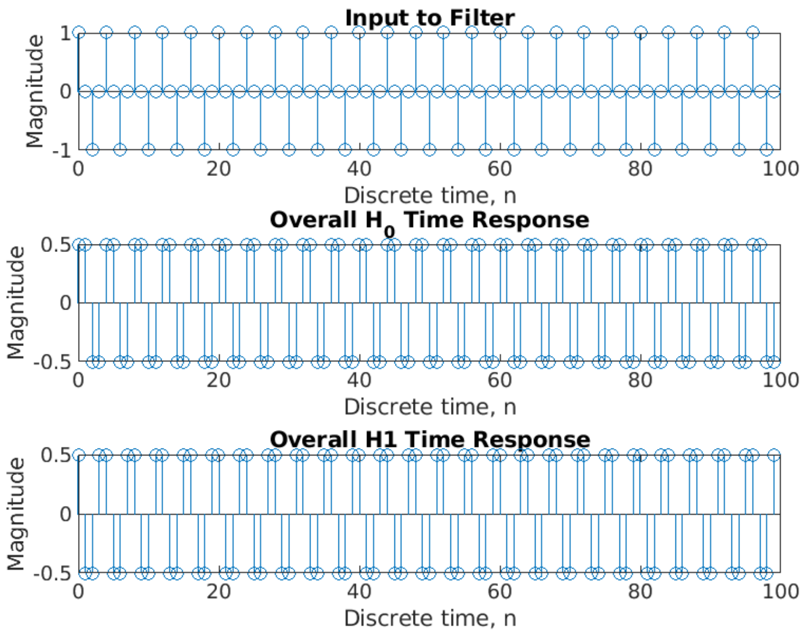

Figure 1.16: The time response of the filters, input frequency is $5\pi/10$

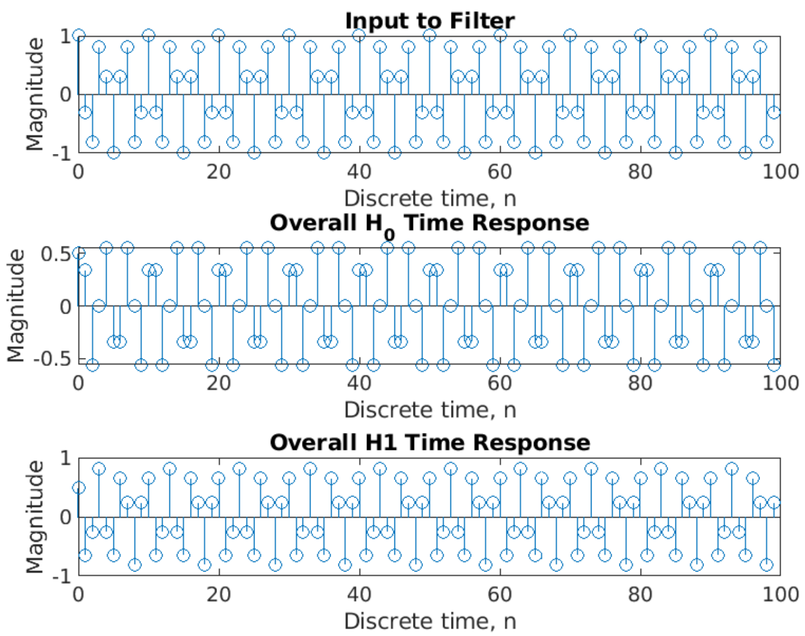

Figure 1.17: The time response of the filters, input frequency is $6\pi/10$

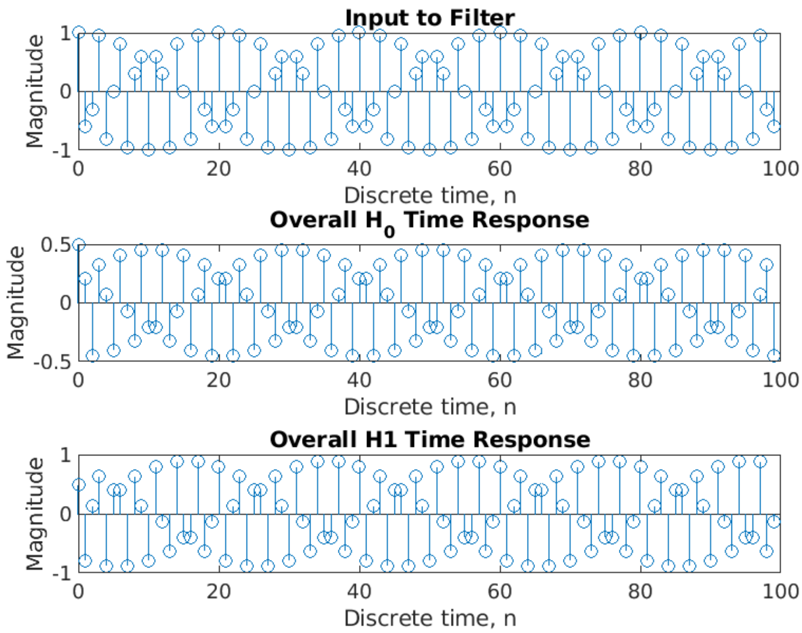

Figure 1.18: The time response of the filters, input frequency is $7\pi/10$

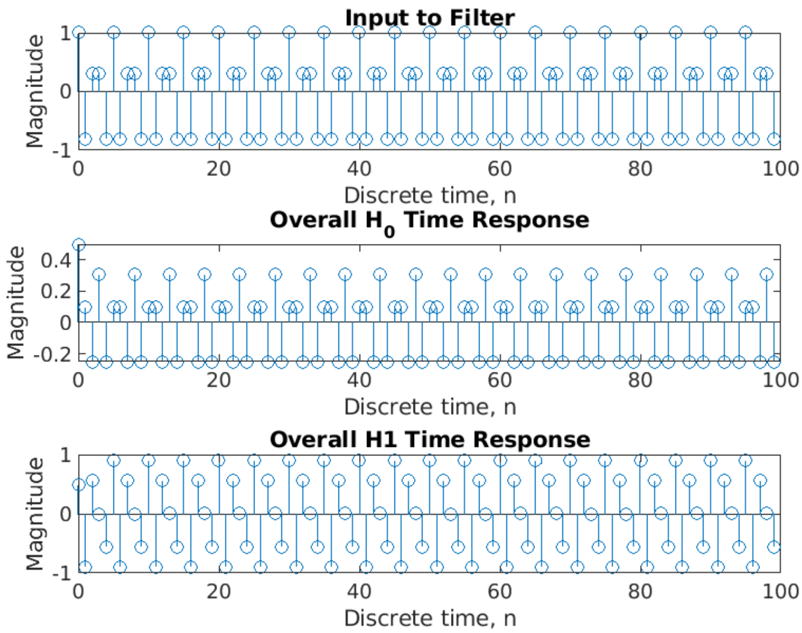

Figure 1.19: The time response of the filters, input frequency is $8\pi/10$

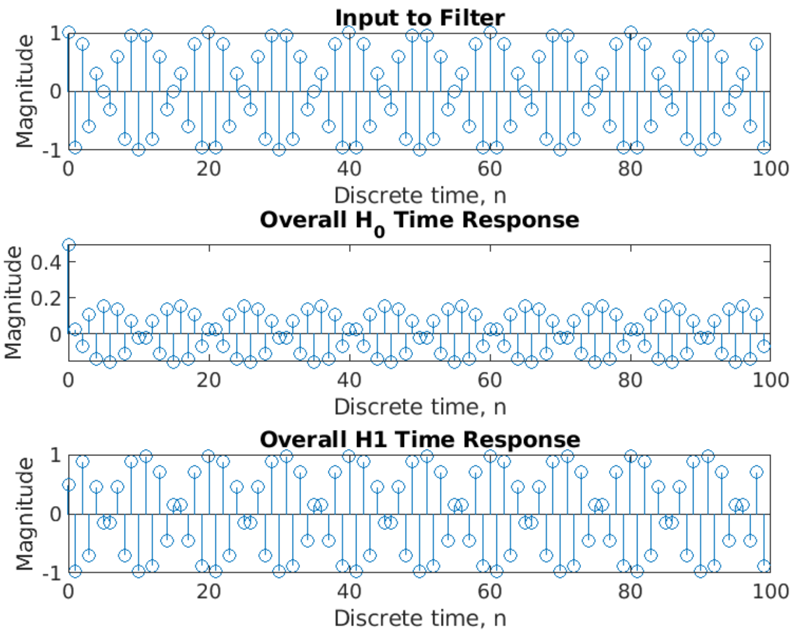

Figure 1.20: The time response of the filters, input frequency is $9\pi/10$

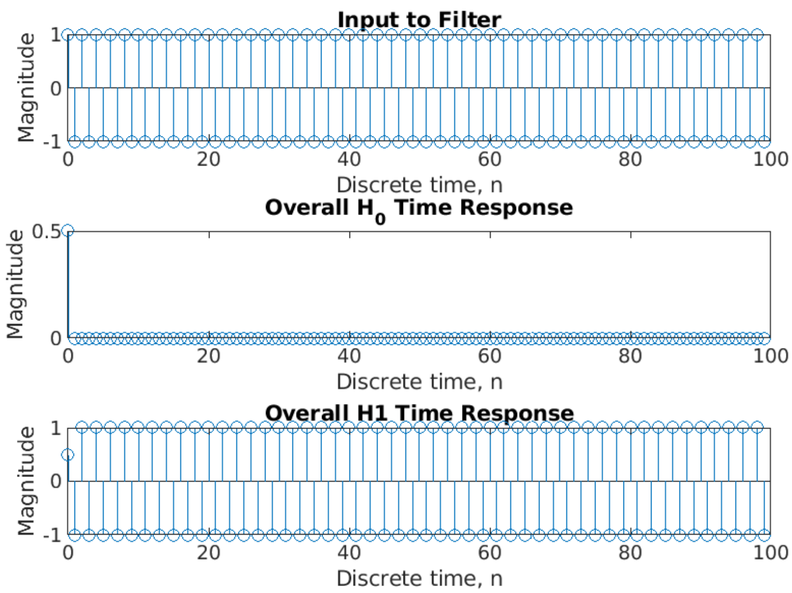

Figure 1.21: The time response of the filters, input frequency is $10\pi/10$

2 SCHEME 2

2.1 Task a

The calculations for the butterworth filter are in Appendix C.
MATLAB code for task A is in butterworthScheme2.m of Appendix A:

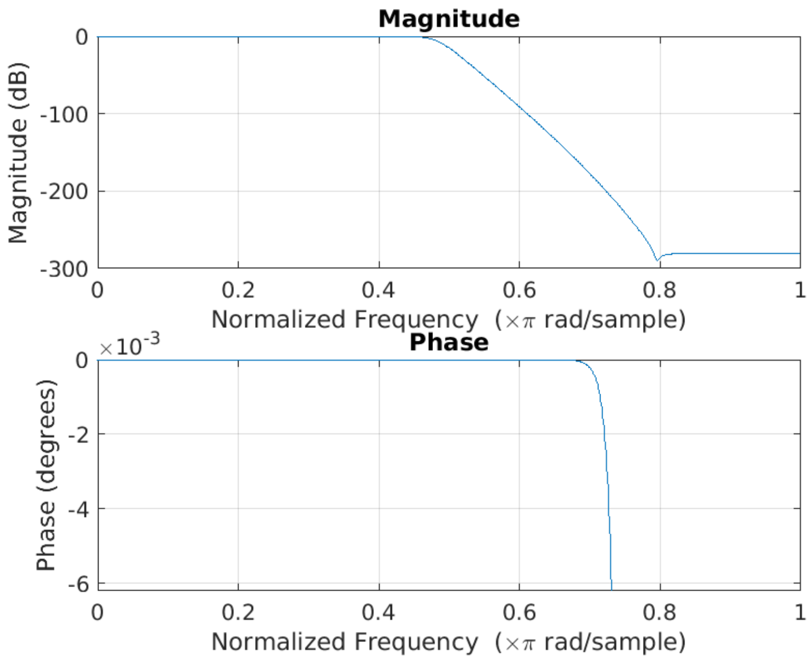

Figure 2.1: The frequency response for the magnitude squared function, as a sanity check before attempting to generate the butterworth filters

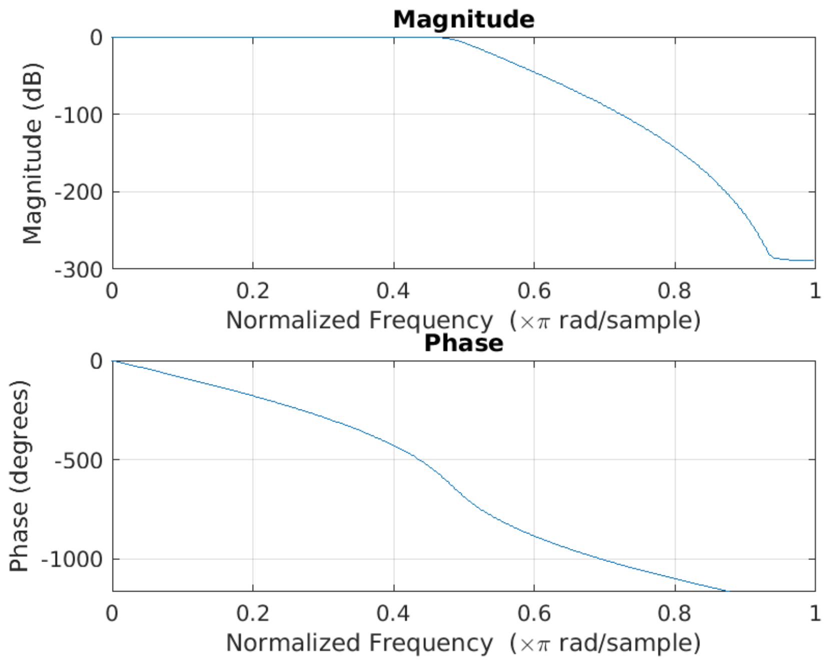

Figure 2.2: The frequency response for H_0, lowpass Butterworth filter

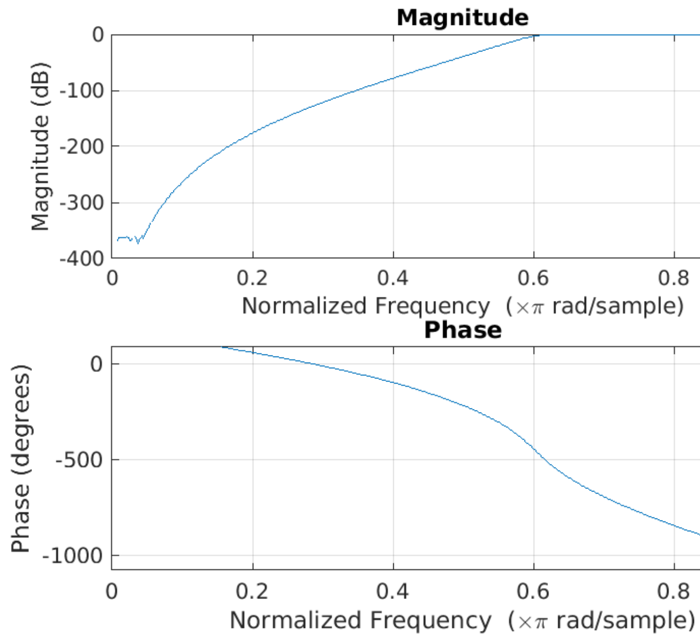

Figure 2.3: The frequency response for H_1, highpass Butterworth filter

2.2 Task b: Test the Filter Bank

The filter bank was tested by generating 10 sinusoidal inputs, all at different frequencies, and observing the filter output. The details of generating the output can be seen in the code, the main thing to note is that the output was generated using the difference equation of the filter.

A note about delay aligning:

There is a MATLAB function called **finddelay()** that is supposed to find the delay between the input and output signals of a system. I found that inspecting the graphs of the output was a simpler, more accurate way to find the time delay for the case of a single sinusoidal input. The phase of these filters is close to linear, so the input signal and output signal resemble each other quite well (aside from the magnitude change). It should be possible to calculate the time delay because the phase is roughly linear. However, I stuck with my simple method of inspecting the graphs visually. Delays are listed in the below table:

input signal DT frequency ω_0	lowpass filter delay	highpass filter delay
$\frac{\pi}{10}$	5	5
$\frac{3\pi}{10}$	12	0
$\frac{6\pi}{10}$	2	4
π	0	0

Table 2.1: output delay for the input signal x = $cos(k\omega_0 n)$ where $\omega_0 = \frac{\pi}{10}$ and k ranges from 1 to 10.

A note about transients:

My method for determining the length of the transients was the same as my method for finding the time delay between input and output. I inspected the graphs visually and settled on a value of 1 sample, which makes sense because this is a FIR filter with a simple difference equation of length 1.

input signal DT freq	lowpass SNR	highpass SNR
$\frac{\pi}{10}$	53.0363	-inf
$\frac{3\pi}{10}$	67.5711	-23.0386
$\frac{6\pi}{10}$	-21.9329	0.0177
π	-23.088	133.6657

Table 2.2: test results for the input signal x = $cos(k\omega_0 n)$ where $\omega_0 = \frac{\pi}{10}$ and k ranges from 1 to 10. The highpass filter attenuates low frequencies well and passes high frequencies, the lowpass filter attenuates high frequencies well and passes low frequencies

All of the time responses were plotted. Note that for the butterworth filter, there are 2 discrete time frequencies where the transition from signals that classified as low frequency pass over to being classified as high frequency signals. These frequencies are:

- $\frac{4\pi}{10}$

- $\frac{5\pi}{10}$

This observation is obvious from figures 2.7 and 2.8. In both cases, the input signal is only attenuated to approximately 1/2 of its input amplitude by one of the 2 filters. In figure 2.7, the $\frac{4\pi}{10}$ signal is not fully attenuated by the lowpass filter, however the highpass filter does attenuate it. In figure 2.8, the $\frac{5\pi}{10}$ signal is not fully attenuated by the highpass filter, however the lowpass filter does attenuate it.

If this filter were used to segment signals, one way to deal with this would be to classify the signals based on if **either** of the filters attenuates them significantly.

For instance in the case of $\frac{4\pi}{10}$, that frequency would be classified as low frequency because it is attenuated almost entirely by the highpass filter.

The opposite case would be true for $\frac{5\pi}{10}$. It would be a high frequency component by the same logic.

As far as using the output of either branch goes, the following are some relevant notes:
The lowpass and highpass branches' outputs can simply be taken to be the high and low frequency portions of a given input signal because there would only be content from a signal that is lower than $\frac{6\pi}{10}$ present in the lowpass output, and only content from a signal that is higher than $\frac{5\pi}{10}$ present in the highpass output. These observations are clear from the plots 2.4 to 2.13.

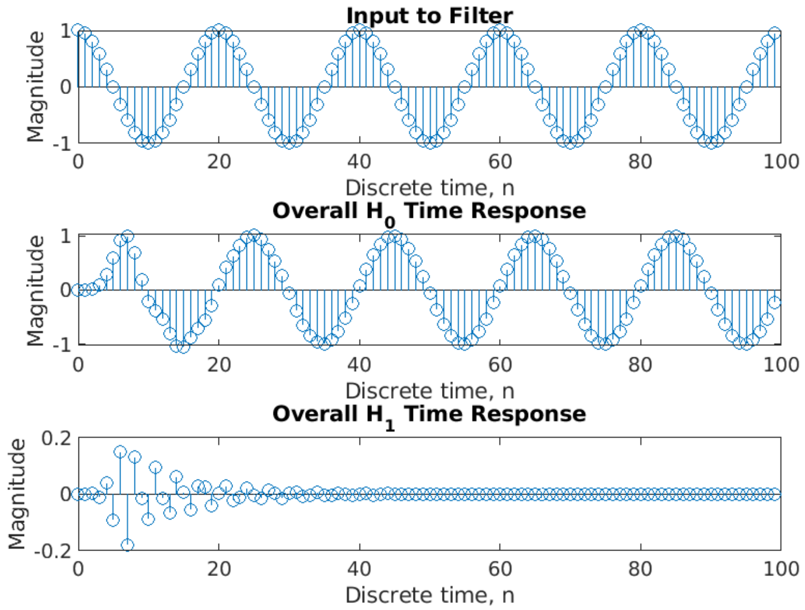

Figure 2.4: The time response of the filters , input frequency is $\pi/10$

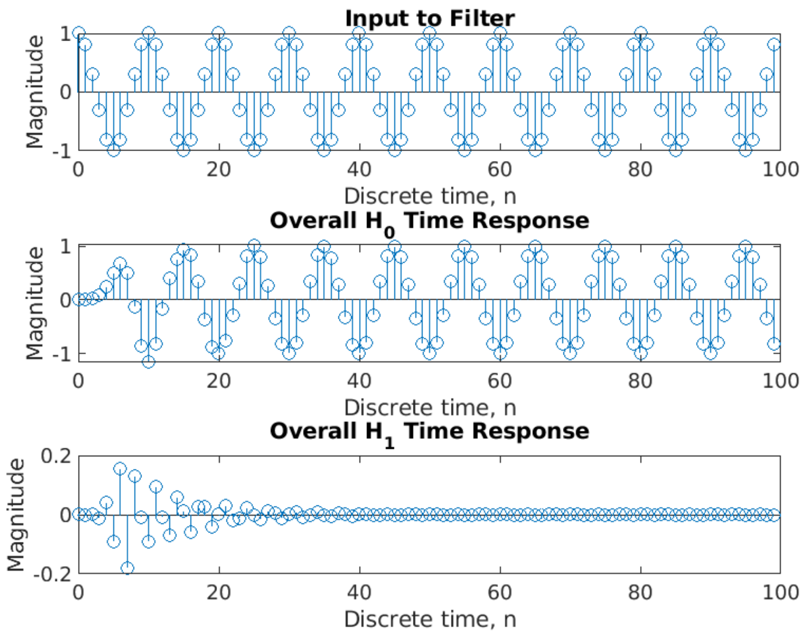

Figure 2.5: The time response of the filters, input frequency is $2\pi/10$

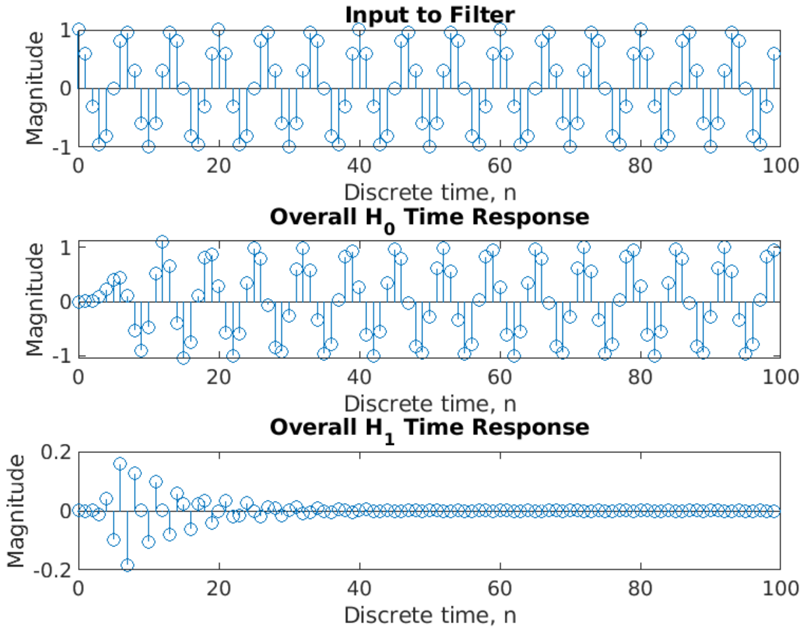

Figure 2.6: The time response of the filters, input frequency is $3\pi/10$

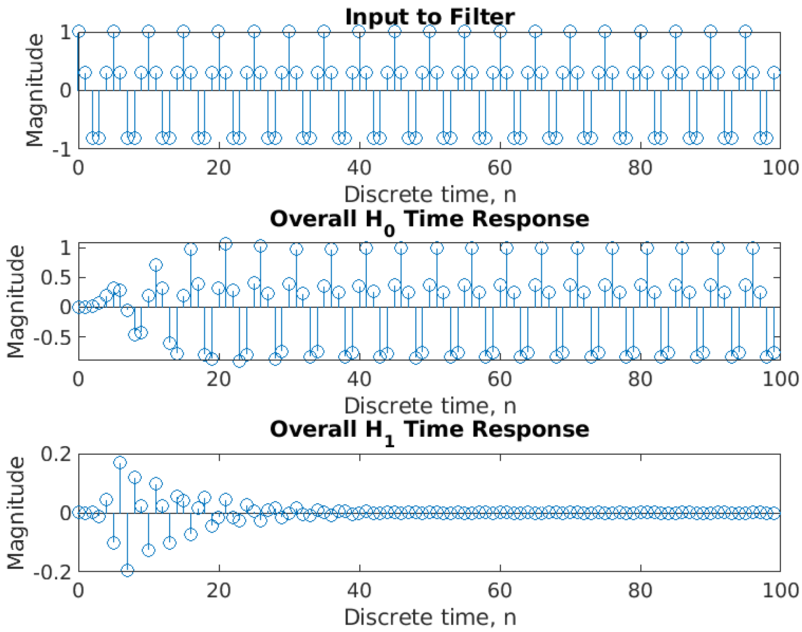

Figure 2.7: The time response of the filters, input frequency is $4\pi/10$

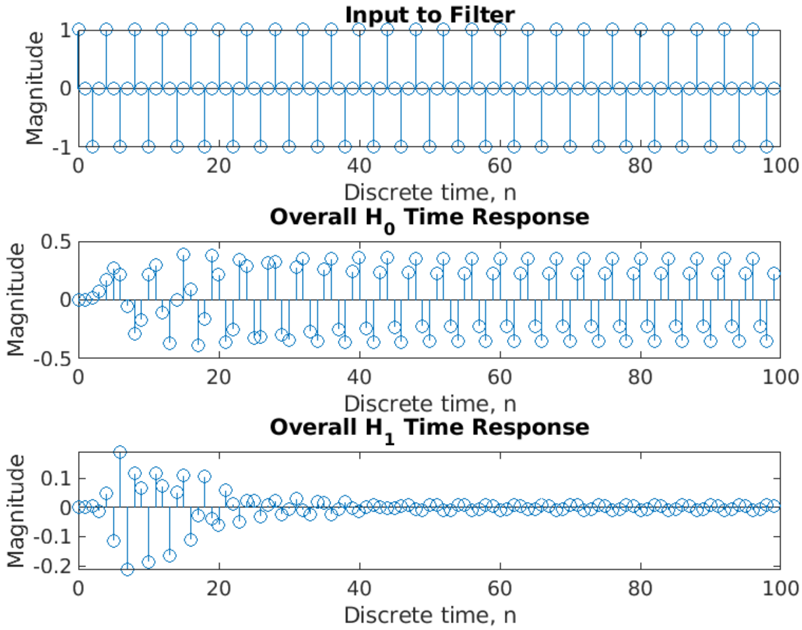

Figure 2.8: The time response of the filters, input frequency is $5\pi/10$

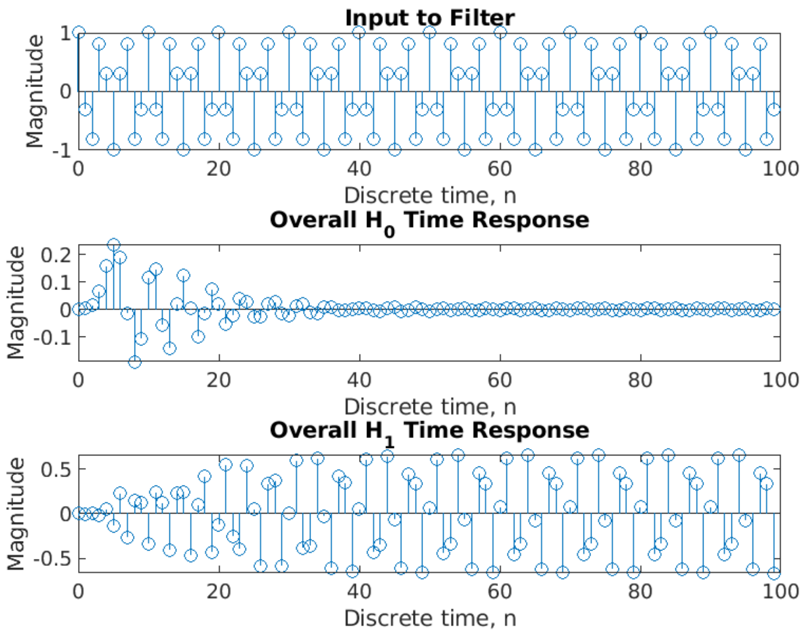

Figure 2.9: The time response of the filters, input frequency is $6\pi/10$

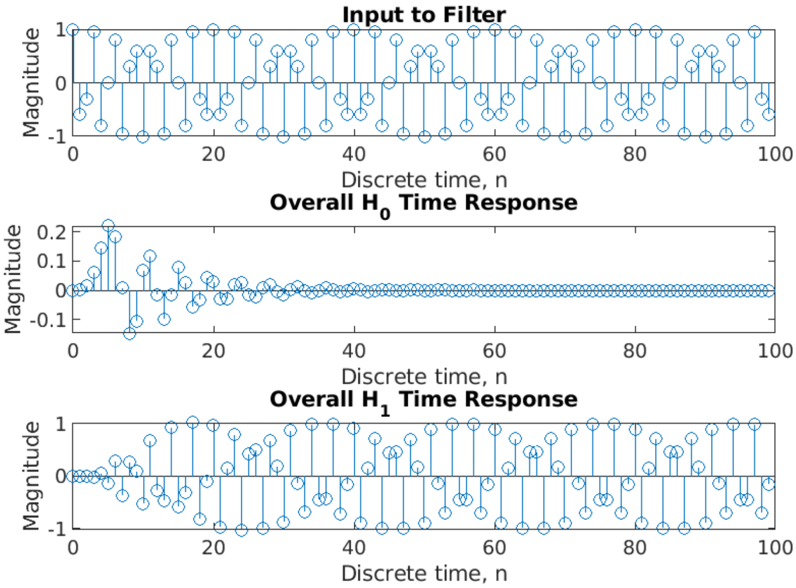

Figure 2.10: The time response of the filters, input frequency is $7\pi/10$

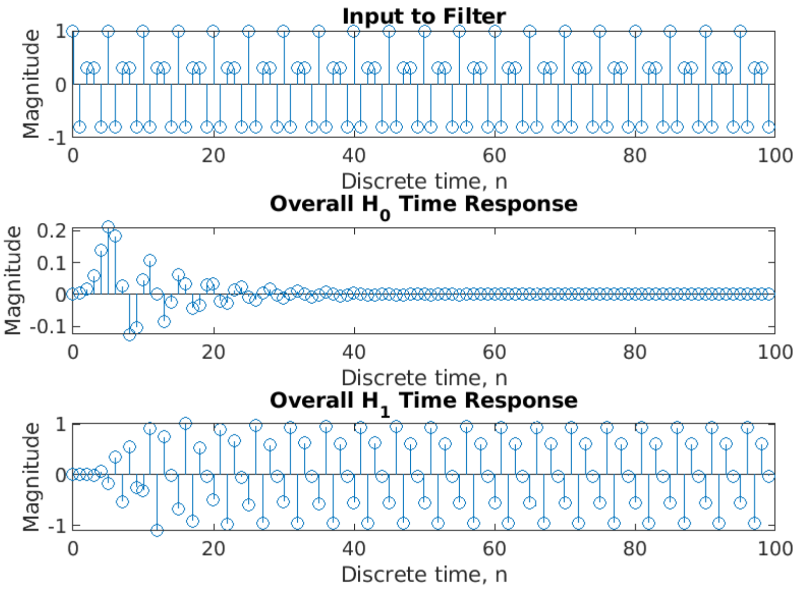

Figure 2.11: The time response of the filters, input frequency is $8\pi/10$

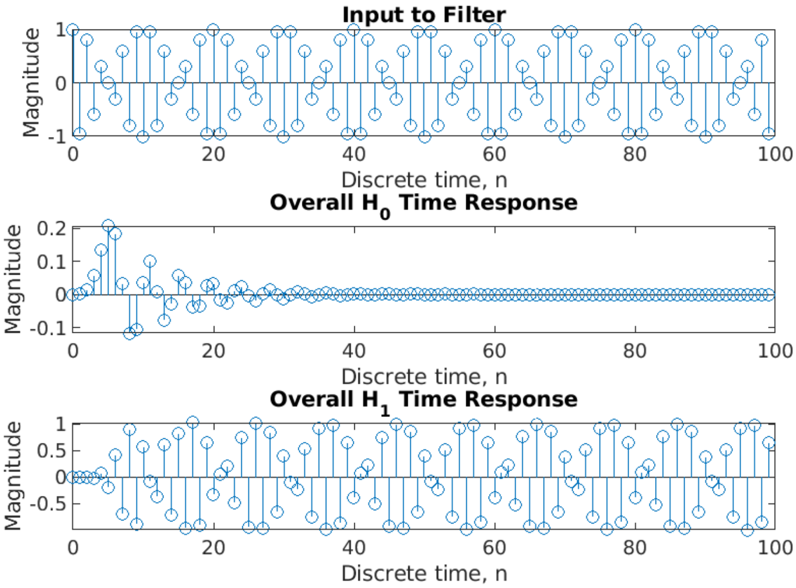

Figure 2.12: The time response of the filters, input frequency is $9\pi/10$

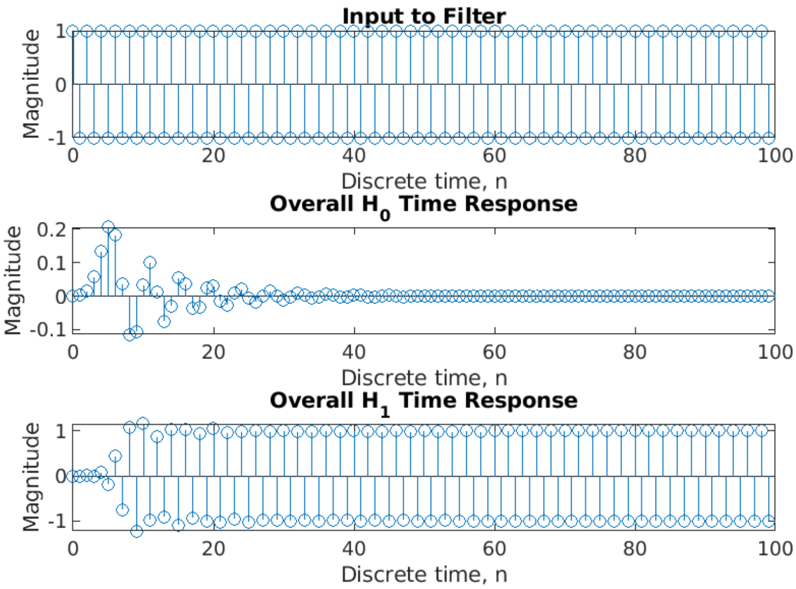

Figure 2.13: The time response of the filters, input frequency is $10\pi/10$

2.3 Task c: comments on designs

The butterworth filters of scheme 2 have been shown to be superior to the Haar filters of scheme 1 when it comes to filtering out all of the content of low or high frequencies. This is the objective of the filters, therefore scheme 2 is superior. One nice feature of the Haar filters that is noted in the assignment is that the input can be recovered from the 2 branches, perfectly. This is not the objective however and therefore carries no weight for the desired application.

3 Appendix A: Main Matlab Scripts

3.1 a3scheme1ResponsePlots.m

```
z = tf('z',1/(2*pi*10000));

H\textunderscore 0 = 1/2*(1+z^(-1));
H\textunderscore 1 = 1/2*(1-z^(-1));

frequencies=(-pi:(2*pi/1000):pi);
frequencies=frequencies(1:1000);
frequencies2=exp(j.*frequencies);
H\textunderscore 0\textunderscore resp  = freqresp(H\textunderscore 0, frequencies2);
H\textunderscore 1\textunderscore resp  = freqresp(H\textunderscore 1, frequencies2);

fignum=1;

plotResp(1, fignum, 1, 1, abs(H\textunderscore 0\textunderscore resp), frequencies, [1000], ...
[Lowpass Filter H_0 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

angle\textunderscore of\textunderscore signal = angle(H\textunderscore 0\textunderscore resp);

plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal, frequencies, [1000], ...
[Lowpass Filter H_0 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);



plotResp(1, fignum+2, 1, 1, abs(H\textunderscore 1\textunderscore resp), frequencies, [1000], ...
[Highpass Filter H_1 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

angle\textunderscore of\textunderscore signal = angle(H\textunderscore 1\textunderscore resp);

plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal, frequencies, [1000], ...
```

```
[Highpass Filter H_1 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);
```

3.2 a3scheme1GRespPlots.m

```
z = tf('z',1/(2*pi*10000));

G\textunderscore 0 = 1/2*(1+z^(-1));
G\textunderscore 1 = 1/2*(-1+z^(-1));

frequencies=(-pi:(2*pi/1000):pi);
frequencies=frequencies(1:1000);
frequencies2=exp(j.*frequencies);
G\textunderscore 0\textunderscore resp  = freqresp(G\textunderscore 0,
    frequencies2);
G\textunderscore 1\textunderscore resp  = freqresp(G\textunderscore 1,
    frequencies2);

fignum=1;

plotResp(1, fignum, 1, 1, abs(G\textunderscore 0\textunderscore resp),
    frequencies, [1000], ...
[Lowpass Filter G_0 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

angle\textunderscore of\textunderscore signal = angle(G\textunderscore 0\
    textunderscore resp);

plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
[Lowpass Filter G_0 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);



plotResp(1, fignum+2, 1, 1, abs(G\textunderscore 1\textunderscore resp),
    frequencies, [1000], ...
[Highpass Filter G_1 Magnitude Response], ...
[Magnitude Response ], ...
[Discrete time frequency],0);

angle\textunderscore of\textunderscore signal = angle(G\textunderscore 1\
    textunderscore resp);

plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
```

```
[Highpass Filter G_1 Phase Response], ...
[Phase Response], ...
[Discrete time frequency],0);
```

3.3 branchResponses.m

```
z = tf('z',1/(2*pi*10000));

H\textunderscore 0 = 1/2*(1+z^(-1));
H\textunderscore 1 = 1/2*(1-z^(-1));

G\textunderscore 0 = 1/2*(1+z^(-1));
G\textunderscore 1 = 1/2*(-1+z^(-1));

identity\textunderscore 0 = H\textunderscore 0*G\textunderscore 0;
identity\textunderscore 1 = H\textunderscore 1*G\textunderscore 1;

frequencies=(-pi:(2*pi/1000):pi);
frequencies=frequencies(1:1000);
frequencies2=exp(j.*frequencies);
identity\textunderscore 0\textunderscore resp = freqresp(identity\textunderscore 0, frequencies2);
identity\textunderscore 1\textunderscore resp = freqresp(identity\textunderscore 1, frequencies2);

fignum=1;
mag1=abs(identity\textunderscore 1\textunderscore resp);
phase1=angle(identity\textunderscore 1\textunderscore resp);
signals1=[transpose(mag1(:)); transpose(phase1(:))];
plotResp(2, fignum, 2, 1, signals1, [frequencies; frequencies], [1000 1000], ...
[Overall branch identity_1 Magnitude Response, Overall branch identity_1 Phase Response], ...
[Magnitude Response, Phase Response], ...
[Discrete time frequency,Discrete time frequency],0);

fignum=2;
mag0=abs(identity\textunderscore 0\textunderscore resp);
phase0=angle(identity\textunderscore 0\textunderscore resp);
signals0=[transpose(mag0(:)); transpose(phase0(:))];
plotResp(2, fignum, 2, 1, signals0, [frequencies; frequencies], [1000 1000], ...
[Overall branch identity_0 Magnitude Response, Overall branch identity_0 Phase Response], ...
[Magnitude Response, Phase Response], ...
[Discrete time frequency,Discrete time frequency],0);

fignum=3;
```

```
mag=abs(identity\textunderscore 1\textunderscore resp+identity\textunderscore 0\textunderscore resp);
phase=angle(identity\textunderscore 1\textunderscore resp+identity\textunderscore 0\textunderscore resp);
signals=[transpose(mag(:)); transpose(phase(:))];
plotResp(2, fignum, 2, 1, signals, [frequencies; frequencies], [1000 1000], ...
[Overall Identity Magnitude Response, Overall Identity Phase Response], ...
[Magnitude Response, Phase Response], ...
[Discrete time frequency,Discrete time frequency],0);
```

3.4 butterworthScheme2.m

```
a=[(1/1.89077219)^28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1];
b=[1];
[resids, poles, k]=residue(b,a);

sys1=tf(b,a);
freqs(b,a,(-pi:(2*pi/1000):pi))
[numd,dend]=bilinear(cell2mat(sys1.Numerator), cell2mat(sys1.Denominator),1);
figure(1);
freqz(numd,dend,1000);

%find all LHP poles of CT magnitude squared function
butterPoles=zeros(1,14);
a=zeros(1,14);
b=zeros(1,14);
for i=1:7
    angle=pi+pi/28+(i-1)*pi/14;
    a(i)=1.89077219*(cos(angle));
    a(i+7)=1.89077219*(cos(angle));
    b(i)=1.89077219*sin(angle);
    b(i+7)=-1.89077219*sin(angle);
end
butterPoles=complex(a,b);

poles=transpose(butterPoles);

denominator=poly(butterPoles);
numerator=[1.89077219^14];

[numerator\textunderscore d\textunderscore lp,denominator\textunderscore d\textunderscore lp]=bilinear(numerator,denominator,1);

figure(2)
freqz(numerator\textunderscore d\textunderscore lp, denominator\textunderscore d\textunderscore lp);

%highpass butterworth filter

```

```

[numerator\textunderscore d\textunderscore hp, denominator\textunderscore d\textunderscore hp]=butter(14,1.89077219/pi,'high');

figure(3)
freqz(numerator\textunderscore d\textunderscore hp, denominator\textunderscore d\textunderscore hp);
```

3.5 testHaarFilters.m

```
base\textunderscore f = pi/10;
matlabIndexOffset=1;
z = tf('z',1/(2*pi*10000));

H\textunderscore 0 = 1/2*(1+z^(-1));
H\textunderscore 1 = 1/2*(1-z^(-1));

n=[0:1:99]

h\textunderscore 0\textunderscore outputs=zeros(10,100);
h\textunderscore 1\textunderscore outputs=zeros(10,100);
input\textunderscore signals=zeros(10,100);

for i=1:10
    % baseband of discrete time fourier transform of cos(i*base\textunderscore f*n) is pi
    % occuring at +/-omega\textunderscore nought. entirely real, phase is 0 degrees

    % from tables, mthe z transform of cos(i*base\textunderscore f*n) is :
    %   cos(w\textunderscore 0*n)u(n)=(1-cos(w\textunderscore 0)z^(-1))/(1-2cos(w\textunderscore 0)z^(-1)+z^(-2))
    w\textunderscore 0=i*base\textunderscore f
    %input\textunderscore signal=(1-cos(w\textunderscore 0)*z^(-1))/(1-2*cos(w\textunderscore 0)*z^(-1)+z^(-2));

    % a very simple approach: find the output using the difference equation
    % from initial rest.

    input\textunderscore signal=cos(w\textunderscore 0.*n);
    output\textunderscore signal\textunderscore 0=zeros(1,100);
    output\textunderscore signal\textunderscore 1=zeros(1,100);

    output\textunderscore signal\textunderscore 0(1)=input\textunderscore signal(1)*0.5;
    output\textunderscore signal\textunderscore 1(1)=input\textunderscore signal(1)*0.5;

    for j = 2:100
```

```
        output\textunderscore signal\textunderscore 0(j)=input\textunderscore
           signal(j)*0.5+0.5*input\textunderscore signal(j-1);

        output\textunderscore signal\textunderscore 1(j)=input\textunderscore
           signal(j)*0.5-0.5*input\textunderscore signal(j-1);

    end

    h\textunderscore 0\textunderscore outputs(i,1:100)=output\textunderscore
       signal\textunderscore 0;
    h\textunderscore 1\textunderscore outputs(i,1:100)=output\textunderscore
       signal\textunderscore 1;
    input\textunderscore signals(i,1:100)=input\textunderscore signal;
end

for i = 1:10
    plotResp(3, i, 3, 1, [input\textunderscore signals(i,:); h\textunderscore
       0\textunderscore outputs(i,:); h\textunderscore 1\textunderscore
       outputs(i,:)], [n; n; n], [100 100 100], ...
    [Input to Filter, Overall H_0 Time Response, Overall H_1 Time
       Response], ...
    [Magnitude, Magnitude, Magnitude], ...
    [Discrete time, n, Discrete time, n,Discrete time, n],1);
end

%based on the output of these filters, the first sample of the output
%should be ommitted from the MSE calculation

% ideally, the h\textunderscore 0 filter attenuates all high frequencies
MSE\textunderscore lp=0
ideal\textunderscore val\textunderscore squared=0;
%lowest frequency signal is pi/10=dt frequency, it should be passed
for i = 2:100
    %the causal delay is 1, but the other problem to deal with here is
    %input signal starting from cos(0) while matlab indexing only allows
    % you to index starting from 1, thus the use of -2 in the ideal cos's
       argument.

    MSE\textunderscore lp=MSE\textunderscore lp+(cos(pi/10*(i-2))-h\
       textunderscore 0\textunderscore outputs(1,i))^2;
    ideal\textunderscore val\textunderscore squared=ideal\textunderscore val\
       textunderscore squared+(cos(pi/10*(i-2)))^2;
end

snr\textunderscore lp = 10*log(ideal\textunderscore val\textunderscore squared/
   MSE\textunderscore lp);

MSE\textunderscore lp2=0
```

```
%assume that the signal's ideal value is just the input attenuated by a factor of 0.01
ideal\textunderscore val\textunderscore squared2=0;
%highest frequency signal is pi it should be completely attenuated
for i = 2:100
    MSE\textunderscore lp2=MSE\textunderscore lp2+(0.05*cos(pi*(i-matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(10,i))^2;
    ideal\textunderscore val\textunderscore squared2=ideal\textunderscore val\textunderscore squared2+0.00025*(cos(pi*(i-matlabIndexOffset)))^2;
end

snr\textunderscore lp2 = 10*log(ideal\textunderscore val\textunderscore squared2/MSE\textunderscore lp2);

%what is the half-power bandwith of the lowpass filter?

MSE\textunderscore lp3=0
ideal\textunderscore val\textunderscore squared3=0;
delay=6
%from graphs, delay is 1 sample and transient is 1 sample
%medium frequency signal is 3pi/10=dt frequency, it should be passed
for i = 2:100
    MSE\textunderscore lp3=MSE\textunderscore lp3+(cos(3*pi/10*(i-delay-matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(3,i))^2;
    ideal\textunderscore val\textunderscore squared3=ideal\textunderscore val\textunderscore squared3+(cos(3*pi/10*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore lp3 = 10*log(ideal\textunderscore val\textunderscore squared3/MSE\textunderscore lp3);
%visually inspect delay from graph
delay=4;
MSE\textunderscore lp4=0;
ideal\textunderscore val\textunderscore squared4=0;
%medium frequency signal is 6pi/10=dt frequency, it is on the high side and should not be passed
for i = 2:100
    MSE\textunderscore lp4=MSE\textunderscore lp4+(0.05*cos(6*pi/10*(i-delay-matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(6,i))^2;
    ideal\textunderscore val\textunderscore squared4=ideal\textunderscore val\textunderscore squared4+0.00025*(cos(6*pi/10*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore lp4 = 10*log(ideal\textunderscore val\textunderscore squared4/MSE\textunderscore lp4);


%do the same for the highpass filter

```

```
%%

%based on the output of these filters, to evaluate performance, you need to
%account for the transient response of the filter

%based on the time output graphs, the max trasient length is about 40
%samples, there is virtually 0 transient response after that in all graphs

% account for the time delay of the filter in order to compare input and
% output as well

delay=finddelay(input\textunderscore signal, h\textunderscore 1\textunderscore outputs(1,:));
%inspect delay visually from graph
delay=15;
% ideally, the h\textunderscore 1 filter attenuates all high frequencies
MSE\textunderscore hp=0;
ideal\textunderscore val\textunderscore squared=0;
%lowest frequency signal is pi/10=dt frequency, it should not be passed
%assume that the signal's ideal value is just the input attenuated by a factor of 0.05
for i = 2:100
    MSE\textunderscore hp=MSE\textunderscore hp+(0.05*cos(pi/10*(i-delay-matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(1,i))^2;
    ideal\textunderscore val\textunderscore squared=ideal\textunderscore val\textunderscore squared+0.00025*(cos(pi/10*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore hp = 10*log(ideal\textunderscore val\textunderscore squared/MSE\textunderscore hp);


%inspect delay visually from graph
delay=0;

MSE\textunderscore hp2=0;
ideal\textunderscore val\textunderscore squared2=0;
%highest frequency signal is pi it should be completely passed
for i = 2:100
    MSE\textunderscore hp2=MSE\textunderscore hp2+(cos(pi*(i-delay-matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(10,i))^2;
    ideal\textunderscore val\textunderscore squared2=ideal\textunderscore val\textunderscore squared2+(cos(pi*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore hp2 = 10*log(ideal\textunderscore val\textunderscore squared2/MSE\textunderscore hp2);

%what is the half-power bandwith of the lowpass filter?
```

```
delay=finddelay(input\textunderscore signal, h\textunderscore 1\textunderscore outputs(3,:));

%inspect delay visually from graph
delay=1;

MSE\textunderscore hp3=0;
ideal\textunderscore val\textunderscore squared3=0;
%medium frequency signal is 3pi/10=dt frequency, it is on the low side and should attenuated
for i = 2:100
    MSE\textunderscore hp3=MSE\textunderscore hp3+(0.05*cos(3*pi/10*(i-delay-matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(3,i))^2;
    ideal\textunderscore val\textunderscore squared3=ideal\textunderscore val\textunderscore squared3+0.00025*(cos(3*pi/10*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore hp3 = 10*log(ideal\textunderscore val\textunderscore squared3/MSE\textunderscore hp3);

%inspect delay visually from graph
delay=3;

MSE\textunderscore hp4=0;
ideal\textunderscore val\textunderscore squared4=0;
%medium frequency signal is 6pi/10=dt frequency, it is on the high side and should be passed
for i = 2:100
    MSE\textunderscore hp4=MSE\textunderscore hp4+(cos(6*pi/10*(i-delay-matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(3,i))^2;
    ideal\textunderscore val\textunderscore squared4=ideal\textunderscore val\textunderscore squared4+(cos(6*pi/10*(i-delay-matlabIndexOffset)))^2;
end

snr\textunderscore hp4 = 10*log(ideal\textunderscore val\textunderscore squared4/MSE\textunderscore hp4);
```

3.6 testIIRfilters.m

```
n=[0:1:99]
matlab_index_offset=1;

h_0_outputs=zeros(10,100);
h_1_outputs=zeros(10,100);
input_signals=zeros(10,100);
base_f = pi/10;
for k=1:10
```

```
    % baseband of discrete time fourier transform of cos(i*base_f*n) is pi
    % occuring at +/—omega_nought. entirely real, phase is 0 degrees

    % from tables, mthe z transform of cos(i*base_f*n) is :
    %   cos(w_0*n)u(n)=(1—cos(w_0)z^(—1))/(1—2cos(w_0)z^(—1)+z^(—2))
    w_0=k*base_f
    %input_signal=(1—cos(w_0)*z^(—1))/(1—2*cos(w_0)*z^(—1)+z^(—2));

    % a very simple approach: find the output using the difference equation
    % from initial rest.

    input_signal=cos(w_0.*n);
    output_signal_0=zeros(1,100);
    output_signal_1=zeros(1,100);

    h_0_outputs(k,1:100)=ccde(input_signal, output_signal_0, numerator_d_lp,
       denominator_d_lp);
    h_1_outputs(k,1:100)=ccde(input_signal, output_signal_1, numerator_d_hp,
       denominator_d_hp);
    input_signals(k,1:100)=input_signal;
end

for i = 1:10
    plotResp(3, i, 3, 1, [input_signals(i,:); h_0_outputs(i,:); h_1_outputs(i
       ,:)], [n; n; n], [100 100 100], ...
    [Input to Filter, Overall H_0 Time Response, Overall H_1 Time Response], ...
    [Magnitude, Magnitude, Magnitude], ...
    [Discrete time, n, Discrete time, n, Discrete time, n],1);
end
%%
%based on the output of these filters, to evaluate performance, you need to
%account for the transient response of the filter

%based on the time output graphs, the max trasient length is about 40
%samples, there is virtually 0 transient response after that in all graphs

% account for the time delay of the filter in order to compare input and
% output as well

delay=finddelay(input_signal, h_0_outputs(1,:));
%inspect delay visually from graph
delay=5

% ideally, the h_0 filter attenuates all high frequencies
MSE_lp=0;
ideal_val_squared=0;
%lowest frequency signal is pi/10=dt frequency, it should be passed
for i = 40:100

```

```
    MSE_lp=MSE_lp+(cos(pi/10*(i-delay-matlab_index_offset))-h_0_outputs(1,i))^2;
    ideal_val_squared=ideal_val_squared+(cos(pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_lp = 10*log(ideal_val_squared/MSE_lp);
%inspect delay visually from graph
delay=0;

delay

MSE_lp2=0;
%assume that the signal's ideal value is just the input attenuated by a factor of 0.05
ideal_val_squared2=0;
%highest frequency signal is pi it should be completely attenuated
for i = 40:100
    MSE_lp2=MSE_lp2+(0.05*cos(pi*(i-delay-matlab_index_offset))-h_0_outputs(10,i))^2;
    ideal_val_squared2=ideal_val_squared2+0.00025*(cos(pi*(i-delay-matlab_index_offset)))^2;
end

snr_lp2 = 10*log(ideal_val_squared2/MSE_lp2);

%what is the half-power bandwith of the lowpass filter?

delay=finddelay(input_signal, h_0_outputs(3,:));
%observe and manually set time delay from graphs
delay=12

MSE_lp3=0;
ideal_val_squared3=0;
%medium frequency signal is 3pi/10=dt frequency, it should be passed
for i = 40:100
    MSE_lp3=MSE_lp3+(cos(3*pi/10*(i-delay-matlab_index_offset))-h_0_outputs(3,i))^2;
    ideal_val_squared3=ideal_val_squared3+(cos(3*pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_lp3 = 10*log(ideal_val_squared3/MSE_lp3);
%after 40 samlples observe delay (roughly, it is hard to know because
%amplitude is roughly constant at this point)
delay=2;

MSE_lp4=0;
ideal_val_squared4=0;
```

```
%medium frequency signal is 6pi/10=dt frequency, it is on the high side and should not be passed
for i = 40:100
    MSE_lp4=MSE_lp4+(0.05*cos(6*pi/10*(i-delay-matlab_index_offset))-h_0_outputs(6,i))^2;
    ideal_val_squared4=ideal_val_squared4+0.00025*(cos(6*pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_lp4 = 10*log(ideal_val_squared4/MSE_lp4);


%%


%based on the output of these filters, to evaluate performance, you need to
%account for the transient response of the filter

%based on the time output graphs, the max trasient length is about 40
%samples, there is virtually 0 transient response after that in all graphs

% account for the time delay of the filter in order to compare input and
% output as well

delay=finddelay(input_signal, h_1_outputs(1,:));
%inspect delay visually from graph
delay=5

% ideally, the h_1 filter attenuates all high frequencies
MSE_hp=0;
ideal_val_squared=0;
%lowest frequency signal is pi/10=dt frequency, it should not be passed
%assume that the signal's ideal value is just the input attenuated by a factor of 0.05
for i = 40:100
    MSE_hp=MSE_hp+(0.05*cos(pi/10*(i-delay-matlab_index_offset))-h_1_outputs(1,i))^2;
    ideal_val_squared2=ideal_val_squared2+0.00025*(cos(pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_hp = 10*log(ideal_val_squared/MSE_hp);
%inspect delay visually from graph
delay=0;

delay

MSE_hp2=0;

ideal_val_squared2=0;
```

```matlab
%highest frequency signal is pi it should be completely passed
for i = 40:100
    MSE_hp2=MSE_hp2+(cos(pi*(i-delay-matlab_index_offset))-h_1_outputs(10,i))^2;
    ideal_val_squared2=ideal_val_squared2+(cos(pi*(i-delay-matlab_index_offset)))^2;
end

snr_hp2 = 10*log(ideal_val_squared2/MSE_hp2);

%what is the half-power bandwith of the lowpass filter?

delay=finddelay(input_signal, h_1_outputs(3,:));

delay

MSE_hp3=0;
ideal_val_squared3=0;
%medium frequency signal is 3pi/10=dt frequency, it is on the low side and should attenuated
for i = 40:100
    MSE_hp3=MSE_hp3+(0.05*cos(3*pi/10*(i-delay-matlab_index_offset))-h_1_outputs(3,i))^2;
    ideal_val_squared3=ideal_val_squared3+0.00025*(cos(3*pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_hp3 = 10*log(ideal_val_squared3/MSE_hp3);


MSE_hp4=0;
ideal_val_squared4=0;
%medium frequency signal is 6pi/10=dt frequency, it is on the high side and should be passed
for i = 40:100
    MSE_hp4=MSE_hp4+(cos(6*pi/10*(i-delay-matlab_index_offset))-h_1_outputs(3,i))^2;
    ideal_val_squared4=ideal_val_squared4+(cos(6*pi/10*(i-delay-matlab_index_offset)))^2;
end

snr_hp4 = 10*log(ideal_val_squared4/MSE_hp4);
```

4 Appendix B: Auxillary Functions Called in main Scripts

4.1 plotResp.m

```
%frequency response plotting

%signals are the 2 dimensional vectors to plot

% n is the number of signals for plotting
%
% %fign is number of the figure within the script to plot

% subP1 number of total subplots

% subP2 number of rows of subplots

% signals is the array of dependent axis signals to plot

% indep_ax is the array of independent axis signals to plot

% len is the array of independent axis lengths

% titles is the array of titles for the subplots

% ylabels is the array of ylabels for the subplots

% xlabels is the arrary of xlabels for the subplots

function plotResp = plotResp(n,fign,subP1,subP2,signals,indep_ax,len,titles,
    ylabels,xlabels, stemP)
    figure(fign)
%checks
    [M,N]=size(signals);
    if length(len)==n %fails silently if badly formatted data is passed
        if n == M %fails silently if badly formatted data is passed
            for i = 1:n
                subplot(subP1, subP2, i)
                if(stemP)
                    stem(indep_ax(i:i,1:len(i)),signals(i:i,1:len(i)));
                else
                    plot(indep_ax(i:i,1:len(i)),signals(i:i,1:len(i)));
                end

                title((titles(i)))
                ylabel(ylabels(i))
                xlabel(xlabels(i))
            end
        end
    end
```

4.2 ccde.m

```
function ccde = ccde(input\textunderscore signal, output\textunderscore signal, numerator\textunderscore d, denominator\textunderscore d)

matlabIndexOffset=1;

for j = 0:(length(input\textunderscore signal)-1)

        for i=0:(length(numerator\textunderscore d)-1)
            if j-i >= 0
                output\textunderscore signal(j+matlabIndexOffset)=output\textunderscore signal(j+matlabIndexOffset)+numerator\textunderscore d(matlabIndexOffset+i)*input\textunderscore signal(j+matlabIndexOffset-i);
            else
                output\textunderscore signal(j+matlabIndexOffset)=output\textunderscore signal(j+matlabIndexOffset)+0;
            end
        end
        %start index from 1 because index zero is on the left side of
        %equality sign (its already there)
        for i=1:(length(denominator\textunderscore d)-1)
            if j-i >= 0
                output\textunderscore signal(j+matlabIndexOffset)=output\textunderscore signal(j+matlabIndexOffset)-denominator\textunderscore d(i+matlabIndexOffset)*output\textunderscore signal(j+matlabIndexOffset-i);
            else
                output\textunderscore signal(j+matlabIndexOffset)=output\textunderscore signal(j+matlabIndexOffset)+0;
            end
        end

end

ccde=output\textunderscore signal;
```

5 Appendix C: Calculations for Lowpass Butterworth Filter

Butterworth requires design specifications:

Choose the following:

$$0.95 \leq |H(e^{j\omega})| \leq 1, \quad 0 \leq \omega \leq 0.45\pi$$

$$|H(e^{j\omega})| \leq 0.05, \quad 0.55\pi \leq \omega \leq \pi$$

Continuous time equivalents

$$0.95 \leq |H(e^{j\omega})| < 1, \quad 0 \leq \Omega \leq \frac{2}{T_d}\tan\left(\frac{0.45\pi}{2}\right)$$ let $T_d = 1$ to make design easy

$$|H(e^{j\omega})| \leq 0.05, \quad \frac{2}{T_d}\tan\left(\frac{0.55\pi}{2}\right) \leq \Omega \leq \infty$$

$$|H_c(j\Omega)|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}}$$

$$\left(1 + (2\tan(0.225\pi)/\Omega_c)^{2N}\right) \leq 0.95^{-2}; \quad \left(1+(2\tan(0.275\pi)/\Omega_c)^{2N}\right) \geq 0.05^{-2}$$

$$2N\left[\log(2\tan(0.225\pi)) - \log(\Omega_c)\right] \leq \log(0.95^{-2}-1); \quad 2N\left[\log(2\tan(0.275\pi)) - \log\Omega_c\right] \geq \log(0.05^{-2}-1)$$

switch to equality to use both equations → will meet specifications still.

$$2N\left[\log(2\tan(0.225\pi)) - \log(2\tan(0.275\pi))\right] = \log(0.95^{-2}-1) - \log(0.05^{-2}-1)$$

$$2N(-0.137002199) = -3.567415491$$

$$N = 13.01955\ldots \rightarrow \text{use } N = 14$$

$$\log \Omega_c = \left(\frac{1}{28}\log(0.05^{-2}-1) - \log(2\tan(0.275\pi))\right) \times -1 \qquad \Omega_c = 1.89077219$$

$$|H_c(j\Omega)|^2 = \frac{1}{1+(\Omega/1.990772191)^{28}}$$

use MATLAB to finish the design...

- Take 10 LHP roots of $1/(1+(\Omega/1.99077219)^{28})$ to get C.T. T.F.
- perform Bilinear transform on the T.F.