

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 discrete 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 filters 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:

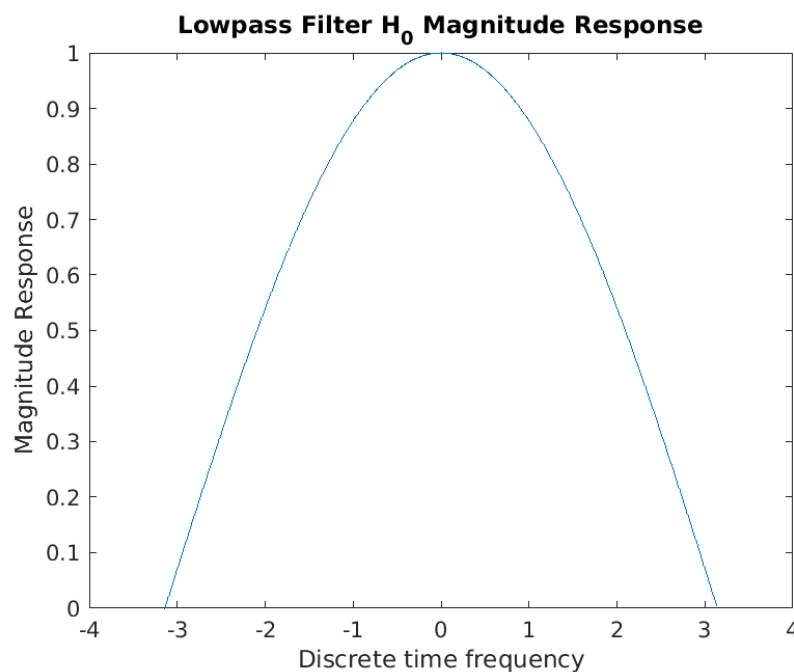


Figure 1.1: Magnitude of the frequency response for H_0

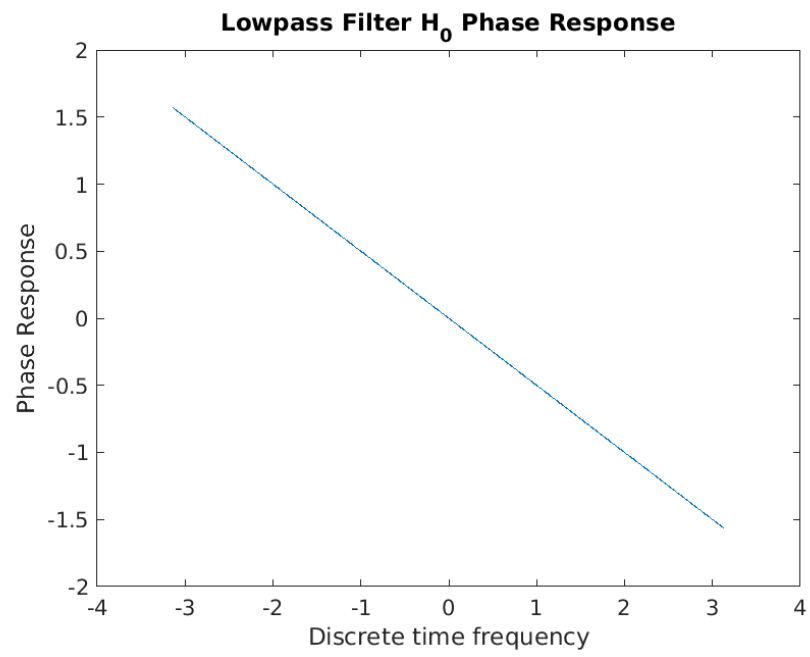


Figure 1.2: Phase of the frequency response for H_0

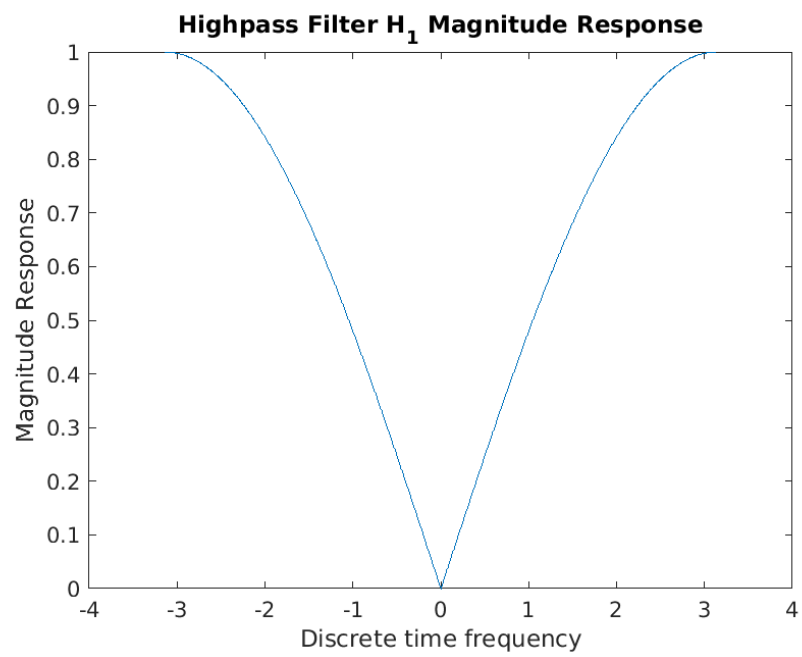


Figure 1.3: Magnitude of the frequency response for H_1

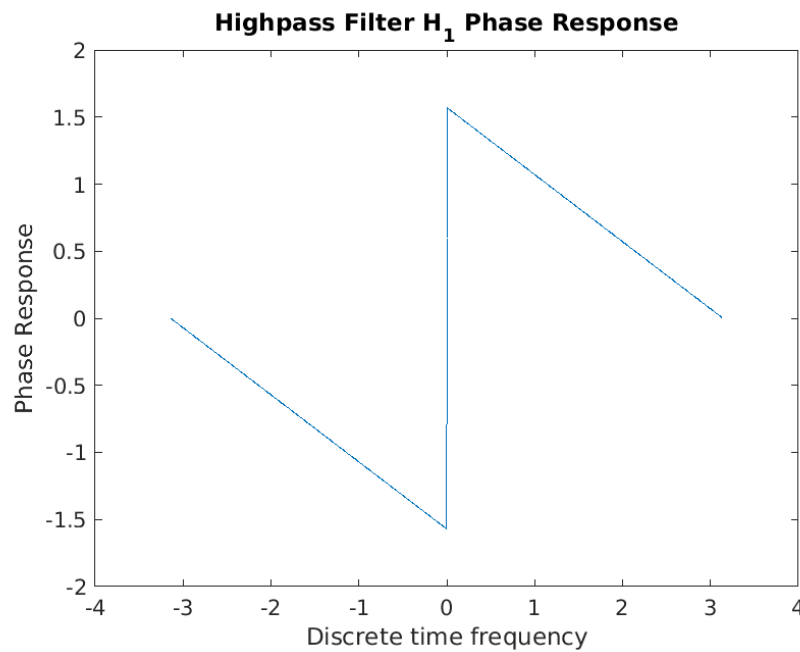


Figure 1.4: Phase of the frequency response H_1

Task b

The filters to be implemented are:

$$G_0 = \frac{1}{2} * (1 + Z^{-1})$$

$$G_1 = \frac{1}{2} * (-1 + Z^{-1})$$

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

```

1  z = tf('z',1/(2*pi*10000));
2
3  G0 = 1/2*(1+z^(-1));
4  G1 = 1/2*(-1+z^(-1));
5
6  frequencies=(-pi:(2*pi/1000):pi);
7  frequencies=frequencies(1:1000);
8  frequencies2=exp(j.*frequencies);
9  G0_resp = freqresp(G0, frequencies2);
10 G1_resp = freqresp(G1, frequencies2);
11
12 fignum=1;
13
14 plotResp(1, fignum, 1, 1, abs(G0_resp),
15         frequencies, [1000], ...
16         [Lowpass Filter G_0 Magnitude Response], ...
17         [Magnitude Response ], ...
18         [Discrete time frequency],0);

```

```

19 angle\textunderscore of\textunderscore signal = angle(G\textunderscore 0\
    textunderscore resp);
20
21 plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
22 [Lowpass Filter  $G_0$  Phase Response], ...
23 [Phase Response], ...
24 [Discrete time frequency],0);
25
26
27
28 plotResp(1, fignum+2, 1, 1, abs(G\textunderscore 1\textunderscore resp),
    frequencies, [1000], ...
29 [Highpass Filter  $G_1$  Magnitude Response], ...
30 [Magnitude Response ], ...
31 [Discrete time frequency],0);
32
33 angle\textunderscore of\textunderscore signal = angle(G\textunderscore 1\
    textunderscore resp);
34
35 plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
36 [Highpass Filter  $G_1$  Phase Response], ...
37 [Phase Response], ...
38 [Discrete time frequency],0);

```

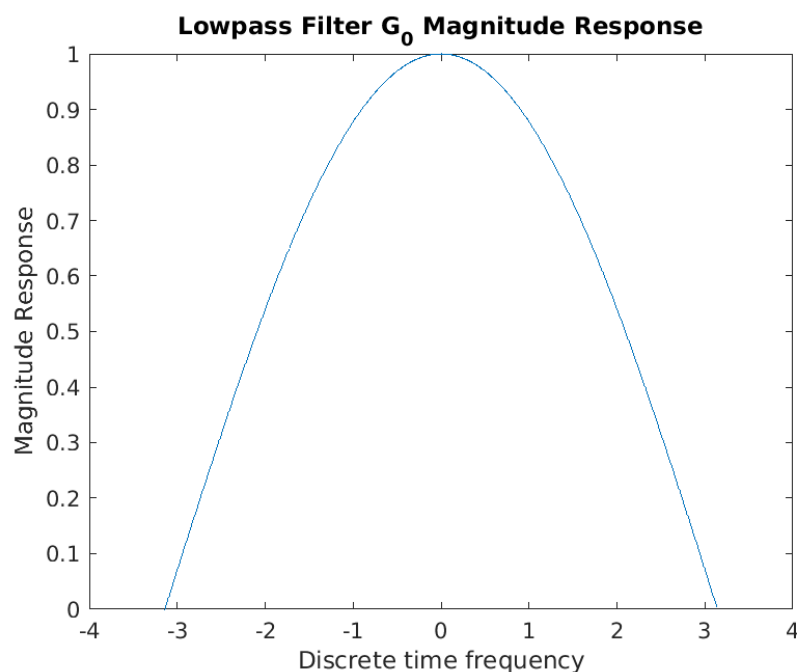


Figure 1.5: Magnitude of the frequency response for G_0

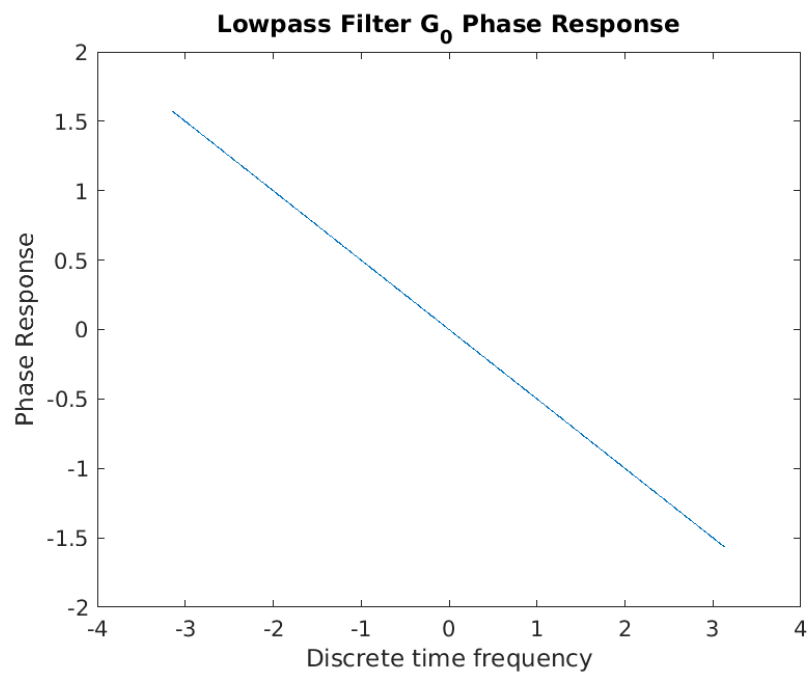


Figure 1.6: Phase of the frequency response for G_0

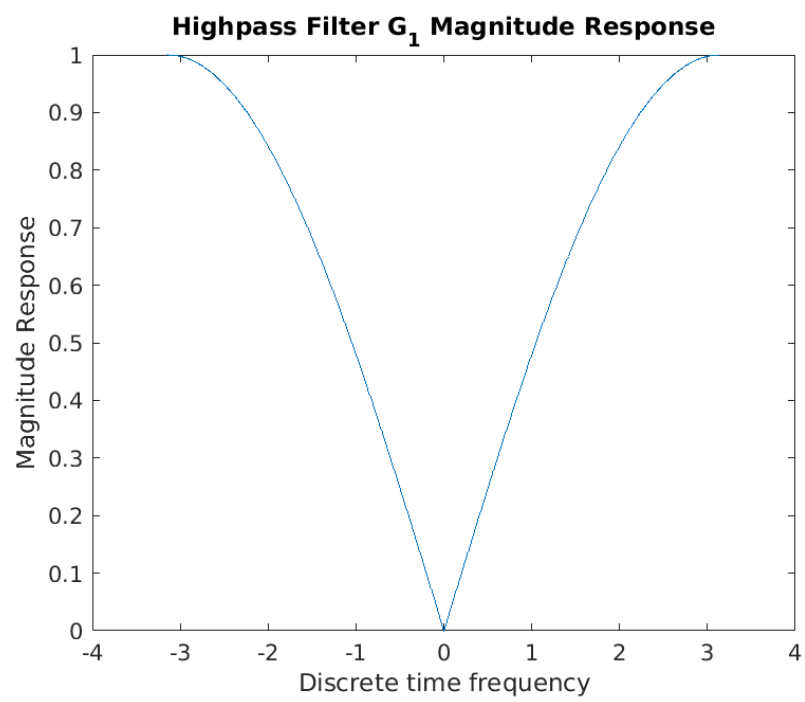


Figure 1.7: Magnitude of the frequency response for G_1

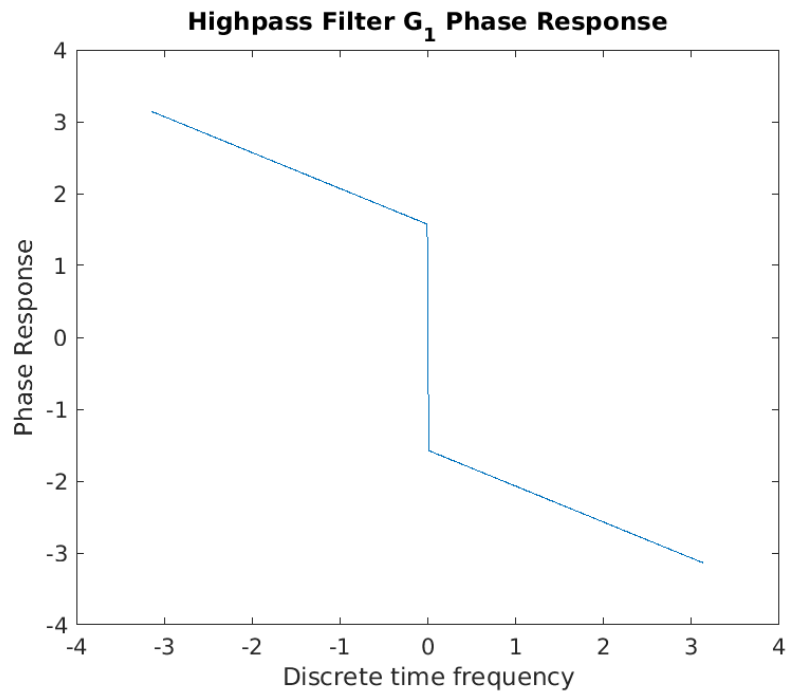


Figure 1.8: Phase of the frequency response for G_1

1.2 Task c: Create a Filter Bank

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

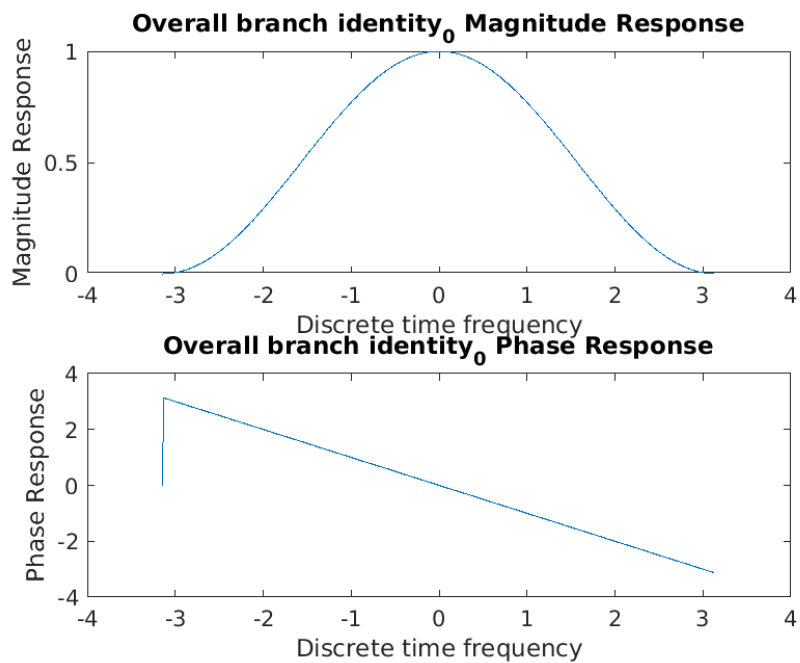


Figure 1.9: The frequency response of the first branch of the parallel filters

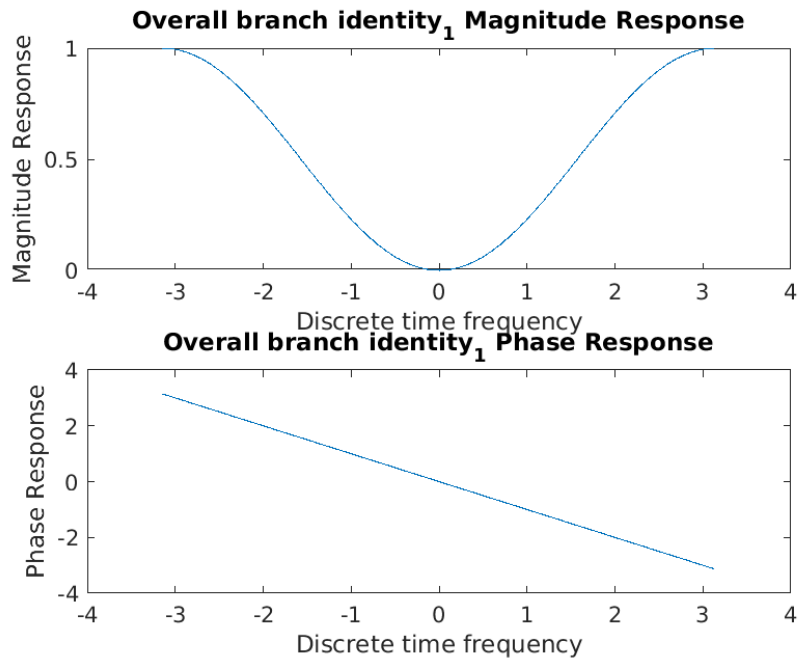


Figure 1.10: The frequency response of the second branch of the parallel filters

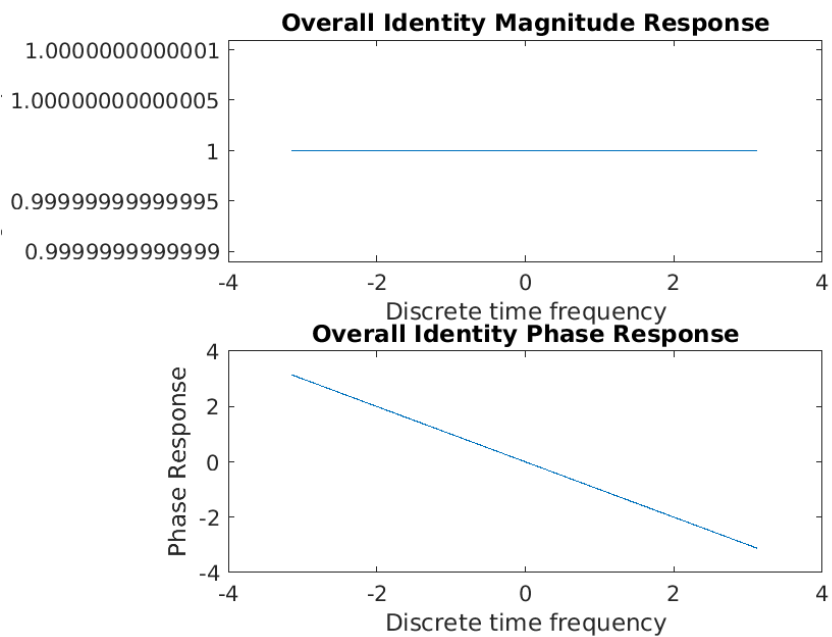


Figure 1.11: The frequency response of the overall filter. Note that the input would be recovered perfectly (with a time delay) with this implementation

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 due to the filter having a passband gain of 1 at DT frequencies of π

The SNR of 0 that occurs for input frequencies of π 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 $0.05 \cdot \cos(\omega n)$ as the ideal value because there would be a 0/0 error otherwise. Instead of $10\log(0/0)$, I end up with $10\log(1/1)$ 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 than $1/2$, 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.

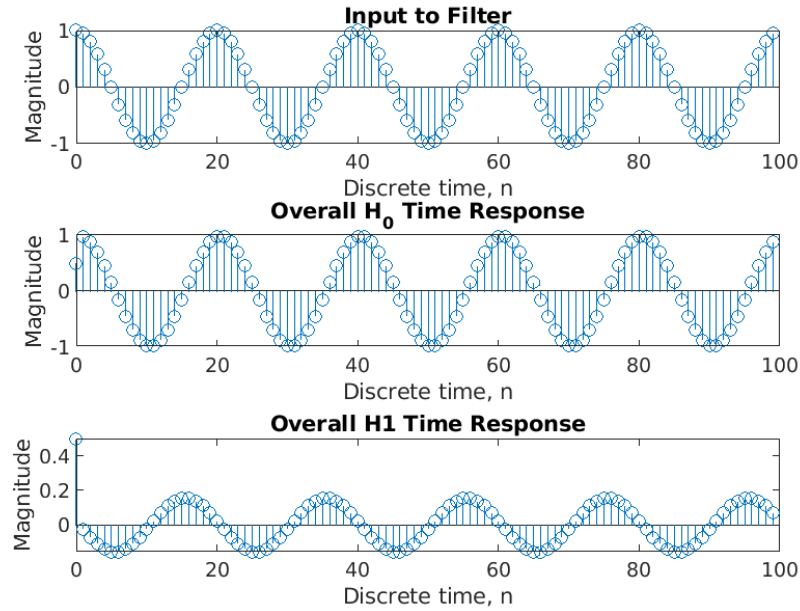


Figure 1.12: The time response of the filters , input frequency is $\pi/10$

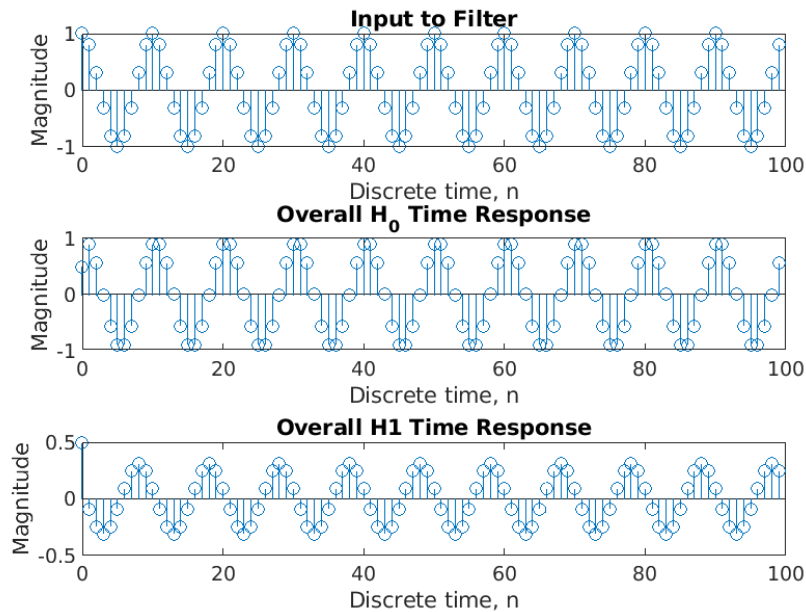


Figure 1.13: The time response of the filters, input frequency is $2\pi/10$

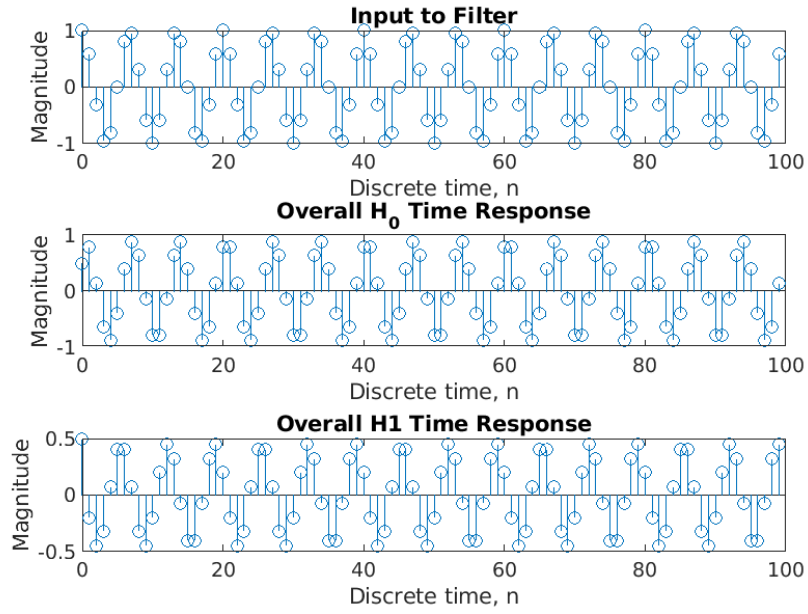


Figure 1.14: The time response of the filters, input frequency is $3\pi/10$

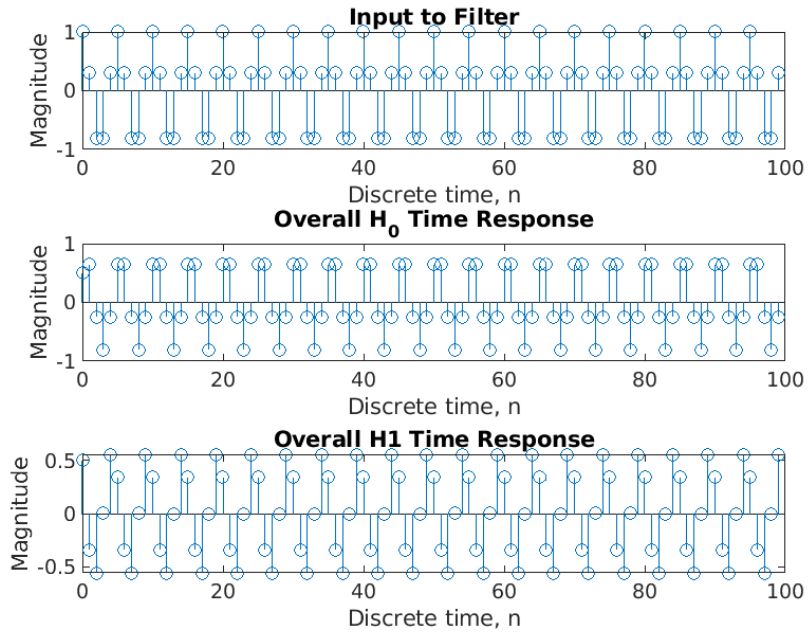


Figure 1.15: The time response of the filters, input frequency is $4\pi/10$

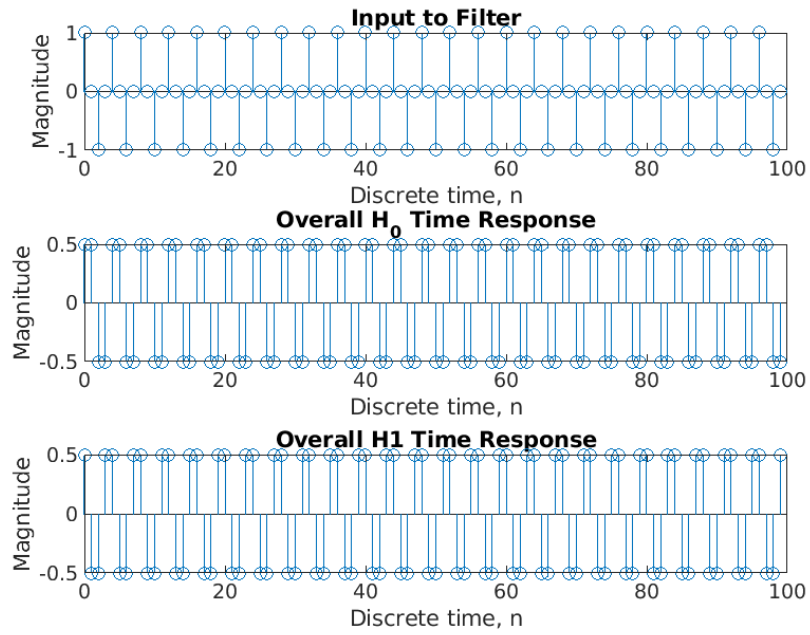


Figure 1.16: The time response of the filters, input frequency is $5\pi/10$

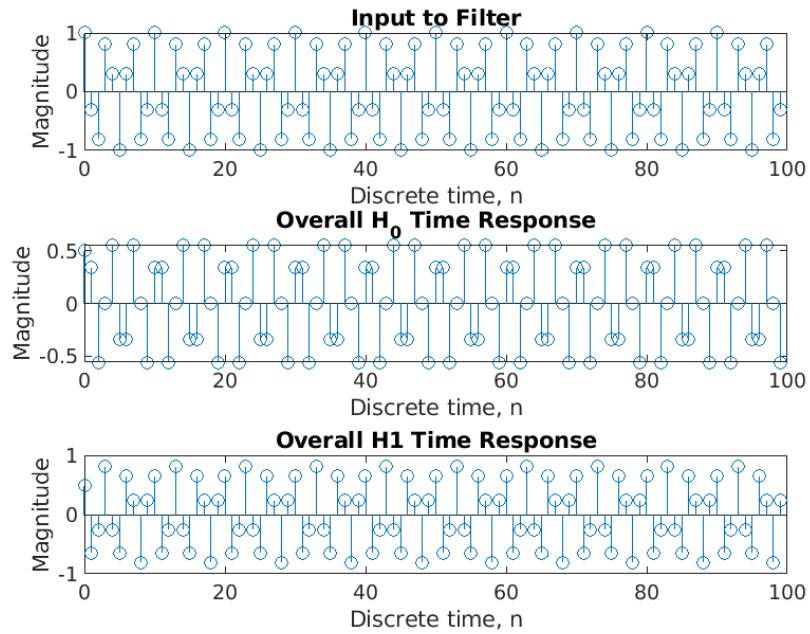


Figure 1.17: The time response of the filters, input frequency is $6\pi/10$

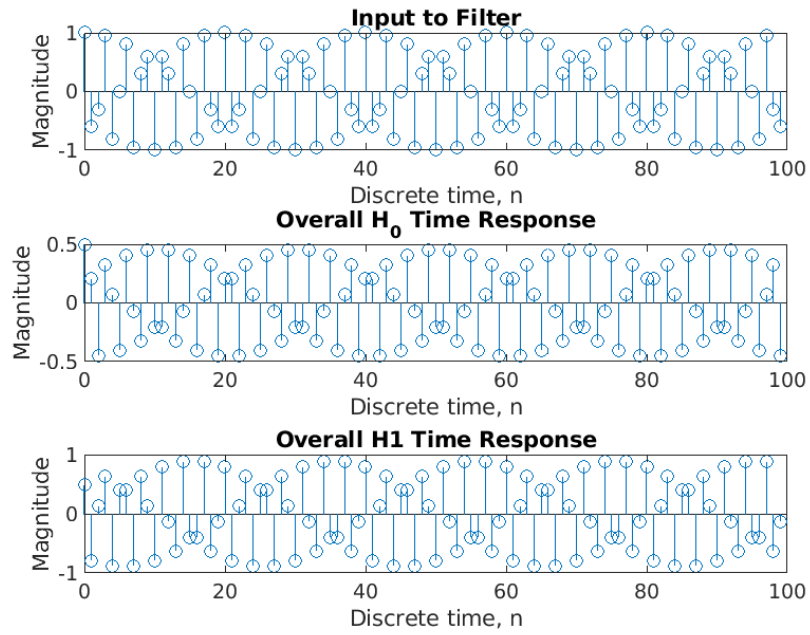


Figure 1.18: The time response of the filters, input frequency is $7\pi/10$

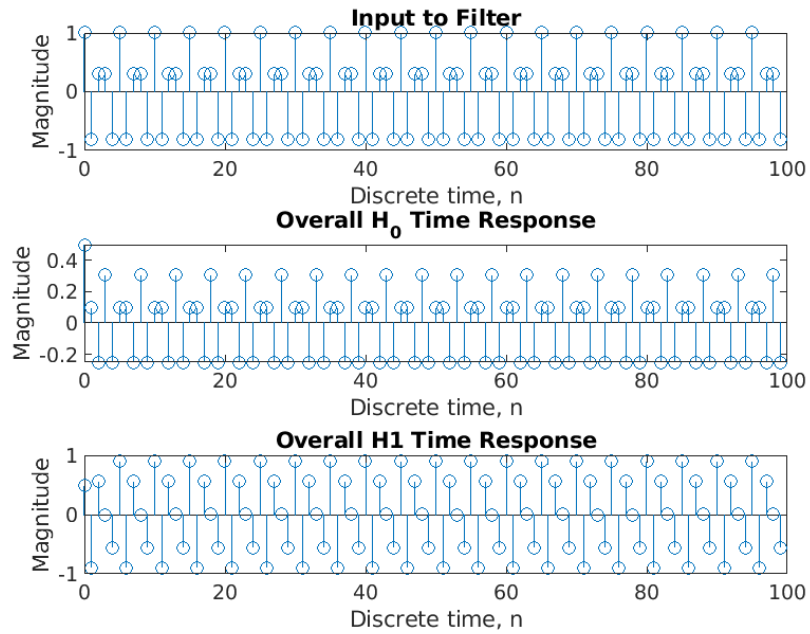


Figure 1.19: The time response of the filters, input frequency is $8\pi/10$

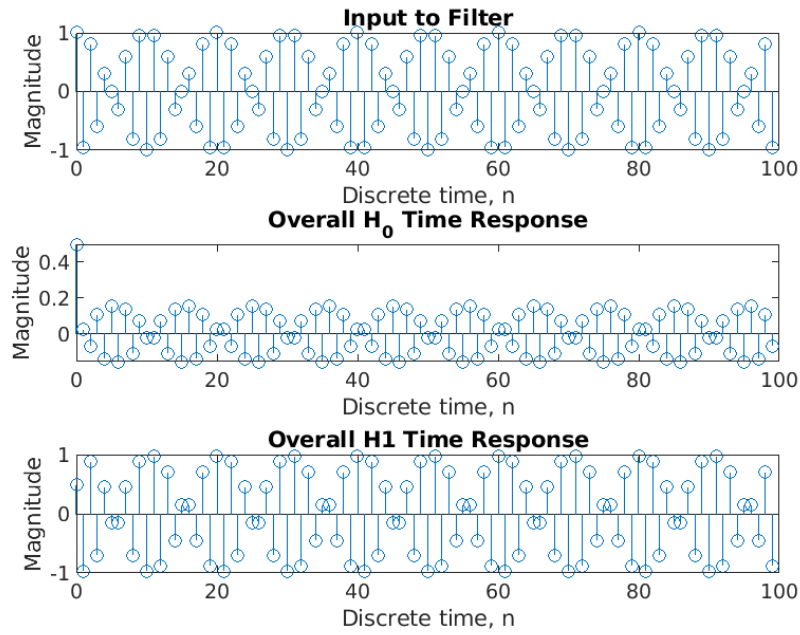


Figure 1.20: The time response of the filters, input frequency is $9\pi/10$

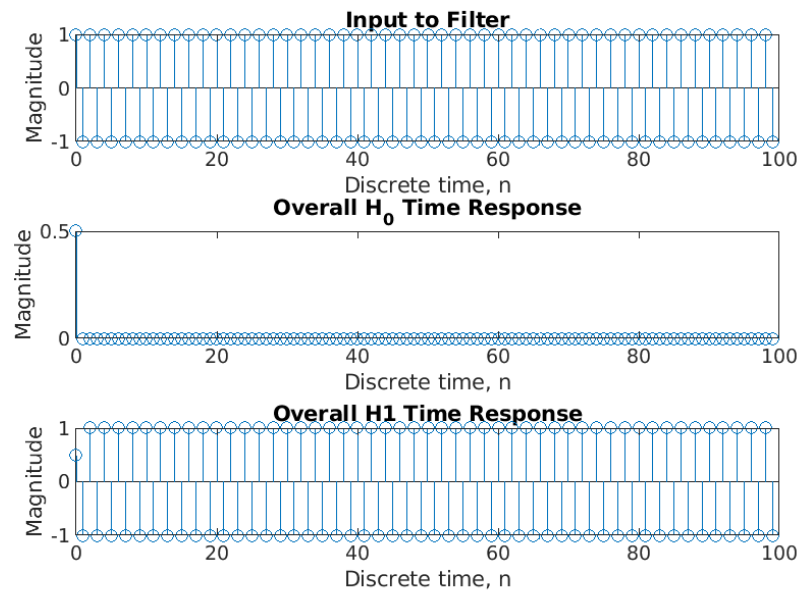


Figure 1.21: The time response of the filters, input frequency is $10\pi/10$

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:

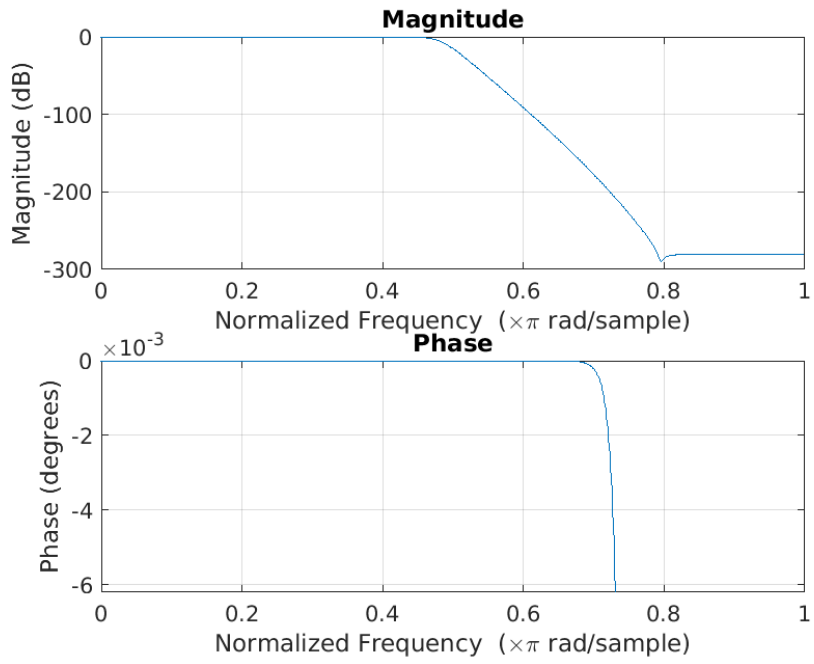


Figure 2.1: The frequency response for the magnitude squared function, as a sanity check before attempting to generate the butterworth filters

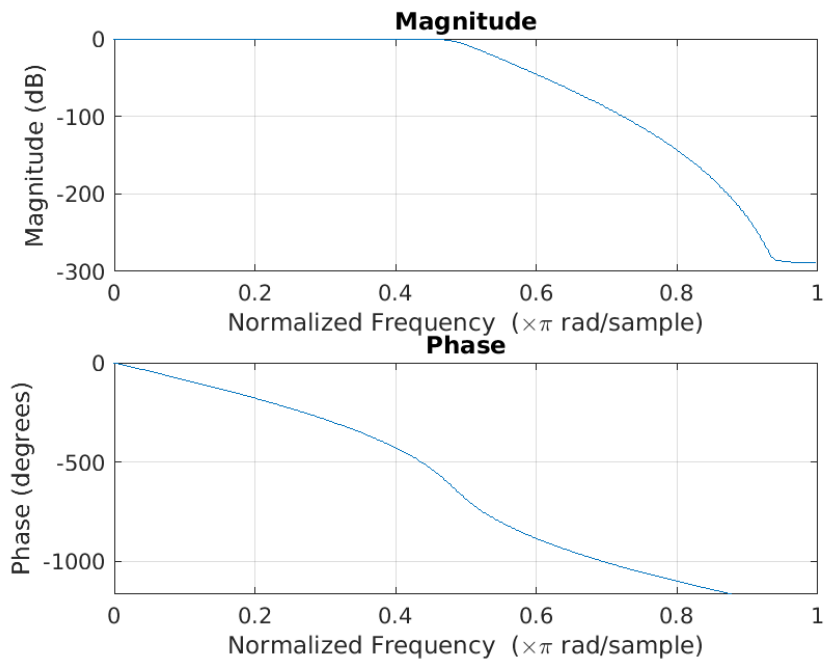


Figure 2.2: The frequency response for H_0 , lowpass Butterworth filter

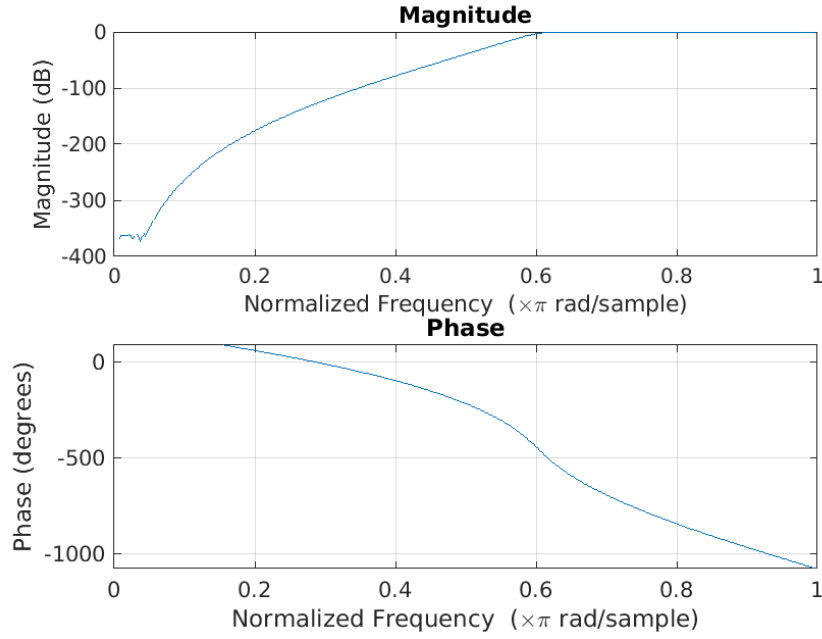


Figure 2.3: The frequency response for H_1 , highpass Butterworth filter

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.

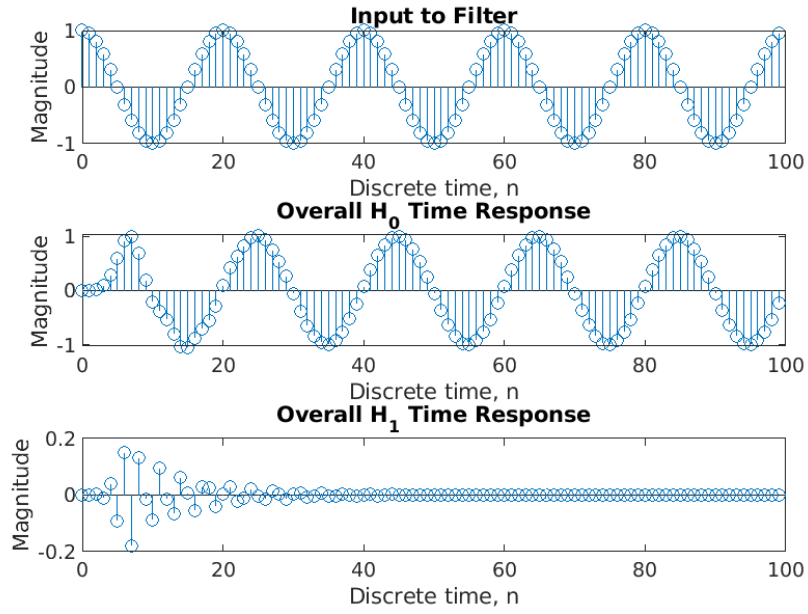


Figure 2.4: The time response of the filters , input frequency is $\pi/10$

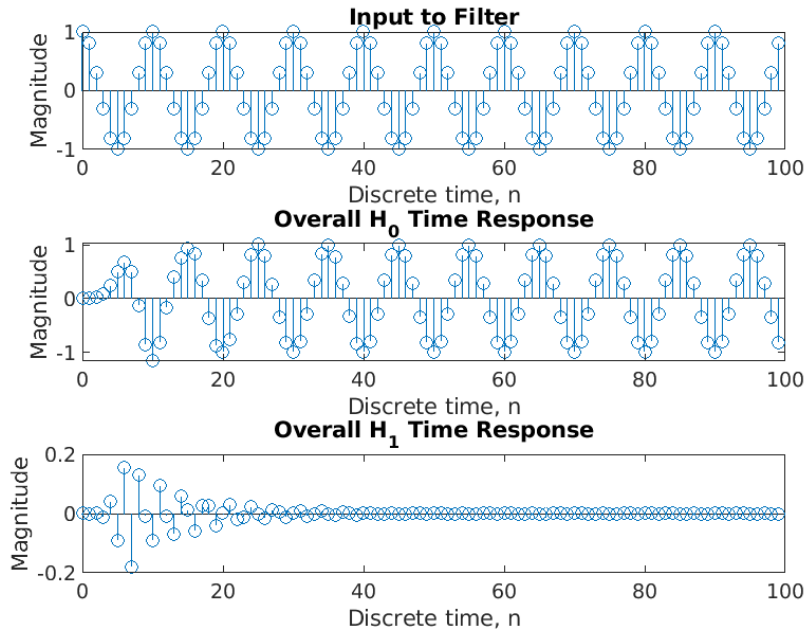


Figure 2.5: The time response of the filters, input frequency is $2\pi/10$

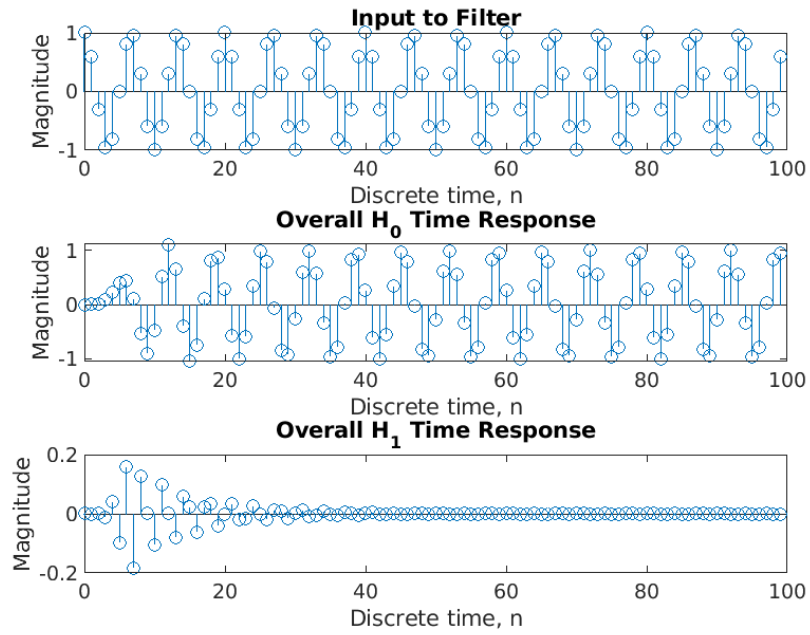


Figure 2.6: The time response of the filters, input frequency is $3\pi/10$

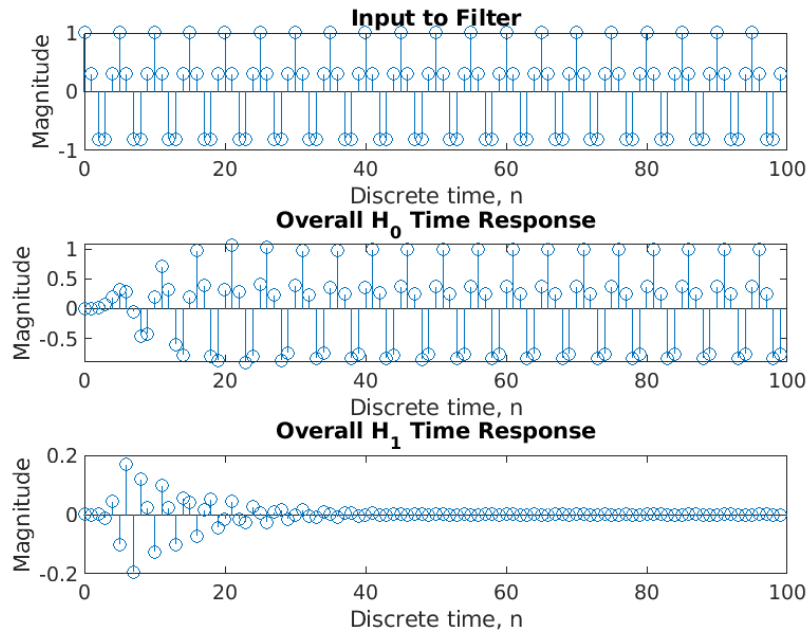


Figure 2.7: The time response of the filters, input frequency is $4\pi/10$

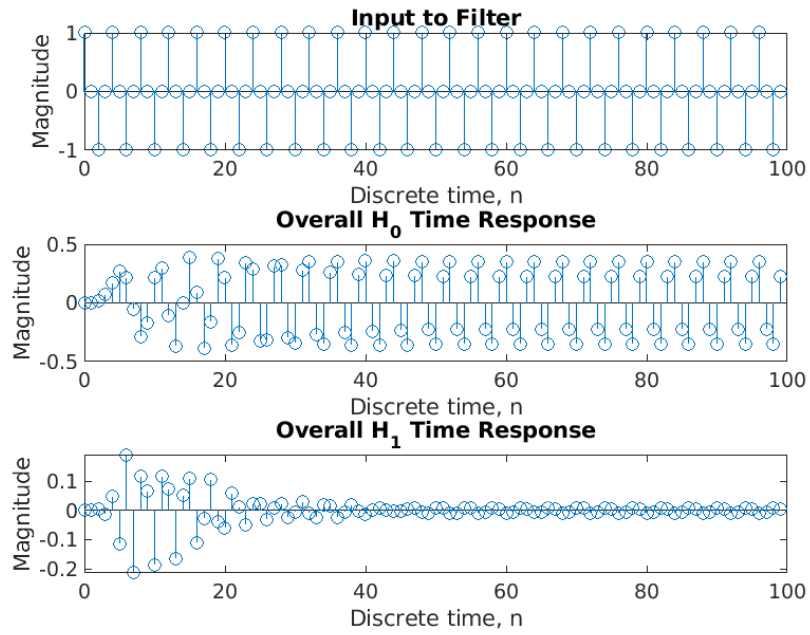


Figure 2.8: The time response of the filters, input frequency is $5\pi/10$

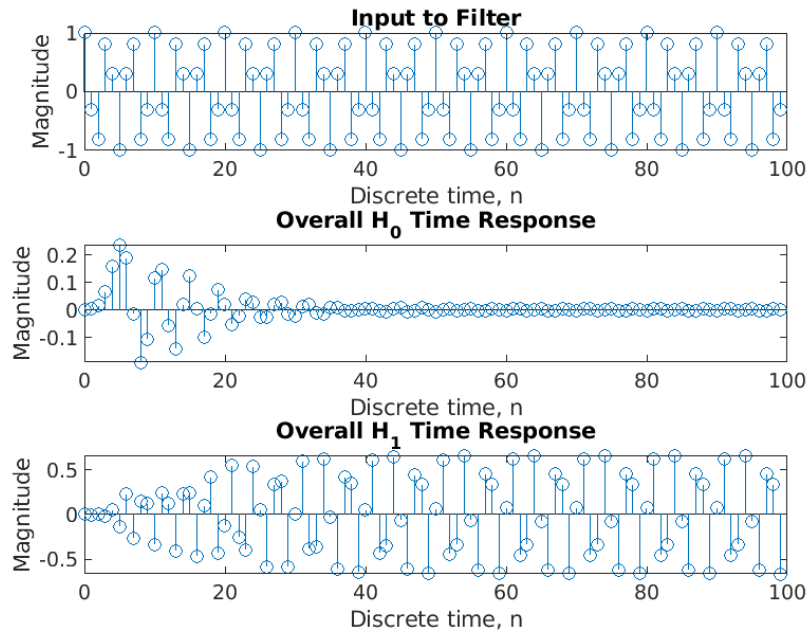


Figure 2.9: The time response of the filters, input frequency is $6\pi/10$

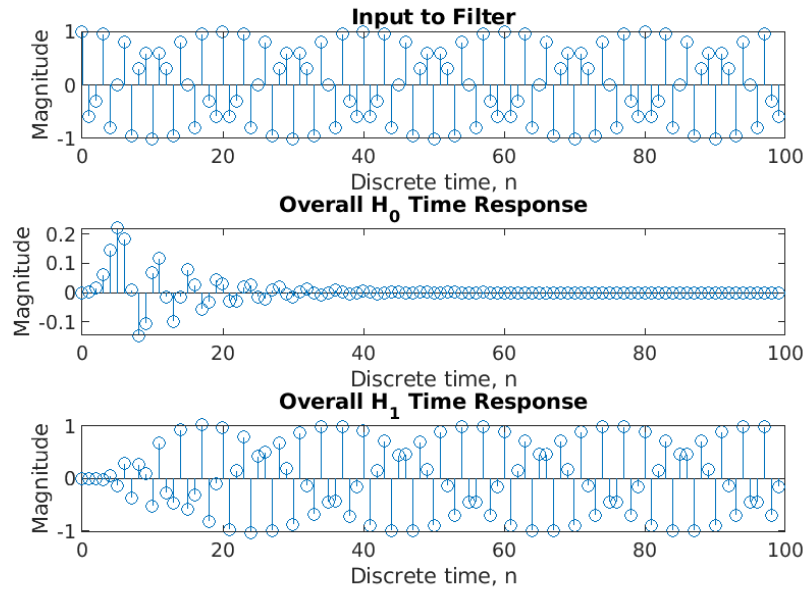


Figure 2.10: The time response of the filters, input frequency is $7\pi/10$

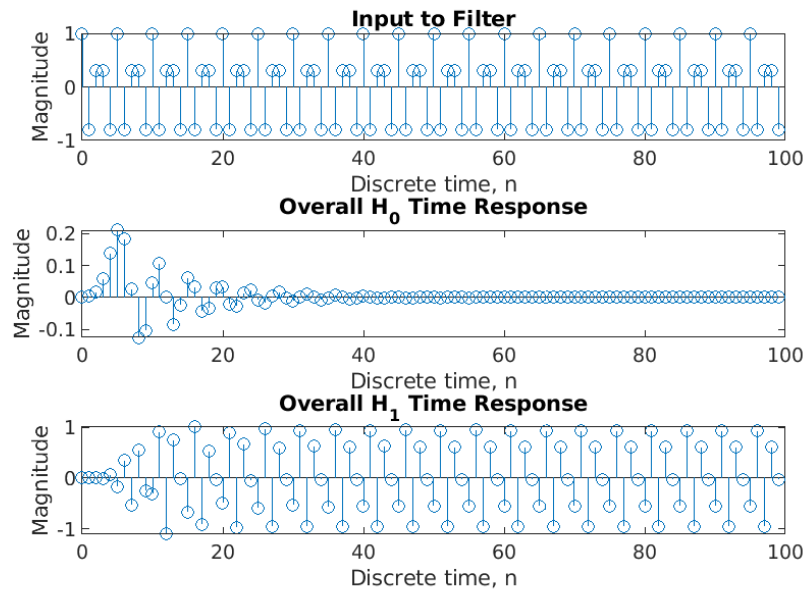


Figure 2.11: The time response of the filters, input frequency is $8\pi/10$

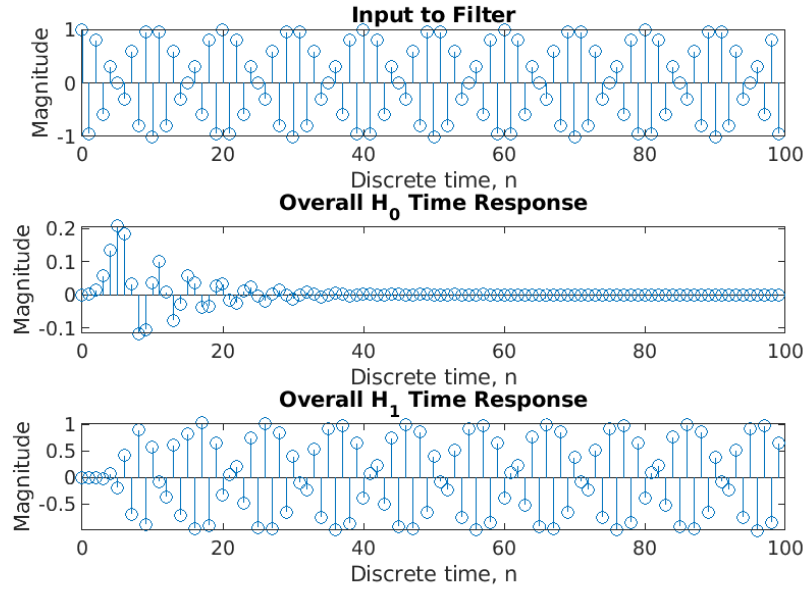


Figure 2.12: The time response of the filters, input frequency is $9\pi/10$

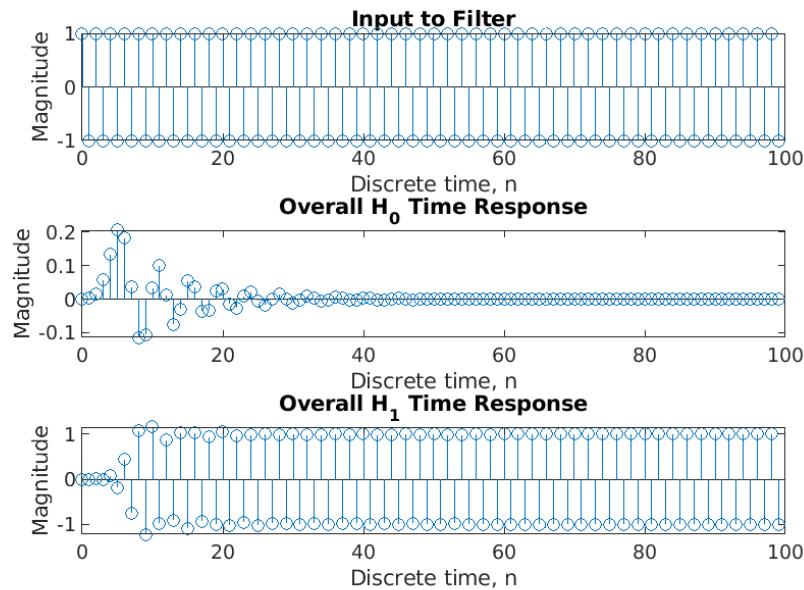


Figure 2.13: The time response of the filters, input frequency is $10\pi/10$

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

```
1
2
3
4 z = tf('z',1/(2*pi*10000));
5
6 H\textunderscore 0 = 1/2*(1+z^(-1));
7 H\textunderscore 1 = 1/2*(1-z^(-1));
8
9 frequencies=(-pi:(2*pi/1000):pi);
10 frequencies=frequencies(1:1000);
11 frequencies2=exp(j.*frequencies);
12 H\textunderscore 0\textunderscore resp = freqresp(H\textunderscore 0,
    frequencies2);
13 H\textunderscore 1\textunderscore resp = freqresp(H\textunderscore 1,
    frequencies2);
14
15 fignum=1;
16
17 plotResp(1, fignum, 1, 1, abs(H\textunderscore 0\textunderscore resp),
    frequencies, [1000], ...
18 [Lowpass Filter  $H_0$  Magnitude Response], ...
19 [Magnitude Response ], ...
20 [Discrete time frequency],0);
21
22 angle\textunderscore of\textunderscore signal = angle(H\textunderscore 0\textunderscore
    resp);
23
24 plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
25 [Lowpass Filter  $H_0$  Phase Response], ...
26 [Phase Response], ...
27 [Discrete time frequency],0);
28
29
30
31 plotResp(1, fignum+2, 1, 1, abs(H\textunderscore 1\textunderscore resp),
    frequencies, [1000], ...
32 [Highpass Filter  $H_1$  Magnitude Response], ...
33 [Magnitude Response ], ...
34 [Discrete time frequency],0);
35
36 angle\textunderscore of\textunderscore signal = angle(H\textunderscore 1\textunderscore
    resp);
37
38 plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
```

```

39 [Highpass Filter  $H_1$  Phase Response], ...
40 [Phase Response], ...
41 [Discrete time frequency],0);

```

3.2 a3scheme1GRespPlots.m

```

1  z = tf('z',1/(2*pi*10000));
2
3  G\textunderscore 0 = 1/2*(1+z^(-1));
4  G\textunderscore 1 = 1/2*(-1+z^(-1));
5
6  frequencies=(-pi:(2*pi/1000):pi);
7  frequencies=frequencies(1:1000);
8  frequencies2=exp(j.*frequencies);
9  G\textunderscore 0\textunderscore resp = freqresp(G\textunderscore 0,
    frequencies2);
10 G\textunderscore 1\textunderscore resp = freqresp(G\textunderscore 1,
    frequencies2);
11
12 fignum=1;
13
14 plotResp(1, fignum, 1, 1, abs(G\textunderscore 0\textunderscore resp),
    frequencies, [1000], ...
15 [Lowpass Filter  $G_0$  Magnitude Response], ...
16 [Magnitude Response ], ...
17 [Discrete time frequency],0);
18
19 angle\textunderscore of\textunderscore signal = angle(G\textunderscore 0\textunderscore
    \textunderscore resp);
20
21 plotResp(1, fignum+1, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...
22 [Lowpass Filter  $G_0$  Phase Response], ...
23 [Phase Response], ...
24 [Discrete time frequency],0);
25
26
27
28 plotResp(1, fignum+2, 1, 1, abs(G\textunderscore 1\textunderscore resp),
    frequencies, [1000], ...
29 [Highpass Filter  $G_1$  Magnitude Response], ...
30 [Magnitude Response ], ...
31 [Discrete time frequency],0);
32
33 angle\textunderscore of\textunderscore signal = angle(G\textunderscore 1\textunderscore
    \textunderscore resp);
34
35 plotResp(1, fignum+3, 1, 1, angle\textunderscore of\textunderscore signal,
    frequencies, [1000], ...

```



```

36 [Highpass Filter  $G_1$  Phase Response], ...
37 [Phase Response], ...
38 [Discrete time frequency],0);

```

3.3 branchResponses.m

```

1  z = tf('z',1/(2*pi*10000));
2
3  H\textunderscore 0 = 1/2*(1+z^(-1));
4  H\textunderscore 1 = 1/2*(1-z^(-1));
5
6  G\textunderscore 0 = 1/2*(1+z^(-1));
7  G\textunderscore 1 = 1/2*(-1+z^(-1));
8
9  identity\textunderscore 0 = H\textunderscore 0*G\textunderscore 0;
10 identity\textunderscore 1 = H\textunderscore 1*G\textunderscore 1;
11
12 frequencies=(-pi:(2*pi/1000):pi);
13 frequencies=frequencies(1:1000);
14 frequencies2=exp(j.*frequencies);
15 identity\textunderscore 0\textunderscore resp = freqresp(identity\
    \textunderscore 0, frequencies2);
16 identity\textunderscore 1\textunderscore resp = freqresp(identity\
    \textunderscore 1, frequencies2);
17
18 fignum=1;
19 mag1=abs(identity\textunderscore 1\textunderscore resp);
20 phase1=angle(identity\textunderscore 1\textunderscore resp);
21 signals1=[transpose(mag1(:)); transpose(phase1(:))];
22 plotResp(2, fignum, 2, 1, signals1, [frequencies; frequencies], [1000 1000],
    ...
23 [Overall branch identity_1 Magnitude Response, Overall branch identity_1 Phase
    Response], ...
24 [Magnitude Response, Phase Response], ...
25 [Discrete time frequency,Discrete time frequency],0);
26
27 fignum=2;
28 mag0=abs(identity\textunderscore 0\textunderscore resp);
29 phase0=angle(identity\textunderscore 0\textunderscore resp);
30 signals0=[transpose(mag0(:)); transpose(phase0(:))];
31 plotResp(2, fignum, 2, 1, signals0, [frequencies; frequencies], [1000 1000],
    ...
32 [Overall branch identity_0 Magnitude Response, Overall branch identity_0 Phase
    Response], ...
33 [Magnitude Response, Phase Response], ...
34 [Discrete time frequency,Discrete time frequency],0);
35
36 fignum=3;

```

```

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

```

3.4 butterworthScheme2.m

```

1 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 1];
2 b=[1];
3 [resids, poles, k]=residue(b,a);
4
5 sys1=tf(b,a);
6 freqs(b,a,(-pi:(2*pi/1000):pi))
7 [numd,dend]=bilinear(cell2mat(sys1.Numerator), cell2mat(sys1.Denominator),1);
8 figure(1);
9 freqz(numd,dend,1000);
10
11 %find all LHP poles of CT magnitude squared function
12 butterPoles=zeros(1,14);
13 a=zeros(1,14);
14 b=zeros(1,14);
15 for i=1:7
16     angle=pi+pi/28+(i-1)*pi/14;
17     a(i)=1.89077219*(cos(angle));
18     a(i+7)=1.89077219*(cos(angle));
19     b(i)=1.89077219*sin(angle);
20     b(i+7)=-1.89077219*sin(angle);
21 end
22 butterPoles=complex(a,b);
23
24 poles=transpose(butterPoles);
25
26 denominator=poly(butterPoles);
27 numerator=[1.89077219^14];
28
29 [numerator\textunderscore d\textunderscore lp,denominator\textunderscore d\
    textunderscore lp]=bilinear(numerator,denominator,1);
30
31 figure(2)
32 freqz(numerator\textunderscore d\textunderscore lp, denominator\textunderscore
    d\textunderscore lp);
33
34 %highpass butterworth filter
35

```

```

36
37 [numerator\textunderscore d\textunderscore hp, denominator\textunderscore d\
    textunderscore hp]=butter(14,1.89077219/pi,'high');
38
39 figure(3)
40 freqz(numerator\textunderscore d\textunderscore hp, denominator\textunderscore
    d\textunderscore hp);

```

3.5 testHaarFilters.m

```

1 base\textunderscore f = pi/10;
2 matlabIndexOffset=1;
3 z = tf('z',1/(2*pi*10000));
4
5 H\textunderscore 0 = 1/2*(1+z^(-1));
6 H\textunderscore 1 = 1/2*(1-z^(-1));
7
8 n=[0:1:99]
9
10 h\textunderscore 0\textunderscore outputs=zeros(10,100);
11 h\textunderscore 1\textunderscore outputs=zeros(10,100);
12 input\textunderscore signals=zeros(10,100);
13
14 for i=1:10
15     % baseband of discrete time fourier transform of cos(i*base\textunderscore
        f*n) is pi
16     % occuring at +/-omega\textunderscore nought. entirely real, phase is 0
        degrees
17
18     % from tables, mthe z transform of cos(i*base\textunderscore f*n) is :
19     % cos(w\textunderscore 0*n)u(n)=(1-cos(w\textunderscore 0)z^(-1))/(1-2cos
        (w\textunderscore 0)z^(-1)+z^(-2))
20     w\textunderscore 0=i*base\textunderscore f
21     %input\textunderscore signal=(1-cos(w\textunderscore 0)*z^(-1))/(1-2*cos(w\
        textunderscore 0)*z^(-1)+z^(-2));
22
23     % a very simple approach: find the output using the difference equation
24     % from initial rest.
25
26     input\textunderscore signal=cos(w\textunderscore 0.*n);
27     output\textunderscore signal\textunderscore 0=zeros(1,100);
28     output\textunderscore signal\textunderscore 1=zeros(1,100);
29
30     output\textunderscore signal\textunderscore 0(1)=input\textunderscore
        signal(1)*0.5;
31     output\textunderscore signal\textunderscore 1(1)=input\textunderscore
        signal(1)*0.5;
32
33     for j = 2:100

```

```

34
35     output\textunderscore signal\textunderscore 0(j)=input\textunderscore
        signal(j)*0.5+0.5*input\textunderscore signal(j-1);
36
37     output\textunderscore signal\textunderscore 1(j)=input\textunderscore
        signal(j)*0.5-0.5*input\textunderscore signal(j-1);
38
39 end
40
41     h\textunderscore 0\textunderscore outputs(i,1:100)=output\textunderscore
        signal\textunderscore 0;
42     h\textunderscore 1\textunderscore outputs(i,1:100)=output\textunderscore
        signal\textunderscore 1;
43     input\textunderscore signals(i,1:100)=input\textunderscore signal;
44 end
45
46 for i = 1:10
47     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], ...
48     [Input to Filter, Overall  $H_0$  Time Response, Overall  $H_1$  Time
        Response], ...
49     [Magnitude, Magnitude, Magnitude], ...
50     [Discrete time, n, Discrete time, n, Discrete time, n], 1);
51 end
52
53 %based on the output of these filters, the first sample of the output
54 %should be omitted from the MSE calculation
55
56 % ideally, the h\textunderscore 0 filter attenuates all high frequencies
57 MSE\textunderscore lp=0
58 ideal\textunderscore val\textunderscore squared=0;
59 %lowest frequency signal is  $\pi/10=dt$  frequency, it should be passed
60 for i = 2:100
61     %the causal delay is 1, but the other problem to deal with here is
62     %input signal starting from  $\cos(0)$  while matlab indexing only allows
63     % you to index starting from 1, thus the use of -2 in the ideal cos's
        argument.
64
65     MSE\textunderscore lp=MSE\textunderscore lp+(cos(pi/10*(i-2))-h\
        textunderscore 0\textunderscore outputs(1,i))^2;
66     ideal\textunderscore val\textunderscore squared=ideal\textunderscore val\
        textunderscore squared+(cos(pi/10*(i-2)))^2;
67 end
68
69 snr\textunderscore lp = 10*log(ideal\textunderscore val\textunderscore squared/
        MSE\textunderscore lp);
70
71 MSE\textunderscore lp2=0

```

```

72 %assume that the signal's ideal value is just the input attenuated by a factor
    of 0.01
73 ideal\textunderscore val\textunderscore squared2=0;
74 %highest frequency signal is pi it should be completely attenuated
75 for i = 2:100
76     MSE\textunderscore lp2=MSE\textunderscore lp2+(0.05*cos(pi*(i-
        matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(10,i))^2;
77     ideal\textunderscore val\textunderscore squared2=ideal\textunderscore val\
        textunderscore squared2+0.00025*(cos(pi*(i-matlabIndexOffset)))^2;
78 end
79
80 snr\textunderscore lp2 = 10*log(ideal\textunderscore val\textunderscore
    squared2/MSE\textunderscore lp2);
81
82 %what is the half-power bandwidth of the lowpass filter?
83
84 MSE\textunderscore lp3=0
85 ideal\textunderscore val\textunderscore squared3=0;
86 delay=6
87 %from graphs, delay is 1 sample and transient is 1 sample
88 %medium frequency signal is 3pi/10=dt frequency, it should be passed
89 for i = 2:100
90     MSE\textunderscore lp3=MSE\textunderscore lp3+(cos(3*pi/10*(i-delay-
        matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(3,i))^2;
91     ideal\textunderscore val\textunderscore squared3=ideal\textunderscore val\
        textunderscore squared3+(cos(3*pi/10*(i-delay-matlabIndexOffset)))^2;
92 end
93
94 snr\textunderscore lp3 = 10*log(ideal\textunderscore val\textunderscore
    squared3/MSE\textunderscore lp3);
95 %visually inspect delay from graph
96 delay=4;
97 MSE\textunderscore lp4=0;
98 ideal\textunderscore val\textunderscore squared4=0;
99 %medium frequency signal is 6pi/10=dt frequency, it is on the high side and
    should not be passed
100 for i = 2:100
101     MSE\textunderscore lp4=MSE\textunderscore lp4+(0.05*cos(6*pi/10*(i-delay-
        matlabIndexOffset))-h\textunderscore 0\textunderscore outputs(6,i))^2;
102     ideal\textunderscore val\textunderscore squared4=ideal\textunderscore val\
        textunderscore squared4+0.00025*(cos(6*pi/10*(i-delay-matlabIndexOffset
        )))^2;
103 end
104
105 snr\textunderscore lp4 = 10*log(ideal\textunderscore val\textunderscore
    squared4/MSE\textunderscore lp4);
106
107
108 %do the same for the highpass filter
109

```

```

110 %%
111
112 %based on the output of these filters, to evaluate performance, you need to
113 %account for the transient response of the filter
114
115 %based on the time output graphs, the max transient length is about 40
116 %samples, there is virtually 0 transient response after that in all graphs
117
118 % account for the time delay of the filter in order to compare input and
119 % output as well
120
121 delay=finddelay(input\textunderscore signal, h\textunderscore 1\textunderscore
    outputs(1,:));
122 %inspect delay visually from graph
123 delay=15;
124 % ideally, the h\textunderscore 1 filter attenuates all high frequencies
125 MSE\textunderscore hp=0;
126 ideal\textunderscore val\textunderscore squared=0;
127 %lowest frequency signal is  $\pi/10=dt$  frequency, it should not be passed
128 %assume that the signal's ideal value is just the input attenuated by a factor
    of 0.05
129 for i = 2:100
130     MSE\textunderscore hp=MSE\textunderscore hp+(0.05*cos(pi/10*(i-delay-
        matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(1,i))^2;
131     ideal\textunderscore val\textunderscore squared=ideal\textunderscore val\
        textunderscore squared+0.00025*(cos(pi/10*(i-delay-matlabIndexOffset)))
        ^2;
132 end
133
134 snr\textunderscore hp = 10*log(ideal\textunderscore val\textunderscore squared/
    MSE\textunderscore hp);
135
136
137 %inspect delay visually from graph
138 delay=0;
139
140 MSE\textunderscore hp2=0;
141 ideal\textunderscore val\textunderscore squared2=0;
142 %highest frequency signal is  $\pi$  it should be completely passed
143 for i = 2:100
144     MSE\textunderscore hp2=MSE\textunderscore hp2+(cos(pi*(i-delay-
        matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(10,i))^2;
145     ideal\textunderscore val\textunderscore squared2=ideal\textunderscore val\
        textunderscore squared2+(cos(pi*(i-delay-matlabIndexOffset)))^2;
146 end
147
148 snr\textunderscore hp2 = 10*log(ideal\textunderscore val\textunderscore squared2/
    MSE\textunderscore hp2);
149
150 %what is the half-power bandwidth of the lowpass filter?

```

```

151
152 delay=finddelay(input\textunderscore signal, h\textunderscore 1\textunderscore
    outputs(3,:));
153
154 %inspect delay visually from graph
155 delay=1;
156
157 MSE\textunderscore hp3=0;
158 ideal\textunderscore val\textunderscore squared3=0;
159 %medium frequency signal is  $3\pi/10=dt$  frequency, it is on the low side and
    should attenuated
160 for i = 2:100
161     MSE\textunderscore hp3=MSE\textunderscore hp3+(0.05*cos(3*pi/10*(i-delay-
        matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(3,i))^2;
162     ideal\textunderscore val\textunderscore squared3=ideal\textunderscore val\
        textunderscore squared3+0.00025*(cos(3*pi/10*(i-delay-matlabIndexOffset
        )))^2;
163 end
164
165 snr\textunderscore hp3 = 10*log(ideal\textunderscore val\textunderscore
    squared3/MSE\textunderscore hp3);
166
167 %inspect delay visually from graph
168 delay=3;
169
170 MSE\textunderscore hp4=0;
171 ideal\textunderscore val\textunderscore squared4=0;
172 %medium frequency signal is  $6\pi/10=dt$  frequency, it is on the high side and
    should be passed
173 for i = 2:100
174     MSE\textunderscore hp4=MSE\textunderscore hp4+(cos(6*pi/10*(i-delay-
        matlabIndexOffset))-h\textunderscore 1\textunderscore outputs(3,i))^2;
175     ideal\textunderscore val\textunderscore squared4=ideal\textunderscore val\
        textunderscore squared4+(cos(6*pi/10*(i-delay-matlabIndexOffset)))^2;
176 end
177
178 snr\textunderscore hp4 = 10*log(ideal\textunderscore val\textunderscore
    squared4/MSE\textunderscore hp4);

```

3.6 testIIRfilters.m

```

1 n=[0:1:99]
2 matlab_index_offset=1;
3
4 h_0_outputs=zeros(10,100);
5 h_1_outputs=zeros(10,100);
6 input_signals=zeros(10,100);
7 base_f = pi/10;
8 for k=1:10

```

```

9      % baseband of discrete time fourier transform of  $\cos(i \cdot \text{base\_f} \cdot n)$  is  $\pi$ 
10     % occuring at  $\pm \omega_{\text{nought}}$ . entirely real, phase is 0 degrees
11
12     % from tables, the z transform of  $\cos(i \cdot \text{base\_f} \cdot n)$  is :
13     %  $\cos(w_0 \cdot n)u(n) = (1 - \cos(w_0)z^{-1}) / (1 - 2\cos(w_0)z^{-1} + z^{-2})$ 
14      $w_0 = k \cdot \text{base\_f}$ 
15     %input_signal=(1-cos(w_0)*z^(-1))/(1-2*cos(w_0)*z^(-1)+z^(-2));
16
17     % a very simple approach: find the output using the difference equation
18     % from initial rest.
19
20     input_signal=cos(w_0.*n);
21     output_signal_0=zeros(1,100);
22     output_signal_1=zeros(1,100);
23
24     h_0_outputs(k,1:100)=ccde(input_signal, output_signal_0, numerator_d_lp,
25                               denominator_d_lp);
26     h_1_outputs(k,1:100)=ccde(input_signal, output_signal_1, numerator_d_hp,
27                               denominator_d_hp);
28     input_signals(k,1:100)=input_signal;
29 end
30
31 for i = 1:10
32     plotResp(3, i, 3, 1, [input_signals(i,:); h_0_outputs(i,:); h_1_outputs(i,
33         :)], [n; n; n], [100 100 100], ...
34     [Input to Filter, Overall  $H_0$  Time Response, Overall  $H_1$  Time Response], ...
35     [Magnitude, Magnitude, Magnitude], ...
36     [Discrete time, n, Discrete time, n, Discrete time, n],1);
37 end
38
39 %%
40 %based on the output of these filters, to evaluate performance, you need to
41 %account for the transient response of the filter
42
43 %based on the time output graphs, the max transient length is about 40
44 %samples, there is virtually 0 transient response after that in all graphs
45
46 % account for the time delay of the filter in order to compare input and
47 % output as well
48
49 delay=finddelay(input_signal, h_0_outputs(1,:));
50 %inspect delay visually from graph
51 delay=5
52
53 % ideally, the h_0 filter attenuates all high frequencies
54 MSE_lp=0;
55 ideal_val_squared=0;
56 %lowest frequency signal is  $\pi/10 = \text{dt}$  frequency, it should be passed
57 for i = 40:100
58

```



```

55     MSE_lp=MSE_lp+(cos(pi/10*(i-delay-matlab_index_offset))-h_0_outputs(1,i))
        ^2;
56     ideal_val_squared=ideal_val_squared+(cos(pi/10*(i-delay-matlab_index_offset
        )))^2;
57 end
58
59 snr_lp = 10*log(ideal_val_squared/MSE_lp);
60 %inspect delay visually from graph
61 delay=0;
62
63 delay
64
65 MSE_lp2=0;
66 %assume that the signal's ideal value is just the input attenuated by a factor
    of 0.05
67 ideal_val_squared2=0;
68 %highest frequency signal is pi it should be completely attenuated
69 for i = 40:100
70     MSE_lp2=MSE_lp2+(0.05*cos(pi*(i-delay-matlab_index_offset))-h_0_outputs(10,
        i))^2;
71     ideal_val_squared2=ideal_val_squared2+0.00025*(cos(pi*(i-delay-
        matlab_index_offset)))^2;
72 end
73
74 snr_lp2 = 10*log(ideal_val_squared2/MSE_lp2);
75
76 %what is the half-power bandwidth of the lowpass filter?
77
78 delay=finddelay(input_signal, h_0_outputs(3,:));
79 %observe and manually set time delay from graphs
80 delay=12
81
82 MSE_lp3=0;
83 ideal_val_squared3=0;
84 %medium frequency signal is 3pi/10=dt frequency, it should be passed
85 for i = 40:100
86     MSE_lp3=MSE_lp3+(cos(3*pi/10*(i-delay-matlab_index_offset))-h_0_outputs(3,i
        ))^2;
87     ideal_val_squared3=ideal_val_squared3+(cos(3*pi/10*(i-delay-
        matlab_index_offset)))^2;
88 end
89
90 snr_lp3 = 10*log(ideal_val_squared3/MSE_lp3);
91 %after 40 samples observe delay (roughly, it is hard to know because
92 %amplitude is roughly constant at this point)
93 delay=2;
94
95 MSE_lp4=0;
96 ideal_val_squared4=0;

```

```

97 %medium frequency signal is  $6\pi/10=dt$  frequency, it is on the high side and
    should not be passed
98 for i = 40:100
99     MSE_lp4=MSE_lp4+(0.05*cos(6*pi/10*(i-delay-matlab_index_offset))-
        h_0_outputs(6,i))^2;
100     ideal_val_squared4=ideal_val_squared4+0.00025*(cos(6*pi/10*(i-delay-
        matlab_index_offset)))^2;
101 end
102
103 snr_lp4 = 10*log(ideal_val_squared4/MSE_lp4);
104
105
106 %%
107
108
109 %based on the output of these filters, to evaluate performance, you need to
110 %account for the transient response of the filter
111
112 %based on the time output graphs, the max transient length is about 40
113 %samples, there is virtually 0 transient response after that in all graphs
114
115 % account for the time delay of the filter in order to compare input and
116 % output as well
117
118 delay=finddelay(input_signal, h_1_outputs(1,:));
119 %inspect delay visually from graph
120 delay=5
121
122 % ideally, the h_1 filter attenuates all high frequencies
123 MSE_hp=0;
124 ideal_val_squared=0;
125 %lowest frequency signal is  $\pi/10=dt$  frequency, it should not be passed
126 %assume that the signal's ideal value is just the input attenuated by a factor
    of 0.05
127 for i = 40:100
128     MSE_hp=MSE_hp+(0.05*cos(pi/10*(i-delay-matlab_index_offset))-h_1_outputs(1,
        i))^2;
129     ideal_val_squared2=ideal_val_squared2+0.00025*(cos(pi/10*(i-delay-
        matlab_index_offset)))^2;
130 end
131
132 snr_hp = 10*log(ideal_val_squared/MSE_hp);
133 %inspect delay visually from graph
134 delay=0;
135
136 delay
137
138 MSE_hp2=0;
139
140 ideal_val_squared2=0;

```

```

141 %highest frequency signal is pi it should be completely passed
142 for i = 40:100
143     MSE_hp2=MSE_hp2+(cos(pi*(i-delay-matlab_index_offset))-h_1_outputs(10,i))
144         ^2;
145     ideal_val_squared2=ideal_val_squared2+(cos(pi*(i-delay-matlab_index_offset)
146         ))^2;
147 end
148
149 snr_hp2 = 10*log(ideal_val_squared2/MSE_hp2);
150
151 %what is the half-power bandwidth of the lowpass filter?
152
153 delay=finddelay(input_signal, h_1_outputs(3,:));
154
155 MSE_hp3=0;
156 ideal_val_squared3=0;
157 %medium frequency signal is 3pi/10=dt frequency, it is on the low side and
158     should attenuated
159 for i = 40:100
160     MSE_hp3=MSE_hp3+(0.05*cos(3*pi/10*(i-delay-matlab_index_offset))-
161         h_1_outputs(3,i))^2;
162     ideal_val_squared3=ideal_val_squared3+0.00025*(cos(3*pi/10*(i-delay-
163         matlab_index_offset)))^2;
164 end
165
166 snr_hp3 = 10*log(ideal_val_squared3/MSE_hp3);
167
168 MSE_hp4=0;
169 ideal_val_squared4=0;
170 %medium frequency signal is 6pi/10=dt frequency, it is on the high side and
171     should be passed
172 for i = 40:100
173     MSE_hp4=MSE_hp4+(cos(6*pi/10*(i-delay-matlab_index_offset))-h_1_outputs(3,i
174         ))^2;
175     ideal_val_squared4=ideal_val_squared4+(cos(6*pi/10*(i-delay-
176         matlab_index_offset)))^2;
177 end
178
179 snr_hp4 = 10*log(ideal_val_squared4/MSE_hp4);

```

4 APPENDIX B: AUXILLARY FUNCTIONS CALLED IN MAIN SCRIPTS

4.1 plotResp.m

```

1 %frequency response plotting
2
3 %signals are the 2 dimensional vectors to plot
4
5 % n is the number of signals for plotting
6 %
7 % %fign is number of the figure within the script to plot
8
9 % subP1 number of total subplots
10
11 % subP2 number of rows of subplots
12
13 % signals is the array of dependent axis signals to plot
14
15 % indep_ax is the array of independent axis signals to plot
16
17 % len is the array of independent axis lengths
18
19 % titles is the array of titles for the subplots
20
21 % ylabel is the array of ylabels for the subplots
22
23 % xlabel is the array of xlabels for the subplots
24
25 function plotResp = plotResp(n,fign,subP1,subP2,signals,indep_ax,len,titles,
    ylabel,xlabel, stemP)
26     figure(fign)
27 %checks
28     [M,N]=size(signals);
29     if length(len)==n %fails silently if badly formatted data is passed
30         if n == M %fails silently if badly formatted data is passed
31             for i = 1:n
32                 subplot(subP1, subP2, i)
33                 if(stemP)
34                     stem(indep_ax(i:i,1:len(i)),signals(i:i,1:len(i)));
35                 else
36                     plot(indep_ax(i:i,1:len(i)),signals(i:i,1:len(i)));
37                 end
38
39                 title((titles(i)))
40                 ylabel(ylabel(i))
41                 xlabel(xlabel(i))
42             end
43         end
44     end

```

4.2 ccde.m

```

1 function ccde = ccde(input\textunderscore signal, output\textunderscore signal,
   numerator\textunderscore d, denominator\textunderscore d)
2
3 matlabIndexOffset=1;
4
5 for j = 0:(length(input\textunderscore signal)-1)
6
7     for i=0:(length(numerator\textunderscore d)-1)
8         if j-i >= 0
9             output\textunderscore signal(j+matlabIndexOffset)=output\
               textunderscore signal(j+matlabIndexOffset)+numerator\
               textunderscore d(matlabIndexOffset+i)*input\textunderscore
               signal(j+matlabIndexOffset-i);
10        else
11            output\textunderscore signal(j+matlabIndexOffset)=output\
               textunderscore signal(j+matlabIndexOffset)+0;
12        end
13    end
14    %start index from 1 because index zero is on the left side of
15    %equality sign (its already there)
16    for i=1:(length(denominator\textunderscore d)-1)
17        if j-i >= 0
18            output\textunderscore signal(j+matlabIndexOffset)=output\
               textunderscore signal(j+matlabIndexOffset)-denominator\
               textunderscore d(i+matlabIndexOffset)*output\textunderscore
               signal(j+matlabIndexOffset-i);
19        else
20            output\textunderscore signal(j+matlabIndexOffset)=output\
               textunderscore signal(j+matlabIndexOffset)+0;
21        end
22    end
23
24 end
25
26 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

$$0.95 \leq |H(e^{j\omega})| \leq 1, \quad 0 \leq \Omega \leq \frac{2}{T_d} \tan\left(\frac{0.45\pi}{2}\right)$$

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

let $T_d = 1$
to make design easy

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

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

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

switch to equality to use both equations \rightarrow will meet specifications still.

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

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

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

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

$$|H_c(j\Omega)|^2 = \frac{1}{1 + (\Omega/1.89077219)^{28}}$$

use MATLAB to finish the design...

- Take 10 LHP roots of $1/(1 + (\Omega/1.89077219)^{28})$ to get C.T. T.F.

- perform Bilinear transform on the T.F.