LAB 5

Discrete - Time Bandpass Filters

ECEn 380 Section 001

Task 1 signoff : 12/7/15

Task 2 signoff : 12/7/2015

11/16/2015

Benjamin Bergeson

LAB 5

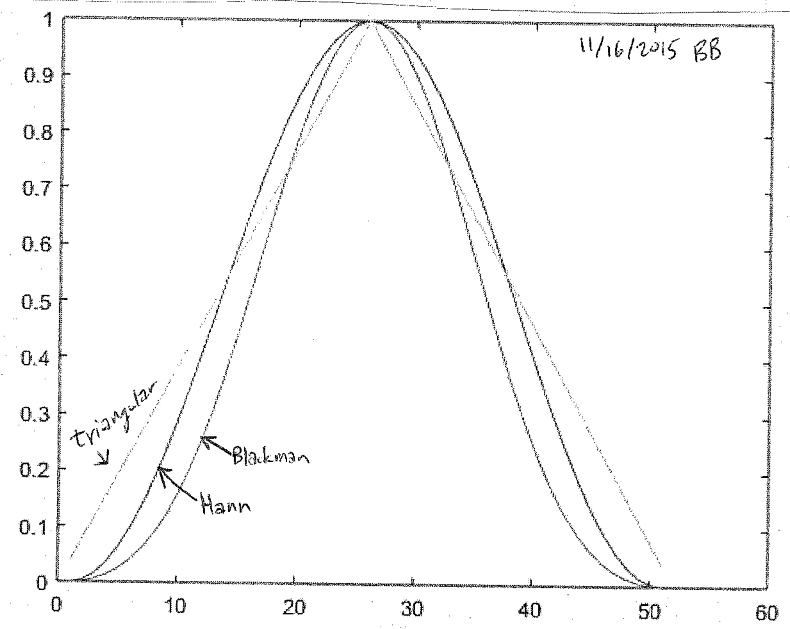
Objective

The purpose of this lab is to design and apply bandpass filters to capture data captured from a laser tag prototype system. The bandpass filters are used to determine the modulating frequency of the received light beam. The results are used to determine who the shooter was.

Task 1

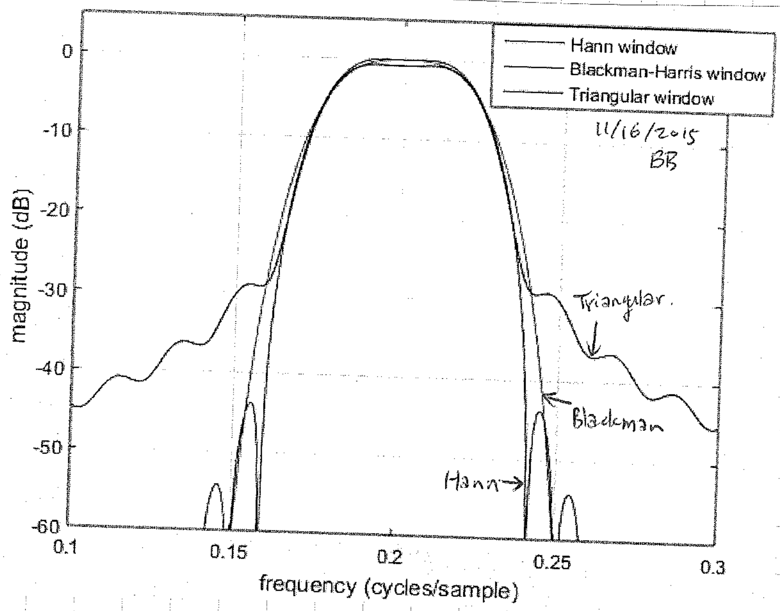
1. We are using the Hann, Blackman, and triangular window.

(a)



amplitude is the same, but the slopes are different.

(b)



11/16/2015

Benjamin Bergeson

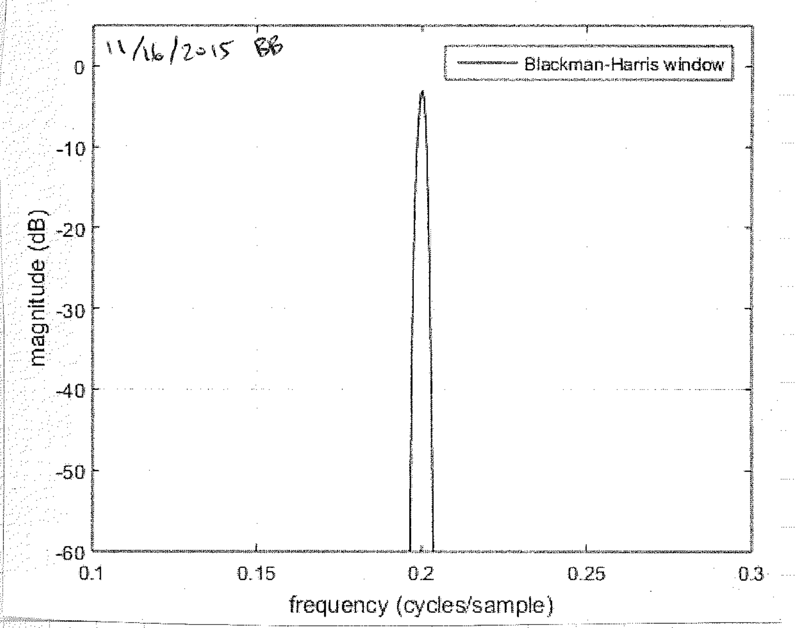
LAB 5

They have the same maximum amplitude. However, the transfer bands are different widths.

Task 1

2. $h_{ideal}[n] = 2B \frac{\sin(\pi B n)}{\pi B n} \cos(2\pi F_0 n)$

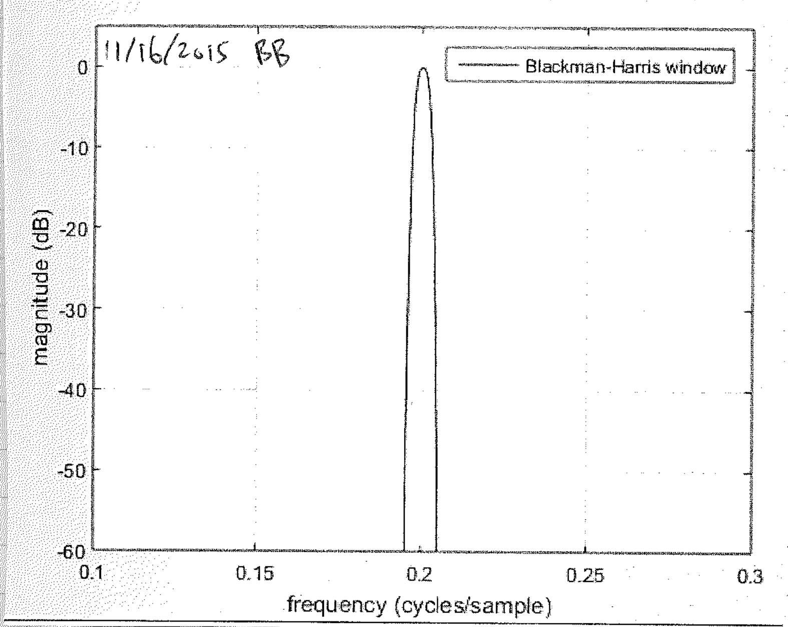
(a) Using Blackman-Harris window.



~~Frequency at~~

Bandwidth = 0.002 cycles/sample

Length = 501



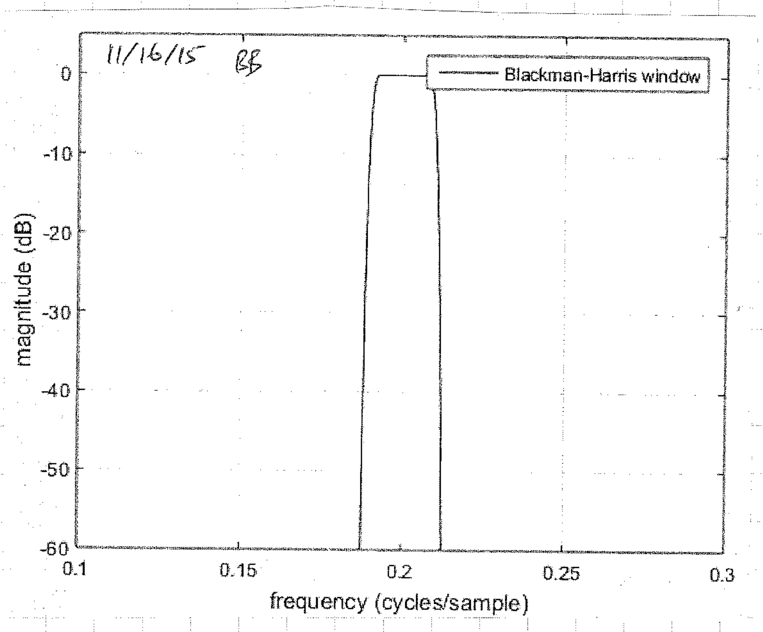
Bandwidth = 0.005 cycles/sample

Length = 501

11/16/2015

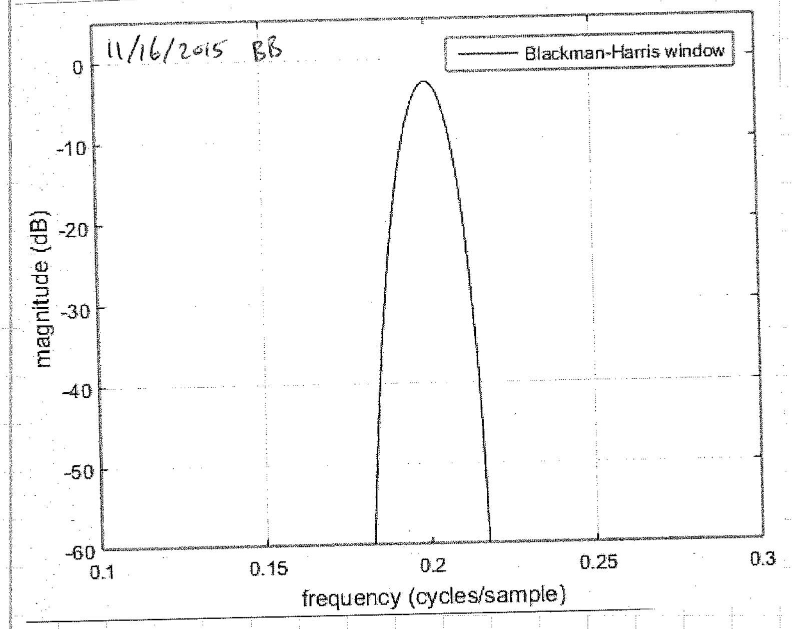
LAB 5

Task 1

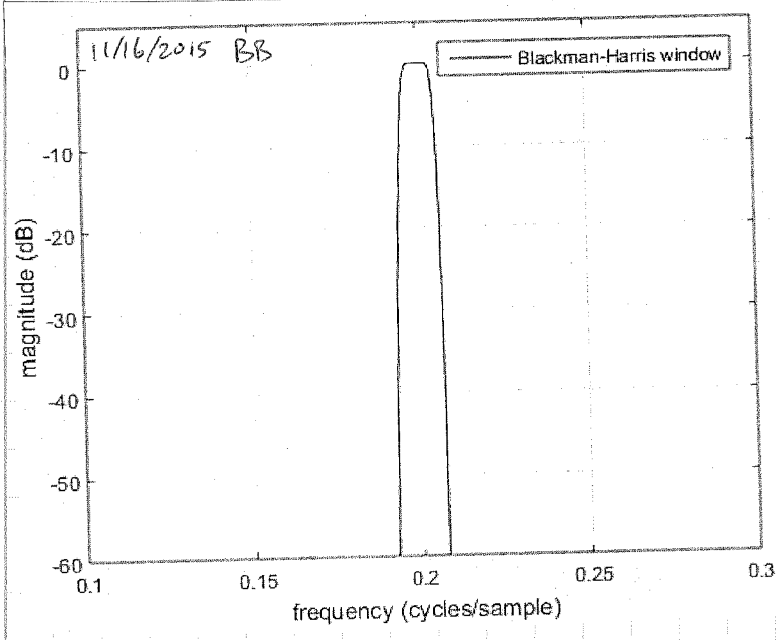


Bandwidth: 0.02 cycles/samp
Length: 501

(b)



Bandwidth: 0.01 cycles/samp
Length: 101

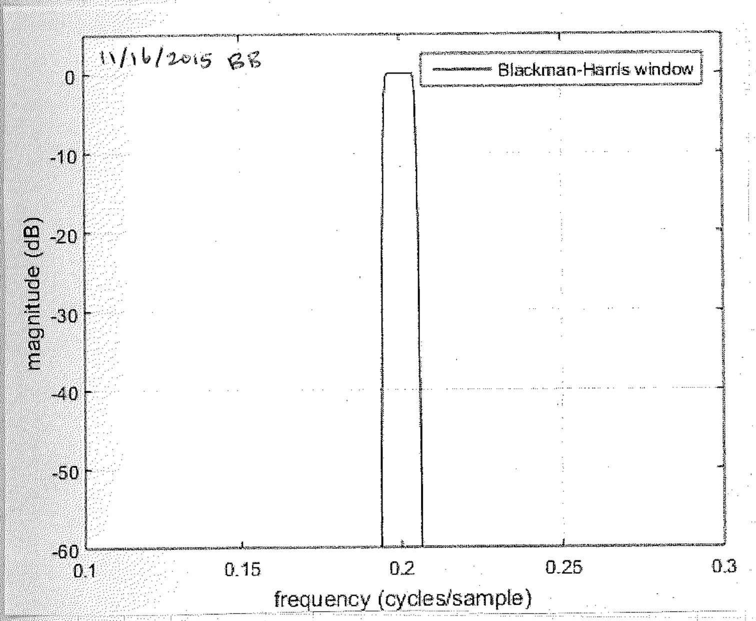


Bandwidth: 0.01 cycles/sam
length: 501

11/16/2015

LAB 5

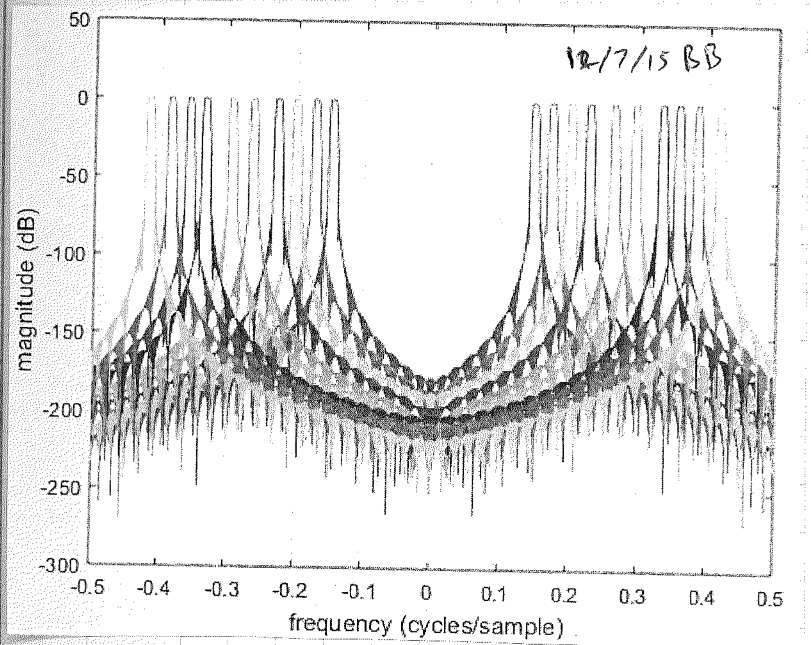
Task 1



11/16/2015 BB

Bandwidth = 0.01 cycles/sample
Length = 1001

~~#~~. 3)



12/7/15 BB

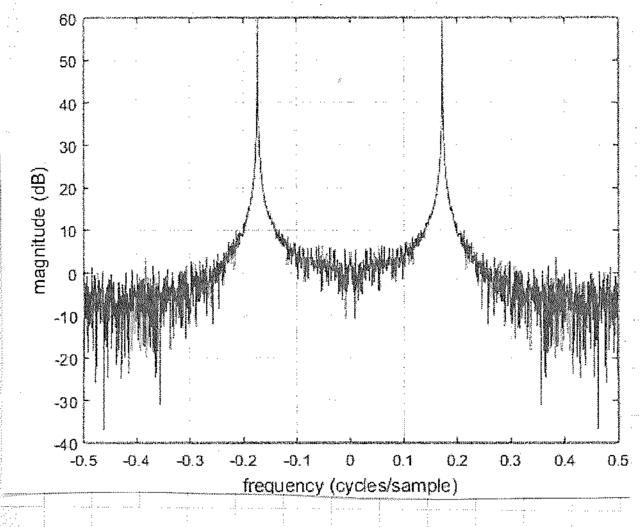
Bandwidth = 0.01
Length = 201
No overlap.

4)

x_easy1

Player 2

12/7/15 BB



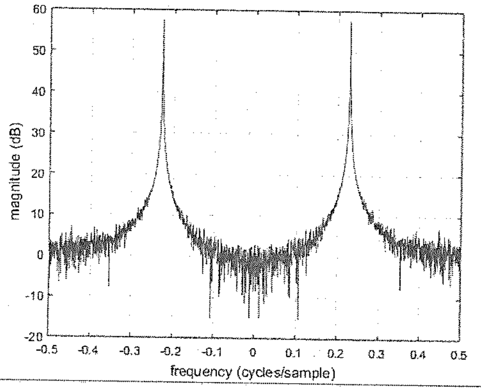
player 2 @ 1.7 kHz

12/7/15

LAB 5

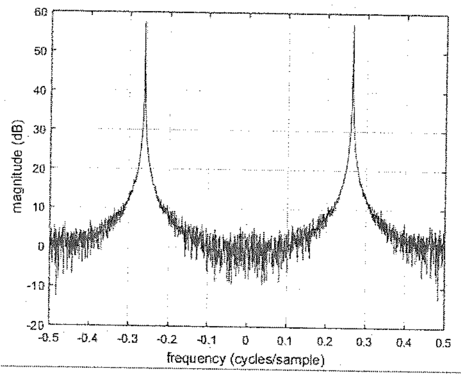
Task 1

X_easy 2 12/7/15 BB



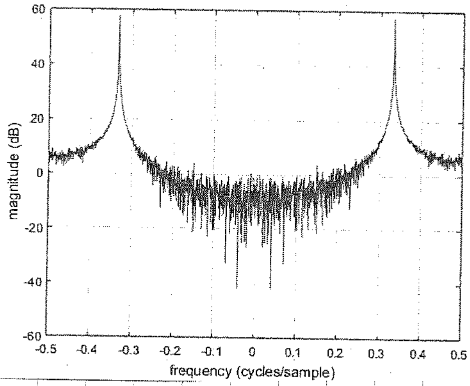
player 4 @ 2.2 kHz

X_easy 3



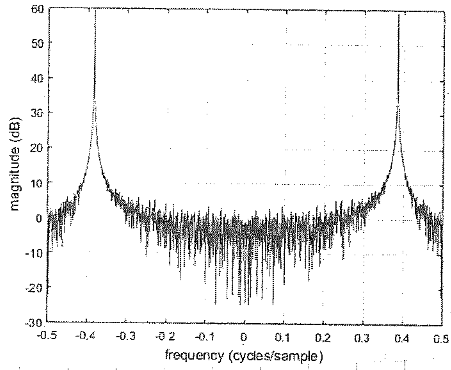
player 5 @ 2.6 kHz

X_easy 4



player 7 @ 3.3 kHz

12/7/15 BB



X_easy 5 Player 9 @ 3.8k

12/7/15

Benjamin Bergman

LAB 5

Task 1

Code to print spectrums

12/7/15 BB

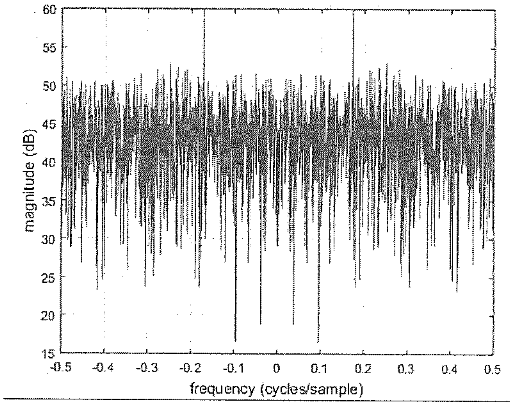
```
Nx = length(x_hard5);
X = fft(x_hard5);

    FF = -0.5:1/Nx:0.5-1/Nx;
    plot(FF,20*log10(abs(fftshift(X))));
    grid on;
    xlabel('frequency (cycles/sample)');
    ylabel('magnitude (dB)');
```

}change these accordingly.

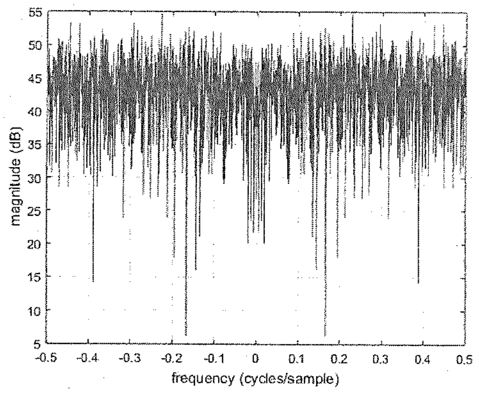
Code for finding player 12/7/15 BB

```
F0 = [0.1471, 0.1724, 0.2, 0.2273, 0.2632,
      0.2941, 0.3333, 0.3571, 0.3846, 0.4167];
B = 0.01; % bandwidith (cycles/sample)
L = 201; % the length parameter L

N = 2*L+1; % the filter length
n = (-L:L)';
hideal = 2*B*cos(2*pi*F0(1)*n).*sinc(B*n);
h0 = blackman(N).*hideal;

power = zeros(1,10);
max_player = -1;
max_power = 0;
for i = 1: length(F0)
    N = 2*L+1; % the filter length
    n = (-L:L)';
    hideal = 2*B*cos(2*pi*F0(i)*n).*sinc(B*n);
    h0 = blackman(N).*hideal;
    y = filter(h0,1, x_hard2);
    power(i) = sum(abs(y.*y));
    if power(i) >= max_power
        max_power = power(i);
        max_player = i;
    end
end

bar(power)
if max_power <= 650
    max_player = -1;
end
max_player
```

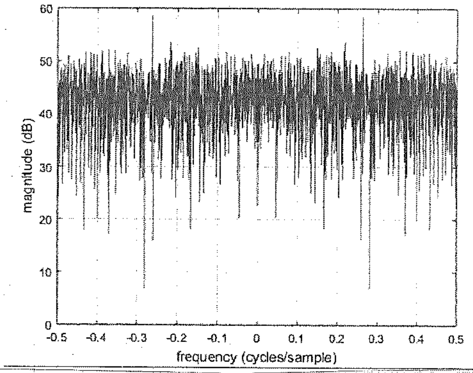
12/7/15 BB x_hard1



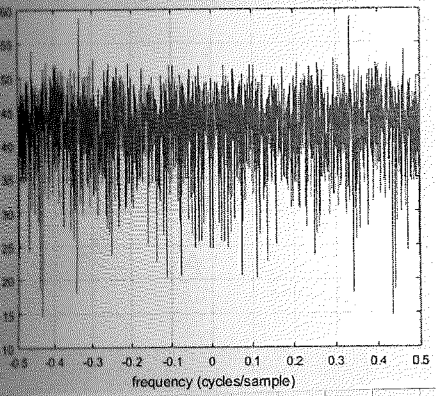
2



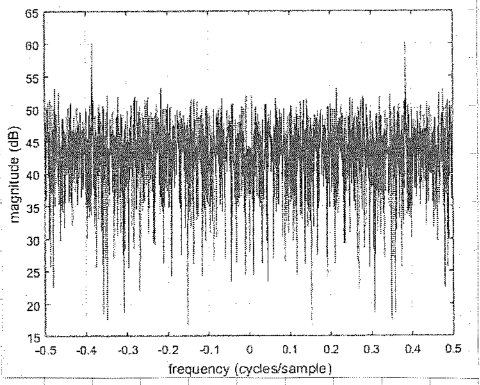
3



X-hard4 12/7/15 BB



12/7/15 BB X-hard5



We got the same answers for the hard data set

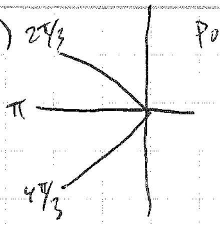
12/7/15

Benjamin Bergeson

LAB 5

TASK 2

1 (a)



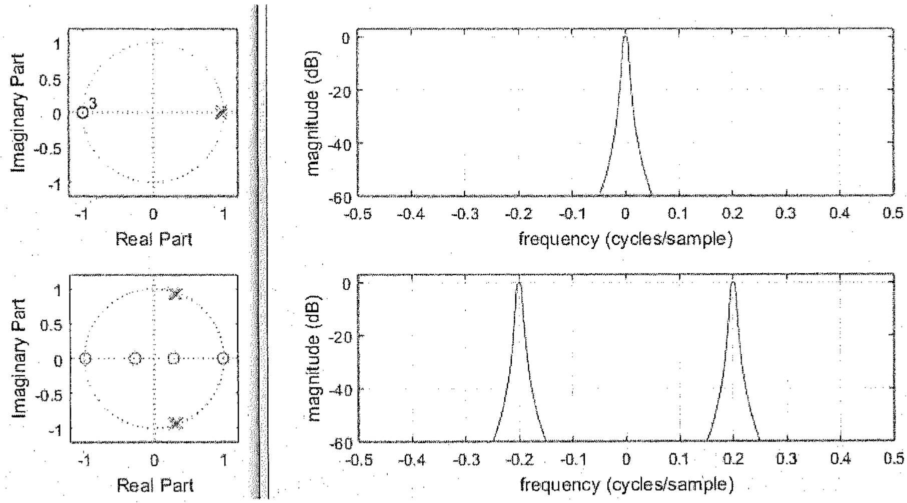
Pole locations of LPF

$\Omega_0 = 2\pi \times 0.2$ rad/sample $W = 2\pi \times 0.01$

$$\frac{(\pi \times 0.01)^3}{(s - W_c e^{j\pi})(s - W_c e^{j2\pi/3})(s - W_c e^{j4\pi/3})} \quad \Big| \quad (\pi \times 0.02)^3 \quad \Big| \quad (\pi \times 0.05)^3$$

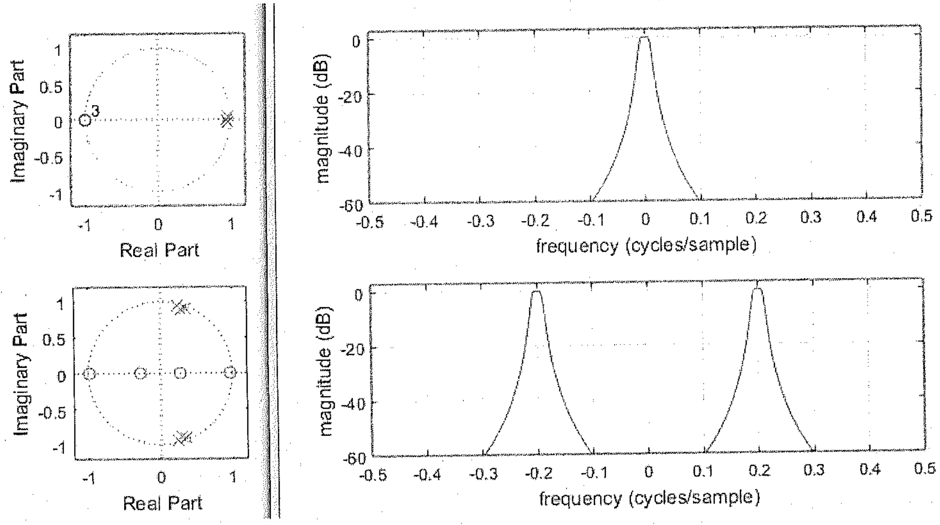
12/7/15 BB

W = 2 * pi * 0.01



12/7/15 BB

W = 2 * pi * 0.02



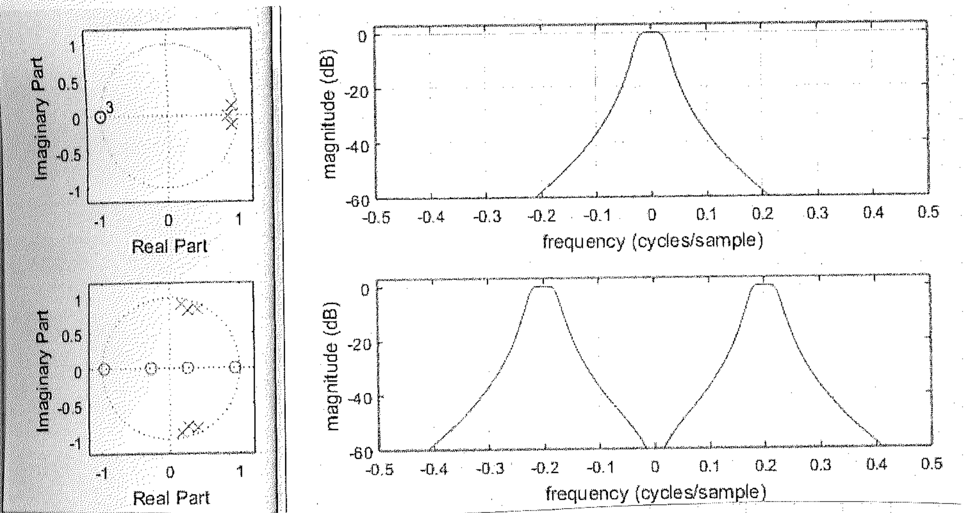
12/7/15

LAB 5

Task 2

W = 2 * pi * 0.05

12/7/15 BB



12/7/15 BB

Code for part (2)

```
n = 3; % LPF filter order
Wc = pi*0.05; % LPF corner frequency
W0 = 2*pi*0.2; % BPF center frequency
[b,a] = butter(n,Wc/pi); % create 3rd order B'worth LPF
aplus = a.*exp(1i*W0*(0:n)); % rotate poles by W0
bplus = b.*exp(1i*W0*(0:n)); % rotate zeros by W0
aminus = a.*exp(-1i*W0*(0:n)); % rotate poles by -W0
bminus = b.*exp(-1i*W0*(0:n)); % rotate zeros by -W0
bb = conv(bplus,aminus) + conv(bminus,aplus); % BPF zeros
aa = conv(aplus,aminus); % BPF poles
aa = real(aa); % eliminate round-off error
figure(1);
subplot(211);
zplane(b,a); % pole-zero plot of LPF
axis(1.2*[-1 1 -1 1]);
subplot(212);
zplane(bb,aa); % pole-zero plot of BPF
axis(1.2*[-1 1 -1 1]);
N = 1024; % # points on unit circle
FF = -0.5:1/N:0.5-1/N; % corresponding freq. axis
H_lpf = freqz(b,a,N,'whole'); % DFT of LPF
H_bpf = freqz(bb,aa,N,'whole'); % DFT of BPF
figure(2);
subplot(211);
plot(FF,20*log10(abs(fftshift(H_lpf)))); % plot DFT of LPF
grid on;
xlabel('frequency (cycles/sample)');
ylabel('magnitude (dB)');
set(gca,'XTick',-0.5:0.1:0.5);
axis([-0.5 0.5 -60 3]);
subplot(212);
plot(FF,20*log10(abs(fftshift(H_bpf)))); % plot DFT of BPF
grid on;
xlabel('frequency (cycles/sample)');
ylabel('magnitude (dB)');
set(gca,'XTick',-0.5:0.1:0.5);
axis([-0.5 0.5 -60 3]);
```

12/7/15

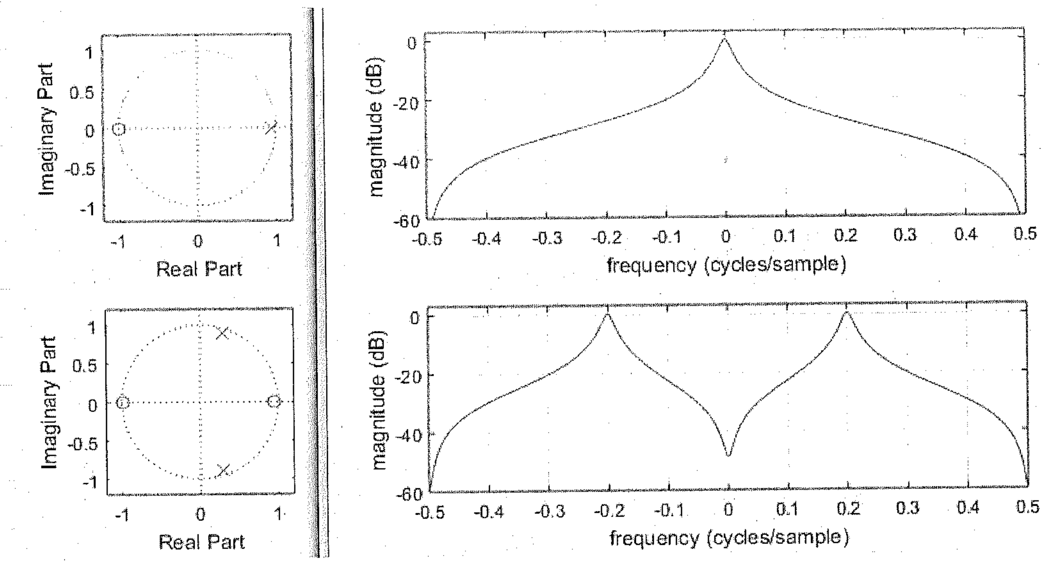
LAB5

Task 2

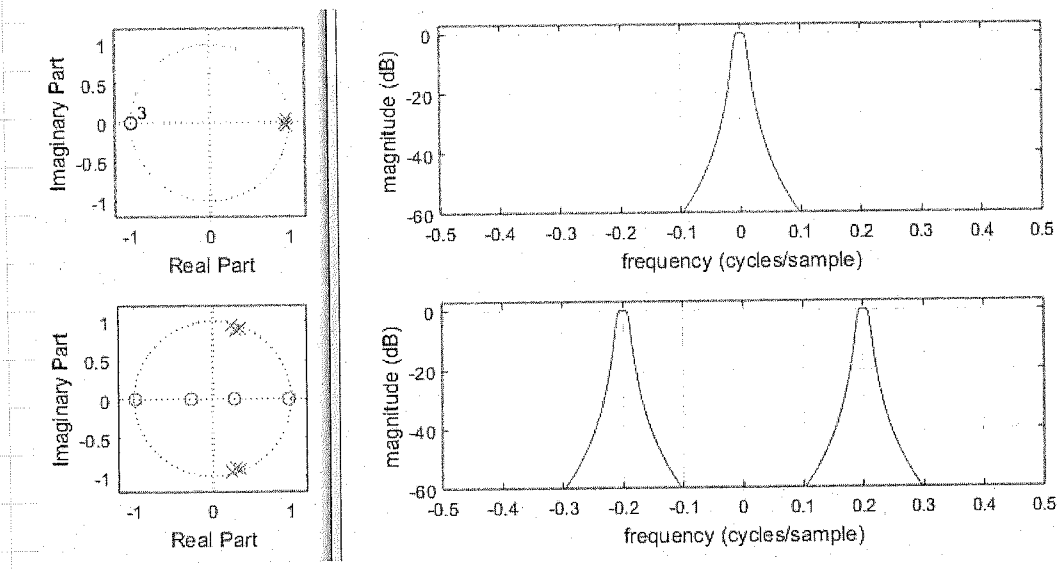
1(b)

2nd order

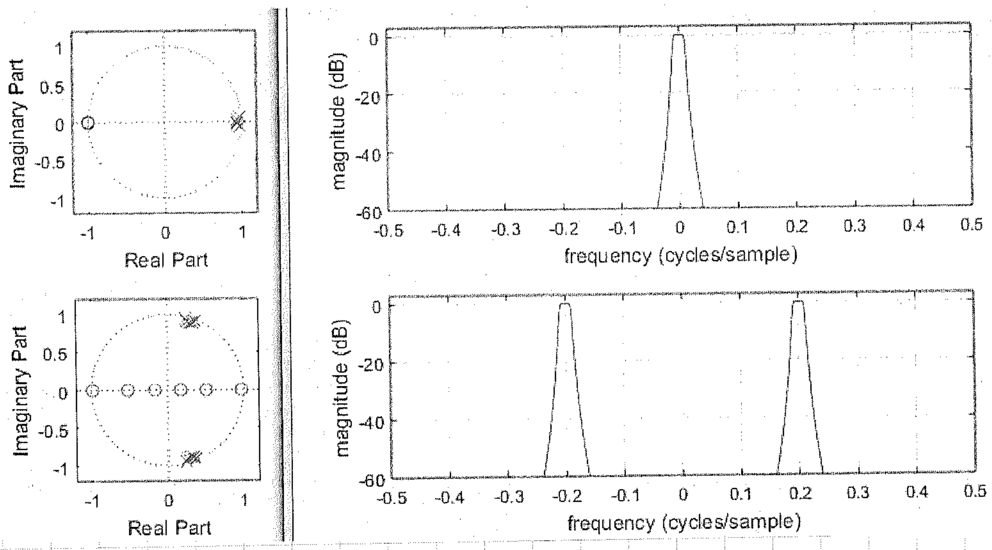
12/7/15 BB



6th Order



10th Order



12/7/15

LAB 5

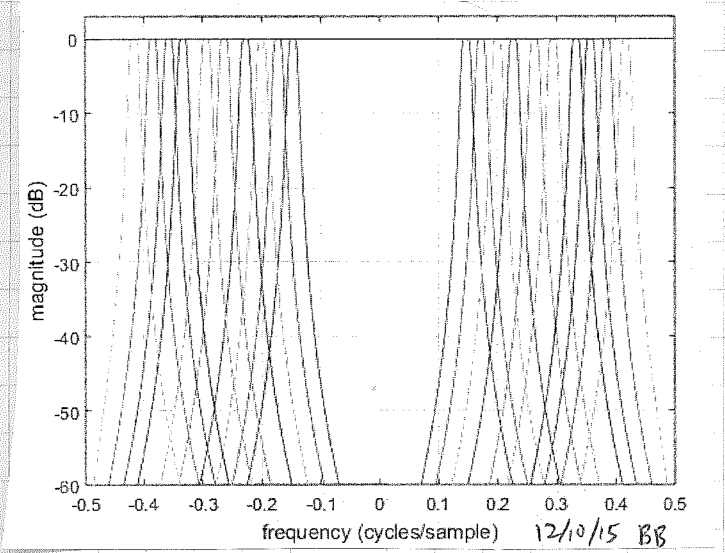
Task 2

1(c) The higher the order, the ~~higher~~ more the filter looks like a brick wall filter. The smaller the bandwidth, the more it looks like a brickwall filter. So the goal would be to pick a bandwidth and an order that would make the most brickwall-like filter without losing too much bandwidth.

Complexity: Σ of multiplications in FIR: 201

Σ of multiplications in IIR: 14

2. For this part we decided to use Butterworth filters. After some trial and error we decided to use a 6th order filter with a bandwidth of 0.005. After doing so, we ran the data sets through our filter and got the following outputs.



12/10/15 BB

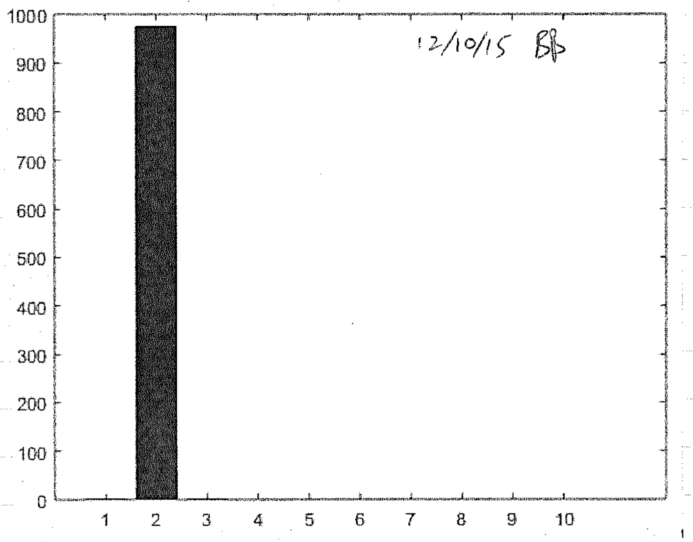
All ten filters on the same plot.

12/10/15

Benjamin Bergeson

LAB 5

Task 2

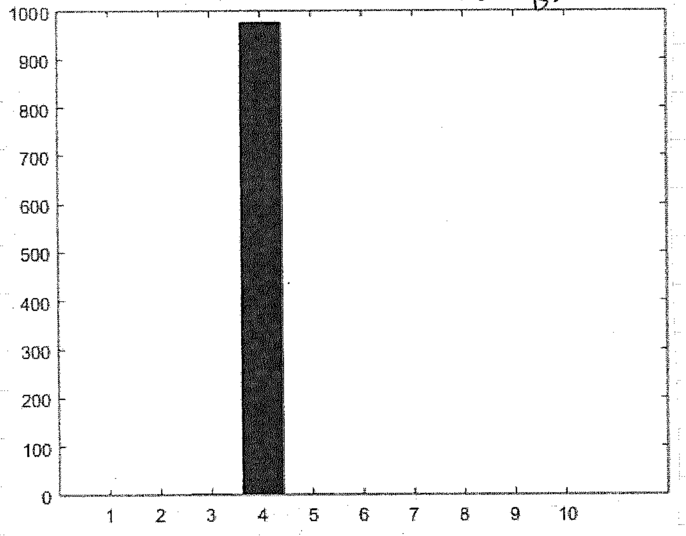


12/10/15 BB

x easy1

player 2

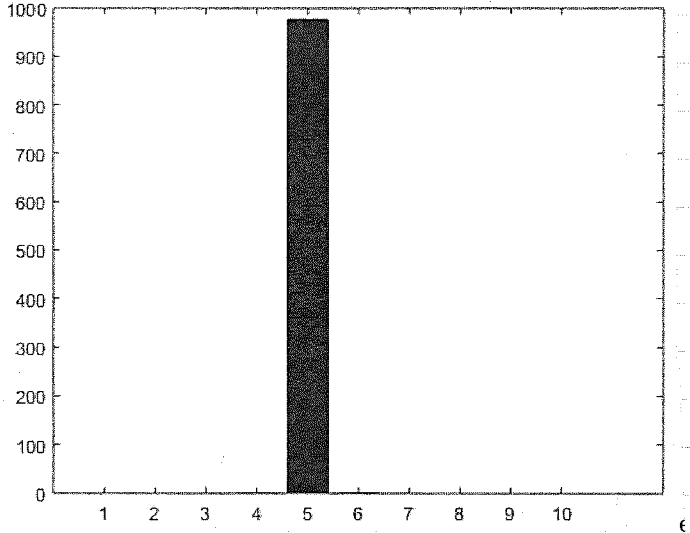
12/10/15 BB



x easy 2

player 4

12/10/15 BB



x easy 3

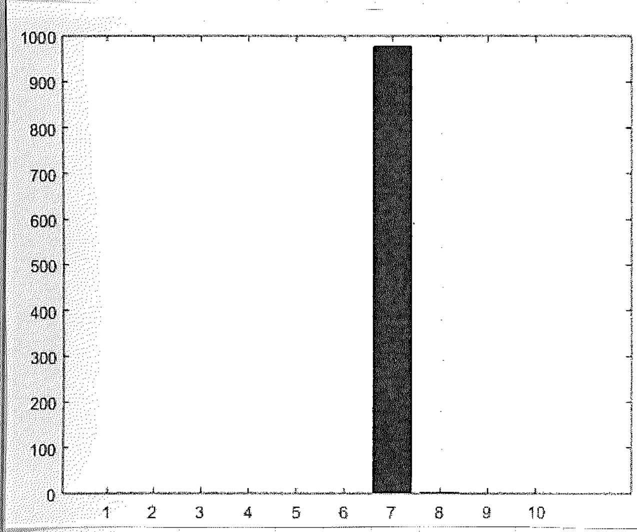
player5

12/10/15

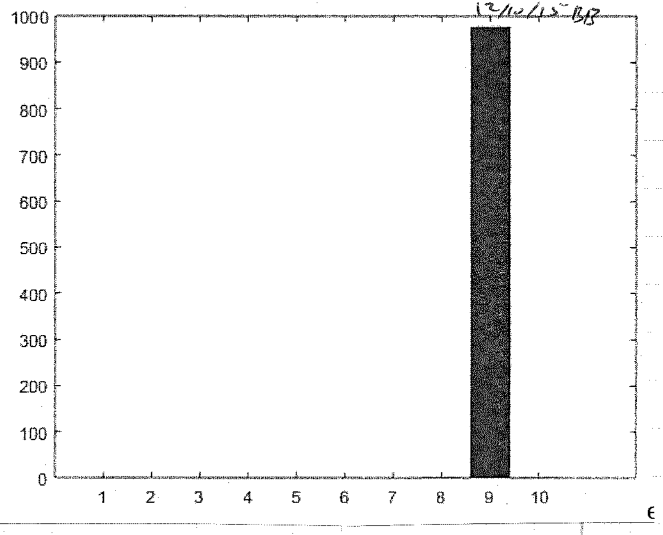
LAB 5

Task 2

xeasy 4
player 7

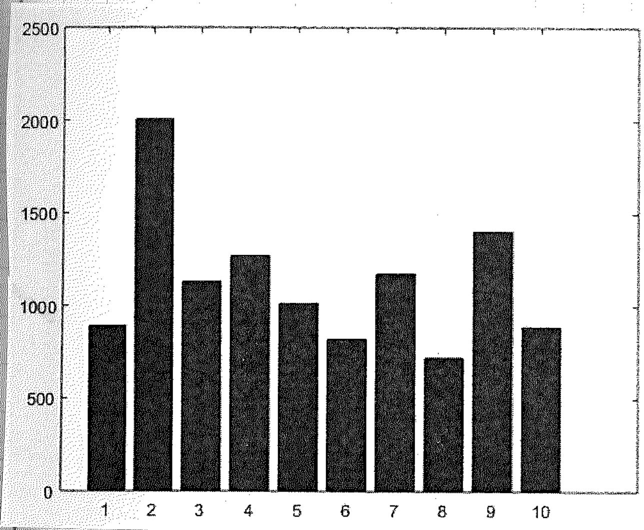


Xeasy 5
player 9

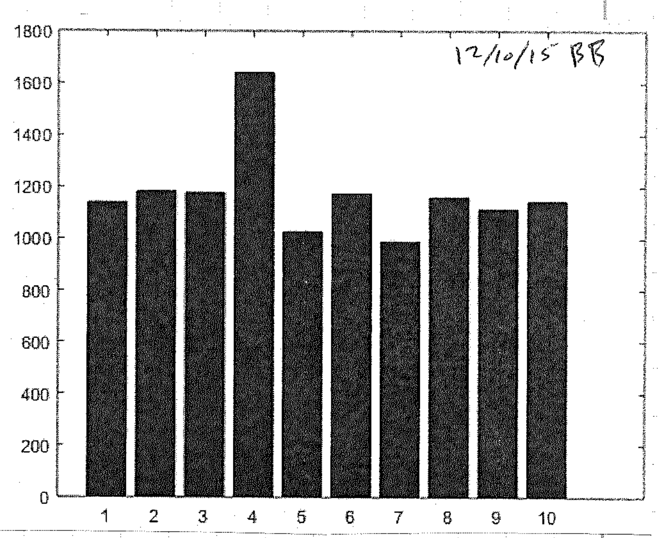


12/10/15 BB

~~xeasy~~ Xhard 1
player 2

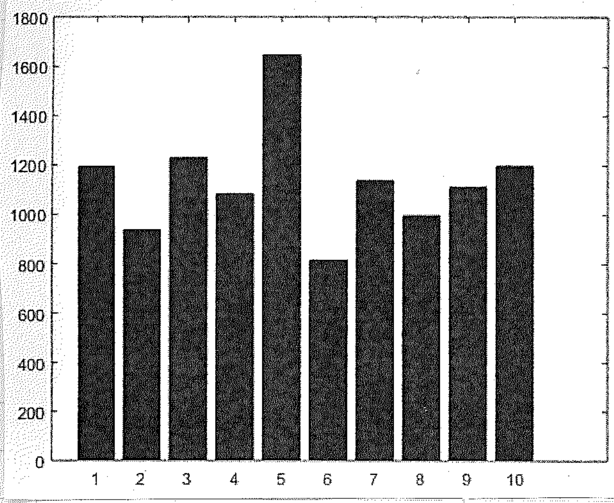


Xhard 2
player 4

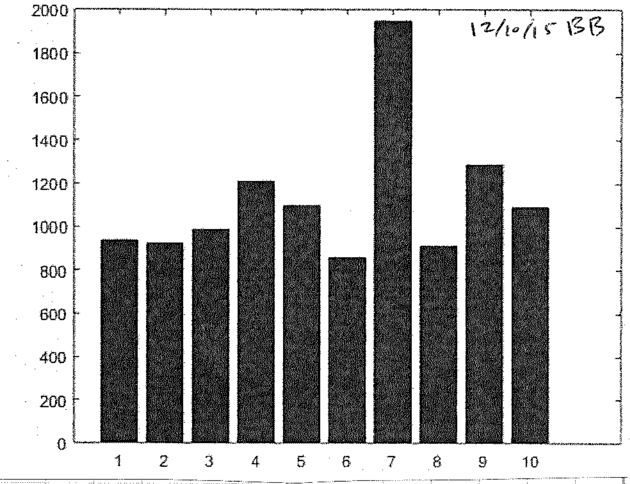


12/10/15 BB

x hard 3
player 5



xhard 4
player 7

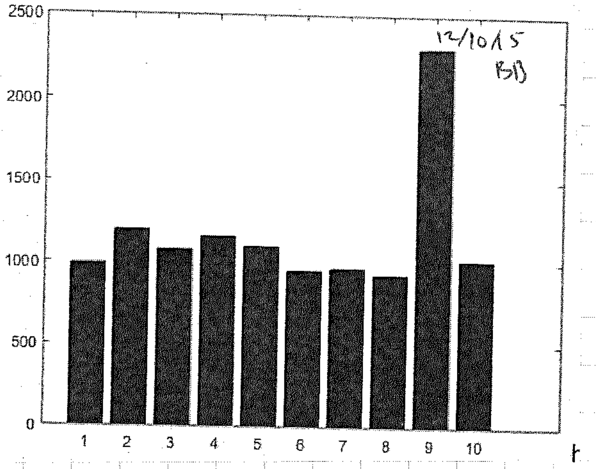


12/10/15 BB

12/10/15
Benjamin Bergeron

LAB 5

Task 2



Xhard 5
player 9

Complexity

The Σ of multiplications in the FIR = 201

~~Ea~~

The Σ of multiplications in the IIR = 14

Based on the total number of multiplications in each filter, it's easy to see that the IIR filter is much less computationally heavy.

Conclusion

Conclusion

In this lab we designed two different filters in order to accomplish the same goal of having small bandwidths to filter out noise so that the frequency of the light detected can be determined.

In task 1, we designed a FIR filter based on the instructions given to us. Using MATLAB

12/10/15
Benjamin [illegible]

Conclusion

We were able to plot the frequency response of the three windows that we randomly picked. From these plots, we decided to use the Blackman-Harris filter because of how clean its frequency response was. Based on the different outputs we got when we used different parameters in the filter we were able to see the tradeoffs involving bandwidth and length.

We had to make many little adjusements in our filter design so that the filter would pass through just barely large enough bandwidth so that no ~~frequency~~ legitimate frequency would be missed. At the same time we didn't want the filter to pass through the wrong frequencies, as this would mess up the algorithm to figure out the shooter.

We repeated similar steps for the IIR filter. However the IIR did take more fine tuning to get it to do what we wanted it to.

Some of the things that we learned from this lab are: to read the help from

12/10/15

LAB 5

Conclusion.

MATLAB thoroughly, since sometimes there is more than one way to use a function. We also learned that MATLAB has a lot of very useful functions built in. It doesn't hurt to check if there are functions that accomplish what you want so that you don't have to write the code yourself.

12/10/15

Benjamin Bergeson