ADAPTIVE FILTERING

ESE 531 FINAL PROJECT

Dinesh Jagai
Department of Computer Science
University of Pennsylvania
Philadelphia, PA
dinesh97@seas.upenn.edu

Pranav Panganamamula
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA
ppranav@seas.upenn.edu

Tuesday April 30^{th} 2020

N.B. The explanation is given by detailed comments in our code.

1 Part A

1.1 a

The baseline model involves a desired sinusoidal signal with frequency given by $\omega = 0.6\pi$ and a strong interference sinusoidal signal with a with a frequency of 0.6π
Different parameters of and r were tried until the optimal were found and the interference was filtered out. This is shown in figures A.1 and A.2.
Upon getting this simple baseline working, parameter for its power, the interfering frequency, , r and the desired signal were all varied to make inferences.

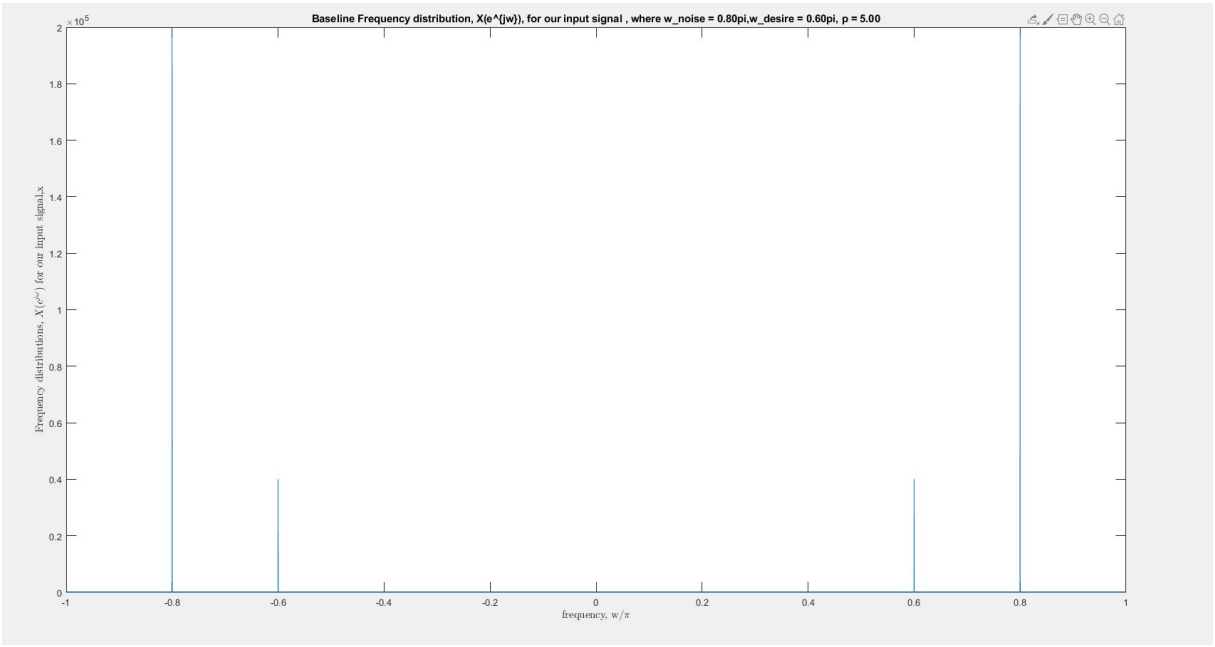


Figure B.1: Baseline Model Input Signal)

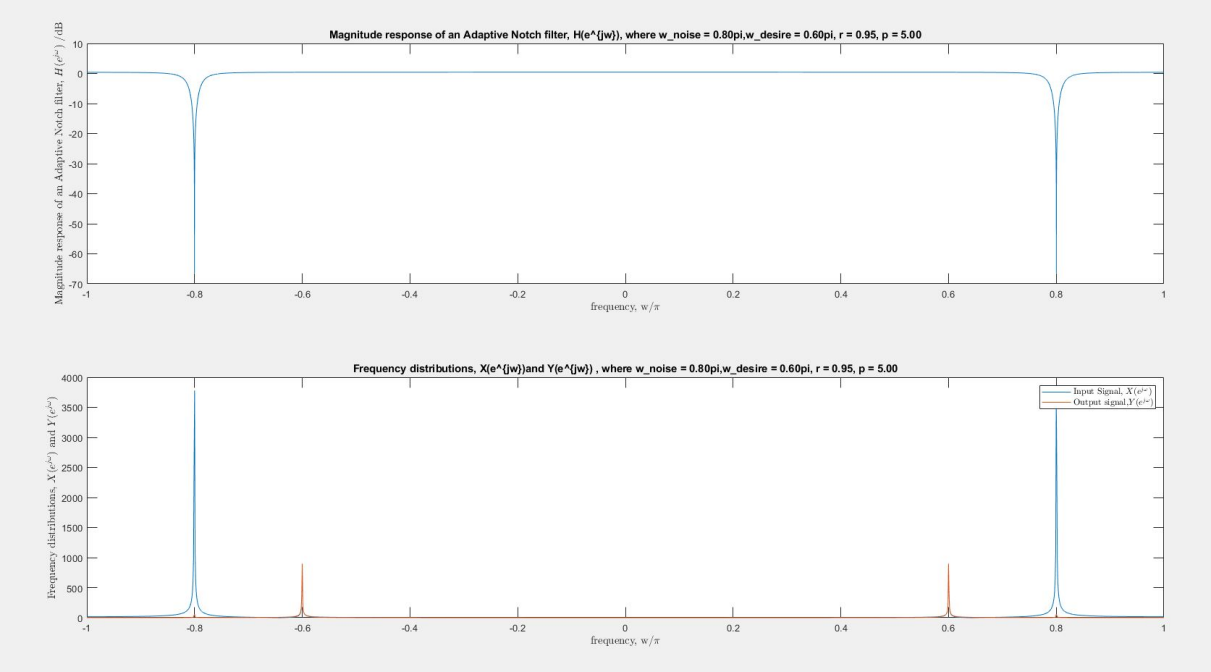


Figure B.2: Baseline Model Filtered Signal)

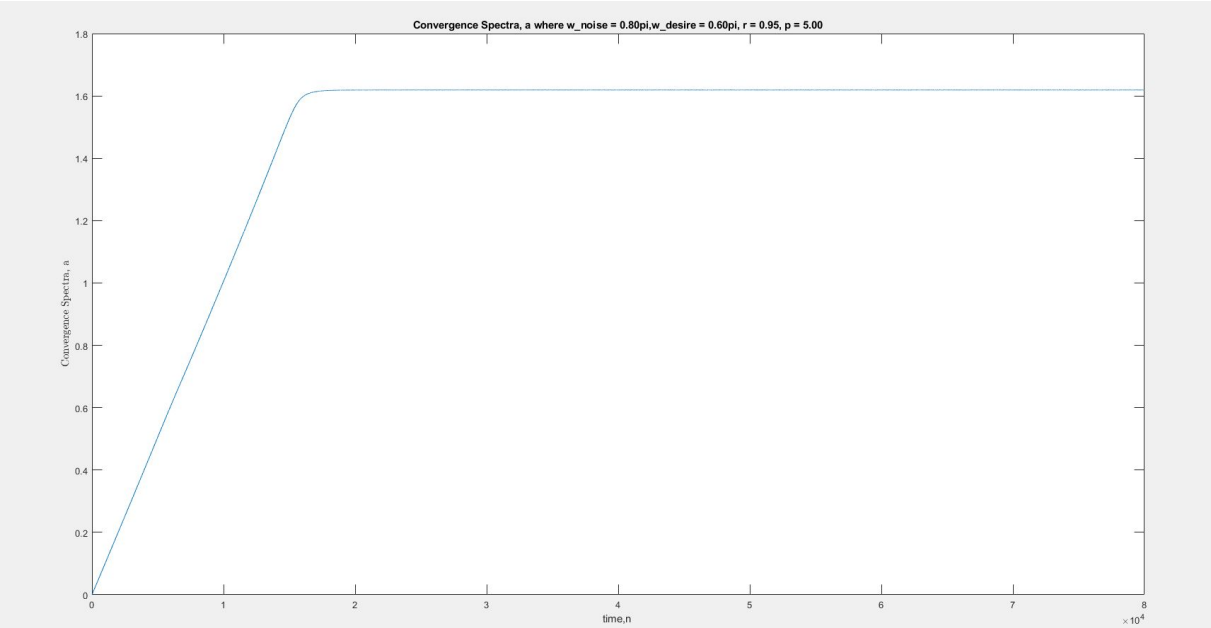


Figure B.3: Baseline Model Convergence)

Note the our adaptive filter is working and it is clearly filtering out the strong noise inference signal

Tuning parameters 1 - Changing the frequency of the interference Signal

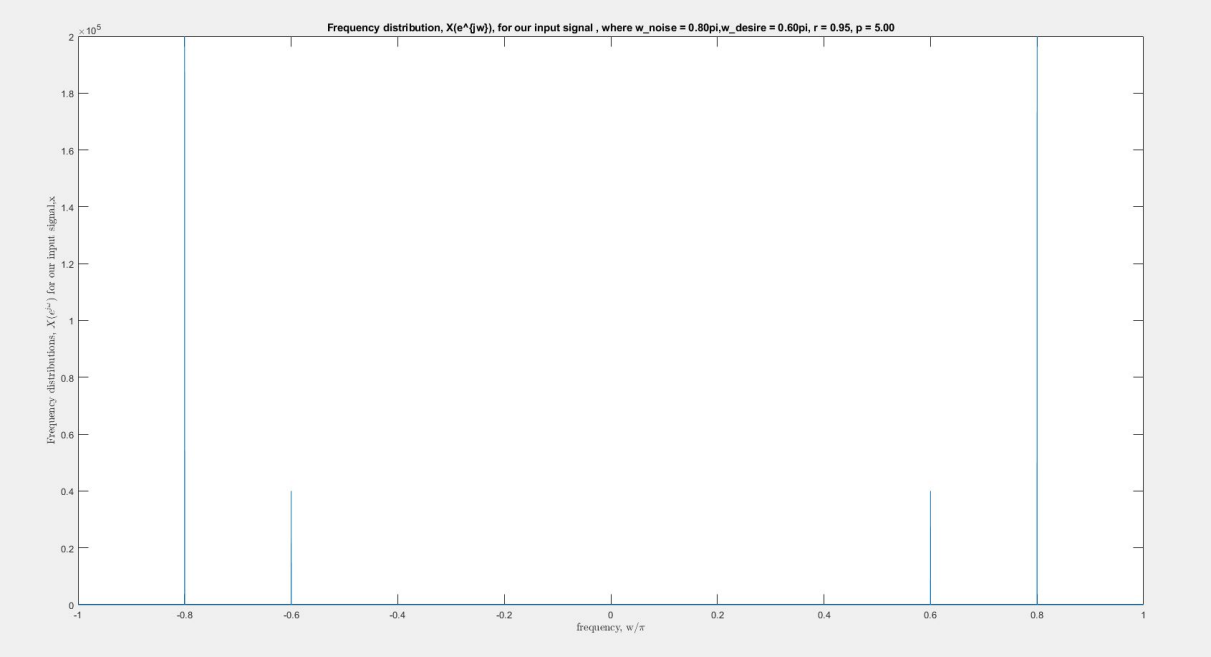


Figure B.4: Input Signal of Model f$_{noise} = 0.40, mu = 0000090, r = 0.95$

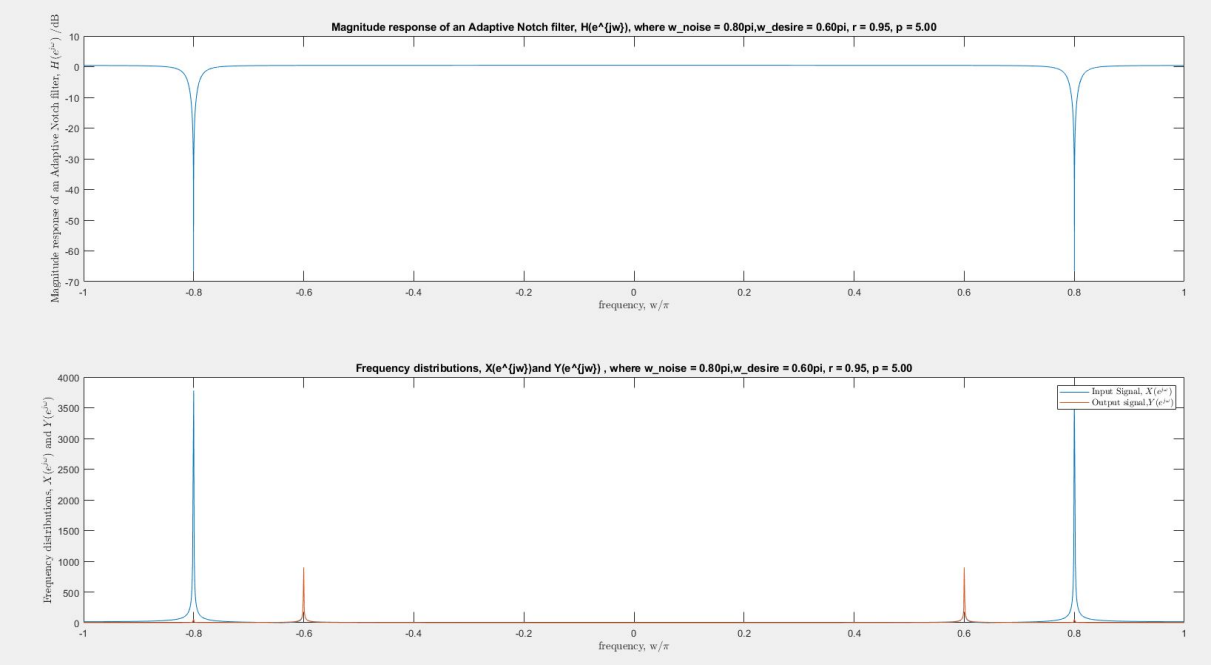


Figure B.5: Filtered Signal of Model f$_{noise} = 0.40, mu = 0000090, r = 0.95$

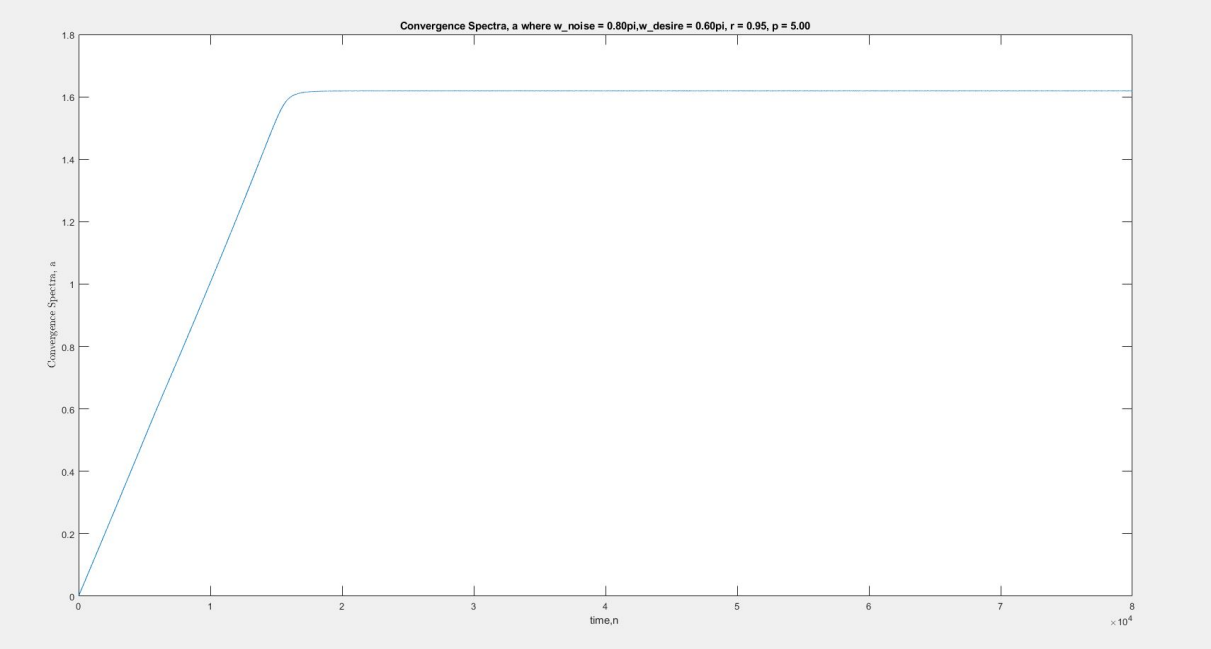


Figure B.6: Convergence of a plot f$_{noise} = 0.40, mu = 0000090, r = 0.95$

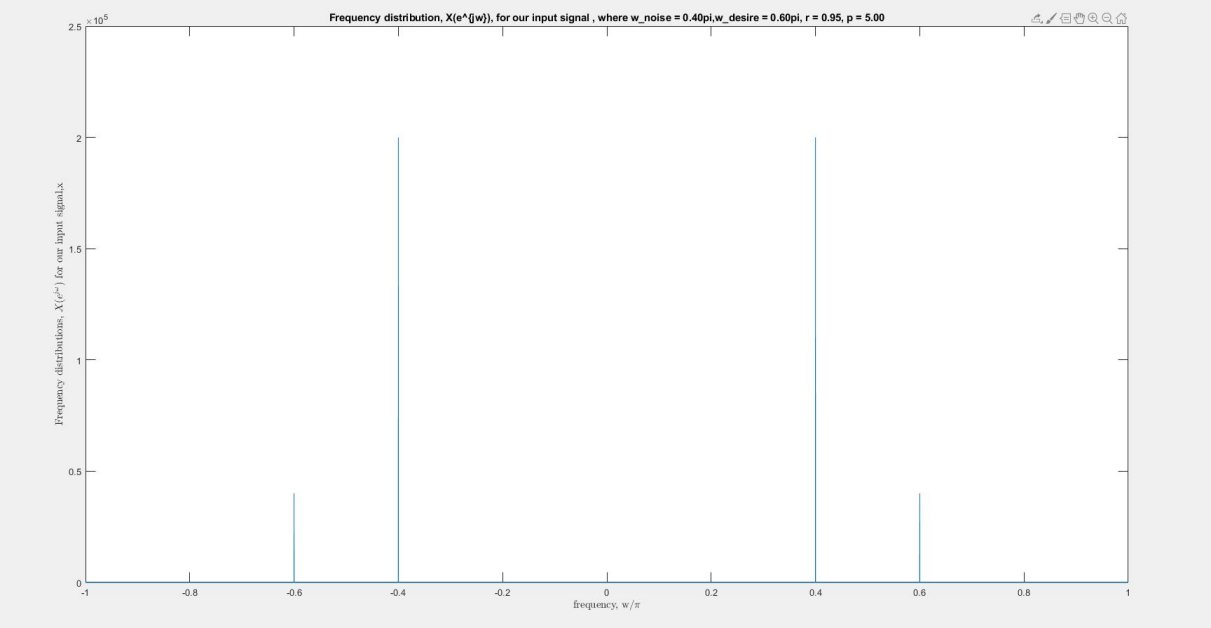


Figure B.7: Input Signal of Model f$_{noise} = 0.20, mu = 0000090, r = 0.95$

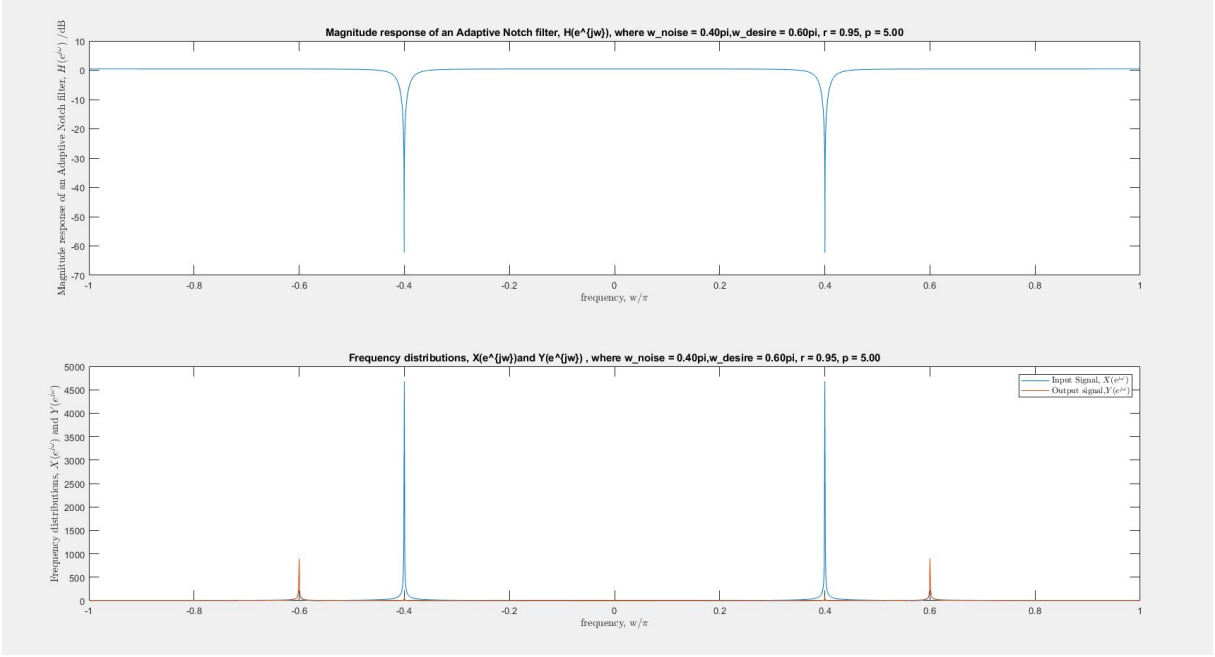


Figure B.8: Filtered Signal of Model f$_{noise} = 020, mu = 0000090, r = 0.95$

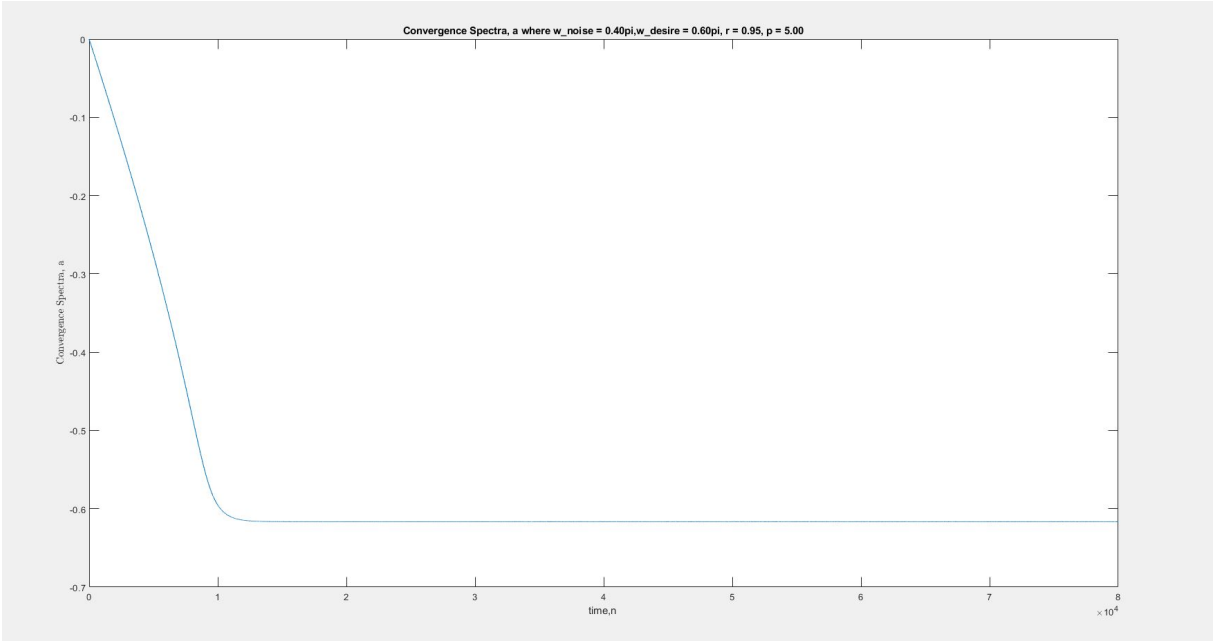


Figure B.9: Convergence of a plot f$_{noise} = 0.20, mu = 0000090, r = 0.95$

Figure B.10: Input Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.95$

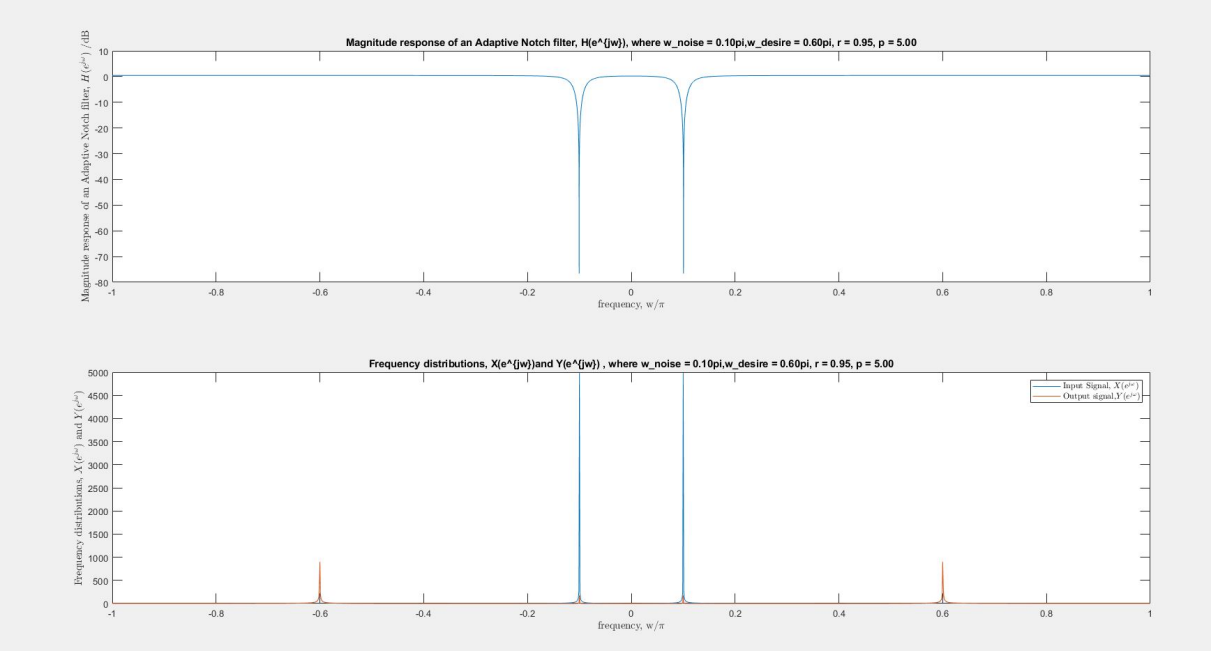


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.95$

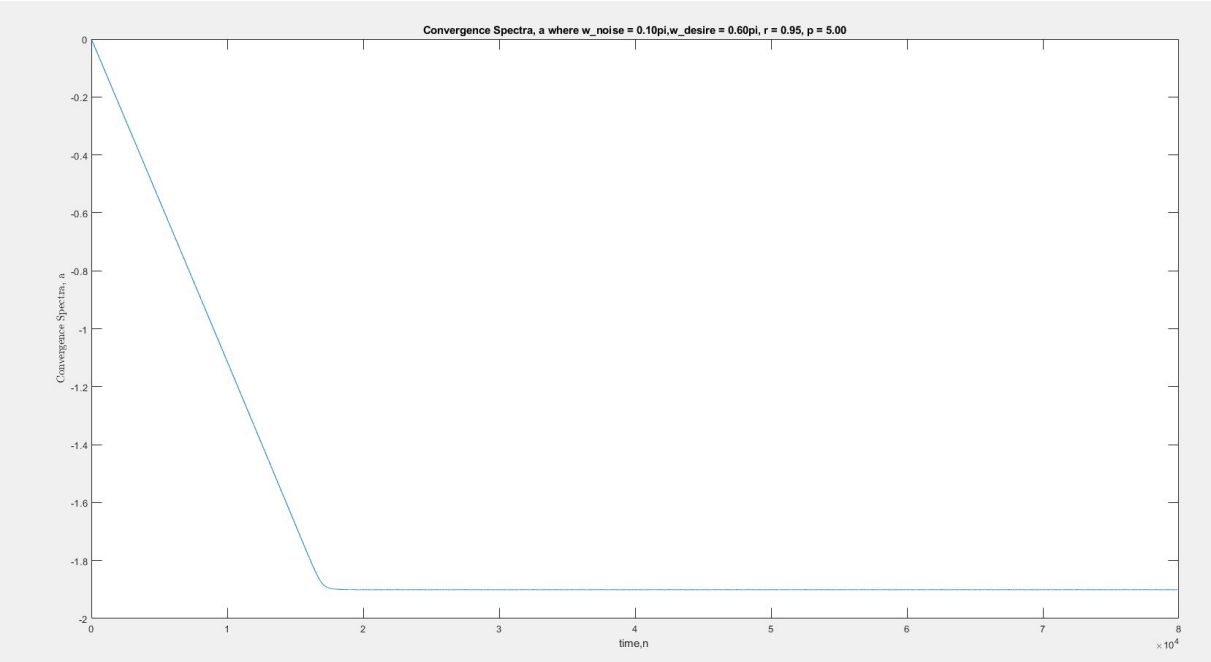


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.95$

Tuning parameters 2 - Changing the r in the Signal

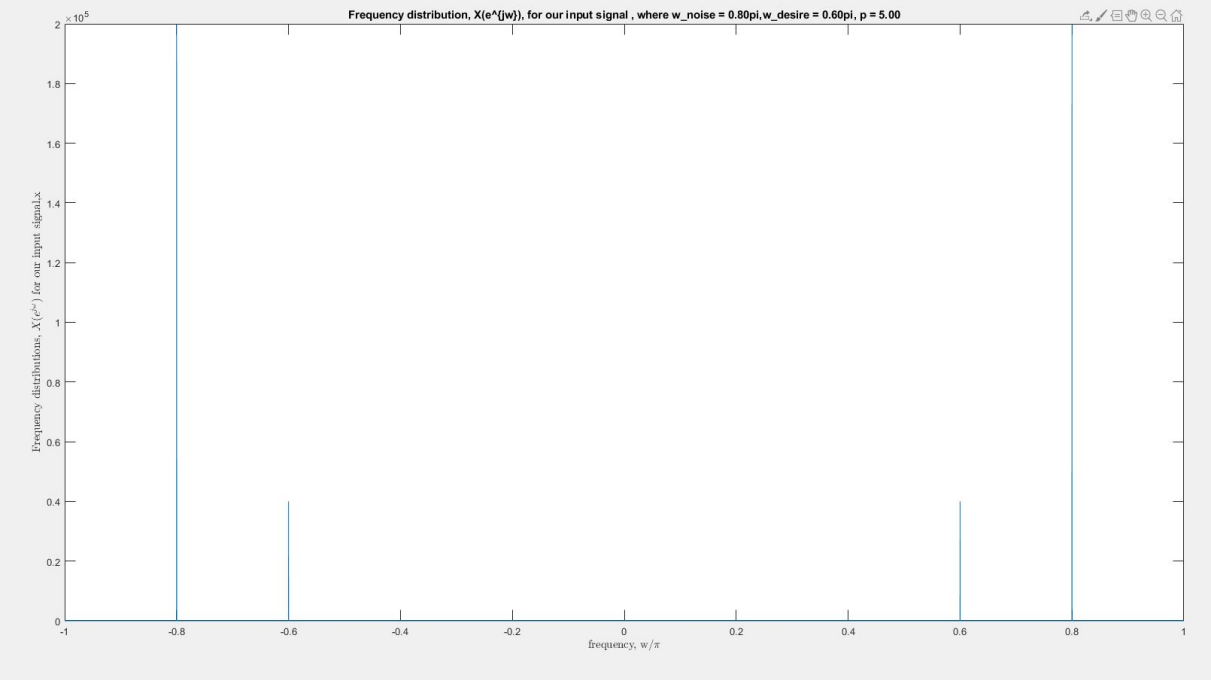


Figure B.10: Constant Input Signal of Model f$_{noise} = 0.40, mu = 0000090$

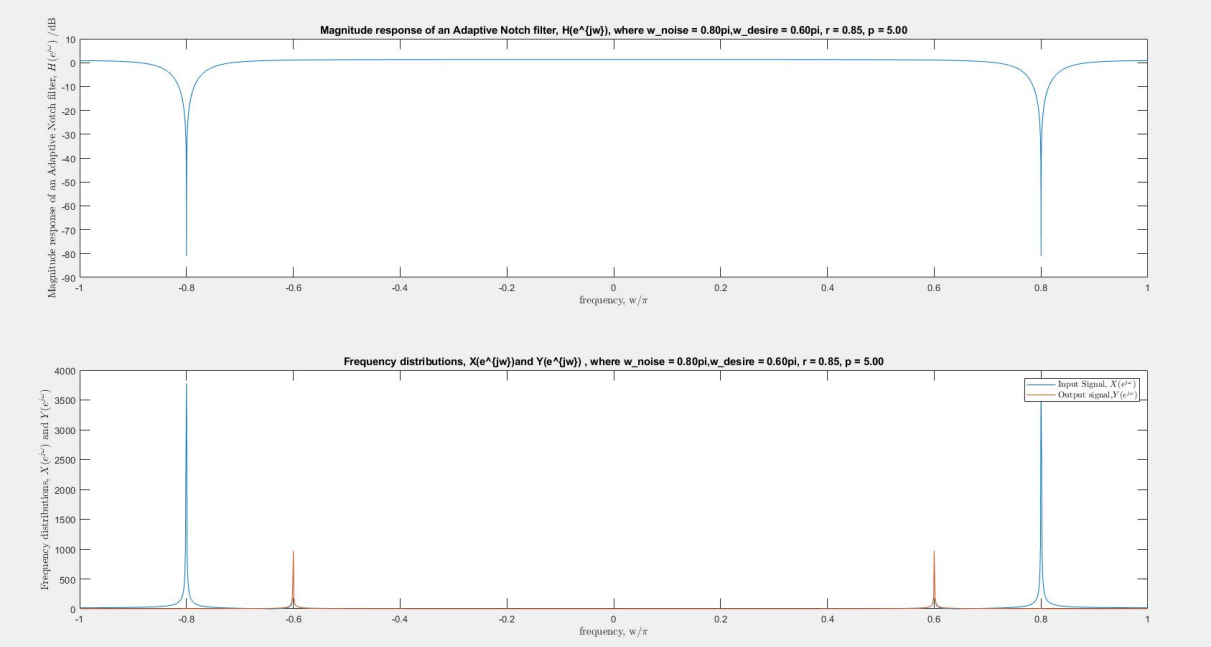


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.85$

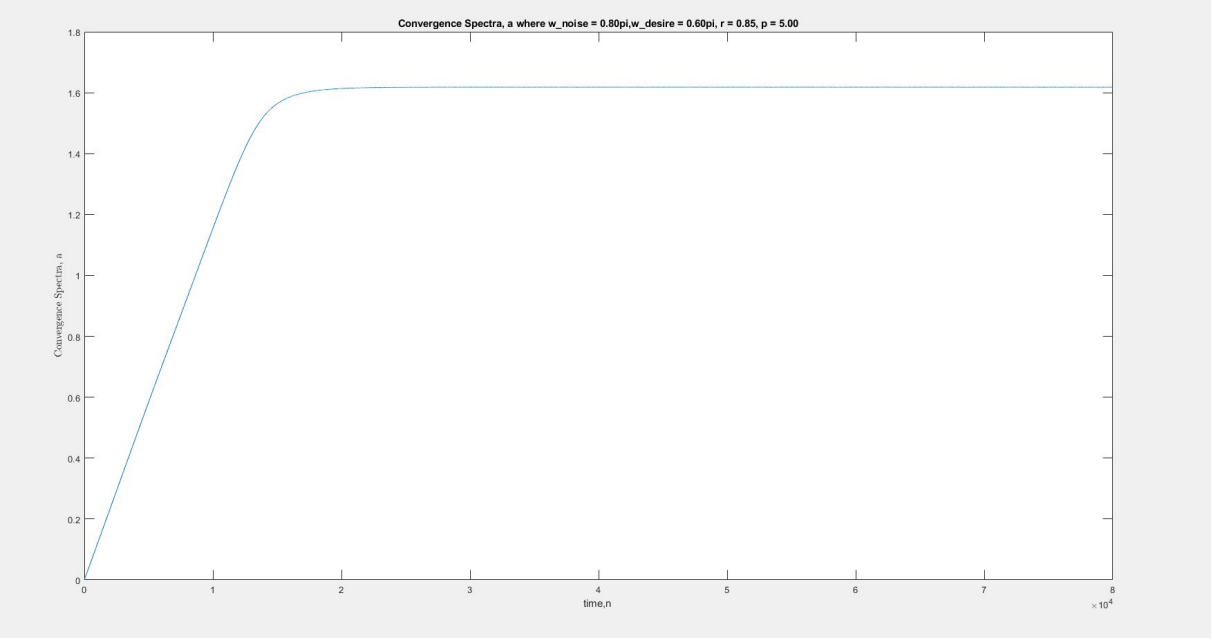


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.85$

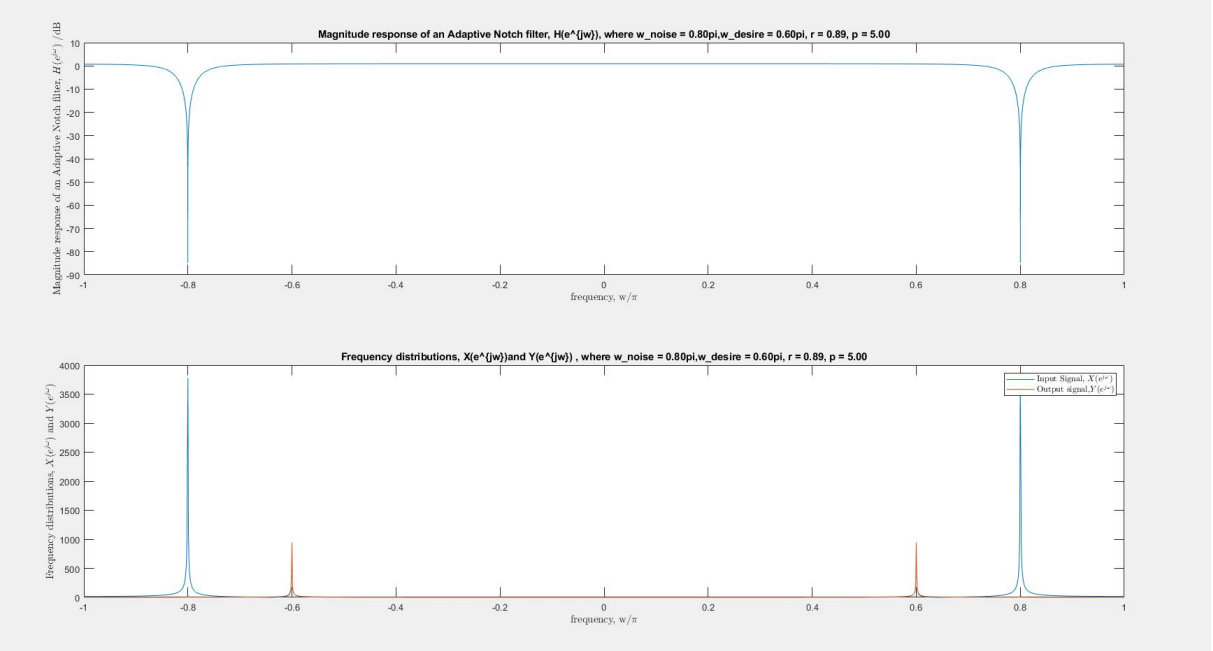


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.89$

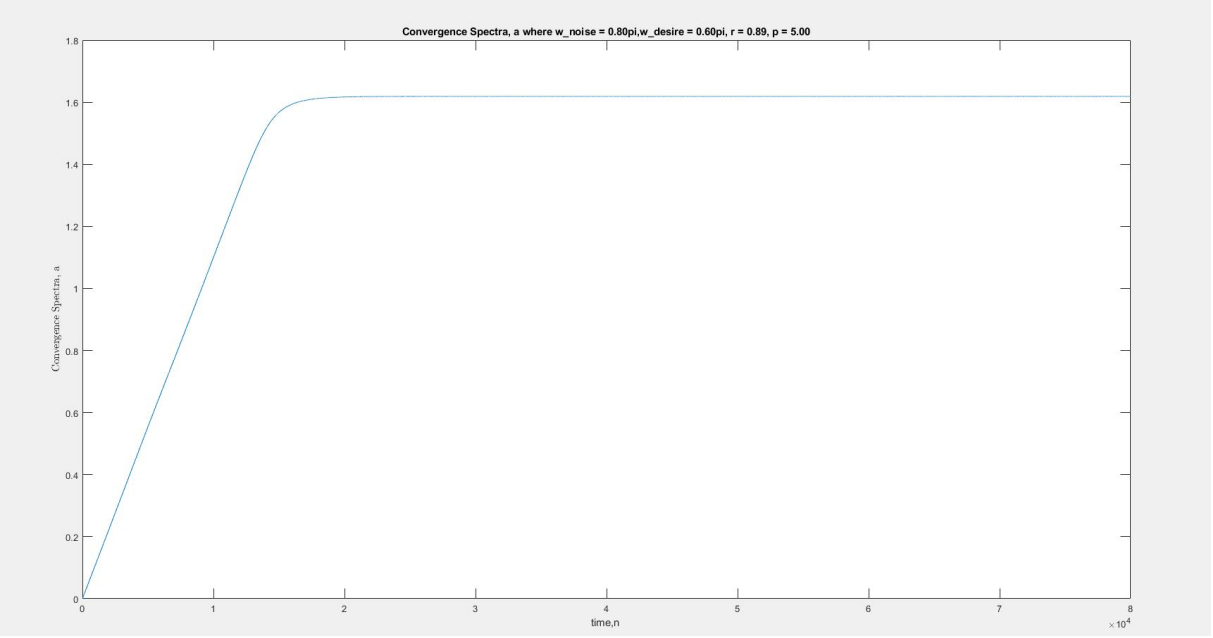


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.89$

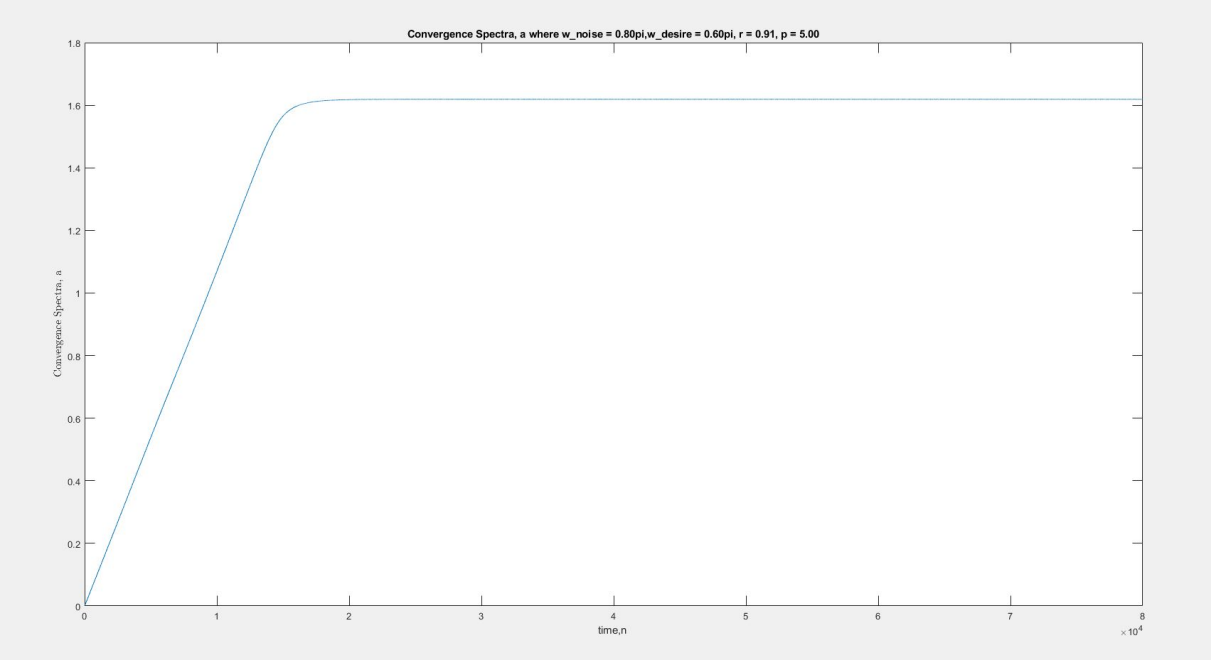


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.91$

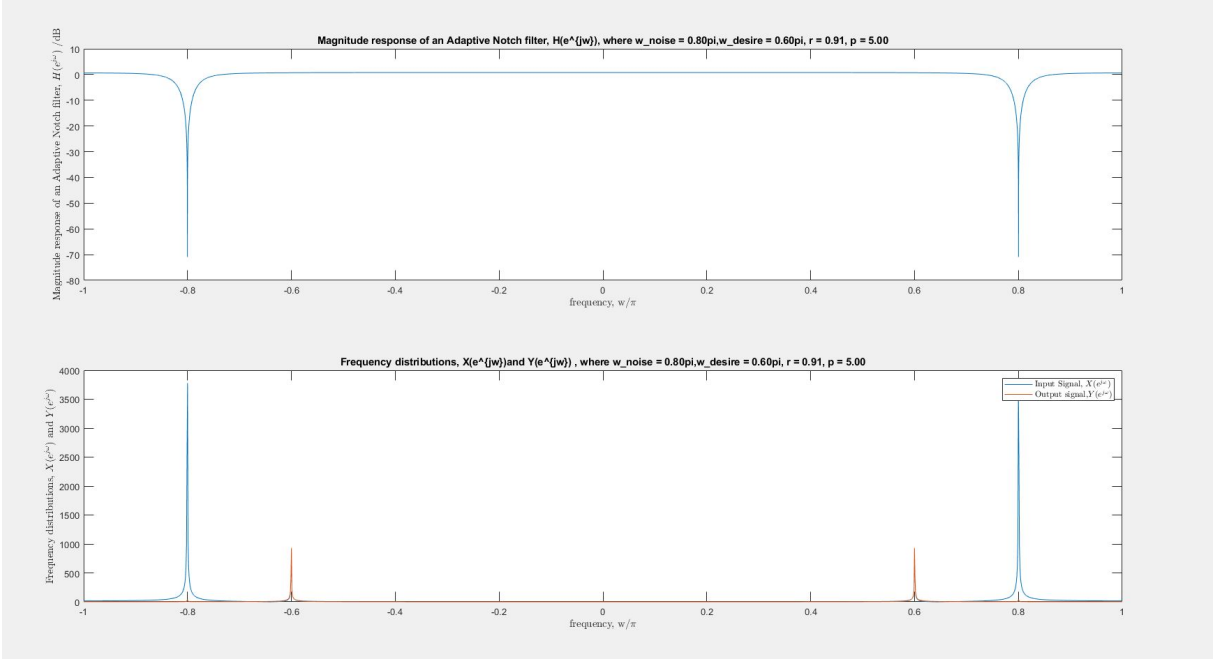


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.91$

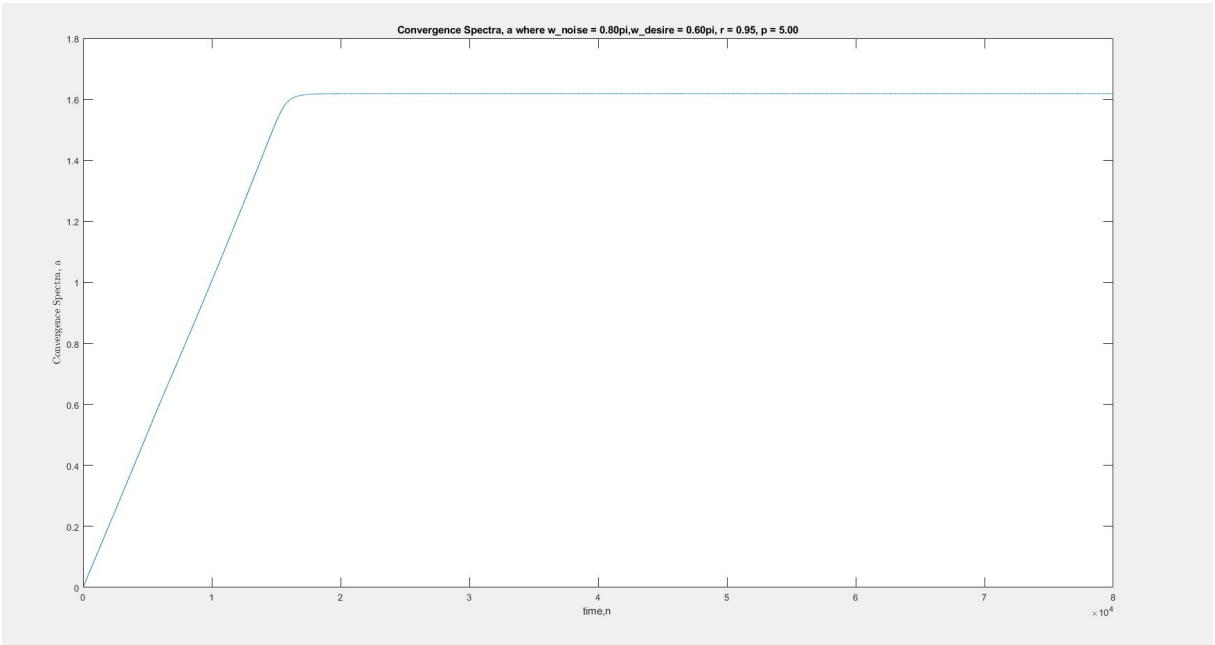


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.95$

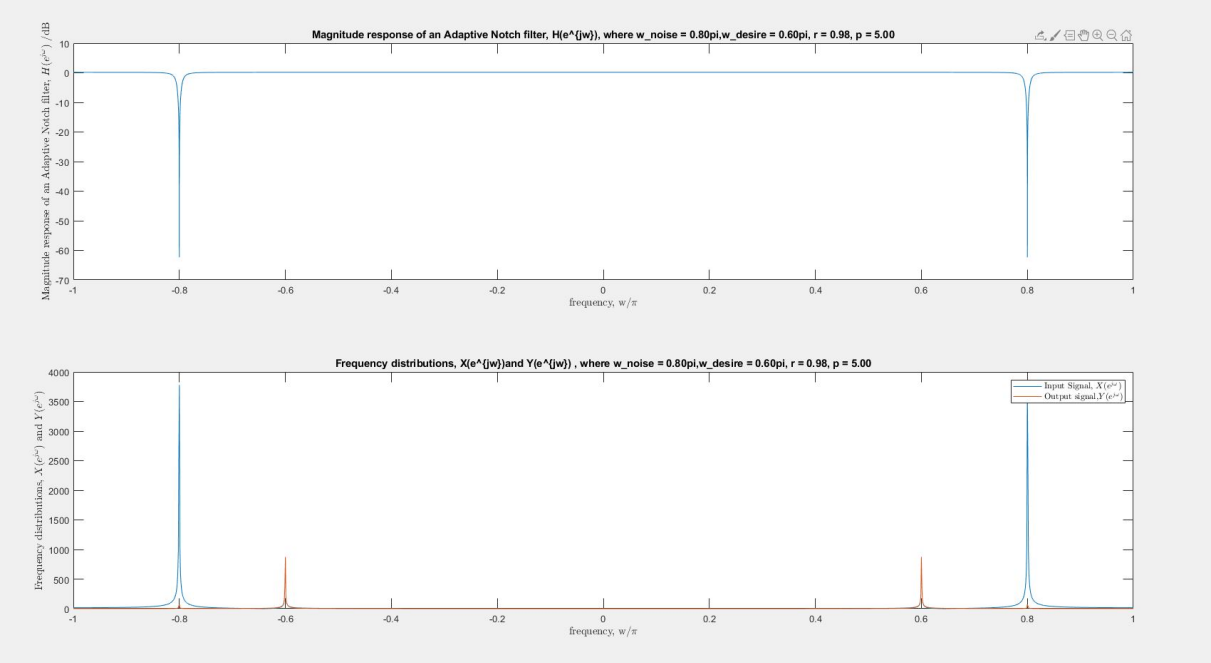


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.91$

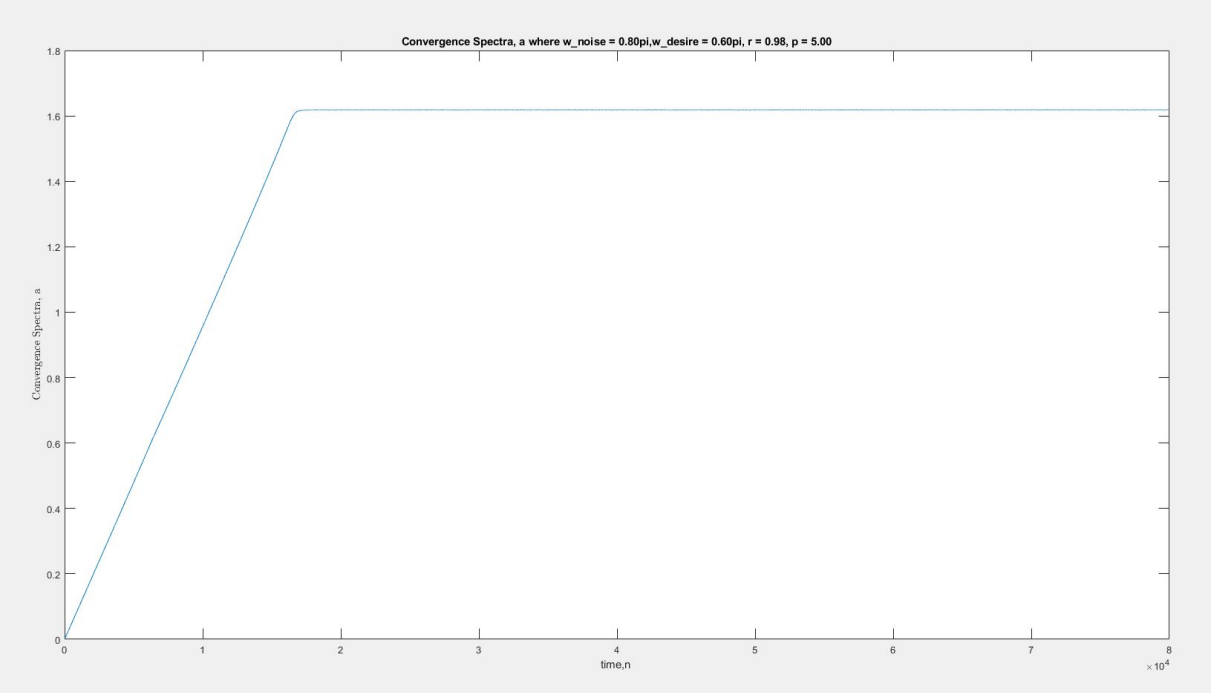


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.95$

Tuning parameters 3 - Changing the mu in the Signal

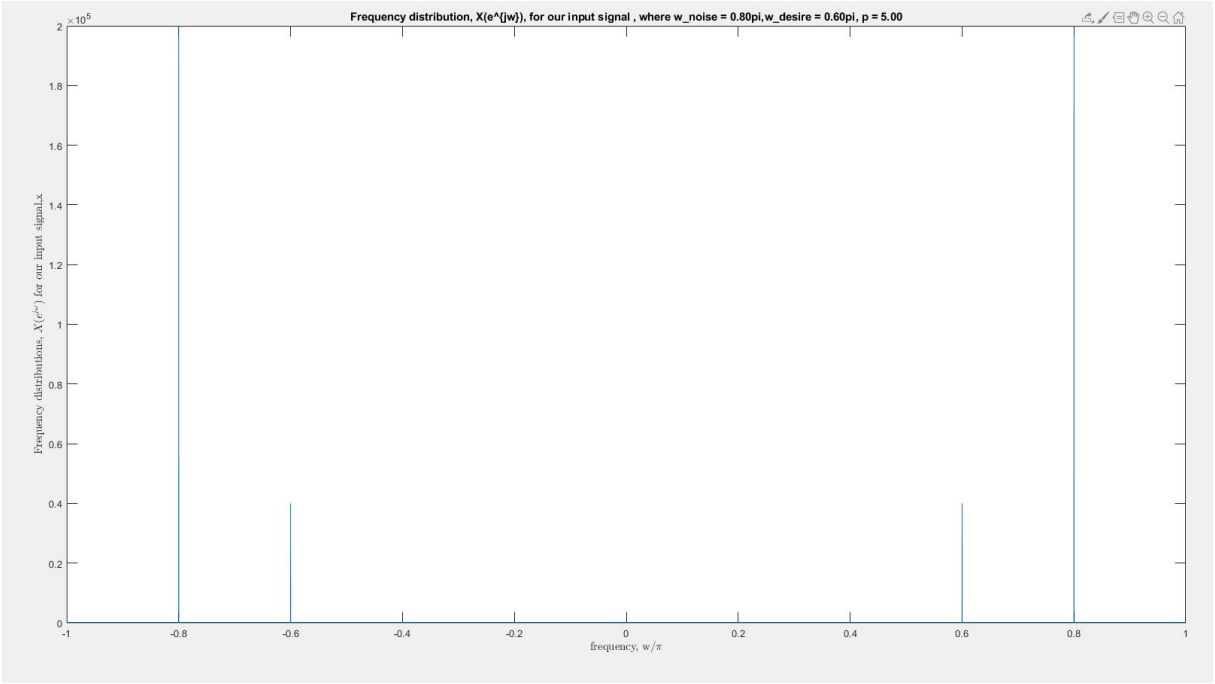


Figure B.10: Constant Input Signal of Model f$_{noise} = 0.40, mu = 0000090$

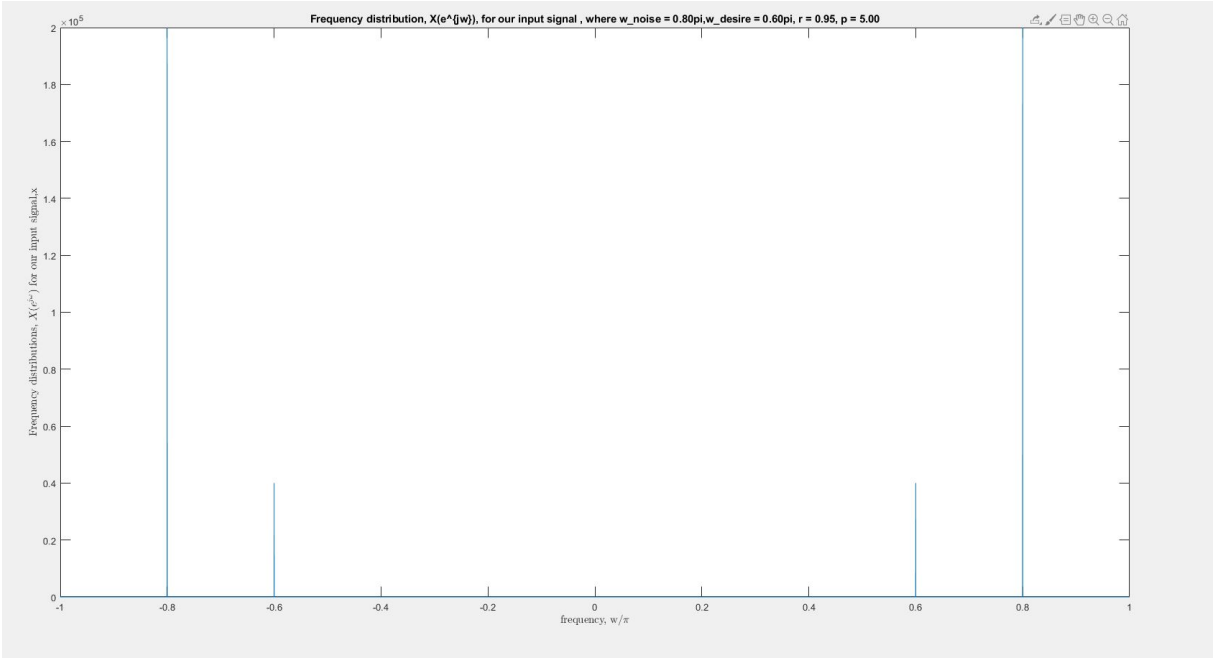


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.91$

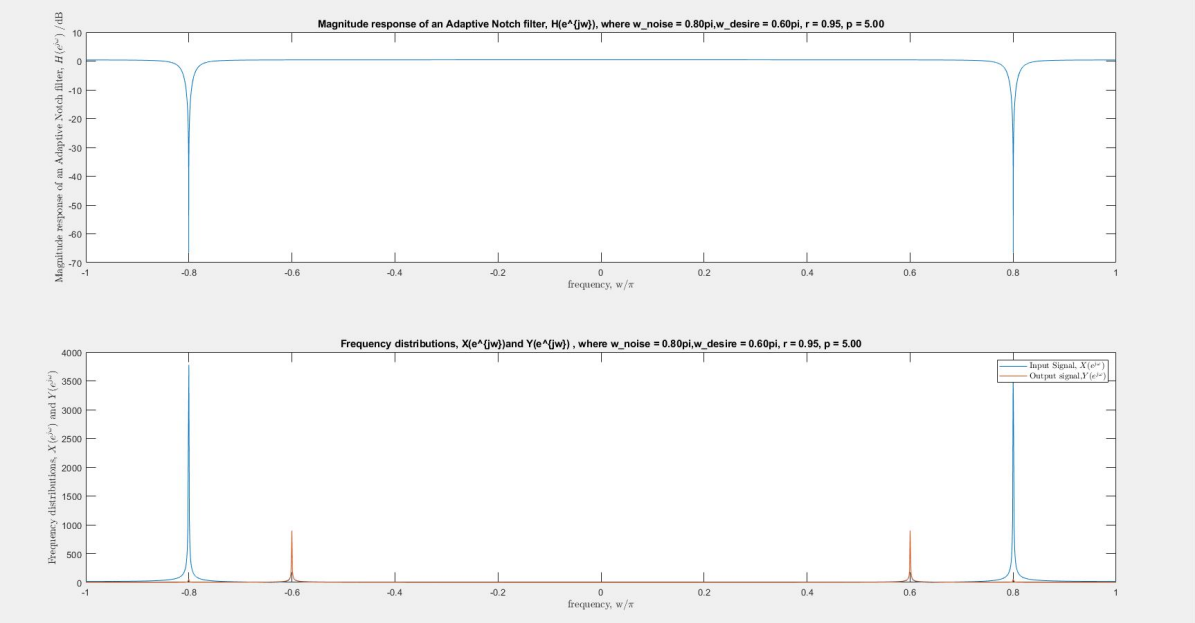


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0000090, r = 0.95$

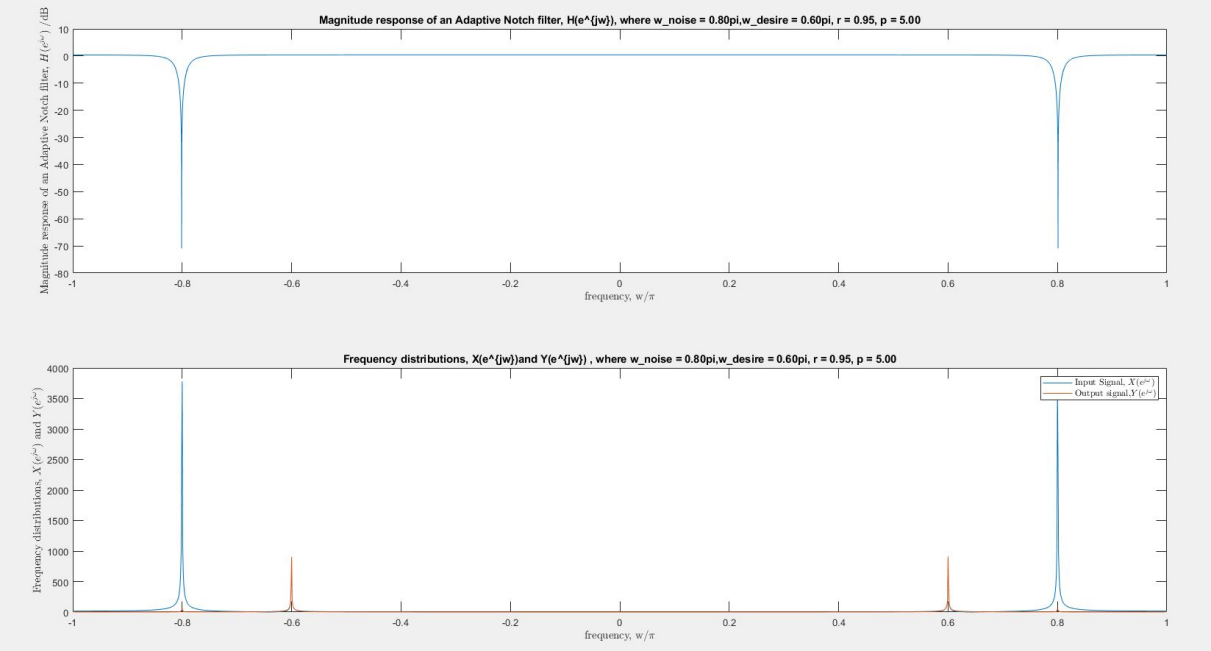


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0000090, r = 0.91$

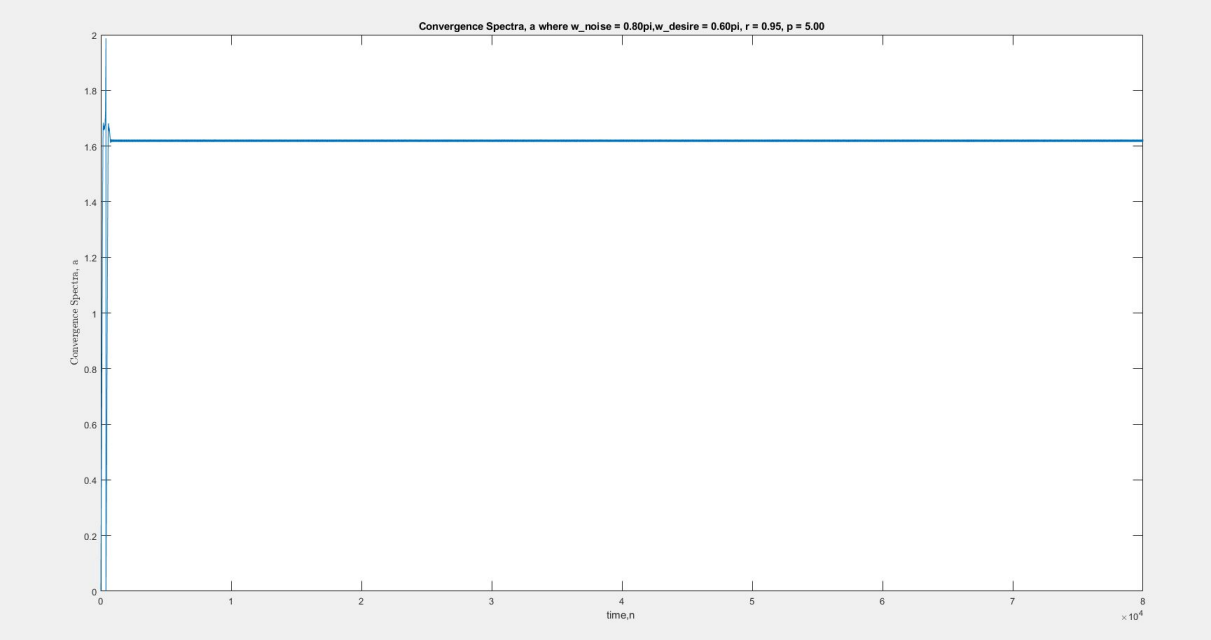


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0009000, r = 0.95$

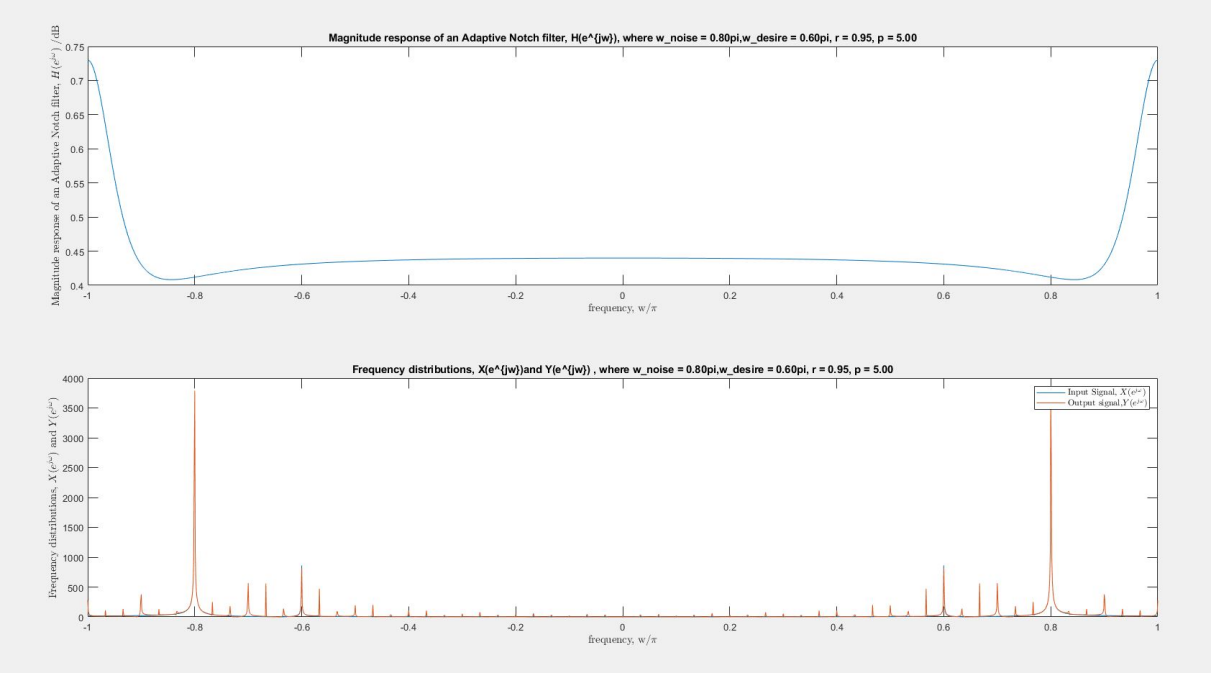


Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0090000, r = 0.91$

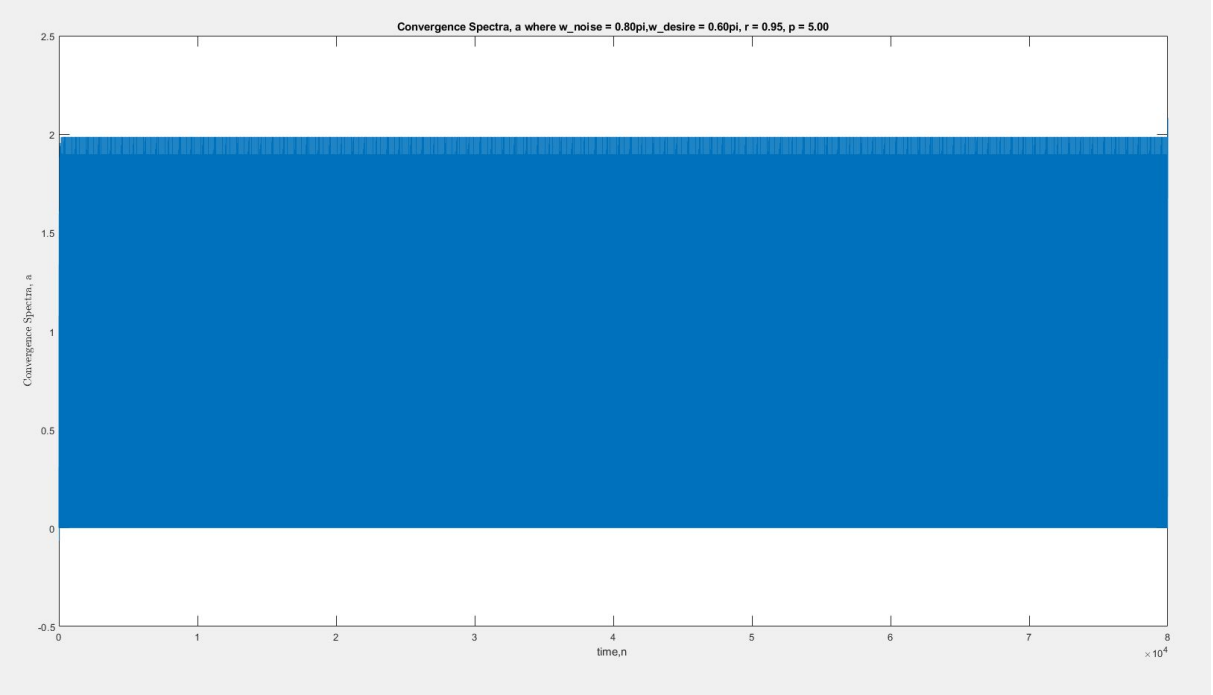


Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0090000, r = 0.95$

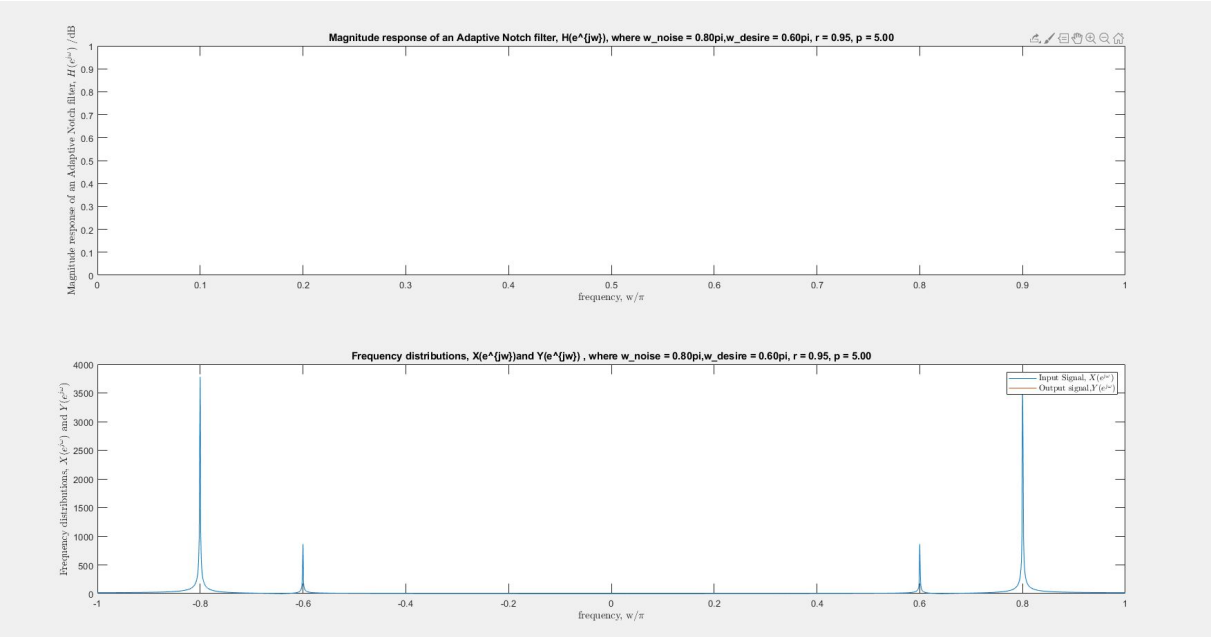
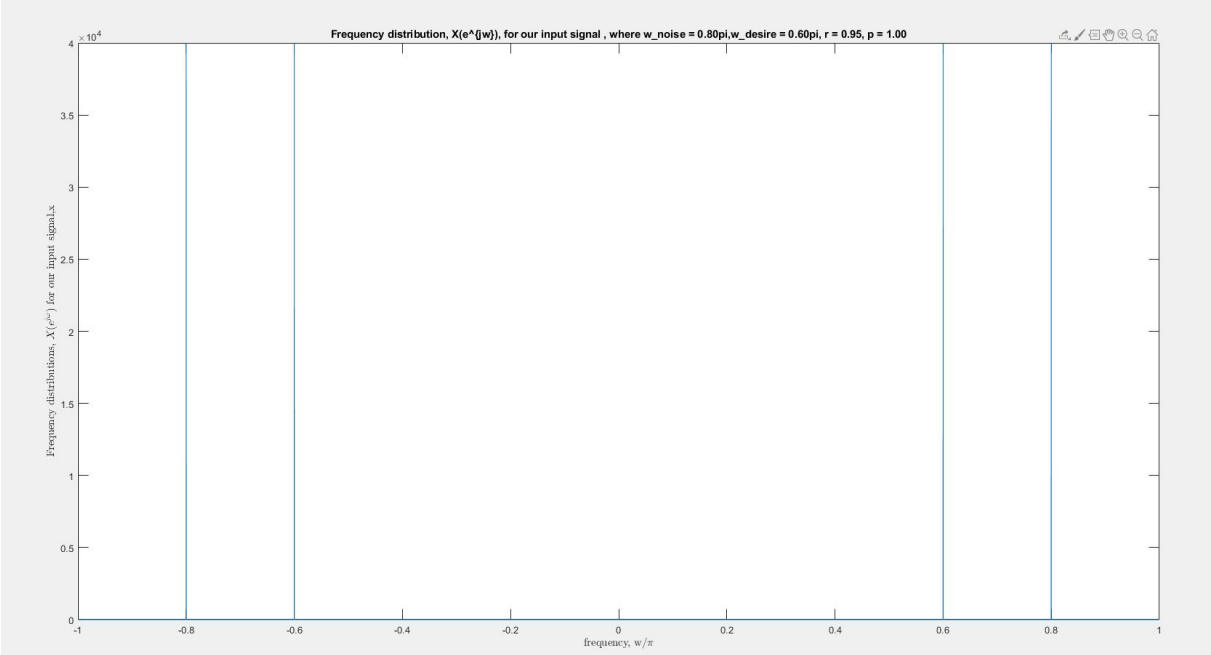
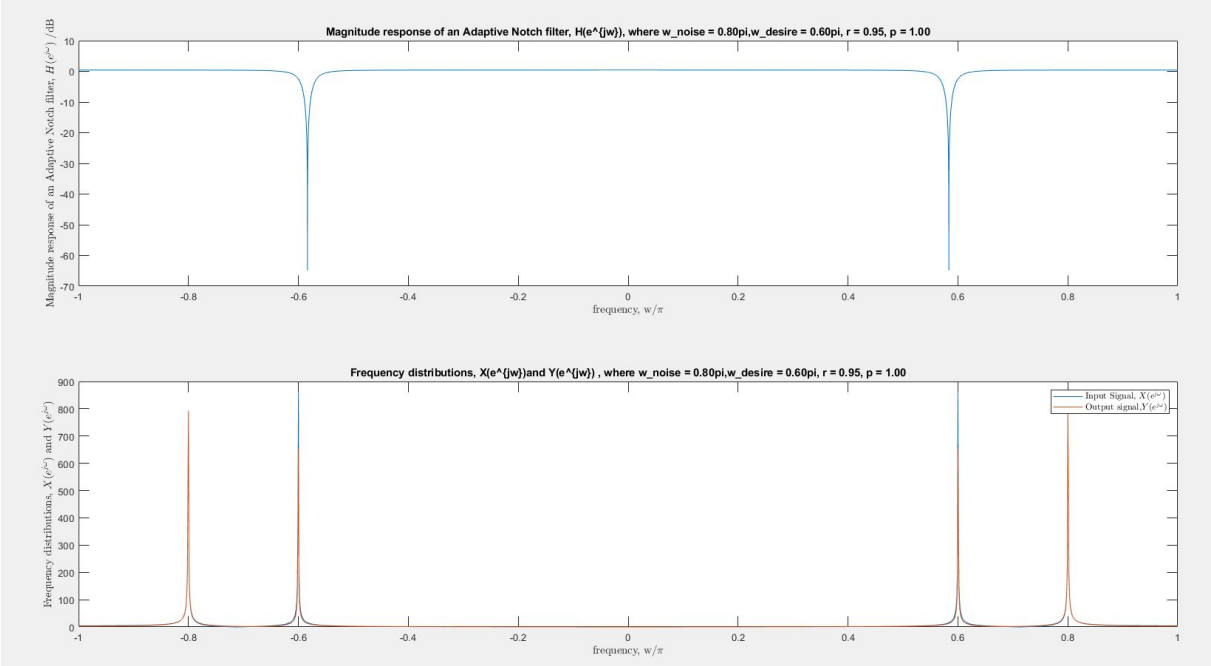


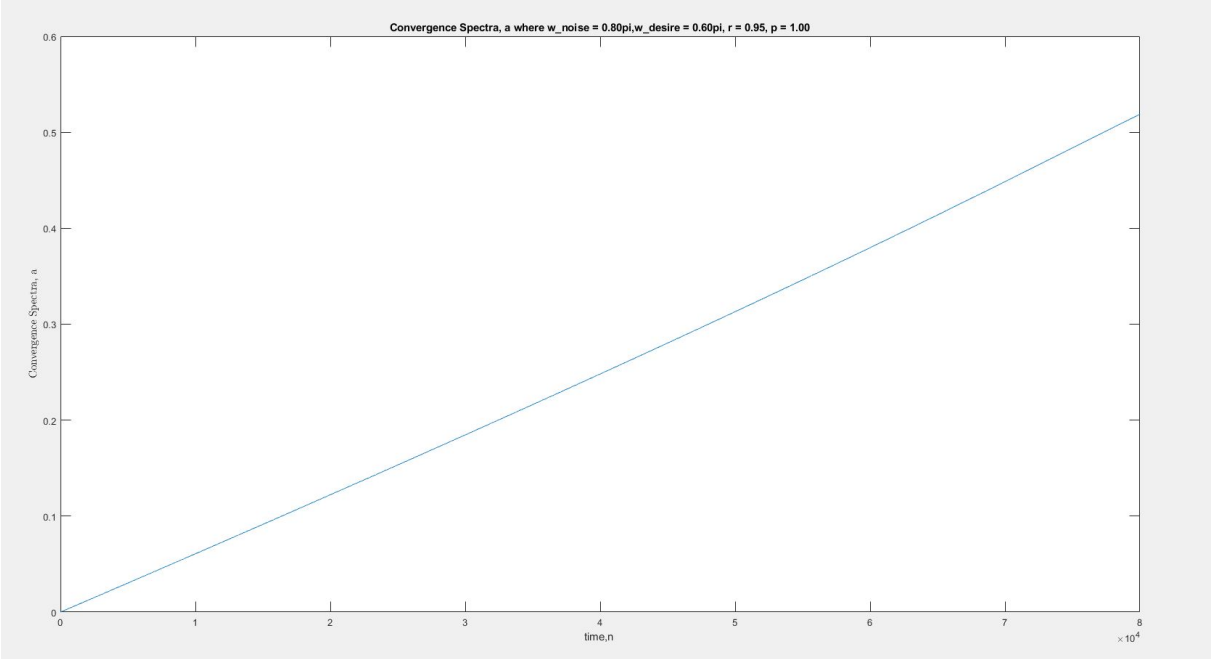
Figure B.11: Filtered Signal of Model f$_{noise} = 0.05, mu = 0090000, r = 0.91$

Figure B.12: Convergence of a plot f$_{noise} = 0.05, mu = 0090000, r = 0.95$

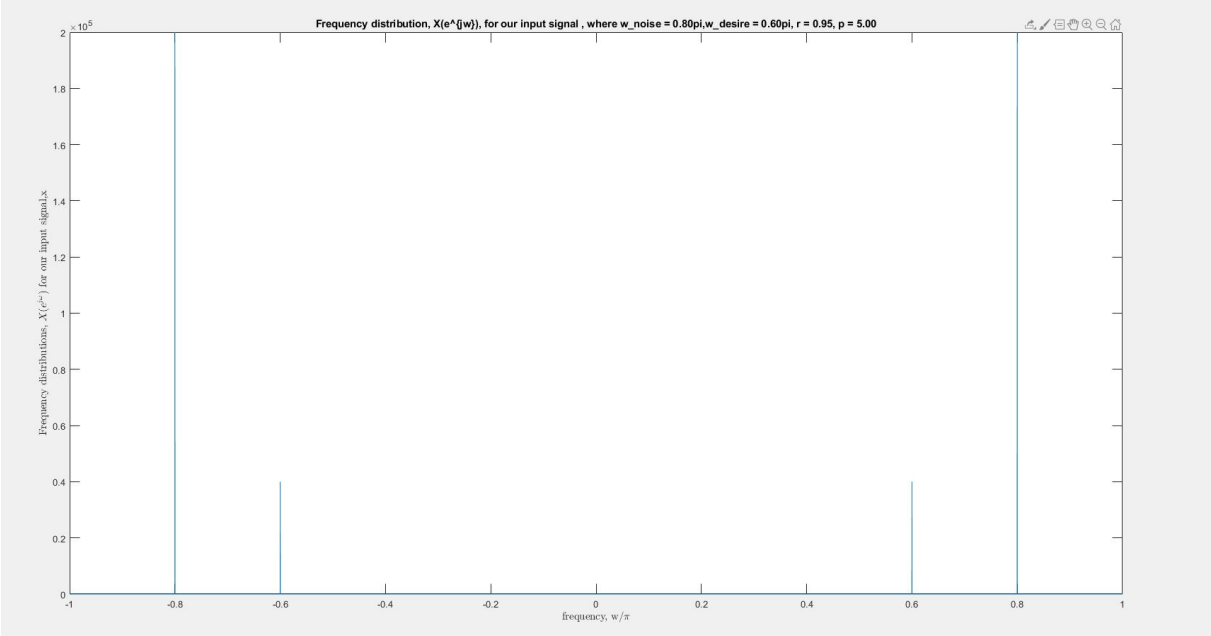
Tuning parameters 4 - Changing the the strength of inference
Baseline Model

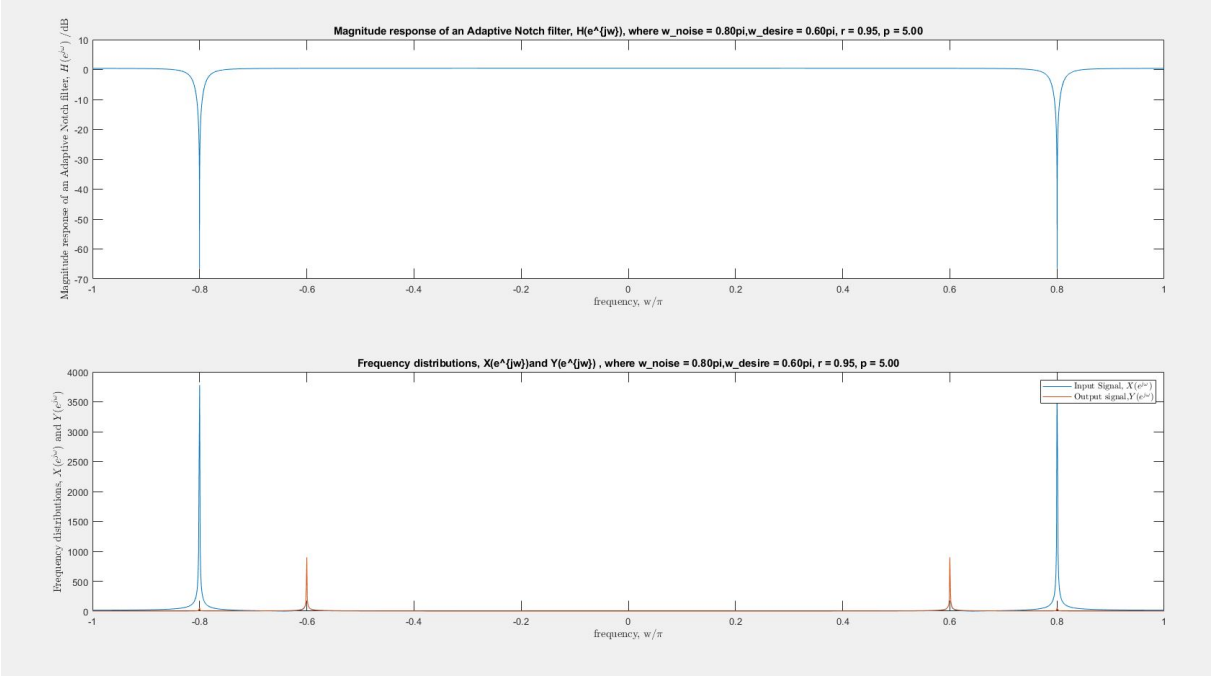

Frequency distribution, X(e^{jw}), for our input signal , where w_noise = 0.80pi,w_desire = 0.60pi, r = 0.95, p = 1.00
Frequency distributions, $X(e^{j\omega})$ for our input signal,x
frequency, w/π


Magnitude response of an Adaptive Notch filter, H(e^{jw}), where w_noise = 0.80pi,w_desire = 0.60pi, r = 0.95, p = 1.00
Magnitude response of an Adaptive Notch filter, $H(e^{j\omega})$ /dB
frequency, w/π
Frequency distributions, X(e^{jw})and Y(e^{jw}) , where w_noise = 0.80pi,w_desire = 0.60pi, r = 0.95, p = 1.00
Frequency distributions, $X(e^{j\omega})$ and $Y(e^{j\omega})$
Input Signal, $X(e^{j\omega})$
Output signal,$Y(e^{j\omega})$
frequency, w/π



Signal with strength 5 times stronger then bASELINE Model




Magnitude response of an Adaptive Notch filter, H(e^{jw}), where w_noise = 0.80pi,w_desire = 0.60pi, r = 0.95, p = 5.00
Magnitude response of an Adaptive Notch filter, $H(e^{j\omega})$ /dB
frequency, w/π
Frequency distributions, X(e^{jw})and Y(e^{jw}) , where w_noise = 0.80pi,w_desire = 0.60pi, r = 0.95, p = 5.00
Frequency distributions, $X(e^{j\omega})$ and $Y(e^{j\omega})$
Input Signal, $X(e^{j\omega})$
Output signal,$Y(e^{j\omega})$
frequency, w/π

partAimgs/a/WithoutNoise/p_change/mu_0.0000090-f_noise_0.95-r_0.40-p_5.00_without_noise_CONV.JPG

Signal with strength 10 times stronger then bASELINE Model

partAimgs/a/WithoutNoise/p_change/mu_0.0000090-f_noise_0.95-r_0.40-p_10.00_without_noise_INPUT.JPG


partAimgs/a/WithoutNoise/p_change/mu_0.0000090-f_noise_0.95-r_0.40-p_10.00_without_noise_FILTER.JPG

As the inference signal strength increases the filter converges faster.

Tuning parameters 5 - Changing the the type of signal

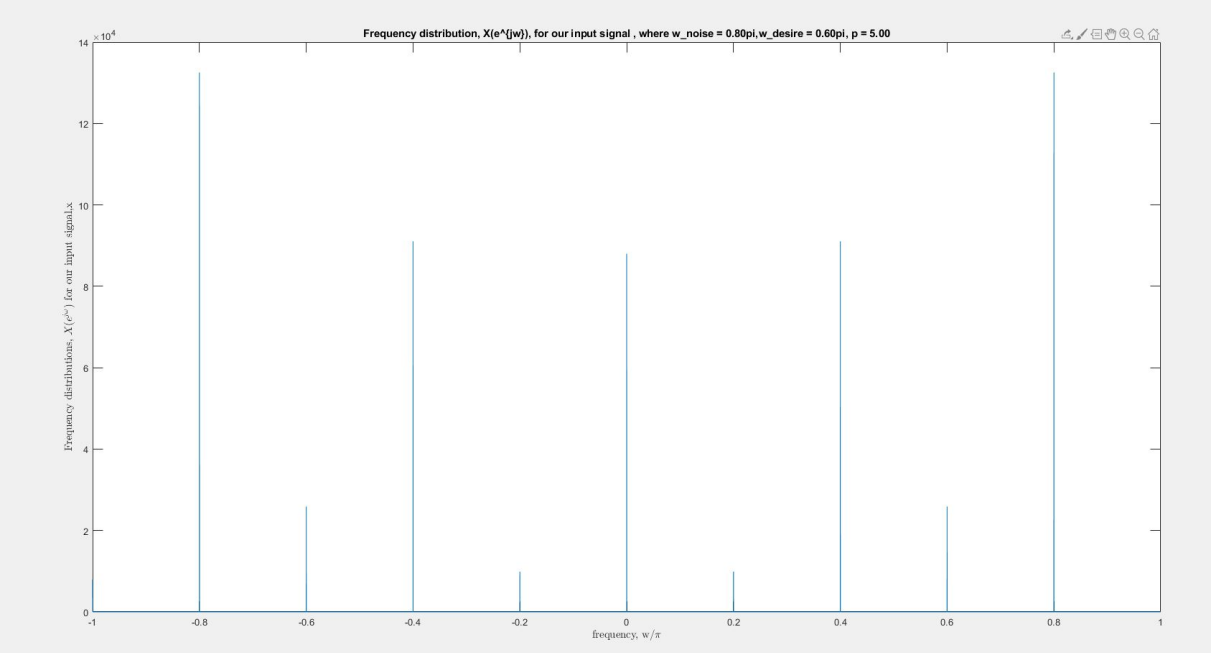


Figure B.10: SawTooth Wave Input Signal of Model f$_{noise} = 0.4, mu = 0000090, r = 0.95$

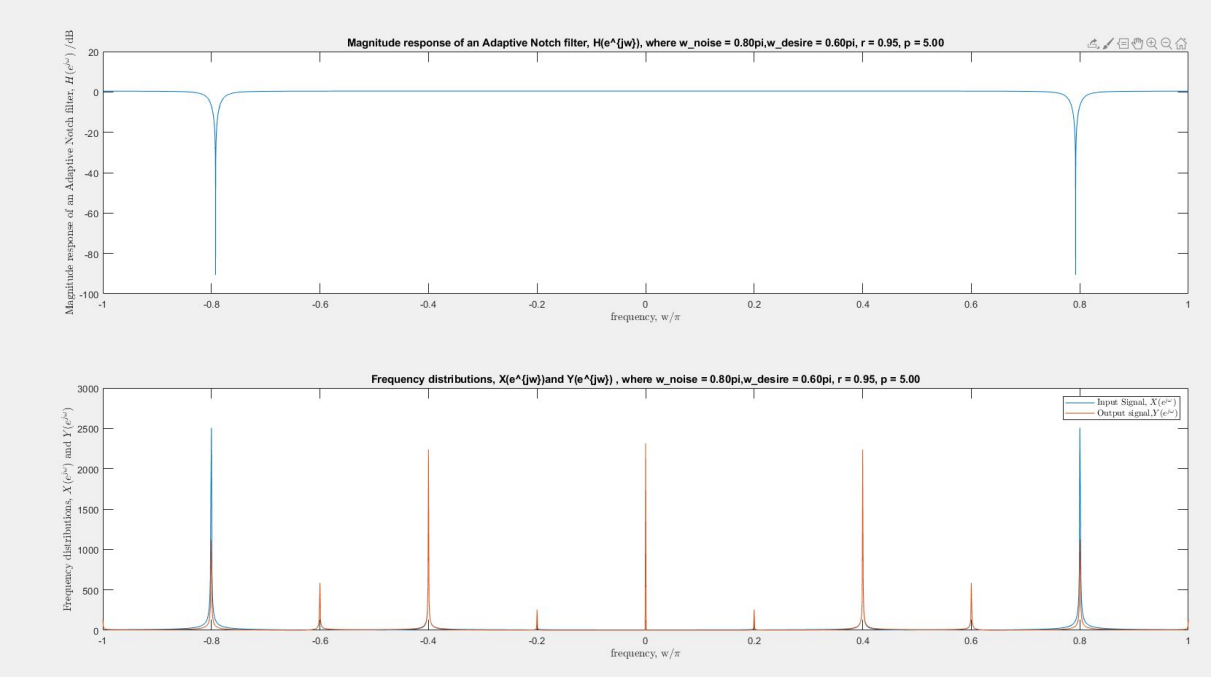


Figure B.11: Sawtootuh Wave Filtered Signal of Model f$_{noise} = 0.4, mu = 0000090, r = 0.95$

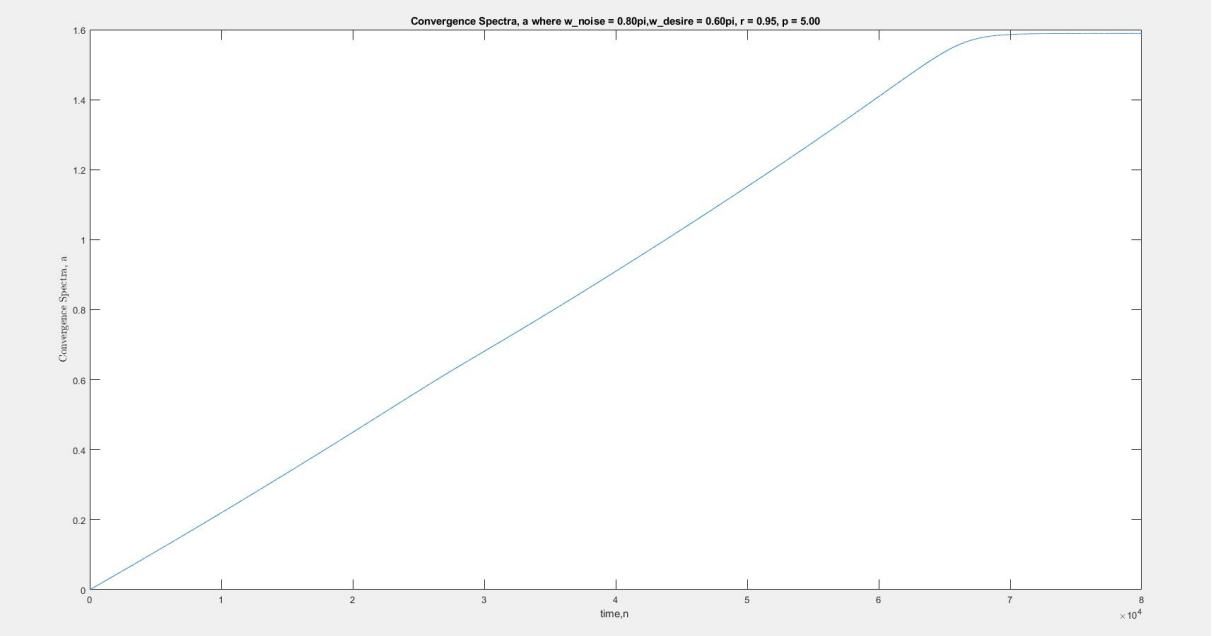


Figure B.12: Convergence of a plot f$_{noise} = 0.4, mu = 0000090, r = 0.95$

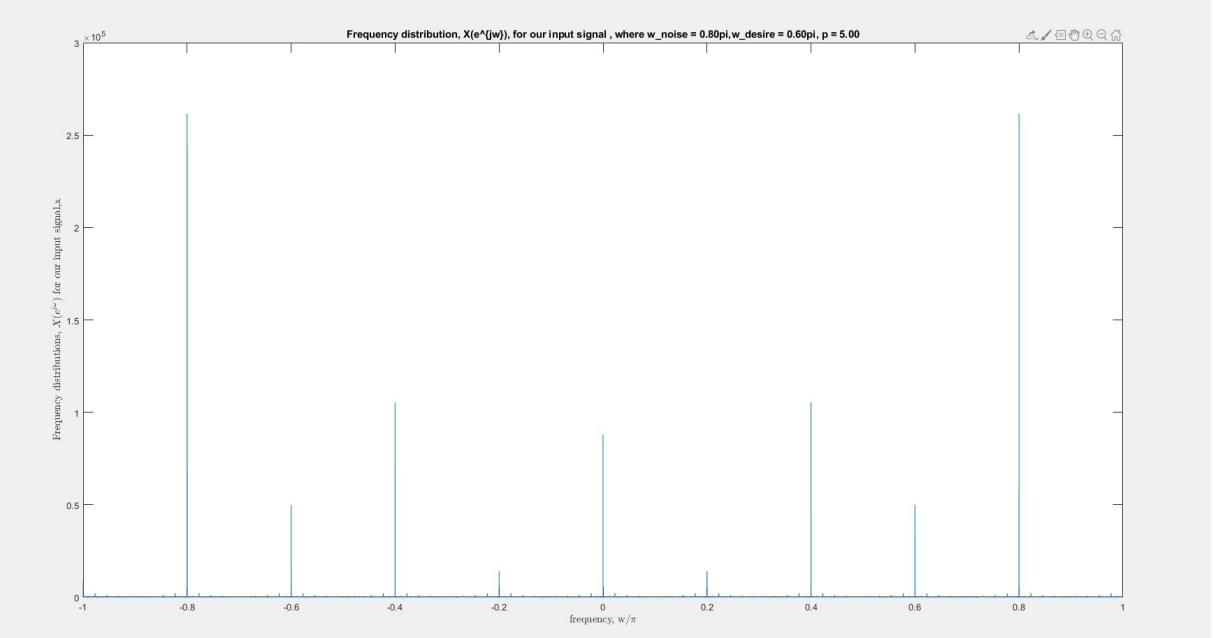


Figure B.10: Square Wave Input Signal of Model f$_{noise} = 0.4, mu = 0000090, r = 0.95$

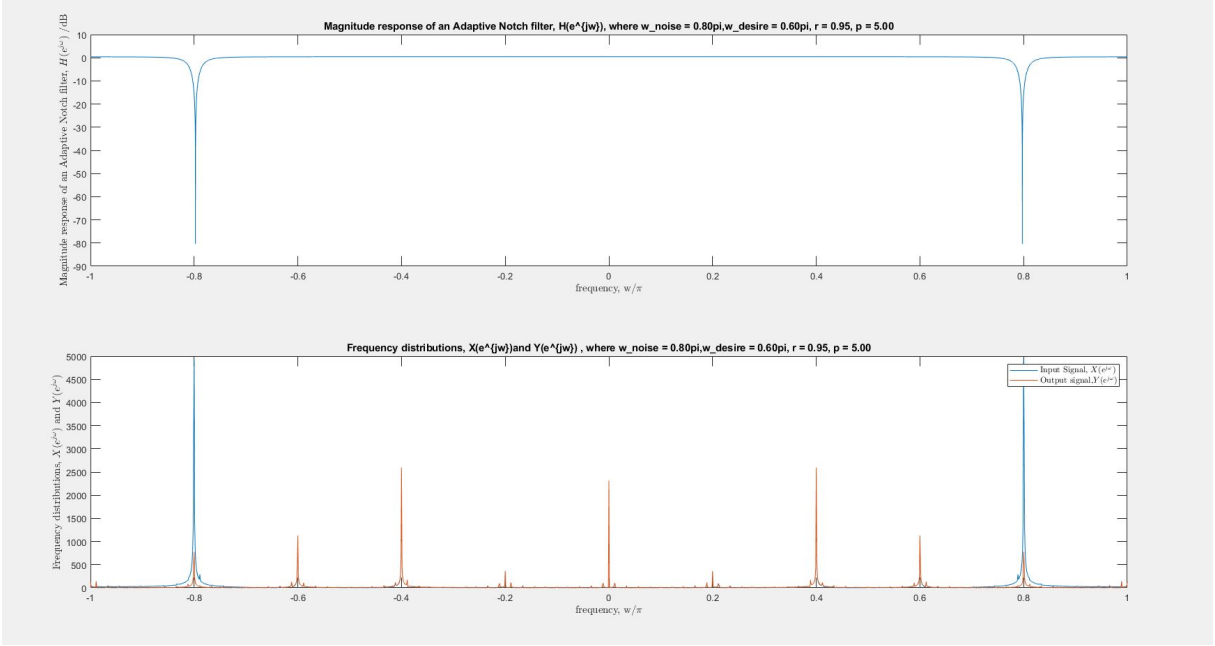


Figure B.11: Square Wave Filtered Signal of Model f$_{noise} = 0.4, mu = 0000090, r = 0.95$

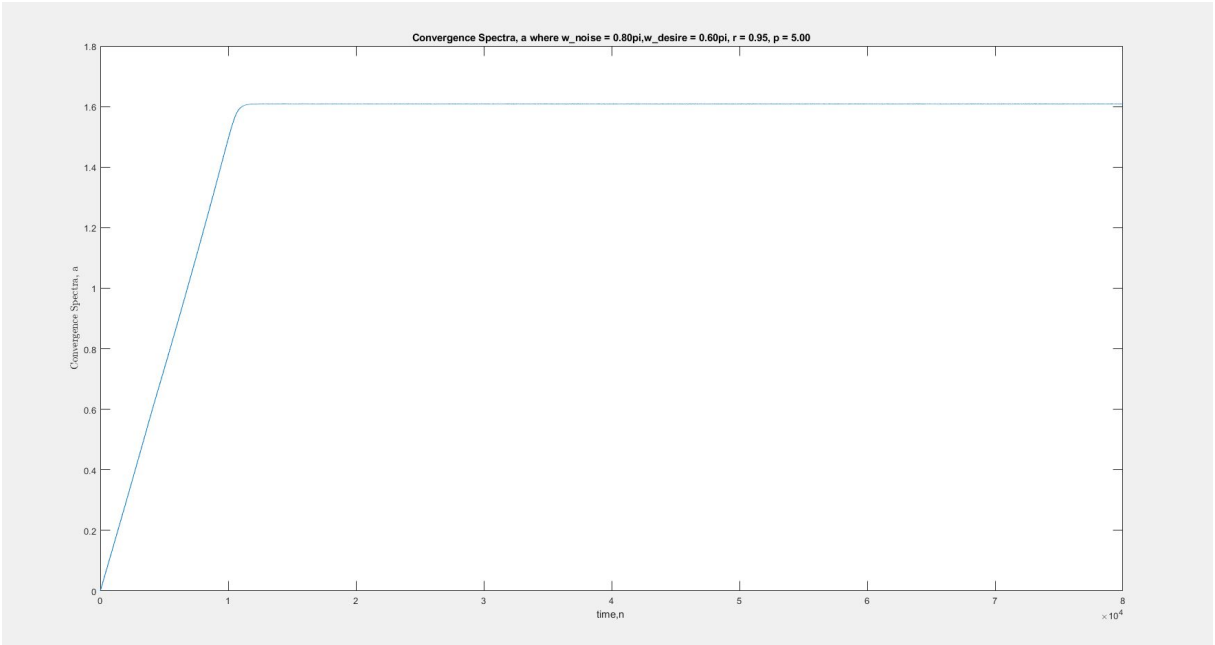


Figure B.12: Square Wave Convergence of a plot f$_{noise} = 0.4, mu = 0000090, r = 0.95$

1.2 b

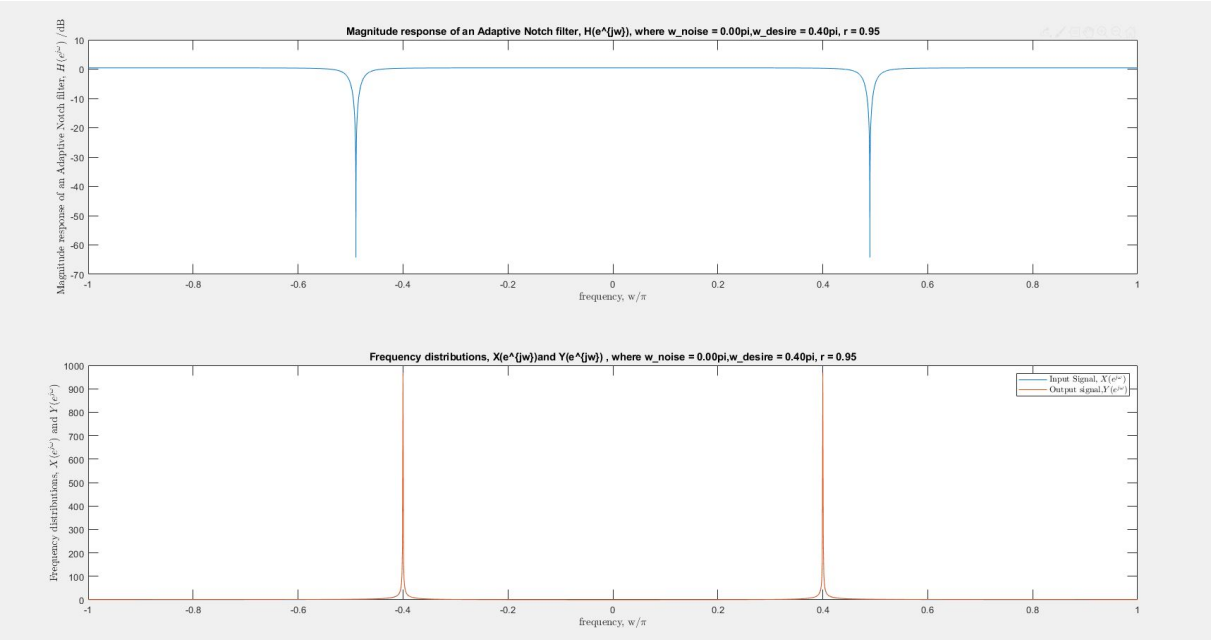


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0, mu = 0000090, r = 0.95$

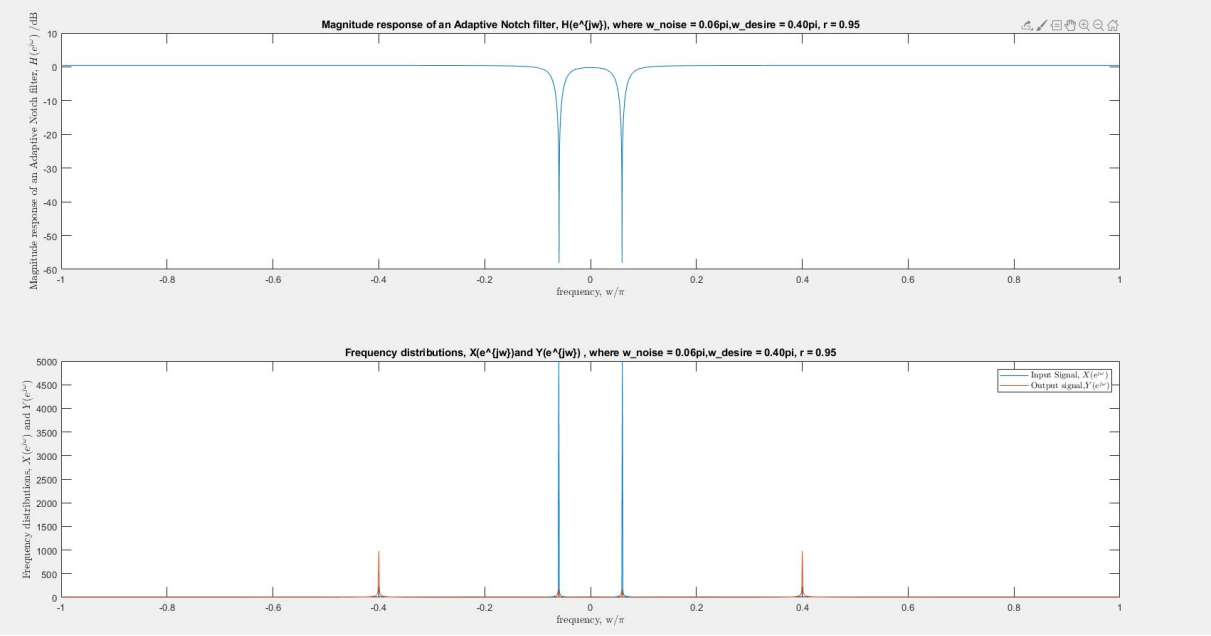


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0, mu = 0000090, r = 0.95$

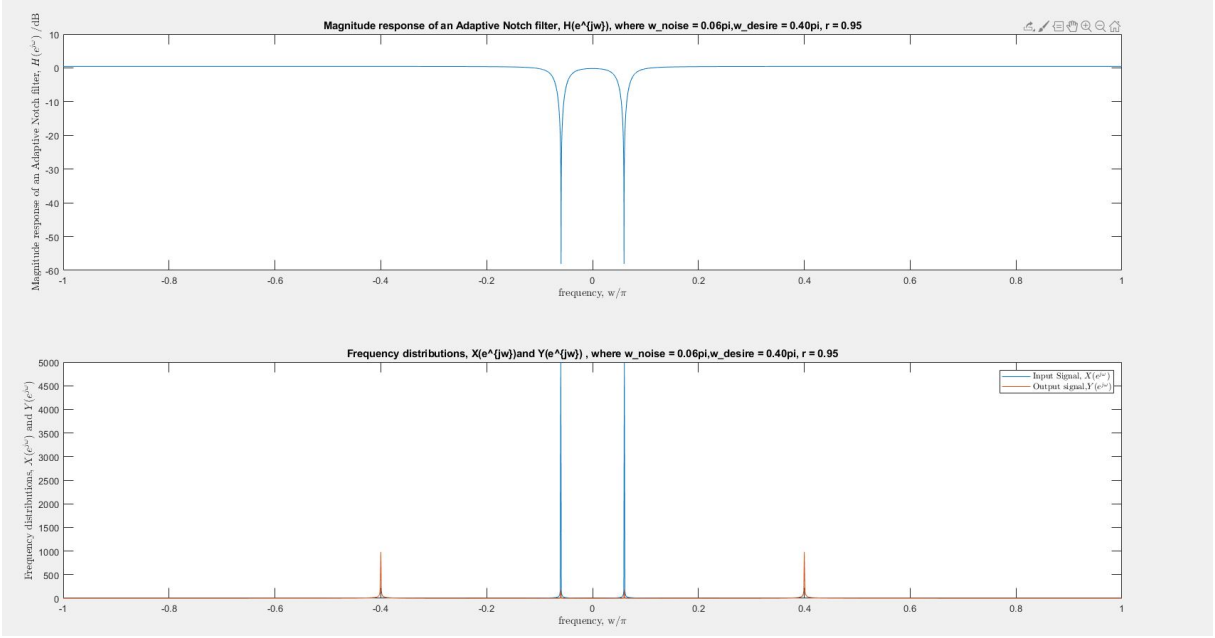


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.03, mu = 0000090, r = 0.95$

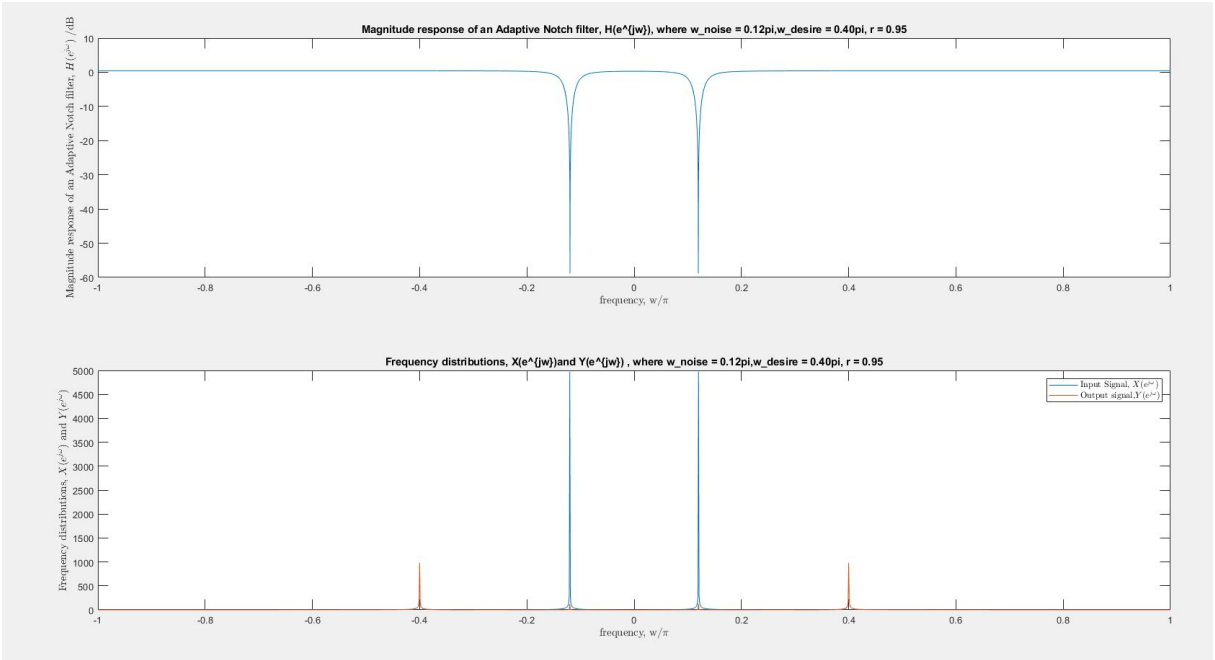


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.06, mu = 0000090, r = 0.95$

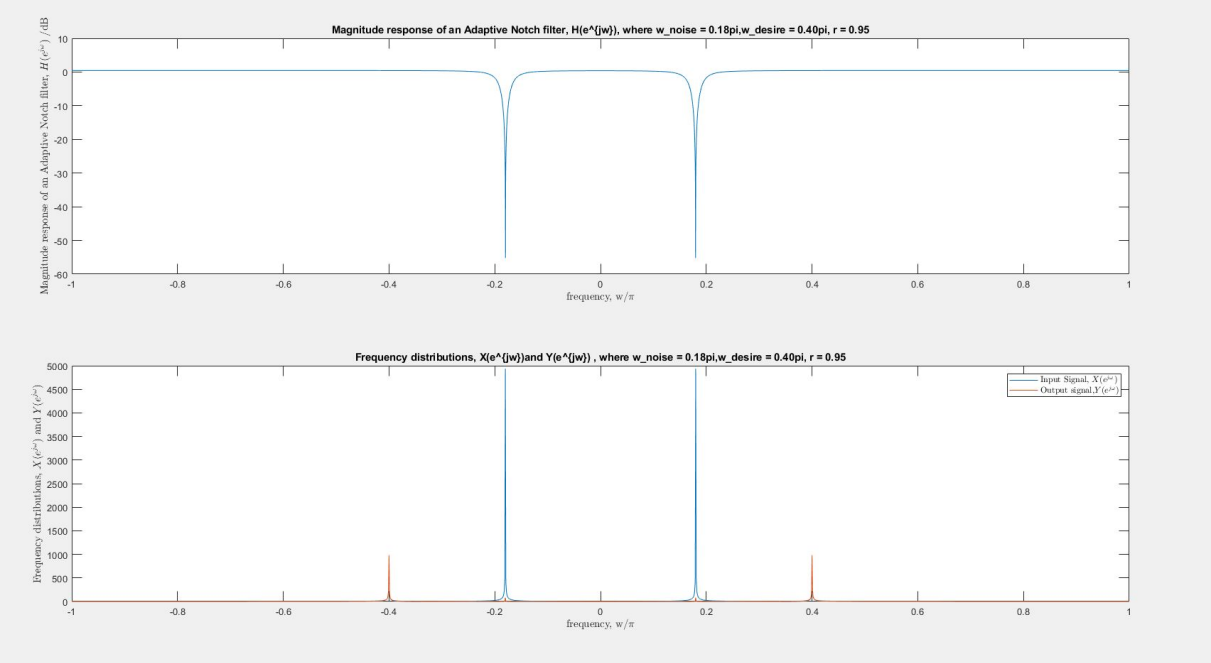


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.09, mu =$ 0000090, $r = 0.95$

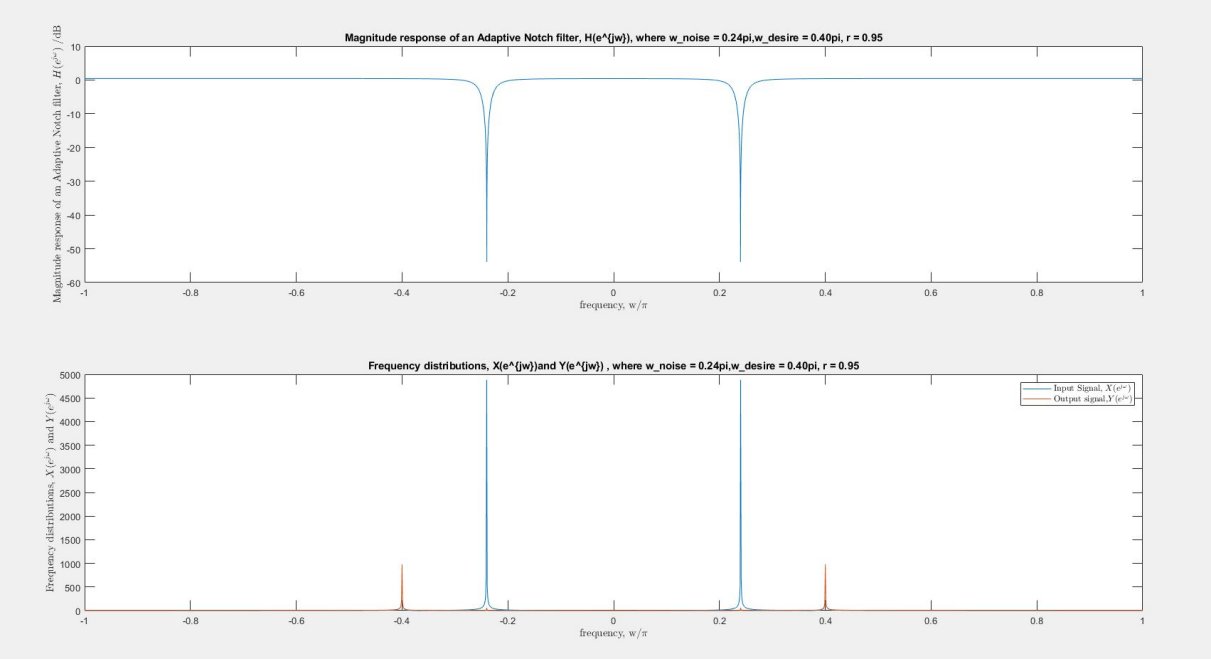


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.12, mu =$ 0000090, $r = 0.95$

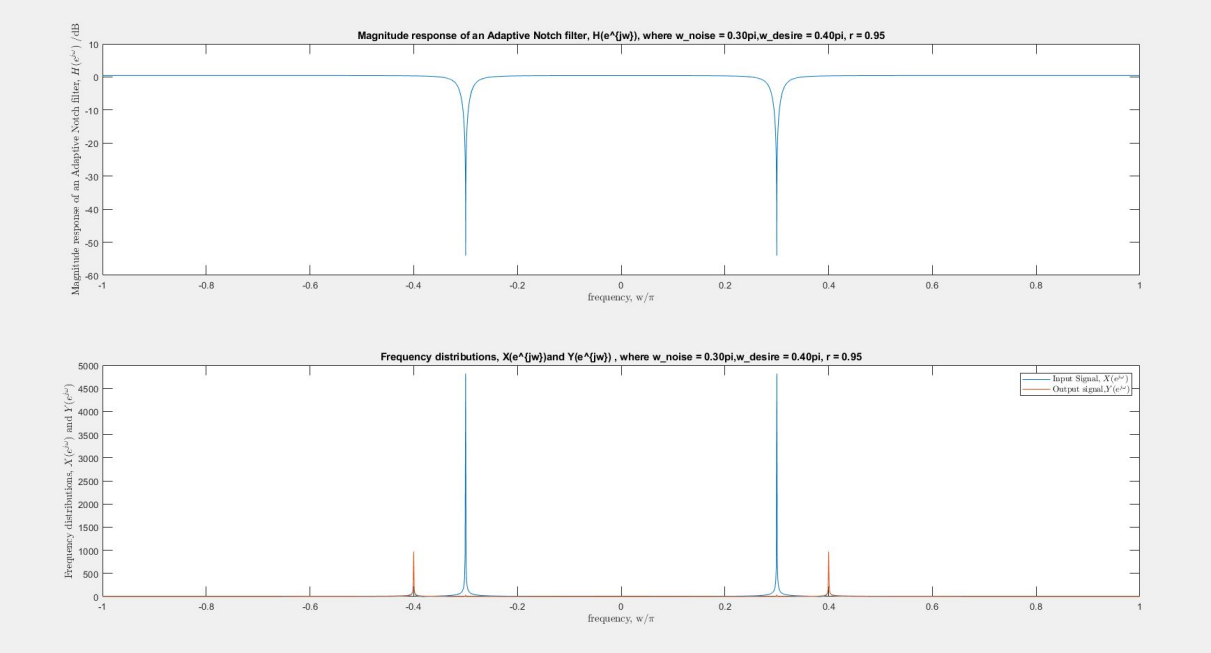


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.15, mu = 0000090, r = 0.95$

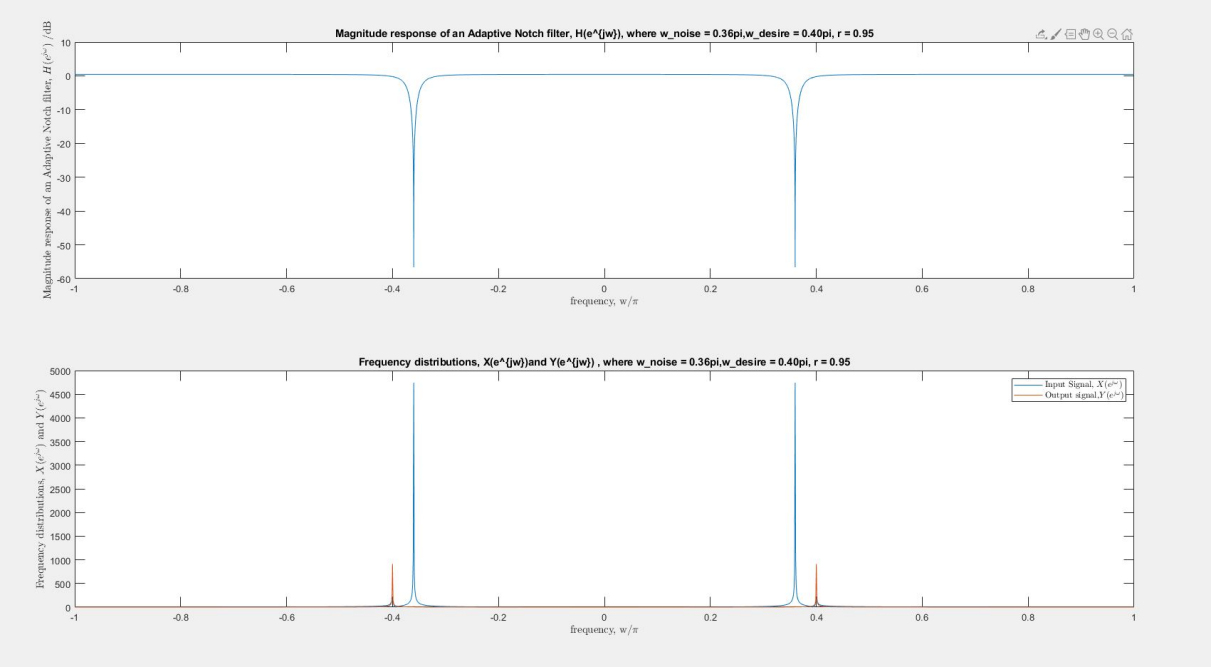


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.18, mu = 0000090, r = 0.95$

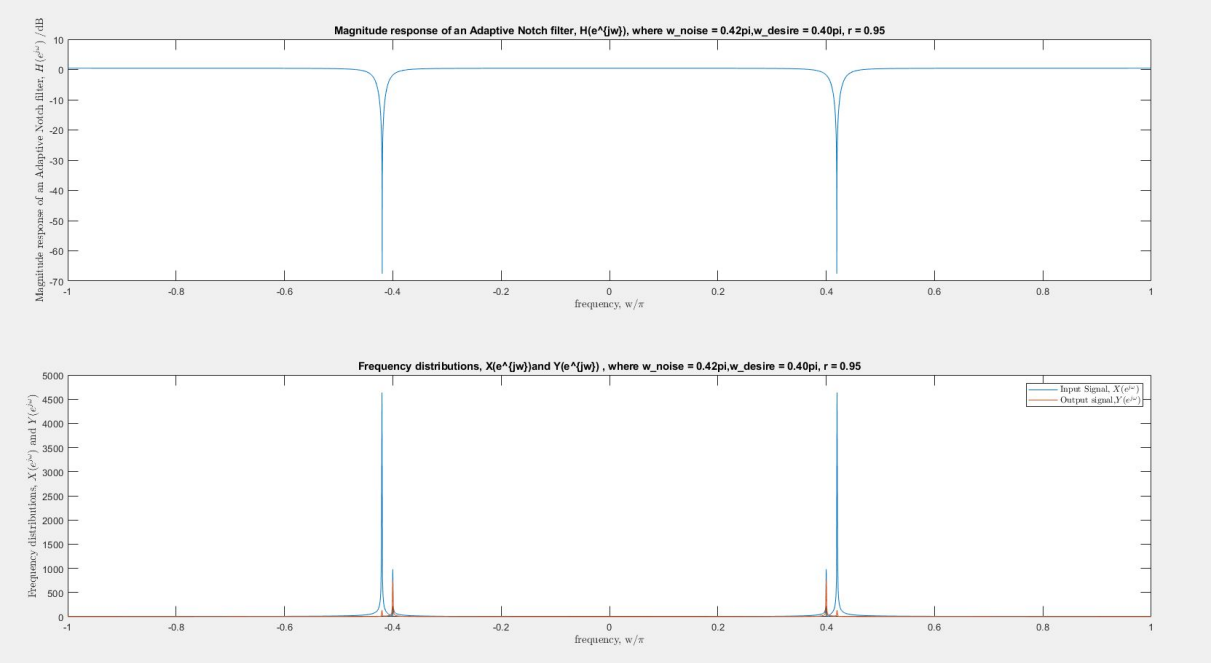


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.21, mu = 0000090, r = 0.95$

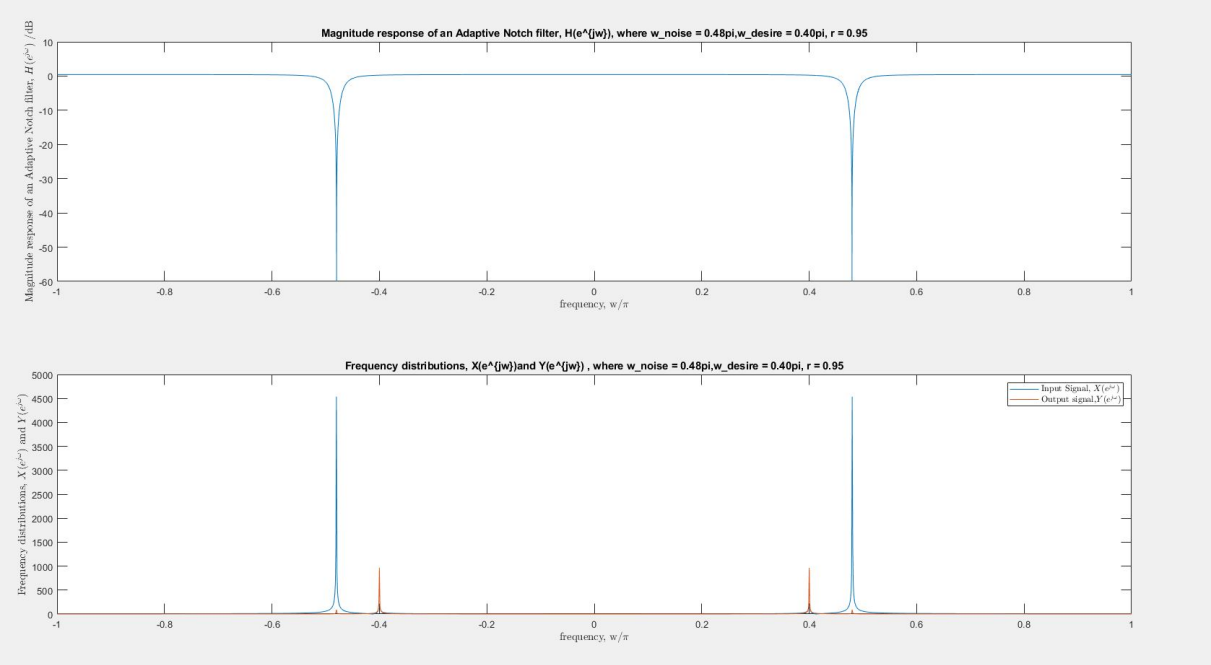


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.24, mu = 0000090, r = 0.95$

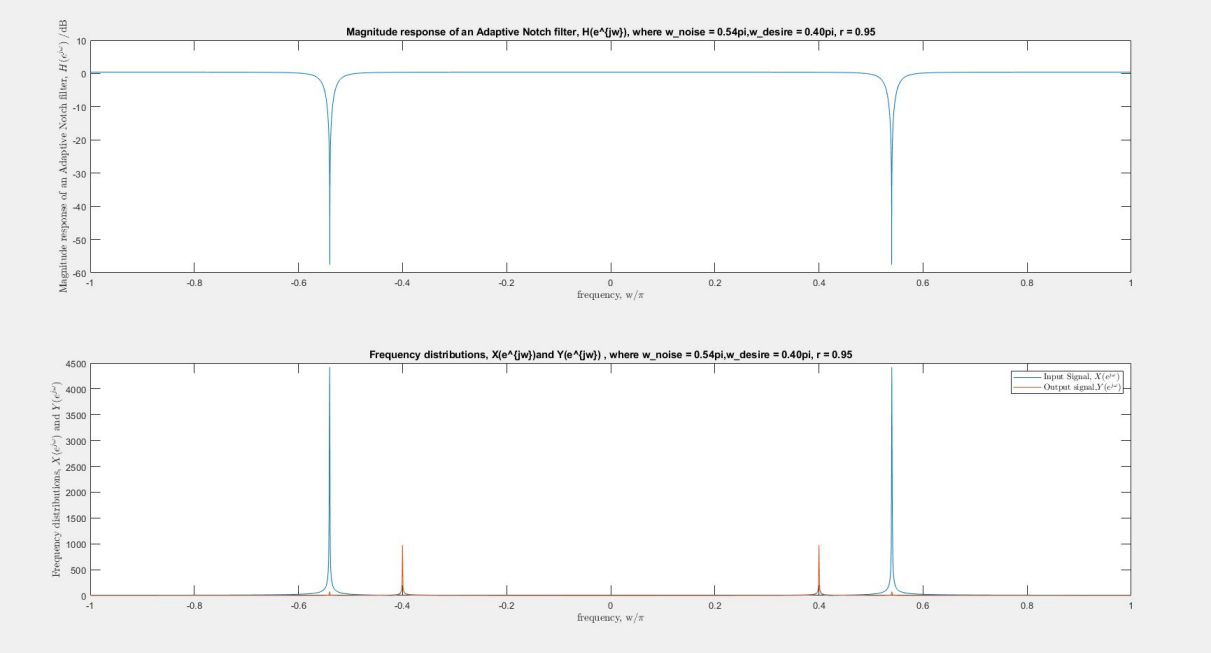


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.27, mu = 0000090, r = 0.95$

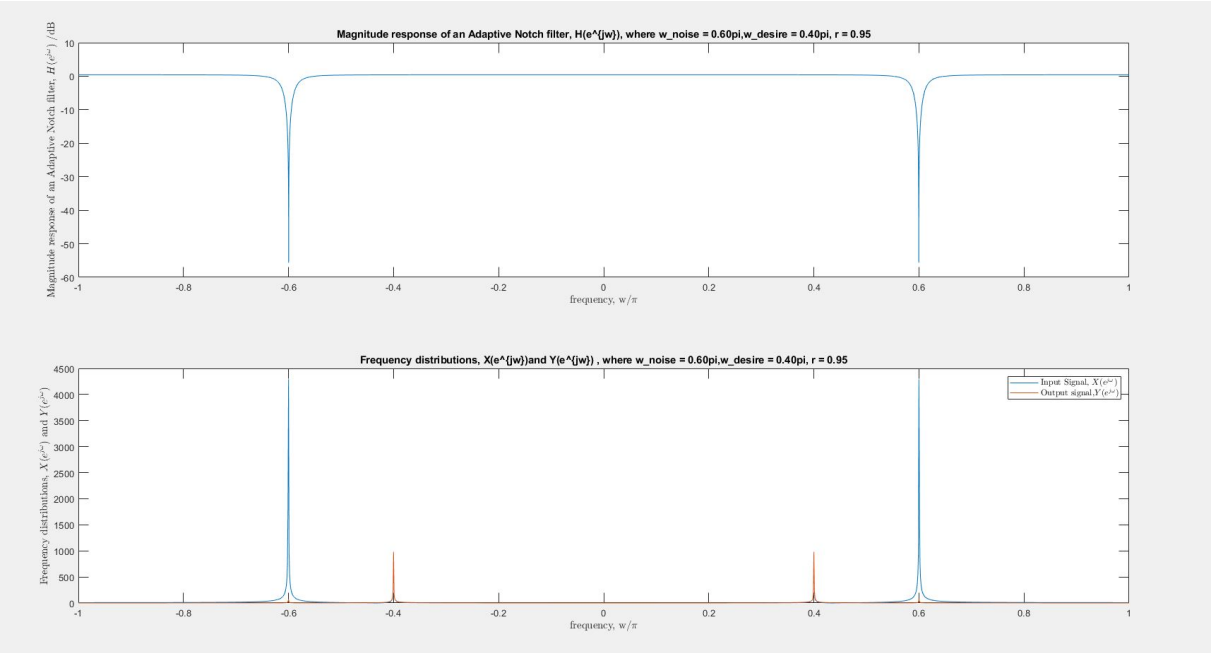


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.30, mu = 0000090, r = 0.95$

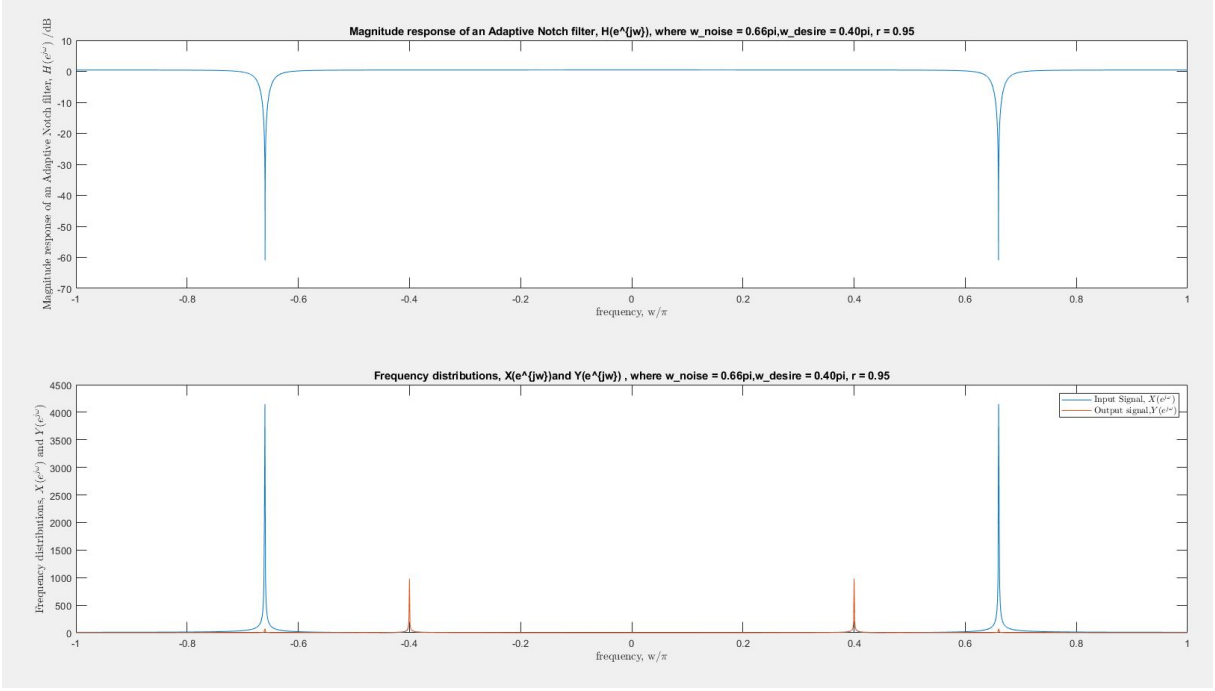


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.33, mu = 0000090, r = 0.95$

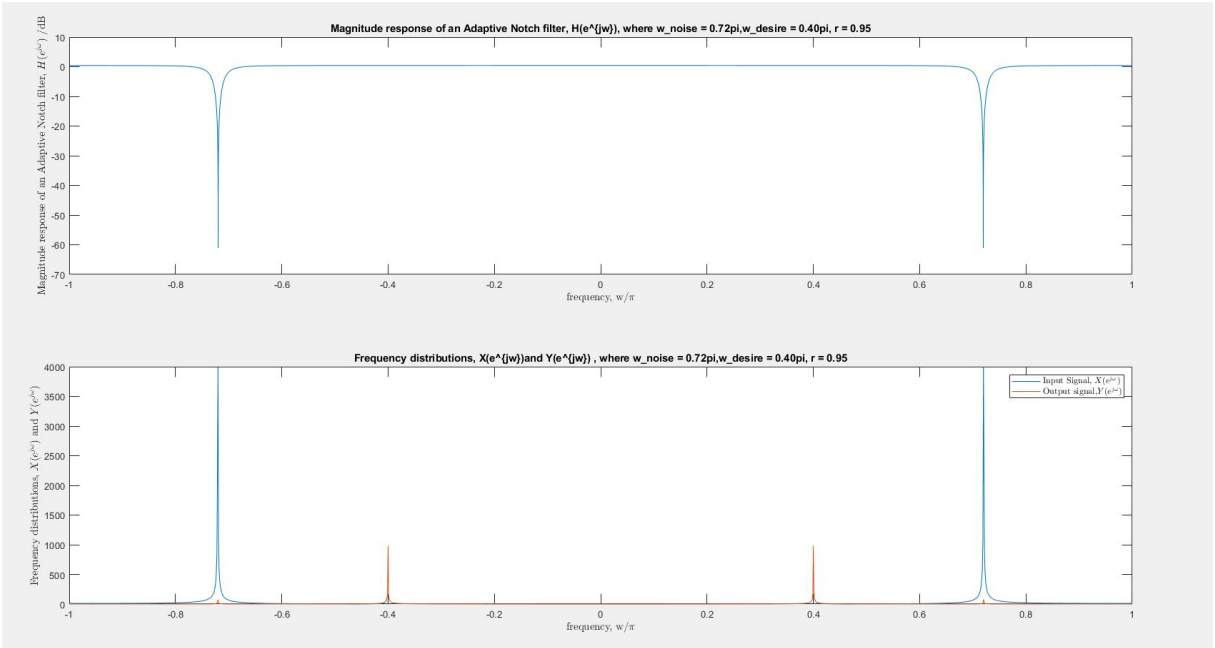


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.36, mu = 0000090, r = 0.95$

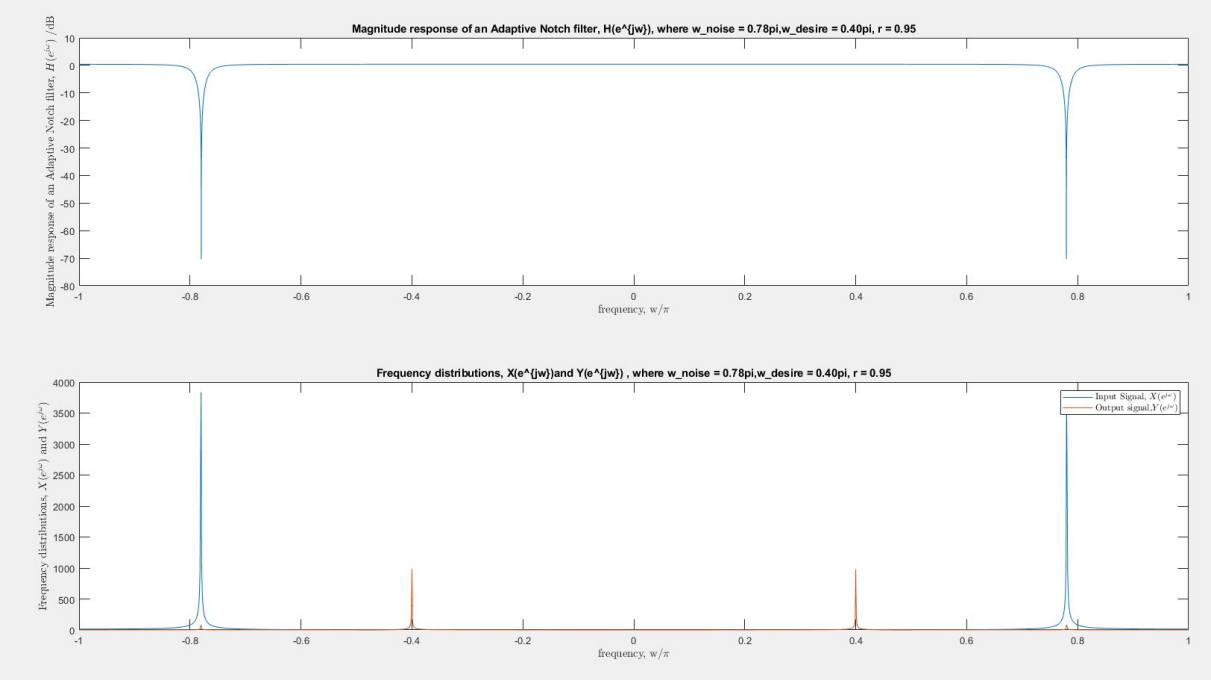


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model $\mathrm{f}_{noise} = 0.39, mu = 0000090, r = 0.95$

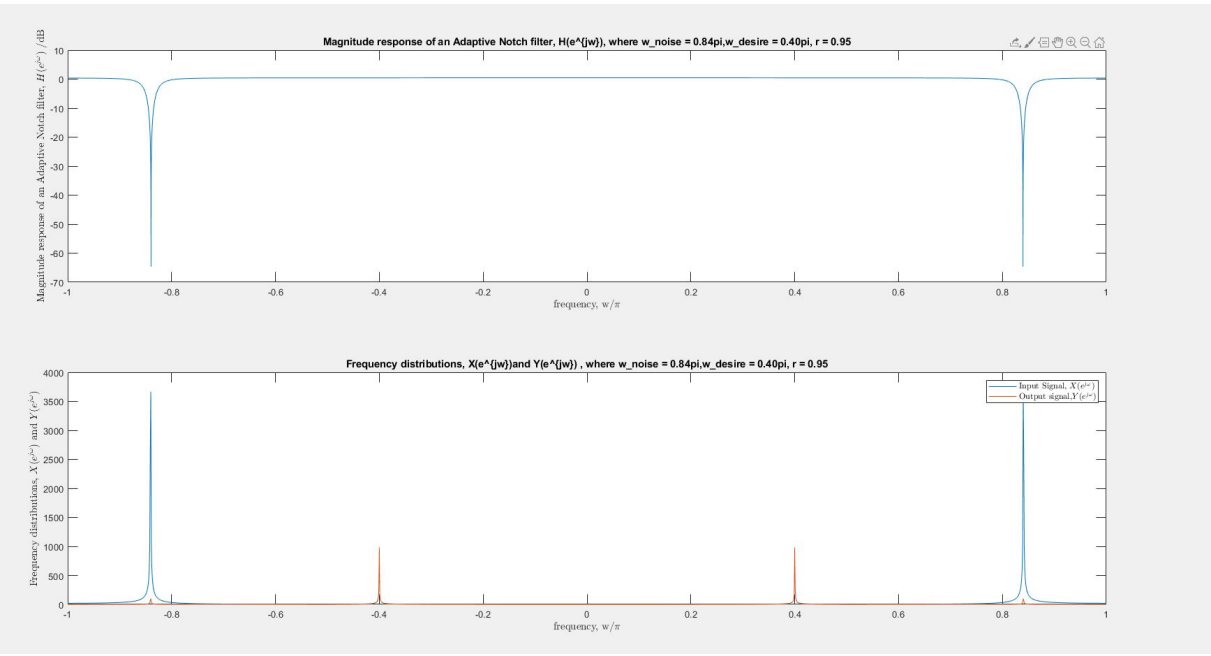


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model $\mathrm{f}_{noise} = 0.42, mu = 0000090, r = 0.95$

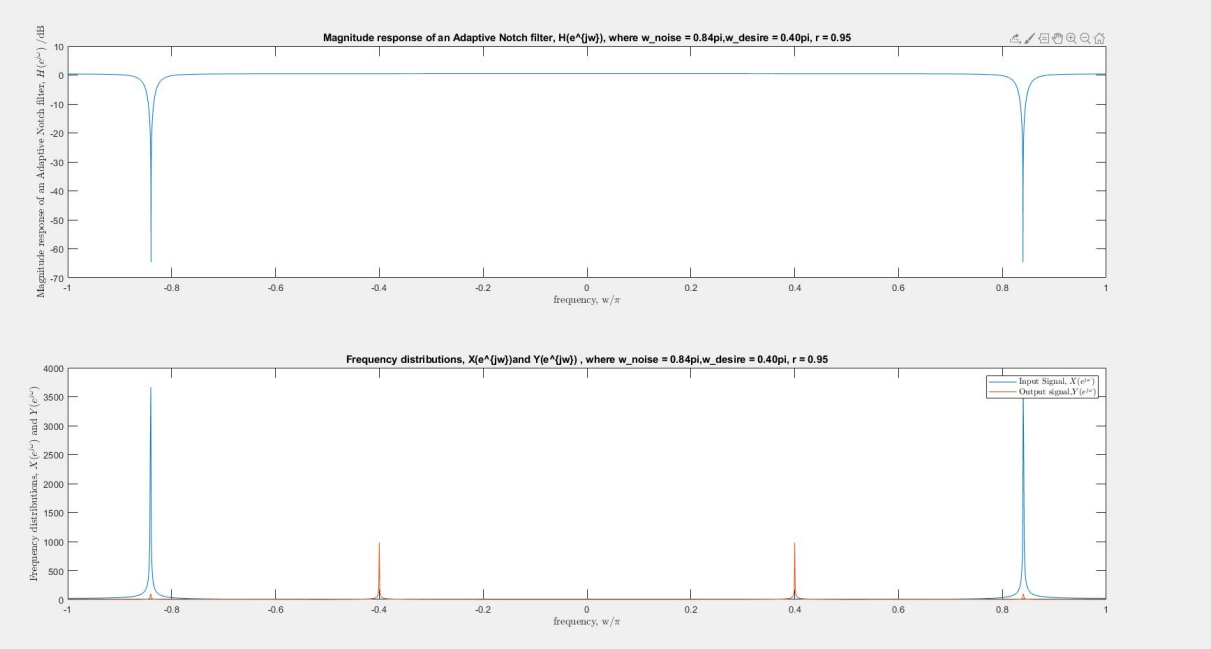


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.42, mu = 0000090, r = 0.95$

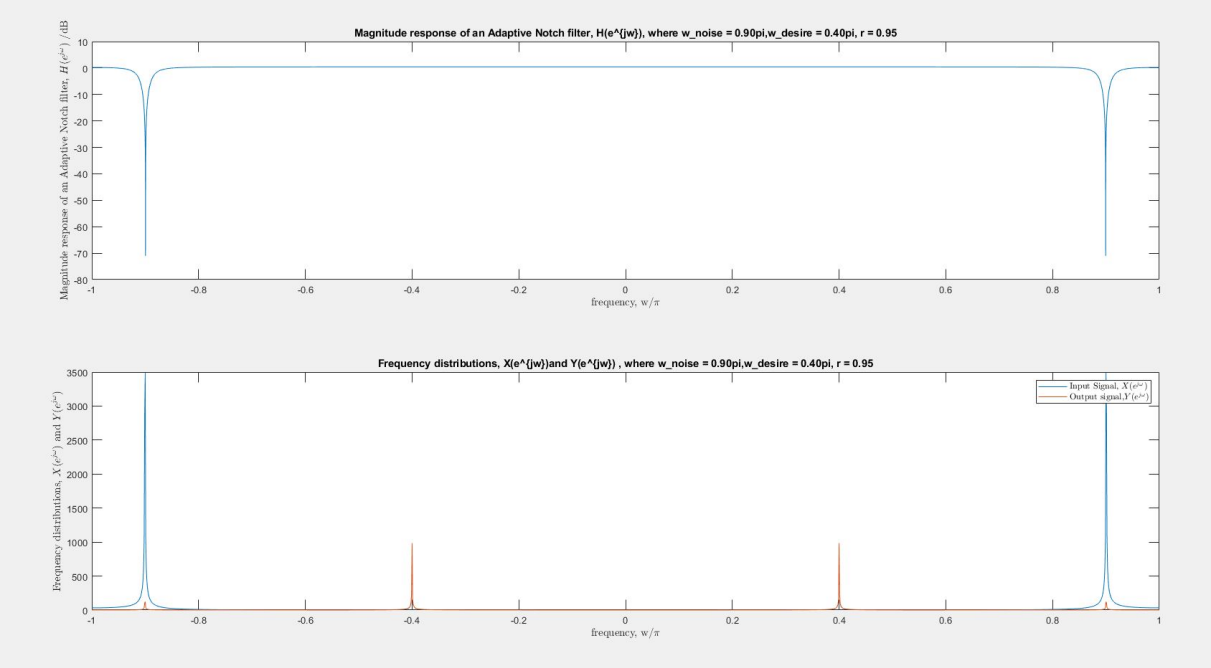


Figure B.12: Input Signal, Output Signal and Filter Response Magnitude for Signal of Model f$_{noise} = 0.45, mu = 0000090, r = 0.95$

Note that as the frequencies of the interfereance signal change the filter adapts to it and almost always produces the right output - the only irregularity is when the frequency of the inference signal is 0, the filter doesn't converge.

1.3 c

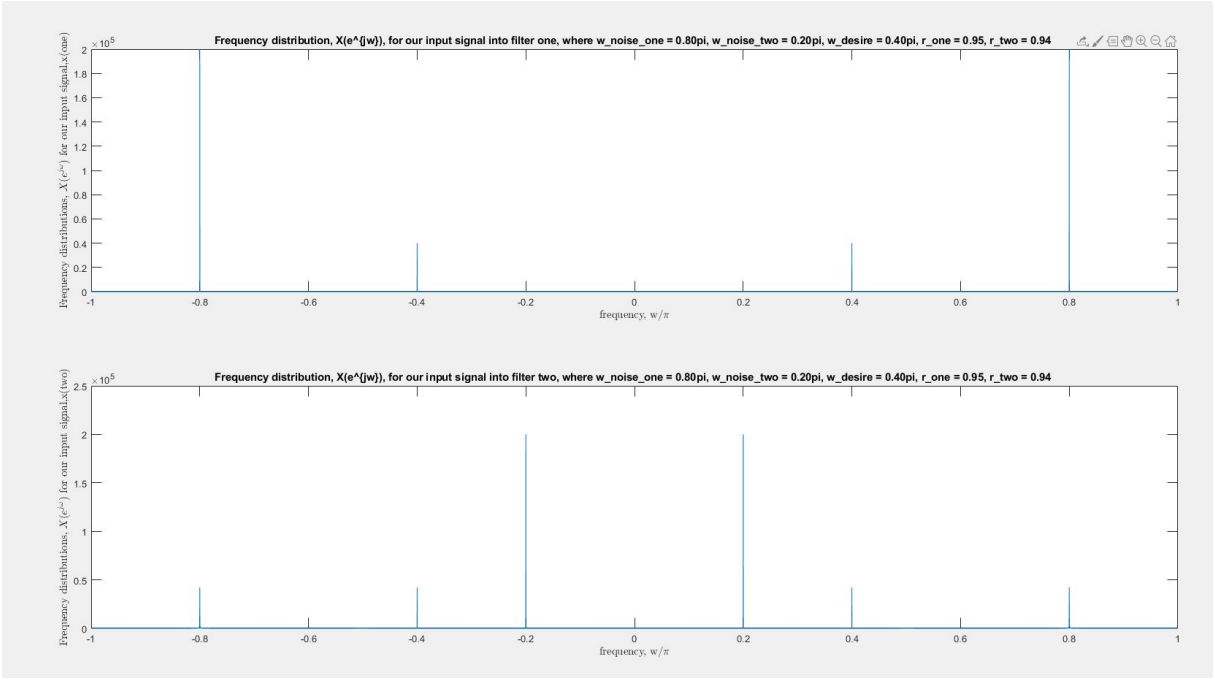


Figure A.12: Input Signal to the both cascaded filters

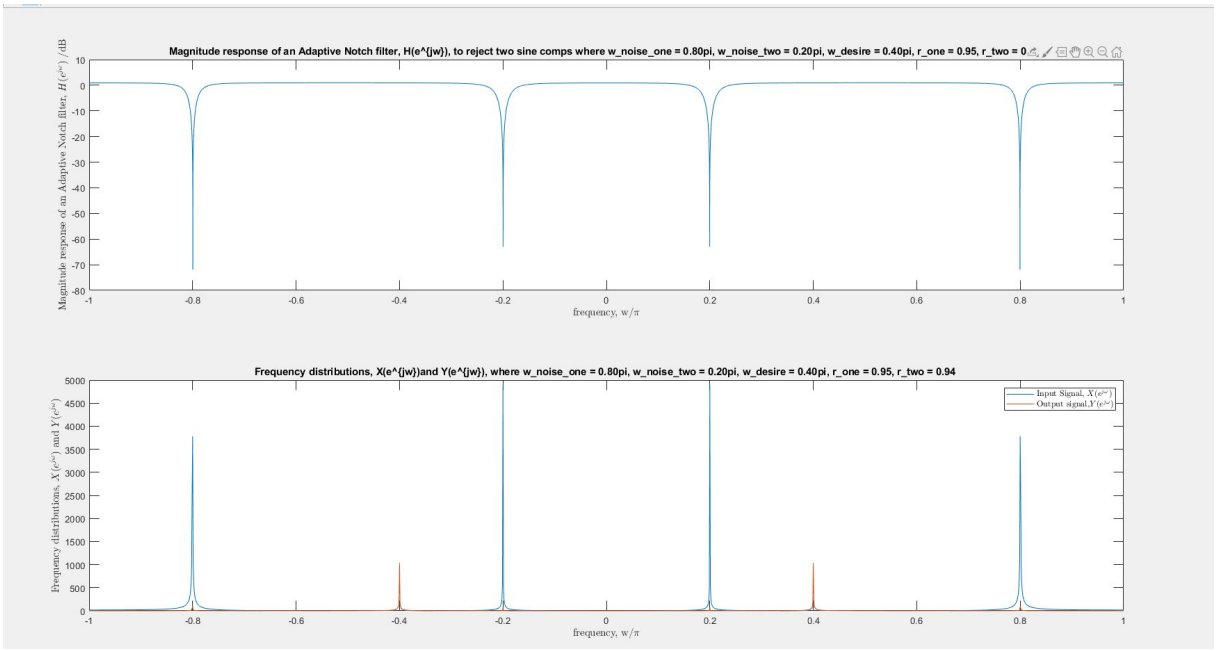


Figure A.12: Effective Magnitude response of filter with the input and output signals

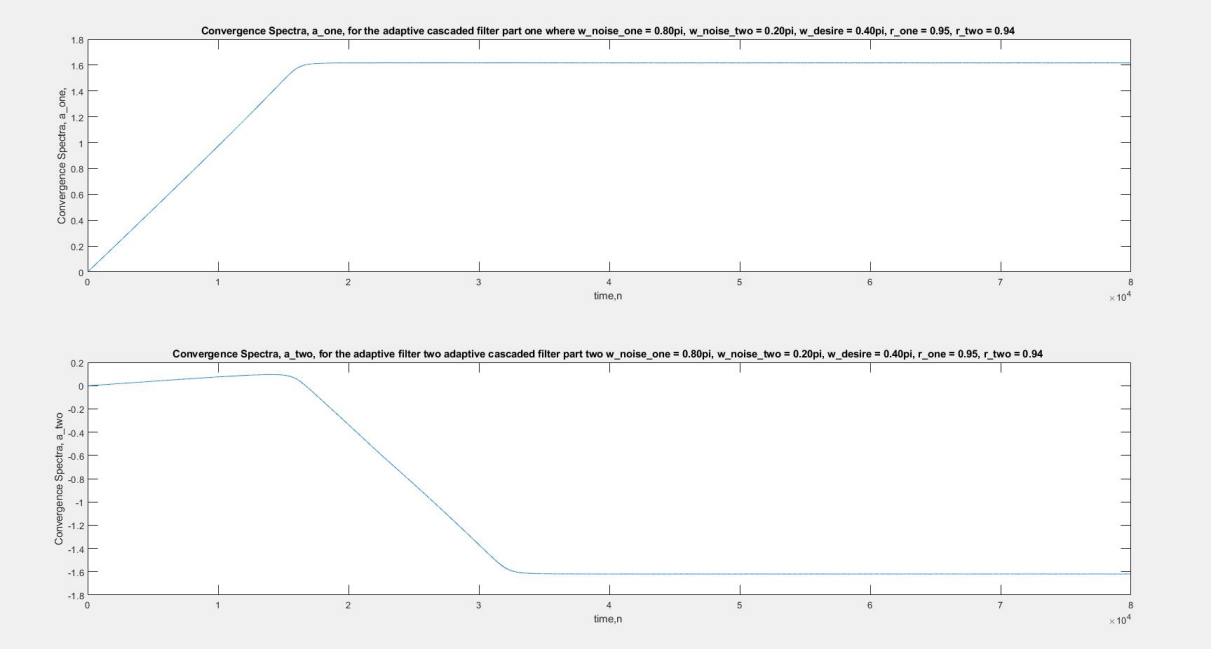


Figure A.12: Convergences of the intermediate cascade filters

The cascaded filter clearly works are we were able to filter out the two sine interference frequencies.

Our approach to part C was as follows: We simply cascaded two notch filters - i,e, the output of the first notch filter was used as input to the second notch filter.

2 Part B

The baseline performance of our equalizer was achieved with the following design parameters: the channel length was 5, the adaptive filter length was 12, the SNR of the interference was 20 dB, the delay of the desired signal to the adaptive filter was 2 samples, and the filter step size was 0.035. The filter coefficients were initialized to 1, and the channel was $h[n] = [0.310.70.30.2]$. In the below figures we display some plots of the channel and filter for the baseline performance.

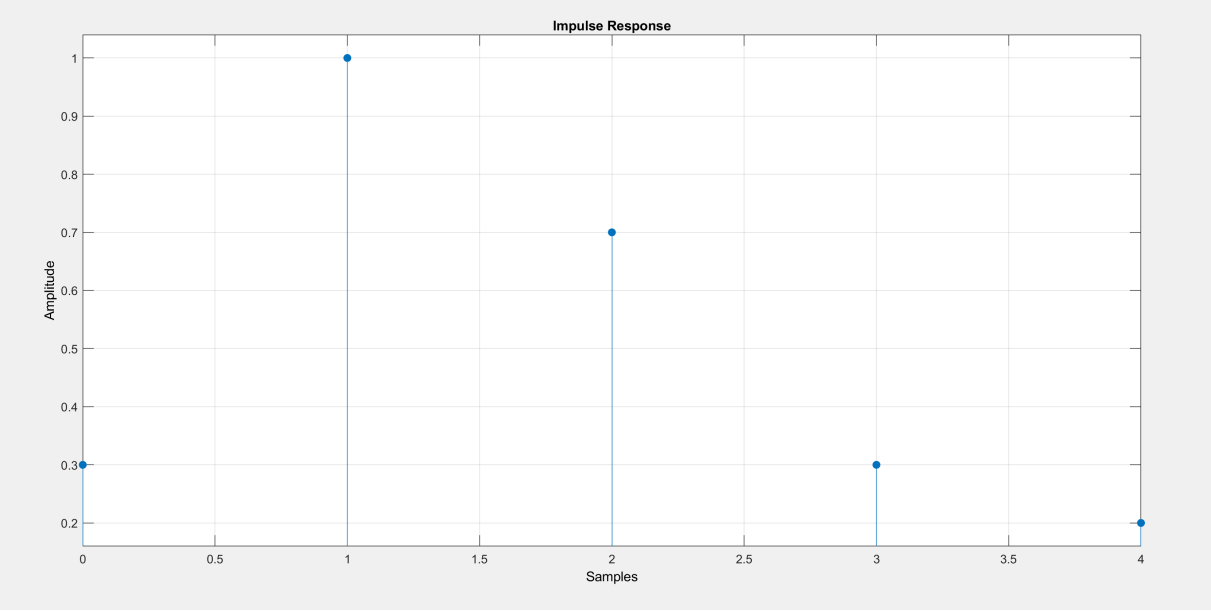


Figure B.1: Channel Impulse Response

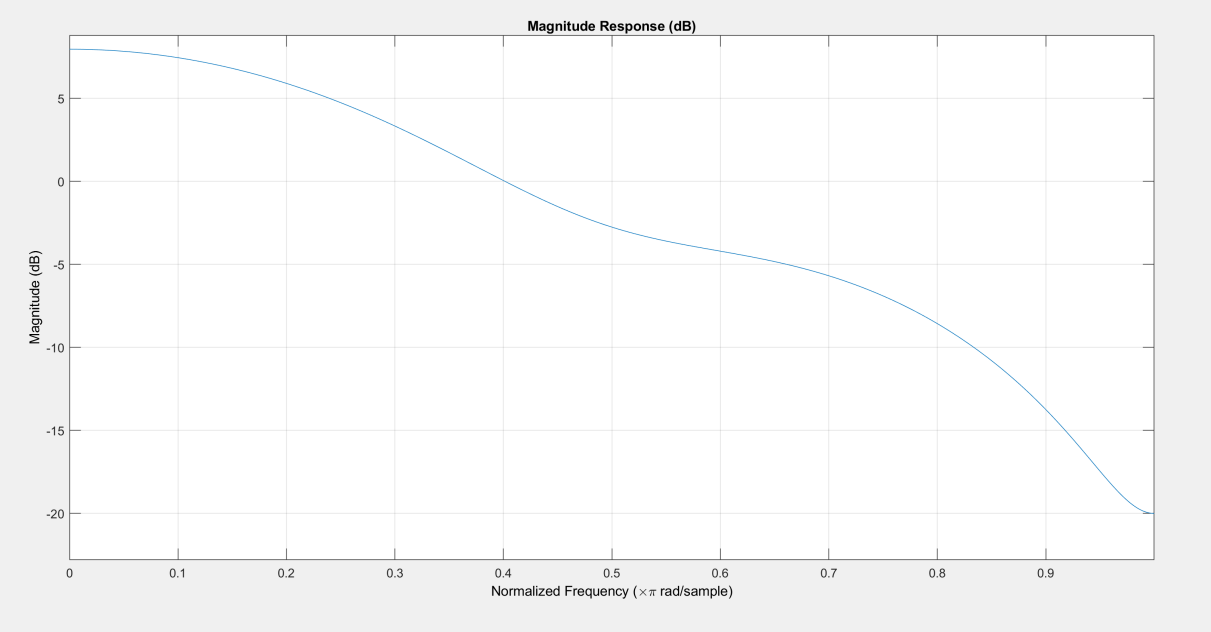


Figure B.2: Channel Magnitude Response

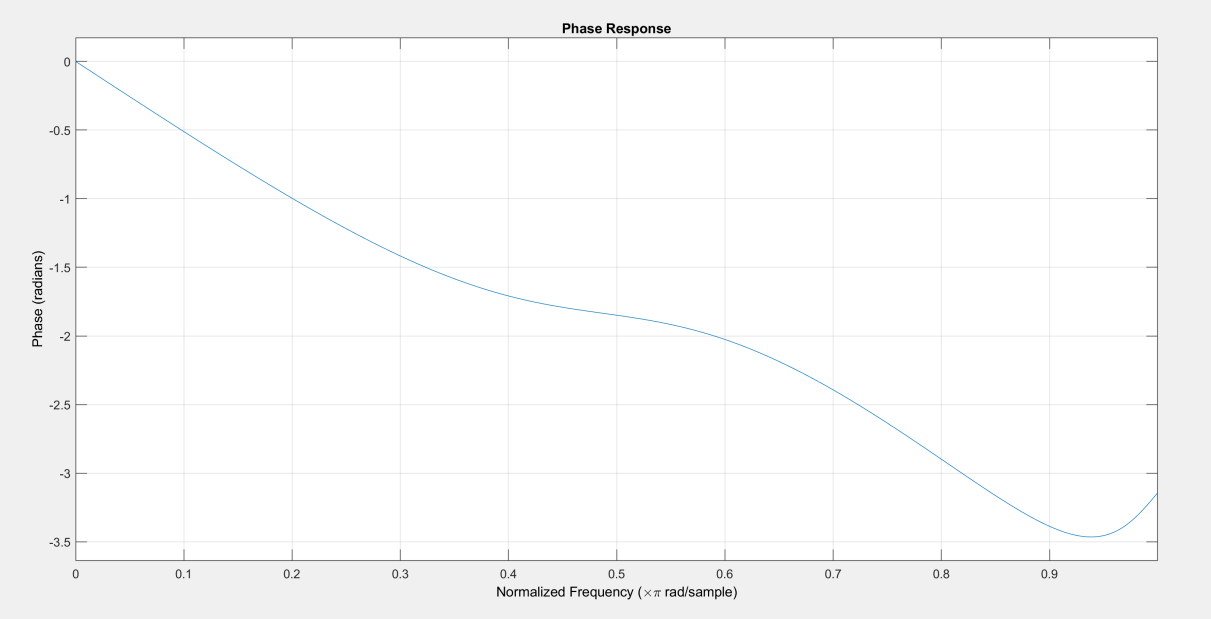


Figure B.3: Channel Phase Response

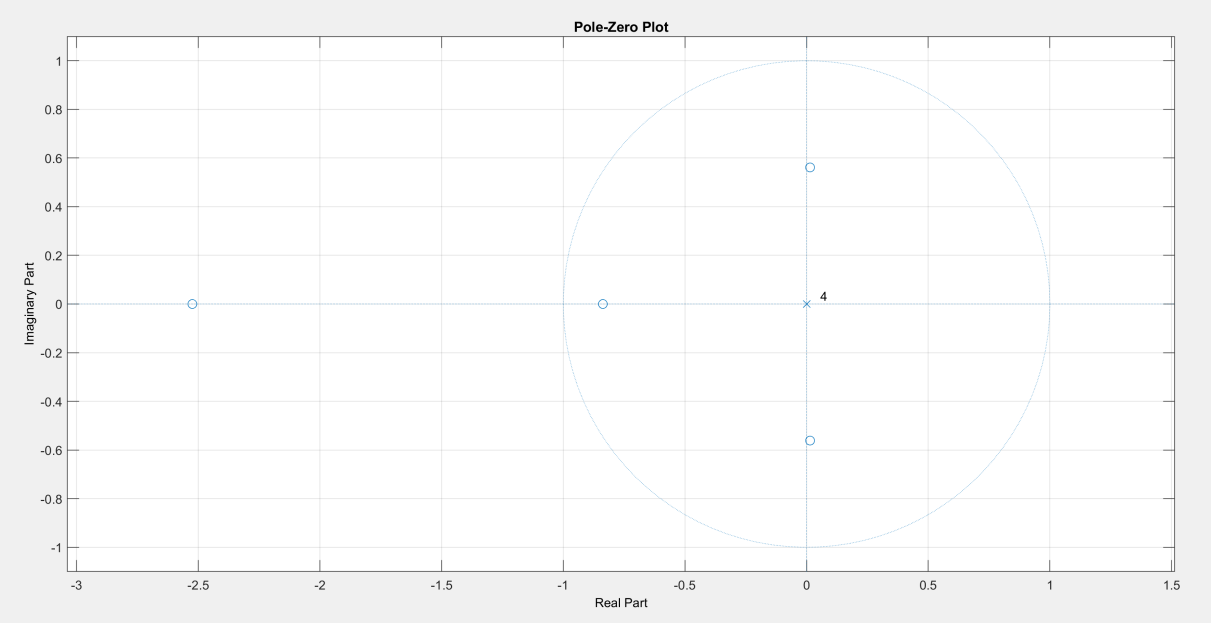


Figure B.4: Channel Pole Zero Plot

In the figures below, we show the characteristics of the equalizer and its performance with this particular channel.

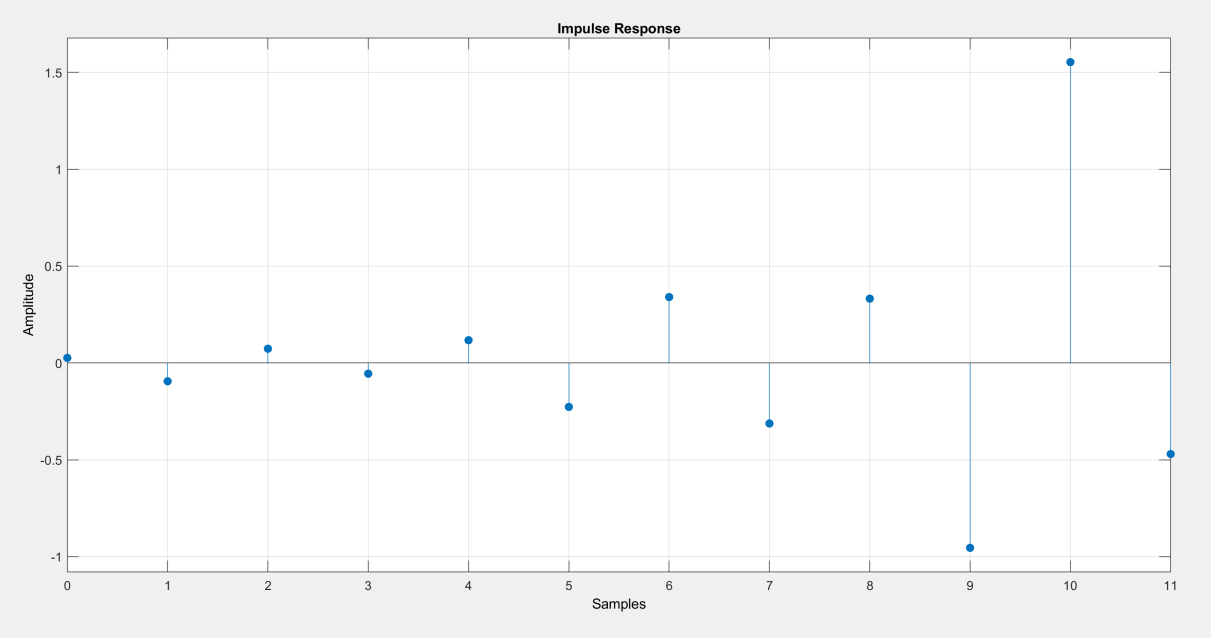


Figure B.5: Filter Impulse Response

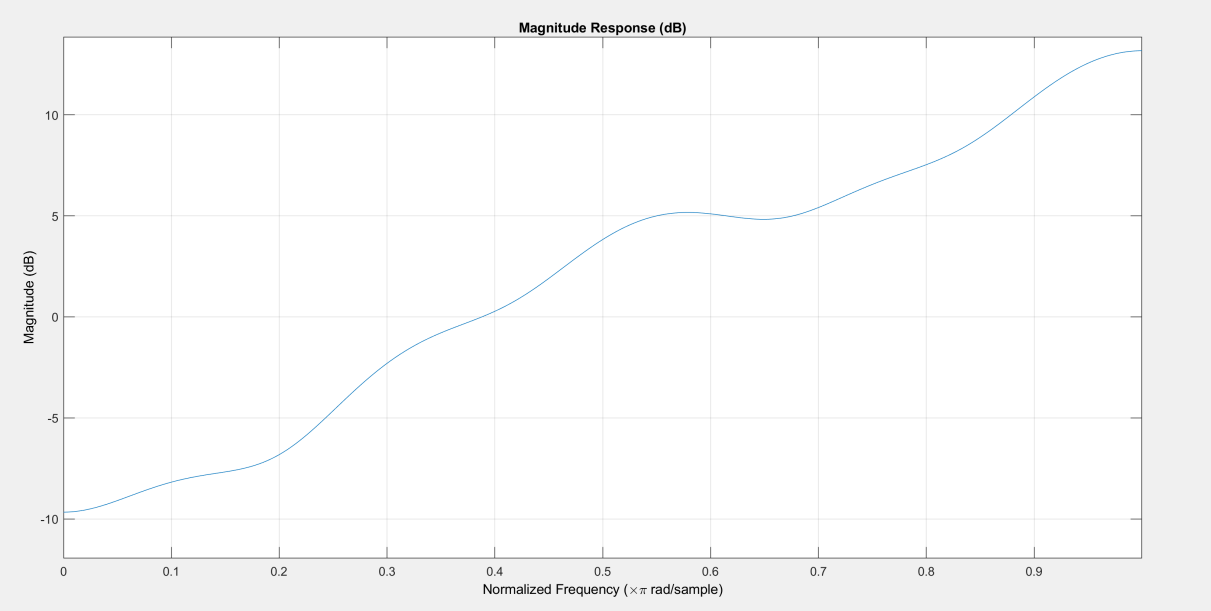


Figure B.6: Filter Magnitude Response

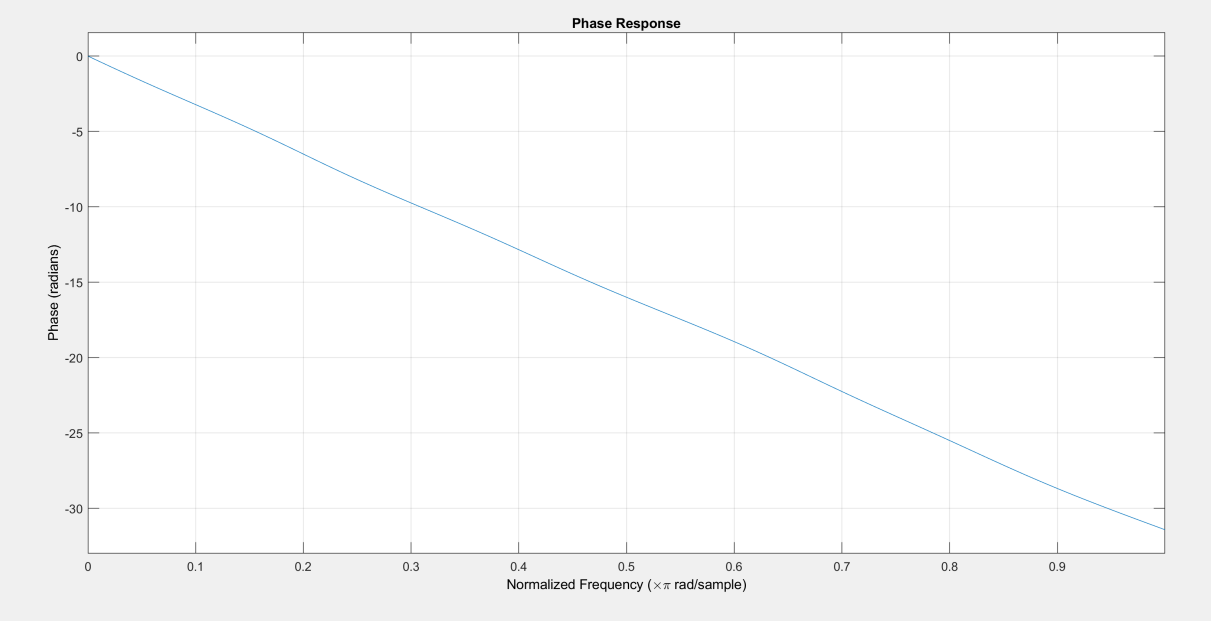


Figure B.7: Filter Phase Response

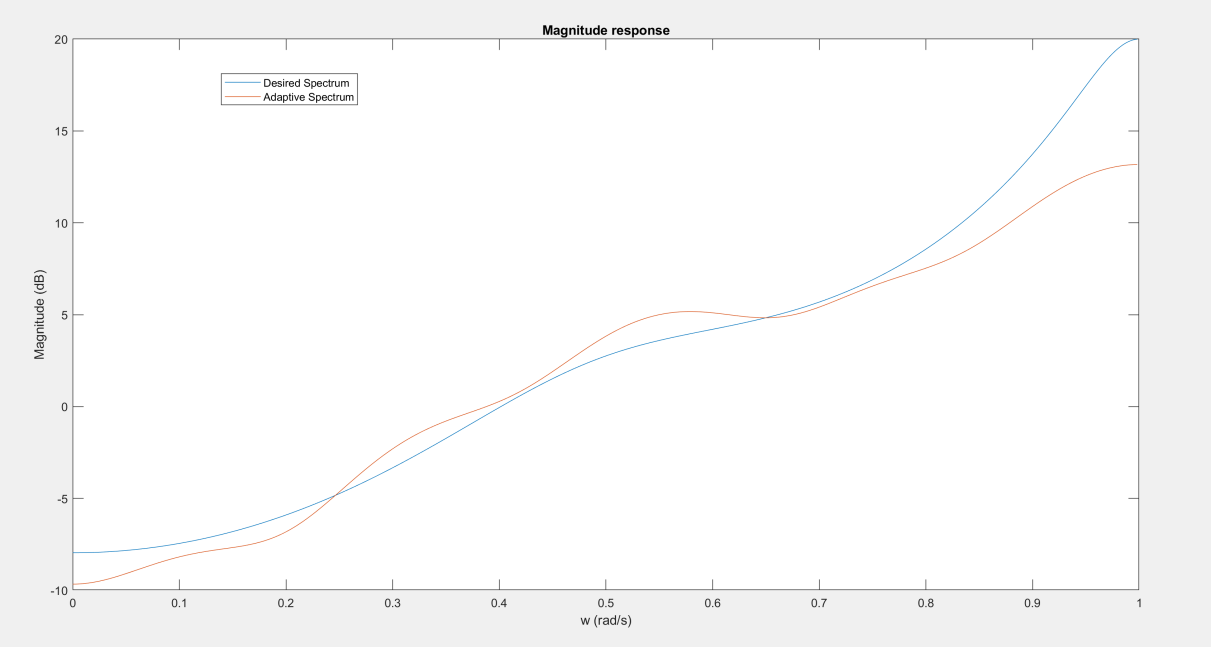


Figure B.8: Inverse Channel vs Filter Response

The above plot shows the magnitude response of the equalizer as compared to the inverse response of the channel, which is the ideal behavior for the equalizer to output the original input signal to the channel. In this baseline it is performing reasonably well, but performance seems to decline with higher frequencies.

We will now examine the performance of the equalizer under varying conditions. First we show figures of the equalizer performance when varying the SNR of the interference, with other design choices fixed. The baseline was with minimum interference of 20 dB so these plots will display performance with increased noise.

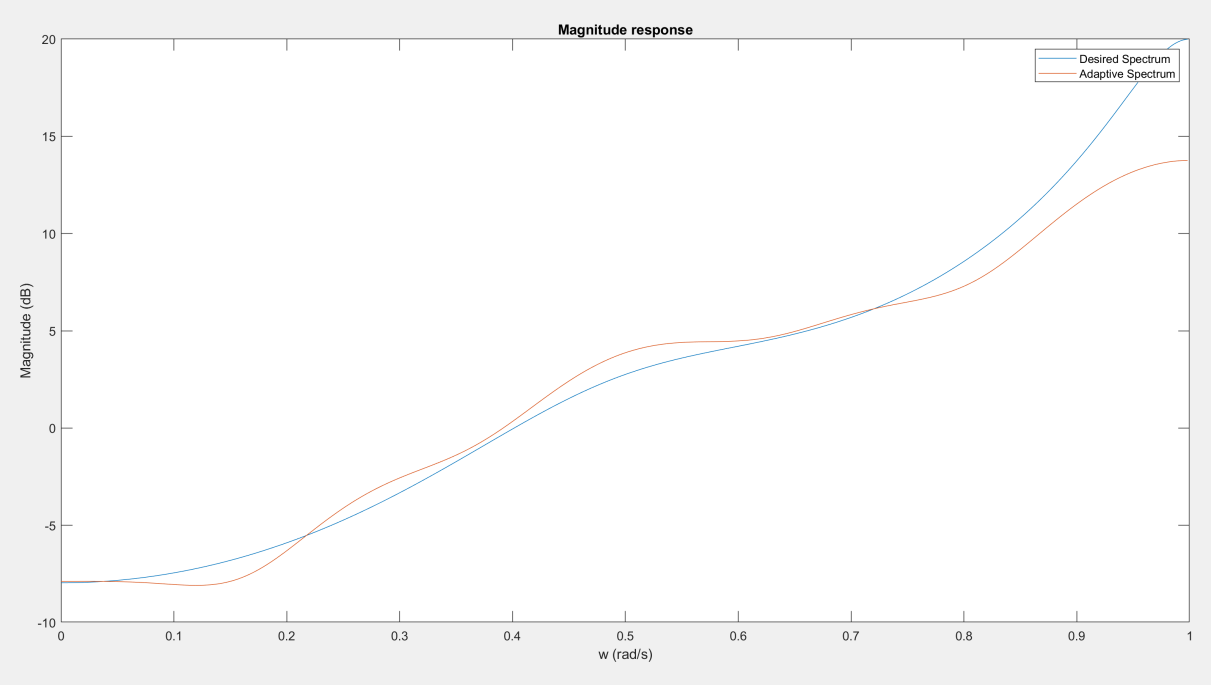


Figure B.9: Inverse Channel vs Filter Response, SNR=25

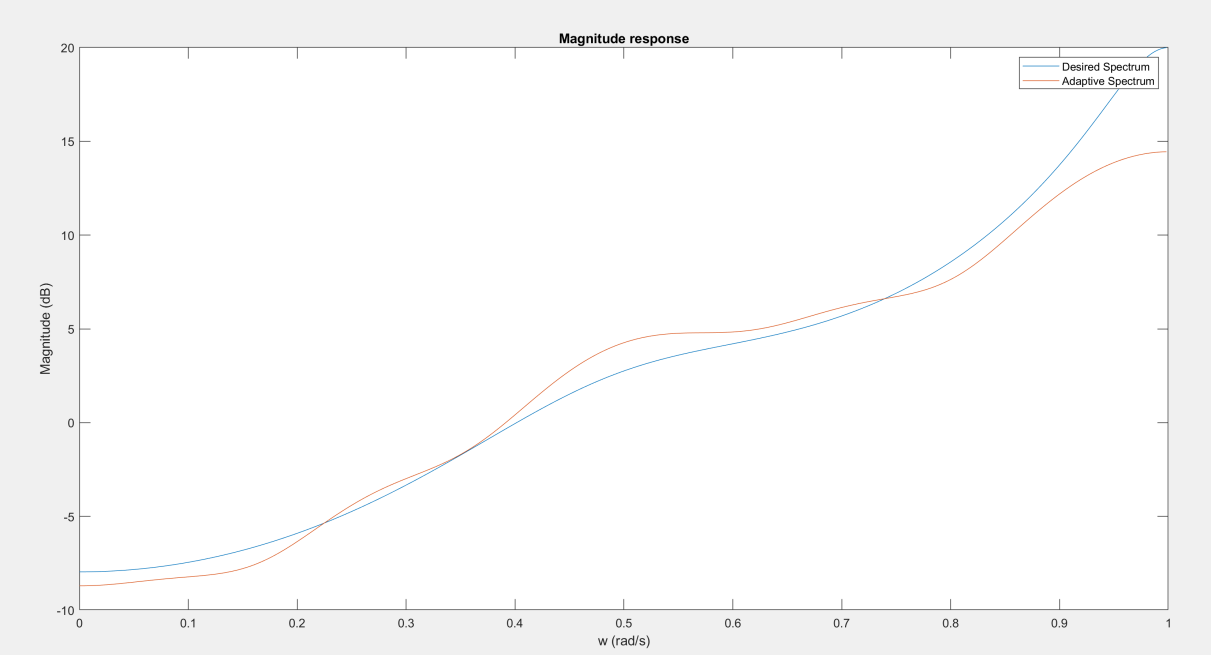


Figure B.10: Inverse Channel vs Filter Response, SNR=35

Next we examine performance with different learning rates, with all other parameters fixed. The baseline learning rate was 0.035.

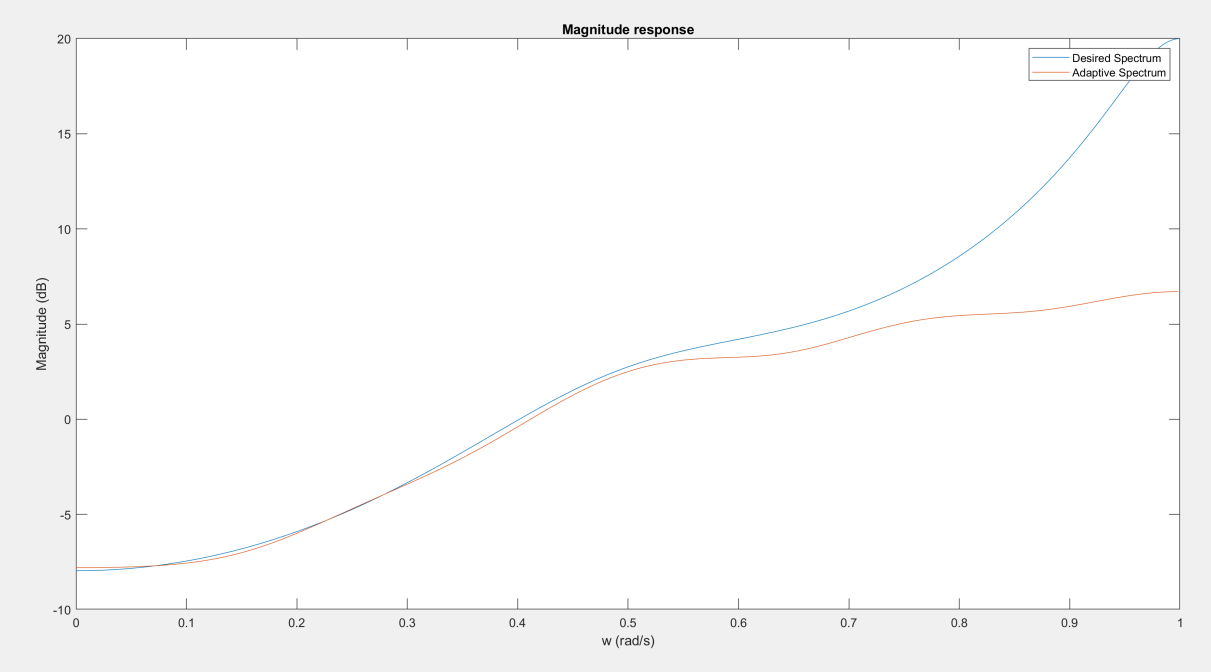


Figure B.11: Inverse Channel vs Filter Response, step size=0.01

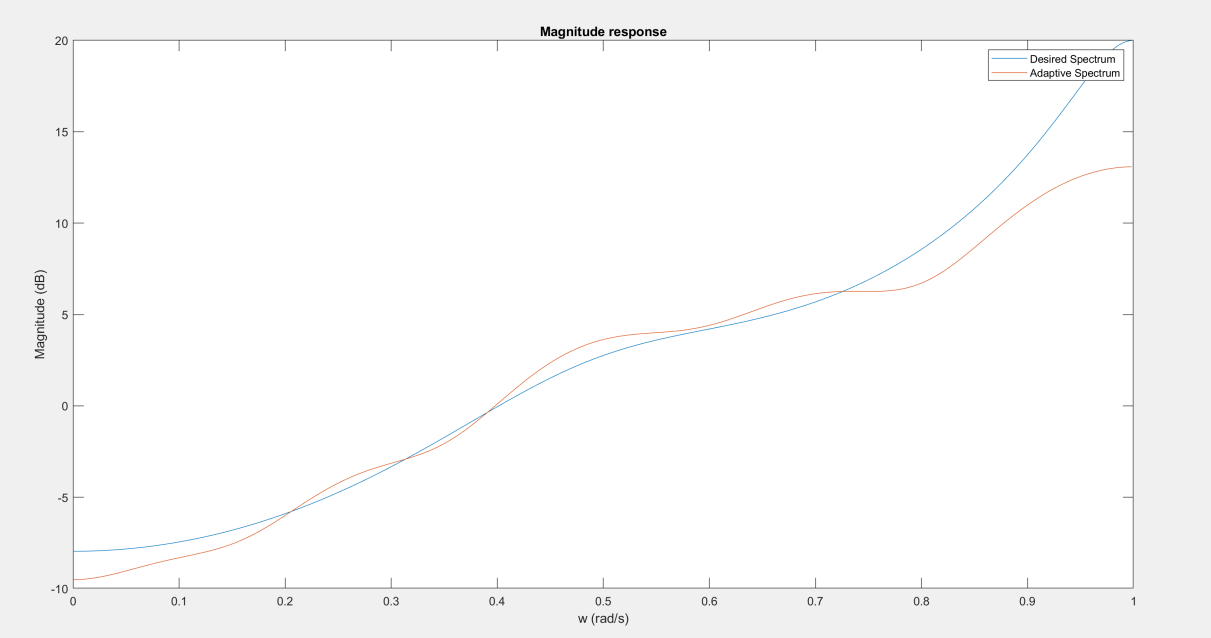


Figure B.12: Inverse Channel vs Filter Response, step size=0.05

Next we examine performance with different filter lengths, with all other parameters fixed. The baseline length was 12, filter order 11.

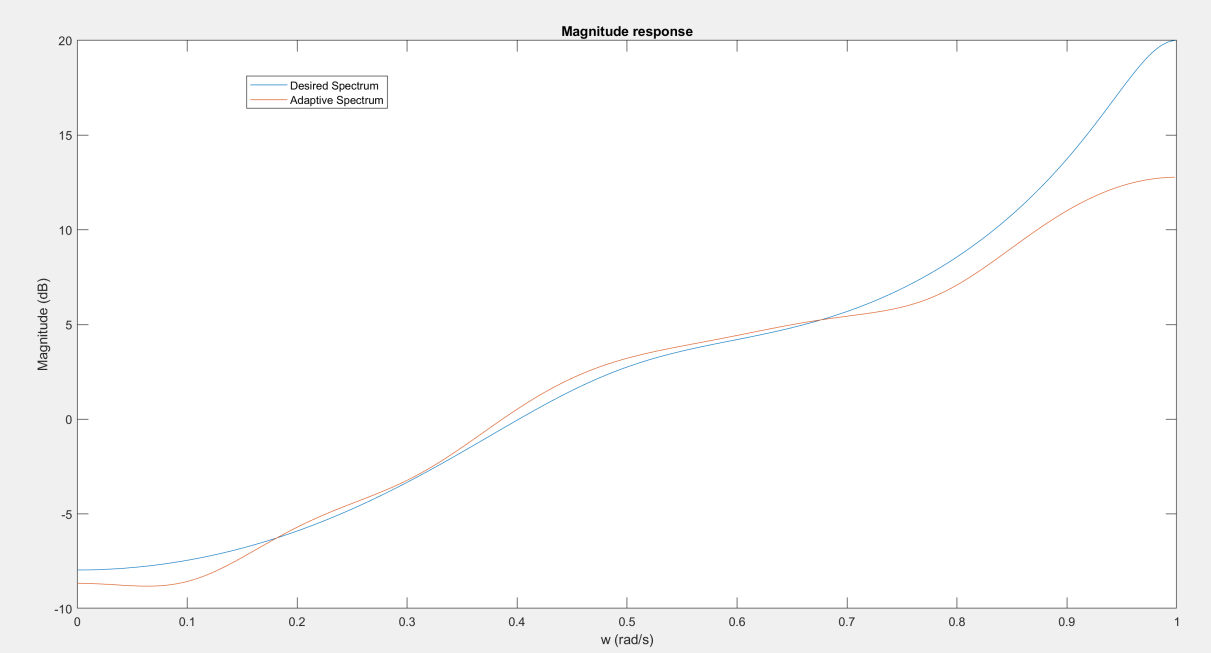


Figure B.13: Inverse Channel vs Filter Response, length=10

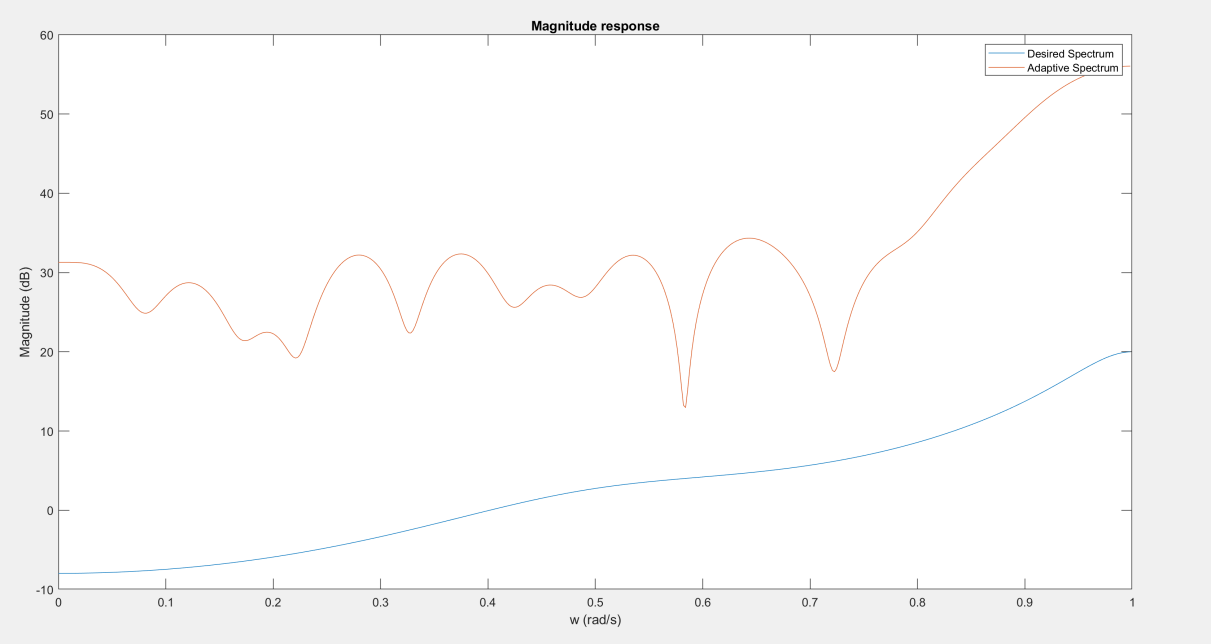


Figure B.14: Inverse Channel vs Filter Response, length=25

Below is a plot showing the performance when the filter is initialized to zeros instead of ones as the baseline performance is.

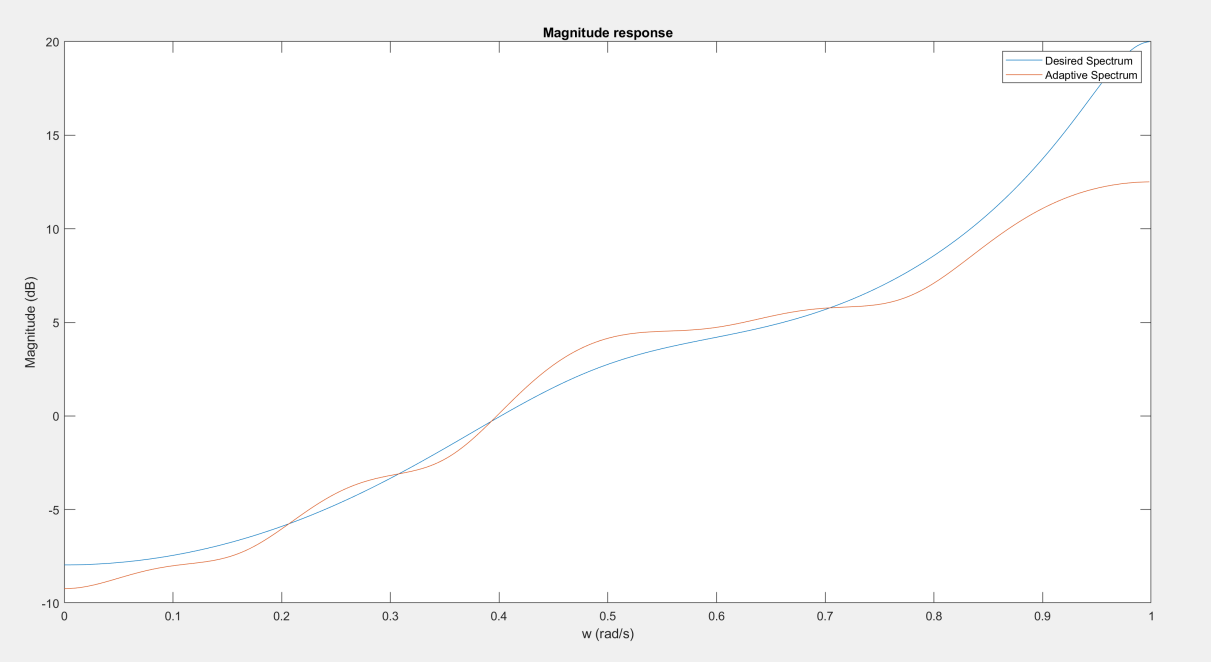


Figure B.15: Inverse Channel vs Filter Response, Initialized to zeros

Now, in the figures below we display the impulse and frequency responses of the overall LTI system between input and output, that is the cascaded system of channel and equalizer.

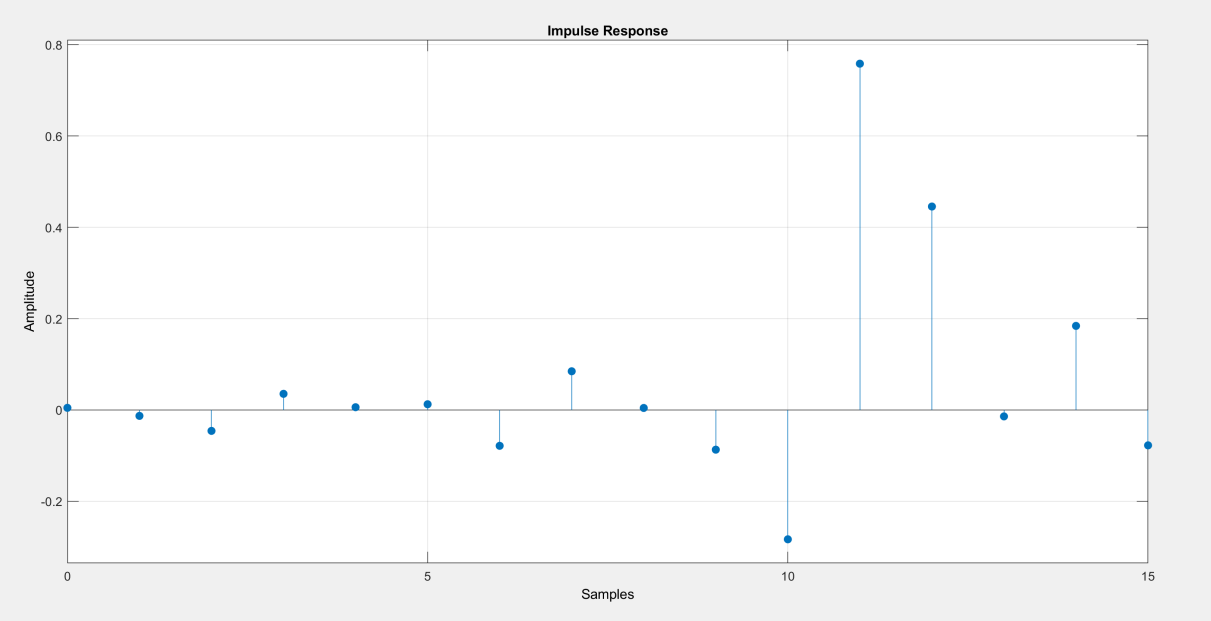


Figure B.16: Cascaded System Impulse Response

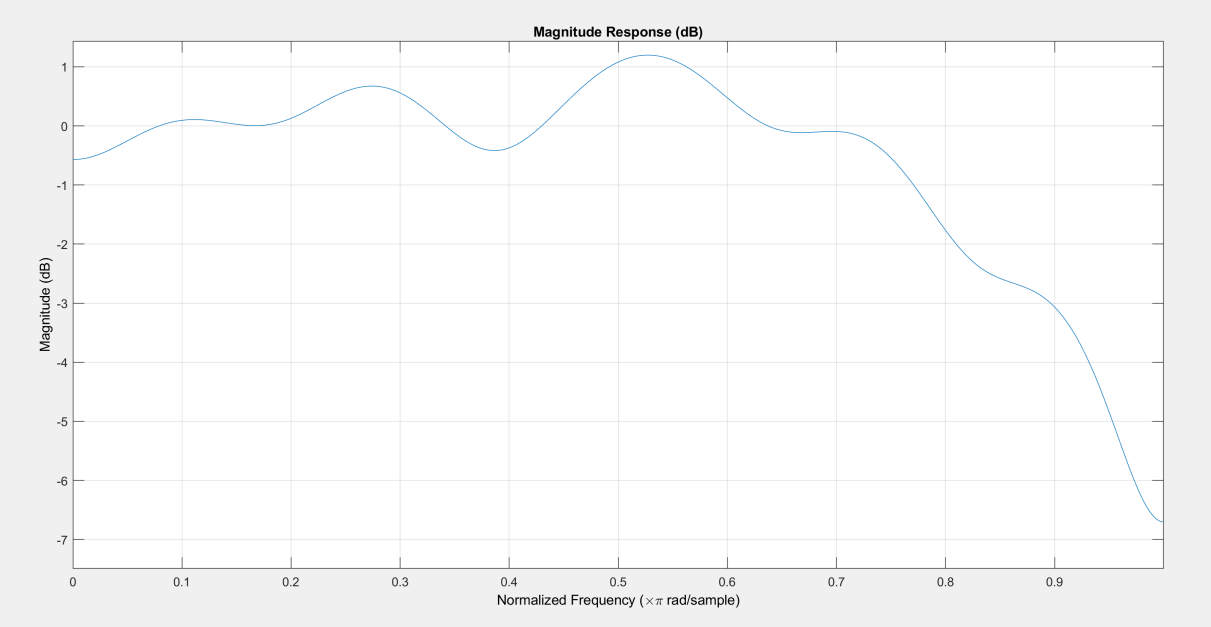


Figure B.17: Cascaded System Magnitude Response

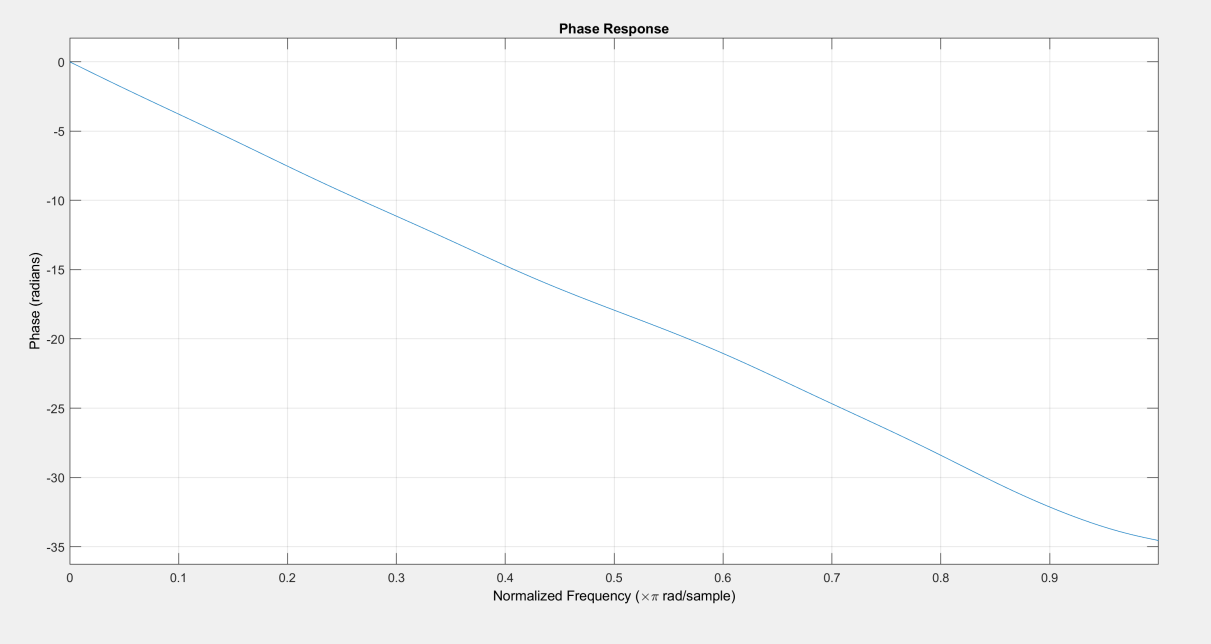


Figure B.18: Cascaded System Phase Response

Finally, to determine if output decisions were better after equalization we used a test signal containing values that were either 1 or -1 and applied the channel and the equalizer to it. We then measured the accuracy of the channel output and the accuracy of the equalizer output. We found that in most cases decisions about output bits were better in the un-equalized case as compared to the equalized case. The channel output had approximately a 60 percent accuracy in determining polarity of the bits and the equalized output had a 50 percent accuracy. Accuracy was measured by

counting how many values in each output signal matched the test samples and then dividing by the number of test samples. This could most likely be improved by training with more samples and using some optimization techniques to fine tune the design parameters such as filter order, learning rate, etc.

3 Part C

3.1 a

Our multirate filter bank was designed as shown in the figure below:

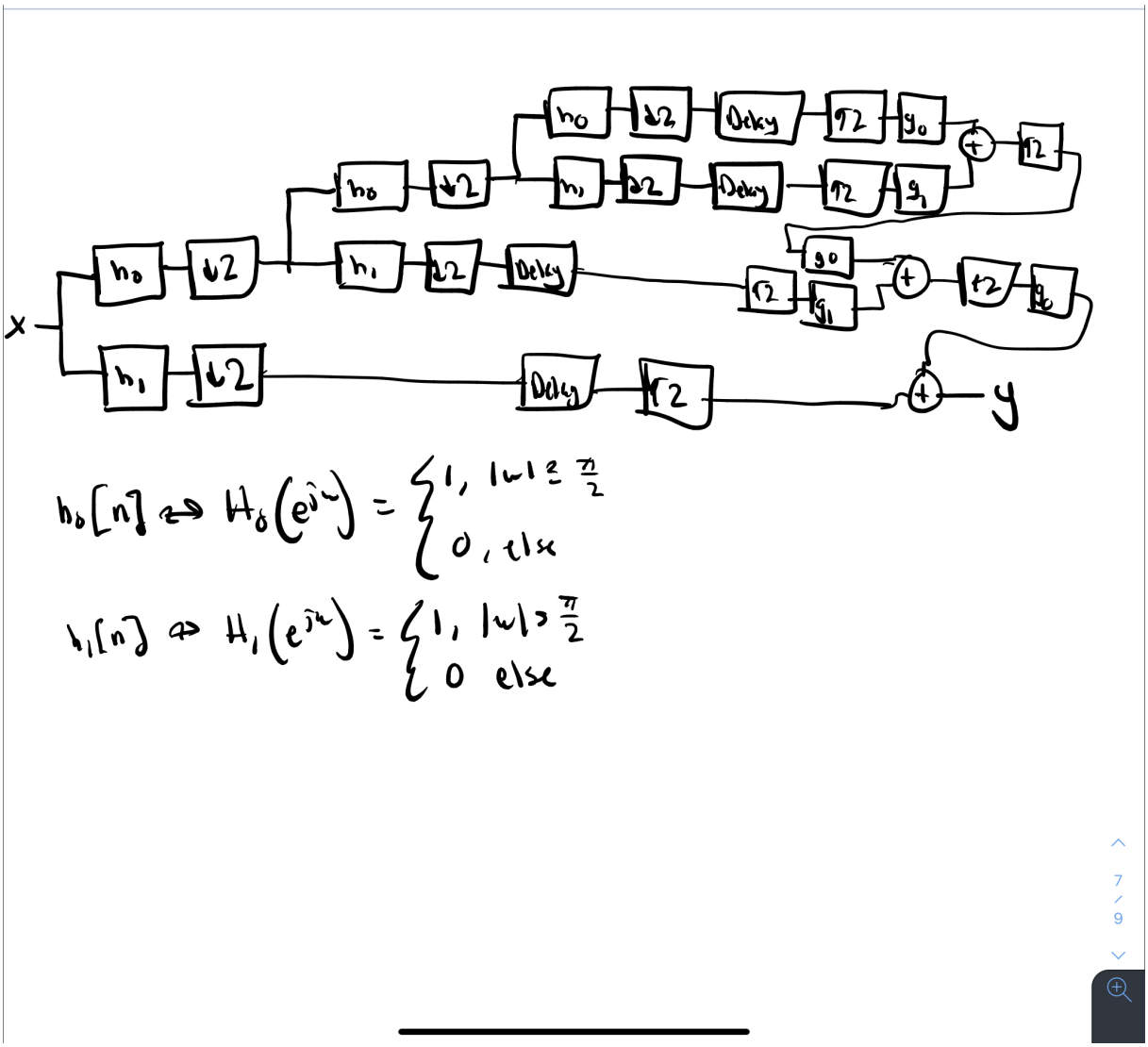


Figure C.1: Multirate Filter Bank

3.2 b

To test the filter bank, we used the following test signal: $x[n] = sin(0.2\pi n) + sin(0.4\pi n)$. After applying the filter bank to this signal, we received the below output:

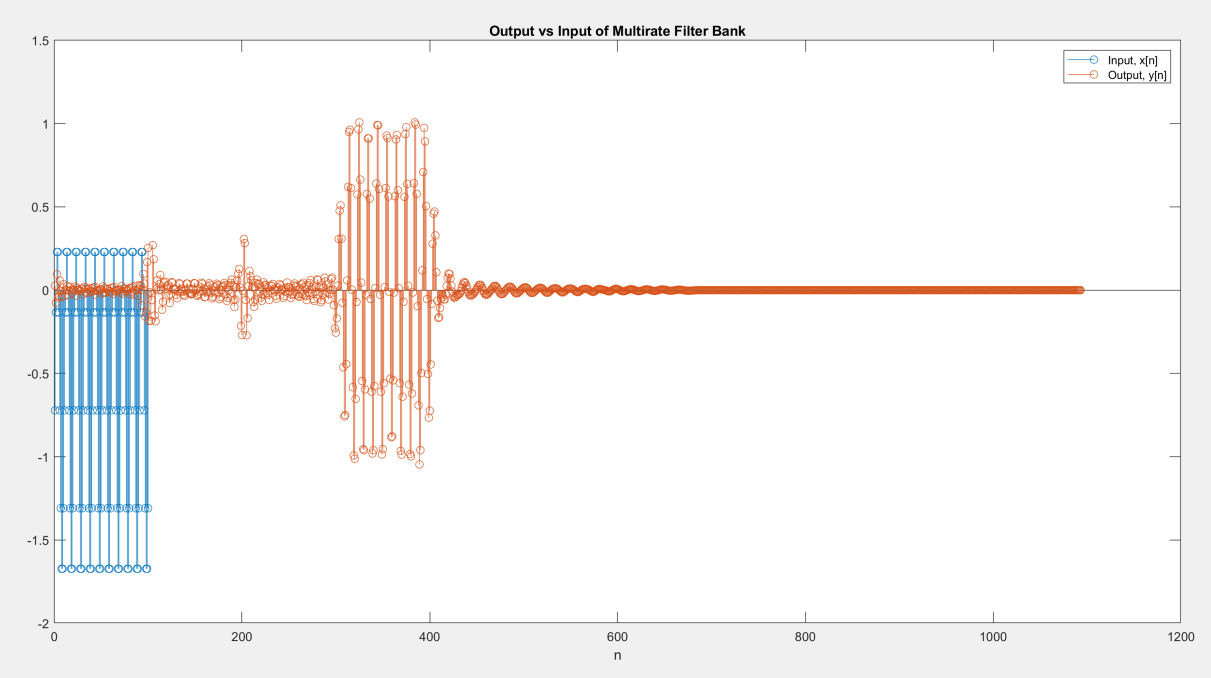


Figure C.2: Filter Output vs Input

Oddly it seems our output incurred a offset on the vertical axis, but it still retains the amplitude from the input. But for the most part the output resembles the input except for some noise, most likely coming from the fact that our filters in the bank are not ideal.

3.3 c

We added gains to each sub frequency band. Gain 1-2 are used for the lowest 2 bands, gain 3 is for the middle quarter band, and gain 4 is for the upper half band. Below shows the magnitude responses of the output when a speech signal from HW7 is input into the filter.

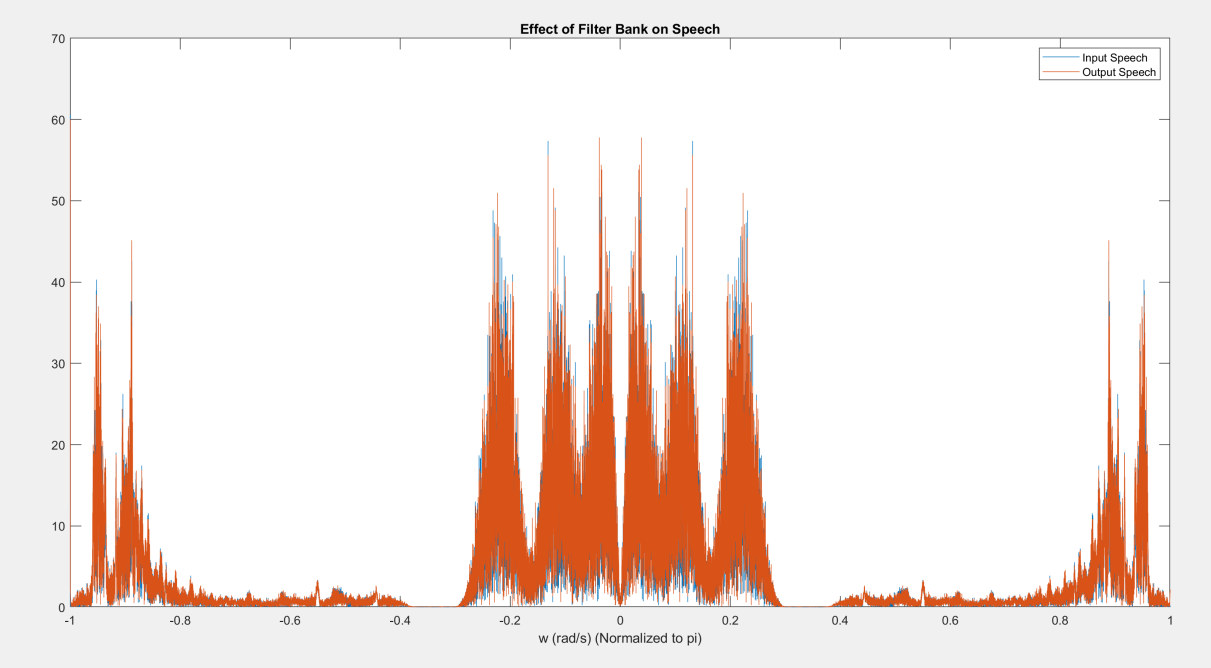


Figure C.3: All Gains set to 1

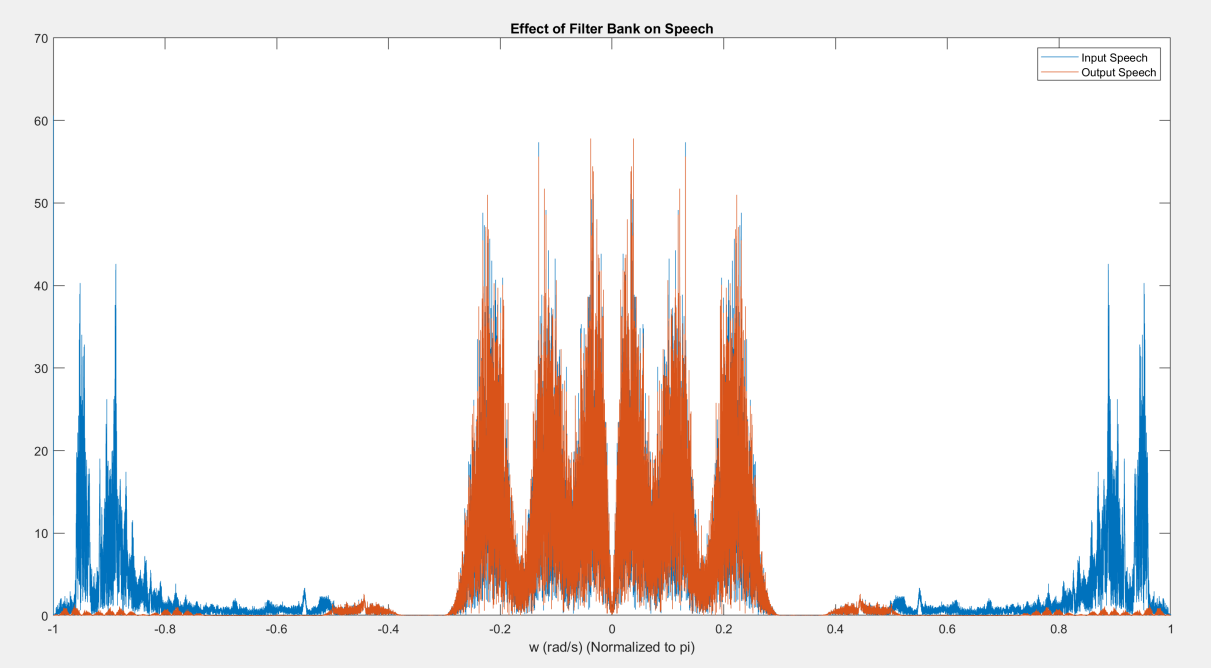


Figure C.4: Lower band gains set to 0.001

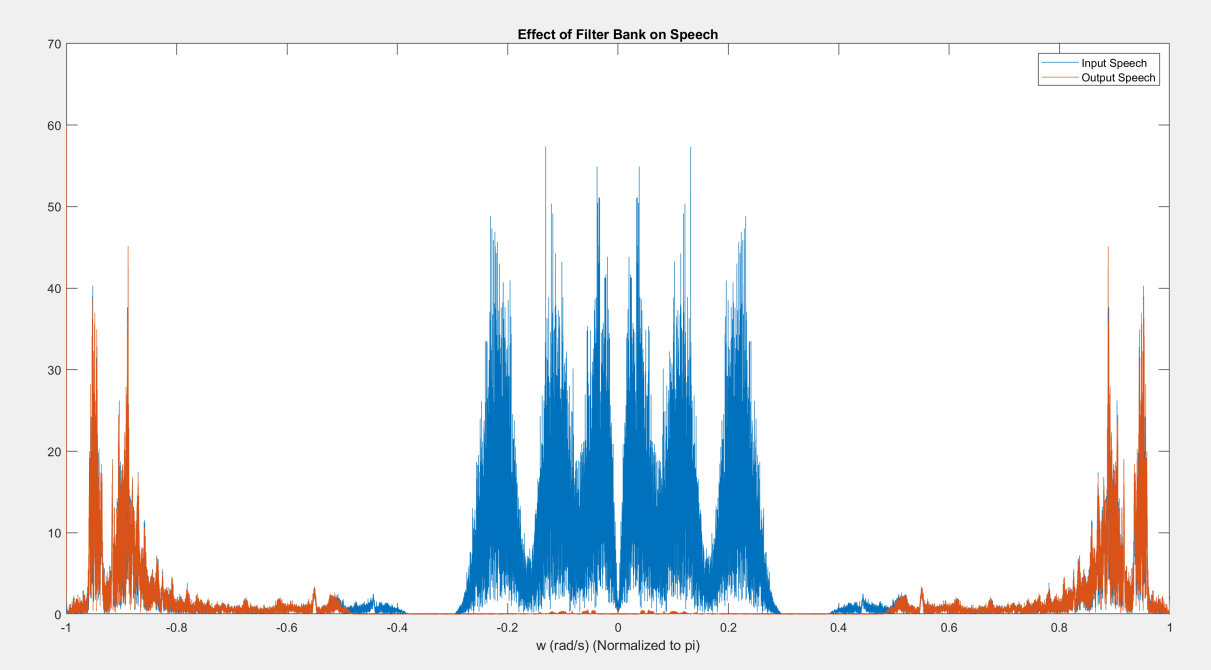


Figure C.5: Upper band gain set to 0.001

When listening to the outputs, we were able to suppress the noisy components of the original speech signal.

Adding additional small time delays does not seem to affect the frequency responses of the speech signal. Below shows a plot with speech filter output with unity gains and delay of 10.

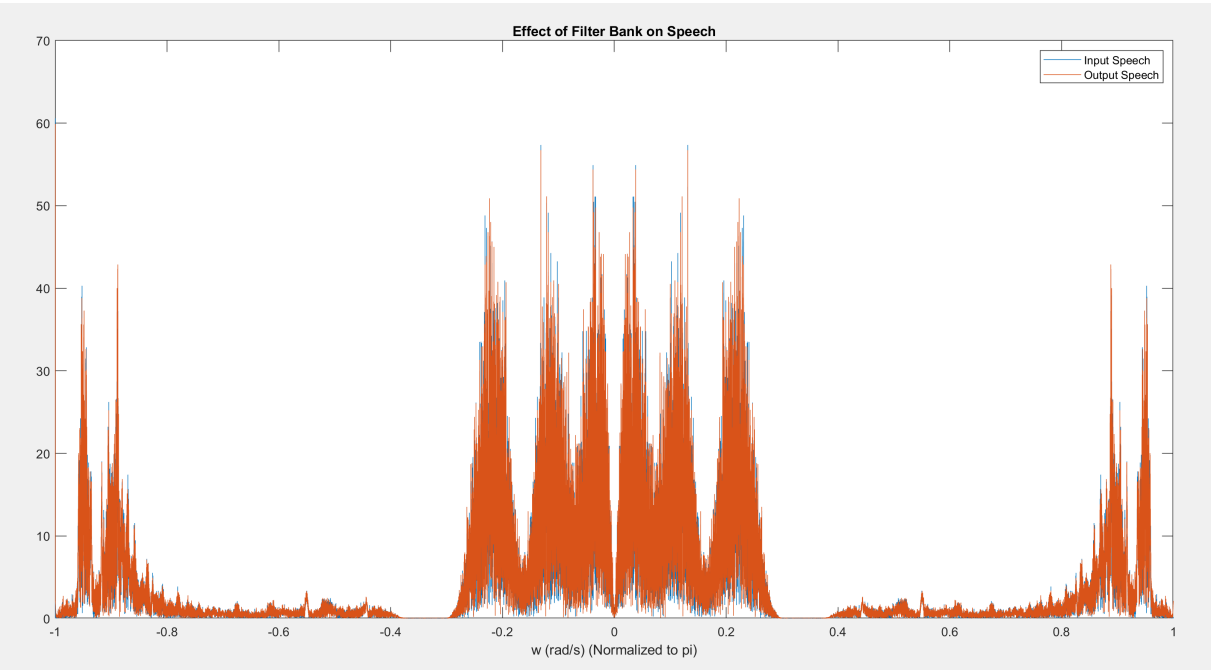


Figure C.5: Unity gains, delay=10

References

4 Appendix

Listing 1: PART A (a,b,c) Code

```
%Main file, used to run our functions
close all
clear variables
clc

%% ------------------------------------------------------------------------%%
%Part A
% constants
num_iter = 15000;
r_range = [0.85, 0.86, 0.87, 0.88, 0.89, 0.90, 0.91, 0.92, 0.93,...
          0.94, 0.95, 0.96, 0.97, 0.98];
r = 0.95;
% w_noise = .2*pi
% w_desire = pi
f_noise = 0.4; % frequency of sinusoid signal we want to remove(High inference)
f_desire = 0.2; % frequency of sinusoid signal we want
mu = 0.000009; %small mu reauired to converge

% sequences
e = zeros(1,num_iter); % intermediate Sequence
y = zeros(1, num_iter); % output sequence
a = zeros(1, num_iter); % Coefficients of a
range = 1:num_iter;
x_noise = 20 * sin(2*pi*f_noise*range); % sinusoidal noise with Strong interference
x_desired = sin(2*pi*f_desire*range); % signal desired without random noise)
% x_desired = x_desired + randn(size(range)); % Add noise to signal

x = x_noise + x_desired;
w = linspace(-pi, pi, num_iter);
% Plot of Signal Generated

figure()
plot(w/pi, abs(fftshift(fft(x))))
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ for our input
    signal,x'),'interpreter','latex');
title(sprintf('Frequency distribution, X(e^{jw}), for our input signal , where w_noise =
    %.2fpi,w_desire = %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');

% Simple Algorithm to minimize E(y[n]^2)
for n = 3:num_iter - 1
    e(n) = x(n) + a(n)*x(n - 1) + x(n - 2);
    y(n) = e(n) - r*a(n)*y(n-1) - r*r*y(n - 2);
    if ((a(n) >= -2) && (a(n) <= 2))
        a(n + 1) = a(n) - mu*y(n)*x(n-1);
    else
        a(n) = 0;
    end

end

a_fin = a(num_iter);

z = exp(1i*w);
H_adapt = ((1+a_fin*z.^(-1)+z.^(-2))./(1+r*a_fin*z.^(-1)+r^2*z.^(-2)));
% For visualizing the filter
fvtool(H_adapt)

subplot(2,1,1)
plot(w/pi, 20*log10(abs(H_adapt)));
xlabel('frequency, w/$\pi$', 'interpreter','latex');
```

```
ylabel(sprintf('Magnitude response of an Adaptive Notch filter, $H(e^{j\\omega})$
    /dB'),'interpreter','latex');
title(sprintf('Magnitude response of an Adaptive Notch filter, H(e^{jw}), where w_noise =
    %.2fpi,w_desire = %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');


% Extract the filtered region from y by taking the last 2k samples
% Extract the sample samples from x for comparison sake
y_filtered_extract = y(num_iter - 2000 :num_iter);
x_extract = x(num_iter - 2000 : num_iter);
% Take the fft of x and y and center them
Y_filter = fftshift(fft(y_filtered_extract, 2001));
X_extract = fftshift(fft(x_extract, 2001));

subplot(2,1,2)
w_filt = linspace(-pi, pi, 2001)/pi;
plot(w_filt, abs(X_extract))
hold on
plot(w_filt,abs(Y_filter));
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ and
    $Y(e^{j\\omega})$'),'interpreter','latex');
title(sprintf('Frequency distributions, X(e^{jw})and Y(e^{jw}) , where w_noise = %.2fpi,w_desire =
    %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');
hold off
l = legend('Input Signal, $X(e^{j\omega})$ ', 'Output signal,$Y(e^{j\omega})$');
set(l, 'interpreter', 'latex')
figure
plot(1:num_iter, a);
xlabel('time,n ')
ylabel(sprintf('Convergence Spectra, a'),'interpreter','latex');
title(sprintf('Convergence Spectra, a where w_noise = %.2fpi,w_desire = %.2fpi, r = %.2f',
    2*f_noise,2*f_desire,r),'interpreter','latex');


%% ------------------------------------------------------------------------%%
%Part B
% constants
num_iter = 15000;
r = 0.95;
mu = 0.000009; %small mu reauired to converge
for f = 0:0.03:0.45
    f_noise = f; % frequency of sinusoid signal we want to remove(High inference)
    f_desire = 0.2; % frequency of sinusoid signal we want


    % sequences
    e = zeros(1,num_iter); % intermediate Sequence
    y = zeros(1, num_iter); % output sequence
    a = zeros(1, num_iter); % Coefficients of a
    range = 1:num_iter;
    x_noise = 20 * sin(2*pi*f_noise*range); % sinusoidal noise with Strong interference
    x_desired = sin(2*pi*f_desire*range); % signal desired without random noise)
    % x_desired = x_desired + randn(size(range)); % Add noise to signal

    x = x_noise + x_desired;
    w = linspace(-pi, pi, num_iter);
    % Plot of Signal Generated

    figure()
    plot(w/pi, abs(fftshift(fft(x))))
    xlabel('frequency, w/$\pi$', 'interpreter','latex');
    ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ for our input
        signal,x'),'interpreter','latex');
```

```
    title(sprintf('Frequency distribution, X(e^{jw}), for our input signal , where w_noise =
        %.2fpi,w_desire = %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');

    % Simple Algorithm to minimize E(y[n]^2)
    for n = 3:num_iter - 1
        e(n) = x(n) + a(n)*x(n - 1) + x(n - 2);
        y(n) = e(n) - r*a(n)*y(n-1) - r*r*y(n - 2);
        if ((a(n) >= -2) && (a(n) <= 2))
            a(n + 1) = a(n) - mu*y(n)*x(n-1);
        else
            a(n) = 0;
        end

    end

    a_fin = a(num_iter);

    z = exp(1i*w);
    H_adapt = ((1+a_fin*z.^(-1)+z.^(-2))./(1+r*a_fin*z.^(-1)+r^2*z.^(-2)));
    % For visualizing the filter
%     fvtool(H_adapt)

    subplot(2,1,1)
    plot(w/pi, 20*log10(abs(H_adapt)));
    xlabel('frequency, w/$\pi$', 'interpreter','latex');
    ylabel(sprintf('Magnitude response of an Adaptive Notch filter, $H(e^{j\\omega})$
        /dB'),'interpreter','latex');
    title(sprintf('Magnitude response of an Adaptive Notch filter, H(e^{jw}), where w_noise =
        %.2fpi,w_desire = %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');


    % Extract the filtered region from y by taking the last 2k samples
    % Extract the sample samples from x for comparison sake
    y_filtered_extract = y(num_iter - 2000 :num_iter);
    x_extract = x(num_iter - 2000 : num_iter);
    % Take the fft of x and y and center them
    Y_filter = fftshift(fft(y_filtered_extract, 2001));
    X_extract = fftshift(fft(x_extract, 2001));

    subplot(2,1,2)
    w_filt = linspace(-pi, pi, 2001)/pi;
    plot(w_filt, abs(X_extract))
    hold on
    plot(w_filt,abs(Y_filter));
    xlabel('frequency, w/$\pi$', 'interpreter','latex');
    ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ and
        $Y(e^{j\\omega})$'),'interpreter','latex');
    title(sprintf('Frequency distributions, X(e^{jw})and Y(e^{jw}) , where w_noise =
        %.2fpi,w_desire = %.2fpi, r = %.2f', 2*f_noise,2*f_desire,r),'interpreter','latex');
    hold off
    l = legend('Input Signal, $X(e^{j\omega})$ ', 'Output signal,$Y(e^{j\omega})$');
    set(l, 'interpreter', 'latex')
    figure
    plot(1:num_iter, a);
    xlabel('time,n ')
    ylabel(sprintf('Convergence Spectra, a'),'interpreter','latex');
    title(sprintf('Convergence Spectra, a where w_noise = %.2fpi,w_desire = %.2fpi, r = %.2f',
        2*f_noise,2*f_desire,r),'interpreter','latex');
end
%% ------------------------------------------------------------------------%%
%Part C
% constants
r_one = 0.95;
r_two = 0.94;
mu_one = 0.000009; %small mu reauired to converge
```

```
mu_two = 0.000009; %small mu reauired to converge
num_iter = 15000;
f_noise_one = 0.4; % frequency of sinusoid signal we want to remove(High inference)
f_noise_two = 0.1; % frequency of second sinusoid signal we want to remove(High inference)
f_desire = 0.2; % frequency of sinusoid signal we want

% sequences
e_one = zeros(1,num_iter); % intermediate Sequence one
e_two = zeros(1,num_iter); % intermediate Sequence two
y_one = zeros(1, num_iter); % output sequence one
y_two = zeros(1, num_iter); % output sequence two
a_one = zeros(1, num_iter); % Coefficients of a one
a_two = zeros(1, num_iter); % Coefficients of a two
range = 1:num_iter;
x_noise_one = 20 * sin(2*pi*f_noise_one*range); % sinusoidal one noise with Strong interference
x_noise_two = 20 * sin(2*pi*f_noise_two*range); % sinusoidal two noise with Strong interference
x_desired = sin(2*pi*f_desire*range); % signal desired
x = x_noise_one + x_desired;
x_full = x_noise_one + x_noise_two + x_desired;
w = linspace(-pi, pi, num_iter);

% Plot of Signal Generated for Filter one
figure()
subplot(2,1,1)
plot(w/pi, abs(fftshift(fft(x))))
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ for our input
    signal,x(one)'),'interpreter','latex');
title(sprintf('Frequency distribution, X(e^{jw}), for our input signal into filter one, where
    w_noise_one = %.2fpi, w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two = %.2f ',
    2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');


% Simple Algorithm to minimize E(y[n]^2)

for n = 3:num_iter - 1
    e_one(n) = x(n) + a_one(n)*x(n - 1) + x(n - 2);
    y_one(n) = e_one(n) - r_one*a_one(n)*y_one(n-1) - r_one*r_one*y_one(n - 2);
    if ((a_one(n) >= -2) && (a_one(n) <= 2))
        a_one(n + 1) = a_one(n) - mu_one*y_one(n)*x(n-1);
    else
        a_one(n) = 0;
    end
end
z = exp(1i*w);
a_fin = a_one(num_iter);
H_adapt_one = ((1+a_fin*z.^(-1)+z.^(-2))./(1+r_one*a_fin*z.^(-1)+r_one^2*z.^(-2)));
x_two = x_noise_two + y_one;

% Plot of Signal Generated
subplot(2,1,2)
plot(w/pi, abs(fftshift(fft(x_two))))
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ for our input
    signal,x(two)'),'interpreter','latex');
title(sprintf('Frequency distribution, X(e^{jw}), for our input signal into filter two, where
    w_noise_one = %.2fpi, w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two = %.2f ',
    2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');

for n = 3:num_iter - 1
    e_two(n) = x_two(n) + a_two(n)*x_two(n - 1) + x_two(n - 2);
    y_two(n) = e_two(n) - r_two*a_two(n)*y_two(n-1) - r_two*r_two*y_two(n - 2);
    if ((a_two(n) >= -2) && (a_two(n) <= 2))
        a_two(n + 1) = a_two(n) - mu_two*y_two(n)*x_two(n-1);
    else
```

```
        a_two(n) = 0;
    end
end

a_fin = a_two(num_iter);
H_adapt_two = ((1+a_fin*z.^(-1)+z.^(-2))./(1+r_two*a_fin*z.^(-1)+r_two^2*z.^(-2)));
% For visualizing the filter
% fvtool(H_adapt)
H_adapt = H_adapt_one .* H_adapt_two;
figure()
subplot(2,1,1)
plot(w/pi, 20*log10(abs(H_adapt)));
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Magnitude response of an Adaptive Notch filter, $H(e^{j\\omega})$
    /dB'),'interpreter','latex');
title(sprintf('Magnitude response of an Adaptive Notch filter, H(e^{jw}), to reject two sine comps
    where w_noise_one = %.2fpi, w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two =
    %.2f ', 2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');

% Extract the filtered region from y by taking the last 2k samples
% Extract the sample samples from x for comparison sake
y_filtered_extract = y_two(num_iter - 2000 :num_iter);
x_extract = x_full(num_iter - 2000 : num_iter);
% Take the fft of x and y and center them
Y_filter = fftshift(fft(y_filtered_extract, 2001));
X_extract = fftshift(fft(x_extract, 2001));

subplot(2,1,2)
w_filt = linspace(-pi, pi, 2001)/pi;
plot(w_filt, abs(X_extract))
hold on
plot(w_filt,abs(Y_filter));
xlabel('frequency, w/$\pi$', 'interpreter','latex');
ylabel(sprintf('Frequency distributions, $X(e^{j\\omega})$ and
    $Y(e^{j\\omega})$'),'interpreter','latex');
title(sprintf('Frequency distributions, X(e^{jw})and Y(e^{jw}), where w_noise_one = %.2fpi,
    w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two = %.2f ',
    2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');
hold off
l = legend('Input Signal, $X(e^{j\omega})$ ', 'Output signal,$Y(e^{j\omega})$');
set(l, 'interpreter', 'latex')

figure()
subplot(2,1,1)
plot(1:num_iter, a_one);
xlabel('time,n ')
ylabel(sprintf('Convergence Spectra, a_one,'),'interpreter','latex');
title(sprintf('Convergence Spectra, a_one, for the adaptive cascaded filter part one where
    w_noise_one = %.2fpi, w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two = %.2f ',
    2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');

subplot(2,1,2)
plot(1:num_iter, a_two);
xlabel('time,n ')
ylabel(sprintf('Convergence Spectra, a_two'),'interpreter','latex');
title(sprintf('Convergence Spectra, a_two, for the adaptive filter two adaptive cascaded filter
    part two w_noise_one = %.2fpi, w_noise_two = %.2fpi, w_desire = %.2fpi, r_one = %.2f, r_two =
    %.2f ', 2*f_noise_one,2*f_noise_two, 2*f_desire,r_one, r_two),'interpreter','latex');

%% ------------------------------------------------------------------------%%
```