DSP PROJECT REPORT

PREPARED BY :
Azlaan Ranjha
Bilal Younas Chaudhary
Rana Abdullah Azhar

PREPARED FOR :
Sir Moiz

NUST COLLEGE OF EME

Title

Design and Implementation of Filters for EEG Signals using MATLAB

Project Statement

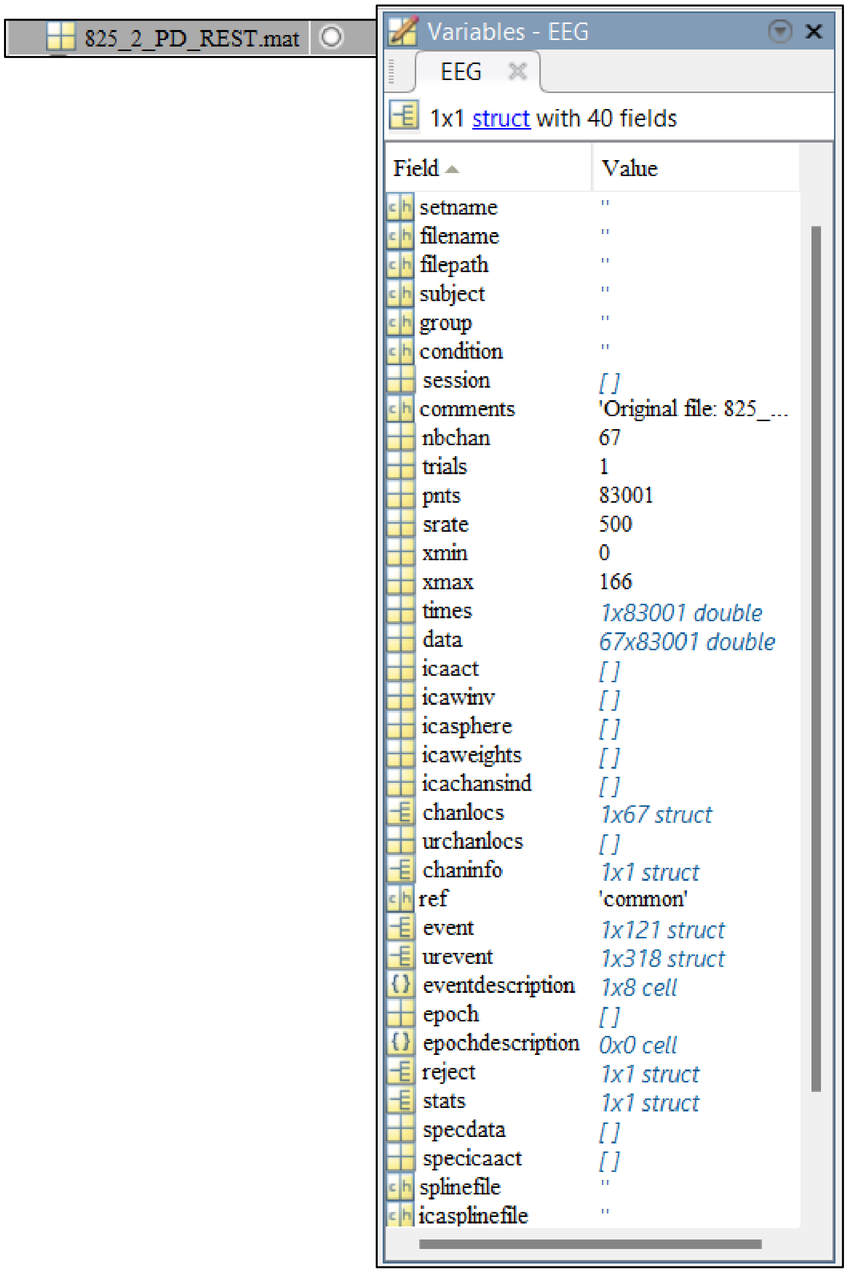
Q: Electroencephalography (EEG) is a critical diagnostic method that captures the brain's electrical activity through electrodes placed on the scalp. However, for effective analysis, EEG signals must be cleansed of various noise sources, including line noise. This project will guide you through the application of Digital Signal Processing (DSP) techniques, with a strong focus on signal filtering and noise reduction, using MATLAB.

The goal of this project is to provide firsthand experience in managing, evaluating, and processing EEG data. You will clean EEG data using several DSP techniques and analyze it across different brain activity frequency bands. By the end of the project, you will have a comprehensive understanding of filter design and implementation, as well as enhanced skills in analyzing and interpreting signal data for improved signal integrity and noise removal.

Objectives

- Learn to use MATLAB for EEG data processing.
- Detect and remove line noise from EEG data.
- Design and implement various filters (notch, high pass, low pass, band pass) tailored to specific EEG frequency bands: delta (0.5-4 Hz), theta (4-7 Hz), alpha (8-12 Hz), sigma (12-16 Hz), and beta (13-30 Hz).
- Apply these filters to selected EEG channels.
- Document the entire process in a comprehensive report, including MATLAB code and graphical representations of EEG signals pre- and post-filtering.

Dataset



825_2_PD_REST.mat

Variables - EEG

EEG

1x1 struct with 40 fields

Field	Value
setname	''
filename	''
filepath	''
subject	''
group	''
condition	''
session	[ ]
comments	'Original file: 825_...
nbchan	67
trials	1
pnts	83001
srate	500
xmin	0
xmax	166
times	1x83001 double
data	67x83001 double
icaact	[ ]
icawinv	[ ]
icasphere	[ ]
icaweights	[ ]
icachansind	[ ]
chanlocs	1x67 struct
urchanlocs	[ ]
chaninfo	1x1 struct
ref	'common'
event	1x121 struct
urevent	1x318 struct
eventdescription	1x8 cell
epoch	[ ]
epochdescription	0x0 cell
reject	1x1 struct
stats	1x1 struct
specdata	[ ]
specicaact	[ ]
splinefile	''
icasplinefile	''

Figure 1: For the project, we had our dataset in the form of the '852_2_PD_REST.mat' file. Inside the .mat file is a struct by the name of 'EEG.' The specifications and contents of the dataset struct can be clearly seen.

Notch Filter Frequency Selection:

```
% Load EEG data

load('825_2_PD_REST.mat');


% Sampling rate
fs = 500;


% Compute PSD using pwelch
channel_to_check = 1; % Channel to check for line noise
[psd, f] = pwelch(EEG.data(channel_to_check, :), [], [], [], fs);


% Find the power at 50 Hz and 60 Hz
[~, idx_50Hz] = min(abs(f - 50));
[~, idx_60Hz] = min(abs(f - 60));
power_50Hz = psd(idx_50Hz);
power_60Hz = psd(idx_60Hz);


% Determine which frequency has higher power
if power_50Hz > power_60Hz
    f0 = 50; % Line noise frequency
else
    f0 = 60; % Line noise frequency
end


% Display the chosen frequency for notch filter
fprintf('Chosen frequency for notch filter: %d Hz\n', f0);
```

Output:

```
Chosen frequency for notch filter: 60 Hz
```

```
% Define the notch filter to remove the chosen line noise
Q = 30; % Quality factor
[notch_b, notch_a] = iirnotch(f0/(fs/2), f0/(fs/2)/Q);


% Apply the notch filter to each channel
filtered_data_notch = zeros(size(EEG.data));
```

```
for i = 1:size(EEG.data, 1)
    filtered_data_notch(i, :) = filtfilt(notch_b, notch_a, EEG.data(i, :));
end


% Plot original vs. notch-filtered signal for one channel
figure;
channel_to_plot = 1;


% Plot original signal
subplot(2, 2, 1);
plot(EEG.data(channel_to_plot, :));
title('Original Signal');
xlabel('Sample Number');
ylabel('Amplitude');


% Compute and plot the FFT of the original signal
subplot(2, 2, 2);
fft_original = fft(EEG.data(channel_to_plot, :));
fft_original_shifted = fftshift(fft_original);
f_fft_original = (-length(fft_original)/2:length(fft_original)/2-1) * (fs /
length(fft_original));
plot(f_fft_original, 20*log10(abs(fft_original_shifted)));
title('FFT of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');


% Plot notch-filtered signal
subplot(2, 2, 3);
plot(filtered_data_notch(channel_to_plot, :));
title('Notch Filtered Signal (50 Hz)');
xlabel('Sample Number');
ylabel('Amplitude');


% Compute and plot the FFT of the signal after notch filter
subplot(2, 2, 4);
fft_notch_filtered = fft(filtered_data_notch(channel_to_plot, :));
fft_notch_filtered_shifted = fftshift(fft_notch_filtered);
f_fft_notch_filtered = (-
length(fft_notch_filtered)/2:length(fft_notch_filtered)/2-1) * (fs /
length(fft_notch_filtered));
plot(f_fft_notch_filtered, 20*log10(abs(fft_notch_filtered_shifted)));
title('FFT after Notch Filtering');
xlabel('Frequency (Hz)');
```

```
ylabel('Magnitude (dB)');
```

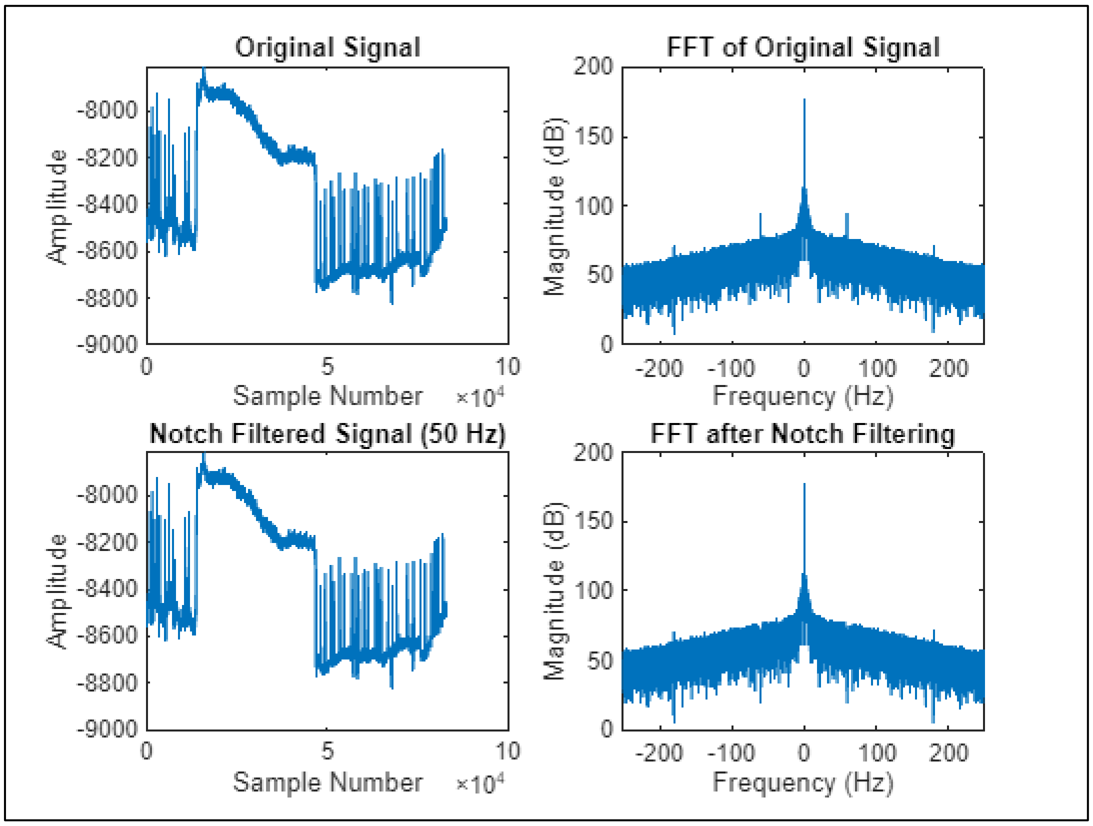
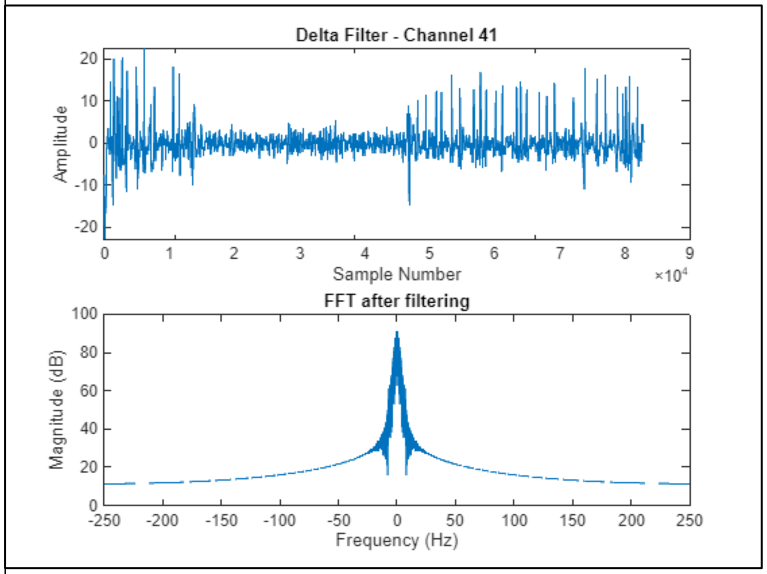
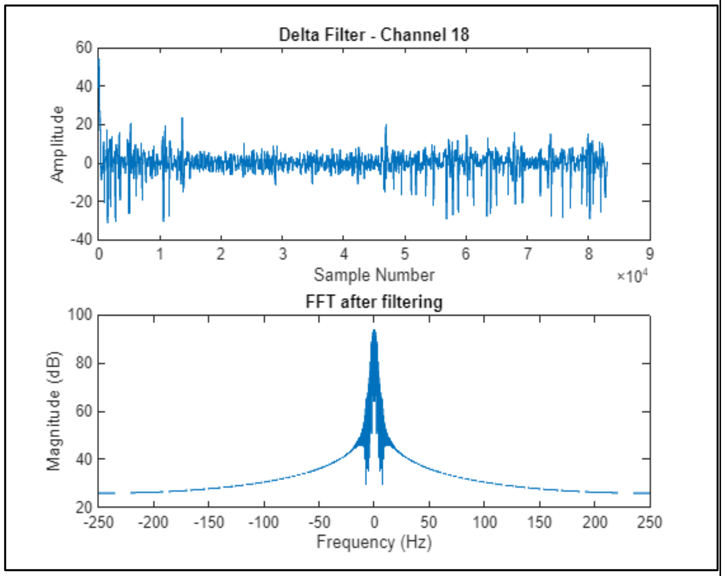


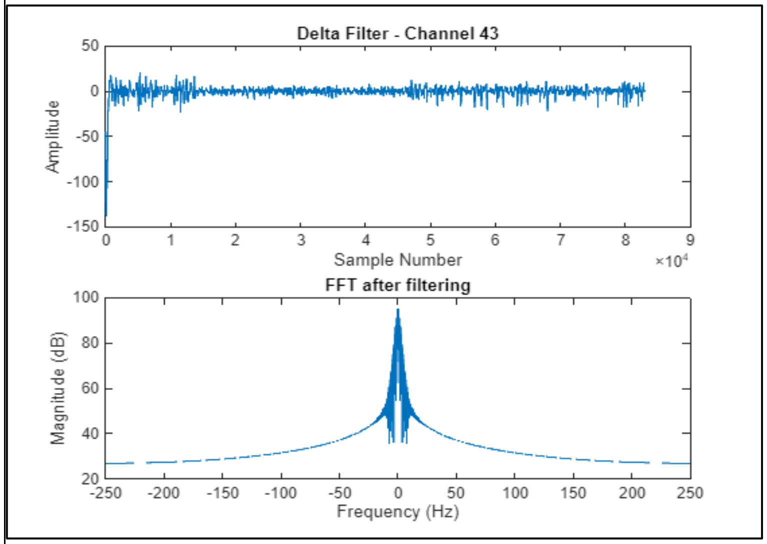
Figure 2: The Original Signal in Time and Frequency Domain. Also, the signal after the notch filter in both time and frequency domain. It can be observed that the spike in FFT at 60 Hz, is removed after the notch filter.

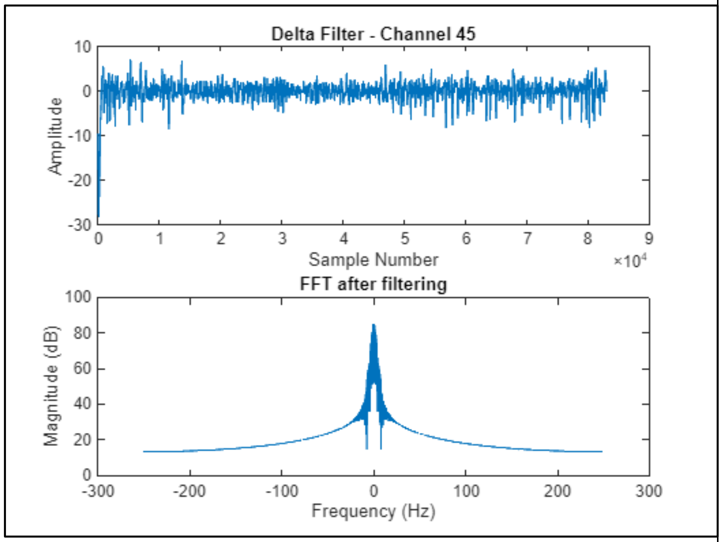
```
% Store the notch filtered data back into the EEG struct for further processing
EEG.data = filtered_data_notch;


% Filter Design and Implementation
% Define the filter bands
bands = {'delta', 'theta', 'alpha', 'sigma', 'beta'};
band_ranges = [0.5 4; 4 7; 8 12; 12 16; 13 30];


% Initialize the filters struct
filters = struct();
for k = 1:length(bands)
    filters.(bands{k}) = designfilt('bandpassiir', 'FilterOrder', 4, ...
        'HalfPowerFrequency1', band_ranges(k, 1), ...
        'HalfPowerFrequency2', band_ranges(k, 2), ...
        'SampleRate', fs);
end
```

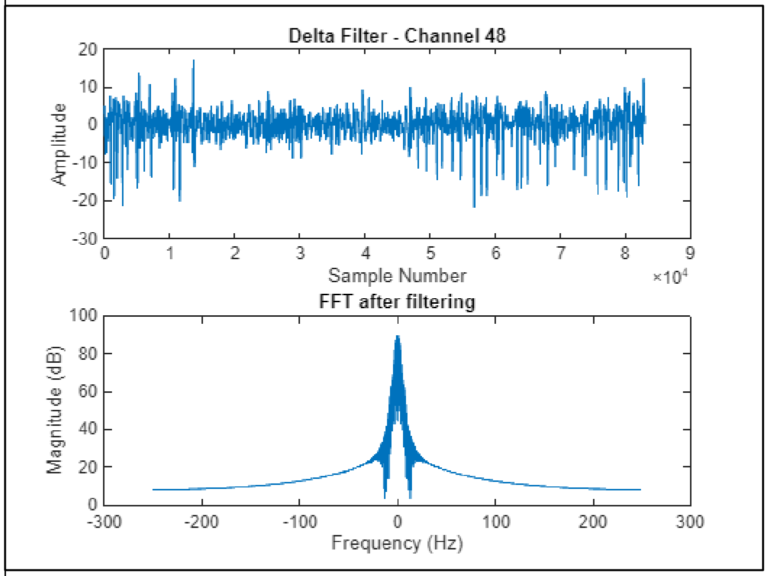
```
% Apply filters to the selected channels and plot
% Select 10 random channels
num_channels = size(EEG.data, 1);
random_channels = randperm(num_channels, 10);


filtered_signals = struct();
all_filters = [bands, {'notch'}];


for k = 1:length(all_filters)
    filter_name = all_filters{k};
    filtered_signals.(filter_name) = zeros(length(random_channels),
size(EEG.data, 2));
    for i = 1:length(random_channels)
        if strcmp(filter_name, 'notch')
            filtered_signals.(filter_name)(i, :) = filtfilt(notch_b, notch_a,
EEG.data(random_channels(i), :));
        else
            filtered_signals.(filter_name)(i, :) =
filtfilt(filters.(filter_name), EEG.data(random_channels(i), :));
        end

        % Plot FFT of the channel after applying the filter
        figure;
        subplot(2, 1, 1);
        plot(filtered_signals.(filter_name)(i, :));
        title([upper(filter_name(1)) filter_name(2:end) ' Filter - Channel '
num2str(random_channels(i))]);
        xlabel('Sample Number');
        ylabel('Amplitude');


        % Compute and plot the FFT
        subplot(2, 1, 2);
        fft_data = fft(filtered_signals.(filter_name)(i, :));
        fft_data_shifted = fftshift(fft_data);
        f_fft = (-length(fft_data)/2:length(fft_data)/2-1) * (fs /
length(fft_data));
        plot(f_fft, 20*log10(abs(fft_data_shifted)));
        title('FFT after filtering');
        xlabel('Frequency (Hz)');
        ylabel('Magnitude (dB)');
    end
end
```

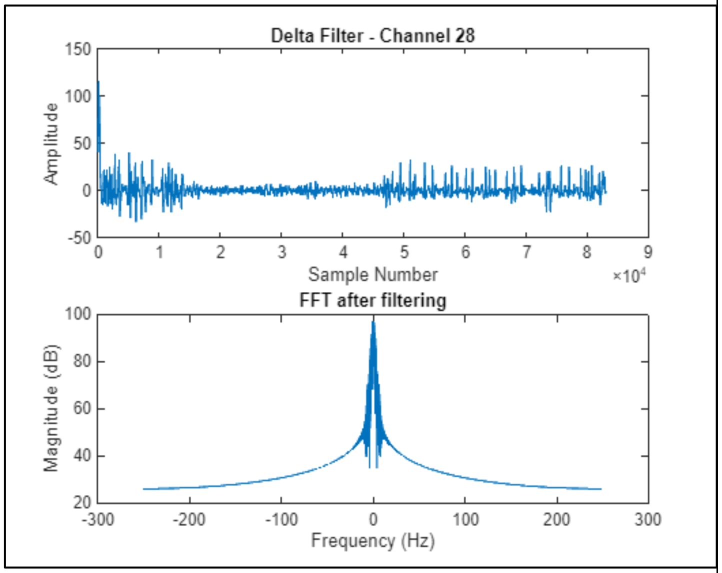

Delta Filter - Channel 41
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

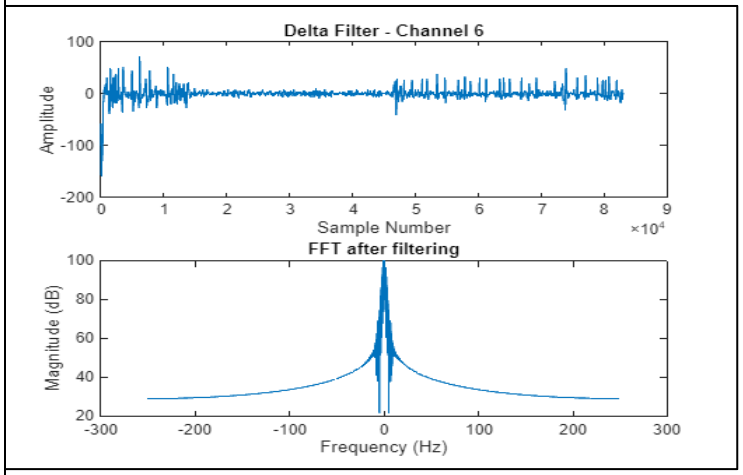

Delta Filter - Channel 18
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

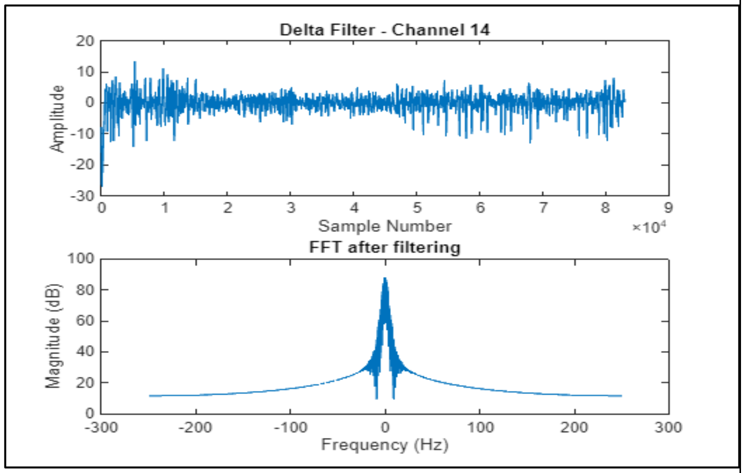

Delta Filter - Channel 43
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

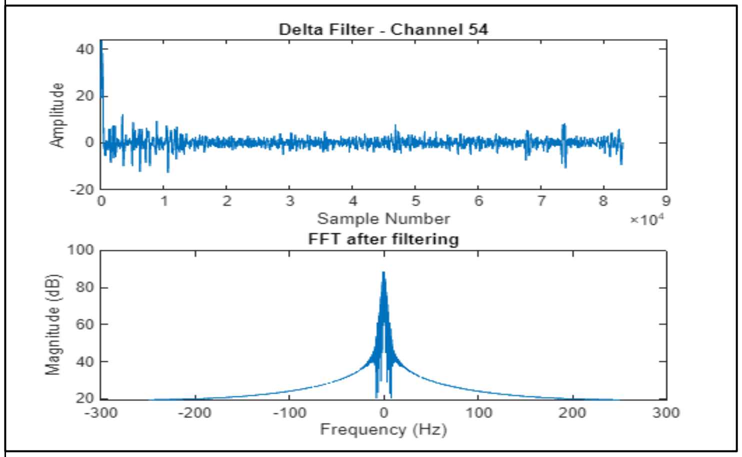

Delta Filter - Channel 45
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Delta Filter - Channel 48
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Delta Filter - Channel 28
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)







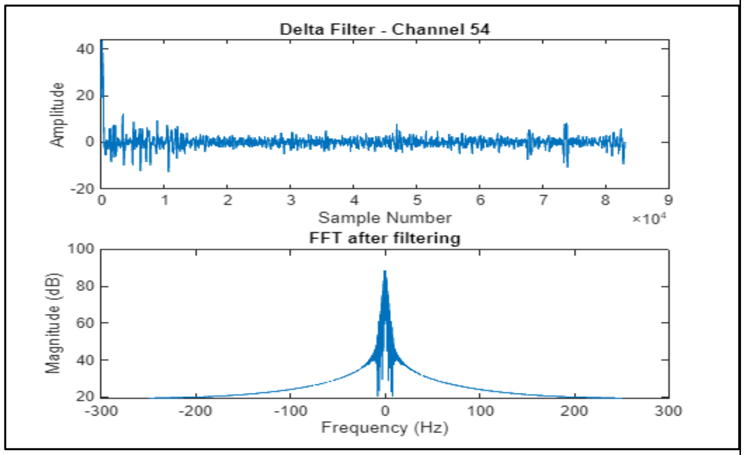
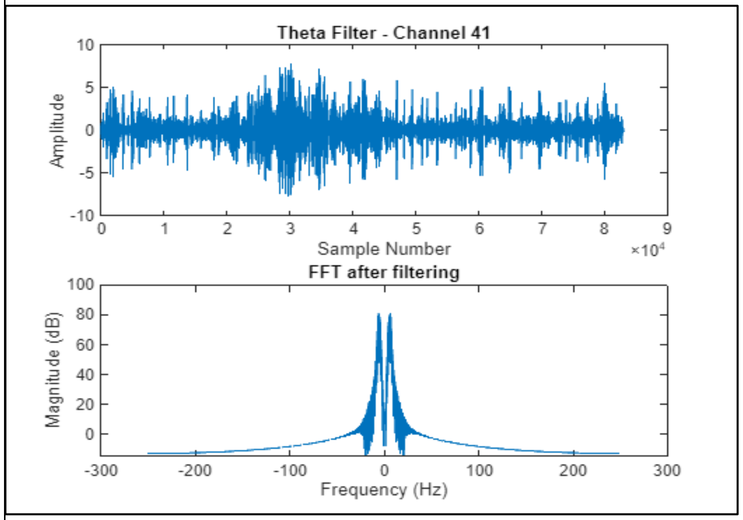
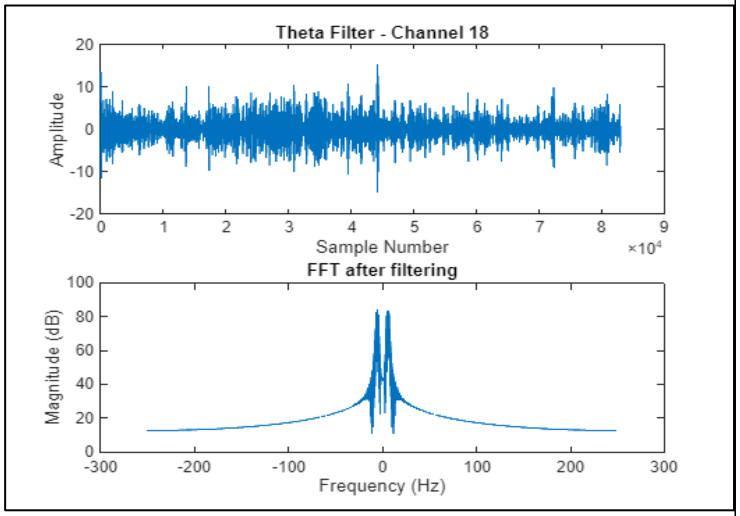
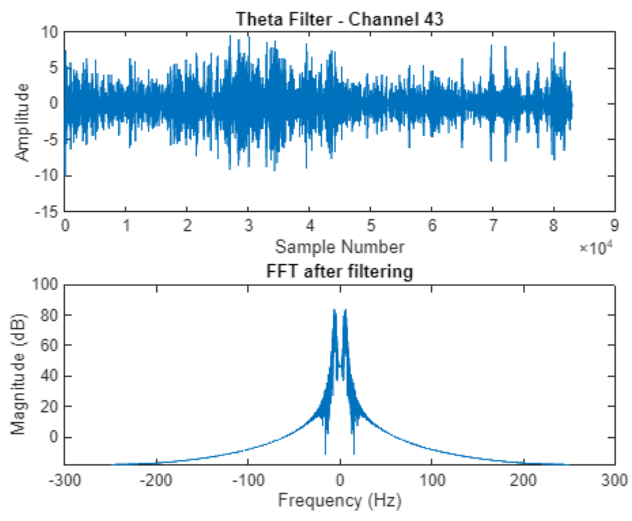
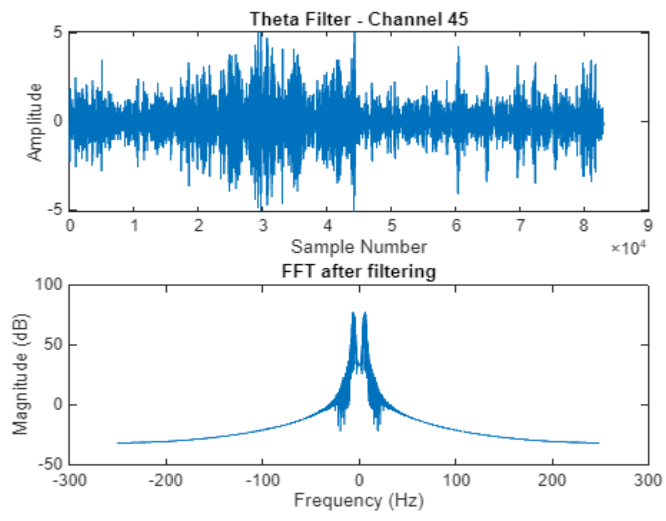


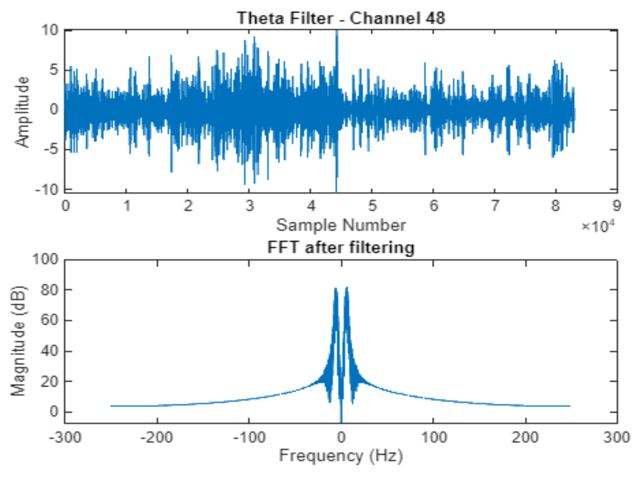
Figure 3: The time domain and frequency domain of the signals of 10 random channels after passing through delta filter

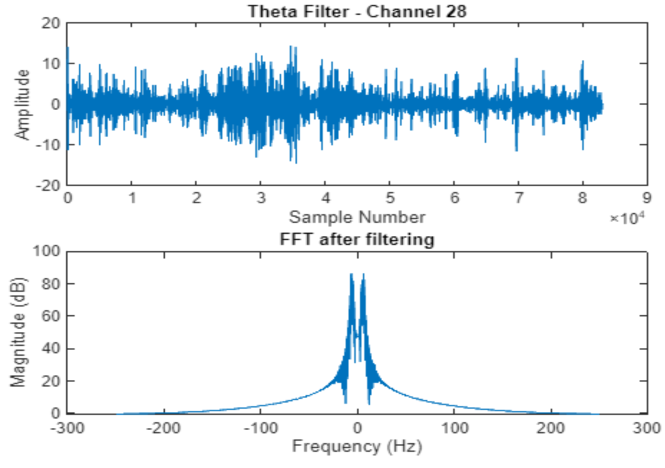


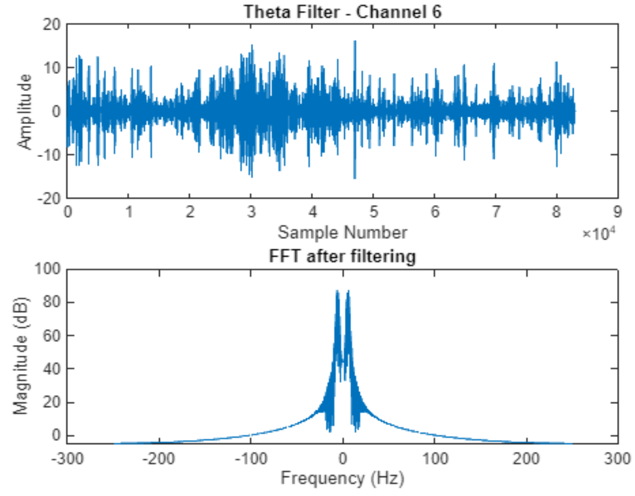


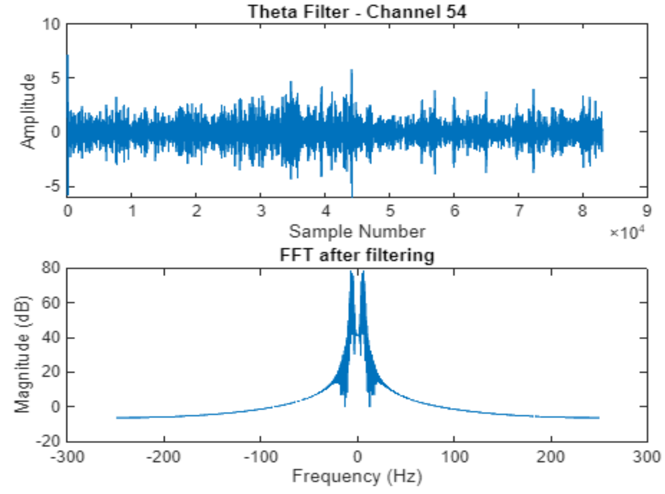

Theta Filter - Channel 43
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)

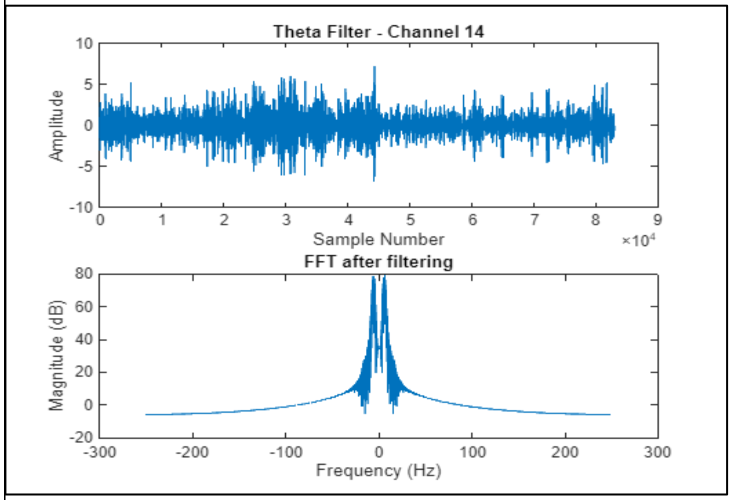

Theta Filter - Channel 45
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Theta Filter - Channel 48
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Theta Filter - Channel 28
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Theta Filter - Channel 6
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Theta Filter - Channel 54
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)



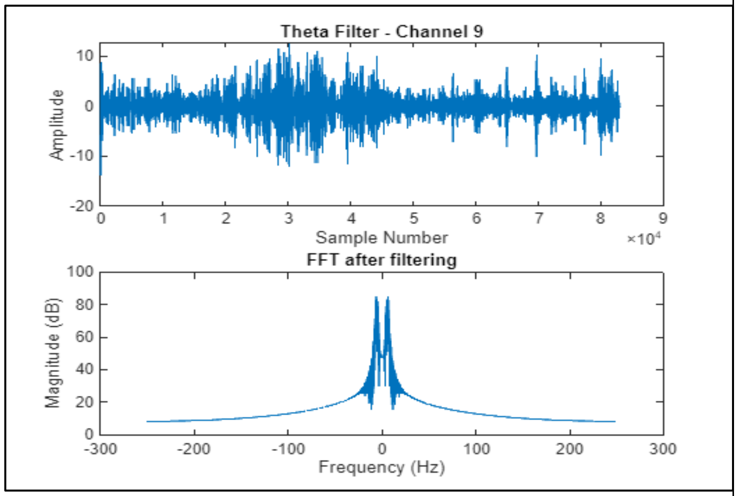
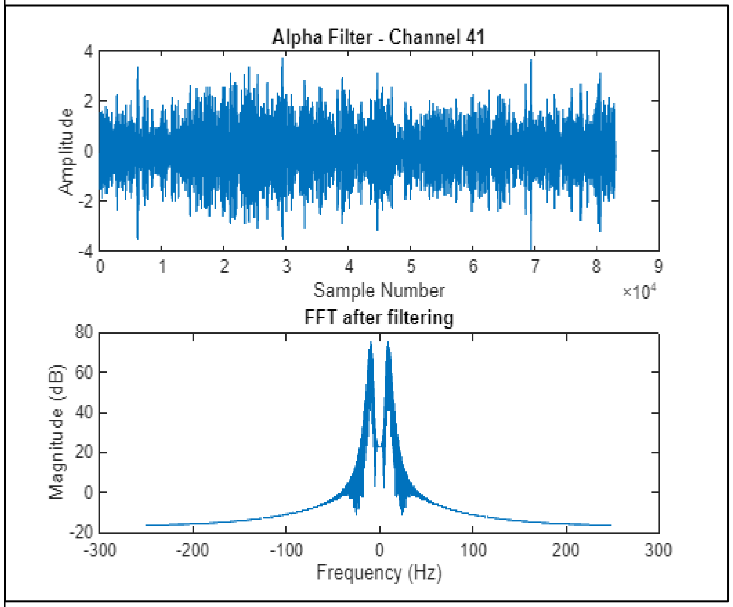
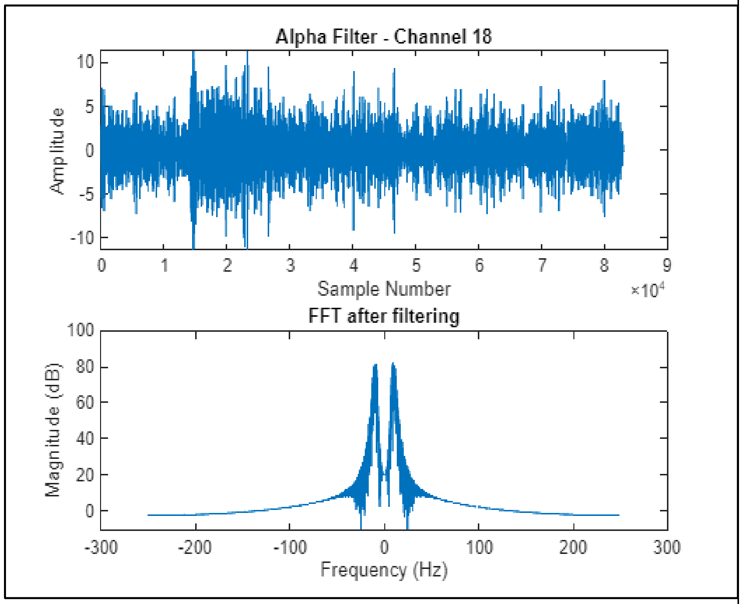
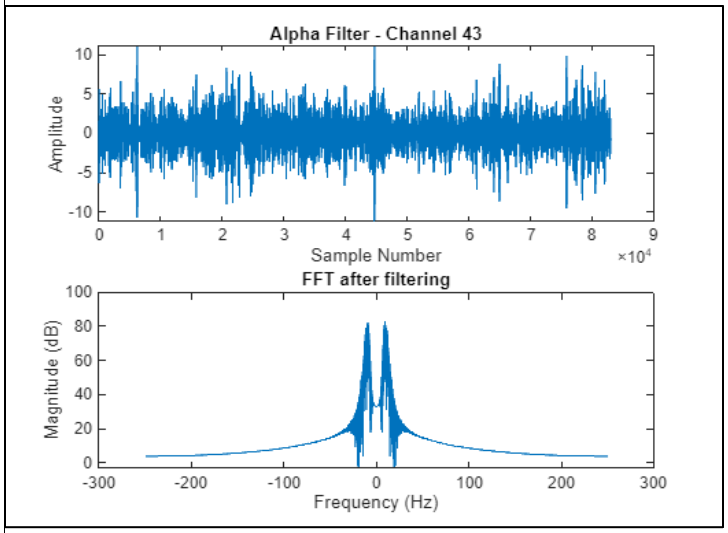
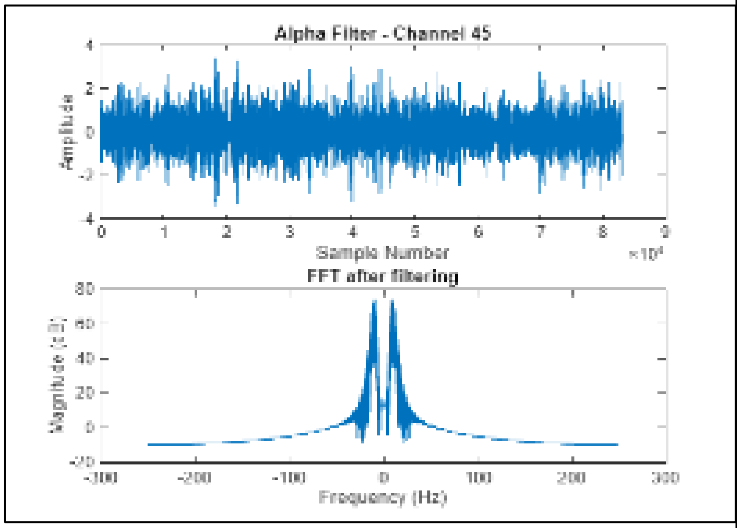


Figure 4: The Original and Filtered Signal in time and frequency domain









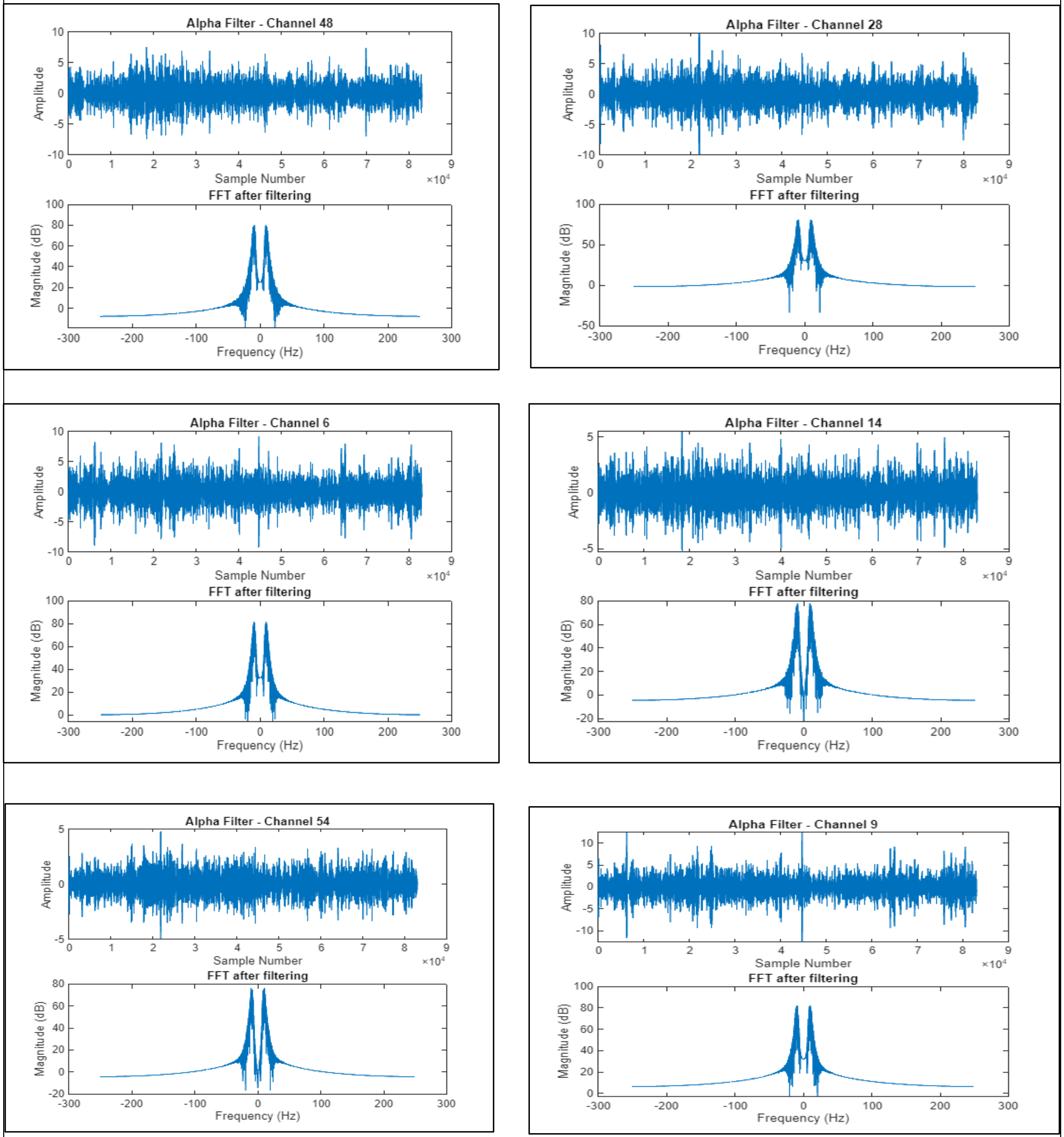
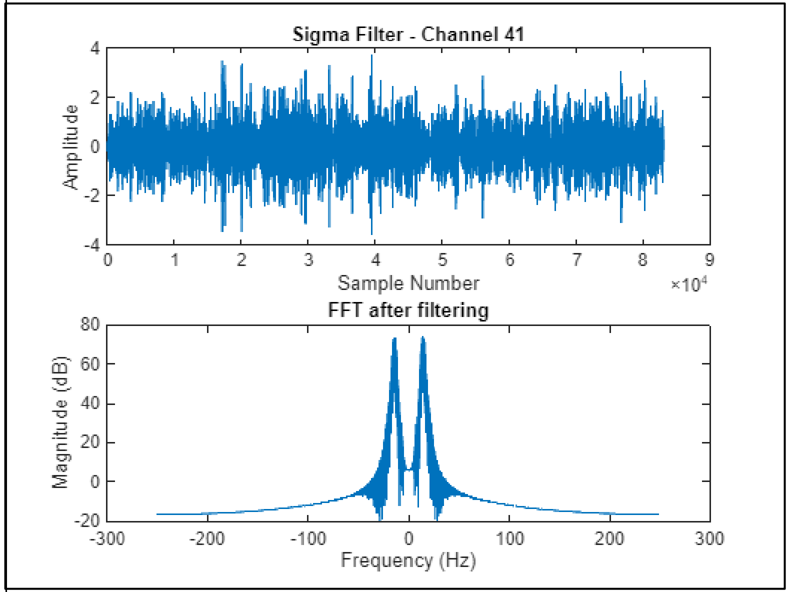
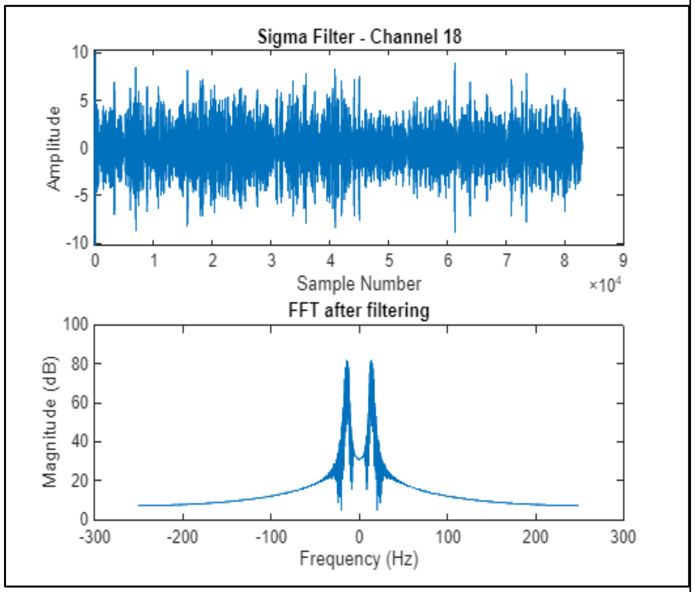
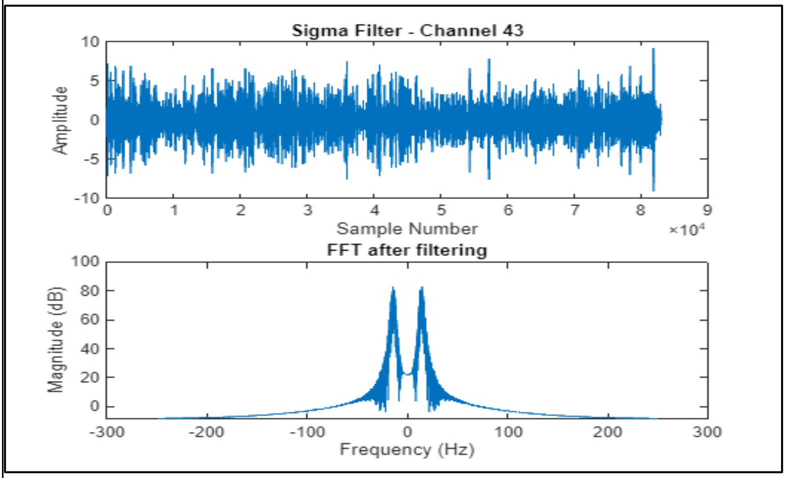
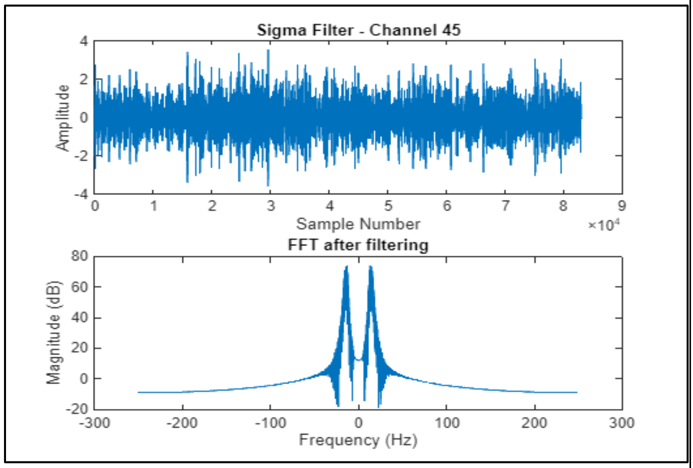


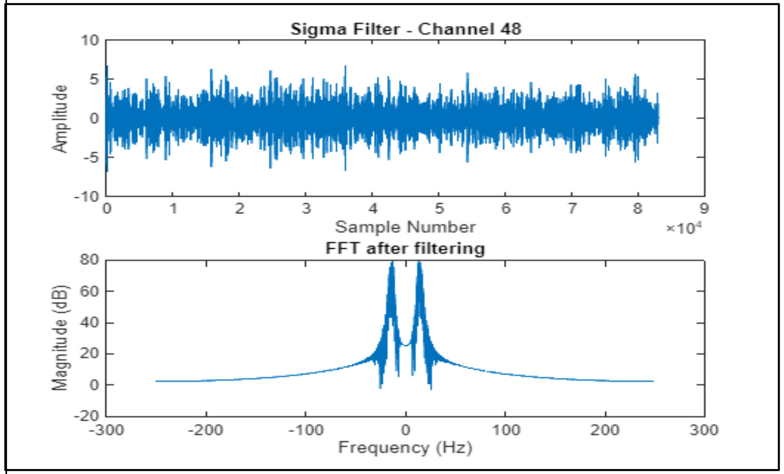
Figure 5: The original signal and signal after alpha filter in time and frequency domain

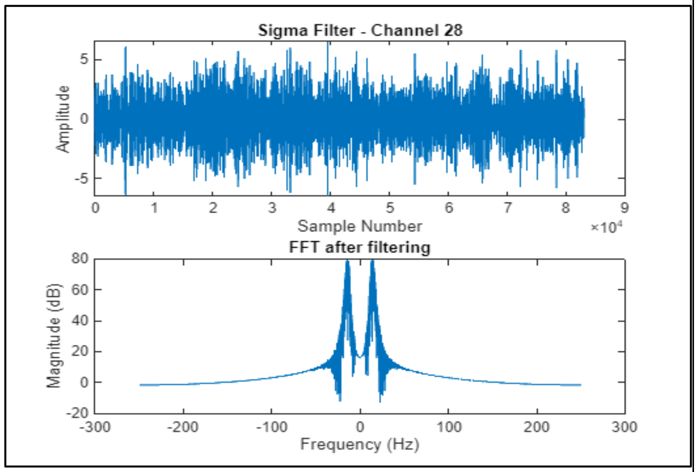

Sigma Filter - Channel 41
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

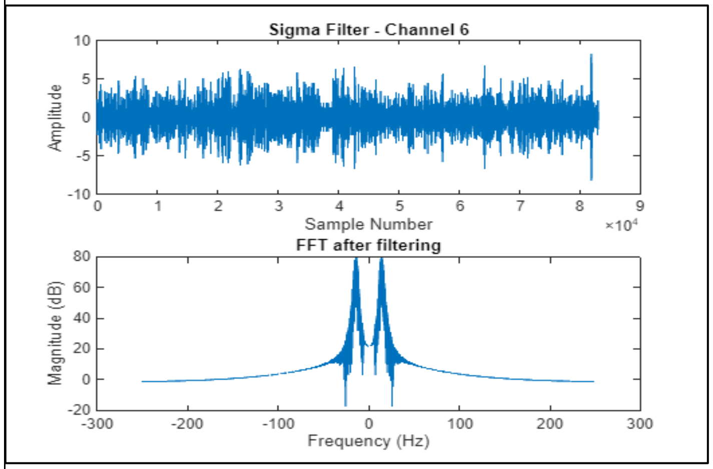

Sigma Filter - Channel 18
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

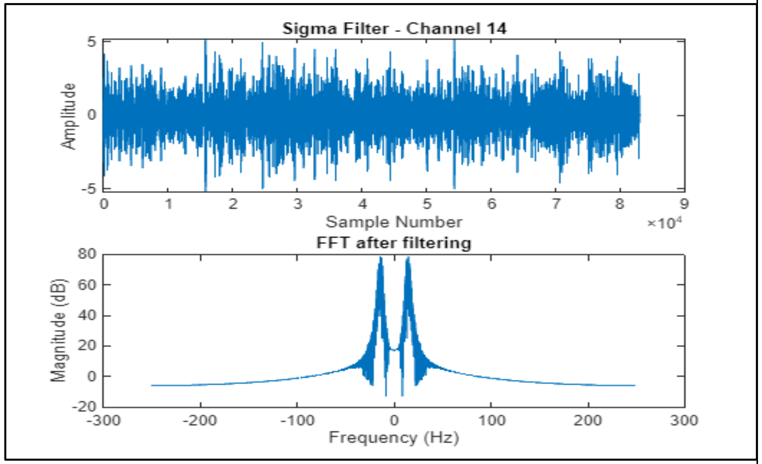

Sigma Filter - Channel 43
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)

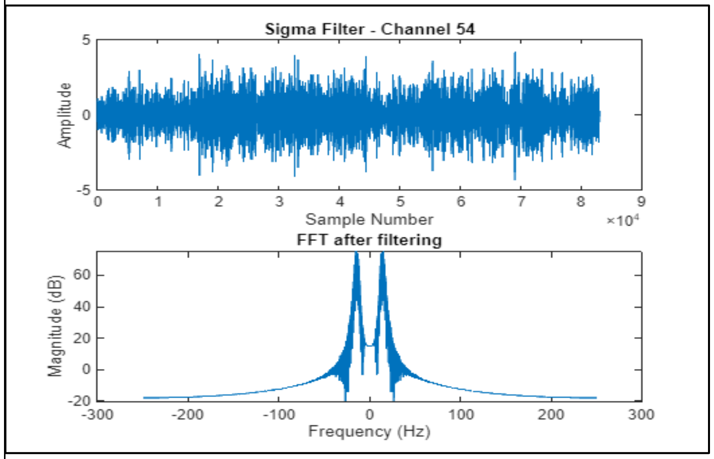

Sigma Filter - Channel 45
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Sigma Filter - Channel 48
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Sigma Filter - Channel 28
Amplitude
Sample Number
×10^4
FFT after filtering
Magnitude (dB)
Frequency (Hz)







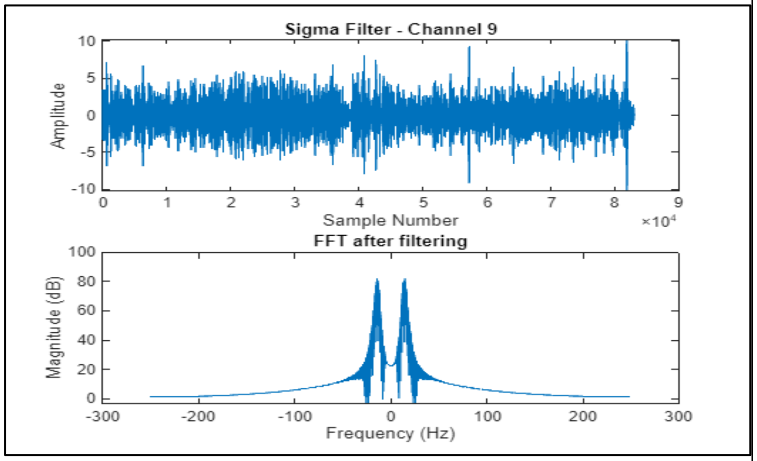
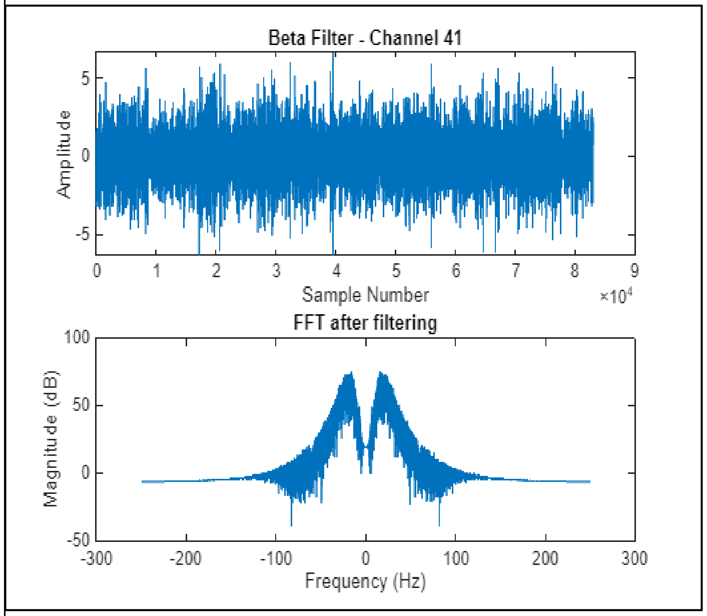
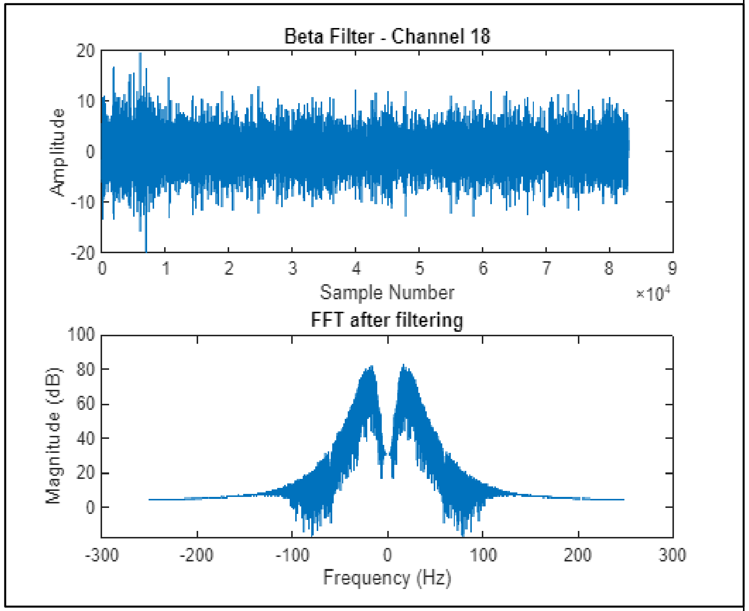
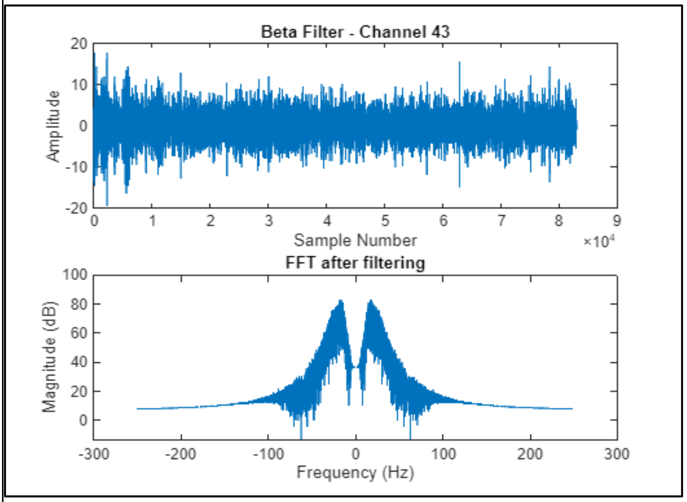
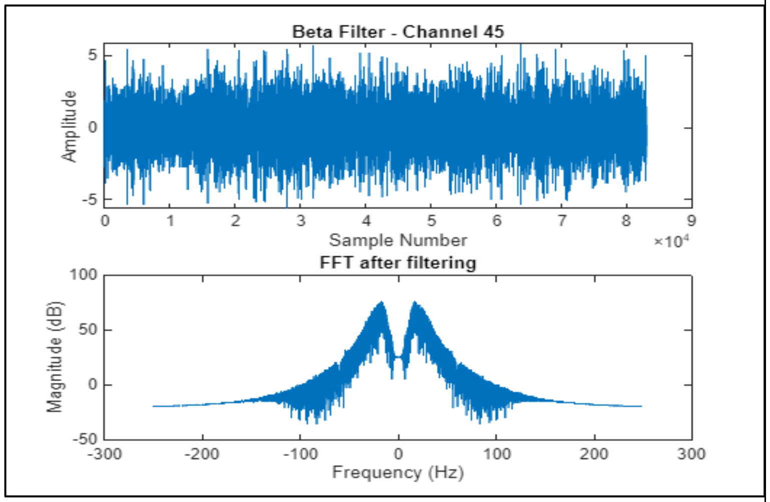


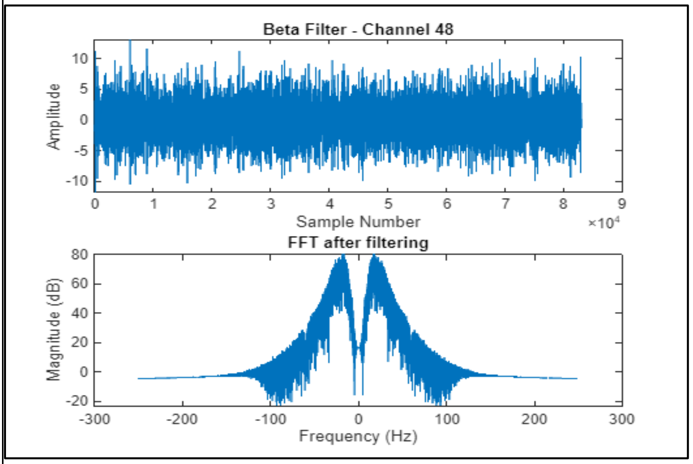
Figure 6: Signal before and after sigma filter in both time and frequency domain

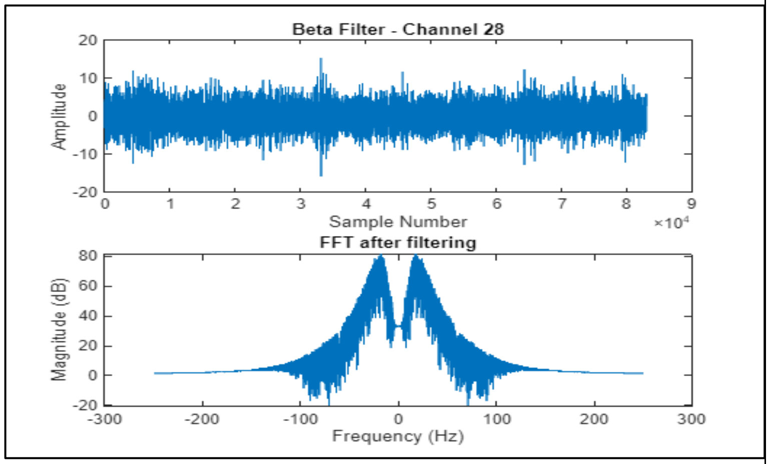


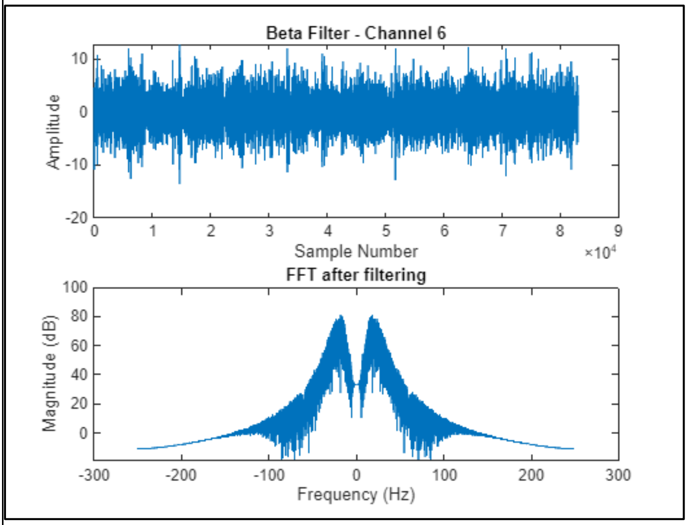


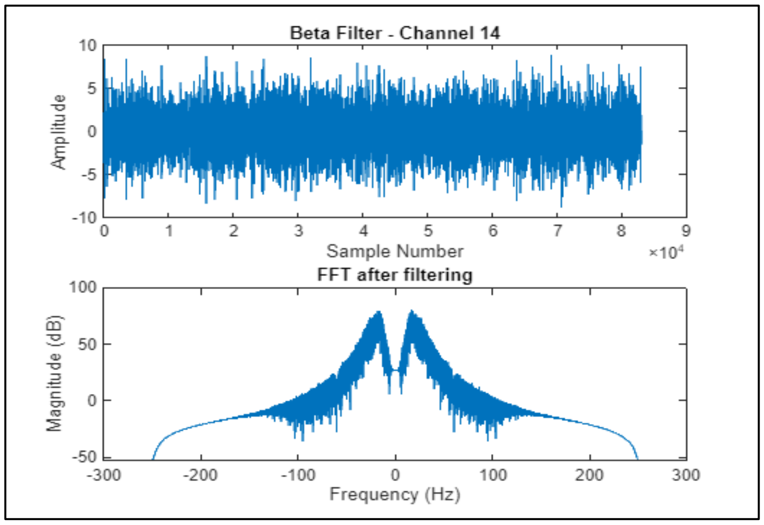

Beta Filter - Channel 43
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)

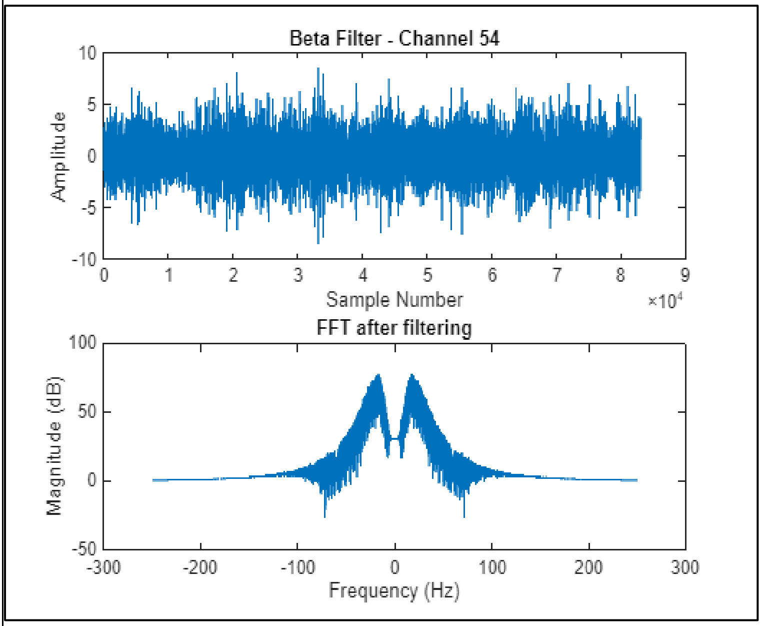

Beta Filter - Channel 45
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Beta Filter - Channel 48
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Beta Filter - Channel 28
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Beta Filter - Channel 6
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)


Beta Filter - Channel 14
Amplitude
Sample Number
×10⁴
FFT after filtering
Magnitude (dB)
Frequency (Hz)



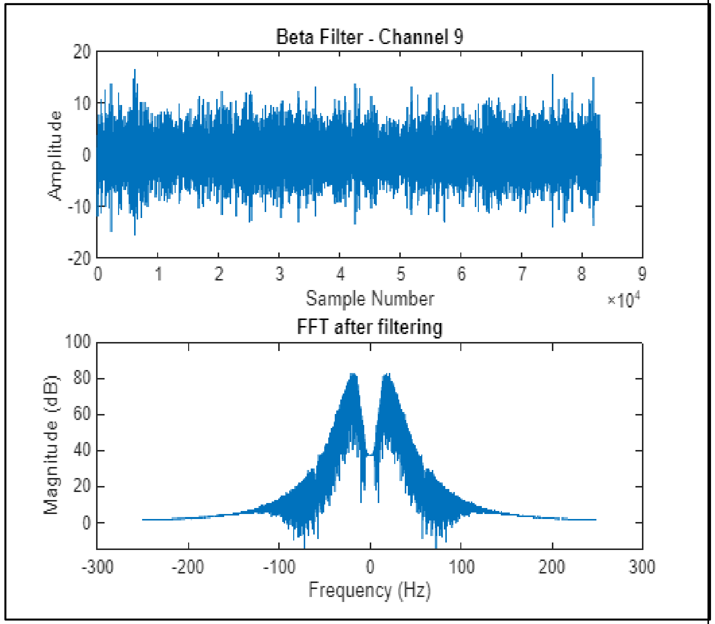
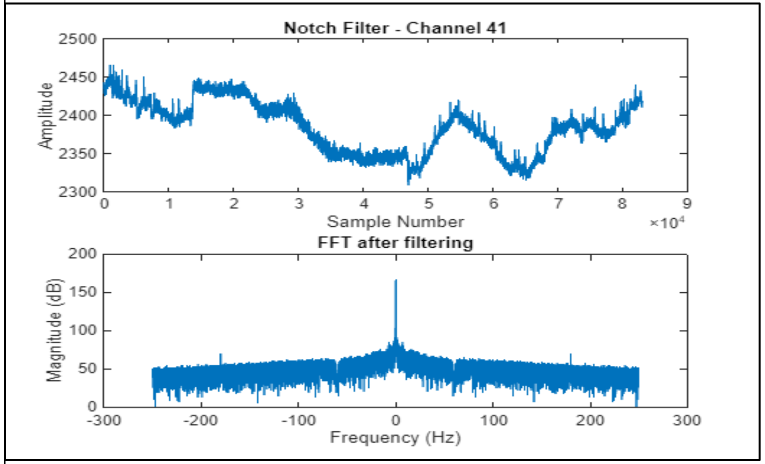
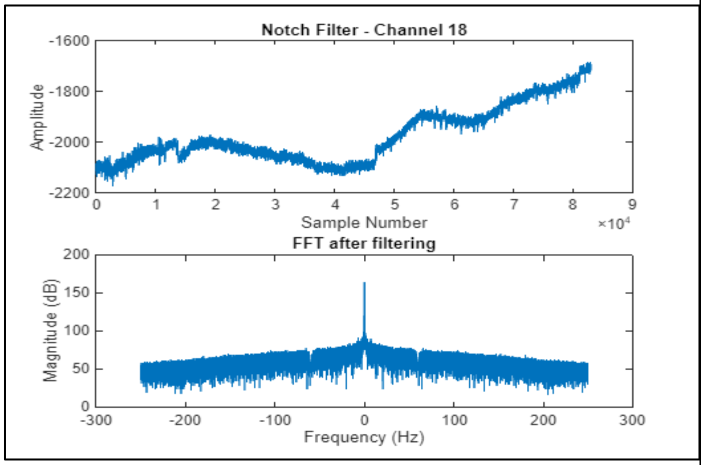
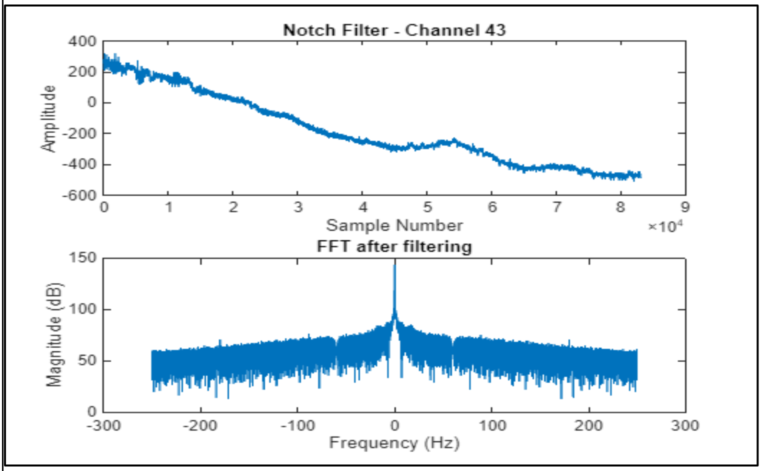
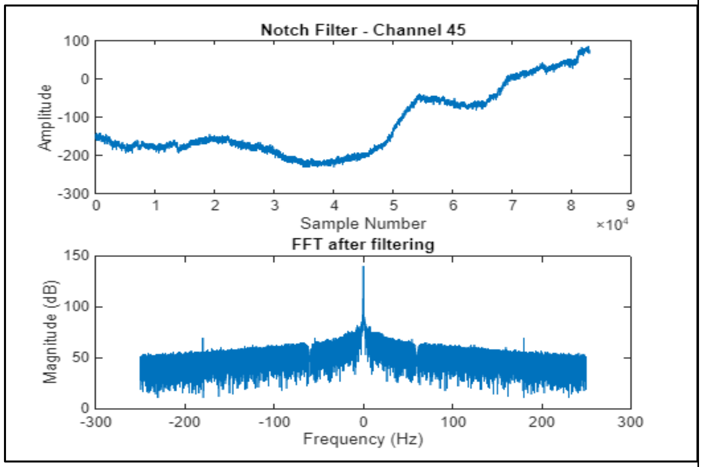


Figure 7: Signal before and after the application of the beta filter in both time and frequency domain









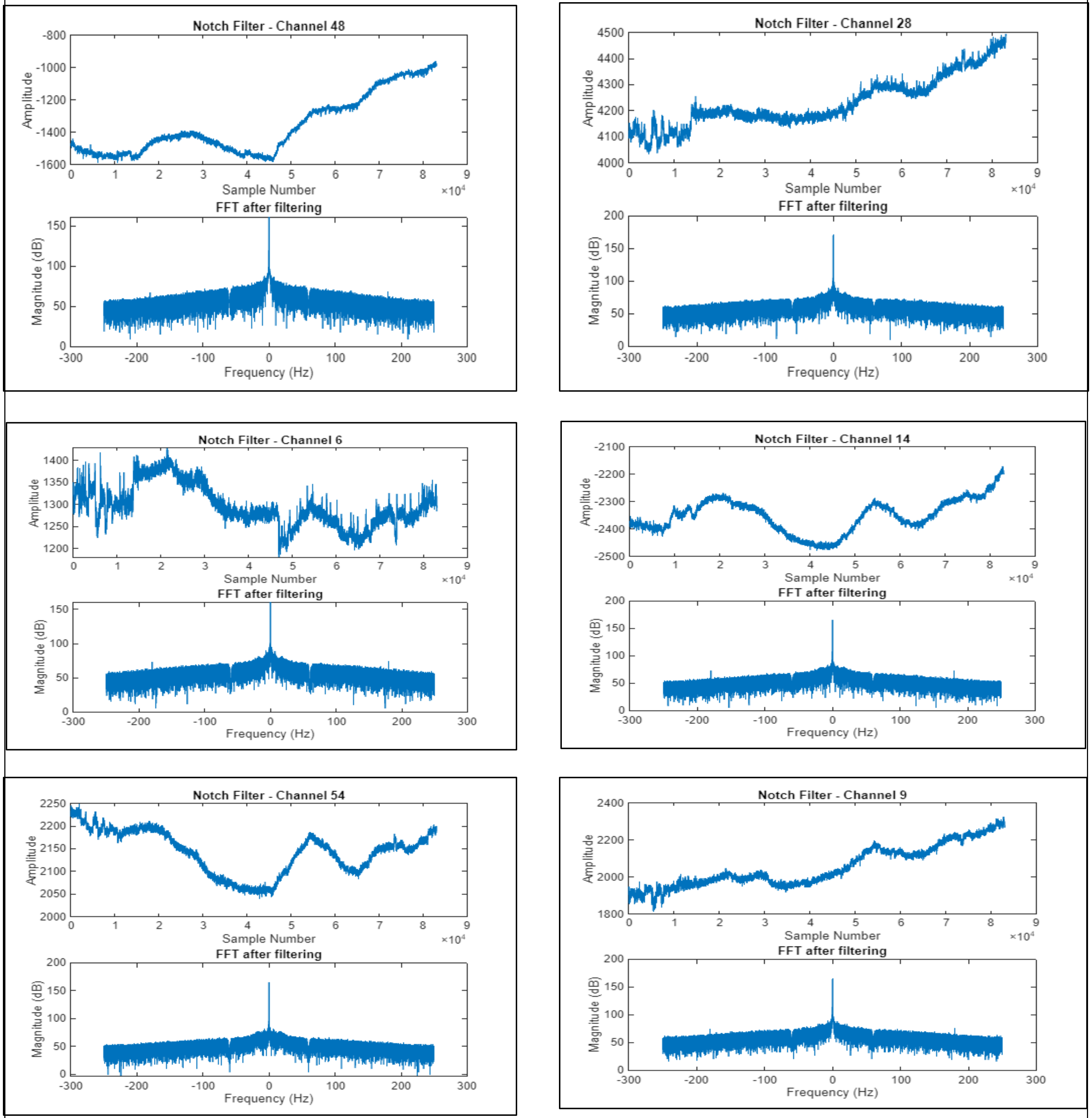


Figure 8: Signal before and after notch filter, in both time and frequency domain

```
% Plot the combined frequency response of each of the designed filters in dB
scale
figure;
hold on;
for k = 1:length(all_filters)
    filter_name = all_filters{k};
    if strcmp(filter_name, 'notch')
        [h, f] = freqz(notch_b, notch_a, 1024, fs);
    else
        [h, f] = freqz(filters.(filter_name), 1024, fs);
    end
    plot(f, 20*log10(abs(h)), 'DisplayName', filter_name);
    saveas(gcf, [all_filters{k} '_Filtered_Signals.png']);
end
legend;
title('Frequency Response of Designed Filters (dB Scale)');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
grid on;
hold off;

saveas(gcf, 'Combined_Frequency_Response_dB.png');
```

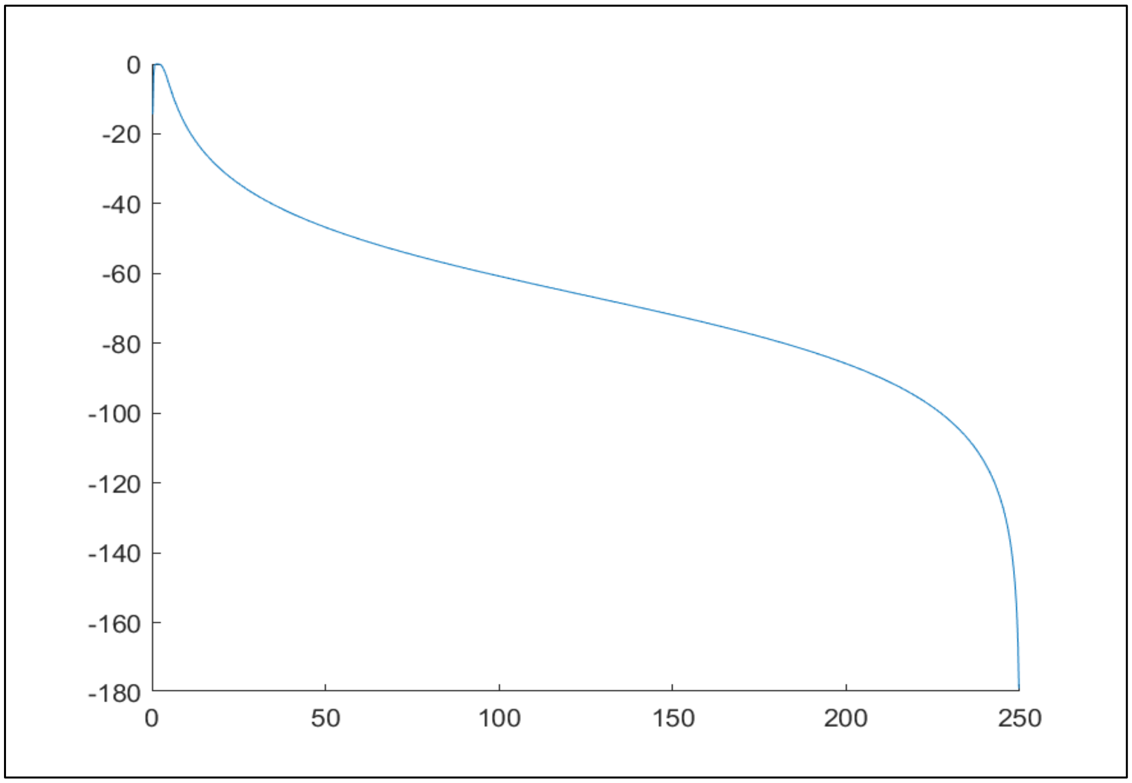


Figure 9: Frequency Response of the Delta Filter

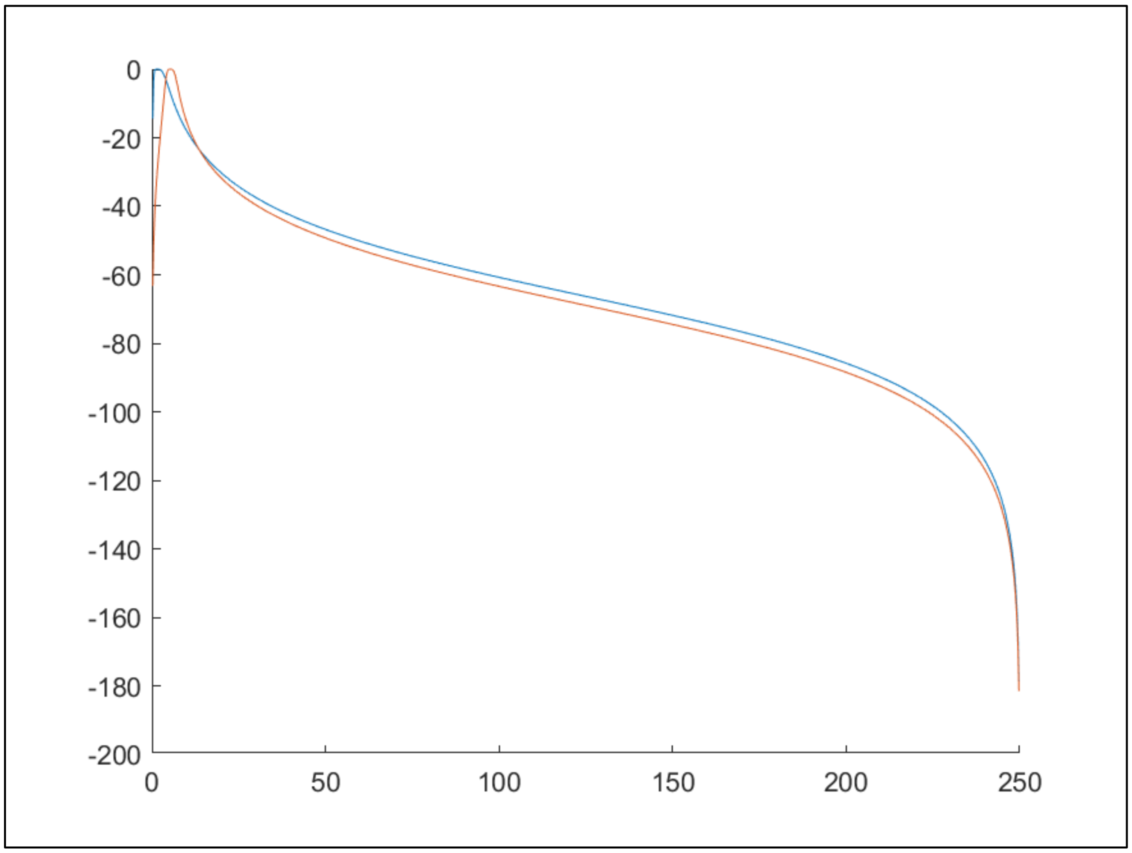


Figure 10: Frequency Response of Theta Filter (red) added

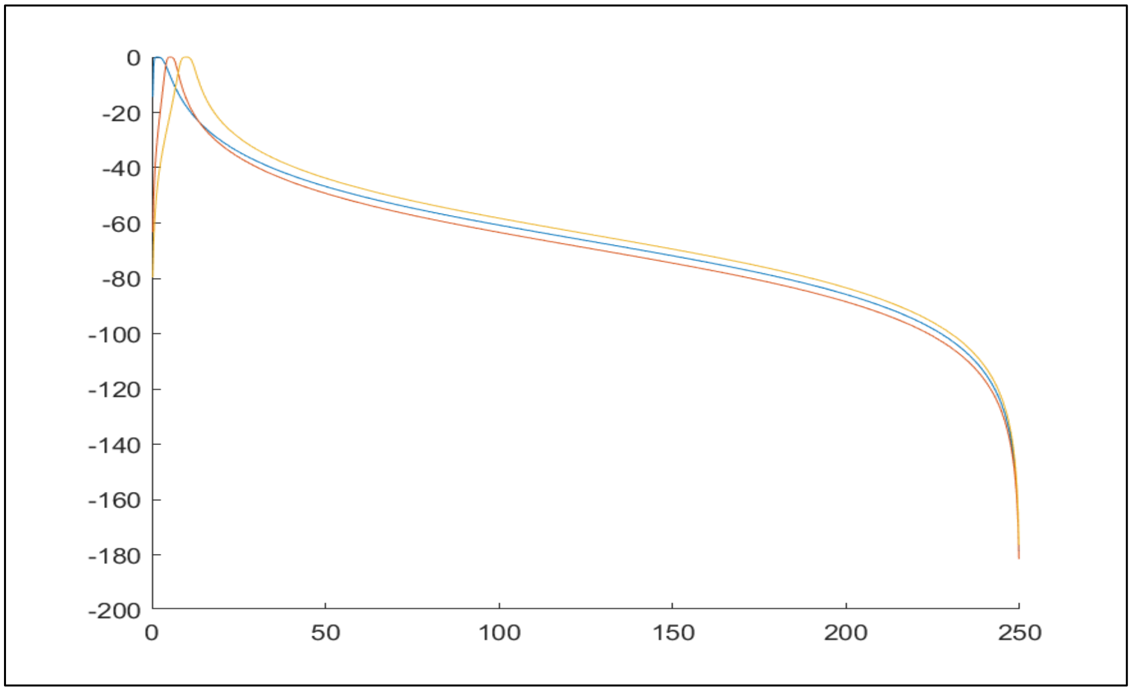


Figure 11: Frequency Response of Alpha Filter (yellow) added

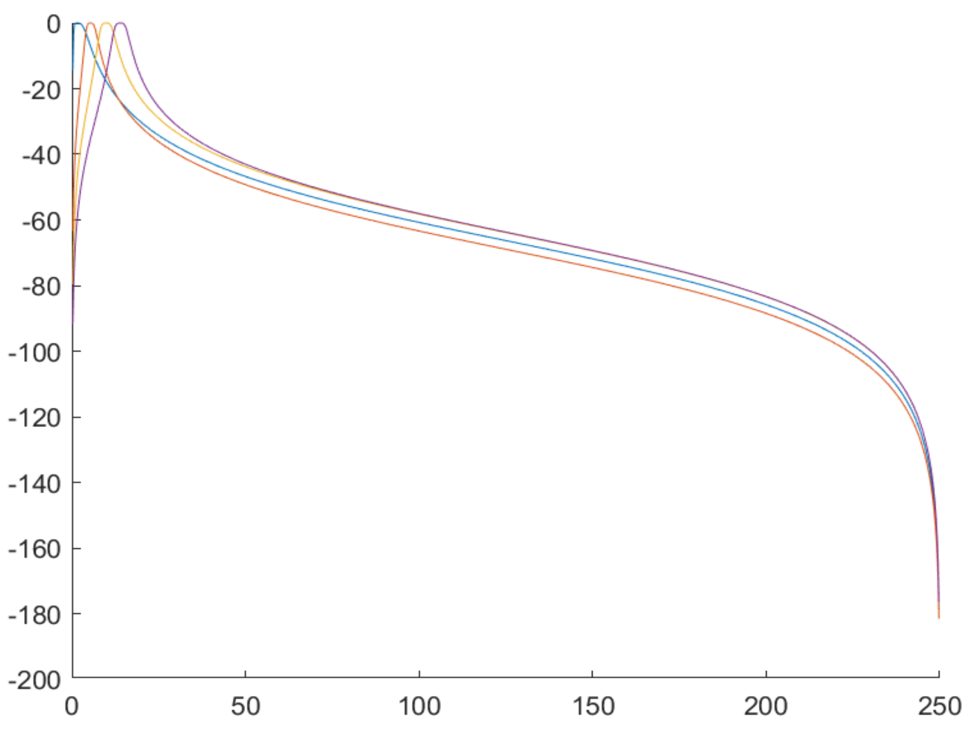


Figure 12: Frequency Response of Sigma Filter (purple) added

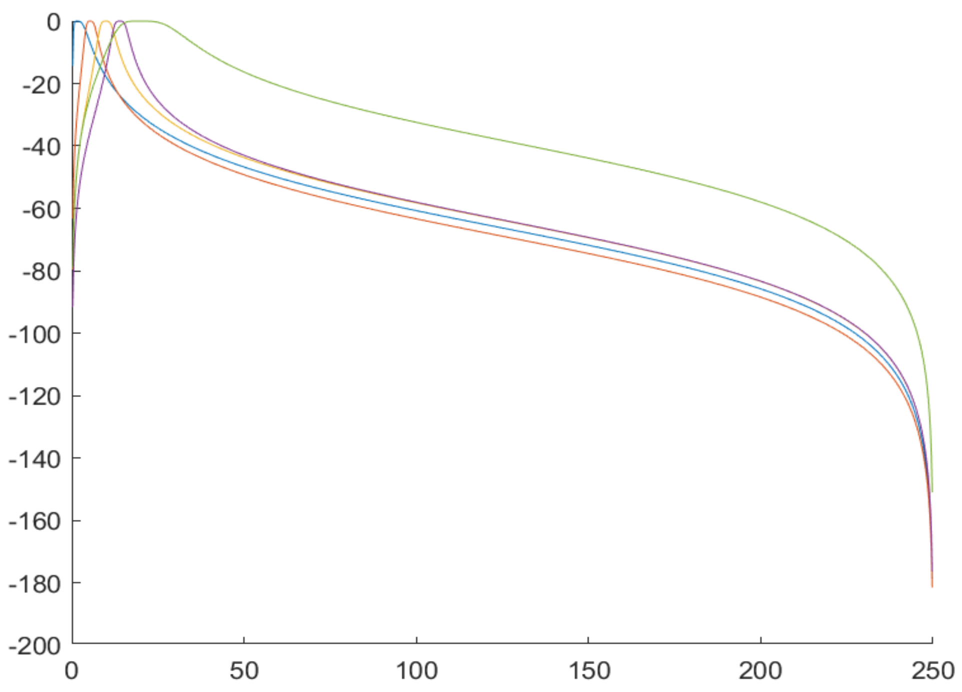


Figure 13: Frequency Response of Beta Filter (green) added

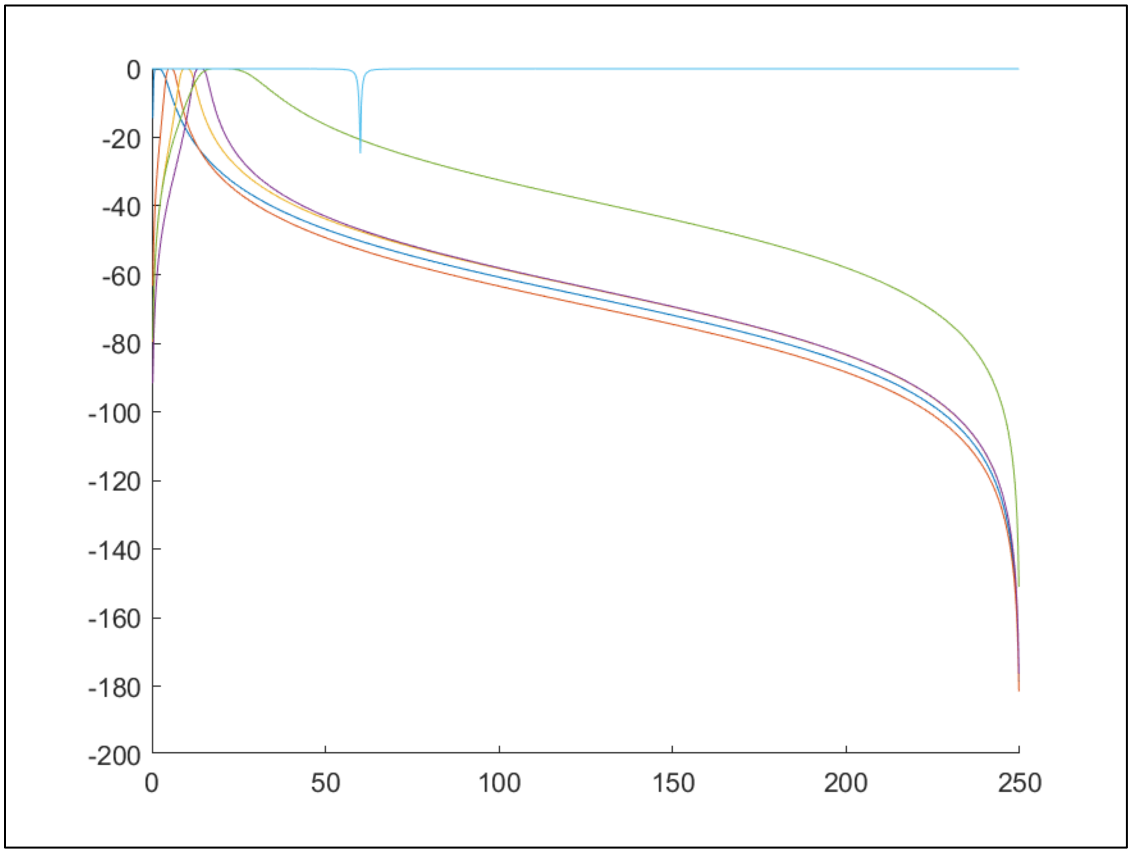


Figure 14: Frequency Response of Notch Filter (sky blue) added

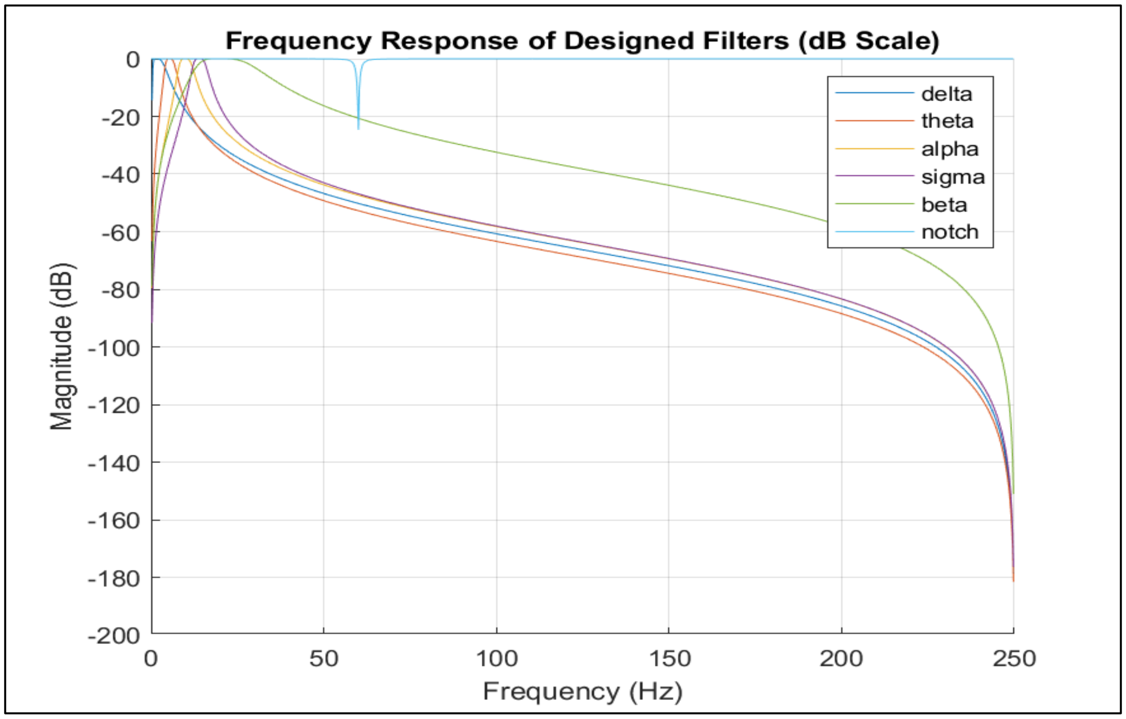
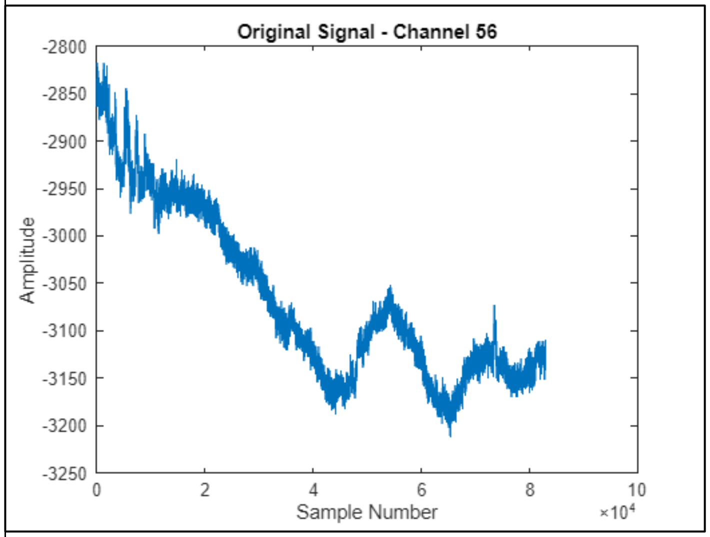
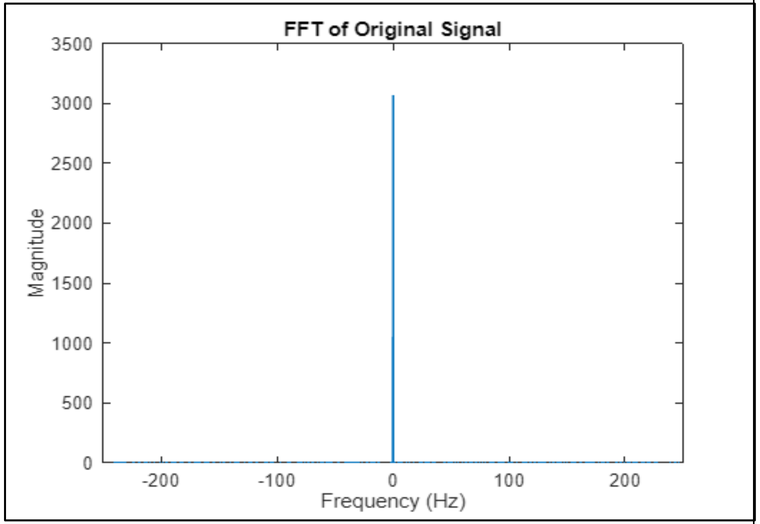
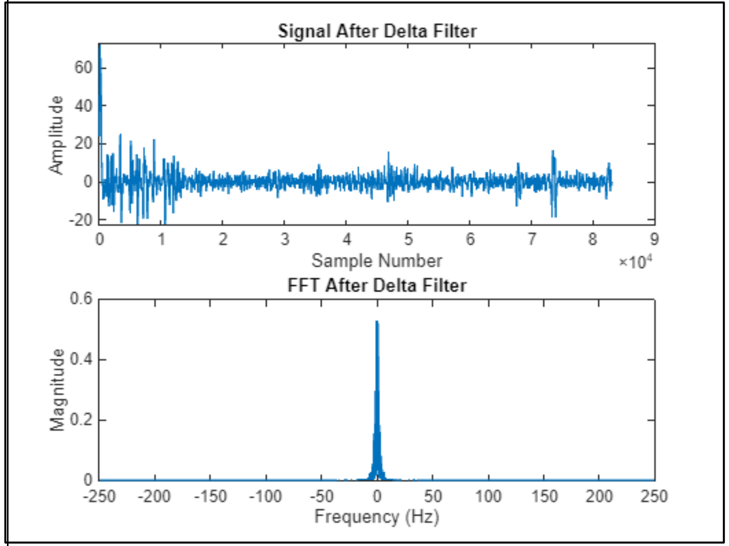


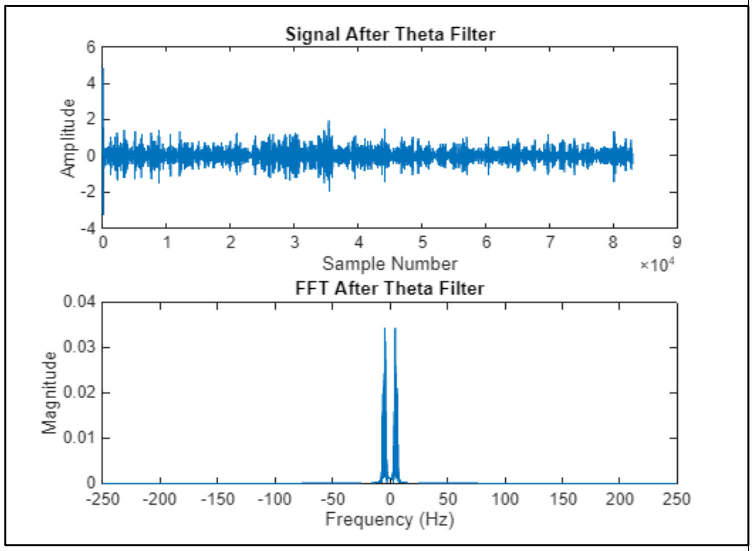
Figure 15: Combined Frequency Response of all the filters.

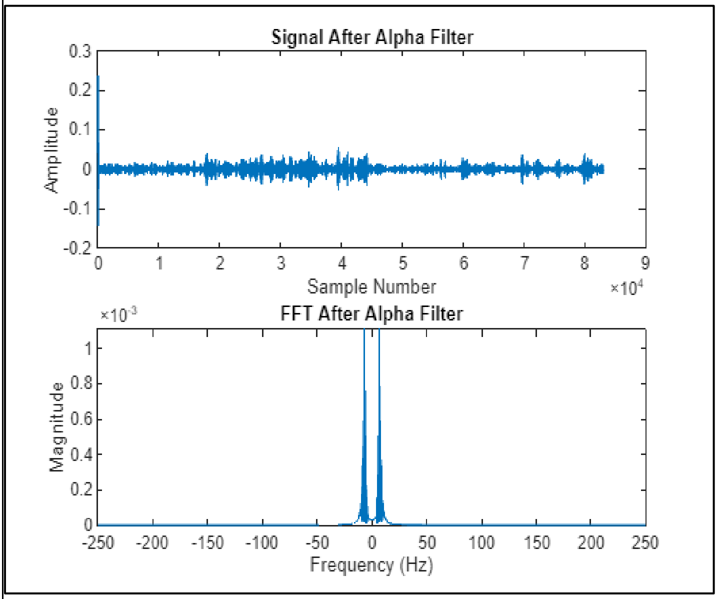
```
% Sequential filtering on one channel
sequential_filtered_signal = EEG.data(channel_to_plot, :);

figure;
plot(sequential_filtered_signal);
title('Original Signal');
xlabel('Sample Number');
ylabel('Amplitude');
saveas(gcf, 'Original_Signal.png');


for k = 1:length(all_filters)
    filter_name = all_filters{k};
    if strcmp(filter_name, 'notch')
        sequential_filtered_signal = filtfilt(notch_b, notch_a, 
sequential_filtered_signal);
    else
        sequential_filtered_signal = filtfilt(filters.(filter_name), 
sequential_filtered_signal);
    end

    % Plot the signal after passing through each filter
    figure;
    plot(sequential_filtered_signal);
    title(['Signal After ' upper(filter_name(1)) filter_name(2:end) ' Filter']);
    xlabel('Sample Number');
    ylabel('Amplitude');
    saveas(gcf, ['Sequential_Filtered_Signal_' filter_name '.png']);
end
```

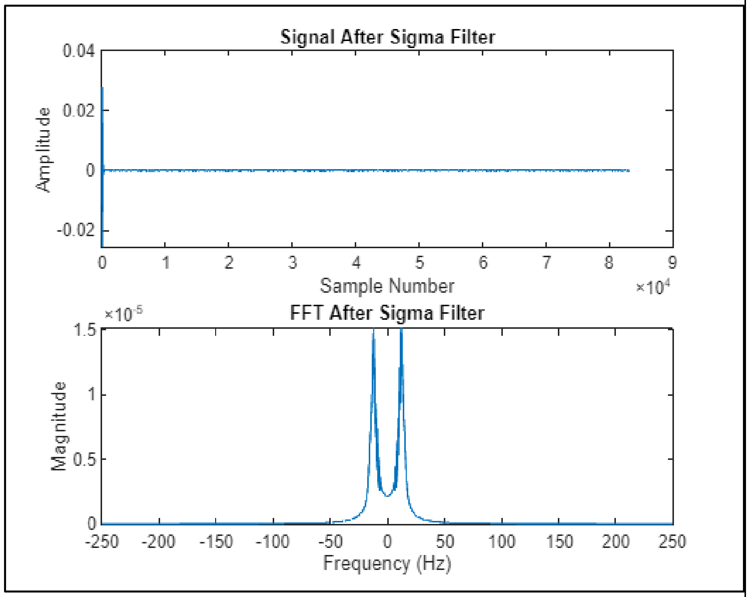


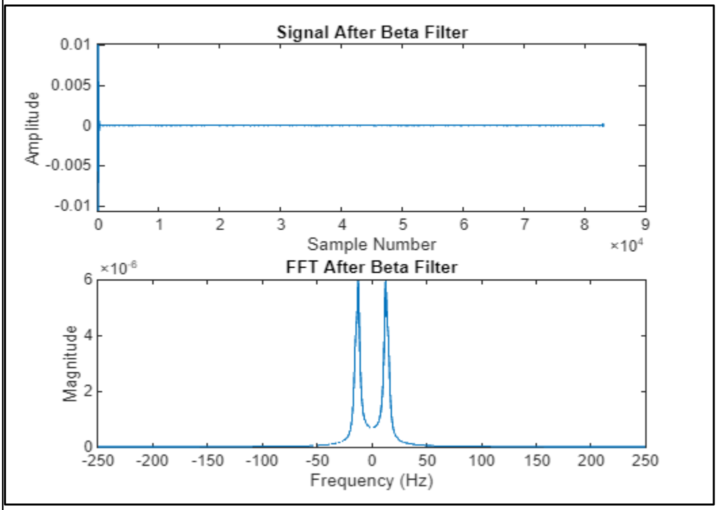


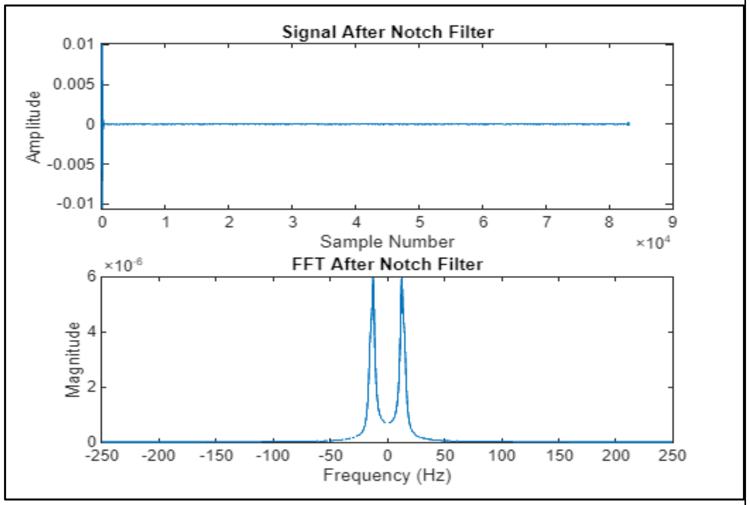

Signal After Delta Filter
Amplitude
Sample Number
×10^4
FFT After Delta Filter
Magnitude
Frequency (Hz)


Signal After Theta Filter
Amplitude
Sample Number
×10^4
FFT After Theta Filter
Magnitude
Frequency (Hz)


Signal After Alpha Filter
Amplitude
Sample Number
×10^4
FFT After Alpha Filter
×10^-3
Magnitude
Frequency (Hz)


Signal After Sigma Filter
Amplitude
Sample Number
×10^4
FFT After Sigma Filter
×10^-5
Magnitude
Frequency (Hz)


Signal After Beta Filter
Amplitude
Sample Number
×10^4
FFT After Beta Filter
×10^-6
Magnitude
Frequency (Hz)


Signal After Notch Filter
Amplitude
Sample Number
×10^4
FFT After Notch Filter
×10^-6
Magnitude
Frequency (Hz)

Comments on the Results

The results demonstrate the effectiveness of the notch filter in removing 60Hz line noise from the EEG signal, enhancing signal clarity significantly. Subsequent application of bandpass, high-pass, and low-pass filters on selected channels isolates specific frequency components, effectively highlighting various brain activity bands. The sequential filtering process on a single channel confirms cumulative noise reduction and signal enhancement, rendering the EEG data more suitable for detailed analysis. Overall, the filtering techniques employed improve EEG signal quality, facilitating more accurate and insightful neurological assessments. These outcomes underscore the significance of advanced digital signal processing techniques in enhancing the quality of EEG data and contributing to more accurate neurological analyses and diagnoses.

Conclusion

In conclusion, the comprehensive filtering approach applied to EEG data successfully mitigates noise interference and enhances signal integrity. By effectively removing line noise and isolating specific frequency bands associated with brain activity, the filtered signals exhibit improved clarity and fidelity. The sequential filtering process further refines the data, resulting in a cleaner and more interpretable EEG signal. These outcomes underscore the significance of advanced digital signal processing techniques in improving the quality of EEG data, contributing to more accurate and insightful neurological analyses and diagnoses. As EEG continues to be a critical tool in neurology and neuroscience research, the utilization of robust filtering methods demonstrated in this project holds immense promise in advancing our understanding of brain function and pathology.