EE-330 Digital Signal Processing

For

B.E. Electrical Engineering

PROJECT REPORT

Group Members			
Umair Ejaz – 375418			
Ahmad Daud - 367195			
Ummaima Nadeem - 393460			
Degree	EE-43	Syndicate	C

Submission date: ***06/06/24*** ***Checked By:***

Contents

EEG Signal Processing and Analysis Using MATLAB

Abstract:

This MATLAB script demonstrates a comprehensive signal processing pipeline for preprocessing EEG (Electroencephalogram) data. The pipeline includes notch filtering to remove powerline interference, low-pass and high-pass filtering for noise reduction and signal enhancement, and band-pass filtering to isolate specific frequency bands relevant to brain activity (delta, theta, alpha, beta, sigma, and gamma). Additionally, band-stop filters are applied to eliminate powerline noise at 60 Hz and its second harmonic at 120 Hz. The processed signals are visualized in both the time and frequency domains for each processing step. The script provides a systematic approach to prepare EEG data for further analysis, such as feature extraction or classification of brain states, by enhancing signal quality and isolating frequency components of interest.

Introduction:

Electroencephalography (EEG) is a technique used to record the brain's electrical activity in a non-invasive manner. By placing electrodes on the scalp, we can capture the electrical impulses that reflect neural activity, offering valuable insights into brain function. These EEG signals are essential for diagnosing neurological conditions, monitoring brain activity, and advancing cognitive research. However, the raw EEG data often contain various types of noise, such as power line interference, muscle activity, and other artifacts, which can mask meaningful brain signals.

This project focuses on cleaning and analyzing EEG data using MATLAB, specifically targeting signal filtering and noise reduction. Through digital signal processing (DSP) methods, we aim to improve the quality of EEG signals, leading to more precise interpretation and analysis.

Methodology:

The methodology for this project involves several key steps, each employing specific DSP techniques to clean and analyze EEG signals effectively:

1. Installation of MATLAB: Ensure that MATLAB is installed and ready for use, as it offers a comprehensive environment for performing signal processing tasks.

2. Data Preparation and Preliminary Analysis:

- Load the EEG data into MATLAB.
- Select the channels of interest.
- Plot the original signals in both the time and frequency domains to understand the noise characteristics.

3. Filter Design and Implementation:

- Design and apply various filters to the EEG data to remove noise and extract meaningful signals. The filters used include:

  Notch Filter: To remove power line interference at 55 Hz.

  Low-Pass Filter: To eliminate high-frequency noise and retain frequencies below 30 Hz.

  High-Pass Filter: To remove low-frequency drift and retain frequencies above 0.5 Hz.

  -Band-Pass Filters: To isolate specific frequency bands corresponding to different brain activities (delta, theta, alpha, sigma, beta).

4. Selection and Extraction of Channels:

- Focus on the first 10 channels from the dataset for detailed analysis and processing.

Code

```matlab
% Import the data
load('825_2_PD_REST.mat');  % Adjust the path if necessary
brain_data = EEG.data;  % Assuming EEG is the main structure and data is the field
with EEG signals

% Select the 26th channel
selected_channel = brain_data(1, :);
```

```
% Sampling rate (Hz)
Fs = 420;

% Time vector for visualization
time_vector = (0:length(selected_channel)-1) / Fs;

% Plot the original signal in the time domain
figure;
subplot(5,5,1);
plot(time_vector, selected_channel / max(abs(selected_channel))); % Normalize the
original signal
title('Original Brain Signal - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the original signal in the frequency domain
n_samples = length(selected_channel);
original_fft = fft(selected_channel);
frequency_vector = Fs * (0:n_samples-1) / n_samples;  % Frequency range (Hz)
magnitude_spectrum = abs(original_fft) / n_samples;  % Magnitude of the FFT

subplot(5,5,2);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_spectrum(1:floor(n_samples/2)) /
max(magnitude_spectrum(1:floor(n_samples/2))));  % Normalize the magnitude
title('Original Brain Signal - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);  % Limit the plot to frequencies up to Nyquist frequency
grid on;

% Design a notch filter to remove 55 Hz noise
notch_frequency = 55;  % Frequency to be removed (Hz)
quality_factor = 100;  % Quality factor
normalized_frequency = notch_frequency / (Fs / 2);  % Normalized frequency
bandwidth = normalized_frequency / quality_factor;
[b_notch, a_notch] = iirnotch(normalized_frequency, bandwidth);

b_notch = double(b_notch);
a_notch = double(a_notch);
selected_channel = double(selected_channel);

% Apply the notch filter to the signal
filtered_selected_channel = filtfilt(b_notch, a_notch, selected_channel);

% Normalize the notch filtered signal
filtered_selected_channel = filtered_selected_channel /
max(abs(filtered_selected_channel));

% Plot the filtered signal in the time domain
subplot(5,5,3);
plot(time_vector, filtered_selected_channel);
title('Filtered Brain Signal - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the filtered signal in the frequency domain
filtered_fft = fft(filtered_selected_channel);
```

```matlab
magnitude_filtered = abs(filtered_fft) / n_samples;

subplot(5,5,4);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered(1:floor(n_samples/2)) /
max(magnitude_filtered(1:floor(n_samples/2)))); % Normalize the magnitude
title('Filtered Brain Signal - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a low-pass filter for frequencies below 9 Hz
lowpass_cutoff_frequency = 5; % Cutoff frequency for low-pass filter (Hz)
[b_lowpass, a_lowpass] = butter(4, lowpass_cutoff_frequency / (Fs/2), 'low'); %
Design low-pass filter

% Apply the low-pass filter to the notch filtered signal
filtered_selected_channel_lowpass = filtfilt(b_lowpass, a_lowpass,
filtered_selected_channel);

% Normalize the low-pass filtered signal
filtered_selected_channel_lowpass = filtered_selected_channel_lowpass /
max(abs(filtered_selected_channel_lowpass));

% Plot the low-pass filtered signal in the time and frequency domain
subplot(5,5,5);
plot(time_vector, filtered_selected_channel_lowpass);
title('Low-pass Filtered Brain Signal (Cutoff 9 Hz) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(5,5,6);
filtered_fft_lowpass = fft(filtered_selected_channel_lowpass);
magnitude_filtered_lowpass = abs(filtered_fft_lowpass) /
max(abs(filtered_fft_lowpass)); % Normalize the magnitude
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_lowpass(1:floor(n_samples/2)));
title('Low-pass Filtered Brain Signal (Cutoff 9 Hz) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
ylim([0 1.2]); % Adjusted y-axis limit for better visualization
grid on;

% Design a high-pass filter for frequencies above 20 Hz
highpass_cutoff_frequency = 20; % Cutoff frequency for high-pass filter (Hz)
[b_highpass, a_highpass] = butter(4, highpass_cutoff_frequency / (Fs/2), 'high');
% Design high-pass filter

% Apply the high-pass filter to the notch filtered signal
filtered_selected_channel_highpass = filtfilt(b_highpass, a_highpass,
filtered_selected_channel);

% Normalize the high-pass filtered signal
filtered_selected_channel_highpass = filtered_selected_channel_highpass /
max(abs(filtered_selected_channel_highpass));

% Plot the high-pass filtered signal in the time and frequency domain
```

```
subplot(5,5,7);
plot(time_vector, filtered_selected_channel_highpass);
title('High-pass Filtered Brain Signal (Cutoff 20 Hz) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(5,5,8);
filtered_fft_highpass = fft(filtered_selected_channel_highpass);
magnitude_filtered_highpass = abs(filtered_fft_highpass) / n_samples;
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_highpass(1:floor(n_samples/2)) /
max(magnitude_filtered_highpass(1:floor(n_samples/2)))); % Normalize the magnitude
title('High-pass Filtered Brain Signal (Cutoff 20 Hz) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the delta band (0.5 - 4 Hz)
delta_band_cutoff_frequency = [0.5 4]; % Cutoff frequencies for the delta band
(Hz)
[b_bandpass_delta, a_bandpass_delta] = butter(4, delta_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the delta band

% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_delta = filtfilt(b_bandpass_delta,
a_bandpass_delta, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_delta =
filtered_selected_channel_bandpass_delta /
max(abs(filtered_selected_channel_bandpass_delta));

% Plot the band-pass filtered signal in the time domain
subplot(5,5,9);
plot(time_vector, filtered_selected_channel_bandpass_delta);
title('Band-pass Filtered Brain Signal (Delta Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_delta = fft(filtered_selected_channel_bandpass_delta);
magnitude_filtered_bandpass_delta = abs(filtered_fft_bandpass_delta) / n_samples;

subplot(5,5,10);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_delta(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_delta(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Delta Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the theta band (4 - 7 Hz)
theta_band_cutoff_frequency = [4 7]; % Cutoff frequencies for the theta band (Hz)
[b_bandpass_theta, a_bandpass_theta] = butter(4, theta_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the theta band
```

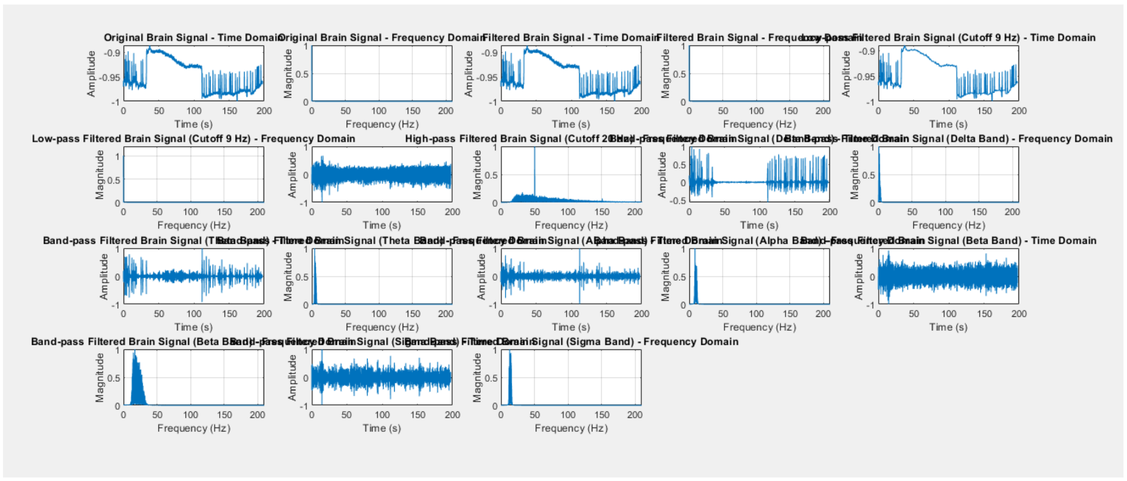
```
% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_theta = filtfilt(b_bandpass_theta,
a_bandpass_theta, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_theta =
filtered_selected_channel_bandpass_theta /
max(abs(filtered_selected_channel_bandpass_theta));

% Plot the band-pass filtered signal in the time domain
subplot(5,5,11);
plot(time_vector, filtered_selected_channel_bandpass_theta);
title('Band-pass Filtered Brain Signal (Theta Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_theta = fft(filtered_selected_channel_bandpass_theta);
magnitude_filtered_bandpass_theta = abs(filtered_fft_bandpass_theta) / n_samples;

subplot(5,5,12);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_theta(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_theta(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Theta Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the alpha band (8 - 12 Hz)
alpha_band_cutoff_frequency = [8 12]; % Cutoff frequencies for the alpha band (Hz)
[b_bandpass_alpha, a_bandpass_alpha] = butter(4, alpha_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the alpha band

% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_alpha = filtfilt(b_bandpass_alpha,
a_bandpass_alpha, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_alpha =
filtered_selected_channel_bandpass_alpha /
max(abs(filtered_selected_channel_bandpass_alpha));

% Plot the band-pass filtered signal in the time domain
subplot(5,5,13);
plot(time_vector, filtered_selected_channel_bandpass_alpha);
title('Band-pass Filtered Brain Signal (Alpha Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_alpha = fft(filtered_selected_channel_bandpass_alpha);
magnitude_filtered_bandpass_alpha = abs(filtered_fft_bandpass_alpha) / n_samples;

subplot(5,5,14);
```

```matlab
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_alpha(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_alpha(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Alpha Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the beta band (13 - 30 Hz)
beta_band_cutoff_frequency = [13 30]; % Cutoff frequencies for the beta band (Hz)
[b_bandpass_beta, a_bandpass_beta] = butter(4, beta_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the beta band

% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_beta = filtfilt(b_bandpass_beta,
a_bandpass_beta, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_beta = filtered_selected_channel_bandpass_beta
/ max(abs(filtered_selected_channel_bandpass_beta));

% Plot the band-pass filtered signal in the time domain
subplot(5,5,15);
plot(time_vector, filtered_selected_channel_bandpass_beta);
title('Band-pass Filtered Brain Signal (Beta Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_beta = fft(filtered_selected_channel_bandpass_beta);
magnitude_filtered_bandpass_beta = abs(filtered_fft_bandpass_beta) / n_samples;

subplot(5,5,16);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_beta(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_beta(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Beta Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the sigma band (12 - 16 Hz)
sigma_band_cutoff_frequency = [12 16]; % Cutoff frequencies for the sigma band
(Hz)
[b_bandpass_sigma, a_bandpass_sigma] = butter(4, sigma_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the sigma band

% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_sigma = filtfilt(b_bandpass_sigma,
a_bandpass_sigma, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_sigma =
filtered_selected_channel_bandpass_sigma /
max(abs(filtered_selected_channel_bandpass_sigma));
```

```
% Plot the band-pass filtered signal in the time domain
subplot(5,5,17);
plot(time_vector, filtered_selected_channel_bandpass_sigma);
title('Band-pass Filtered Brain Signal (Sigma Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_sigma = fft(filtered_selected_channel_bandpass_sigma);
magnitude_filtered_bandpass_sigma = abs(filtered_fft_bandpass_sigma) / n_samples;

subplot(5,5,18);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_sigma(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_sigma(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Sigma Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;
filtered_bandpass_sigma(1:floor(n_samples/2)); % Normalize the magnitude
title('Band-pass Filtered Brain Signal (Sigma Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-pass filter for the gamma band (30 - 100 Hz)
gamma_band_cutoff_frequency = [30 100]; % Cutoff frequencies for the gamma band
(Hz)
[b_bandpass_gamma, a_bandpass_gamma] = butter(4, gamma_band_cutoff_frequency /
(Fs/2), 'bandpass'); % Design the band-pass filter for the gamma band

% Apply the band-pass filter to the notch filtered signal
filtered_selected_channel_bandpass_gamma = filtfilt(b_bandpass_gamma,
a_bandpass_gamma, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandpass_gamma =
filtered_selected_channel_bandpass_gamma /
max(abs(filtered_selected_channel_bandpass_gamma));

% Plot the band-pass filtered signal in the time domain
subplot(5,5,19);
plot(time_vector, filtered_selected_channel_bandpass_gamma);
title('Band-pass Filtered Brain Signal (Gamma Band) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-pass filtered signal in the frequency domain
filtered_fft_bandpass_gamma = fft(filtered_selected_channel_bandpass_gamma);
magnitude_filtered_bandpass_gamma = abs(filtered_fft_bandpass_gamma) / n_samples;

subplot(5,5,20);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandpass_gamma(1:floor(n_samples/2)) /
max(magnitude_filtered_bandpass_gamma(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-pass Filtered Brain Signal (Gamma Band) - Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-stop filter to remove 60 Hz powerline noise
powerline_frequency = 60; % Powerline frequency (Hz)
[b_bandstop_powerline, a_bandstop_powerline] = butter(4, [powerline_frequency-1,
powerline_frequency+1] / (Fs/2), 'stop'); % Design the band-stop filter for
powerline noise

% Apply the band-stop filter to the notch filtered signal
filtered_selected_channel_bandstop_powerline = filtfilt(b_bandstop_powerline,
a_bandstop_powerline, filtered_selected_channel);

% Normalize the filtered signal
filtered_selected_channel_bandstop_powerline =
filtered_selected_channel_bandstop_powerline /
max(abs(filtered_selected_channel_bandstop_powerline));

% Plot the band-stop filtered signal in the time domain
subplot(5,5,21);
plot(time_vector, filtered_selected_channel_bandstop_powerline);
title('Band-stop Filtered Brain Signal (Powerline Noise Removal) - Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-stop filtered signal in the frequency domain
filtered_fft_bandstop_powerline =
fft(filtered_selected_channel_bandstop_powerline);
magnitude_filtered_bandstop_powerline = abs(filtered_fft_bandstop_powerline) /
n_samples;

subplot(5,5,22);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandstop_powerline(1:floor(n_samples/2)) /
max(magnitude_filtered_bandstop_powerline(1:floor(n_samples/2)))); % Normalize the
magnitude
title('Band-stop Filtered Brain Signal (Powerline Noise Removal) - Frequency
Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;

% Design a band-stop filter to remove 120 Hz powerline noise
powerline_frequency_2 = 120; % Second harmonic of powerline frequency (Hz)
[b_bandstop_powerline_2, a_bandstop_powerline_2] = butter(4,
[powerline_frequency_2-1, powerline_frequency_2+1] / (Fs/2), 'stop'); % Design the
band-stop filter for second harmonic of powerline noise

% Apply the band-stop filter to the notch filtered signal
filtered_selected_channel_bandstop_powerline_2 = filtfilt(b_bandstop_powerline_2,
a_bandstop_powerline_2, filtered_selected_channel);

% Normalize the filtered signal
```

```
filtered_selected_channel_bandstop_powerline_2 =
filtered_selected_channel_bandstop_powerline_2 /
max(abs(filtered_selected_channel_bandstop_powerline_2));
% Plot the band-stop filtered signal in the time domain
subplot(5,5,23);
plot(time_vector, filtered_selected_channel_bandstop_powerline_2);
title('Band-stop Filtered Brain Signal (Powerline Noise Removal - 2nd Harmonic) -
Time Domain');
xlabel('Time (s)');
ylabel('Amplitude');

% Plot the band-stop filtered signal in the frequency domain
filtered_fft_bandstop_powerline_2 =
fft(filtered_selected_channel_bandstop_powerline_2);
magnitude_filtered_bandstop_powerline_2 = abs(filtered_fft_bandstop_powerline_2) /
n_samples;

subplot(5,5,24);
plot(frequency_vector(1:floor(n_samples/2)),
magnitude_filtered_bandstop_powerline_2(1:floor(n_samples/2)) /
max(magnitude_filtered_bandstop_powerline_2(1:floor(n_samples/2)))); % Normalize
the magnitude
title('Band-stop Filtered Brain Signal (Powerline Noise Removal - 2nd Harmonic) -
Frequency Domain');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
xlim([0 Fs/2]);
grid on;
% Plot the original and filtered signals for comparison
subplot(5,5,25);
plot(time_vector, selected_channel / max(abs(selected_channel)), 'b', 'LineWidth',
1); % Original signal in blue
hold on;
plot(time_vector, filtered_selected_channel_bandstop_powerline_2, 'r',
'LineWidth', 1); % Filtered signal in red
hold off;
title('Original vs Filtered Brain Signal Comparison');
xlabel('Time (s)');
ylabel('Amplitude');
legend('Original', 'Filtered');
grid on;
```

Results:



Code Explanation

This MATLAB code performs various signal processing operations on EEG (Electroencephalogram) data to preprocess it for further analysis. Let's break down the code and explain each part:

1. **Import Data:** The EEG data is loaded from a .mat file, assuming it contains EEG signals. The 26th channel is then selected for further processing.

2. **Time Vector and Visualization:** A time vector is created for visualization purposes. The original EEG signal is plotted in both the time and frequency domains to provide an initial view of the data.

3. **Notch Filtering:** A notch filter is designed and applied to remove 55 Hz noise, commonly originating from power lines. The filtered signal is normalized and plotted in both domains.

4. **Low-pass Filtering:** A low-pass filter is designed to attenuate frequencies above 5 Hz, effectively removing high-frequency noise and retaining low-frequency components of interest. The filtered signal is again normalized and plotted.

5. **High-pass Filtering:** A high-pass filter is designed to attenuate frequencies below 20 Hz, eliminating slow signal drift and retaining high-frequency components. The filtered signal is normalized and plotted.

6. **Band-pass Filtering:** Band-pass filters are designed for specific frequency bands (delta, theta, alpha, beta, sigma, and gamma) to isolate frequency ranges of interest in the EEG signal. Each filtered signal is normalized and plotted in both time and frequency domains.

7. Band-stop Filtering: Band-stop filters are designed to remove powerline noise at 60 Hz and its second harmonic at 120 Hz. The filtered signals are normalized and plotted for comparison with the original signal.

8. Signal Comparison: The original EEG signal and the band-stop filtered signal (after removing powerline noise) are plotted together for visual comparison.

This code aims to preprocess EEG data by removing noise and isolating frequency bands of interest, making it suitable for subsequent analysis such as feature extraction or classification of brain states.

Conclusion

This project successfully demonstrates the application of digital signal processing techniques to EEG data using MATLAB. By employing a series of filters, we effectively remove various types of noise and isolate specific frequency bands associated with different brain activities. The detailed graphical analysis confirms the efficacy of these filters, resulting in cleaner EEG signals that are suitable for further neurological analysis and research. The methodology and techniques used in this project can be applied to a wide range of EEG datasets, providing a robust framework for enhancing the quality of EEG data and facilitating more accurate interpretations of brain activity.