3rd exp baseline wander and high frequency noise

```
% Load ECG data (Replace with your actual file path if needed)
ecg_data = load('101m.mat'); % Load ECG data
ecg_signal = ecg_data.val(1, :); % Use the first channel of the loaded ECG signal
% Sampling frequency
fs = 250; % Replace with the actual sampling frequency if different
% Define the time vector based on the length of the signal and sampling frequency
t = (0:length(ecg_signal)-1) / fs; % Time vector in seconds
% Plot the original ECG signal
figure;
subplot(3,1,1);
plot(t, ecg_signal);
title('Original ECG Signal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
% 1. Remove Baseline Wander
% Use a high-pass filter to remove baseline wander (e.g., cutoff frequency = 0.5 Hz)
fc_baseline = 0.5; % Cutoff frequency for baseline wander in Hz
[b_high, a_high] = butter(2, fc_baseline / (fs / 2), 'high');
ecg_no_baseline = filtfilt(b_high, a_high, ecg_signal);
% 2. Remove High-Frequency Noise
% Use a low-pass filter to remove high-frequency noise (e.g., cutoff frequency = 40 Hz)
fc_noise = 40; % Cutoff frequency for high-frequency noise in Hz
[b_low, a_low] = butter(4, fc_noise / (fs / 2), 'low');
ecg_filtered = filtfilt(b_low, a_low, ecg_no_baseline);
% Plot the results
subplot(3,1,2);
plot(t, ecg_no_baseline);
title('After Baseline Wander Removal');
```

```
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
subplot(3,1,3);
plot(t, ecg_filtered);
title('After High-Frequency Noise Removal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
```

4th exp heart rate

```
% Load ECG Data
ecg_data = load('101m.mat'); % Load ECG signal
ecg_signal = ecg_data.val(1, :); % Use the first channel of the ECG signal
fs = 250; % Sampling frequency in Hz
t = (0:length(ecg_signal)-1) / fs; % Time vector in seconds

% Plot the ECG Signal
figure;
plot(t, ecg_signal);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('ECG Signal');
grid on;

% Detect R-peaks
% Using MATLAB's findpeaks function
[~, r_peaks] = findpeaks(ecg_signal, 'MinPeakHeight', 0.5, 'MinPeakDistance', 0.6*fs);

% Mark R-peaks on the ECG signal
hold on;
```

```
plot(r_peaks / fs, ecg_signal(r_peaks), 'ro');
legend('ECG Signal', 'R-peaks');

% Calculate RR Intervals
rr_intervals = diff(r_peaks) / fs; % RR intervals in seconds
heart_rate = 60 ./ rr_intervals; % Heart rate in bpm

% Average Heart Rate
avg_heart_rate = mean(heart_rate);

% Display Results
disp('Heart Rate Analysis:');
disp(['RR Intervals (s): ', num2str(rr_intervals)]);
disp(['Instantaneous Heart Rates (bpm): ', num2str(heart_rate)]);
disp(['Average Heart Rate (bpm): ', num2str(avg_heart_rate)]);

% Annotate the graph for easier understanding
for i = 1:length(r_peaks)-1
   xline(r_peaks(i) / fs, '--g', 'LineWidth', 1); % Mark RR intervals
   text((r_peaks(i)+r_peaks(i+1))/(2*fs), max(ecg_signal)*0.9, ...
      ['RR = ', num2str(rr_intervals(i), '%.2f'), ' s'], ...
      'HorizontalAlignment', 'center');
end
```

5th exp eeg

```
% Load EEG data
data = load('chb01_02_edfm.mat'); % Corrected filename
disp('Loaded data successfully!');

eeg_signal = data.val; % Assuming EEG data is stored in 'val'

% Get the number of channels in the EEG data
```

```
num_channels = size(eeg_signal, 1);

% Set up the subplot grid
rows = 11; % Number of rows in the subplot grid
cols = 2;  % Number of columns in the subplot grid

% Plot each channel
figure;
for channel = 1:num_channels
    subplot(rows, cols, channel); % Create subplot
    plot(eeg_signal(channel, :)); % Plot the signal for the current channel
    title(['EEG Channel ' num2str(channel)], 'FontSize', 10); % Title with channel number
    xlabel('Sample Number', 'FontSize', 10); % X-axis label
    ylabel('Amplitude (μV)', 'FontSize', 10); % Y-axis label
    grid on; % Enable grid
end

% Add a global title to the figure
sgtitle('EEG Signals for All Channels', 'FontSize', 16);
```

7th exp fft

```
% Load the ECG data
data = load('101m.mat'); % Replace with your ECG file
ecg_signal = data.val(1, :); % Assuming the ECG signal is stored in the first channel
fs = 250; % Sampling frequency in Hz (adjust as per your data)

% Time vector
N = length(ecg_signal); % Number of samples
t = (0:N-1) / fs; % Time in seconds

% Plot the original ECG signal
figure;
```

```
subplot(3, 2, 1);
plot(t, ecg_signal);
title('Original ECG Signal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;

% --- Filtering the ECG Signal ---
% Remove baseline wander using a high-pass filter (cutoff frequency 0.5 Hz)
fc_baseline = 0.5; % Cutoff frequency for baseline wander in Hz
[b_high, a_high] = butter(2, fc_baseline / (fs / 2), 'high');
ecg_filtered = filtfilt(b_high, a_high, ecg_signal);

% Remove high-frequency noise using a low-pass filter (cutoff frequency 40 Hz)
fc_noise = 40; % Cutoff frequency for high-frequency noise in Hz
[b_low, a_low] = butter(4, fc_noise / (fs / 2), 'low');
ecg_filtered = filtfilt(b_low, a_low, ecg_filtered);

% Plot the filtered ECG signal
subplot(3, 2, 2);
plot(t, ecg_filtered);
title('Filtered ECG Signal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;

% --- Compute FFT for Original ECG ---
ecg_fft = fft(ecg_signal); % FFT of original signal
frequencies = (0:N-1) * (fs / N); % Frequency axis in Hz
ecg_fft_magnitude = abs(ecg_fft / N); % Magnitude of FFT
ecg_fft_magnitude = ecg_fft_magnitude(1:N/2+1); % Single-sided spectrum
```

```
frequencies = frequencies(1:N/2+1); % Only keep positive frequencies

% Plot FFT of Original ECG Signal
subplot(3, 2, 3);
plot(frequencies, ecg_fft_magnitude);
title('FFT of Original ECG Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% --- Compute FFT for Filtered ECG ---
ecg_filtered_fft = fft(ecg_filtered); % FFT of filtered signal
ecg_filtered_fft_magnitude = abs(ecg_filtered_fft / N); % Magnitude of FFT
ecg_filtered_fft_magnitude = ecg_filtered_fft_magnitude(1:N/2+1); % Single-sided spectrum

% Plot FFT of Filtered ECG Signal
subplot(3, 2, 4);
plot(frequencies, ecg_filtered_fft_magnitude);
title('FFT of Filtered ECG Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Optional: Display Filter Characteristics
figure;
subplot(2,1,1);
fvtool(b_high, a_high, 'Fs', fs); % High-pass filter response
title('High-pass Filter Response (Baseline Wander Removal)');

subplot(2,1,2);
fvtool(b_low, a_low, 'Fs', fs); % Low-pass filter response
```

```
title('Low-pass Filter Response (Noise Removal)');
```

8th exp artifacts

```
% Load ECG data (Replace with your own data file)
data = load('101m.mat'); % Example ECG data
ecg_signal = data.val(1, :); % Assuming ECG signal is in the first channel
fs = 250; % Sampling frequency (adjust according to your data)
% Time vector
N = length(ecg_signal); % Number of samples
t = (0:N-1) / fs; % Time in seconds
% Plot the original ECG signal
figure;
subplot(3, 2, 1);
plot(t, ecg_signal);
title('Original ECG Signal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
% --- 1. Remove Baseline Wander (High-pass filter) ---
fc_baseline = 0.5; % Cutoff frequency for baseline wander in Hz
[b_high, a_high] = butter(2, fc_baseline / (fs / 2), 'high');
ecg_no_baseline = filtfilt(b_high, a_high, ecg_signal);
% Plot the ECG after baseline wander removal
subplot(3, 2, 2);
plot(t, ecg_no_baseline);
title('ECG After Baseline Wander Removal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
% --- 2. Remove High-frequency Noise (Low-pass filter) ---
fc_noise = 40; % Cutoff frequency for high-frequency noise in Hz
```

```
[b_low, a_low] = butter(4, fc_noise / (fs / 2), 'low');
ecg_filtered = filtfilt(b_low, a_low, ecg_no_baseline);
% Plot the ECG after high-frequency noise removal
subplot(3, 2, 3);
plot(t, ecg_filtered);
title('ECG After High-Frequency Noise Removal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
% --- 4. Remove Motion Artifacts (Band-pass filter) ---
fc_low = 0.5; % Low cutoff frequency for ECG signal
fc_high = 50; % High cutoff frequency for ECG signal
[b_bandpass, a_bandpass] = butter(4, [fc_low / (fs / 2), fc_high / (fs / 2)], 'bandpass');
ecg_bandpass_filtered = filtfilt(b_bandpass, a_bandpass, ecg_filtered);
% Plot the ECG after motion artifact removal
subplot(3, 2, 5);
plot(t, ecg_bandpass_filtered);
title('ECG After Motion Artifact Removal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
% Powerline Interference (50 Hz)
fc_line = 50; % Powerline frequency (50 Hz)
Q_factor = 35; % Quality factor (higher Q gives a narrower notch)
% Normalized frequency (W0) for the notch filter
W0 = fc_line / (fs / 2); % W0 is normalized by Nyquist frequency
BW = W0 / Q_factor; % BW is related to Q-factor
[b_notch, a_notch] = iirnotch(W0, BW); % W0 and BW define the notch filter
ecg_no_interference = filtfilt(b_notch, a_notch, ecg_filtered);
% Plot the ECG after powerline interference removal
subplot(3,2,4)
```

```
plot(t, ecg_no_interference);
title('ECG After Powerline Interference Removal');
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
```