12/5/2025

Surya Ghosh
52426797

# EE3053 Design Exercise

Table of Contents

3. Motivation and Approach

4. Execution

10. Evaluation

11. Conclusion

12. Sources

13. Appendix

Surya Ghosh
52426797

## Motivation

In modern society, digital technology has become a fundamental part of daily life and has shaped the medium through which we interact with and define the world. It has especially revitalized and revolutionized our healthcare, giving us the tools necessary to maintain human longevity. The electrocardiogram (ECG) is a fundamental tool in today's practice; it gives practitioners critical insight into a patient's physiology and allows for rapid prognosis. An ECG, however, is rife with unwanted information, noise, that must be delicately filtered before the signal can be used. In this exercise, we will determine and justify a digital signal processing pipeline that will target bands of noise while minimizing impact on critical ECG segments. We will also inspect which frequencies should be considered as noise and evaluate the success of our pipeline in eliminating them.

## Approach

There were 3 given sources of noise in the assessment outline; baseline wander, mains hum, and a harmonic, that we were tasked with filtering out.

The baseline wander is a slow fluctuation caused by varying sources such as patient respiration, movement, and electrode-skin contact instability. Due to its very low frequency, it appears as a slow wave that the individual pulses ride upon. To identify the specific frequencies that the wander occupies, we can compare the provided clean signal (*Figure 1*) with the noisy signal (*Figure 2*) and see which frequencies below 0.7 Hz are minimized. In doing so, we see that 0.2 Hz is affected, while frequencies above and below are preserved, identifying it as the baseline wander. We will see that filtering this will be challenging as the frequency (0.2 Hz) is considerably below the sampling frequency (360 Hz) and demands a high attenuation to not affect neighbouring frequencies. We decided to try both IIR and FIR filters in removing the baseline wander, then proceeded with the most sensible result. In order to remove the mains hum at 50 Hz we used an IIR notch filter given the high selectivity needed. We approached the harmonic at 100 Hz similarly. Finally, to filter the white noise present, we tried to use a wavelet transform to identify what frequencies the noise occupied. As we will see, this proved challenging, due to which we ended up using a final low pass filter.

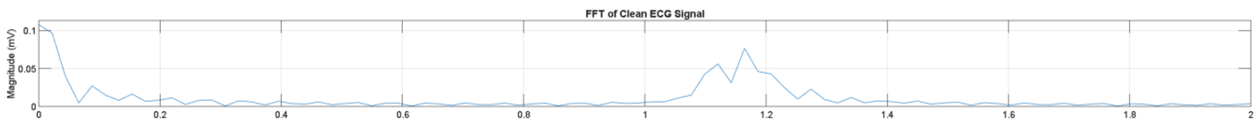


*Figure 1 FFT of clean ECG signal*

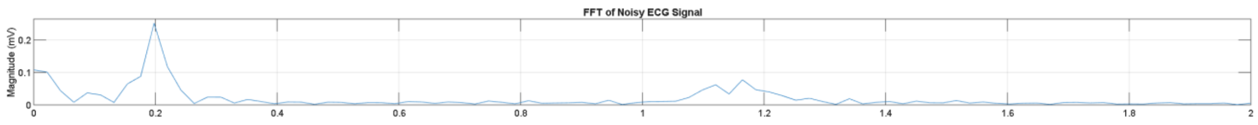


*Figure 2 FFT of noisy ECG signal*

## Execution

In removing the baseline wander, we iterated through a variety of filters and filter combinations. First, we attempted to remove the $0.2$ Hz spike using an FIR band-stop filter. We decided to start with an FIR filter as the stop band was very narrow, and an IIR filter may have become unstable while executing the very precise calculations. In respect to the clean signal, we set the lower edge of the stop band to $0.1$ Hz and the upper edge to $0.7$ Hz. The required filter order can be calculated using *Equation 1*, where $A = 3.3$ for Hamming windows, $f_s = 360$ Hz, and $\Delta f = 0.2$ Hz (transition band), giving us $N = 5{,}940$.

$$N = \frac{A \times f_s}{\Delta f}$$

*Equation 1*

In creating the filter, we can use the command 'fir1' within MATLAB to create the desired FIR filter with the required frequencies and filter order. After creating and applying the band-stop filter through convolution, we have our filtered signal (*Figure 3)*. As FIR filters impart a linear phase delay (group delay), they must be non-causal (type 1) in order to apply the delay uniformly, this results in the entire signal being delayed/shifted as shown. The delay time can be calculated through *Equation 2*, giving us $\tau_{delay} = 8.24$ seconds, which we can confirm visually in *Figure 3*. In order to use the filtered signal, we need to apply a delay correction by shifting the signal by $\frac{N}{2}$ samples and subsequently removing the extra data, giving us the processed and aligned signal (*Figure 4)*.

$$\tau_{delay} = \frac{{}^{N}\!/_{2}}{f_s}$$

*Equation 2*

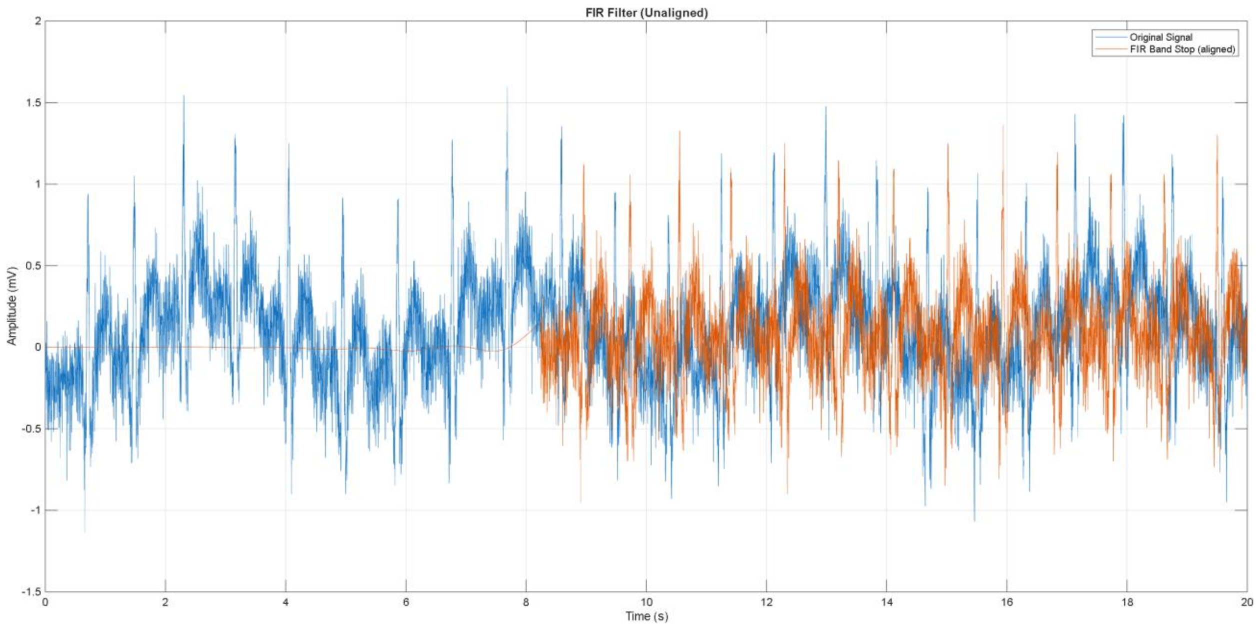


*Figure 3 FIR Filter (unaligned)*

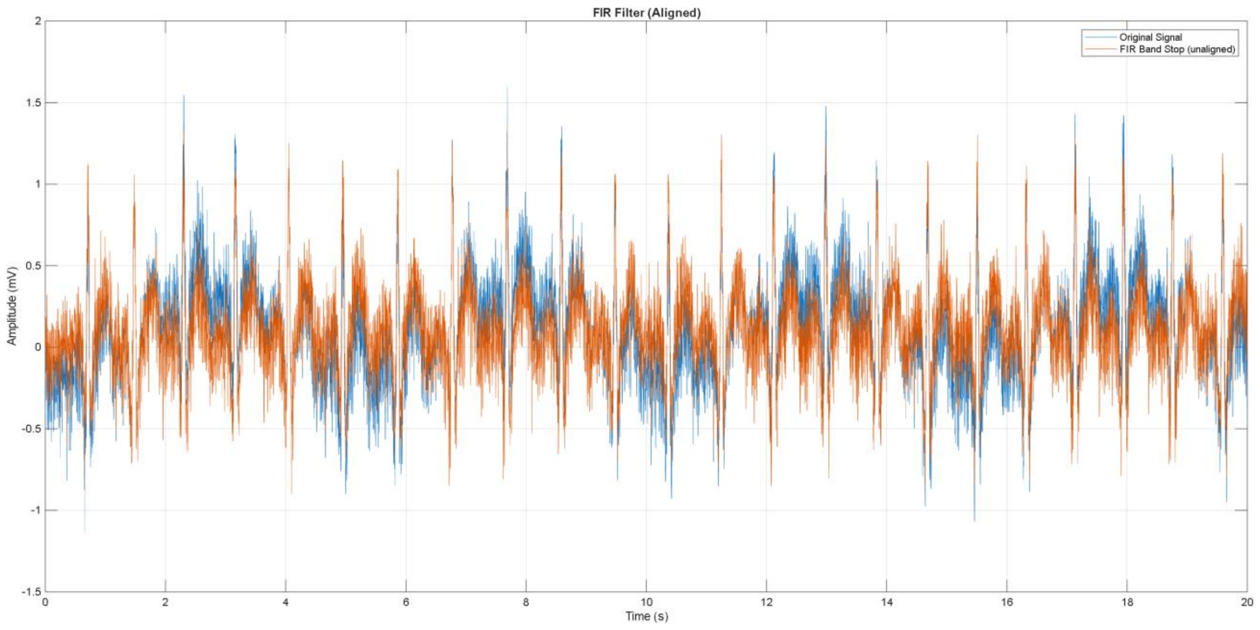


*Figure 4 FIR Filter (aligned)*

As briefly mentioned, we decided to use a Hamming window to design the FIR filter. We choose the Hamming window because of its high side lobe attenuation, which suppressed frequencies from leaking into neighbouring ones, and maintained a narrow enough main lobe such that the desired frequencies were isolated. We also considered the Hanning window due to its strong suppression of the sidelobes, but we decided not to use it as the lower stopband attenuation on the first lobe may have affected ECG frequencies <0.1 Hz. We can see this in *Figure 5*, where the first lobe for Hanning is considerably higher than that of Hamming.

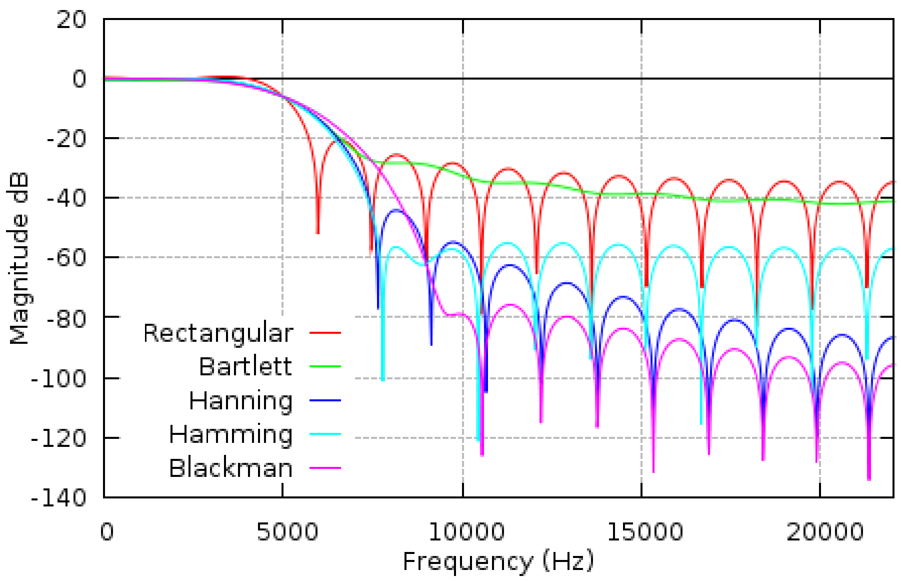


*Figure 5 Frequency response for different windows ("FIR Filters by Windowing - the Lab Book Pages")*

After filtering and shifting the signal, we can see how our signal was affected by looking at its FFT (*Figure 6*). Comparing it to *Figure 1*, the FFT of the given clean signal, we can see that the frequency at 0.2 Hz was effectively taken out. However, there was a slight effect on frequencies right outside of the stop band (< 0.2 Hz), which may be reduced by increasing the filter order, but for our needs, this effect is acceptable.

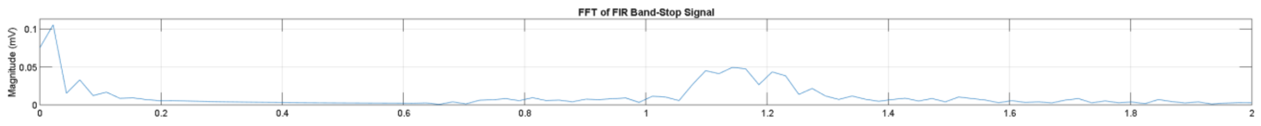
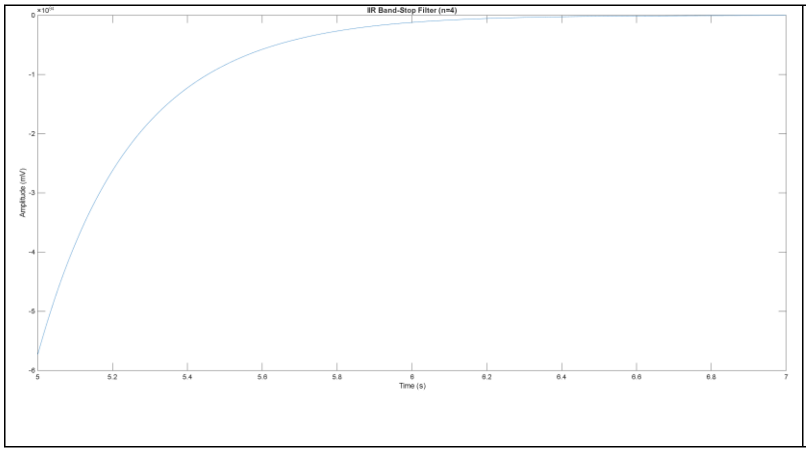


*Figure 6 FFT of FIR band-stop signal*

Our approach of using an FIR band-stop filter gave us acceptable results, but at the cost of high computational resources and a large delay. Let us try using an IIR band-stop filter and determine if it has acceptable performance. We began with an 4th order Butterworth IIR band stop filter with a stop band between 0.1 Hz and 0.7 Hz. From *Figure 7*, we can see that this resulted in instability, with the output growing to very large numbers. As the filter coefficients are dependent upon the sampling rate and cutoff frequencies, the relatively small frequencies with respect to the sampling rate resulted in roundoff errors and numerical instability. The double numerical precision used within MATLAB is not sufficient to represent the numbers within our calculations. We can confirm this through the pole-zero plot (*Figure 8*) and see that there are poles/zeros shifted outside of the unit circle. As we increase the order, the errors grow, and the poles/zeros are shifted further away (*Figures 9/10*). Additionally, IIR filters impart a non-linear phase distortion upon the signal which must be corrected. This was done by using the ‘filtfilt’ command, which mitigated any phase distortions through processing the signal in both forward and reverse directions, causing the distortion to cancel out.



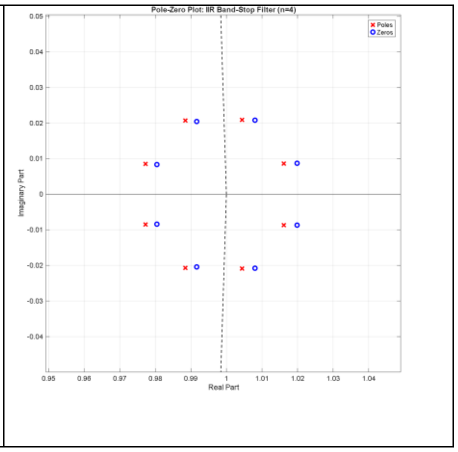
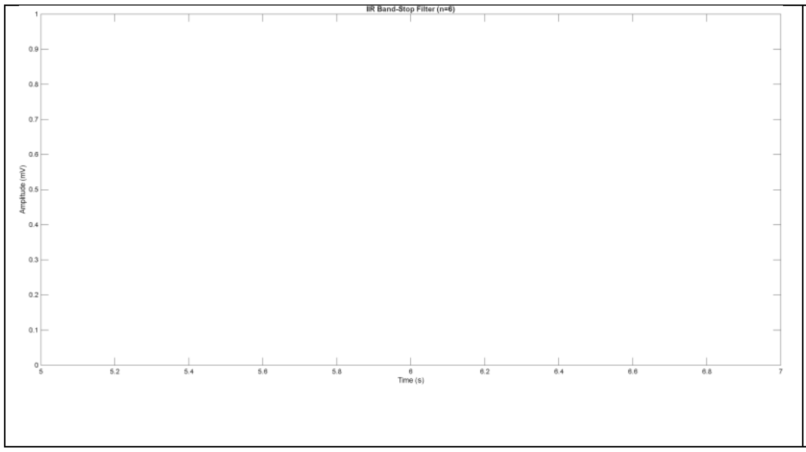


*Figure 7 IIR band-stop filter [n=4] (left)*
*Figure 8 IIR band-stop filter pole/zero plot [n=4] (right)*



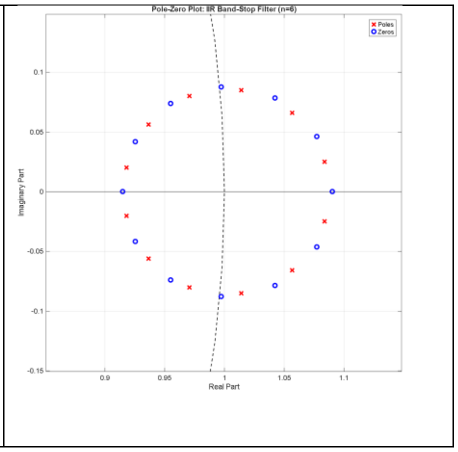
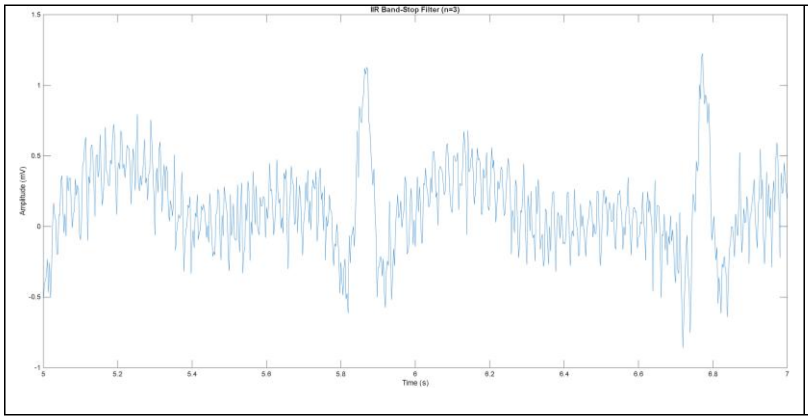


*Figure 9 IIR band-stop filter [n=6] (left)*
*Figure 10 IIR band-stop filter pole/zero plot [n=6] (right)*

We can increase the stability by either expanding the stop band or reducing the filter order. In reducing the filter order, we reduce the number of computations carried out, leading to less truncations (as we are dealing with irrational numbers and trigonometric operations), and therefore lower error/instability. In widening stop pass band (i.e. 1-5 Hz), the poles of the filter are shifted inside the unit circle, stabilizing the filter; however, the stop band is already the widest that it may be without affecting the ECG frequencies. As we want to preserve the neighbouring frequencies and maintain a stable filter, we decide to reduce the filter order to 3.

The response is shown in *Figures 11/12*, where the 3rd order filter stays stable. From *Figure 13*, we can see that the baseline wander is effectively taken out, however, due to its lower order and poorer stopband attenuation, we can see that it affects neighbouring frequencies and especially distorts the lower ones.



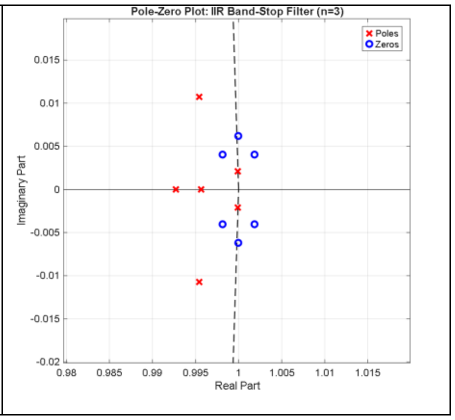


*Figure 11 IIR band-stop filter [n=3] (left)*
*Figure 12 IIR band-stop filter pole/zero plot [n=3] (right)*

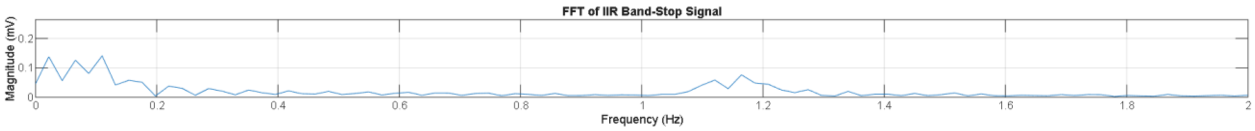


*Figure 13 FFT of IIR band stop signal (N=3)*

As shown, it is difficult to effectively filter the frequencies between 0.1 Hz and 0.7 Hz given the relatively high sampling frequency. The 3rd order IIR band stop filter and the 5,940th order FIR filter both succeed in doing so, but each with their downsides. The FIR filter effectively preserves frequencies outside of the stop band, and has high pass band attenuation, but it has a very large delay. The IIR filter is sufficient in filtering the noise and does not impart a delay but distorts neighbouring frequencies. The FIR delay makes any real time use impractical, so we will use the IIR band stop filter.

After filtering out the baseline wander, we can further process our signal to remove the mains hum. The mains hum is caused by the constant 50 Hz AC power that is used to power the ECG device. As there aren't any critical frequencies of the ECG near 50 Hz, we don't need to be overly cautious in considering attenuation, ripple, or in picking a filtering window. We used a 2nd order IIR notch filter given that the range of frequencies we wanted to eliminate was very small. We could have used an FIR filter, which would also be appropriate as we do not need high attenuation or performance. From *Figure 14*, we can see that the IIR filter was adequate.

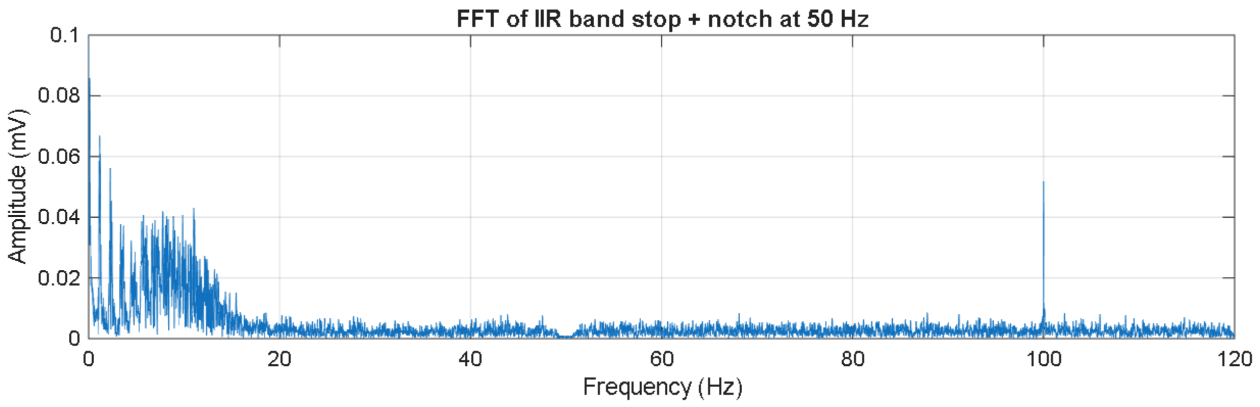


*Figure 14 FFT of IIR band stop and notch at 50 Hz*

We can treat the harmonic at 100 Hz similarly and use an IIR notch filter with the same order and bandwidth. After doing so, we can see that our signals' FFT (*Figure 15*) is like that of the clean one (*Figure 16)*.

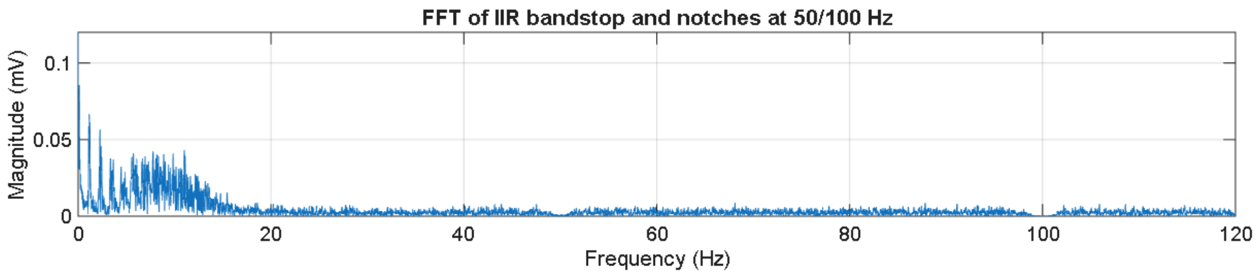


*Figure 15 FFT of IIR band stop and notch at 50/100 Hz*

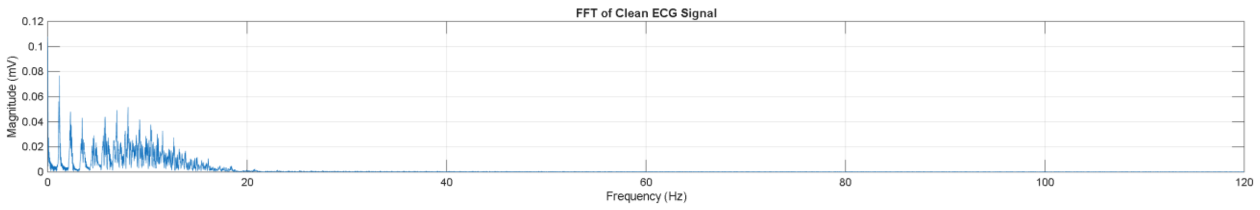


*Figure 16 FFT of clean ECG signal*

After all of the specified sources of noise have been removed, there is still a constant low magnitude noise throughout the entire frequency spectrum. This is white noise, which distorts all of the frequencies within our signal. As it permeates through the ECG frequencies, the FFT cannot identify between noise and ECG components. Therefore, the FFT approach will not work, however we can approach this through Wavelet Transform (WT). The WT separates the ECG components from noise by analyzing the signal at different frequencies and times. As wavelets describe the time duration for which frequencies exist, the noise appears as small components on all time scales, whereas the ECG appears as a few large components. Therefore, we can set a threshold based upon these coefficients and filter the noise out. Through online resources and artificial intelligence, we produced the signal shown in *Figure 17*. We can see that the resulting signal is quite distorted, but the possibility for a corrected filter remains. As tuning the WT filter may be outside of our scope, we decided to implement a low pass filter to filter out frequencies above 25 Hz, as no frequency above this has significant magnitude in the clean signal. Therefore, we produced the final signal (*Figure 18*) and its corresponding FFT plot (*Figure 19*).

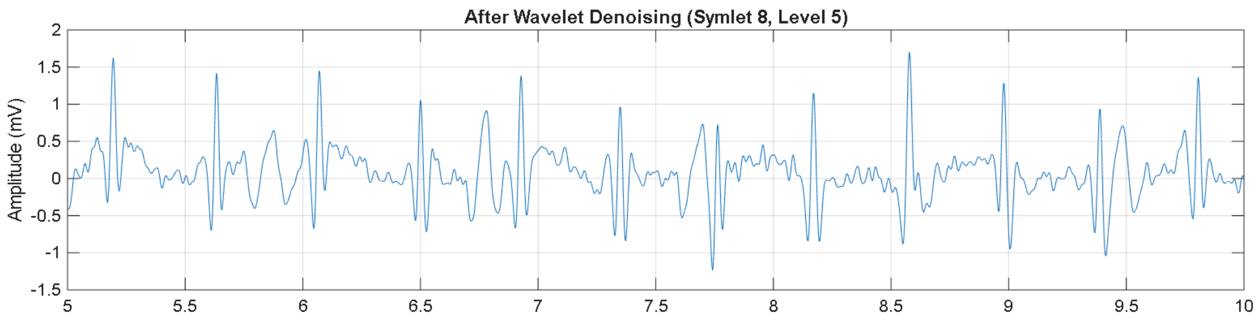


*Figure 17 ECG signal after wavelet denoising*

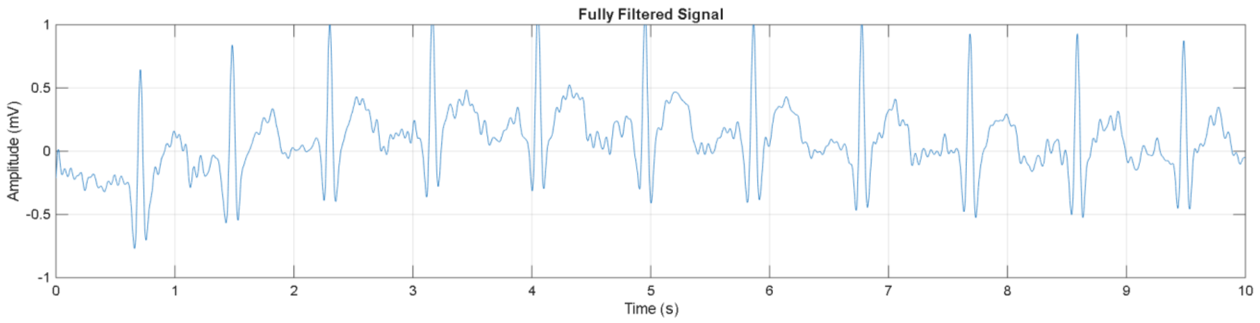


*Figure 18 Fully filtered signal*

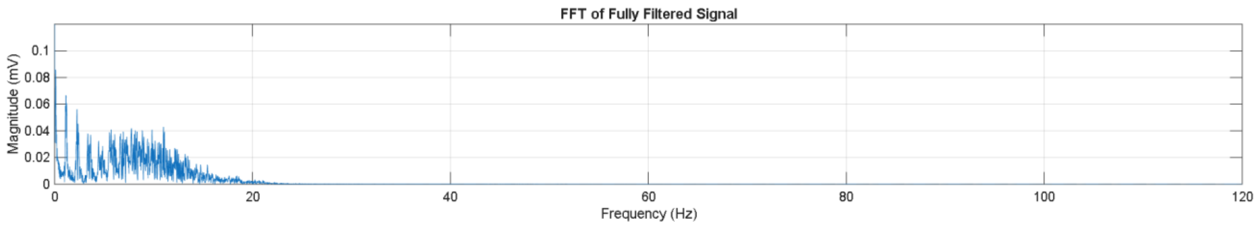


*Figure 19 FFT of fully filtered signal*

## Evaluation

As we are given a clean reference signal, we can use the root mean square error (*Equation 3*) to evaluate our filtering.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

*Equation 3 RMSE*

From *Equation 3*, we get an RMSE of $0.0979$ mV and a percent error of $54.79$ %, indicating a poor performance. The majority of our error comes from the distortions caused by the band stop filter used to eliminate the baseline wander. It was challenging to have a stop band that was very small relative to our sampling frequency. Although it reduced the wander considerably, it greatly distorted the frequencies neighbouring the stop band, which are unfortunately critical in representing the ECG. We could have used the FIR filter to stop the wander and greatly reduce distortions, but the time delay that it imparted was impractical. In using the IIR band stop filter, we also had to ensure that the filter stayed stable, and from the pole zero plot in *Figure 11*, we saw that there were two poles on the edge of the unit circle, indicating that the filter is marginally stable and may become unstable if there are large variations in the system. In evaluating our system, we can also measure the performance through the signal to noise ratio (SNR), which compares the parts of the signal that we want to preserve to the noise. Through *Equation 4* we calculated a $7.99$ dB SNR improvement, which demonstrates effective noise removal, but doesn't take the deformation of the wave into account. In the final low pass filter that we employed to reduce the white noise, we filtered out all frequencies above $25$ Hz on the basis that they do not appear to have significant magnitude in the clean FFT. However, this eliminates/affects parts of the ECG, specifically the QRS complex. We could have also used wavelet denoising here to target only the white noise.

$$SNR_{dB} = 10\log_{10}\frac{P_{signal}}{P_{noise}}$$

*Equation 4 SNR*

## Conclusion

Our filtering process successfully removed the baseline wander, the mains hum, and the harmonic. In removing the baseline wander, we faced challenges in balancing computational resources, filter stability, and signal preservation. Given that we must preserve frequencies below 0.2 Hz and produce a signal with a reasonable delay, we had to find a balance between filter stability, signal attenuation, and computation time. We proposed several solutions, of which the IIR band stop filter was the most practical. We did not have much difficulty in removing the mains hum at $50$ Hz or the harmonic. In order to improve our filtering pipeline, we could have further employed the wavelet denoising, which may have accurately eliminated the baseline wander. Our final filtered signal also had a very high percent error, which may make it unusable in a clinical setting. Overall, we explored a wide range of different filters and filtering methods, and created a pipeline that efficiently filtered our given signal.

Sources

"FIR Filters by Windowing - the Lab Book Pages." *Www.labbookpages.co.uk*, www.labbookpages.co.uk/audio/firWindowing.html.

Buendía-Fuentes, F., et al. "High-Bandpass Filters in Electrocardiography: Source of Error in the Interpretation of the ST Segment." *ISRN Cardiology*, vol. 2012, 2012, pp. 1–10, www.ncbi.nlm.nih.gov/pmc/articles/PMC3388307/, https://doi.org/10.5402/2012/706217.

"Practical Introduction to Digital Filtering - MATLAB & Simulink Example - MathWorks United Kingdom." *Uk.mathworks.com*, uk.mathworks.com/help/signal/ug/practical-introduction-to-digital-filtering.html.

"Wavelet Denoising - MATLAB & Simulink - MathWorks United Kingdom." *Uk.mathworks.com*, uk.mathworks.com/help/wavelet/ug/wavelet-denoising.html.

Appendix

```
% Surya Ghosh

% 52426797

%

% EE3053 Design Exercise 2025


clc;

close all;

clear;


%% 1. Load Data

load("ecg_noisy.mat");

Fs = fs; % Use the sampling frequency loaded from the file


% Ensure data is column vector and rename for clarity

x = x(:);

y_clean = y_clean(:); % Provided clean signal (reference)


% 1.1 Plot raw data

t = (0:length(x)-1) / Fs;


figure;

plot(t,x);

hold on;

plot(t,y_clean);

xlim([0,t(end)]);

xlabel("Time (s)");

ylabel("Amplitude (mV)");

legend("Noisy (x)", "Clean (Reference y\_clean)")
```

```
title("ECG Given Data");



%% 2. IIR Band Stop filter for baseline wander (n=3)
Fs = fs;            % Sampling frequency (loaded from .mat file)
f_stop_low = 0.1;   % Lower edge of the stopband (Hz)
f_stop_high = 0.7;  % Upper edge of the stopband (Hz)
N_order = 3;        % Filter Order (3rd order Butterworth for efficiency)

% normalizes cutoff frequencies
Wn_stop = [f_stop_low, f_stop_high] / (Fs/2);

[num_3, den_3] = butter(N_order, Wn_stop, 'stop');

% Use filtfilt to apply the filter forward and backward, eliminating phase distortion
y_iir_bandstop_filtered_3 = filtfilt(num_3, den_3, x);

figure;
t = (0:length(x)-1) / Fs;
plot(t, y_iir_bandstop_filtered_3);
xlim([5, 7]);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('IIR Band-Stop Filter (n=3)');

%% 2.5 Pole/zero plot for IIR band stop filter (n=3)
figure('Position', [100, 100, 600, 600]);
```

```
% Get poles and zeros of the filter
poles = roots(den_3);
zeros = roots(num_3);

% Plot unit circle
theta = linspace(0, 2*pi, 100);
plot(cos(theta), sin(theta), 'k--', 'LineWidth', 1);
hold on;

% Plot real and imaginary axes
plot([-1.2, 1.2], [0, 0], 'k-', 'LineWidth', 0.5);
plot([0, 0], [-1.2, 1.2], 'k-', 'LineWidth', 0.5);

plot(real(poles), imag(poles), 'rx', 'MarkerSize', 8, 'LineWidth', 2);
plot(real(zeros), imag(zeros), 'bo', 'MarkerSize', 6, 'LineWidth', 2);

% Formatting
axis equal;
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);
grid on;
xlabel('Real Part');
ylabel('Imaginary Part');
title('Pole-Zero Plot: IIR Band-Stop Filter (n=3)');

% Add legend
legend('', '', '', 'Poles', 'Zeros');
```

```
%% 3 IIR Band Stop filter for baseline wander (n=6)
Fs = fs;          % Sampling frequency (loaded from .mat file)
f_stop_low = 0.1;   % Lower edge of the stopband (Hz)
f_stop_high = 0.7;  % Upper edge of the stopband (Hz)
N_order = 6;       % Filter Order (4th order Butterworth for efficiency)

% normalizes cutoff frequencies
Wn_stop = [f_stop_low, f_stop_high] / (Fs/2);

[num_6, den_6] = butter(N_order, Wn_stop, 'stop');

% Use filtfilt to apply the filter forward and backward, eliminating phase distortion
y_iir_bandstop_filtered_6 = filtfilt(num_6, den_6, x);

figure;
t = (0:length(x)-1) / Fs;
plot(t, y_iir_bandstop_filtered_6);
xlim([5, 7]);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('IIR Band-Stop Filter (n=6)');

%% 3.5 Pole/zero plot for IIR band stop filter (n=6)
figure('Position', [100, 100, 600, 600]);

% Get poles and zeros of the filter
poles = roots(den_6);
```

```
zeros = roots(num_6);

% Plot unit circle
theta = linspace(0, 2*pi, 100);
plot(cos(theta), sin(theta), 'k--', 'LineWidth', 1);
hold on;

% Plot real and imaginary axes
plot([-1.2, 1.2], [0, 0], 'k-', 'LineWidth', 0.5);
plot([0, 0], [-1.2, 1.2], 'k-', 'LineWidth', 0.5);

plot(real(poles), imag(poles), 'rx', 'MarkerSize', 8, 'LineWidth', 2);
plot(real(zeros), imag(zeros), 'bo', 'MarkerSize', 6, 'LineWidth', 2);

% Formatting
axis equal;
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);
grid on;
xlabel('Real Part');
ylabel('Imaginary Part');
title('Pole-Zero Plot: IIR Band-Stop Filter (n=6)');

% Add legend
legend('', '', '', 'Poles', 'Zeros');

%% 4 IIR Band Stop filter for baseline wander (n=4)
Fs = fs;          % Sampling frequency (loaded from .mat file)
```

```
f_stop_low = 0.1;   % Lower edge of the stopband (Hz)
f_stop_high = 0.7;  % Upper edge of the stopband (Hz)
N_order = 4;        % Filter Order (4th order Butterworth for efficiency)

% normalizes cutoff frequencies
Wn_stop = [f_stop_low, f_stop_high] / (Fs/2);

[num_4, den_4] = butter(N_order, Wn_stop, 'stop');

% Use filtfilt to apply the filter forward and backward, eliminating phase distortion
y_iir_bandstop_filtered = filtfilt(num_4, den_4, x);

figure;
t = (0:length(x)-1) / Fs;
plot(t, y_iir_bandstop_filtered);
xlim([5, 7]);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('IIR Band-Stop Filter (n=4)');

%% 4.5 Pole/zero plot for IIR band stop filter (n=4)
figure('Position', [100, 100, 600, 600]);

% Get poles and zeros of the filter
poles = roots(den_4);
zeros = roots(num_4);

% Plot unit circle
```

```
theta = linspace(0, 2*pi, 100);
plot(cos(theta), sin(theta), 'k--', 'LineWidth', 1);
hold on;

% Plot real and imaginary axes
plot([-1.2, 1.2], [0, 0], 'k-', 'LineWidth', 0.5);
plot([0, 0], [-1.2, 1.2], 'k-', 'LineWidth', 0.5);

plot(real(poles), imag(poles), 'rx', 'MarkerSize', 8, 'LineWidth', 2);
plot(real(zeros), imag(zeros), 'bo', 'MarkerSize', 6, 'LineWidth', 2);

% Formatting
axis equal;
xlim([-1.2, 1.2]);
ylim([-1.2, 1.2]);
grid on;
xlabel('Real Part');
ylabel('Imaginary Part');
title('Pole-Zero Plot: IIR Band-Stop Filter (n=4)');

% Add legend
legend('', '', '', 'Poles', 'Zeros');

%% 5 FIR Band Stop filter for baseline wander

N_fir = 5940;

% Normalize the cutoff frequencies
```

```
Wn_stop_fir = [f_stop_low, f_stop_high] / (Fs/2);

% FIR band stop filter using the Hamming window.
b_fir_bs = fir1(N_fir, Wn_stop_fir, 'stop', hamming(N_fir + 1));

% Apply Filtering
y_fir_bs = filter(b_fir_bs, 1, x);

% FIR Delay Correction: Group Delay = N / 2 samples
delay_samples_fir = N_fir / 2;

% Trim the delayed output and the original signal to align them
y_fir_bandstop_aligned = y_fir_bs(delay_samples_fir + 1 : end);
x_aligned = x(1 : end - delay_samples_fir);
t_aligned = t(1 : end - delay_samples_fir);



%% 9. Plotting FIR Band-Stop


figure;
plot(t_aligned, x_aligned, 'DisplayName', 'Original Signal');
hold on;
plot(t_aligned, y_fir_bandstop_aligned, 'DisplayName', 'FIR Band Stop (aligned)');

xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('FIR Filter (Aligned)');
legend('show');
```

```
xlim([0 20]);

grid on;


%% 9.1 FIR Band-stop with delay


figure;

plot(t_aligned, x_aligned, 'DisplayName', 'Original Signal');

hold on;

plot(t, y_fir_bs,'DisplayName', 'FIR Band Stop (unaligned)');


xlabel('Time (s)');

ylabel('Amplitude (mV)');

title('FIR Filter (Unaligned)');

legend('show');

xlim([0 20]);

grid on;


%% 10. FFT of Noisy vs Clean vs FIR band stop vs IIR band stop

% FFT of FIR band stop

N_fft = 2^nextpow2(length(x));

Y_fir = fft(y_fir_bs, N_fft);

Ym_fir = abs(Y_fir(1:N_fft/2+1)) / length(y_fir_bs);

Ym_fir(2:end- (mod(N_fft,2)==0)) = 2 * Ym_fir(2:end- (mod(N_fft,2)==0));


% FFT of IIR band stop n=3

Y_bs3 = fft(y_iir_bandstop_filtered_3, N_fft);

Ym_bs3 = abs(Y_bs3(1:N_fft/2+1)) / length(y_iir_bandstop_filtered_3);

Ym_bs3(2:end- (mod(N_fft,2)==0)) = 2 * Ym_bs3(2:end- (mod(N_fft,2)==0));
```

```
% FFT of clean

Y_ref = fft(y_clean, N_fft);

Ym_ref = abs(Y_ref(1:N_fft/2+1)) / length(y_clean);

Ym_ref(2:end- (mod(N_fft,2)==0)) = 2 * Ym_ref(2:end- (mod(N_fft,2)==0));


% FFT of noisy

X = fft(x, N_fft);

Xm = abs(X(1:N_fft/2+1)) / length(x);

Xm(2:end- (mod(N_fft,2)==0)) = 2 * Xm(2:end- (mod(N_fft,2)==0));


% Update y-axis limits based on all signals

combined_signals = [Xm; Ym_ref; Ym_fir; Ym_bs3];

ylim_max = max(combined_signals) * 1.05;


% Frequency vector

N_fft = 2^nextpow2(length(x));

f = (0:(N_fft/2)) * (Fs / N_fft);


subplot(4,1,1);

plot(f, Xm);

ylabel('Magnitude (mV)');

title('FFT of Noisy ECG Signal');

grid on;

xlim([0, 2]);

ylim([0, ylim_max]);


subplot(4,1,2);
```

```
plot(f, Ym_ref);

ylabel('Magnitude (mV)');

title('FFT of Clean ECG Signal');

grid on;

xlim([0, 2]);

ylim([0, ylim_max]);


subplot(4,1,3);

plot(f, Ym_fir);

ylabel('Magnitude (mV)');

title('FFT of FIR Band-Stop Signal (n=5940)');

grid on;

xlim([0, 2]);

ylim([0, ylim_max]);


subplot(4,1,4);

plot(f, Ym_bs3);

xlabel('Frequency (Hz)');

ylabel('Magnitude (mV)');

title('FFT of IIR Band-Stop Signal (n=3)');

grid on;

xlim([0, 2]);

ylim([0, ylim_max]);


%% 10.1 Time domain plots of IIR band stop vs FIR Band-Stop vs. Clean Reference vs. Noisy vs
IIR High pass

y_clean_aligned = y_clean(1 : length(t_aligned));

y_iir_bandstop_aligned_3 = y_iir_bandstop_filtered_3(1 : length(t_aligned));
```

```
y_lim = [-1.5, 1.5];

figure;

subplot(4,1,1);
plot(t_aligned, x_aligned);
title('Original');
xlim([5, 7]);
ylim(y_lim);
ylabel('Amplitude (mV)');
grid on;

subplot(4,1,2);
plot(t_aligned, y_fir_bandstop_aligned);
xlim([5, 7]);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
title('FIR Band Stop');
grid on;

subplot(4,1,3);
plot(t_aligned, y_iir_bandstop_aligned_3);
title('IIR Band Stop (n=3)');
xlim([5, 7]);
ylim(y_lim);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;
```

```
subplot(4,1,4);
plot(t_aligned, y_clean_aligned);
title('Clean Signal');
xlim([5, 7]);
ylim(y_lim);
ylabel('Amplitude (mV)');
grid on;

%% 11. Removing mains hum at 50 Hz

% 50 Hz notch filter
f_notch = 50;
Q = 25;

% bandwidth needed from Q factor
BW = f_notch / Q; % 1.43 Hz

% Cutoff frequencies
f_low = f_notch - (BW / 2);   % 49.285 Hz
f_high = f_notch + (BW / 2);  % 50.715 Hz

% creating filter
N_order_notch = 2;
Wn_notch = [f_low, f_high] / (Fs/2);
[b_n50, a_n50] = butter(N_order_notch, Wn_notch, 'stop');

% phase correction
```

```
y_50hz_removed = filtfilt(b_n50, a_n50, y_iir_bandstop_aligned_3);

% FFT of the 50 Hz Removed Signal
Y_notch = fft(y_50hz_removed, N_fft);
Ym_notch = abs(Y_notch(1:N_fft/2+1)) / length(y_50hz_removed);
Ym_notch(2:end- (mod(N_fft,2)==0)) = 2 * Ym_notch(2:end- (mod(N_fft,2)==0));

% Plotting the 50 Hz Removal Effect (FFT Comparison)
figure;

subplot(2,1,1);
plot(f, Xm);
title('FFT of Noisy ECG Signal');
xlim([0, 120]);
ylim([0, 0.26]);
xlabel('Frequency (Hz)');
ylabel('Magnitude (mV)');
hold on;
grid on;

subplot(2,1,2);
plot(f, Ym_notch);
title('FFT of IIR band stop + notch at 50 Hz');
xlim([0, 120]);
ylim([0, 0.1]);
xlabel('Frequency (Hz)');
ylabel('Amplitude (mV)');
grid on;
```

```
%% 12. Removing harmonic at 100 Hz
f_notch_100 = 100;
Q = 35;

% bandwidth needed from Q factor
BW_100 = f_notch_100 / Q;

% Cutoff frequencies
f_low_100 = f_notch_100 - (BW_100 / 2);
f_high_100 = f_notch_100 + (BW_100 / 2);

% Creating filter
N_order_notch = 2;
Wn_notch_100 = [f_low_100, f_high_100] / (Fs/2);
[b_n100, a_n100] = butter(N_order_notch, Wn_notch_100, 'stop');



y_100hz_removed = filtfilt(b_n100, a_n100, y_50hz_removed);

% FFT of the 100 Hz Removed Signal
Y_notch_100 = fft(y_100hz_removed, N_fft);
Ym_notch_100 = abs(Y_notch_100(1:N_fft/2+1)) / length(y_100hz_removed);
Ym_notch_100(2:end- (mod(N_fft,2)==0)) = 2 * Ym_notch_100(2:end- (mod(N_fft,2)==0));

%% 13. Final low pass filter
fc_lpf = 20;
N_order_lpf = 6;
```

```
Wn_lpf = fc_lpf / (Fs/2);
[b_lpf, a_lpf] = butter(N_order_lpf, Wn_lpf, 'low');

y_final_denoised = filtfilt(b_lpf, a_lpf, y_100hz_removed);

Y_final_fft = fft(y_final_denoised, N_fft);
Y_final = abs(Y_final_fft(1:N_fft/2+1)) / length(y_final_denoised);

Y_final(2:end- (mod(N_fft,2)==0)) = 2 * Y_final(2:end- (mod(N_fft,2)==0));

%% 13.5 Plotting FFTs

figure;
subplot(3,1,1);
plot(f, Ym_notch_100);
title('FFT of IIR bandstop and notches at 50/100 Hz');
xlim([0, 120]);
ylim([0, 0.12]);
xlabel('Frequency (Hz)');
ylabel('Magnitude (mV)');
grid on;

subplot(3,1,2);
plot(f, Y_final);
xlabel('Frequency (Hz)');
ylabel('Magnitude (mV)');
title('FFT of Fully Filtered Signal');
```

```
grid on;
xlim([0, 120]);
ylim([0, 0.12]);

subplot(3,1,3);
plot(f, Ym_ref);
xlabel('Frequency (Hz)');
ylabel('Magnitude (mV)');
title('FFT of Clean ECG Signal');
grid on;
xlim([0, 120]);
ylim([0, 0.12]);


%% 14. Fully filtered signals time domain plot
figure;

subplot(2,1,1);
plot(t, y_clean);
title('Clean Signal');
xlim([0, 10]);
ylim([-1.0, 1.0]);
ylabel('Amplitude (mV)');
grid on;

subplot(2,1,2);
plot(t_aligned(1:length(y_final_denoised)), y_final_denoised);
title('Fully Filtered Signal');
```

```
xlim([0, 10]);
ylim([-1.0, 1.0]);
xlabel('Time (s)');
ylabel('Amplitude (mV)');
grid on;


%% 15. Performace evaluation (RMSE and Percent Error)

N_final = length(y_final_denoised);
N_clean = length(y_clean);

y_clean_for_rmse = y_clean(1:N_final);
rmse = sqrt(mean((y_clean_for_rmse - y_final_denoised).^2));
avg_clean_amplitude = mean(abs(y_clean_for_rmse));

percent_error = (rmse / avg_clean_amplitude) * 100;

fprintf('RMSE: %.4f mV\n', rmse);
fprintf('Percent RMSE Error: %.4f%%\n', percent_error);

initial_noise = x(1:N_clean) - y_clean;
initial_noise_power = mean(initial_noise.^2);

final_noise = y_final_denoised - y_clean_for_rmse;
final_noise_power = mean(final_noise.^2);

SNR_improvement = 10 * log10(initial_noise_power / final_noise_power);
```

```
fprintf('Initial Noise Power: %.6f mV^2\n', initial_noise_power);
fprintf('Final Noise Power: %.6f mV^2\n', final_noise_power);
fprintf('SNR Improvement: %.4f dB\n', SNR_improvement);

%% 15. (AI generated) Wavelet Denoising for White Noise Removal

% 15.1 Signal Preparation - Use ALIGNED signals
% We need to use the aligned signal to match time vectors
y_for_wavelet = y_100hz_removed;  % Use your final denoised signal

% Create corresponding time vector
t_for_wavelet = t_aligned(1:length(y_for_wavelet));

% 15.2 Wavelet Parameters
wname = 'sym8'; % Symlet 8 wavelet (good for ECG)
level = 5;      % Decomposition level

% 15.3 Decomposition (Discrete Wavelet Transform - DWT)
[C, L] = wavedec(y_for_wavelet, level, wname);

% 15.4 Threshold Calculation (Universal Threshold/VisuShrink)
% Estimate noise from finest detail coefficients
detail_coeffs = detcoef(C, L, 1);  % Get level 1 detail coefficients
sigma = median(abs(detail_coeffs)) / 0.6745;
lambda = sigma * sqrt(2 * log(length(y_for_wavelet)));
```

```
% 15.5 Apply soft thresholding to DETAIL coefficients only
% (preserve approximation coefficients)
C_denoised = C;  % Start with all coefficients

% Apply threshold to each detail level
for i = 1:level
    % Get detail coefficients at level i
    coeffs = detcoef(C, L, i);

    % Apply soft thresholding with scaled threshold
    coeffs_thresh = wthresh(coeffs, 's', lambda/(2^(i/2)));

    % Put back into coefficient vector
    start_idx = sum(L(1:end-i)) + 1;
    end_idx = sum(L(1:end-i)) + L(end-i);
    C_denoised(start_idx:end_idx) = coeffs_thresh;
end

% 15.6 Reconstruction (Inverse DWT)
y_wavelet_denoised = waverec(C_denoised, L, wname);

% Ensure same length
if length(y_wavelet_denoised) > length(y_for_wavelet)
    y_wavelet_denoised = y_wavelet_denoised(1:length(y_for_wavelet));
elseif length(y_wavelet_denoised) < length(y_for_wavelet)
    y_wavelet_denoised(end+1:length(y_for_wavelet)) = 0;
end
```

```
%% (AI generated) 15.7 Time Domain Comparison
figure();

% Before wavelet
subplot(2,1,1);
plot(t_for_wavelet, y_for_wavelet);
title('Before Wavelet Denoising');
xlim([5, 10]);  % Zoom on a few beats
ylabel('Amplitude (mV)');
grid on;

% After wavelet
subplot(2,1,2);
plot(t_for_wavelet, y_wavelet_denoised);
title(['After Wavelet Denoising (Symlet 8, Level ' num2str(level) ')']);
xlim([5, 10]);  % Same zoom
ylabel('Amplitude (mV)');
grid on;
```