Electrical & Electronics Engineering Department
EE434 – Biomedical Signal Processing - Midterm Takehome Exam
Deadline: December 7th, 2023 23:59
Instructor: M. Zübeyir Ünlü

Name: Harun Durmuş Student No: 270206025

Honor statement: *I pledge that I have not used any notes, text, or any other reference materials during this exam. I pledge that I have neither given nor received any aid from any other person during this examination, and that the work presented here is entirely my own.*

Signature:

[illegible]

Q1. Apply a 10-point moving average filter to ecg_2.mat.

a) Plot the filtered signal and the original signal on top of each other on the same plot (with a different color). Comment!

b) Plot the filtered signal and the original signal frequency spectrums on top of each other on the same plot (with a different color). Compare them!

ANS1.

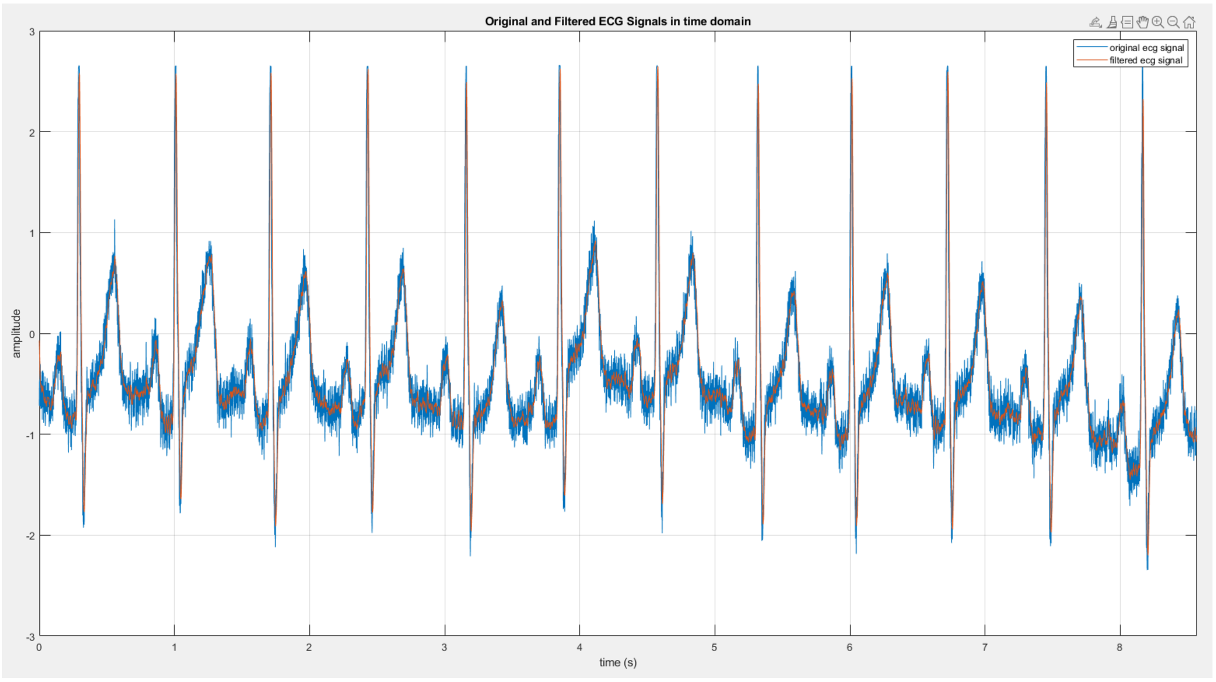

Figure 1: Original and Filtered ECG Signals in time domain

a) The first figure shows the original ECG signal and the filtered ECG signal overlaid on the same plot. This visualization allows for a direct comparison of the signals before and after filtering.

According to the provided code[1] that is attached in appendices, the moving average filter, being a low-pass filter, smoothens the high frequency noised ECG signal. This smoothing is expected to reduce high-frequency noise and minor fluctuations in the signal. As a result, the filtered signal might lose some of its sharp features (like the sharp peaks of the QRS complex), but it will better represent the overall trend of the heart rhythm. The effect of the filter is more pronounced in areas with rapid changes in the signal amplitude.

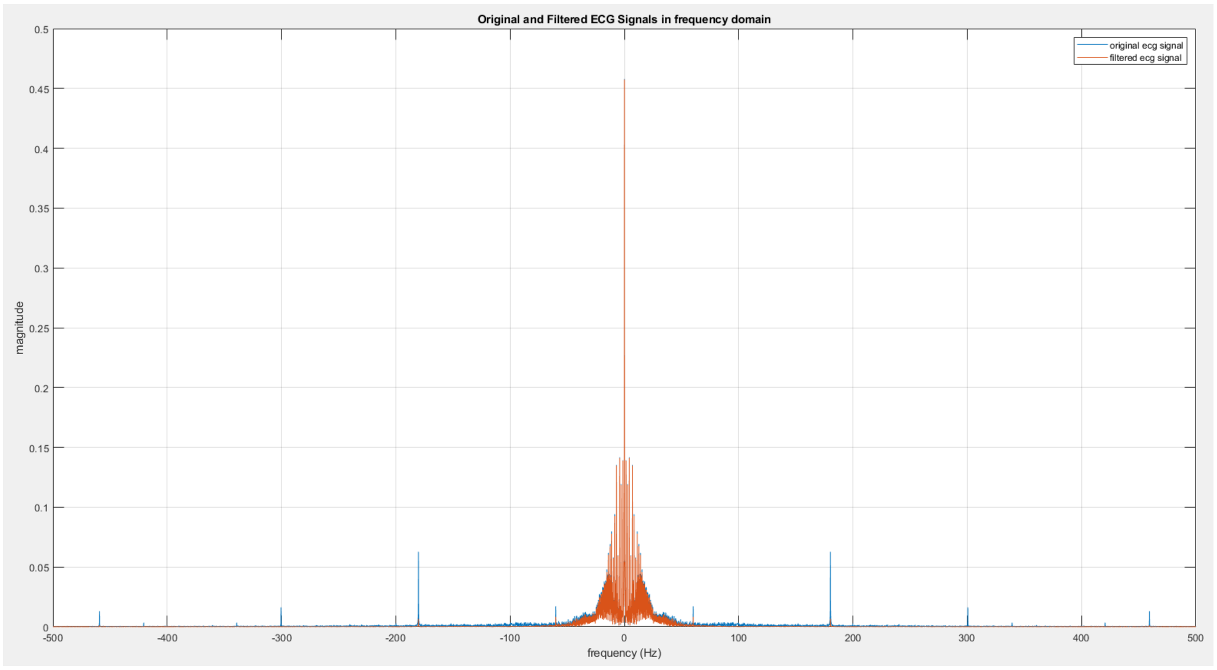

Figure 2: Original and Filtered ECG Signals in frequency domain

b) The second figure compares the frequency spectrums of the original and filtered ECG signals. The spectrums are obtained using the Fast Fourier Transform (FFT), which transforms the signals from the time domain to the frequency domain.

According to the provided code[1], in the frequency domain, the impact of the moving average filter can be observed as a reduction in the magnitude of the higher frequency components. The filtered signal's spectrum will have diminished peaks at higher frequencies, indicating the attenuation of those components due to the low-pass characteristics of the moving average filter. This is consistent with the filter's purpose of reducing high-frequency noise. However, the essential components of the ECG signal, which are typically in the lower frequency range, remain largely preserved. It can be observed from the zoomed verison of the figure 2.

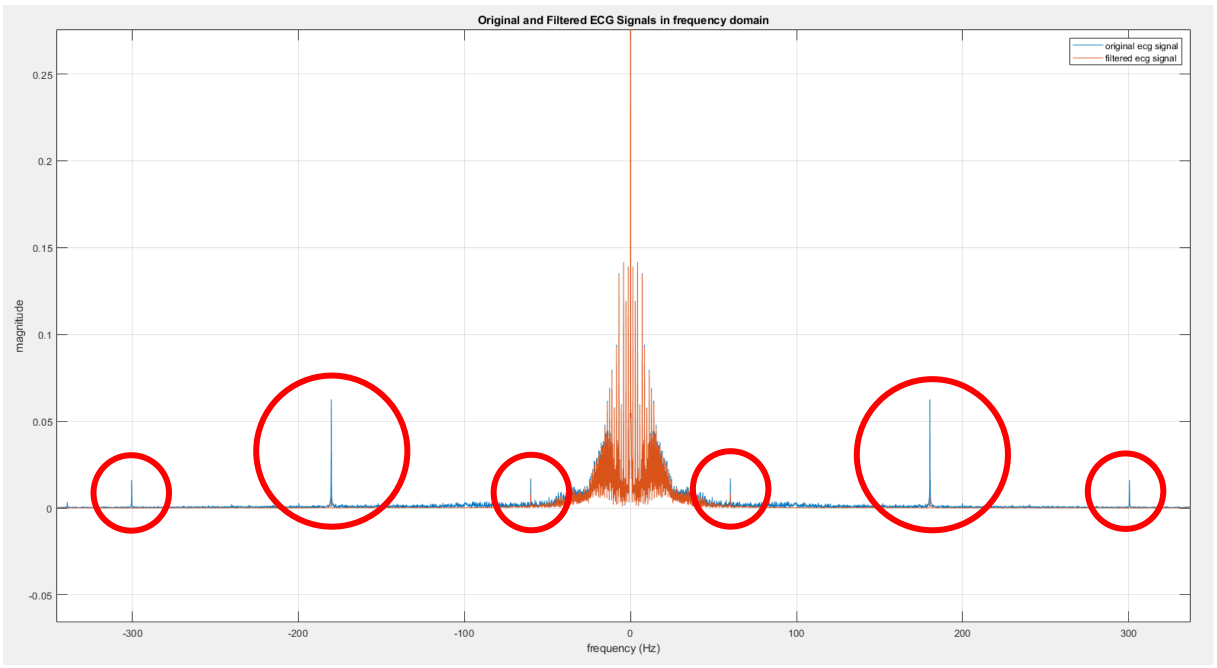

Figure 3: Zoomed Version of the Figure 2

As it seems from the figure above, the unwanted frequency components are removed with the filter. These noise components can occur due to 50-60 Hz and 220 V electric network or other organs and muscles.

In overall conclusion, in the time domain, the filtering process results in a smoother signal, potentially making it easier to identify key features like the R peaks in a noisy ECG signal. In the frequency domain, the comparison highlights the filter's effectiveness in attenuating high-frequency noise while retaining the primary components of the ECG signal.

This analysis is critical in clinical settings and biomedical signal processing, where extracting clear, noise-free signals is essential for accurate diagnosis and monitoring. The moving average filter serves as a simple yet effective tool for achieving this, particularly in scenarios where high-frequency noise is a concern.

Q2. In this question, we will design digital IIR filters using Butterworth and Chebychev filter approximations. Which type of Chebychev filter (type I or II) should you prefer? Why? Compare the filter characteristics to those of FIR filters. What differences are most prominent?
a) Apply each of the above designed filters to the ECG signal you downloaded. Is the signal denoised? Play around with the filter specs, and find out whether other specs may give you better denoised signal.
b) Plot the filtered signal and the original signal on top of each other on the same plot (with a different color). Comment!
c) Plot the filtered signal and the original signal frequency spectrums on top of each other on the same plot (with a different color). Compare them!

ANS2. The MATLAB code[2] provided illustrates the design and application of both Butterworth and Chebyshev Type I filters to an ECG signal, and it presents the comparison of these filters in both the time and frequency domains.

a) Choosing Filter Parameters

- **Butterworth Filter:** The Butterworth filter is designed with a low-pass characteristic and a specific cutoff frequency. The key advantage of this filter is its maximally flat frequency response in the passband, meaning no ripples, which results in a smoother signal without distortions in the passband.
- **Chebyshev (Type I) Filter:** The Chebyshev Type I filter is designed with a certain amount of ripple in the passband. It offers a sharper cutoff compared to the Butterworth filter, which can be beneficial in applications requiring a more stringent attenuation of frequencies outside the passband.

When choosing between these filters, one must consider the trade-offs: the Butterworth filter for a smoother passband response and the Chebyshev filter for a sharper cutoff but with some passband ripple.

Several filter parameter values are tried until the best response is obtained. Here are some outputs with different adjustment of the filter parameters:

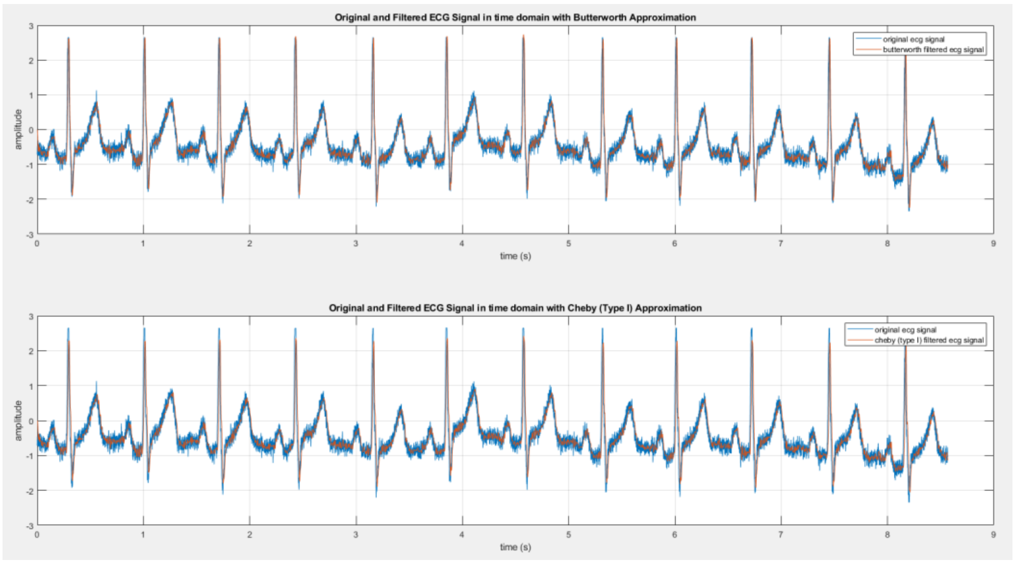

Figure 4: Filter with cut-off frequency is 100 Hz

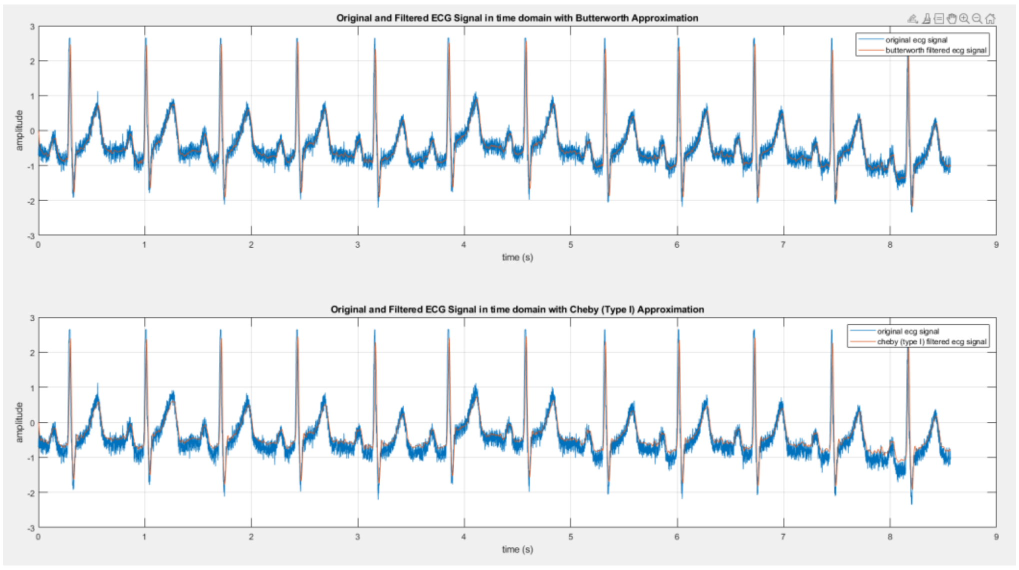

Figure 5: Filter with order is 2

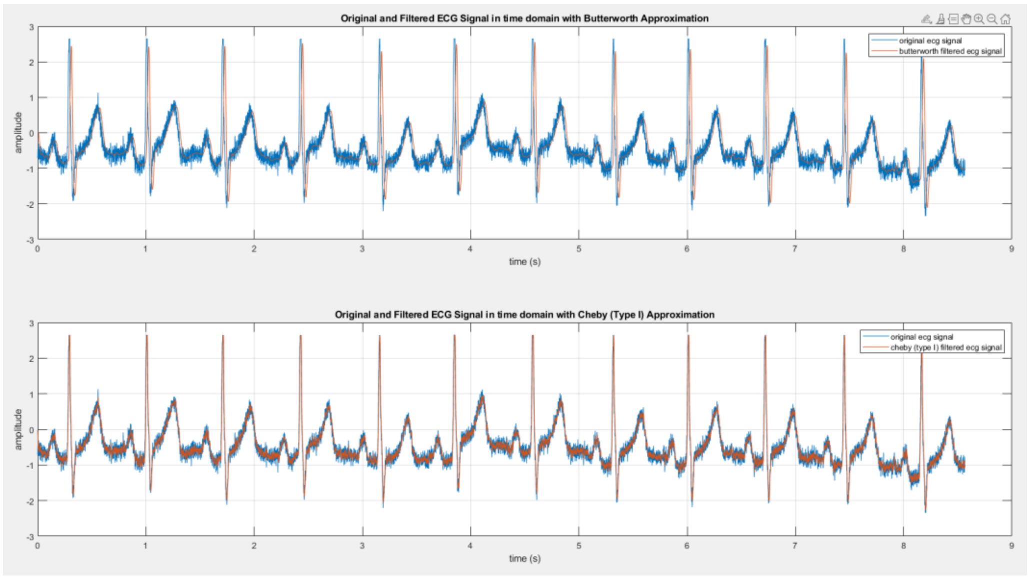

Figure 6: Filter with passband ripple is 0

By analyzing the figures above, it is obtained that the optimum response should indicate P-Q-R-S-T intervals clearly without any information loss or unwanted noise. So, after several adjustment the optimum condition for filter parameters are founded like in the code[2]:

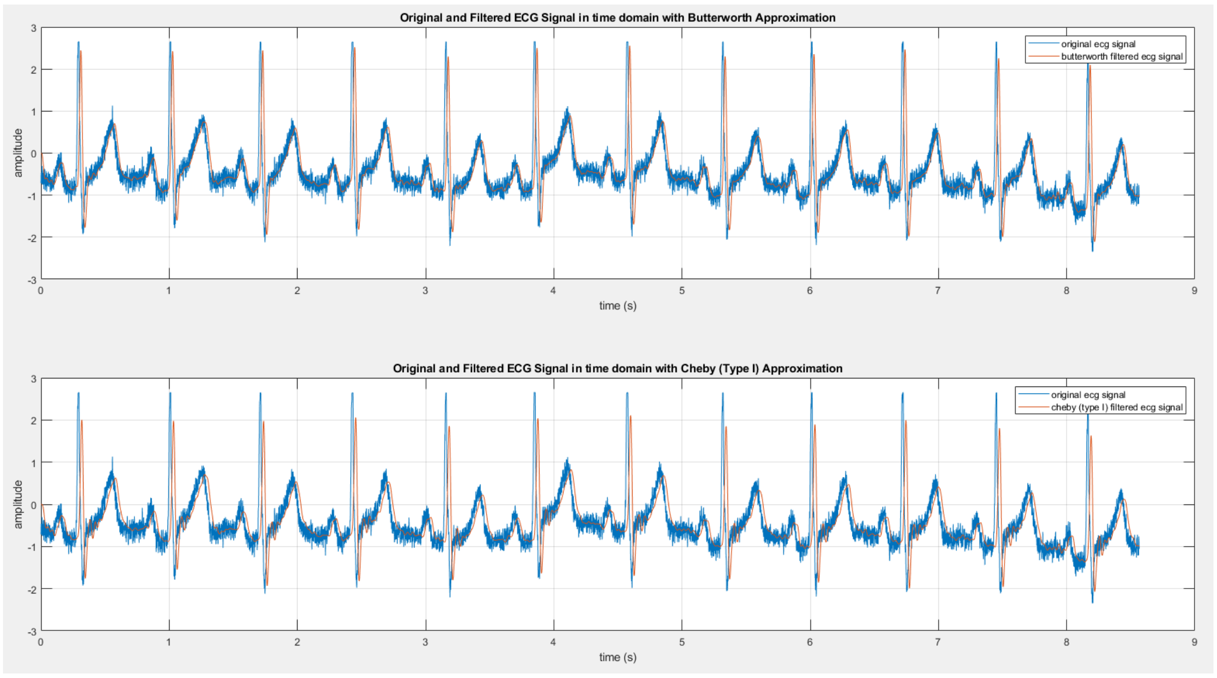

Figure 7: Output with optimum filter parameters

b) In the time domain, the Butterworth-filtered signal likely shows a more gradual transition and less distortion, maintaining the original shape of the ECG signal while effectively reducing high-frequency noise. The Chebyshev-filtered signal might exhibit slight distortions due to passband ripples but would more aggressively attenuate unwanted frequencies.

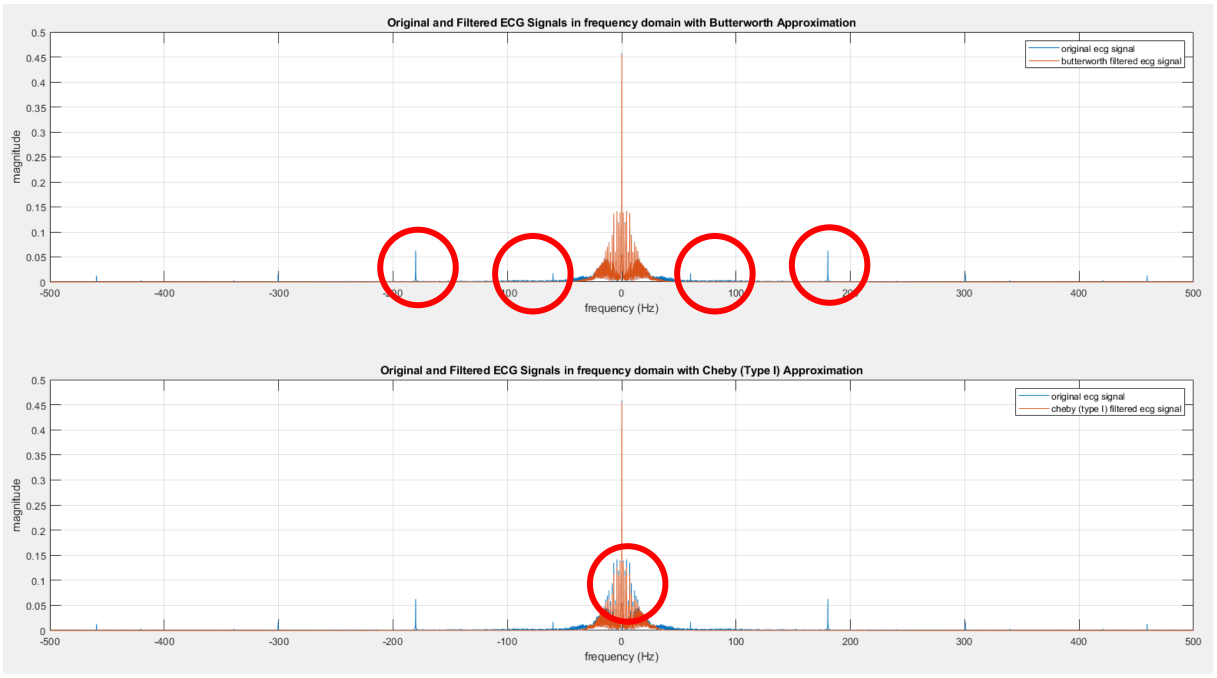

Figure 8: Frequency Spectrum of Output Signal

c) It is also observed that there is a small amount of magnitude decrease in lower frequencies when Chebychev Filter is used. That reminds the trade-off which is explained earlier. The frequency response of the Butterworth filter demonstrates its flat passband up to the cutoff frequency and a gradual roll-off, which is evident in the filtered signal's spectrum. The Chebyshev filter shows a steeper roll-off past the cutoff frequency, but ripples in the passband can be observed. This characteristic is advantageous for more precise frequency cutoff but at the expense of some passband fidelity.

In overall conclusion, both filters effectively denoise the ECG signal by attenuating high-frequency components. The choice between them depends on the specific requirements for the ECG analysis – whether a smoother passband (Butterworth) or a sharper cutoff (Chebyshev) is more critical.

FIR filters typically offer a linear phase response, which is beneficial for preserving the waveform shape. In contrast, IIR filters like Butterworth and Chebyshev can introduce phase distortions but require a lower filter order to achieve a similar magnitude response.
Experimenting with different filter specifications (like cutoff frequency and filter order) can further optimize signal denoising. The optimal settings depend on the characteristics of the noise and the ECG signal itself.

This analysis demonstrates the importance of filter selection in signal processing tasks, where each filter type brings its own advantages and trade-offs. The choice should be guided by the specific requirements of the ECG signal analysis and the nature of the noise present in the signal.

Q3. Short-Time Fourier Transform Analysis
a) Using a rectangular window with a length of 256 samples and length of overlap as 128 samples: Plot the STFT (spectrogram).
b) Using a hamming window with a length of 256 samples and length of overlap as 128 samples: Plot the STFT (spectrogram).
c) Compare the spectrograms you obtained in parts (a) and (b).
d) If you make a comparison between the spectrograms you obtained in Question #3 and the spectrums in previous questions and in Homework #1, what are the advantages of spectrograms and the disadvantages of spectrums?

ANS3. The Short-Time Fourier Transform (STFT) is a fundamental tool in signal processing that extends the concept of the Fourier Transform to non-stationary, time-varying signals. It works by dividing a longer time signal into shorter segments of equal length and then computing the Fourier Transform separately on each of these segments. This division is typically achieved using a sliding window function that moves across the signal. By applying the Fourier Transform to these windowed segments, the STFT captures not only the frequency content of the signal but also how this content changes over time. The result is a two-dimensional representation of the signal, showing both time and frequency information. This makes the STFT particularly useful for analyzing signals whose frequency characteristics vary over time, such as speech or music, and for applications like noise reduction and signal compression. The trade-off in the STFT is between the time and frequency resolution, determined by the size of the window: larger windows provide better frequency resolution but poorer time resolution, and vice versa.

The provided MATLAB code[3] demonstrates the use of the Short-Time Fourier Transform (STFT) for analyzing three different ECG signals (labeled as ecg_lfn, ecg_hfn, and ecg_noisy) using both rectangular and Hamming windows. This analysis offers insights into how windowing affects the time-frequency representation of signals.

a) Using Rectangular Window

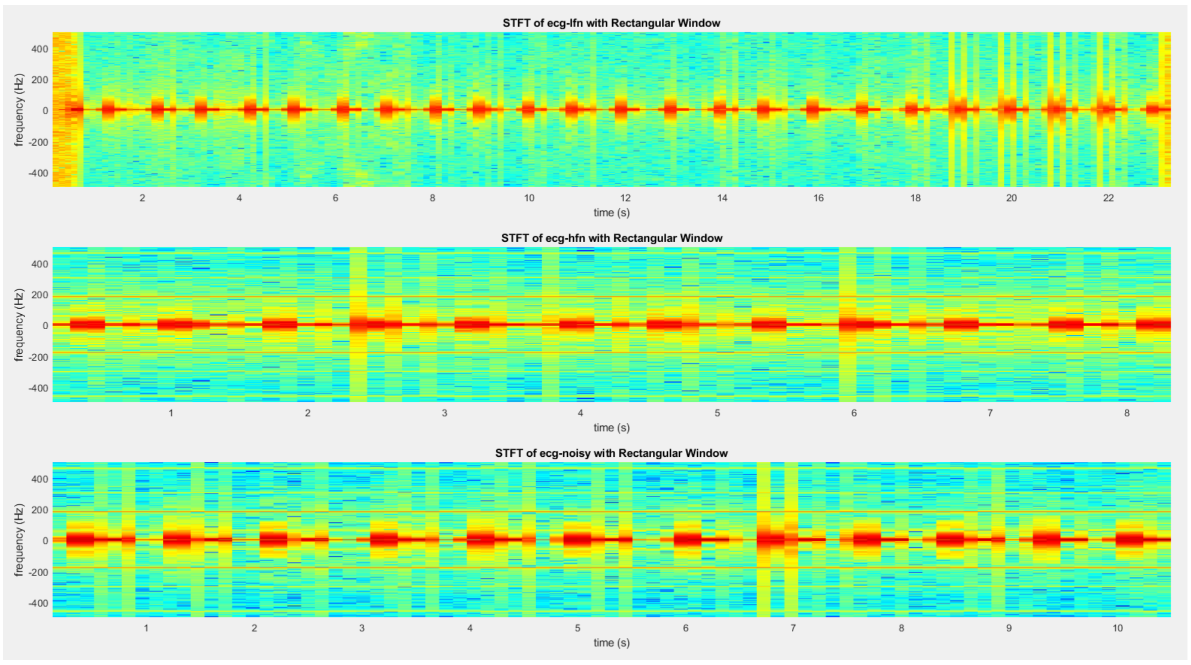

Figure 9: STFT with rectangular window of the given ECG signals

- The STFT of each ECG signal is computed using a rectangular window of 256 samples with an overlap of 128 samples.
- The spectrograms are plotted in dB scale, providing a clear visualization of the frequency components over time.
- The rectangular window tends to produce a spectrogram with sharper frequency resolution but can introduce discontinuities at the window edges, possibly leading to spectral leakage.

b) Using Hamming Window

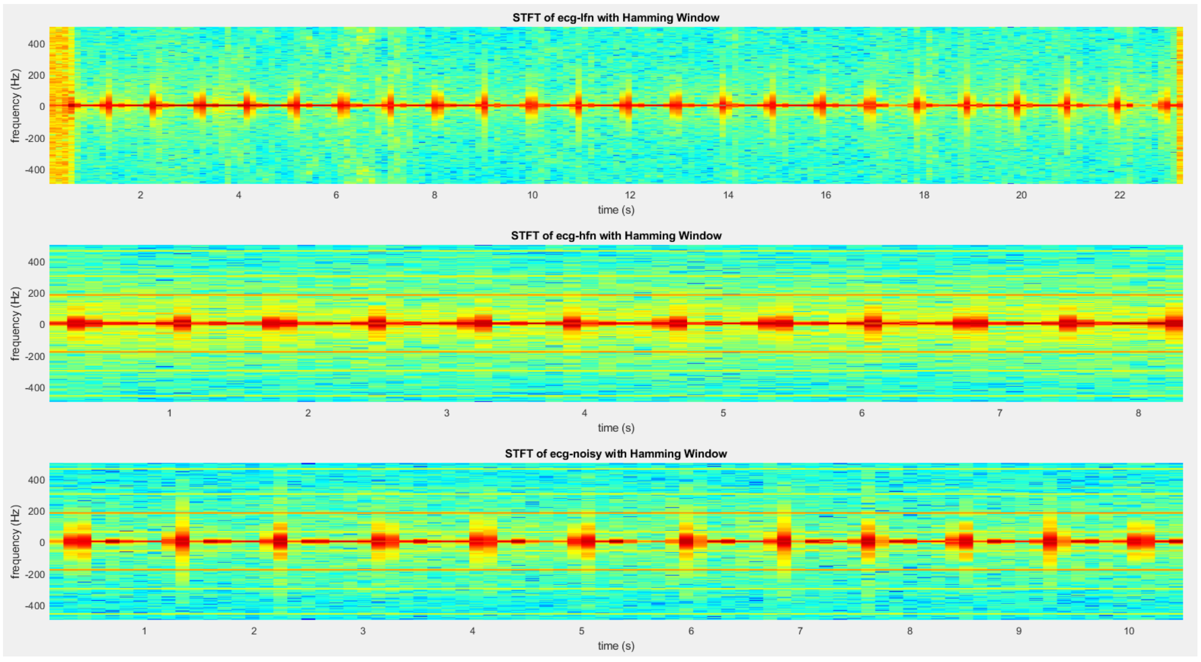

Figure 10: STFT with Hamming window of the given ECG signals

- The same ECG signals are analyzed using a Hamming window, known for its smoother characteristics compared to the rectangular window.
- The Hamming window reduces spectral leakage due to its tapered shape, providing a smoother spectral representation.
- This windowing technique might offer a more accurate depiction of the energy distribution across frequencies, especially in noisy signals.

c) Comparison between Rectangular and Hamming Window Spectrograms

- **Rectangular Window:** Offers higher spectral resolution but may include more spectral leakage, which can be misleading in energy distribution across frequencies.
- **Hamming Window:** Provides a smoother spectral representation, reducing leakage, and potentially giving a more accurate depiction of the signal's frequency content over time.

d) Disadvantages of Spectrums

- **Lack of Time Information:** Traditional spectrums fail to capture the time-varying nature of signals, making them less suitable for analyzing signals whose spectral content changes over time.
- **Averaging Effect:** Spectrums provide an averaged view over the entire signal duration, potentially masking transient or evolving features of the signal.

In summary, the choice of window in STFT analysis significantly impacts the spectral representation, and the use of spectrograms offers a comprehensive view of both time and frequency characteristics of a signal, which is essential for detailed analysis in applications like ECG signal processing.

Q4. Estimating Power Spectral Density (PSD) of the ECG signals
a) Calculate and plot the PSD of the above data using periodogram method.
b) Calculate and plot the PSD of the above data using modified periodogram method.
c) Calculate and plot the PSD of the above data using Welch’s method.
d) Calculate and plot the PSD of the above data using Levinson-Durbin Algorithm. Compare all of the above results among themselves and also with the results obtained before.

ANS4. The MATLAB code[4] provided demonstrates the application of different methods for estimating the Power Spectral Density (PSD) of ECG signals, including the periodogram, modified periodogram, Welch’s method, and the Levinson-Durbin algorithm. Each method offers a unique perspective on the spectral content of the ECG signals.

a) Periodogram method provides a basic estimation of the PSD using a rectangular window. This method is straightforward but can suffer from variance issues and spectral leakage. The periodogram plots for ecg_lfn, ecg_hfn, and ecg_noisy reveal their frequency content but might not accurately represent power distribution due to the limitations of the method.

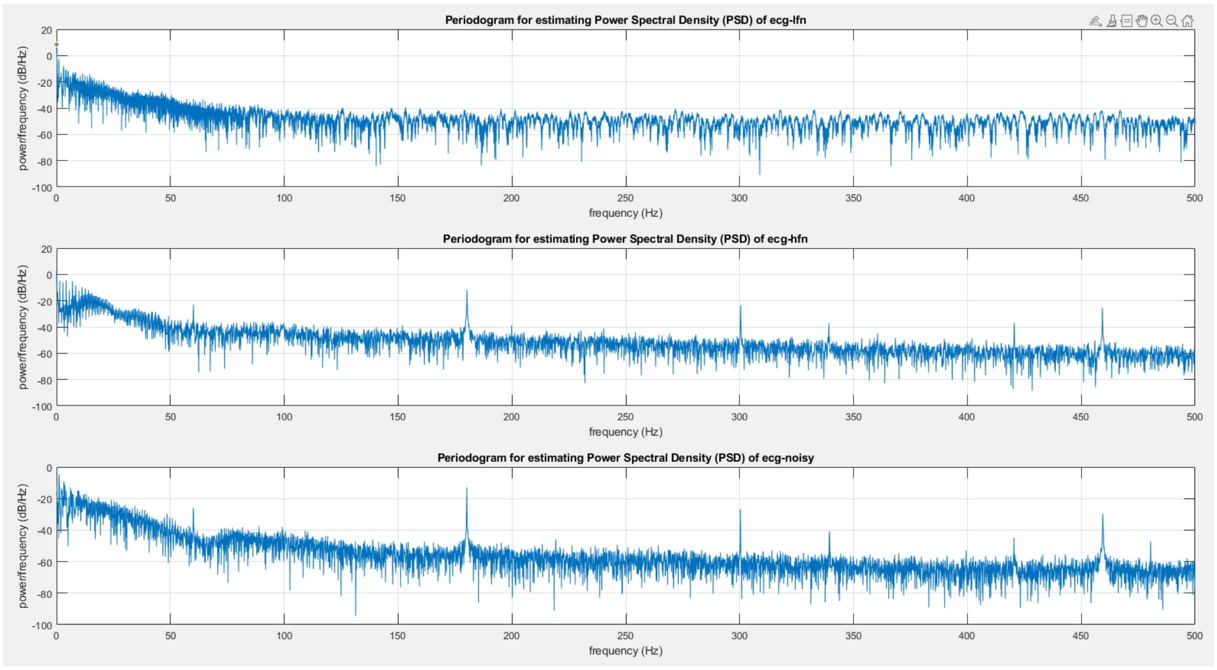

Figure 11: Periodogram method for estimating power spectral density of each signals

b) Modified periodogram methos uses a Hamming window to reduce spectral leakage compared to the standard periodogram. This approach offers a smoother PSD estimate, potentially providing a more accurate representation of the power distribution in the ECG signals.

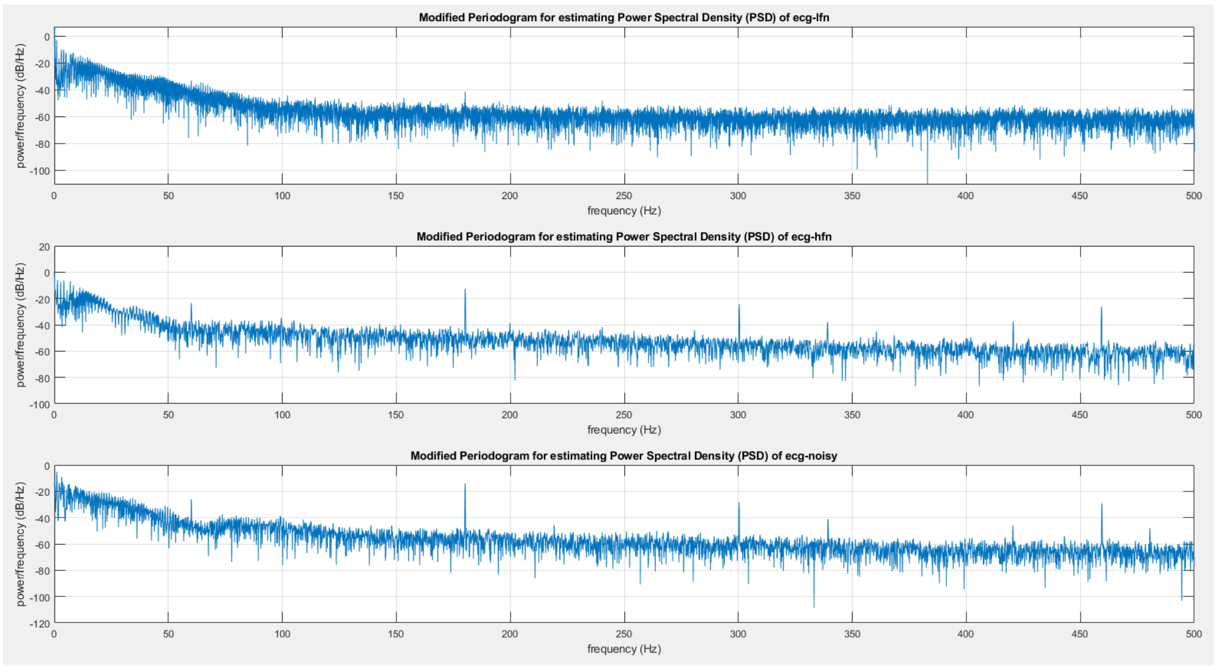

Figure 12: Modified periodogram for estimating power spectral density of each signals

c) Welch's method is an improvement over the basic periodogram, Welch's method segments the signal, applies windowing, and averages the periodograms of these segments. This method reduces variance in the PSD estimate, resulting in a more consistent and reliable representation of the ECG signals' spectral content.

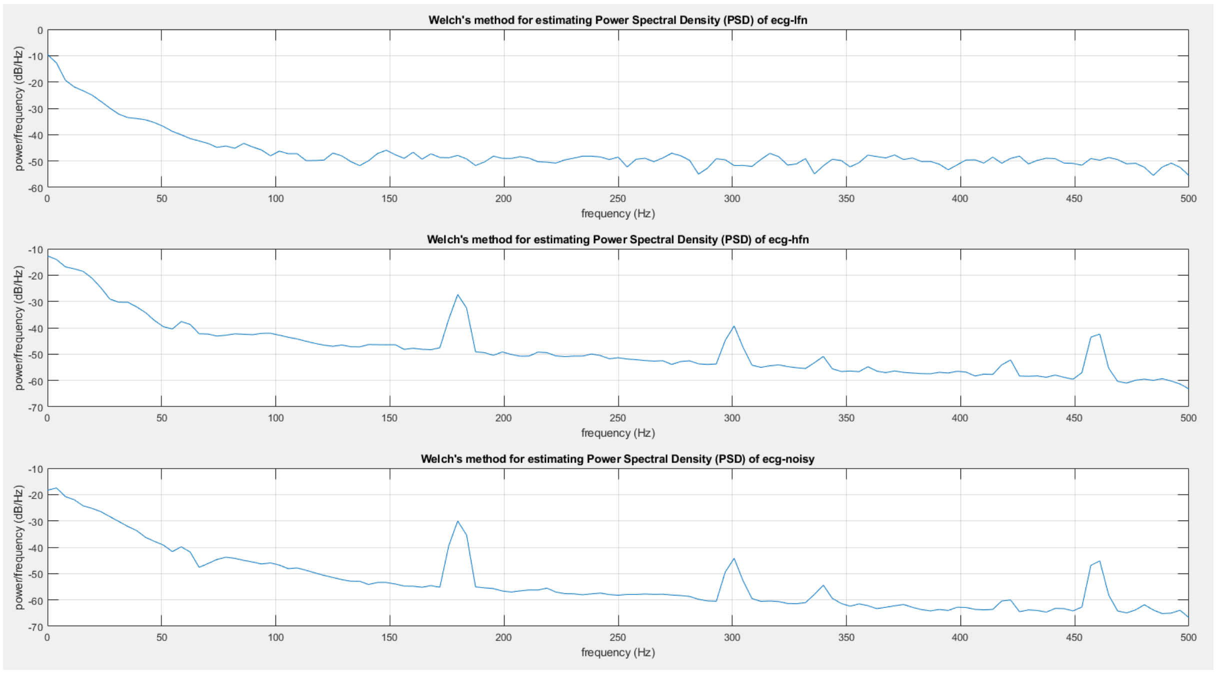

Figure 13: Welch's method for estimating power spectral density

d) Levinson-Durbin algorithm is an autoregressive method that models the signal as a system output driven by white noise. The order of the model, chosen separately for each signal (order_lfn, order_hfn, order_noisy), significantly impacts the PSD estimate. It Offers a parametric approach to PSD estimation, which can be particularly effective when the signal has a clear rhythmic component like an ECG.

Choosing Parameters for Levinson-Durbin Algorithm

- **Model Order:** The choice of the order is crucial. A higher order may capture more signal details but risks overfitting and increasing computational complexity. A lower order may miss important signal characteristics. Model order can be selected based on criteria like Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), or experimentally by observing the fit quality.
- **Signal Characteristics:** Considering the nature of the signal, ECG signals with more complex patterns might require a higher order for accurate modeling.

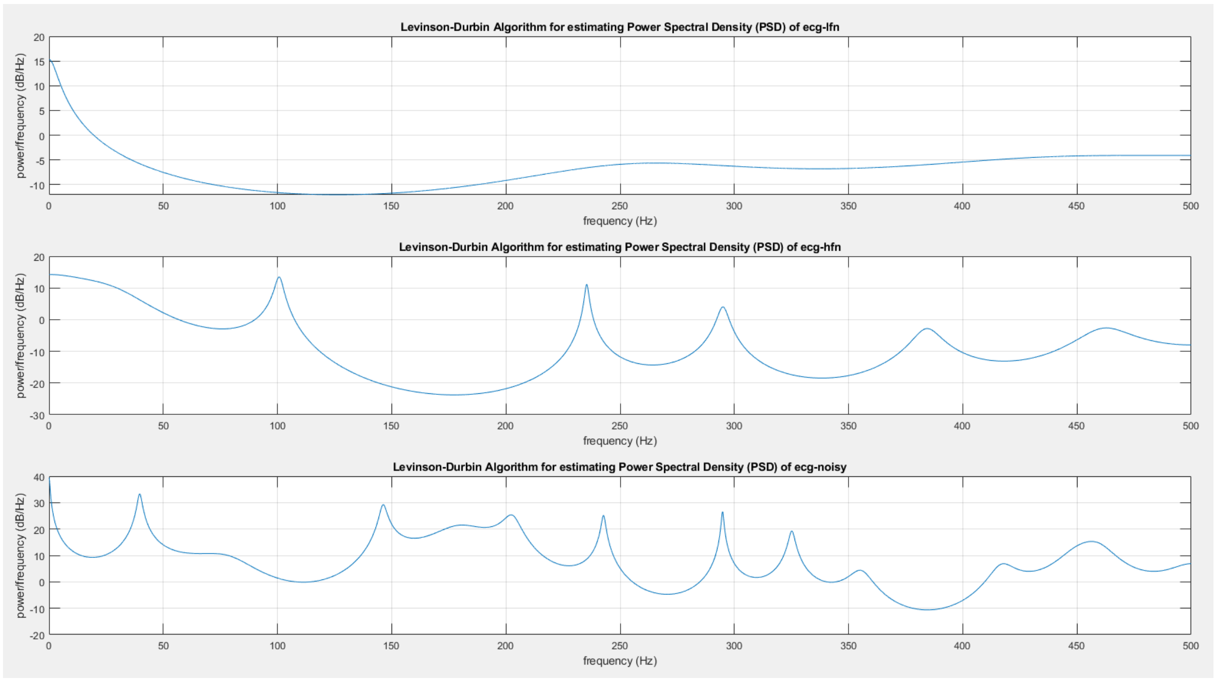

Figure 14: Levinson-Durbin Algorithm for estimating power spectral density of each signal

In overall conclusion, each method has its strengths and limitations. The periodogram is simple but prone to variance, the modified periodogram offers smoother estimates, Welch's method provides more reliable estimates for noisy signals, and the Levinson-Durbin algorithm is effective for signals with clear patterns or rhythms. Spectrogram vs. Spectrum provide a more detailed view of the spectral content compared to a basic Fourier Transform or spectrogram, which is crucial for analyzing the frequency characteristics of ECG signals in detail.

The choice of method depends on the specific requirements of the signal analysis, the nature of the ECG data, and the level of detail required in the spectral representation.

Q5. Please implement the same "QRS Detection Algorithm" on the given ecg signal.
ANS5. The MATLAB code[5] provided implements a QRS detection algorithm on an ECG signal (ecg_hfn). This algorithm is a multi-step process involving filtering, differentiation, squaring, and moving average filtering to highlight and detect the QRS complexes characteristic of ECG signals.

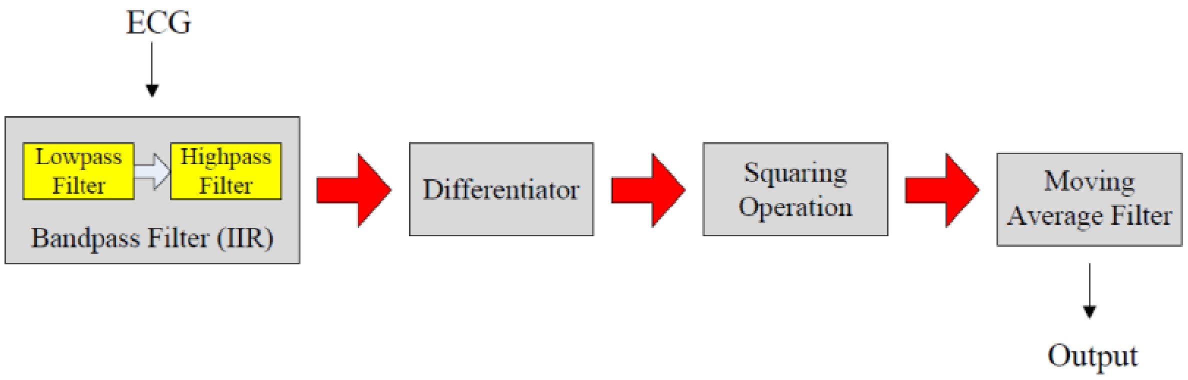

Figure 15: The block diagram scheme of the process to be applied

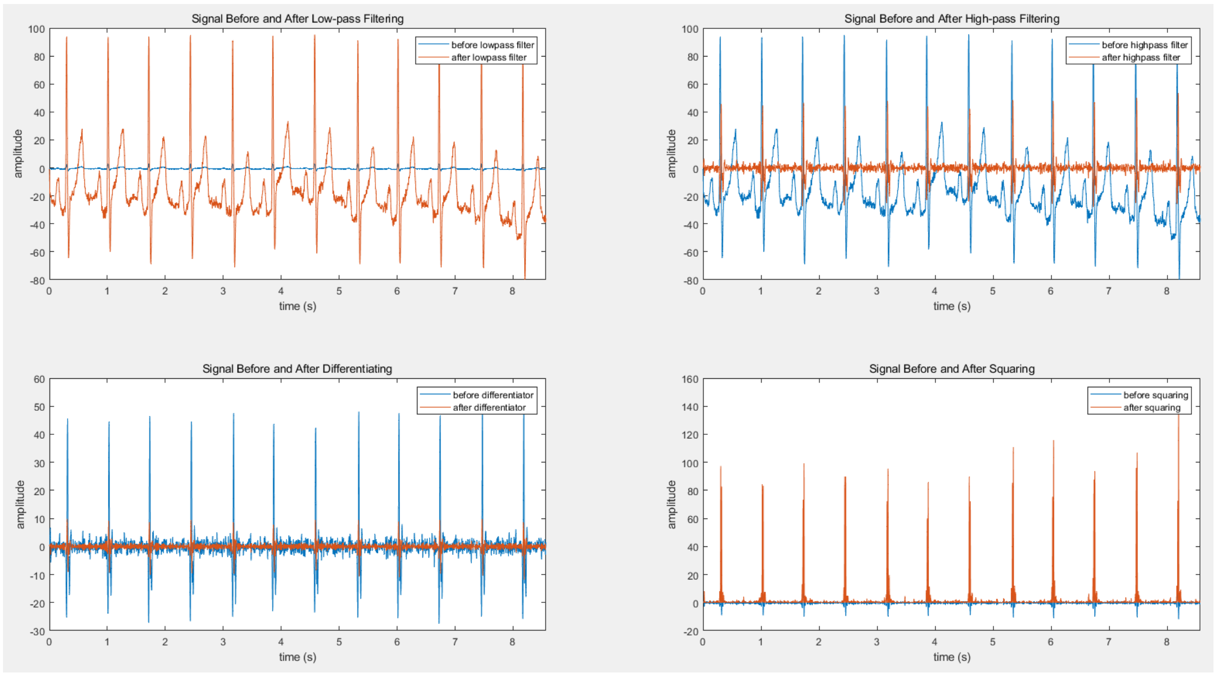

Figure 16: Results of each steps that are defined in block diagram

Low-pass filter is designed to remove high-frequency noise. The nominator coefficients (nom_low) are set to allow frequencies below a certain cutoff while attenuating higher frequencies. The denominator coefficients (denom_low) determine the filter's stability and frequency response. High-pass filter is aimed at removing low-frequency trends from the signal, like baseline wander. The coefficients (nom_high and denom_high) are chosen to preserve frequencies above a certain threshold. Differentiator is used to emphasize the high-frequency components of the QRS complex. The differentiator's coefficients (nom_diff and denom_diff) are set to calculate a finite difference approximation, enhancing the slope of the QRS complex. Squaring the signal after differentiation serves to make all values positive and emphasize larger values, which are likely to be the QRS complexes.

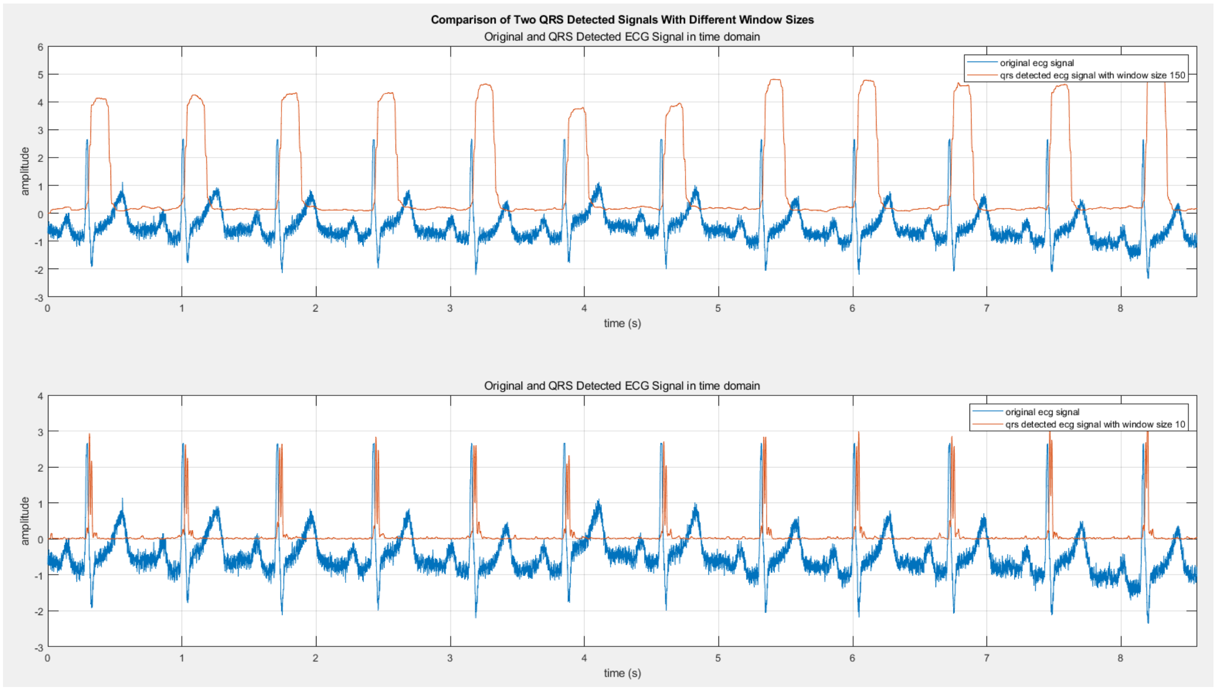

Figure 17: Results after moving average filter

Finally, averaging over a window smooths the signal and helps in identifying the regions where the QRS complexes are located. Two window sizes are used (size_window_1 and size_window_2), corresponding to different durations (150 ms and as defined in question 1). This step helps in delineating the QRS complexes more clearly.

In overall conclusion, the implementation of the QRS detection algorithm in MATLAB demonstrates the effectiveness of signal processing techniques in biomedical applications. The choice of filter parameters and window sizes is crucial in accurately detecting QRS complexes. The two different window sizes in the moving average filter provide an interesting comparison in how the duration of averaging affects QRS detection. This process is essential in automated ECG analysis systems, where accurate QRS detection is key to identifying heart rate, rhythm, and other cardiac abnormalities.

Appendices

[1] MATLAB Code "Q1_harun_durmus.m":

```
%% Comments are added to the report (pdf file)
%%
clc;
clear all;
close all;
%% uploading images
%%
load('ecg_2.mat'); % ecg_hfn is from ecg_2.mat file
%% Question 1. Applying a 10-point moving average filter
%% step a) time domain signals
%%
size_window = 10; % Defined window size of the filter
nom = ones(1,size_window); % nominator coefficient of the filter
denom = size_window; % denominator coefficient of the filter
ecg_hfn_filtered = filter(nom, denom, ecg_hfn);
N_hfn = length(ecg_hfn);
Fs = 1000; % sample frequency
Ts = 1/Fs;
t = 0:Ts:(N_hfn-1)*Ts;

figure (1)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_hfn_filtered);
grid on;
xlabel("time (s)");
ylabel("amplitude");
title('Original and Filtered ECG Signals in time domain');
legend("original ecg signal","filtered ecg signal");
xlim([0 N_hfn*Ts]);
%% step b) frequency domain signals
%%
ecg_hfn_freq = abs(fftshift(fft(ecg_hfn,N_hfn)))/N_hfn;
ecg_hfn_filtered_freq = abs(fftshift(fft(ecg_hfn_filtered,N_hfn)))/N_hfn;
f = linspace(-Fs/2,Fs/2,N_hfn);

figure (2)
plot(f, ecg_hfn_freq);
hold on;
plot(f, ecg_hfn_filtered_freq);
grid on;
xlabel("frequency (Hz)");
```

```
ylabel("magnitude");
title('Original and Filtered ECG Signals in frequency domain');
legend("original ecg signal","filtered ecg signal");
```

[2] MATLAB Code “Q2_harun_durmus.m”:

```
%% Comments are added to the report (pdf file)
%%
clc;
clear all;
close all;
%% uploading images
%%
load('ecg_2.mat'); % ecg_hfn is from ecg_2.mat file
%% Question 2. Designing digital IIR filters using Butterworth and Chebychev
filter approximations
%% step a) Butterworth and Chebyshev (Type I) filter parameters
%%
Fs = 1000; % sample frequency
fc = 30; % cutoff frequency
filter_order = 5; % choosing filter order
fc_normalized = fc / (Fs/2); % normalized cutoff frequency
pb_ripple = 2; % ripple in passband
[nom_butter, denom_butter] = butter(filter_order, fc_normalized, 'low'); %
designing butterworth filter
ecg_filtered_butter = filter(nom_butter, denom_butter, ecg_hfn);
[nom_cheby1, denom_cheby1] = cheby1(filter_order, pb_ripple, fc_normalized,
'low'); % designing chebyshev type I filter
ecg_filtered_cheby1 = filter(nom_cheby1, denom_cheby1, ecg_hfn);
%% step a) time domain signals
%%
N_hfn = length(ecg_hfn);
Ts = 1/Fs;
t = 0:Ts:(N_hfn-1)*Ts;
figure (1)
subplot(211)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_filtered_butter);
grid on;
xlabel("time (s)");
ylabel("amplitude");
title('Original and Filtered ECG Signal in time domain with Butterworth
Approximation');
legend("original ecg signal","butterworth filtered ecg signal");
subplot(212)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_filtered_cheby1);
grid on;
xlabel("time (s)");
ylabel("amplitude");
title('Original and Filtered ECG Signal in time domain with Cheby (Type I)
Approximation');
legend("original ecg signal","cheby (type I) filtered ecg signal");
%% step b) frequency domain signals
%%
ecg_hfn_freq = abs(fftshift(fft(ecg_hfn,N_hfn)))/N_hfn;
ecg_butter_filtered_freq = abs(fftshift(fft(ecg_filtered_butter,N_hfn)))/N_hfn;
```

```
ecg_cheby1_filtered_freq = abs(fftshift(fft(ecg_filtered_cheby1,N_hfn)))/N_hfn;
f = linspace(-Fs/2,Fs/2,N_hfn);
figure (2)
subplot(211)
plot(f, ecg_hfn_freq);
hold on;
plot(f, ecg_butter_filtered_freq);
grid on;
xlabel("frequency (Hz)");
ylabel("magnitude");
title('Original and Filtered ECG Signals in frequency domain with Butterworth
Approximation');
legend("original ecg signal","butterworth filtered ecg signal");
subplot(212)
plot(f, ecg_hfn_freq);
hold on;
plot(f, ecg_cheby1_filtered_freq);
grid on;
xlabel("frequency (Hz)");
ylabel("magnitude");
title('Original and Filtered ECG Signals in frequency domain with Cheby (Type I)
Approximation');
legend("original ecg signal","cheby (type I) filtered ecg signal");
```

[3] MATLAB Code “Q3_harun_durmus.m”:

```
%% Comments are added to the report (pdf file)
%%
clc;
clear all;
close all;
%% uploading images
load('ecg_1.mat'); % ecg_lfn    is from ecg_1.mat file
load('ecg_2.mat'); % ecg_hfn    is from ecg_2.mat file
load('ecg_3.mat'); % ecg_noisy  is from ecg_3.mat file
%% Question 3. About time-and-frequency analysis, Short-time Fourier Transform
(STFT) and plotting the spectrogram.
%% step a) Using rectangular window
%%
n_window = 256;     % window length
n_overlap = 128;    % overlap length
Fs = 1000;          % sample frequency
[rect_lfn, f_lfn, t_lfn] = stft(ecg_lfn, Fs, 'Window', rectwin(n_window),
'OverlapLength', n_overlap);
[rect_hfn, f_hfn, t_hfn] = stft(ecg_hfn, Fs, 'Window', rectwin(n_window),
'OverlapLength', n_overlap);
[rect_noisy, f_noisy, t_noisy] = stft(ecg_noisy, Fs, 'Window', rectwin(n_window),
'OverlapLength', n_overlap);
% [s_lfn, f_lfn, t_lfn] = spectrogram(ecg_lfn, rectwin(n_window), n_overlap, [],
Fs);
% [s_hfn, f_hfn, t_hfn] = spectrogram(ecg_hfn, rectwin(n_window), n_overlap, [],
Fs);
% [s_noisy, f_noisy, t_noisy] = spectrogram(ecg_noisy, rectwin(n_window),
n_overlap, [], Fs);
rect_lfn_dB = 10*log10(abs(rect_lfn));        % converting magnitude of STFT of
the ecg_lfn signal to dB scale
rect_hfn_dB = 10*log10(abs(rect_hfn));        % converting magnitude of STFT of
the ecg_hfn signal to dB scale
```

```
rect_noisy_dB = 10*log10(abs(rect_noisy));    % converting magnitude of STFT of
the ecg_noisy signal to dB scale

figure(1)
subplot(311)
surf(t_lfn, f_lfn, rect_lfn_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-lfn with Rectangular Window');

subplot(312)
surf(t_hfn, f_hfn, rect_hfn_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-hfn with Rectangular Window');

subplot(313)
surf(t_noisy, f_noisy, rect_noisy_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-noisy with Rectangular Window');
%% step b) Using hamming window
%%
[hamming_lfn, f_lfn, t_lfn] = stft(ecg_lfn, Fs, 'Window', hamming(n_window),
'OverlapLength', n_overlap);
[hamming_hfn, f_hfn, t_hfn] = stft(ecg_hfn, Fs, 'Window', hamming(n_window),
'OverlapLength', n_overlap);
[hamming_noisy, f_noisy, t_noisy] = stft(ecg_noisy, Fs, 'Window',
hamming(n_window), 'OverlapLength', n_overlap);
% [s_lfn, f_lfn, t_lfn] = spectrogram(ecg_lfn, rectwin(n_window), n_overlap, [],
Fs);
% [s_hfn, f_hfn, t_hfn] = spectrogram(ecg_hfn, rectwin(n_window), n_overlap, [],
Fs);
% [s_noisy, f_noisy, t_noisy] = spectrogram(ecg_noisy, rectwin(n_window),
n_overlap, [], Fs);
hamming_lfn_dB = 10*log10(abs(hamming_lfn));        % converting magnitude of STFT
of the ecg_lfn signal to dB scale
hamming_hfn_dB = 10*log10(abs(hamming_hfn));        % converting magnitude of STFT
of the ecg_hfn signal to dB scale
hamming_noisy_dB = 10*log10(abs(hamming_noisy));    % converting magnitude of STFT
of the ecg_noisy signal to dB scale

figure(2)
subplot(311)
surf(t_lfn, f_lfn, hamming_lfn_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-lfn with Hamming Window');

subplot(312)
surf(t_hfn, f_hfn, hamming_hfn_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-hfn with Hamming Window');
```

```
subplot(313)
surf(t_noisy, f_noisy, hamming_noisy_dB, 'EdgeColor', 'none');
axis xy; axis tight; colormap(jet); view(0, 90);
xlabel('time (s)');
ylabel('frequency (Hz)');
title('STFT of ecg-noisy with Hamming Window');
```

[4] MATLAB Code “Q4_harun_durmus.m”:

```
%% Comments are added to the report (pdf file)
%%
clc;
clear all;
close all;
%% uploading images
load('ecg_1.mat'); % ecg_lfn    is from ecg_1.mat file
load('ecg_2.mat'); % ecg_hfn    is from ecg_2.mat file
load('ecg_3.mat'); % ecg_noisy  is from ecg_3.mat file
%% Question 4. Power Spectral Density Analysis by using periodogram, modified
periodogram, welch and levinson-durbin methods
%% step a) Using periodogram method (rectangular window)
fs = 1000; % sample frequency in Hz
N_lfn = length(ecg_lfn);
N_hfn = length(ecg_hfn);
N_noisy = length(ecg_noisy);

[PSD_lfn_a, f_lfn_a] = periodogram(ecg_lfn, rectwin(N_lfn), N_lfn, fs);
[PSD_hfn_a, f_hfn_a] = periodogram(ecg_hfn, rectwin(N_hfn), N_hfn, fs);
[PSD_noisy_a, f_noisy_a] = periodogram(ecg_noisy, rectwin(N_noisy), N_noisy, fs);
PSD_lfn_dB_a = 10*log10(PSD_lfn_a);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_hfn_dB_a = 10*log10(PSD_hfn_a);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_noisy_dB_a = 10*log10(PSD_noisy_a); % converting magnitude of psd of the
ecg_lfn signal to dB scale

figure(1)
subplot(311)
plot(f_lfn_a, PSD_lfn_dB_a);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Periodogram for estimating Power Spectral Density (PSD) of ecg-lfn');

subplot(312)
plot(f_hfn_a, PSD_hfn_dB_a);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Periodogram for estimating Power Spectral Density (PSD) of ecg-hfn');

subplot(313)
plot(f_noisy_a, PSD_noisy_dB_a);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Periodogram for estimating Power Spectral Density (PSD) of ecg-noisy');
%% step b) Using modified periodogram method (hamming window)
```

```
[PSD_lfn_b, f_lfn_b] = periodogram(ecg_lfn, hamming(N_lfn), N_lfn, fs);
[PSD_hfn_b, f_hfn_b] = periodogram(ecg_hfn, hamming(N_hfn), N_hfn, fs);
[PSD_noisy_b, f_noisy_b] = periodogram(ecg_noisy, hamming(N_noisy), N_noisy, fs);
PSD_lfn_dB_b = 10*log10(PSD_lfn_b);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_hfn_dB_b = 10*log10(PSD_hfn_b);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_noisy_dB_b = 10*log10(PSD_noisy_b); % converting magnitude of psd of the
ecg_lfn signal to dB scale

figure(2)
subplot(311)
plot(f_lfn_b, PSD_lfn_dB_b);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Modified Periodogram for estimating Power Spectral Density (PSD) of ecg-
lfn');

subplot(312)
plot(f_hfn_b, PSD_hfn_dB_b);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Modified Periodogram for estimating Power Spectral Density (PSD) of ecg-
hfn');

subplot(313)
plot(f_noisy_b, PSD_noisy_dB_b);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title('Modified Periodogram for estimating Power Spectral Density (PSD) of ecg-
noisy');
%% step c) Using welch's method
[PSD_lfn_c, f_lfn_c] = pwelch(ecg_lfn, hamming(256), 128, 256, fs);
[PSD_hfn_c, f_hfn_c] = pwelch(ecg_hfn, hamming(256), 128, 256, fs);
[PSD_noisy_c, f_noisy_c] = pwelch(ecg_noisy, hamming(256), 128, 256, fs);
PSD_lfn_dB_c = 10*log10(PSD_lfn_c);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_hfn_dB_c = 10*log10(PSD_hfn_c);     % converting magnitude of psd of the
ecg_lfn signal to dB scale
PSD_noisy_dB_c = 10*log10(PSD_noisy_c); % converting magnitude of psd of the
ecg_lfn signal to dB scale

figure(3)
subplot(311)
plot(f_lfn_c, PSD_lfn_dB_c);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title("Welch's method for estimating Power Spectral Density (PSD) of ecg-lfn");

subplot(312)
plot(f_hfn_c, PSD_hfn_dB_c);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title("Welch's method for estimating Power Spectral Density (PSD) of ecg-hfn");
```

```
subplot(313)
plot(f_noisy_c, PSD_noisy_dB_c);
grid on;
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
title("Welch's method for estimating Power Spectral Density (PSD) of ecg-noisy");
%% step d) Using levin-durbin algorithm
order_lfn = 5;          % order of the AR model
order_hfn = 15;          % order of the AR model
order_noisy = 30;        % order of the AR model
nom_lfn = 1;        % nominator coefficient of freqz(.) is 1
nom_hfn = 1;        % nominator coefficient of freqz(.) is 1
nom_noisy = 1;      % nominator coefficient of freqz(.) is 1
[denom_lfn, error_lfn] = levinson(xcorr(ecg_lfn), order_lfn);
[denom_hfn, error_hfn] = levinson(xcorr(ecg_hfn), order_hfn);
[denom_noisy, error_noisy] = levinson(xcorr(ecg_noisy), order_noisy);
[PSD_lfn_d, f_lfn_d] = freqz(nom_lfn, denom_lfn, N_lfn, fs);
[PSD_hfn_d, f_hfn_d] = freqz(nom_hfn, denom_hfn, N_hfn, fs);
[PSD_noisy_d, f_noisy_d] = freqz(nom_noisy, denom_noisy, N_noisy, fs);
PSD_lfn_dB_d = 10*log10(abs(PSD_lfn_d).^2/error_lfn);          % converting
magnitude of psd of the ecg_lfn signal to dB scale
PSD_hfn_dB_d = 10*log10(abs(PSD_hfn_d).^2/error_hfn);          % converting
magnitude of psd of the ecg_lfn signal to dB scale
PSD_noisy_dB_d = 10*log10(abs(PSD_noisy_d).^2/error_noisy);    % converting
magnitude of psd of the ecg_lfn signal to dB scale

figure (4)
subplot(311)
plot(f_lfn_d, PSD_lfn_dB_d);
grid on;
title('Levinson-Durbin Algorithm for estimating Power Spectral Density (PSD) of
ecg-lfn');
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');

subplot(312)
plot(f_hfn_d, PSD_hfn_dB_d);
grid on;
title('Levinson-Durbin Algorithm for estimating Power Spectral Density (PSD) of
ecg-hfn');
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');

subplot(313)
plot(f_noisy_d, PSD_noisy_dB_d);
grid on;
title('Levinson-Durbin Algorithm for estimating Power Spectral Density (PSD) of
ecg-noisy');
xlabel('frequency (Hz)');
ylabel('power/frequency (dB/Hz)');
```

[5] MATLAB Code “Q5_harun_durmus.m”:

```
%% Comments are added to the report (pdf file)
%%
clc;
clear all;
close all;
```

```
%% uploading images
load('ecg_2.mat'); % ecg_hfn is from ecg_2.mat file
N_hfn = length(ecg_hfn);
fs = 1000; % sampling rate in Hz
ts = 1/fs;
t = 0:ts:(N_hfn-1)*ts;
%% Question 5. QRS Detection
%% low-pass Filter
% nominator part exist from the coefficients of x[n] and denominator part exist
% from the coefficients of y[n]:
% y[n] = 2y[n-1] - y[n-2] + x[n] - 2x[n-6] + x[n-12]
nom_low = [1,0,0,0,0,0,-2,0,0,0,0,0,1];
denom_low = [1,-2,1];
ecg_hfn_lowpass = filter(nom_low,denom_low,ecg_hfn);
%% high-pass filter
% y[n] = y[n-1] - x[n]/32 + x[n-16] - x[n-17] + x[n-32]/32
nom_high = [-1/32,zeros(1, 15),1,-1,zeros(1, 14),1/32];
denom_high = [1,-1];
ecg_hfn_highpass = filter(nom_high,denom_high,ecg_hfn_lowpass);
%% differentiator
% 8y[n] = 2x[n] + x[n-1] - x[n-3] - 2x[n-4]
nom_diff = [2,1,0,-1,-2];
denom_diff = 8;
ecg_hfn_diff = filter(nom_diff,denom_diff,ecg_hfn_highpass);
ecg_hfn_diff_0 = ecg_hfn_diff;
%% squaring
N_diff = length(ecg_hfn_diff);
for k=1:N_diff
    ecg_hfn_diff(k) = ecg_hfn_diff(k)^2;
end

figure (1)
subplot(221)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_hfn_lowpass);
xlabel("time (s)");
ylabel("amplitude");
subtitle("Signal Before and After Low-pass Filtering");
legend("before lowpass filter","after lowpass filter");
xlim([0 N_hfn*ts]);

subplot(222)
plot(t, ecg_hfn_lowpass);
hold on;
plot(t, ecg_hfn_highpass);
subtitle("Signal Before and After High-pass Filtering");
xlabel("time (s)");
ylabel("amplitude");
legend("before highpass filter","after highpass filter");
xlim([0 N_hfn*ts]);

subplot(223)
plot(t, ecg_hfn_highpass);
hold on;
plot(t, ecg_hfn_diff_0);
subtitle("Signal Before and After Differentiating");
xlabel("time (s)");
ylabel("amplitude");
```

```
legend("before differentiator","after differentiator");
xlim([0 N_hfn*ts]);

subplot(224)
plot(t, ecg_hfn_diff_0);
hold on;
plot(t, ecg_hfn_diff);
subtitle("Signal Before and After Squaring");
xlabel("time (s)");
ylabel("amplitude");
legend("before squaring","after squaring");
xlim([0 N_hfn*ts]);
%% moving average filter
duration = 0.15; % 150 ms moving average filter parameter
size_window_1 = fs * duration; % defined window size of the filter
size_window_2 = 10; % the window size from the question 1
nom_maf_1 = ones(1,size_window_1); % nominator coefficient of the filter
nom_maf_2 = ones(1,size_window_2); % nominator coefficient of the filter
denom_maf_1 = size_window_1; % denominator coefficient of the filter
denom_maf_2 = size_window_2; % denominator coefficient of the filter
ecg_hfn_filtered_1 = filter(nom_maf_1, denom_maf_1, ecg_hfn_diff);
ecg_hfn_filtered_2 = filter(nom_maf_2, denom_maf_2, ecg_hfn_diff);
ecg_hfn_filtered_1 = ecg_hfn_filtered_1./2;
ecg_hfn_filtered_2 = ecg_hfn_filtered_2./20;

figure (2)
subplot(211)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_hfn_filtered_1);
grid on;
xlabel("time (s)");
ylabel("amplitude");
title("Comparison of Two QRS Detected Signals With Different Window Sizes");
subtitle('Original and QRS Detected ECG Signal in time domain');
legend("original ecg signal","qrs detected ecg signal with window size 150");
xlim([0 N_hfn*ts]);

subplot(212)
plot(t, ecg_hfn);
hold on;
plot(t, ecg_hfn_filtered_2);
grid on;
xlabel("time (s)");
ylabel("amplitude");
subtitle('Original and QRS Detected ECG Signal in time domain');
legend("original ecg signal","qrs detected ecg signal with window size 10");
xlim([0 N_hfn*ts]);
```