EEE 491 - Biomedical Engineering

Final Project

Taha Ahmed 19015866

1 Introduction

Electrocardiography (ECG) is a non-invasive medical test used to detect and diagnose cardiovascular diseases. It measures the electrical activity of the heart and produces a graph, called an electrocardiogram, that shows the heart's rhythm and any abnormalities in its function. ECG is a vital tool for medical professionals in diagnosing and monitoring heart conditions.
However, ECG signals are often contaminated with noise and interference, making it difficult to analyze them accurately. In this report, we will explore different signal processing techniques to enhance the quality of ECG signals and extract useful information from them. Specifically, we will focus on removing 50 Hz noise using a notch filter, optimizing the signal-to-noise ratio by limiting the bandwidth, and detecting the heart rate using autocorrelation. We will apply these techniques to both a noisy ECG signal and a processed signal and compare the results.
The main objective of this report is to demonstrate how signal processing techniques can improve the quality of ECG signals and facilitate the accurate detection of heart rate. By achieving this objective, we hope to contribute to the ongoing efforts to enhance the effectiveness and reliability of ECG as a diagnostic tool for cardiovascular diseases.

2 Procedure

2.1 Removing signals from muscle movement

Plot the signal:

```
% load the signal
EKG1 = [struct2cell(load('ecg.mat')){:}];
fs = 500;
time = (0:length(EKG1)-1)/fs; % time vector
plot(time, EKG1);
set(gca, "fontsize", 24) % increase the fontsize of the plot
xlabel("Time (s)");
ylabel("Voltage (V)");
```

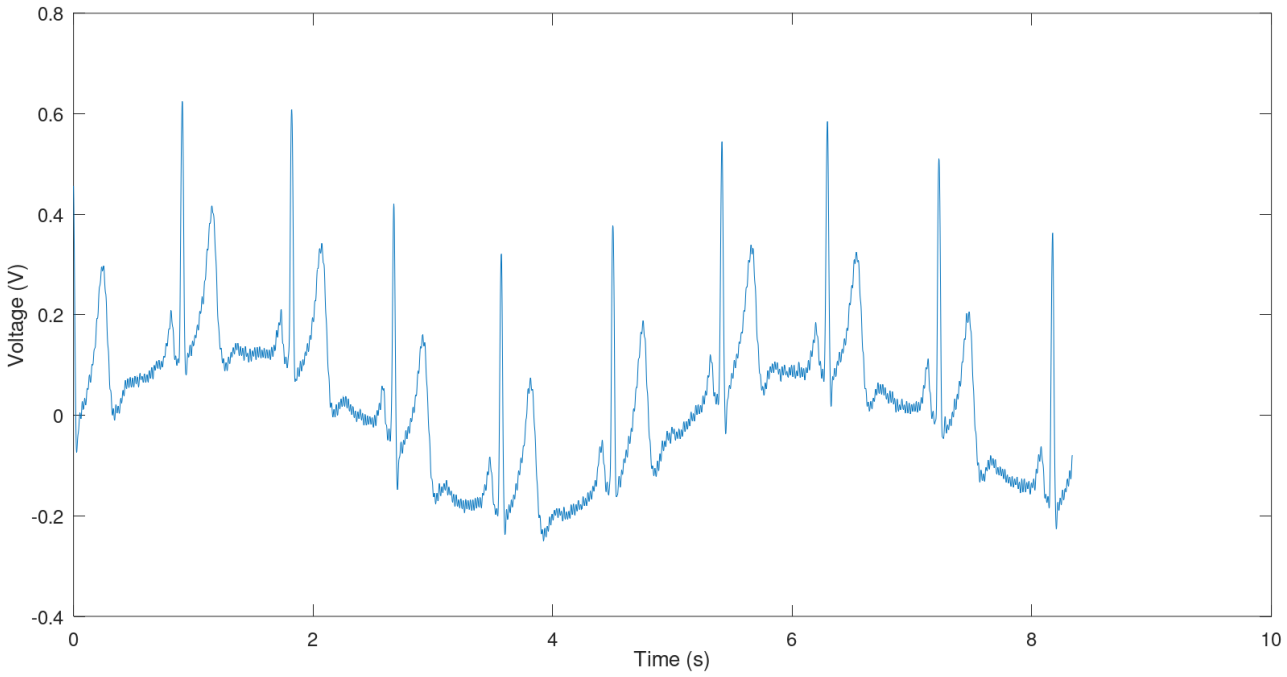

Figure 1: Original Signal

Zoom on one period:

```
t_start = 1.5; % start time for zoomed in plot
t_end = 1.52; % end time for zoomed in plot
idx_start = round(t_start*500); % index corresponding to start time
idx_end = round(t_end*500); % index corresponding to end time
plot(time(idx_start:idx_end), EKG1(idx_start:idx_end), "linewidth", 2);
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Voltage (V)");
```

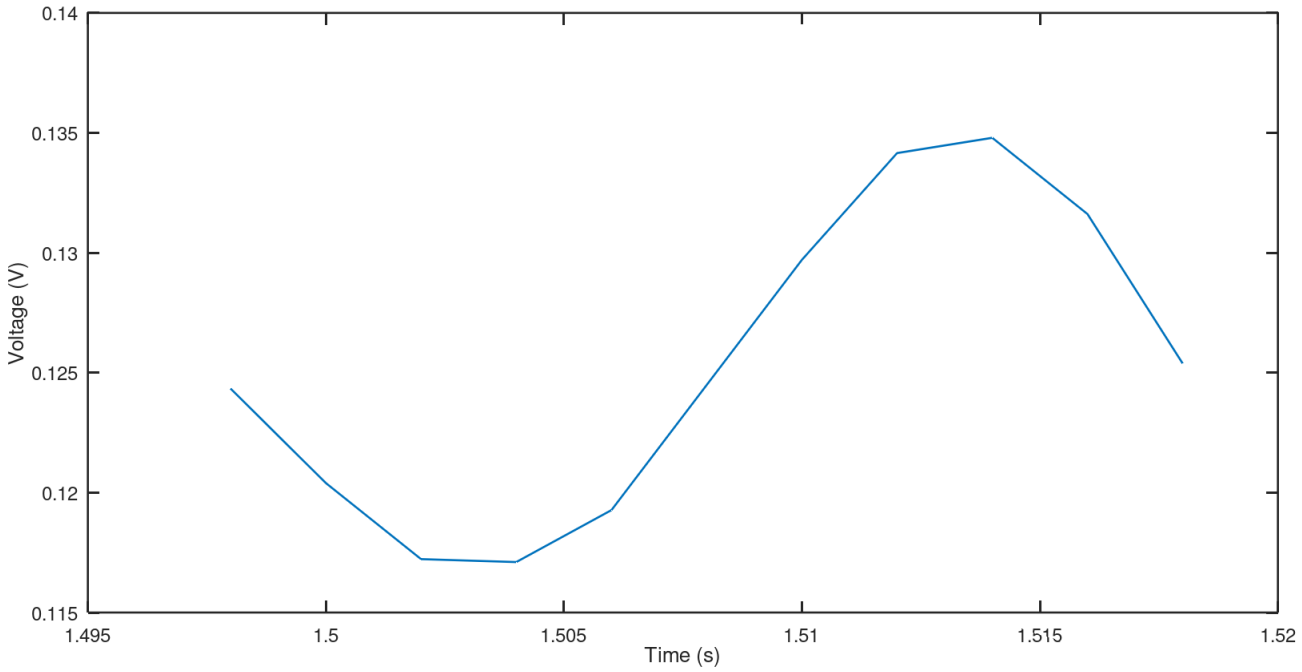

Figure 2: One interval of the signal, from `t=1.5` to `t=1.52`

Remove noise generated by muscles:

```
freq = (0:length(EKG1)-1)/length(EKG1)*500; % frequency vector
ecg_fft = fft(EKG1);
ecg_fft(freq < 0.5) = 0; % set frequencies below 0.5 Hz to zero
ecg1 = ifft(ecg_fft);
plot(time, EKG1, 'b', time, ecg1, 'r');
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Voltage (V)");
legend("Original signal", "Filtered signal");
```

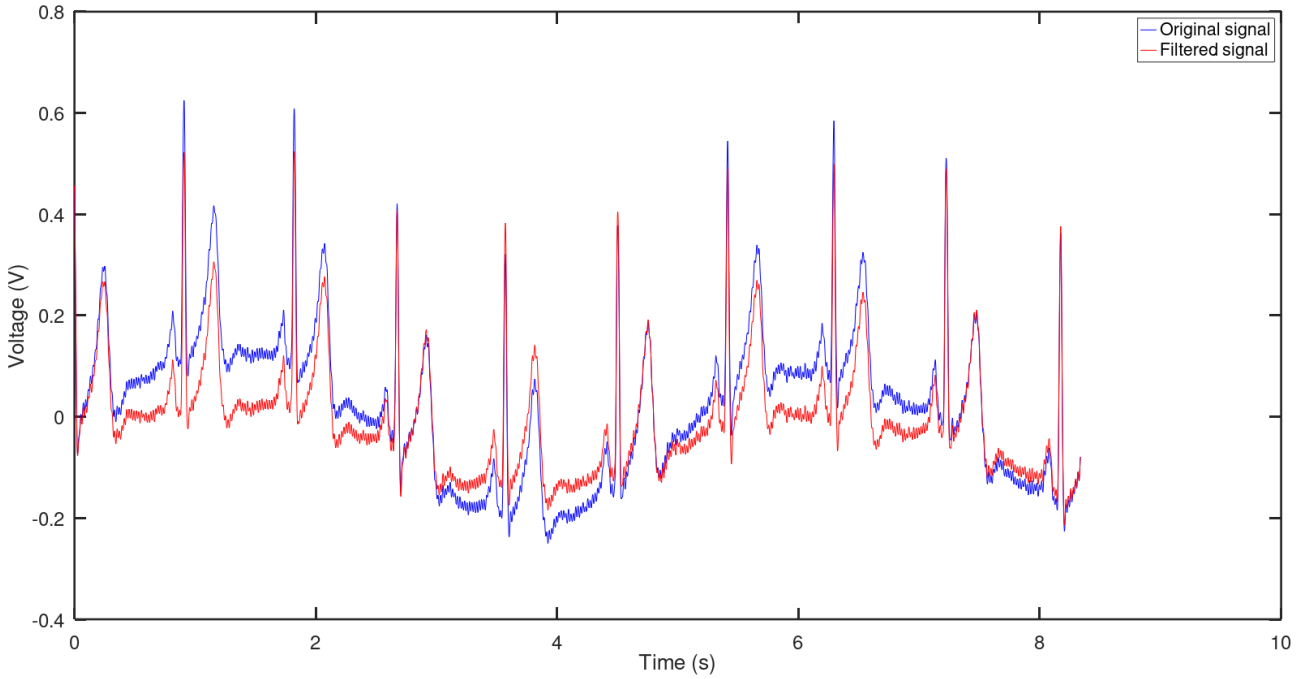

Figure 3: Filtered signal vs original signal

Make sure that you filter has a real impulse response by making an inverse Fourier transform of the transfer function.

```
filter_fft = ones(size(EKG1)); % initialize filter to all ones
filter_fft(freq < 0.5) = 0; % set frequencies below 0.5 Hz to zero
filter_ifft = ifft(ifftshift(filter_fft)); % inverse Fourier transform of filter
t = (-length(filter_ifft)/2:length(filter_ifft)/2-1)/fs; % create time vector
plot(t, real(fftshift(filter_ifft)), "linewidth", 2); % center filter in time
    domain before plotting
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Amplitude");
```

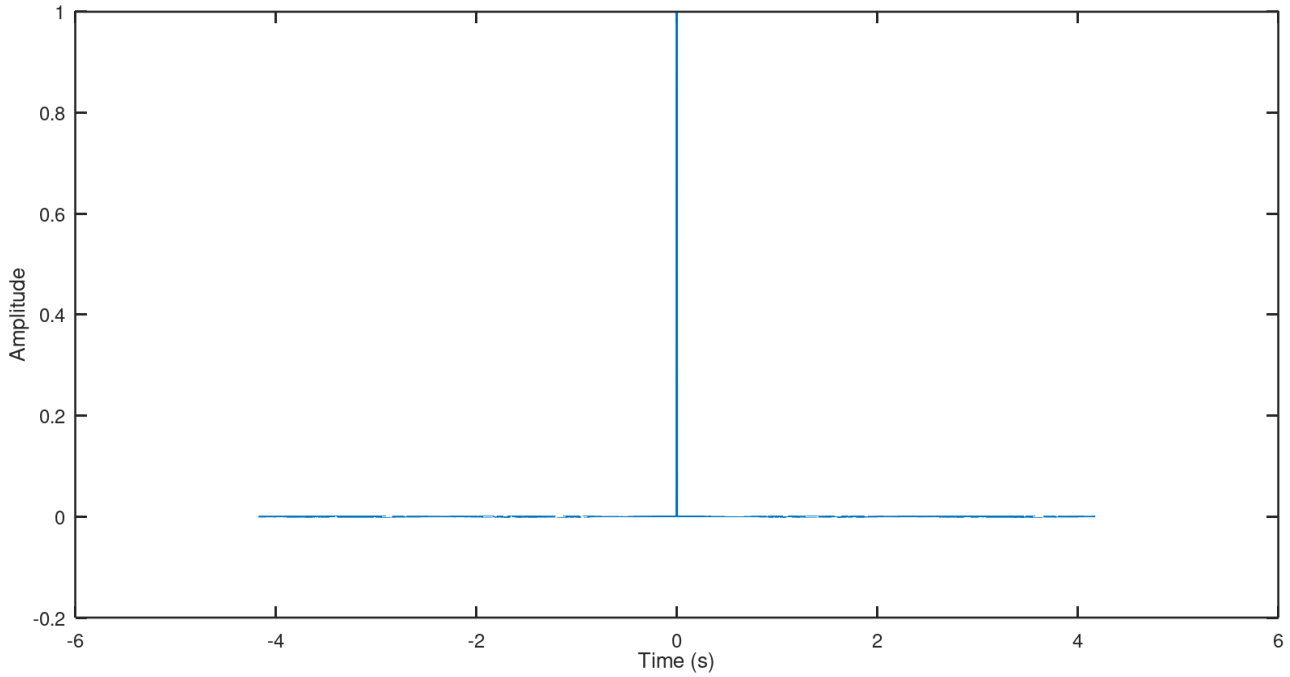

Figure 4: real impulse response

2.2 Removing 50 Hz interference:

Design a notch filter with a zero at 50 Hz. We will design it with butterworth filter

```
pkg load signal % I am using octave   \_(   )_/
fs = 500; % sampling frequency
f0 = 50; % notch frequency
bw = 1/(fs/2); % normalized notch bandwidth
Q = f0/bw; % quality factor
wo = f0/(fs/2); % normalized notch frequency
[b,a] = butter(2, [wo-bw/2, wo+bw/2], 'stop'); % design notch filter coefficients
ECG_signal_filt = filter(b, a, EKG1); % apply notch filter using filtfilt
plot(time, EKG1, 'b', time, ECG_signal_filt, 'r');
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Voltage (V)");
legend("Original signal", "Filtered signal");
```

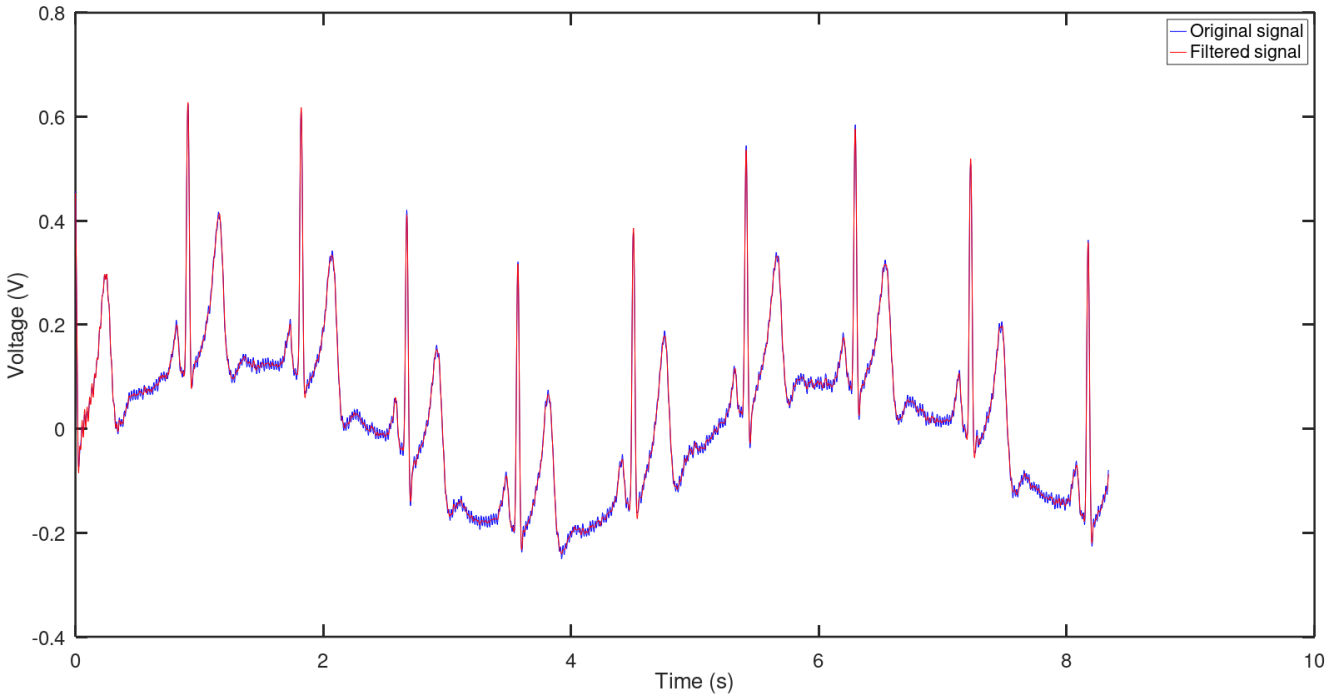

Figure 5: Apply notch filter to remove 50Hz noize

Zoom in with your mouse to see the smoothed signal

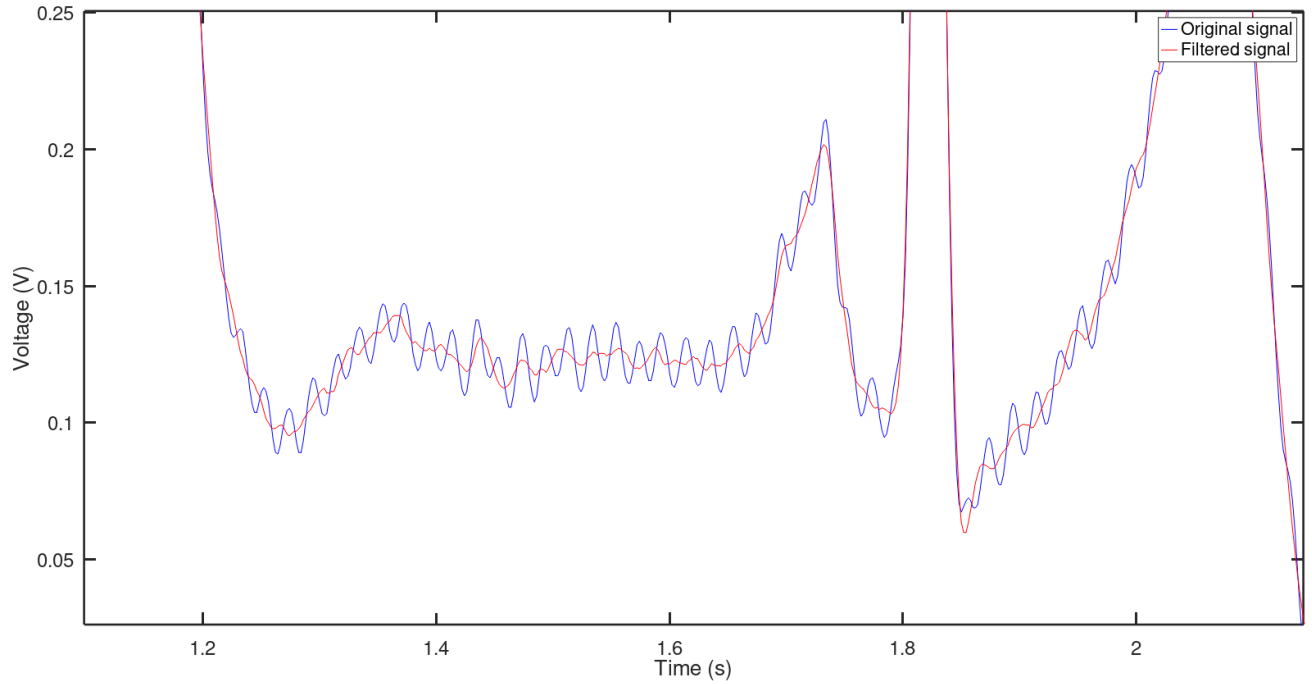

Figure 6: A little zoom

2.3 Increasing The SNR

```
clf;
fs = 500; % sampling frequency in Hz
order = 2; % filter order
fc = [5 20 100]; % cut-off frequencies in Hz
```

```
colors = ['r', 'g', 'y'];
figure;
hold on;
plot(time, ECG_signal_filt, 'b', "linewidth", 2);
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Voltage (V)");
legend("Original signal");

for i = 1:length(fc)
    [b,a] = butter(order, fc(i)/(fs/2), 'low');
    ecg_filt = filter(b, a, ECG_signal_filt);
    plot(time, ecg_filt, colors(i));
    set(gca, "linewidth", 2, "fontsize", 24);
end

title("Filtered Signals with Different Cut-off Frequencies");
legend("Original signal", "5", "10", "100");
hold off;
```

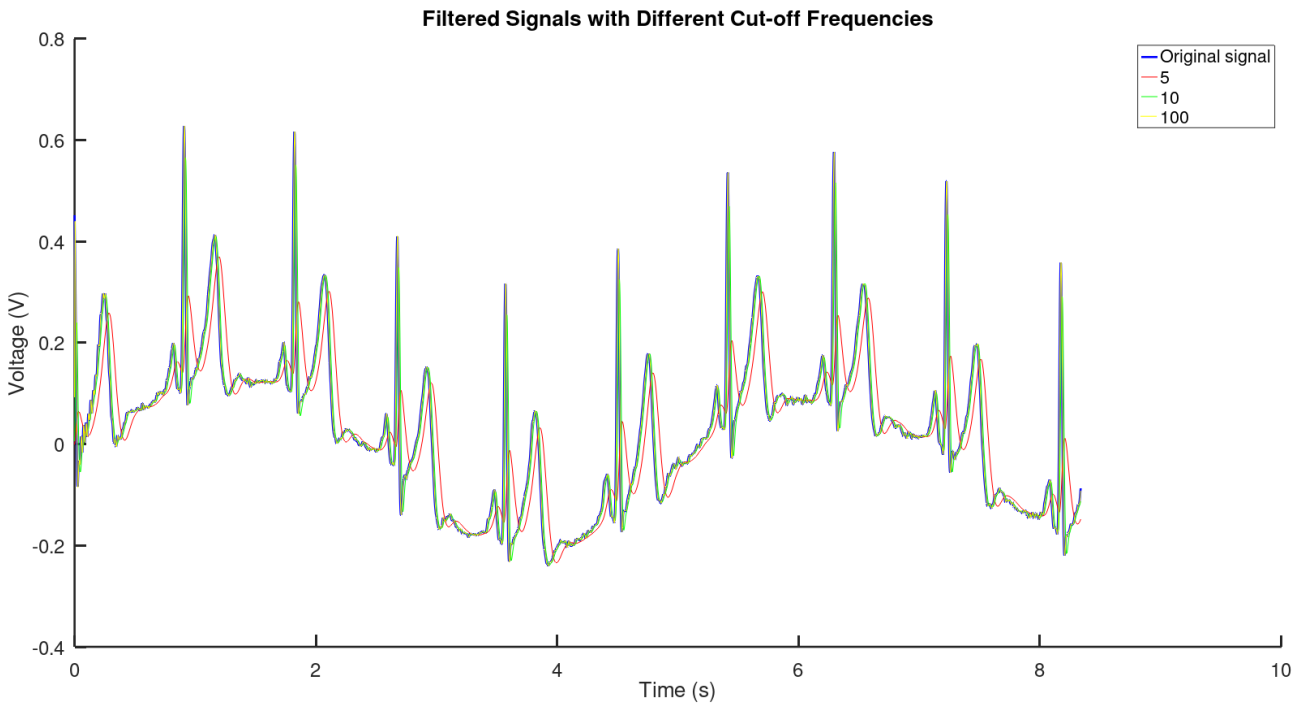

Figure 7: Try to increase the SNR value by applying low pass filter with different cutoff frequencies

Zoom in with your mouse to see the smoothed signal

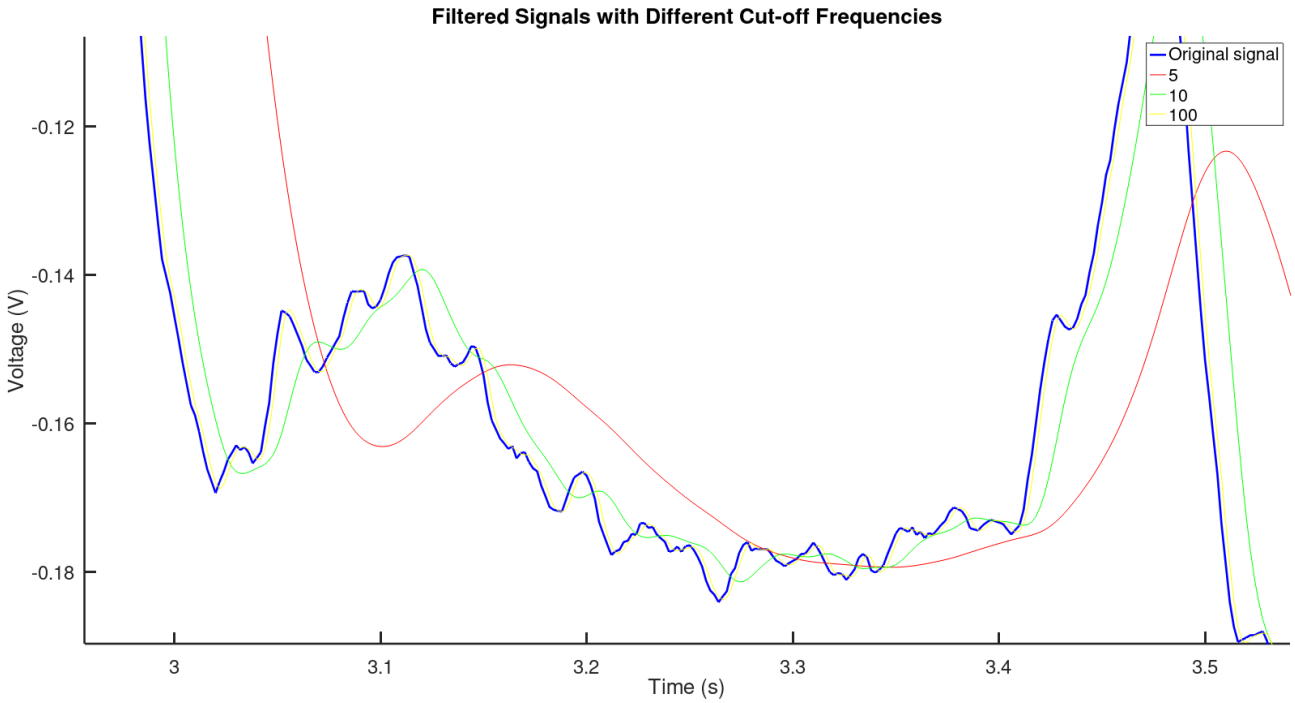

Figure 8: A little zoom

2.4 Finding the heart rate using autocorrelation:

```
[b, a] = butter(5, 3/(fs/2), 'low');
ecg3 = filter(b, a, ECG_signal_filt);
% plot the filtered signal
plot(time, ecg3, 'b');
set(gca, "linewidth", 2, "fontsize", 24);
xlabel("Time (s)");
ylabel("Voltage (V)");
legend("Filtered signal");
% get the autocorrelation of the filtered signal
[autocor,lags] = xcorr(ecg3,'coeff');
% plot the autocorrelation
plot(lags/fs,autocor, "linewidth", 2)
xlabel('Lag')
ylabel('Autocorrelation')
% find the peaks of the autocorrelation
[peaks,locs] = findpeaks(autocor, "DoubleSided");
% find the average time between peaks
period = mean(diff(locs)) / fs
% find the heart rate by dividing 60 seconds by the average period
heart_rate = 60 / period
% plot the peaks on top of the autocorrelation
hold on
pks = plot(lags(locs)/fs,peaks,'or', "linewidth", 2);
```

```
24 set(gca, "linewidth", 2, "fontsize", 24);
25 title(sprintf('Heart rate: %.1f bpm', heart_rate));
26 hold off
27 legend(pks, 'peaks')
```

Looking at the console, we can find that `heart_rate = 104.53`

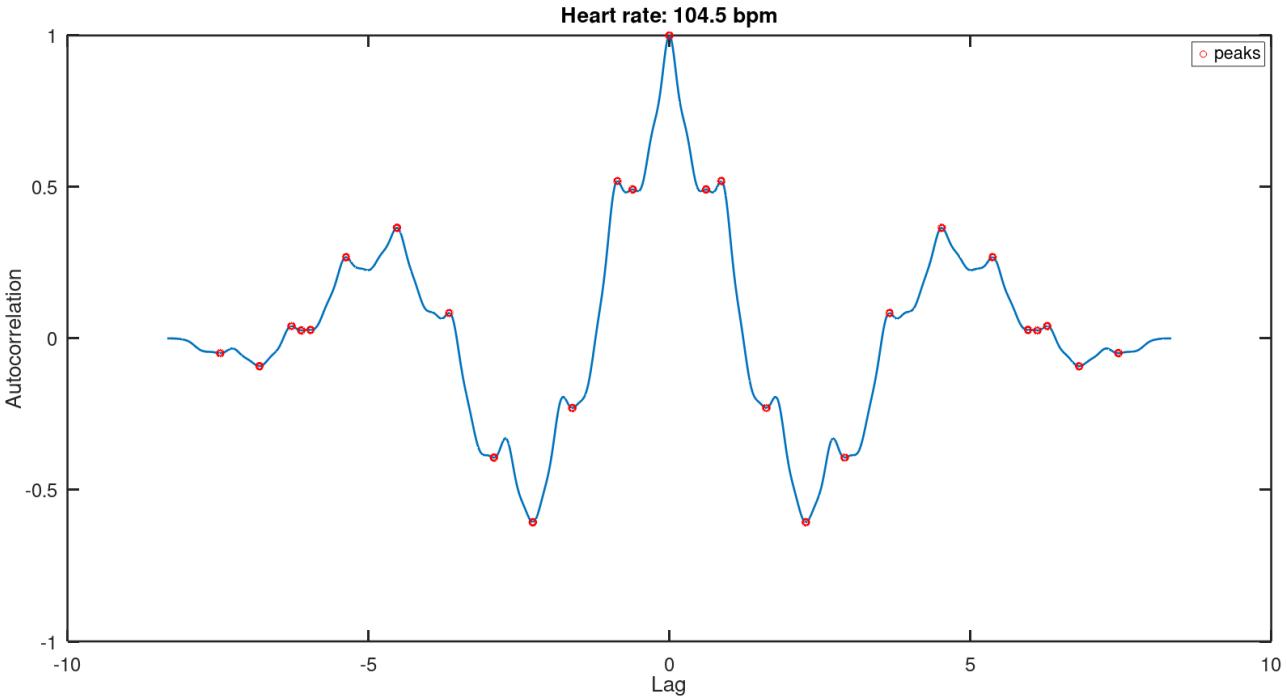

Figure 9: autocorrelation can help verify the presence of cycles and determine their durations.