Linköping University

TBMT01 - BIOMEDICAL SIGNAL PROCESSING

SPT1: ECG ANALYSIS

Group D1

Mirelle Ximena Soriano Pérez - mirso643

Alexandra Roser - alero536

Alberto Zilli - albzi339

2015, December 21st

Table of contents

1. Introduction

This task is about using the software MATLAB to create an algorithm so we can analyze 5 ECG that can be from healthy patients or with an arrhythmia or ST-shift, but also the algorithm should be useful to any ECG. The signals from this 5 ECG contains some noise and baseline drift. We needed to create an algorithm to detect and visualize the gradual change in ST level.

1.1 The ECG

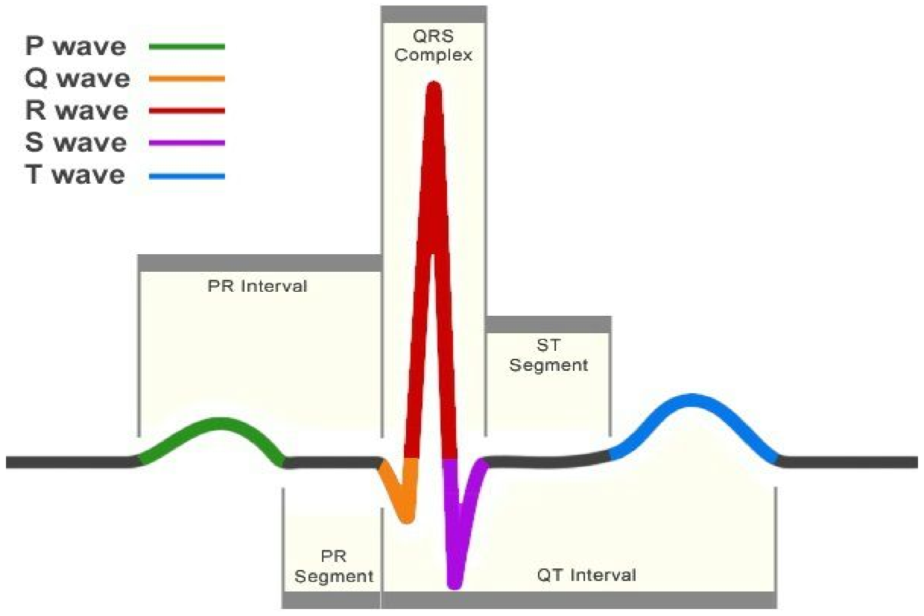

Figure 1: Typical ECG [1]

An electrocardiogram (ECG) records electrical signals caused by action potentials in the heart, visualising them as distinctive waves and peaks. The first wave, the so-called P wave, shows the atrial depolarization, the QRS complex, consisting of the Q, R, and S peaks, show the rapid ventricular depolarization, and the T wave the ventricular repolarization [2, p.776]. The so-called ST segment is the time of ventricular contraction, also known as ventricular systole [2, p. 777]

1.2 ECG Anomalies

ECGs can be used to detect anomalies in the heart beats or signal conduction by analysing the waves and peaks. Since the ECG recording can be divided into different sections with their respective meanings, anomalies can be detected by comparing those to common values.
ECG provides important information: the duration of the electrical wave and the amount of the electrical activity. The first one can tell if the rhythm is too slow, too fast or irregular (arrhythmia). The second can suggest malfunctioning of any part of the heart, for example inferring by the T wave amplitude.

2. Background

2.1 Signal Issues

When obtaining ECG signals, many factors can affect the waveforms, which make the interpretation difficult. The interferences that present with major frequency are:

- Power line noise, at a constant frequency of 50Hz or 60Hz and its multiples.
- Electrode noise and motion artifacts.
- Chest wall muscle noise.
- Baseline wandering due to respiration

Therefore, before proceeding with any analysis, the ECG must be manipulated as necessary, for example filter it.

2.2 Pathologies (arrhythmia)

One of the most common problem regarding the heart, is the alteration of the normal cardiac pace (arrhythmia), which can be easily detected by looking at a patient's ECG. The normal value of beats per minute goes from 60 bpm to 100 bpm, when this value is lowered, the pathology is called bradycardia, and when it is exceeded is called tachycardia. When analyzing an ECG, you can observe if the length of the R-R intervals varies considerably, if this fulfills it is likely that there is a problem with the pacemaker.

2.3 ST Shift

The ST segment indicates the time between the end of the ventricles contraction and the beginning of the repolarization or rest, in normal conditions it is isoelectric. Another anomaly is the ST shift, which can be either elevated or depressed, indicating myocardial infarction or not enough oxygen delivery to the cardiac muscle [2, p. 777].
To find the ST segment, it is necessary to know when the S peak ends and the T wave starts. After this, the average of the ST segment is calculated and compared to the average baseline. If the average of the ST segment is above the baseline,we can say it is elevated and if it is below the baseline we can say is depressed.

3. Algorithm description

3.1 Peak detection and arrhythmias

This algorithm is constructed to detect two of the most common arrhythmias, bradycardia and tachycardia, either too slow or too fast heart beat. Deviations of heart paces with a normal value of around 60 to 100 beats per minute can be detected by spotting the QRS complex in each heartbeat, calculating the average time between them and then comparing the time between each heartbeats to the normal heart rate.

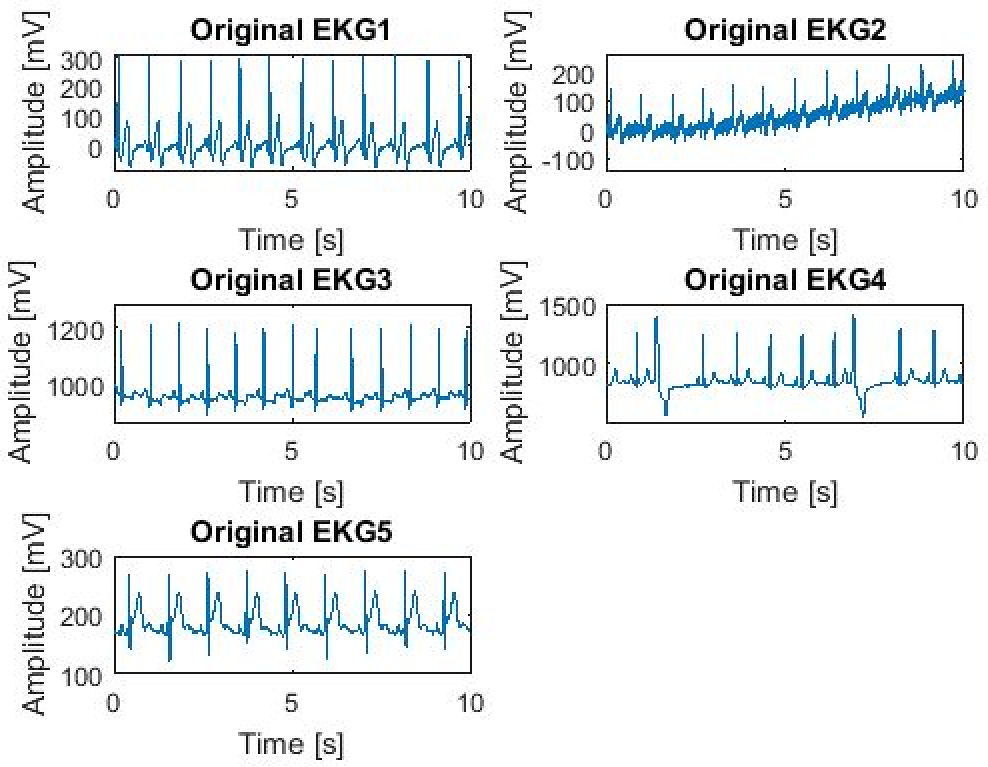

Figure 2: Unfiltered EKG signals

To get rid of the baseline wander and reduce the noise of the input signal respectively, a bandpass filter as a combination of a lowpass and highpass filter were used. With this filter, only a certain frequency range is let through. This filter also sets the baseline to zero, which will be important for the ST shift later.

It should also be mentioned that a lot of the code can be made more automatic with arrays, while we decided to calculate everything for each EKG signal manually, since it is only five EKG signals. This also means that the sampling frequencies are set in the code to each EKG respectively. Should the code be used for more or other signals, this has to be adjusted accordingly.

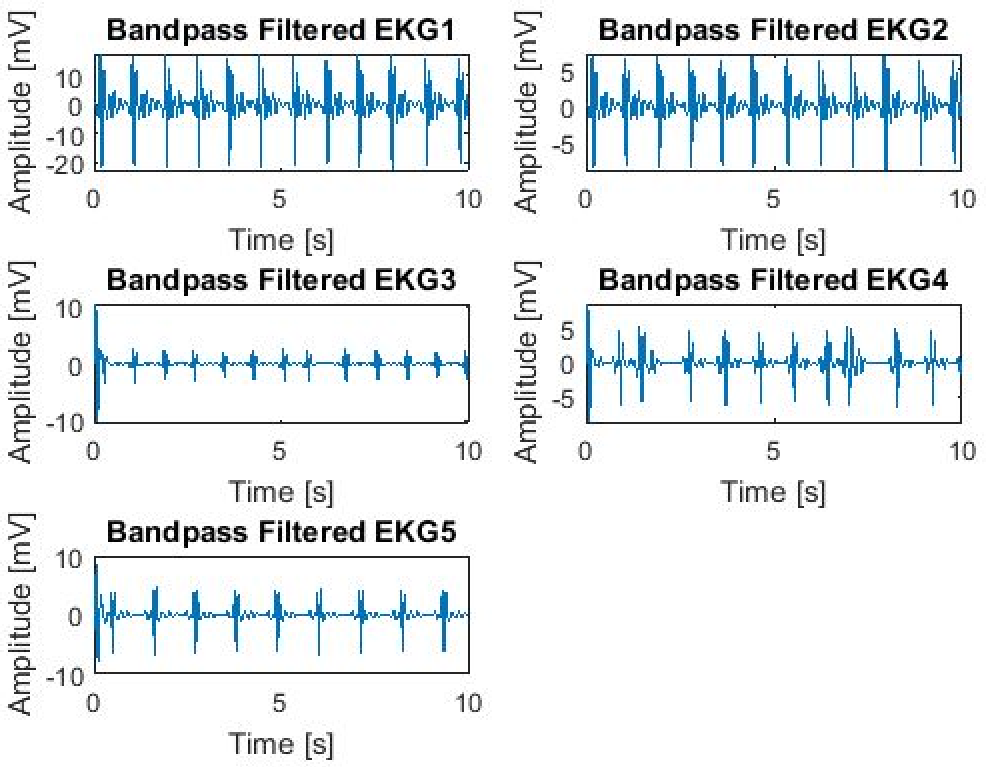

Figure 3: EKG signals after bandpass filter

Afterwards, a derivative filter is applied to evidence the slopes' changes in direction and highlight the QRS complex.

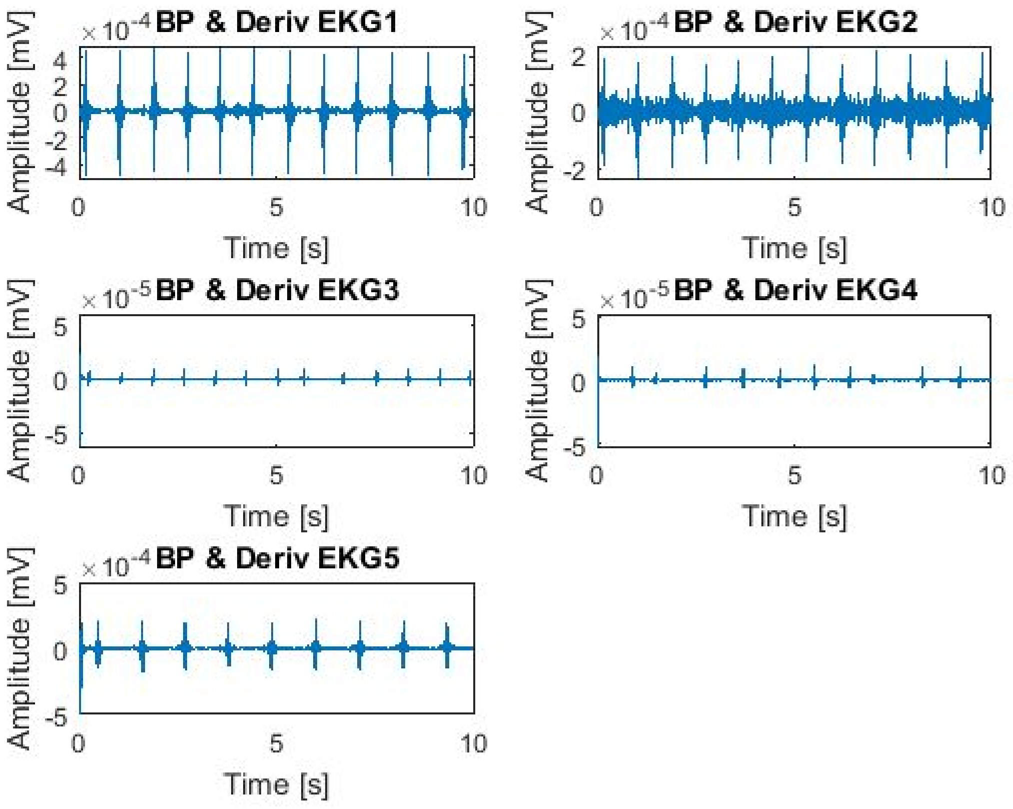

Figure 4: EKG signals after derivative filter

To enhance the peak amplitudes further and distinguish them better from smaller ones, a non-linear transformation in form of a squarer is used.

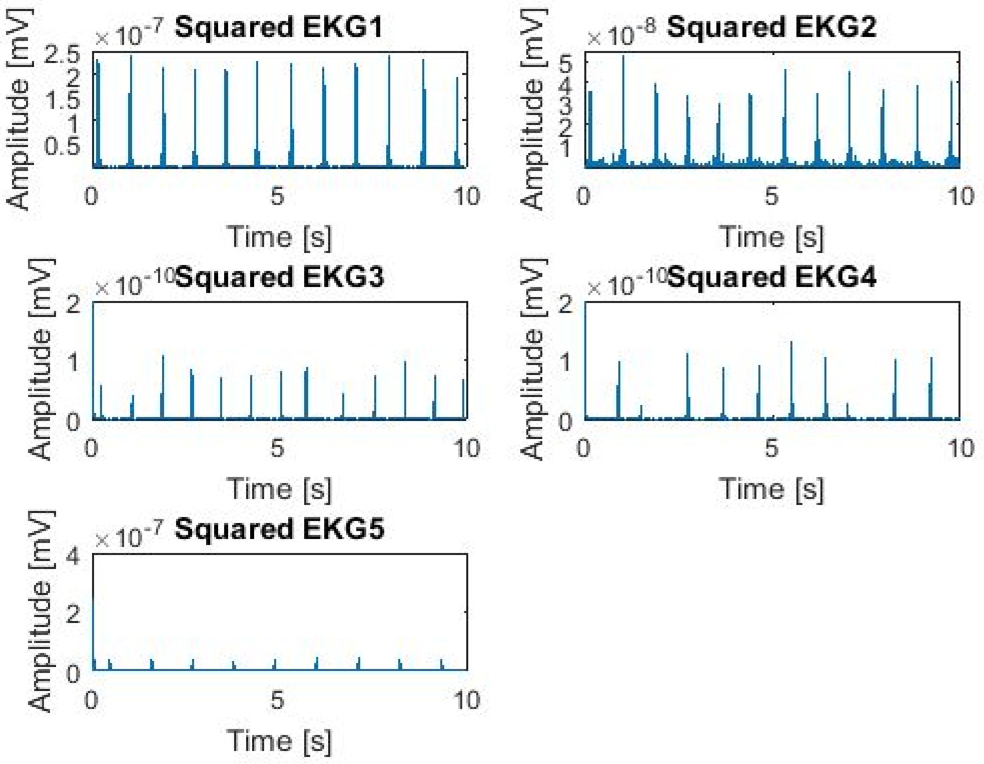

Figure 5: EKG signals after squarer

The resulting filtered and squared ECG is averaged through a convolution with the Hamming window. This window is used to minimise the closest peak, since we only want to detect the R peaks.

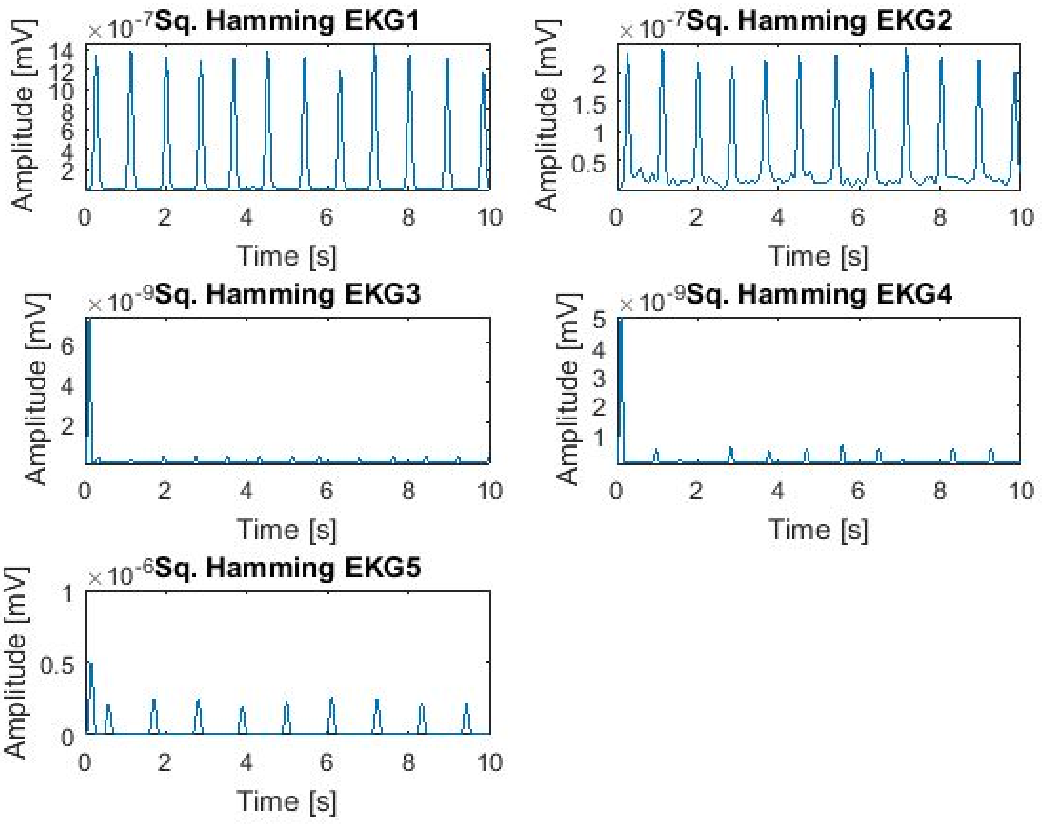

Figure 6: EKG signals after Hamming window

With help of the findpeak function, the actual R peaks can be detected, which are the biggest peaks. To eliminate smaller peaks and avoid false detections, a threshold filter is used.

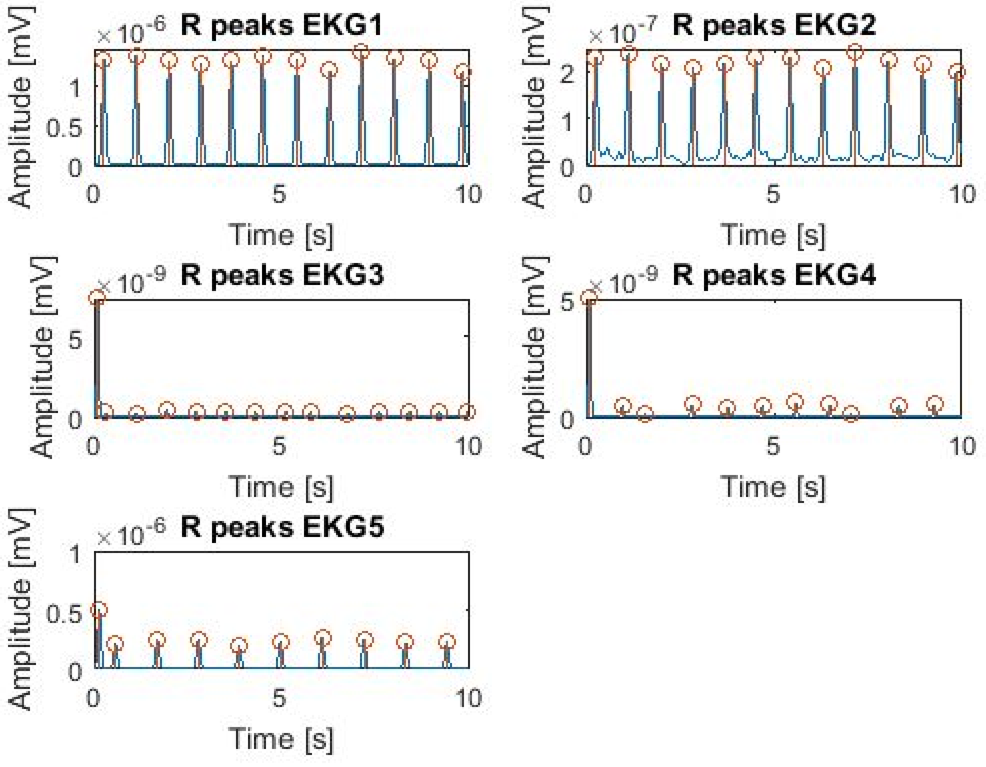

Figure 7: EKG signals after finding the R peaks

Once the R peaks have been detected, the R-R intervals can be calculated and therefore the average time between each heartbeat.
In a next step, each R-R interval is compared to the calculated average time between heartbeats. This is done by summing up the times between R peaks and dividing it by the number of intervals. If the R-R interval time is shorter than the average time calculated in bpm, the tachycardia counter goes up by 1. In case it is longer, the bradycardia counter goes up by 1. After the calculation, the number of occurrences for tachycardia and bradycardia is displayed.

It should be mentioned that since our algorithm calculates the average time of all intervals between heartbeats, some minor amounts of bradycardia or tachycardia will go unnoticed if the average is still in a normal pace (as can be seen at EKG3). Also unnoticed will be cases where the count of tachycardia and bradycardia is roughly the same, cancelling each other out (as can be seen at EKG4). Should one count be much higher than the other one, only one of them will be registered as arrhythmia (as can be seen at EKG5). These problems should be fixed, but to not let them go unnoticed at all, we have included a message that any arrhythmia has been detected, even if the average is rhythmic.

3.2 Arrhythmia description present in EKG 3 and EKG 4 [new]

The arrhythmias present in EKG3 and EKG4 seem to be premature atrial contraction (PAC) and premature ventricular contraction (PVC) respectively. Both arrhythmias are characterised by depolarisations that occur too early and are initiated in the atrium or ventricle.
PACs tend to have narrow QRS complexes while PVCs have wider than normal complexes [3]. Additionally, the QRS complex of a PVC is over 120ms wide and is followed by a compensatory pause [4].
A so-called compensatory pause is the time following a PVC which is long enough to compensate for the premature contraction, so that the interval between the R peaks preceding and following the PVC equals two normal heartbeats. [5]

The time period following a PAC is also longer than normal, however the abovementioned interval between the preceding and following R peaks does not equal two normal heartbeats, which is why this is shorter than a compensatory pause and therefore a noncompensatory pause, increasing the heart rate. [6]

In EKG4, as the PVCs are followed by a large, negative dip, the output of the signal after the squarer and Hamming window that was used for the R peak detection can also be used to detect the incidents of PVC. These incidents are clearly visible as smaller-than-average R peaks because the following dips are included into these peaks, affecting their amplitude.
A way to detect these decreased R peaks is to calculate an average of the amplitude of the peaks after the squarer and Hamming window and only detect the peaks that are below this average. As the average turned out to be too high, a certain percentage of it has to be used, here e.g. 50%.

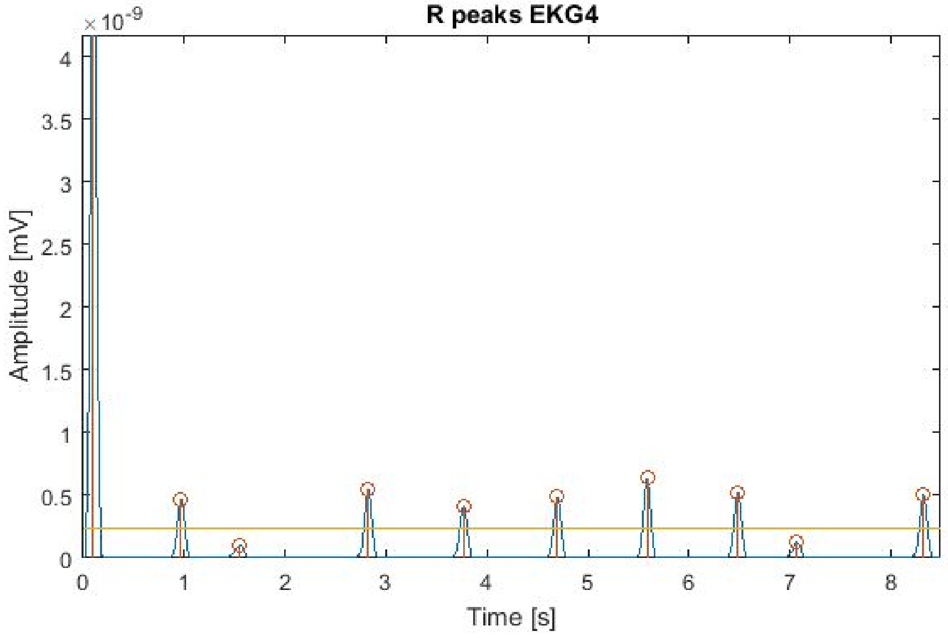

Figure 8: Detected R peaks in EKG4 and their average divided by 2

During the implementation of the ST Shift it became apparent that the x-values of the detected R peaks do not actually hold when plotted against the original ECGs. Therefore, the PVC detection has to be done in another way, even though this seemed to be a good solution.

To detect PVCs and PACs in one algorithm, the shorter-than-average R-R intervals preceding a premature contraction have to be detected. This is done by calculating the time period between each R-R interval and comparing it to the average time, which has already been done in our first code to detect tachycardia. If such a premature contraction has been detected, two things can be done to determine whether it is a PVC.
Firstly, as described above, the QRS complex of a PVC is over 120ms wide, which is why the time of the QRS complex can be calculated and then compared to 120ms by finding the Q peak in the same way the S peak is found in the ST Shift (Q: looking for minima in a certain time interval left of the R peak). As the width of the R peak also tends to be greater than normal, the width can be calculated and compared to a normal value. The width is calculated e.g. by comparing the x-values on both sides of the detected peak where the y-values equal a certain percentage of the peak's value.
Secondly, the time period following the contraction has to be classified as a compensatory pause. This is done by calculating the time between the preceding and the following R peak and comparing it to the duration of two average heartbeats.
If a premature contraction with a QRS complex wider than 120ms and a compensatory pause following has been detected, it can be classified as PVC. Otherwise, the premature contraction is a PAC.

These two arrhythmias have not been implemented in the code.

3.3 ST Shift [corrected]

As mentioned above, the R peaks were actually detected in the wrong place due to our former filters. Therefore, we have rewritten the code to detect the R peaks and neither the squarer nor Hamming window was applied anymore. Nonetheless, the R peaks are still detected in the same way by calculating the average of all peaks and then only classify those as R peaks which are above this average.

To detect an ST shift, the ST segment is compared to the baseline to see whether it is elevated or depressed in respect it. After the bandpass filtering, our baseline is zero, which can also be seen in Figure 9.

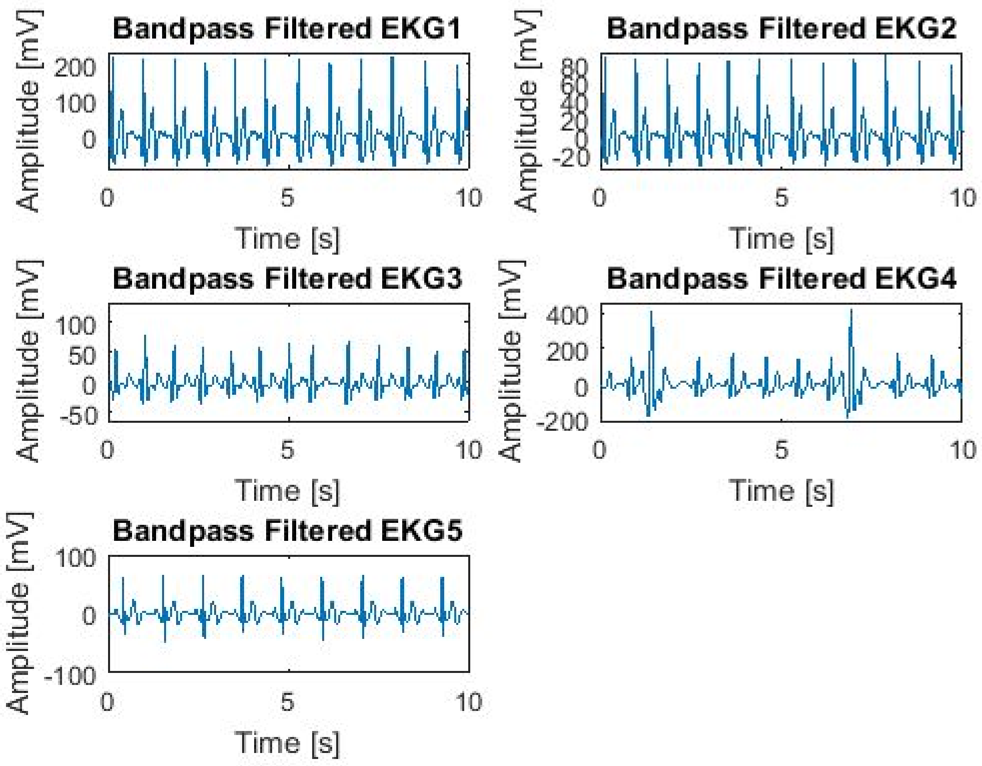

Figure 9: EKG signals bandpass filtered

To identify the ST segment, it is first necessary to detect the end of the S peak and the beginning of the T wave. An easy but not so accurate way is to only detect the S peak by taking a time interval after the R peak in which the S peak normally occurs, search for the minimum in

it, and use it as a starting point. As end point, the same is done with another interval in which the T wave normally occurs, and detect the maximum, which indicates the T wave.
This way also includes half of the S peak and half of the T wave into the ST segment, while it should actually exclude these. The accurate way would be to find the end point of the S peak and the beginning of the T wave, which is more difficult and not done here.

In a normal ECG, the T wave has a duration of 0.16s [7] and occurs roughly 0.15s after the R peak. The time interval is therefore set at [0.16s 0.3s] after the R peak. For the S peak, the interval starts at the next sample after the detected R peak and is roughly as long as the remaining QRS complex. Since the S peak is negative and the used function “findpeaks” only detects maxima, the signal is inverted beforehand.

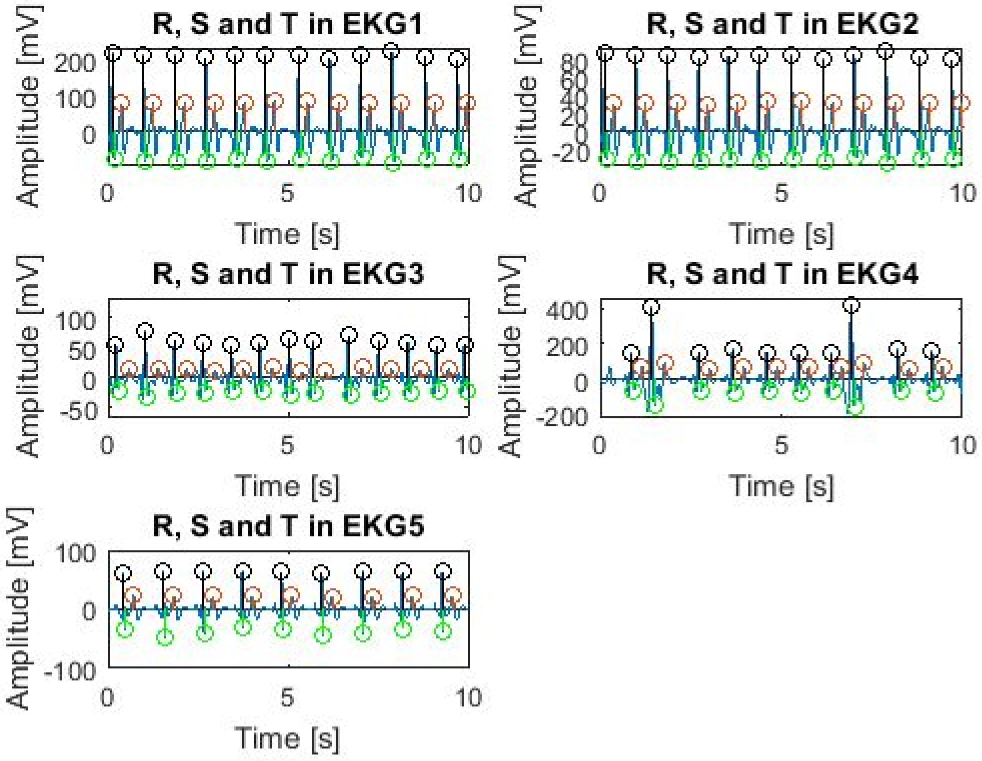

Figure 10: R peaks, S peaks and T waves

It has to be noted that some of these intervals had to be improved by trial and error for an accurate detection.

After the ST segment has been detected, its average can be calculated and compared to the baseline. Should the average be positive, so is the ST segment elevated. Should it be negative, so is the ST segment depressed. As half the S peak and half the T wave are included in the ST segment, calculating an average might skew the results. Therefore, a better solution is to only take the y-value of the x-value exactly in the middle between the S peak and the T maximum. These results are then plotted.

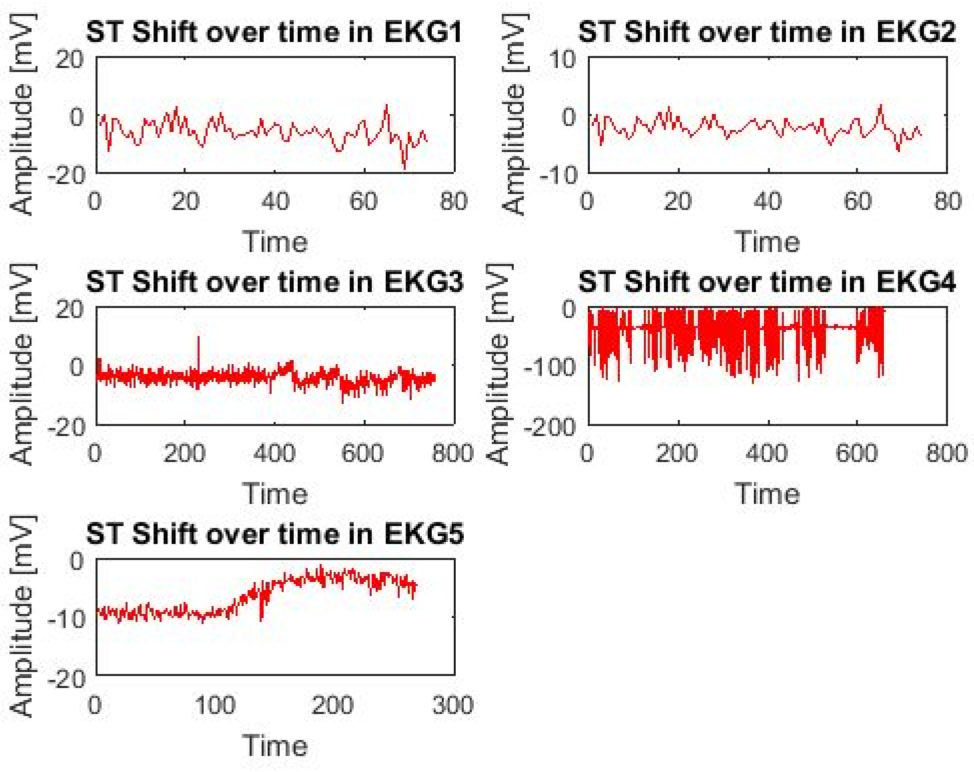

Figure 11: ST Shift middle value over time

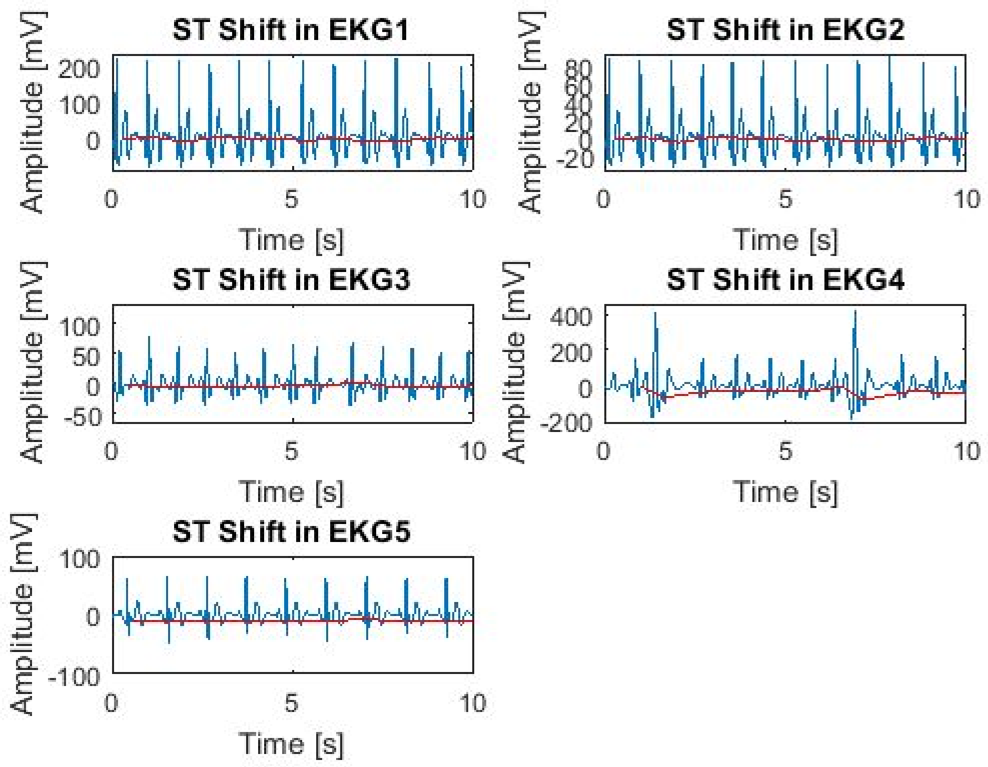

Figure 12: Middle values plotted on ECGs

4. Conclusions [corrected]

There are many things to improve, as we have already stated throughout the report. First of all, this algorithm is very specific to the five ECG signals. It should be modified to be applied to any given signal with different sampling frequencies. We should also have used loops and indices for the signals 1-5.

Additionally, some time intervals and calculated averages had to be modified manually to meet the required accuracy.

As the signal of EKG5 gets hard to handle for the algorithm at some point, we have to decided to cut it off when the amplitude of the T waves exceed the amplitude of the R peaks as it obstructs the proper detection of the following R peaks.

We have not modified the old algorithm regarding the bradycardia and tachycardia except for adding an average and a figure for EKG4 to demonstrate our first solution to detect PVCs. Since we also noticed that our detected R peaks were slightly off, we have rewritten the code including the formerly missing ST shift and added it as Appendix A.

Word count: 2411 including headings and figure descriptions

References

[1] Small Animal Cardiology. http://research.vet.upenn.edu/smallanimalcardiology/ECGTutorial/tabid/4930/Default.aspx. Accessed December 7, 2015

[2] Gerard J. Tortora, Bryan Derrickson. Anatomy & Physiology. 2012. 978-0470-56510-0 (13th Edition)

[3] Healio. http://www.healio.com/cardiology/learn-the-heart/ecg-review/ecg-topic-reviews-and-criteria/premature-atrial-contractions-review. Accessed December 19, 2015

[4] Life in the Fastlane. http://lifeinthefastlane.com/ecg-library/basics/pvc/. Accessed December 19, 2015

[5] TheFreeDictionary. http://medical-dictionary.thefreedictionary.com/compensatory+pause. Accessed December 19, 2015

[6] Life in the Fastlane. http://lifeinthefastlane.com/ecg-library/premature-atrial-complex-pac/. Accessed December 19, 2015

[7] Wikipedia. https://en.wikipedia.org/wiki/Electrocardiography. Accessed December 17, 2015

Appendix A: New Code

```
% clear all windows
close all;
clear all;

% load ECG data
ECG = load('ECG.mat', 'EKG1', 'EKG2', 'EKG3', 'EKG4', 'EKG5');

EK1 = ECG.EKG1;
EK2 = ECG.EKG2;
EK3 = ECG.EKG3;
EK4 = ECG.EKG4;
EK5 = ECG.EKG5;

% Figure 1: Unfiltered ECGs
figure
p1 = subplot(3, 2, 1);
plot([1:length(EK1)]./250, EK1); % transforming the axes into time
title('Original EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p2 = subplot(3, 2, 2);
plot([1:length(EK2)]./250, EK2);
title('Original EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p3 = subplot(3, 2, 3);
plot([1:length(EK3)]./360, EK3);
title('Original EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p4 = subplot(3, 2, 4);
plot([1:length(EK4)]./360, EK4);
title('Original EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p5 = subplot(3, 2, 5);
```

```
plot([1:length(EK5)]./250, EK5);
title('Original EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p1,p2,p3,p4,p5],[0 10 -inf inf]) % displaying time from 0-10s with adjustable y axis
axis 'auto y'

%% Filtering

% cutting off the signal as the T waves become bigger than the R peaks
% at x=295s, where the T wave will be as high as the R peak
EK5 = EK5(1:73750); % 295s * 250 samples/s

D = [4 40]/(250);
F = [4 19]/360;
[y,z] = butter(3,D); % bandpass filter
[t,u] = butter(3,F); % bandpass filter

EK1ST = filtfilt(y, z, EK1);
EK2ST = filtfilt(y, z, EK2);
EK3ST = filtfilt(t, u, EK3);
EK4ST = filtfilt(t, u, EK4);
EK5ST = filtfilt(y, z, EK5);

% Figure 2: Filtered ECGs
figure
p11 = subplot(3, 2, 1);
plot([1:length(EK1ST)]./250, EK1ST);
title('Bandpass Filtered EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p21 = subplot(3, 2, 2);
plot([1:length(EK2ST)]./250, EK2ST);
title('Bandpass Filtered EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p31 = subplot(3, 2, 3);
plot([1:length(EK3ST)]./360, EK3ST);
title('Bandpass Filtered EKG3')
xlabel('Time [s]')
```

```
ylabel('Amplitude [mV]')

p41 = subplot(3, 2, 4);
plot([1:length(EK4ST)]./360, EK4ST);
title('Bandpass Filtered EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p51 = subplot(3, 2, 5);
plot([1:length(EK5ST)]./250, EK5ST);
title('Bandpass Filtered EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p11,p21,p31,p41,p51],[0 10 -inf inf])
axis 'auto y'


%% Peak detection

% R peak detection

% only detects one peak every 0.2s
[y1ST, x1ST] = findpeaks(EK1ST, 'MINPEAKDISTANCE', round(0.2 * 250));
[y2ST, x2ST] = findpeaks(EK2ST, 'MINPEAKDISTANCE', round(0.2 * 250));
[y3ST, x3ST] = findpeaks(EK3ST, 'MINPEAKDISTANCE', round(0.2 * 360));
[y4ST, x4ST] = findpeaks(EK4ST, 'MINPEAKDISTANCE', round(0.2 * 360));
[y5ST, x5ST] = findpeaks(EK5ST, 'MINPEAKDISTANCE', round(0.2 * 250));


% Calculating average as threshold above which all other R peaks should be

peak_average_1 = 0;
for i=1:length(y1ST)
        peak_average_1 = peak_average_1 + y1ST(i);
end
peak_average_1 = peak_average_1/length(y1ST);

peak_average_2 = 0;
for i=1:length(y2ST)
        peak_average_2 = peak_average_2 + y2ST(i);
end
peak_average_2 = peak_average_2/length(y2ST);
```

```
peak_average_3 = 0;
for i=1:length(y3ST)
        peak_average_3 = peak_average_3 + y3ST(i);
end
peak_average_3 = peak_average_3/length(y3ST);

peak_average_4 = 0;
for i=1:length(y4ST)
        peak_average_4 = peak_average_4 + y4ST(i);
end
peak_average_4 = peak_average_4/length(y4ST);

peak_average_5 = 0;
for i=1:length(y5ST)
        peak_average_5 = peak_average_5 + y5ST(i);
end
peak_average_5 = peak_average_5/length(y5ST);
% has to be modified as the T wave grows as big as the R peak with time
peak_average_5 = peak_average_5*1.65;


% Detecting R peaks above threshold

xR1ST=[];
yR1ST=[];
inc = 1;
for i = 1:length(x1ST)
        tmp = EK1ST(x1ST(i));
  if i > 0 && tmp > peak_average_1
        yR1ST(inc) = EK1ST(x1ST(i));
        xR1ST(inc) = x1ST(i);
        inc = inc + 1;
  end
end

xR2ST=[];
yR2ST=[];
inc = 1;
for i = 1:length(x2ST)
        tmp = EK2ST(x2ST(i));
  if i > 0 && tmp > peak_average_2
        yR2ST(inc) = EK2ST(x2ST(i));
```

```
            xR2ST(inc) = x2ST(i);
            inc = inc + 1;
  end
end

xR3ST=[];
yR3ST=[];
inc = 1;
for i = 1:length(x3ST)
            tmp = EK3ST(x3ST(i));
  if i > 0 && tmp > peak_average_3
            yR3ST(inc) = EK3ST(x3ST(i));
            xR3ST(inc) = x3ST(i);
            inc = inc + 1;
  end
end

xR4ST=[];
yR4ST=[];
inc = 1;
for i = 1:length(x4ST)
            tmp = EK4ST(x4ST(i));
  if i > 0 && tmp > peak_average_4
            yR4ST(inc) = EK4ST(x4ST(i));
            xR4ST(inc) = x4ST(i);
            inc = inc + 1;
  end
end

xR5ST=[];
yR5ST=[];
inc = 1;
for i = 1:length(x5ST)
            tmp = EK5ST(x5ST(i));
  if i > 0 && tmp > peak_average_5
            yR5ST(inc) = EK5ST(x5ST(i));
            xR5ST(inc) = x5ST(i);
            inc = inc + 1;
  end
end


% S peak detection
```

```
xS1 = [];
yS1 = [];
yPeaks_S1 = [];
xPeaks_S1 = [];
EK1ST_inv = -EK1ST;
inc = 1;

% only to second last R peak, as the last R peak might have been detected
% but no S peak is following any more
for i = 1:length(xR1ST)-1
        % start and end of interval after R peak
        start_interval = xR1ST(i) + 1; % next sample after R peak
        end_interval = xR1ST(i) + 20; % 20 = 0.08s*250 samples/s (approx)

        % Look for local minimum
        [yPeaks_S1, xPeaks_S1] = findpeaks(EK1ST_inv(start_interval:end_interval));
        if (length(xPeaks_S1) > 0)
        xS1(inc) = xPeaks_S1(1)+start_interval;
        yS1(inc) = -yPeaks_S1(1);
        inc = inc+1;
        end
end

xS2 = [];
yS2 = [];
yPeaks_S2 = [];
xPeaks_S2 = [];
EK2ST_inv = -EK2ST;
inc = 1;

for i = 1:length(xR2ST)-1
        % start and end of interval after R peak
        start_interval = xR2ST(i) + 1; % next sample after R peak
        end_interval = xR2ST(i) + 20; % 20 = 0.08s*250 samples/s (approx)

        % Look for local minimum
        [yPeaks_S2, xPeaks_S2] = findpeaks(EK2ST_inv(start_interval:end_interval));
        if (length(xPeaks_S2) > 0)
        xS2(inc) = xPeaks_S2(1)+start_interval;
        yS2(inc) = -yPeaks_S2(1);
        inc = inc+1;
        end
```

```
end

xS3 = [];
yS3 = [];
yPeaks_S3 = [];
xPeaks_S3 = [];
EK3ST_inv = -EK3ST;
inc = 1;

for i = 1:length(xR3ST)-1
        % start and end of interval after R peak
        start_interval = xR3ST(i) + 1; % next sample after R peak
        end_interval = xR3ST(i) + 30; % 30 = 0.08s*360 samples/s (approx)

        % Look for local minimum
        [yPeaks_S3, xPeaks_S3] = findpeaks(EK3ST_inv(start_interval:end_interval));
        if (length(xPeaks_S3) > 0)
        xS3(inc) = xPeaks_S3(1)+start_interval;
        yS3(inc) = -yPeaks_S3(1);
        inc = inc+1;
        end
end

xS4 = [];
yS4 = [];
yPeaks_S4 = [];
xPeaks_S4 = [];
EK4ST_inv = -EK4ST;
inc = 1;

for i = 1:length(xR4ST)-1
        % start and end of interval after R peak
        start_interval = xR4ST(i) + 1; % next sample after R peak
        end_interval = xR4ST(i) + 40; % 30 = 0.08s*360 samples/s (approx)

        % Look for local minimum
        [yPeaks_S4, xPeaks_S4] = findpeaks(EK4ST_inv(start_interval:end_interval));
        if (length(xPeaks_S4) > 0)
        xS4(inc) = xPeaks_S4(1)+start_interval;
        yS4(inc) = -yPeaks_S4(1);
        inc = inc+1;
        end
end
```

```
xS5 = [];
yS5 = [];
yPeaks_S5 = [];
xPeaks_S5 = [];
EK5ST_inv = -EK5ST;
inc = 1;

for i = 1:length(xR5ST)-1
        % start and end of interval after R peak
        start_interval = xR5ST(i) + 1; % next sample after R peak
        end_interval = xR5ST(i) + 20; % 20 = 0.08s*250 samples/s (approx)

        % Look for local minimum
        [yPeaks_S5, xPeaks_S5] = findpeaks(EK5ST_inv(start_interval:end_interval));
        if (length(xPeaks_S5) > 0)
        xS5(inc) = xPeaks_S5(1)+start_interval;
        yS5(inc) = -yPeaks_S5(1);
        inc = inc+1;
        end
end


% T wave detection

xT1 = [];
yT1 = [];
yPeaks_T1 = [];
xPeaks_T1 = [];
inc = 1;

for i = 1:length(xR1ST)-1
        % start and end of interval after R peak
        start_interval = xR1ST(i) + 40; % 40 = 0.16s*250 samples/s
        end_interval = xR1ST(i) + 75; % 75 = 0.3s*250 samples/s

        % Look for local maximum
        [yPeaks_T1, xPeaks_T1] = findpeaks(EK1ST(start_interval:end_interval));
        if (length(xPeaks_T1) > 0)
        xT1(inc) = xPeaks_T1(1)+start_interval;
        yT1(inc) = yPeaks_T1(1);
        inc = inc+1;
        end
```

```
end

xT2 = [];
yT2 = [];
yPeaks_T2 = [];
xPeaks_T2 = [];
inc = 1;

for i = 1:length(xR2ST)-1
        % start and end of interval after R peak
        start_interval = xR2ST(i) + 40; % 40 = 0.16s*250 samples/s
        end_interval = xR2ST(i) + 75; % 75 = 0.3s*250 samples/s

        % Look for local maximum
        [yPeaks_T2, xPeaks_T2] = findpeaks(EK2ST(start_interval:end_interval));
        if (length(xPeaks_T2) > 0)
        xT2(inc) = xPeaks_T2(1)+start_interval;
        yT2(inc) = yPeaks_T2(1);
        inc = inc+1;
        end
end

xT3 = [];
yT3 = [];
yPeaks_T3 = [];
xPeaks_T3 = [];
inc = 1;

for i = 1:length(xR3ST)-1
        % start and end of interval after R peak
        start_interval = xR3ST(i) + 110; % trial and error
        end_interval = xR3ST(i) + 150; % trial and error

        % Look for local maximum
        [yPeaks_T3, xPeaks_T3] = findpeaks(EK3ST(start_interval:end_interval));
        if (length(xPeaks_T3) > 0)
        xT3(inc) = xPeaks_T3(1)+start_interval;
        yT3(inc) = yPeaks_T3(1);
        inc = inc+1;
        end
end

xT4 = [];
```

```
yT4 = [];
yPeaks_T4 = [];
xPeaks_T4 = [];
inc = 1;

for i = 1:length(xR4ST)-1
        % start and end of interval after R peak
        start_interval = xR4ST(i) + 85; % trial and error
        end_interval = xR4ST(i) + 150; % trial and error

        % Look for local maximum
        [yPeaks_T4, xPeaks_T4] = findpeaks(EK4ST(start_interval:end_interval));
        if (length(xPeaks_T4) > 0)
        xT4(inc) = xPeaks_T4(1)+start_interval;
        yT4(inc) = yPeaks_T4(1);
        inc = inc+1;
        end
end

xT5 = [];
yT5 = [];
yPeaks_T5 = [];
xPeaks_T5 = [];
inc = 1;

for i = 1:length(xR5ST)-1
        % start and end of interval after R peak
        start_interval = xR5ST(i) + 41; % 40 = 0.16s*250 samples/s
        end_interval = xR5ST(i) + 75; % 75 = 0.3s*250 samples/s

        % Look for local maximum
        [yPeaks_T5, xPeaks_T5] = findpeaks(EK5ST(start_interval:end_interval));
        if (length(xPeaks_T5) > 0)
        xT5(inc) = xPeaks_T5(1)+start_interval;
        yT5(inc) = yPeaks_T5(1);
        inc = inc+1;
        end
end


% Figure 3: ST filtered signal and R peaks/S peaks/T waves
figure
p12 = subplot(3, 2, 1);
```

```
plot([1:length(EK1ST)]./250, EK1ST);
title('R, S and T in EKG1');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
stem(xS1./250, yS1, 'g'); % displays S peaks
stem(xT1./250, yT1, 'o'); % displays T wave max
stem(xR1ST./250, yR1ST, 'k'); % displays R peaks
% plot([0 xR1ST(end)], [peak_average_1 peak_average_1]); % plots peak average of EK1ST
hold off

p22 = subplot(3, 2, 2);
plot([1:length(EK2ST)]./250, EK2ST);
title('R, S and T in EKG2');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
stem(xS2./250, yS2, 'g'); % displays S peaks
stem(xT2./250, yT2, 'o'); % displays T wave max
stem(xR2ST./250, yR2ST, 'k'); % displays R peaks
% plot([0 xR2ST(end)], [peak_average_2 peak_average_2]); % plots peak average
hold off

p32 = subplot(3, 2, 3);
plot([1:length(EK3ST)]./360, EK3ST);
title('R, S and T in EKG3');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
stem(xS3./360, yS3, 'g'); % displays S peaks
stem(xT3./360, yT3, 'o'); % displays T wave max
stem(xR3ST./360, yR3ST, 'k'); % displays R peaks
% plot([0 xR3ST(end)], [peak_average_3 peak_average_3]); % plots peak average
hold off

p42 = subplot(3, 2, 4);
plot([1:length(EK4ST)]./360, EK4ST);
title('R, S and T in EKG4');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
stem(xS4./360, yS4, 'g'); % displays S peaks
stem(xT4./360, yT4, 'o'); % displays T wave max
stem(xR4ST./360, yR4ST, 'k'); % displays R peaks
% plot([0 xR4ST(end)], [peak_average_4 peak_average_4]); % plots peak average
hold off

p52 = subplot(3, 2, 5);
plot([1:length(EK5ST)]./250, EK5ST);
title('R, S and T in EKG5');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
stem(xS5./250, yS5, 'g'); % displays S peaks
stem(xT5./250, yT5, 'o'); % displays T wave max
stem(xR5ST./250, yR5ST, 'k'); % displays R peaks
% plot([0 xR5ST(end)], [peak_average_5 peak_average_5]); % plots peak average
hold off

axis([p12,p22,p32,p42,p52],[0 10 -inf inf])
axis 'auto y'


%% ST Shift

% calculating the middle value between S and T

xST1_middle = [];
yST1_middle = [];

for i = 1:length(xS1)
        xST1(i) = xT1(i) - xS1(i); % difference between S and T
        xST1_middle(i) = xS1(i) + round(xST1(i)/2);
        yST1_middle(i) = EK1ST(xST1_middle(i));
end


xST2_middle = [];
yST2_middle = [];

for i = 1:length(xS2)
        xST2(i) = xT2(i) - xS2(i); % difference between S and T
        xST2_middle(i) = xS2(i) + round(xST2(i)/2);
        yST2_middle(i) = EK2ST(xST2_middle(i));
end
```

```
xST3_middle = [];
yST3_middle = [];

for i = 1:length(xS3)
        xST3(i) = xT3(i) - xS3(i); % difference between S and T
        xST3_middle(i) = xS3(i) + round(xST3(i)/2);
        yST3_middle(i) = EK3ST(xST3_middle(i));
end


xST4_middle = [];
yST4_middle = [];

for i = 1:length(xS4)
        xST4(i) = xT4(i) - xS4(i); % difference between S and T
        xST4_middle(i) = xS4(i) + round(xST4(i)/2);
        yST4_middle(i) = EK4ST(xST4_middle(i));
end


xST5_middle = [];
yST5_middle = [];

for i = 1:length(xS5)
        xST5(i) = xT5(i) - xS5(i); % difference between S and T
        xST5_middle(i) = xS5(i) + round(xST5(i)/2);
        yST5_middle(i) = EK5ST(xST5_middle(i));
end


% Figure 4: ST middle value over time
figure
subplot(3, 2, 1);
plot([1:length(xST1_middle)], yST1_middle, 'r');
title('ST Shift over time in EKG1');
xlabel('Time');
ylabel('Amplitude [mV]');

subplot(3, 2, 2);
plot([1:length(xST2_middle)], yST2_middle, 'r');
title('ST Shift over time in EKG2');
xlabel('Time');
```

```
ylabel('Amplitude [mV]');

subplot(3, 2, 3);
plot([1:length(xST3_middle)], yST3_middle, 'r');
title('ST Shift over time in EKG3');
xlabel('Time');
ylabel('Amplitude [mV]');

subplot(3, 2, 4);
plot([1:length(xST4_middle)], yST4_middle, 'r');
title('ST Shift over time in EKG4');
xlabel('Time');
ylabel('Amplitude [mV]');

subplot(3, 2, 5);
plot([1:length(xST5_middle)], yST5_middle, 'r');
title('ST Shift over time in EKG5');
xlabel('Time');
ylabel('Amplitude [mV]');


% Figure 5: Middle value over EKG1
figure
p14 = subplot(3, 2, 1);
plot([1:length(EK1ST)]./250, EK1ST);
title('ST Shift in EKG1');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
plot(xST1_middle./250, yST1_middle, 'r');
% legend('EKG1 filtered', 'Middle values between S and T');
% stem(xST1_middle./250, yST1_middle, 'r');
hold off

p24 = subplot(3, 2, 2);
plot([1:length(EK2ST)]./250, EK2ST);
title('ST Shift in EKG2');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
plot(xST2_middle./250, yST2_middle, 'r');
% legend('EKG2 filtered', 'Middle values between S and T');
% stem(xST2_middle./250, yST2_middle, 'r');
```

```
hold off

p34 = subplot(3, 2, 3);
plot([1:length(EK3ST)]./360, EK3ST);
title('ST Shift in EKG3');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
plot(xST3_middle./360, yST3_middle, 'r');
% legend('EKG3 filtered', 'Middle values between S and T');
% stem(xST3_middle./360, yST3_middle, 'r');
hold off

p44 = subplot(3, 2, 4);
plot([1:length(EK4ST)]./360, EK4ST);
title('ST Shift in EKG4');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
plot(xST4_middle./360, yST4_middle, 'r');
% legend('EKG4 filtered', 'Middle values between S and T');
% stem(xST4_middle./360, yST4_middle, 'r');
hold off

p54 = subplot(3, 2, 5);
plot([1:length(EK5ST)]./250, EK5ST);
title('ST Shift in EKG5');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
plot(xST5_middle./250, yST5_middle, 'r');
% legend('EKG5 filtered', 'Middle values between S and T');
% stem(xST5_middle./250, yST5_middle, 'r');
hold off

axis([p14,p24,p34,p44,p54],[0 10 -inf inf])
axis 'auto y'
```

Appendix B: Old Code

B.1 Arrhythmia detection

```
% clear all windows
close all;
clear all;

% load ECG data
ECG = load('ECG.mat', 'EKG1', 'EKG2', 'EKG3', 'EKG4', 'EKG5');

EK1 = ECG.EKG1;
EK2 = ECG.EKG2;
EK3 = ECG.EKG3;
EK4 = ECG.EKG4;
EK5 = ECG.EKG5;

% Figure 1
figure
p1 = subplot(3, 2, 1);
plot([1:length(EK1)]./250, EK1); % transforming the axes into time
title('Original EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p2 = subplot(3, 2, 2);
plot([1:length(EK2)]./250, EK2);
title('Original EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p3 = subplot(3, 2, 3);
plot([1:length(EK3)]./360, EK3);
title('Original EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
```

```
p4 = subplot(3, 2, 4);
plot([1:length(EK4)]./360, EK4);
title('Original EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p5 = subplot(3, 2, 5);
plot([1:length(EK5)]./250, EK5);
title('Original EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p1,p2,p3,p4,p5],[0 10 -inf inf]) % displaying time from 0-10s with adjustable y axis
axis 'auto y'


% Time will be used at the end to calculate the heart rate
Time_Base_1 = 1/250;
Time_Base_2 = 1/250;
Time_Base_3 = 1/360;
Time_Base_4 = 1/360;
Time_Base_5 = 1/250;


%%Preprocessing -> peak detection

% Low pass filter

[b,a] = butter(6, 15/250, 'low');
[c,d] = butter(6, 5/250, 'high');

[f,e] = butter(6, 15/360, 'low');
[g,h] = butter(6, 5/360, 'high');


F1 = filter(b,a,[1 zeros(1,16)]);
F2 = filter(c,d,[1 zeros(1,32)]);
```

```
F3 = filter(f,e,[1 zeros(1,16)]);
F4 = filter(g,h,[1 zeros(1,32)]);

EK11 = conv (EK1, F1);
EK21 = conv (EK2, F1);
EK31 = conv (EK3, F3);
EK41 = conv (EK4, F3);
EK51 = conv (EK5, F1);

EK11 = conv (EK11, F2);
EK21 = conv (EK21, F2);
EK31 = conv (EK31, F4);
EK41 = conv (EK41, F4);
EK51 = conv (EK51, F2);


D = [5 15]*2/250;
F = [5 15]*2/360;
[y,z] = butter(3,D); % bandpass filter
[t,u] = butter(3,F); % bandpass filter

EK11 = filtfilt(y, z, EK11);
EK21 = filtfilt(y, z, EK21);
EK31 = filtfilt(t, u, EK31);
EK41 = filtfilt(t, u, EK41);
EK51 = filtfilt(y, z, EK51);

% Figure 2
figure
p11 = subplot(3, 2, 1);
plot([1:length(EK11)]./250, EK11);
title('Bandpass Filtered EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p21 = subplot(3, 2, 2);
plot([1:length(EK21)]./250, EK21);
```

```
title('Bandpass Filtered EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p31 = subplot(3, 2, 3);
plot([1:length(EK31)]./360, EK31);
title('Bandpass Filtered EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p41 = subplot(3, 2, 4);
plot([1:length(EK41)]./360, EK41);
title('Bandpass Filtered EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p51 = subplot(3, 2, 5);
plot([1:length(EK51)]./250, EK51);
title('Bandpass Filtered EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p11,p21,p31,p41,p51],[0 10 -inf inf])
axis 'auto y'


% derivative filter of order 10

EK12 = diff(EK11, 10);
EK22 = diff(EK21, 10);
EK32 = diff(EK31, 10);
EK42 = diff(EK41, 10);
EK52 = diff(EK51, 10);

% Figure 3
figure
p12 = subplot(3, 2, 1);
```

```
plot([1:length(EK12)]./250, EK12);
title('BP & Deriv EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p22 = subplot(3, 2, 2);
plot([1:length(EK22)]./250, EK22);
title('BP & Deriv EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p32 = subplot(3, 2, 3);
plot([1:length(EK32)]./360, EK32);
title('BP & Deriv EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p42 = subplot(3, 2, 4);
plot([1:length(EK42)]./360, EK42);
title('BP & Deriv EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p52 = subplot(3, 2, 5);
plot([1:length(EK52)]./250, EK52);
title('BP & Deriv EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p12,p22,p32,p42,p52],[0 10 -inf inf])
axis 'auto y'


% We change the signal to positive to make the activity zone visible

EK13 = (EK12.^2);
EK23 = (EK22.^2);
```

```
EK33 = (EK32.^2);
EK43 = (EK42.^2);
EK53 = (EK52.^2);

% Figure 4
figure
p13 = subplot(3, 2, 1);
plot([1:length(EK13)]./250, EK13);
title('Squared EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p23 = subplot(3, 2, 2);
plot([1:length(EK23)]./250, EK23);
title('Squared EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p33 = subplot(3, 2, 3);
plot([1:length(EK33)]./360, EK33);
title('Squared EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p43 = subplot(3, 2, 4);
plot([1:length(EK43)]./360, EK43);
title('Squared EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p53 = subplot(3, 2, 5);
plot([1:length(EK53)]./250, EK53);
title('Squared EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p13,p23,p53],[0 10 -inf inf])
```

```
axis 'auto y'
axis([p33,p43],[0 10 0 0.0000000002]) % adjusting axes to able to see the peaks


% applying Hamming window

w = hamming(50);
EK14 = conv(EK13, w);
EK24 = conv(EK23, w);
EK34 = conv(EK33, w);
EK44 = conv(EK43, w);
EK54 = conv(EK53, w);

% Figure 5
figure
p14 = subplot(3, 2, 1);
plot([1:length(EK14)]./250, EK14);
title('Sq. Hamming EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p24 = subplot(3, 2, 2);
plot([1:length(EK24)]./250, EK24);
title('Sq. Hamming EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p34 = subplot(3, 2, 3);
plot([1:length(EK34)]./360, EK34);
title('Sq. Hamming EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p44 = subplot(3, 2, 4);
plot([1:length(EK44)]./360, EK44);
title('Sq. Hamming EKG4')
xlabel('Time [s]')
```

```
ylabel('Amplitude [mV]')

p54 = subplot(3, 2, 5);
plot([1:length(EK54)]./250, EK54);
title('Sq. Hamming EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p14,p24,p34,p44,p54],[0 10 -inf inf])
axis 'auto y'

[y1, x1] = findpeaks(EK14, 'MINPEAKDISTANCE', round(0.2 * 250));
[y2, x2] = findpeaks(EK24, 'MINPEAKDISTANCE', round(0.2 * 250));
[y3, x3] = findpeaks(EK34, 'MINPEAKDISTANCE', round(0.2 * 360));
[y4, x4] = findpeaks(EK44, 'MINPEAKDISTANCE', round(0.2 * 360));
[y5, x5] = findpeaks(EK54, 'MINPEAKDISTANCE', round(0.2 * 250));


% Computation of the global average for the threshold

average1 = 0;
average2 = 0;
average3 = 0;
average4 = 0;
average5 = 0;


for i = 1:length(EK14)
        average1 = average1 + EK14(i);
end
average1 = average1 / length(EK14);

for i = 1:length(EK24)
        average2 = average2 + EK24(i);
end
average2 = average2 / length(EK24);
```

```
for i = 1:length(EK34)
        average3 = average3 + EK34(i);
end
average3 = average3 / length(EK34);

for i = 1:length(EK44)
        average4 = average4 + EK44(i);
end
average4 = average4 / length(EK44);

for i = 1:length(EK54)
        average5 = average5 + EK54(i);
end
average5 = average5 / length(EK54);


xM1=[];
yM1=[];
inc = 1;
for i = 1:length(x1)
        tmp = EK14(x1(i));
 if i > 0 && tmp > average1
        yM1(inc) = EK14(x1(i));
        xM1(inc) = x1(i);
        inc = inc + 1;
 end
end

xM2=[];
yM2=[];
inc = 1;
for i = 1:length(x2)
        tmp = EK24(x2(i));
 if i > 0 && tmp > average2
        yM2(inc) = EK24(x2(i));
        xM2(inc) = x2(i);
```

```
            inc = inc + 1;
 end
end

xM3=[];
yM3=[];
inc = 1;
for i = 1:length(x3)
            tmp = EK34(x3(i));
 if i > 0 && tmp > average3
            yM3(inc) = EK34(x3(i));
            xM3(inc) = x3(i);
            inc = inc + 1;
 end
end

xM4=[];
yM4=[];
inc = 1;
for i = 1:length(x4)
            tmp = EK44(x4(i));
 if i > 0 && tmp > average4
            yM4(inc) = EK44(x4(i));
            xM4(inc) = x4(i);
            inc = inc + 1;
 end
end

xM5=[];
yM5=[];
inc = 1;
for i = 1:length(x5)
            tmp = EK54(x5(i));
 if i > 0 && tmp > average5
            yM5(inc) = EK54(x5(i));
            xM5(inc) = x5(i);
```

```
        inc = inc + 1;
 end
end


% Figure 6
figure
p15 = subplot(3, 2, 1);
plot([1:length(EK14)]./250, EK14);
title('R peaks EKG1');
xlabel('Time [s]');
ylabel('Amplitude [mV]');
hold on
s1 = stem(xM1./250, yM1);
hold off

p25 = subplot(3, 2, 2);
plot([1:length(EK24)]./250, EK24);
title('R peaks EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
hold on
s2 = stem(xM2./250, yM2);
hold off

p35 = subplot(3, 2, 3);
plot([1:length(EK34)]./360, EK34);
title('R peaks EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
hold on
s3 = stem(xM3./360, yM3);
hold off

p45 = subplot(3, 2, 4);
plot([1:length(EK44)]./360, EK44);
```

```
title('R peaks EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
hold on
s4 = stem(xM4./360, yM4);
hold off

p55 = subplot(3, 2, 5);
plot([1:length(EK54)]./250, EK54);
title('R peaks EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
hold on
s5 = stem(xM5./250, yM5);
hold off

axis([p15,p25,p35,p45,p55],[0 10 -inf inf])
axis 'auto y'


%% For corrected report: calculating average of detected R peaks

Rpeak_sum4 = 0;

for i=1:length(yM4)
        Rpeak_sum4 = Rpeak_sum4 + yM4(i);
end
Rpeak_average4 = Rpeak_sum4/length(yM4);

% Figure of detected R peaks with half their average (ECG4)
figure
plot([1:length(EK44)]./360, EK44);
title('R peaks EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')
hold on
```

```
stem(xM4./360, yM4);
plot([0 xM4(end)], [Rpeak_average4 Rpeak_average4]./2);
hold off
axis([0 10 -inf inf])
axis 'auto y'


%% Rythme detection

Raverage1 = 0;
bradycardia_cnt1 = 0;
tachycardia_cnt1 = 0;
cmp = 0;
for i = 2:(length(xM1) - 1)
        Raverage1 = Raverage1 - xM1(i - 1) + xM1(i);
        arr1 = xM1(i)-xM1(i-1);
        if (arr1*Time_Base_1 > 1)
                bradycardia_cnt1 = bradycardia_cnt1 + 1;
        elseif (arr1*Time_Base_1 < 0.6)
                tachycardia_cnt1 = tachycardia_cnt1 + 1;
        end
        cmp = cmp + 1;
end
Raverage1 = (Raverage1 * Time_Base_1) / cmp;

if (Raverage1 > 1)
        display('EKG1: Bradycardia');
elseif (Raverage1 < 0.6)
        display('EKG1: Tachycardia');
else
        display('EKG1: No Arrythmia');
end

display(bradycardia_cnt1); % displaying how many bradycardia incidents there are
display(tachycardia_cnt1); % displaying how many tachycardia incidents there are

if (bradycardia_cnt1 > 0);
```

```
	display('EKG1: Bradycardia detected based on counts')
end
if (tachycardia_cnt1 > 0);
	display('EKG1: Tachycardia detected based on counts')
end


Raverage2 = 0;
bradycardia_cnt2 = 0;
tachycardia_cnt2 = 0;
cmp = 0;
for i = 2:(length(xM2) - 1)
	Raverage2 = Raverage2 - xM2(i - 1) + xM2(i);
	arr2 = xM2(i)-xM2(i-1);
	if (arr2*Time_Base_2 > 1)
		bradycardia_cnt2 = bradycardia_cnt2 + 1;
	elseif (arr2*Time_Base_2 < 0.6)
		tachycardia_cnt2 = tachycardia_cnt2 + 1;
	end
	cmp = cmp + 1;
end
Raverage2 = (Raverage2 * Time_Base_2) / cmp;

if (Raverage2 > 1)
	display('EKG2: Bradycardia');
elseif (Raverage2 < 0.6)
	display('EKG2: Tachycardia');
else
	display('EKG2: No Arrythmia');
end
display(bradycardia_cnt2);
display(tachycardia_cnt2);

if (bradycardia_cnt2 > 0);
	display('EKG2: Bradycardia detected based on counts')
end
```

```
if (tachycardia_cnt2 > 0);
        display('EKG2: Tachycardia detected based on counts')
end


Raverage3 = 0;
bradycardia_cnt3 = 0;
tachycardia_cnt3 = 0;
cmp = 0;
for i = 2:(length(xM3) - 1)
        Raverage3 = Raverage3 - xM3(i - 1) + xM3(i);
        arr3 = xM3(i)-xM3(i-1);
        if (arr3*Time_Base_3 > 1)
                bradycardia_cnt3 = bradycardia_cnt3 + 1;
        elseif (arr3*Time_Base_3 < 0.6)
                tachycardia_cnt3 = tachycardia_cnt3 + 1;
        end
        cmp = cmp + 1;
end
Raverage3 = (Raverage3 * Time_Base_3) / cmp;

if (Raverage3 > 1)
        display('EKG3: Bradycardia');
elseif (Raverage3 < 0.6)
        display('EKG3: Tachycardia');
else
        display('EKG3: No Arrythmia');
end
display(bradycardia_cnt3);
display(tachycardia_cnt3);

if (bradycardia_cnt3 > 0);
        display('EKG3: Bradycardia detected based on counts')
end
if (tachycardia_cnt3 > 0);
        display('EKG3: Tachycardia detected based on counts')
```

```
end


Raverage4 = 0;
bradycardia_cnt4 = 0;
tachycardia_cnt4 = 0;
cmp = 0;
for i = 2:(length(xM4) - 1)
        Raverage4 = Raverage4 - xM4(i - 1) + xM4(i);
        arr4 = xM4(i)-xM4(i-1);
        if (arr4*Time_Base_4 > 1)
                bradycardia_cnt4 = bradycardia_cnt4 + 1;
        elseif (arr4*Time_Base_4 < 0.6)
                tachycardia_cnt4 = tachycardia_cnt4 + 1;
        end
        cmp = cmp + 1;
end
Raverage4 = (Raverage4 * Time_Base_4) / cmp;

if (Raverage4 > 1)
        display('EKG4: Bradycardia');
elseif (Raverage4 < 0.6)
        display('EKG4: Tachycardia');
else
        display('EKG4: No Arrythmia');
end
display(bradycardia_cnt4);
display(tachycardia_cnt4);

if (bradycardia_cnt4 > 0);
        display('EKG4: Bradycardia detected based on counts')
end
if (tachycardia_cnt4 > 0);
        display('EKG4: Tachycardia detected based on counts')
end
```

```
Raverage5 = 0;
bradycardia_cnt5 = 0;
tachycardia_cnt5 = 0;
cmp = 0;
for i = 2:(length(xM5) - 1)
        Raverage5 = Raverage5 - xM5(i - 1) + xM5(i);
        arr5 = xM5(i)-xM5(i-1);
        if (arr5*Time_Base_5 > 1)
                bradycardia_cnt5 = bradycardia_cnt5 + 1;
        elseif (arr5*Time_Base_5 < 0.6)
                tachycardia_cnt5 = tachycardia_cnt5 + 1;
        end
        cmp = cmp + 1;
end
Raverage5 = (Raverage5 * Time_Base_5) / cmp;

if (Raverage5 > 1)
        display('EKG5: Bradycardia');
elseif (Raverage5 < 0.6)
        display('EKG5: Tachycardia');
else
        display('EKG5: No Arrythmia');
end
display(bradycardia_cnt5);
display(tachycardia_cnt5);

if (bradycardia_cnt5 > 0);
        display('EKG5: Bradycardia detected based on counts')
end
if (tachycardia_cnt5 > 0);
        display('EKG5: Tachycardia detected based on counts')
end
```

B.2 ST Shift

```
% clear all windows
close all;
clear all;

% load ECG data
ECG = load('ECG.mat', 'EKG1', 'EKG2', 'EKG3', 'EKG4', 'EKG5');


EK1 = ECG.EKG1;
EK2 = ECG.EKG2;
EK3 = ECG.EKG3;
EK4 = ECG.EKG4;
EK5 = ECG.EKG5;


Time_Base_1 = 1/250;
Time_Base_2 = 1/250;
Time_Base_3 = 1/360;
Time_Base_4 = 1/360;
Time_Base_5 = 1/250;

% leaving out the butterworth filter to get a better image

D = [4 40]/(250);
F = [4 40]/360;
[y,z] = butter(3,D); % bandpass filter
[t,u] = butter(3,F); % bandpass filter

EK11 = filtfilt(y, z, EK1);
EK21 = filtfilt(y, z, EK2);
EK31 = filtfilt(t, u, EK3);
EK41 = filtfilt(t, u, EK4);
EK51 = filtfilt(y, z, EK5);

% Figure 8
figure
p11 = subplot(3, 2, 1);
```

```
plot([1:length(EK11)]./250, EK11);
title('Bandpass Filtered EKG1')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p21 = subplot(3, 2, 2);
plot([1:length(EK21)]./250, EK21);
title('Bandpass Filtered EKG2')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p31 = subplot(3, 2, 3);
plot([1:length(EK31)]./360, EK31);
title('Bandpass Filtered EKG3')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p41 = subplot(3, 2, 4);
plot([1:length(EK41)]./360, EK41);
title('Bandpass Filtered EKG4')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

p51 = subplot(3, 2, 5);
plot([1:length(EK51)]./250, EK51);
title('Bandpass Filtered EKG5')
xlabel('Time [s]')
ylabel('Amplitude [mV]')

axis([p11,p21,p31,p41,p51],[0 10 -inf inf])
axis 'auto y'
```