Introduction:

This lab is based around an algorithm developed by Pan and Tompkins that allows for the detection of the QRS complex from an ECG. The algorithm accomplishes this identification based off the slope, amplitude and width at various filtering stages [1]. The first stage is a bandpass filter that increases the signal to noise ratio by removing muscle noise, 60 Hz powerline interference (high frequency) and baseline drift (low frequency) from the signal [1]. The second stage is a derivative filter which captures the slope of the signal [1]. This highlights the QRS complex as it is the part of the signal with the highest rate of change. The third stage is the squaring operation which amplifies higher values and suppresses the lower values [1]. The last stage is the moving average filter which will smooth out the signal outputted by the squaring operation [1]. The output of this signal will allow for us to accomplish the lab's goals which is to identify RR intervals as well as QRS complex width. Being able to do this is a powerful method to be used for analysis of electrocardiogram signals.

Results:

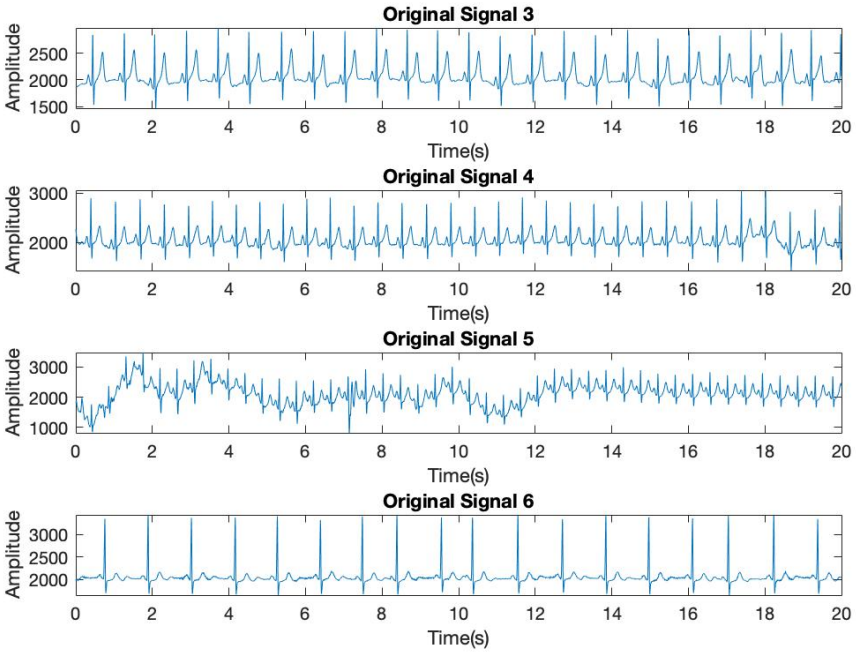


Figure 1: Original ECG signals

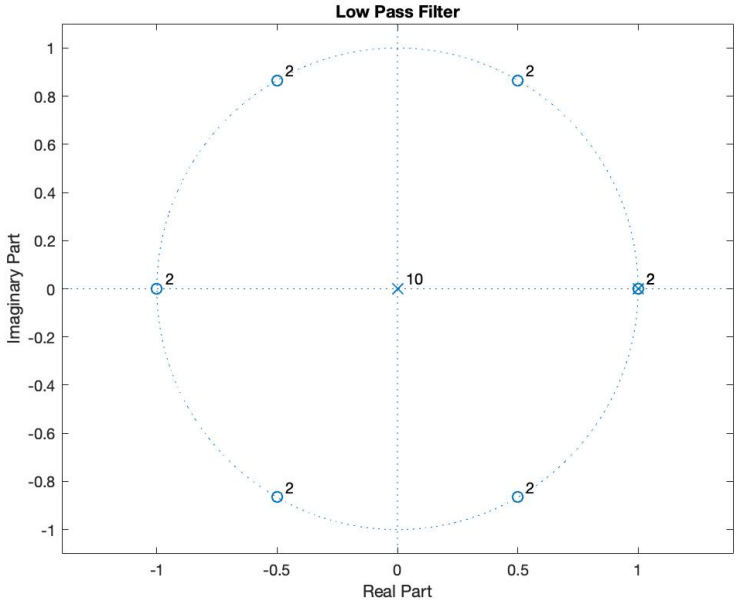


Figure 2: Pole zero plot for low pass filter

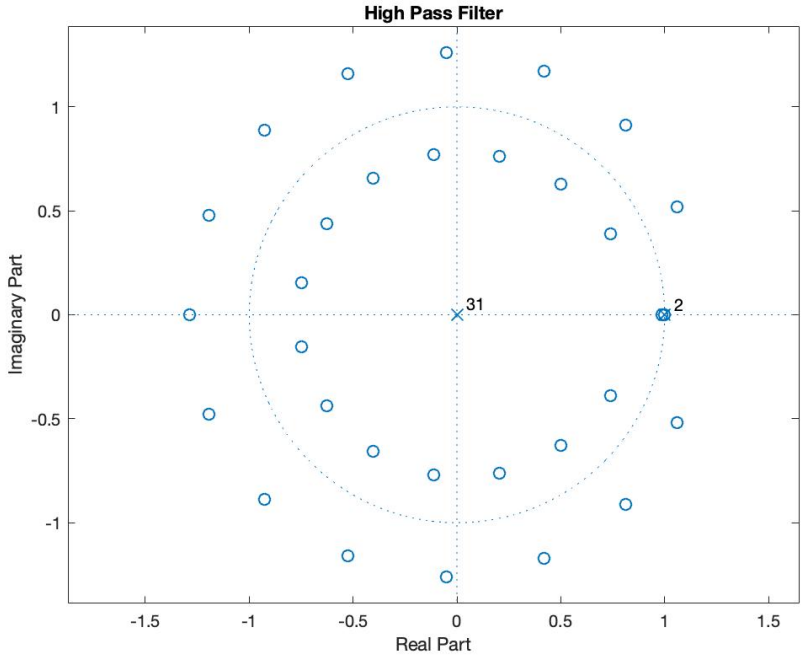


Figure 3: Pole zero plot for high pass filter

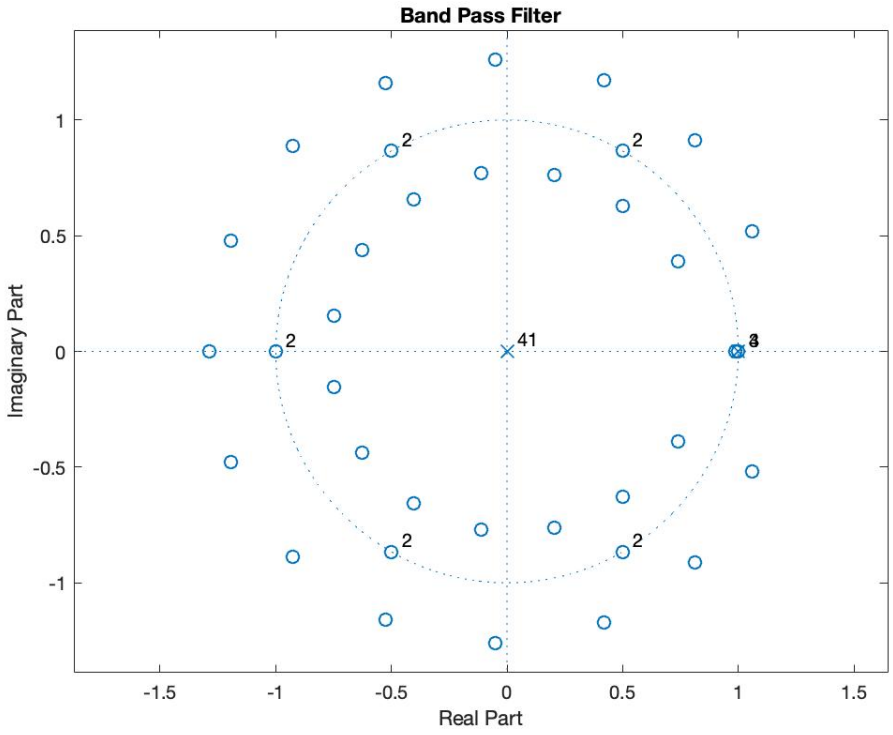


Figure 4: Pole zero plot for band pass filter

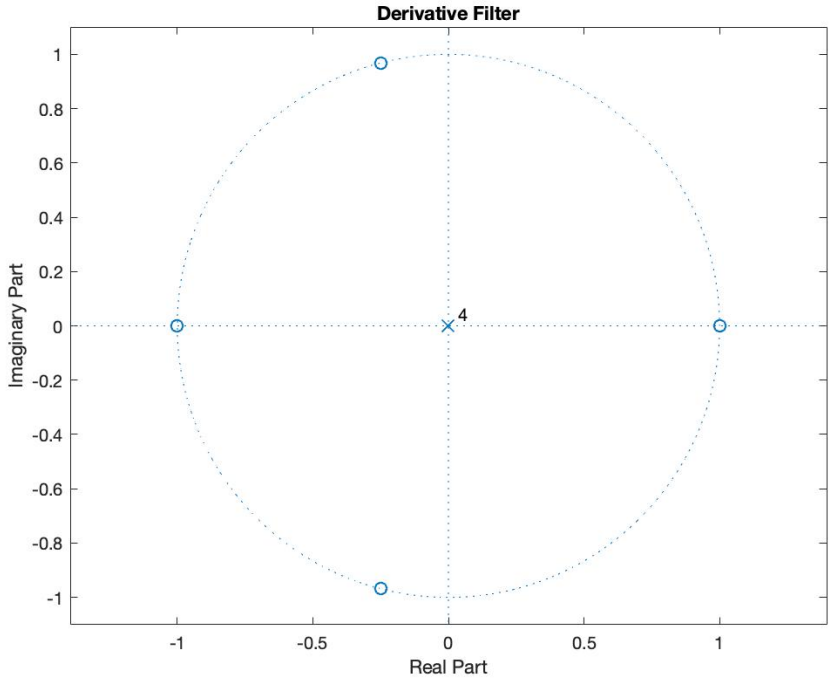


Figure 5: Pole zero plot for derivative filter

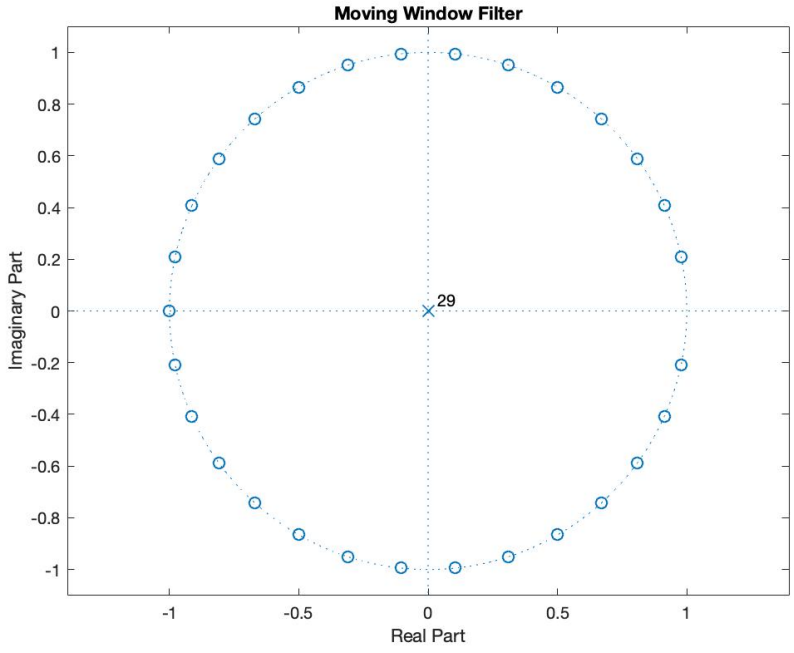


Figure 6: Pole zero plot for moving window filter

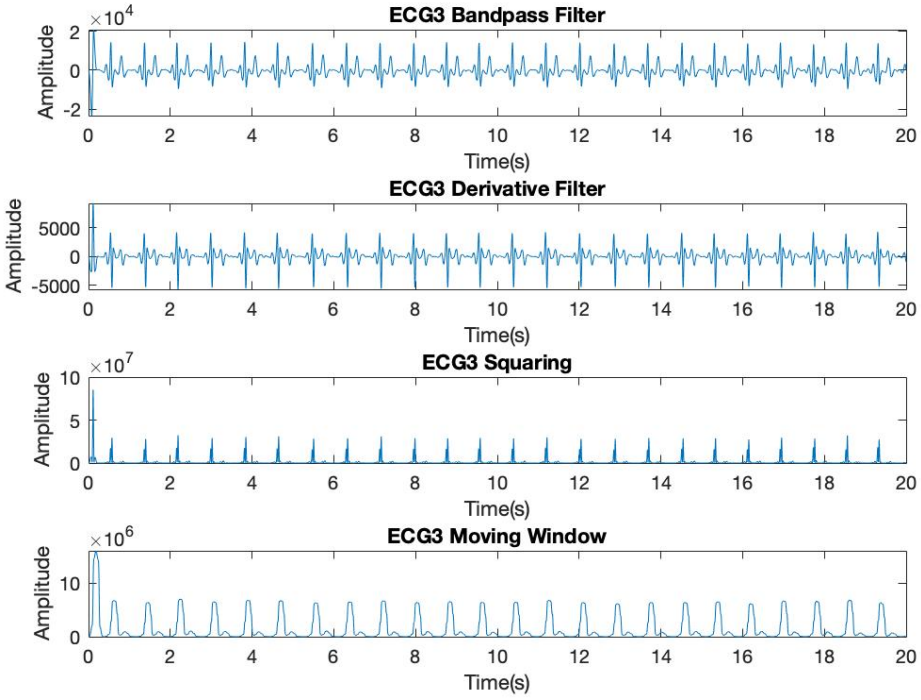


Figure 7: Outputs after each filtering stage for ECG3

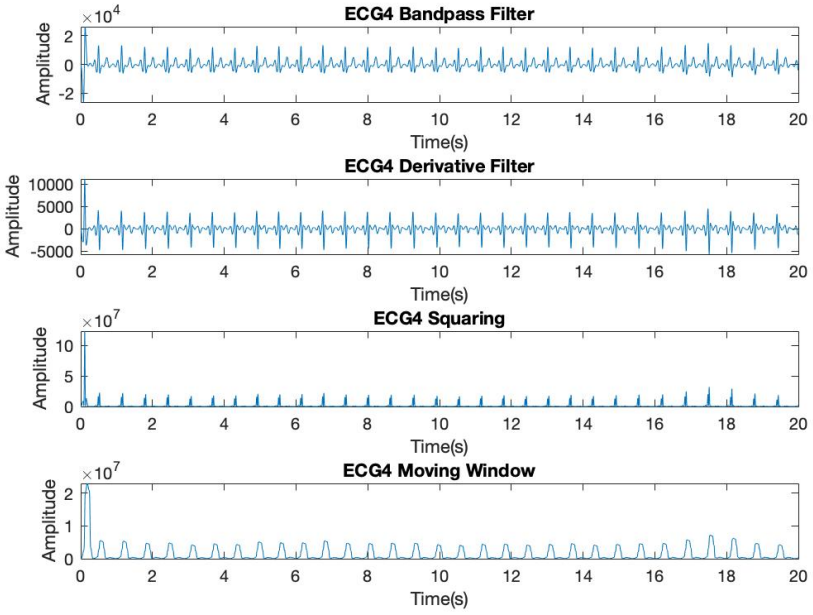


Figure 8: Outputs after each filtering stage for ECG4

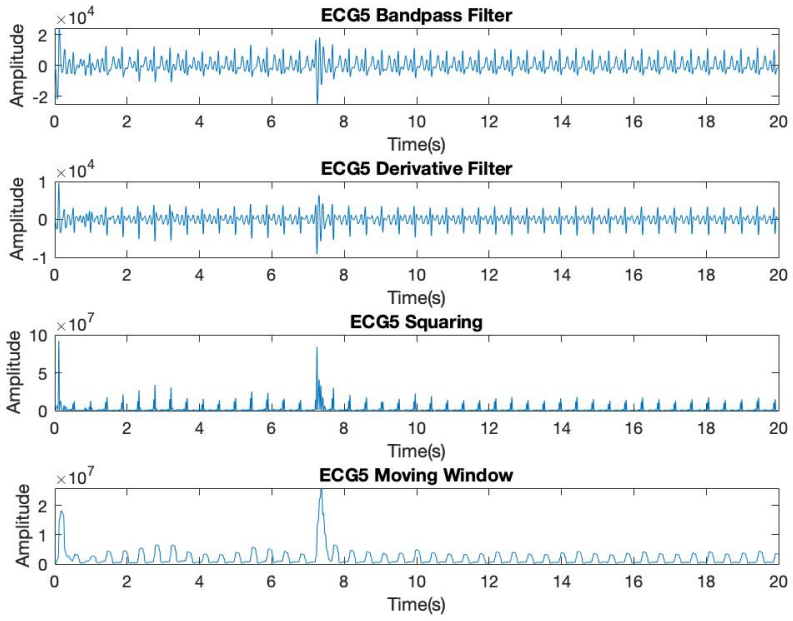
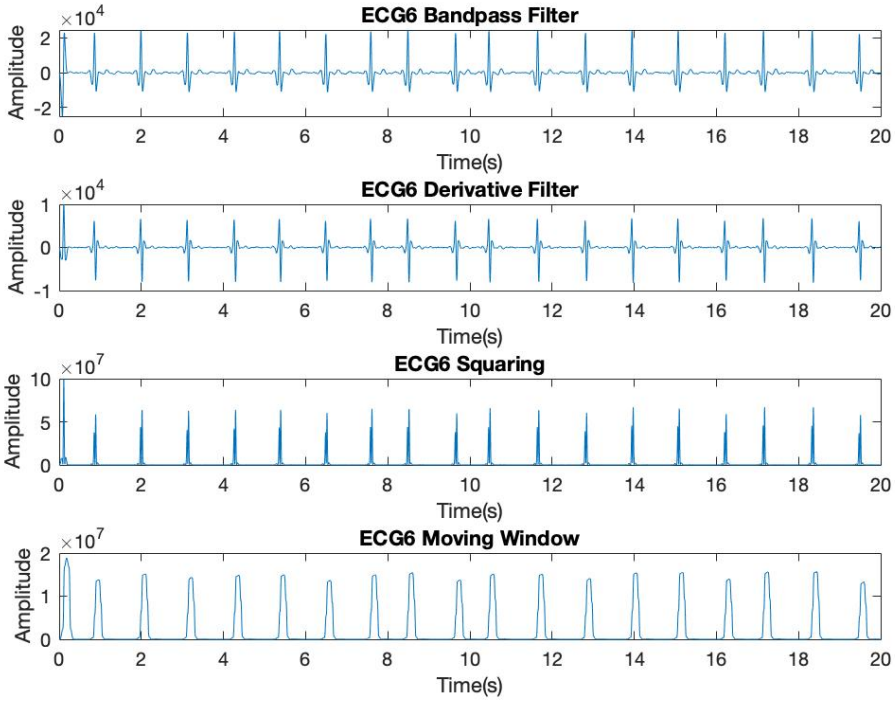


Figure 9: Outputs after each filtering stage for ECG5



Filter 10: Outputs after each filtering stage for ECG6

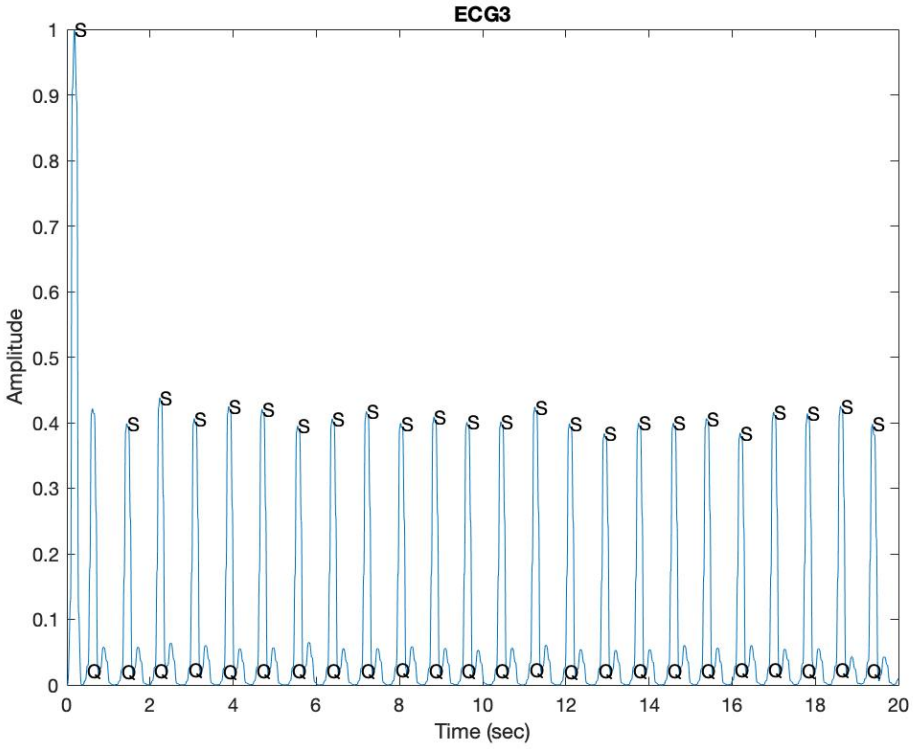


Figure 11: ECG3, Q and S waves labeled

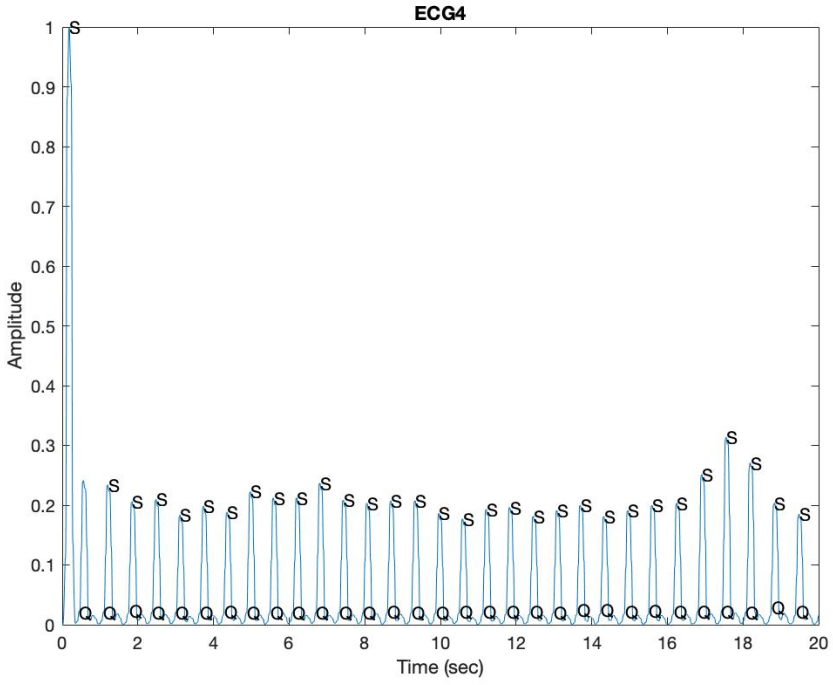


Figure 12: ECG4, Q and S waves labeled

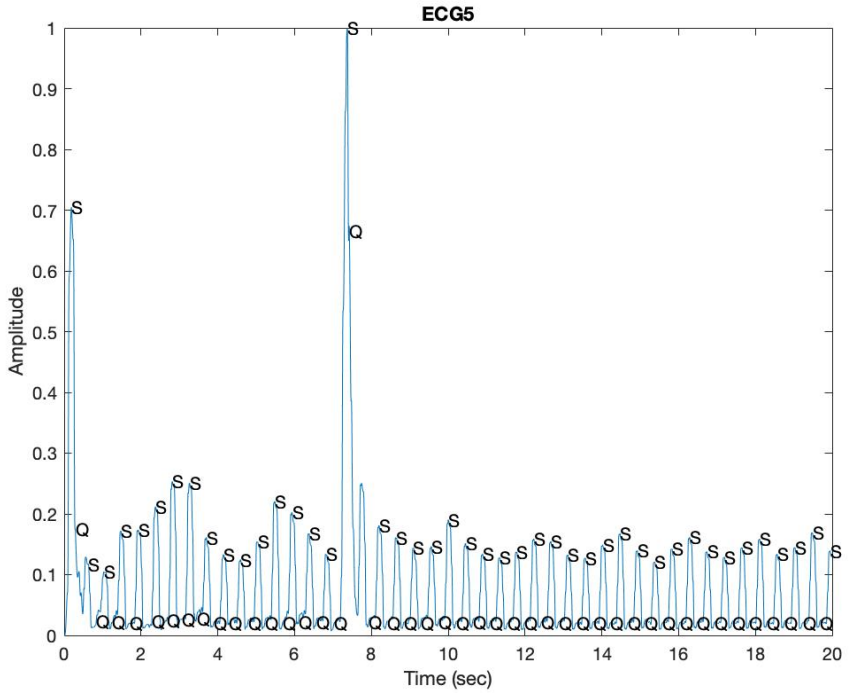


Figure 13: ECG5, Q and S waves labeled

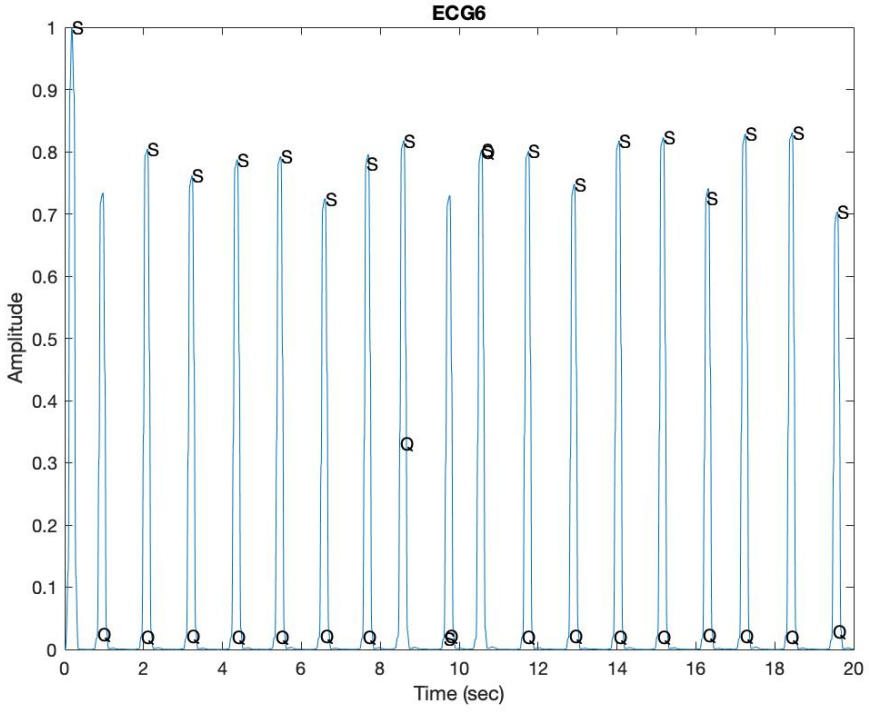


Figure 14: ECG6, Q and S waves labeled

Table 1: ECG parameters for all 4 signals

	ECG3	**ECG4**	**ECG5**	**ECG6**
Avg QRS width (ms)	119.79	86.77	81.02	77.78
Avg RR interval (ms)	815.96	630.5	455.68	1095
Standard deviation RR interval (ms)	19.11	12.52	90.17	106.01
Number of beats	24	31	45	18
BPM	95.16	95.16	134.65	54.78

Conclusion:

By applying Pan Tompkins method, it allowed for us to isolate the QRS complex of ECGs and perform an analysis upon them. In particular it allowed for identification of the time point for the Q, R and S waves in the signal. Once those three different time points were identified for every heartbeat in all 4 ECGs, all was left was to do some calculations to acquire the features of interest. To calculate BPM the average RR interval was inverted and multiplied by 60. To find the average RR interval the difference between consecutive R peaks and then the mean was taken for all intervals. To find the average QRS complex width, once the Q and S waves were identified and the time point of the Q wave was subtracted from the time point S wave. Once this was done for every consecutive complex it was averaged. With respect to the four ECG signals, they all exhibited different types of noise. They all had a DC offset built into the signal, ECG5 also experienced baseline wander possibly due to motion artifacts or breathing etc. ECG3 had a very pronounced T wave which could have contributed to a false R wave identification. ECG5 as well had big disturbance around 7.2 seconds. Some advantages of the Pan Tompkins method is that it allows for proper identification of time points of the QRS complex but the amplitude of these is lost in the process. A disadvantage would be the inability to identify anything to do with the P and T waves to analyze for an pathologies associated with them.

Appendix:

MAIN

```
%% Lab 3: QRS Detection and ECG Rhythm Analysis
% BME 772 Biomedical Signal Analysis
% Andrew Mullen

close all;
clear all;
clc;

%% Preprocessing
% Load Signals
ECG3 = load('ECG3.txt');
ECG4 = load('ECG4.txt');
ECG5 = load('ECG5.txt');
ECG6 = load('ECG6.txt');

% Create Time Vector
fs = 200;
time = 0:length(ECG3)-1;
time = time./fs;

% Plot Original Signals
subplot(411);
plot(time, ECG3); title('Original Signal 3');
xlabel('Time(s)'); ylabel('Amplitude');

subplot(412);
plot(time, ECG4); title('Original Signal 4');
xlabel('Time(s)'); ylabel('Amplitude');

subplot(413);
plot(time, ECG5); title('Original Signal 5');
xlabel('Time(s)'); ylabel('Amplitude');

subplot(414);
plot(time, ECG6); title('Original Signal 6');
xlabel('Time(s)'); ylabel('Amplitude');

% Filter and plot Signals
ECG3_filter = Lab3Filter(ECG3, time, '3');
ECG4_filter = Lab3Filter(ECG4, time, '4');
ECG5_filter = Lab3Filter(ECG5, time, '5');
ECG6_filter = Lab3Filter(ECG6, time, '6');

%% Pole Zero Plots
% Low Pass Filter Coefficients
b_low = [1,0,0,0,0,0,-2,0,0,0,0,0,1];
a_low = [1,-2,1,0,0,0,0,0,0,0,0,0,0];
figure;
zplane(b_low, a_low);
title('Low Pass Filter');

% High Pass Filter Coefficients
```

```
b_high = [-1/32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1/32];
a_high = [1,-1];
figure;
zplane(b_high, a_high);
title('High Pass Filter');

% Band Pass Filter Coefficients
b_band = conv(b_low, b_high);
a_band = conv(a_low, a_high);
figure;
zplane(b_band, a_band);
title('Band Pass Filter');

% Derivate Filter Coefficients
b_der = [0.25, 0.125, 0, -0.125, -0.25];
a_der = [1,0,0,0,0];
figure;
zplane(b_der, a_der);
title('Derivative Filter');

% Moving Window Filter Coefficients
b_mw = ones(1,30)./30;
a_mw = 1;
figure;
zplane(b_mw, a_mw);
title('Moving Window Filter');

%% Visualize Single Heart Beat
org = ECG3(770:929);
band = ECG3_filter(770:929, 1);
der = ECG3_filter(770:929, 2);
sq = ECG3_filter(770:929, 3);
mw = ECG3_filter(770:929, 4);

figure;
subplot(511); plot(org); title('Original ECG');
subplot(512); plot(band); title('Bandpass Filtered ECG');
subplot(513); plot(der); title('Derivative filtered ECG');
subplot(514); plot(sq); title('Squaring');
subplot(515); plot(mw); title('Moving Window');


%% Feature Extraction
% R peak indexing
[ECG3_R_index, ECG3_pulse] = RpeakIndexing(ECG3_filter(:,2), 2000);
[ECG4_R_index, ECG4_pulse] = RpeakIndexing(ECG4_filter(:,2), 2000);
[ECG5_R_index, ECG5_pulse] = RpeakIndexing(ECG5_filter(:,2), 2000);
[ECG6_R_index, ECG6_pulse] = RpeakIndexing(ECG6_filter(:,2), 2000);

%% # of beats, BPM, RR interval, std RR interval
[ECG3_num_beat, ECG3_BPM, ECG3_RR, ECG3_std_RR] = ECG_Data(ECG3_R_index);
[ECG4_num_beat, ECG4_BPM, ECG4_RR, ECG4_std_RR] = ECG_Data(ECG4_R_index);
[ECG5_num_beat, ECG5_BPM, ECG5_RR, ECG5_std_RR] = ECG_Data(ECG5_R_index);
[ECG6_num_beat, ECG6_BPM, ECG6_RR, ECG6_std_RR] = ECG_Data(ECG6_R_index);
```

```
%% Q detection
[ECG3_Q_val, ECG3_Q_loc] = Qidentification(ECG3_filter(:,4), ECG3_RR);
[ECG4_Q_val, ECG4_Q_loc] = Qidentification(ECG4_filter(:,4), ECG4_RR);
[ECG5_Q_val, ECG5_Q_loc] = Qidentification(ECG5_filter(:,4), ECG5_RR);
[ECG6_Q_val, ECG6_Q_loc] = Qidentification(ECG6_filter(:,4), ECG6_RR);

%% S detection
[ECG3_S_val, ECG3_S_loc] = Sidentification(ECG3_filter(:,4), ECG3_RR);
[ECG4_S_val, ECG4_S_loc] = Sidentification(ECG4_filter(:,4), ECG4_RR);
[ECG5_S_val, ECG5_S_loc] = Sidentification(ECG5_filter(:,4), ECG5_RR);
[ECG6_S_val, ECG6_S_loc] = Sidentification(ECG6_filter(:,4), ECG6_RR);

%% QRS Duration
ECG3_QRS = mean(ECG3_S_loc - ECG3_Q_loc')*5;
ECG4_QRS = mean(ECG4_S_loc - ECG4_Q_loc')*5;
ECG5_QRS = mean(ECG5_S_loc - ECG5_Q_loc')*5;
ECG6_QRS = mean(ECG6_S_loc - ECG6_Q_loc')*5;

%% Plot MA and label Q and S wave
time = 0:length(ECG3)-1;
time = time./fs;
norm = (ECG3_filter(:,4)-min(ECG3_filter(:,4)))/(max(ECG3_filter(:,4))-
min(ECG3_filter(:,4)));
figure;
plot(time, norm);
xlabel('Time (sec)'); ylabel('Amplitude'); title('ECG3');
for i = 1:length(ECG3_Q_loc)
    text(ECG3_Q_loc(i)/fs, ECG3_Q_val(i), 'Q');
    text(ECG3_S_loc(i)/fs, ECG3_S_val(i), 'S');
end

norm = (ECG4_filter(:,4)-min(ECG4_filter(:,4)))/(max(ECG4_filter(:,4))-
min(ECG4_filter(:,4)));
figure;
plot(time, norm);
xlabel('Time (sec)'); ylabel('Amplitude'); title('ECG4');
for i = 1:length(ECG4_Q_loc)
    text(ECG4_Q_loc(i)/fs, ECG4_Q_val(i), 'Q');
    text(ECG4_S_loc(i)/fs, ECG4_S_val(i), 'S');
end

norm = (ECG5_filter(:,4)-min(ECG5_filter(:,4)))/(max(ECG5_filter(:,4))-
min(ECG5_filter(:,4)));
figure;
plot(time, norm);
xlabel('Time (sec)'); ylabel('Amplitude'); title('ECG5');
for i = 1:length(ECG5_Q_loc)
    text(ECG5_Q_loc(i)/fs, ECG5_Q_val(i), 'Q');
    text(ECG5_S_loc(i)/fs, ECG5_S_val(i), 'S');
end

norm = (ECG6_filter(:,4)-min(ECG6_filter(:,4)))/(max(ECG6_filter(:,4))-
min(ECG6_filter(:,4)));
figure;
plot(time, norm);
xlabel('Time (sec)'); ylabel('Amplitude'); title('ECG6');
for i = 1:length(ECG6_Q_loc)
```

```
    text(ECG6_Q_loc(i)/fs, ECG6_Q_val(i), 'Q');
    text(ECG6_S_loc(i)/fs, ECG6_S_val(i), 'S');
end
```

Lab3Filter.m function

```
function [ECG_output] = Lab3Filter(ECG_input, time, num)
% Applies all the filters applicable to Lab 3
% Bandpass, Derivative, Squaring, Moving Window
    % Inputs - ECG signal and its number
    % Output - Filtered Signals (4000, 4)
        % Order is 1-Bandpass 2-Derivative 3-Sqaring 4-Moving Window

    % Preallocate ECG_Output
    ECG_output = zeros(length(ECG_input), 4);

    % Low Pass Filter Coefficients
    b_low = [1,0,0,0,0,0,-2,0,0,0,0,0,1];
    a_low = [1,-2,1,0,0,0,0,0,0,0,0,0,0];

    % High Pass Filter Coefficients
    b_high = [-1/32,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,-
1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1/32];
    a_high = [1,-1];

    % Band Pass Filter Coefficients
    b_band = conv(b_low, b_high);
    a_band = conv(a_low, a_high);

    % Derivate Filter Coefficients
    b_der = [0.25, 0.125, 0, -0.125, -0.25];
    a_der = [1,0,0,0,0];

    % Moving Window Filter Coefficients
    b_mw = ones(1,30)./30;
    a_mw = 1;

    % Filtering
    ECG_output(:,1) = filter(b_band, a_band, ECG_input); % Band Pass applied
to ECG input
    ECG_output(:,2) = filter(b_der, a_der, ECG_output(:,1)); % Derivative
applied to bandpass output
    ECG_output(:,3) = ECG_output(:,2).^2; % Squaring applied to derivative
output
    ECG_output(:,4) = filter(b_mw, a_mw, ECG_output(:,3)); % Moving Window
applied to squaring output

    % Plot
    figure;
    subplot(411);
    plot(time, ECG_output(:,1)); title(['ECG',num,' Bandpass Filter']);
    xlabel('Time(s)'); ylabel('Amplitude');

    subplot(412);
    plot(time, ECG_output(:,2)); title(['ECG',num,' Derivative Filter']);
    xlabel('Time(s)'); ylabel('Amplitude');
```

```
    subplot(413);
    plot(time, ECG_output(:,3)); title(['ECG',num,' Squaring']);
    xlabel('Time(s)'); ylabel('Amplitude');

    subplot(414);
    plot(time, ECG_output(:,4)); title(['ECG',num,' Moving Window']);
    xlabel('Time(s)'); ylabel('Amplitude');
end
```

RpeakIndexing.m function

```
function [index, pulse_train] = RpeakIndexing(signal, threshold)
%RPEAKINDEXING Summary of this function goes here
   % Input - Derivative filtered version of ECG
   %       - Initial value of the R peak threshold

   sig_length = length(signal);
   i = 50;
   index = [];

   while i <= sig_length
      if signal(i) > threshold
          if signal(i) < signal(i-1) && signal(i) > signal(i+1)
              index = [index, i];
              i = i+40;
          end
      end
      i = i+1;
   end
   pulse_train =  zeros(sig_length, 1);
   pulse_train(index) = 3500;

end
```

ECG_Data.m function

```
function [num_beats, BPM, RR, std_RR] = ECG_Data(indices)
%ECG_DATA Summary of this function goes here
    % Input - indices - array of the indexes where R peak locations
    %           occur
    % Output - num_beats = number of beats in the signal
    %        - BPM = beats per minute of the individual
    %        - RR = Average time length between R peaks
    %        - std_RR

    % Number of beat calculations
    num_beats = length(indices);

    % Calculate time period between R peaks
```

```
    for i = 2:length(indices)
        RR_period(i-1) = (indices(i) - indices(i-1));
    end

    RR_period = RR_period.*(1000/200);

    RR = mean(RR_period);
    std_RR = std(RR_period);
    BPM = (60/RR)*1000;

end
```

Qidentification.m function

```
function [Q_val, Q_loc] = Qidentification(MA,RR)
%QIDENTIFICATION Summary of this function goes here
    MA = (MA - min(MA))./(max(MA)-min(MA));
    i = 50;
    j = 1;

    while i < 3999

       if MA(i) > 0.02
           if MA(i-1) < MA(i)
               Q_loc(j) = i;
               Q_val(j) = MA(i);
               i = i + round(0.85*RR/5);
               j = j+1;
           end
       end
       i = i+1;
    end



end
```

Sidentification.m function

```
function [pks, locs] = Sidentification(MA, RR)
%SIDENTIFICATION Summary of this function goes here
%   Detailed explanation goes here
    MA = (MA - min(MA))./(max(MA)-min(MA));
    [pks, locs] = findpeaks(MA,'MinPeakDistance',0.86*RR/5);
end
```

QRSduration.m function

```
function [QRS] = QRSduration(Q, S)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
    QRS = S - Q';
    QRS = mean(QRS)*5;


end
```

References

[1] “Lab 3: QRS Detection and ECG Rhythm Analysis”, BME772, Ryerson University, 2018.