Final Project:
Atrial Fibrillation Detection Using Entropy

Natalie Tipton

Grand Valley State University
EGR 534 & EGR 635
Dr. Samhita Rhodes

December 6, 2019

Abstract

Atrial fibrillation is a cardiac dysrhythmia where the atria do not contract properly. It affects more people than any other dysrhythmia, and accurate detection of it is incredibly important for their well being. This algorithm looks at entropy, or uncertainty, in normal sinus rhythms versus signals with atrial fibrillation as well as signals with other dysrhythmias. Analysis found that eliminating the ventricular activity through means of either wavelets or cancellation of the QRS-T segment and then finding the entropy of only the atrial activity was not conclusive enough for classification. Instead, classifying normal versus atrial fibrillation was successful 100 percent of the time by calculating the entropy of the Q to Q interval. Heart rate variability was much higher in signals with atrial fibrillation than normal signals. If further research could be done to improve atrial activity isolation, entropy of the wavelet determined atrial activity could be informative enough to make a classification. This could make an algorithm that is much more robust and one that is able to definitively detect atrial fibrillation in lieu of any other dysrhythmia.

Introduction

Atrial fibrillation (AF) is the most common cardiac dysrhythmia. It is characterized by uncoordinated atrial activation, leading to inefficient pumping of blood into the ventricles and out into the rest of the body [1]. Since this condition is so common, the ability to accurately detect it in an electrocardiogram (ECG) signal is imperative. This problem has been solved in many different ways, but each is subject to certain issues that arise, making them less reliable. One common method is to detect the R to R time of the ventricular activity [1]. As atrial activity in a patient with AF is erratic, the R to R interval is less consistent than in a patient with regular atrial activity. However, this is also the case in many other dysrhythmias, so that single factor is not enough to definitively diagnose AF [2]. Therefore, with this method, it must be known that AF is a possibility and there must be other ways to ensure that AF is the true cause of the R to R invariability. Another common method is to cancel out the ventricular activity in the ECG and then analyze the behavior of only the atrial activity [3]. In a signal that has AF, p-waves may come multiple at a time or be skipped between ventricular contractions. This is a reliable factor to investigate for the diagnosis of AF. This method, however, can be susceptible to inconsistencies in results due to noise. Additionally, detection based on p-wave activity may be inconsistent because of the episodic nature of AF [4]. Since irregular atrial activity may come and go, it may not be consistent enough for an algorithm that looks only for p-wave activity as enough of the signal may have normal activity to result in a normal classification.

In an attempt to mitigate the issues that other methods face, this project focused on the comparison of classification for time and frequency domain methods. In the time domain method, the QRS-T portion of the ECG, or the ventricular activity, was detected. All of these portions from each signal were averaged to create a sample of the average QRS-T portion. Subtracting this average from each pulse allowed for most of the ventricular activity to be cancelled out, leaving only the atrial activity for further analysis. The frequency domain method involved wavelets. The ECG signal was broken down into ten different frequency bands from zero up to half of the sample frequency of the data, 150 Hz. The frequency bands that make up the atrial activity were summed in the time domain and the frequency bands that make up the ventricular activity were left out. The results of this step had the same intent as cancelling out the QRS-T portion from the time domain. In both methods, once atrial activity was the only thing remaining in the signal, entropy was used to determine the amount of uncertainty that existed. Ideally, in a normal sinus rhythm, the uncertainty and, therefore, the entropy, should be low due to the consistent nature of the signal. However, in a signal that displays AF, atrial activity is more inconsistent, so uncertainty and entropy should high. This would allow for simple classification of the signals. Additionally, entropy calculations were performed on Q to Q intervals in each signal as another metric to investigate.

This analysis was performed on single-lead ECG data that was obtained from over 12,000 people. Data in this set included normal sinus rhythm, atrial fibrillation data, noisy data, and data with other dysrhythmias [5]. For the purpose of this project, only low noise, normal sinus rhythm and atrial fibrillation data was considered along with signals that exhibited other dysrhythmias for comparative analysis.

Methods

The data used for this project was obtained from the Physionet Computing in Cardiology Challenge 2017 dataset. This dataset contained 12,186 ECG recordings that were donated by AliveCor. The recordings were taken by individuals who had purchased an AliveCor single-channel ECG recording device. These individuals held an electrode in each hand to create a lead I equivalent circuit. The device checked for signal quality and then recorded data for 30 seconds. This data was sent acoustically to a smartphone with a 19.1 kHz carrier frequency and a 200 Hz/mV modulation index. This data was stored with a 300 Hz sampling frequency in 16 bit files [2].

The dataset included labels for each signal to denote normal sinus rhythm, atrial fibrillation, other dysrhythmias, and signals that were too noisy to be classified. These labels were used as benchmarks to test algorithm accuracy against. Ten normal sinus rhythms, ten atrial fibrillation rhythms, and ten other dysrhythmia signals were selected as the training set for this project and ten more of each were selected as the test set.

This algorithm was split into three methods. For the first two methods, the goal was to remove all ventricular activity, leaving only atrial activity to be analyzed. This was split into two methods to determine if either the time or frequency domain was the best to complete the ventricular removal. The third method looked at classification without the removal of the ventricular activity.

The first method that was used to complete AF classification involved the frequency domain. For this analysis, no preliminary data manipulation such as filtering was implemented aside from normalizing the ECG data so that it was between zero and one. Since frequency analysis was being used, it was assumed that any noise would be filtered out through removal of the high frequency content. The original ECG signal was sent through wavelet decomposition. The discrete approximation of the Meyer wavelet was used as the mother wavelet for this technique. Nine different detail coefficients and one approximate coefficient was found. These coefficients led to ten different frequency bands from zero to half of the sampling frequency. As it is known that the QRS complex lies from about 8 to 50 Hz [7], the bands that included these frequencies were eliminated from further analysis. Beyond that, a trial and error method was used to determine which of the low frequency coefficients were necessary to capture atrial activity and eliminate ventricular activity. It was found that the approximate coefficient, a9, and the first two detail coefficients, d9 and d8, were the coefficients that encompassed only the atrial activity. The values of these coefficients were added together to create a new signal that would, theoretically, display only the desired content.

Once the atrial activity signal was obtained, wavelet entropy and entropy were both calculated for it. For wavelet entropy, the three coefficients that were used to create the atrial activity signal were put through equations 1 and 2 in order to determine the resulting amount of uncertainty in the signal [4].

$$E_j = \frac{\sum_{k=1}^{P_j} C(j,k)^2}{\sum_{j=1}^{N}\sum_{k=1}^{P_j} C(j,k)^2} \tag{1}$$

$$WE = -\sum_{j=1}^{N} E_j \log(E_j), \tag{2}$$

An additional form of entropy was calculated that did not take any of the wavelet coefficients into account. Instead, it only looked at the probabilities of the atrial activity signal. This was done for windows of one second of data in order to reduce the effects of significant spikes in the data. The entropy value of every second was averaged to obtain one final result. The probabilities for values in each second of data were put through equation 3 to determine the result.

$$H = \sum_{i=0}^{N} p_i * log(p_i) \tag{3}$$

Next, the second method was completed. This method removed the ventricular activity using the signal in the time domain. In order to account for some of the noise, the first thing that was done in this method was bandpass filter the data. The pass band went from 5 to 15 Hz. This allowed for baseline wander and other low frequency noise along with high frequency noise to be removed from the signal. Then, to account for any signals that contained outliers, the peaks of

the signal were found. A threshold of 0.6 was set to allow a decision to be made on whether the peaks were a QRS complex or a smaller T or P wave peak. Next, any of those peaks that were greater than 1.5 times the median value of the QRS peaks were determined to be outliers, or noisy data. That peak was then removed from the data by setting all of the points involved with it to the average value of the signal. This allowed for the peaks to be a consistent amplitude for further analysis.

Next, the Pan-Tompkins method was applied [6]. Here, the derivative of the ECG signal was taken. Then, the derivative of that result was taken again. The peaks in that second derivative of the ECG signal were found. A threshold of 0.3 times the maximum peak value was set in order to determine which of the peaks in the Pan-Tompkins derivative were significant. Those peaks were in the same location as the Q and S points on the ECG for each pulse, respectively. A test was done to determine if the first peak and, therefore each subsequent peak, were classified properly as a Q or S point. By default, the first peak was labelled a Q peak, however, if the next peak was more than 100 data points away, it was clear that it was not close enough to be an S point on the other end of the same QRS complex. Therefore, it was shown that the first peak must have been an S point and the classification of each peak was switched. Finally, before the Q and S points could be solidified, the algorithm went through each peak to determine if the set threshold had missed any points. It did this by checking if each Q and S point of the same index were within 100 points of one another. If they were not, it showed that one of them had been missed. Therefore, a new S point was added into the S points array at a location equal to the average Q to S interval. Then, the classification of each of the following points as Q or S was switched to account for the added point. Since T-wave location will be determined based on the location of S points, this step was very important for the integrity of locating the ventricular activity.

After the Q and S points were detected, the data was smoothed twice using a moving average filter. This was necessary because the next step was to find the end of the T-wave by looking at the slope of the signal. Before smoothing, the signal had jagged noise on the T-wave, which caused the algorithm to falsely believe the slope had switched directions. After smoothing, the data was consistent enough to check the difference between each data point. The first time the difference became negative after the S point indicated the location of the top of the T-wave. To ensure that no noise slipped through the smoothing filter and indicated a false peak, the next ten differences were checked to ensure that they were also negative. Then, to find the end of the T-wave, the first time the difference between points turned positive after the peak of the wave indicated the local minimum.

After the Q, S, and T points were found, an average of all QRS-T segments in the data could be determined. This was done by separating out every Q to T interval and averaging them point by point. That average QRS-T segment was then subtracted from each pulse in the signal to, effectively, cancel out the ventricular activity. At this point, the same moving entropy calculation that was discussed along with Equation 3 was completed on this QRS-T removed data.

The third and final method that was completed in this project picked up after the Q, S, and T points were found. The length of each Q to Q interval was determined. Equation 3 was used to calculate the entropy value of those intervals without the use of windowing to split the data up second by second. When all entropy values were determined, they were compared between normal sinus rhythm and atrial fibrillation data. A classification was made of normal

sinus rhythm if the Q to Q interval entropy was less than four. If it was greater than four, the signal was classified as AF data.

After the four different entropy calculations were complete, statistical analysis was completed on the values obtained for each signal tested. In total, 20 signals from groups of normal sinus rhythm, atrial fibrillation, and other dysrhythmias, respectively were tested. A student's T-test was completed on these three sets of data to determine if the average entropy was significantly different between any two pairings. A flow chart of the described methods is shown in Figure 1.

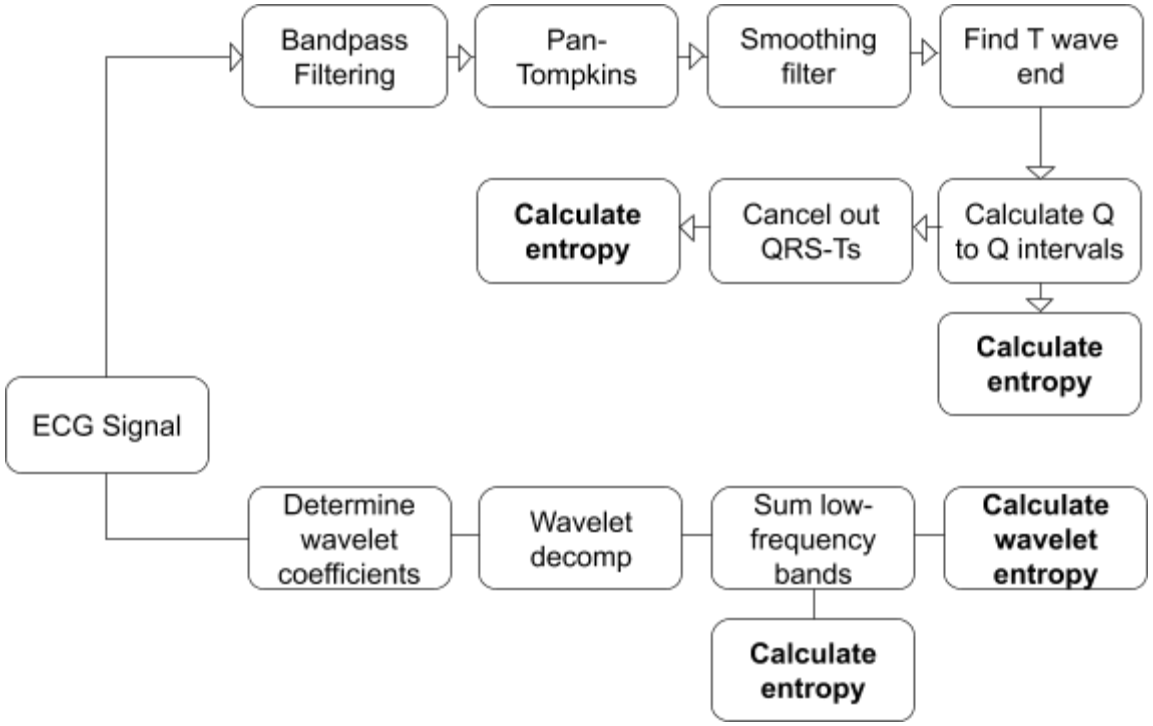

Figure 1. Flow chart of methods for algorithm

Results

The first thing that was done in this algorithm was to read in the data and normalize it. Data was normalized so that its maximum and minimum values were between zero and one. Then, that data was plotted against time to obtain an idea of what the signal looked like. An example of a normal sinus rhythm is shown in Figure 2 and a rhythm with AF is shown in Figure 3.

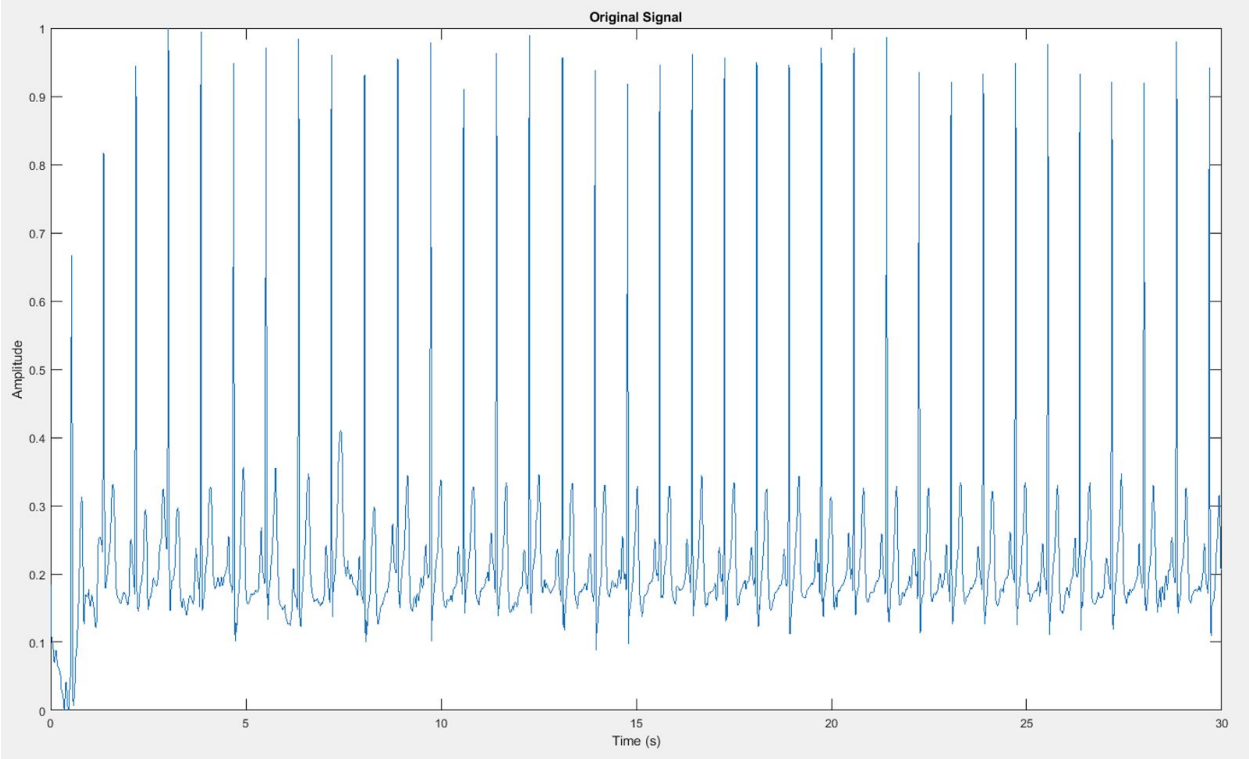

Figure 2. Normalized raw data of a normal sinus rhythm

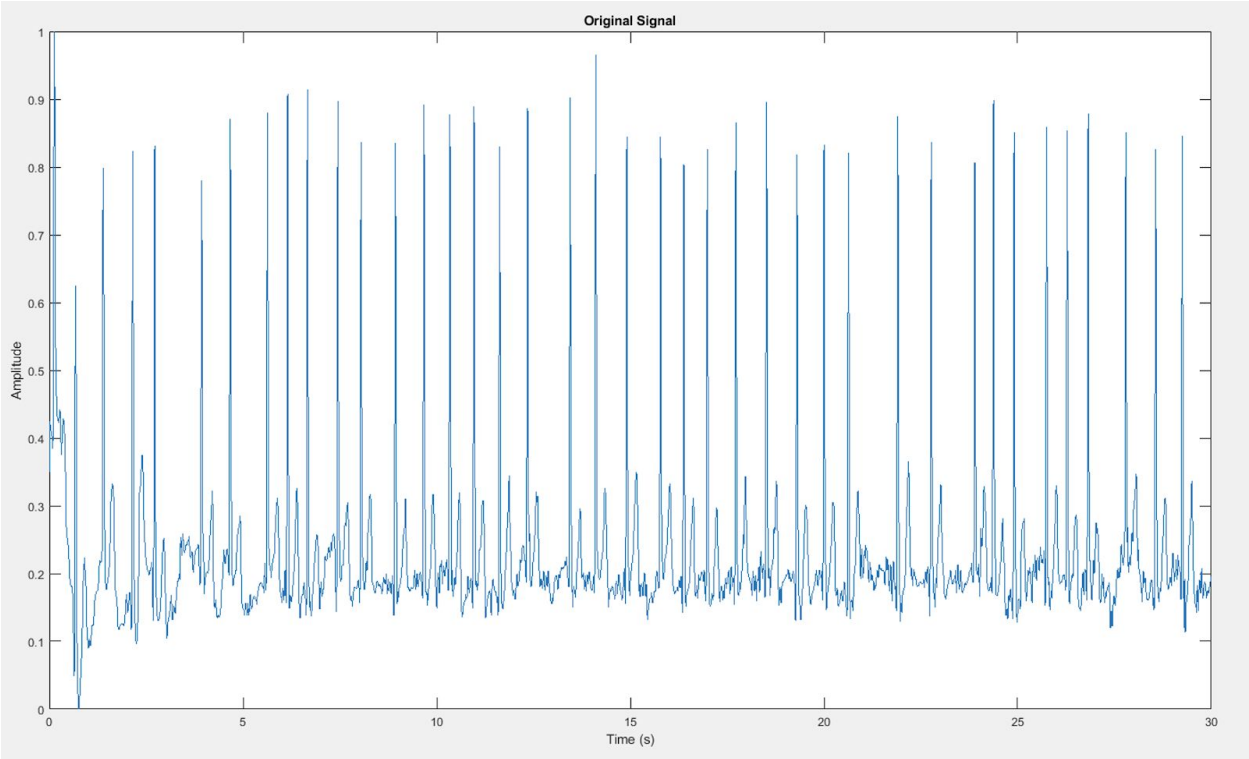

Figure 3. Normalized raw data of an atrial fibrillation signal

Next, the wavelet decomposition was completed to separate the signal into ten different frequency bands. An example of this breakdown is shown in Figure 4.

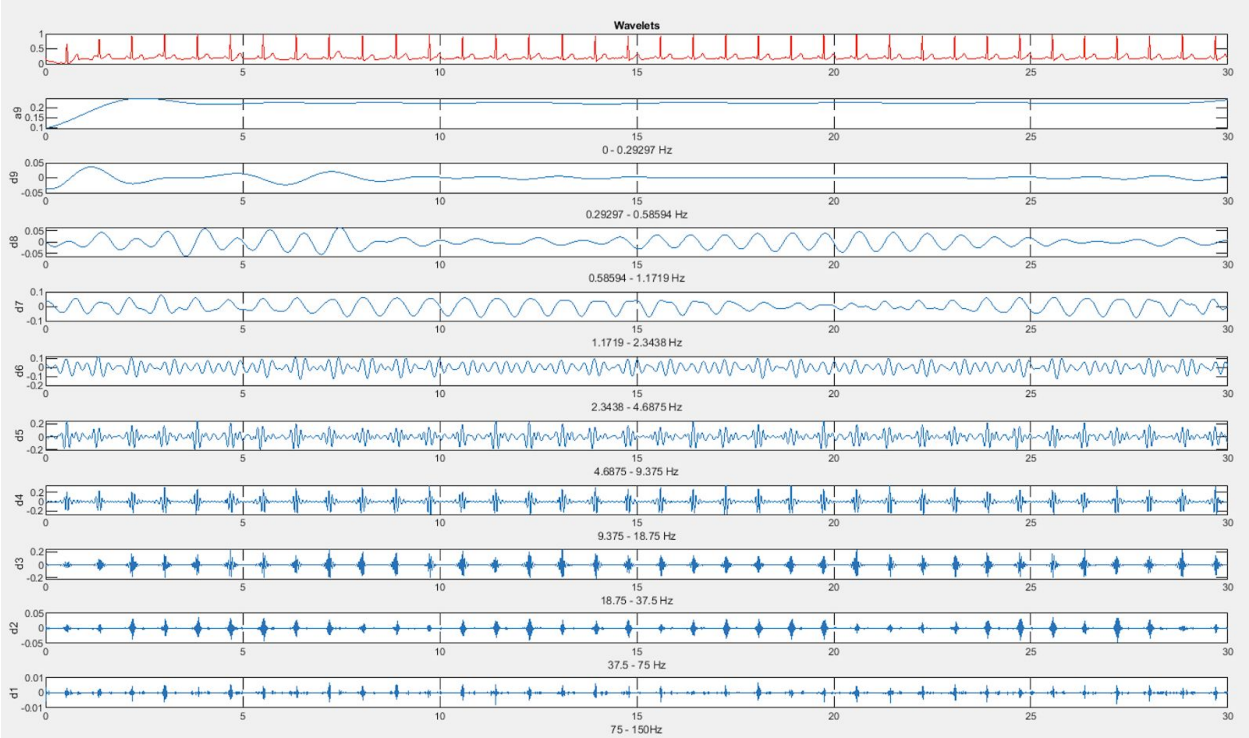

Figure 4. Wavelet decomposition of ECG into ten frequency bands

The three lowest frequency bands were summed to create a signal that contained only the p-waves of the original ECG. This also removed excess noise on the p-waves. This atrial activity is shown in Figure 5 for a normal ECG and in Figure 6 for an ECG with atrial fibrillation.

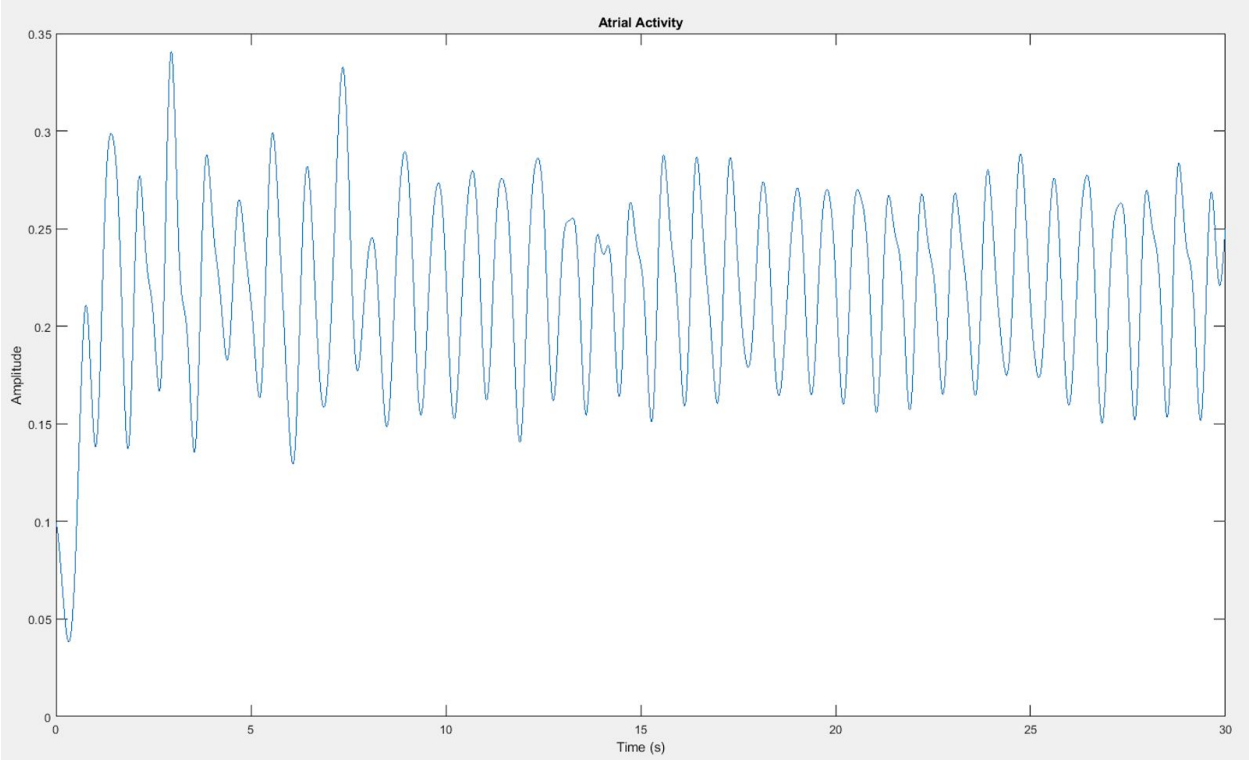

Figure 5. Atrial activity with ventricular activity removed using wavelets for a normal rhythm

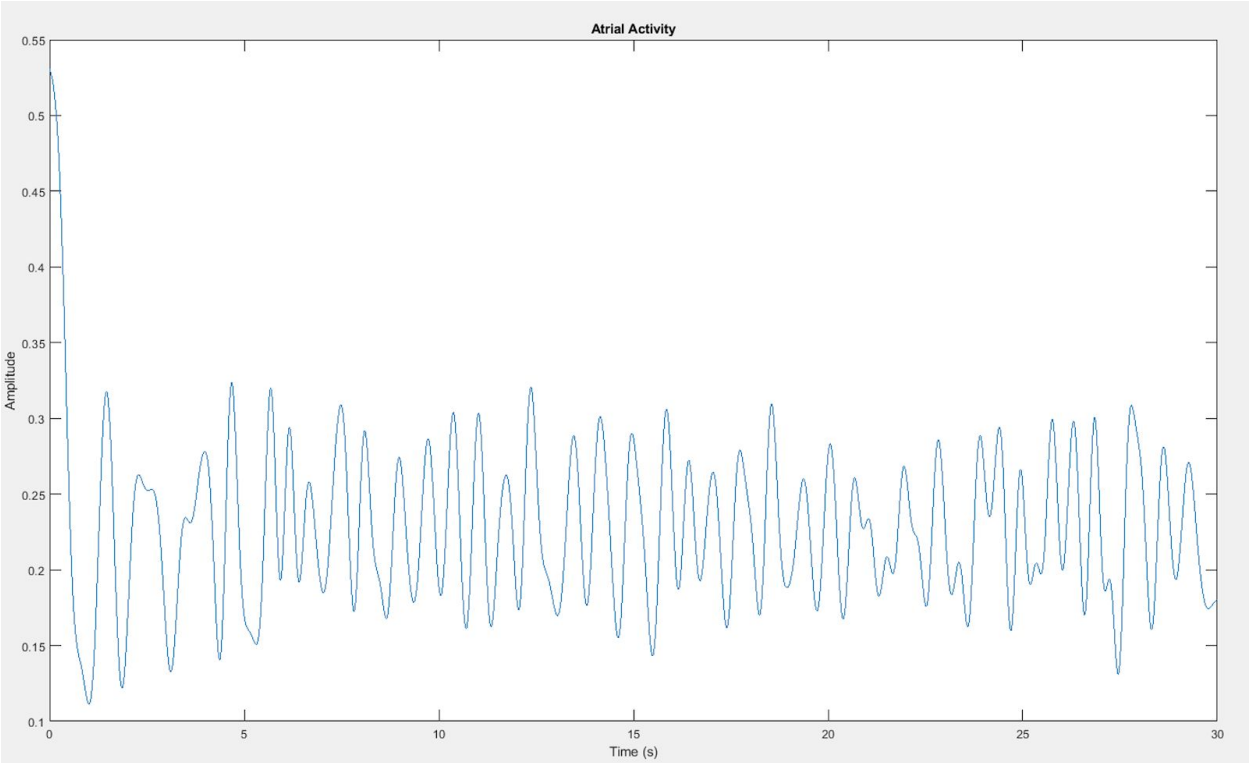

Figure 6. Atrial activity with ventricular activity removed using wavelets for atrial fibrillation

Once the atrial activity was separated from the rest of the signal, wavelet entropy was calculated on the atrial activity by looking at each value of the three coefficients that were summed to create the signal. This was done using Equations 1 and 2. The resulting uncertainty of this atrial activity for each signal tested with normal rhythm, AF, and other dysrhythmias is plotted in Figure 7. The wavelet entropy was not significantly different between the three categories of data. It was inconsistently between approximately 0.1 and 0.5 for each classification of signal.

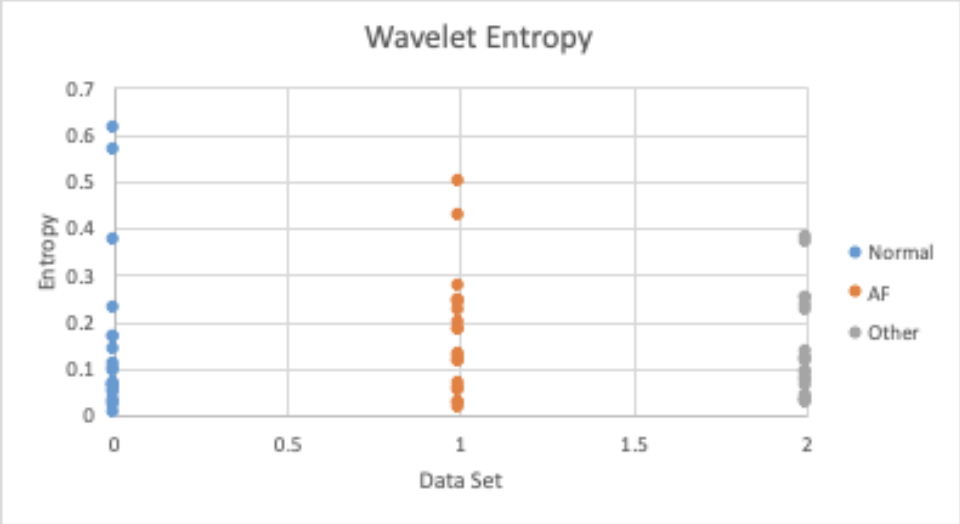

Figure 7. Wavelet entropy of wavelet determined atrial activity for all three signal types

Additionally, normal entropy was calculated by looking at the probabilities of different values within the ECG signal as opposed to the coefficients. It does not appear is if there is a significant difference between the entropy for the atrial activity in any of the three categories. The values are all relatively similar as they were all around 8.1. It is shown, however, that AF signals tend not to drift as low as normal or other dysrhythmic signals. Without further analysis, though, it does not appear that this method will work for classifying the signal. The calculated entropy values for the three signal types are seen plotted in Figure 8.

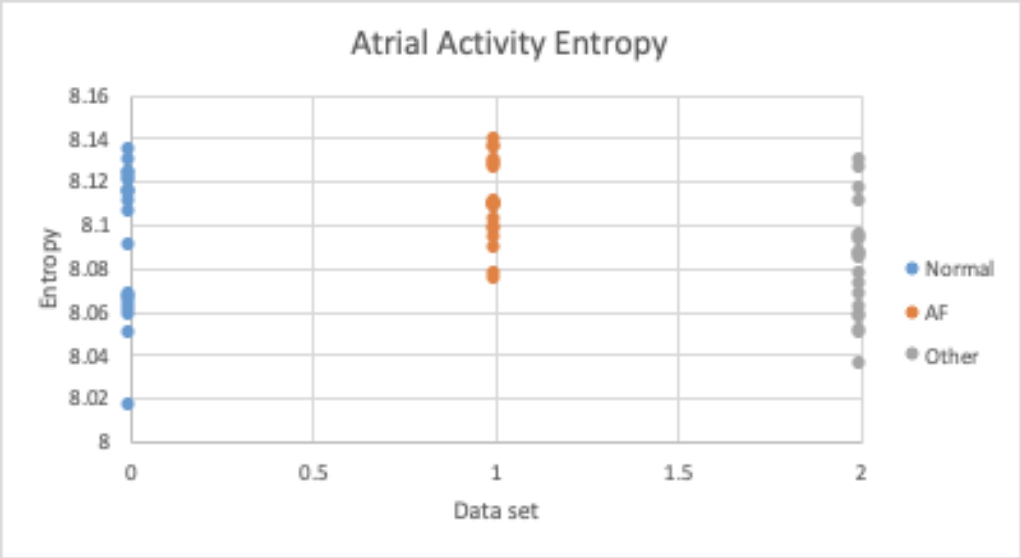

Figure 8. Normal entropy of wavelet determined atrial activity for all three signal types

Next, the original normalized signal was filtered before time domain analysis was performed. The signal was bandpass filtered with a pass band from 5 to 15 Hz. This eliminated a lot of noise as well as any baseline wander that existed. A filtered normal signal is shown in Figure 7 and a filtered AF signal is shown in Figure 8.

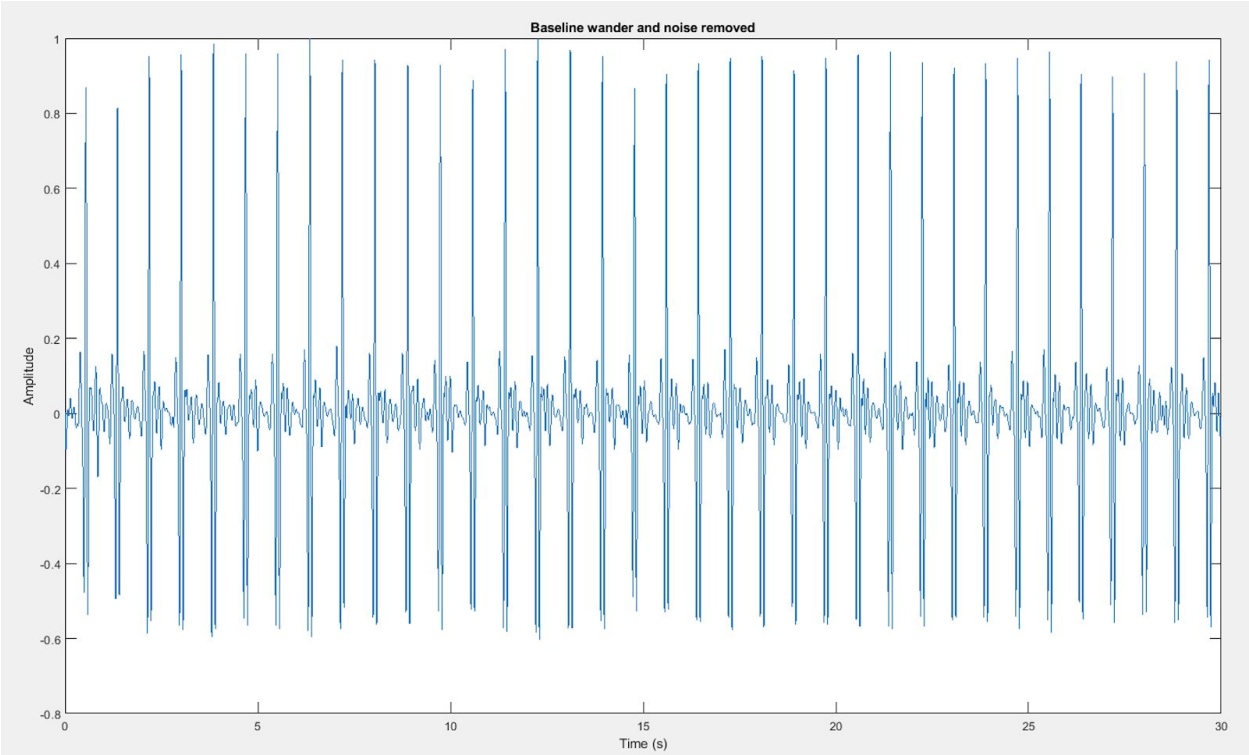

Figure 9. Normal sinus rhythm bandpass filtered from 5 to 15 Hz

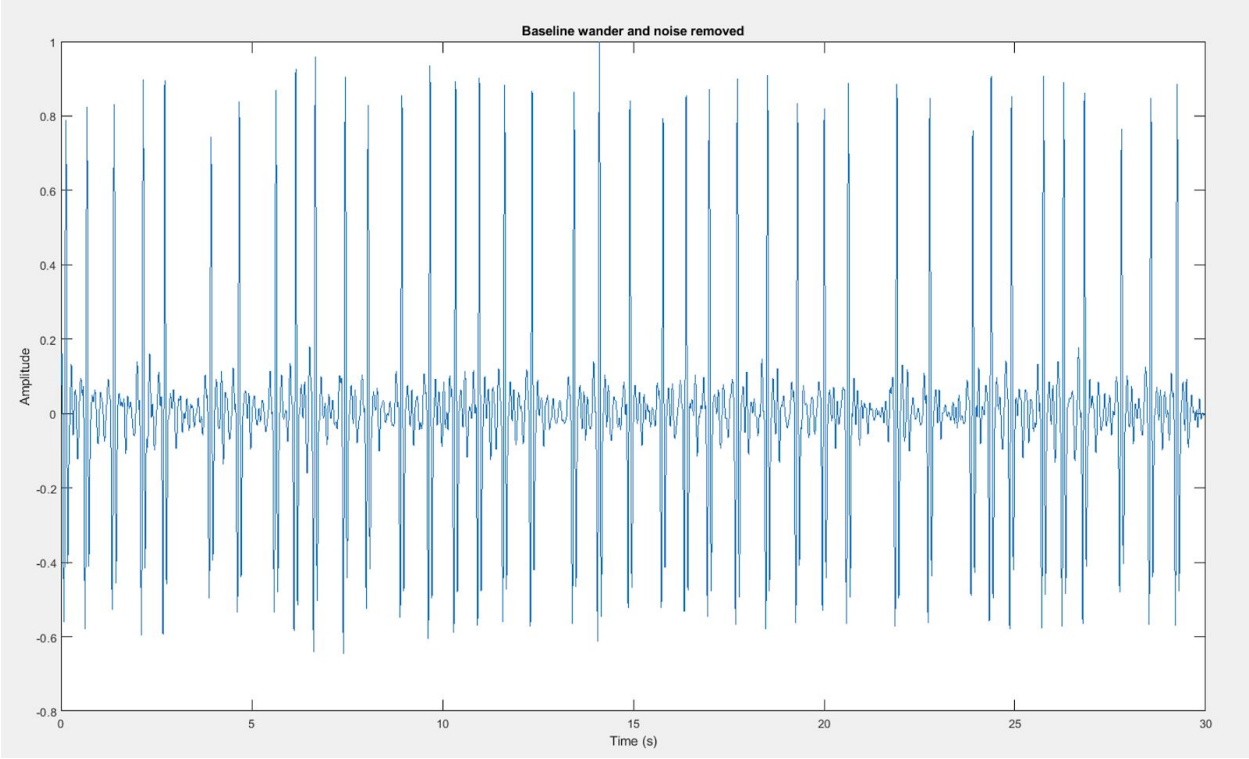

Figure 10. Atrial fibrillation rhythm bandpass filtered from 5 to 15 Hz

After filtering, all peaks in the signal were located and all peaks greater than 0.6 were classified as the peak of a QRS complex. The QRS peaks are shown for a normal signal in Figure 8 and an AF signal in Figure 10.

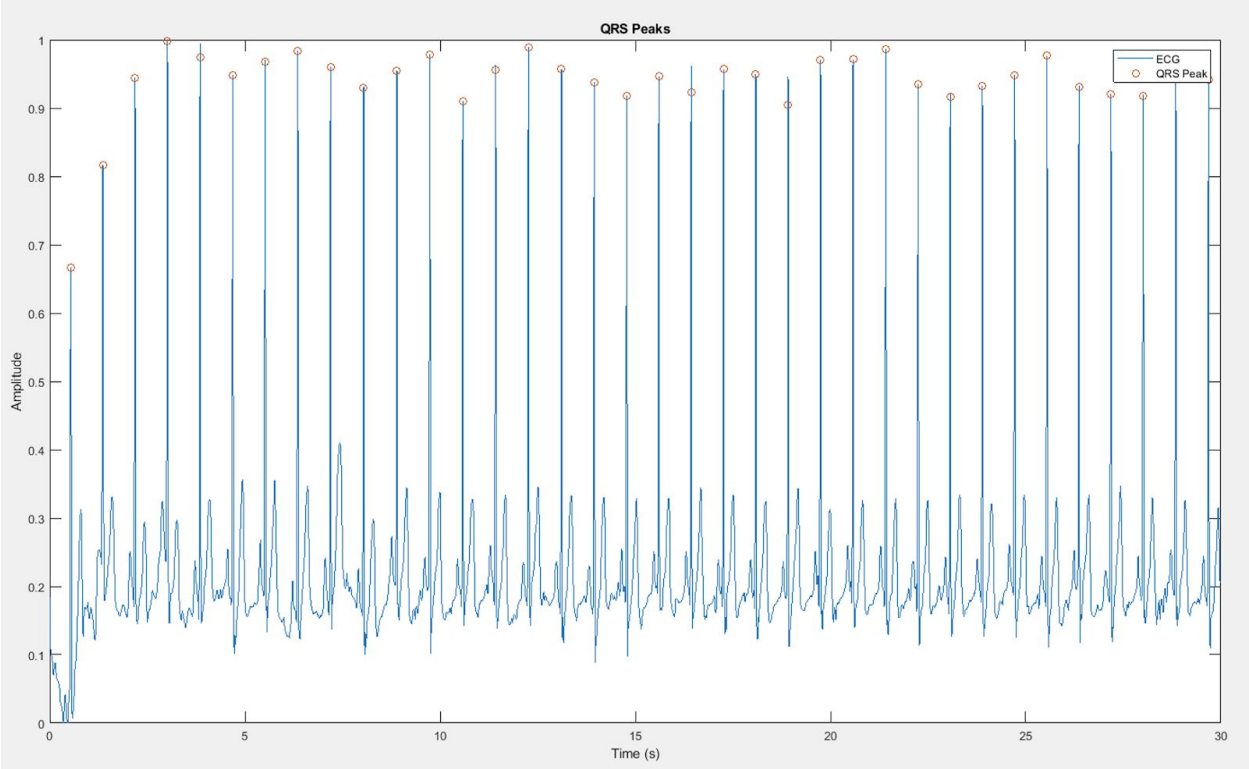

Figure 11. QRS peaks for normal sinus rhythm

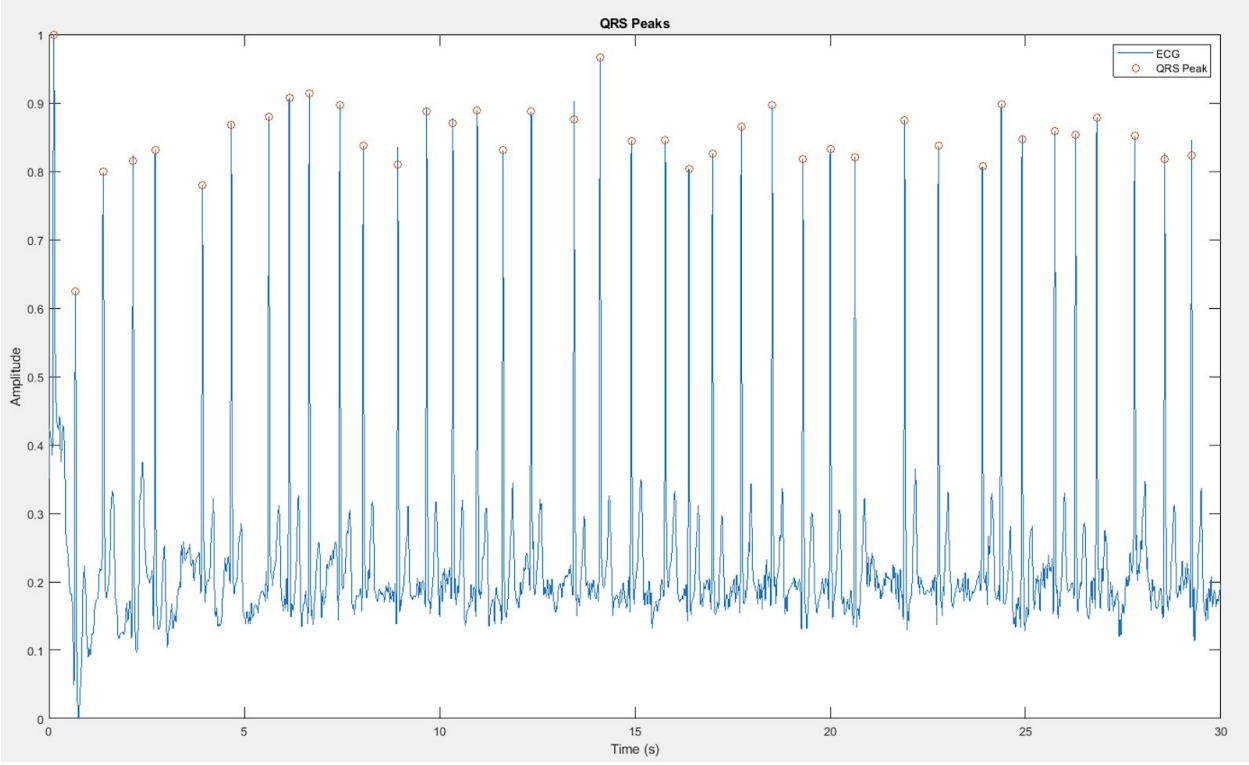

Figure 12. QRS peaks for atrial fibrillation signal

Determining the QRS peaks shown above was also helpful for determining if any outliers existed that were significantly larger value than the actual QRS peaks. The data that was used in this set, however, was generally very clean data without outliers. Therefore, no outliers were detected in the data used.

Once it was verified that no outliers existed, or if any did, they were removed, the Pan-Tompkins method was utilized. Two derivatives were taken of the data and the peaks of that result were located. These points signified the locations of the Q and S points in the ECG data. The Pan-Tompkins derivative with the Q and S peaks shown are seen in Figure 11 for a normal rhythm and Figure 12 for an AF signal.

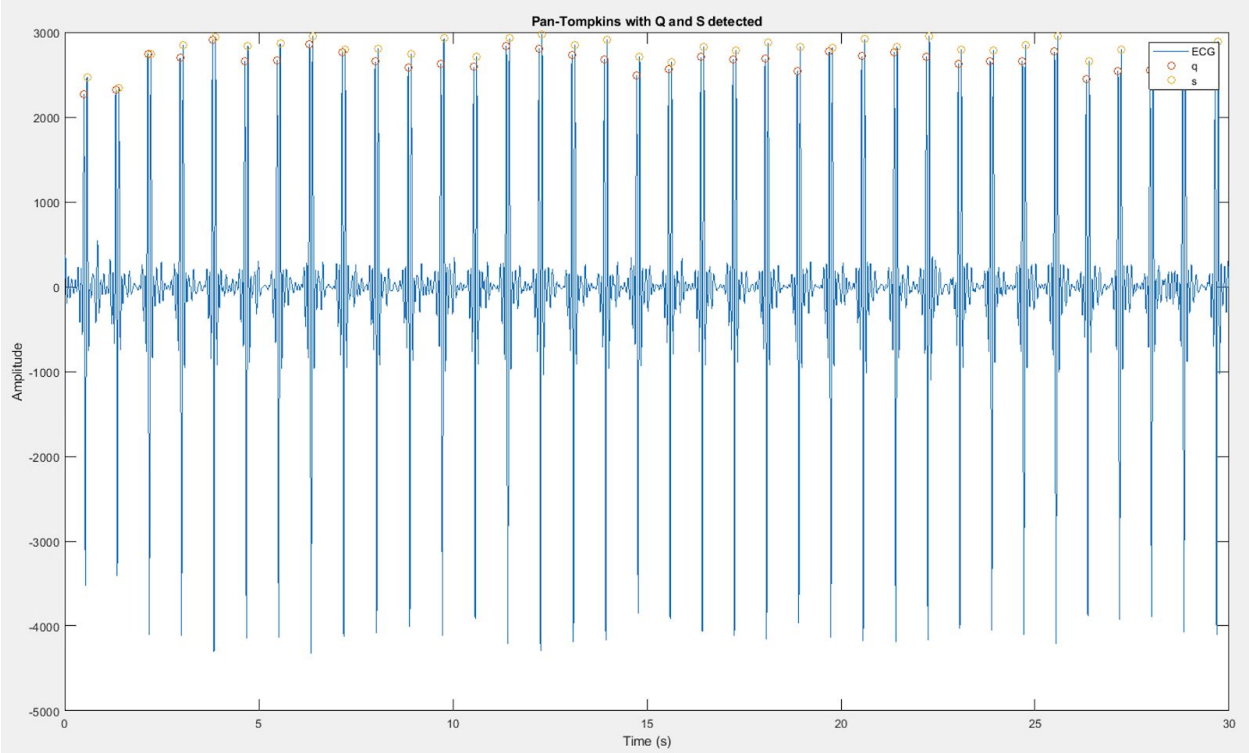

Figure 13. Pan-Tompkins result with Q and S point locations shown for normal rhythm

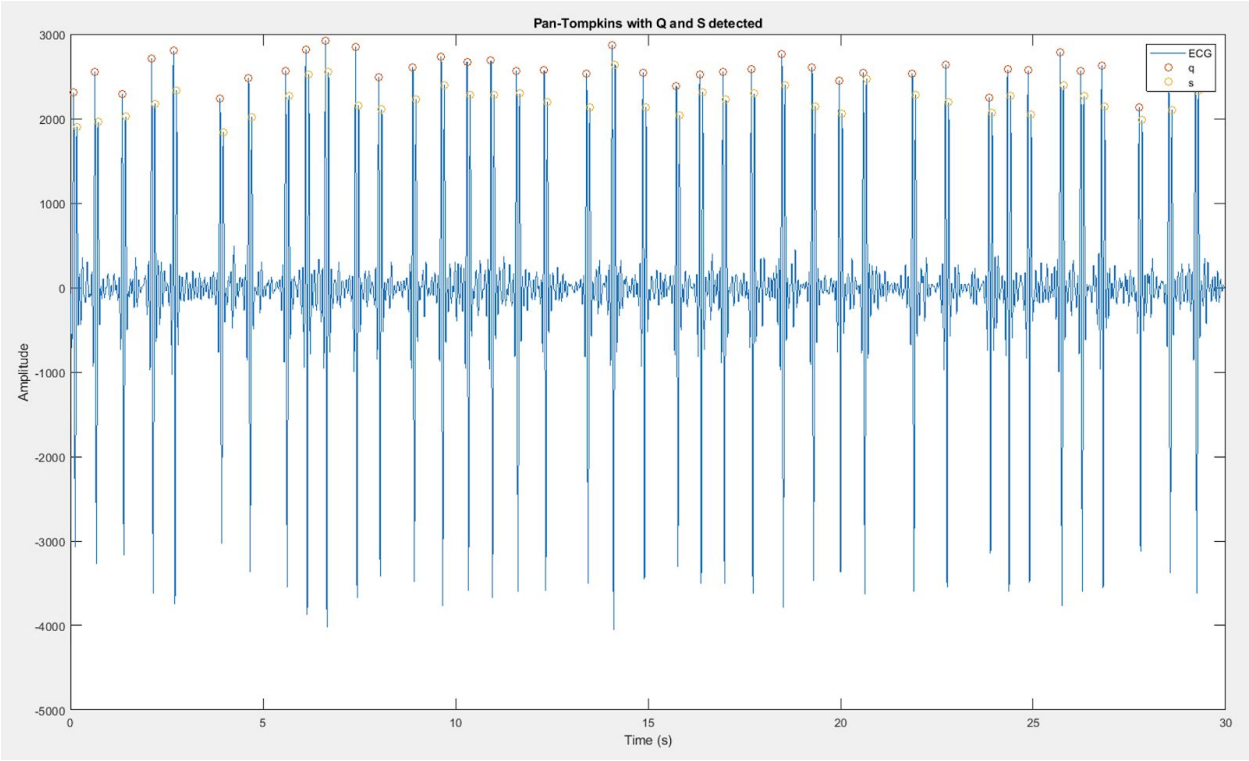

Figure 14. Pan-Tompkins result with Q and S point locations shown for AF rhythm

Once the locations of the Q and S points were known, the T-wave could be found. First, the ECG signal was smoothed with a moving average filter. An example of a smoothed signal is shown in Figure 13.

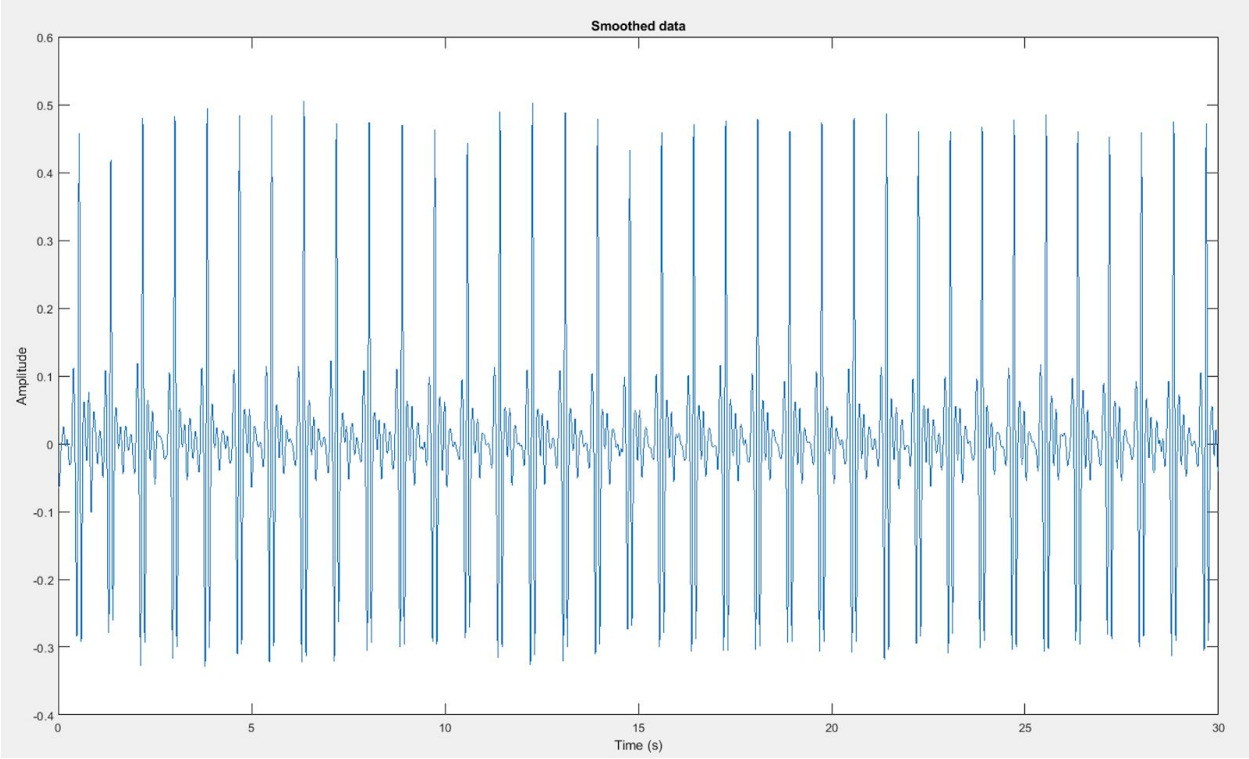

Figure 15. Smoothed ECG signal

By checking for inflection points in the ECG signal between an S-point and the following Q-point, the trough after the conclusion of the T-wave could be determined successfully. The normalized ECG signal with detected Q, S, and T points are shown in Figures 14 and 15 for a normal and AF signal, respectively.

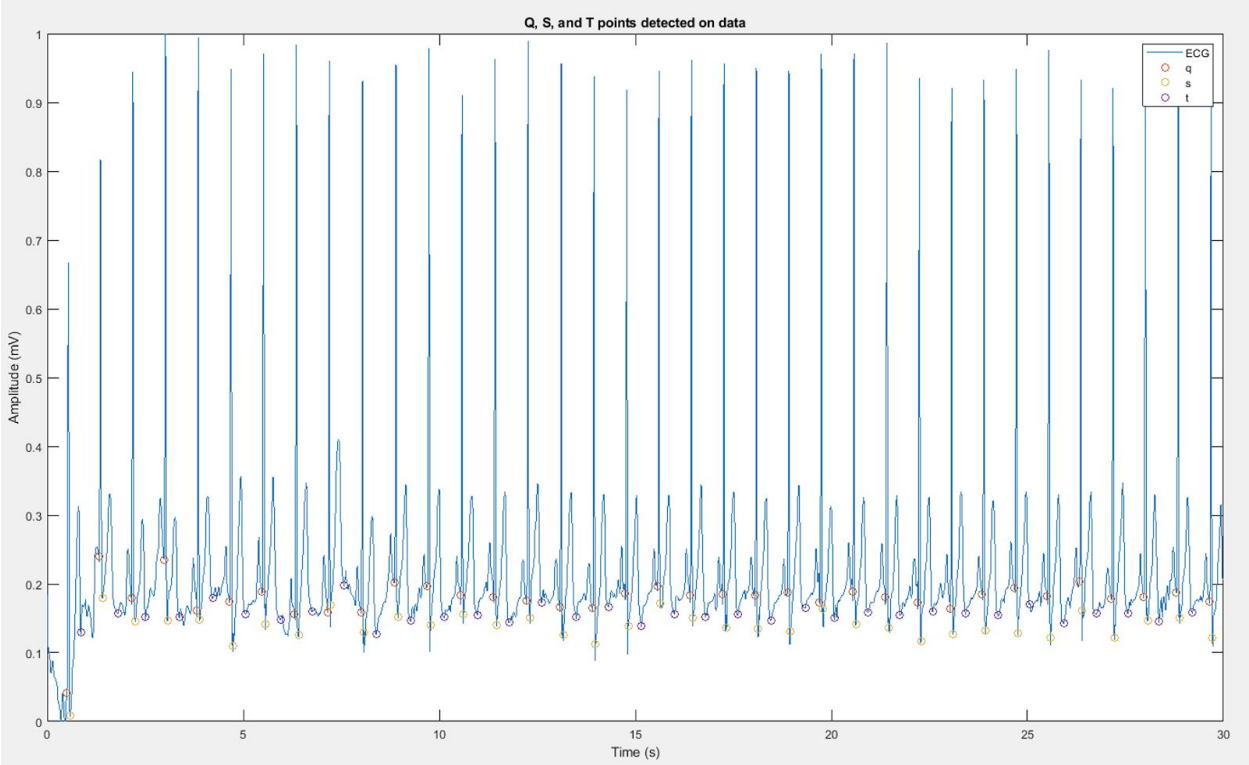

Figure 16. Q, S, and T points for normal signal

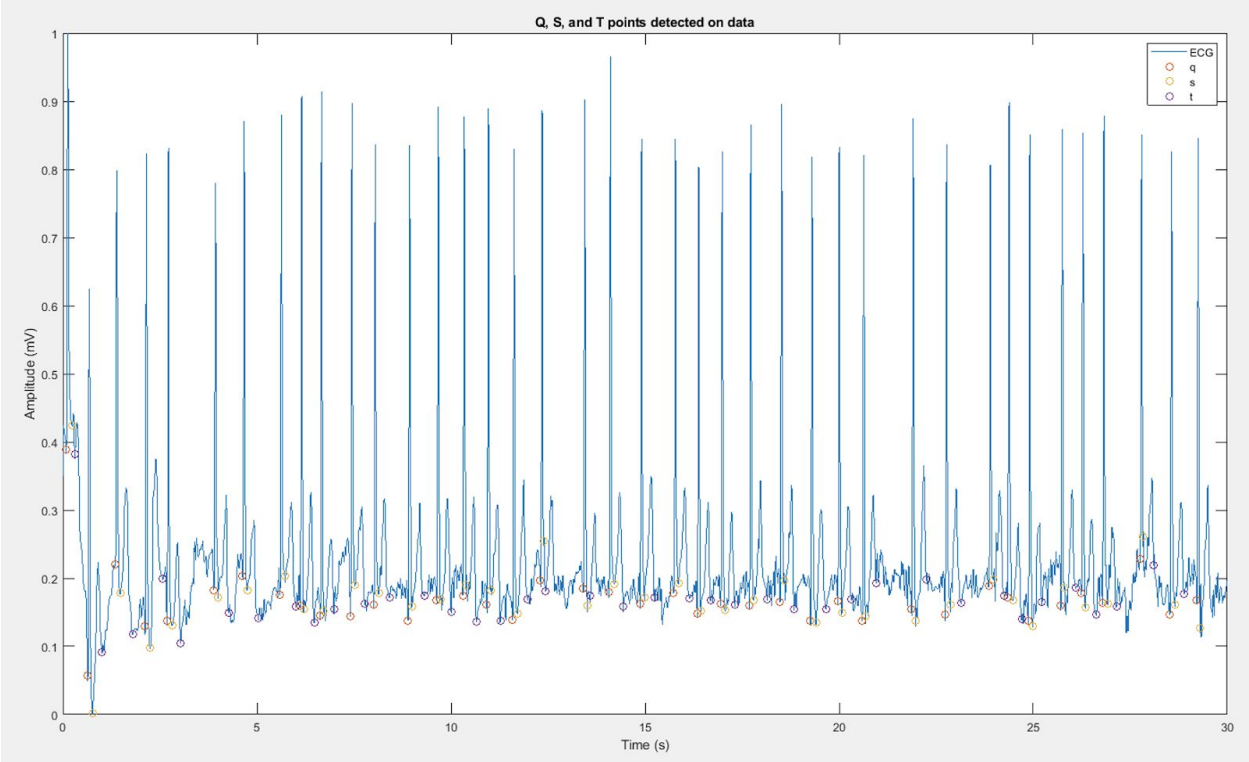

Figure 17. Q, S, and T points for AF signal

Next, each Q to T interval was determined for the signal and an average of all of those intervals was found. An example of an average QRS-T segment is shown in Figure 16.

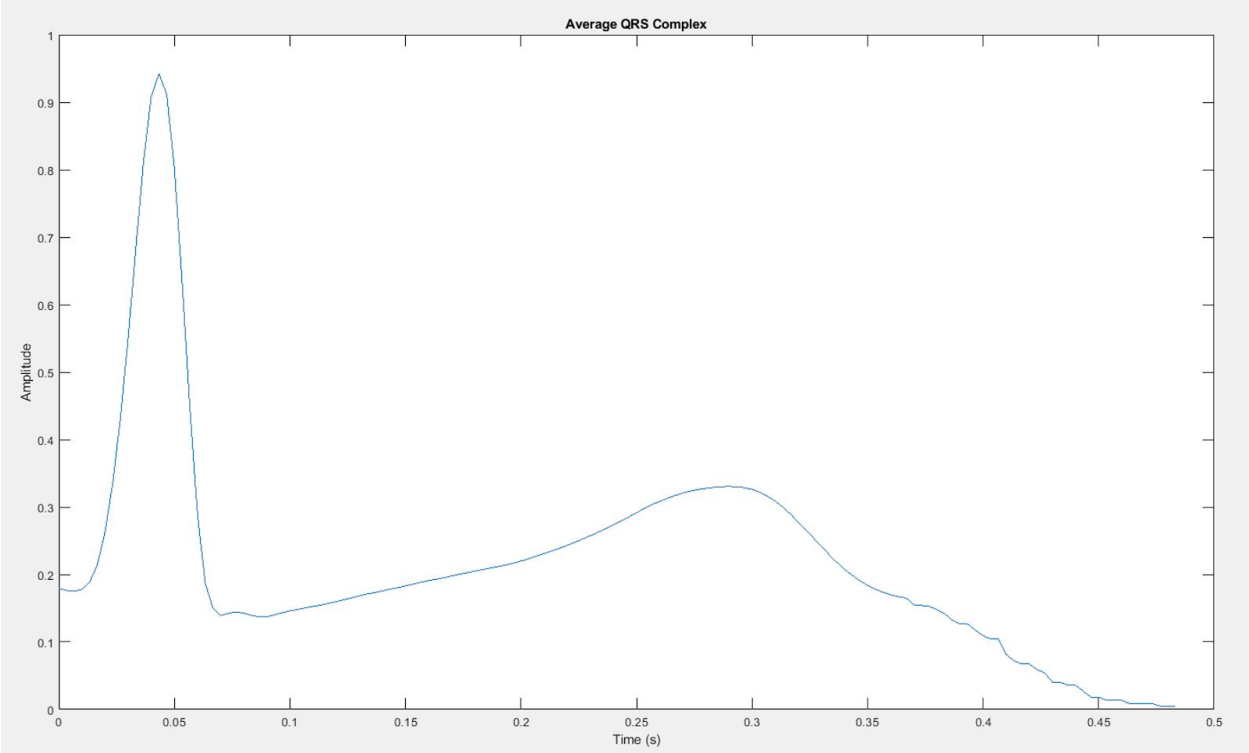

Figure 18. Average QRS-T signal for a normal ECG

That average QRS-T segment allowed for the ventricular activity to be cancelled out by subtracting it from each individual QRS-T segment within the signal. The ECG signal with ventricular activity cancelled out is shown in Figure 17 for a normal sinus rhythm and in Figure 18 for an AF signal.

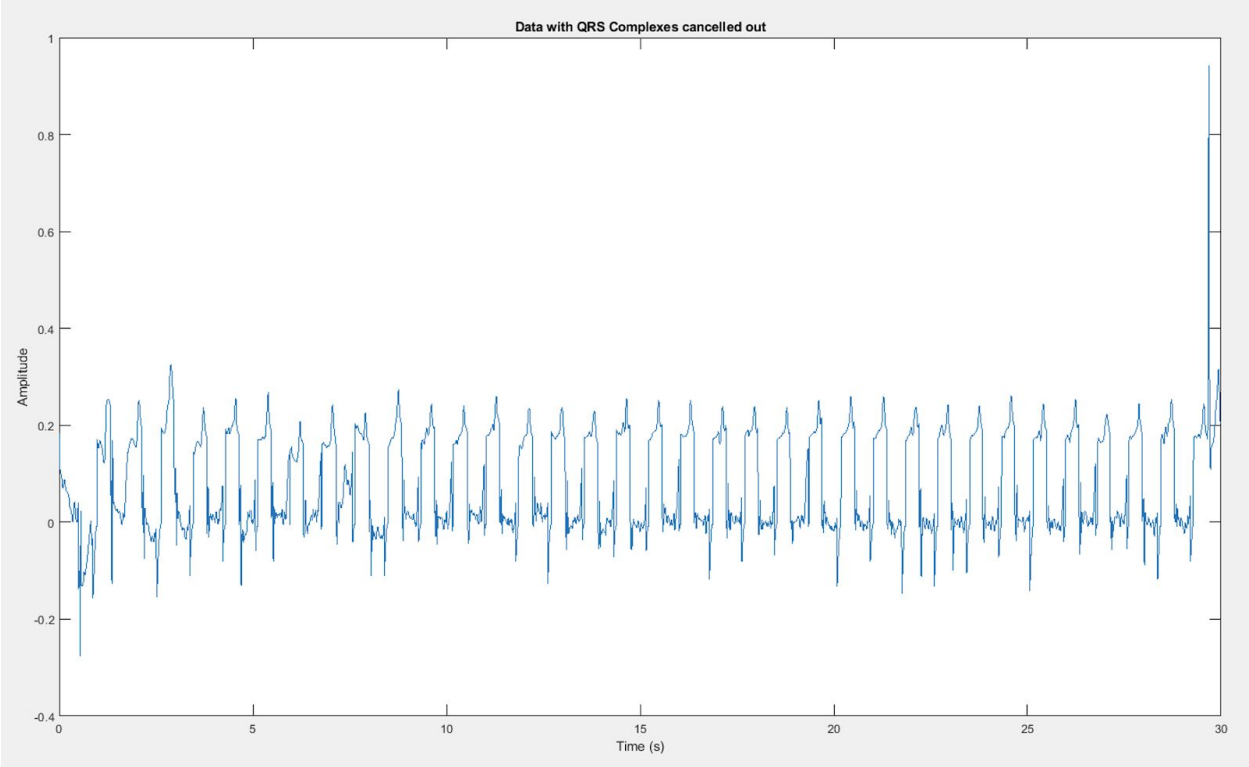

Figure 19. Ventricular activity cancelled out of normal sinus rhythm

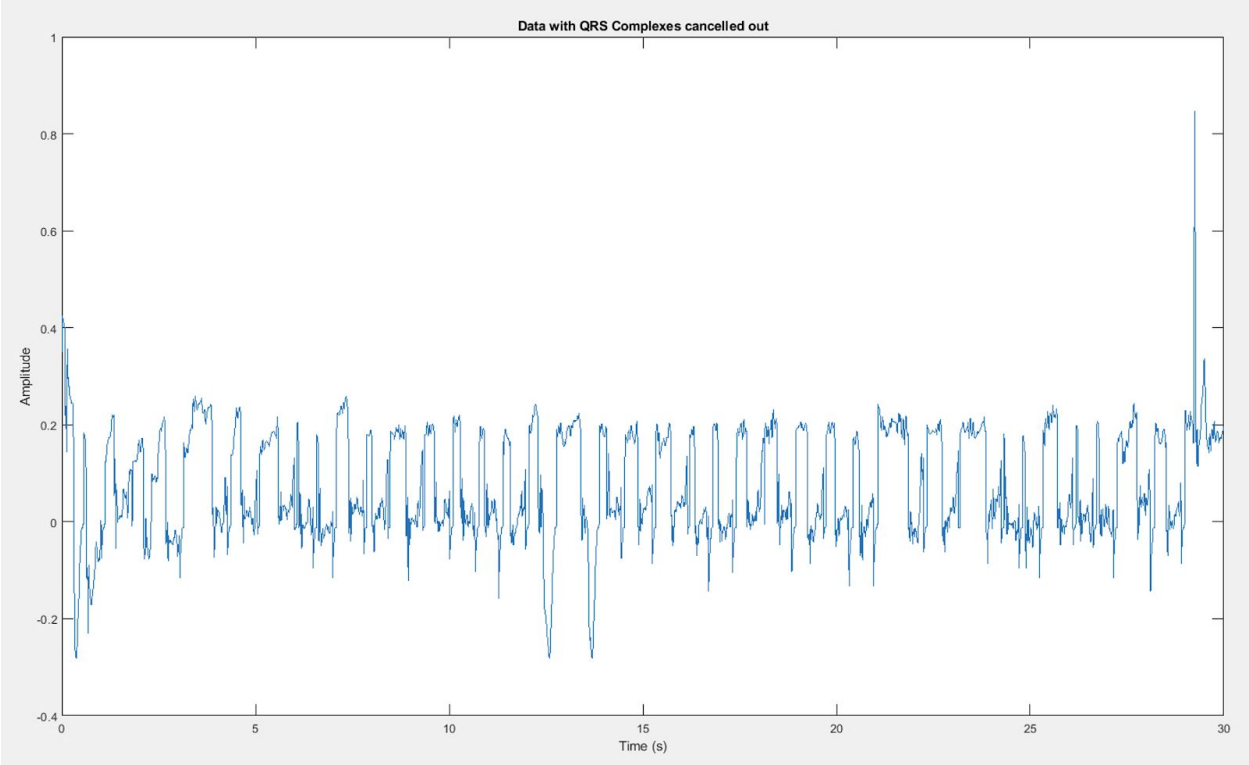

Figure 20. Ventricular activity cancelled out of AF rhythm

At this point, the average entropy from one second windows of the signal with ventricular activity cancelled out was calculated. Similar to the wavelet entropy results in Figure 7, all of these entropy values are very nearly the same for all three categories. These results are found in Figure 21 for each of the normal, AF, and other dysrhythmia signals.

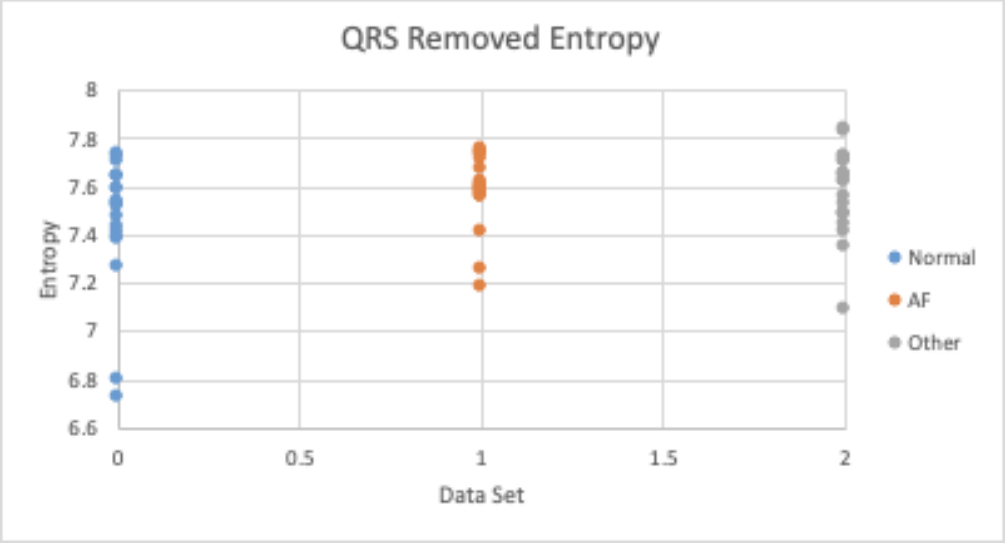

Figure 21. Entropy of QRS-T cancelled data for all three signal types

The final method was to determine each Q to Q interval using the previously detected Q points. The entropy of those values through a signal was obtained. These values are found in Figure 22, below. Differences are seen in these entropy values with normal sinus rhythm data tending to have Q to Q interval entropy of less than four and AF data tending to have Q to Q interval entropy greater than four. The data from other dysrhythmias, however, was unpredictably above and below four.

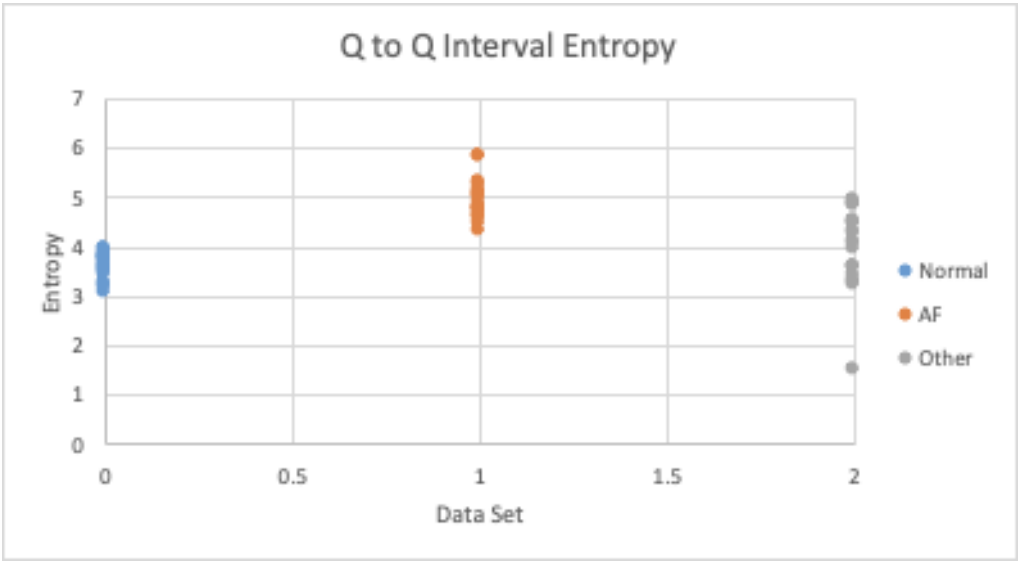

Figure 22. Entropy of Q to Q intervals for all three signal types

The final statistic that was performed on the data was a Student's two tailed t-test on each of the three pairs of categories in order to determine if the means of the different calculated entropy values were significantly different between normal, AF, and other dysrhythmic data. The p-values that resulted from these t-tests are shown in Table 1. The Q to Q interval entropy had, by far, the lowest p-value in all three categories of data.

Table 1. T-test results

P-value: Normal vs. AF	P-value: Normal vs. Other	P-value: AF vs. Other
0.044590169	0.26146149	**0.000290852**
0.091130757	0.123624539	0.906100869
3.76E-15	0.069690697	**2.21E-05**
0.118204319	0.814274883	0.071666756
0.666376419	0.775112273	0.371641762

Discussion

The data in Table 1 shows information that goes against the original hypothesis that the entropy of the atrial activity in an ECG exhibiting atrial fibrillation would be higher than that of a normal sinus rhythm. Instead, it shows that entropies of that data, no matter the method used to obtain it, were all relatively equal for the categories of normal sinus rhythm, atrial fibrillation, and other dysrhythmias.

Figure 4 shows that all of the ventricular data was encapsulated in high frequency bands that were not included in the atrial activity as the shape of the QRS complex can be seen in wavelets with coefficients smaller than d7. However, there is a possibility that some low frequency noise still existed in the end signal. This could add more consistent peaks into the data and disrupt the entropy calculation. It is also possible that some of the atrial activity was left out by not including additional frequencies. This could lead to false results if additional, erratic atrial behavior existed in the original signal, but was not incorporated into the atrial activity.

Figures 14 and 15 show that the detection of the Q, S, and T points was successful as all three points were located correctly very nearly 100 percent of the time. The Q and the S points had 100 percent specificity and sensitivity. The T wave detection was very close behind with 97.4 percent sensitivity and specificity. Therefore, the average QRS complex that was developed was likely to be a good representation of the typical QRS complex in the signal. If the average complex and each individual complex were not perfectly lined up upon cancellation, that could have left additional ventricular activity in the signal, which may have affected entropy calculation.

An additional possibility for why entropy calculations did not have the expected results is that the atrial activity, alone, was not uncertain enough. When looking closely, the atrial activity in an atrial fibrillation signal is just a number of repeating small peaks. By eliminating the ventricular activity, the inconsistency of how much activity occurred at a time between QRS complexes is removed, so all that is left is a series of small waves. This may have simply not been inconsistent enough on its own to be reflected in entropy values.

The t-tests that were completed showed strong significance in the difference of means between normal and atrial fibrillation entropies for the Q to Q interval. That, along with the fact that a threshold

was able to accurately classify those two categories with 100 percent sensitivity and specificity, shows that Q to Q interval entropy is a good metric to use. The plot in Figure 19 shows a clear divide between all of the entropy values for normal signals and all of the entropy values for AF signals. The t-tests also showed statistical significance in the Q to Q interval between ECGs with atrial fibrillation and other dysrhythmias. While there is statistical significance, there was still a lot of overlap in entropy values between the two, so this may not be a good metric to classify between them as there is not a clear line to divide the categories between.

Another area that also showed statistical significance was in the entropy of the wavelet determined atrial activity signal in normal versus atrial fibrillation data and atrial fibrillation versus other dysrhythmia data. In both cases, there was still a lot of overlap between the entropy among the pairs, as seen in Figure 20, so at this point that metric is still not useful for classification. However, it does show that there is potential for further research. If the ventricular removal can be strengthened to ensure that absolutely no ventricular activity gets through, but all of the necessary atrial activity still does, that may be a useful tool for classification.

The two additional methods of wavelet entropy on the wavelet determined atrial activity and entropy on the QRS-T cancelled data using an averaged segment were not significant between any of the pairs. As Figures 21 and 22 both display, the entropy values for all three categories of each of these entropy methods are very similar. They are relatively equivalent across the board, so neither of these methods would be of any use for AF classification.

The fact that none of the entropy values showed up as significant in the t-testing between normal signals and other dysrhythmias as well as the fact that the statistically significant methods for atrial fibrillation signals versus other dysrhythmias still showed a lot of overlap in entropy values proves that this algorithm is not robust enough to discern between anything more than normal and atrial fibrillation at this point. Therefore, this could be a tool in diagnosing atrial fibrillation in conjunction with other methods, but not on its own. In some cases, using only this algorithm, a different dysrhythmia could be classified as atrial fibrillation and in others, a signal with some dysrhythmia could be classified as normal. Further analysis would have to be performed on the data, looking at many other features of the ECG in order to improve the ability to pick out atrial fibrillation, specifically.

This algorithm was dependent on the input signals being relatively noise and anomaly free. As long as the data fit that requirement, the software was quite successful. This success is very important because early detection of AF is necessary for the health of individuals who are suffering from it, which is approximately 2 percent of the world population [2]. Accurate detection of dysrhythmias could lead to improved quality of life for millions of people.

Acknowledgements

I would like to thank Jarred Parr, Oyinda Owoeye, and Dr. Rhodes for their vital help and support toward the completion of this project.

References

[1] Jekova, Irena, et al. “Arrhythmia Classification via Time and Frequency Domain Analyses of Ventricular and Atrial Contractions.” *2017 Computing in Cardiology Conference (CinC)*, 2017, doi:10.22489/cinc.2017.345-029.

[2] Clifford, Gari, et al. “AF Classification from a Short Single Lead ECG Recording: the Physionet Computing in Cardiology Challenge 2017.” *2017 Computing in Cardiology Conference (CinC)*, 2017, doi:10.22489/cinc.2017.065-469.

[3] Alcaraz, Raúl, and José Joaquín Rieta. “Adaptive Singular Value Cancelation of Ventricular Activity in Single-Lead Atrial Fibrillation Electrocardiograms.” *Physiological Measurement*, vol. 29, no. 12, 2008, pp. 1351–1369., doi:10.1088/0967-3334/29/12/001.

[4] Ródenas, Juan, et al. “Wavelet Entropy Automatically Detects Episodes of Atrial Fibrillation from Single-Lead Electrocardiograms.” *Entropy*, vol. 17, no. 12, July 2015, pp. 6179–6199., doi:10.3390/e17096179.

[5] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals (2003). Circulation. 101(23):e215-e220.

[6] Pan, Jiapu, and Willis J. Tompkins. “A Real-Time QRS Detection Algorithm.” *IEEE Transactions on Biomedical Engineering*, BME-32, no. 3, Mar. 1985, pp. 230–236., doi:10.1109/tbme.1985.325532.

[7] Tereshchenko, Larisa G., and Mark E. Josephson. “Frequency Content and Characteristics of Ventricular Conduction.” *Journal of Electrocardiology*, vol. 48, no. 6, 28 Aug. 2015, pp. 933–937., doi:10.1016/j.jelectrocard.2015.08.034.

[8] Alcaraz, Raúl, and José Joaquín Rieta. “Wavelet Bidomain Sample Entropy Analysis to Predict Spontaneous Termination of Atrial Fibrillation.” *Physiological Measurement*, vol. 29, no. 1, Jan. 2008, pp. 65–80., doi:10.1088/0967-3334/29/1/005.

[9] Kropf, Martin, et al. “ECG Classification Based on Time and Frequency Domain Features Using Random Forrests.” *2017 Computing in Cardiology Conference (CinC)*, 2017, doi:10.22489/cinc.2017.168-168.

[10] Husser, Daniela et al. “Frequency analysis of atrial fibrillation from the surface electrocardiogram.” *Indian pacing and electrophysiology journal* vol. 4,3 122-36. 1 Jul. 2004

Appendix A - Source Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Title: Final Project
% Filename: Tipton_FinalProject.m
% Author: Natalie Tipton
% Class: EGR 635/534
% Date: 12/5/19
% Instructor: Dr. Rhodes
% Description: This algorithm detects the presence of Atrial Fibrillation
%   or normal sinus rhythm in an ECG signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; close all;

path = cd;
files = dir('Test\Normal\*.mat');   %open all .csv files in current folder w/ CS0 in name
[num_files,z] = size(files);          %determine number of files read

for x = 1:num_files        % run code for each data file
        if x > 1             % after first file, delete all variable values except cumulative ones
        clearvars -except ent_atrial_activity ent_qrs_removed ent_raw_data ent_qq_interval...
        ent_wavelet x path num_files files classification;
        end

        cd('Test\Normal');
        load(files(x).name);       %read in all images in directory
        cd(path);

        val = (val - min(val)) ./ (max(val) - min(val));

        entropy_data = shannon_ent(val, 0.001);

        fs = 300;                    % sampling frequency
        len = length(val);           % length of data
        t = 0:1/fs:len/fs - 1/fs;   % time vector

        % plot original data
        figure
        plot(t,val);
        title('Original Signal'); xlabel('Time (s)'); ylabel('Amplitude');

        %wavelet entropy calculations on original data
        [aa, coeffs, entropy_wavelet] = wavelets(val, t, fs);
```

```
% revectorize data into overlapping vectors of size 300 with 50% overlap
aa_windowed = buffer(1:numel(aa),300,150);
aa_overlap = aa(aa_windowed(:,all(aa_windowed)))';
[row,col] = size(aa_overlap);   % find size of vectorized atrial activity

% find entropy of each second of data
for i = 1:row
entropy_moving_aa(i) = shannon_ent(aa_overlap(i,:), 0.001);
end

% average moving entropy values
entropy_aa = mean(entropy_moving_aa);

% plot atrial activity from wavelet decomposition
figure
plot(t,aa);
title('Atrial Activity'); xlabel('Time (s)'); ylabel('Amplitude');

% bandpass filter the original data for time domain calculations
ecg = removeNoise(5, 15, fs, t, val);

% find peaks in data
[pks, locs] = findpeaks(ecg);

% plot peaks
figure
findpeaks(ecg);
title('Peaks in ECG'); xlabel('Time (s)'); ylabel('Amplitude');

% pick out the QRS peaks and their location index in data
qrs_pks = pks(pks > 0.6);
qrs_locs = locs(pks > 0.6);

% plot QRS peaks over original data
figure
plot(t, val)
hold on
plot(t(qrs_locs), val(qrs_locs), 'o');
title('QRS Peaks'); xlabel('Time (s)'); ylabel('Amplitude');
legend('ECG', 'QRS Peak');

% find outliers due to noise that are much higher than avg QRS peak
outlier = find(qrs_pks > 1.5 * median(qrs_pks));

% % plot the outlier peaks
```

```
% figure
% plot(t,val)
% hold on
% plot(t(qrs_locs(outlier)), val(qrs_locs(outlier)), 'o');
% title('Outliers Detected'); xlabel('Time (s)'); ylabel('Amplitude');
% legend('ECG', 'Outlier');

% remove the outlier data up to the preceding and following QRS peak
for i = length(outlier) : -1:  1
        ecg(qrs_locs(outlier(i)-1) : qrs_locs(outlier(i)+1)) = mean(ecg);
end

% % plot data with outliers removed
% figure
% plot(t,ecg)
% title('ECG With Outliers Removed'); xlabel('Time (s)'); ylabel('Amplitude');

% find Q and S points
[q_points, s_points, t_2, v2] = QS_pointDetect(ecg, t);

qq_int = diff(q_points(1:end));       % find Q to Q intervals
entropy_qq = shannon_ent(qq_int, 0.001);   % calculate entropy of Q to Q intervals

% Plot Q and S points
figure
plot(t_2,v2)
hold on
plot(t_2(q_points(:)), v2(q_points(:)), 'o');
plot(t_2(s_points(:)), v2(s_points(:)), 'o');
title('Pan-Tompkins with Q and S detected');
legend('ECG', 'q', 's'); xlabel('Time (s)'); ylabel('Amplitude');

% smooth data for slope analysis
smoothed = smoothdata(ecg);
smoothed = smoothdata(smoothed);

% plot smoothed data
figure
plot(t,smoothed);
title('Smoothed data'); xlabel('Time (s)'); ylabel('Amplitude');

% refine location of the S point by finding where slope goes positive
for i = 1:length(q_points) - 1
        s_points(i) = s_points(i) + find(diff(val(s_points(i):q_points(i+1))) > 0,1,'first');
end
```

```
    % based on S-point location, find the peak of the t-wave
    for i = 1:length(q_points) - 1
            done = 0;
            start = s_points(i);        % start looking for slope change at s point
while done == 0
        % find where slope goes negative (peak of T-wave)
        top_peak = start + find(diff(val(start:q_points(i+1))) < 0,1,'first');

        % check if next 5 differences are also negative
        if (all(diff(val(top_peak:top_peak + 10)) < 0))
            t_top(i) = top_peak;      % if so, it is really the peak
            done = 1;
        else
            start = top_peak;     % if not, it was from noise, recheck starting from that point
        end
end
    end

    % based on peak of the t-wave, find the end of the t-wave
    for i = 1:length(q_points) - 1
            t_end(i) = t_top(i) + find(diff(val(t_top(i):q_points(i+1))) > 0,1,'first');
    end

    % subtract an average QRS complex from each pulse to remove ventricular activity
    [qrs, qrs_avg, qrs_removed] = QRSaveraging(val, q_points, t_end ,fs);

    % revectorize data into overlapping vecotrs of one second length
    qrs_rem_windowed = buffer(1:numel(qrs_removed),300,150);
    qrs_rem_overlap = qrs_removed(qrs_rem_windowed(:,all(qrs_rem_windowed)))';
    [row,col] = size(qrs_rem_overlap);

    % find entropy of each second of data with QRS-T removed
    for i = 1:row
            entropy_moving_qrs(i) = shannon_ent(qrs_rem_overlap(i,:), 0.001);
    end

    % find average entropy of all thirty seconds
    entropy_qrs_rem = mean(entropy_moving_qrs);

    % plot Q, S, and T points
    figure
    plot(t, val)
    hold on
    plot(t(q_points(:)), val(q_points(:)), 'o');
```

```
        plot(t(s_points(:)), val(s_points(:)), 'o');
        plot(t(t_end(:)), val(t_end(:)), 'o');
        title('Q, S, and T points detected on data');
        legend('ECG', 'q', 's', 't');
        xlabel('Time (s)'); ylabel('Amplitude (mV)');

        % Save all entropy values to an array
        ent_atrial_activity(x) = entropy_aa;
        ent_qrs_removed(x) = entropy_qrs_rem;
        ent_raw_data(x) = entropy_data;
        ent_qq_interval(x) = entropy_qq;
        ent_wavelet(x) = entropy_wavelet;

        % classify rhythm based on Q to Q interval entropy
        if(entropy_qq < 4)
    classification(x) = 0;     % normal sinus rhythm
        else
    classification(x) = 1;     % atrial fibrillation
        end

end

% writes data to an excel file
% myfile = 'Project_Results.xlsx' ;
% xlswrite(myfile,classification,'A42:J42')
% xlswrite(myfile,ent_qrs_removed,'A38:J38')
% xlswrite(myfile,ent_qq_interval,'A39:J39')
% xlswrite(myfile,ent_raw_data,'A40:J40')
% xlswrite(myfile,ent_wavelet,'A41:J41'

function [aa, coeffs, we] = wavelets(val, t, fs)

% This function computes 9th order wavelets.
% Inputs:
%       val = input signal
%       t = time vector for input signal
%       fs = sampling frequency of input signal
% Outputs:
%       aa = atrial activity of input signal. All content less than 10 Hz
%       aa_ent = entropy of atrial activity
%       aa_went = wavelet entropy of atrial activity
%       plots the content of each of the 9 wavelets
%

% obtain wavelet coefficients
```

```
[c,l] = wavedec(val, 9, 'dmey');

% determine the frequency ranges of each wavelet for labelling
range_start = fs/2;
r1 = range_start / 2;
r2 = r1 / 2;
r3 = r2 / 2;
r4 = r3 / 2;
r5 = r4 / 2;
r6 = r5 / 2;
r7 = r6 / 2;
r8 = r7 / 2;
r9 = r8 / 2;

% getting detailed and approximate coefficients
a9 = wrcoef('a', c, l, 'dmey', 9);
d1 = wrcoef('d', c, l, 'dmey', 1);
d2 = wrcoef('d', c, l, 'dmey', 2);
d3 = wrcoef('d', c, l, 'dmey', 3);
d4 = wrcoef('d', c, l, 'dmey', 4);
d5 = wrcoef('d', c, l, 'dmey', 5);
d6 = wrcoef('d', c, l, 'dmey', 6);
d7 = wrcoef('d', c, l, 'dmey', 7);
d8 = wrcoef('d', c, l, 'dmey', 8);
d9 = wrcoef('d', c, l, 'dmey', 9);

% plot and label all wavelets
figure
subplot(11,1,1)
plot(t, val, 'r');
title('Wavelets');
subplot(11,1,2)
plot(t,a9); ylabel('a9');xlabel(['0 - ', num2str(r9), ' Hz']);
subplot(11,1,3)
plot(t,d9); ylabel('d9');xlabel([num2str(r9), ' - ', num2str(r8), ' Hz']);
subplot(11,1,4)
plot(t,d8); ylabel('d8');xlabel([num2str(r8), ' - ', num2str(r7), ' Hz']);
subplot(11,1,5)
plot(t,d7); ylabel('d7');xlabel([num2str(r7), ' - ', num2str(r6), ' Hz']);
subplot(11,1,6)
plot(t,d6); ylabel('d6');xlabel([num2str(r6), ' - ', num2str(r5), ' Hz']);
subplot(11,1,7)
plot(t,d5); ylabel('d5');xlabel([num2str(r5), ' - ', num2str(r4), ' Hz']);
subplot(11,1,8)
plot(t,d4); ylabel('d4');xlabel([num2str(r4), ' - ', num2str(r3), ' Hz']);
```

```
subplot(11,1,9)
plot(t,d3); ylabel('d3');xlabel([num2str(r3), ' - ', num2str(r2), ' Hz']);
subplot(11,1,10)
plot(t,d2); ylabel('d2');xlabel([num2str(r2), ' - ', num2str(r1), ' Hz']);
subplot(11,1,11)
plot(t,d1); ylabel('d1'); xlabel([num2str(r1), ' - ', num2str(range_start), 'Hz']);


coeffs = [a9; d9; d8];  % coefficients that contribute to atrial activity

[nj, nk] = size(coeffs);

bottom = 0;       % set denominator to zero

% cycle through all coefficients
% double summation of C(j,k)^2 for k=1:Pj and j=1:N
for j = 1:nj
        for k = 1:nk
    bottom = bottom + coeffs(j,k)^2;
        end
end

top = zeros(1,nj);  % set numerator to all zeros

%find the numerator for each value of j
for j = 1:nj
        for k = 1:nk
    top(j) = top(j) + coeffs(j,k)^2; % numerator = sum of all k for each j
    E(j) = top(j)/ bottom;  % find E(j)
        end
end

% calculate wavelet entropy
we = -sum(E .* log2(E));

% recreate atrial activity by including only the low frequency bands
aa = d8+d9+a9+d7;

function ecg = removeNoise(f1, f2, fs, t, val)

% This function creates a bandpass filter for given corner frequencies
% Inputs:
%       f1 = low end corner frequency
%       f2 = high end corner frequency
%       fs = sampling frequency of signal
```

```
%       t = time vector for signal
%       val = signal being filtered
% Outputs:
%       ecg = filtered signal
%       plots filtered signal

% filter to remove baseline wander and noise from muscles
cutoff =[f1 f2]*2/fs;               % cutoff based on fs
order = 3;                  % order of 3 less processing
[a,b] = butter(order,cutoff);       % create filter coefficients
ecg = filtfilt(a,b,val);            % filter data
ecg = ecg/ max( abs(ecg));          % zero mean the data

% plot filtered data
figure
plot(t,ecg);
title('Baseline wander and noise removed'); xlabel('Time (s)'); ylabel('Amplitude');

function [qrs, qrs_avg, qrs_removed] = QRSaveraging(val, q_points, t_end,fs)

% This function finds the average QRS complex for a given input signal
% Inputs:
%       val = input ECG signal
%       q_points = Detected locations of Q points
%       t_end = Detected locations of end of T wave
% Outputs:
%       qrs = the matrix with all QRS-T complexes, 1 in each row
%       qrs_avg = the average of all QRS-T complexes
%       qrs_removed = signal with GRS-T segments cancelled out

% find length of each QRS-T segment
for i = 1 : length(t_end)
        qrs_size(i) = t_end(i) - q_points(i);
end
% find longest QRS-T portion
max_size = max(qrs_size);

% create a vector for average QRS-T portion the length of largest QRS-T
qrs = zeros(length(t_end), max_size);

% separate out each QRS-T segment
for i = 1 : length(t_end)
        cur_len = t_end(i) - q_points(i) + 1;   % find length of current QRS-T
        qrs(i,1:cur_len) = val(q_points(i):t_end(i));   % place QRS-T in new row
end
```

```
% find average of all QRS-T segments
qrs_avg = mean(qrs(1:length(t_end),:));

% find size of QRS-T matrix
[m,n] = size(qrs);

% duplicate input signal for removal of QRS-T segments
qrs_removed = val;

% replaces each QRS-T segment in input signal with the difference between
% that segment and the average QRS-T segment
for i = 1 : m
        qrs_removed(q_points(i):q_points(i) + n - 1) = qrs(i,:) - qrs_avg;
end

% removes any partial QRS-T segments that were not cancelled out
%qrs_removed(end-300:end) = 0;
%qrs_removed(1:300) = 0;

% histo = hist(qrs_removed, min(qrs_removed):max(qrs_removed)); % Form a histogram of the input
% qrs_p = histo/sum(histo); % Estimate the probabilities (Pi)
% %aa_p = abs(qrs_p(:) / sum(qrs_p));
% qrs_p = qrs_p(qrs_p ~= 0);
% qrs_ent = -sum(qrs_p.*log2(qrs_p));

% time vector for average QRS-T segment
t_avg = 0:1/fs:length(qrs_avg)/fs-1/fs;

% plot average QRS-T segment
figure
plot(t_avg,qrs_avg);
title('Average QRS Complex'); xlabel('Time (s)'); ylabel('Amplitude');

% time vector for signal with QRS-T segments removed
t = 0:1/fs:length(qrs_removed)/fs - 1/fs;

% plot signal wit QRS-T segments removed
figure
plot(t,qrs_removed);
title('Data with QRS Complexes cancelled out'); xlabel('Time (s)'); ylabel('Amplitude');

function [q_points, s_points, t_2, v2] = QS_pointDetect(ecg, t)

% This function detects the location of Q and S points in an ECG signal
```

```
% Inputs:
%       ecg = ECG signal on which Q and S points are being detected
%       t = time vector for input ECG signal
% Outputs:
%       q_points = array of index locations of input signal where Q points are
%       s_points = array of index location of input signal where S points are
%       t_2 = time vector for Pan-Tompkins double derivative result
%       v2 = Pan-Tomplins double derivative results

% for completion of pan-tompkins method find derivative of data
dx = ecg(2:end) - ecg(1:end-1);
dt = t(2:end) - t(1:end-1);
v1 = dx./dt;

% create time vector for derivative
t_2 = t;
t_2(end) = [];

% find derivative of derivative of data
dx = v1(2:end) - v1(1:end-1);
dt = t_2(2:end) - t_2(1:end-1);
v2 = dx./dt;

% adjust time vector again
t_2(end) = [];

% % plot the pan-tompkins result
% figure
% plot(t_2, v2);
% title('Pan-Tompkins'); xlabel('Time (s)'); ylabel('Amplitude');

% % plot all peaks
% figure
% findpeaks(v2);
% title('Peaks in Pan-Tompkins result'); xlabel('Time (s)'); ylabel('Amplitude');

% find all large peaks in pan-tompkins and the location of them
[pks_pan, locs_pan] = findpeaks(v2);
qs_peaks = pks_pan(pks_pan > (0.3 * max(pks_pan)));
qs_locs = locs_pan(pks_pan > (0.3 * max(pks_pan)));

% find number of Q and S peaks in pean-tompkins
len_pan = length(qs_peaks);

% every other pan-thomkins peak is a Q or S point
```

```
q_points = ceil(qs_locs(1:2:end));
s_points = ceil(qs_locs(2:2:end));

% if there is more than 100 points of distance between first Q and S
% points, then the first point is actually an S point, not a Q point
if(abs(q_points(1) - s_points(1)) > 100)
        q_points(1) = [];      % get rid of first point because an S without a Q doesn't help us
        temp = s_points;       % temporarioy save the s_points
        s_points = q_points;      % make s_points equal to q_points
        q_points = temp;       % make q_points equal to temporarily saved s_points
end

qtos_dist = 30;                % an average Q to S distance is 30 for initial estimation

% finding if any Q or S points were not detected originally
for i = 1 : length(q_points)-1
        if((s_points(i) - q_points(i)) < 100)   % if less than 100 dist btwn same index Q and S
    qtos_dist(i) = s_points(i) - q_points(i);   % both correctly detected, record the distance
        else      % if distance more than 100 btwen same index Q and S, something was missed
    avg_dist = ceil(mean(nonzeros(qtos_dist))); % calculate the average Q to S distance
    qtos_dist(i) = avg_dist;
    s_points = [s_points(1:i-1), q_points(i) + avg_dist, s_points(i:end)]; % add in missing point at average
distance away

    % we must switch following Q and S points to make room for added point
    temp = q_points(i+1:end);
    q_points = [q_points(1:i), s_points(i+1:end)];
    s_points = [s_points(1:i), q_points(i+1:end)];
        end
End

function hx = shannon_ent(x,r)
%hx = shannon_ent(x,r)
%
%This function will calculate Shannon's entropy as the sum of p(xi)*logb
%INPUTS: signals X and tolerance r [default is 0.001].
%
%OUTPUT: hx entropy
%
%
%Define function name
% FuncName = 'shannon_ent';
%
% Written by Dr. Samhita Rhodes
% Adapted by Natalie Tipton
```

```
%Check input parameters
if(nargin<2)
        fprintf(1,'Must give all input parameters\nType help Entropy\n\n');
        return
end
if(~exist('r') | isempty(r))          % default value of r = 0.001
        r=.001;
end;

%Obtaining Entropy
x=(x-mean(x))./std(x);  % normalize data
N = length(x);              % find length of data
x1 = unique(x);             % returns x with no repetititive values
hx = 0;             % initialize entropy variable
for i=1:length(x1)
        nx(i) = sum(abs(x1(i)-x)<=r);   % finds all values within r of original normalized data
end
px = nx./sum(nx);           % calculated pdf
px1 = px(find(px));         % gets locations of each pdf value
hx = hx + sum(px1.*log2(px1));  % calculates entropy
hx = -hx;                   % makes entropy positive
```