The Art of Scientific Computation - Respiratory Rates Final Report

Tyson Field (695118)

Contents

1 Introduction

An electrocardiogram (ECG) is a graph of voltage over time that is produced by the electrical activity of the heart. Here, the data has been obtained from [1] and [2]. The data is in A/D units and there are 100 samples per second. It is expected that there are 200 A/D units per millivolt for this data. Labelling the units A/D refers to the fact that the signal has undergone analogue to digital conversion and so 200 A/D units may not correspond exactly to 1 millivolt. For this reason, we will use A/D units for our data. The plot below shows an ECG signal of two heart beats.

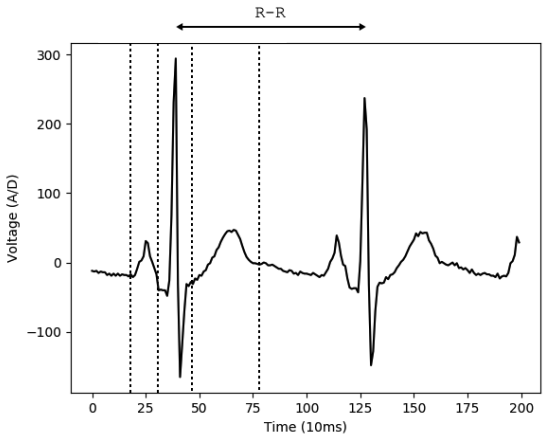

Figure 1: Two heartbeats

The basic physiological aspects of the ECG signal are described in [3] as follows. The change in electric charge distribution in a cell is called depolarisation. When this occurs in a muscle fibre, it creates a potential between the stimulated (electrically negative) part of the fibre and the

unstimulated part of the fibre. By placing electrodes on the surface of the muscle fibre, the current can be measured using a galvanometer. In the case of an ECG, one there are three primary defining characteristics of a heartbeat that can be measured. These are indicated by the dotted lines in Figure 1. The first interval is the P-wave. This occurs during depolarisation of the left and right atria. The left and right atria then contract as both ventricles undergo depolarisation. The second interval is known as the QRS complex. This is when the ventricles become depolarised and contract, while the atria re-polarise and relax. Finally, the ventricles are re-polarised and relax. This is indicated by the third interval: the T-wave.

The arrow at the top of Figure 1 defines the R-R interval, which is the distance between two adjacent peaks. The R-R interval is useful to us, as it gives an estimate of the heart rate at any given time in the data. Our aim is to accurately estimate the heart rate from the ECG data, and then find correlations between the heart rate data (R-peak amplitudes and R-R intervals) and respiratory rate. This should allow us to estimate the respiratory rate of an individual from ECG data. We hope to develop algorithms to efficiently and accurately compute the respiratory rate of an individual from an ECG signal. The data obtained from [1] and [2] will be used as a benchmark. This data was obtained from individuals with apnoea, which is useful for us as we would like our algorithm to detect respiratory issues.

2 Finding Peaks

One of the challenges when analysing the ECG data is finding the peaks. Figure 2 is a flowchart that shows an algorithm that we have developed to do this. Below, we have shown a plot of five heartbeats with the corresponding peaks that have been located using our algorithm.

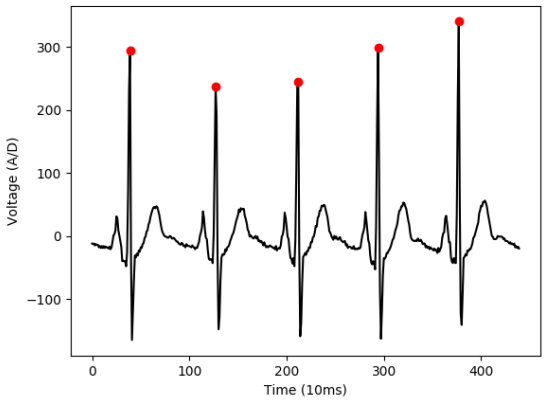

Figure 3: Five heartbeats with peaks located

We can test this algorithm statistically by computing a sufficient number of peaks. Using our algorithm to find the peaks from the first 100,000 samples in one of our ECG data files, we find that the mean and standard deviation of the R-R distance is

$$\mu_{R-R} = 827.25ms \tag{1}$$
$$\sigma_{R-R} = 108.56ms. \tag{2}$$

In our 100,000 samples, we found 13 $R-R$ distances that differ significantly from the mean ($\pm 3\sigma_{R-R}$). Let us take a closer look at some of these. In the plot below, we see that our algorithm has incorrectly detected some peaks.

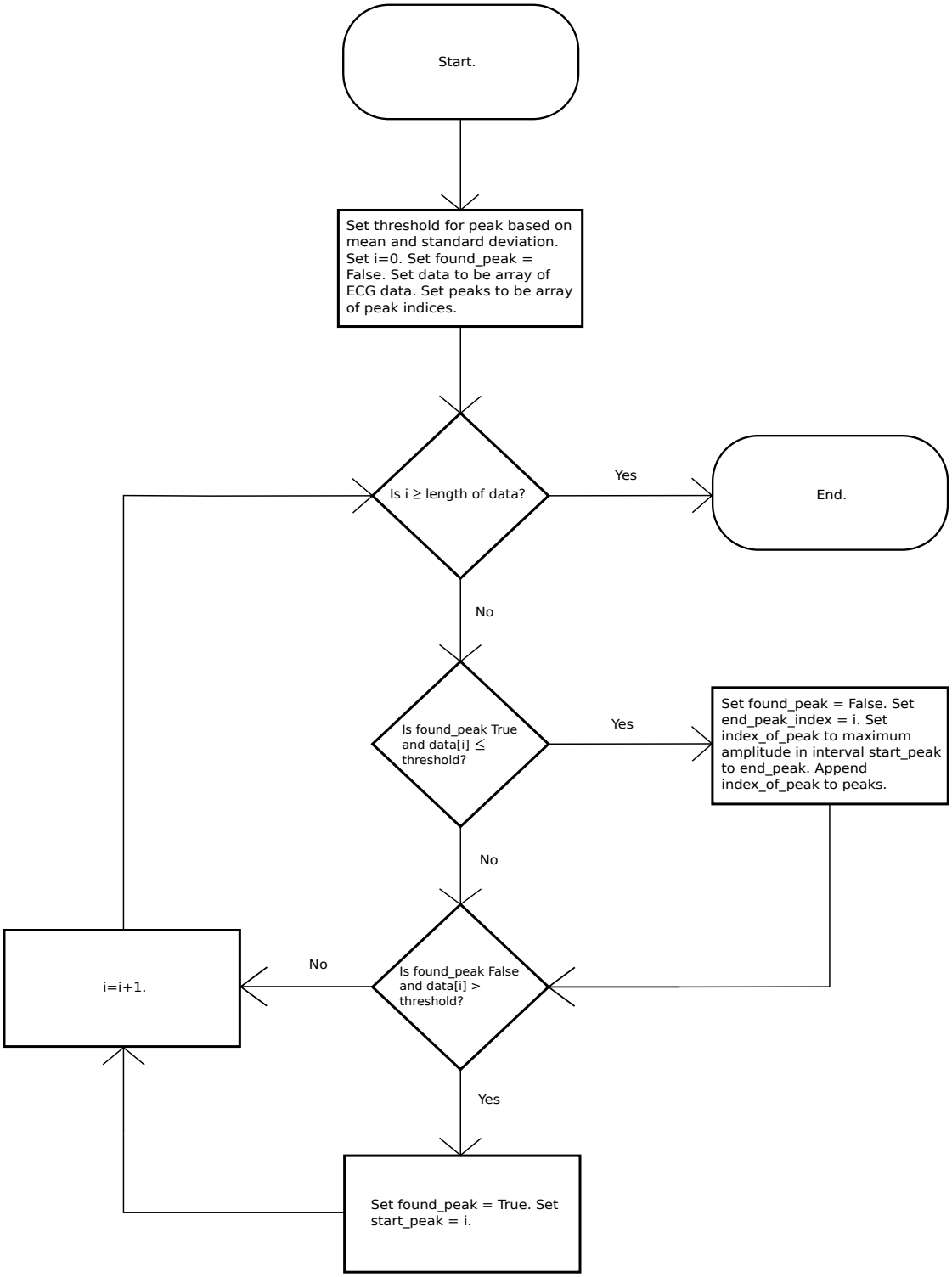

Figure 2: Flow Chart for Peak Detection Algorithm

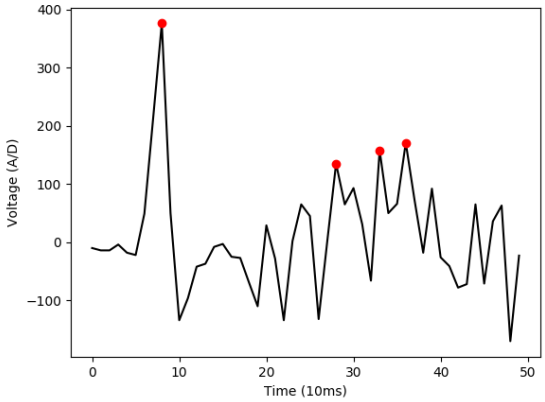

Figure 4: False peak detection

Here, $t = 0$ corresponds to sample number 46450. There are multiple ways in which one could address this problem. One method is to increase the threshold for a peak so that less peaks are detected. However, this might result in our algorithm missing true peaks. Perhaps a better solution is to recompute the mean and standard deviation of the data in segments. If we look at the samples $46450 - 46500$ in isolation, our algorithm detects only the true peak. This is a great improvement, and provides us motivation to read our data in segments.

To implement this, we define an ECG class which corresponds to a given segment of data. This class has methods that allow us to read data, detect peaks, and more. We read the data in segments with an overlap region. For example, we can use events 0-1000 for the first data segment and events 800-1800 for the second data segment. The overlap region is chosen to be sufficiently large to ensure that all peaks are detected. Without an overlap region, our algorithm would not detect peaks that occur between two segments. The following is an outline for a function that reads the data in segments.

```
define extract_ecg_data(filename, outfilename, offset, events):
    for each segment in filename:
        create instance of ECG class corresponding to given data segment
        create instance of ECG class corresponding to next data segment
        call find_peaks method for both instances and store peaks
        remove peaks in overlap region
        write peak and amplitude data to outfilename
```

After implementing this technique for reading data, our algorithm improved significantly. We processed data for the entire file a01.dat, which consisted of 2,957,000 events. Of these, we found only four falsely detected peaks. An example of such a falsely detected peak occurs at the T wave of the first heartbeat in Figure 5.

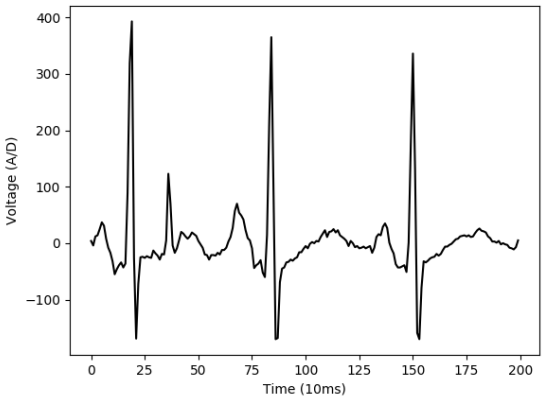

Figure 5: False peak detection with buffer

Unfortunately, we were unable to resolve this by decreasing the size of the data segments we processed. Although these false peaks are very rare in this data set, some of the other data sets contain false peaks at the T waves. For example, Figure 6 shows a sample of data from the data set a02.dat.

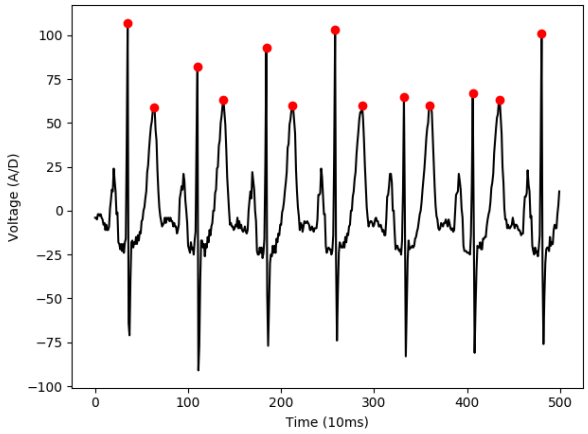

Figure 6: T wave false peak detection

This is therefore an important issue to resolve. A useful approach is to notice that the R peaks are immediately followed by large troughs and that this is not the case for the T wave peaks. So, we can build on our peak detection algorithm by verifying that a given peak is followed by a significant trough before appending it to our array of peaks. We chose our cutout for a trough to be 1 standard deviation below the mean. Testing this, we find that the algorithm works on a02.dat but not as well as on a01.dat. For a02.dat, the number of falsely detected peaks is on the order of 10. These mostly occur to be due to noise.

Although we have designed an effective peak finding algorithm for the data sets a01.data and a02.dat, there is still work to be done in making the algorithm sufficiently robust. For example, in the data set b01.dat there are segments of data where the true R peaks are not followed by significant troughs. Because of this, our modified algorithm will not detect the true peaks in b01.dat. The challenge is that a good characterisation of the T peaks appears to depend on the data set. In b01.dat for example, we could modify our algorithm to look at distances between the R and T peaks. In b01.dat, this distance when a R peak follows an T peak is significantly shorter than when a T peak follows an R peak. However, in a02.dat this effect does not appear to be pronounced enough to distinguish the R and T peaks stochastically in this manner. A promising approach is for a more robust peak detection algorithm is to compute the width of the peaks (or equivalently compute the derivative of the peaks).

3 Filtering

Filtering is a common technique in signal processing. The idea is to take the Fourier transform of the signal and attenuate unwanted frequencies. Then, taking the inverse Fourier transform gives us back a signal with the unwanted frequencies attenuated. Given a list of complex numbers $x_0, \ldots, x_{n-1}$ (in our case this corresponds to our values for voltage), the discrete Fourier transform is defined by

$$X_k = \sum_{j=0}^{n-1} x_j e^{-2\pi ijk/n}, \qquad k = 0, 1, \ldots, n-1. \tag{3}$$

Computing this naively is an $O(n^2)$ algorithm. However, there are faster algorithms for this known as fast Fourier transform (FFT) algorithms. An important application of filtering is to remove noise. This can be done by the use of a low-pass filter, which attenuates high frequencies. The high frequencies can be attenuated by multiplying the Fourier transform of the data by the Gaussian kernel

$$Ce^{-ax^2}. \tag{4}$$

Below is a plot of a Gaussian kernel low-pass filter applied to Figure 4.

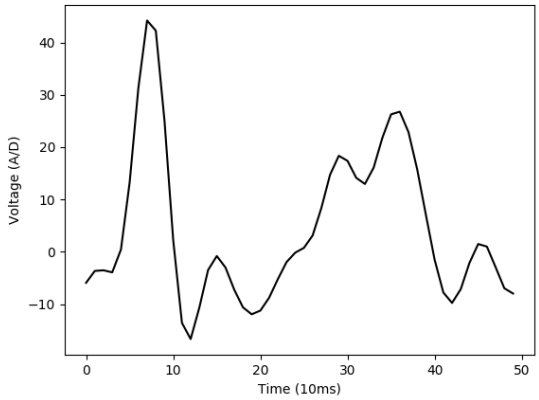

Figure 7: Low pass filter to remove noise

We can see that this effectively removes noise in the data while maintaining the important features of the ECG signal such as the T-wave. Another application of filtering is to remove baseline wander. This occurs when the baseline signal (i.e. the part of the signal not corresponding to the P wave, T wave, or QRS complex) drifts in either direction. Removing baseline wander is useful for extracting respiratory data from an ECG signal as baseline wander is a non-respiratory effect that occurs due to variation of action between the sensor and body [6, pg. 2]. Fortunately, the data we have been working with does not appear to exhibit a significant amount of baseline wander.

4 Respiratory Data

Accompanied with the ECG data are four sets of respiratory data. These include chest and abdominal signals obtained using respiratory inductance plethysmography, oronasal airflow, and SpO_2 oxygen saturation. Oronasal airflow refers to the pressure measured from the nose and mouth of the participant. SpO_2 oxygen saturation is a measure of arterial oxygen saturation. Oxygen saturation is a measure of how much haemoglobin is oxygen-saturated (as a percentage). Respiratory inductance plethysmography (RIP) is a method of obtaining the volume inspired by a participant at any given moment by studying the self-inductance of a pair of wire coils attached to the body of the participant. We will not discuss the physiological details of this data in detail, and we will instead focus on the corresponding data.

Let us look at some respiratory data samples without the presence of apnoea.

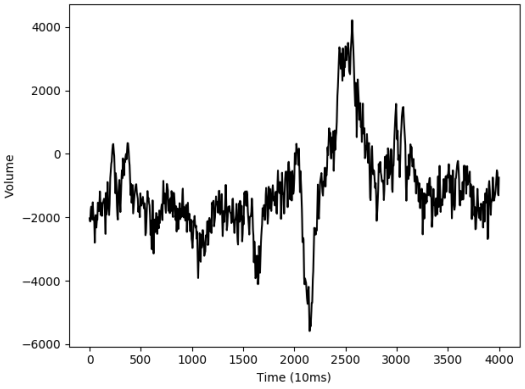

Figure 8: Chest RIP data

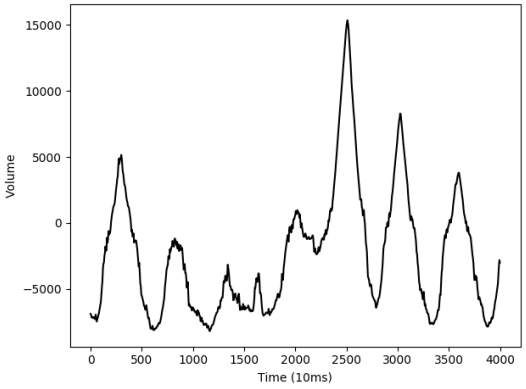

Figure 9: Abdominal RIP data

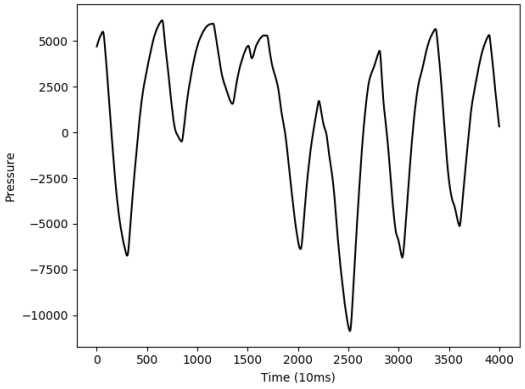

Figure 10: Oronasal airflow

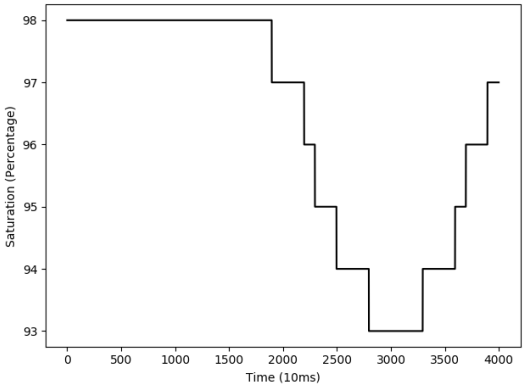

Figure 11: Oxygen saturation

We have left out the units of this data as they are unknown. Fortunately, the units are not important for our purposes. The chest RIP data appears to be quite noisy and it is not clear how one should determine the respiratory rate in this case. The abdominal RIP data appears easier to work with, but we have decided to focus on using the oronasal airflow data as our benchmark. We can see the connection between the oronasal airflow data and the oxygen saturation data. Looking at the oronasal airflow data, we see a pause in inhalation between 5 and 15 seconds. Roughly 5 seconds later, we see a delayed drop in oxygen saturation caused by the pause in breathing. This is expected since the oxygen saturation data is SpO_2 data, and it is known that SpO_2 desaturation events are delayed compared to irregular breathing events [5, pg. 3].

One method for finding correlations between this data and the ECG data is to compute the cross-correlation of the two time series. Given two real-valued discrete functions f and g defined on the integers 0 to N, their cross-correlation is defined as

$$(f \star g)(n) = \sum_{m=0}^{N} f(m)g(m+n). \tag{5}$$

The cross-correlation helps us to find correlations in two signals independent of their phase. The variable n acts as the phase shift of the signal g relative to f and so if a correlation between f and g occurs at some phase shift, this correlation will be seen in the function $f \star g$. Figure 12 shows the cross-correlation of oronasal airflow and the ECG signal for the first 10 seconds of data.

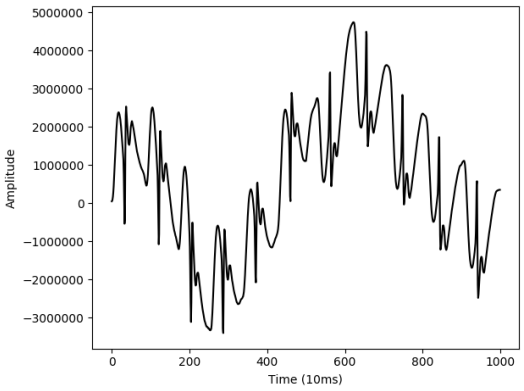

Figure 12: Cross-correlation 10 seconds

In Figure 12, we see a clear correlation between the two data sets. The sinusoidal behaviour of the cross-correlation suggests that the amplitude of the ECG peaks is modulated. We will make use of this fact in Section 5.

Recall that Figures 8 to 11 do not exhibit signs of apnoea. To compute the respiratory rate during periods of apnoea, we will need to look at data samples containing signs of apnoea. Figure 13 shows sample osonasal airflow data when apnoea is present.

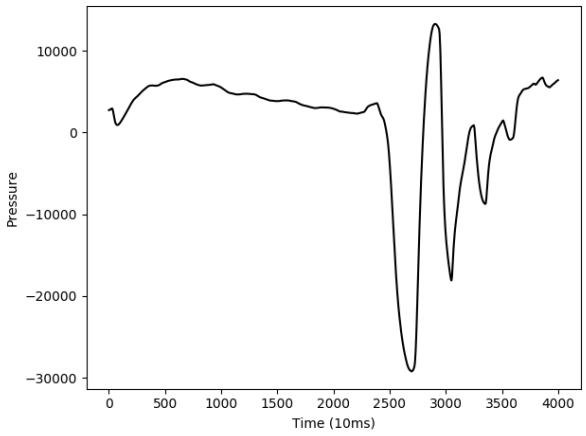

Figure 13: Oronasal airflow with apnoea

In Section 5, we will demonstrate an algorithm that computes the respiratory rate from the ECG signal. Before doing this, we need a benchmark respiratory rate. This can be obtained from the oronasal airflow data. The algorithm is essentially the same as the peak detection algorithm used for the ECG data, except now we will count sufficiently large troughs. These troughs correspond to inhalation. We check that the trough is sufficiently large so that we expect the inhalation to be followed by expiration. The algorithm for locating the breaths in the airflow data is as follows:

1. Iterate through array of airflow data. If the pressure falls sufficiently low (below some threshold), we mark the beginning of a trough. Our threshold for the start of a trough is mean - (tolerance $\times$ std dev) where mean is the mean of the airflow data in the given data segment, tolerance is a constant factor that can be modified, and std dev is the standard deviation of the airflow data in the given data segment.

2. Once we have the start of a trough, continue to iterate through the data until the pressure increases back above the threshold. Once this occurs, mark the end of the trough.

3. Locate the index of the minimum pressure between the start and end of the trough. Append this index and continue to iterate through the array.

4. Return the array of trough indices. Note that these indices include the offset so they give the absolute position of the trough within the .dat file.

Figures 14 and 15 show sample oronasal airflow data with the located breaths identified using the algorithm just described.

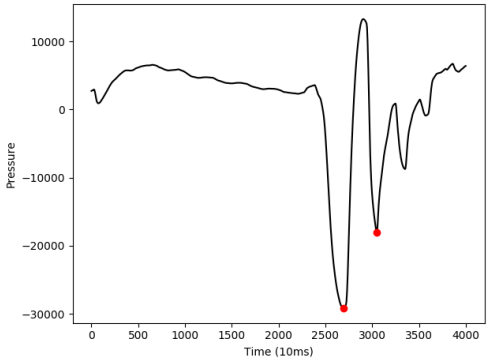

Figure 14: Oronasal airflow with apnoea and breaths identified

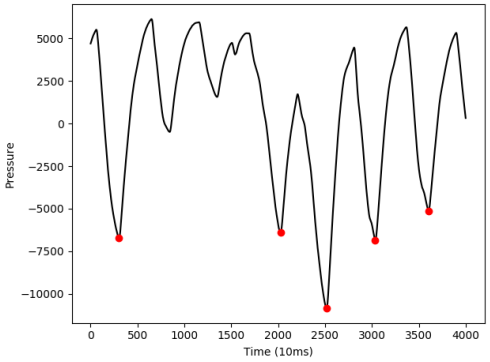

Figure 15: Oronasal airflow with no apnoea and breaths identified

For these plots, our threshold was set to 1 standard deviation for the no apnoea data and 2 standard deviations for the apnoea data. We arrived at these thresholds through trial and error on small segments of the a01r.dat data. To ensure that the algorithm is effective for both apnoea and no apnoea data, we have chosen to set a default threshold of 1.5 standard deviations for our algorithm. This benchmark algorithm appears to work very well and so we will now focus on the algorithms used to compute the respiratory rate from the ECG signal.

5 Respiratory Rate from ECG Amplitude Modulation

Our first method for computing the respiratory rate from the ECG data will study the amplitude modulation of the ECG peaks. Our motivation for this comes from studying the cross-correlation of the ECG signal with the respiratory data. Furthermore, studying the frequency and amplitude modulation of the ECG signal to compute respiratory rate are known approaches in the literature [4]. We will attempt to implement these approaches using our own statistical methods. The diagram below shows the ECG signal from the first 20 seconds of a01.dat with peaks identified.

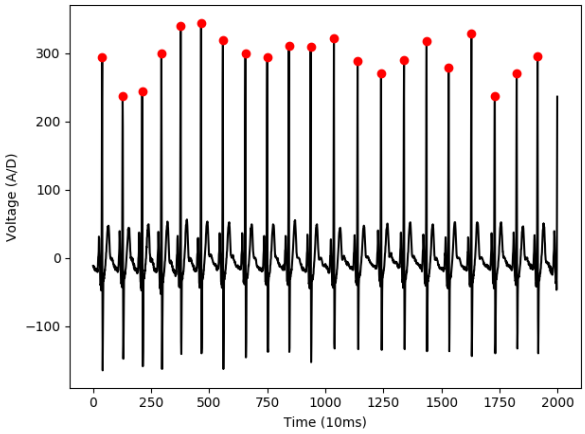

Figure 16: ECG peaks

From this, we can see that the peaks of the ECG signal are roughly sinusoidal. In fact, if we compare with Figures 9 and 10, we see that the peaks and troughs of the ECG peaks correspond roughly to expiration and inhalation respectively. Implementing an algorithm to find the relevant peaks and troughs is less straightforward than finding the ECG peaks. First, we need to be clear about what we mean by the respiratory rate. If we look at times between 12 seconds and 17 seconds in Figures 9 and 10, we can see a small respiratory signal. However, these events are followed by the oxygen desaturation event starting at 17 seconds in Figure 11. Hence, a good characterisation

for poor respiration over some time period is small variance in the RIP signal. This is useful as it picks up poor respiration when the oronasal airflow pressure fluctuates around zero. A naive method that looks for the peaks and troughs in the oronasal airflow data would pick up this noise as a false respiration signal.

Given this characterisation for good respiration in the RIP signals, we would hope that something similar arises in the ECG signal. Figures 17 and 18 show the running variance of the ECG peaks and oronasal airflow data respectively. This data has been sampled from a region without the presence of apnoea and the running variance computes the variance of the next 1000 samples at any given time.

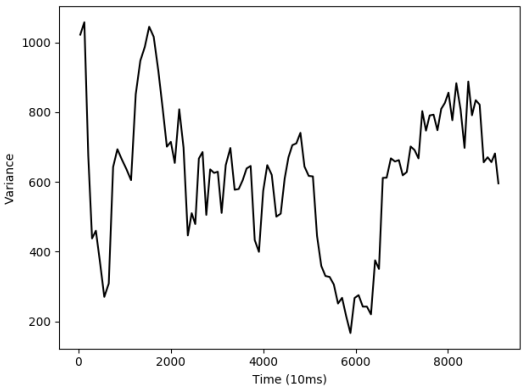

Figure 17: ECG peaks running variance

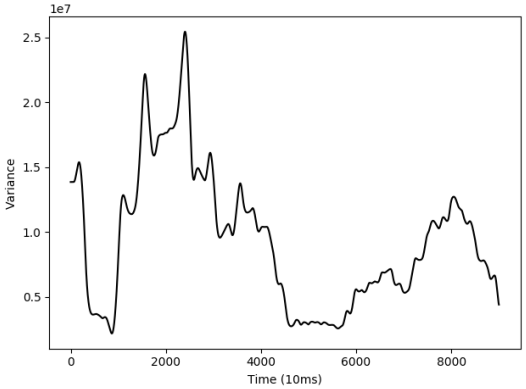

Figure 18: Oronasal airflow running variance

It appears that the variance of the ECG peaks roughly coincides with the variance of the oronasal airflow data. This is promising as we can attempt to detect respiration by searching for sufficiently large variance in the peaks of the ECG signal. The algorithm we have used works as follows:

1. Iterate through array of ECG peaks and search for peaks. Our condition for finding the start of a peak is that there are two successive increases in the amplitude of the data with a possible gap of one event between the two increases. In other words, index i is the start of a peak if $\text{data}[i+1] > \text{data}[i]$ and either $\text{data}[i+2] > \text{data}[i+1]$ or $\text{data}[i+3] > \text{data}[i+1]$ where data is an array of ECG R-peak indices.

2. If we have found a peak, continue until we find the end of the peak. We assume that we are still at a peak if there is an increase in the data with a possible gap of one event. In other words, $\text{data}[i+1] > \text{data}[i]$ or $\text{data}[i+2] > \text{data}[i]$.

3. Given the start and end points of the peak, compute its variance and compare with the variance of the current segment of data.

4. If the variance is sufficiently large (compared to the standard deviation of the amplitude data), store the index of the end of the peak.

The purpose of the "gaps" is to account for the noise in the signal. It appears that there are not enough R-peaks in a given time interval to effectively use filtering to locate the peaks, and so we needed this alternative approach to account for noise. The algorithm described above is implemented using the resp_peaks_amplitude amplitude method in our code.

In Section 7, we test our algorithm on large samples of data. Here, we present some plots of the located breaths using our algorithm. We show enough plots to demonstrate examples of the algorithm working as expected and examples of the algorithm not working as expected. Our examples of the algorithm working well are taken from a01.dat. Our examples of the algorithm working not so well are taken from a02.dat. This is because the pitfalls of our algorithm are easy to see when applied to the more challenging data a02.dat.

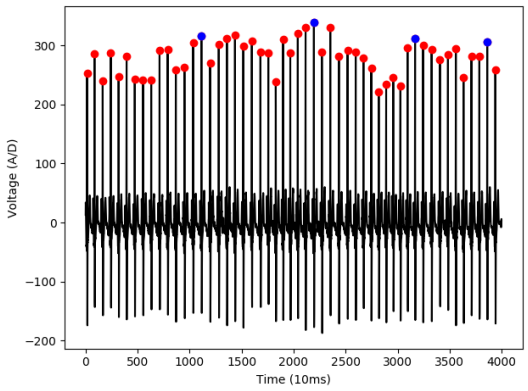

Figure 19: Amplitude modulation ECG data no apnoea a01.dat

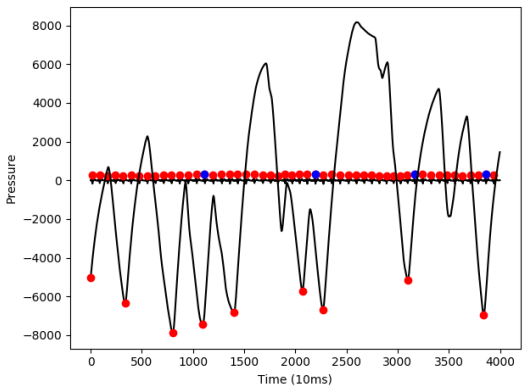

Figure 20: Amplitude modulation airflow data no apnoea a01r.dat

Figures 19 and 20 are great examples of our algorithm working well. The red and blue dots in Figure 19 correspond to R-peaks and breaths found using our amplitude modulation algorithm respectively. The troughs with red dots in Figure 20 correspond to breaths found using our benchmark algorithm applied to the oronasal airflow data. Recall that our benchmark algorithm finds breaths by searching for troughs of sufficiently large size in the oronasal airflow data. We have overlaid Figure 19 onto 20 to compare our algorithm to its benchmark.

It appears that in this case our algorithm outperforms the benchmark. In Figure 20, we see falsely detected breaths (from our benchmark algorithm) between 0 and 15 seconds. During this period, it appears that the airflow data corresponds to multiple instances of inspiration without expiration. Hence, we should only count one breath. Our amplitude modulation algorithm happened to count exactly one breath in this period. A similar event also occurs between 17 and 25 seconds. Finally, our amplitude modulation detects the two breaths between 30 and 40 seconds.

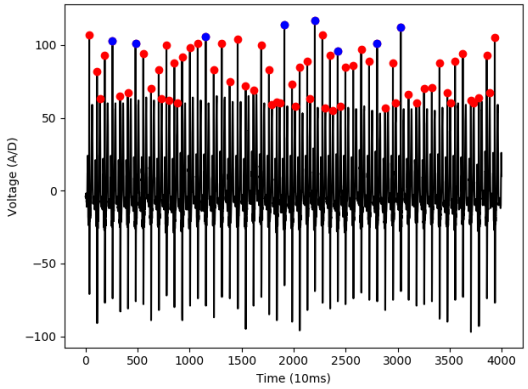

Figure 21: Amplitude modulation ECG data no apnoea a02.dat

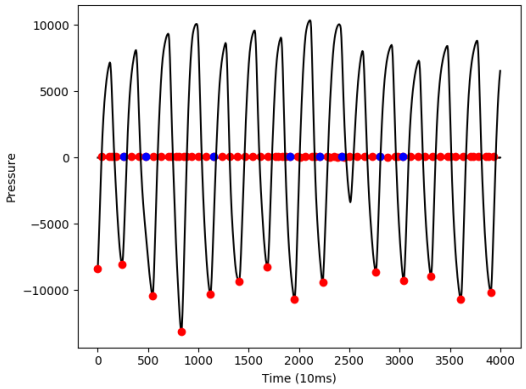

Figure 22: Amplitude modulation airflow data no apnoea a02r.dat

Figures 21 and Figures 22 show examples of our algorithm not working as expected. Looking at Figure 22, we see that our amplitude modulation algorithm underestimates the respiratory rate. This appears to be an issue with not having enough R-peaks data to allow a large number of respiratory peaks to form within a given time interval. In Figure 21, we can see that the R-peaks data fluctuates between high and low voltage many times as a result of this. We cannot easily modify our amplitude modulation algorithm to resolve this. If we relaxed the condition on finding a respiratory peak (so that for example, we only require data$[i+1]$ to be sufficiently larger than data$[i]$ to have found a respiratory peak), then we would expect our algorithm to overestimate the respiratory rate when the respiratory rate is low (i.e. in the presence of apnoea). We accept this as a fundamental limitation of our analysis: for a sufficiently large respiratory rate we will not be able to locate all of the respiratory information in the ECG R-peaks data. Fortunately, our algorithm manages to capture at least half of the breaths in this instance so that it is still partially useful.

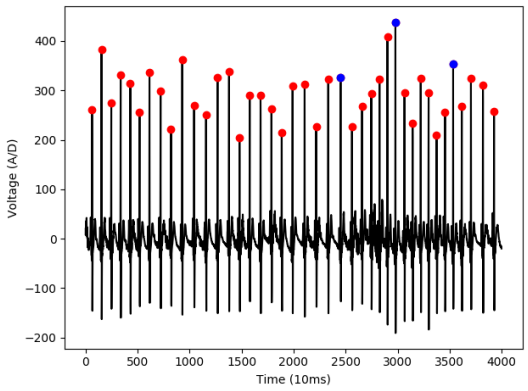

Figure 23: Amplitude modulation ECG data apnoea a01.dat

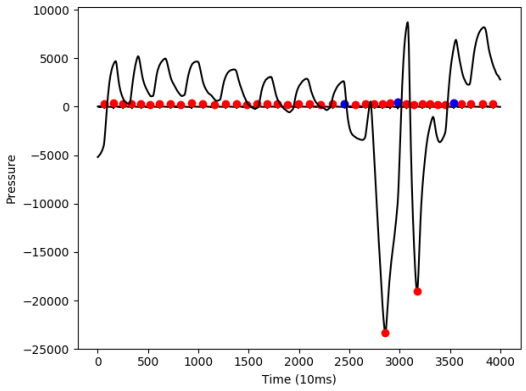

Figure 24: Amplitude modulation airflow data apnoea a01r.dat

Figures 23 and 24 show our algorithm working as expected in the presence of apnoea. In Figure 24, we see that our algorithm has detected the two breaths found by the benchmark algorithm. There is one false breath detected at approximately 24 seconds, but the algorithm does well to not falsely detect any breaths between 0 and 24 seconds. Unfortunately, the success of the algorithm in the presence of apnoea depends on the ECG R-peaks data being sufficiently well-behaved.

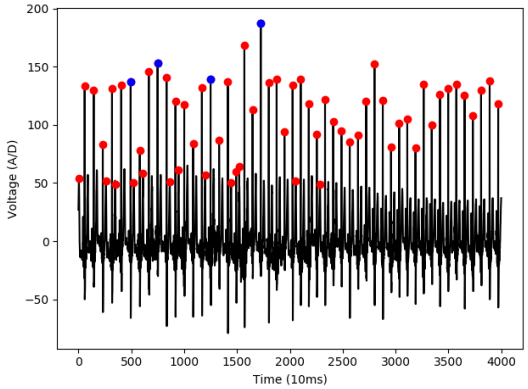

Figure 25: Amplitude modulation ECG data apnoea a02.dat

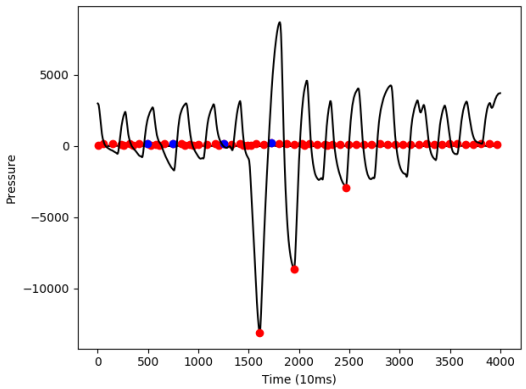

Figure 26: Amplitude modulation airflow data apnoea a02r.dat

Figures 25 and 26 demonstrate examples of our algorithm falsely detecting breaths in the presence of apnoea. We detect at least one true peak, but there are at least two examples (between 5 and 10 seconds) where we have falsely detected a breath. In this case, one potential cause are the low amplitude R-peaks in the ECG signal. If we ignore these R-peaks, we can see that our algorithm should have only counted a single peak between 0 and 10 seconds. These plots demonstrate a significant limitation of our algorithm: it tends to overestimate the respiratory rate in the presence of apnoea for sufficiently complicated ECG signals.

Our amplitude modulation approach appears to have been moderately successful. In Section 7, we will test the algorithm on significantly larger samples of data. Naturally, one might ask if studying the frequency modulation of R-R intervals could provide a better approach. We will investigate this in the next section.

6 Respiratory Rate from ECG Frequency Modulation

There is physiological motivation for studying the frequency modulation of the R-peaks. It is known that the R-R interval decreases during inspiration and increases during expiration [9]. The algorithm we have used is essentially the same as the algorithm we used for studying amplitude modulation. For completeness, we will provide an outline of the algorithm:

1. Iterate through array of R-R intervals and search for troughs. Our condition for finding the start of a trough is that there is a decrease in the data (i.e. $\text{data}[i+1] < \text{data}[i]$).

2. If we have found a the start of a trough, continue until we find the end of the trough. We assume that we are still at a trough if there is a decrease in the data with a possible gap of one event. In other words, $\text{data}[i+1] < \text{data}[i]$ or $\text{data}[i+2] < \text{data}[i]$.

3. Given the start and end points of the trough, compute its variance and compare with the variance of the current segment of data.

4. If the variance is sufficiently large (compared to the standard deviation of the R-R interval data), store the index of the end of the trough.

Figures 28 to 34 demonstrate the strengths and weaknesses of the frequency modulation algorithm. Here, we see strong similarities with the amplitude modulation algorithm. The blue dots represent the breaths located using the frequency modulation algorithm and the red dots represent the breaths located using the benchmark algorithm. In Figures 27, 28, 31, and 32, we see the algorithm working well. However, we see the same issue of under-counting in Figures 29 and 30, and over-counting in Figures 33 and 34.

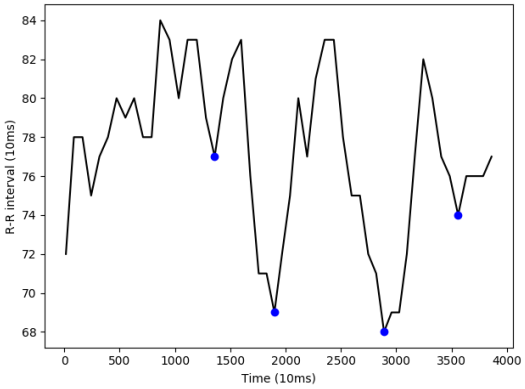

Figure 27: Frequency modulation ECG data no apnoea a01.dat

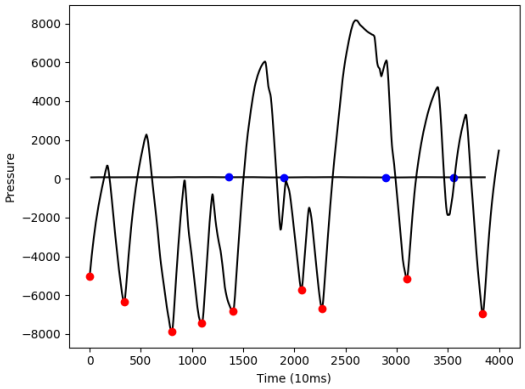

Figure 28: Frequency modulation airflow data no apnoea a01r.dat

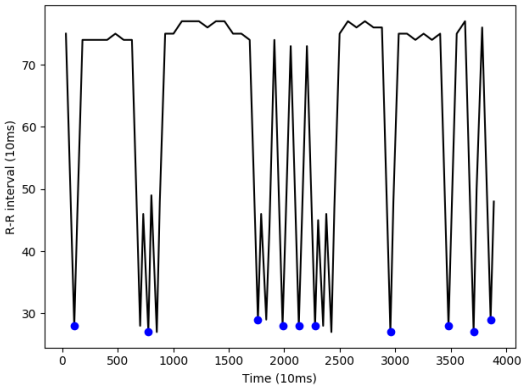

Figure 29: Frequency modulation ECG data no apnoea a02.dat

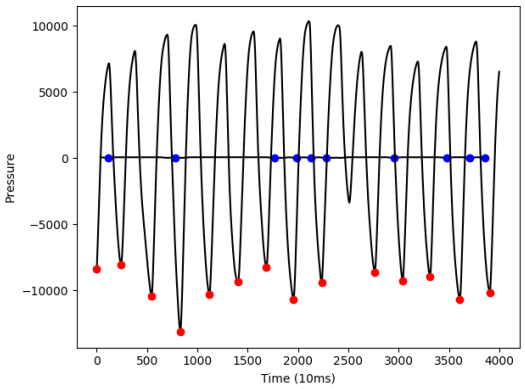

Figure 30: Frequency modulation airflow data no apnoea a02r.dat

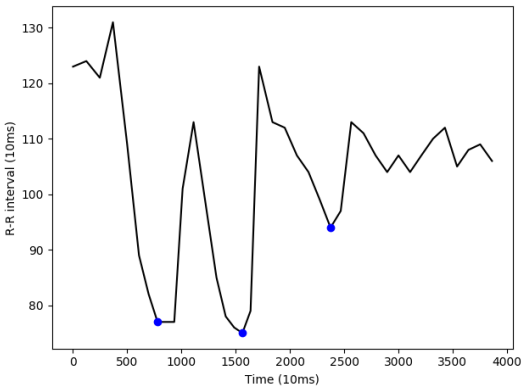

Figure 31: Frequency modulation ECG data apnoea a01.dat

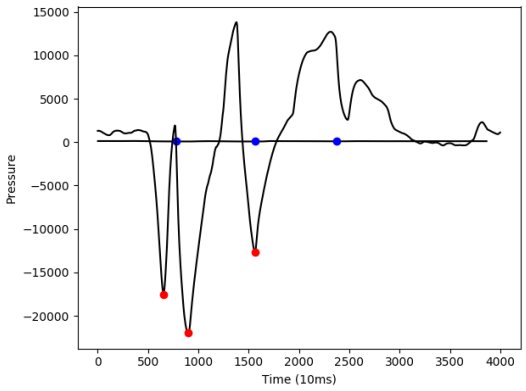

Figure 32: Frequency modulation airflow data apnoea a01r.dat

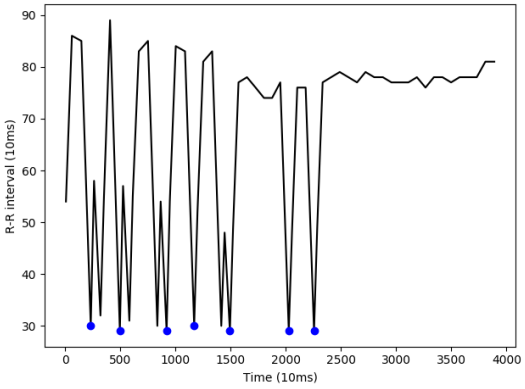

Figure 33: Frequency modulation ECG data apnoea a02.dat

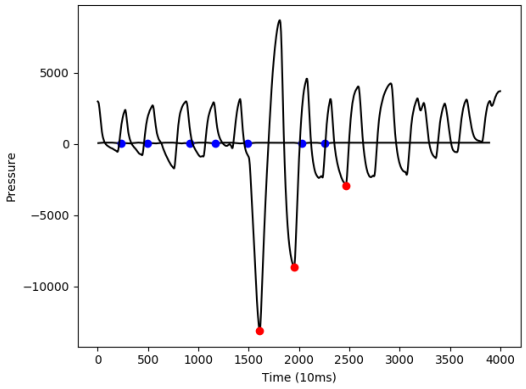

Figure 34: Frequency modulation airflow data apnoea a02r.dat

In the next section, we will test both the amplitude and frequency modulation algorithms on larger samples of data.

7 Testing the Algorithms

Our algorithms for computing respiratory rate were created by studying small segments of the ECG data to avoid overfitting. We will now test the algorithms on larger samples of data. Perhaps the simplest measure of validity is the root mean square (RMS) of the difference between the respiratory rate (in breaths per minute) computed using our algorithms and the benchmark respiratory rate computed from the respiratory data. The respiratory rate is computed every 120 seconds using the function extract_ecg_data in our code. By default, the function uses a 120 second time interval but allows for using time intervals of a different length.

To test our algorithm, we compute the RMS difference for varying tolerance values. Here, the tolerance defines the threshold for locating breaths with our algorithms. The last step of verifying that we have found a breath using the amplitude/frequency modulation algorithm is checking that the corresponding peak/trough height is greater than the product of the tolerance and standard deviation of the amplitude/frequency data in the given data segment. By sampling small segments of data in a01.dat, we initially decided to use a tolerance of 2 standard deviations for amplitude modulation and 1 standard deviation for frequency modulation.

a01.dat					
Amplitude Modulation Tolerance (std dev)	2	2.25	2.5	2.75	3
RMS of Amplitude-Airflow Difference	2.62	2.17	2.38	2.96	3.50
Total number of breaths	2731	2126	1515	1020	657
Frequency Modulation Tolerance (std dev)	0.5	0.75	1	1.25	1.5
RMS of Frequency-Airflow Difference	4.36	2.67	2.49	2.72	2.98
Total number of breaths	3592	2344	1597	1178	946
Airflow RMS	4.39	4.39	4.39	4.39	4.39
Benchmark total number of breaths	2059	2059	2059	2059	2059

Table 1: RMS difference between amplitude/frequency modulation algorithms and benchmark algorithm for various tolerance values in a01.dat

a02.dat					
Amplitude Modulation Tolerance (std dev)	1	1.25	1.5	1.75	2
RMS of Amplitude-Airflow Difference	5.23	4.72	4.33	4.17	4.59
Total number of breaths	6404	5842	5158	4361	3634
Frequency Modulation Tolerance (std dev)	0.125	0.25	0.5	0.75	1
RMS of Frequency-Airflow Difference	5.49	4.68	4.84	4.81	5.15
Total number of breaths	6600	5458	4853	4405	3944
Airflow RMS	9.87	9.87	9.87	9.87	9.87
Benchmark total number of breaths	4874	4874	4874	4874	4874

Table 2: RMS difference between amplitude/frequency modulation algorithms and benchmark algorithm for various tolerance values in a02.dat

Table 1 shows the result of our testing on a01.dat. From the table, we see that a tolerance of between 2 and 2.5 for AM and 0.75 and 1.25 works reasonably well. Outside of these values, however, the algorithms begin to breakdown. Table 2 shows the result of our tests on a02.dat. Here, we see a wider range of tolerance values for which the algorithms still work reasonably well. The acceptable tolerance values for the AM algorithm differ between the a01.dat data and the a02.dat data. This makes it hard to set a tolerance value that will work reasonably well across multiple data sets. There is less of a disparity for the FM algorithm and so this may turn out to be more robust than the AM algorithm.

The primary reason for the disparity between the a01.dat data and the a02.dat data appears to be the larger variance (both amplitude and frequency) of the a02.dat data compared to the a01.dat data. This larger variance means that we should use a lower tolerance to account for this. It also appears that the larger variance is more significant in the amplitude data as opposed to the frequency data. This results in the FM algorithm potentially being more robust.

We could test our algorithms on the other data files to find ideal tolerance values for those files. However, it is not clear how we could design a more robust algorithm that works well for multiple data sets without knowing the ideal tolerance values in advance.

8 Further Work

Further work should be done on our algorithms to increase their robustness. We would like our algorithms to be effective at analysing more complex ECG data sets. Examples of these data sets include the more challenging learning records obtained from [1] and [2]. We previously discussed filtering for removing noise in the data. A greater issue is the presence of baseline wander in ECG data. Baseline wander can be removed using filtering techniques. We did not encounter baseline wander with the data that we tested. However, this problem would need to be addressed when using the algorithms on more complex data sets. To test baseline wander removal algorithms, a good resource is the MIT-BIH noise stress test database [7], [8].

9 Conclusion

To conclude, we have implemented two algorithms to compute the respiratory rate from the ECG signal. These were the amplitude and frequency modulation algorithms. We have found them to be moderately successful when applied to a combination of apnoea and no apnoea data. However, there are obvious limitations to these algorithms. The primary limitations are under-counting breaths during a period of high respiratory rate and over-counting breaths in the presence of apnoea. There is a tradeoff between accuracy in the no apnoea data and accuracy in the apnoea data. We could weaken our condition for locating breaths to resolve the issue of under-counting in the no apnoea data but this will cause even more over-counting in the presence of apnoea. Similarly, if we strength our condition for locating breaths we will see increased performance in the apnoea data and decreased performance in the no apnoea data. Limiting our algorithms to only the no apnoea data and lowering the tolerance values, we find them to be much more accurate.

Our algorithms have been implemented with the intention of striking a balance between accuracy in the apnoea data and accuracy in the no apnoea data. One reason for this is the potential application of ECG analysis to detecting respiratory issues. If our algorithm is only applicable to regular respiratory signals or only applicable to irregular respiratory signals, then we will not be able to distinguish between good and poor respiration. We would like to be able to distinguish between good and poor respiration using our algorithms, as this has potential applications to detecting pneumonia in low-resource clinics.

References

[1] Penzel T, Moody GB, Mark RG, Goldberger AL, Peter JH. The Apnea-ECG Database. Computers in Cardiology 2000;27:255-258.

[2] 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. Circulation 101(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13).

[3] Halhuber M, Günther R, Ciresa M. *ECG An Introductory Course: A Practical Introduction to Clinical Electrocardiography*. Springer-Verlag Berlin Heidelberg, 1979.

[4] Charlton PH, Bonnici T, Tarassenko L, Clifton DA, Beale R, Watkinson PJ. An assessment of algorithms to estimate respiratory rate from the electrocardiogram and photoplethysmogram. Physiol Meas. 2016;37(4):610-626. doi:10.1088/0967-3334/37/4/610.

[5] Penzel T, Kantelhardt JW, Bartsch RP, Riedl M, Kraemer JF, Wessel N, Garcia C, Glos M, Fietze I, Schbel C. Modulations of Heart Rate, ECG, and Cardio-Respiratory Coupling Observed in Polysomnography. Front Physiol. 2016 Oct 25;7:460. doi: 10.3389/fphys.2016.00460. PubMed PMID: 27826247; PubMed Central PMCID: PMC5078504.

[6] Ding S, Zhu X, Chen W, Wei D. Derivation of Respiratory Signal from Single-Channel ECGs Based on Source Statistics. International Journal of Bioelectromagnetism. 2004;6(1):41-48.

[7] Moody GB, Muldrow WE, Mark RG. A noise stress test for arrhythmia detectors. Computers in Cardiology 1984; 11:381-384.

[8] 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. Circulation 101(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/content/101/23/e215.full]; 2000 (June 13).

[9] Billman GE. Heart rate variability - a historical perspective. Front Physiol. 2011 Nov 29;2:86. doi: 10.3389/fphys.2011.00086. PubMed PMID: 22144961; PubMed Central PMCID: PMC3225923.