# 6.555 Lab 1, Imran Ahmed

1. A block diagram for this pipeline is as below:
2. For a 16-bit quantiser with a Vmax of 5V:

Thus, it is clear that the rounding of ECG data values to the nearest millivolt is due to some additional resolution loss after quantization.

1. Analogue filters tend to be imperfect (as opposed to ideal low pass filters) and have a large transition band. Therefore, it is likely the case that analogue filters won’t fully attenuate frequencies close above the cut-off frequency. Thus to avoid the bulk of the aliasing effects, it is necessary to use a cut-off frequency for the analogue filter that is lower than 125Hz to allow for as-close-to-perfect sampling as possible.
2. Our bandpass filter was a simple filter with order 500 and low frequency cut-off of 5Hz and a high frequency cut-off of 30Hz. This particular order was chosen as a trade-off between the filter’s transition band or “sharpness” and the time delay (computational cost is another disadvantage but this wasn’t an issue for us due to small sample size). In addition, we decided to stick with the Hamming window as it provided the right balance between amplitude resolution and frequency resolution - we made the assumption that amplitude and frequency would both be equally useful in characterising the power spectrum of the ECG signal.

Our cut-off frequencies were chosen because they removed the baseline fluctuations in the ultra-low frequency regions, and also removed the high frequency noise that starts to overwhelm the signal past the 30 Hz region. In addition, we chose our bandpass such that we would preserve as much of the signal energy as possible and thus above 30 Hz, we could only attribute the output to noise. To implement this filter, we used the Matlab ***fir1*** function. The output filter had the frequency spectrum we were expecting and there is a suitable amount of frequency attenuation outside the passband. The transition band was still quite wide however (suggesting a higher filter order may have been more appropriate).

Both the frequency and impulse response for our filter can be found below. Note that the frequencies on the x-axis are normalised and that freal / fsample = fnormal (where fsample is 250Hz).

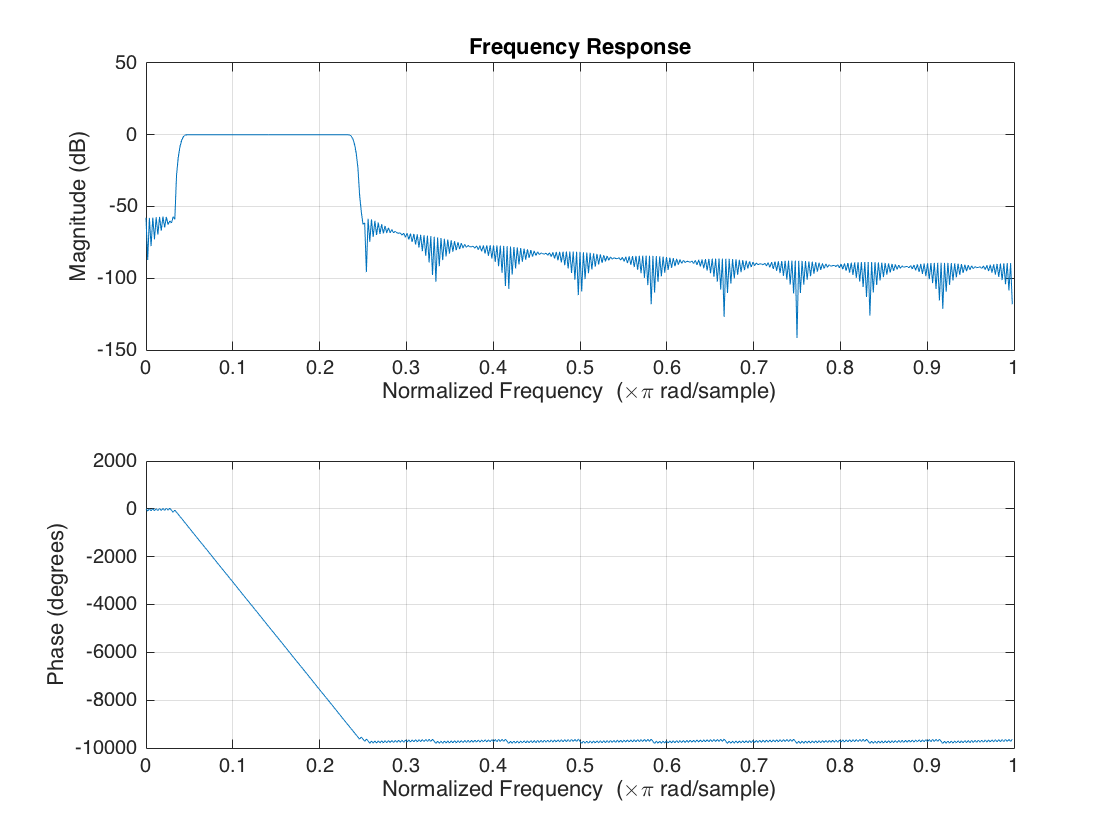
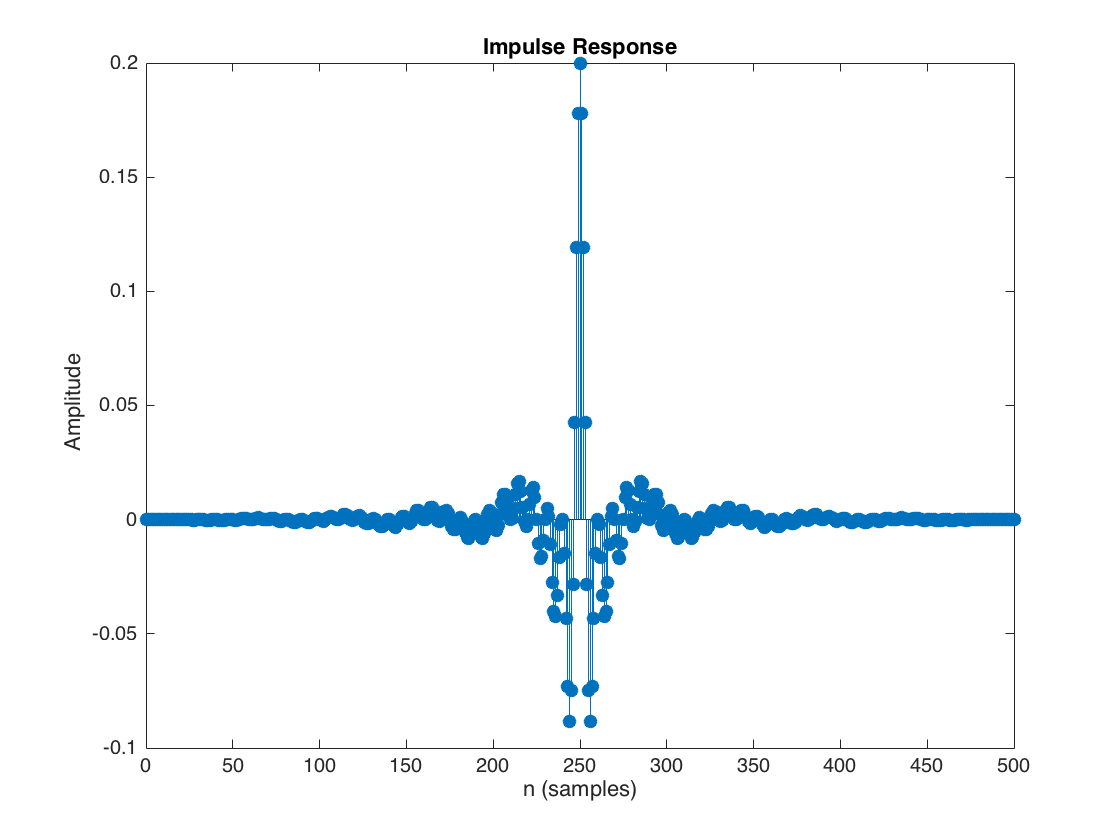


Figure 2: Impulse response of bandpass filter

Figure 1: Frequency response of bandpass filter

1. The effect of the filter can be seen below:

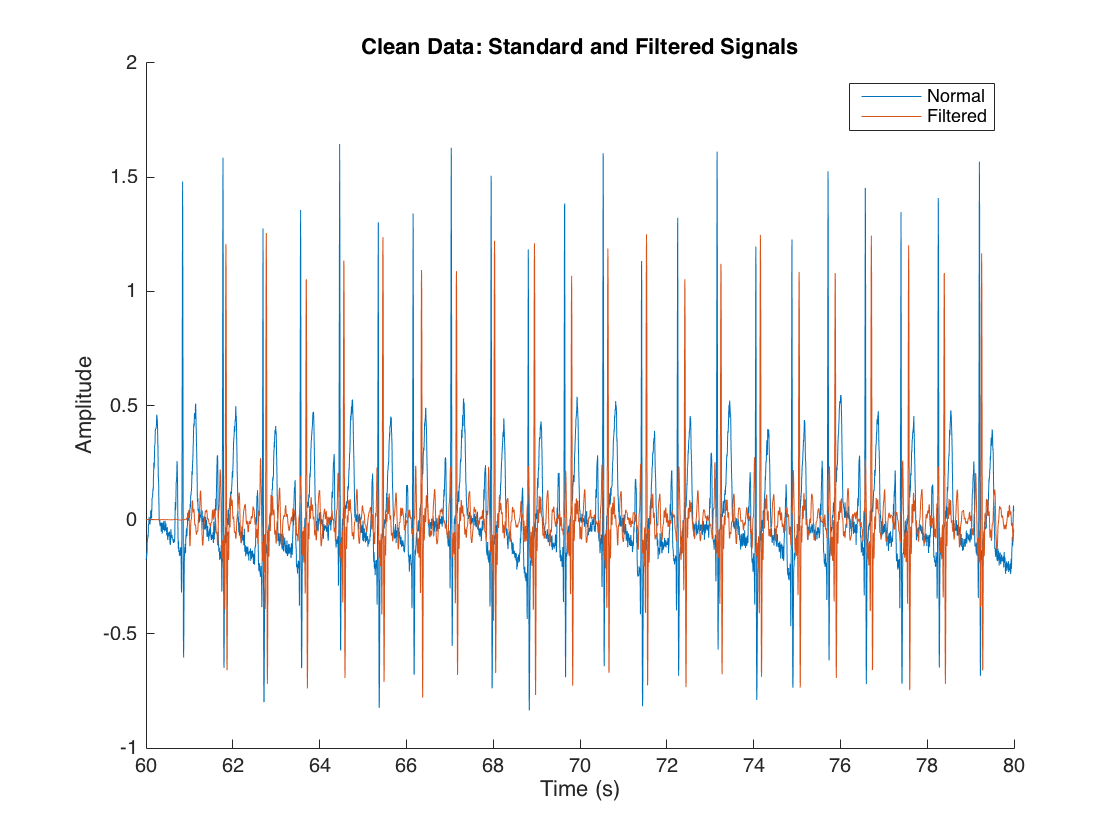


Figure 3: Comparison between standard and filtered 'clean' signal in time domain

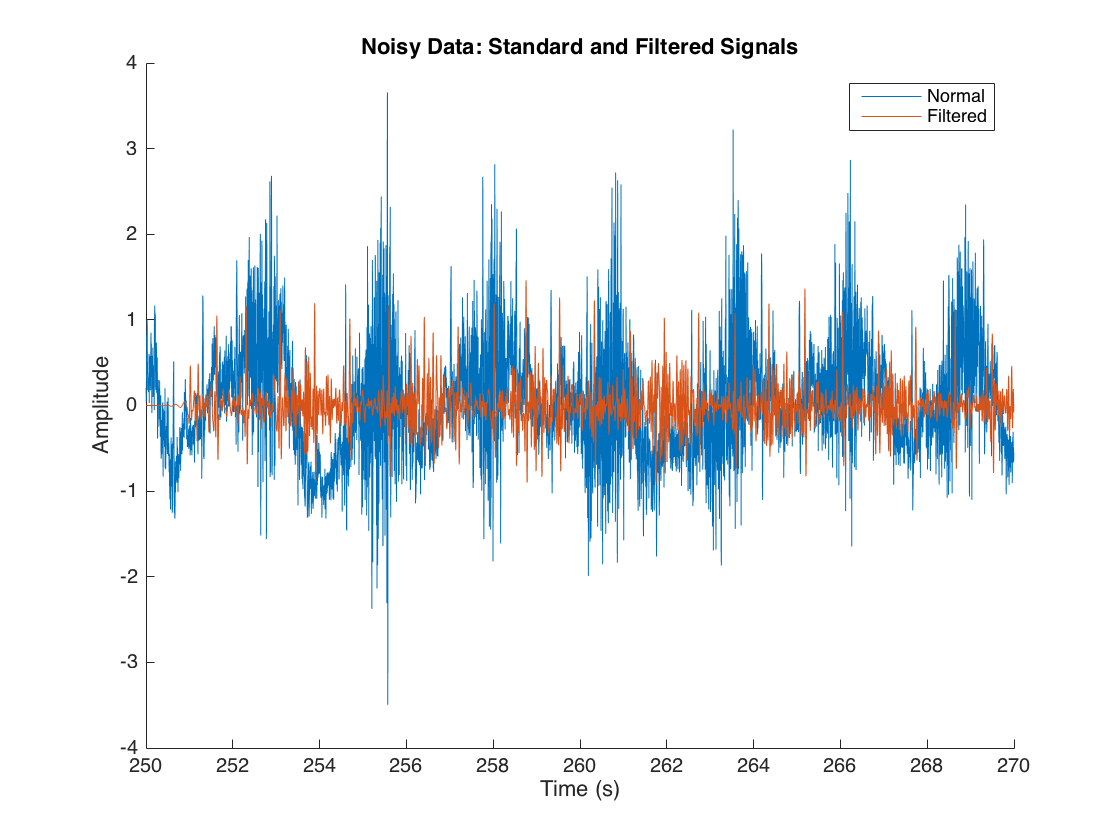


Figure 4: Comparison between standard and filtered 'noisy' signal in time domain

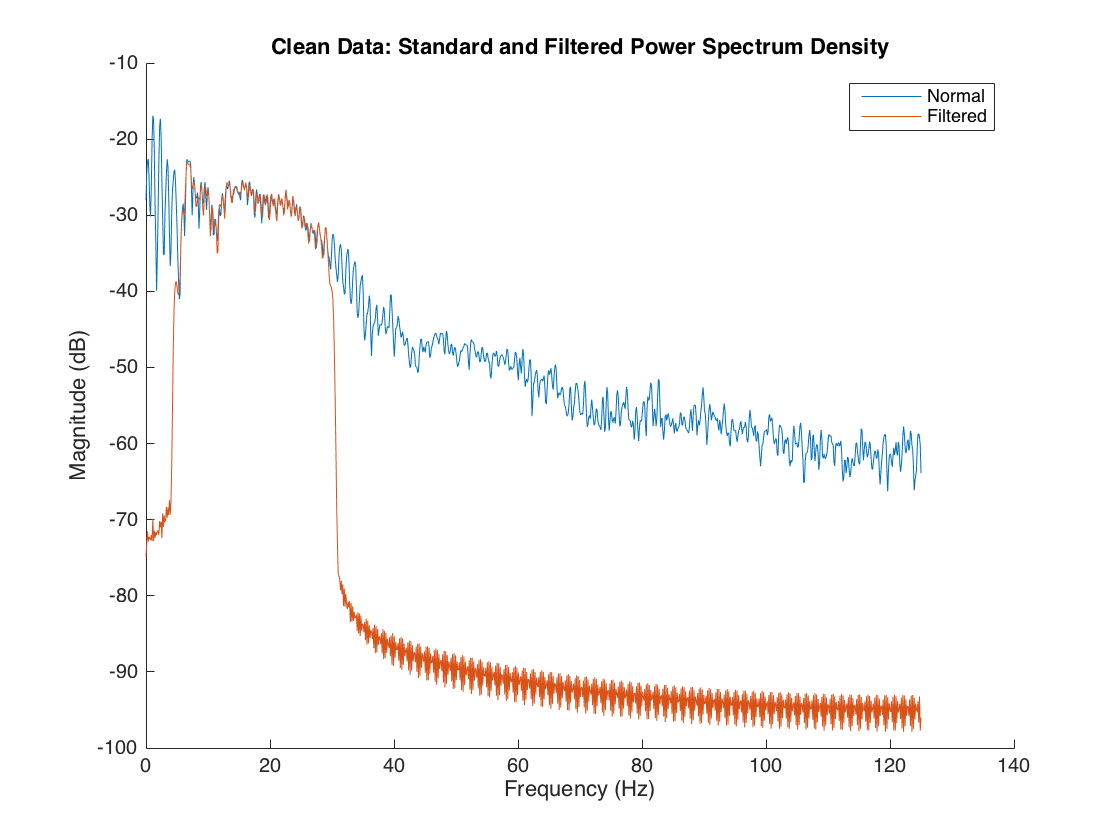


Figure 5: Comparison between 'pwelch' PSD of standard and filtered 'clean' signal

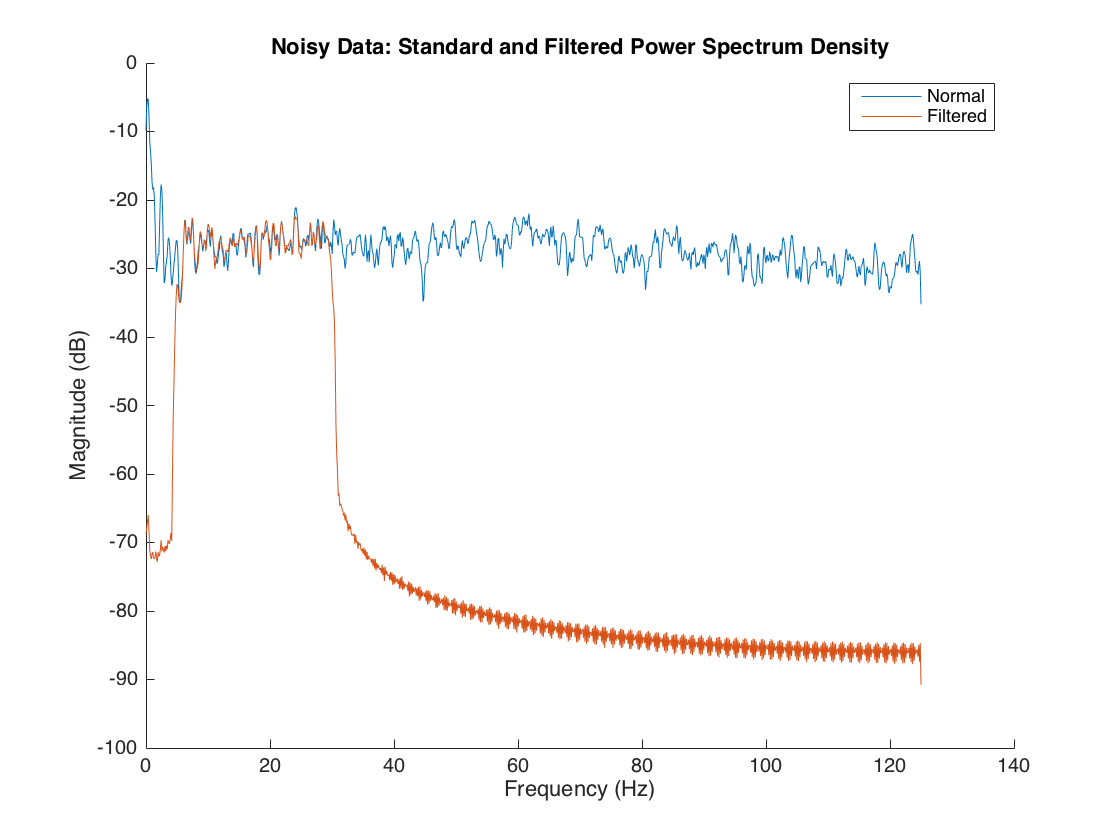


Figure 6: Comparison between 'pwelch' PSD of standard and filtered 'noisy' signal

As can be seen by the comparison in both figures 3 and 4, adding the filter (especially a filter with such a high order) introduces a delay to the filtered signal. In addition, the QRS complexes are slightly attenuated and the low-frequency signal associated with the PR interval is significantly attenuated. However, at the same time, a lot of the very high frequency noise that can be seen occurring during the non-QRS-complex stages is also removed (which is as expected given the nature of our filter). This is far more evident in the “noisy” time-comparison (the low frequency variation is almost entirely removed). Unfortunately, there is still a lot of noise prevalent which falls within the band of frequencies that we are interested in – which cannot be resolved by a bandpass filter.

Perhaps the best illustration of the effect of the bandpass filter can be seen by looking at figures 5 and 6 which show the power spectrum densities for both the standard & filtered clean and noisy signals respectively. It is apparent from these plots that the bandpass filter is serving its purpose: there is a clear low frequency cut-off that effectively eliminates low-frequency noise (by attenuating it to -70dB) and there is also a high-frequency cut-off. The effects of the filter very closely mimic the frequency response of the filter (Figure 1) which is promising. Interestingly the transition band width does not seem to have a substantially detrimental effect on the PSD’s – the transition is relatively sharp which is favourable.

1. Irrespective of how ideal a bandpass filter can be, the fundamental flaw of bandpass filtering is that it can only remove noise from signals if they either fall into a bin of low frequency or into a bin of high frequency. If there is noise in the signal which affects the signal in the same range of frequencies at which the information in the signal is conveyed, then bandpass filtering will not be able to remove this noise. In this case, the bandpass filter would simply remove “easy-to-remove” noise (which falls outside of a range of frequencies) and would leave the remaining noise unattenuated – which may not be useful
2. We chose our parameters as follows: we decided to use a ‘Hamming’ window as opposed to using a custom window. This is because the ‘Hamming’ window has an acceptable side-lobe amplitude decay rate compared to a simple rectangular window. Thus it is possible to maintain a reasonable frequency resolution (for a given window length) as well as a good amplitude resolution (which we need as we intend to discriminate arrhythmia from normal rhythm by looking at the amplitude output of the PSD). More importantly, ‘Hamming’ is integrated by default into ‘pwelch’ – thus allowing us to find a reasonable representation of the PSD with minimal changes. If we were to choose another window, we would not be able to use pwelch as this would window twice which makes little sense.

While our ideal window length would be chosen to maintain a frequency resolution of 0.1Hz (corresponding to **4.Fsample/0.1 = 10000**), we are limited by the number of samples of that we have available in each time segment. Because we are taking segments every 10 seconds, we have **10. Fsample = 2500** samples per segment. As a result, our maximum possible DFT length is the largest power of two below this (i.e. **2048**)which gives us a frequency resolution due to our Hamming window of **0.5Hz**.

1. Ventricular arrhythmia segments differ from the normal segments in that they have substantially different profiles in both the frequency and time domain – as illustrated by the plots below.

In the time domain, the peaks are often far closer together which is indicative of a very fast heartbeat. Figure 7 below seems to include some low frequency drift during normal rhythm but that can be safely ignored (as it is unrelated to the heart). After looking at several power density spectrums, we determined that this was within the range of 3.5 Hz to 6.5 Hz; this is abnormal as normal resting heart rates are typically 2 Hz at their highest.

This observation is echoed in the PSD comparison in Figure 8 below – there is a clear peak at 5 Hz. Note that this PSD also reveals several other outliers at higher frequencies but we were unable to find a consistent pattern with these higher frequency peaks when looking at data across multiple patients (whereas, peaks in and around the 5 Hz region were consistent).

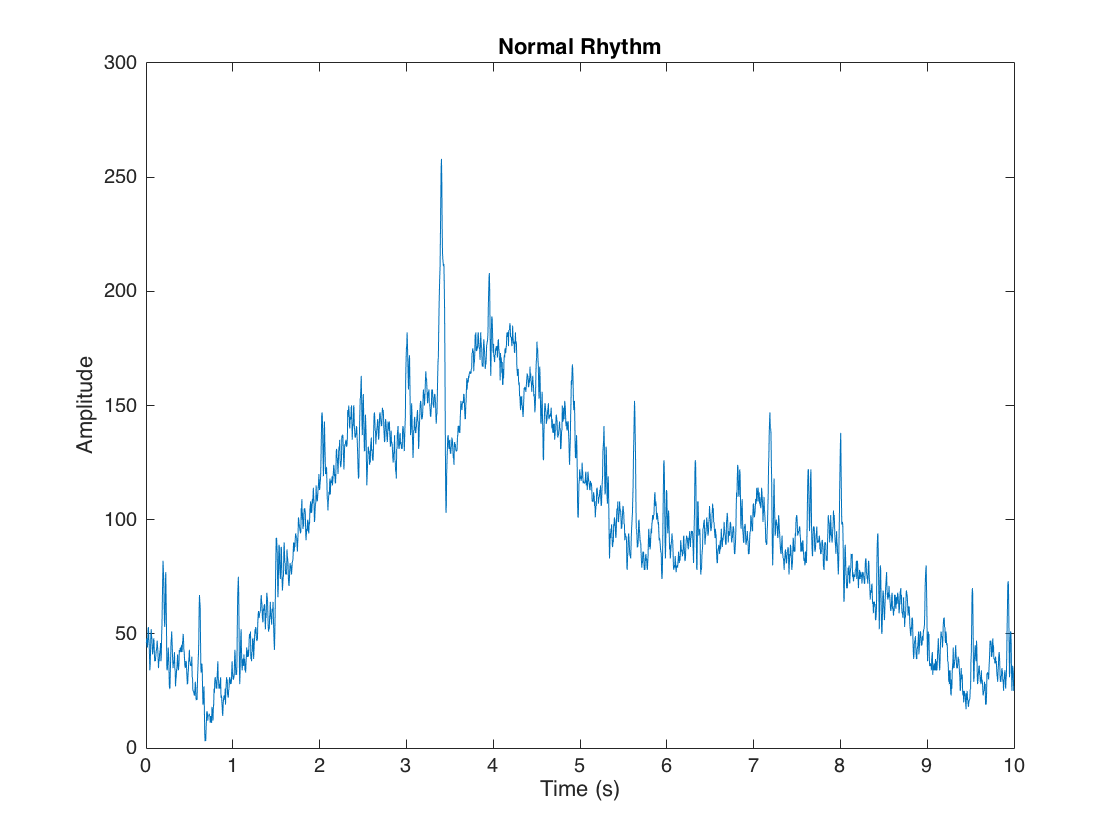
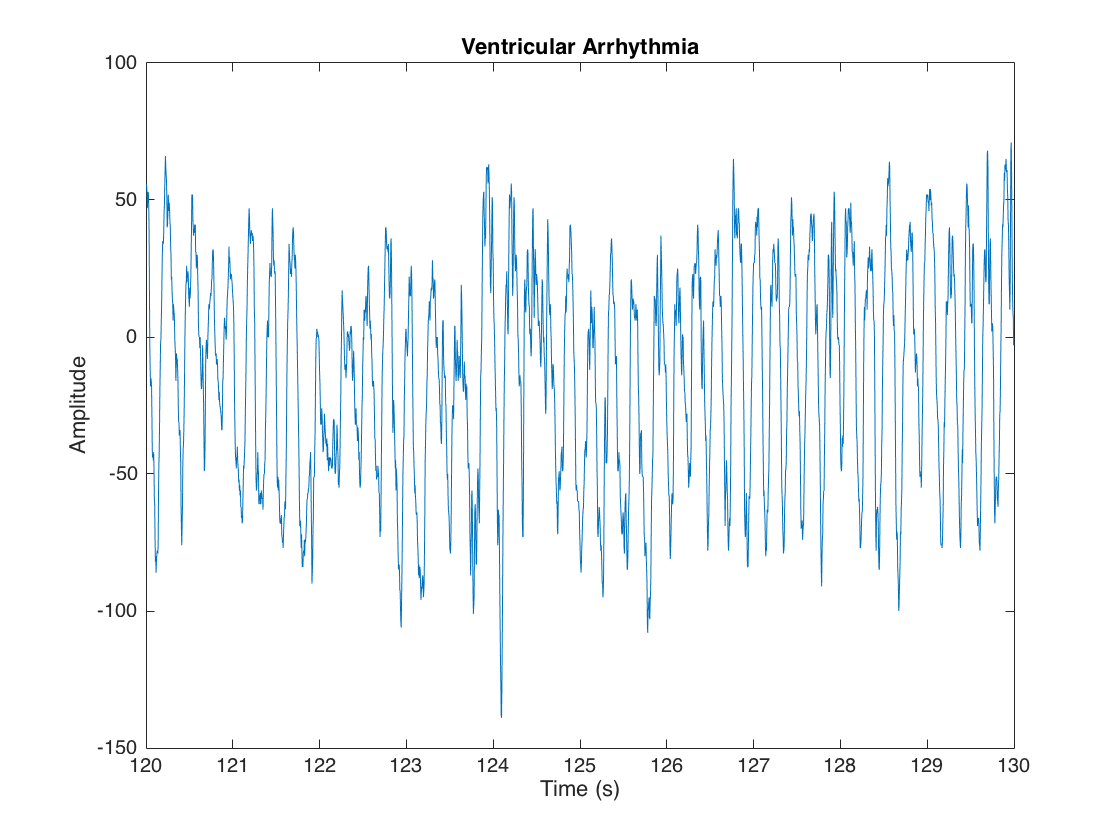


Figure 7: Comparison in time between normal rhythm and typical ventricular arrhythmia

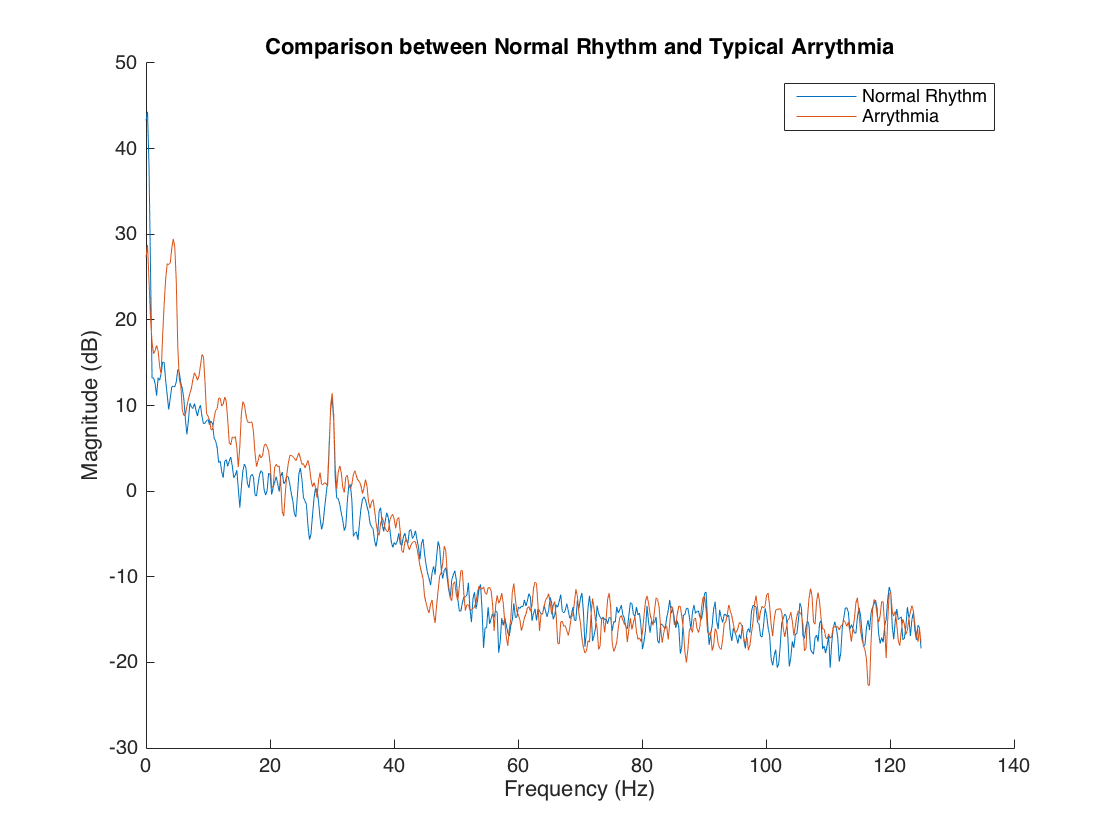


Figure 8: Comparison between normal rhythm and arrhythmia

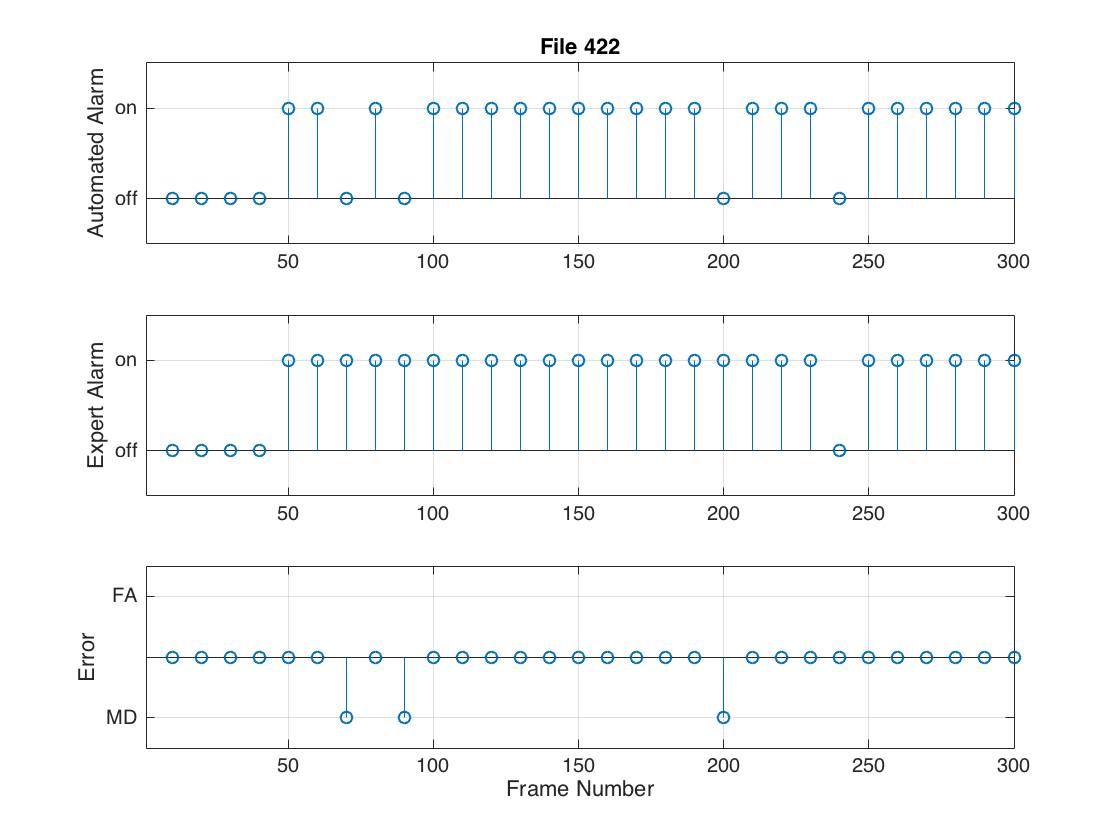
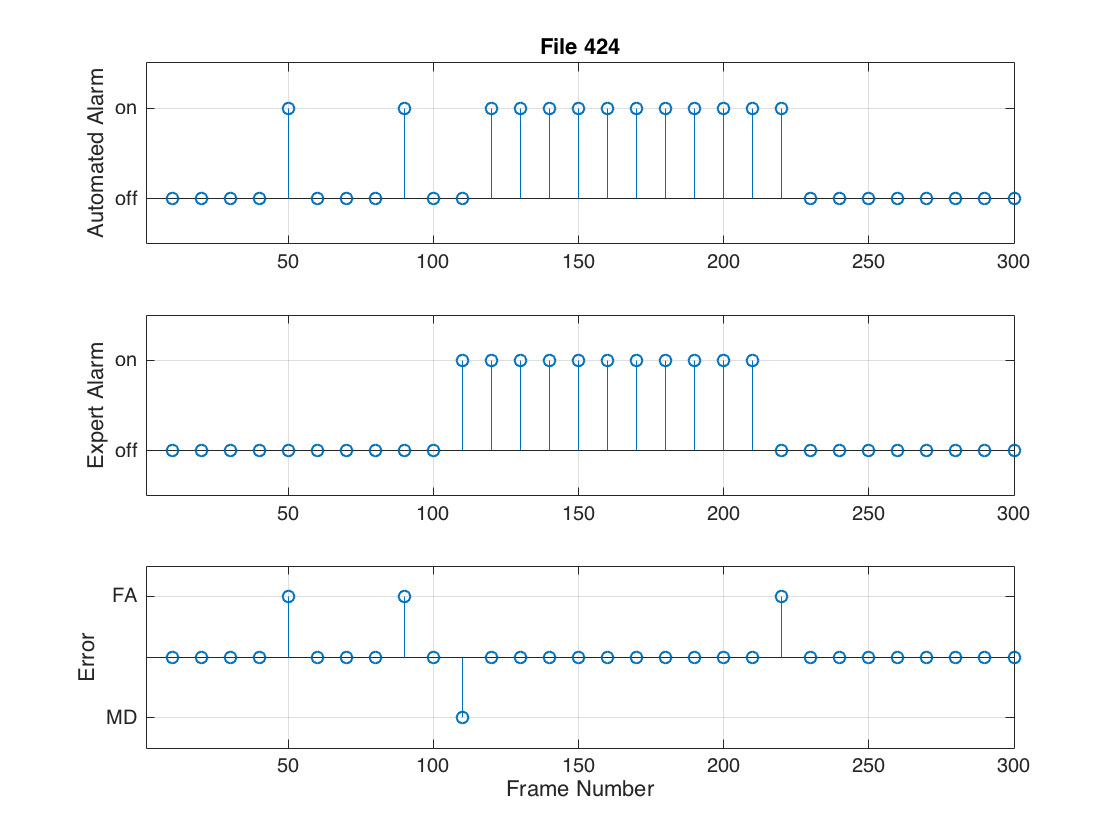
1. The main approach of our detector was based on our observation from the previous question that there is a noticeable peak in the 5 Hz region of the PSD during an arrhythmia. We surmised that if this peak’s amplitude varies from a patient’s typical spectral amplitude in that region (calculable when the patient first uses the ECG machine), it would be possible to “flag” an arrhythmia. As a result of this approach, the algorithm is built on the assumption that the first 20 seconds of a patient’s ECG readings are deemed as “normal” – which is not necessarily valid but this was a trade-off we chose to make.

Our algorithm in *va\_detect* is as follows:

1. Find the PSD of the current time window using ‘pwelch’ – defined with **no overlap,** with a frequency bins of 0.005 Hz (due to the efficiency of the algorithm, this is a tractable calculation). The window length is 2048 and DFT length is 2049 (both of which are smaller than the sample size at 2500).
2. Extract a small window of the PSD between 3.5 and 6.5 Hz (i.e. a 1.5 Hz buffer around the 5 Hz mark).
3. Use the MATLAB function *envelope* to find an envelope for the plot (with a granularity of 3 – which means envelopes are taken at every **0.3 Hz**.
4. Compute the maximum envelope magnitude: menv = max(envmax(i)-envmin(i) ∀ i)
5. Calculate the mean menv for all the windows within the first 30 “safe” seconds of ECG data – use this as a baseline for subsequent comparison.
6. Flag an alarm for all subsequent windows if the menv for that particular window exceeds the menv-avg by a certain threshold factor.

It is worth noting that while we experimented with applying a filter, we realised that this wasn’t the issue – the issue we faced was noise *within* the frequency band we were interested in and thus a bandpass filter would not have been useful. We also tried a number of other heuristics, but the heuristic proposed above produced the strongest results.

1. Our detector performed relatively well with respect to the expert annotations although there is certainly room for improvement. Comparisons between our detector and the expert output can be found below:



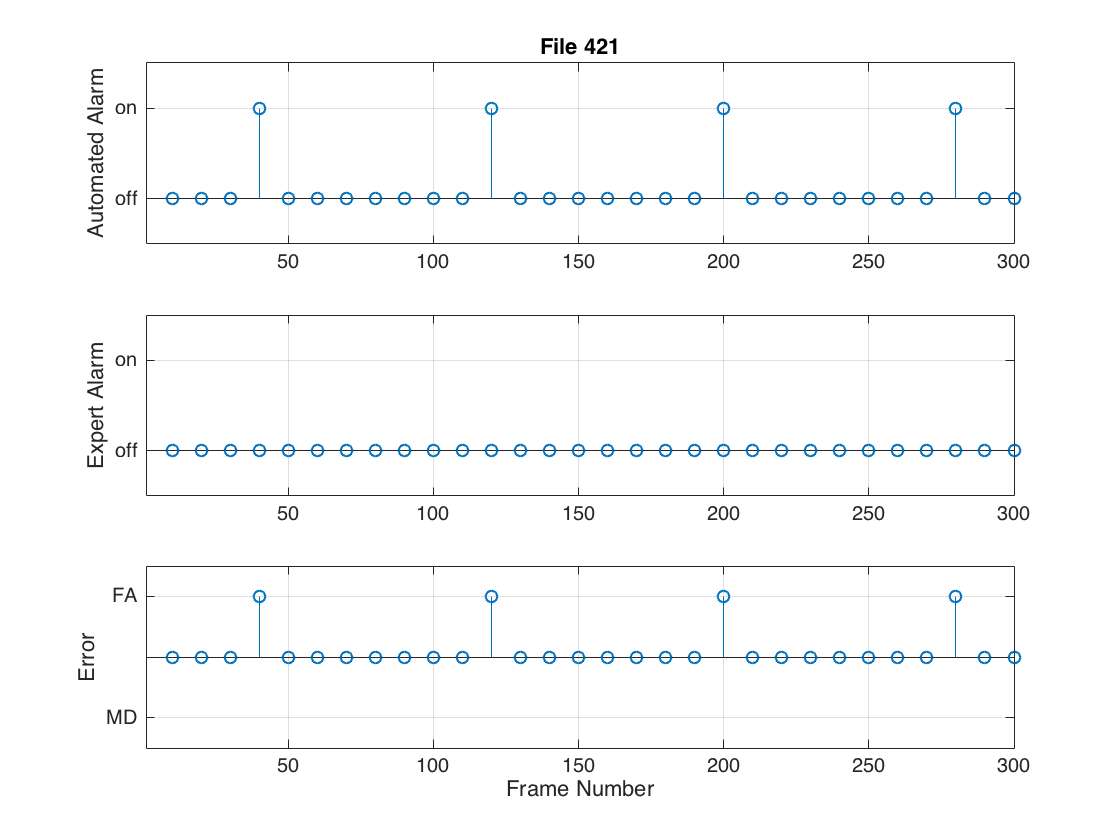
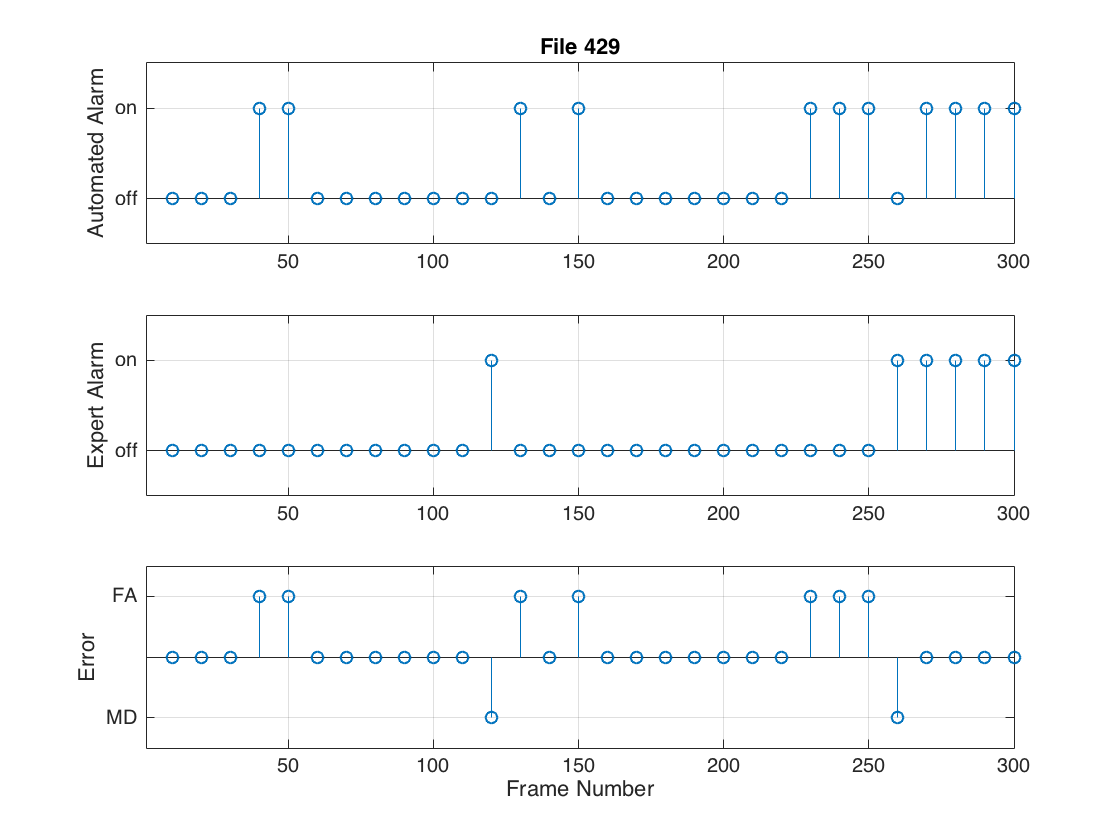


Figure 9: Comparison of our detector's performance and the annotations of an expert

As can be clearly seen from the results above, our detector does perform quite well and can correctly detect lengthy segments of arrhythmia. However, at the same time, our detector produces a substantial number of false positives – suggesting that we are undercompensating with our max envelope threshold. However, when we tried to increase this threshold, we didn’t observe any noticeable change (and in fact, we produced more false negatives). This means that we would have to find another way of determining whether a positive detection was a false positive. Interestingly, false positives often occur in the “lead-up” to an arrhythmia which suggests that the transition between normal rhythm and arrhythmia is gradual and our detector is inadvertently detecting this in advance.

False positives also seem to occur in the test cases where there is some form of noise which indicates that our particular method is sensitive to noise – particularly noise at low frequency. This makes sense as we only look at a small window between 3.5 and 6.5Hz (which is in the low frequency region) and thus low frequency noise in this region will likely perturb the max envelope used for detection.

In terms of false negatives, we ensured that the threshold was low enough that we could mitigate them as much as possible as we judged that it would be better to have more false positives than false negatives. However, we weren’t able to remove all of the false negatives: in situations where the user is in fibrillation for only a short time period (i.e. for just one segment of 10 seconds), our detector isn’t [for the most part] sensitive enough to detect it although it tends to detect something in the segments immediately following a short fibrillation event.

1. If we had more time to improve this detector we would look into analysing signal energy as we anticipate the signal energy would be higher during arrhythmia than during normal rhythm. In addition, it would be interesting to look into the time domain in more detail and see if there is any way of distinguishing arrhythmia in time (by looking at relative magnitudes or frequency of 0-amplitude ‘crossings’ for example). As opposed to using a simple max-envelope heuristic, we think that more complex solutions may involve using SVMs/other machine learning techniques to classify signals (after running spectral analysis on the signal to characterise it) or perhaps using more traditional methods such as Hidden Markov Models or some form of decision tree.
2. Primarily I learnt that while it may be possible to develop a simple heuristic that can work “most” of the time, to reduce false positives and false negatives to an acceptable level of tolerance for real use (i.e. 1 in 1000 for example) requires a step-up in effort and algorithmic complexity. This is perhaps why ECG detection is still a real-world problem that hasn’t been solved.
3. It was very interesting to work with with windows and see how Matlab’s signal processing toolkit can help clean very noisy inputs and allow for relatively accurate detection with a few lines of code. However, we did spend a lot of our lab time in our last session trying to implement a window only to realise that we didn’t really need it to identify fibrillation – that was a bit exasperating.