Etude 09: Pulses Counting

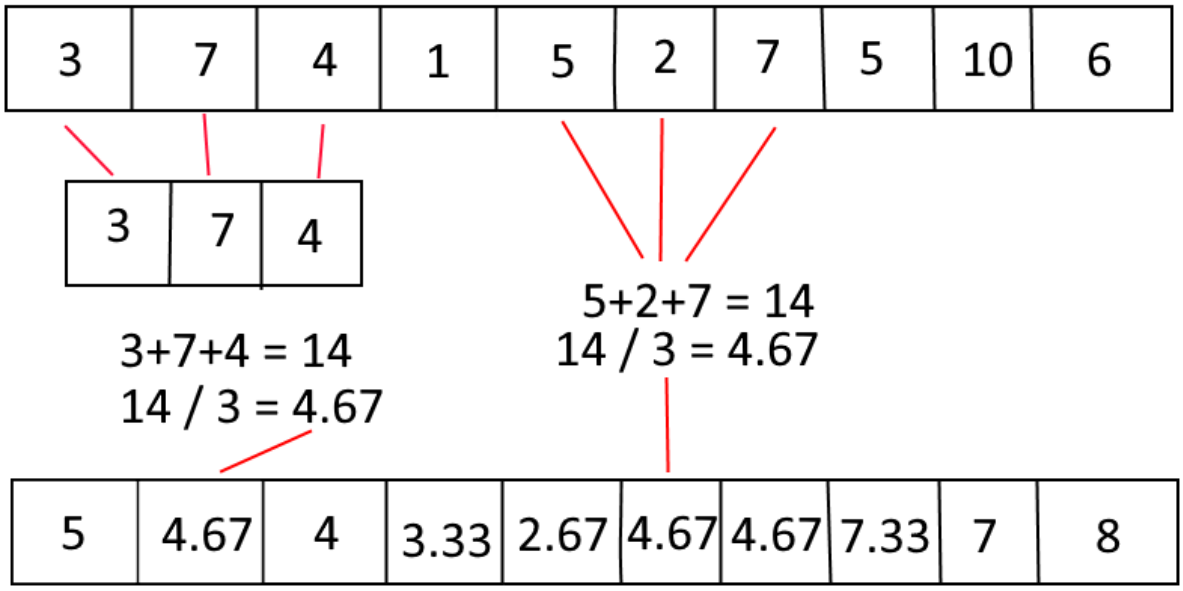
COSC 326

Blake MacDade (8548310) Sam Fern (8555433)

Filters:

For our approach, used a combination of two filters a smoothing filter and a Bandpass filter.

- The first filter we used was a sliding window-based averaging filter whose job is provide a general smoothing effect. This works by getting the values of three neighbouring cells and averaging them. For the extremities, that is the first and last index, we simply get the one neighbouring cell and average both. The diagram below shows how this works in more detail:



- The second filter we used was a Butterworth Bandpass filter, this works by performing a Fourier Transform on the input data (that is, switching to the frequency domain) and applying both a High and Low cut to the data. A High cut stops any frequencies above a specified point contributing to the resulting signal, it cuts them out, or blocks them. A Low Cut does a similar job, just instead of cutting frequencies above a point, it cuts frequencies below a point. The effect of combining these two filters means we are cutting frequencies below and above the frequency we want to isolate. This means that if we choose our high and low-cut numbers correctly, we can effectively completely isolate our desired frequency. How we chose these came down to effectively trial and error, seeing what produced a nice-looking graph, as we were not sure exactly the frequency we were trying to isolate. We were seeing what numbers cut out the most noise. There were constraints on what our high and low-cut values could be however, they had to be above zero (0) and below our Nyquist frequency, where the Nyquist frequency is half the sampling rate of 10Hz. This upper constraint is because once we get above the Nyquist frequency, we don't have enough information to accurately describe the wave, as we don't have enough 'resolution' to sample it correctly.
  A bandpass filter also has what is called the 'order' this is effectively the cut-off slope. Ideally we would just cut off any frequency outside of the range that we want to keep, however it

isn't that easy, we have to have some kind of slope in there somewhere, the order is a representation of the 'steepness' of the slope. The higher the order, the 'steeper' the slope.

The below image is intended to try explain how a bandpass filter works:

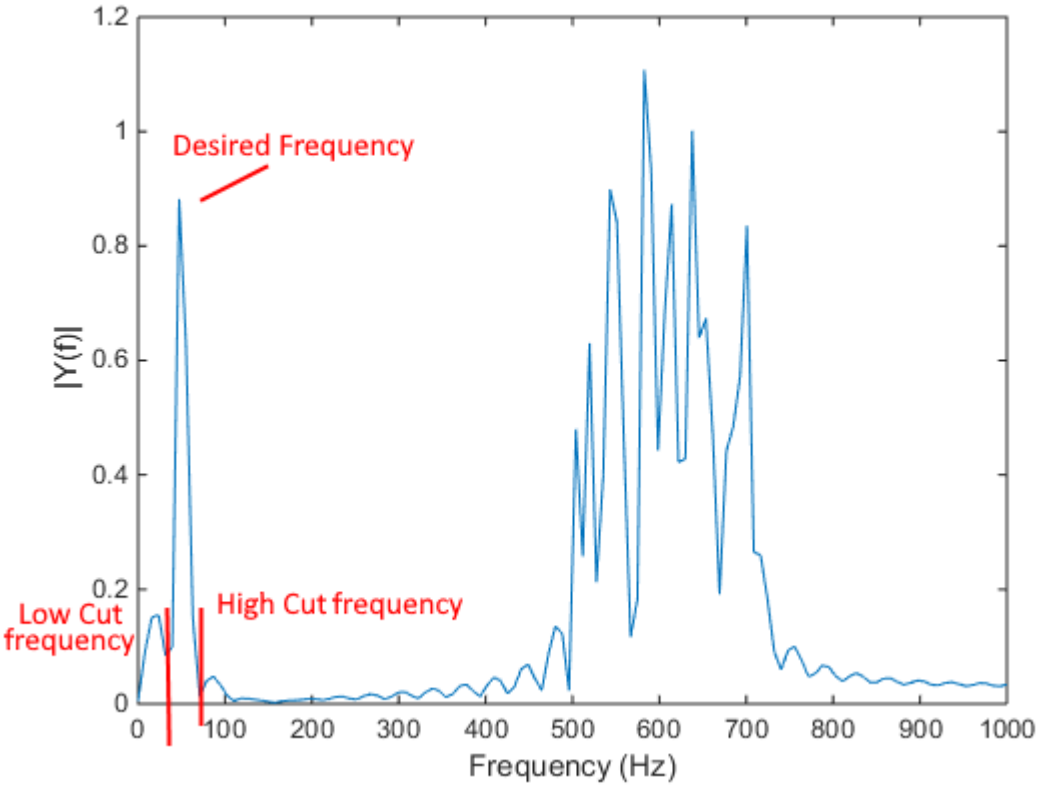


Image is licensed under CC Creative Commons from Stack Overflow.

Our approach was then to apply the smoothing moving window filter three times, then apply the bandpass filter, and then finally apply the smoothing moving window filter another five times.

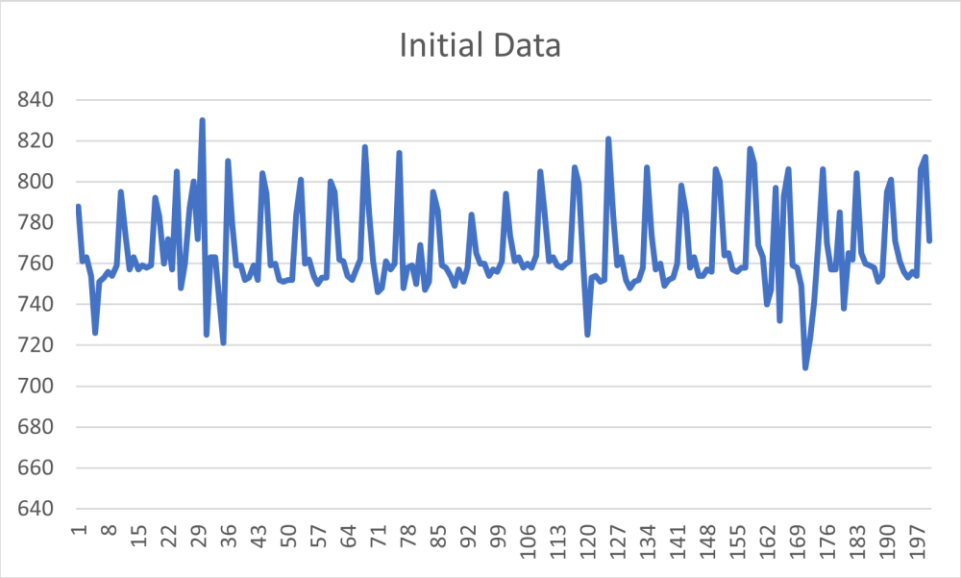
We found that the initial runs of the smoothing moving window filter helped to clean up and smoothen out the input signal just enough to help filter some initial noise. The second runs of the smoothing filter helped to clear out any local maxima that may mess with our peak counting algorithm, it does this by smoothing everything so that the small non-peaks will no longer be counted.

Peak Counting:

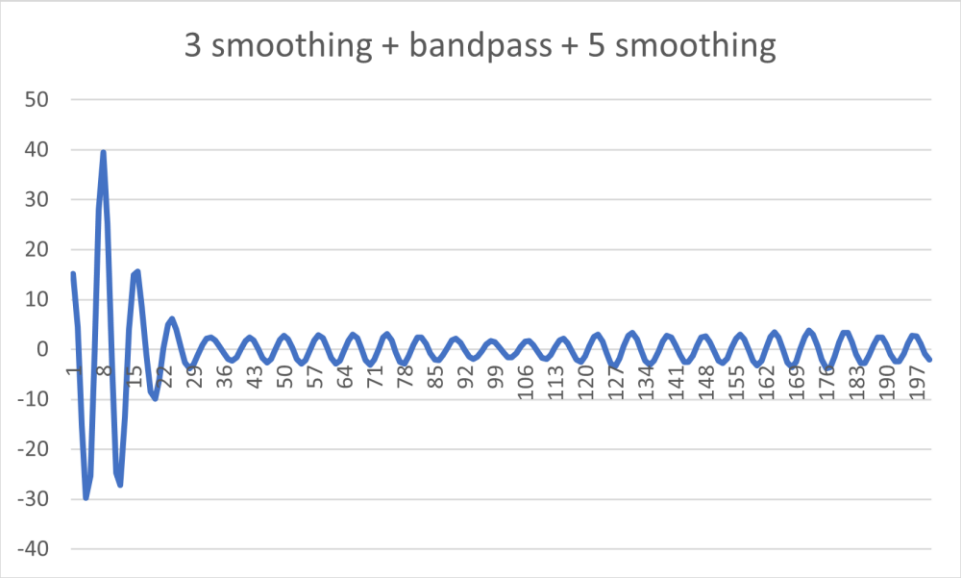
To count peaks, we have used quite a rudimentary algorithm that works quite well. The approach is also a sliding window-based method, we simply go through and see if the index that we are at is greater in value than the ones beside it. That is, if the value we are currently looking at is greater than the value to its left, and greater than the one to its right we can say we have found a peak as the value we are looking at is a local maximum.

Results:

For this showcase, I will be using the dataset “pulsedata1”. Below is what the data looks like initially, before any filtering is applied:



The next image is what the data looks like after an initial three smoothing passes, a bandpass filtering pass, and a final five smoothing passes:



Note that without counting the first index, there are 24 pulses counted for this data.

The data you see there, has the following settings for the band pass filter:

- Low-cut of 1.2Hz (0.24 in Nyquist Normalized Units)
- High-cut of 2.25Hz (0.45 in Nyquist Normalized Units)
- Order of 3.

Nyquist Normalized Units are the frequencies we want represented as a percentage of the Nyquist frequency.

Special Cases:

- We decided not to count the first and last indices as we could not be sure if they are indeed pulses. This is because there could be a higher data point just before the first index, or just after the last index that our recording has missed. When we initially were counting the first and last indices, we encountered a few false positives where the program thought we had a peak but we in fact didn't.