Pulses Counting Report

15/10/21
Pair: Toby Meyrick (1998580), Jake Perkins (2713308)

We started off this etude with a program that simply summed up the data points, averaged them and then anything above the (average + 1 standard deviation) would likely be a pulse. The results were much too variable though. We then decided to use a simple moving average. This involves getting the average of every two data points and finding the pulses in the resulting averages. Now, the resulting pulses counts were much more consistent, although still two of the text files yielded 29 pulses, which was too inaccurate.

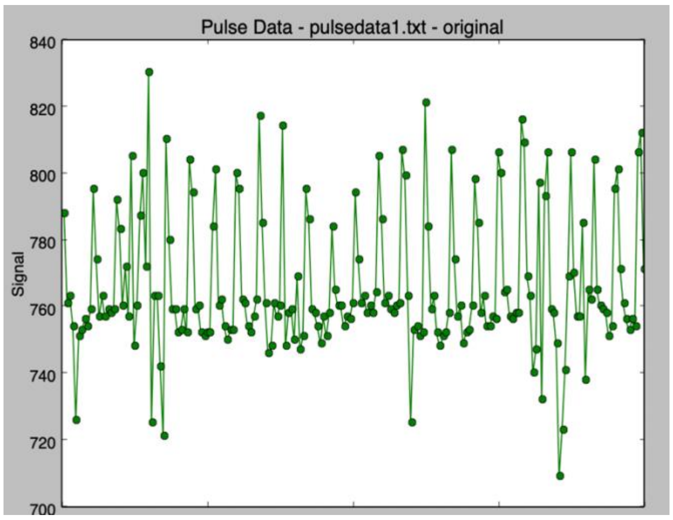
Possible Alternatives
One alternative to make the pulse counting more accurate would be utilising a filtering mechanism such as band-pass filtering. It is a common filtering technique often used in data transmission to minimise bandwidth signal therefore decreasing noise. Band-pass filters are each made of two different types of filters which when used together, form a window of accepted data. A low-pass filter allows data through that is under a margin while a high-pass filter allows data through that is over a margin. Our utilisation of the band-pass filter would count data points between two and any other data points would be considered noise and would be discarded allowing us to precisely report back the number of pulses. Because we were using Java, we decided a bandpass filter would be too difficult to implement (compared to Python, which has special libraries that are well supported), but as you can see below, we were still able to implement a very good count anyway. The next two paragraphs outline how we would have implemented such a filter.

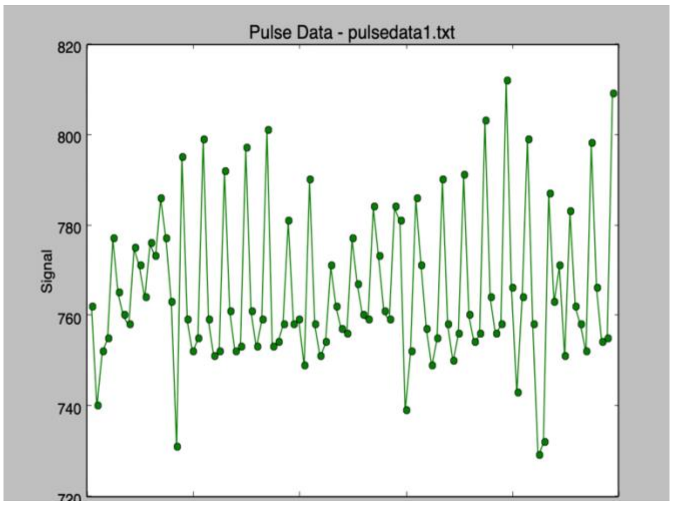
The first step for our band-pass filter programme would be to read in the pulses and define the bandwidth. Ideally, this would be a narrow window of data points which includes the pulses and excludes everything else. The programme would either need to define the minimum and maximum margins for the bandwidth relative to the existing data points or ask the user for them to provide the lower and upper bounds of the window after showing them all the given points on a graph.

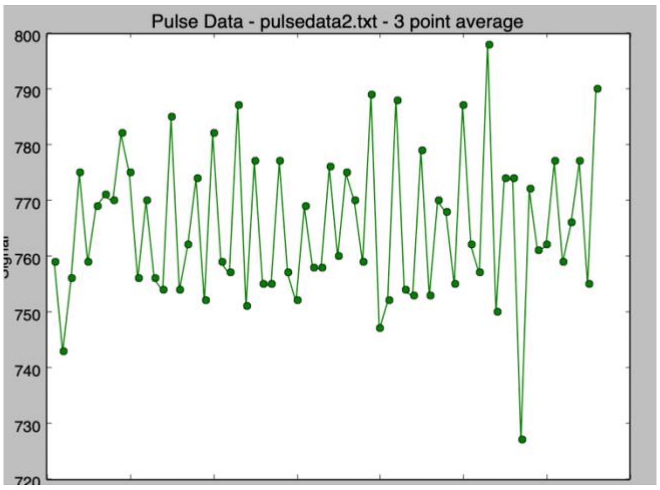
Our programme at first was using a high-pass filter to reduce noise - it is calculating the standard deviation of the data and then only accepting data above it. By assuming that the highest recorded data point is an accepted pulse, the bandwidth for our band-pass filter would be set relative to it. The higher margin would be equal to the highest pulse and the lower would be calculated. So far, we have been using the standard deviation of the data and while this is not the formal high-pass calculation, it defines an acceptable window.

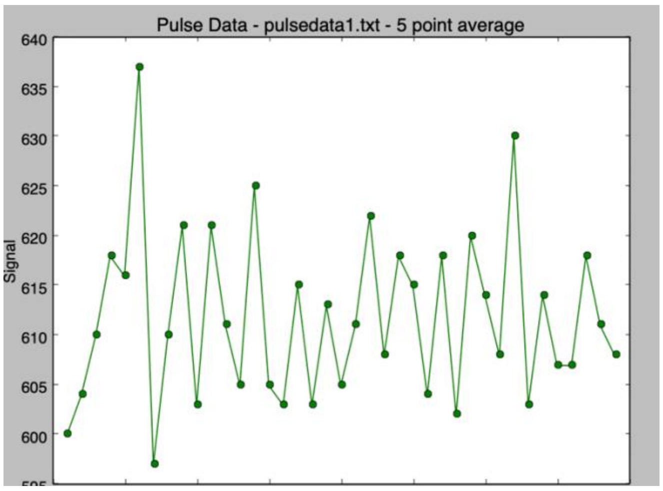
Final Solution
We have come up with a solution, whereby a simple moving average reduces the noise, and each data point above the mean is then compared with its neighbours, and if it has neighbours that are less than it, it must be a pulse. This enables there to be accurate pulse reading even when there is noise recorded on either side of a pulse as we identified that this would have been finding false pulses in the data. Combining this with a simple moving average made our program much more accurate. The results show below. As you can see, the results are very consistent. If we were to make it more accurate, we would need to ensure that the first and last data points are also considered because the first and last data points will not have two neighbours.

Input	Pulses
Pulsedata1.txt	25
Pulsedata2.txt	25
Pulsedata3.txt	23
Pulsedata3.txt	23
Pulsedata4.txt	23
Pulsedata5.txt	24
Pulsedata6.txt	23









The first graph is the original, the second graph is the simple moving average for every two points, which is what we used and the third graph is the simple moving average for every 3 points. We show the simple moving average for 5 points to illustrate that reducing so much of the data by averaging can lead to data loss instead of the noise reduction we were aiming for (this is the last/4th graph).

We found the 3-point average to be less accurate. The reason we think this is, is because some actual pulses may be lost when you average too much. It seems like either 3-point simple moving average or a 2-point as we used could work, but the simple moving average of every 2 points yielded the most accurate results.

Because of this we opted for a method whereby every 2 points are averaged and then used to count the pulses via the neighbour counting we explained above. If the point is above both of it's neighbours as well as being above the mean, we can conclude that it is most likely a pulse. This is especially true once the excess noise has been reduced via the simple moving average.