Pulses Counting

Approach

The fundamental way we are counting the pulses is by finding local maxima or minima and calling them the ‘peak’ (or ‘trough’) of a pulse. A local maximum is a point in the data whose immediate neighbors both have a lower value. Conversely, a local minimum is a point in the data whose immediate neighbors both have a higher value.

Unfortunately, this approach alone is not going to be enough to work accurately on noisy data. We decided to use a moving average smoothing function wherein each point of the data is rewritten as an average of itself and the points around it (to a certain width). We initially implemented a rectangular moving average smoothing function, however this distorted the data too much. We instead switched to a triangular smoothing function, which gave less weight in the average to points farther away from the original point. This appeared to work well enough and didn’t lose too much information in the data.

At this point our implementation worked quite well, but there were a couple of things we did to improve it. Firstly, our smoothing function discarded endpoints who did not have enough points on either side to input into the average function. We altered our smoothing function to make progressively less wide smooths on the endpoints, allowing us to keep the (slightly less smooth) data. Lastly, our extrema detection function detected some of the smaller peaks that we didn’t want to count as pulses. We could have used a threshold function to ensure that only peaks of a certain height were counted, however we found that simply having a minimum distance between peaks worked just as well.

We had a couple of issues with our initial submission. One of them was a simple index out of bounds error that was simple to fix. The other was an issue with outliers in the data. Violent spikes or troughs would reduce the significance of other pulses and would consequently affect the number of pulses we counted. We fixed this by adding a simple minimum and maximum threshold on each point, removing any which crossed this threshold.

Unfortunately, our stop-gap method for removing outliers was too hacky and needed to be more robust. We initially looked at signal correlation, however due to a lack of understanding in how it worked, we deemed it too complex to be reliable and switched to a more statistical approach. Our new method calculates the mean and standard deviation of *N* subsets of the pulse data. From these subsets, we choose the set with the median standard deviation - eliminating subsets with skewed, noisy pulses. With this information we remove any points that are more than 5 standard deviations away from the mean. Thanks to Chebyshev's Inequality, we are guaranteed in the worst case that 96% of all data will fall within 5 standard deviations of the mean, in practice, none of our test data came even close to crossing this threshold while comfortably eliminating any major outliers. We felt that this method - while not perfect - did a fantastic job given the scenario.

Results

Our code is set up to work ‘pretty well’ on all 5 sample sets, however I tuned our code for each dataset (parameters shown below) to count each as precisely as possible.

File	Smooth Width	Min Peak Dist	Extrema	Pulses
pulsedata1.txt	1	6	Maxima	24
pulsedata2.txt	1	5	Minima	25
pulsedata3.txt	1	None	Minima	24
pulsedata4.txt	1	6	Maxima	24
pulsedata5.txt	2	None	Minima	25

External Resources

https://terpconnect.umd.edu/~toh/spectrum/Smoothing.html
https://stackoverflow.com/a/31070779