Lab 2: Step Counter

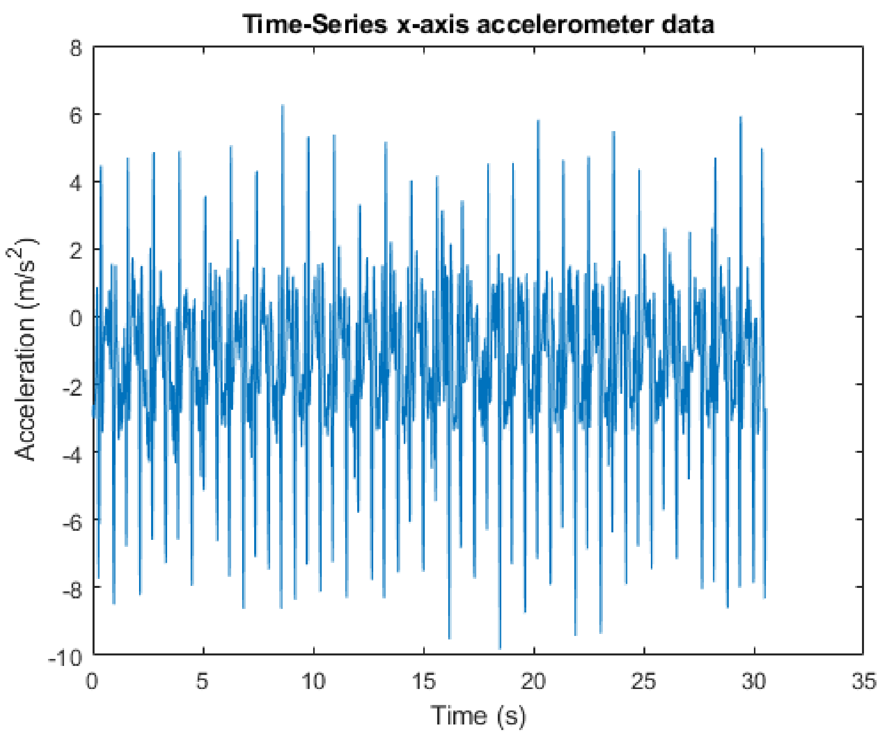
Omar Ramos Escoto

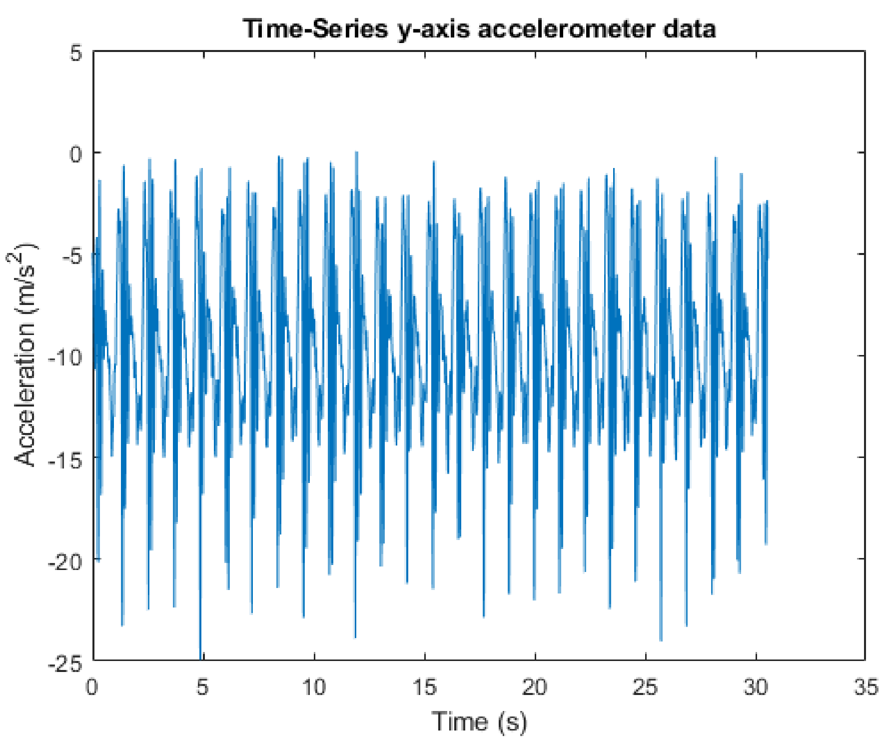
Due: May 3rd, 2023

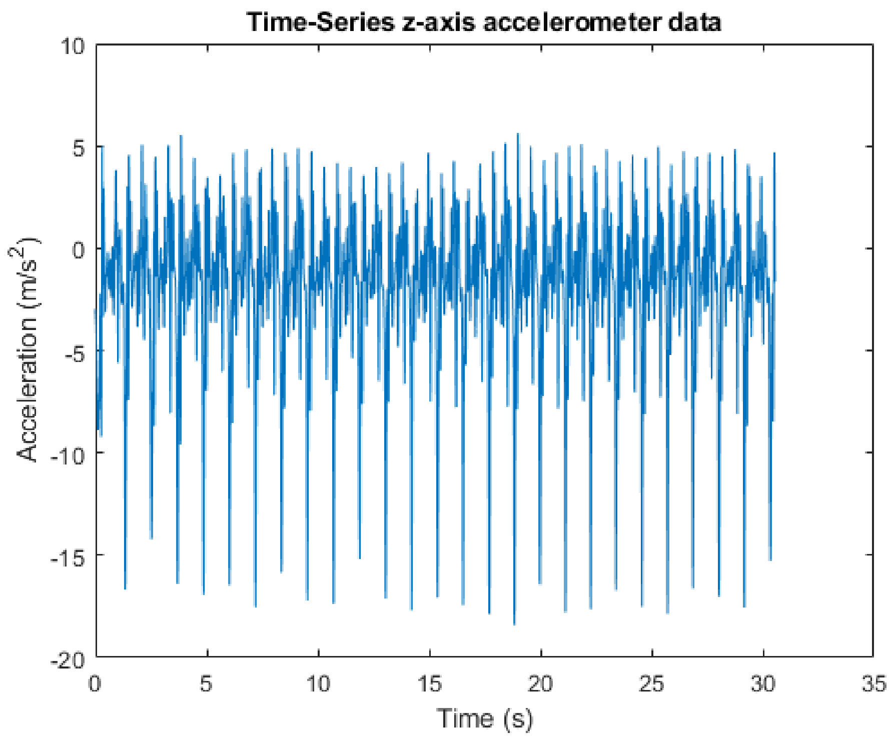
ME 220
Introduction to Sensors
Spring 2023

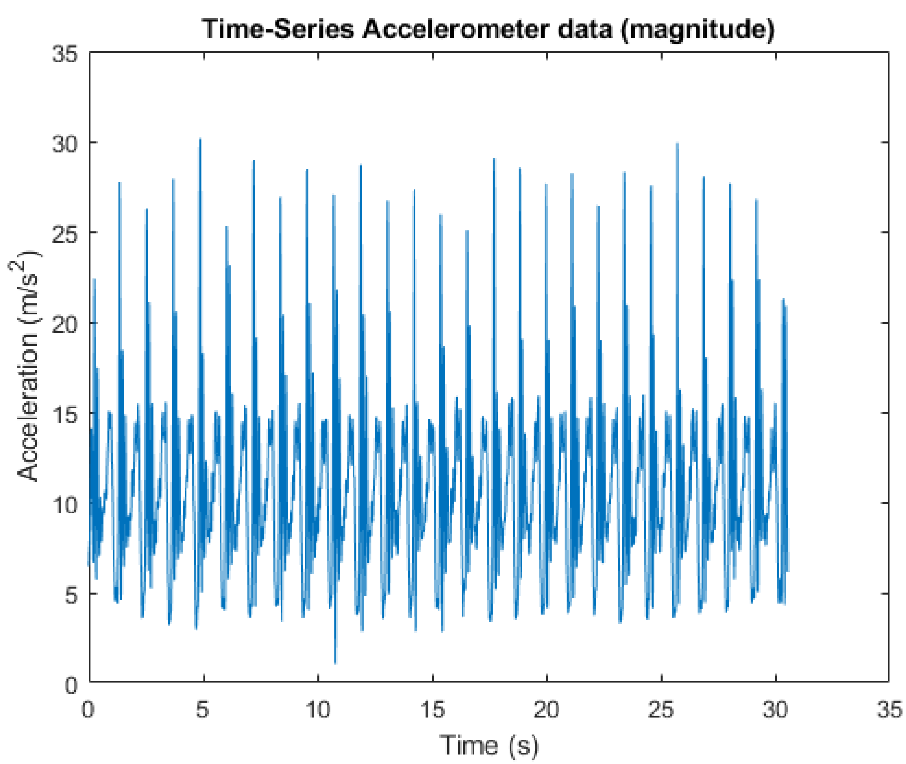
Part 1: A Basic Algorithm

Q2.



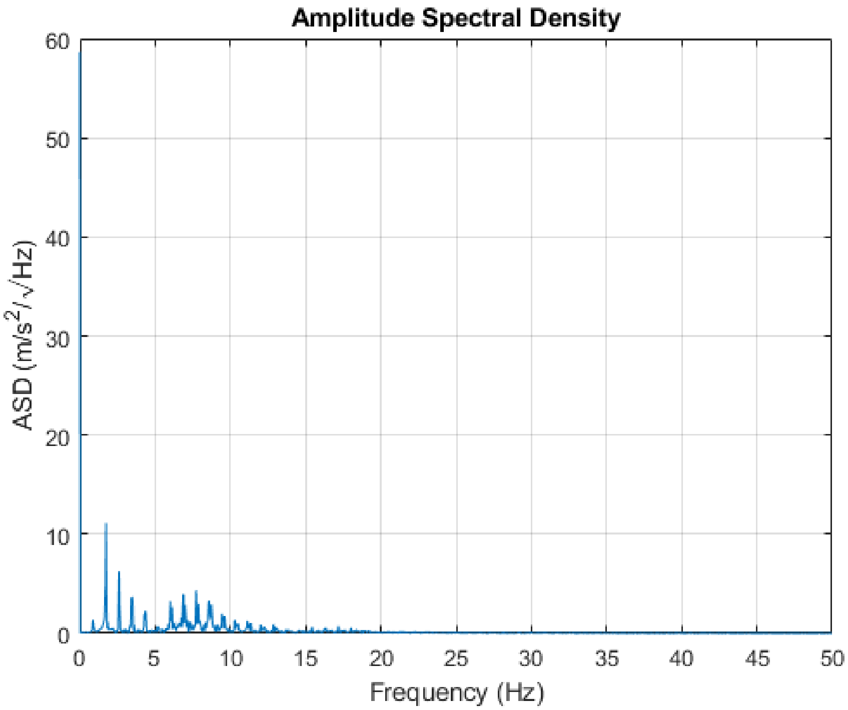


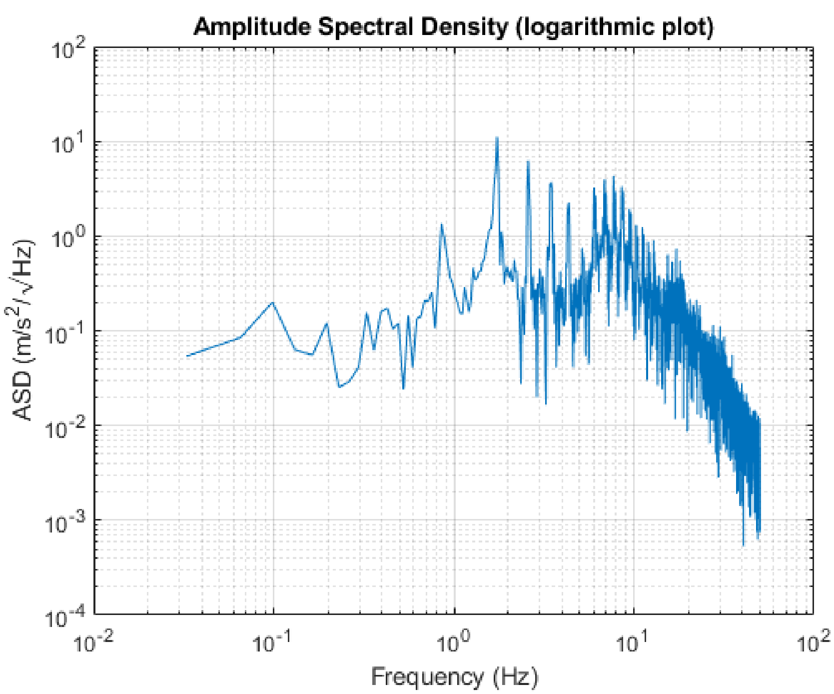




The signal looks like it has many periodic peaks across all axes which most likely correspond to the individual steps.

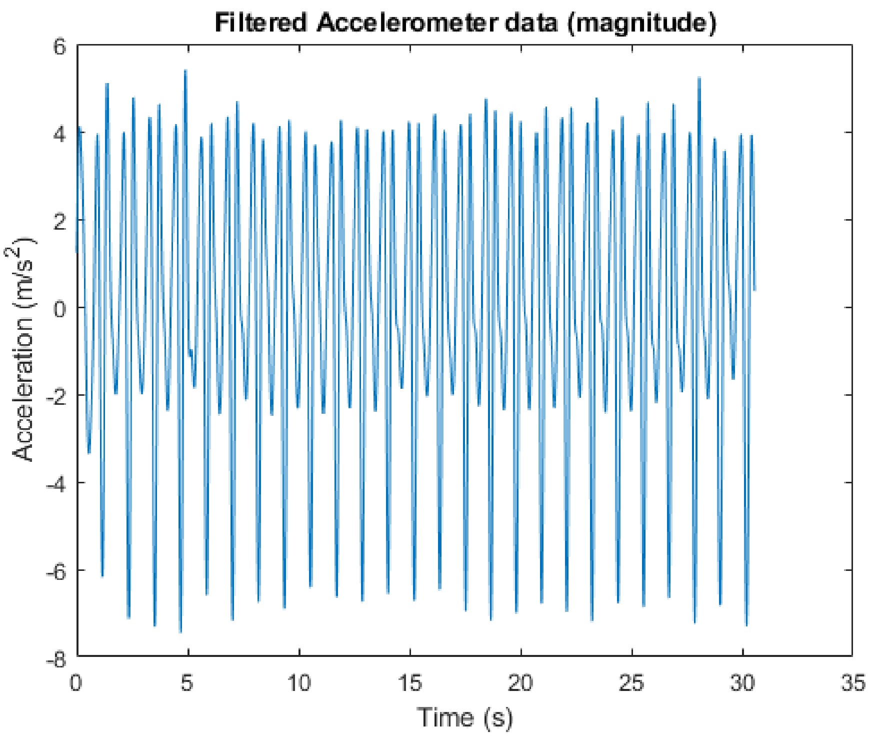
Q3.





From the plot, we can see that there is a very large amplitude signal in the data at 0 Hz. This is most likely the constant offset of acceleration due to gravity that needs to be filtered out. Most of the energy in the signal is the 1-10 Hz range, peaking between 2-3 Hz, with very small relative amplitudes past that range.

Q4. I designed a Butterworth bandpass filter (4th order) to separate the steps from other signals, such as gravity acceleration, with as flat of a frequency response as possible in the bandwidth of the signal I wanted to extract. Butterworth filters are often used for filtering motion data, so applying it for this problem seemed appropriate and filtered the data as desired. The order was chosen empirically, with 4th order providing plenty of attenuation graphically for the non-desired frequencies. The low frequency cut-off is set to .7 Hz since this gets rid of the constant gravity signal and is below expected step frequency. The high cutoff frequency is set to 3.5 Hz since steps should be below this threshold.

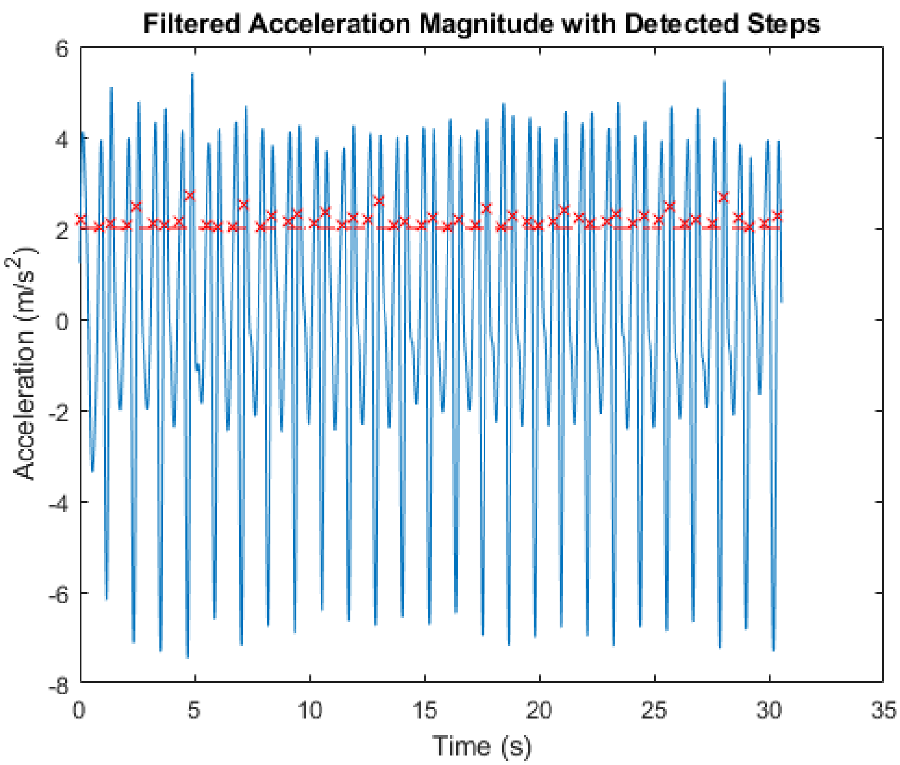


Q5. I implemented an algorithm to count the amount of rising edge crossings at a threshold of 2 m/s^2.

```
crossings = diff(magnitude_filt > 2);
steps = sum(crossings == 1); % Count positive crossings
fprintf('Number of steps: %d\n', steps);
```

Q6. My algorithm counts 53 steps, which is an under count by only a single step from what I counted manually! I might have cut off that extra step when trimming out the data or it might have simply not met the threshold for the algorithm when compared to background noise that I am not counting due to my threshold. Since I am counting rising edges, it might also be the case that the acceleration magnitude did not go below the threshold before coming back up so the step would be missed.

The graph below shows the filtered acceleration magnitude with a dotted line indicating the threshold and the 'x' marks tallying the indices at which the acceleration magnitude crosses the threshold values (counting rising edges, so the acceleration needs to go below the threshold before being counted again).



Q7.

Actual Step Count	Estimated Step Count	Difference
60	62	+2
60	61	+1
60	62	+2
60	59	-1
60	62	+2

Lab 2: Step Counter

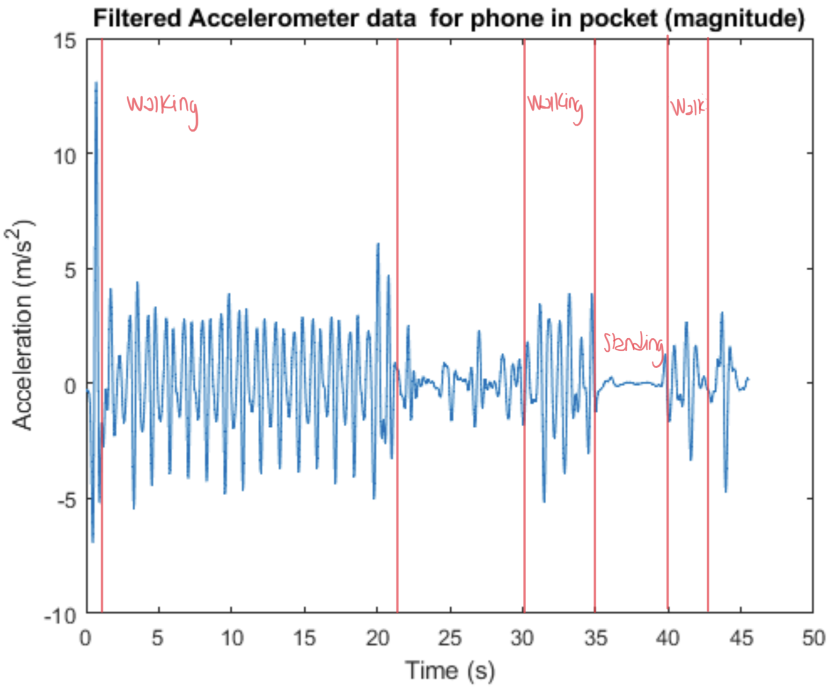
Omar Ramos Escoto

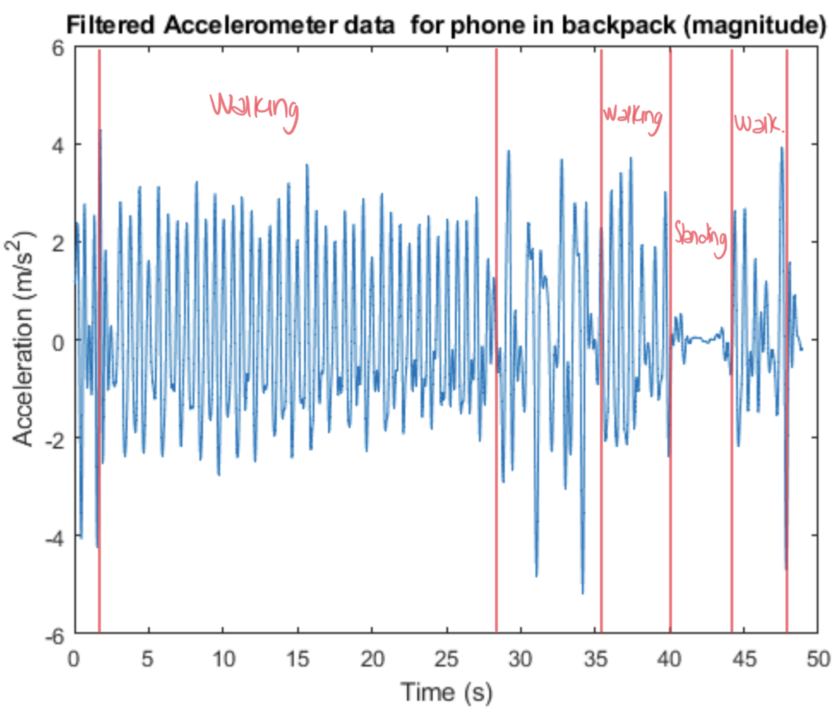
Due: May 3rd, 2023

ME 220
Introduction to Sensors
Spring 2023

Part 2: Beyond Thresholding

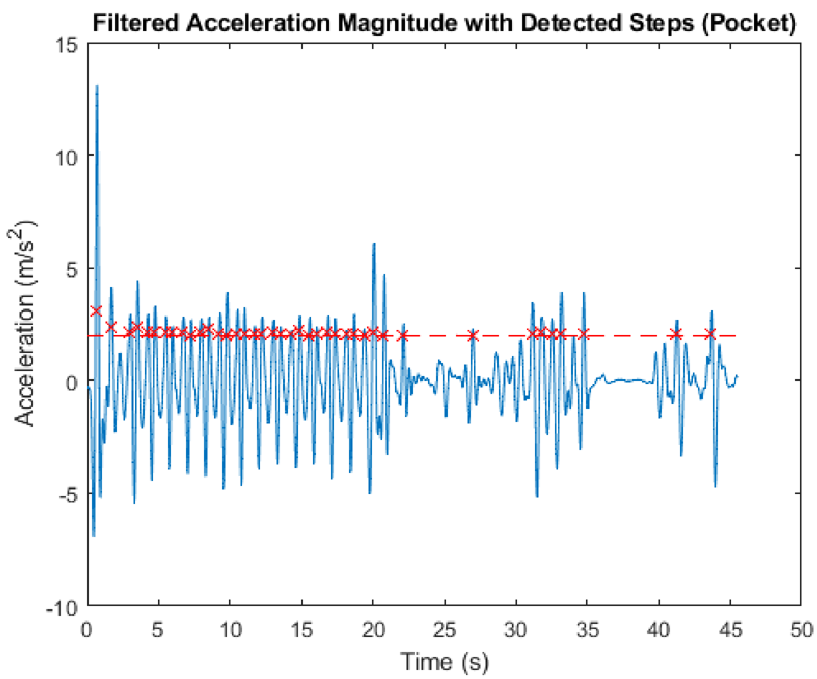
Q3.

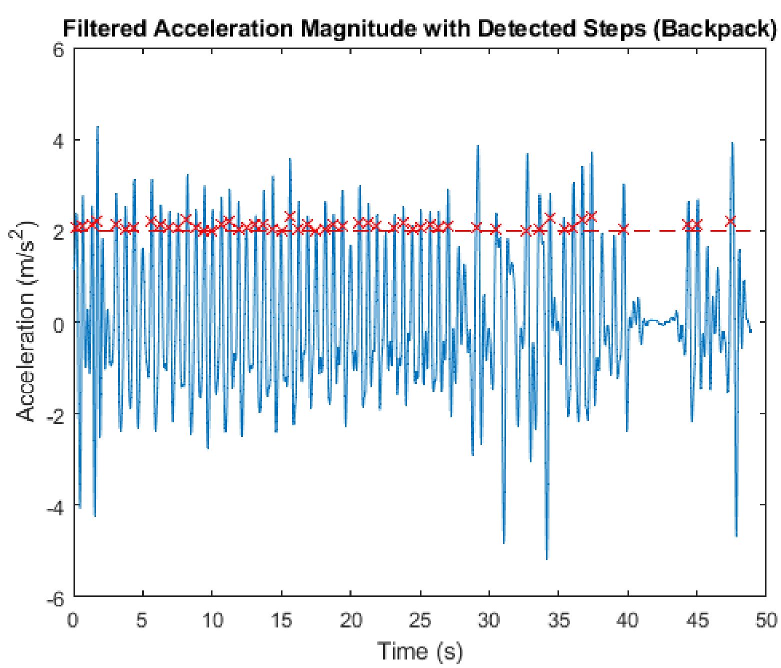




I can distinguish steps for motion mainly from the periodic nature of the steps and similar magnitude. There are big spikes when standing up and down, as well as when putting my phone in my pocket, and also flat areas where I was standing still.

Q4.

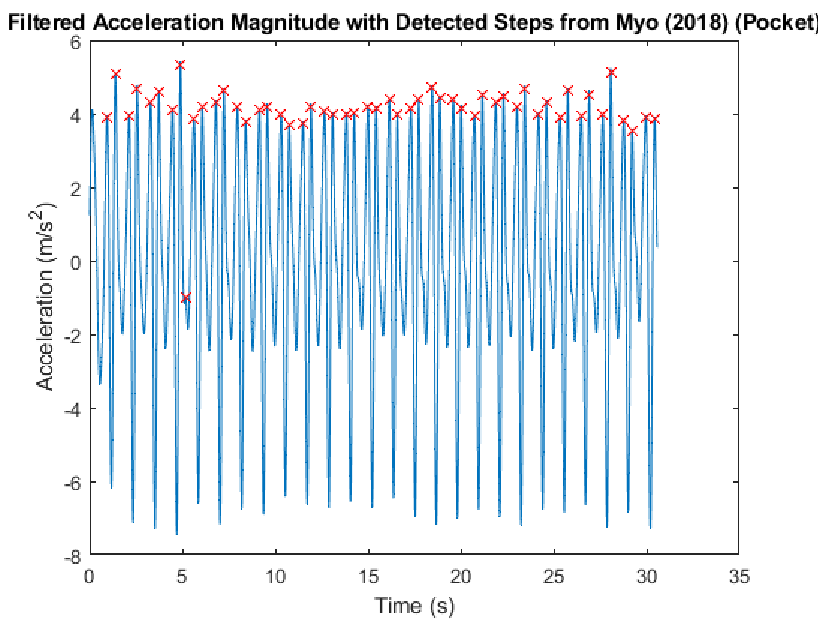




My algorithm from part 1 actually performed much better than expected, only adding 4 steps to the pocket data set and 3 steps to the backpack data set from what I counted manually (36 steps in my pocket, 50 in my backpack). It seems like it added those steps for both data sets when I put my phone in my pocket or bag, and I believe one step when I am standing up and sitting down as well as an extra step when taking my phone out for the pocket data set. I would imagine this has to do with those motions having large magnitudes of acceleration and my algorithm relying on an acceleration threshold.

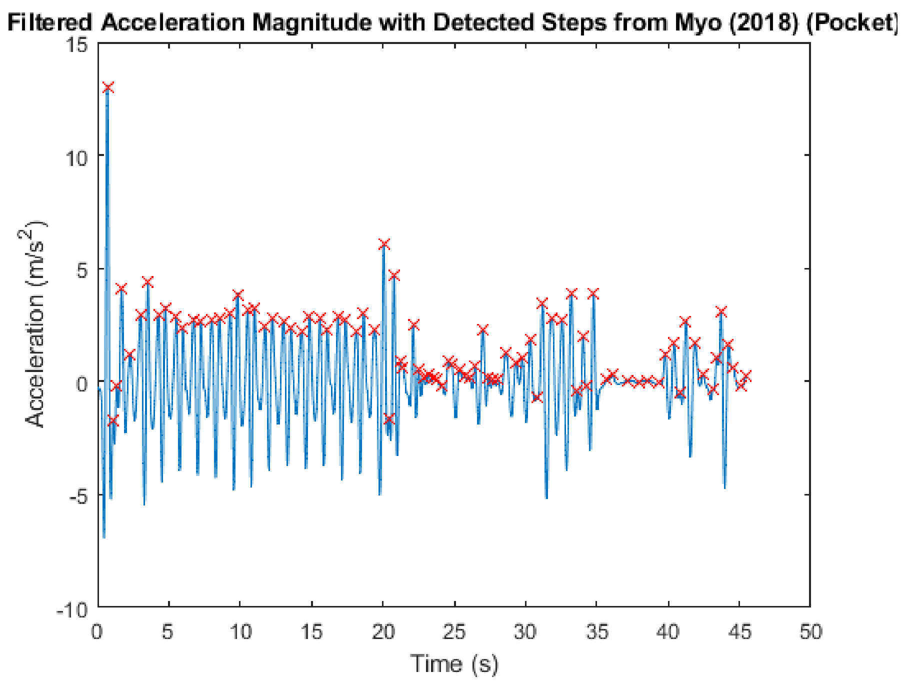
Q5. I chose to implement the method outlined in Myo (2018), which is somewhat of an adaptive threshold method.

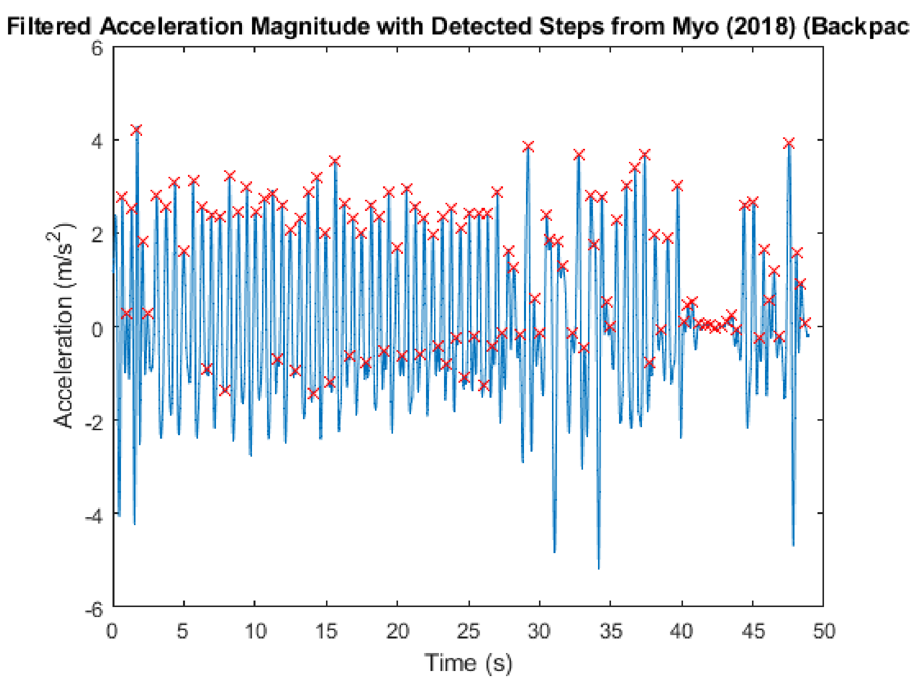
Q6. The algorithm counted the steps perfectly, calculating 54 steps. Therefore, it does perform better. However, graphing where this extra step comes from, we can see that it most likely comes from a false step, a peak found in a valley between steps. Thus, even though the algorithm does perform similarly to the rising edge algorithm I used in part 1, it seems like it is not for the right reasons and could be extremely sensitive to peaks anywhere in the signal that might not be a step. The points where the steps are counted can be seen below:



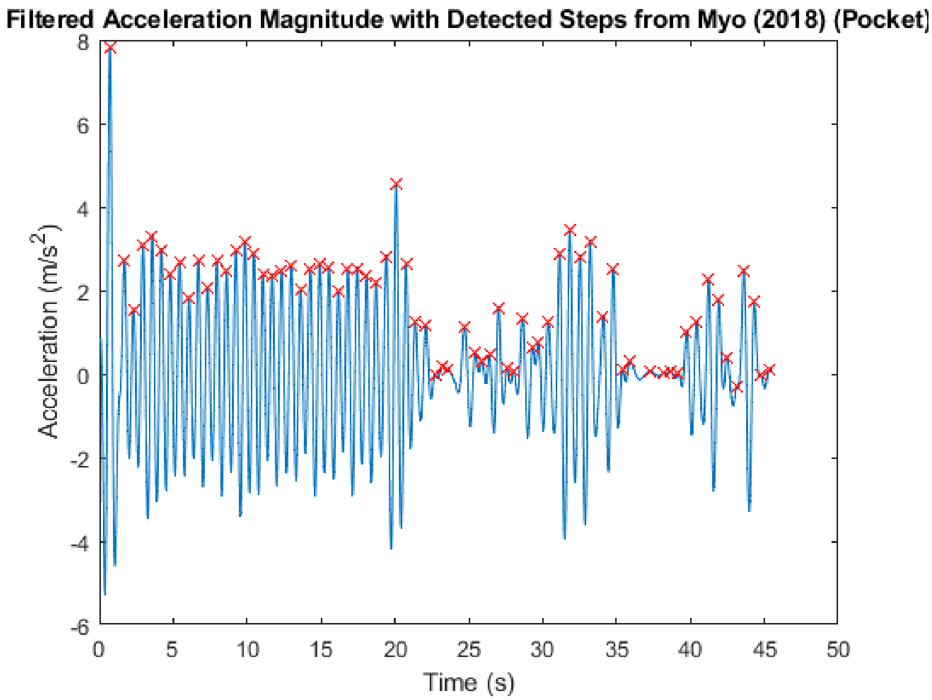
Q7. The improved algorithm proposed by the Myo (2018) paper seemed extremely promising, especially since I had had relative success with processing steps through thresholds. The algorithm takes as input the filtered data, its magnitude and uses peak detection to count steps as long as the step signature can be detected between peaks (two "occurrence points" happen between the peaks). Because it finds the peaks as it goes, it is somewhat of an adaptive threshold method, which seemed like the most intuitive way to improve upon my earlier algorithm and extremely promising from the paper results. However, the algorithm did not work very well, essentially doubling the estimated step count for both data sets. It did not perform any better on the areas in which my earlier algorithm didn't work because it is way too sensitive.

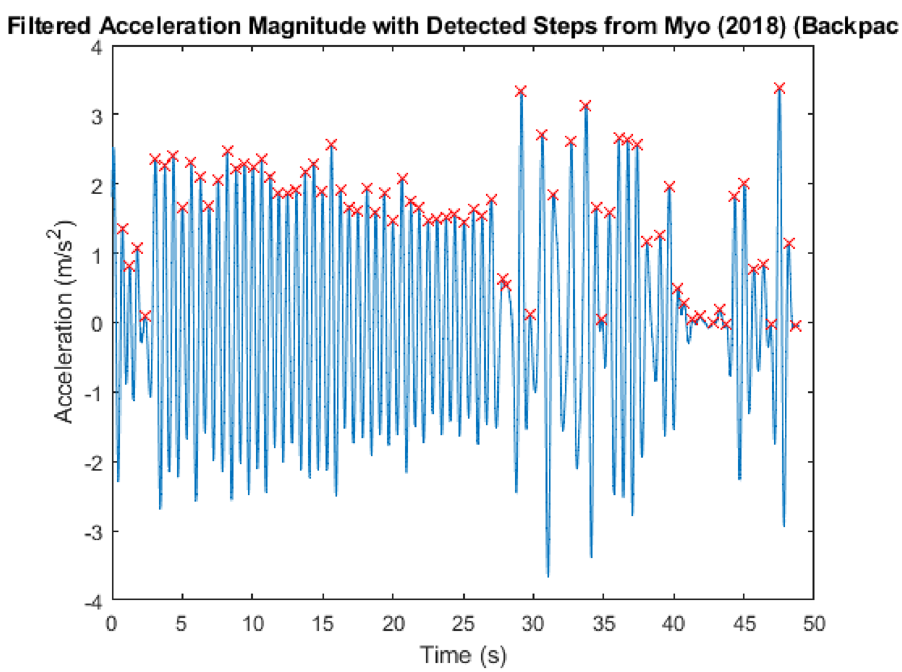
Using the same filtered data from my original algorithm, this new algorithm detected 87 steps when my phone was in my pocket and 114 on my backpack (36 steps in my pocket, 50 in my backpack was the actual count). The data can be seen below, where it is obvious that this algorithm is extremely sensitive to peaks since it is all it is doing:





If I want to get a better count, I need to filter the data more aggressively to begin with to smooth out the peaks that are sometimes found in the valleys. After filtering the algorithm data with almost no data outside of the .7-2 Hz range, it is still overcounting but less, with 71 steps in pocket and 76 in backpack. The plots for where it counts these steps is below:





With more filtering, the algorithm performs better on backpack data since there are less peaks in general, but the original algorithm still outperformed it.