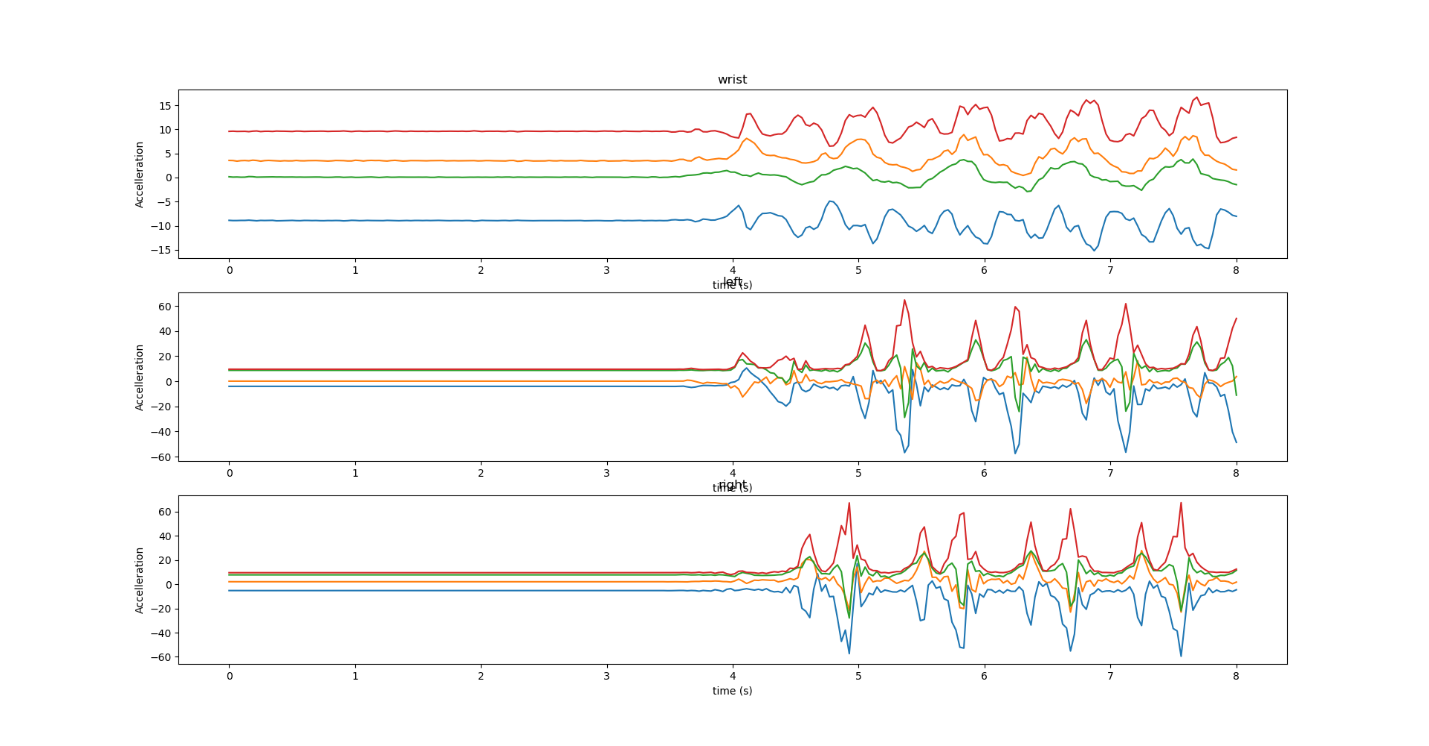
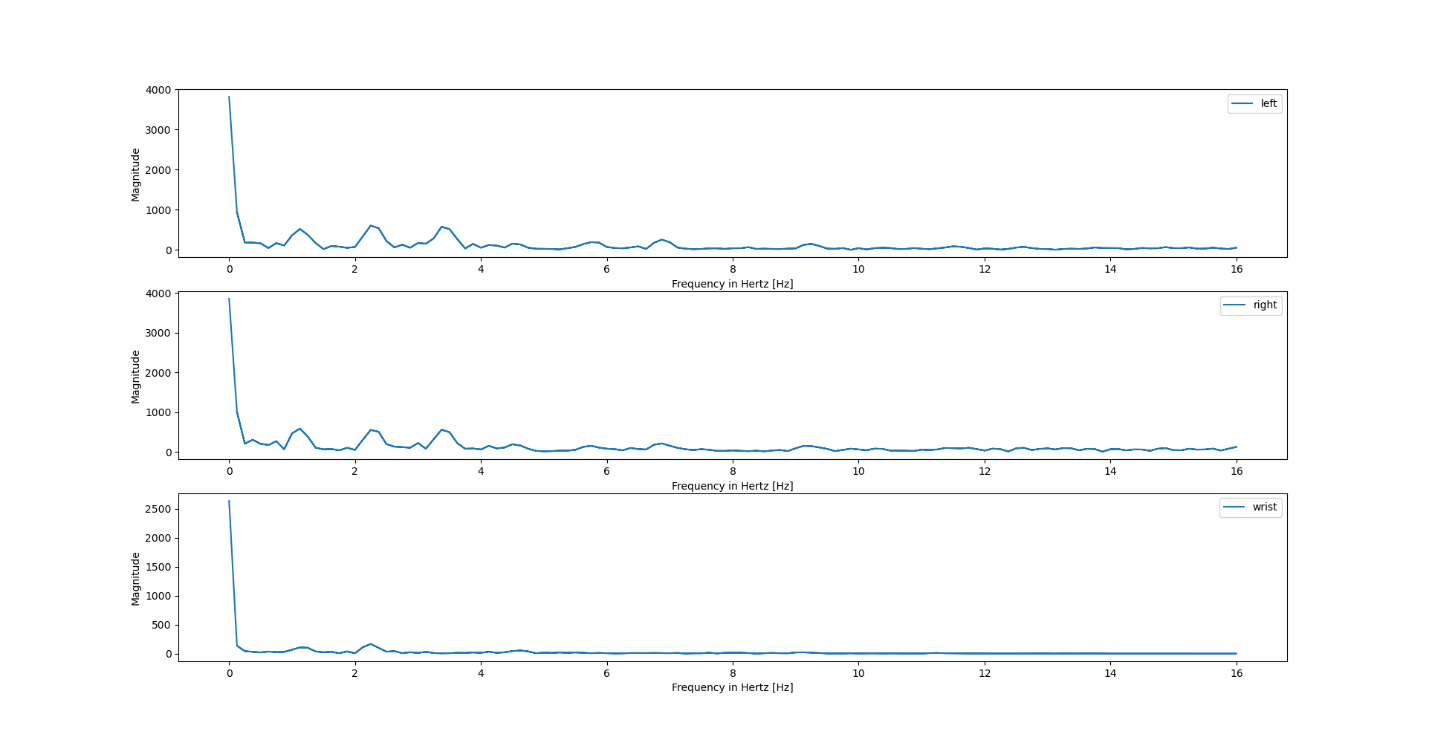
Mikhail Mayers

EE523

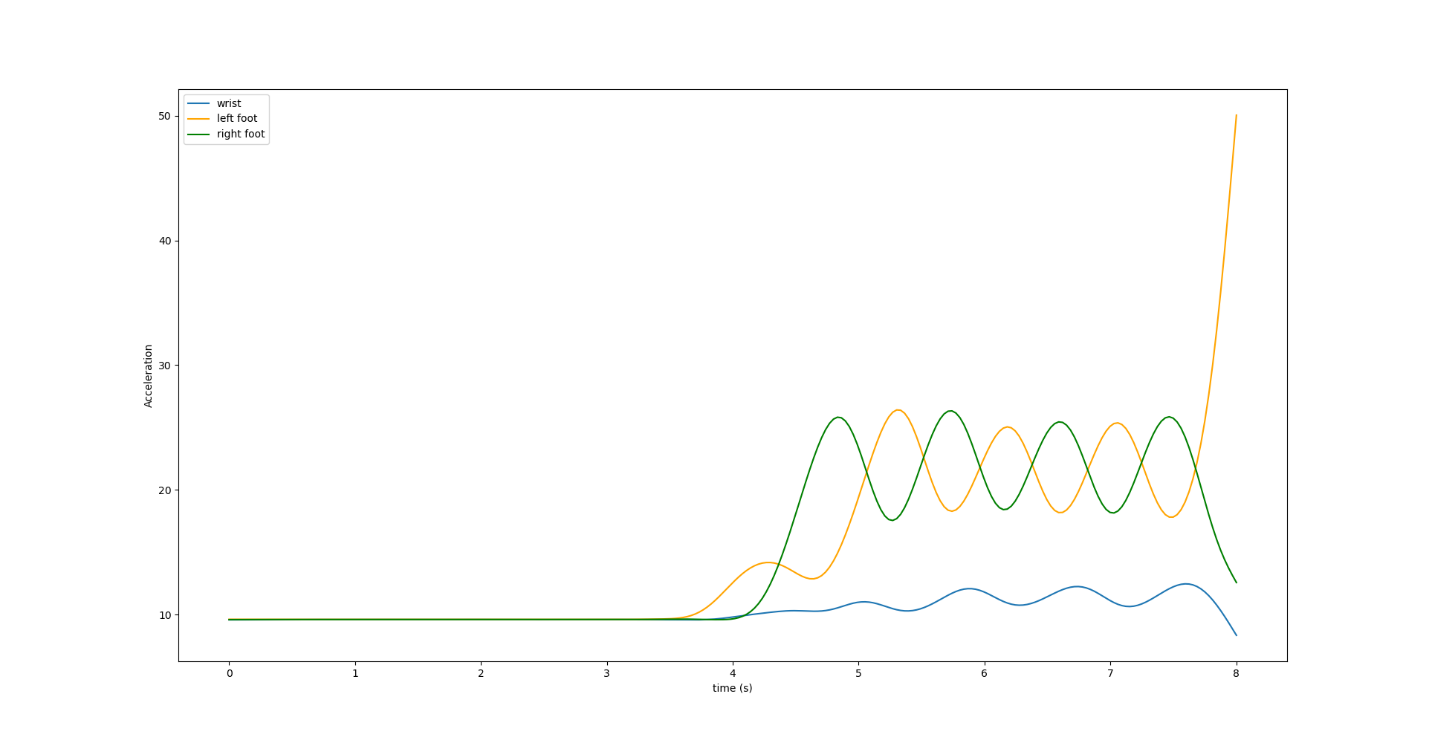
Hw1

Starting out, I immediately looked at the data and thought about the noise. Rather than looking at all of the data in the 3 axis individually, I decided to focus only on the L2-Norm of each appendage. The next step, I knew if I was going to detect anything, I would need to first filter the data before I even attempt. At first glance, I realized I needed a low pass filter to remove a lot of the jitters in the data. I looked only at the first data table to start and took a DFT to find out the magnitude in each frequency.

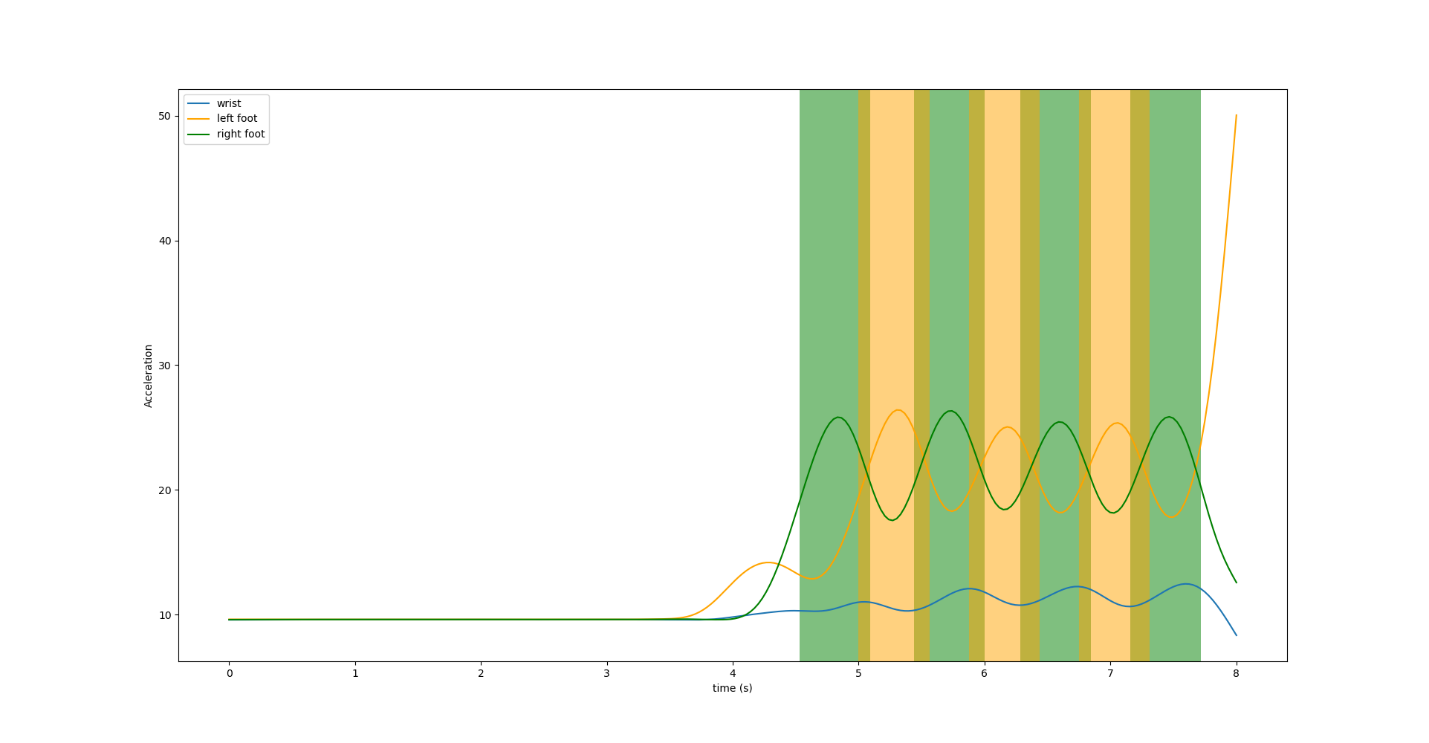




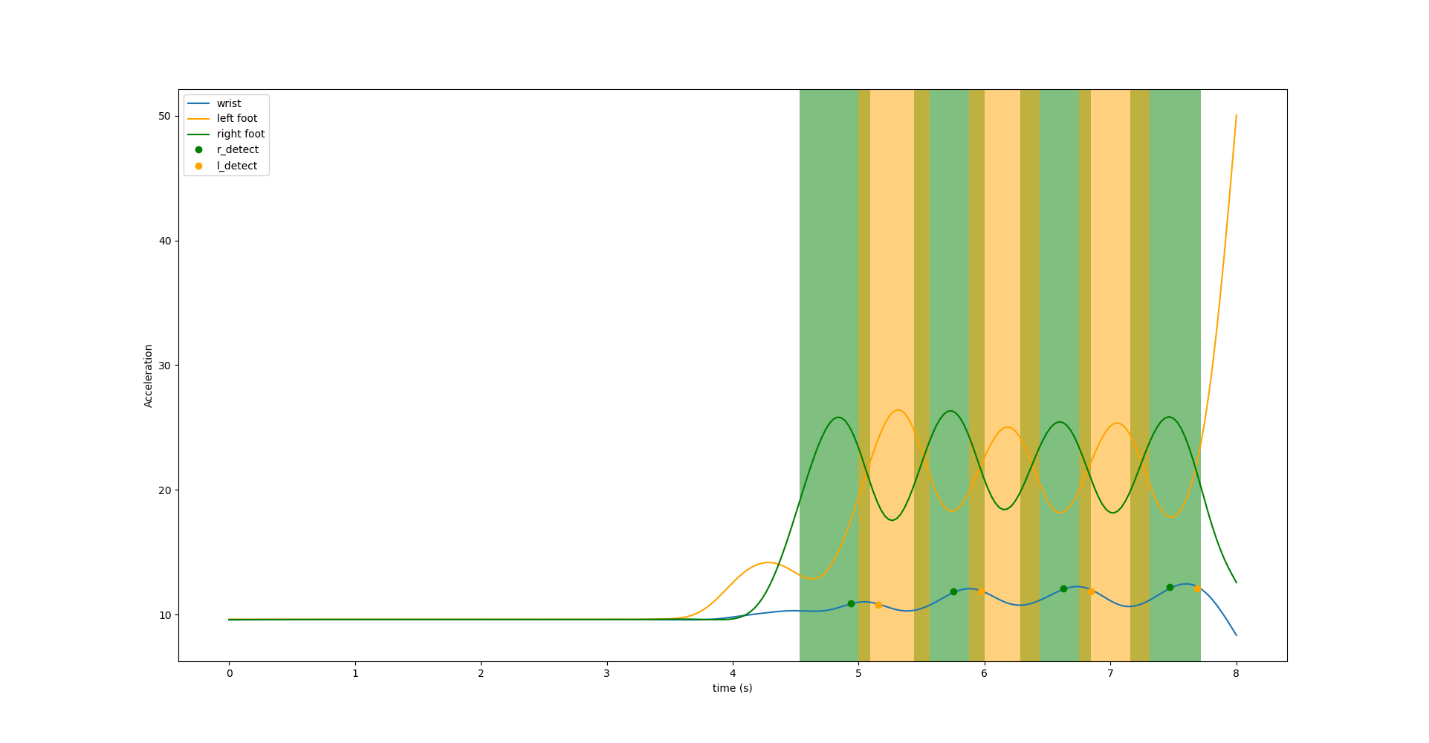
The noise looks gaussian, because it starts off very high, then tapers off as the frequencies get higher. So, I continue with my idea to use a low pass filter, and the graph looks pretty empty and I don’t see any obvious patterns.



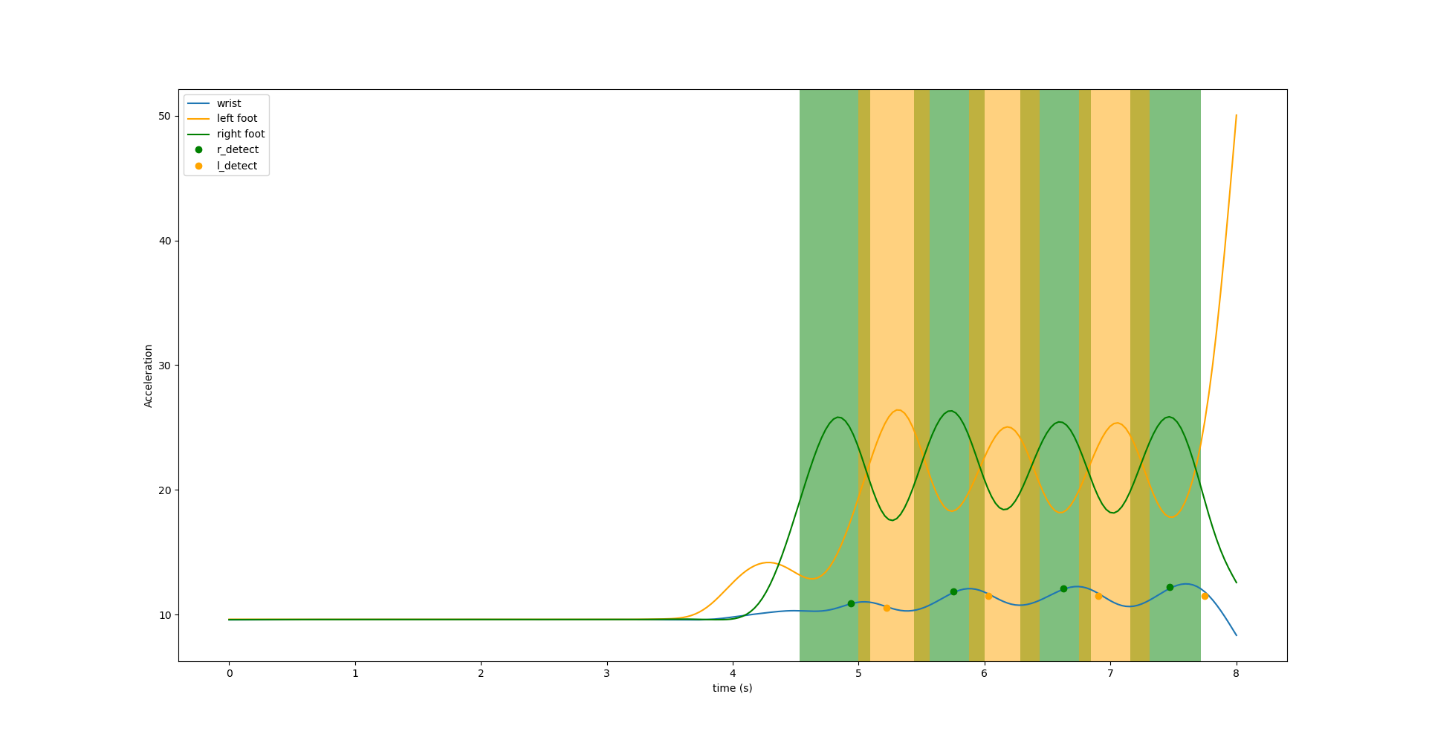
So, I decided that now is the time to find my true steps in data and mark them off. I start with finding the peaks on each foot, then I gave a threshold to the left and right, saying that “x time is a full step”. I chose the region roughly 25% the distance of the next peak before and after.



And now the pattern is obvious. The wrist signal is deterministic. There is a clear pattern when steps are taken. I noticed that the peaks of the wrist data, is situated at the intersection of the left and right foot. My assumption is that the timing of those intersections is when the step is at enough of a point that it will be a step and not just the subject shifting their weight. So as in computers, I decide to detect a rising edge as a right foot indicator, and a falling edge as a left foot indicator. This seemed to hold for most events. I accomplish this is a similar way to the step detection. I find the peaks and determine that 3 points behind the peak is a right step, and 3 points forward is a left step.



As I plot this, I noticed that the left foot detector is always lagging, and almost touching the border. But I remembered that the wrist sensor is actually on the right side, so I guess this is what is skewing the data, so I doubled the number points to 6 points forward (to the right) from the peak. This made my accuracy go up.



I the tried to quantify my accuracy, so I had to find the sensitivity and specificity of the data. I first had to find the true positives, which were how many times I was right, and my steps fell in the regions. The false positives, which were every time I predicted and was wrong. The false negatives, which is every time I have a region that was not detected. And last is the true negatives, which are every point in time that is not an event.

The true positives were simple, I just used regions like bins in a histogram and saw if the bin was occupied by a prediction. The false positives were the times I guessed but didn’t land in a bin, or bins greater than 1. The false negatives were the unoccupied bins, or bins that had zero. And the true negatives were all the points minus the regions of guessing.

With these numbers, I just used equations learned in class. For sensitivity I used:

sensitivity = tp/(tp + fn)

|  |  |  |
| --- | --- | --- |
| label | Sensitivity | Specificity |
| 10015369 | 1 | 1 |
| 11053032 | 0.571429 | 0.994885 |
| 100044194 | 1 | 0.990769 |
| 105945872 | 0.888889 | 1 |
| 110414272 | 1 | 0.996 |
| 110835409 | 0.8 | 1 |
| 110949721 | 0.875 | 0.995708 |
| 111417456 | 0.956522 | 1 |
| 111810216 | 0.909091 | 1 |
| 113654115 | 0.901961 | 0.993827 |
| 114515454 | 0.9 | 0.996226 |
| 114624795 | 0.888889 | 1 |
| 114719341 | 1 | 0.996109 |
| 120031387 | 0.84 | 0.981572 |
| 120601909 | 1 | 0.941414 |
| 132015917 | 0.813433 | 0.99782 |
| 132455848 | 0.780523 | 0.998081 |
| 115031838 | 1 | 0.996587 |
| 115419186 | 1 | 1 |
| 115615296 | 0.066667 | 0.992638 |
| 120026579 | 0.857143 | 0.94108 |
| 121629814 | 0.888889 | 1 |

specificity = 1-(fp/(fp+tn)).

As I was programming, I was thinking about how nice it would be if I could tweak a lot of the datapoints automatically. A lot of the hard part in this process was making small changes and seeing if that gave me better results. I think that it’s nice to have an idea about how to predict something but having to find a “sweet spot” for all the data would be very time consuming. So, I’m hoping that this class will help with that, but right now I’m still a little lost on how to do any of that.

