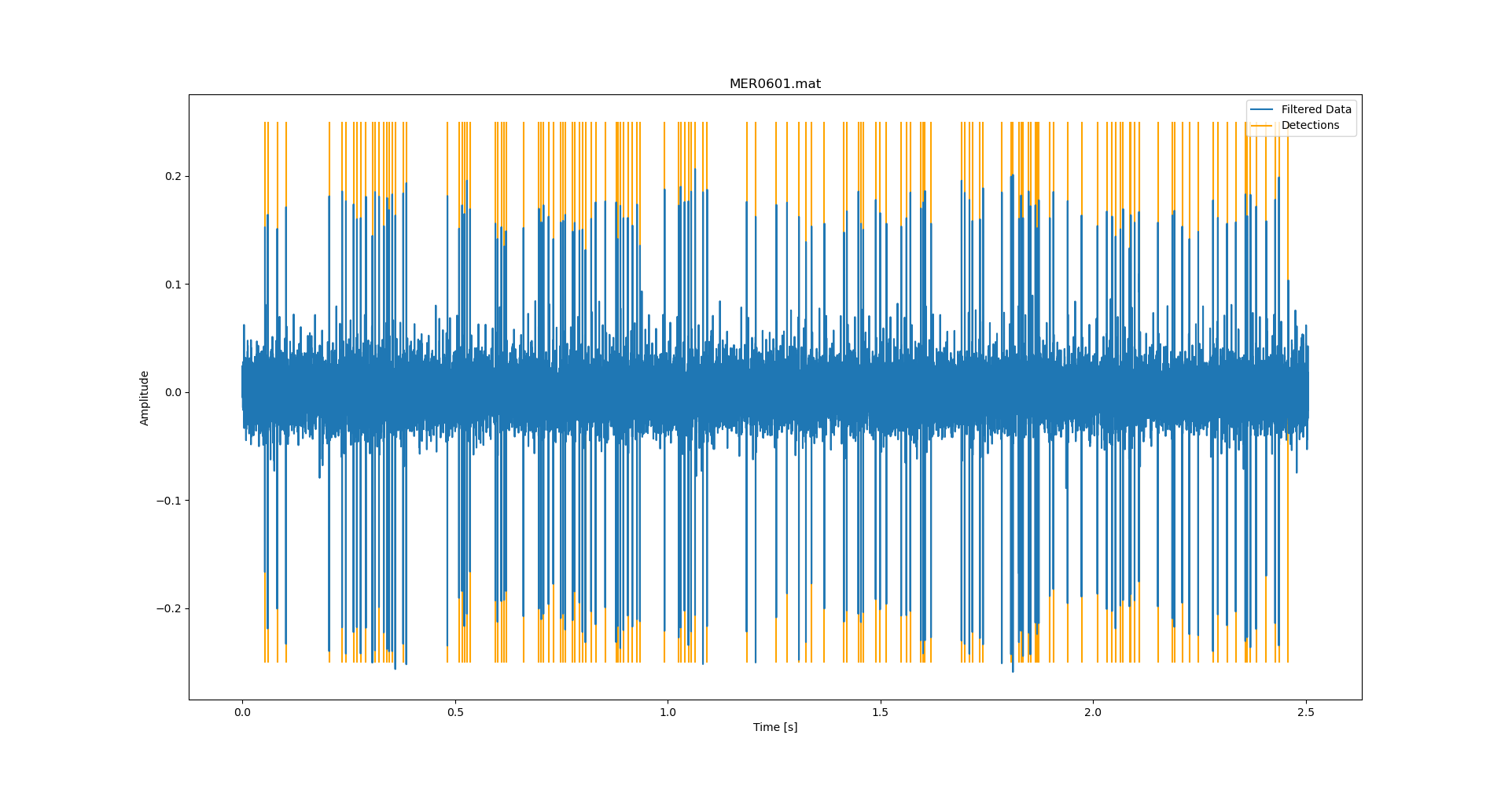
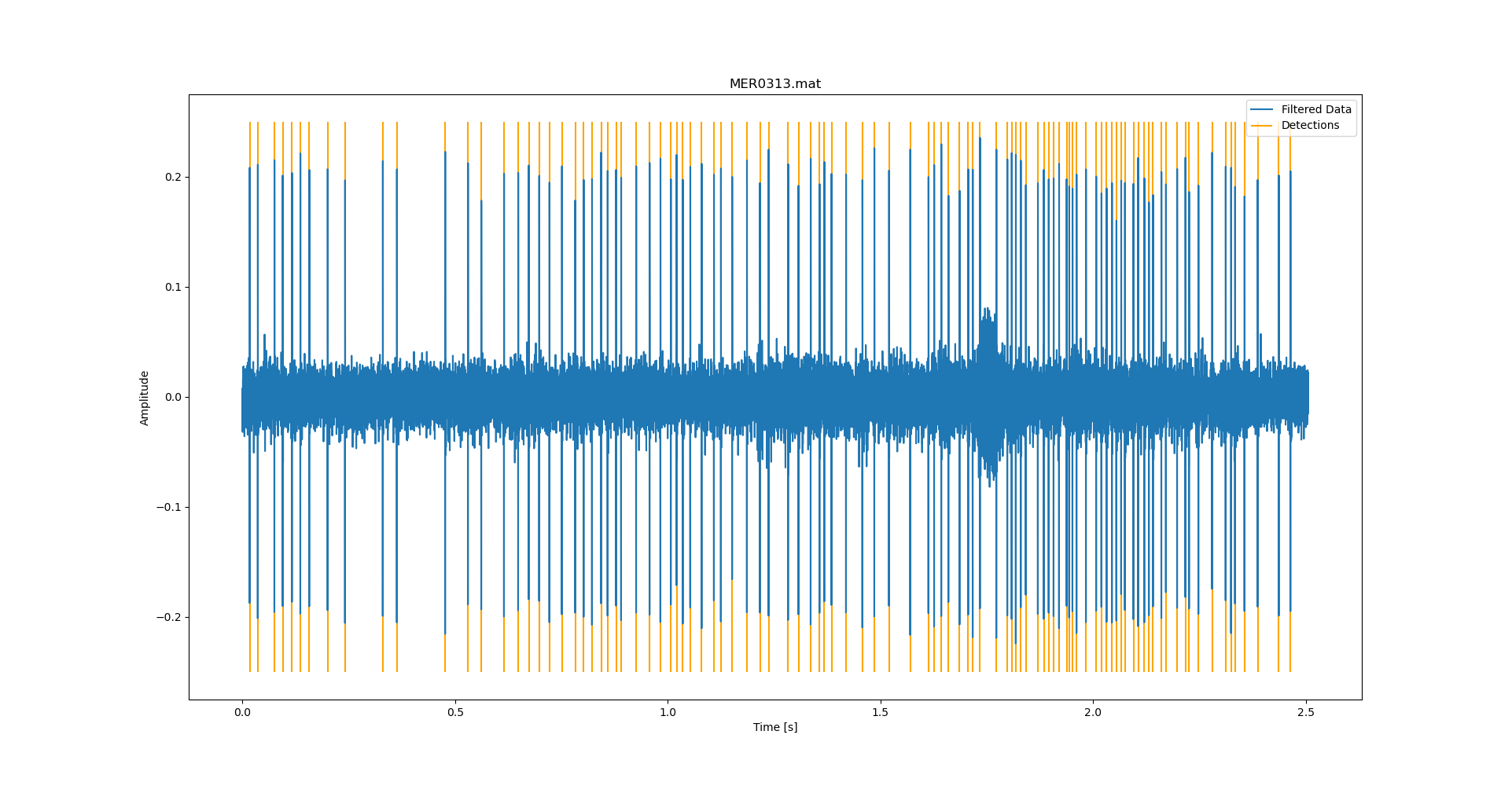
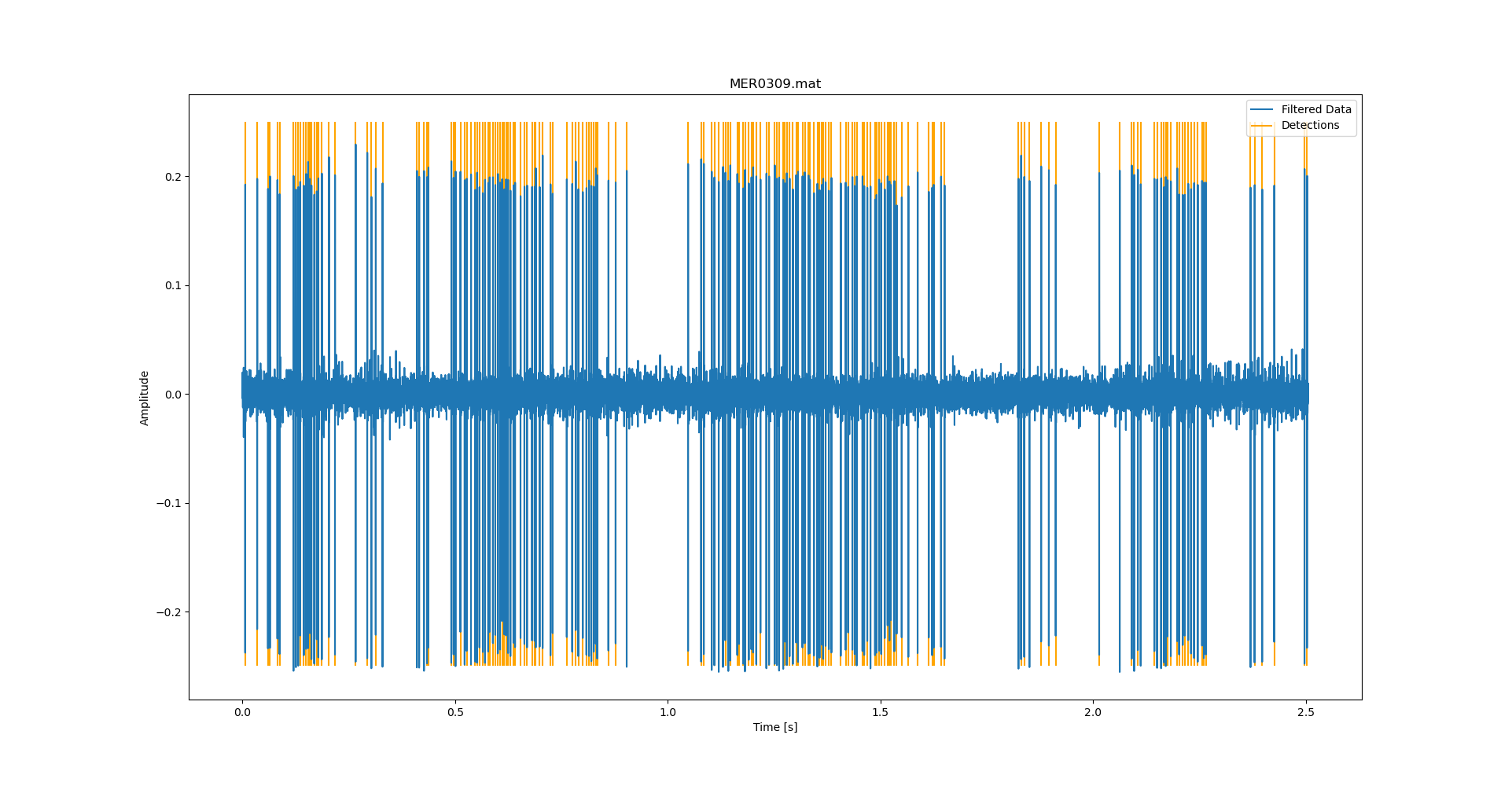
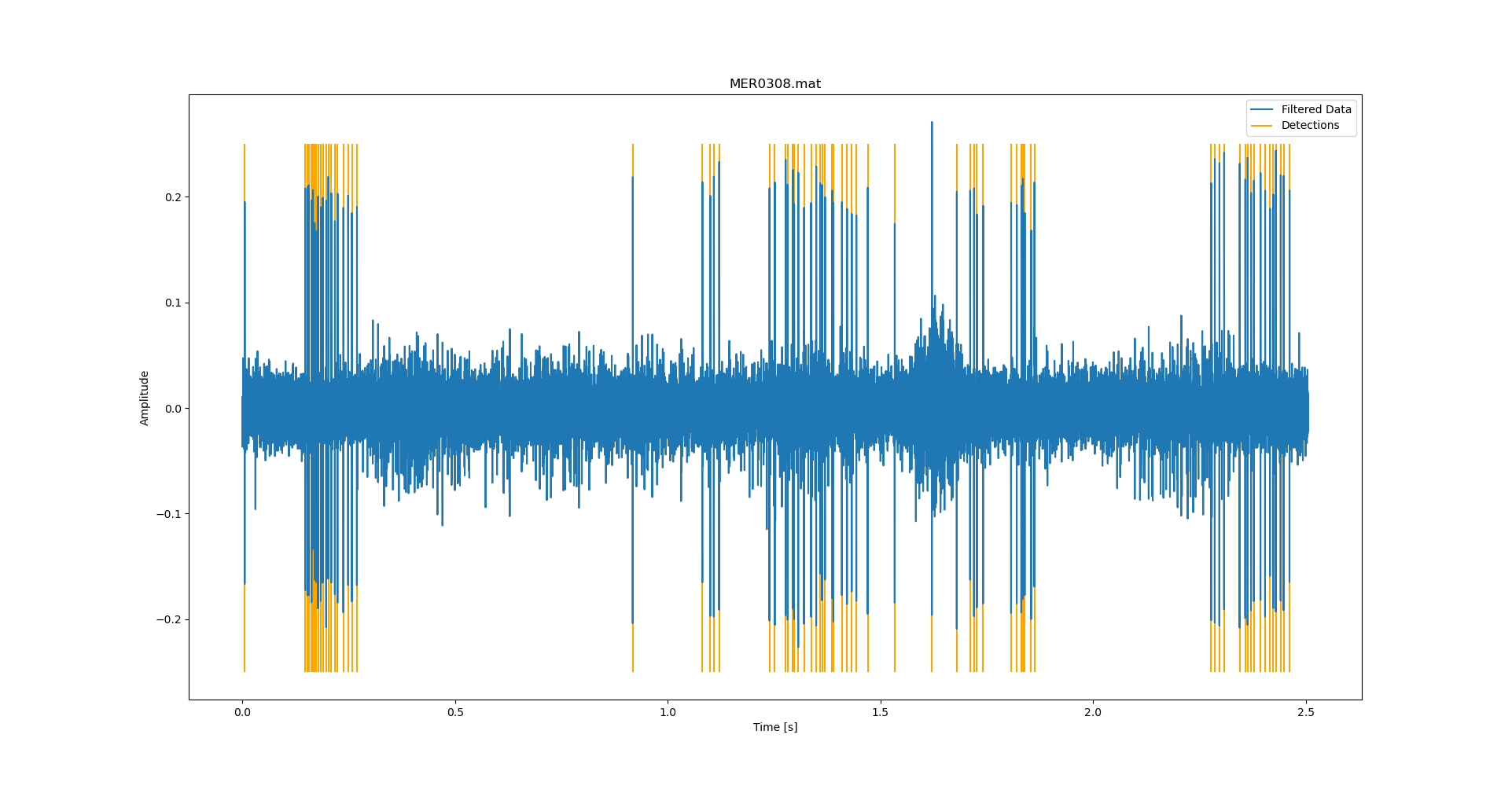
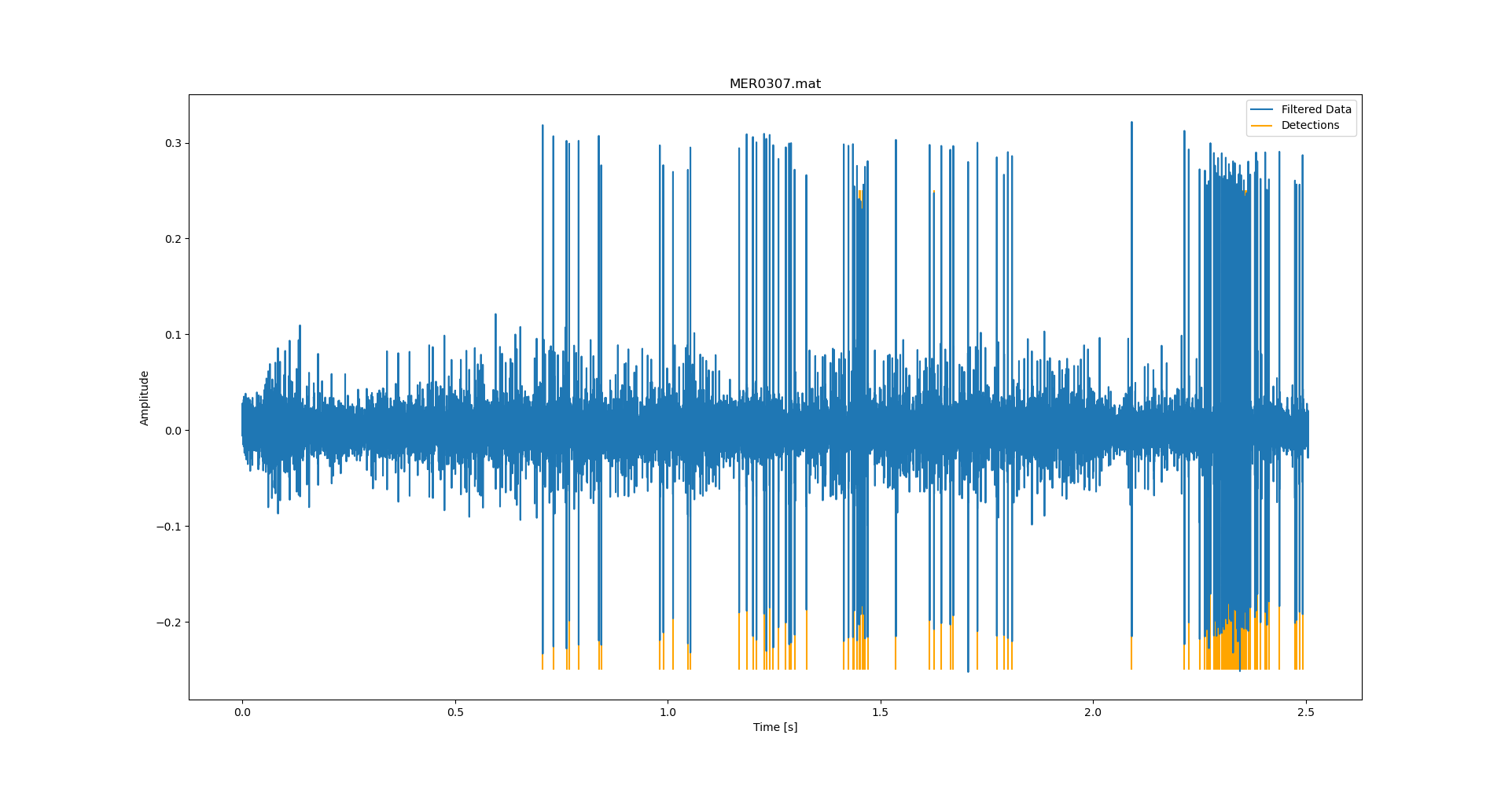
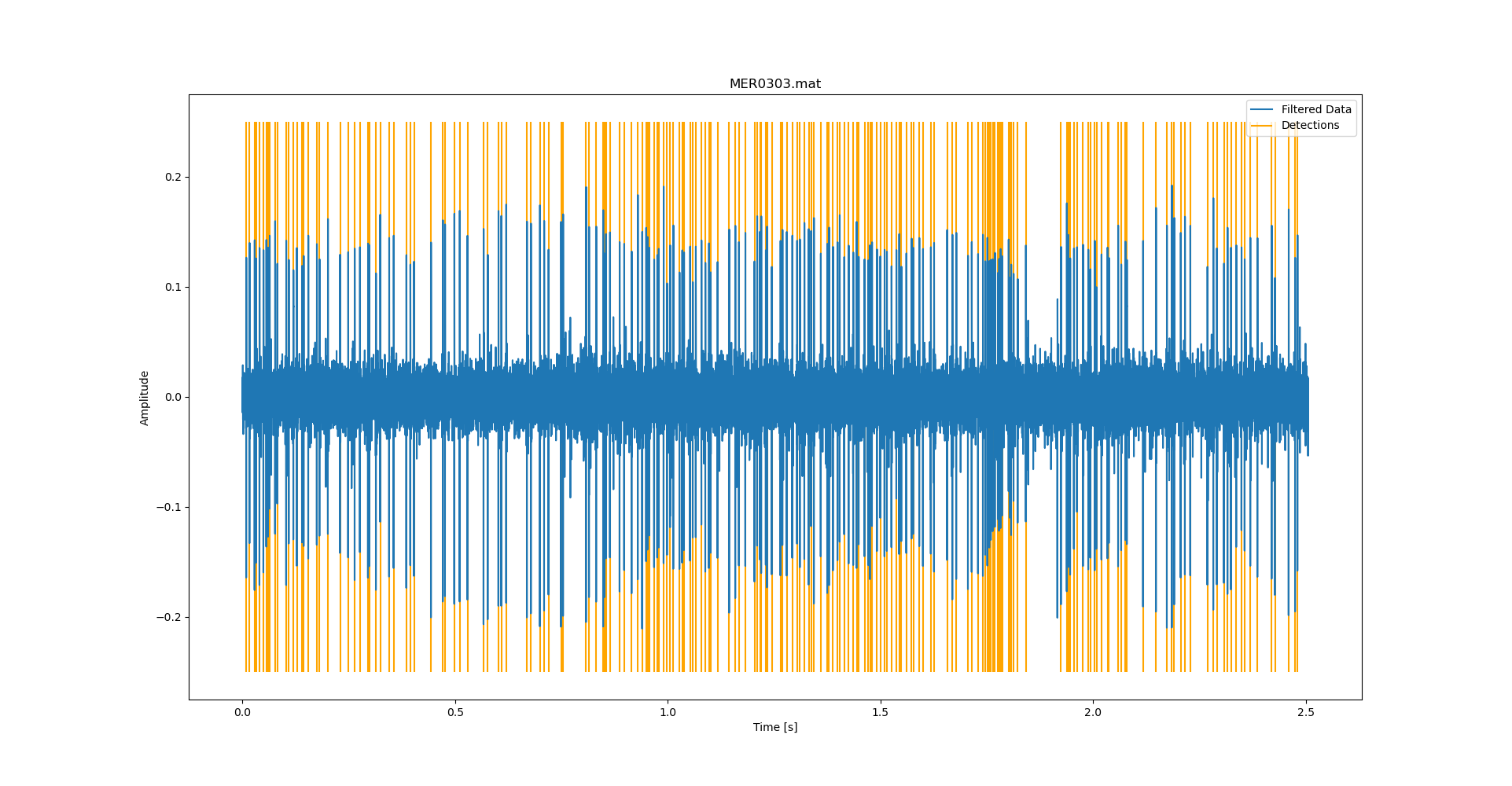
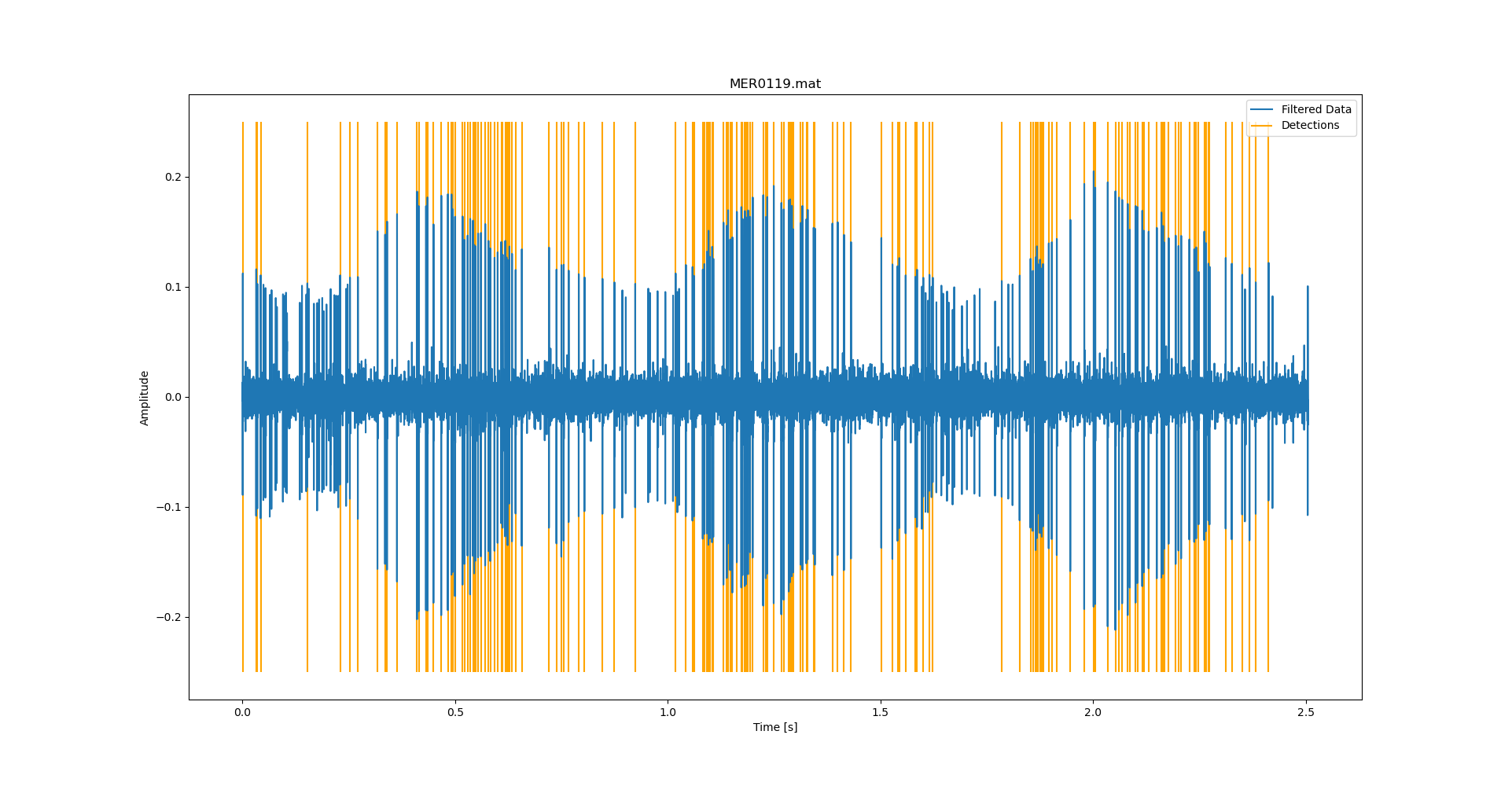
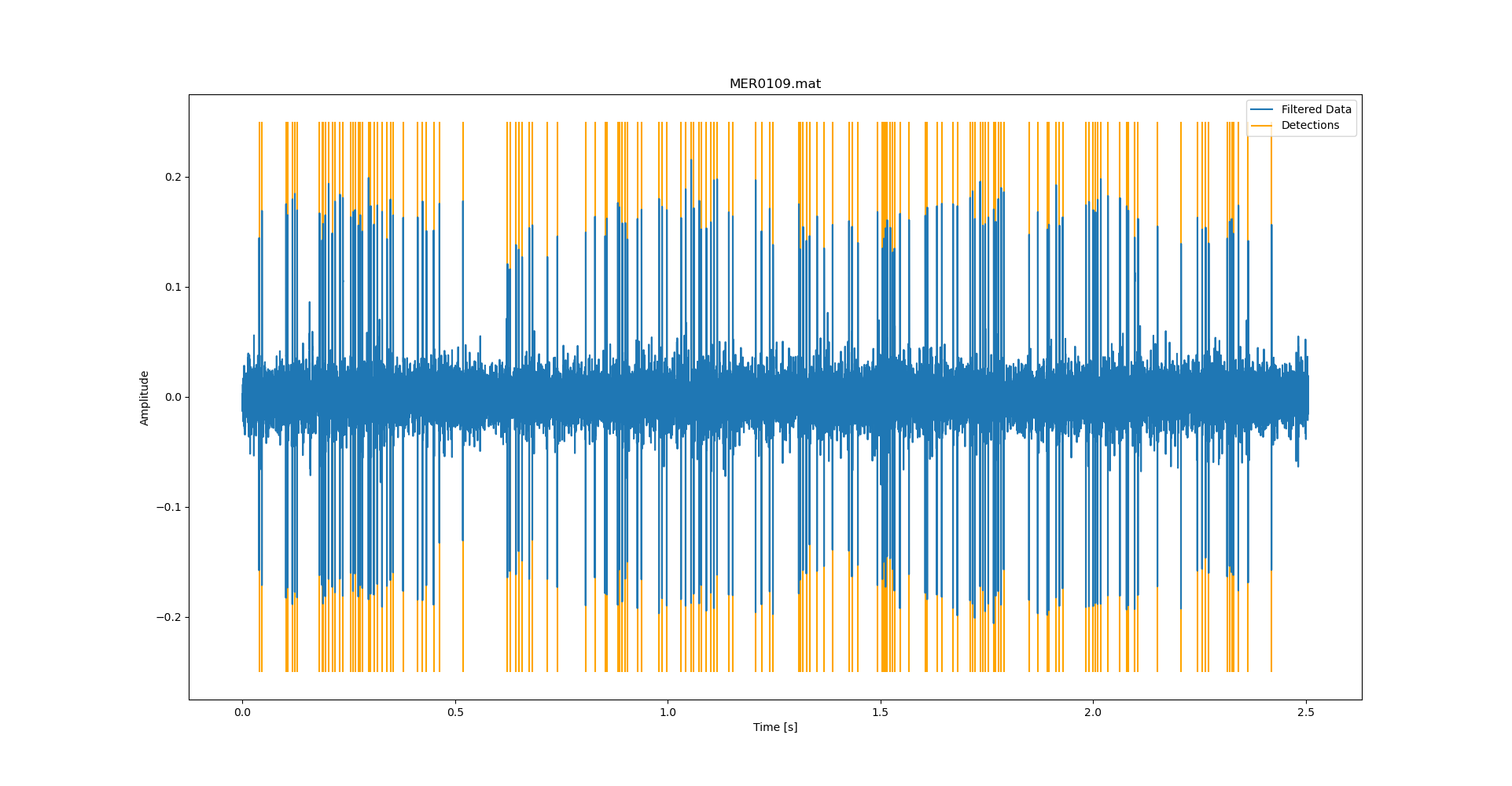
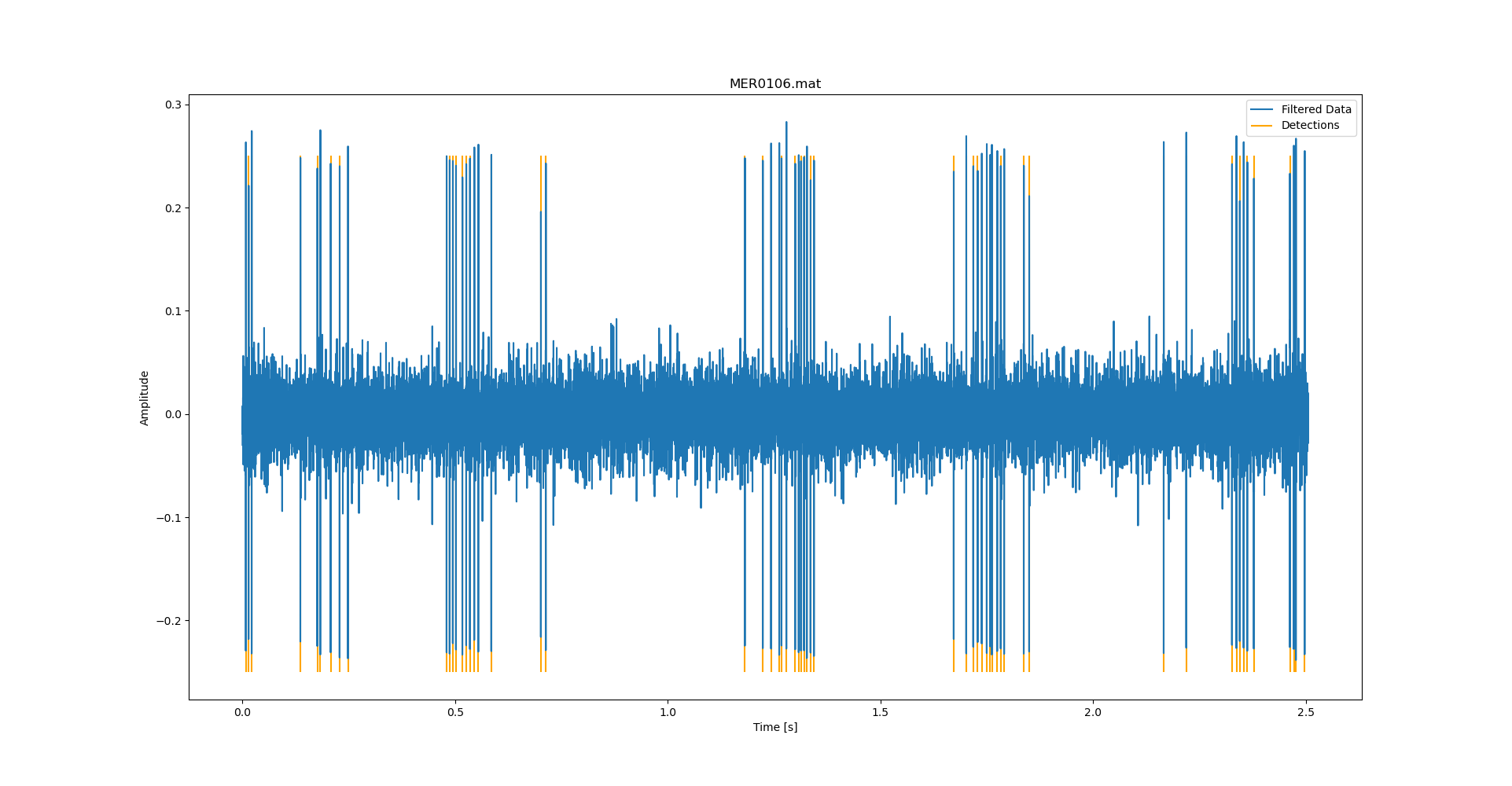
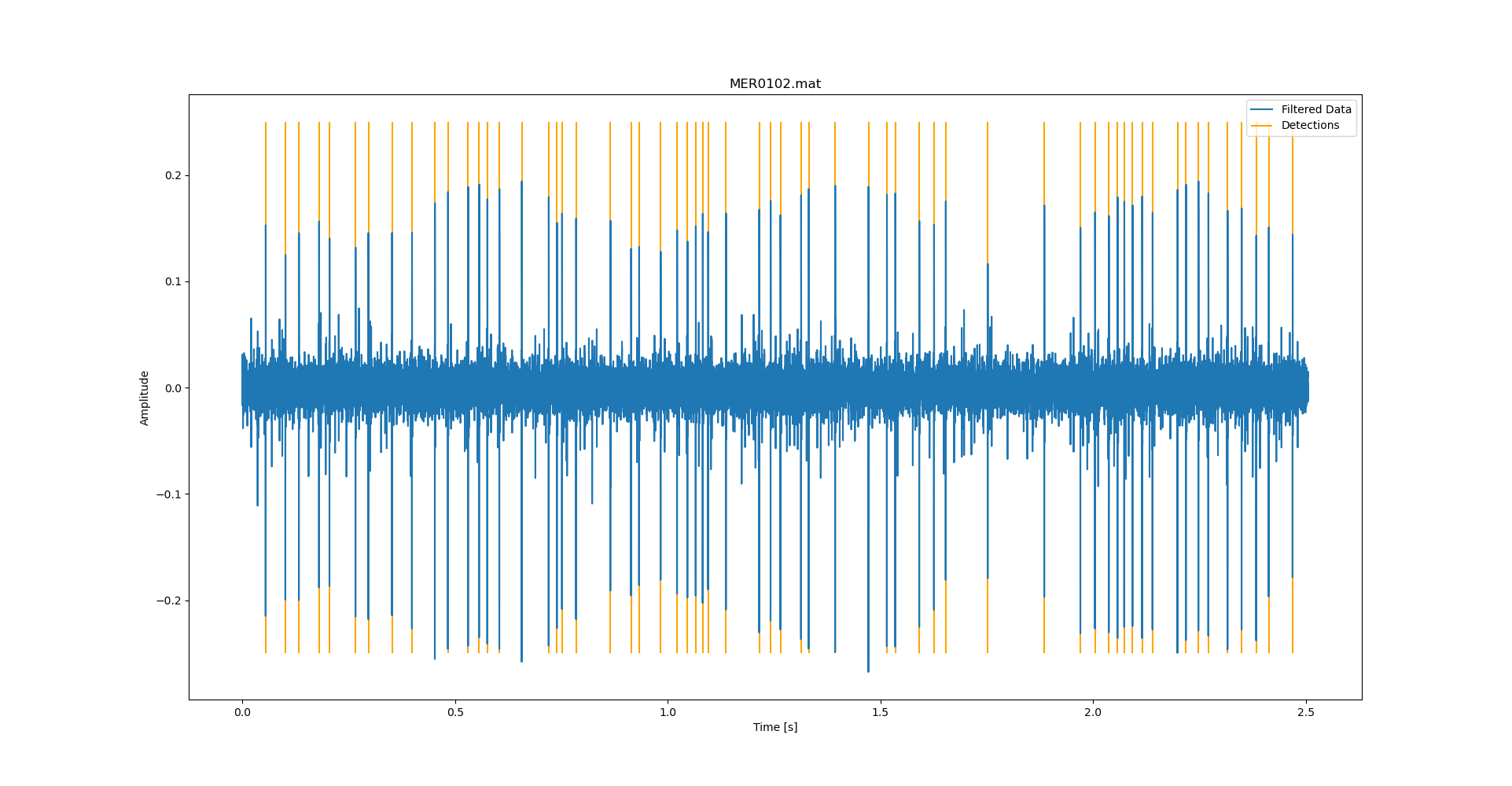
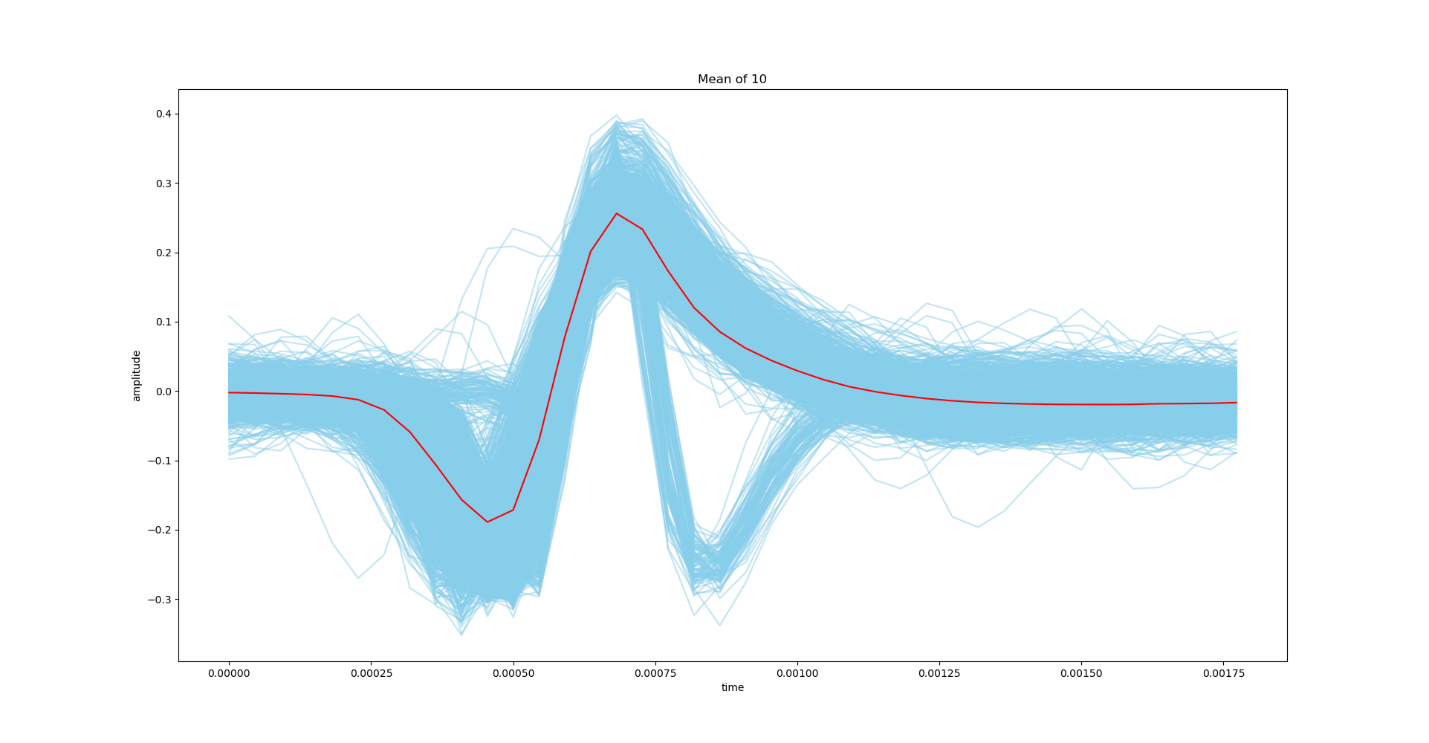
Mikhail Mayers

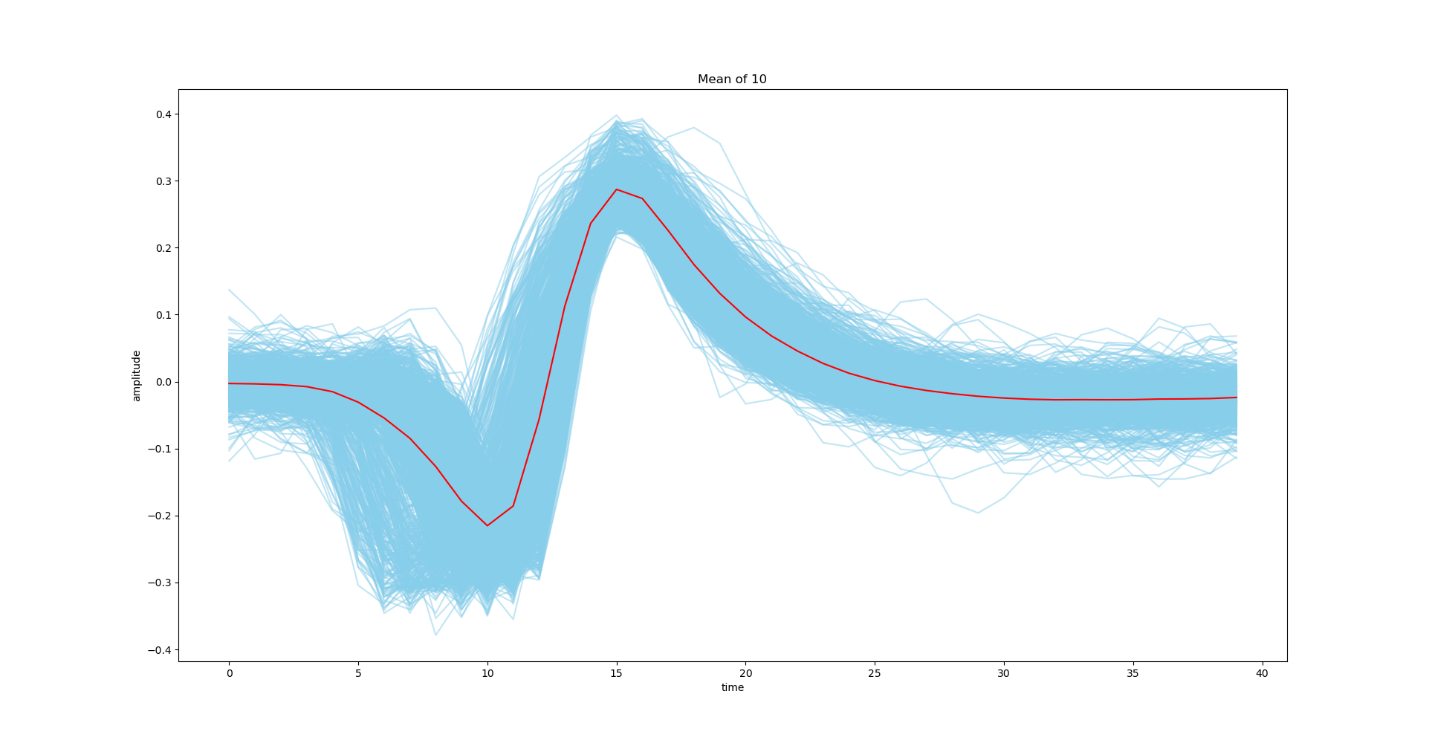
2/10/2021

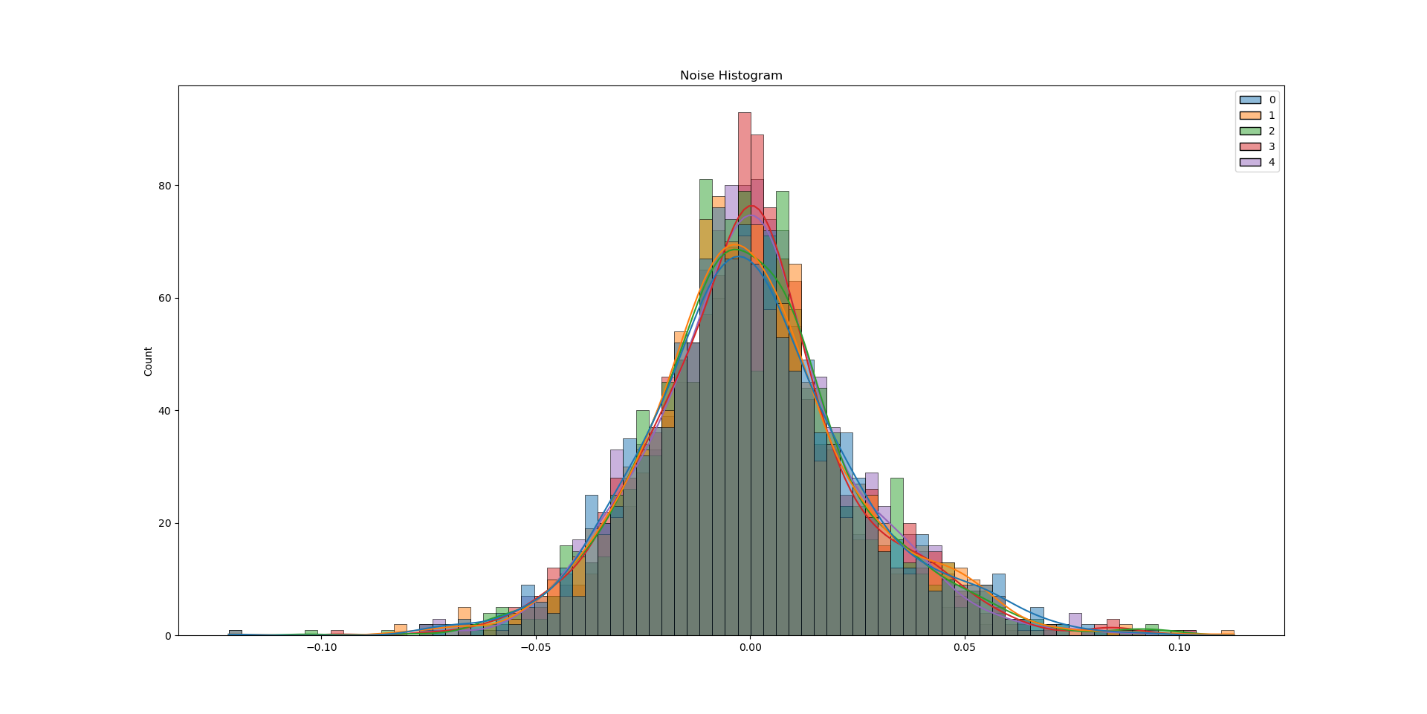
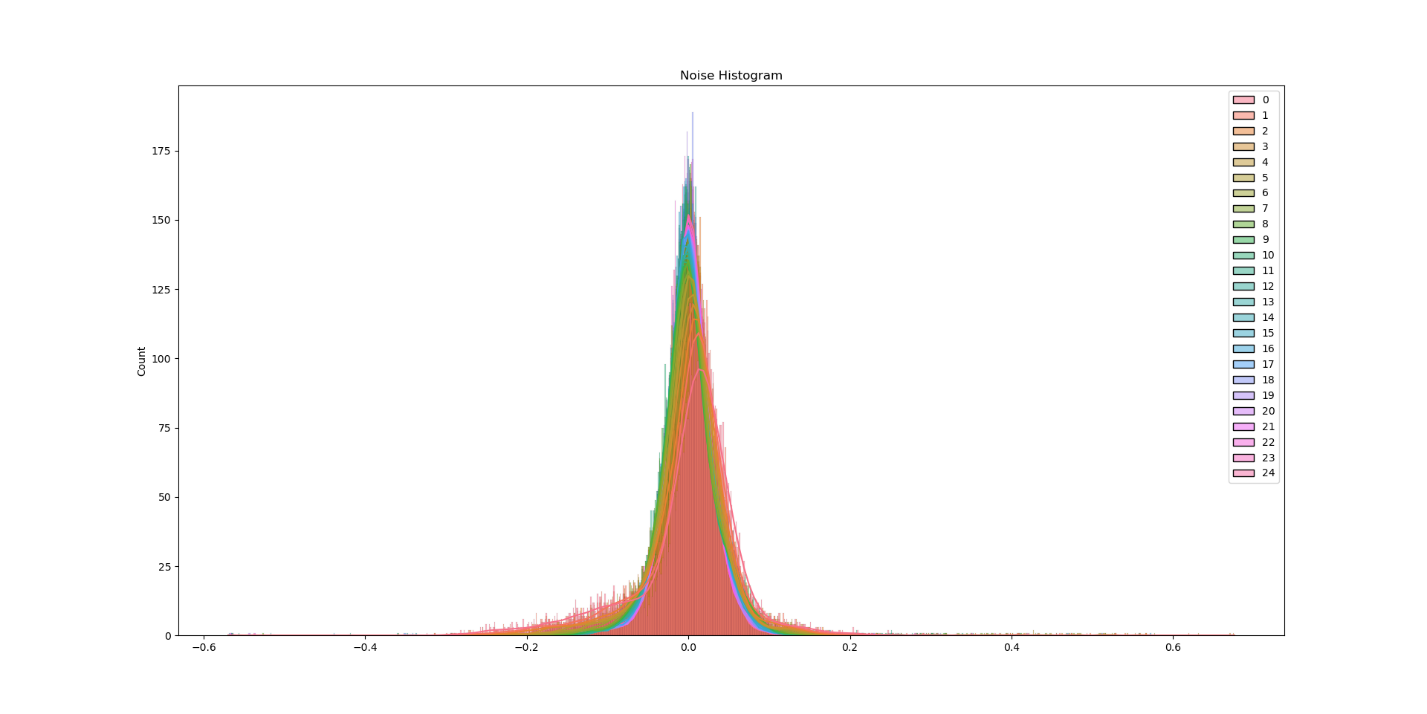
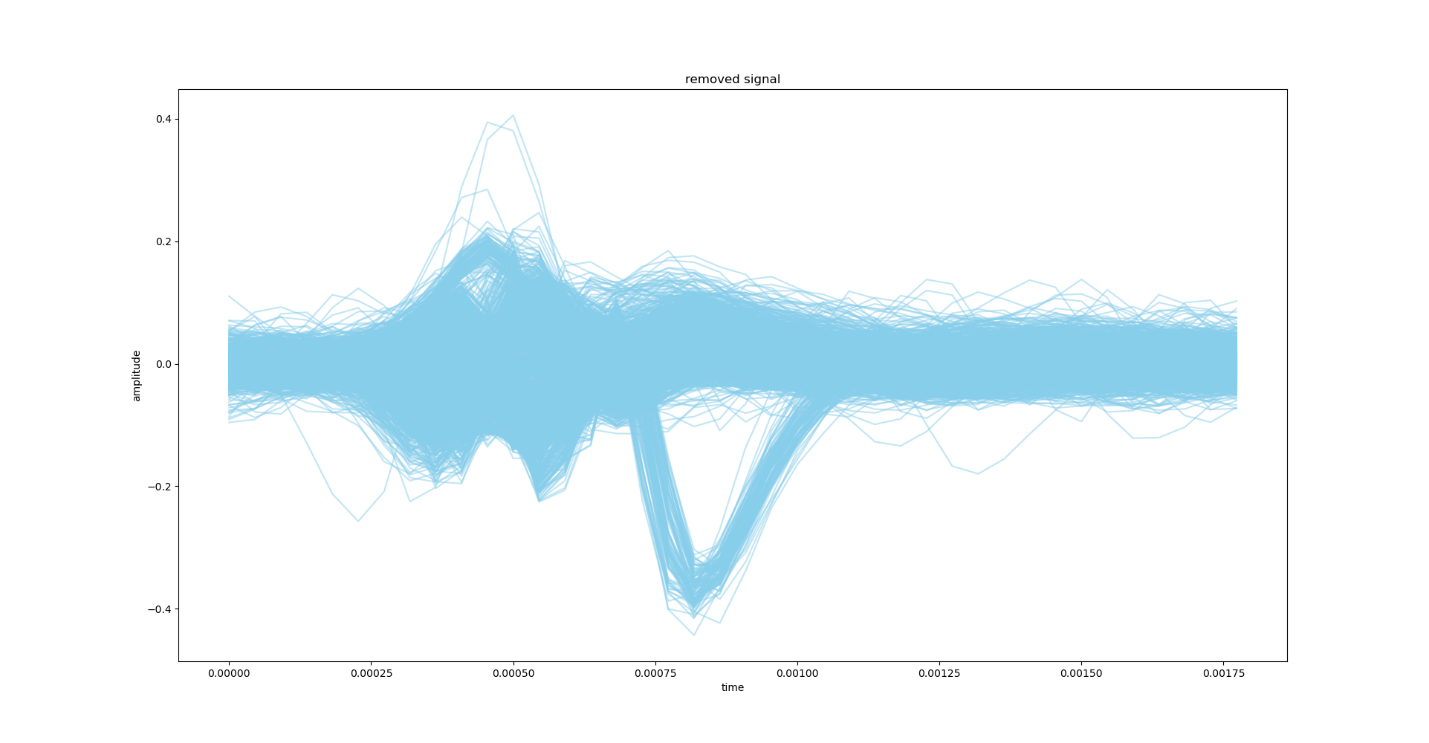
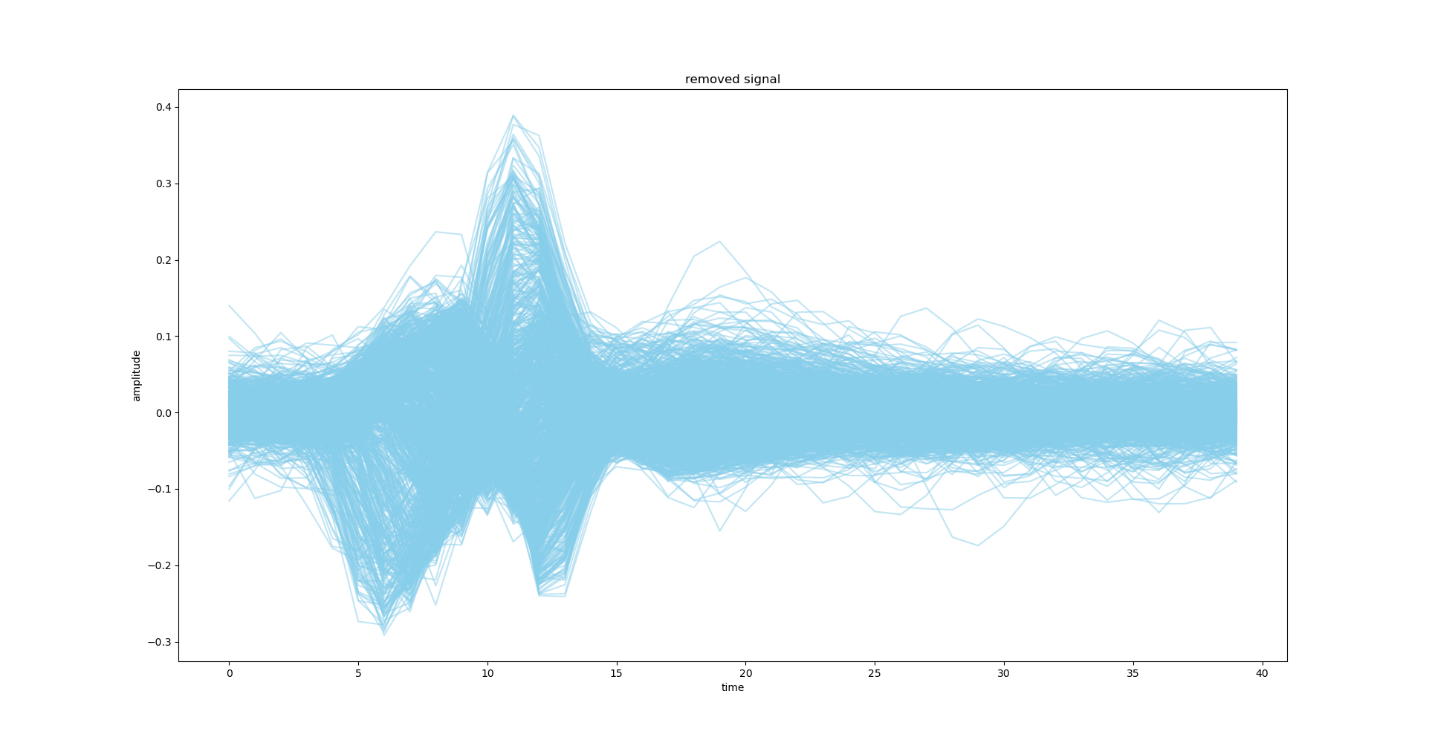
hw4

1. The initial heuristic approach I created was just my HW2 assignment from last term. We had the same assignment last term, with one waveform called Henderson. Luckily, I write all my code modularly, so it ended up being plug and play. I ended up removing unnecessary plots to make it run faster, but I didn’t have to change much more than that.

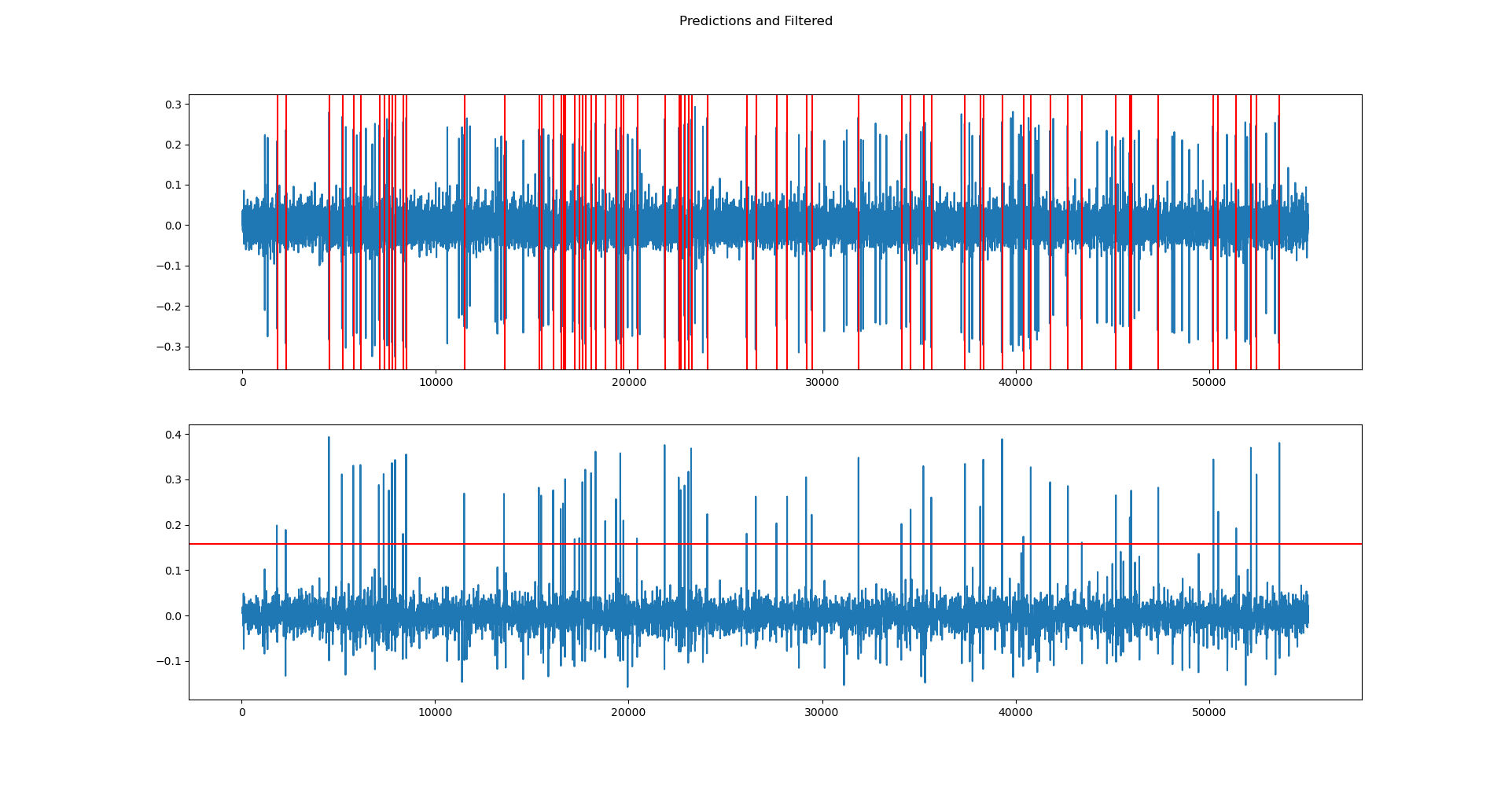
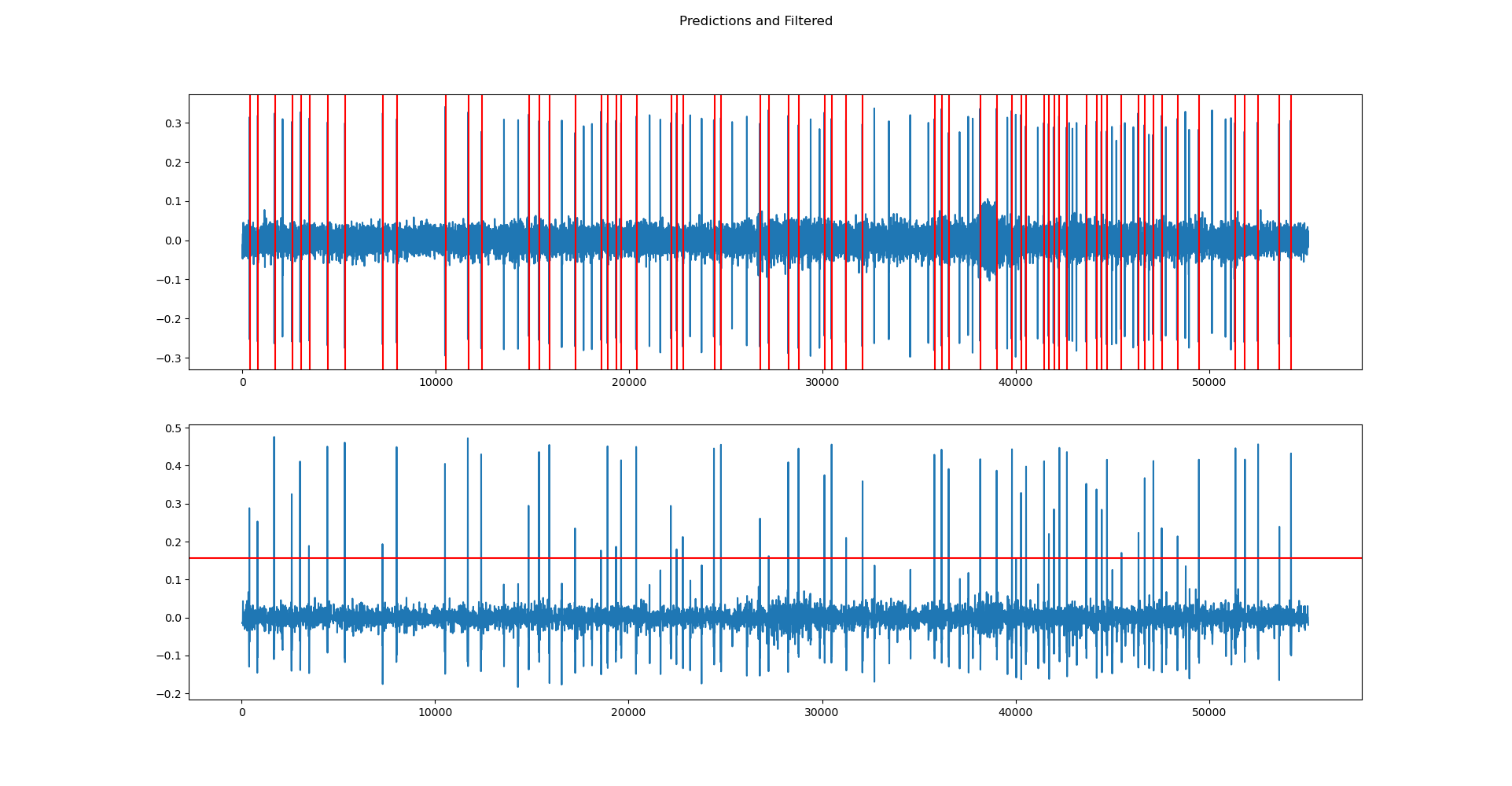
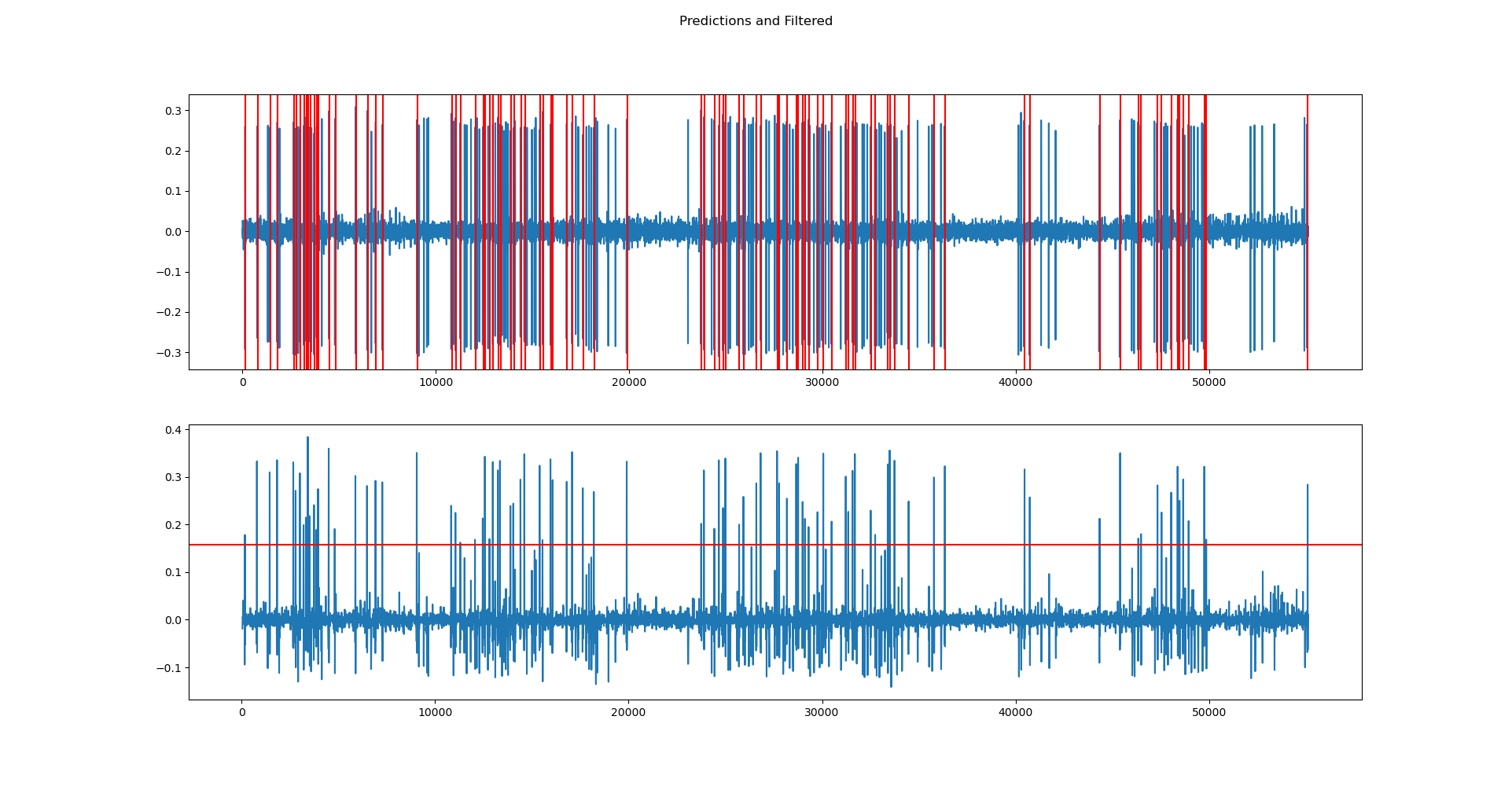
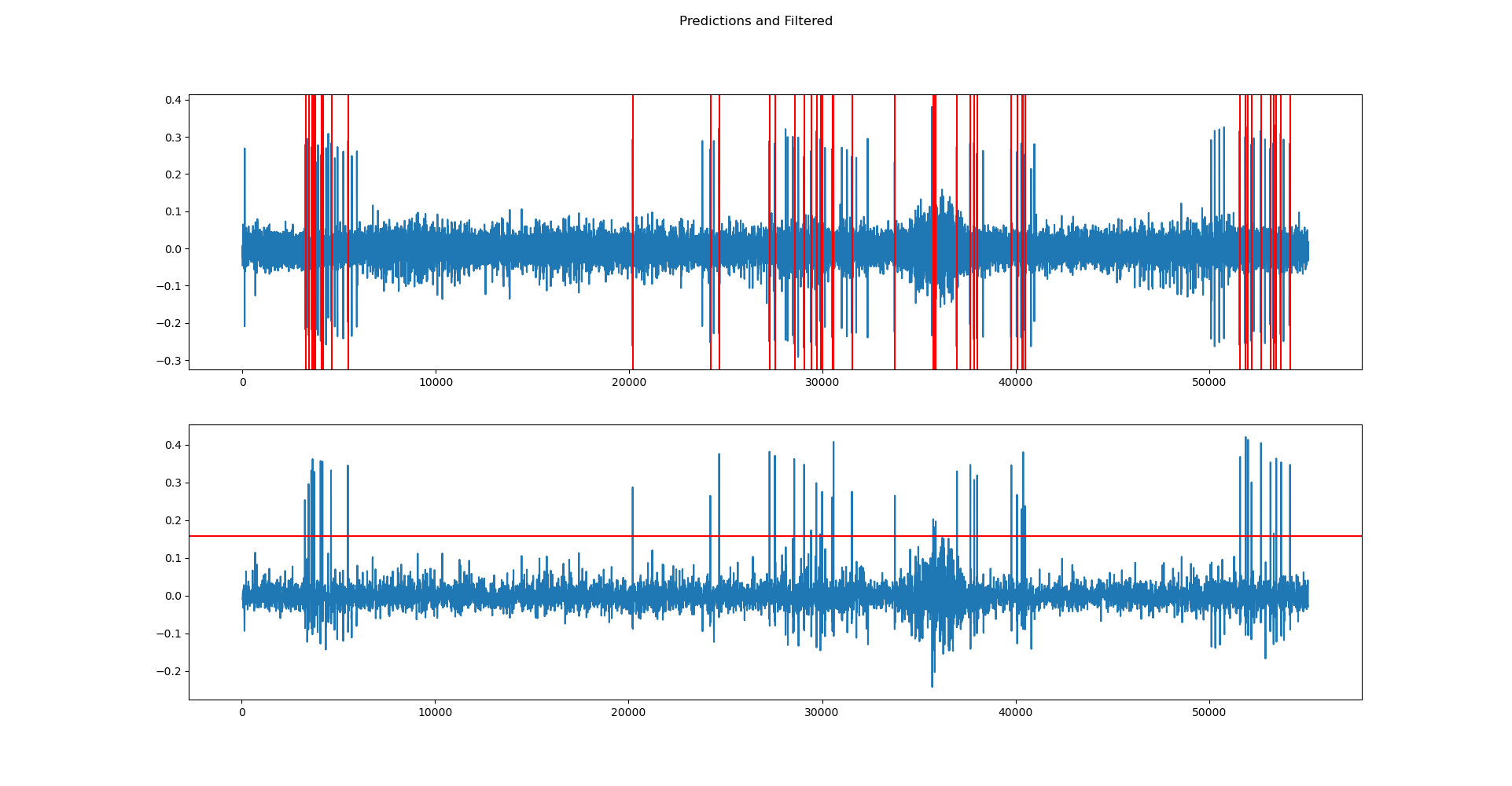
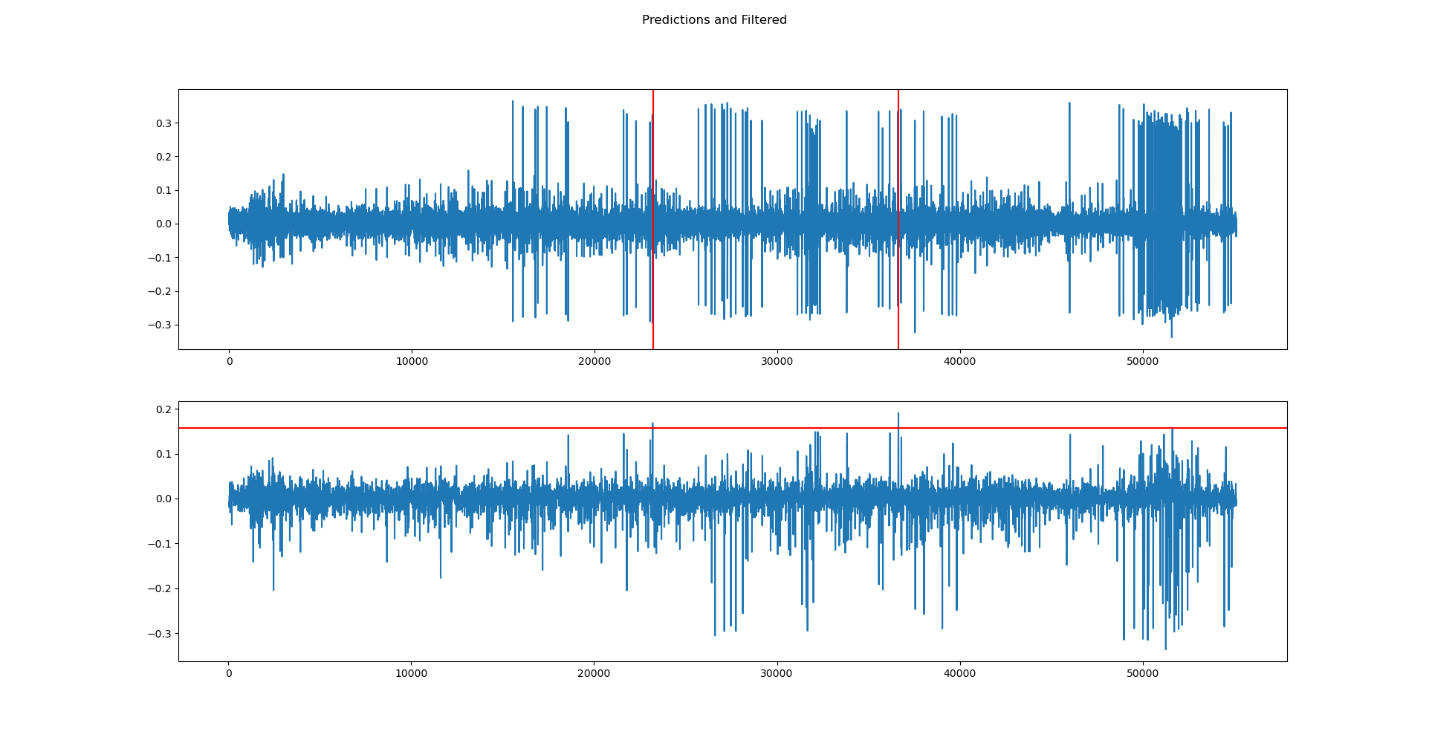
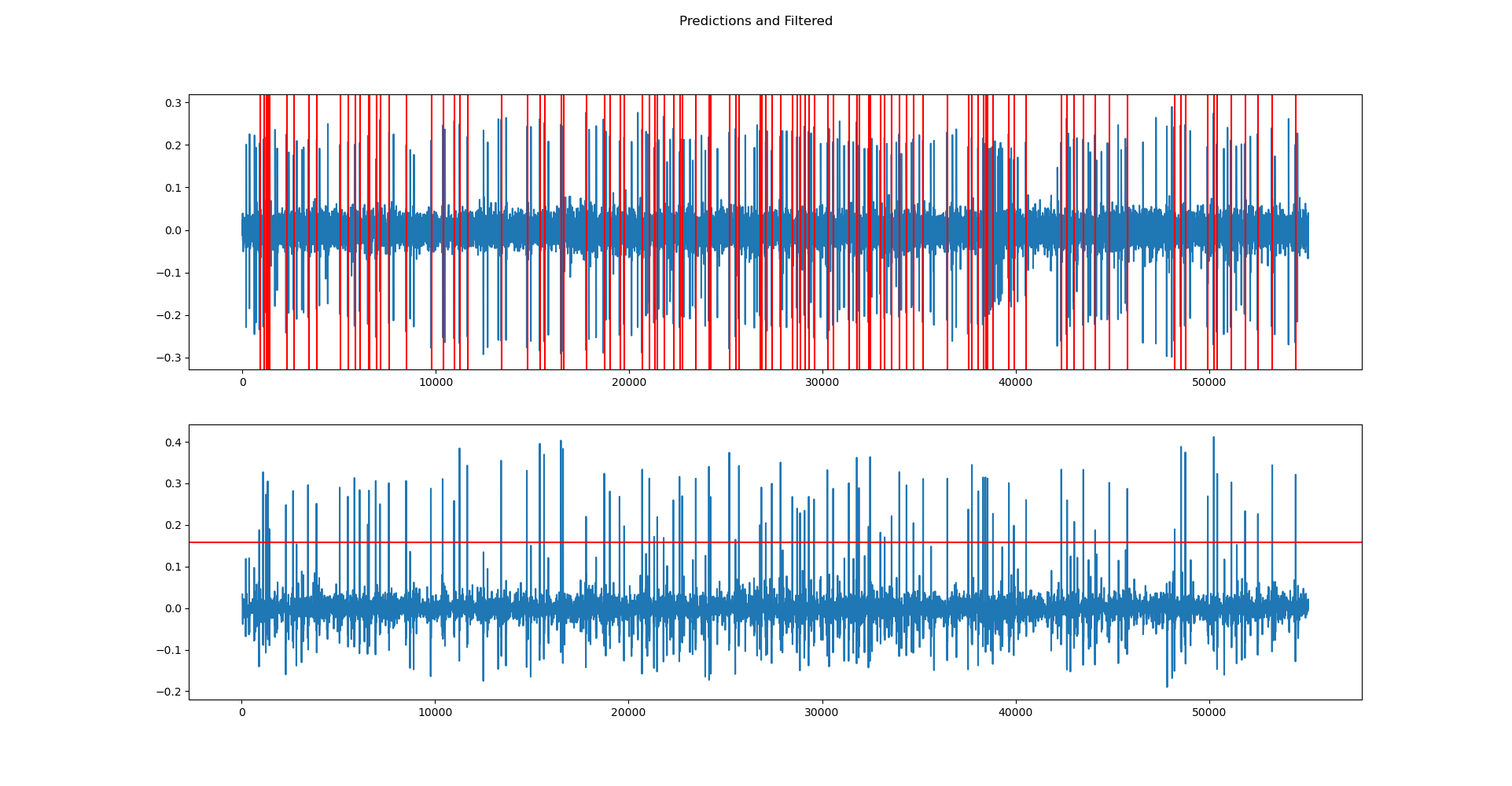
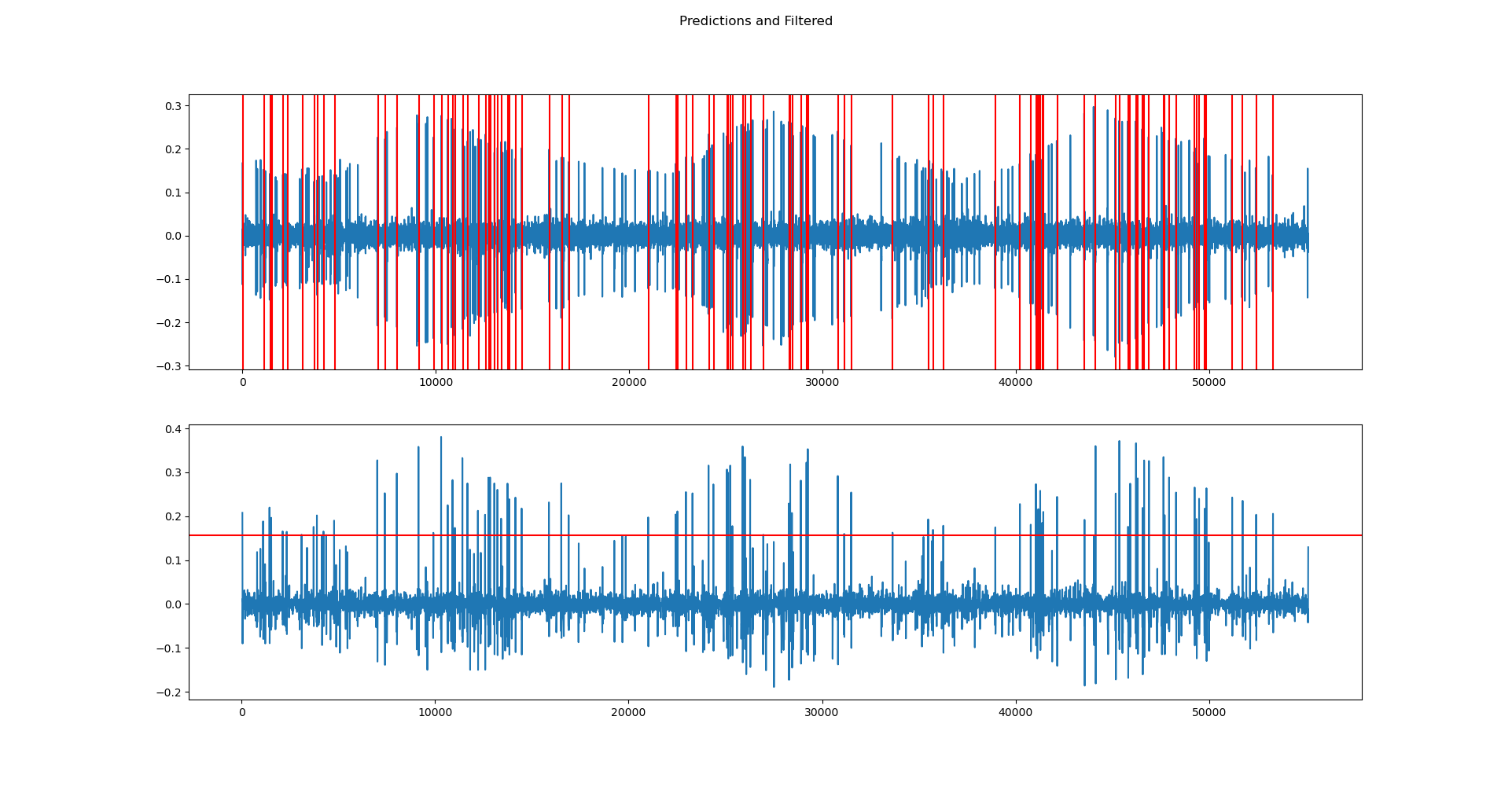
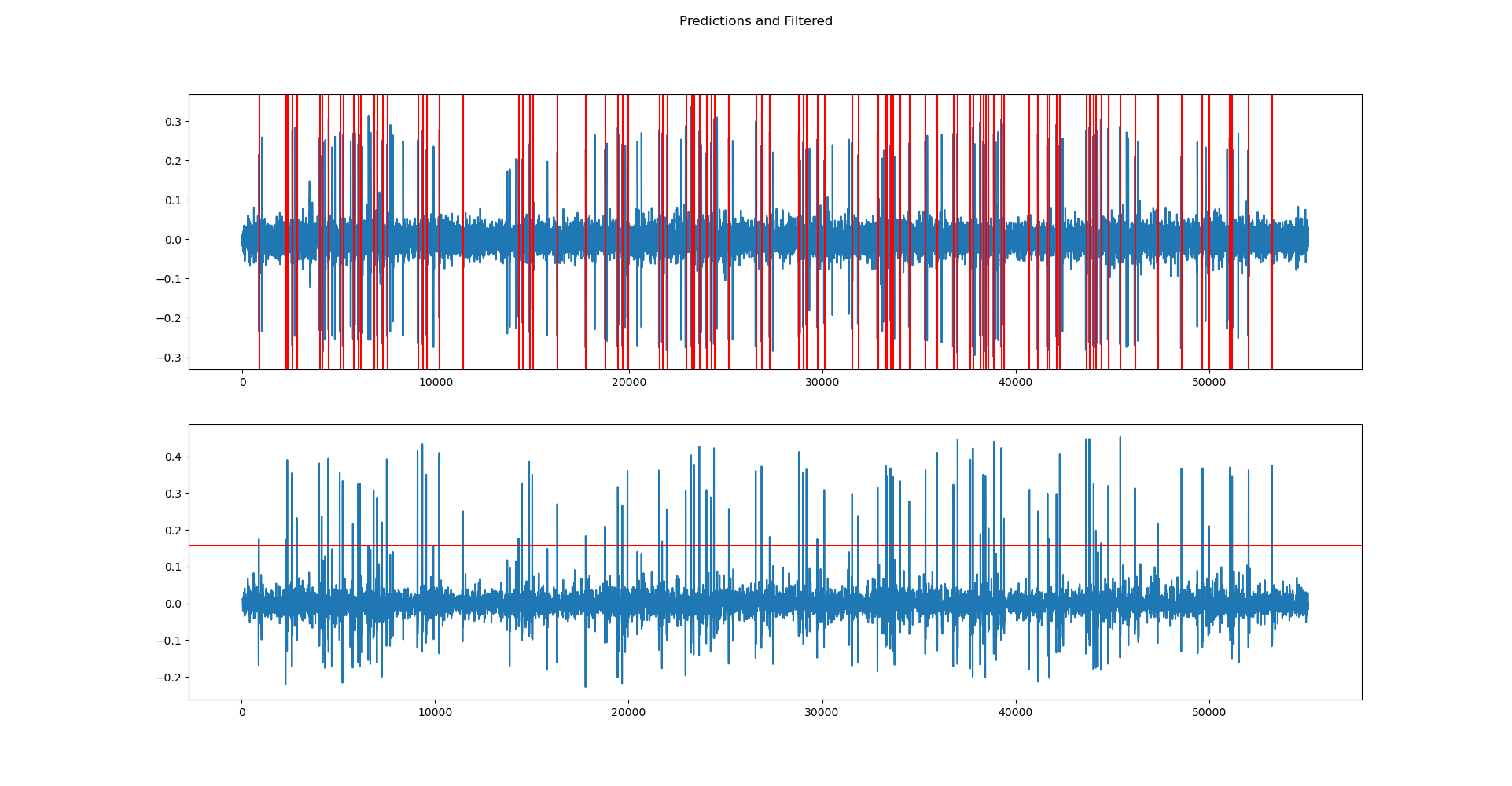
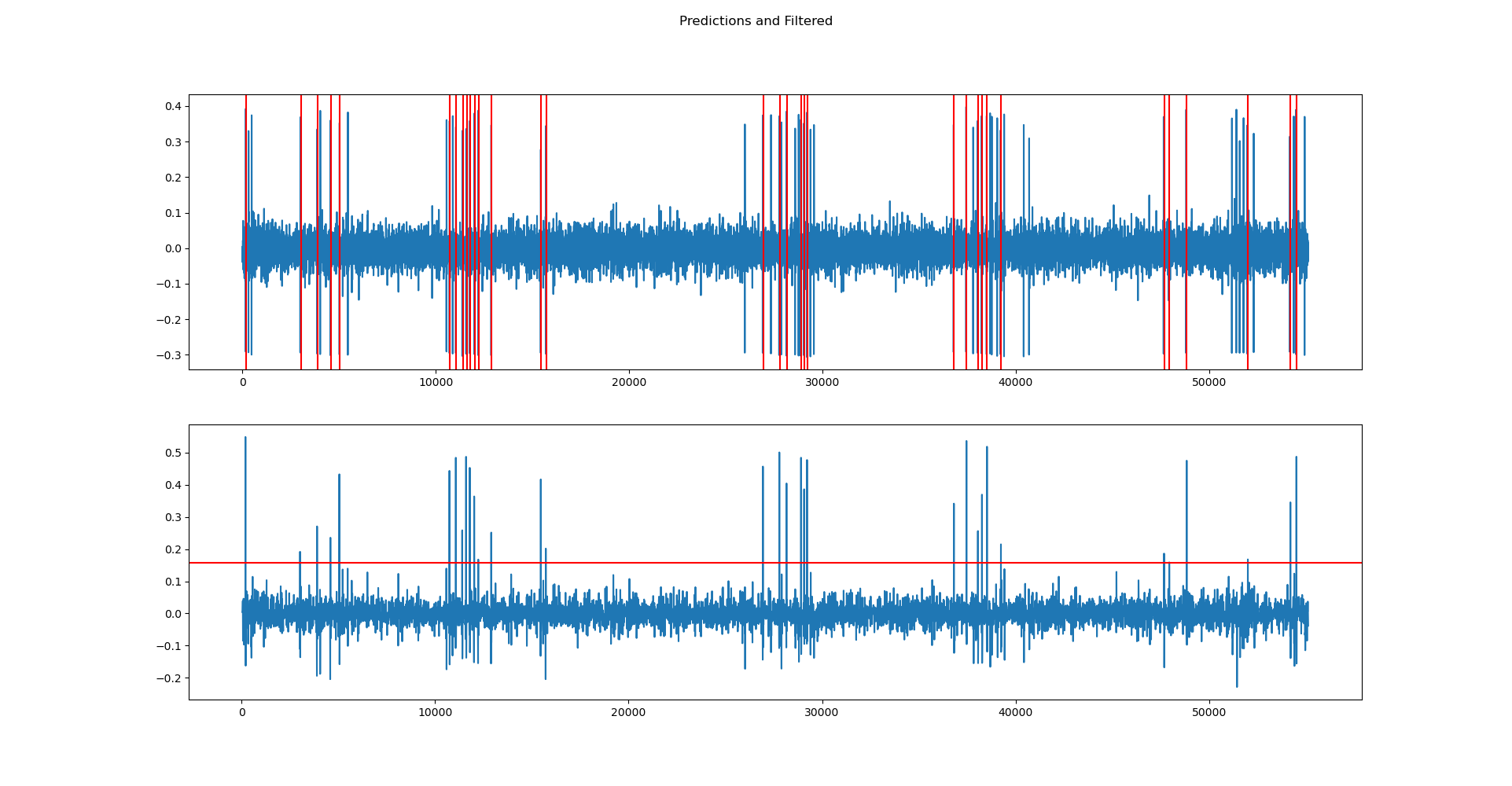
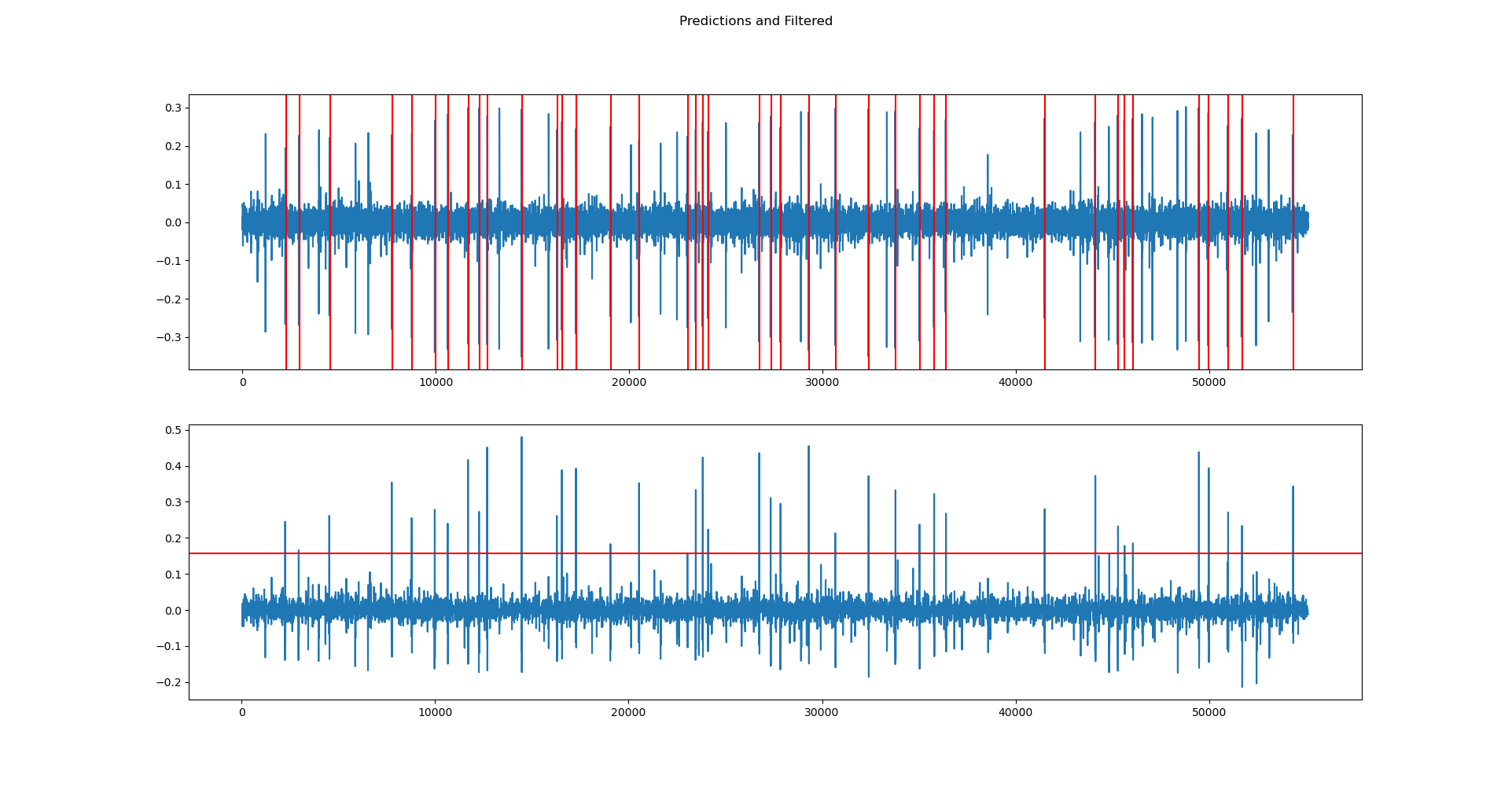
The old assignment from last term, was to find all the tall spikes in the waveform that were overactive parts of the patient’s brain. I accomplished this by using a low pass filter which put all the spikes on a straight line, and then detecting the large spikes above a threshold. One big change I made to the program was to the threshold. Rather than just being a fixed number that I enter in, I found maximum value in the wave, and made the threshold be anything at 60% of its height. That as the spikes change, the threshold would be relative to the individual.

1. 
2. Looking at this plot, we see that there’s some weird distortion between .75ms and 1ms. This was caused by waveform #5(MER0307). I assume this is from some kind of data misalignment.

This is an over lay plot with number 5 removed.

1. The duration of my sample mean is about 1.75 ms, or roughly 40 samples. I picked this number to get it close to the wave form, but not worry about cutting into my mean(in case I wanted to test this on some of the other waves). But made this work variably, so I could change the later if I want.
2. So, to find the standard deviation of the noise, looked at the part of the sample mean that didn’t have the signal(the flat patch between 1.25ms-1.75ms) and used the std function in NumPy, and then too a mean of the array that I got back. I chose this section because it was absent of the signal and had the highest likelihood of being just noise. I ended up getting ~0.024 I tested this by checking the same region in all of the datafiles, and I got approximately the same number, so I assume this was correct.
3. Yes, I think that the noise is pretty close. The shape looks Gaussian and the std looks like it’s in the right place at around ~0.024. I think this makes sense, because the mean should be at zero, and most of the noise visually is 0.024 away from the x axis. This is a histogram of all the noise in all signals.
4. There’s a lot of junk left over in my residuals. I think this has to do more with a misalignment. I have that really gross knee opposite the large opposite spike. I know it’s from recording 5, but I don’t actually know what it is beyond that. I tried removing the file, but I still have a lot of junk on the left. At this point I decided to continue with the program, and save my mean in a csv file, so that I can call it later, without having to recalculate every time. This also allows me to have multiple means and save them for later use. As before, this is my signal with removed number 5, this is what leads me to believe that some of my data points were misaligned. It looks like another wav that was too close.
5. I think it’s reasonable to assume the noise is zero mean gaussian with a ~0.024 std throughout. As stated before, I looked through all of the files to see what I would get, and that’s what I came up with.
6. So the threshold that I chose was based on the Eta variable in Maximum Likelihood. I liked it the most because it moves with the standard deviation and finds the midpoint of the mass of the wave form. I think it seems like the most reasonable threshold to start with. That being said, the gamma in the equation is the log of Theta(the threshold), and I needed to choose my variables. I left my costs pretty basic to start, that way I could see where it lands…. But for my prior probabilities, I tried to figure out what would be best. I decided that I should see how many spikes there were on average from my heuristic. I was going to try with just the annotation data…. But that felt like cheating. So, I used my heuristic, and I found that in the first 10 recordings, there’s an average of about 125 large spikes per recording.

This also led to other questions. I knew for my program I was going to use a sliding window, so I chose my window to be the size of my sample mean 1.75ms, with a 75% overlap(so 40 points, and it shifts over by 10 through each iteration). With 55125 points in the data divided by 10, I look in around 5512 regions, and 125 of those have signals, so my prior probability of detecting a signal is 0.0226(125/55125), and my probability of not finding the signal is ~.9774.

1. For a lot of these, it worked pretty well, except for number 5, which (surprise surprise) was the signal that was creating that gross distortion. For most, it was guessing too much which I could adjust with the threshold, but when I would change my costs to compensate for one of the waveforms, I would greatly mess up number 5, and wouldn’t detect anything.
2. My acceptance interval ended up being the length of my mean. I thought this would keep it simple, and I was hoping that this would help to match exactly what I was looking for. I did notice, that some of the patients, whom had a lot of activity where signals were very close to each other would have many more detections, and this may be partially because I detected more times and got more false positives.

12.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Matched | Sensitivity | Specificity | NPV | PPV |
|  | 1 | 0.9979 | 0.3443 | 0.6557 |
|  | 0.9697 | 0.9973 | 0.4576 | 0.5424 |
|  | 1 | 0.9948 | 0.3537 | 0.6463 |
|  | 1 | 0.9884 | 0.5442 | 0.4558 |
|  | 1 | 0.9902 | 0.4901 | 0.5099 |
|  | 0 | 0.9902 | 1 | 0 |
|  | 0.9348 | 0.9967 | 0.4342 | 0.5658 |
|  | 1 | 0.9906 | 0.5053 | 0.4947 |
|  | 1 | 0.9962 | 0.3725 | 0.6275 |
|  | 1 | 0.9932 | 0.4964 | 0.5036 |
|  | 0.9926 | 0.9859 | 0.5162 | 0.4838 |
|  | 1 | 0.9891 | 0.8209 | 0.1791 |
|  | 0.9913 | 0.9961 | 0.2549 | 0.7451 |
|  | 1 | 0.9985 | 0.2727 | 0.7273 |
|  | 0.9701 | 0.9958 | 0.3925 | 0.6075 |
|  | 0.9857 | 0.9977 | 0.25 | 0.75 |
|  | 0.9412 | 0.9988 | 0.2727 | 0.7273 |
|  | 0.9881 | 0.9918 | 0.5 | 0.5 |
|  | 0.9714 | 0.9992 | 0.1905 | 0.8095 |
|  | 0.9231 | 0.9987 | 0.2653 | 0.7347 |
|  | 1 | 0.997 | 0.411 | 0.589 |
|  | 1 | 0.9971 | 0.4085 | 0.5915 |
|  | 1 | 0.9915 | 0.4943 | 0.5057 |
|  | 1 | 0.99 | 0.6645 | 0.3355 |
|  | 1 | 0.9978 | 0.6667 | 0.3333 |
|  | 0.9925 | 0.9912 | 0.4027 | 0.5973 |
|  | 0.8889 | 0.9966 | 0.8095 | 0.1905 |
|  | 0.3333 | 0.964 | 0.992 | 0.008 |
|  | 0.9091 | 0.996 | 0.6667 | 0.3333 |
|  | 1 | 0.9884 | 0.3343 | 0.6657 |
|  | 1 | 0.9959 | 0.3306 | 0.6694 |
|  | 1 | 0.9922 | 0.8404 | 0.1596 |
|  | 1 | 0.9959 | 0.3661 | 0.6339 |
|  | 1 | 0.9929 | 0.4965 | 0.5035 |
|  | 1 | 0.99 | 0.4449 | 0.5551 |
|  | 1 | 0.9839 | 0.6082 | 0.3918 |
|  | 1 | 0.9842 | 0.5882 | 0.4118 |
|  | 0.9125 | 0.9961 | 0.3482 | 0.6518 |
|  | 1 | 0.9972 | 0.4242 | 0.5758 |
|  | 1 | 0.9958 | 0.4615 | 0.5385 |
|  | 1 | 0.998 | 0.3571 | 0.6429 |
|  | 0.8345 | 0.9984 | 0.1168 | 0.8832 |
|  | 1 | 0.9967 | 0.4231 | 0.5769 |
|  | 1 | 0.9951 | 0.3657 | 0.6343 |
|  | 1 | 0.9988 | 0.5217 | 0.4783 |
|  | 0.3636 | 0.9963 | 0.6981 | 0.3019 |
|  | 1 | 0.9972 | 0.3784 | 0.6216 |
|  | 0.9565 | 0.9957 | 0.4943 | 0.5057 |
|  | 0.6566 | 0.9998 | 0.0299 | 0.9701 |
| Ave | 0.91031 | 0.973936 | 0.457562 | 0.522438 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Heuristic | 0 | 0 | 0 | 0 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 0.9944 | 0.2605 | 0.7395 |
|  | 1 | 0.9999 | 0.005 | 0.995 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9928 | 1 | 0 | 1 |
|  | 1 | 0.9999 | 0.0036 | 0.9964 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9942 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9897 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9934 | 0.9999 | 0.0066 | 0.9934 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9955 | 1 | 0 | 1 |
|  | 0.9762 | 0.9999 | 0.0238 | 0.9762 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9677 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9826 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 0.9928 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 0.9997 | 0.0448 | 0.9552 |
| Ave | 0.977698 | 0.979874 | 0.006886 | 0.973114 |

13. I think that it’s feasible to use the matched filter. I think that my heuristic method, was flawed at returning the correct signals. I think that my signals ended up being misaligned, but if I had spent more time correcting this failure, and having completely aligned signals, I would have had better results.

14. From this homework I learned a lot. After seeing a lot of this used last time, and getting more pointer during the homework presentation, I think I had a better idea how to do this this time around. This homework taught me how much actually goes into making the right matched filter. There’s a lot of preprocessing that needs to be done, and if automated can make things go faster. However, with automation, there’s extra problems that can arise like misaligned data.

15. I’d like to know more about implementation. Last term, you had code samples that you showed us, and I think code samples for just the basic things like implementing likelihood ratio test would be very helpful.