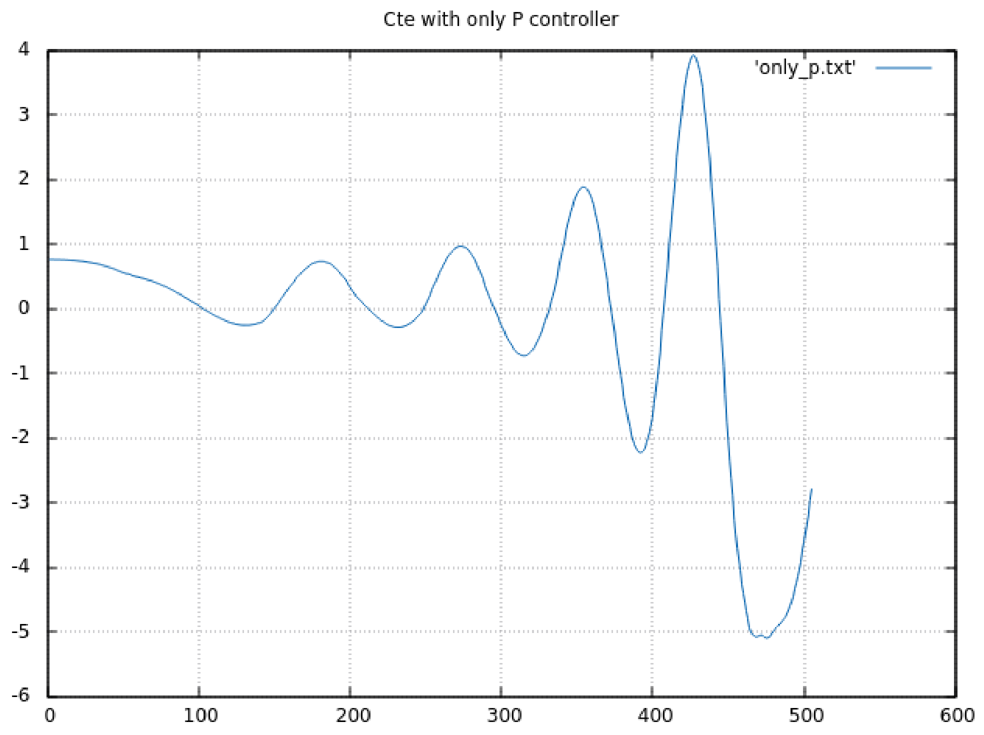
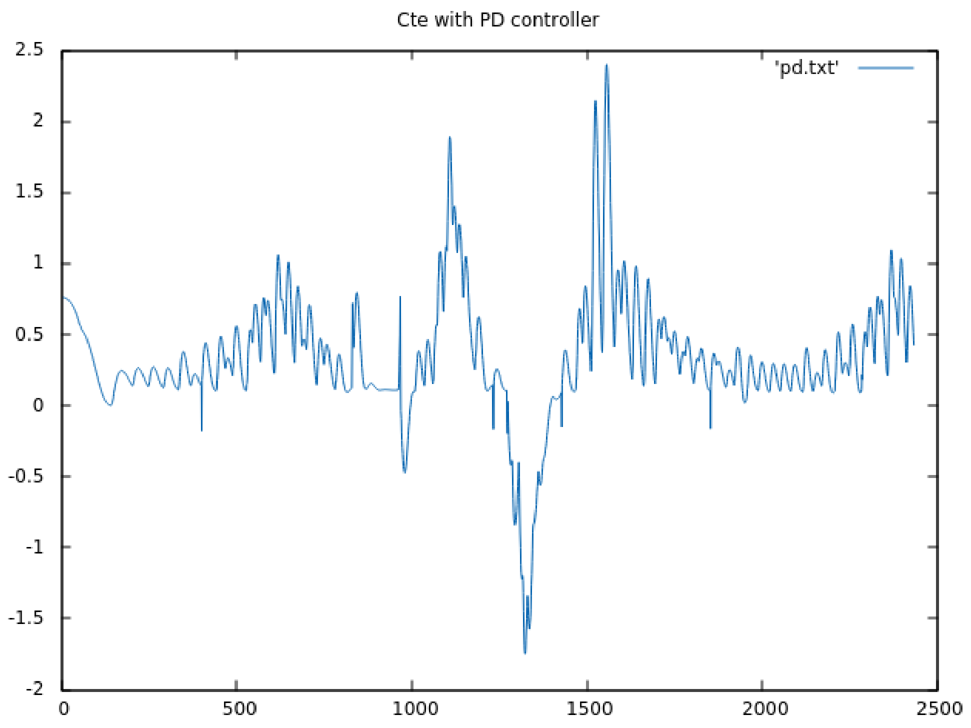
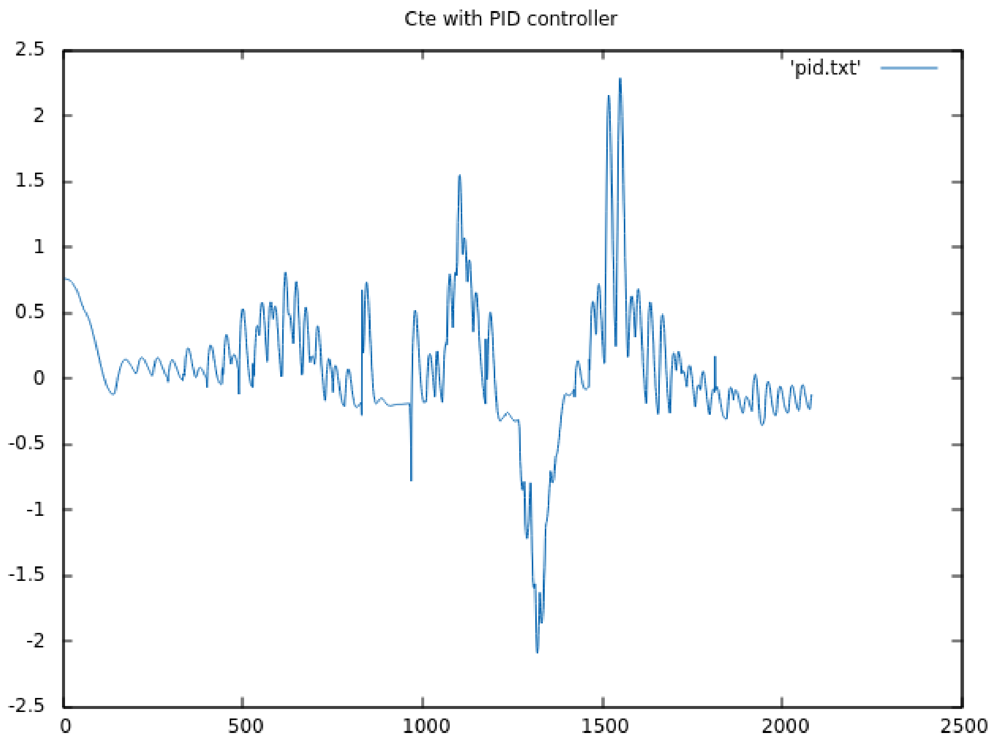
PID Controller Project

1. Effect of the P, I, D components had in the implementation
The P component mainly works on bringing the car back to the center of the lane, the bigger the P component, the faster it will bring the car back to the center of the lane. However, the bigger the P component, the more over shot will happen. And in the project, if use only the P component, with constant throttle, the can will leave the track very soon due to over shot. The following figure shows the case when we use only the P component, we can see that the cte value gets larger and larger and overshoot happens.



The D component for derivate is the change in cte from previous value to the current value. If the car is moving away from the center line, the current cte is bigger than previous cte value, the d error has the same sign as the P error, and this will increase the steering value, so it will bring the car back to the center of the line. And if the car is moving towards the center, the current cte is smaller than the previous cte value, the d error has different sign as the P error, and this will smooth the steering angle. If the D coefficient is too small, the car will still oscillate and overshoot. If the D coefficient is too large, it will take a long time for the car to correct the offsets.





The I component is the integral sum of all cte up to that point. Therefore, if there are too many negative cte values, it will increase the steering value and cause the car to turn back towards the center of the line. As shown above, the up figure is cte value change when the I component is not present, and the bottom figure shows the case when I component is present. We can see the cte with I component is shift down a bit towards to y = 0 line. When the coefficient for I component is too low, the car is tend to drift to one side of the lane or other for long time. And when the coefficient for I component is too high, the car tends to have quicker oscillations and not tend to get up to a quick speed.

2. How the final hyper-parameters were chosen.

I have implemented twiddle algorithm in the PID to help for hyper parameter tuning. However, for the environment like this project, the twiddle needs to run for a long time. Because for each new parameter, we need to let the car to run through the track for at least one lap to get a comparable error value. Finally, I decided to tune those hyper parameters manually.

First, I used only the PID for steering, and fix the throttle to be 0.3. After some tuning, I got the following values tau_p = 0.16, tau_I = 0.0003, tau_d = 3.0.

After that, I added the PID for throttle. The idea to use the PID for throttle is to modify the speed based on PID error value. When error is big, we should reduce the throttle and when error is small, we should increase the throttle. In the implementation, I used $0.9*(1.0-|PID_{value}|)$ to set the throttle value and used the steering PID with the tuned parameters. After some tuning, I got the parameters for throttle PID as following: tau_P = 1.0, tau_I = 0.0001, and tau_D = 25.0. The big D component make the throttle react very fast to the error, so the car can break in case of big cte value. By applying those hyper parameters mentioned above, the car can run through the whole track without leaving the boundary of the lane.