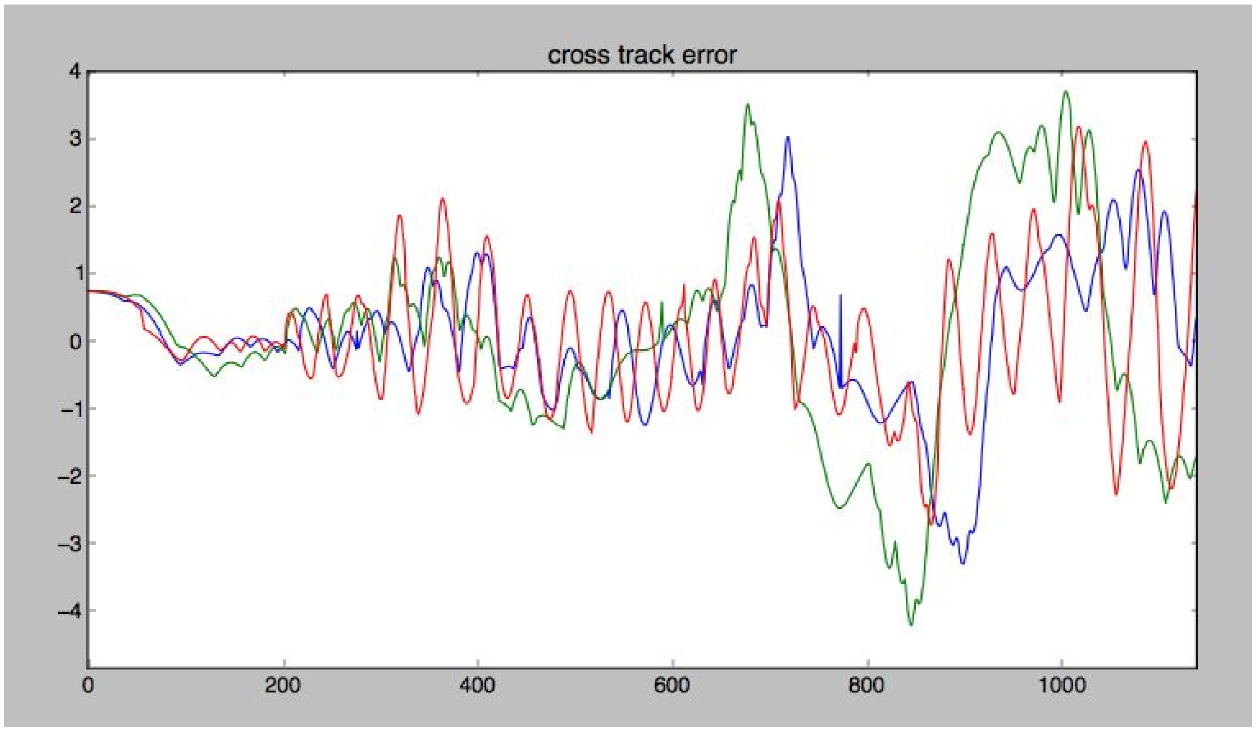
Project 4 PID Controller Writeup

Udacity Self-Driving Car Nanodegree Program, Term 2
Gregory Katz

Describe the effect each of the P, I, D components had in your implementation.

The primary effect of the P component is to steer the vehicle toward the center of the road. By itself, however, it causes an oscillatory behavior as it doesn't drive the cross track velocity to 0 at the same time as driving the cross track position to 0.

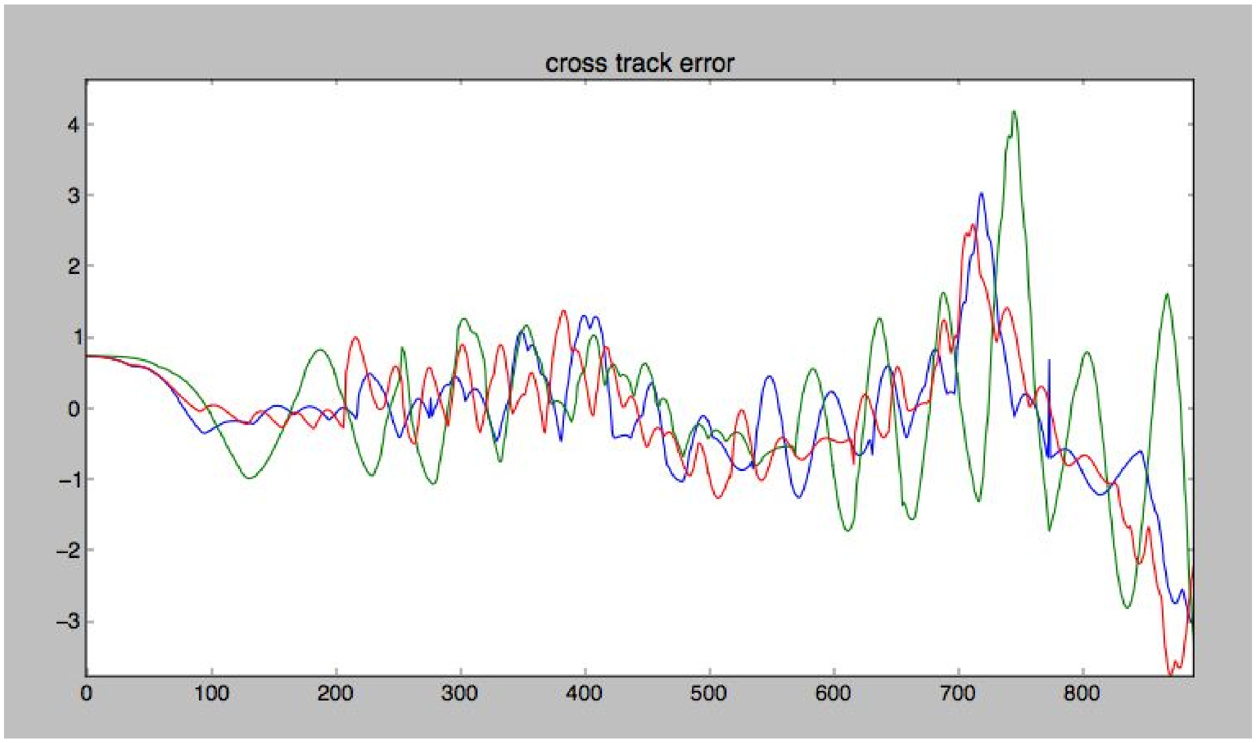
The following plot shows the effect on cross track error of increasing the P component. The green trace has low gain and it is slower to reduce the error to 0 in the beginning and has a higher peak on the large turns. The red trace has high gain and tends to oscillate too much but also performs better on the largest turns.



Green: Kp = 0.04, Ki = 0.001, Kd = 1.8
Blue: Kp = 0.08, Ki = 0.001, Kd = 1.8
Red: Kp = 0.16, Ki = 0.001, Kd = 1.8

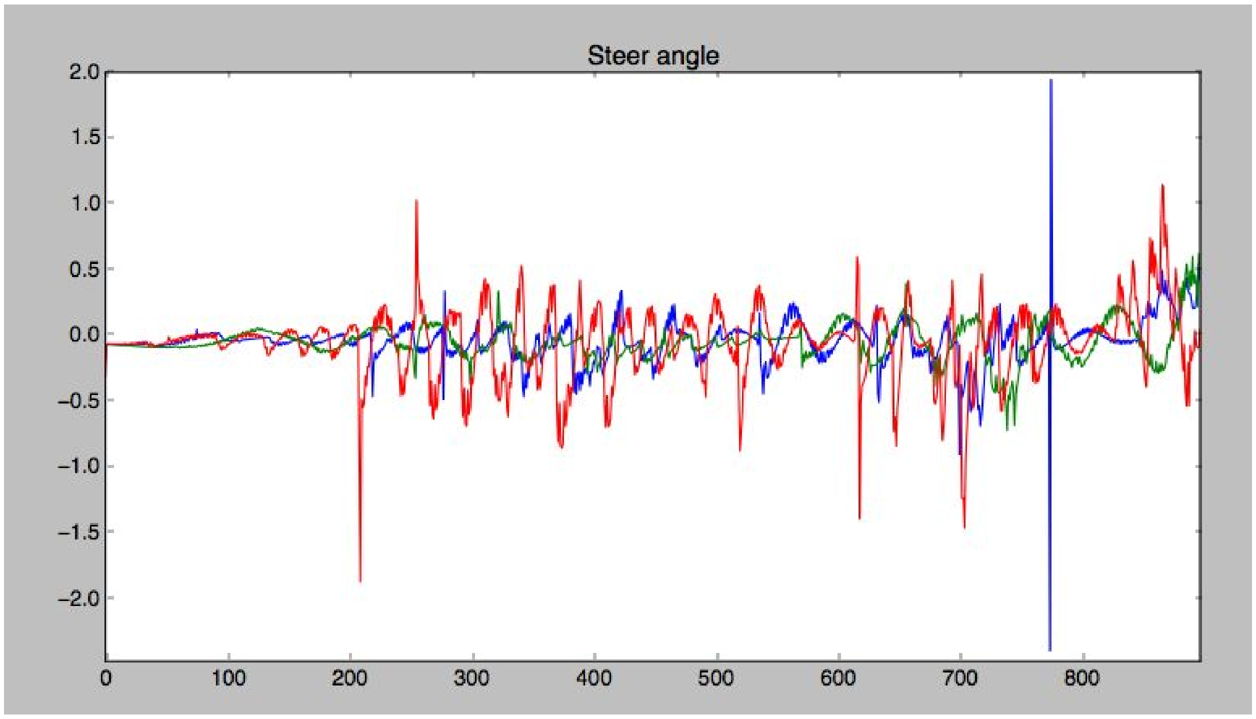
The primary effect of the I component is to reduce steady state cross track error to 0. In this implementation and simulation there doesn't seem to be any misalignment or disturbance that causes a steady state error, however since the track itself is not a linear input to the controller the I component can help keep the car closer to the center of the road during parts of the track with constant curvature. If the I gain is too high it destabilized the controller.

The primary effect of the D component is to reduce the overshoot and oscillation caused by the P component. The following plot shows the effect on cross track error of increasing the D component. Somewhat opposite to the P component the lower gain (green) is more oscillatory. In the plot it seems the higher gain (red) is generally better than the blue at keeping the cross track error close to 0, however the problem with high gain is that the vehicle behaves more jittery and the steering tends to saturate more. A moderate derivative gain balances between effectively reducing the large oscillations of the proportional gain while also keeping the steering input smooth and away from saturation. ›



Green: Kp = 0.08, Ki = 0.001, Kd = 0.9
Blue: Kp = 0.08, Ki = 0.001, Kd = 1.8
Red: Kp = 0.08, Ki = 0.001, Kd = 3.6

This plot shows the effect on steering angle of increasing the D component. Here you cansee that the steering angles are larger and change more quickly for the red trace with high derivative gain.



Green: Kp = 0.08, Ki = 0.001, Kd = 0.9
Blue: Kp = 0.08, Ki = 0.001, Kd = 1.8
Red: Kp = 0.08, Ki = 0.001, Kd = 3.6

I found that these components didn't always behave exactly as I expected. I think the main reason for this is that as the gain is increased the steering angle tended to saturate. It was especially difficult to keep the controller from oscillating at high speeds. I needed a high P gain to be able complete the turns without going off the track but to keep from oscillating I also needed a high D gain which tended to saturate the controller and cause bad results. I think the non-linear input that is the track itself was also causing some difficulty.

Describe how the final hyperparameters were chosen.

I used manual tuning to choose the gain values. In my process I ran a sweep of many Kp and Kd values leaving Ki at 0. For each run I had the cross track error print to a file so I could it with a python script to compare the performance of the gain parameters. I started my sweep with a large range of values and progressively narrowed it down with successive iterations of this process.

This method worked quite well at low speeds (throttle fixed at 0.3 or 0.4). As I tried to increase the throttle, I found that I needed to change the gains every time I increased the speed. At higher speeds gains that previous worked were unstable or not strong enough to complete the sharpest turns.

It is difficult to achieve high speeds with the basic PID control with constant throttle because it is hard to make the sharpest turns at high speed and because the gains perform differently at different speeds and with different radii of curvature. A more advanced algorithm would perhaps scale gains based on the speed and would brake before large turns and accelerate out of turns and straights. Additionally it would really help if the algorithm could look ahead somehow because the controller doesn't initiate the turn until the vehicle has built up some error from not turning early enough. Playing catch up makes it difficult to stay on the track and requires extreme steering at high speeds.

Ultimately, I was able to improve the speed up to 50-80 mph by implementing a simple throttle control law. In my implementation the throttle is proportional to the absolute value of the cross track error. This way the vehicle slows down when off the center and speeds up when on center. This allows for high speed on the straights without going way too fast on the corners. In addition I implemented a minimum speed below which the throttle is always max to keep the car from slowing down more than necessary. This method was a big improvement over constant throttle but was still sub-optimal because it does not look ahead. The throttle has a lot of delay and really needs to slow down before the turn not during the turn. I did not use braking, just reduced throttle, that could have improved performance.