Project Basics

In this C++ project, I use a Proportional-Integral-Derivative Controller, or PID for short, in order to drive a simulated car around a virtual track. The project involves implementing the controller primarily for the steering angle of the car (although I used the value from this controller to also determine throttle), as well as tuning coefficients for each PID value in order to calculate a steering angle that keeps the car on the track.

Project Steps

- Implement PID Controller for Steering (optional: controlling throttle as well)
- Optimize init parameters for each PID coefficient

Results / Reflection

Components of PID

The P, or "proportional" means that the car will steer in proportion to the cross-track error (CTE). The CTE is error between desired value and measured one. So in terms of simulator, CTE is essentially how far from the middle line of the road the car is. Too high P gain results in oscillations because output signal constantly overshoots and overcorrect the desired value. If P gain is too low, output signal can't reach desired value (car may react too slowly when the car reach the curve).

The "D" for derivate is the change in CTE from one value to the next. This means that if the derivative is quickly changing, the car will correct itself by increasing steering angle. Also "D" makes the car's steering angle smoothed, leading to a more smoother driving experience. But too high of "D" gain leads to almost constant steering angle. Too low of a "D" gain will lead to the car oscillations.

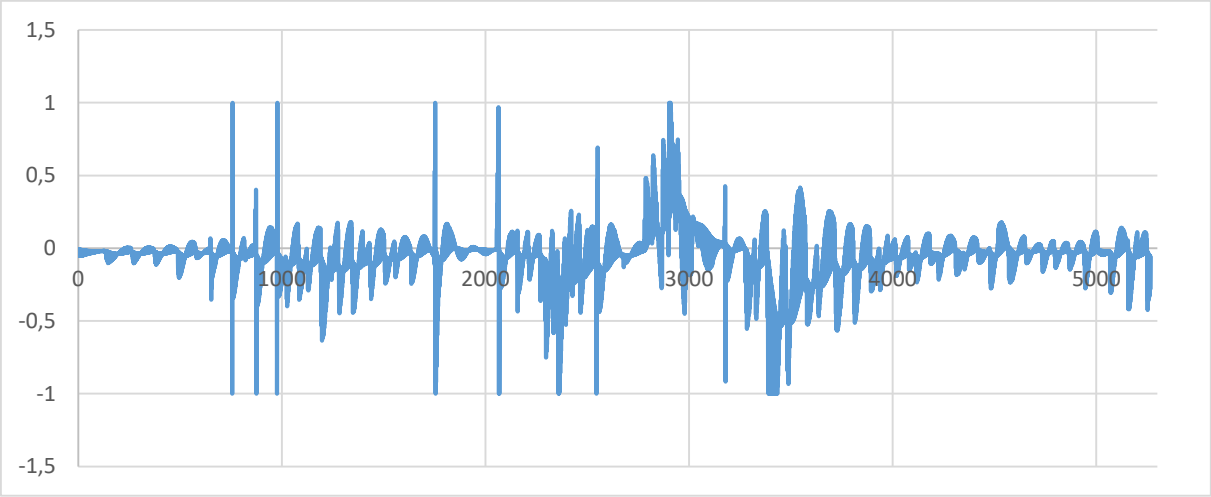
The "I" for integral sums up all CTEs up to that point, such that too many negative CTEs will drive up this value, causing the car to turn back toward the middle, preventing the car from driving on one side of the lane the whole time. If the coefficient is too high for "I", the car tends to have quicker oscillations. Too low "I" gain will cause the car to tend to drift.

How it was tuned

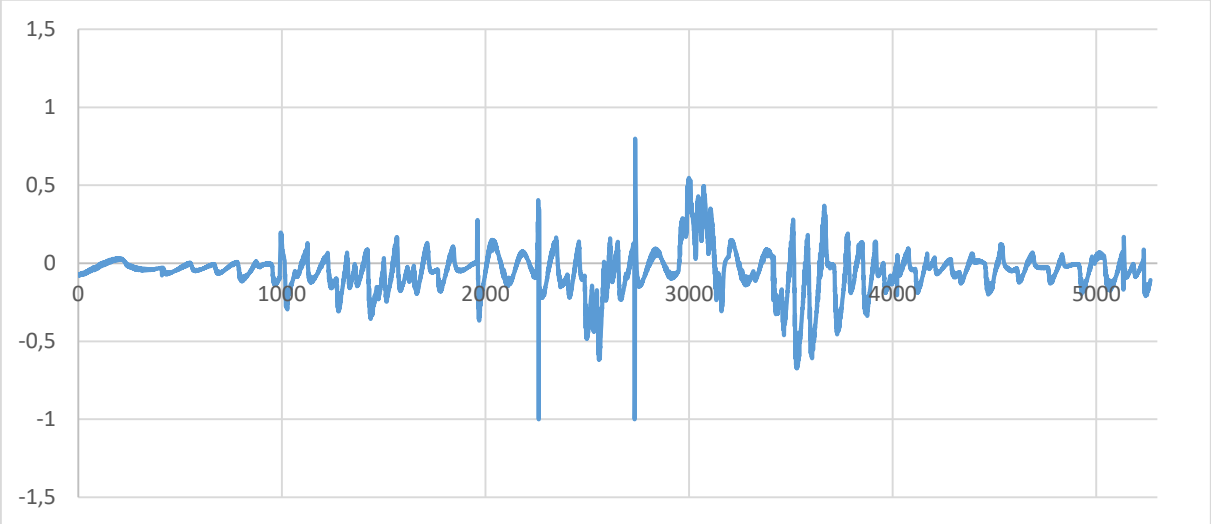
All coefficients were tuned manually. First of all I set "I" and "D" gain to zero and "P" gain to 10. When the car stops oscillating and reached first curve I set "D" gain also to 10. I repeated the process and I ended up with 0.1 for "P" and 6.0 for "D". At this point car could complete full lap but controlling signal (steering angle) was very jerky and car made sharp turns. I set "I" gain to very small value – 0.001 and started to fine-tuning. Final values are P – 0.105, I – 0.0002, D – 6.0.

Some addition features

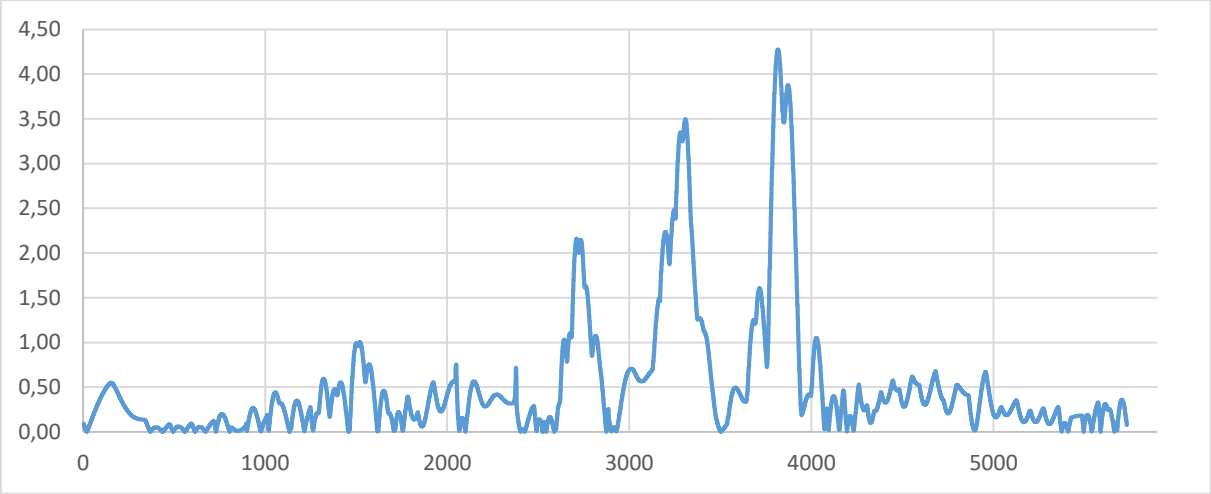
I noticed that with my final values output signal from PID controller is still in some situations very jerky (Pic. 1). So I filter steering signal with simple 1D Kalman filter (Pic. 2). With these simple feature I get more stable steering angle signal and also lowest overall CTE (sum of all absolute values of CTE). Without KF overall CTE equals to 2954.99 and with KF overall CTE equals to 2313.99 (Pic. 3)(Pic. 4). It means that with KF, car position was more equal to ground truth than without KF add-on. I also implement simple throttle control algorithm. The throttle value is dependent on steering value. If steering control signal is low than throttle value is high, and if steering control signal starts to blowing up car started to brake.



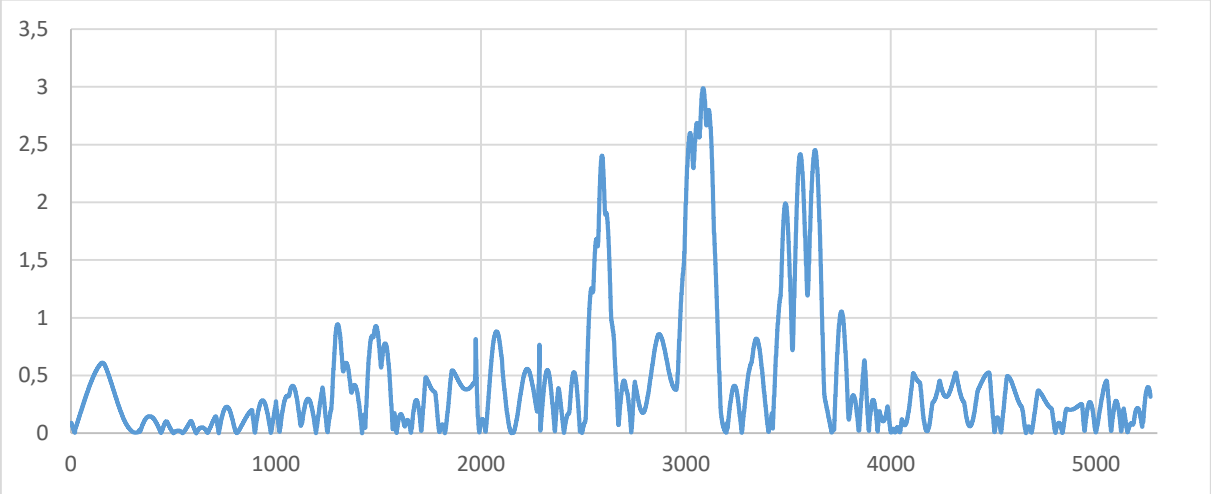
Pic. 1. Steering signal without Kalman filtering



Pic. 2. Steering signal with Kalman filtering



Pic.3. Absolute value of CTE without Kalman filtering



Pic.4. Absolute value of CTE with Kalman filtering