# Reflections

April 3, 2018

## 1 Self-driving Car Nanodegree

### 1.1 Project: PID Controller

#### 1.1.1 Reflections

Steering angle of the vehicle in the simulator is controlled using a PID controller, whose parameters are tuned using Twiddle algorithm discussed in the class. The input to the controller is "cte" value obtained from the simulator as the car moves on the track. This input is fed into the Twiddle algorithm in the form of consecutive batches of a fixed sample size.

Besides the steering algorithm, the throttle is also controlled using a quasi-proportional controller whose input is the "cte" value as well the steering angle. The parameters of this throttle controller are manually and crudely fixed.

Below are a few interesting observations regarding the tuning process:

**PID parameters - Kp, Ki, Kd** The twiddle algorithm in this case turns out to be very sensitive to the initialization of the search space, that is, the initial [dKp, dKi, dKd], as well as to the initial input to the vehicle in the simulator. This is probably because the batches for tuning the algorithm are themselves dependent on the initial conditions. Therefore, the initial choice of [dKp, dKi, dKd] and throttle should be reasonable to allow the vehicle to stay on the track during the initial tuning phase, and therefore, generate data for further tuning. "dKp = 0.1", "dKi = 0.0001" and "dKd = 1.0" is one such choice.

The integral parameter, Ki, is not particularly amenable to tuning in this project. In fact, it is not straightforward to define the integral error here. I tried two methods: - Once the "cti" value from simulator changes signs, I discard the previous sum of "cti" values and start summing anew until the next change of signs. - Alternatively, I heavily weigh the more recent "cti" values in the sum.

Even though in both the cases, the tuned controller had a reasonable performance, it had a somewhat delayed response. Since the simulator doesn't seem to have a system bias, I would prefer to ignore the integral error altogether. To take it into account, "integral_control" variable in the PID class should be set to "true".

**Hyperparameter "sample_size"** The size of the batch over which the algorithm observes the performance of a particular set of PID parameters turns out to be a crucial hyperparameter. If the batch size is too small, the average error generated over this batch may not be an accurate estimate of the performance of the PID parameters set. If the batch size is too large, sub-optimal set of parameters will control the car for too long a time and might end up derailing it. For this project, the batch size of 100 works well.

**Possible improvements**

- *Exploration-exploitation:* Once the controller has been tuned to the extent that it keeps the car on track below a certain speed threshold, further tuning can be done by collecting samples for the batch non-consecutively over many short random intervals. This would enable having a large batch size to accurately estimate the performance of a set of PID parameters, without the fear of having the car go off-track.
- *Performance evaluation under similar conditions:* If possible, performace of different sets of PID parameters should be compared over the same length of track, with the same initial conditions.