WriteUp

PID Tuning

In all following plots: left figure shows yaw error over time (in rad) and right shows speed error over time (in m/s).

The Kp value is responsible for reacting on big control error where the Ki is able to minimize steady state error and Kd reduces overshooting.
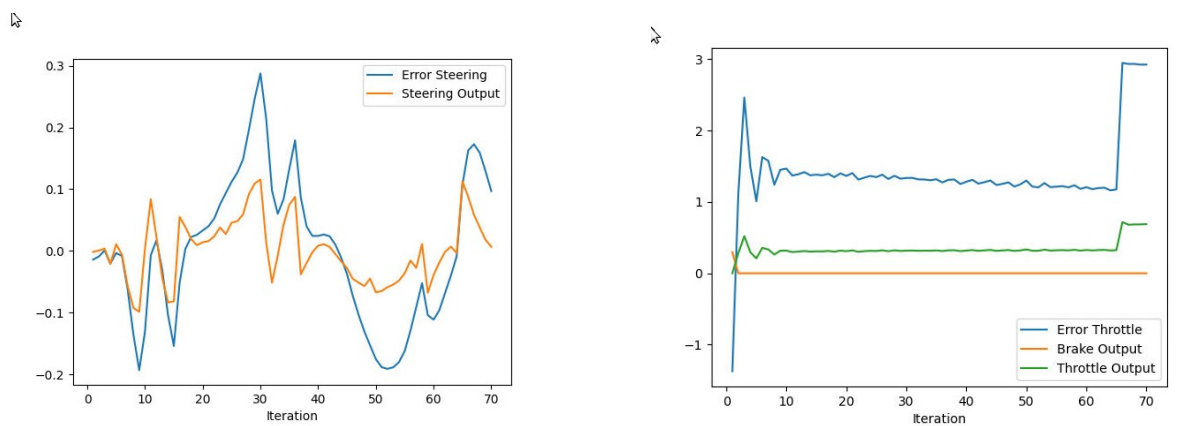
Run 1:
Starting parameters are inspired by: https://knowledge.udacity.com/questions/852715
pid_steer.Init(0.3f, 0.001f, 0.7f, 1.2f, -1.2f)
pid_throttle.Init(0.2f, 0.001f, 0.02f, 1.0f,-1.0f)

Note: Collision with 3rd car. Speed target not reached (Error Throttle >>0). Error steering reacts as expected to control input (gets smaller after steering is applied).
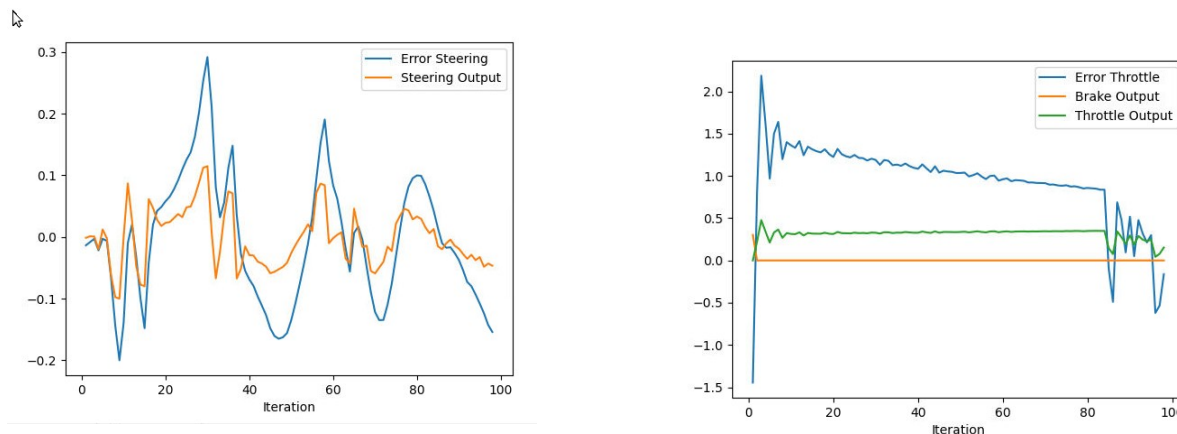

Run 2:
Increased throttle Kii to reach target speed faster
pid_steer.Init(0.3f, 0.001f, 0.7f, 1.2f, -1.2f)
pid_throttle.Init(0.2f, 0.002f, 0.02f, 1.0f,-1.0f)

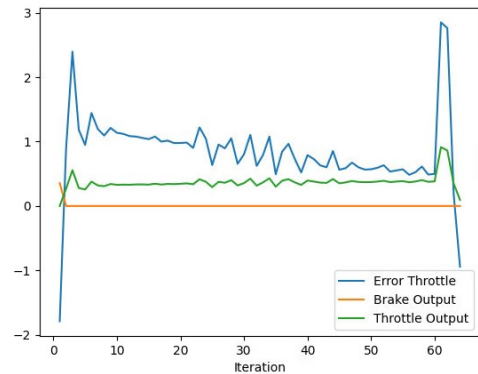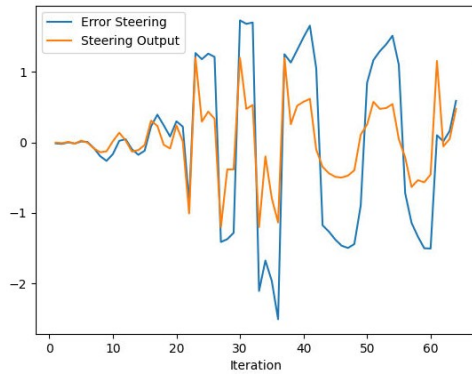Note: reached intersection without crash. Speed target not reached however error is smaller.

Run 3:
Increased throttle Kpi part to reach target speed faster
pid_steer.Init(0.3f, 0.001f, 0.7f, 1.2f, -1.2f)
pid_throttle.Init(0.2f, 0.005f, 0.02f, 1.0f,-1.0f)

Note: crashed into 3rd car and steering is very unstable. Target speed not reached.

As the 2nd run produced satisfying results I decided to stick with the obtained parameters.

Automatic Tuning of PID parameters

An automatic tuning could be performed by calculating the integral of the squared control error and using an algorithm (e.g. twiddle) to minimize it. To simplify this the control parameters should not be hard coded but passed as arguments to the binary.

Pros and Cons of PID controllers

Pros:

- easy to understand
- simple implementation
- low computational resources required

Cons:

- manual tuning needed
- only current state is used no looking ahead
- carrying over to other vehicles is not trivial, i.e. new tuning require

Ways to improve the controller

- Implement a feedforward controller for vehicle speed
- Implement gain scheduling (e.g. parameters become function of vehicle speed)
- Invest more effor to tune the parameters