Report of PID Conbtroller Project

Introduction

In this project, I used two PID controllers to control the vehicle, one for throtle and the other for steer angle. The “twiddle” algorithm was applied to update PID parameters in real time.

To simply the problem, only the PID controller of steer angle was able to *train* parameters. The parameters of throtle PID controller was manually tuned by trial and error.

Effects of the P, I, D coefficients

Generally speaking, the proportional gain detemines the “strength” to pull the control variable back to the desired value. A greater proportional gain will make the control variable reach the desired value faster; however, it may lead to a bigger overshoot.

To reduce overshoot, a derivative gain is applied because it compensates the control variable with a term

$$\propto -\frac{de}{dt}$$

. For example, if the error is decreasing, the compensation term is positive which “pushes” the control variable away from the desired value, damping the overshoot.

The integration gain is applied to remove system errors and sensor noises. For example, when the model of a system is not accurate, or the actuator has a bias, the control variable will have a constant offset to the desired variable. If the offset is small, the proportional gain and derivative gain can not compensate the error. To remove the offset, an integration gain accumulates all historical error by integration. In this way, the integration gain “magnifies” the offset and compensate the error.

Twiddle

In the class of `PID` (see PID.cpp), I defined a variable called `Type` to let the user update PID parameters or not. When `Type` set to zero, the PID parameters will be updated using the “twiddle” algorithm.

The algorithm was implemented in line `44-114` of PID.cpp. The twiddle algorithm updates PID parameters based on the total cross track error over N = 1000 steps.

Line `90-97` of main.cpp increases the velocity of the car every 40 laps.

Notes: In order to *train* PID parameters, the initial settings is critical. If initial parameter were not set well, the car could easily go out of the track.

Result

A log file of how “twiddle” updates PID parameters is included. As shown in the log file, the car can go at a speed of 46mph in the end.

In the submitted code, the reference velocity is set to 45mph because the car moves more stable at a bit slower speed.

Here are a few reasons that the car cannot go faster:

- the totoal error of every N steps is not consistent, because the car keeps going and does not start at the same point. The car rougly moves one lap over N steps. As a result, the road segment that N-step covers keeps driftting. Since the cross track error is related to the shape of the road segment, the total cross track error over N steps is not an accurate measure of performance of the PID controller.
- the simulator system has small tolerence; the car easily moves out of road and cannot move back to the road.
- algorithm `twiddle` needs too many iterations, which is not practical using the simulator.