PID controller for steering angle

Purpose

The purpose of this project is to use PID controller to control the steering angle to make the vehicle stay in the center of the lane as much as possible.

The PID controller is a three-term controller.
Term P stands for proportion of the current value of the SP – PV error. For example, if the error is large and positive, the control output will be proportionately large and positive, taking into account the gain factor (coefficient). Using proportional control alone in the process with compensation, will result in an error between the SP (the desired setpoint) and PV (the actual process value), because it requires an error to generate the proportional response. If there is no error, there is no corrective response.

Term I accounts for past value of the SP – PV error and integrates them over time to produce the term. For example, if there is a residual SP – PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.

Term D is a best estimate of the future trend of the SP – PV error, based on its current rate of change. It's sometimes called "anticipatory control", as it's effectively seeking to reduce the effect of the SP – PV error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or dampening effect.

The overall control function can be expressed mathematically as:
Kp * error + Ki * integral + Kd * derivative

Kp is the proportional gain, a tuning parameter,
Ki is the integral gain, a tuning parameter,
Kd is the derivative gain, a tuing parameter.

Implementation

I firstly implemented the PID class which takes the cte error from the simulator, and calculate the steering angle as return for simulator to change the steer.

The steering angle calculation follows the formula in the previous page, the 3 parameters initialized as 0, and have 2 variables to save previous cte and accumulated cte.

The next challenge is to tune the 3 parameters, I implemented the twiddle in a separate class, which takes the initial 3 parameters and defines the delta update for each parameter; since the twiddle requires same input to calculate the error and compare the error to decide how to update the parameter, I changed the PID class a bit to make it work for tuning mode, every time if the twiddle changed the parameter, I will send simulator reset message to let the simulator start from the beginning that way I make sure twiddle gets same input each time.

However, I wasn't able to make twiddle figure out best parameters for me, I defined 2000 circles for twiddle to calculate the error, it ends up with an infinite loop for no improving on error.

In the end, I manually tuned the parameters and set PID running in the driving mode (non-tuning mode). A negative value for Kp, a negative value for Kd and 0 for Ki.

Ki seems very sensitive, which is understandable since it would time the accumulated error.

I also normalized the steering value to make sure it stays in between -1 and 1.