Write Up

** PID Controller Project **

The Rubric Points of this project are the following:
* Your code should compile.
* The PID procedure follows what was taught in the lessons.
* Describe the effect each of the P, I, D components had in your implementation.
* Describe how the final hyperparameters were chosen.
* The vehicle must successfully drive a lap around the track.
* Reflection of the PID Controller (You are reading it)

Rubric Points

Here I will consider the rubric points (https://review.udacity.com/#!/rubrics/1972/view) individually and describe how I addressed each point in my implementation.

Files Submitted & Code Quality

1. Your code should compile.

Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:
* Write Up called “PID Controller Project - Write Up
* main.cpp
* PID.cpp
* PID.h
* json.hpp

Submission includes functional code

Using the Udacity provided simulator and the main.cpp file together with the other src Files from the GitHub Rep the car steers itself around the track in the simulator “Project 4: PID Controller”

if you use the entry “./pid 0” or “./pid 1” in the console after going to the right path (using “cd /work/CarND-PID-Control-Project/build”). (0 means no speed limit, 1 means a speed limit of 22 mph, see point 4 for details.)

1. The PID procedure follows what was taught in the lessons.

The PID controller is implemented using the proportional, differential and integral part of the equation, using the same logic as it was presented in the lessons. The main difference is mainly through the framework of a class given as a starting point from the authors of the project. However despite the names (like the steering angle now being mostly referred to as TotalError) the same algorithm is implemented.

2. Describe the effect each of the P, I, D components had in your implementation.

The proportional part of the equation (the “P”), makes sure that the car steers towards the center line of the road, which was the reference line in the lessons. The further the car turns away from the center line the stronger the steering back will “counter” it. This part alone will generally speaking keep the car in a curvy course around the reference line, but this part alone oscillates and overshoots further and further as Sebastian pointed out (and is clearly visible if one only uses the “P” part of the controller.. So to drive around the track the car obviously needs to stay on the road and not further and further drive out of it.

This is where the “D” in PID comes in and corrects that oversteering and smoothes out the driving. However this part of the mathematical equation (respectively the code) alone is only producing improved results, if there is no bias. As soon as there is a bias the differential part alone is not enough and by itself would likely produce less favourable results than the “P”.

In the project I am fairly certain, that the car does not have a bias and hence effectively a PD controller is already sufficient to get the car to drive around the track several times, within the rubric requirements.

In the real world of course there is a good chance that there is a (slight) bias due to the physical mounting of wheels and axis might not be as exact as it would be in the simplified and theoretical mathematical calculation or the idealised coding environment. So taking this into account the “I” in PID will take care of any biases.

I have implemented a full PID controller, however due to the setting of the coefficients the part of the integral equals 0 and hence the car is steered by a de facto PD controller alone. However if the car had a bias, one could simply change the coefficient and the PID controller would be able to handle the less ideal real world scenario.

3. Describe how the final hyperparameters were chosen.

I used the twiddle algorithm to find the best possible values with and without drift/bias. So I used twiddle, but manually tested some assumptions, so you can speak of a mix of these two.

I used the reading from the initial Cross Track Error, that is given by main.cpp of 0.7598 to initiate a more accurate twiddle calculation, by setting the robot.y equal to the cte after the prev_cte variable:

```
# NOTE: We use params instead of tau_p, tau_d, tau_i
def run(robot, params, n=100, speed=1.0):
    x_trajectory = []
    y_trajectory = []
    err = 0
    prev_cte = robot.y
    robot.y = 0.7598
    int_cte = 0
    for i in range(2 * n):
        cte = robot.y
        diff_cte = cte - prev_cte
        int_cte += cte
        prev_cte = cte
        steer = -params[0] * cte - params[1] * diff_cte - params[2] * int_cte
        robot.move(steer, speed)
        x_trajectory.append(robot.x)
        y_trajectory.append(robot.y)
        if i >= n:
            err += cte ** 2
```

console

The result was essentially a minute, practically 0 value for the integral coefficient if using no drift, which makes sense. Since the simulation was astonishingly reactive even to small values I decided to keep it on 0 to practically only use the PD part of the PID controller, as described above.

The final values chosen are:
P: -0.1585734579 (proportional)
I: 0 (integral)
D: -3.1312509606 (differential)

While testing I used the console to not always have to rebuild the executable like this:
./pid -0.1585734579, 0, -3.1312509606 0
(The last value is explained further down in detail, it basically switches the throttle control bit on or off.)

4. The vehicle must successfully drive a lap around the track.

On my local machine it worked both with and without the throttle control.

If you start the code with:

./pid 0

The throttle control portion of my code is not activated and the car will drive as fast as it is possible, only restricted by the simulations slowing down in steering situations. This way the car goes to top speed above 34 mph on a straight part of the road and usually is driving with give or take 30 mph, even through bends.

If the code is supposed to make the theoretical passengers of the car less seasick from the sometimes rather sudden and harsh steering in curves, the throttle control active will let the car not go faster than 22 mph.

The latter can be activated by using:

./pid 1