PID Controller

Writeup

PID Controller Project

In this project, the goal is to implement a PID controller in C++ to maneuver the vehicle around the lake track.

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Compilation

1. Your code should compile.

The project can be compiled using both cmake and Visual Studio. For more information, check out the readme.md.

Implementation

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

For the implementation, the PID controller and the Twiddle algorithm from Lesson "PID Control" have been used as reference. The Twiddle algorithm has been adapted to work in an online context.

Reflection

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

The proportional component of the controller attempts to correct the steering in order to bring the vehicle to the center of the road. The magnitude of the signal is proportional to the cross track error. In order to avoid oversteering, the differential component further tunes the signal relative to the cross track error differential. Finally, the integral component uses a "history" of all cross track errors and attempts to compensate any long term errors such as a systematic bias. In the current implementation, the I-component is used to correct a steering bias of roughly 0.44 degrees to the right.

2. Describe how the final hyperparameters were chosen.

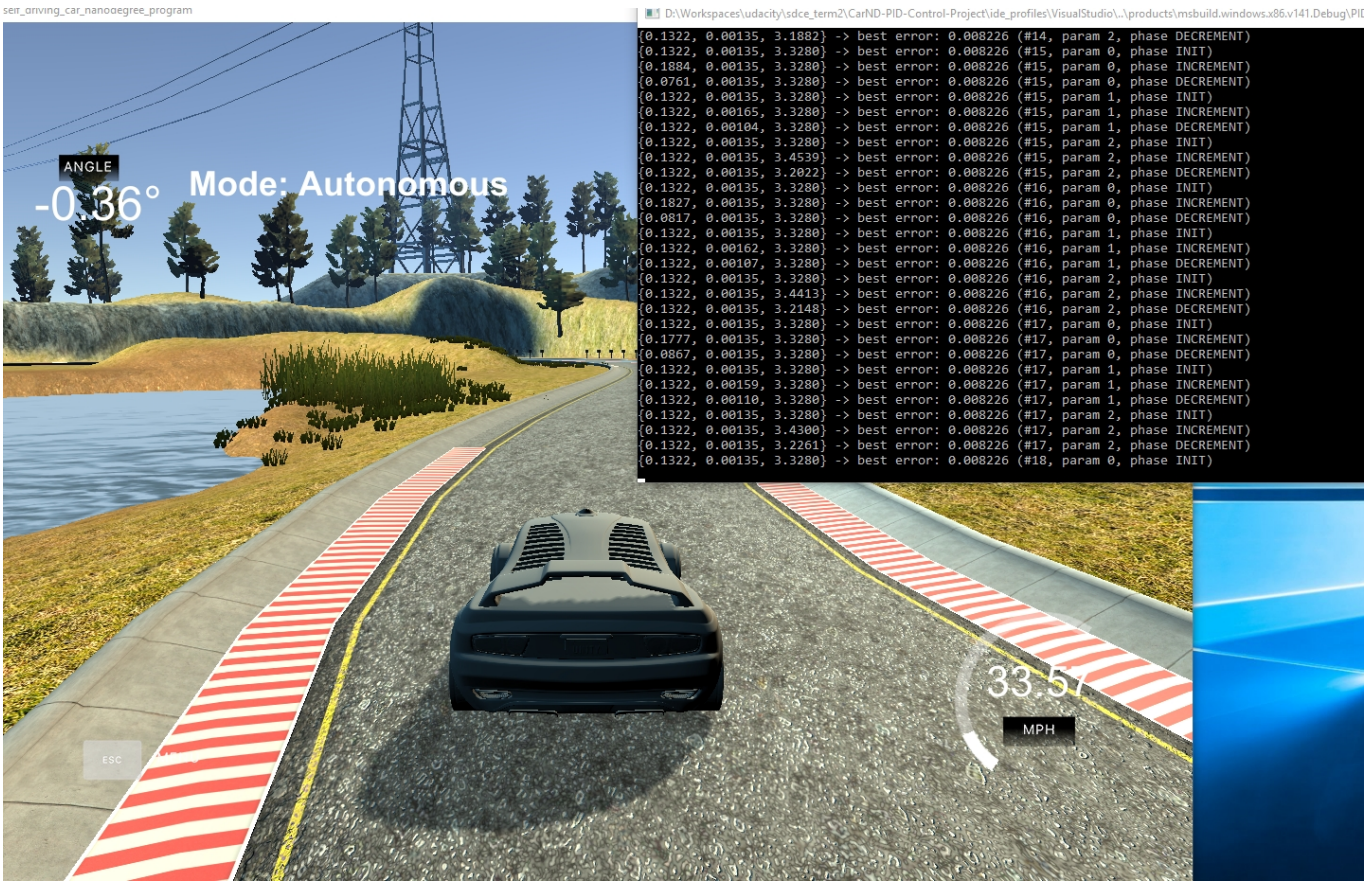
The initial values of the parameters have been empirically set to *Kp=0.2*, *Ki=0.0*, *Kd=3.0*. This yields acceptable results, allowing the car to complete a lap of the track without any incident.

However, in order to improve the performance and achieve a smoother trajectory for the car, an online version of the twiddle algorithm has been implemented. For this, the algorithm has been split into 3 phases: INIT, INCREMENT and DECREMENT. In the INIT phase, a parameter is first incremented using its

corresponding delta value. Then, the algorithm progresses to the INCREMENT phase, where the result of the parameter increment is evaluated and, if needed, the delta is adjusted. If the increment was unsuccessful, the parameter is decremented. Finally, the algorithm progresses to the DECREMENT phase where the effect of the parameter decrement is evaluated. All phases are repeated for each parameter sequentially.

Between each phase, the simulator is allowed to run for 800 ticks. At the end of each phase, the total error is determined which is then used to evaluate the effect of the parameter tuning. Furthermore, the PID controller is re-initialized and the results of each phase are applied.

In the current implementation, twiddle is initialized with the above mentioned parameter values as well as the following deltas: *0.1*, *0.001*, *0.5*. After 18 iterations, twiddle managed to lower the total error from 0.079 to 0.0082. The final parameters identified by twiddle are *Kp=0.132*, *Ki=0.00157*, *Kd=3.328*.



Simulation

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

The car successfully drives around the lap both with the initial parameter set and with the twiddle optimization. However, to avoid the car crashing during the initial optimization phases, the throttle of the car is set to 0.3 for the first 100 twiddle iterations. Afterwards, the throttle is increased to 0.5.