PID Control Project - Reflection

Udacity Self Driving Car Nanodegree

Goals

In main goal of this project is to build a PID controller and tune the hyperparameters.

Constraints:

1. No tire may leave the drivable portion of the track surface.
2. Always ensure a safe driving route from source to destination.
3. Tune the gains of PID controller using Optimization techniques after random initialization.

Inputs:

- Cross track error
- Speed of vehicle
- Steering angle

Implementation Details:

- **Init**() - Initialize default values to all the parameters.
- **UpdateError**() : Updates the error during each cycle and Activates twiddle algorithm to find optimal gain values.
- **Twiddle**() : Function to find the optimal gain parameters .
- **TotalError**() : calculate the total error based on Kp,Kd,Ki and all 3 error values. Total error should be within the range [-1,+1]

Reflection on Controller Generation and Parameter Tuning:

P Controller: Effects of Proportional(P) Term

- The work of Proportional P component is to **reduce** the **cross-track error** (cte).
- It tries to **pull the car** towards its intended steering angle to reduce the error.
- **Drawbacks**: Due to Physical constraints, the car's path will experience **oscillations**.

This is clearly shown in the video attached video **P_Controller.mp4** attached in the path **\CarND-PID-Control-Project\videos\.**

PD Controller: Effects of Derivative(D) Term

- The work of Derivative D component is to **reduce** the **deviation** between successive errors values, i.e it **reduces the derivative** of the **cross-track error** (cte).
- **Pull more** when car is in **wrong direction** (**high derivative**)

- **Pull less** when car is already in **right direction** (**less derivative**)
- This helps in **reducing** the car's **oscillations**.
- **Drawbacks**: Due to some issues like car wheel alignment, slip angle, high cross winds, the car's path will experience some **bias (offset),** due to **accumulation of errors over time**. Car may fail to travel in exact centre, but might have some offset in the track.

This is clearly shown in the video attached video **PD_Controller.mp4** attached in the path **\CarND-PID-Control-Project\videos\**.

PID Controller: Effects of Integral(I) Term

- The work of Integral I component is to **reduce** the accumulation of **cross-track error** over time
- **Pull more** when the **steady state error** is too high**.**
- This helps in **reducing** the bias or offset caused by long time accumulation of errors.
- **Drawbacks**: The Integral part may increase the error during some time interval, but it will get reduced when nearing the steady state of the system.

This is clearly shown in the video attached video **PID_Controller.mp4** attached in the path **\CarND-PID-Control-Project\videos\\.**

Need for Tuning:

As you can see from the implementation, the car's path purely depends on the values of the gain parameters Kp, Ki, Kd. In reality it is not an easy task to find the optimal values for all 3 parameters.

Hence generally the parameters are initialized with some **trial and error** based **reasonable random values** (at least car is able to drive in the track). Then use some optimization techniques to find the best values for the parameters which reduces total error over time.

A simple example of choosing poor random values is shown in **Accident_PID_Controller.mp4** in the path **\CarND-PID-Control-Project\videos\\.**

From the video, you can understand that even though random initialization values lead to following path for some extent same as shown in **PID_Controller.mp4,** there was a **collision** with the road side ledges and car rolls over outside the track. This is of high safety concern which demands the need for tuning the parameters.

Twiddle Algorithm:

There are many kinds of optimization algorithms for such kind of problems like different versions of Gradient Descent, Newton method, Gauss-Newton, Lagrange method and so on. As most methods are complex and some needs higher order derivatives, I have implemented the Twiddle algorithm which is discussed in classroom.

This is a simple algorithm, which works as explained below.

- Increase each gain parameters with some delta values.
- If the error gets reduced, increase the delta value by a factor > 1.0 (1.1 is used in code).
- If increasing does not work, decrease the gain and do the same process.
- If both increasing / decreasing does not work, reduce delta by a factor < 1.0 (0.9 is used).
- During twiddle updation, the car may experience oscillations and other problems due to change in gain parameters in each cycle, which lasts only until it reaches the optimal values.
- The behaviour during updation can be seen in the video **During_Twiddle_Update.mp4** added in the path **\CarND-PID-Control-Project\videos\\.**

The tuning is done with a random initial value and it converges after around 500 iterations.

The optimal tuning values prevents collision with the barriers and make the Car to stay in the track.

This can be seen in the video **Accident_Prevented_With_Twiddled_Values.mp4** in the path **\CarND-PID-Control-Project\videos\\.**

Result:

The Car is able to travel safely inside the track without any collisions or roll over.

Values used while capturing videos and execution of the code is given in the below table.

Controller	Kp	Ki	Kd
P controller	**1.0**	0.0	0.0
PD controller	**0.1**	0.0	**1.0**
PID Controller without Tuning	0.04	0.00005	0.75
PID Controller + Twiddle Tuning	0.18631	0.000259837	3.02721

Future Works:

- As we very well know that all optimization methods listed are only **Local optimization** methods. The optimal values mainly depend on the **Initialization**.
- **Global optimization methods** like simulated annealing or Black box based global optimization techniques can be used to get the Best gain parameters.
- Also, as you witness that the car may experience some oscillations during big turns, which is due to the fact that **speed parameter is not tuned** here. Turning with high speeds, causes car to move away from intended path. Other reasons being the **initialization of parameters** and other **physical constraints**.