PID Final Project

Author: Nolan Lunscher
Date: 2021-10-30
Repo:https://github.com/nlunscher/nd013-c6-control-starter

Introduction

In this project we are tasked with applying what was learned about PID controllers to a simulated self-driving car. Here two PID controllers are used to control the throttle and steering of the vehicle such that it can successfully navigate down a street and avoid stationary obstacles.

Question Answers

Add the plots to your report and explain them (describe what you see)

Below are the plots showing the error and controller output for steering and throttle. As can be seen, the controller output signal is applied in the opposite direction of the error. This makes sense because the control is applied to minimize the signal error. In both cases the magnitude of the control output is less than that of the error. This is because I tuned the Proportional component to be less than one.

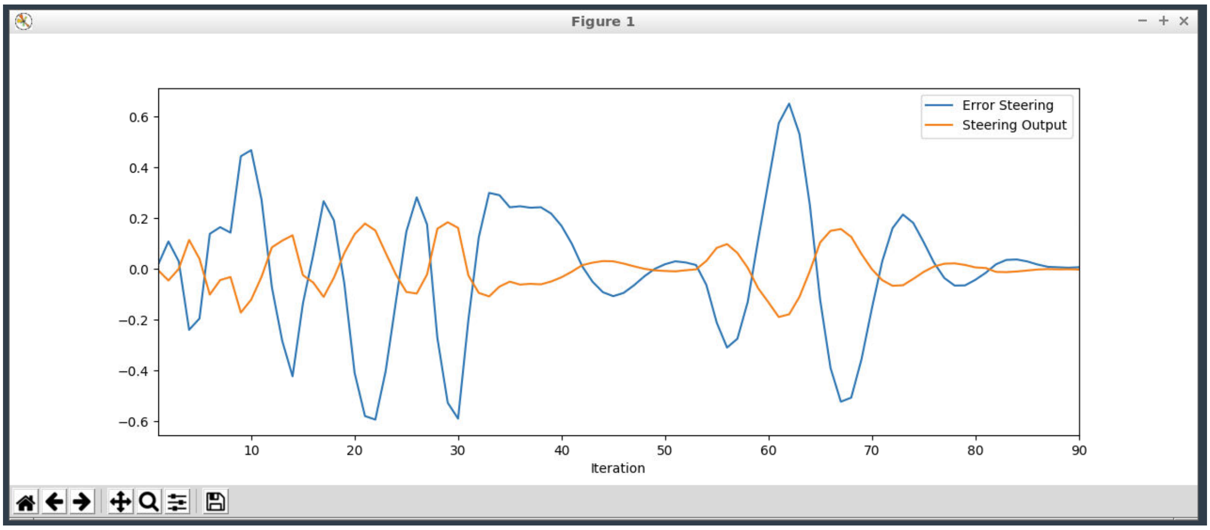

Figure 1: Steering error and controller output.

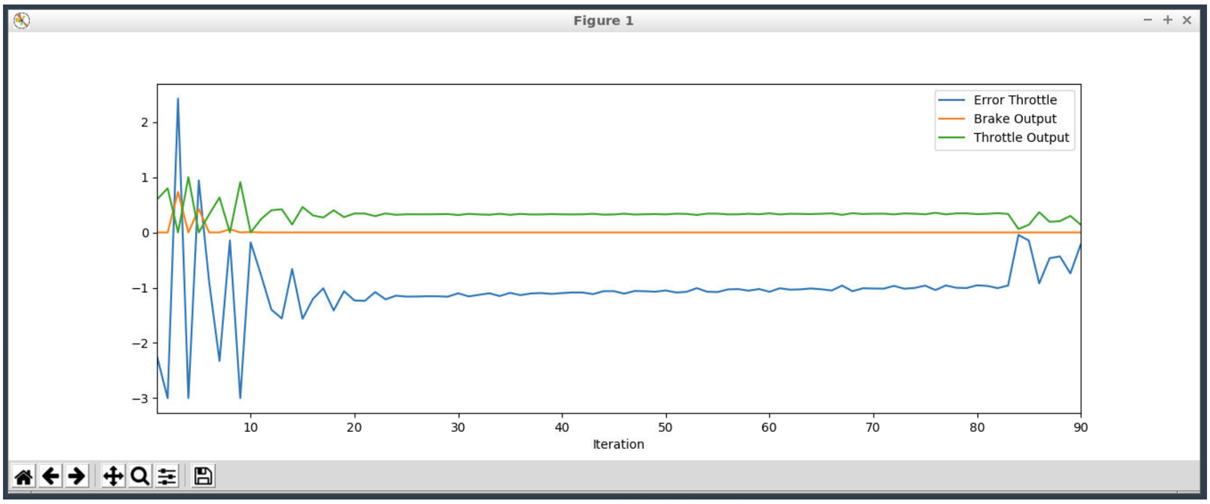

Figure 2: Throttle/Brake error and controller output.

What is the effect of the PID according to the plots, how each part of the PID affects the control command?

Each part of the PID params controls a different part of the response of the PID controller. The Proportional component controls how aggressively the controller will attempt to correct the immediate error. I tuned this part to be less than 1, and therefore the controller outputs are always less than the measured error. The Integral component is used to correct for the total error over time, and to avoid system biases. I tuned this part to be quite low, so the effects aren't really seen in my plots. The Differential part is used to dampen the control output such that it doesn't overshoot too much. In my plots this can be seen its its behaviour to decay rather than oscillate, especially near the end of the path in the steering plot.

How would you design a way to automatically tune the PID parameters?

I would use the twiddle algorithm. Given an overall evaluation function to maximize, the twiddle algorithm is able to optimize the PID parameters by iterating on a set of controller inputs until a stable and optimal set of behaviours is achieved. It does this by taking small steps up or down on each of P, I and D and looking at their effects on the evaluation function. In this project, the evaluation function could be the total control error for the drive down the street. The twiddle algorithm would wrap the whole process, and drive down the street many times with different PID parameters, adjusting them each until the optimal parameters are found that can drive down the street with the least amount of error.

PID controller is a model free controller, i.e. it does not use a model of the car. Could you explain the pros and cons of this type of controller?

The benefits of a model free controller are that they are significantly more portable than a controller that requires a car model. This means that the same basic PID controller algorithm can be applied to other unrelated applications (e.g. planes, toys, ovens) without significant modification. This is good because with an understanding of PID in one domain, you will also have a good idea about how to use a PID in any number of other domains.

The down side to this however is that there might be domain specific knowledge that is not being leveraged in order to optimize the controller's performance. An example of this could be specific knowledge about the physical characteristics of the vehicle, such as its mass and the friction properties of its wheels and brakes, as well as the position and turning characteristics of the wheels themselves. Knowledge of these types of variables could be used by the controller to model, predict and optimize the behaviour of the vehicle to various controller outputs before deciding on what outputs to provide.

(Optional) What would you do to improve the PID controller?

I think the first thing to do would be to wrap the system in such a way that the Twiddle algorithm could be run. This would further optimize the system without requiring fundamentally changing the PID controller.

Another might be to better incorporate more information from the trajectory. I found the best results by measuring cross track error with the nearest trajectory point, however this throws away a lot of information provided by the planner. Perhaps there is more that could be done by looking at more of the trajectory.

One other thing that could be done would be to use different PIDs or cross track error models for different behaviours. For example, a different PID could be used when following a lane vs when navigating around an obstacle or passing vs when coming to an intersection. In this way the parameters could be tuned differently for tasks with different needs.