Reflection

Implementing the controller

The implementation of the PID controller was rather straight forward. And I believe within about half an hour I was able to get the code up and running. So I started to tune and tried to get it to run at least one lap, so that I could speed up and kept on tuning it.

The process was, however, not as smooth as I expected. I could only get the car to run at 0.1 of the full thrust without the car going haywire. And I thought I found a set of parameters to make the controller work, at least for 0.1 thrust, so I started implementing the twiddle algorithm.

Twiddle

The twiddle algorithm I implemented was slightly less efficient than the one in the lecture as I wasn't really familiar with socket programming so I had to modify it a bit to get it to work. The version I implemented seemed to be working fine, but it would fail as soon as the thrust was increased to a threshold. And I didn't know why. I thought about a few possibilities:

1. The gap between each tune is too big
2. The cost function is not convex and my initial pick was bad so I was stuck in the local minimum
3. There's a bug in the twiddle code that I wasn't aware of

SGD

My solution was to implement SGD, which would seem to be able solve all of the above at once. And I was wrong. The same problem happened to SGD. While I was able to eliminate possibility 2., I still didn't know how to pick the right initial values and whether there was a bug.

Reaching out for help

So my last resort was manual tuning. Manual tuning allowed me to tune the parameters any way I wanted, so whatever reason it was to have caused the issue could be solved. And I was wrong again! I used the most common way of tuning: P ➜ D ➜ I. But no matter what I tried I couldn't get it to stabilise.

After a few days of trying I decided to give up working on my own and went on to Slack to see if anyone else had the same problem. And just a few scrolls I found the problem which seemed to be the problem of a lot of people—the implementation of the PID controller itself!

There is supposed to be a delta t term in both D controller and I controller, and most people are missing it. And the reasons I believe are twofold:

1. In the class, delta t has always been set to 1 for simplicity
2. In the starter code, the function is declared without delta t so it's a bit misleading

After the fix, the tuning progress was a lot smoother. Unfortunately I didn't keep my optimizer code (neither twiddle nor SGD as I thought they were probably too buggy…what a mistake), and the deadline was approaching. So I decided to tune it manually with some extra constraints on the hard turns.

Conclusion

I think the project is really good fun and it was also my first time implementing SGD. There are a lot of implementation details of SGD that I had never really thought of are actually quite important, such as initialisation of weights and deltas, the singularity problem, and also when to stop the algorithm. I've also learned a lesson about debugging and the problem of working solo, haha.