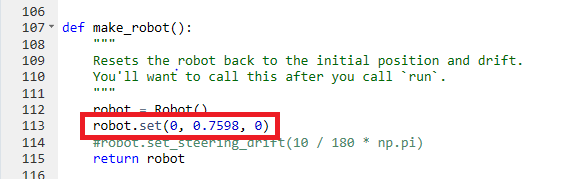
|  |  |
| --- | --- |
| Describe how the final hyperparameters were chosen. | Student discusses how they chose the final hyperparameters (P, I, D coefficients). This could be have been done through manual tuning, twiddle, SGD, or something else, or a combination! |
| Describe the effect each of the P, I, D components had in your implementation. | Student describes the effect of the P, I, D component of the PID algorithm in their implementation. Is it what you expected? |

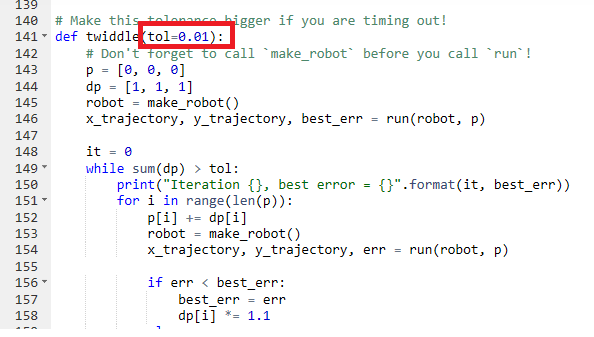
to choose the final parameters I used „twiddle” method that is part of the PID controller lesson. I have tried various combination of parameters like:

* speed
* initial CTE
* tolerance
* steering drift

my assumption was that moving a robot is just like driving a car. Based on the simulator the initial CTE value is 0.7568 therefore I set this value as the “y” coordinate of the robot.

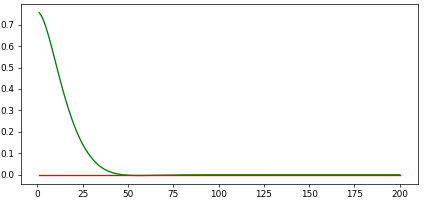


I reduced the tolerance from the original 0.2 to 0.01 to get more accurate tau values for P, I, D. Further reduction of tolerance causing timeout.

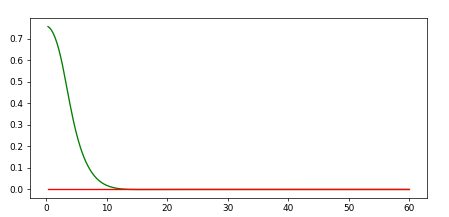


Changing speed from “1.0” like to “0.3” results higher PID values, means that the car gets closer to “0” CTE value much earlier as the calculated steering angle is higher. As a “disadvantage” the movement of the car is more “uncertain”, lots of correction is happening as the car overshoots “0” CTE more frequently and it needs to come back. It also slows down the car.

twiddle PID control diagram with speed “1.0”



twiddle PID control diagram with speed “0.3”



I also set steering drift to 0, assuming there is no “issue” with the car. As there is nothing to compensate from that point of view, “I” is set to 0.

Finally, I found the following parameters working:

P: 0.19

I: 0.0

D: 3.14

Higher “P” results more oscillation. The car overshoots all the time and makes lots of correction.

with higher “D” value the car movement is smoother after car got closer to “0” CTE.