Step 1: Build the PID controller object

Step 2: PID controller for throttle:

For the PID controller for throttle, I did some tests with different values but it always ended up with the car hitting the car in front, so I decided to leave it to optimize after I had the steering control, so that the car could go around the car in front.

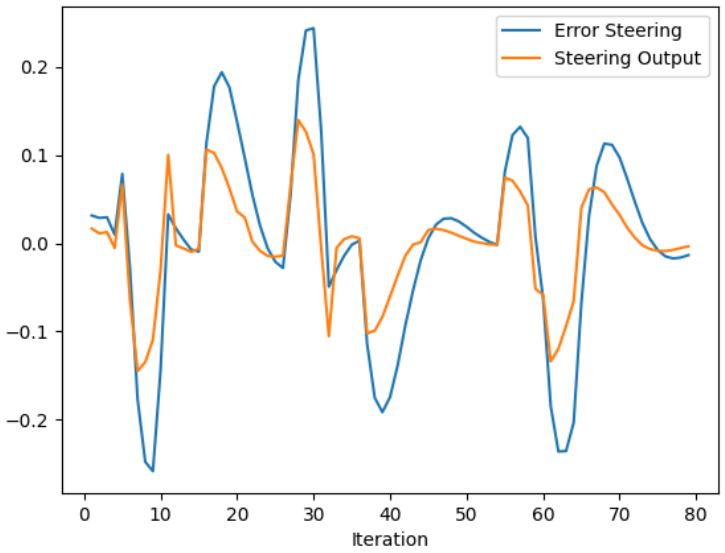
Step 3: PID controller for steer:

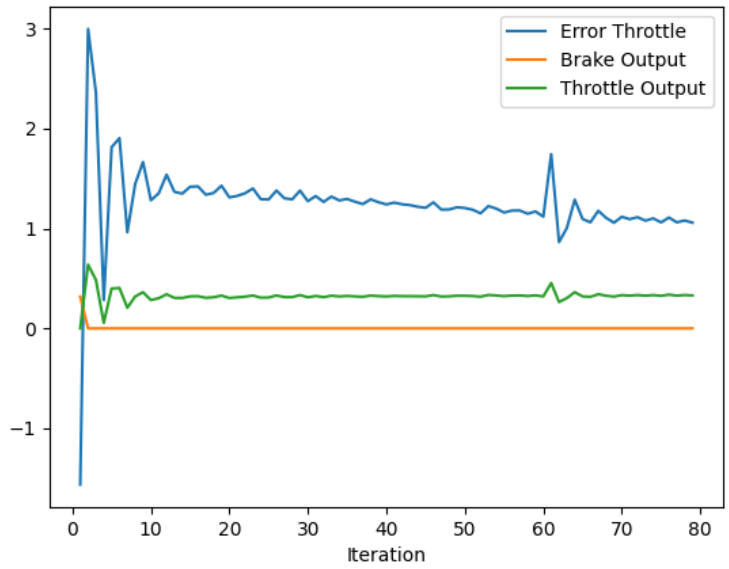
For the PID controller for steering, it took many tests with different values (mainly for the throttle pid) so that I could control the car without hitting any obstacle in the scenario (car, wall, etc).

Example of unsuccessful test: pid_steer.Init(0.2, 0.01, 0.04, 1.2, -1.2) and pid_throttle.Init(0.4, 5, 0.0098, 1.0, -1.0)

Test that succeeded: pid_steer.Init(0.4, 0.002, 0.5, 1.2, -1.2) and pid_throttle.Init(0.2, 0.001, 0.01, 1.0, -1.0)

Step 4: Evaluate the PID efficiency





1. **Plot descriptions:** The error of the PID control for steering remained relatively low during all the time, it had some oscillations between zero, but not very high.
The PID control for throttle not converged to a sufficiently low error, it visibly needs a very long time to converge. In theory could easily be corrected with an increase in the coefficient for kp and kI, but in the case where the PID was implemented with a high value with kp or kI, it led the car to hit the car in front.

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

- Term **P** is proportional to the current error, if the error is large, the control output will be proportionately large by using the gain factor "K_p". Using proportional control alone will result in an error between the set point and the process value because the controller requires an error to generate the proportional output response.
- Term **I** accounts for past errors and integrates them over time to produce the **I** term. For example, if there is a residual error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error.
- Term **D** is a best estimate of the future, based on its current rate of change, as it is effectively seeking to reduce the proporcional error by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or damping effect.

So in the example evaluated in this course, theoretically an increase in the kp and ki factor could make the error for throttle converge faster, however we have here many correlated processes, which cannot be controlled so simply with isolated PID processes. First, the yaw rate is correlated with vehicle speed, and here they were contracted separately. according to other control factors listed here, such as not hitting obstacles, which should also be modeled in the error calculation.

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

I would use some optimization algorithm, as for example, Twiddle, given in the course, but focusing on optimizing the PDI for the two processes together.

4. 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?

Pros: It is a simple, low-cost model to implement, and it is easy to understand how it will operate according to the values of the configured parameters.

cons: as it is a very simple model, it has limitations to control more complex systems. the example where the algorithm was implemented had a low complexity, but it was enough for the algorithm to show its limitations in the control. The fact that it does not model the correlation/dependency between the two modeled controls (steering and throttle) shows its limitation in the example applied here.

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

Model the dependency between steering and throttle.