

PID Control Project

Ximing Chen

I. GOALS

The goal of this project is to implement a PID controller to control a vehicle in virtual environment to prevent it from driving off the road.

II. PID BASICS AND TUNING

A. Preliminaries on PID Control

Proportional, integral, and derivative controller is often used in industry as a feedback control mechanism. More specifically, the controller $u(t)$ leverages the past trajectory of the system in the following form:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}, \quad (1)$$

where $e(t)$ is the difference/error between the actual trajectory and the desired trajectory. In discrete-time, the above equation writes:

$$u[k] = K_p e[k] + K_i \sum_{\ell=0}^k e[\ell] + K_d (e[k] - e[k-1]). \quad (2)$$

The term K_p governs that the system is moving towards the desired trajectory. For example, consider a 1-D system with reference trajectory $y(t) \equiv 0$. In this case, when the state $x(t)$ is positive, the error is positive. Subsequently, a negative feedback will drive $x(t)$ towards zero, vice versa. In particular, a larger K_p aims to drive $x(t)$ to desired trajectory as quick as possible.

The term K_d decides how much the system is overshooting. The introduction of K_d arises because of following reason. Notice that when only K_p term is used. Then, even when the error is small, the system may drive across the desired trajectory due to momentum of the system. Subsequently, one may expect that the trajectory oscillates around the desired trajectory when only K_p is used. Introducing K_d term efficiently mitigates such defects as it measures the “speed” of the system. When the system moves “slowly”, it is expected that

the system operates around the desired trajectory, whereas when the system deters from the desired trajectory by a large amount, then a large control should be used.

The term K_i ensures convergence to the steady state. Notice that given a desired trajectory $y(t)$ one can consider $e(t) = x(t) - y(t)$. When there is system bias, using only K_p and K_d cannot drive the system towards exact trajectory. However, if the system has constant error, i.e., does not operate at the desired trajectory, then these constant error will accumulate overtime. We then leverage them to construct a control signal to indicate the system to move towards $y(t)$. In other words, when the accumulated error is large enough, the integral error becomes significant in the control signal.

B. Selecting PID Parameters

From the above analysis, we see that K_p and K_d decides whether the system can operate around the desired trajectory efficiently and asymptotically. Subsequently, we first set $K_i = 0$. On the other hand, as we want less oscillations around desired trajectory, we set $K_d > K_p$. In the project, I manually tried initializing $K_d \approx 2.5$ and $K_p \approx 0.5$.

From the simulation, I observed that the car oscillates vastly and drives off the road when turning. This indicates that K_p is probably too large. Therefore, I tried a grid-search idea on (K_p, K_d) and found the working parameter set.

Another possible automatic way of implement this is to use twiddle. The steps of twiddle is described as follows:

- 1) Set a counter on how many iterations should we evaluate the current parameter set;
- 2) Compute the total error using the sums-of-squares of cte obtained from the program;
- 3) Adjust using twiddle on a single parameter, e.g., K_p first and K_d second. The change on these parameters are set to be around 0.1 of the original values.