1 Getting started

In this session, we will implement a PID controller on a simple simulation of a self-driving car. Next, we will optimize the parameters of the PID controller to make sure our car is as accurate as possible while being robust to disturbances.

First of all, make sure you have all the necessary files, which you can download from Brightspace. Next, install the *pygame* Python package.

In our simulation, we will work with the concept of a self-driving car with a constant speed. It will be our job to make sure that our car remains centered on the road. Sensors on the car allow us to determine our *cross-track error* (CTE): the perpendicular distance to our desired track. In other words, our task is to minimize our CTE, and we can only do so by turning the steering wheel. Our task then becomes: what inputs do we give to the steering wheel to minimize CTE, given that we can only sense our current CTE?

We will start with the `main.py` file. Open the file and take a look. It contains a function `move_player`, in which the current CTE is calculated. Now look at the `pid.py` file. This contains the framework for implementing your PID controller. Feel free to experiment with the example values.

2 Implementing a P controller

A P controller calculates the required steering angle proportional to the observed CTE, parameterized by one P parameter. Implement it in the `PIDcontroller` class in the `pid.py` file. Experiment with different values for the P parameter. What do you observe?

3 Implementing a PD controller

Obviously, we want to do better. We can do so by implementing a differential term in our controller. The steering angle will then be determined both proportionally to the CTE, but also by the derivative of the CTE with regard to time. Our PD controller will then be parameterized by both a P parameter and a D parameter. What do you observe?

Now we will set the `steer_bias` variable to `True`. Note that your controller might not work as well now. To solve this problem, we will need to implement a full PID controller.

4 Implementing a PID controller

We will need to include the integrative term in our controller to compensate for systematic bias. Once implemented, you now have a controller that is parameterized by three parameters: a P, I, and D parameter. Experiment with different values for these parameters. How can we determine which parameter combination is the best?

5 Implementing Twiddle

For our parameter optimization, we will use Twiddle. You can find multiple implementations around the web, but make sure you understand the principle behind it, and refer to the explanation given in the lecture.

Implement it in your code. You are allowed to make changes to the structure of the code, but are not allowed to change the dynamics of the car and track. In other words, only change code to implement Twiddle. For example, you may want to wrap a large part of the code in a function that returns a value representing the goodness of your parameterization.

5.1 Lab report template

Use the Google Docs template provided here. Look over the template to see general formatting guidelines. You can just use a *Heading 1* for each item discussed. You can remove the abstract, as it is not necessary for this assignment.

5.2 What should be in the lab report?

Make sure you include the following in your lab report:

1. Visualizations for and explanations of the concepts of P, PD, and PID controllers.
2. A high-level explanation of the Twiddle algorithm.
3. Visualizations of experimental manipulations of the PID parameters. What happens when you manipulate your parameters? Use *matplotlib* to graph your car's behavior.
4. Visualization of your optimal solution, after having your parameters optimized using Twiddle.
5. Implementation of a PID controller on a physical mBot. Describe difficulties encountered and performance. More information will be given during the session on 19 April.

You will be graded on the clarity and completeness of your report. Do not hesitate to ask the lab session instructors for help, either with writing or getting through the code.

5.3 Lab report submission

You should submit your lab report no later than **Friday, 28 April 2023 at 23:59 CET.** The lab report should be submitted through Brightspace in PDF format. You should also submit your *main.py* and *pid.py* files through Brightspace so that we can check for correctness and plagiarism.