Lab 6: PID Controller

Due: Tuesday, December 2nd 11:59pm EST

The objective of this lab is to implement a closed loop control system with a PID controller for Cozmo. Starting from random locations, you will be controlling the speed of Cozmo's wheels in order to drive on a straight line and stop in a pre-specified region in front of a cube.

Background

PID is one of the most widely used controllers across the world. In this lab, you will get to implement and test the PID controller on a physical robot system - the Cozmo. Refer to the lecture slides to understand the role of each term in the controller and the effects of changing their gain values. While certainly not necessary, feel free to use this tool to get a better sense of how the controller gains affect a system's performance.

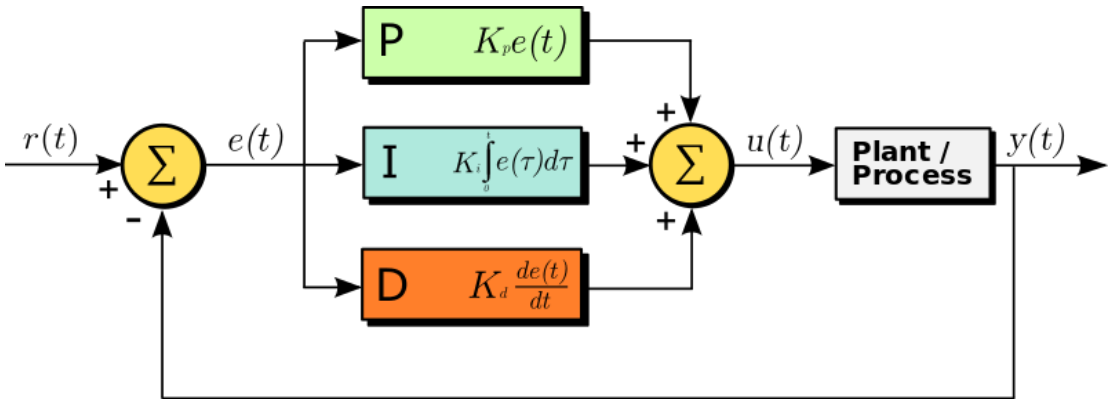

Figure 1. A block diagram of the PID controller

Part 1 – Tuning

For the tuning part, we have provided the file `tuning.py`. This file sets the values of the control gains K_p, K_i and K_d. You can either tune these gains manually or use any tuning method, such as the Zeigler-Nichols (ZN) method. For instance, if you choose to use the ZN method, you will have to write code to measure the oscillation frequency of a controller or other system dynamics. Once tuning has been completed, you can use the `CozmoTuning` method to create a dictionary of control gains and write them into `config.json`. This method is executed by the `RobotThread` and should contain all your Cozmo behavior. You are not required to use `tuning.py` and are free to use any tuning process you wish to. Irrespective of how you choose to tune the controller, you do not need to submit this file, but you need to submit your `config.json` containing the values of your control gains.

Part 2 – PID

For the PID portion, you will complete the `CozmoPID` method in `pid.py`. This method is executed by the `RobotThread` and should contain all your Cozmo behavior. Your implementation needs to drive Cozmo and stop inside the orange region in front of a cube (see Fig. 2). You do **not** need to turn. During the demo, your cozmo will be placed at three different points on the road (see example points A, B, and C in Fig. 2). **You will not be given the locations of these three points before the demo - you will have to tune your controller without knowing the start locations.** The time required to reach and stop within the orange region will determine your grade. The time does not stop until Cozmo stops completely. You should detect the initial distance to the cube at the start of the program by reading the cube position. Once the initial distance is perceived, you will longer need to rely on perception and may instead rely on `robot.pose.position.x` or equivalent.

The objective is similar to the automated stop sign braking system example in the Intro to Control slides. **You cannot use `drive_straight` or `stop_all_motors` for this implementation.** If you use the `drive_straight` function or any functions for controlling the wheel motors other than `drive_wheel_motors`, you will not get any credit for this lab. Instead, you must use the provided `move` method that uses `drive_wheel_motors` to control the speed of the left and right wheels of the robot. **You should only use the `move` method for wheel speed control.**

Each Cozmo will likely require different gain values given the hardware differences. However, **you cannot share or discuss your controller gain values with other teams.**

You can write any helper functions you wish to as well. At each iteration of the loop, you must print the Cozmo's speed to ensure that the speed is being dictated by your controller.

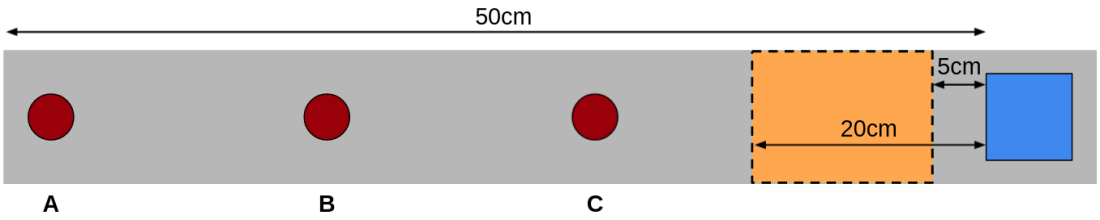

Figure 2. A, B, and C are example (unknown) initial positions for the Cozmo. The Cozmo is expected to drive straight and stop to be fully within the orange region in front of the goal cube.

Part 3 – Disturbances!

For this portion, we will introduce disturbances (scary)! At random points in time, we will introduce an impulse (either "push" or "pull"), and your PID will have to adjust its wheel speed to compensate. Each push or pull will be applied for approximately 0.3 seconds shortly after Cozmo starts moving. (this is set by `start_dist` in `move`) We will be grading based on how quickly and effectively this compensation is administered in your PID system. We will test Cozmo on two different disturbances.

The provided `move` method integrates disturbance for you. You can apply disturbances by adding a disturbance argument in your command. For example, `python pid.py -10` would apply a pull of -10mm/sec to the Cozmo wheel speed for about 0.3 seconds. In the demo, we will use disturbances around -200mm/s ~ 200mm/s to test your controller's reaction. Cozmo will start from a reasonably long distance to give enough time for Cozmo to compensate for the disturbances.

Grading:
Your grade will solely be based on your demo performance. We will be testing your controller's performance by placing Cozmo at three (unknown) starting points in front of the cube. For each of these starting points, you will have 3 opportunities to record the best (fastest) time. **You will lose points if Cozmo gets closer than 5 cm to the cube (i.e. overshoots the orange region), or crashes into the cube.** If Cozmo either crashes into the cube, or doesn't fully stop in the stopping region, you will get no points for that run. For the disturbance part, we will Cozmo on two different disturbance values. You will be given three opportunities per run, and will receive full points if the Cozmo stops within the stopping region. **There will be a small oral component in this lab**: you will be expected to explain how you tuned your controller. You will have the opportunity to score a maximum of 6 extra credit points in this lab (2 points for each starting point).

Cozmo reaches goal from point A in:	4-5s 5-7s 7-10s >10s	25 pts 20 pts 10 pts 0 pts
Cozmo reaches goal from point B in:	3-4s 4-6s 6-9s >9s	25 pts 20 pts 10 pts 0 pts
Cozmo reaches goal from point C in:	2-3s 3-5s 5-8s >8s	25 pts 20 pts 10 pts 0 pts
Cozmo compensates for disturbances for:	2 runs 1 run 0 runs	15 pts 10 pts 0 pts
Oral explanation of controller tuning		10 pts
- Crashes into the cube		-5 pts each
- Overshoots without crashing into cube		-5 pts each

- Excessive oscillations around the stopping point	-5 pts each
+ Reaches goal faster than required for full credit	+2 pts each

Submission: Create a zip file that includes both `pid.py` and `config.json`. Make sure you enter the names of both partners in a comment at the top of the files. Make sure the files remain compatible with the autograder. Name the zip file `lastname1firstname1_lastname2firstname2_pid.zip`. Only one partner should upload the file to Gradescope. **Please make sure to do a group submission by adding your partner to your team in Gradescope.** If you relied significantly on any external resources to complete the lab, please reference these in the submission comments.