CSCI 445 Lab 5 — Go to Goal

Prof. Culbertson, University of Southern California Spring 2020

In this lab we will add an integral term to last week's PD controller to get a full PID controller. We will use the controller in various settings: go-to-angle, and go-to-goal.

1 Prerequisites

- Review your findings from the previous lab.
- Review feedback control (i.e. Lectures 7 and 8).
- Review Odometry (Lecture 6).
- Bring your laptop.

2 Go To Angle — PD Controller

We want to use a PD Controller to go to a specified angle.

2.1 Controller Design

What is the state here and how can you measure it? How do you use the controller output to drive the robot?

2.2 Simulation

Implement your controller by leveraging the `PDController` class from last week. Test your controller in simulation for angles π, $-\frac{\pi}{2}$, and $\frac{\pi}{2}$. Tune your gains until you get a reasonable output and provide a plot of the angle over time using those gains. In your plot, show current and desired state on the y-axis and time on the x-axis.

3 PID Controller

Now we would like to extend our controller to include an integral (I controller) term and compare it to the PD controller.

3.1 Controller Implementation

Copy your existing `PDController` into a new file called `pid_controller.py`. Change the class name to `PIDController` and update the code such that your controller includes the integral term.

Hint: Consider the case where your goal state suddenly changes. Your error would be very big until you manage to reach the goal state. Unfortunately, the integral portion "remembers" the error and might grow very large. Hence, you need to make sure that the accumulated error can be clamped independently (where the clamping range is user-specified and depends on your application).

3.2 Go To Angle in Simulation

Repeat your experiment from the last section this time using the PID controller. Now you have three gains and two clamping ranges to tune. Provide plots of at least three different sets of gains and report a good set of PID gains. How does the PID controller compare to the P and PD Controllers?

3.3 Robot

Try your controller from 3.2 on the robot and re-tune the gains if required. As before, also provide a plot of the angle over time.

4 Go To Goal

We would like to go to a goal rather than rotating just in place.

4.1 Simulation — Base Line

Update your code to go to a specified goal location. As in Lab 4, use a base velocity v_0 and control only the angular velocity of the robot (ω).

What happens when you reach your goal location? Create a plot of the current angle of the robot versus the dynamically changing desired angle. What is causing this issue?

4.2 Simulation — Improved Solution 1

A simple solution is to tell the robot to stop when it gets close enough to the goal. Implement and test this in simulation. As in 4.1, generate a plot of the current angle of the robot versus the dynamically changing desired angle.

4.3 Simulation — Improved Solution 2

If we just tell the robot to stop when it gets near the goal, it may never reach the goal. A better solution is to use a second PID controller to control the robot's velocity. Then, it should move faster when it is farther from the goal and slow down as it gets near the goal. Implement and test this solution in simulation. As in 4.1, generate a plot of the current angle of the robot versus the dynamically changing desired angle.

4.4 Robot

Now you are ready to try it on the robot. We will ask you to show us one of the solutions, and we will pick which one.
As in 4.1, generate a plot of the current angle of the robot versus the dynamically changing desired angle.