LAB 5: PID CONTROLLER

PROFESSOR HARISH RAVICHANDAR
CS 3630, FALL 2023

Due: November 14th 2023, 11:59 pm EST

SETTING

You are a robot trapped at sea, with only a thin path of ice that you can navigate on. Don't fall off the icy path! You will have to make it to the red helicopter landing zone within a certain amount of time in order to survive the cold! Make sure to stop within the landing zone, without overshooting it, or else you will fall into the sea.

The objective of this lab is to implement a closed loop control system with a PID controller to get the robot safely to the goal zone. In this assignment, we assume you have implemented a Kalman Filter, particle filter, or any other kind of localization algorithm so we give you perfect sensor readings. However, the robot's actuators are very noisy and must run with acceleration constraints.

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 simulated robot system. Refer to the lecture slides to understand the role of each term in the controller and the effects of changing their gain values.

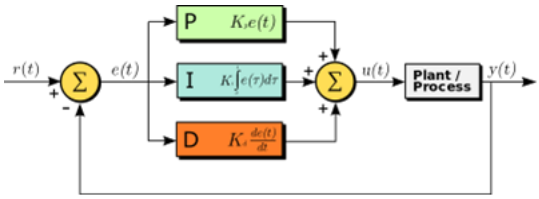

Figure 1. A block diagram of the PID controller

PROJECT SETUP

Download the project zip and unzip. Then, run `pip install -r requirements.txt` in the directory same directory as `requirements.txt`

GENERAL GUIDANCE

You must create a PID controller to complete 6 unique scenarios. You wneed one controller that will modulate the robot's linear velocity, and one controller that will modulate the robot's angular velocity.

The two PID controllers (linear and angular) will likely have different gains. **However, you are not tuning different PID controllers for each scenario.** You will create one linear and one angular controller that should work for all scenarios.

You can write any helper functions or create any variables you want. You will need to write code that can calculate your respective errors, keep track of your previous errors, utilize the gains, and more.

The program will terminate a scenario when the robot's linear velocity is less than .005m/second.

The robot must drive over ice so it accelerates slowly. This means if your controller tells the robot to move at max speed one time step and then 0 m/s the next time step, it will take time for the robot to decelerate to actually move at 0 m/s.

Note: Just implementing two basic PID controllers with carefully tuned gains should be enough for an A on this assignment. However, you will need to do something more for full credit. Specifically, you will need to figure out how to get the robot to stop within the target region without being too slow. There are multiple ways to accomplish this – we will accept any viable solution.

SCENARIOS

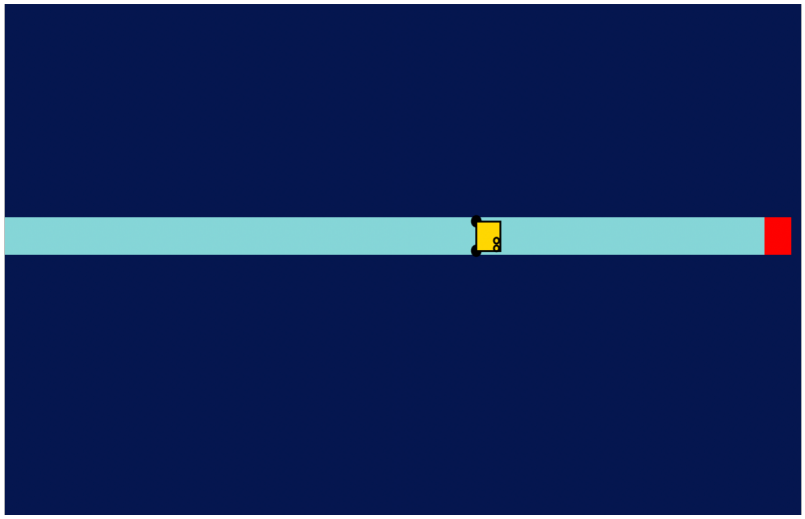

Figure 2. Map for scenarios 1-4.

SCENARIO 1

For Scenario 1, the robot must drive straight along a road and stop within the red region, as seen in Figure 2. Note that you will need at least need to have implemented the proportional component of the linear controller and have tuned the respective gain constant to pass scenario 1. This will involve calculating linear error.

SCENARIO 2

For Scenario 2, the robot must again drive straight along a road, however, this time one of the robot's wheels is malfunctioning causing it to arc! To compensate for this, your controller will need to be able to handle constant angular noise, and stop within the red region. You will at least need the proportional and integral components of both the linear and angular controller, and have tuned their respective gains, to pass scenario 2.

SCENARIO 3

For Scenario 3, the robot must again drive straight along a road but this time one of its wheels is malfunctioning again *and* there's unpredictable wind that it has to drive through. Therefore, the robot will need to handle angular noise and occasional random bursts of wind (i.e angular noise), and stop within the red region. You may also want to consider using the derivative terms to pass this scenario.

SCENARIO 4

For Scenario 4, the robot must drive straight along a road and stop within the red region, but this time it is starting at an angle. Make sure the robot doesn't immediately drive off the road!

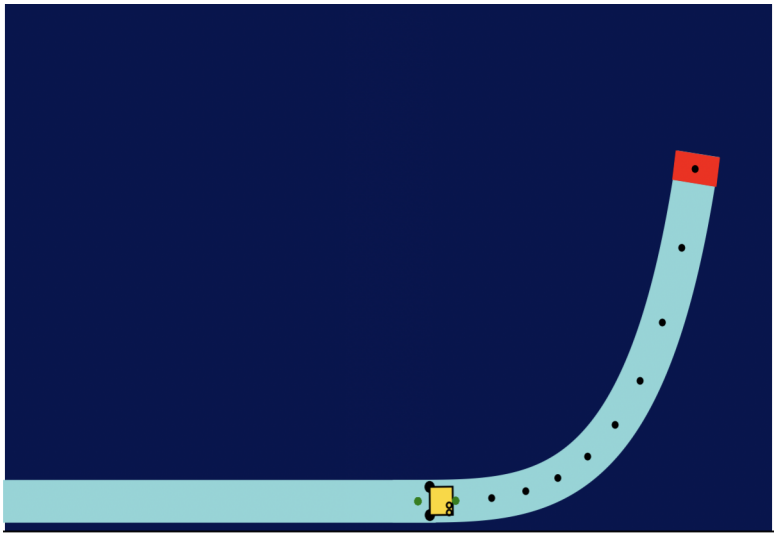

Figure 3. Map for scenarios 5-6.

SCENARIO 5

For Scenario 5, the robot must drive along a curved road, and stop within the red region, as seen in Figure 3. Note that the dots are representative of the waypoints the robot must pass through prior to stopping fully within the red region. The black points on the path represents the waypoints the robot must reach. The points turn green when the robot reaches the previous waypoint.

Hint: you will likely want to use next waypoint as the setpoint for your angular velocity controller and the goal point as the set point for your linear velocity controller.

SCENARIO 6

For Scenario 6, the robot must drive again along a curved road, but one of its wheels is acting up again! The robot will need to be able to handle constant angular noise and stop within the red region.

CODE STRUCTURE

In what files do I need to implementing code?
The file you will be working in is `pid_controller.py` where you will implement a linear and an angular controller.

How do I test my code locally?
To test your code with a single seed, you can run `main.py`. This is very helpful for debugging your code.

- To visualize the scenario, run with the `-v` flag
- To run your scenario slower (useful for debugging purposes), run with the `-r` flag
- To run just a specific scenario, run with the `-s <scenario number (1-6)>`

For example, to run a single, random seed of scenario 6 with visualization on, you would run

```
python3 main.py -v -s 6
```

To rigorously test your code in the same way we will evaluate it in Gradescope, you can run `local_grader.py`. You can use the same arguments when running `local_grader.py` as with `main.py` except that it evaluates 30 seeds instead of just 1. You can use this to evaluate each scenario individually with

```
python3 local_grader.py -s <scenario_number>
```

or all scenarios at once with

```
python3 local_grader.py
```

GRADING

The grader is very precise with regards to final position: the center of the robot's wheels must be within 5cm of the center of the red zone to score.

For each scenario we evaluate your controller for 30 seeds. In each scenario, to get any points, you must pass that scenario at least 29 out of 30 seeds (if you pass all 30, we will only consider your fastest 29). The exact scoring breakdown for each scenario can be seen below:

Scenario 1:

10 × min (1, 480 ÷ average time steps for scenario 1 across best 29 seeds)

Scenario 2:

18 × min (1, 480 ÷ average time steps for scenario 2 across best 29 seeds)

Scenario 3:

18 × min (1, 480 ÷ average time steps for scenario 3 across best 29 seeds)

Scenario 4:

18 × min (1, 500 ÷ average time steps for scenario 4 across best 29 seeds)

Scenario 5:

18 × min (1, 550 ÷ average time steps for scenario 5 across best 29 seeds)

Scenario 6:

18 × min (1, 550 ÷ average time steps for scenario 6 across best 29 seeds)

SUBMISSION TO GRADESCOPE (REQUIRED)

To submit, upload `pid_controller.py` file to Gradescope. Make sure that the `set_velocity` method in the `PidController` class returns a list with `[linear_velocity, angular_velocity]` in the final submission as that is what will be called by the autograder. Otherwise, feel free to make changes as needed.

SUBMITTING TO THE ROBOTARIUM (OPTIONAL)

The Robotarium is a multi-robot testbed here at Georgia Tech that anyone in the world can upload programs to run on real robots for free! This code was created using the Robotarium's simulator which means that you can evaluate your code on real robots if you choose to do so. **Please note that your grade *will not be affected* in any way if you choose to evaluate it on the Robotarium.** Additionally, real robots run very different than how they run in simulation. As a result, there is a high chance that the real world robots will drive off the road after you have fine-tuned your gains for the simulated robots.

If you are interested in seeing how your controller runs on real robot hardware, you can complete the following steps:

1. Go to https://www.robotarium.gatech.edu/sign-in and sign up for an account.
2. Fill out your profile. In the "Reasons for using the Robotarium field", state that you are a current student in CS 3630 and want to run your PID controller for Project 5 on the Robotarium.
3. Wait for your account to be approved (this should take around a day or two).
4. Once you have access, navigate to your Robotarium dashboard and click on "Submit Experiment." Fill out the fields as follows:
    a. Title: CS 3630 PID Controller
    b. Estimated Duration: 240
    c. Experiment Description: Testing a PID controller for following a path in various scenarios with applied noise
    d. Number of Robots: 1
    e. Experiment Files: `main.py`, `pid_controller.py`, `robotarium_main.py`

Make sure to mark `robotarium_main.py` as your main file.

5. Once your experiment has been successfully submitted, it will be in the Robotarium's queue. After it is run, you will be able to view a video of your code running on a real robot for the 6 scenarios!

   From the Robotarium FAQ: The Robotarium operates on a first come first served basis. Depending on what time you submit and the current experiment queue, you can expect results as soon as 10 minutes after your submission (if the queue is empty) or usually within the day (if you submit during operational hours and there is not a large queue).