CS 8803 – O01: Artificial Intelligence for Robotics
Mini Project: PID
Summer 2018

Due: Monday July 9th, Midnight AOE (Anywhere On Earth)

Project Description

The goal of MP: PID is to give you more practice implementing your own Proportional-Integral-Derivative Controller that you learned about me from Lesson 5 on Udacity. The main objective of this project is to write your own PID controllers to solve certain problems. The project is split into two sections, Part A and Part B.

PART A:
All rockets need precise control of the pressure for the feed of liquid oxygen into the engines' turbopumps or it can spell disaster for the launch. Part A will involve maintaining the pressure going into the rocket's turbopumps. In order for the rocket to leave the launchpad, the pumps must be pressurized up to their max level of 100 as quickly as possible. Once there, they must be maintained for the duration of the launch. Therefore, the goal will be to create a simple PD based controller that can make adjustments to the pump's pressure in order to meet these demands. The PD controller will give adjustments that will be in the range of -1.0 to 1.0. Where 1.0 is the max adjustment per time step to increase the pump's pressure, and -1.0 is to reduce the pump's pressure. These adjustments will modify the pressure change rate to the turbopumps. The pressure change rate can operate between -10.0 and 10.0. For example, if for the past 10 time steps the PD controller has given a +1.0 command, then the pressure change rate will now be maxed out at 10.0. This means for every time step the pressure will increase in the turbopump by this amount, 10.0. If the turbopump pressure goes below 0 or above 105, it will result in damage to the engines (and your grade).
For Part A complete the "pressure_pd_solution" in RocketPIDStudent_submission.py. To test your pressure PD controller, see the "test_running_pumps" test case located in "RocketPIDStudent_tester.py
Note: You will not implement the integral part of the traditional PID controller for part A.

PART B:
The second part of the assignment is a slightly more difficult creation of the PID controller for controlling a rocket. It requires your controller be able to maintain the velocity of a simulated rocket ship such that it is able to successfully maintain speeds through different layers of the atmosphere and as different mechanisms act on the vehicle. You will be given an evaluation file that you can use to determine how well your PID controller and parameters are working. Various information about the rocket's flight path and other telemetry can be viewed in a matplotlib output similar to as seen in Lesson 5 in the testing file (if your system has the matplotlib package installed). The controller can only operate the throttle in a range of [0,1] and only acts in the vertical path and only in the upwards direction from the "ground". When enough throttle is applied, the rocket will to ascend and continue to ascend until it runs out of fuel or "lands". The flight plan required for the rocket consists of maintaining a velocity of 0.25 km/s and then 0.5 km/s. Once the rocket has completed it's flight plan, it will be required to descend through gravitational forces and "land" by approaching the ground below a "safe" landing velocity of 0.1 km/s. For Part B, complete "rocket_pid_solution" in RocketPIDStudent_submission.py that satisfies the given constraints. To test your PID controller, use the test case "test_launch" in the RocketPIDStudent_submission.py file to import your throttle controls and test it on a simulated rocket.

Submitting your Assignment

You should complete the "pressure_pd_solution" and "rocket_pid_solution" ifunctions n the RocketPIDStudent_submission.py file to satisfy the specifications described therein (by implementing a PD and PID controller), and then you should submit these this file to the MP: PID Assignment on T-Square.

Do not create an archive (zip,tar, etc) , and only submit RocketPIDStudent_submission.py without changing the file naming. Files that do not follow this format will receive a penalty of 25%. Your submission should use **Python 2.7** and may optionally use external modules such as matplotlib, numpy, scipy, etc that are present in the respective evaluation files. Your python file must execute NO code when it is imported. We encourage you to keep any testing code in a separate file that you do not submit. Your code should also **NOT** display a GUI or Visualization when we import the file or call your function under test. If we have to manually edit your code to comment out your own testing harness or visualization you will receive a 25% penalty.

Testing Your Code

We have provided a testing suite similar to the one we'll be using for grading the project, which you can use to ensure your code is working correctly. These testing suites are NOT guaranteed to be exactly the same as the final grader, but you are guaranteed the points if you satisfy the tester AND the conditions laid out for passing each test. You should ensure that your code consistently succeeds on the test suite before turning it in as we will only run your code once. For the test, you code must complete execution within the proscribed time limit (10 seconds) or it will receive no credit.

The tester can be run from the shell or by using a testing framework in an IDE (like Nose or Py.test).

Grading

Grading for Part A involves seeing a stable pump pressure as quickly as possible. If the pump's pressure drops below 0 or above 105 during any part of the test, it will result in 0 points. Otherwise a score based on how well the pump's pressure is maintained at 100 for the duration of the test will be used. Part A is weighted at 25% of the total grade for the mini-project.

The grading for Part B is broken into two parts: following the optimal velocity course set for the rocket and then allowing the rocket to free fall back to the ground and avoiding excessive speed on landing. If your PID controller fails to produce valid throttle control between [0,1] you may receive zero credit. If your PID controller is able to stay on course for 130 seconds during the simulated flight, you will receive credit for the correct throttling for a total of 65 points. After that if your planner successfully makes a "good" landing, you will receive the remaining credit of 35 points. You will not receive partial credit for a "crash" landing. Part B is weighted at 75% of the total grade.

Academic Integrity

You must write the code for this project alone. While you may make limited usage of outside resources, keep in mind that you must cite any such resources you use in your work (for example, you should use comments to denote a snippet of code obtained from StackOverflow).
You must not use anybody else's code for this project in your work. We will use code-similarity detection software to identify suspicious code, and we will refer any potential incidents to the Office of Student Integrity for investigation.

Moreover, you must not post your work on a publicly accessible repository; this could also result in an Honor Code violation [if another student turns in your code]. (Consider using the GT provided Github repository or a repo such as Bitbucket that doesn't default to public sharing.)