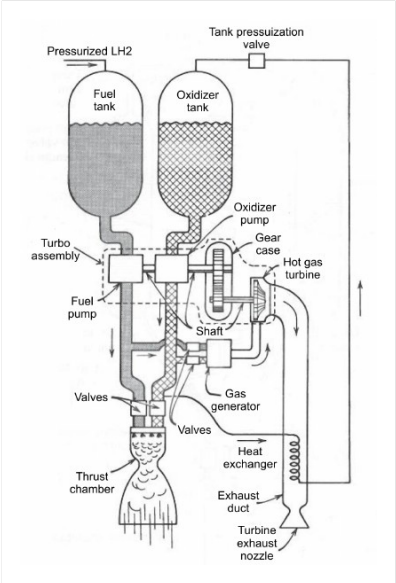
CS 8803 – O01: Artificial Intelligence for Robotics

Mini Project: PID Fall 2018

Due: Monday October 29th, 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 from the PID lesson on Udacity. The main objective of this project is to write your own PID controllers to solve certain problems that are more easily solved using such controllers. The project is split into two sections, Part A and Part B. It is suggested that you use a PD controller to solve Part A and a PID controller to solve part B but any approach that can solve the student testing suite will be accepted. You may also use Twiddle to find an optimal solution, but it is not required. This project requires the SciPy library and you will need the Matplotlib library to get the full benefit of the testing suite (graph output).

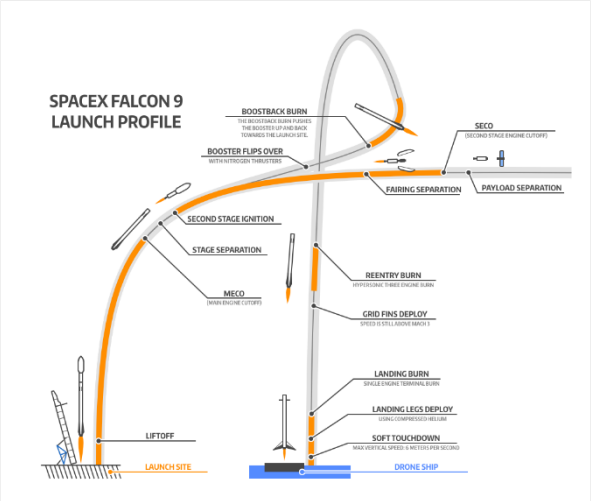


PART A:

All rockets need precise control of the pressure for the feed system of liquid oxidizer and fuel into the engine turbopumps. A failure of engine turbopumps to supply the pressure to meet feed demands of the engine could force an abort of the launch or a disaster after liftoff. The main purpose of Part A will be to solve the problem of maintaining the fuel mixture pressure supplied by 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 output in order to meet pressure 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 to the turbopumps, 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" function in RocketPIDStudent_submission.py. To test your pressure PD controller, see the "test_running_pumps" test case located in "RocketPIDStudent_tester.py

PART B:



The second part of the assignment is a slightly more difficult adaption of the PID controller for controlling a rocket launch and reentry. It requires your controller be able to control the output of rocket engines on a simulated rocket ship such that it is able to successfully maintain speeds through different atmospheric flight regimes and as parameters such as thrust, weight, and air drag are constantly changing. You will be given an evaluation file that you can use to determine how well your PID solution 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 on Udacity in the testing file. The controller can only operate the throttle on the rocket's engines in a range of [0,1] and the rocket engine only pushes the rocket upwards (only gravity pulls the rocket downward). When enough throttle is applied, the rocket will "liftoff" and continue to ascend until it runs out of fuel or "lands". The flight plan required for the rocket consists of maintaining an upward velocity of 0.25 km/s +/- 0.01 for 90 seconds and then 0.5 km/s +/- 0.01 for 40 seconds. Once the rocket has completed its flight plan, it will be required to descend through gravitational forces and NOT by using the rocket (which only act in the upward direction, and can be used to slow the rocket as it approaches the ground). The rocket should then "land" by approaching the ground below a "safe" landing velocity of 0.1 km/s +/- 0.01. 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” functions in the RocketPIDStudent_submission.py file to satisfy the specifications described therein (by implementing a PID controller), and then you should submit the file to the MP: PID Assignment on Canvas. Do not place the file into an archive (zip, tar, etc.), and only submit RocketPIDStudent_submission.py without changing the file name. You may submit any number of times before the deadline and can safely ignore changes in the stored file name Canvass makes to multiple submissions. Your submission should use **Python 2.7** and may optionally use external modules (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. They should also **NOT** display a GUI or Visualization when we import them or call your function under test. If submissions do not follow naming convention or we have to manually edit your code to comment out your own testing harness or visualization you will receive a -20 point 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. You are guaranteed the points on the project you receive from the testing suite 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). You are encouraged to modify your testing suite to aid in your visualization of the solution such as with slider functionally from https://matplotlib.org/api/widgets_api.html#matplotlib.widgets.Slider for PID parameters. (But be sure that your code works correctly with the provided, unmodified testing suite).

Grading

Grading for Part A involves maintaining a stable pump pressure as quickly as possible. If the pump’s pressure drops below 0 or climbs 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 worth 25% of the grade.

The grading for Part B is broken into two parts: following the optimal velocity plan 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, 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 score of 65. Partial credit will be awarded for less than desirable flight control. After that if your planner successfully makes a “good” landing, you will receive the landing score of 35. (For a total maximum of 100 points on part B). You will not receive partial credit for a “crash” landing. Part B is worth 75% of the total grade.

The grader also includes an assert true warning that you may ignore, but it is present to alert you that all conditions of the solution have not been met. Final grading will use the same parameters as in the student testing.

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.)