CS 7638 - AI for Robotics - Mini Project: PID

Summer 2021 - Deadline: Friday July 2, Midnight AOE (8am EST)

Introduction

This projects involves creating a Proportional-Integral-Derivative (PID) controller that:

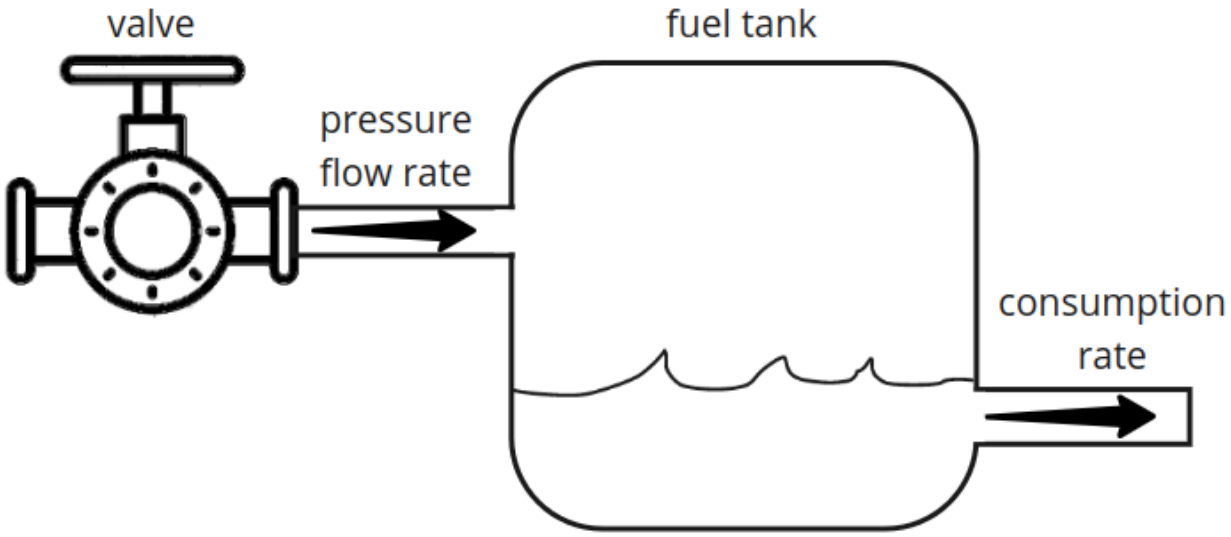
- regulates pressure in a rocket fuel tank
- controls a rocket's thrust

Project Description

The goal of Rocket PID is to give you more practice implementing your own Proportional-Integral-Derivative Controller that you learned about from the PID lesson. 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 three sections, Part A, Part B and Part C. It is suggested that you use a PD controller to solve Part A and a PID controller to solve parts B and C 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 can display a graph using turtle or optionally the Matplotlib library.

PART A:

A rocket engine requires precise control of the pressure inside the fuel tank. A supply system controls the rate at which pressure is increased (or decreased) in the fuel tank. If the fuel tank pressure drops too low (or goes too high) then the rocket will inevitably fail. Your task in part A is to design a controller that will regulate the rate of pressure flow into the fuel tank. In order for the rocket to leave the launchpad, the fuel tank must be pressurized up to its max required level as quickly as possible. Once pressurized, the fuel tank pressure must be maintained for the duration of the launch. The goal is to create a simple PD controller that can make adjustments to the flow rate of pressure into the fuel tank in order to meet the fuel tank pressure demands.



The fuel tank starts with an initial pressure of 10 units. The engine consumes fuel from the fuel tank which causes the pressure inside the fuel tank to drop at a constant rate of 5 units per time step. Regardless of the operating conditions, the pressure inside the fuel tank is always decreasing at the same constant rate. If the flow rate is not adjusted then after 2 times steps the pressure inside the fuel tank will reach 0 units. If the pressure inside the fuel tank drops below 0 units, the fuel tank will implode, causing the rocket launch to end

in catastrophic failure (but more importantly, failure of the test case). Likewise, if the pressure inside the fuel tank goes above 105, the fuel tank will explode.

You can prevent this however, as you have been given control of the supply system for the fuel tank. The supply system has the ability to add (or subtract) pressure from the fuel tank. The maximum amount of pressure that can be added (subtracted) to the fuel tank in a single time step is 10 units (-10 units). Since the supply system is able to add and relieve (remove) pressure from the fuel tank it is a bit of a misnomer (the **supply** system is actually a **supply and relief** system).

You are given control of a valve to operate the supply system. The valve controls the **flow rate** of pressure into the fuel tank. The adjustment of this valve is limited to increments between -1 and +1 each time step (this is an inclusive continuous range). This means if the flow rate of pressure into the fuel tank is $\boldsymbol{x}$ then the range of possible flow rates into the tank after adjusting the valve is [x-1, x+1] (inclusive continuous range). This shows that drastic changes to the flow rate can **not** happen in a single time step: for example, if the flow rate is currently 1 unit per time step into the tank, then you can **not** increase the flow rate to 8 units per time step with a single valve adjustment.

The flow rate of pressure into the fuel tank initially starts at 0 units per time step. Therefore, in order to avoid catastrophic failure (of the test case), the valve **must** adjust the flow rate by +1 for the first 5 time steps. Doing so would prevent the fuel tank pressure from dropping below 0 units. Note that any other value used for the valve adjustment in these first 5 time steps would result in the fuel tank pressure dropping below 0 at time step 4 or 5.

time	valve adjustment	supply system flow rate	consumption rate	fuel tank pressure
start	-	0	-5	10
1	+1	1	-5	6
2	+1	2	-5	3
3	+1	3	-5	1
4	+1	4	-5	0
5	+1	5	-5	0

At each time step, the steps occur in the following order:

1. the valve adjustment is used to update the supply system flow rate
2. the supply system flow rate is used to update the fuel tank pressure
3. the consumption rate is used to update the fuel tank pressure

For Part A complete the "pressure_pd_solution" in RocketPIDStudent_submission.py. To test your pressure PD controller, see the "PressurePDTestCase" test case located in "RocketPID_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 two specific velocity profiles through different atmospheric flight regimes as parameters such as thrust, weight, and air drag change over the time of flight. You will be given an evaluation file that you can use to determine how well your PID solution and parameters are working.

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 (i.e. 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 is set in the Test Cases. Once the rocket has completed its flight plan, it will be required to descend through gravitational forces and NOT by using the rocket engines (which only act in the upward direction and can only be used to slow the rocket's descent on reentry). 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 "RocketPIDTestCase" in the RocketPIDStudent_submission.py file to import your throttle controls and test it on a simulated rocket.

PART C:

The third part of the assignment is an extension of Part B. The rocket is a bi-propellant liquid rocket, which uses a liquid fuel as well as a liquid oxidizer. Both the fuel as well as the oxidizer will be mixed in a fixed ratio, and the excess fuel or oxidizer will be wasted. This part requires two controllers, one to control the output of fuel and another to control the output of oxidizer such that it is able to successfully maintain one specific velocity profile through different atmospheric flight regimes as parameters such as thrust, weight, and air drag change over the time of flight.

The controller can only operate the throttle for the fuel and the throttle for the oxidizer on the rocket's engines, both in a range of [0,1] and the rocket engine only pushes the rocket upwards (i.e. only gravity pulls the rocket downward). When enough throttle for each propellant is applied, the rocket will "liftoff" and continue to ascend until it runs out of fuel or runs out of oxidizer or "lands". The flight plan required for the rocket consists of maintaining an upward velocity of 0.25 km/s +/- 0.01 for 130 seconds for period identified in the velocity profile. Once the rocket has completed its flight plan, it will be required to descend through gravitational forces and NOT by using the rocket engines (which only act in the upward direction and can only be used to slow the rocket's descent on reentry). The rocket should then "land" by approaching the ground below a "safe" landing velocity of 0.1 km/s +/- 0.01.

Note that it differs from Part B in two aspects. One is that here the controller needs to operate the throttle of the oxidizer in addition to the throttle of the fuel. Second is that the flight plan consists of maintaining only one upward velocity unlike Part B which required maintaining two upward velocities.

For Part C, complete "bipropellant_rocket_pid_solution" in RocketPIDStudent_submission.py that satisfies the given constraints. To test your PID controller, use the test case "BiPropellantPIDTestCase" in the RocketPIDStudent_submission.py file to import your throttle controls and test it on a simulated rocket.

Visualization Options

Various information about the supply system, rocket's flight path, and other telemetry can be viewed by enabling graphing using the "–showgraph" argument. By default it will try to use matplotlib if installed or will use turtle graphics otherwise. To specify which version you want to use, "–showgraph turtle OR –showgraph matplotlib".

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. 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). Be sure that your code works correctly with the provided, unmodified testing suite.

Submitting your Assignment

You should complete the "pressure_pd_solution", "rocket_pid_solution", "bipropellant_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 Rocket PID Assignment on Canvas -> Gradescope. 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. Your submission should use Python version 3 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.

Grading

Grading for Part A is based on how well your controller is capable of maintaining a stable fuel tank pressure. If the fuel tank pressure drops below 0 or climbs above 105 during any part of the test, it will result in 0 points. Otherwise your score will be calculated based on how closely the tank's pressure is maintained at 100 for the duration of the test. Part A is worth 25% of the grade.

The grading for Part B is broken into two parts:

- maintaining two specific velocity stages
- 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 a score of 65 which is full credit for that section of part B. Partial credit will be awarded for less than desirable flight control. In the landing section, your controller must make a controlled descent to land under the acceptable velocity for which you will receive the landing score of 35. You will not receive partial credit for a "crash" landing. These two scores will add up to your final score for part B with a maximum possible score of 100 which denotes full credit for Part B. Part B is worth 65% of the total grade for the PID project.

The grading for Part C is broken into two parts: * maintaining the upward specific velocity stage * 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 a score of 65 which is full credit for that section of part C. Partial credit will be awarded for less than desirable flight control. In the landing section, your controller must make a controlled descent to land under the acceptable velocity for which you will receive the landing score of 35. You will not receive partial credit for a "crash" landing. These two scores will add up to your final score for part C with a maximum possible score of 100 which denotes full credit for Part C. Part C is worth 10% of the total grade for the PID project.

Note that 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. Also, the velocity profile provided is simply a guideline and you may use whatever system to define throttling you desire. Final grading will use the same controller parameters as in the student submission.

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, lecture videos, etc)

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.)