ROCKETLAB

GNC Engineer - Neutron Algorithms test

Contents

1 Introduction

1.1 Scope

This take home test is designed to give the candidate the opportunity to demonstrate their capabilities, which are relevant to the role of a Neutron GNC Engineer.

This test focuses on a rocket landing problem, where the only actuator is a thrust vectored rocket engine. The main goals of this test is to demonstrate that the candidate is able to:

- section 2: model the vehicle's motion;
- section 3: design a simple guidance and control algorithm to steer the rocket towards the landing target;
- section 4: assess the algorithm performance and robustness to different scenarios and discuss the results;
- section 5: derive reasonable actuators and sensors requirements;
- section 6: implement a C++ version of the control algorithm designed to run on an embedded computer.

The main questions are marked **Question #** and the candidate should do their best to answer them. Extra questions marked Optional Question # are to be addressed last. These are optional questions and not strictly required for a successful submission. Once the main questions of the test have been answered and documented the extra questions can be answered.

1.2 Guidelines

The candidate must respect the following guidelines.

- The purpose of this test is to reflect the true abilities of the candidate. **Therefore, true personal work with no external help is fundamental in this exercise.** Far more value will be attributed to a submission with seemingly less successful results, but reflecting personal thinking and self-made content rather than a more complete but impersonal submission.
- **The test is designed to be completed including documentation within 48 hours after receiving the material and therefore has to be submitted within that timeframe.**
- The candidate can use either Python or MATLAB for the sections 3 and 4 and C++ for section 6.
- The candidate must include all code and resources used for the submission. The use of any non-publicly available code should be avoided, or at a minimum be included in the submission for the recruiter to reproduce the results. MATLAB toolboxes are accepted but should not be required. Simulink is accepted.
- This test is designed to be relatively simple to solve; it is well known and documented in the literature. Reading too much about it is discouraged; re-using a complex project already

solved in the past will be less valued than following the flow of the test and implementing things from scratch.

- **Generative AI tools should not be used for the submission.** Much more value will be given to personal and concise writing than generic AI-generated lengthy paragraphs. Any mismatch between the test quality and the demonstrated knowledge of the candidate during the interview will be obvious.

1.3 Tips

Here are additional tips to help the candidate in their submission.

- The choice of all control techniques used should be clearly justified. **Advanced control theory concepts can be applied, but are not required to successfully complete the test.** They are only expected to be useful in the Optional Questions.
- This test is designed to show your way of thinking and the approaches you take to tackle engineering problems. Thus, explanations of your design choices and documentation of the difficulties encountered or even dead ends, are highly encouraged in the submission.
- Code optimality and performance is of no concern here, a submission with code that is un-optimized but nicely architectured, documented and easy to review is preferred.
- Effective communication is a highly regarded skill at Rocket Lab and is key to our renowned efficiency. Hence, a concise and effective writing style is preferred.

2 Problem formulation and modelling

This section deals with the first step of the rocket landing control problem: modelling the physics of the vehicle.

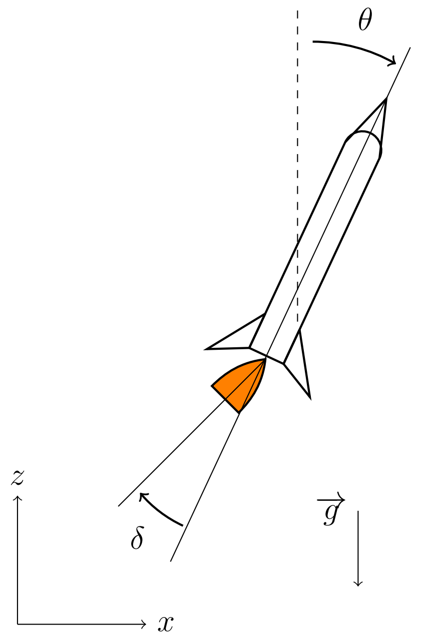
2.1 Problem statement

The vehicle motion is assumed to be described by only 3 Degrees of Freedom (3DoF), as follows:

- x: lateral position
- z: altitude
- θ: attitude angle

The rocket is actuated by a single engine mounted on a gimbal point at its lower end. The thrust can be oriented with a Thrust Vector Control system (TVC). The angle of the thrust vector with respect to the rocket body is noted by δ.

Here is a drawing describing the conventions used for the rotation angles:



Additional assumptions:

- the Earth is flat;
- gravity is uniform at 9.81 m/s^2 along the z axis;
- the mass, inertia and centre of mass of the rocket are considered constant during the whole landing phase: thrust is "magically" created without using propellant to simplify simulation modelling;
- no aerodynamic forces are to be considered.

The mass properties of the rocket are:

- J: rotational inertia, 7.5e5 kg.m^2
- m: mass, 5.0e3 kg
- Centre of mass is located 3.0 m up from the engine nozzle gimbal point

2.2 Model implementation

Question 1 Formulate the equations of motion for this system and implement their time-based propagation.

Question 2 Demonstrate your modelling works as expected on an example case.

3 Controller implementation

Using the propagation model from the previous section, this section defines the guidance and control algorithm requirements to land the rocket straight on a chosen target, with the lowest sink rate possible at touchdown.

3.1 Assumptions

Some assumptions are made:

- the controller can control TVC angle and thrust with no restriction;
- the controller has access to the full state of the vehicle; as if there was a perfect observer in the control loop;

Note: *As a time management advice, it is strongly suggested to implement the simplest solution that meets the landing requirements.*

3.2 Requirements

The controller shall be able to land the rocket in the situation defined below.

The initial state of the rocket is defined as follows, including dispersions:

- lateral position $x = 0 \pm 5$ m;
- altitude $z = 150 \pm 10$ m;
- lateral speed $\dot{x} = 0 \pm 1$ m/s;
- vertical speed $\dot{z} = -10 \pm 1$ m/s;
- attitude angle $\theta = 0 \pm 5$ deg.
- angular rate $\dot{\theta} = 0$ deg/s.

The rocket shall land at the origin $(0, 0)$ with the following accuracy:

- position x within ± 3 m;

- lateral velocity $\dot{x}$ within ± 0.5 m/s;
- attitude angle θ within ± 2 deg;
- angular rate $\dot{\theta}$ within ± 1 deg/s;

As mentioned earlier, no fuel consumption is to be modeled here. However, to represent a limited amount of fuel available on board, **the rocket should reach the target within 35.0 seconds.**

3.3 Algorithm Implementation

Question 3 Implement an algorithm of your choice that can steer the rocket towards the given landing point and that meets the requirements. Explanations on the choice of controller architecture and gains are expected.

4 Robustness and sensitivity analyses

At this stage, the candidate should have the ability to land the rocket within the provided requirements.

4.1 Initial conditions expansion

Question 4 How much can the set of initial conditions be expanded while still be able to land within the requirements? E.g. how much further away can you initiate the powered descent? At which speed? With which tilt angle? Discuss.

4.2 Path to better performance

Question 5 What are the key limiting factors preventing the controller to land the rocket with a larger subset of initial states, or a better accuracy? What type of Guidance and Control (G&C) architecture would be better?

Optional Question 1 Provide a full G&C stack for the architecture that you suggested. If enough time is still available, attempt to implement it in the simulation, and compare the results with the previous implementation. That may include some trajectory planning, or advanced control techniques.

5 Actuators and sensors requirements

5.1 Actuators

So far no assumptions have been made on the actuators present on the rocket in the simulations. However, real rockets have actual engines as real actuators, and they need to be manufacturable while staying lightweight.

This means that the GNC engineers need to determine requirements for the subsystems they interact with, guaranteeing performance of the rocket GNC systems.

Question 6 Derive and justify *reasonable* requirements for:

- TVC min/max angles.
- Thrust min/max value.

Question 7 Implement those limits in your modelling and show how much this impacts your landing performance.

Mechanical and propulsion systems like TVC systems and rocket engines obey rules of physics and also have their own internal dynamics.

Optional Question 2 Assuming the TVC actuators are modelled with a second order model, derive a reasonable -3dB bandwidth and damping that do not impair drastically your controller performance and robustness. Implement this second order model in the simulation, and compare the new simulations with the previous results.

Optional Question 3 What is the minimum -3dB bandwidth value that would be incompatible with the landing requirements, regardless of the employed algorithm? How would you prove that this value is indeed physically limiting?

5.2 Sensors

The controller was assumed to have perfect knowledge of the vehicle state in position, velocity and attitude. This is not realistic: measurements have noise, bias or are sometimes only indirectly observable.

Question 8 Break down the different state variables and give examples of sensors that could be used, or alternatively examples of indirect observation method that would provide suitable estimates.

Optional Question 4 Using your simulation setup, study the impact on estimation errors on vehicle state, mass properties and thrust estimation. What are the most important variables to get right?

6 Embedded algorithm implementation

As a GNC Engineer, you will be responsible for writing software that includes the core control algorithms used by the rocket, which follow the definition of "flight critical software". One bug, floating point exception, untested branch or memory error can very easily cause a mission failure, or worse, damage to launch infrastructure or the surrounding environment.

For this reason, GNC Software is architected, written, documented, tested and reviewed to the highest standards to ensure it works all the time, exactly as intended. It also needs to satisfy constraints to run on an embedded flight computer in real time to ensure the rocket is controlled at all times.

Question 9 Re-implement the control algorithm you developed in section 3 but this time in C++.

Additional tips and constraints for this C++ version:

- you should use an object-oriented approach;
- think about methods, encapsulation, maintainability, testability;
- your implementation needs to satisfy all constraints you deem necessary to successfully and safely run on an embedded flight computer: list them.

Question 10 Write unit tests and provide proof showing that your C++ implementation is performing as expected, matching your MATLAB/Python implementation.