ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Department of Electrical, Electronic and Information Engineering, "Guglielmo Marconi"

Master's Degree in

Automation Engineering

Development of a sequential learning and control scheme for a quadrotor altitude model

Supervisor

Prof. Giuseppe Notarstefano

Defended by

Simone Cenerini

Co-supervisors

Ing. Janani Venkatasubramanian
Dr. Simone Baroncini
Prof. Dr.-Ing. Frank Allgöwer

Graduation session IV - July/2024

Academic Year 2023/2024

Abstract

The contribution of this thesis lies in the field of system identification techniques, which require estimating the parameters of a dynamical system from data collected during experiments. This is essential in many fields of engineering since, in real-life experiments, the perfect knowledge of the coefficients related to the equations describing the dynamics of the system is not always available. Specifically, this thesis investigates a recent system identification technique in the presence of non-stochastic energy-bounded disturbances, to obtain a desired error bound on the estimated parameters. The first objective of this thesis lies in developing a Software In the Loop (SIL) simulator to test this technique based on optimization problems, on a quadrotor model. This SIL simulator is based on an existing simulator (RotorS developed by ETH Zurich) running on ROS infrastructure, adapted to collaborate with MATLAB where the optimization techniques are implemented. Moreover, the quadrotor model is derived, and a pre-stabilizing controller is implemented to conduct the desired experiments without accounting for the unstable nature of the quadrotor. Finally, the techniques to generate the targeted exploration signal for the system identification are implemented and tested only on the altitude model of the quadrotor. The results of the simulation show that the exploration signal generated by the optimization techniques is able to reduce the uncertainty of the model and achieve the desired accuracy. This thesis is a first step toward implementing a robust control strategy based on the desired uncertainty obtained from the system identification techniques known in the literature as dual control.

Acknowledgements

The results presented in this thesis are the outcome of my activity at the Institute for Systems Theory and Automatic Control (IST) at the University of Stuttgart. There are numerous people who supported me and to whom I would like to express my gratitude.

First of all, I would like to express my special thanks to my academic supervisor Prof. Giuseppe Notarstefano and co-supervisor Dr. Simone Baroncini for their important support and guidance. Furthermore, I am grateful to the people who have supported me during the development of this thesis in Stuttgart. First of all Ing. Janani Venkatasubramanian for the guidance throughout the whole project. Moreover, I am grateful to Prof. Dr.-Ing. Frank Allgöwer for having hosted me at IST, giving me the possibility of working in his institute during the last months. This has allowed me to gain great experience in a different country, as well as to experience the world of dynamical systems control.

Finally, I would like to thank family and friends for supporting me with love all these years. Their support was more than important, it has been fundamental. They inspired me, encouraged me, and helped me follow my way.

July 2024

Simone

Contents

Introduction

Motivations

The modeling of a dynamical system is fundamental in many fields of engineering, especially in control theory. Accurate models are important to understand and predict the behavior of a system. In particular, in the automatic control field, they are used to properly design a regulator to drive the system toward desired performances. In real-life experiments, the perfect knowledge of the coefficients related to the equations describing the dynamics of the system is not always available. Especially when the dynamic equations of the system are simplified to have linear relations between variables, the new coefficients, even if representing some known constants, are modified to account for some nonlinear effects that are not described by the linearized model. For this reason, the scope of this thesis is to implement an existing method [1] for system identification and test it on a quadrotor model. This technique ensures a desired error bound on the estimated parameters given a sufficient condition on the exploration signal and without stochastic assumptions on the disturbance acting on the system. This uncertainty bound makes it possible to design a robust control goal. Additionally, this targeted exploration technique does not excite the system arbitrarily but rather in such a way that the information gathered is useful for achieving the goal. In real-world applications, this is essential for not compromising the safety and reliability of the system.
In general, the linearized dynamics of a quadrotor model are described by six differential equations representing the evolution of the position and the orientation of the quadrotor in space. For the sake of simplicity, the model is reduced where the system identification techniques are applied, testing it only on the altitude model, describing the motion of the quadcopter with respect to the vertical direction.

Literature

The study of multi-rotor helicopters has experienced remarkable development over the last decade and has been applied in military and commercial fields [2], [3]. Due to the unstable nature of the quadrotor, various techniques of pre-stabilization and control have been widely studied [4]–[7].
To test and study these techniques, several simulation environments have been

developed [8]–[10], where it is possible to study the behavior of the quadrotor and simulate it with high accuracy. Even in simulation, it is necessary to identify the coefficients that are correlated to the dynamic equations used to study the evolution of the quadrotor involved in simulation. Several techniques of system identification exist depending on how the model is represented and how the data are collected. In the presence of energy-constrained noise, a historical study is [11], which is used in [1] to design a targeted exploration strategy. In particular, this targeted exploration input is designed such that the consequent reduction of the model uncertainty can ensure a desired accuracy and the achievement of a desired control goal, as studied in [12]–[17]. This technique is known in the literature as dual control, and a direct use of [1] is present in [16].

Contributions

The main contributions of this thesis lie in the analysis and implementation of the system identification method [1] on an altitude model of a quadrotor. To achieve this result, the necessary infrastructure to perform the tests was developed, including the simulation environment, control framework, and optimization strategy needed to achieve the targeted exploration signal.
The linear dynamical model describing the evolution of the quadrotor during hovering conditions has been developed, and the control framework has been implemented to stabilize the attitude dynamics of the system, thereby enabling the exploration tests on the altitude dynamics. A Software In the Loop (SIL) simulator has been realized to execute the optimization techniques in [1] without constraints on execution time. Starting from an existing simulator (RotorS, developed by ETH Zurich) running on the ROS infrastructure, it has been adapted to collaborate as a SIL simulator with MATLAB, where the optimization techniques are implemented. One of the adaptations involves a dynamic estimator that allows the SIL controller to evaluate the dynamic evolution of the quadrotor for an almost fixed discrete time. Finally, once the optimal exploration signal is generated, it is tested, and the results analyzed, demonstrating the achievement of the desired accuracy. A comparison with another exploration signal, generated by a Gaussian stochastic process, has been performed at the end, providing further insights into the effectiveness of the approach.

Organization

This section provides the reader with a guide to the organization of the thesis, showing where the previously mentioned themes are located.
In Chapter 1, the technical preliminaries are presented. The quadrotor model is described, and its control structure is derived. The simulation environment is

explained, starting from the explanation of the existing RotorS simulator to how it is adapted to perform as an SIL simulator. Finally, the system identification technique [1] is presented.
In Chapter 2, the methodology is presented. The problem formulation is described based on the explanations achieved in the previous chapter. The altitude model setup is presented, and the design of a stabilizing controller for the attitude dynamics is performed. Finally, the exploration strategy explained at the end of Chapter 1 is implemented and tested.
In Chapter 3, the results of the simulation are presented, analyzing the achievement of the objectives explained at the beginning of Chapter 2. Finally, a comparison between different exploration signals is presented.

Chapter 1

Technical Preliminaries

The aim of this chapter is to provide the reader with the necessary background required to properly understand the following chapters of this thesis. First, the quadrotor model is presented, followed by a detailed description of the simulation setup, mainly focused on the interconnection between MATLAB and Gazebo tools. Finally, the necessary mathematical background (strongly inspired by [1]) is presented and explained.

1.1 Quadrotor Model

In this section, the quadrotor dynamical model and the linearized equations around a specific equilibrium point are presented. At the end, the general control structure adopted for the quadrotor will also be described (for further details see [4]–[7], [18]).

The multi-rotor helicopter is a special kind of Unmanned Aerial Vehicle (UAV) which has experienced remarkable development over the last decade. The most commonly used and studied multi-rotor helicopter is the quadcopter, also known as a quadrotor. This family of UAVs has been widely applied in both military and commercial fields. Quadrotor helicopters have hovering and vertical takeoff and landing (VTOL) capabilities, which are also characteristics of conventional helicopters. However, the quadrotor is an under-actuated system with six degrees of freedom (three translational and three rotational) but only four independent inputs (the rotational speed of each propeller). This results in a very strong coupling between rotational and translational dynamics. In what follows, the rotor working principle is described in detail.

1.1.1 Rotor Working Principle

As highlighted before, the quadrotor utilizes four rotors as direct power for flight. These rotors have the same structure and are placed in a symmetric configuration, as can be seen in Figure 1.1. The propellers rotate in clockwise and counterclockwise directions in pairs; the two propellers on the same axis (opposite each other) rotate in the same direction, while the other two rotate in the opposite direction.

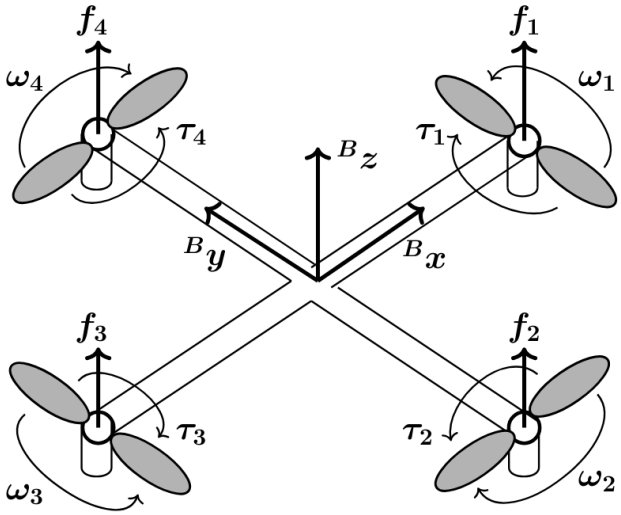

Figure 1.1: The structure of a quadrotor [19, Figure 1]

This configuration allows the quadrotor to control the yaw (ψ) angle (rotation around the z-axis) produced by the drag force of the propellers. As seen in Figure 1.1, the reaction torques are caused by the propeller's rotation, and their directions are exactly opposite to the rotor's rotation directions.

As it is well known in aerodynamics, the lifting force f_i and the reaction torque τ_i, generated by the i^{th} rotor, are proportional to the square of the rotor angular velocity:

$$\begin{aligned} f_i &= k_f \omega_i^2 \\ \tau_i &= k_m \omega_i^2 \end{aligned} \tag{1.1}$$

where k_f and k_m are the force and torque coefficients, while ω_i is the angular velocity of the i-th rotor. For the hovering of the quadrotor, all the propellers rotate at the same (hovering) angular velocity to counterbalance the gravitational acceleration. Thus, the quadrotor performs stationary flight and no forces or torques move it from its position. The altitude of the quadrotor, representing the vertical position of the origin of the body-fixed reference system with respect to the inertial ground-fixed reference system, is governed by the lifting force of the four rotors, generating a total thrust T. The attitude, defined by the Euler roll (ϕ) and pitch (θ) angles obtained by the rotation of the quadrotor about the x and y axes of the body-fixed reference system, can be determined by the torques τ_ϕ and τ_θ. Thus, in accordance with Figure 1.1, it can be expressed as

$$\begin{aligned} T &= f_1 + f_2 + f_3 + f_4 \\ \tau_\phi &= l\,(-f_2 + f_4) \\ \tau_\theta &= l\,(-f_1 + f_3) \\ \tau_\psi &= -\tau_1 + \tau_2 - \tau_3 + \tau_4 \end{aligned} \tag{1.2}$$

where l is the moment arm, namely the distance between the propeller and the origin of the body reference system. From the sets of equations (1.1) and (1.2),

it is possible to obtain the non-linear matching between the four rotor speeds w_i and the variable U, henceforth referred to as the control input, thus

$$U \coloneqq \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \underbrace{\begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & -l \cdot k_f & 0 & l \cdot k_f \\ -l \cdot k_f & 0 & l \cdot k_f & 0 \\ -k_m & k_m & -k_m & k_m \end{bmatrix}}_{\coloneqq K} \cdot \underbrace{\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}}_{\coloneqq \Omega^2} \tag{1.3}$$

where the matrix K is nonsingular as long as $l\, k_f k_m \neq 0$.

1.1.2 Dynamic Equations

The motion of the quadrotor is described by the Newton-Euler equations, which are a set of Ordinary Differential Equations that describe the translational and rotational motion of a rigid body. Because of the four inputs and the six degrees of freedom, the quadrotor is an under-actuated nonlinear system. The rotational motion is fully actuated by the three torque-components (τ_ϕ , τ_θ , τ_ψ) components of the control input U and it is independent of the translational motion. The translational motion, on the contrary, is under-actuated and it is governed by the total thrust (T) component of the control input U and by the rotational motion.

The following quadrotor motion equations can be obtained using the Newton-Euler formalism without considering the rotor dynamics during the model definition [5, Eq.(12)]:

$$\begin{cases} \ddot{\phi} = \left(\dfrac{I_y - I_z}{I_x}\right) \dot{\theta}\dot{\psi} - \dfrac{J_r}{I_x}\dot{\theta}\Omega_r + \dfrac{1}{I_x}U_2 \\ \ddot{\theta} = \left(\dfrac{I_z - I_x}{I_y}\right) \dot{\phi}\dot{\psi} + \dfrac{J_r}{I_y}\dot{\phi}\Omega_r + \dfrac{1}{I_y}U_3 \\ \ddot{\psi} = \left(\dfrac{I_x - I_y}{I_z}\right) \dot{\phi}\dot{\theta} + \dfrac{1}{I_z}U_4 \\ \ddot{x} = \dfrac{U_1}{m}(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \\ \ddot{y} = \dfrac{U_1}{m}(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi) \\ \ddot{z} = \dfrac{U_1}{m}(\cos\phi \cos\theta) - g \end{cases} \tag{1.4}$$

where I_x, I_y, and I_z are the moments of inertia of the quadrotor, characterized by a diagonal inertia matrix due to the choice of the reference system coincident with the principal axes of inertia of the quadrotor itself. The second term of the first and second equations in (1.4) represents the gyroscopic effect caused by the inertia of the rotors J_r and the relative speed $\Omega_r = \omega_1 - \omega_2 + \omega_3 - \omega_4$. Additionally, m is the mass of the quadrotor and g is the gravity acceleration.

It is possible to notice that the dynamic equations of the quadrotor are highly non-linear and strongly coupled. To have a tractable solution from the control

point of view, a linearization around an equilibrium point is performed. To achieve a simplified mathematical model and decoupled dynamics, some prior assumptions are made:

Ass 1. The quadrotor is in hovering condition, and this corresponds to the definition of the desired equilibrium point. By considering small attitude angles and linear velocity, the following can be assumed:

$$\phi \approx 0,\ \theta \approx 0,\ \dot{x} \approx 0,\ \dot{y} \approx 0.$$

Ass 2. The linear motion along the x-y plane is mutually exclusive with the linear motion along the z-axis; thus, any non-zero velocity along one requires zero velocities along the other.

Ass 3. The body frame of the quadrotor is rigid, and the axes of the frame are the principal axes of inertia.

Ass 4. The center of gravity of the quadrotor coincides with the origin of the body reference system.

Ass 5. The quadrotor leans towards the direction of the slow-spinning rotor, making gyroscopic effects negligible.

Under these assumptions, the linearized dynamical equations around the hovering equilibrium point result in [4, Eq.(13)]:

$$\begin{cases} \ddot{\phi} = \dfrac{1}{I_x} U_2 \\ \ddot{\theta} = \dfrac{1}{I_y} U_3 \\ \ddot{\psi} = \dfrac{1}{I_z} U_4 \\ \ddot{x} = \dfrac{1}{m} U_1(\theta) = g\theta \\ \ddot{y} = \dfrac{1}{m} U_1(-\phi) = -g\phi \\ \ddot{z} = \dfrac{1}{m} U_1 - g \end{cases} \tag{1.5}$$

Therefore, by defining the state vector as

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^\top \in \mathbb{R}^{12},$$

and the control input as

$$U = \begin{bmatrix} U_1 & U_2 & U_3 & U_4 \end{bmatrix}^\top = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^\top \in \mathbb{R}^4.$$

The state space representation of the linearized dynamical model can be expressed as

$$\dot{X} = A\,X + B\,U - g\,\hat{e}_6\,,$$

with $U = K \cdot \Omega^2$ and where $\hat{e}_j$ is the j^{th} canonical basis vector. No output equation is present and it is assumed that the states are directly measurable. The state and input matrices appear as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}.$$

1.1.3 Control Structure

According to the under-actuated nature of the quadrotor, the control of the translational motion is coupled with the rotational one. Due to the input matching of the rotor dynamics (1.3), the linearized dynamic equations result to be partially decoupled.

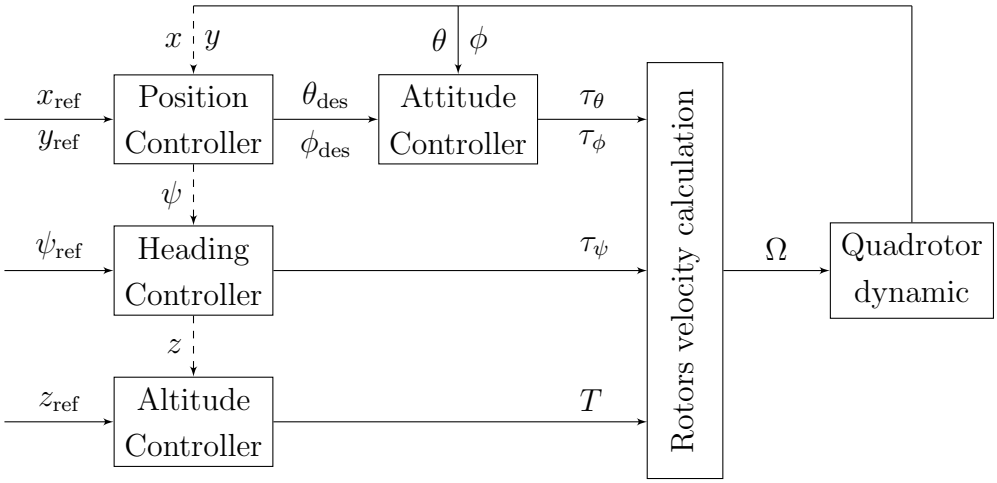

Figure 1.2: Quadrotor control structure

Therefore, a control structure with inner and outer loops is designed (Figure 1.2). The inner control loop, named *Attitude controller*, ensures asymptotic stability of the attitude dynamics, which is also necessary to satisfy the first assumption of the linearization. The outer loop, indicated as *Position, Heading and Altitude controller*, manages the navigation of the quadrotor by following the given reference position $[x_{\text{ref}}, y_{\text{ref}}, z_{\text{ref}}, \psi_{\text{ref}}]$. Finally, these controllers generate the control input

$U = [T, \tau_\phi, \tau_\theta, \tau_\psi]$, which is then converted into the angular rotors velocities $\Omega = [\omega_1, \omega_2, \omega_3, \omega_4]$ by the non-linear input matching block indicated as *Rotors velocity calculation* in Figure 1.2. This conversion is performed using the inverse of the matching input matrix K^{-1}, which is a non-singular matrix, and the square root of the velocities.

1.2 Simulation Environment

In this section, the Software In The Loop (SIL) simulator is explained. The Robotic Operating System (ROS) and its Gazebo physical simulator are briefly described, together with the correlation of these two components with the RotoS simulator. At the end of the chapter, it will be presented how this simulator has been adapted to collaborate with MATLAB and to behave as a SIL simulator [8], [10], [20], [21].

1.2.1 Robotics Operating System (ROS)

Robot Operating System (ROS) is an open-source middleware framework specifically designed for robotics research and development applications [21]. Despite the name, it is not a real operating system but a collection of libraries, tools, and conventions that allow robotic developers to share, reuse, and manage applications. Being a communication and middleware layer, it serves as an important intermediary layer between the operating system and software applications.

One of the key features of ROS is its modular architecture, which promotes code reuse and modularity by breaking down robot applications into smaller, reusable software components called "nodes". A node is an individual specific and independent program written in Python or C++ that executes a specific task or function within the framework of a more complex application. The nodes can communicate with each other using a publish-subscribe messaging system, allowing for flexible and decentralized control architectures. In particular, some nodes can publish messages to topics (publishers) and other nodes can subscribe to these topics to receive and process the data (subscribers). Another way for the ROS nodes to communicate is by using the services. They are specifically used for synchronous and request-response interactions rather than continuous data exchange, as it was for the topics. In particular, a node (called client) sends a request to a service and waits for a response while another node (called server) handles this request, performs a specific action and sends back a response.

1.2.2 Gazebo

Gazebo is a free, open-source robot simulation environment. The project is run by Open Robotics, the same group looking after ROS. However, the projects are managed separately and Gazebo is not a "part of" ROS. With Gazebo, a virtual world

can be created and simulated versions of our robots can be loaded into it. Simulated sensors can detect the environment and publish the data to the same ROS topics as real sensors would, allowing easy testing of algorithms. Then, external forces can be applied to the simulated actuators of the robot, taking into account complex physical phenomena like friction. Gazebo uses the Open Dynamics Engine as one of its physics engines to simulate the dynamics of rigid body systems. When Gazebo simulates a physical scenario, it translates the physical properties and interactions of objects into a set of Ordinary Differential Equations (ODEs). In the following, a high-level overview of how this process works is presented:

Model Definition: Gazebo models are defined using the Simulation Description Format (SDF). The SDF file specifies the physical properties of each object, such as mass, inertia, and friction, as well as their geometric properties and initial conditions.

Physics Engine Initialization: When a simulation starts, Gazebo initializes the physics engine based on the specified parameters in the SDF file. For Open Dynamics Engine, this includes parameters like the solver type, iterations, time step size, and other constraints.

Equation Setup: The physics engine sets up the ODEs that govern the motion of each rigid body. This involves defining equations for translational and rotational motion based on Newton's laws of motion (in the context of the thesis these are similar to Equation (1.4)). For each rigid body, the engine calculates forces and torques resulting from gravity, joint constraints, collisions, and other applied forces (like Equation (1.2)).

Solver Execution: The Open Dynamics Engine solver iteratively solves these differential equations at each time step. It computes the next state (positions, velocities, orientations) of each body by integrating the forces and torques over time. The solver can use various methods like Euler integration or more complex iterative solvers, depending on the required accuracy and computational resources.

Constraint Handling: Constraints such as joints and contact forces are incorporated into the system of equations. The Open Dynamics Engine handles these using methods like Lagrange multipliers or penalty methods to ensure that the constraints are satisfied throughout the simulation.

State Update: After solving the ODEs, the physics engine updates the state of the simulation world, including the new positions and orientations of all objects. These updates are then used to render the next frame in Gazebo's graphical interface.

This process allows Gazebo to accurately simulate complex interactions between multiple bodies in real-time, providing a robust platform for testing robotics algorithms and performing physics-based simulations.

1.2.3 RotorS

Rotor Simulator (RotorS) is a modular Gazebo Micro Aerial Vehicle (MAV) simulator framework, developed by the Autonomous Systems Lab at ETH Zurich [8]. It enables a quick start to perform research on MAVs, to reduce field testing times and to separate problems for testing, making debugging easier, and finally reducing undesired crashes of real MAVs. The simulator is designed in a modular way, such that different controllers and state estimators can be used interchangeably, while incorporating new MAVs is reduced to a few steps.

An overview of the main components of the RotorS simulator is shown in Figure 1.3. All components found on real MAVs are simulated by Gazebo plugins and

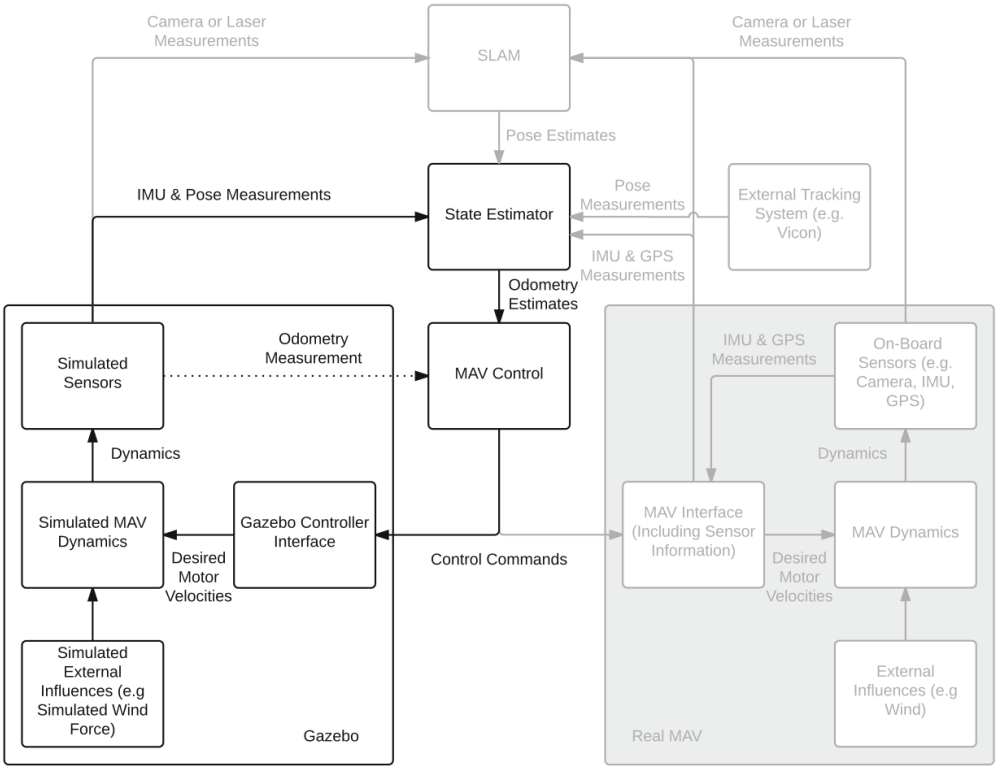

Figure 1.3: MAV simulator framework [8, Figure 2]. All the *black* parts are covered by the RotorS simulator and are available open source, while the real MAV structure can be compared to the *right* part.

by the Gazebo physics engine (Open Dynamics Engine). A MAV consists of a body, a fixed number of rotors (actuators), which can be placed at user-specified locations, and some sensors attached to the body. Each rotor has motor dynamics and accounts for the most dominant aerodynamic effects. In order to facilitate the development of different control strategies, a simple interface is provided. The control input acts on the desired rotor speeds, while the output is the odometry of the MAV, which means position, orientation, linear, and angular velocity. To simulate realistic conditions, a noise model for the applied sensors is implemented, which motivates the presence of a state estimation block. The estimation is crucial for real MAVs and it is used to obtain information about the state of the MAV

at a high rate. In the simulation, this part can be replaced by a generic (ideal) odometry sensor directly provided by a Gazebo plugin. It mimics any generic on-board or off-board tracking system such as a Global Position System (GPS) or Vicon, respectively.

Regarding the Gazebo block, to compute the forces and moments that are acting on the MAV, the forces and moments are split into the respective forces and moments acting on each rotor, and the gravitational force acting on the Center of the Gravity (CoG) of the MAV. The forces and the moments acting on the rotors are computed locally by RotorS just for the propellers, then they are sent to Open Dynamics Engine within the rest of the simulation state to compute the dynamics. RotorS generates the mapping between the rotor dynamics output and input, as described by Equation (1.1). The rotor dynamics output is the lifting force f and the reaction torques τ generated by the propeller, while the input is the angular velocity of the rotor squared ω^2. However, RotorS simulates these dynamics with high accuracy, not only according to (1.1), but also by taking into account the drag force and the rolling moment originated from the drag of the rotor blade. This new force and moment are proportional to the angular velocity ω of the propeller and not to the square of it, as for the other [8, Eq.(4)].

Following the description of the RotorS simulator, the next step is to focus on some components that have been used for the thesis work. As highlighted before, in simulation an ideal odometry sensor is used, which is a generic sensor that provides the odometry of the MAV. It is important to notice that it is placed at the center of the reference frame of the simulated MAV. Since it is represented as a link, it must have a certain weight (at least $10^{-5}kg$); otherwise, it gets omitted by the physical engine. This will result in not being able to find the link by the plugins.

Given the forces and the torques acting on the rotors, it is important to explain how to generate the structure of the robot describing a generic MAV. The robot is described using a Unified Robot Description Format (URDF), with a macro language (Xacro) of Extensible Markup Language (XML), which is used to generate more readable and often shorter XML code. Internally, Gazebo converts the URDF files to SDF files that are used by the physical engine.

During the experiments developed in the context of this thesis, the Hummingbird model quadcopter was used (one of the four MAVs available into RotorS) with the odometry sensor attached to the CoG.

1.2.4 Software In The Loop Simulator

Given the explanation of the framework of the RotorS simulator, the next step is to explain how this simulator was adapted to collaborate with MATLAB and to work as a Software In The Loop (SIL) simulator. During the thesis work, the necessity of interfacing the RotorS, usually working on ROS, with MATLAB was

required due to the implementation of the optimization algorithm (explained in Section 1.3) that was performed on MATLAB.

First, it is important to explain what a SIL simulator is. Software In The Loop is a simulation technique that allows developers to test their software on a host computer rather than on the target hardware. This allows testing and validating the software's functionality, performance, and interaction with other components in a simulated environment before deployment on the physical system. In this case, RotorS is used as a simulated environment taking into account only the *Gazebo* part of the framework in Figure 1.3, while the *MAV Control* part is substituted by the control algorithm implemented in MATLAB. As explained, the *State Estimation* part is omitted, using directly the ideal simulated odometry sensor.

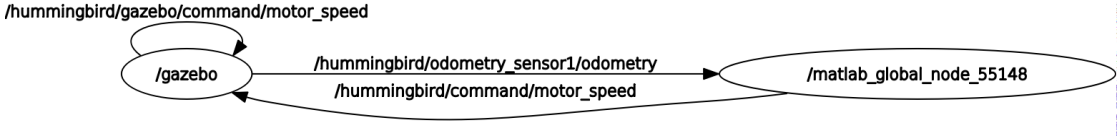

Figure 1.4: Graph of ROS nodes and topics of the SIL simulator

In Figure 1.4, a drawn output of the `rqt_graph` command of ROS when the SIL simulator is running is shown. This tool gives an overview of all ROS nodes that are running, and the topics on which the nodes are communicating. Gazebo is only shown as one ROS node, but internally all the Gazebo plugins are running, such as the odometry sensor and individual motors that are mounted on the frame. The node `/matlab_global_node_55148` is the interface with the SIL simulator running on MATLAB thanks to the Robotics System Toolbox. These two nodes are subscribed to the topics `/hummingbird/odometry_sensor1/odometry` and `/hummingbird/command/motor_speed`, to exchange sensor data and control input, respectively. Finally, due to the use of the service of ROS, the MATLAB node is able to pause and unpause the Gazebo node simulation, in order to guarantee a constant discrete time step of the simulation while also getting the simulation time. Since it is a SIL simulation, the MATLAB node has no real-time constraints. Once the control input is computed and published on the topic `/hummingbird/command/motor_speed`, the Gazebo node will run for a controlled discrete time and uses the topic `/hummingbird/gazebo/command/motor_speed` to have a constant input during the simulation. Once the simulation is unpaused by the MATLAB node, the new state of the system is read from the topic `/hummingbird/odometry_sensor1/odometry`.

Time Management

Before presenting the structure of the simulator, it is important to explain how the time management is handled. As presented before, the simulator (more precisely the MATLAB node) is able to pause and unpause the Gazebo node performing

the dynamic simulation, by means of a ROS service. The services, within the ROS infrastructure, require a non-zero and stochastic time (dT_{ROS}) to be executed. This time, added to a control input time (dT_{ctrl}) of the SIL simulator, can lead to an almost constant discrete time step (dT_{sim}) of the simulation.

It is possible to consider dT_{ROS} as a noise of the model. Ideally, if it was zero, the optimal control input $u \coloneqq dT_{\text{ctrl}}$ would coincide with the desired discrete time dT_{des}. The execution time of the services can be expressed as

$$dT_{\text{ROS}} \coloneqq x_k^{\text{noise}} = \overline{x}_k^{\text{n}} + \widetilde{x}_k^{\text{n}} \,,$$

where x_k^{noise} is composed of a variable mean value $\overline{x}_k^{\text{n}}$ and a residual part $\widetilde{x}_k^{\text{n}}$, while the subscript k represents the discrete time index of the signal. By means of the services, it is possible to compute the simulation time $dT_{\text{sim}} \coloneqq x_k$. This can be rewritten as

$$\begin{aligned} dT_{\text{sim}} &= dT_{\text{ROS}} + dT_{\text{ctrl}} \,, \\ x_k &= \overline{x}_k^{\text{n}} + \widetilde{x}_k^{\text{n}} + u_k \,. \end{aligned}$$

By imposing dT_{sim} as $dT_{\text{des}} \coloneqq x_k^{\star}$, the control input u_k can be computed as

$$\begin{aligned} x_k^{\star} &= \overline{x}_k^{\text{n}} + \widetilde{x}_k^{\text{n}} + u_k \,, \\ u_k &= x_k^{\star} - \overline{x}_k^{\text{n}} - \widetilde{x}_k^{\text{n}} \,. \end{aligned}$$

The best estimation of an unknown stochastic signal is its mean value, but in this case, a weighted mean value has been computed in order to track it in a better way,

$$\hat{x}_k^{\text{n}} = \frac{1}{W} \sum_{i=1}^{k} \lambda^{k-i} x_i^{\text{noise}} = \frac{1}{W} \sum_{i=1}^{k} \lambda^{k-i} (x_i - u_i) \,,$$

where

$$W = \sum_{i=1}^{k} \lambda^{k-i}$$

and $\lambda \in (0, 1)$ is the so called forgetting factor.
Finally, the control input u_k can be computed as

$$u_k = x_k^{\star} - \hat{x}_{k-1}^{\text{n}} \,.$$

The control scheme in Figure 1.5 is used to manage the discrete time of the simulation. The evolution of the discrete time during a SIL simulation is shown in Figure 1.6, while the stochastic noise generated by the ROS infrastructure is shown in Figure 1.7. In that figure, it is also plotted the weighted mean value of the stochastic noise, used to compute the control input dT_{ctrl}. The SIL simulation has been generated with a desired discrete time dT_{des} of $0.15s$, for a simulation time of $38s$. The estimated dT_{ROS} is computed via weighted mean with a forgetting factor λ of 0.94. The average obtained from the controlled discrete time $mean(dT)$ signal is equal to $0.14992s$.

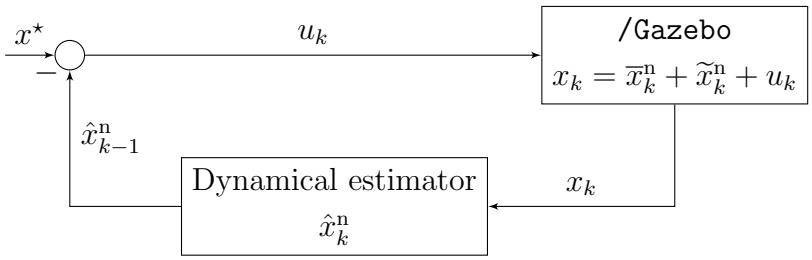

Figure 1.5: Time management control scheme

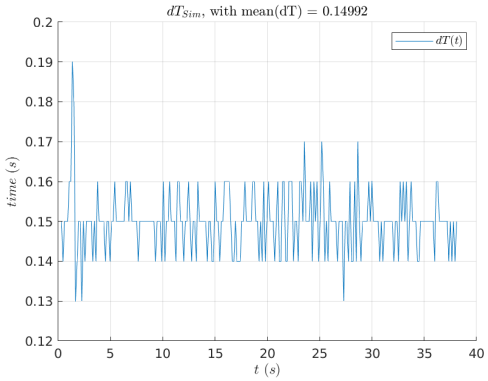

Figure 1.6: Evolution of the controlled discrete time during the SIL simulation.

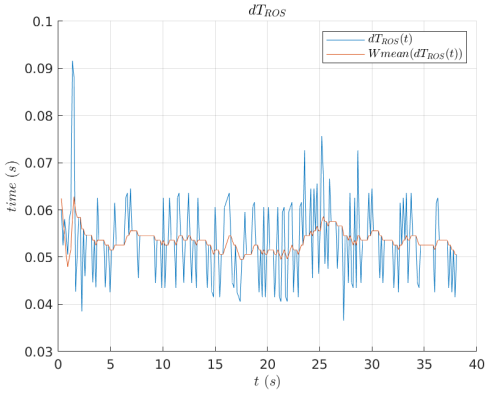

Figure 1.7: Evolution of the stochastic noise generated by the ROS infrastructure during the SIL simulation.

Simulator

Following it is explained how the simulator for the SIL is implemented. First, a RotorS launch file is created and executed, where the desired MAV (`hummingbird`) is loaded into the Gazebo simulator with the desired sensors (`odometry`). This can be seen as the *Gazebo* block of Figure 1.3, since the *MAV Control* block is substituted by the MATLAB node.
Next, a MATLAB script performing the following steps is launched:

1. *Simulation Setup*:
   This phase consists of the ROS communications setup. Initially, the MATLAB node is created, then it subscribes to the topics, finally the services client are generated. After that, the parameters for the simulation, such as the simulation time and the discrete time step, as well as all the parameters needed during the simulation, are set.

2. *Initialization Phase*:
   The initial conditions of the MAV (X_0, U_0), defined during the setup, are imposed.

3. *Simulation Loop*:

   (a) Read and save sensor data.

(b) PAUSE (Gazebo).

(c) Process sensor data.

(d) Control algorithm:
It consists of the prestabilizable controller, the online system identification, and the optimization for the exploration input. This concept will be explained in the next chapters.

(e) Write actuator data.

(f) Plot and save variables during simulation time.

(g) Timing management.

(h) START (Gazebo).

(i) Wait (dT_{ctrl}).

(j) Read and save control input feedback data.

4. *Plot Results*

5. *Save Simulation Data*

Due to the development of the Software In The Loop simulator, the control algorithm implemented in MATLAB does not need to satisfy the real-time constraints, while the MAV is simulated in Gazebo for an almost fixed discrete time.

1.3 System Identification Technique

The aim of this section is to present the paper *Towards non-stochastic targeted exploration* [1] in order to provide the necessary background for the following chapters.

1.3.1 Problem statement

The paper presents a strategy to compute a targeted exploration input for linear time-invariant systems without stochastic assumptions on the noise (for example deterministic model misspecifications). It takes into account classical data-dependent uncertainty bounds on the least-squares parameter estimates in the presence of energy-bounded noise, in order to provide a sufficient condition on the exploration data that ensures a desired error bound on the estimated parameters. For a tractable solution, using common approximations, a semidefinite program to compute the optimal sinusoidal input excitation is derived.

Setup

Consider a discrete-time, linear time-invariant system of the following form

$$x_{k+1} = A_{\text{tr}} x_k + B_{\text{tr}} u_k + w_k \,,$$

where $x_k \in \mathbb{R}^{n_x}$ is the state, $u_k \in \mathbb{R}^{n_u}$ the input and $w_k \in \mathbb{R}^{n_x}$ the energy-bounded disturbance. It is assumed that the states are directly measurable.

Since the disturbance w_k is energy-bounded, there exists a known constant $\gamma_w > 0$ such that:

$$\sum_{k=0}^{T-1} ||w_k||^2 \leq \gamma_w \,. \tag{1.6}$$

For the sake of the explanation, it is important to define what a data-dependent uncertainty bound is. It is a bound on the least-squares parameter estimates in the presence of energy-bounded noise. Given observed data

$$\mathcal{D}_T = \{\{x_0, u_0\}, \ldots, \{x_{T-1}, u_{T-1}\}, \{x_T\}\}$$

with dimension $(T+1) \in \mathbb{N}$ and the least-squares estimate $\hat{\theta}_T = (\hat{A}_T, \hat{B}_T)$ of the system matrix A_{tr} and B_{tr} and its covariance matrix P_θ, the data-dependent uncertainty bound is a set of parameters Θ_T such that:

$$\Theta_T \coloneqq \left\{\theta : (\theta - \hat{\theta}_T)^\top P_\theta^{-1} (\theta - \hat{\theta}_T) \leq G\right\} \,, \tag{1.7}$$

where

$$G \coloneqq \left(\gamma_w + ||\hat{\theta}_T||^2_{P_\theta^{-1}} - \sum_{k=0}^{T-1} ||x_{k+1}||^2\right) \in \mathbb{R} \,. \tag{1.8}$$

Objective

The objective is to design an exploration input u_k that allows estimating the system matrix A_{tr} and B_{tr} while satisfying the following condition:

$$(\theta_{\text{tr}} - \hat{\theta}_T)^\top D_{\text{des}} (\theta_{\text{tr}} - \hat{\theta}_T) \leq 1 \,, \tag{1.9}$$

where

$$D_{\text{des}} \succeq 0 \quad \in \mathbb{R}^{(n_\phi n_x) \times (n_\phi n_x)}$$

is a desired closeness matrix of $\hat{\theta}_T$ to θ_{tr}, and $n_\phi = n_x + n_u$.

1.3.2 Targeted exploration input

The exploration input sequence takes the form

$$u_k = \sum_{i=1}^{L} \alpha_i \cos(2\pi f_i k) \qquad k = 0, \ldots, T-1 \,,$$

where $T \in \mathbb{N}$ is the length of the exploration sequence, $L \in \mathbb{N}$ is the number of harmonics, $\alpha_i \in \mathbb{R}$ is the amplitude of the i-th harmonics, and $f_i \in \mathbb{R}$ is the frequency of the i-th harmonics. The frequencies f_i are selected from a set $\mathcal{F}_T \coloneqq \{0, 1/T, \ldots, (T-1)/T\}$ and are known a priori, thus before the optimization problem is solved, while the amplitudes α_i are the decision variables of the optimization problem.

One of the constraints imposed in the optimization problem requires that the control energy at each time instant does not exceed a constant value γ_e^2. This can be written as

$$\sum_{i=1}^{L} ||\alpha_i||^2 \leq \gamma_e^2 \,,$$

with the energy-bound $\gamma_e \geq 0$ imposed as a cost function and minimized.

The second constraint is necessary to give a sufficient condition on the exploration data that ensures the desired error bound on the estimated parameter, satisfying $\theta_{\text{tr}} \in \hat{\Theta}_T$. It imposes:

$$\begin{aligned} P_\theta^{-1} &\succeq G \cdot D_{\text{des}} \quad [1, \text{Eq.}(22)] \\ G \cdot P_\theta &\preceq D_{\text{des}}^{-1} \,. \end{aligned} \tag{1.10}$$

Combining the exploration constraint (1.10) with the uncertainty bound (1.7) the following result can be obtained:

$$(\theta_{\text{tr}} - \hat{\theta}_T)(\theta_{\text{tr}} - \hat{\theta}_T)^\top \preceq G \cdot P_\theta \preceq D_{\text{des}}^{-1} \,, \tag{1.11}$$

which satisfies the objective of the paper (1.9).

This constraint, as well as the previous one, can be expressed as a linear matrix inequality (LMI) and its proof is provided in [1].

1.3.3 Algorithm

Given the setup and the constraints, a tractable solution to the optimization problem will be presented together with the explanation of their effects.

Initial definition

- Consider a rough initial estimate of the dynamics $\hat{\theta}_0 = [vec(\hat{A}_0); vec(\hat{B}_0)]$.
- Specify the exploration length T of the desired exploration input sequence. The lower is the time horizon, the higher will be the energy of the exploration signal.
- Select $L \in \mathbb{N}$ frequencies f_i, from the set $\mathcal{F}_T$, where $i \in \{1, \ldots, L\}$. The frequencies are selected in order to cover the bound of the system dynamics, by previously studying the frequency response of the initial estimation $\hat{A}_0$, $\hat{B}_0$.
- Define an estimate of the energy-bound γ_w of the disturbance.
  A rough estimate can be computed by running a simulation and evaluating

$$\overline{\gamma}_w = \sum_{k=0}^{T-1} ||x_{k+1} - \hat{x}_{k+1}||^2 \,, \tag{1.12}$$

where

$$\begin{aligned}\hat{x}_{k+1} &= \hat{A}_0\hat{x}_k + \hat{B}_0 u_k \\ &= (A_{\text{tr}} - \Delta_{A_0})\hat{x}_k + (B_{\text{tr}} - \Delta_{B_0})u_k\,.\end{aligned}$$

This estimate will take into account also the mismatch of the parameters, which can be computed as

$$\begin{aligned}\overline{\gamma}_w &= \sum_{k=0}^{T-1} ||x_{k+1} - \hat{x}_{k+1}||^2 \\ &= \sum_{k=0}^{T-1} ||A_{\text{tr}}x_k + B_{\text{tr}}u_k + w_k - (A_{\text{tr}} - \Delta_{A_0})\hat{x}_k - (B_{\text{tr}} - \Delta_{B_0})u_k||^2 \\ &\approx \sum_{k=0}^{T-1} ||w_k + (\Delta_{A_0}\hat{x}_k + \Delta_{B_0}u_k)||^2 \\ &\geq \sum_{k=0}^{T-1} ||w_k||^2 = \gamma_w\,.\end{aligned}$$

In case it is possible to have previous knowledge of the disturbance, it is possible to use Equation (1.6) directly.

- Select the desired accuracy of the parameters estimation D_{des}. A possible selection could be obtained by studying the initial uncertainty bound $\hat{D}_0$ of $\hat{\theta}_0$ and select $D_{\text{des}} \succeq \hat{D}_0$.

Initial setup

- Simulating the open-loop initial system $(\hat{A}_0, \hat{B}_0)$ with $w = 0$ and

$$u_k^{(i)} = \cos(2\pi f_i k)$$

for $i = 1, \ldots, L$ and $k = 0, \ldots, T-1$ and recording the resulting values in $\hat{X}^{(i)}$, $\hat{\Phi}^{(i)}$.

- Select an initial candidate amplitude $\tilde{\alpha}_i$, $i = 1, \ldots, L$.

Iterative optimization

while amplitude α_i are not converged **do**

1. Compute $L_1(\tilde{\alpha}_i)$, $L_2(\tilde{\alpha}_i)$ [1, Eq.(31)] necessary for a convex relaxation in order to have the constraint linear in the decision variables α_i.

2. Solve the optimization problem as a semidefinite program (SDP) in order to find the optimal amplitude $\alpha_i^\star$:

$$\begin{aligned}\inf_{\alpha_i, \gamma_e} \quad & \gamma_e \\ \text{s.t.} \quad & S_{\text{energy-bound}}(\gamma_e, U_e) \succeq 0 \\ & S_{\text{exploration}}(\hat{\Phi}(U_e), \hat{Y}(U_e), L_1(\tilde{\alpha}_i), L_2(\tilde{\alpha}_i), \gamma_w, D_{\text{des}}) \succeq 0\end{aligned} \tag{1.13}$$

where $U_e = \text{diag}(\alpha_1, \ldots, \alpha_L) \in \mathbb{R}^{Ln_u \times L}$

3. Set $\tilde{\alpha}_i = \alpha_i^\star$.

end while

Generate exploration input

Once the amplitudes are converged to the optimal values $\alpha_i^\star$, the exploration input can be generated as follows:

$$u_k = \sum_{i=1}^{L} \alpha_i^\star \cos(2\pi f_i k) \qquad k = 0, \ldots, T-1\,. \tag{1.14}$$

Chapter 2

Methodology

In this chapter, the methodology used in this thesis work to test the system identification algorithm proposed in [1] is presented. The theoretical methodology is combined with the SIL simulator and a prestabilizable controller for the quadrotor.

2.1 Problem formulation

The linearized dynamical model of the quadrotor is decoupled, specifically its evolution in the vertical direction is independent of the evolution of the other states. This dynamic is referred to as altitude dynamic, and can be described by the time evolution of the vertical acceleration variable in the Equation (1.5), resulting in

$$\ddot{z} = \frac{1}{m} U_1 - g \ , \tag{2.1}$$

where the states are the altitude z and the velocity $\dot{z}$, while the input is the overall vertical thrust $U_1 \coloneqq T$.
The objective of this thesis is to identify the parameters of the discrete-time model describing the quadrotor altitude dynamic (2.1). To achieve this goal, a prestabilizable controller for the attitude dynamic is needed to guarantee the stability of the system during the identification process and, moreover, to fulfill Assumption 1. Specifically, the technique used to identify the parameters of the quadrotor is the exploration strategy explained in [1].

2.2 Design of a prestabilizing controller

As explained in Section 1.1, without the use of an attitude controller for the quadrotor, Assumption 1 is not satisfied and the dynamics is not decoupled, resulting in an unstable quadrotor. For this reason, a prestabilizable controller for the attitude dynamics has been designed. Due to the SIL simulator, implemented with the ideal odometry sensor, the feedback signals are all the states, thus the position and the velocity of all the 6-DoF of a body in space. For the attitude

controller, given the error signals of the attitude state variables, two Proportional-Integral (PI) controllers have been implemented: one for controlling τ_θ and one for τ_ϕ. Specifically, the PI controller is implemented as

$$\tau[k] = \begin{bmatrix} K_P^p & K_I^p & K_P^v & K_I^v \end{bmatrix} \cdot \begin{bmatrix} e_P^p[k] \\ e_I^p[k] \\ e_P^v[k] \\ e_I^v[k] \end{bmatrix} ,$$

where the appendix p and v refer to the position and velocity of the attitude state variables, respectively, while the P and I refer to the proportional and integral parts of the controller. Moreover, K rappresent the gain of the controller while $[k]$ is the discrete time index. The integral error can be computed as

$$e_I[k] = e_{I_0} + \sum_{i=1}^{t} e_P[i]\, dT .$$

As shown in Figure 2.1, the position controller has not been implemented, since, during the simulation, once initialized in the desired position, the only disturbance acting on the quadrotor is gravity together with the numerical error of the simulator. Moreover, this controller should be carefully designed in order to guarantee the Assumption 1 ($\phi \approx 0,\ \theta \approx 0,\ \dot{x} \approx 0,\ \dot{y} \approx 0$) to be satisfied.

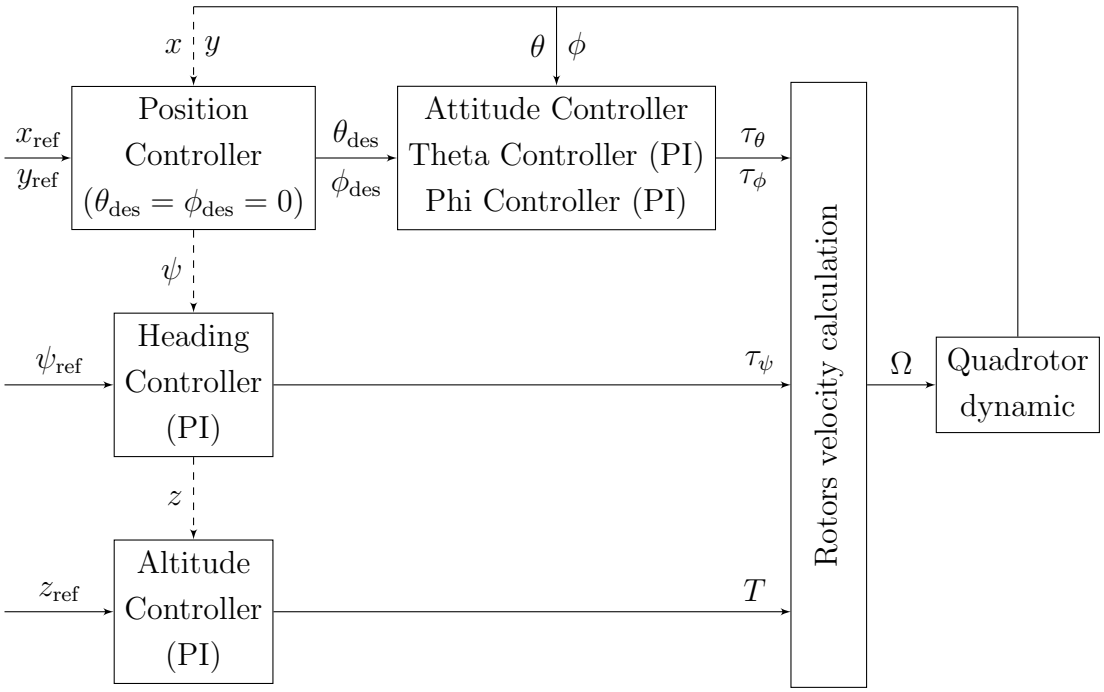

Figure 2.1: Quadrotor prestabilizable control structure

In Figure 2.2, the behavior of the *Attitude* and *Heading* controllers during a generic simulation is shown. It is possible to notice that the magnitude of the control signal ($\frac{N}{m}$) is almost null. In fact, once the drone is initialized in the desired position, the controller has only to compensate for the numerical error generated by the

Gazebo simulator. However, these controllers are crucial in performing any kind of simulation, because of the unstable nature of the quadrotor. In Figure 2.3, the behavior of the *Altitude controller* during a simulation is shown: the quadrotor is stabilized at a reference step altitude. It can be noticed from the plot that, once the transient phase is accomplished and the desired reference altitude is reached, the control effort results to be exactly equal to mg, thus the gravity disturbance acting on the system. The U_1^{FB} signal, present in the figure, is the reconstruction of the control input. It is computed from the sensor feedback signal of the actual rotor speeds, which are measured by the rotor sensors. The U_1^{FB} signal incorporates the saturation effect of the motor and the possible error that could have occurred during the simulation. In Figure 2.3, the saturation effects do not occur.

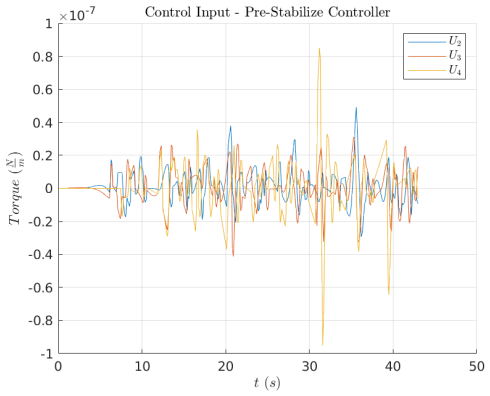

Figure 2.2: Control input of the prestabilizable controllers' time evolutions

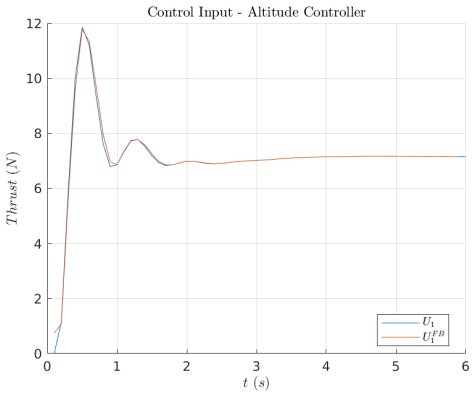

Figure 2.3: Control input of the Altitude controller's time evolutions

2.3 Altitude model setup

The altitude dynamic described by Equation (2.1) is a double integrator time-invariant linear system with a single input and a constant disturbance due to gravity. It can be defined in the continuous time state-space representation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{:=A^{\text{CT}}} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{:=B^{\text{CT}}} U_1 + \underbrace{\begin{bmatrix} 0 \\ -g \end{bmatrix}}_{:=W^{\text{CT}}}$$

where the state $x_1 := z$ is the altitude position and the state $x_2 := \dot{z}$ is the altitude velocity. The parameter m is the mass of the quadrotor and g is the gravitational acceleration.

Given the assumption of constant input during the intervals $[k\,dT, (k+1)\,dT]$ $\forall k \in \mathbb{N}$, and the structure of a continuous time double integrator, it is possible to

evaluate the state update as

$$\begin{aligned} X((k+1)\,dT) = e^{A^{\mathrm{CT}}\,dT} \cdot X(k\,dT) \\ + \int_{k\,dT}^{(k+1)\,dT} e^{A^{\mathrm{CT}}\,((k+1)\,dT-\tau)} B^{\mathrm{CT}} d\tau \cdot U_1(k\,dT) \\ + \int_{k\,dT}^{(k+1)\,dT} e^{A^{\mathrm{CT}}\,((k+1)\,dT-\tau)} W^{\mathrm{CT}} d\tau \end{aligned}$$

where the solution of the integrals are computed as [22]

$$\begin{bmatrix} e^{A^{\mathrm{CT}}\,dT} & \int_{k\,dT}^{(k+1)\,dT} e^{A^{\mathrm{CT}}\,((k+1)\,dT-\tau)} B^{\mathrm{CT}} d\tau \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A^{\mathrm{CT}} & B^{\mathrm{CT}} \\ 0 & 0 \end{bmatrix} \cdot dT \right).$$

According to the principle of superposition, the same can be applied to the constant disturbance term W^{CT}, which leads to the discrete-time update,

$$\begin{aligned} \begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} &= \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2} dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix} U_1[k] + \begin{bmatrix} -g \cdot \frac{1}{2} dT^2 \\ -g \cdot dT \end{bmatrix}, \\ X[k+1] &= A \cdot X[k] + B \cdot U[k] + W \end{aligned} \tag{2.2}$$

where dT is the sampling time and, from now on, when referring to the dynamic model of altitude, the variable U_1 will be represented by U for simplicity in the explanation.

Starting from the control scheme in Figure 2.1, the altitude controller is modified by adding a feedforward term ($T_{FF} = \hat{m}g$) in order to partially compensate for the constant disturbance of gravity. The term $\hat{m}$ is the estimated mass of the quadrotor. The resulting control input is the overall vertical thrust:

$$U_1[k] \equiv T[k] = T_{\mathrm{eff}}[k] + T_{\mathrm{FF}}$$

where the effective thrust, T_{eff}, is the thrust force generated by the drone's motors minus the estimated gravitational force acting on the drone. This effective thrust represents the resulting upward force ideally available for vertical acceleration. The overall control scheme for the altitude dynamic is shown in Figure 2.4. The control input (T_{eff}) generated by the altitude controller can be computed with the PI-Controller or by following the exploration signal. The generation of this signal is explained more in detail in Section 2.4.2. It is important to notice that, as shown in Figure 2.2, the actions U_2 and U_3 are almost zero but are necessary to satisfy Assumption 1 of the linearization and, together with the matching input, to guarantee a decoupled dynamics.

2.4 Exploration technique implementation

Once the altitude model derivation and the control scheme setup are explained, this section focuses on the system identification performed according to the techniques explained in [1]. The state space representation of Equation (2.2) highlights

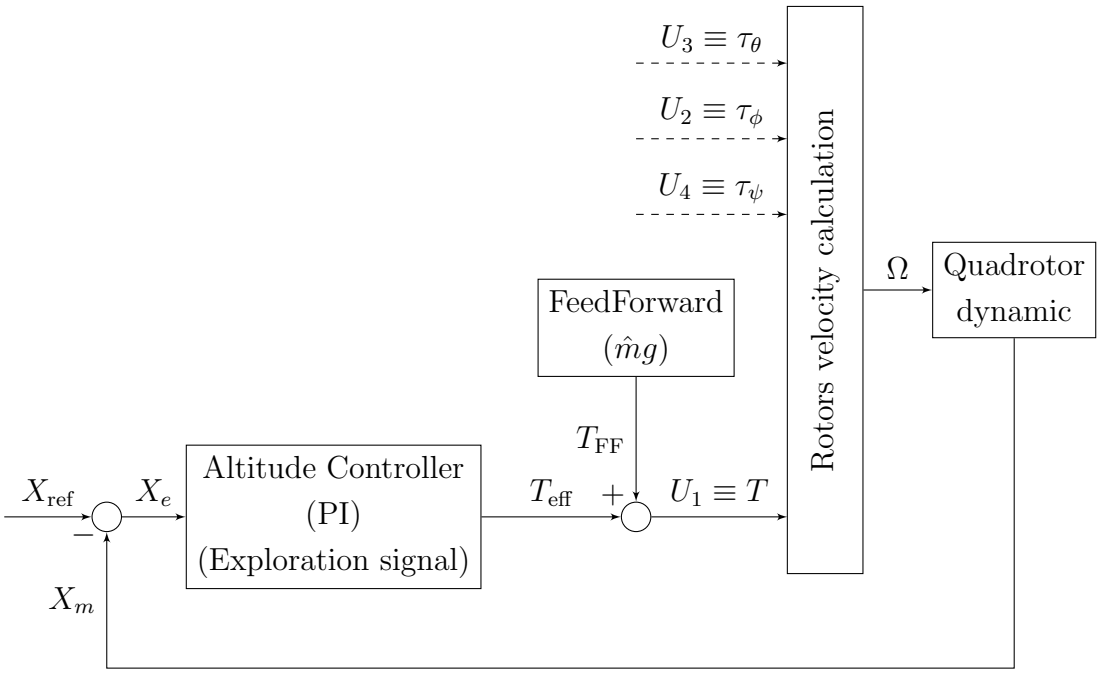

Figure 2.4: Altitude control scheme

the presence of a disturbance. This disturbance satisfies the assumption for the exploration strategy, due to its non-stochastic nature and the possibility of calculating an energy-bound on it over a defined exploration time.
Given the state space representation of the altitude model (2.2) and the true parameters used by the SIL simulator, it is possible to compute the true quadrotor altitude dynamic as

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2} dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix} U[k] + \begin{bmatrix} -g \cdot \frac{1}{2} dT^2 \\ -g \cdot dT \end{bmatrix},$$

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.1 \\ 0 & 1 \end{bmatrix}}_{:=A_{\text{tr}}} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \underbrace{\begin{bmatrix} 0.0068 \\ 0.1370 \end{bmatrix}}_{:=B_{\text{tr}}} U[k] + \underbrace{\begin{bmatrix} -0.0490 \\ -0.98 \end{bmatrix}}_{:=W_{\text{tr}}}$$

where the parameters are $m = 0.73\ kg$ for the mass of the quadrotor, $dT = 0.1\ s$ for the sampling time imposed during the simulations, and $g = 9.8\ \frac{m}{s^2}$ for the gravity term. The goal is to estimate the parameters dT, m, as well as the matrices A_{tr} and B_{tr}, in order to analyze the performance of the system identification algorithm proposed in [1]. The feedforward action added to the control architecture is needed to partially compensate for the disturbance (W_{tr}) in such a way as to obtain numerically tractable results for the simulations.

2.4.1 Initial definition

To derive the exploration signal from the optimization algorithm, it is necessary to set several initial values for the algorithm. This process includes establishing

the initial estimates of the parameters and their corresponding initial closeness matrix, defining the desired closeness matrix to be achieved after the exploration, setting the frequencies of the exploration signal, and determining the associated exploration time length. Finally, the energy bound of the disturbance after partial compensation must be computed.
The coefficients of the state space representation (the A and B matrices) are grouped as follows

$$\theta = \text{vec}[A, B] \in \mathbb{R}^{n_x n_\phi}$$

where $n_\phi = (n_x + n_u)$.

Initial Estimation

An initial estimation of the parameters must be set in order to run the optimization algorithm. A potential approach involves computing an initial estimate value of the parameters ($\hat{\theta}_0$) and its initial closeness matrix D_0 from observed data. To ensure repeatable experiments with comparable results, the initial estimate values have been obtained by selecting $\hat{dT}$ and $\hat{m}$ from a normal distribution with the mean equal to the true value

$$\hat{dT} \sim \mathcal{N}(dT, \sigma^2)\,,$$
$$\hat{m} \sim \mathcal{N}(m, \sigma^2)$$

and the covariance such that the following condition is satisfied

$$(\theta_{\text{tr}} - \hat{\theta}(\sigma))^\top D (\theta_{\text{tr}} - \hat{\theta}(\sigma))^\top = 1\ .$$

Table 2.1 shows the estimates of the parameters for different imposed values of the closeness matrix D, which is computed as

$$D = \lambda_D \cdot I_{n_\phi n_x}\ .$$

Starting from the first column on the left of the table, the following are reported: the coefficient of the imposed value of the D matrix, the parameters obtained from the normal distribution, the resulting feedforward action from the selected mass, and the norm of the error. The norm of the error is computed only on the coefficients of the state space representation containing the parameters to be estimated (only 3 out of 6) as follows:

$$\|e\| = \left\| \begin{bmatrix} A_{\text{tr}\,1,2} - \hat{A}_{1,2} \\ B_{\text{tr}\,1,1} - \hat{B}_{1,1} \\ B_{\text{tr}\,1,2} - \hat{B}_{1,2} \end{bmatrix} \right\|\ .$$

For the reported simulation, $D = 50 \cdot I_{n_\phi n_x}$ was selected. Thus, the initial estimate of the altitude model results in the following state space representation

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.1635 \\ 0 & 1 \end{bmatrix}}_{\coloneqq \hat{A}_0} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \underbrace{\begin{bmatrix} 0.0214 \\ 0.2617 \end{bmatrix}}_{\coloneqq \hat{B}_0} T_{\text{eff}}[k]$$

λ_D	$\hat{dT}\ [s]$	$\hat{m}\ [kg]$	$\hat{m}g\ [N]$	$\|e\|$
1	-0.6432	-0.8596	-8.4241	0.9936
10	0.1411	0.3141	3.0778	0.3158
50	0.1635	0.6247	6.1221	0.1407
100	0.0512	0.2287	2.2408	0.0997
500	0.1430	0.9694	9.4999	0.0444
1000	0.0829	0.5071	4.9696	0.0315
10000	0.0906	0.6779	6.6438	0.0100
1000000	0.0993	0.7281	7.1351	0.0010
∞	0.1	0.73	7.1541	0

Table 2.1: Initial estimate of the parameters of the altitude dynamical model

with $U[k] = T_{\text{eff}}[k] + \underbrace{6.1221}_{T_{\text{FF}}=\hat{m}_0 \cdot g}$. The parameters $\hat{dT}_0$ and $\hat{m}_0$ are

$$\hat{dT}_0 = 0.1635\ s,$$
$$\hat{m}_0 = 0.6247\ kg.$$

Initial Closeness Matrix

The inverse of the initial closeness matrix D_0^{-1} for the initial estimation obtained earlier is computed as

$$D_0^{-1} = (\theta_{\text{tr}} - \hat{\theta}_0)(\theta_{\text{tr}} - \hat{\theta}_0)^\top$$

which results in:

$$D_0^{-1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0040 & 0 & 0.0009 & 0.0079 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0009 & 0 & 0.0002 & 0.0018 \\ 0 & 0 & 0.0079 & 0 & 0.0018 & 0.0156 \end{bmatrix}.$$

Due to the fact that three coefficients of the state space representation are exactly known, it results in being singular. However, to execute the optimization algorithm, it is necessary to compute a positive definite matrix D_{des} such that $D_{\text{des}}^{-1} \preceq D_0^{-1}$. For this reason, a new relaxed initial closeness matrix D_{init}^{-1} needs to be defined and computed as

$$D_{\text{init}}^{-1} \succeq D_0^{-1}$$
$$D_{\text{init}}^{-1} = \lambda(D_0^{-1}) \cdot I_{n_\phi n_x}$$

where $\lambda(D_0^{-1})$ is the maximum eigenvalue of the D_0^{-1} matrix, and its inverse will provide the chosen coefficient for the computation of the initial parameters. This

results in

$$\lambda(D_0^{-1}) = \max(\Lambda(D_0^{-1})) = 0.0198,$$
$$\lambda(D_0) = \frac{1}{\lambda(D_0^{-1})} = 50.4868\ .$$

The resulting initial closeness matrix is

$$D_{\text{init}}^{-1} = \begin{bmatrix} 0.0198 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0198 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0198 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0198 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0198 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0198 \end{bmatrix}$$

This means that a non-perfect estimation of the coefficients known from the physical description of the model is allowed in the system identification phase. However, these coefficients are not taken into account for the calculation of the error.

Desired Closeness Matrix

The desired closeness matrix D_{des} needs to be selected positive definite such that $D_{\text{des}} \succeq D_{\text{init}}$ in order to achieve a better estimation of the parameters. Note that the greater the value of D_{des}, the greater the energy of the exploration signal will be for the same exploration time length. By analyzing Table 2.1, $D_{\text{des}} = 100 \cdot I_{n_\phi n_x}$ was selected. The inverse desired closeness matrix is

$$D_{\text{des}}^{-1} = 0.01 \cdot I_6\ .$$

Frequencies and Exploration Time Length

The other parameters that need to be set up for the optimization algorithm are the frequencies f_i of the exploration signal, and their amount L, from which it is possible to compute the exploration time length T_{exp}. To define which frequencies to select, the frequency response of the model needs to be analyzed. For the sake of the experiment, the frequency response of the true altitude model and not the one of the initial estimation has been studied. The system shows a low-pass filter behavior with a cut-off frequency at $f_c = 0.218\ Hz$. To achieve a good exploration of the system, it is necessary to select the frequencies f_i in the range of the cut-off frequency. Following this criteria the frequencies are selected as

$$f_i \in \left\{ \frac{3}{8} \cdot f_c, \frac{6}{8} \cdot f_c, 1 \cdot f_c, \frac{10}{8} \cdot f_c \right\} \subset \mathcal{F}_T, \quad \text{with} \quad i = 1, \ldots, L.$$

Remember that the set of frequencies $\mathcal{F}_T$ is defined as

$$\mathcal{F}_T \coloneqq \{0, 1/T, \ldots, (T-1)/T\}.$$

From this, it is possible to identify the exploration time as the period of the lowest non-zero frequency:

$$T_{\text{exp}} = \frac{1}{f_{\min}} = \left(\frac{1}{8} \cdot f_c\right)^{-1} \approx 37s.$$

Moreover, given the imposed sampling time $dT = 0.1s$ of the simulations, the number of exploration samples of the SIL simulator can be computed as

$$N_{T_{\text{exp}}} = \frac{T_{\text{exp}}}{dT} = 370 \ .$$

Energy-Bound of the Disturbance

As previously pointed out, the non-stochastic disturbance of the altitude dynamical model is gravity. By using Equation (1.6), the energy of the disturbance W during the exploration time length can be computed as

$$\begin{aligned} \gamma_w^{\text{tr}} &\geq \sum_{k=0}^{N_{T_{\text{exp}}}-1} \|W\|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \left\| \begin{bmatrix} -g \cdot \frac{1}{2}dT^2 \\ -g \cdot dT \end{bmatrix} \right\|^2 = 356.23 \ . \end{aligned}$$

However, due to the feedforward action tuned with a value of $(\hat{m}_0 \cdot g)$, it is partially compensated. The following equation represents how the disturbance results after the partial compensation,

$$X[k+1] = A \cdot X[k] + B \cdot U[k] + W$$

by substituting

$$U[k] = T_{\text{eff}}[k] + T_{FF} = T_{\text{eff}}[k] + (\hat{m}_0 \cdot g) \, ,$$

it is possible to achive,

$$\begin{aligned} \begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} &= \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2}dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix} (T_{\text{eff}}[k] + (\hat{m}_0 \cdot g)) + \begin{bmatrix} -g \cdot \frac{1}{2}dT^2 \\ -g \cdot dT \end{bmatrix} \\ &= \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2}dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix} T_{\text{eff}}[k] \\ &+ \begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2}dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix} (\hat{m}_0 \cdot g) + \begin{bmatrix} -g \cdot \frac{1}{2}dT^2 \\ -g \cdot dT \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{1}{m} \cdot \frac{1}{2}dT^2 \\ \frac{1}{m} \cdot dT \end{bmatrix}}_{B} T_{\text{eff}}[k] + \underbrace{\begin{bmatrix} \left(\frac{\hat{m}_0}{m} - 1\right) \cdot g \cdot \frac{1}{2}dT^2 \\ \left(\frac{\hat{m}_0}{m} - 1\right) \cdot g \cdot dT \end{bmatrix}}_{:= \tilde{W}} . \end{aligned}$$

It can be noticed that in the case of a perfect estimation of the mass (i.e., $\hat{m}_0 = m$), the disturbance term $\tilde{W}$ results to be zero and is thus perfectly compensated.

Once again, by using Equation (1.6), the energy of the partially compensated disturbance can be calculated as follows:

$$\begin{aligned}\gamma_{\tilde{w}} &\geq \sum_{k=0}^{N_{T_{\text{exp}}}-1} \|\tilde{W}\|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \left\| \begin{bmatrix} \left(\frac{\hat{m}_0}{m} - 1\right) \cdot g \cdot \frac{1}{2} dT^2 \\ \left(\frac{\hat{m}_0}{m} - 1\right) \cdot g \cdot dT \end{bmatrix} \right\|^2 = 7.41 \ .\end{aligned} \tag{2.3}$$

Finally, it is possible to define the disturbance energy-bound for the experiment as

$$\gamma_w = 20 \geq \gamma_{\tilde{w}} \ . \tag{2.4}$$

The bound selected is increased relative to the exact value of the energy calculated in Equation (2.3), but it is still significantly lower than the full energy γ_w^{tr} computed without partial compensation. This increase in energy accounts for small disturbances due to factors such as the non-constant sampling time of the SIL simulator and the less accurate calculation of rotor velocities compared to the RotorS simulator.

This disturbance energy bound (γ_w) is used, along with the previously determined initial values, to compute the optimal exploration signal with the optimization algorithm. Moreover, this value is also necessary to compute the uncertainty bound of the estimated parameters during the system identification phase.

2.4.2 Exploration Input Generation

Once the initial setup is computed, the semidefinite program 1.13 can be solved (see Figures 2.5, 2.6, and 2.7). The meta optimization, necessary for the convex relaxation of the problem, is stopped once the cost function and the descent direction are lower than the defined threshold values. The results obtained from the

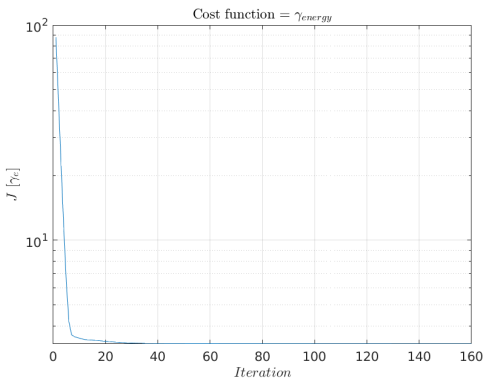

Figure 2.5: Cost function (γ_e)

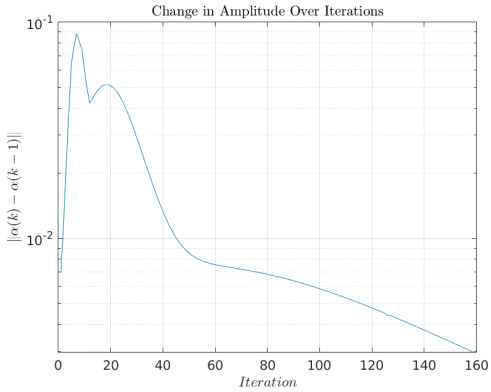

Figure 2.6: Change in amplitude

optimization algorithm are the following optimal amplitudes

$$\alpha^\star = \begin{bmatrix} 0.04 & 0.94 & 1.76 & -2.62 \end{bmatrix}^\top \ ,$$

and the corresponding energy

$$\gamma_e = 3.29\,.$$

Given the results of the optimization problem, it is possible to generate the exploration signal with Equation (1.14):

$$T_{\text{eff}}^{\text{exp}}[k] = \sum_{i=1}^{L} \alpha_i^{\star} \sin(2\pi f_i \cdot k\, dT), \quad \text{with} \quad k = 0, \ldots, N_{T_{\text{exp}}} - 1\,.$$

The obtained exploration signal $T_{\text{eff}}^{\text{exp}}$ is shown in Figure 2.8.

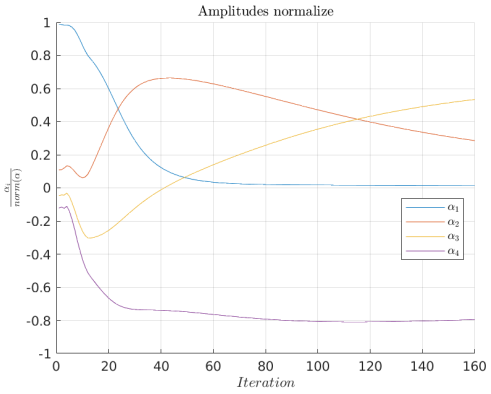

Figure 2.7: Normalized optimized amplitude

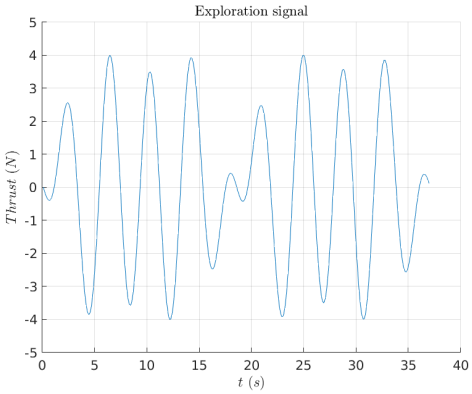

Figure 2.8: Optimized exploration signal

2.4.3 Exploration Phase

The exploration input U_1^{exp}, computed as $T_{\text{eff}}^{\text{exp}} + (\hat{m}_0 \cdot g)$, is used to excite the system and collect the data needed to estimate the coefficients of the altitude model. The resulting exploration input applied during the simulations can be seen in Figure 2.9.

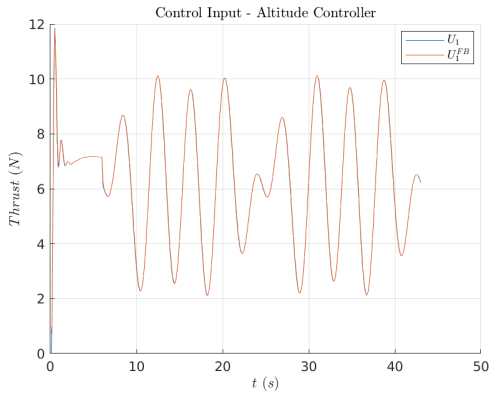

Figure 2.9: Exploration input

The evolution of the states $X = [z, \dot{z}]^\top$ is reported in Figures 2.10 and 2.11, respectively. As shown in Figure 2.10, during the simulation, there is a constant drift in the position due to the non-perfect compensation of the constant disturbance caused by gravity.

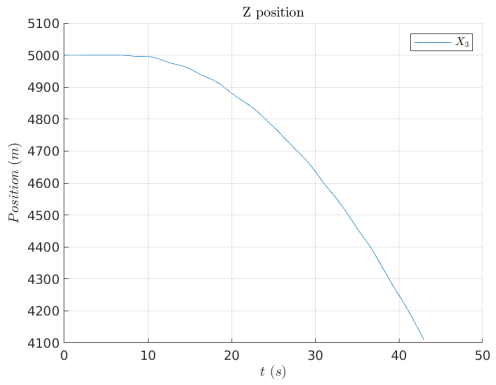

Figure 2.10: Evolution of the altitude

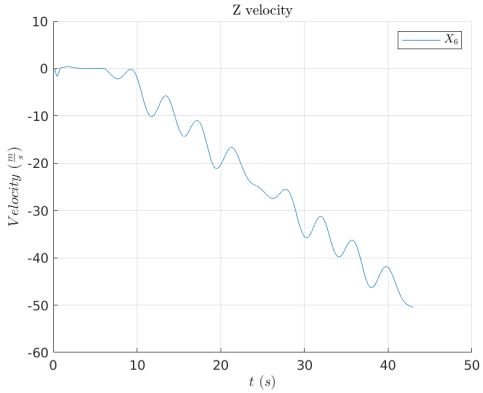

Figure 2.11: Evolution of the velocity

2.4.4 System Identification

By collecting the data during the experiment (Figures 2.9, 2.10, and 2.11) and using only the data generated while the exploration signal is applied, it is possible to solve the standard least-squares problem to estimate the coefficients of the altitude model. The exploration signal have been applied only from $6s$ onwarw from the beginning of the simulation, since during this initial time slot a PI-controller has been used to reach a fixed desired position. The estimated state space representation is

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.09989 \\ -2.605 \times 10^{-5} & 1.001 \end{bmatrix}}_{:= \hat{A}^{\text{LS}}_{T_{\text{exp}}}} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \underbrace{\begin{bmatrix} 0.002517 \\ 0.1363 \end{bmatrix}}_{:= \hat{B}^{\text{LS}}_{T_{\text{exp}}}} T^{\text{FB}}_{\text{eff}}[k]\,,$$

where $T^{\text{FB}}_{\text{eff}}$ is computed as

$$T^{\text{FB}}_{\text{eff}}[k] = U^{\text{FB}}_1[k] - \hat{m}_0 \cdot g\,.$$

From $\hat{\theta}^{\text{LS}}_{T_{\text{exp}}}$, it is possible to identify the estimated parameters

$$\begin{aligned} \hat{dT}_{T_{\text{exp}}} &= 0.09989\ s\,, \\ \hat{m}_{T_{\text{exp}}} &= 0.7331\ kg \end{aligned}$$

and the norm of its representing coefficient error results in

$$\|e_{T_{\text{exp}}}\| = \left\| \begin{bmatrix} A_{\text{tr}\,1,2} - \hat{A}_{T_{\text{exp}}\,1,2} \\ B_{\text{tr}\,1,1} - \hat{B}_{T_{\text{exp}}\,1,2} \\ B_{\text{tr}\,1,2} - \hat{B}_{T_{\text{exp}}\,1,2} \end{bmatrix} \right\| = 0.0044\,.$$

This result can be obtained only in simulation by having prior knowledge of the true parameter values, as it is computed by comparing the actual estimation with them.

Chapter 3

Simulation Results

In this chapter, the results of the thesis are presented and analyzed. First, some comments about the SIL simulator used for the experiments are reported. Then, the results of the experiments are presented, verifying the initial assumptions regarding the energy bound of the disturbance and the objective of the system identification technique presented in [1]. The chapter ends with a comparison with a different exploration signal, generated by a gaussian stochastic process.

3.1 SIL Simulator Analysis

The SIL simulator was essential for solving "online" the optimization problem during the simulation, eliminating the need for offline or parallel execution. The modular structure used for the implementation results to be ideal for employing multiple controllers, each one for a specific control input. Additionally, the time management was precise, resulting in an almost constant discrete time interval. However, due to communication between MATLAB and ROS, the minimum discrete time that can be set is $dT = 0.1\,s$, which corresponds to a controller frequency of $10\,Hz$. This results from the exact discrete time evolution of the state space representation, making the second-order term (dT^2) not negligible. This is a limitation that can be improved in future works.

3.2 Results of the Experiment

In this section, the results obtained during the experiments are reported and analyzed. Starting from the verification of the assumed energy bound for the disturbance and sequentially the objective for the system identification technique presented in [1].

3.2.1 Disturbance Energy

Once data is collected, the energy of the disturbance acting on the system during the exploration phase can be computed to verify if the disturbance energy bound (γ_w) imposed in the optimization problem is satisfied. It is important to add a new term representing the noise of the simulator acting on the system evolution in the state space representation (2.2),

$$X[k+1] = A \cdot X[k] + B \cdot U[k] + W + W_{\text{sim}}[k]\,,$$

where W represents the constant disturbance of gravity acting on the system, while W_{sim} represents the noise introduced by the SIL simulator, representing the difference between the simulated and the ideal evolution of Equation (2.1). This noise accounts for factors such as the non-constant sampling time of the SIL simulator and the less accurate calculation of rotor velocities compared to the RotorS simulator or, in a practical case, the noise of the sensors. Note that RotorS uses a more accurate non-linear model of the propeller dynamics compared to the input matching model used in the SIL simulator.
As highlighted in the previous chapter, the exploration input is composed of the exploration signal and a feedforward action to compensate for the constant disturbance of gravity,

$$\begin{aligned} U^{\text{FB}}[k] &= T_{\text{eff}}^{\text{FB}}[k] + T_{\text{FF}} \\ X[k+1] &= A \cdot X[k] + B \cdot (T_{\text{eff}}^{\text{FB}}[k] + T_{\text{FF}}) + W + W_{\text{sim}}[k] \\ X[k+1] &= A \cdot X[k] + B \cdot T_{\text{eff}}^{\text{FB}}[k] + \underbrace{B \cdot T_{\text{FF}} + W}_{\tilde{W}} + W_{\text{sim}}[k]\,. \end{aligned}$$

The energy of the attenuated disturbance $\tilde{W}$ is computed in Equation (2.3) and was the only possible computation executable a priori. With the data collected during the exploration, it is now possible to compute the full energy of the disturbances acting on the system during the exploration phase,

$$\begin{aligned} \gamma_{w_{\text{exp}}} &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \left\| X[k+1] - \left(\begin{bmatrix} X[k] \\ T_{\text{eff}}^{\text{FB}}[k] \end{bmatrix} \otimes I_{n_x} \right) \theta_{\text{tr}} \right\|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \| X[k+1] - (A \cdot X[k] + B \cdot T_{\text{eff}}^{\text{FB}}[k]) \|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \| B \cdot T_{\text{FF}} + W + W_{\text{sim}}[k] \|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \| \tilde{W} + W_{\text{sim}}[k] \|^2 = 17.87\,. \end{aligned} \tag{3.1}$$

Moreover, it is possible to compute the energy of the noise acting during the simulation,

$$\begin{aligned}\gamma_{w_{\text{sim}}} &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \left\| X[k+1] - \left(\begin{bmatrix} X[k] \\ U^{\text{FB}}[k] \end{bmatrix} \otimes I_{n_x} \right) \theta_{\text{tr}} - W \right\|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \| X[k+1] - (A \cdot X[k] + B \cdot U^{\text{FB}}[k] + W) \|^2 \\ &= \sum_{k=0}^{N_{T_{\text{exp}}}-1} \| W_{\text{sim}}[k] \|^2 = 10.62\,.\end{aligned}$$

From this result, it can be concluded that the energy bound of the disturbance $\gamma_w \geq \gamma_{w_{\text{exp}}}$ imposed as initial parameter for the optimization problem is satisfied. Note that these a posteriori calculations are possible only because the true parameters are known in the simulation. In Figure 3.1, the evolution of the noise W_{sim}

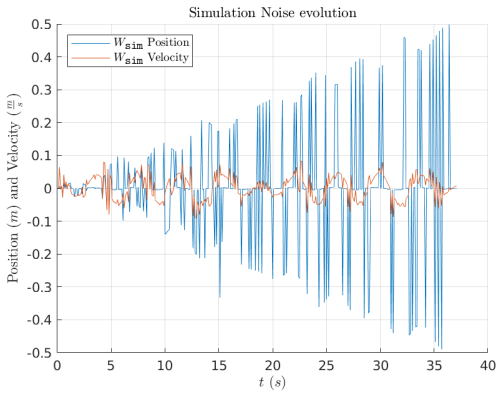

Figure 3.1: Noise acting on the state evolution

acting on the state computation is shown. It is possible to notice that the noise acting on the computation of the position is always increasing due to the increase in velocity during the experiment as reported in Figure 2.11. This introduces an error in the computation of the position for the physical engine of Gazebo, due to the non-constant sampling time of the SIL simulator, which is fundamental in the integration of the velocity itself.

3.2.2 Objective of the Exploration

The results presented in the *System Identification* section 2.4.4 after the exploration phase are analyzed here.
To verify the correctness of the solution provided by the optimization algorithm, it is necessary to check if the constraint (1.10) imposed during the optimization problem is satisfied, i.e. if the following holds:

$$G \cdot P_\theta \preceq D_{\text{des}}^{-1}\,.$$

The left-hand side of the inequality can be interpreted as a closeness matrix, $D_{\text{bd}}^{-1} \coloneqq G \cdot P_\theta$, representing a data-dependent bound on the uncertainty of the estimated parameters in the presence of energy-bounded disturbances. While the right-hand side represents the desired accuracy to the true parameters imposed at the beginning of the experiment.
To calculate the closeness matrix D_{bd}^{-1}, the following matrices need to be defined:

$$\begin{aligned}
&\phi[k] = [X[k]^\top, U[k]^\top]^\top \in \mathbb{R}^{n_\phi}\,,\\
&\Phi = [\phi[0], \dots, \phi[T_{\text{exp}} - 1]] \in \mathbb{R}^{n_\phi \times T_{\text{exp}}}\,,\\
&\overline{X}^\top = [X[1]^\top, \dots, X[T_{\text{exp}}]^\top] \in \mathbb{R}^{1 \times T_{\text{exp}} \cdot n_x}\,.
\end{aligned}$$

The components G and P_θ can be calculated as follows,

$$\begin{aligned}
P_\theta &= (\Phi\Phi^\top \otimes I_{n_x})^{-1}\,,\\
G &\overset{(1.8)}{=} \left(\gamma_w - \overline{X}^\top\overline{X} + \overline{X}^\top((\Phi^\top(\Phi\Phi^\top)^{-1}\Phi) \otimes I_{n_x})\overline{X}\right)\,,
\end{aligned}$$

where the energy bound of the disturbance γ_w is calculated in Equation (2.4). The matrix D_{bd}^{-1} computed in this way is symmetric and positive definite, as is characteristic of a closeness matrix. Moreover, verifying that $(D_{\text{bd}}^{-1} - D_{\text{des}}^{-1})$ is negative semi-definite confirms the initial Equation (1.10).
To provide a comparable result of the quality of the estimation, the inverse of the maximum eigenvalue of the D_{bd}^{-1} matrix can be computed, representing the uncertainty in the direction where it is maximum,

$$\lambda(D_{\text{bd}}) = \frac{1}{\max(\Lambda(D_{\text{bd}}^{-1}))} = 212.8\,,$$

which is greater than the imposed desired uncertainty $\lambda(D_{\text{des}}) = 100$.
Once verified, Equation (1.10) provides a sufficient condition to ensure the objective of [1], reported in Equation (1.9),

$$(\theta_{\text{tr}} - \hat{\theta}_T)^\top D_{\text{des}}(\theta_{\text{tr}} - \hat{\theta}_T) \leq 1\,.$$

Since the true parameters are known in the simulation, it is possible to compute this directly, resulting in

$$(\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}})^\top D_{\text{des}}(\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}}) = 0.0056 \leq 1\,.$$

To complete the analysis, by using the true parameter values, it is also possible to compute the closeness matrix $D_{T_{\text{exp}}}^{-1}$ as

$$D_{T_{\text{exp}}}^{-1} = (\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}})(\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}})^\top\,.$$

It is important to highlight that the symbol of the inverse in $D_{T_{\text{exp}}}^{-1}$ does not mean the inverse of the matrix itself but has been used only for consistency of notation,

since the matrix is not invertible by construction. Now it is possible to compute the exact maximum uncertainty,

$$\lambda(D_{T_{\text{exp}}}) = \frac{1}{\max(\Lambda(D_{T_{\text{exp}}}^{-1}))} = 89\,768\,,$$

where $\max(\Lambda(D_{T_{\text{exp}}}^{-1}))$ is the only eigenvalue different from zero. Once verified that $(D_{T_{\text{exp}}}^{-1} - D_{\text{bd}}^{-1})$ is negative semi-definite, it is possible to verify Equation (1.11),

$$D_{T_{\text{exp}}}^{-1} \preceq D_{\text{bd}}^{-1} \preceq D_{\text{des}}^{-1} \preceq D_{\text{init}}^{-1}\,.$$

These results show that the goal ($D_{T_{\text{exp}}}^{-1} \preceq D_{\text{des}}^{-1}$) is highly satisfied, as the imposed constraint (1.10) ensures $D_{\text{bd}}^{-1} \preceq D_{\text{des}}^{-1}$. Note that, in the case of unknown true parameters, the only closeness matrix that can be computed is D_{bd}, which is a conservative result compared to $D_{T_{\text{exp}}}$.

3.3 Comparison with a Different Exploration Signal

In this section, the results from an experiment that utilizes a random exploration signal are reported. For a fair comparison with the designed exploration strategy, a normally distributed random signal with the same amount of energy as the targeted exploration signal generated from the optimization algorithm in Section 2.4.2 is used. The objective is to verify if the random exploration signal achieves the desired accuracy.

3.3.1 Generation of Normally Distributed Random Exploration Signal

The energy of the optimal exploration signal during the exploration time length T_{exp} is computed as

$$\gamma_{\text{sin}}^2 = \sum_{k=0}^{N_{T_{\text{exp}}}-1} \|T_{\text{eff}}^{\text{exp}}[k]\|^2 = 2012\,N^2\,. \tag{3.2}$$

Note that this energy is different from the cost function γ_e of the optimization problem, which is the energy bound at each time instant of the exploration signal. In order to do a comparison, a new exploration signal $T_{\text{eff}}^{\text{rnd}}$ is generated, starting from

$$x^{\text{rnd}}[k] \sim \mathcal{N}(0,1), \quad \text{with} \quad k = 0, \ldots, N_{T_{\text{exp}}} - 1\,,$$

it is possible to compute $T_{\text{eff}}^{\text{rnd}}[k]$ as

$$T_{\text{eff}}^{\text{rnd}}[k] = x^{\text{rnd}}[k] \cdot \frac{\gamma_{\text{sin}}}{\gamma_{x^{\text{rnd}}}}\,, \tag{3.3}$$

where $\gamma_{x^{\text{rnd}}}$ is computed as in Equation (3.2) but with the exploration signal x^{rnd}. In this way, it can be achived the same energy both for the optimal exploration signal computed according to the exploration technique of [1], and for the normally random distributed exploration signal generated in Equation 3.3. The new exploration signal $T_{\text{eff}}^{\text{rnd}}$ is shown in Figure 3.2.

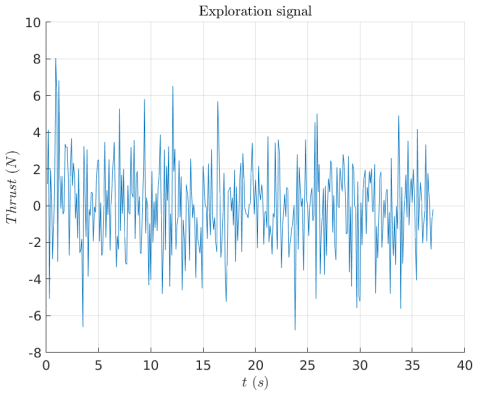

Figure 3.2: Normally random distributed exploration signal

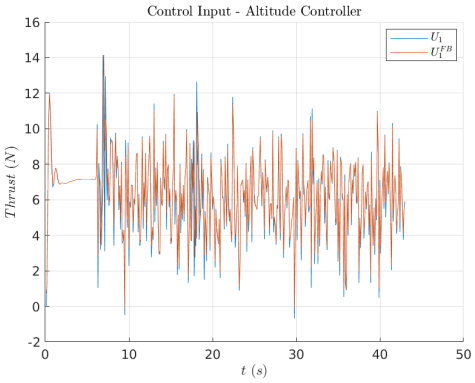

Figure 3.3: Exploration input

3.3.2 Exploration Phase

The exploration input U_1^{rnd}, computed as $T_{\text{eff}}^{\text{rnd}} + (\hat{m}_0 \cdot g)$, is used to excite the system and collect the data needed to estimate the coefficients of the altitude model. The resulting exploration input applied during the simulations can be seen in Figure 3.3. It is possible to notice that in this case, the tracking of the exploration input by the rotor is not as accurate as that obtained with the optimal exploration signal in Figure 2.9.
The evolution of the states $X = [z, \dot{z}]^\top$ is reported in Figures 3.4 and 3.5, respectively. As shown in Figure 3.4, during the simulation, there is a constant drift

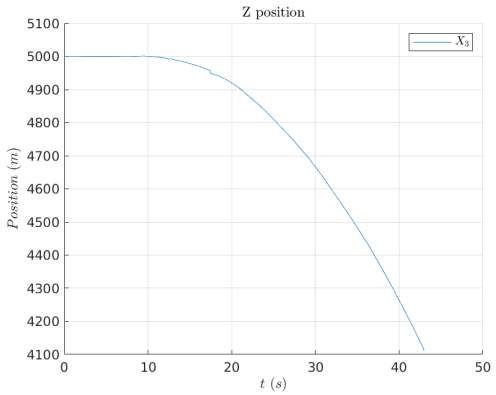

Figure 3.4: Evolution of the altitude

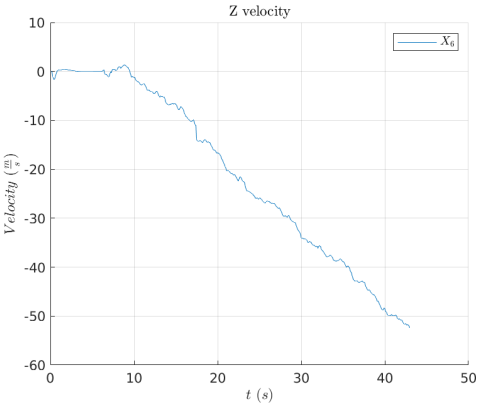

Figure 3.5: Evolution of the velocity

in the position due to the non-perfect compensation of the constant disturbance

caused by gravity. While in Figure 3.5, it can be seen how the exploration input is attenuated due to the nature of the system (low-pass filter).

3.3.3 System Identification

By collecting the data during the experiment (Figures 3.3, 3.4, and 3.5) and using only the data generated while the exploration signal is applied (from 6 s onward), it is possible to solve the standard least-squares problem to estimate the coefficients of the altitude model. The estimated state space representation is

$$\begin{bmatrix} x_1[k+1] \\ x_2[k+1] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.10003 \\ -2.685 \times 10^{-5} & 1.001 \end{bmatrix}}_{:=\hat{A}^{\mathrm{LS}}_{\mathrm{rnd}}} \begin{bmatrix} x_1[k] \\ x_2[k] \end{bmatrix} + \underbrace{\begin{bmatrix} 0.01219 \\ 0.0615 \end{bmatrix}}_{:=\hat{B}^{\mathrm{LS}}_{\mathrm{rnd}}} T^{\mathrm{FB}}_{\mathrm{eff}}[k]$$

where $T^{\mathrm{FB}}_{\mathrm{eff}}$ is computed as

$$T^{\mathrm{FB}}_{\mathrm{eff}}[k] = U^{\mathrm{FB}}_{\mathrm{rnd}}[k] - \hat{m}_0 \cdot g\,.$$

From $\hat{\theta}^{\mathrm{LS}}_{\mathrm{rnd}}$, it is possible to identify the estimated parameters

$$\hat{dT}_{\mathrm{rnd}} = 0.10003\,s,$$
$$\hat{m}_{\mathrm{rnd}} = 1.6266\,kg\,,$$

and the norm of its representing coefficient error results in

$$\|e_{\mathrm{rnd}}\| = \left\| \begin{bmatrix} A_{\mathrm{tr}\,1,2} - \hat{A}_{\mathrm{rnd}\,1,2} \\ B_{\mathrm{tr}\,1,1} - \hat{B}_{\mathrm{rnd}\,1,1} \\ B_{\mathrm{tr}\,1,2} - \hat{B}_{\mathrm{rnd}\,1,2} \end{bmatrix} \right\| = 0.0757\,.$$

3.3.4 Comparison of the Results

As computed for the optimal exploration signal, it is possible to verify if the condition of Equation (1.10) is satisfied,

$$G \cdot P_\theta \preceq D^{-1}_{\mathrm{des}}\,.$$

It can be defined the closeness matrix $(D^{\mathrm{rnd}}_{\mathrm{bd}})^{-1} := G \cdot P_\theta$, computed as explained in Section 3.2 with the exploration data collected using the normally random distributed exploration input. The matrix $(D^{\mathrm{rnd}}_{\mathrm{bd}})^{-1}$ results symmetric but negative definite, not satisfying the conditions that a closeness matrix should satisfy as in the case of the optimal exploration signal. However, even if Equation (1.10) is not verified, it is a sufficient condition and not a necessary one to ensure the objective of [1], thus this does not imply that the objective is not met. Accordingly, the following objective results satisfied:

$$(\theta_{\mathrm{tr}} - \hat{\theta}_{T_{\mathrm{exp}}})^\top D_{\mathrm{des}} (\theta_{\mathrm{tr}} - \hat{\theta}_{T_{\mathrm{exp}}}) = 0.5727 \leq 1\,.$$

To complete the analysis, using the knowledge of the true parameter values, it is also possible to compute the closeness matrix $D_{T_{\text{exp}}}^{-1}$ as

$$D_{T_{\text{exp}}}^{-1} = (\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}})(\theta_{\text{tr}} - \hat{\theta}_{T_{\text{exp}}})^{\top} ,$$

where again the inverse notation is used only for consistency of notation. Now it is possible to compute the exact maximum uncertainty,

$$\lambda(D_{T_{\text{exp}}}) = \frac{1}{\max(|\Lambda(D_{T_{\text{exp}}}^{-1})|)} = 174.6 ,$$

which is greater than the imposed desired uncertainty $\lambda(D_{\text{des}}) = 100$. However, it is highly reduced compared to the optimal exploration signal.

Conclusion

This thesis has presented the implementation and study of an existing system identification technique in the presence of an energy-bounded disturbance. To achieve a system modeled with these characteristics, the study of a quadrotor dynamic model has been performed, focusing the system identification only on the altitude dynamics. In order to perform the experiment, a SIL simulator has been developed, providing the capability to execute the optimization techniques for the generation of the exploration signal without constraints on the execution time while simultaneously evaluating the dynamic evolution of the quadrotor for an almost fixed discrete time. Moreover, to perform the system identification on the altitude dynamics, a stabilizing controller has been designed and implemented for the attitude model to satisfy the assumptions of the linearization and decoupling of the dynamics. The results of the simulation have shown that the exploration signal generated by the optimization techniques in [1] is able to reduce the uncertainty of the model and achieve the desired accuracy. Additionally, this result has been compared with another exploration signal, demonstrating that even if the new exploration signal has the same amount of energy, it is not able to satisfy the constraints of the exploration, resulting in lower accuracy.

Future developments of this work can be pursued in different directions. From the model perspective, system identification can be extended to other dynamics of the quadrotor, such as the position and the orientation. For the SIL simulator, a more consistent evolution of the discrete time can be achieved, or a complete interface with only ROS can be developed, avoiding the use of the MATLAB environment. Moreover, due to the interface of RotorS, it could be possible to test this technique in a real environment, like a flight arena, obtaining the odometry of a real quadrotor using sensors like Vicon. Finally, the techniques of dual control can be applied starting from the strategy of the exploration implemented, making it possible to build a robust controller based on the desired uncertainty obtained from the system identification techniques.

Bibliography

[1] J. Venkatasubramanian, J. Köhler, M. Cannon, and F. Allgöwer, *Towards non-stochastic targeted exploration*, 2023. arXiv: 2312.05947 [eess.SY].

[2] K. Peng, G. Cai, B. M. Chen, M. Dong, K. Y. Lum, and T. H. Lee, "Design and implementation of an autonomous flight control law for a uav helicopter," *Automatica*, vol. 45, no. 10, pp. 2333–2338, 2009, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2009.06.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109809003082.

[3] A. Isidori, L. Marconi, and A. Serrani, "Robust nonlinear motion control of a helicopter," *IEEE Transactions on Automatic Control*, vol. 48, no. 3, pp. 413–426, 2003. DOI: 10.1109/TAC.2003.809147.

[4] P. Wang, Z. Man, Z. Cao, J. Zheng, and Y. Zhao, "Dynamics modelling and linear control of quadcopter," in *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, 2016, pp. 498–503. DOI: 10.1109/ICAMechS.2016.7813499.

[5] Z. he and L. Zhao, "A simple attitude control of quadrotor helicopter based on ziegler-nichols rules for tuning pd parameters," *TheScientificWorldJournal*, vol. 2014, p. 280 180, Dec. 2014. DOI: 10.1155/2014/280180.

[6] A. Tayebi and S. McGilvray, "Attitude stabilization of a vtol quadrotor aircraft," *IEEE Transactions on Control Systems Technology*, vol. 14, no. 3, pp. 562–571, 2006. DOI: 10.1109/TCST.2006.872519.

[7] B. Erginer and E. Altug, "Modeling and pd control of a quadrotor vtol vehicle," in *2007 IEEE Intelligent Vehicles Symposium*, 2007, pp. 894–899. DOI: 10.1109/IVS.2007.4290230.

[8] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot Operating System (ROS): The Complete Reference (Volume 1)*, A. Koubaa, Ed. Cham: Springer International Publishing, 2016, pp. 595–625. [Online]. Available: https://doi.org/10.1007/978-3-319-26054-9_23.

[9] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

[10] G. Silano. (Oct 31, 2018). Crazys, crazyflie simulator - wiki, [Online]. Available: https://github.com/gsilano/CrazyS/wiki.

[11] E. Fogel, "System identification via membership set constraints with energy constrained noise," *IEEE Transactions on Automatic Control*, vol. 24, no. 5, pp. 752–758, 1979. DOI: 10.1109/TAC.1979.1102164.

[12] X. Bombois, G. Scorletti, M. Gevers, P. V. den Hof, and R. Hildebrand, "Least costly identification experiment for control," *Automatica*, vol. 42, no. 10, pp. 1651–1662, 2006, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2006.05.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109806002238.

[13] M. Barenthin and H. Hjalmarsson, "Identification and control: Joint input design and H-infinity state feedback with ellipsoidal parametric uncertainty via lmis," *Automatica*, vol. 44, no. 2, pp. 543–551, 2008, ISSN: 0005-1098. DOI: 10.1016/j.automatica.2007.06.025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109807003421.

[14] C. A. Larsson, A. Ebadat, C. R. Rojas, X. Bombois, and H. Hjalmarsson, "An application-oriented approach to dual control with excitation for closed-loop identification," *European Journal of Control*, vol. 29, pp. 1–16, 2016, ISSN: 0947-3580. DOI: 10.1016/j.ejcon.2016.03.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0947358016300024.

[15] M. Ferizbegovic, J. Umenberger, H. Hjalmarsson, and T. B. Schön, "Learning robust lq-controllers using application oriented exploration," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 19–24, 2020. DOI: 10.1109/LCSYS.2019.2921512.

[16] J. Venkatasubramanian, J. Köhler, J. Berberich, and F. Allgöwer, *Sequential learning and control: Targeted exploration for robust performance*, 2023. arXiv: 2301.07995 [eess.SY]. [Online]. Available: https://arxiv.org/abs/2301.07995.

[17] J. Venkatasubramanian, J. Köhler, J. Berberich, and F. Allgower, "Robust dual control based on gain scheduling," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 2270–2277. DOI: 10.1109/CDC42340.2020.9304088.

[18] Z. Tahir, W. Tahir, and S. A. Liaqat, *State space system modelling of a quad copter uav*, 2019. arXiv: 1908.07401 [cs.RO].

[19] T. Nguyen, I. Prodan, F. Stoican, and L. Lefévre, "Reliable nonlinear control for quadcopter trajectory tracking through differential flatness," vol. 50, Jul. 2017. DOI: 10.1016/j.ifacol.2017.08.1338.

[20] F. Furrer. (Apr 27, 2017). Rotors simulator - wiki, [Online]. Available: https://github.com/ethz-asl/rotors_simulator/wiki.

[21] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: An open-source robot operating system," in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[22] M. M. Lund University Department of Automatic Control. (4 February 2020). Sampling of linear systems, [Online]. Available: `https://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTN01/lectures/L06_slides1.pdf`.