UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

SYSTEM IDENTIFICATION OF A DRONE MOUNTED ON A GYROSCOPIC TEST BENCH

MASTER FINAL THESIS

DOCUMENT:

Master's Report

AUTHOR:

Di Giuseppe, Gabriel Adrián

TUTOR - COTUTOR:

Pérez Magrané, Ramon - Morcego Seix, Bernardo

DEGREE:

Master's degree in Automatic Systems and Industrial Electronics Engineering

EXAMINATION SESSION:

Autumn, 2023

ABSTRACT

This dissertation applies an experimental methodology to study and identify a quadrotor mounted on a gyroscopic test bench. Its aim is to find suitable data-driven linear time-invariant (LTI) models in both continuous and discrete time for pitching, rolling and yawing motions separately and to validate them.

A literature review is done to assess the significance of UAVs in general, the interest in them, and their impact. The work also studies the current methodologies utilized in research to study, identify and control UAVs. Prior documentation and knowledge of the chosen drone is assessed to understand the building foundations of the dissertation.

An experimental, drone-independent methodology is developed for the work. New user-friendly interfaces and tools for experimentation and control of the drones at UPC ESEIAAT are developed in the MathWorks' framework, which are independent of the selected approach. Guidelines for them are written for future use. In accordance with it, basic physical parameters (i.e., friction coefficient and inertia) are estimated based on an equation that explains the motion of the drone-gyroscope system without the actuations of the motors.

Thereafter, data is collected in a system identification framework, and model structures are chosen. The models are then estimated for pitch, roll and yaw separately. Based on them, Proportional-Integral-Derivative (PID) controllers are tuned using the pole placement method and iterative algorithms. Moreover, the models are successfully validated in the platform with the help of the attitude controllers.

The platform demonstrates to be a useful tool for both educational and research purposes. It allows the implementation of the proposed methodology in a controlled environment. Also, the system shows signs of nonlinear behaviour, such as nonlinear proportional gains and dead zones. The first two angles linear dynamics were identified to be of second order, while the yaw proved to have a natural integrator, together with a first order dynamic. In accordance with the influence of the gyroscope's inertia on the drone, the roll was the fastest, and the pitch was in a middle point. Both proved to have one equilibrium point, while the yaw had an infinite amount. Additionally, the rotational friction was pinpointed as the cause of its dead zone, which provoked an offset during the validation phase.

RESUMEN

Este trabajo final de máster aplica una metodología experimental para estudiar e identificar un cuadricóptero montado en un banco de pruebas giroscópico. Su objetivo es encontrar modelos lineales invariantes en el tiempo (LTI) basado en datos en tiempo continuo y discreto para cada ángulo de Euler y validarlos.

Se realiza una revisión del estado del arte y literatura para evaluar la importancia de los Vehículos Aéreos No Tripulados (UAVs), el interés en ellos y su impacto. El trabajo también estudia las metodologías utilizadas en la investigación para estudiar, identificar y controlar los UAV. Se evalúa la documentación previa y el conocimiento del dron elegido para comprender las bases del TFM.

Se desarrolla una metodología experimental independiente del dron. Se desarrollan nuevas interfaces para la experimentación y control de los drones en UPC ESEIAAT dentro de Matlab y Simulink, independientes del enfoque seleccionado. Se redactan guías para su uso. Se estiman parámetros físicos (coeficiente de fricción e inercia) en base a ecuaciones que explican el movimiento de sistema sin las actuaciones de los motores.

A continuación, se recopilan datos para la identificación del sistema y se eligen las estructuras del modelo. Luego, los modelos se estiman por separado para cabeceo, alabeo y guiñada. En base a ellos, se ajustan controladores Proporcional-Integral-Derivativo (PID) utilizando el método de asignación de polos y un algoritmo iterativo. Además, los modelos se validan exitosamente en la plataforma con la ayuda de los controladores de actitud.

La plataforma demuestra ser una herramienta útil tanto para fines educativos como para la investigación. Permite la implementación de la metodología propuesta en un entorno controlado. Además, el sistema muestra signos de comportamiento no lineal, como parámetros no lineales y zonas muertas. Las dinámicas lineales del cabeceo y alabeo se identificaron como de segundo orden, mientras que la guiñada demostró poseer un integrador natural, sumada a una dinámica de primer orden. El alabeo demostró tener la dinámica más rápida, con el cabeceo en segundo lugar. Ambos poseen un punto de equilibrio, mientras que la guiñada tenía una cantidad infinita. Además, se identificó que la fricción rotacional fue la causa de su zona muerta, lo que provocó un desplazamiento durante la fase de validación.

INDEX

TABLE INDEX

EQUATION INDEX

FIGURES INDEX

GLOSSARY

AscTec	Ascending Technologies GmbH
BS	Back-Stepping
CTC	Coordinate Transformation Conversion block
DOF	Degree Of Freedom
EC	European Commission
ESC	Electronic Speed Controller
ESEIAAT	Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa
ETH	Swiss Federal Institute of Technology (Eidgenössische Technische Hochschule)
EU	European Union
FLC	Fuzzy Logic Controller
GHG	Green-house gas
GWP	Global Warming Potential
HLP	High-level processor
IAE	Integral Absolute Error
IMU	Inertial Measurement Unit
IP	Internet Protocol
ITAE	Integral Time Absolute Error
LCV	Light commercial vehicles
LiPo	Lithium Polymer (battery)
LLP	Low-level processor
LQR	Linear Quadratic Regulator
LTI	Linear time-invariant
MIMO	Multiple-input multiple-output
MSE	Mean Square Error
NRMSE	Normalized Root Mean Square Error
OS	Operating System
PV	Process value
ROS	Robot Operating System
RST	Reference Signal Tracking

SF	State Feedback
SISO	Single-input single-output
SITB	MathWorks' System Identification Toolbox
SP	Set point
SSH	Secure Shell
UAV	Unmanned Aerial Vehicle
UPC	Universitat Politècnica de Catalunya

1. INTRODUCTION

This dissertation aims to apply an experimental methodology to study and identify a quadrotor mounted on a gyroscopic test bench. Its aim is to find suitable data-driven linear time-invariant (LTI) models in both continuous and discrete time for each separate Euler angle of the system and to validate them. The main feature of the work is that it develops Unmanned Aerial Vehicles (UAVs) models and tests flight controllers in non-destructive experiments under controlled conditions using data-driven system identification techniques.

The main equipment consists near-human-size gyroscopic test bench and a quadrotor which is mounted on top (Figure 1). This allows the study of the system's mathematical and physical properties. The test bench consists of three outer rings or gimbals (Figure 2).

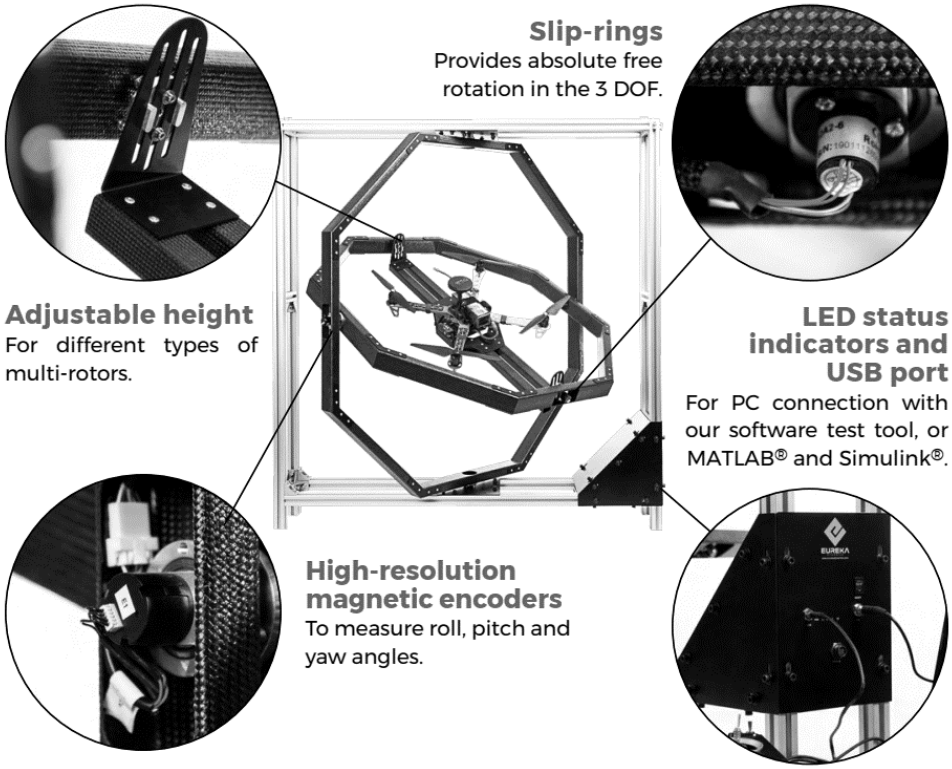

Figure 1: Gyroscopic test bench with mounted quadrotor

Source: [1]

For these purposes, the laboratory had an Ascending Technologies Hummingbird quadrotor (Figure 3), which was modified by the professor [2] with the help of former students [3], [4]. The modification consisted of an additional board called Odroid-XU4 [5]. It

had Ubuntu 16.04 installed and could run the Robot Operating System (ROS) middleware. The user could then connect to this ROS network through Wi-Fi in a safe SSH (Secure Shell) connection [6], which is also possible thanks to the board. Further details of the quadcopter, and its hardware and software configurations can be found in other studies done at the institution [7]–[9].

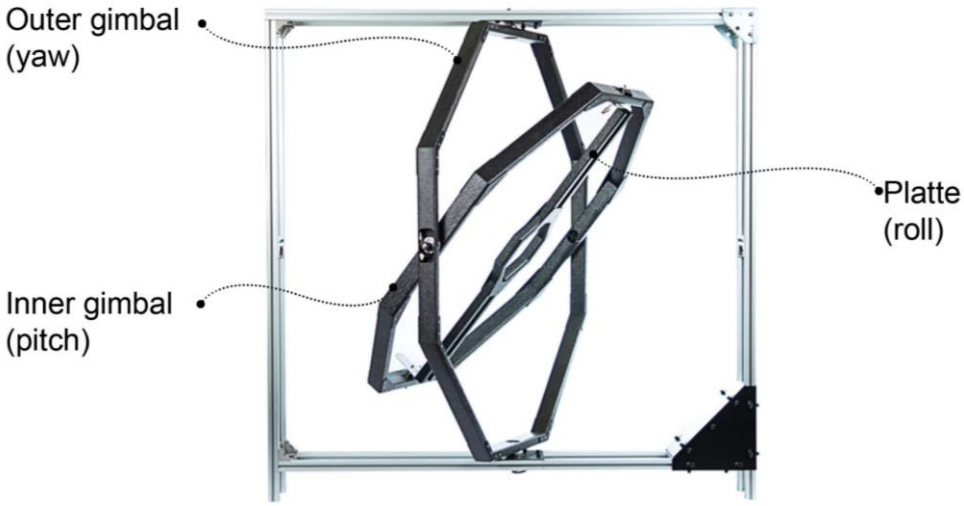

Figure 2: FFTGyro at UPC ESEIAAT

Source: Manufacturer's website

The platform's manufacturer Eureka Dynamics provided three 10-bit, 360° encoders, which are equally distributed across the ring's spinning axes (Figure 1). Additionally, step motors also were provided to the university (Figure 2), which are interchangeable with the commented encoders. The motors can also provide readings of the Euler angles, although they have a dead zone between 280 and 360°. Unfortunately, the system cannot operate with a mixture of both actuators and sensors. The motors will not be used in this work. Eureka also provided a way to easily collect data through a serial communication (USB connection) built into the machine. The encoders' readings could then be displayed through Matlab scripts or Simulink interface for further analysis, such as parameter estimation and model identification.

Figure 3: Modified AscTec Hummingbird quadrotor mounted on gyroscopic test bench at UPC

1.1 OBJECTIVES

The main objectives of this dissertation are:

- To setup the communications channel between (chapter 4):
  - ROS ←→ Matlab/Simulink
  - Gyroscope ←→ Matlab/Simulink
- To develop user-friendly interfaces for the communications with the quadrotor through ROS, scripts, and Simulink interfaces for use in further research and educational purposes (chapter 4, section 7.2 and Appendix C).
- To determine, calculate, and estimate missing physical parameters of the platform with the mounted quadrotor (chapter 5).
- To identify the system's parametric models (chapter 6) in continuous and discrete time alike.
- To develop flight controllers for the quadrotor-gyroscope system based on the identified models and test them on the real platform (chapter 7) to further validate the models.
- To write a manual on how to properly experiment with the system and test controllers in the platform (Appendix C).

1.2 SCOPE

The scope of the dissertation is to develop a methodology to test and validate drone models through system identification and control techniques. In system identification theory, these kinds of parameterized models are called ‘ready-made’, which are of general applicability, have no physical interpretation and are expressed in discrete time [10]. Moreover, process models are continuous time models [11], analogous to the ‘ready-made’. On the other hand, analytical models (‘tailor-made’ models in the literature) are constructed from basic physical principles and will not be within the scope of this work. These models are the most common in the scientific community for UAV identification and control [12]–[16]. The term ‘system’ will be referred to the quadrotor mounted on the test bench (Figure 3) throughout the text, unless specified otherwise.

The ‘ready-made’ and process models are going to be estimated, parametrized, and validated inside the laboratory based on gathered experimental data (i.e. ‘data-driven’). Additionally, flight controllers are going to be selected, developed, analytically tuned, and tested in controlled conditions in the laboratory based on these models. This will be considered the final validation phase of the system identification process. The drone is always going to be mounted on the platform (drone-gyroscope system) throughout the experiments. The data will be gathered through safe, non-destructive experiments given the unique opportunity to test the system repeatedly without damaging the UAV or its user. For that reason, UAV simulations are not within the scope of work.

A decoupler has already been developed by Ascending Technologies, the drone’s manufacturer, and will not be part of the scope of this work. In this case, the decoupler assists the user by enabling the indirect control of the quadrotor motors’ voltages, thus allowing the user to control the pitching, rolling and yawing motions of the drone. However, it should be noted that the decoupler does not replace the flight controller, which still needs to be developed and is within the scope (chapter 7).

Due to safety reasons, the dissertation will not cover field tests of the controllers developed in the laboratory. As mentioned in the section 1.3, it is required by the university that a person with piloting experience in remote control of UAVs to conduct field tests. Instead, that time has been dedicated to experimentation, validation of the results, and development of interfaces with their respective guidelines. In addition, the dissertation focuses on research and development of several techniques and methodologies (chapter

3) instead of field usage. Nevertheless, the controllers could be tested in the field in the future. To assist the development of methodologies and techniques, user-friendly interfaces will developed in the MathWorks' framework (chapter 4 and section 7.2). The main controller will be a Proportional-Integral-Derivative (PID) controller. However, the resulted models could be used for many other techniques, under similar conditions (section 1.4).

Although the results centred around one of the university's AscTec Hummingbird UAV available, the approach applied in the dissertation could prove useful to any UAV in any bench test. The drone must have angular velocity and position sensors (an IMU is sufficient), communications capabilities to its remote controller and/or a computer, and a (preferably gyroscopic) test bench, such as the ones found here or in [14]–[17]. The test bench must comply with basic safety standards to be able to evaluate the drone with any input, especially high frequency white noise such as the one utilized in chapter 6. It is not a requirement that the test bench has encoders, but it certainly helps to have a second output to the drone's sensors to contrast and compare both measurements. If safety standards are not met, the approach cannot be applied and the standard paradigm of model-based testing of UAVs should be further utilized [18]–[20].

In addition to that, similar results of this work can be reproduced through many system identification algorithms in different frameworks. The following examples could be mentioned: Simulink's System Identification toolbox (SITB) [21], a Python open library [22] or a C++ open library [23]. The interface developed for the dissertation can only be used for AscTec drones employing the ROS package 'mav_framework' [24].

1.3 LIMITATIONS

The following limitations are observed at the beginning of the dissertation: a) hardware, b) software, and c) economy (of time and money).

Three quadrotors are available at the university, all of which were bought at the same time to Ascending Technologies GmbH. Two of them have received hardware and software updates but have different morphologies and physical characteristics. Nevertheless, all of them share the same basic components: motors, propellers, and so on. The third has been discarded because it does not meet the requirements of hardware and software (section 1.4).

Since the identification process is of experimental nature, it is not allowed to interchange the two remaining drones. One drone is to be chosen for the study. This could prove to be a limitation in the future if there is any problem with the chosen drone. For that reason, it is advisable to proceed with caution when dealing with it. Therefore, its electronics, cables and other sensitive components should be handled with extra care; for example, it is imperative that the propellers do not collide at any moment with the cabling. If a problem were to arise in the drone used in the dissertation, the proposed work would have to start again.

Additionally, the university has limited number of spare parts for its drone, and no spare parts are available in the market. More on the subject is spoken about at the end of chapter 2.

The batteries for both the quadrotor and the remote controller are swollen, which is an indication that they are old and/or were misused. A limitation then arises with the estimated time of flight (or rather, experiment time) that each battery will provide. To counteract part of the problem, the remote controller will be kept plugged in and charging at all times during the tests.

On top of that, according to the person responsible for the laboratory, the gyroscope had presented software and hardware difficulties in the past. This leads to believe some difficulties might arise in this subject. Note that the platform's manufacturer is from Mexico. A hardware problem could delay the dissertation significantly until a technician fixes it.

Combining all of the above limitations, if the stipulated problems were to arise during the dissertation, solving them could cost money and time, and would therefore diminish the effective time to study the quadrotor-gyroscope system.

1.4 REQUIREMENTS

Basic requirements of the dissertation are:

- ROS-enabled UAV with built-in Inertial Measurement Unit (IMU).
- Test bench with encoders.
- Linux/Ubuntu 16.04 installed on a computer.
- Matlab/Simulink R2020b license with full access to MathWorks' toolboxes.

- Drone batteries with charger.
- Drone controller with charger.

1.5 JUSTIFICATION

In the last twenty years, there has been an exponential increase in interest in Unmanned Aerial Vehicles (UAVs), which have gained the attention from the scientific community [25], [26]. These vehicles offer high flexibility and manoeuvrability, making them attractive for various applications, mostly outdoors. For example, they have been used for reforestation [27], agriculture [28], reconnaissance and rescue [29], and aircraft inspection [30]. In 2018, due to their rapidly growing applications and interest, the EU saw the necessity to implement regulations and civil aviation rules for all UAVs, regardless of their weight [31].

Despite their incredible engineering and scientific potential, UAVs have been difficult to study and control, given their mathematical and physical complexity [32]. Nowadays, the scientific community has perfected several models that explain their behaviour [18], [19]. These are commonly utilized to develop flight controllers for quadrotors and other types of UAVs [25]. Despite that, the equations of the open-loop system are mathematically hard to comprehend and is easy to get lost in physical properties, like the inertia tensor matrix of the system [33]. Their difficulty increases after the flight controller is developed, making the closed-loop system physically unintuitive.

The current methodologies for studying and controlling UAVs are based on analytical (‘tailor-made’) models, in which only the parameters are unknown [32]. This is often called the conventional model-based approach. On the other hand, the model-free approach has been understudied [34]. This dissertation aims to provide a different data-driven approach on the study, validation, and control of UAVs, which does not rely on open-field flight data nor high-fidelity simulations.

To implement the approach, new tools are going to be developed and implemented. The approach will also lower the amount of experience, expertise and funds required to study UAVs. Additionally, their goal is to provide a suitable safe and user-friendly environment for educational and research purposes. In the future, new flight controller’s tuning techniques could be studied, and their results fined-tuned before they are tested in the open field.

2. LITERATURE REVIEW

This chapter is dedicated to the analysis of the literature related to UAVs, their importance, common control and validation techniques, and the economic impact of drones and test benches. Additionally, prior documentation of the available quadrotor and test bench was studied to assess the possibilities and extent of the dissertation.

2.1 INTEREST AND APPLICATIONS

In recent decades, Unmanned Aerial Vehicles have developed quickly. Because of this, they have become quite popular for both military and civilian purposes. They enable to reach previously inaccessible locations while also saving time and lives [35]. Their growing interest amongst industries and society alike is demonstrated through their implementation into various human activities such as:

- Agriculture [28]: five key areas encompassing the primary agricultural industry are to be benefited from the use of UAVs: topographic mapping, physiological evaluations, biophysical analyses, surveillance of living organisms, and the application of plant protection products and biological inputs.
- Reforestation [27]: an operation was able to plant an additional 4 million mangrove plants in 2019 alone with the help of drones. Ten drones were flown by two operators at a rate of up to 400,000 trees each day.
- Rescue missions [29]: most recently, the company Garuda Aerospace deployed their UAVs on earthquake-hit Turkey, which saved countless human lives.
- Inspection of aircrafts [30]: generation of 3D CAD models of aircrafts for maintenance routines such as fault finding, which replaced manual work prone to error.
- Supply chain, logistics and warehouse management [36], [37]: they have reported to: a) support humanitarian logistics, b) shorten delivery time and c) augment sustainability, d) reduce cost and e) augment flexibility.

These applications are only a few of many. The environmental and social impact both positive and negative are analysed thoroughly in Appendix B.

2.2 ECONOMIC TERMS

In economic terms, the interest in UAVs has been highlighted in many reports, science magazines articles and papers. The European Commission (EC) has set a drone strategy for the European large-scale market in November 2022 [38]. The UAV industry is predicted to create a direct workforce of over 100,000 individuals and generate an economic impact of more than €10 billion annually in Europe, primarily through services, within the next two decades [31], [39]–[41].

This has captured the attention of the university, which is committed to allocating resources to develop expertise and tools in the field. Fortunately, a huge savings possibility has opened up with the new equipment bought at the UPC. This dissertation aims to provide the necessary tools to mitigate the economic endeavour that it represents the study and control of drones in open field. This, in turn, would enable the university to offer a strong platform for students, fostering the development of practical and valuable skills that are relevant in the industry.

2.3 IMPORTANCE OF THE GYROSCOPE

Studying the gyroscope in particular, it could prove to be reliable and safe tool, while being economically efficient for the mentioned purposes. Several studies have assessed its potential benefits, to the extent that numerous institutions have decided to build their own test bench [14]–[16], [42]–[44]. According to the articles, the gyroscope unlocks several new data-driven techniques that were previously not available given their characteristic. One of them is the model-free approach. The test bed has also been implemented by these institutions to instruct students about the fundamentals of autonomous control in multirotor aircraft, as it is intended at Universitat Politècnica de Catalunya. They offer a transitional stage between the UAV's simulation process and real-time flight; they enable testing and solution evaluation in a controlled and safe environment. Thus, several methods to study and control of UAVs are now unlocked by its potential. The mentioned equipment utilized during tests with multiple control loop constructions was reported by the mentioned studies to have reduced the danger of causing damage during field testing both to the drone and the users.

Additionally, the test bed has been reported to have little effect on the UAV moments of inertia. It also allowed the vehicle to maintain its individual degrees of freedom for angular stability dynamics (rolling, pitching, and yawing) with little to no interference from it [16]. The shape of the gyroscope has been reported to be the best suited for safe testing (Figure 4), after a thorough comparison between different test benches [45].

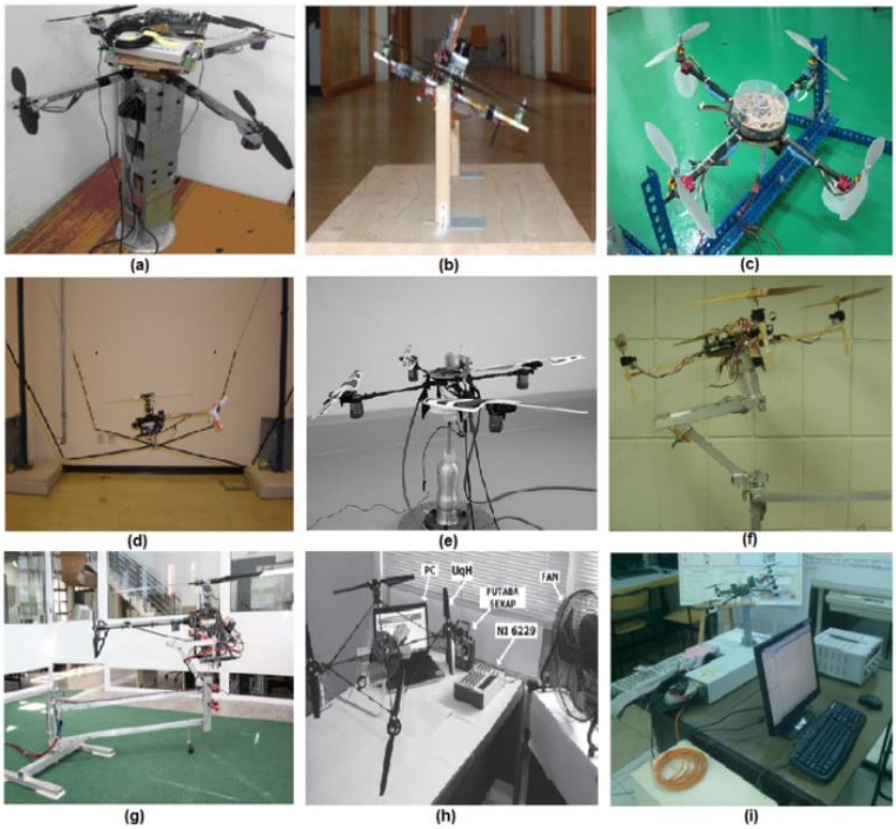

Figure 4: Different test benches possibilities

Source: [45]

The development of a user-friendly Graphic User Interface (GUI) for the UAVs and test bed has also been particularly helpful for sending test commands and receiving data from all sensors. Test GUIs allowed users to view the graphical analysis of the data swiftly and easily [45], which reportedly reduced the complexity of the system overall and opened up new possibilities in the field. The dissertation aims to provide one, suited to the university's needs.

Additionally, studies have shown a worrying rise in drone accidents in conjunction with the spike in public and private interest. It is then crucial to better assess drone controls since

many of these mishaps are due to operational problems and could be avoided [46], while keeping in mind the benefits presented on this subsection.

2.4 DRONE STUDY AND CONTROL TECHNIQUES

This section comments on popular study and control techniques utilised to better understand its application framework and select one suited to the conditions of the dissertation. Parallelly, a literature review on the required software is carried out.

It is important before diving into this subsection that the reader understands the following concepts. More information on the concepts can be found on chapter 6. There are three kinds of parametrised models which are commonly used for system identification [10], [11]:

a) Tailor-made: constructed based on known physical phenomena or properties that affect a given system. Then, the parameters explain the extent to which these phenomena manifest themselves. Control techniques based on them are typically referred to as ‘model-based’. They are typically expressed as discrete time.
b) Ready-made: family of parametrised models that do not necessarily have a physical explanation, in comparison with tailored models. Their structure is used as vehicles to describe the general dynamics of the real system and work on an input-output structure. Algorithms try to find the best suited constant values for their parameters. The number of them is imposed by the user. They are strongly data-driven models and sometimes referred to as ‘grey-box’. They are typically expressed as discrete time.
c) Process models: are common in many industries for modelling system dynamics and are applicable to different production contexts. These models' benefits include simplicity, help for estimating transport delays, and straightforward interpretations of the model coefficients as poles and zeros. They are expressed in continuous time.

Typically, a mathematical model of the controlled aircraft is necessary for model-based tuning techniques in order to accurately map the system's highly complex and nonlinear dynamic behaviour. In reality, it is very challenging to create an accurate model, and even

minor modelling errors can result in compromising control performance or an unstable closed-loop system [47], [48].

The current paradigm for the dynamic nonlinear system modelling, control strategy design and testing through simulations is found on [20]. The article describes a model-based approach to solving the problem of identifying a tri-rotor UAV. The work then proposes common control techniques to the UAV field: linear quadratic regulator (LQR), back-stepping (BS) and fuzzy logic controller (FLC) designs. It later validates the models through simulations done in closed-source software and does not put it to test on a real platform nor on open field. This is the case for many papers in the scientific community [25], [26], [49]. Some papers follow the same paradigm (nonlinear approach), and later validate their results through control techniques implemented in real flight using a test bench [50] and some on open field [7].

On the other hand, although the ‘grey-box’ approach has been studied before as a mechanism to reduce the amount of time and experience required to develop flight controllers, the algorithms have not been tested in the open field or even in a controlled environment such as a gyroscopic test bench [34], [47], [48], [51]. In summary, there seems to be a gap from the study of the ‘ready-made’ models outside of simulation software and in test benches.

Furthermore, to implement a manoeuvring control model and accomplish various altitude and attitude objectives (setpoints, SP), traditional proportional-integral-derivative (PID) control or a linear quadratic regulator (LQR) optimal control approaches have been frequently employed. PID controllers are frequently utilized in a variety of industrial settings, including aircraft manoeuvring control, thanks to their straightforward design and understandable parameter definitions [43], [52]. Nevertheless, the fine-tuning of their parameters can be challenging and time-consuming because of the need to balance the effects of each term to satisfy the transient response of the system. Since the new platform allows endless testing, the controllers (whichever their structure is) could be fined-tuned. Although the possibility is available, it proves to be an ambitious goal for the work given the time constrains, for which the activity will not be present in the objectives. Moreover, the

As mentioned before, PID controllers are widely employed in literature and they have been applied to UAV altitude and attitude control in many scenarios, such as a gyroscopic

testbench, as is the case here. Additionally, to avoid the time-consuming effort of the nonlinear model-based (‘tailor-made’) approach, a linear data-driven (‘ready-made’) approach is chosen. For that reason, they are chosen as a starting point for the implementation of the platform at UPC. This is intended to make use of the benefits of the gyroscope to the dissertation’s advantage. The chosen methodology is expected to allow the identification of the AscTec Hummingbird quadrotor’s model on proven practical structures, as explained earlier in the section.

Moreover, techniques such as state-feedback and LQR have also been present in the literature, which could also be tested in the platform on further studies, as have done others in similar settings [17]. Further venues of studies in the conditions of the dissertation are found on section 8.2.

Parallelly, the need of an open-source simulation software was assessed in a survey at [53], where the use of proprietary closed-source flight controllers on many UAV platforms and the lack of standardized flight controller architectures reported to render the work on UAV a paramount challenge as to port one paper’s solution to another without extensive additional development and testing. The quadrotor will not be 3D simulated for the dissertation, as the gyroscopic platform enables testing on three DOF of the equipment. In regards to the identification software to be used, although the MathWorks’ framework is proprietary, it offers a compact solution for the identification process [21] in the form of a GUI. This is not the case for open-source libraries [22], [23], which offer relatively low possibilities. For that reason, this software will be utilized in this case.

2.5 MAIN BIBLIOGRAPHY, PRIOR DOCUMENTATION AND KNOWLEDGE

After the conscious definition of the objectives (section 1.1) with the help of the literature review, it is imperative to choose the right bibliography to tackle them. The following books and research papers [10], [11], [54]–[57] will play a pivotal role in laying the foundation for this dissertation on the topic of parametric model system identification, serving as key references and authoritative sources. They provide the necessary insights and knowledge essential to achieve the objectives. Particularly, the parametric model-free approach is studied for its correct application in the work. It contains information on both continuous and discrete time systems alike (‘ready-made’ and process models). The designed experiments

employed in the dissertation will be a direct consequence of the contents of the mentioned references.

Furthermore, [58] can be employed to analyse the experimental results (chapter 5). After the necessary knowledge and insights are gathered, the identification and initial validation experiments can be put to test. Concepts of control theory will be extracted from [59]–[63] to understand the underlying dynamics of the models that were identified.

Moreover, the available AscTec Hummingbird quadrotor had prior documentation, analysed thoroughly. Some parts of which were published and others were not [7]–[9], [64]. The necessary parts of them were explained throughout the chapters (e.g. section 7.2). The design of the flight controllers was done according to the digital implementation of them explained on section 3.6 of [65], which was referenced in the mentioned documentation.

In regard to the test bench, the gyroscope's manufacturer provided documentation, which included tables of parameters, 3D models, scripts, and models for the MathWorks' framework. This allowed its basic implementation. A detailed guide for its utilization in further studies was developed for the dissertation and can be found on Appendix C.

A final remark: An important fact was unravelled after a search on Ascending Technologies GmbH, the drone manufacturer. Unfortunately, they were bought and dismantled by Intel in January 2016 [66], which has prevented further baseline improvements to the hardware and software. One example is the brushless motors' drivers, whose last update was in 2016. Other examples are the Electronic Speed Controllers (ESC), motors and propellers (blades), which were custom-made by AscTec's suppliers. Their production is discontinued. This means that the equipment needs to be handled with extra care because no replacement can be ordered, aside from the few spare parts that the university still owns.

3. METHODOLOGY

This chapter is dedicated to the explanation of the work methodology chosen to conduct the study. It employs a quantitative approach to collect, analyse, and present the data and results. The dissertation aims to provide a practical solution to the system identification and control of drones in a safe and controlled environment. In order to achieve this, the development of new tools and interfaces was necessary. This chapter only focuses on the step-by-step process of the dissertation and description of it. The theoretical framework and in-depth analysis will be presented within each relevant chapter. This was done to improve readability and to familiarize the reader with the topics and theory at each chapter.

The equipment utilized for the study was: a) modified AscTec Hummingbird quadrotor, b) Futaba FF-7 2.4GHz remote controller, c) 2300 mAh 11.1V LiPo batteries, d) Eureka Dynamics' gyroscopic test bench FFT Gyro 450 Pro Aluminium and its relevant documentation, e) HP ProBook 450 G5 laptop with Ubuntu and ROS software, and f) full licence for MathWorks' products. The drone was selected because it was available at the university, and it had been studied before within the institution [7]–[9]. This meant that there was already information and tools available that could be utilized as a base for this research. The gyroscope was not previously studied nor utilized at the university. The test bench permitted the development of an experimental framework without the risks and costs associated with the test of the UAV in open field.

The first step in the process was to gather the information available and what had already been studied about the quadrotor and the test bench. The study of the available documentation of the equipment showed that the documentation described the function of both to some extent. In particular, there was documentation about the UAV, which included information about the scripts to be used in an experimental framework. The test bench's manufacturer had also developed scripts and models for the MathWorks' framework, which were improved upon.

The next step was to read the scripts upon which the experimental interface was based on, which primarily ran on C++ coding. The code utilized the ROS middleware to build up communications between the computer and the quadrotor. The communications were then meant to be carried out through the MathWorks' framework. For that, the test bench's manufacturer was contacted.

Meetings with the manufacturer were conducted to understand the equipment. Matlab scripts and Simulink interfaces were obtained from these meetings and the manufacturer's GitLab [67], which allowed the communication between test bench and computer through MathWorks' framework. Documentation regarding the gyroscope were also obtained from these meetings. This included physical parameters (Table 1 and Table 2) and 3D models.

Utilizing the information gathered up to that point in time, the new interface could be built. As explained earlier, the decision was taken to do it through the MathWorks' framework. New tools, scripts, models, and guidelines were developed to that end (chapter 4 and Appendix C). Their end goal was not only to serve the main purposes of the dissertation, but to set up a user-friendly interface for educational and research purposes at Universitat Politècnica de Catalunya. The tools have already been implemented in several courses.

Employing Newton's second law of motion for rotation, an equation was developed based on physical and mathematical analysis of the quadrotor-gyroscope pair (Equation 1). Initial experiments were then conducted based on this equation and the parameters obtained from the gyroscope's manufacturer. A series of observations, hypotheses and new parameters were produced from these initial experiments (chapter 5), which were then utilized as prior knowledge to design experiments and understand the results of chapters 6 and 7.

The new series of experiments was conducted based on the results of the initial experiments (section 6.1). This was an iterative process, where different hypotheses were tested, and the experiment design was redefined as the system identification progressed. Parametrized models for each Euler angle (pitch, roll and yaw) in continuous and discrete time were identified from these experiments. They were initially validated according to the system identification methodology (section 6.2).

The final validation process was based on the parametrized models, which were utilized to develop the quadrotor's controllers (chapter 7). A new interface for the controller's tests was developed. This enabled the final validation of the methodology utilized for the dissertation. This new interface was accompanied with guidelines, which also had educational and research purposes in mind (section 7.2 and Appendix C, item VIII).

Based on the knowledge gathered during the study, conclusions were drawn (section 8.1). New study venues were advised for the future (section 8.2).

4. EXPERIMENTATION INTERFACE

This chapter is dedicated to the development of the communications and experiment interface for the dissertation. It also shows how its results were affected by the past experiment interface, which was developed as executable C++ scripts. The section will also deal with communication tools such as the MathWorks' framework, the Robot Operating System (ROS) and the programming language C++ to communicate with the drone and platform more flexibly and easier throughout experimentation. For a more intelligible understanding of the workflow in the upcoming explanation, the progress will be presented as faithful to the chronological order as possible.

As a starting point, the AscTec Hummingbird quadrotor contained code written in C++ in its memory, developed by the Swiss Federal Institute of Technology (ETH) in Zurich, with relevant contributions made by the university professor and dissertation' co-tutor, which has been used for previous papers involving the UAVs at Universitat Politècnica de Catalunya [7]–[9], [68], [69]. Those changes and additions have been documented and they include the coding of helpful control modes for the quadrotor, such as the open loop ('DirectMotorControl' mode in the drone's documentation) explained on section 4.4, and a PID control algorithm, which will later be implemented as the flight controller (chapter 7 and Appendix C, item VIII). Unfortunately, the quadrotor's manual and documentation developed at UPC have not been published and cannot be referenced.

Although papers, articles and studies have used the laboratory's UAVs, the gyroscopic test bench was not utilized previously due to hardware problems. As the current work started, the problems had to be resolved; an example for a software problem is explained on the paragraph above Figure 19 on section 4.4. The main hardware problem consisted in unforeseen severed cables, which the manufacturer had to replace.

4.1 REQUIREMENTS

Aside from the platform and the quadrotor themselves, the following requirements were needed to develop the communication tools and channels explained in this section:

- Laptop with Ubuntu 16.04 OS and ROS 1 Indigo releases.
- Python 2.7 and CMake 3.10.x.

- Matlab R2020b (modified) with the following add-ons:
    - ROS Toolbox
    - Robotics System Toolbox
    - Signal Processing Toolbox
    - Instrument Control Toolbox
    - Control System Toolbox
    - System Identification Toolbox
    - UAV Toolbox

Matlab's modifications will be explained in subsection 4.3.2.1. Detailed hardware requirements (e.g., batteries) are found on Appendixes A and C.

4.2 DEVELOPMENT OF THE INTERFACE

A user-friendly interface for the experimentation with the drones at UPC was needed to alleviate the amount of background knowledge and experience in both control theory and programming that was required to carry out experiments. The development of one in a familiar setting for professors, students, and researchers at UPC ESEIAAT was necessary. MathWorks' products presented themselves as an ideal substitute to replace the former interface the university had developed. Note that the university has full licence to the software.

Initially, a control manual for the drone was facilitated by the professors, which documented every step of the code execution process in an Ubuntu OS. It described a short but useful guide on how to work with the UAV and its framework in open loop. Because the manual was prequel to the existence of the gyroscopic test bench, it proved insufficient for experimentation in the current setting.

Therefore, an extensive step-by-step guide was therefore written as part of the dissertation's objectives. It contains the latest advances in the experimentation process and can be found in Appendix C. It was developed as a standalone document to be used for educational and research purposes. This will avoid having to read through the main body of this dissertation. As explained earlier, this chapter only explains the decision process of the interface. Professors have already developed laboratory classes for the course

'Modelling and Analysis of Dynamic Systems' [70] based on this Appendix. The interface is expected to be used for numerous courses.

4.3 ROBOT OPERATING SYSTEM (ROS)

The previous interface made use of the Robot Operating System (ROS), an open-source middleware. It was utilized to connect equipment such as the quadrotor with the laptop through Wi-Fi (Figure 5).

The first efforts for the implementation of the interface of this dissertation were dedicated to try and run the C++ scripts upon which the ROS middleware was working on. Nevertheless, this proved to be infructuous. The second and final attempt was intended to facilitate the use of ROS by minimizing the number of C++ scripts that were executed while replacing them with Matlab scripts and Simulink interfaces. Both attempts will be explained.

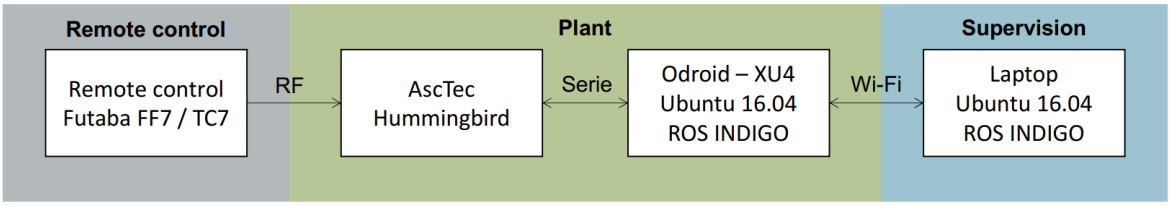

Figure 5: Work environment diagram

Adapted from source: [8]

As general information, the middleware offers all the features an operating system (OS) should have, such as hardware abstraction, low-level device control, common functionality implementation, message-passing between processes, and package management. Additionally, it offers resources and libraries for finding, creating, writing, and executing code across numerous machines [71].

The main component of its architecture is the ROS Master node, which offers naming and registration services to other nodes within the ROS framework (Figure 6). It maintains a record of publishers, subscribers, and services associated with topics. The purpose of the Master is to facilitate inter-node discovery, allowing individual ROS nodes to locate one another. Once the nodes have identified each other, they engage in peer-to-peer communication [72]. The middleware provides a mechanism for initiating the master and multiple nodes simultaneously, utilizing a launch file. This launch file is an XML document. The utilization of these type of files is prevalent in numerous ROS packages. Any system

that employs more than a couple of nodes is prone to employ launch files to define and set up the nodes that are required [73].

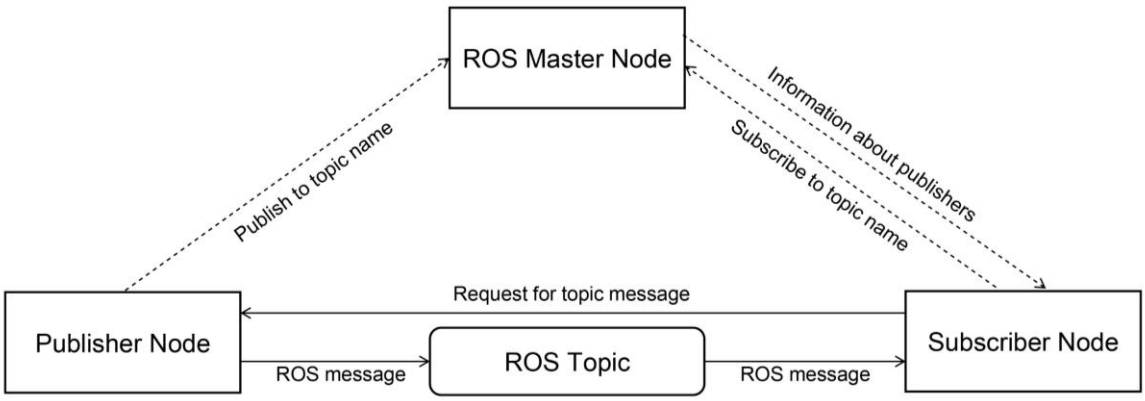

Figure 6: ROS Master node

Adapted from source: [74]

4.3.1 FIRST ATTEMPT

After the decision was made regarding the software to be used for the interface, the first intention was to execute the old manuals' scripts through the Matlab command window. For this purpose, MathWorks' functionalities were analysed, and an attempt was made to develop a Matlab script. After some research, it was found that the software had a suitable command called 'system' that enables the execution of Operating System commands (in this case Linux) and returns its output in the command window, similarly to the execution of a script on a Linux terminal [75]. Prior to proceeding, certain aspects of the script will be explained to understand why the endeavour was unsuccessful.

After connecting the battery and powering the quadrotor up, the ROS Master Node had to be ran inside it. To do that, a secure shell (SSH) connection through a Wi-Fi channel on a Linux terminal was established, as explained thoroughly in Appendix C, section IV. The prompt inside Matlab's command window would look like this:

```
command = 'ssh odroid@10.0.0.200'
system(command)
```

After the connection was established, the ROS master node could be executed as follows:

```
roslaunch asctec_hl_interface fcu.launch
```

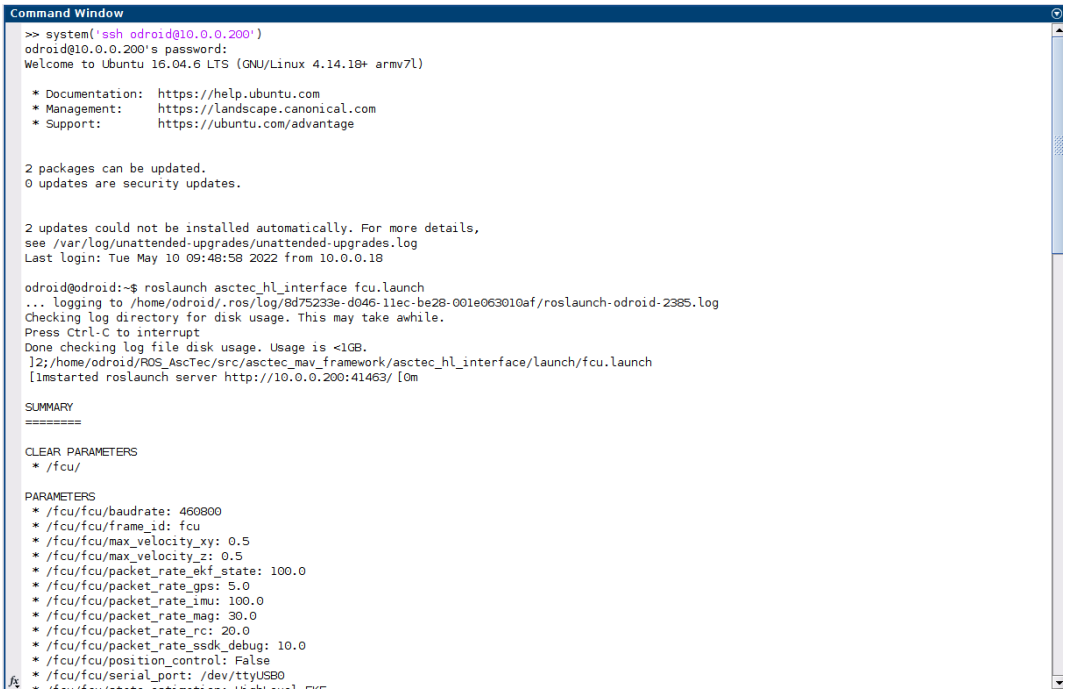

Figure 7: Execution of ROS Master in Matlab

After that, the command window began to reproduce the command's outputs, that is, the Master Node's status. This contained relevant information for the user, such as the IMU's status or the topics on the network. Because of this reproduction, Matlab's command window was taken over by the command itself and did not allow further use of the window. No further scripts were allowed to be executed. Basically, this command turns the command window of Matlab into a Linux terminal (Figure 7). This is because, as explained, the 'system' command returns an output for a given input. The experience then proved that the command window was not a flexible interface to work with, as it could only execute one script at a time. The execution of the ROS master node for the quadrotor was therefore not done through Matlab. It was then done through an Ubuntu terminal. An advantage of this is that the node's status (including the battery voltage) may be consulted on necessity in the above-mentioned terminal.

4.3.2 SECOND ATTEMPT

Other functionalities of the MathWorks' framework were explored to build the interface around it. The dedicated ROS toolbox [76] was found. Note that it is still in development and software bugs are being addressed constantly by the company. However, it provided the necessary tools to build an interface after several problems were resolved. A very helpful command that the toolbox did have was 'rosinit', which allowed the connection of the

MathWorks' framework with the ROS Master executed on a given IP address [77], which was running on an Ubuntu terminal, as explained earlier. The command execution could look like this:

```
ipaddress = '10.0.0.200'; % drone's IP address to which the laptop
connects
rosinit(ipaddress)
```

The result of this is further explained on the corresponding code found on Appendix D, item I. The interaction of the toolbox did, indeed, set back the start of the experimentation phase because the pre-existing scripts were not executable in Matlab, and because the toolbox had major compatibility problems, as it will be explained in the following subsection.

4.3.2.1 Message registration

According to the quadrotor's control manual, a script called 'ctrl_test.cpp' was needed to provide the UAV with the necessary inputs. It can be found on Appendix D, item III. The script was then analysed with the intention of reproducing its contents inside Matlab and Simulink.

However, a new problematic arose: the ROS messages. The middleware is built to work with compound data structures. This makes it more efficient in terms of data transfer, as the structure can have all the necessary information for a particular application. Some basic messages are built into Matlab, such as the 'geometry_msgs/Quaternion' message. For example, to create a new empty message of the type, the following code line can be executed on the command window:

```
msg = rosmessage("geometry_msgs/Quaternion","DataFormat","struct")
```

Which would output the following:

```
>> msg = rosmessage("geometry_msgs/Quaternion","DataFormat","struct")

msg =

  struct with fields:

    x: 0
    y: 0
    z: 0
    w: 0
```

Figure 8: Creation of an example ROS message inside Matlab

The ROS middleware also allows the creation of custom messages, which are used to develop libraries. That was the case for the library ‘asctec_mav_framework’, coded by ETH Zurich [24], [78], and the resulting custom messages were not part of the ROS toolbox. However, Matlab did offer a solution for the problem. The toolbox had a command that enabled the registration of custom ROS messages. It was called ‘rosgenmsg’, for which a guide was available [79]. Additionally, the toolbox had inflexible requirements [80] to that end (Figure 9). The command could only be used on Matlab R2020b release or above, and the software Python 2.7 and CMake 3.15.5+ releases were expressively needed to register the ROS custom messages according to the step-by-step guide.

MATLAB R2020b to R2021b

From R2020b to R2021b, ROS Toolbox supports the Melodic Morenia distribution for ROS.

ROS System Requirements

Platform	Platform Versions	Python Version	C++ Compilers	CMake
Windows	Windows 10	2.7	Visual Studio 2017 and 2019 (recommended)	CMake 3.15.5+
Linux	Ubuntu 18.04 Bionic Beaver (recommended)		GNU Compiler Collection (GCC) 6.3+	
macOS	Mac OS X		Xcode 10+	

Figure 9: ROS toolbox requirements

Source: [80]

These requirements presented several compatibility issues with the laboratory’s computer. First, the Matlab release installed at the time was R2018a. An update was then necessary. Secondly, it was imperative that the laptop’s CMake release stayed as it was before (release 3.10.0). This was because the quadrotor needed this particular release to compile its workspace correctly and, subsequently, execute its scripts properly. A workaround was then required to force Matlab to work on CMake 3.10.0 release instead of the 3.15.5+. The solution was documented on a step-by-step guide on Appendix C, item IX.

To summarize, Matlab’s internal scripts had to be accessed and altered manually to force the software to make use of a lower release of CMake for the compilation process involved while executing the registration of messages. Note that, although it is not against MathWorks’ copyright to make the changes on internal scripts, they oppose to the idea of users altering or editing them [81], [82]. The toolbox’s Python (2.7) release requirement had

no compatibility issues with the computer since they both used the same release. Note that, even though Matlab releases R2020b through R2021b could be used for the same purpose, the Ubuntu OS used in the laboratory (release 16.04) worked better with older versions of the software.

After forcing the system's compatibility, the MathWorks' guide for the registration of custom messages [79] could be followed without further problems. The newly registered messages were then available to be employed inside the MathWorks' framework. This allowed the creation of the 'mav_ctrl' message (Figure 10) used in the quadrotor's code (Appendix D, item III).

```
>> msg = rosmessage("asctec_hl_comm/mav_ctrl")

msg =

  ROS mav_ctrl message with properties:

                    MessageType: 'asctec_hl_comm/mav_ctrl'
              DirectMotorControl: 6
    DirectIndividualMotorControl: 7
                    Acceleration: 1
                        Velocity: 2
                        Position: 3
                    VelocityBody: 4
                    PositionBody: 5
                          Header: [1×1 Header]
                            Type: 0
                               X: 0
                               Y: 0
                               Z: 0
                             Yaw: 0
                           VMaxXy: 0
                            VMaxZ: 0
```

Figure 10: 'mav_ctrl' message after registration

To turn on the motors, the ROS custom service called 'fcu/motor_control' was used. It was registered alongside the custom messages, as previously explained. As explained, in ROS, the services make use of a client-server type of architecture for communication (Figure 6). The client (Matlab) creates a request and sends it to the server (ROS master node running in the quadrotor). In this case, the client demands the server to turn on its blades. Then, the server produces a response, either positive or negative. The code is printed on the command window. This can be interpreted in the following Matlab code shown in Appendix D, item I:

```
client = rossvcclient("/fcu/motor_control","DataFormat","struct");
request = rosmessage(client);
request.startMotors = true
response = call(client,request,"Timeout",10);
if response.motorsRunning == true
       disp("The motors are ON")
```

```
elseif response.motorsRunning == false
       disp("The motors were turned OFF")
else
       disp("The motors have not been able to be turned ON")
 end
```

In the next section, a Simulink interface will be developed to deal with the corresponding signals of the system. Although, it could be possible to control the motors' status (ON, OFF) through the interface, it is safer and easier through the code explained above. For that reason, the model does not possess this functionality.

4.4 SIMULINK INTERFACE FOR EXPERIMENTS

After the registration, a model could be developed in Simulink (Figure 11). The interface consists of three segments to handle the input and output signals of the quadrotor-gyroscope system.

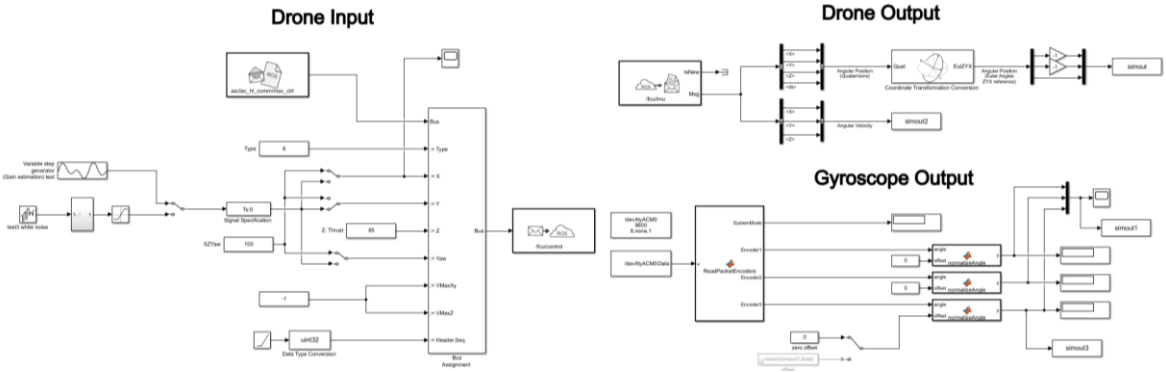

Figure 11: Experiment interface in Simulink. Model view

Firstly, the Drone Input segment deals with the test signals. Figure 12 offers a closer look. This script made use of the 'mav_ctrl' message, which is published on the 'fcu/control' topic in the ROS network. It is then picked up by the quadrotor, which turns it into four voltages applied to each of its motors. To replicate that same workflow, an empty message is created by the Blank Message block and filled with the desired input values using the Bus Assignment block. Moving forward, publishing on the commented topic is done through the Publish block.

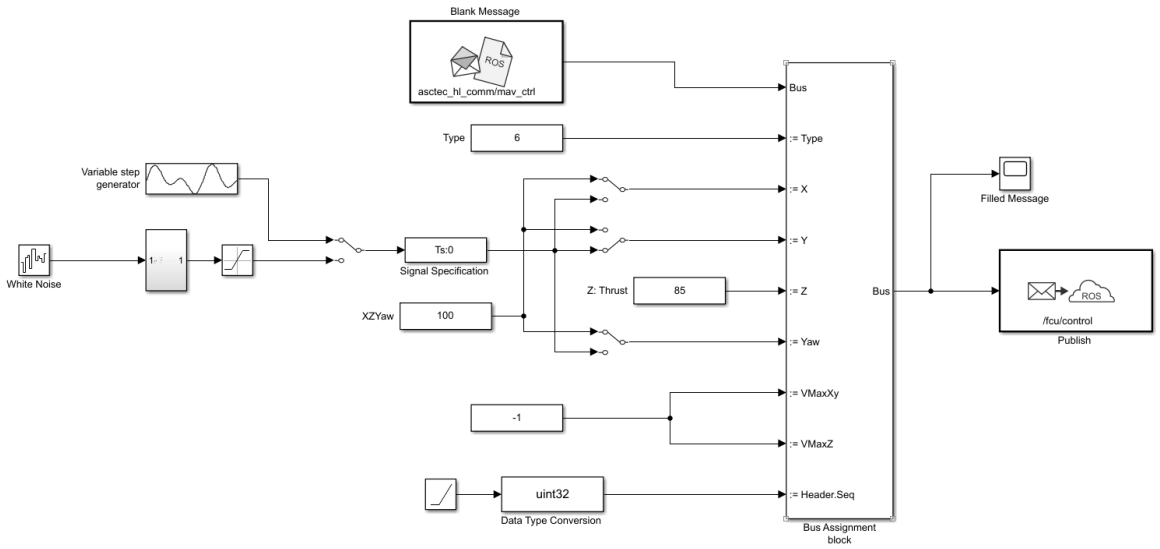

Figure 12: Drone Input segment

Specifically, the empty message is mainly filled with the following information. Keep in mind that the data types for all the variables is 'single' in the MathWorks' framework.

- **Type**: it represents the type of command the user is sending to the drone. Number 6 is intended for the 'DirectMotorControl' sequence, namely the open loop function.
- **X input (%)**: it represents the **pitch** angle, but it is, in fact, the action of the motors to produce such a movement. Its range goes from 0 (-100%) to 200 (100%). Therefore, it is unitless.
- **Y input (%)**: it represents the **roll** angle, but it is in fact the action of the motors to produce such a movement. Its range goes from 0 (-100%) to 200 (100%). Therefore, it is unitless.
- **Z input (%)**: it represents the **thrust**, which is the simultaneous action on the four motors of the quadrotor to produce an altitude change. Its range goes from 0 (0%) to 100 (100%). Therefore, it is unitless. It is currently set to 85 (42,5%), which is the approximate thrust needed to make the drone hover in the air in an idle mode.
- **Yaw input (%)**: it represents the **yaw** angle, but it is in fact, the action of the motors to produce such a movement. Its range goes from 0 (-100%) to 200 (100%). Therefore, it is unitless.

The system can therefore be divided into subsystems. For example the first subsystem would be the one made of X input with the pitch's response. It is imperative to understand that the quadrotor does not have a control algorithm on the message is type 6, as will be

explained on chapter 6. That means that it is working in open loop, rather than in closed loop. Its input is independent from its output for any of the variables explained before. For example, given a pitch (X) input signal, the pitch gimbal tends to change, but so do the other angles, although in a much smaller scale, and therefore, negligible in a practical sense when considering the subsystems separately. This is because the manufacturer built in an algorithm in the low-level processor (LLP). It combined the input variables to actuate indirectly on the motors, turning the drone from a coupled MIMO to a decoupled MIMO that resembles four SISO. The other modes were not necessary for the dissertation and will not be explained.

In Figure 12 there is a manual switch at the entrance of each input variable. The Waveform Generator produces a complex input signal (Figure 13). It was used to generate a gradual step signal. The second block on the far-left side, called White Noise, was utilized to generate a random step signal based on the White Gaussian Noise block seen in Figure 14. Their shape will be further discussed in section 6. Both must be transformed from a continuous to a discrete signal with the Signal Specification block.

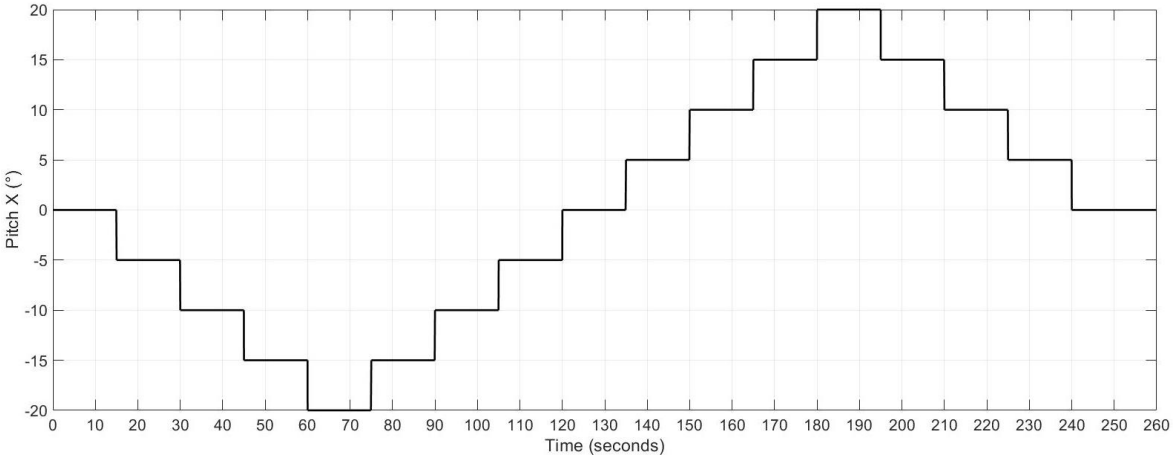

Figure 13: Waveform generator block

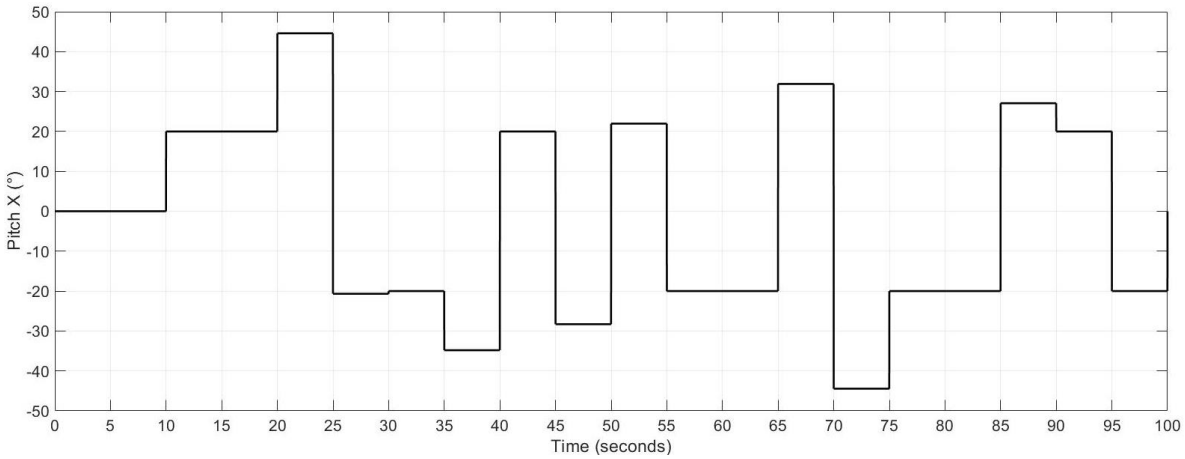

Figure 14: White Noise generator block

The second segment of the model is the Drone Output (Figure 15). The drone publishes the IMU sensors' data in the '/fcu/imu' topic. There are two important measurements that should be pointed out from it. The quadrotor publishes its angular velocity in radians per seconds (rad/s). Additionally, its angular position is also published and can be retrieved as quaternions in a ZYX reference. This is then transformed into Euler angles through an additional block, called Coordinate Transformation Conversion (CTC) block.

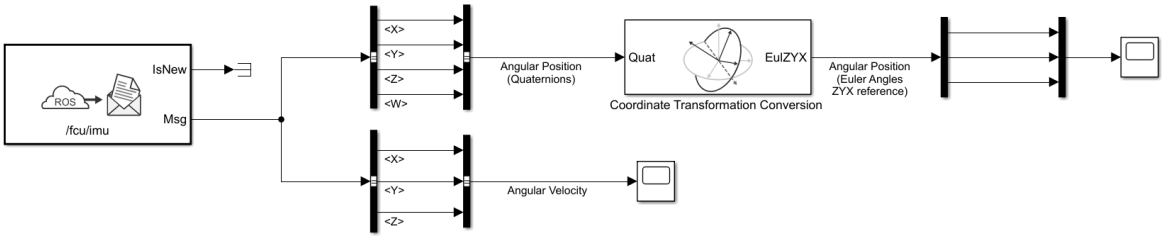

Figure 15: Drone Output segment

The last segment of the model is called Gyroscope Output (Figure 16). Eureka Dynamics, the gyroscope's manufacturer, has provided the means to send and receive signals through Matlab scripts and Simulink interfaces [67]. The default configuration found on its source was not ideal to work in all computers. The segment was therefore adjusted and reconfigured. It can also work as a standalone model if need be. In the next chapter, it will be used to estimate physical parameters such as friction coefficients and inertia.

Particularly this segment captures the output signals from each encoder at their respective rings on the test bench through the Serial Receive and Serial receive blocks. This blocks are explained in the following paragraphs. Moving on, the information of the

encoders is passed through a function called ‘ReadPacketEncoders’, which is responsible for the processing of the signal. It outputs the readings coming from the encoders in an organized manner. Taking into account that the encoders are on a random position at the start of each experiment (e.g., random start in Figure 19), their offset can be subtracted using the ‘normalizeAngle’ function seen in Figure 16. The measurement units are expressed in degrees (°). The datasets gathered here will be converted to radians later on.

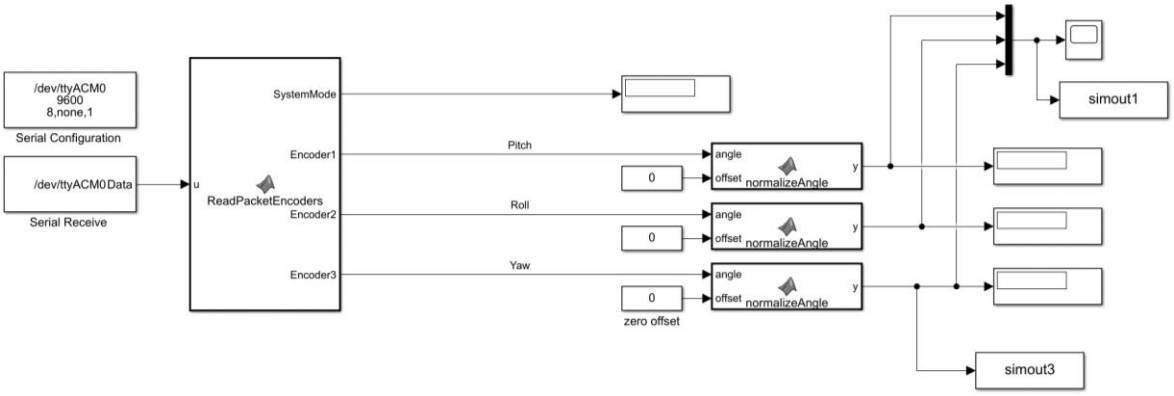

Figure 16: Gyroscope Output segment

Adapted for source: [67]

Moving backwards, the first block on the left is the Serial Receive block. This block has several functionalities which are crucial for the model to work properly (Figure 18). First, it establishes the communications with the bench’s board through a serial connection (USB). Secondly, it maintains a fixed sample time for the model. This functionality is the most important one, because it provides real-time simulation inside the model in the interface. It also configures some of the parameters in the serial connection with the computer.

For experimentation, the sample time for the datasets needs to be at least 2 to 5 times faster than the system dynamics it tries to describe. Ideally, it should be 10 times faster [54]. This was also part of the system identification process because the system’s dynamics were unknown at the start of the process. According to [10], if data collection is cheap and the system’s dynamics are unknown, then it is better to sample faster than slower. Figure 17 displays the consequences of a sizeable sampling rate (in this case $T_s = 0{,}4\ s$), where the continuous signal is the real system, and the dots and discrete signal are the sampled data and interpolated signal, respectively. In chapter 6, it was discovered that the time base was so fast that the resulting angular velocity vector was too noisy for its use in the system

identification [62]. This could be avoided if the signal were filtered, which would add a delay to the sample, which should be taken into account during the identification.

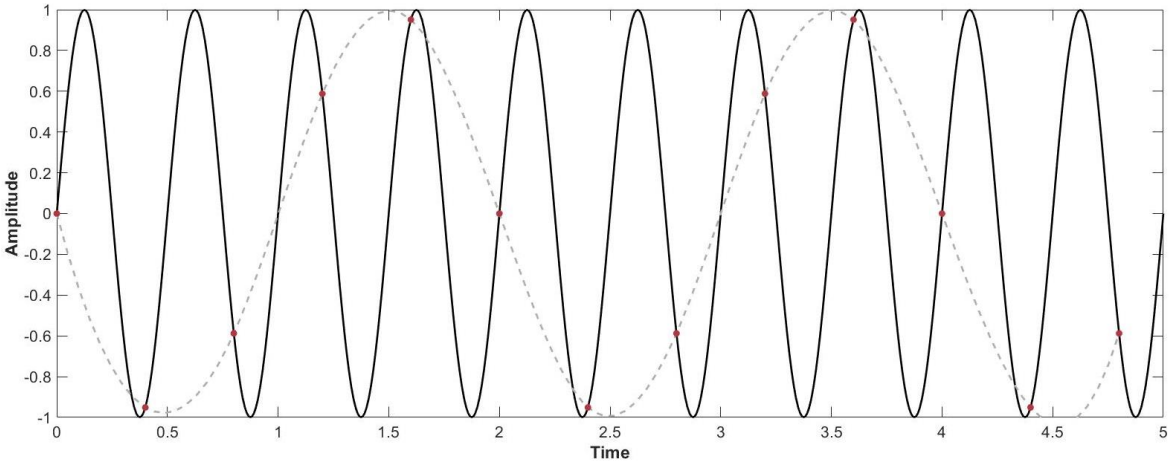

Figure 17: Sizeable sampling time

Source: [55]

The second block on the left is the Serial Configuration block and allows the configuration of additional channel parameters (Figure 18). It also needed the configuration of the specific port to which the test bench was connected to function properly.

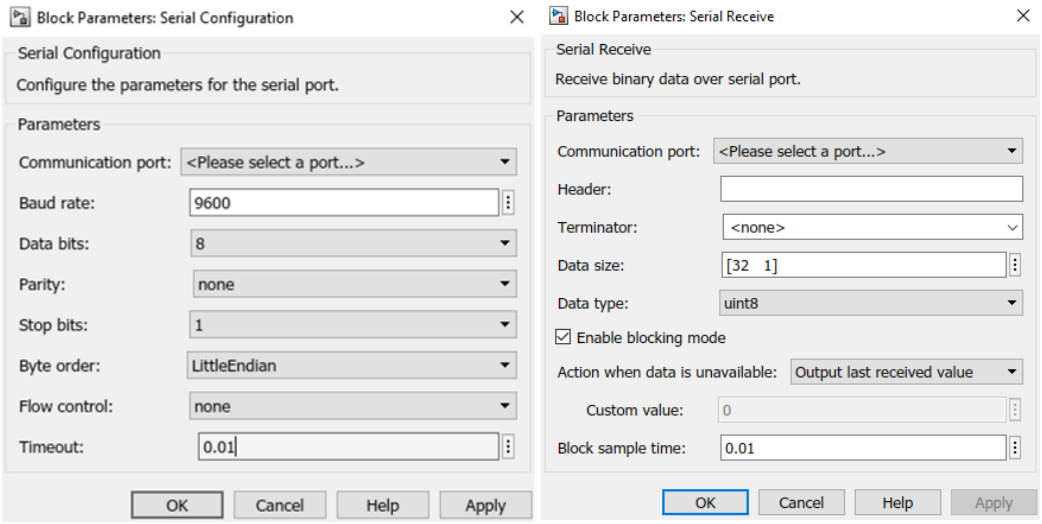

Figure 18: Serial Configuration and Serial Receive blocks' parameters

Both blocks caused a massive problem, which lasted months. With no apparent reason, there were sudden jumps or direction changes in the encoders' readings, as it can be seen in Figure 19. Eureka Dynamics, the test bench manufacturer, was contacted to resolve the issue. After four months, the cause was detected. The solution consisted of reducing the

'Timeout' parameter inside the Serial Configuration block. It needed to be reduced from 10 to 0,01 seconds (Figure 18) to match the sample time of the model. This lowered the data quality, as it will be explained later. The resulted signal after the solution can be observed in Figure 22.

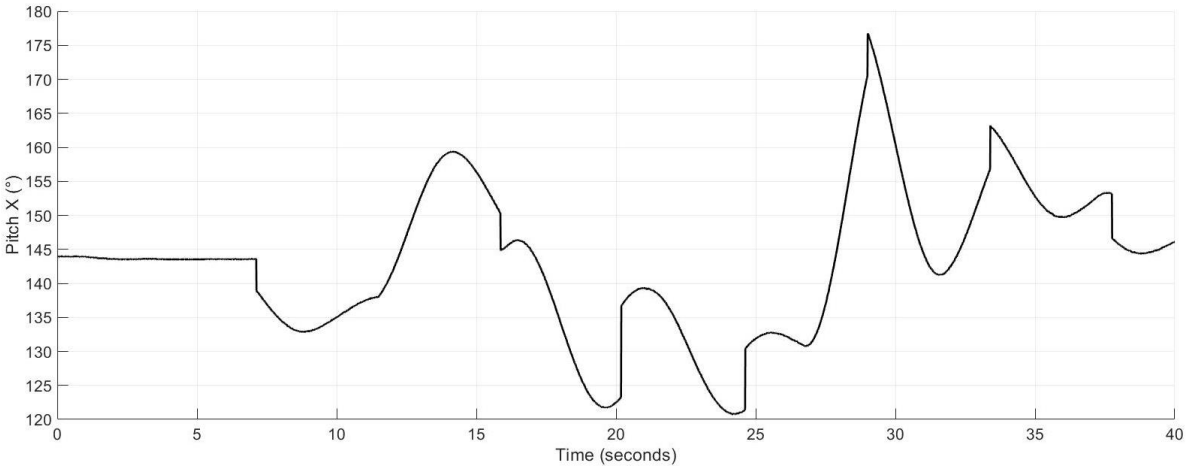

Figure 19: Encoder jumps

It is important to point out that the system identification process experienced difficulties because the gyroscope's output was unreliable for a long period of the dissertation's timespan. Note that the angular position data collected from the encoders has a much lower noise level than the data from the quadrotor's IMU sensors and was therefore the preferred option to work as input in the identification process. The consequences of this fact will be further explained during the system identification (chapter 6).

Lastly, the system mode seen in Figure 16 displays whether the system is working with encoders (mode equal to 2) or with motors (mode equal to 1). Even though the test bench has interchangeable motors, they were not used in this dissertation because the actuators of the system are the quadrotor's motors. Nevertheless, the gyroscope's motors could be used in further research to simulate disturbances (i.e., wind) and see how they affect the system. This will be further commented on chapter 8. Note that the motors cannot work with encoders at the same time.

5. PARAMETER ESTIMATION

This chapter is dedicated to the explanation of the gimbals' motion through an equation. It uses it for the estimation of parameters such as inertia and friction coefficients for the gimbals. It helps to understand the quadrotor-gyroscope system before the identification process starts.

This chapter uses the interface developed in the previous chapter. The platform's manufacturer was contacted. Basic physical parameters (i.e. inertia, mass) and 3D models (Figure 20) were delivered by Eureka Dynamics.

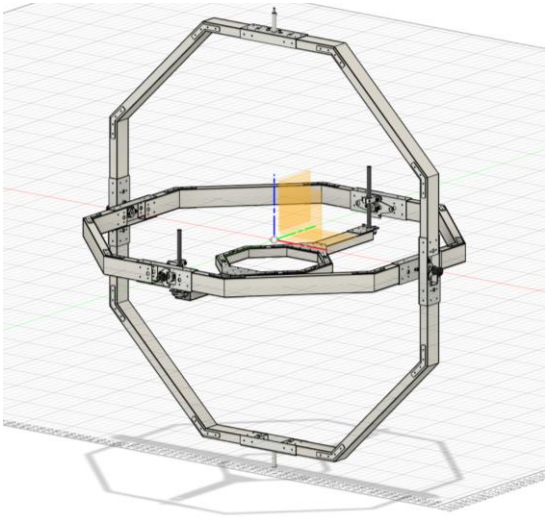

Figure 20: 3D model of the platform

5.1 SYSTEM DATA COLLECTION

Before any estimation could be done, some simple tests needed to be conducted to get a glimpse of what physical elements had an effect on the system when certain initial conditions were imposed on the equipment and infer the nature of the motion through mathematical equations.

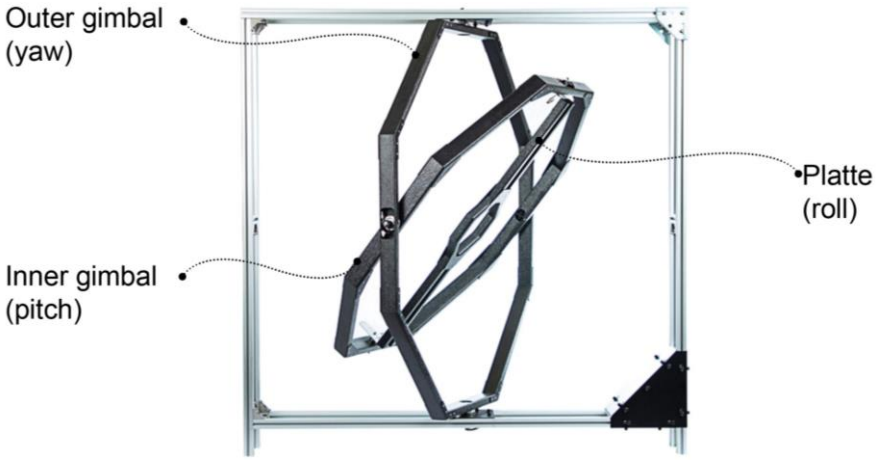

Figure 21: FFTGyro at UPC ESEIAAT

Source: Manufacturer's website

At first, the outer gimbal (yaw) and the plate (roll) were virtually fixed in their movement and only the inner gimbal (pitch) was moved away from its horizontal equilibrium point (Figure 21). Therefore this last axis was allowed to move freely with an initial angular velocity equal to zero. Figure 22 is a plot of one of the datasets gathered from this experiment. The gathered data has minimal noise and is not visible in this scale. Additionally, and since the gimbals' resting position do not coincide with the encoders' zero angle reading, the data was displaced on the pitch axis and a compensation was necessary.

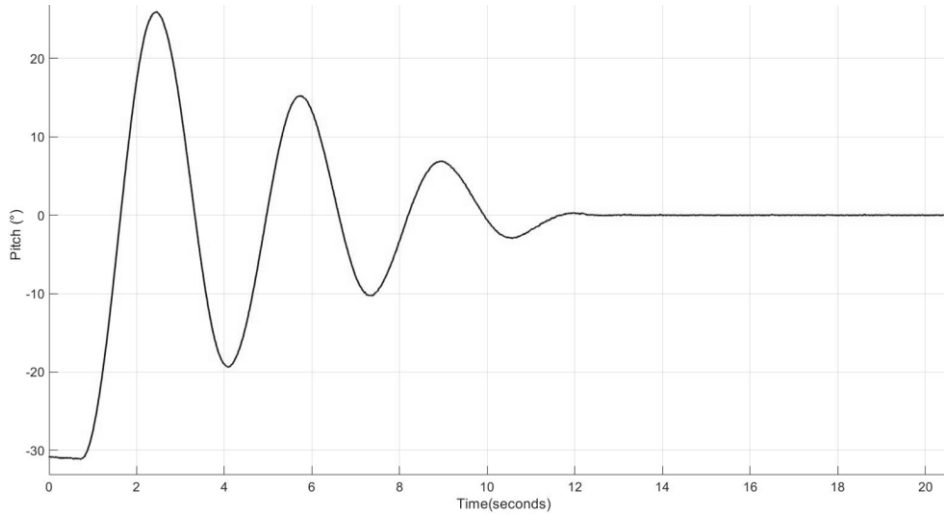

Figure 22: Plot of the inner gimbal motion

It is clear that the gimbal has an angular velocity equal to zero for the first part of the test. This is because the system is artificially stopped and then released. There are several problems with this data as it is shown. One of them is knowing the exact moment when the

ring was released, and the motion started. For that purpose, a simple data treatment was necessary. The first semi cycle going from negative to positive angles is discarded. The cusp of the first oscillation will be the starting point. This point will be regarded as $t = 0$, as seen in Figure 23. This is because the velocity is zero by definition and only gravity and friction will influence the system at this point forward. In the following section, the use of the experimental data as shown will be further explained.

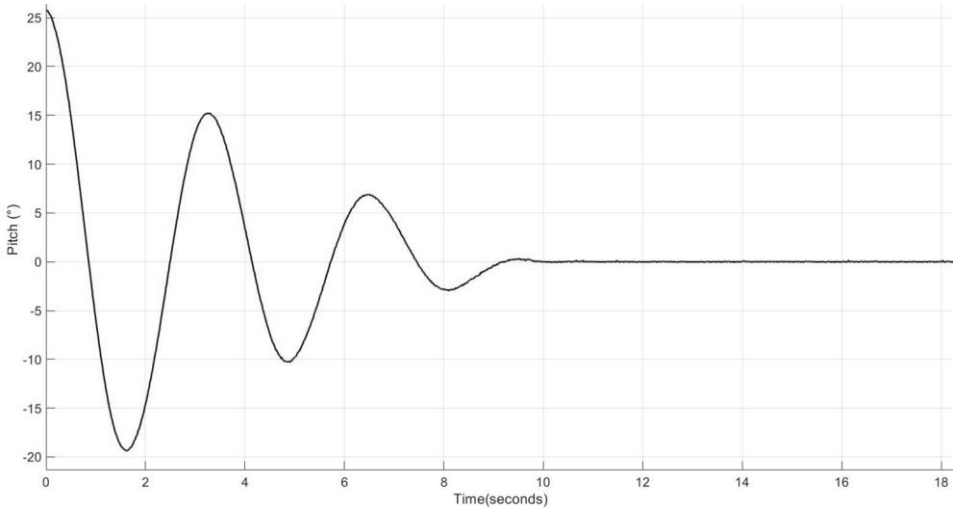

Figure 23: Data treatment on one of the gathered datasets

5.2 SIMULATION AND MATCHING

The end goal there is to estimate parameters of the machine with the drone strapped to it. One of the most relevant physical properties that are missing from the system is the friction coefficient. For that purpose, Newton's 2nd law for rotational motion was used. By analysing the system, a simple yet nonlinear system was found:

$$J\ddot{\theta} = \vec{F} \times \vec{r} - B\dot{\theta}$$

Equation 1: Gimbals' rotational motion

Where J = moment of inertia, kg*m² $\dot{\theta}$ = angular velocity, rad/s

$\ddot{\theta}$ = angular acceleration, rad/s² B = rolling friction coefficient, kg*m²/s

$\vec{F} \times \vec{r}$ = torque due to gravity, N*m $B\dot{\theta}$ = torque due to friction, N*m

$\vec{F}$ = force vector due to mass displacement, Newton

$\vec{r}$ = distance between mass displacement and the system's centre, meter

Here, the torque due to friction was modelled as linear. If treated as nonlinear, the component could have a very different shape depending on the physical system at hand. Note that only linear systems and components are within the scope of the work. Additionally, $\vec{F} \times \vec{r}$ is the torque resulting of the displaced centre of mass for any given angle. Since the system's mass and its distance r cannot be measured, it will be presented as a single constant F_r. This is because they are both constants and cannot be separated. Moreover, little could be gain from knowing the exact values of both separately.

Moving forward, the equation represents the sum of torques present in the system around a given axis, which is equal to the moment of inertia J multiplied by the axis' angular acceleration as follows:

$$J\ddot{\theta} = -F_r \sin\theta - B\dot{\theta}$$

Equation 2: Simplified equation

Where $-F_r$ = simplified term for $\vec{F} \times \vec{r}$ in a given direction, N*m

The simplified Equation 2 represents the motion of both the pitching and rolling. The terms are negative because they oppose the motion. For the yawing, the torque $-F_r$ disappears because gravity does not act directly on this ring. The only torque acting on it is due to the friction at the joint of the spinning axis.

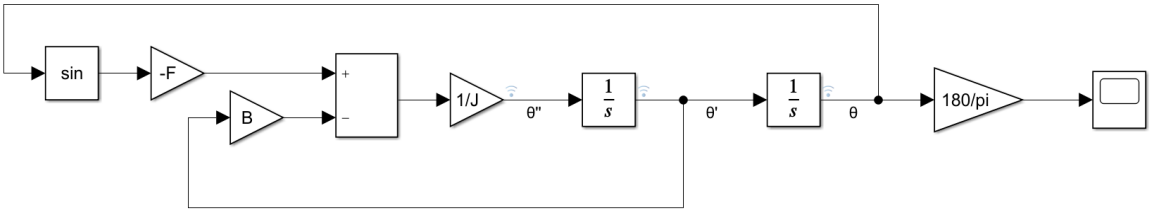

Figure 24: Simulated system inside Simulink interface

Where $\frac{1}{s}$ = integrator in Laplace domain θ = angular position, rad

$\dot{\theta}$ = angular velocity, rad/s $\ddot{\theta}$ = angular acceleration, rad/s²

As expected, the machine has an equilibrium point were the sum of torques is equal to zero. Ideally that would be the horizontal plane. This is not the case in the laboratory. With

and without the drone, the system was fairly unbalanced in the inner gimbal (pitch), as observed in Figure 25.

Figure 25: Inner gimbal stationary unbalance

The equation marked another odd effect in the machine. For example, the inner gimbal (pitch). If the centre of mass coincided perfectly with the axis, then it would mean that the first term on the right (F_r) would be equal to zero, which in turn would render the second term (friction) equal to zero because of its dependency on angular velocity, resulting in the gyroscope's balance. Extrapolating to every possible angle where that condition is met, the system would have an infinite number of equilibrium points. This was not the case in the laboratory and will be explained next.

The reason of the practical inexistence of infinite number of equilibrium points came from the commented imbalance, which acted as a torque due to gravity. The first torque is then only equal to zero when the machine is in the horizontal plane, aside from the mass displacement, which rendered resulting resting point not equal to zero. This was true for the pitch and roll, but not for the yaw, because the yaw was not affected by gravity.

Contemplating the system with or without drone, its resting point seems natural on the outside. Nevertheless, further analysis proved that the described motion was only possible when the centre of mass is below the axis in relative terms. In this case, if the roll axis were to be tilted upside down, then the torque F_r would point in the opposite direction, turning the ring completely around its axis in search for its equilibrium point.

After the analysis, tests were carried out to see if the equation adjusted itself fairly to the data set. The starting point of the tests was the following: the initial angle was measured, and the angular velocity was equal to zero ($\dot{\theta}_0 = 0$). Additionally, the platform's inertia was information given by the manufacturer. Adding the drone to the set could only increase the inertia of system, so the overall inertia of it when the drone is mounted would be higher than the rings' alone. In this case, the pitch was tested first, and the results can be seen as follows.

5.2.1 PITCH, ROLL AND YAW

Reproducing the previous equation inside a Simulink interface allowed for parameter estimation. For it, the Parameter Estimator App was used [83]. The toolbox allowed to set the starting point for the parameters' constant values for the estimation. Setting a strict limit to the possible parameter values was crucial to get reliable results in this process. The inertia was the only reference of a constant value that there was prior to the experiments.

To that end, and starting with the inner gimbal, its inertia (I_{zz} in this case) was extracted from the gyroscope's datasheet, as seen in Table 1. It was necessary to add the pitch axis' moment of inertia to the roll axis' moment of inertia (Table 2) around the z and x, respectively. The difference is that the pitch gimbal spined around its origin with respect to the z axis and the roll gimbal does it around its centre of mass with respect to the x axis. It is to be noted that the manufacturer only provided the characteristics of the gimbals separately, not as an assembled machine.

Item	Description
Dimensions – L x W x H	1016,55 mm x 54,80 mm x 1082,30 mm
Mass	4512,086 g
Volume	$7{,}408 \cdot 10^5$ mm³

World X, Y, Z	0,00 mm, 0,00 mm, 0,00 mm
Centre of mass	0,645 mm, 0,01 mm, 534,87 mm
Moment of Inertia at Centre of Mass (g mm²)	Ixx $5{,}096 \cdot 10^8$ Ixy 27,877 Ixz $1{,}069 \cdot 10^6$ Iyx 27,877 Iyy $1{,}053 \cdot 10^9$ Iyz 19,506 Izx $1{,}069 \cdot 10^6$ Izy 19,506 Izz $5{,}468\text{E} \cdot 10^8$
Moment of Inertia at Origin (g mm²)	Ixx $1{,}800 \cdot 10^9$ Ixy 0,001 Ixz $-4{,}865 \cdot 10^5$ Iyx 0,001 Iyy $2{,}344 \cdot 10^9$ Iyz -23113,911 Izx $-4{,}865 \cdot 10^5$ Izy -23113,911 Izz $5{,}468 \cdot 10^8$

Table 1: Inner gimbal (Pitch) specifications.

Source: Eureka Dynamics (internal communications)

Both axes were moving in the first experiment and need to be considered as a starting point for J as follows:

$$J = I_{zzpitch} + I_{xxroll} = 5{,}468 \cdot 10^8 gr \cdot mm^2 + 1{,}255 \cdot 10^8 gr \cdot mm^2 = 0{,}6723\ kg \cdot m^2$$

Where $J =$ system's inertia in the pitch gimbal, kg·m²

$I_{zzpitch} =$ pitch gimbal's inertia without the drone, gr·mm²

$I_{xxroll} =$ roll gimbal's inertia in the pitch rotating axis without the drone, gr·mm²

Item	Description
Dimensions – L x W x H	295,40 mm x 194,40 mm x 980,00 mm
Mass	1885,897 g
Volume	$3{,}645 \cdot 10^5$ mm³
World X, Y, Z	0,00 mm, 0,00 mm, 0,00 mm
Centre of mass	0,00 mm, 56,648 mm, 490,00 mm

Moment of Inertia at Centre of Mass (g mm²)	Ixx $1{,}255 \cdot 10^{8}$ Ixy $-7{,}062 \cdot 10^{-9}$ Ixz 10956,25 Iyx $-7{,}062 \cdot 10^{-9}$ Iyy $1{,}321 \cdot 10^{8}$ Iyz $3{,}752 \cdot 10^{-7}$ Izx 10956,25 Izy $3{,}752 \cdot 10^{-7}$ Izz $9{,}539 \cdot 10^{6}$
Moment of Inertia at Origin (g mm²)	Ixx $5{,}843 \cdot 10^{8}$ Ixy $-4{,}187 \cdot 10^{-7}$ Ixz 10956,25 Iyx $-4{,}187 \cdot 10^{-7}$ Iyy $5{,}849 \cdot 10^{8}$ Iyz $-5{,}235 \cdot 10^{7}$ Izx 10956,25 Izy $-5{,}235 \cdot 10^{7}$ Izz $1{,}559 \cdot 10^{7}$

Table 2: Plate (Roll) gimbal specifications.

Source: Eureka Dynamics (internal communications)

The other variables (F_r and B) are expected to land in a similar range with their respective units. The following constants were established as a starting point:

$$F_r = 1\ N \cdot m \qquad\qquad B = 0{,}1 \frac{kg \cdot m^2}{s}$$

The angle is in radians, as it is also the case inside the model seen in Figure 24. Just before the scope's entry, there is a change of units from radians to degrees (through the 180/π gain) necessary for the unit match of both the simulation and the dataset gathered from the encoders.

The experiment was set up inside the Parameter Estimation App [83] to minimize the error according to a Sum Square Error (SSE) algorithm built inside it.

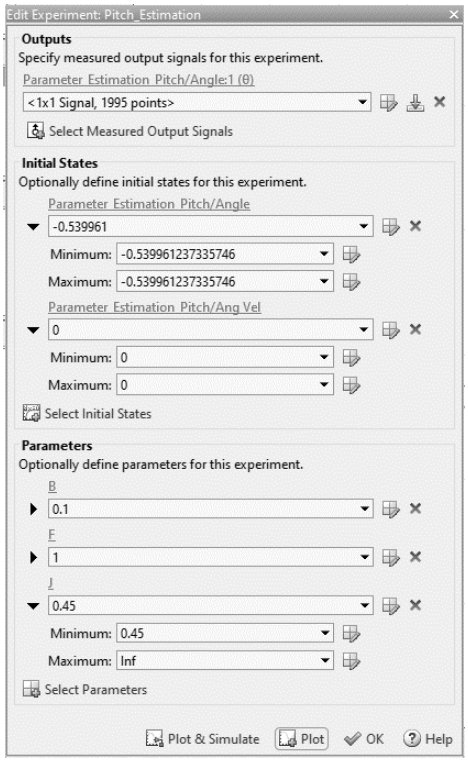

Figure 26: Experiment setup inside Parameter Estimator App

The software had the settings as shown in Figure 26. The output is the data gathered from the experiments, and the initial states are the initial pitch angle and initial velocity. Given that both initial states were known, they were fixed at the start of the estimation. Otherwise, the system could find a local minimum to the cost function which is minimizing. This may not be the real values of the constants, which is the reason why the data from Eureka Dynamics was so important. Then, initial constant values are established as a starting point, setting the gimbal inertia to a minimum value according to its datasheet (Table 1, I_{zz}). This value is only expected to increase due to the presence of the drone. Given the moment of inertia's equation for a solid as follows [33]:

$$I_{zz} = \int (x + y) dm$$

Equation 3: Moment of inertia. Formula.

Where dm = infinitesimal mass, kg

x = distance of the mass dm to the z axis, m

y = distance of the mass dm to the z axis, m

I_{zz} = moment of inertia of the gimbal around z axis, kg·m²

This increase in inertia due to the presence of the quadrotor's mass is expected be counterbalanced by its particles' small distance to the rotation axis z (Figure 27). The result of the initial states can be seen in Figure 28.

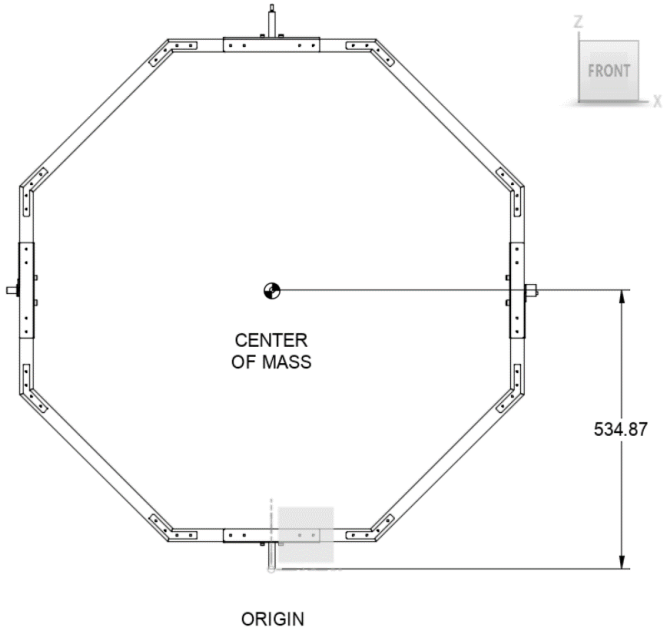

Figure 27: Pitch Gimbal and its axis' references

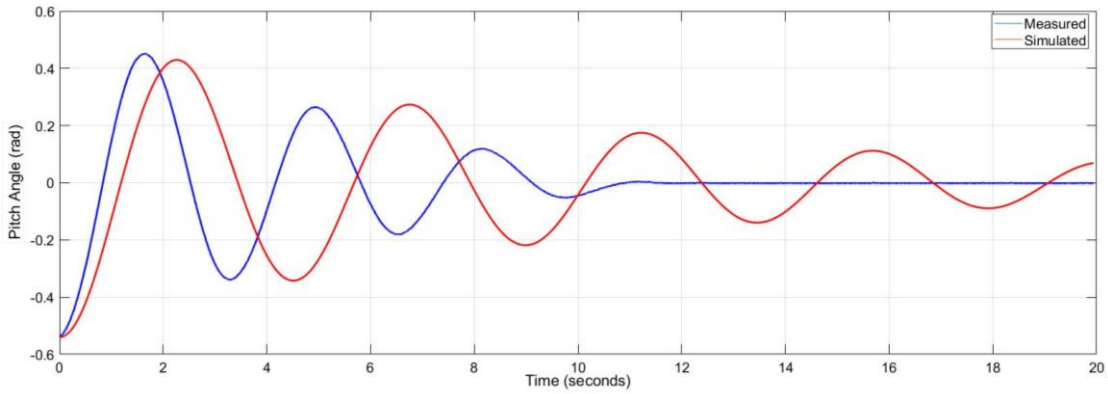

Figure 28: Parameter Estimation. Initial state.

The parameters' change over the iterations can be seen in Figure 29. It can be inferred from the first estimation (Figure 30) that the interface fairly reproduced the result of the experiment until a certain point. After that point, the estimated parameters present a divergence from the actual data. This could be caused by the nonlinear nature of the rolling friction at the axis, particularly at low speeds. Another dry friction model could be tested to

see whether the hypothesis is correct. [58] presents an example of one. Nevertheless, it is a far more complex dry friction model for B than it was needed.

This was not tested since it did not provide further insights for the proposed methodology. Note that the results of the system only provide a guidance to the design of identification and validation experiments of chapter 6.

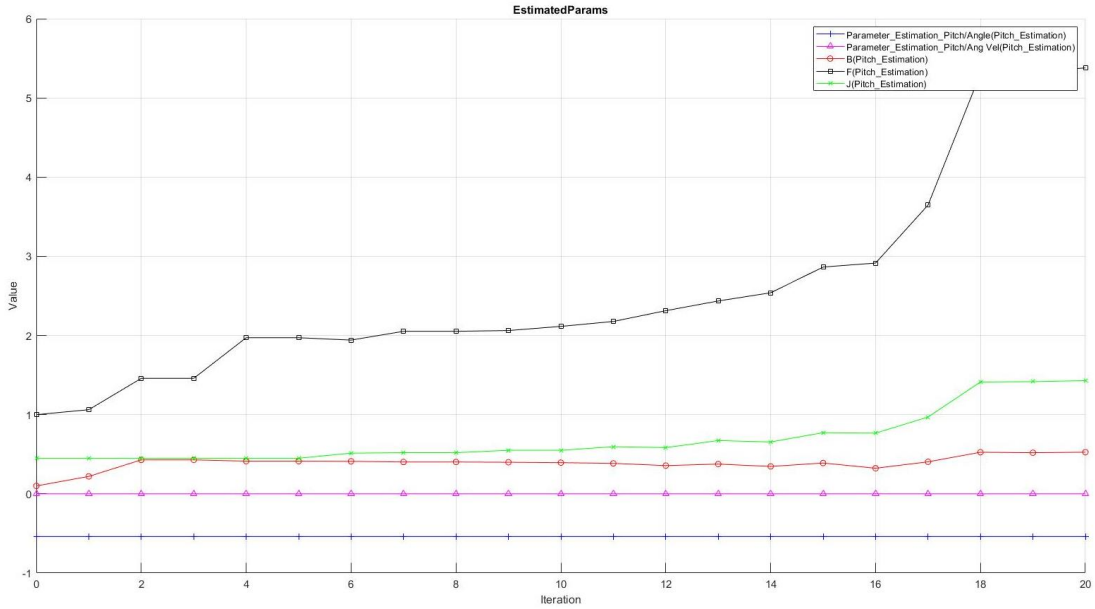

Figure 29: Parameter change

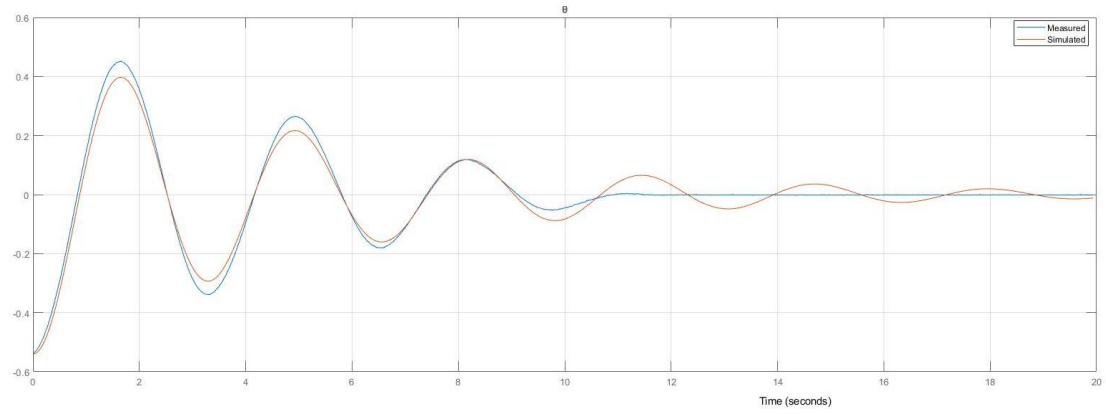

Figure 30: Estimation vs. dataset after optimization

Rounding up the numbers, the parameters were estimated to be:

$$F_r = 5{,}4\ N \cdot m \qquad B = 0{,}5\frac{kg \cdot m^2}{s} \qquad J = 1{,}4\ kg \cdot m^2$$

Table 2 was used again as a reference for the starting point. As it was expected that the plate would not add much inertia to the drone, after an analysis of the gyroscope's shape and its inertia through Equation 3. Taking the data from the corresponding table, the starting point for system in the roll direction was (I_{zzroll}) and the parameters according to the estimation:

$$I_{zzroll} = 9{,}539 \cdot 10^6 gr \cdot mm^2 = 0{,}009539\ kg \cdot m^2$$

$$F_r = 7\ N \cdot m \qquad B = 0{,}5 \frac{kg \cdot m^2}{s} \qquad J = 0{,}8\ kg \cdot m^2$$

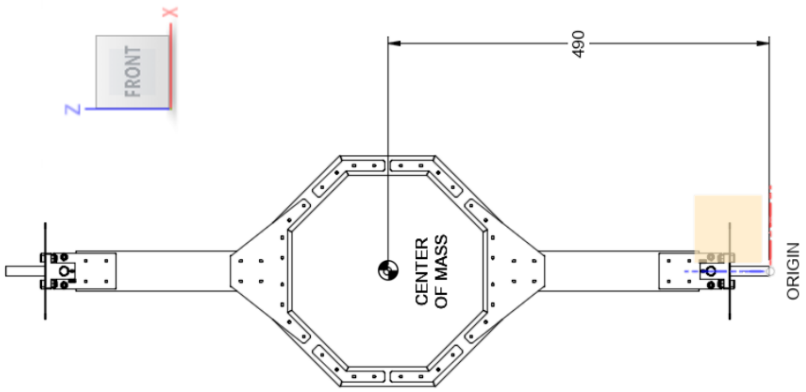

Figure 31: 3D model of the plate (roll)

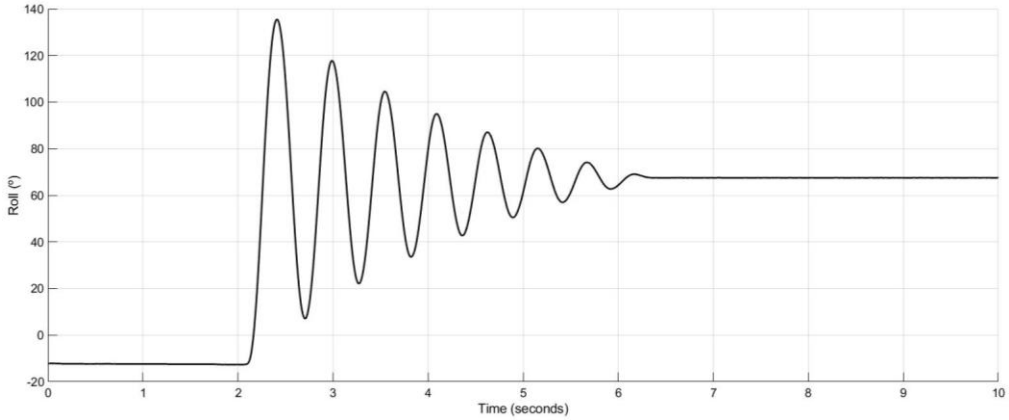

Figure 32: Raw data for roll angle

Since the yaw's motion was the most affected by the gyroscope, the estimation will be based only on Table 3. The gravity component F_r known to be equal to null, and the friction coefficient had already been established to be same for the pitch and roll, the inertia of the yaw is taken out of the manufacturer's datasheet as follows:

$$J = I_{zz,yaw,origin} = 7.630 \cdot 10^8 gr \cdot mm^2 = 0{,}763\ kg \cdot m^2$$

Item	Description	
Dimensions – L x W x H	1105.35 mm x 54.80 mm x 1250.70 mm.	
Mass	5100.588 g.	
Volume	8.158E+05 mm^3.	
World X, Y, Z	0.00 mm, 0.00 mm, 0.00 mm.	
Center of Mass	0.621 mm, 0.008 mm, 618.654 mm.	
Moment of Inertia at Center of Mass (g mm^2)	Ixx	7.961E+08
	Ixy	26.451
	Ixz	15869.044
	Iyx	26.451
	Iyy	1.556E+09
	Iyz	2.022
	Izx	15869.044
	Izy	2.022
	Izz	7.630E+08
Moment of Inertia at Origin (g mm^2)	Ixx	2.748E+09
	Ixy	-0.001
	Ixz	-1.944E+06
	Iyx	-0.001
	Iyy	3.508E+09
	Iyz	-26346.711
	Izx	-1.944E+06
	Izy	-26346.711
	Izz	7.630E+08

Table 3: Outer gimbal's (yaw) specifications

6. SYSTEM IDENTIFICATION

This chapter is dedicated to the experimentation process, the (re)design of experiments to accommodate them to the necessities of each angle, and the first two phases of the system identification process. The process was divided into three parts: identification, initial validation, and final validation phases. The first two were done through MathWorks' System Identification Toolbox (SITB). The desired outcome of the process was to get a good mathematical parametric model for every Euler angle in the system.

The chapter also gathers not only information of good practices when handling the system, but also practices that should be avoided if good data (i.e. informative enough) is to be gathered [54]. It was not within the scope of this dissertation to study statistical parameters in data (i.e. variance). The final validation is articulated in chapter 7, where the models enabled the tuning of flight controllers.

The chapter will be developed in a chronological order to emphasize the conclusions drawn from the most relevant iterations on the system identification process. The term 'system' will be referred to the quadrotor mounted on the test bench, i.e. drone-gyroscope pair. The terms 'position' and 'velocity' are referred to angular position and angular velocity, respectively, since there was no translation of the quadrotor, only rotation along its axes.

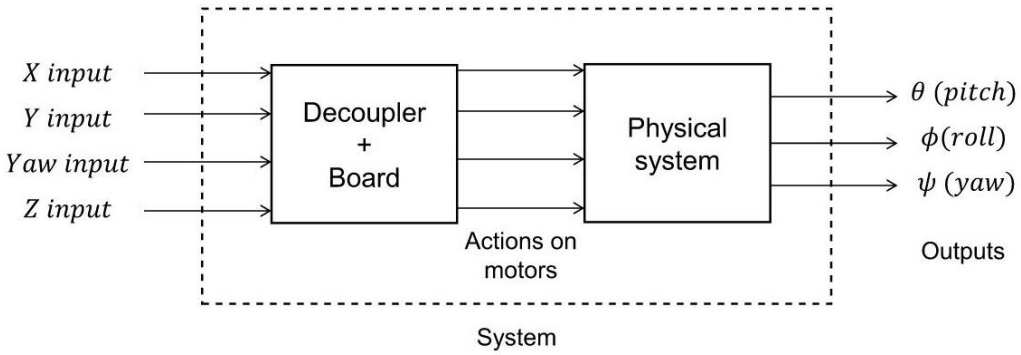

Figure 33: System representation

Prior knowledge (quadrotor's manual) asserted that the system had a built-in decoupler that allowed the actuation on the outputs (Euler angles) through discerned inputs. The system can therefore be subdivided into subsystems. Physically, that means that the user does not control the voltages applied to the system, rather they are calculated by the board and applied correspondingly based on the inputs. A representation of this can be observed

in Figure 33. However, without it the user would have to indicate what voltage to apply to what motor for any given moment. Therefore, the identification process would be multivariable and far too complex for the dissertation's duration. In short terms, this fact turns the structure from a highly coupled multiple-input multiple-output (MIMO) to a decoupled one, equivalent to four single-input single-output (SISO) systems. To clarify the subject and avoid confusions, note that the subsystems are the following:

a) 'X' input affects the inner gimbal (pitch θ),

b) 'Y' input affects the plate (roll ϕ),

c) 'Yaw' input affects the outer gimbal (yaw ψ, Figure 21), and

d) 'Z' input produces thrust (elevation).

The input variables are expressed in percentage and do not have a strict physical interpretation, although they can be understood as the actions on the motors that produce a change in each Euler angle, in the case of X, Y and Yaw inputs. Since it is not possible to make both the test bench and the quadrotor's centre of gravity coincide, the inputs could induce a change in other outputs. For example, the 'X' input could provoke a response of the roll ϕ, which is unwanted. This fact was considered negligible.

6.1 EXPERIMENTATION

Although the experimentation process started in chapter 5, the identification process is documented in this section. The chapter utilizes the system dynamics studied in Equation 1 to understand the underlying dynamics of the system and better adjust the experiments. The angular position for every Euler angle was available both from the test bench's encoders and the quadrotor's IMU sensor, the latter was published on a ROS topic to be gathered by Matlab as explained earlier.

It is necessary to point out that the experiments were not entirely repetitive for any interchange of battery. The battery change displaced the system in all directions. An example is displayed in Figure 25. That meant that the system could be unbalanced between two consecutive experiments, especially when an interchange was involved. A practical approach and adjustment were needed to keep the system as horizontal as

possible. The end goal was to find suitable models that made it plausible to tune flight controllers mathematically. A robust control technique was needed to counteract the limitations of the system identification algorithms here employed.

It was decided that all input variables would be tested separately. The decoupler built in by Ascending Technologies (AscTec) allowed this type of system identification. Additionally, it was decided that the first variable to be tested was the pitch θ. It would be followed by roll ϕ and yaw ψ, respectively (Figure 34). One of the hypotheses that resulted from section 5.2 was that the pitch θ and roll ϕ angles present a similar response to a given input. The yaw angle was left last due to the different nature of the movement. That came from the fact that gravity produced no torque opposed to the system's rotation around the Z axis other than the friction's torque, unlike the pitch and roll rotations. The results will be discussed later in subsection 6.2.3, where the outcome of the outer gimbal's movement is further explained. Note that the quadrotor can have positive and negative values for all inputs in terms of percentage, from 100% to -100%. The thrust input (Z) is the only exception, with a range of 0% to 100%, because it controls the altitude of the UAV. For more information, refer to section 4.4.

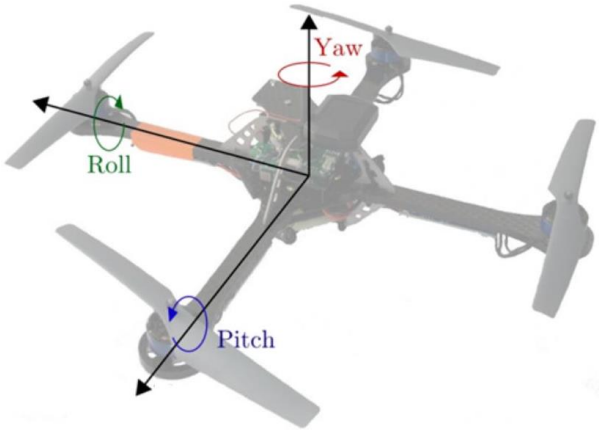

Figure 34: AscTec Hummingbird quadrotor's rotations

Adapted from source: [84]

6.1.1 PITCH ANGLE EXPERIMENTS

The first experimentation conducted was therefore applied to the pitch input variable (X in percentage referred to Figure 33). The input and output datasets for the first iteration are displayed in Figure 35. The first 15 seconds was the stabilization period, which was carried out in every test. The data clearly shows that the pitch angle resembles a nonlinear second

order system, because the jumps between 0 to 5 do not coincide with the jump between 5 to 10 and on. If it were linear, the inputs would produce the same proportional reaction in the pitch angle. That raised the question whether the initial experiments were taking the apparatus too far and into a nonlinear region of operation. The nonlinear nature was studied in chapter 5. The system's motion around the pitch and roll gimbals is explained by Equation 2, which had nonlinear components.

A change in the experiments was introduced to test this hypothesis. It assumed that the system could be regarded as linear around an equilibrium point. Note that this assumption does not work for every system [62]. Setting a threshold of ±5%, the hypothesis was that the system could be approximated as a linear system if the angles were small ($\sin\theta \cong \theta$, Equation 2). It was then tested with the input signal seen in Figure 36. The result (output signal) is also displayed. From these lines of experimentation two main conclusions were drawn for the system: a) it could not be regarded as linear, even if it were working around an equilibrium point, and b) it did not have a dead zone for this gimbal, which dismissed this potential source of nonlinearity.

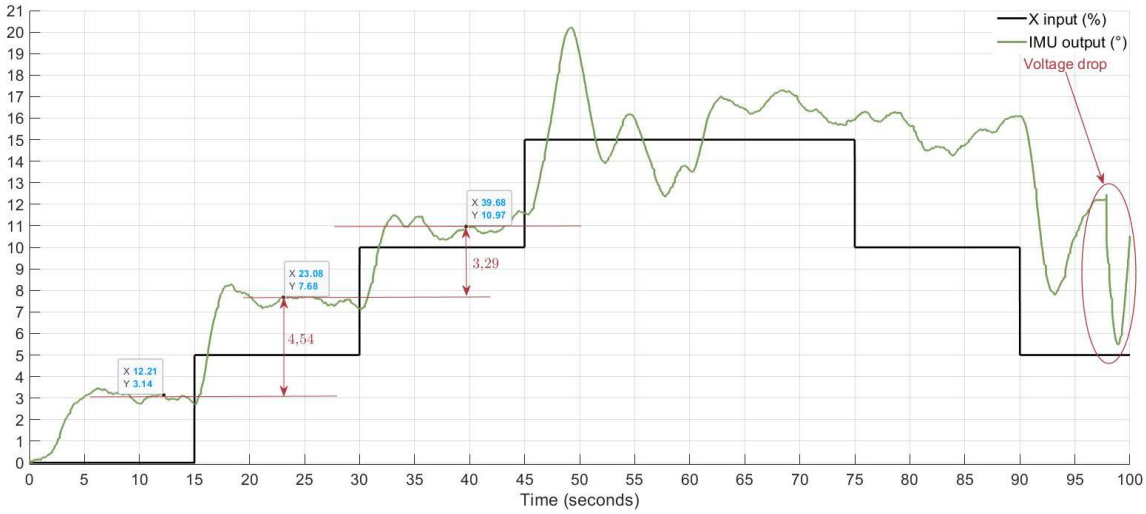

Figure 35: First iteration's input and output

Another hypothesis for this apparent nonlinearity was that the thrust in those initial experiments was insufficient to replicate the hovering state of the drone in the air. The value was set at 15% in the first iteration. To put this second hypothesis to test, a measure of the input values that produced the hovering state was needed. For that, an informal experiment was carried out. The UAV was dismounted and left on the ground. Then, while keeping the

X, Y and Yaw inputs at 0%, the thrust input (Z) was increased until the hovering state was reached. The percentage value of thrust at which that happened was noted, which happened to be 42,5%. The third iteration applied the results from this test and was carried out with this value.

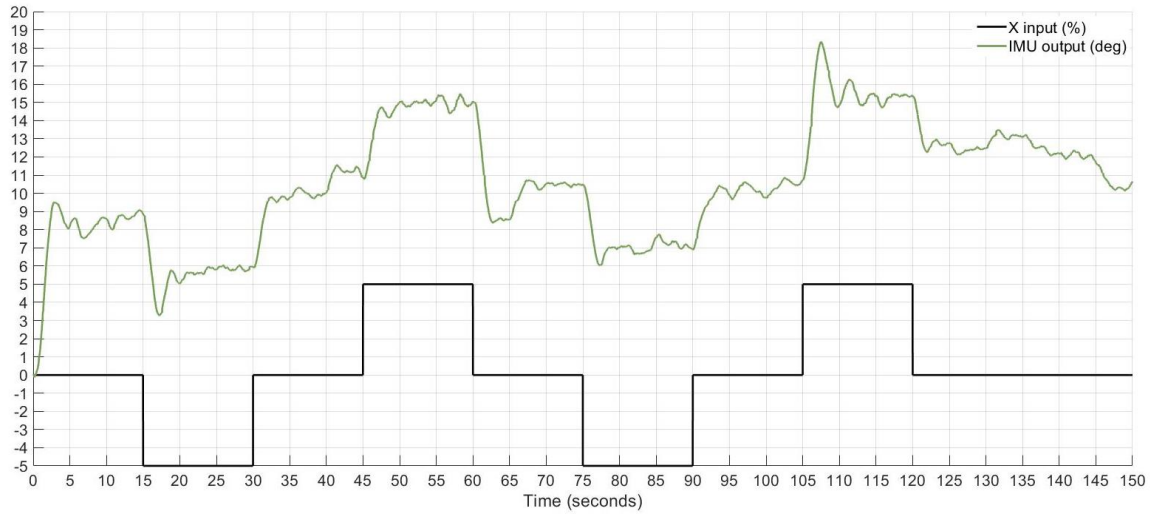

Figure 36: Input and output signals for second iteration

Some important notes gathered from the first iterations can be laid out here. It was quickly noted that the batteries did not last long with the new thrust value imposed on all tests. In fact, they only lasted up to seven minutes of raw experimentation per battery. Unfortunately, several experiments had to be stopped prematurely. Lean time was, therefore, around four minutes of useful data per battery. This was not caused by a communication problem between Matlab and the quadrotor. The problem resided in the drone's board. Either the low-level or high-level processors (LLP, HLP) would lose connection to each other and had an impact on the motors, causing them to stop working for a brief moment (voltage drop pointed in Figure 35). This could happen at any moment during the tests and for any input and for no apparent reason. That meant that the data could only be trusted up to that timeframe, even if the processors continue to work properly.

The source of the problem was found by recording the control data being published in the corresponding ROS topic ('/fcu/control'). Matlab did publish the right commands, but the drone did not respond. Even though, it was not a MathWorks' framework problem, and to minimize it, the implementation of the controller's algorithm would be done exclusively inside the quadrotor. Only the setpoints (SP) would be delivered through Matlab. Fortunately, the quadrotor did have the necessary code to set up a control algorithm inside it. Controllers,

such as PID or any of its variants (PD, PI, etc.), were possible with relative simplicity after tuning them (see chapter 7 for tuned algorithms). Nevertheless, the quadrotor would have to work solely based on the sensors' measurements for that part. The test bench encoders could still be used for the model identification that is presented in this chapter.

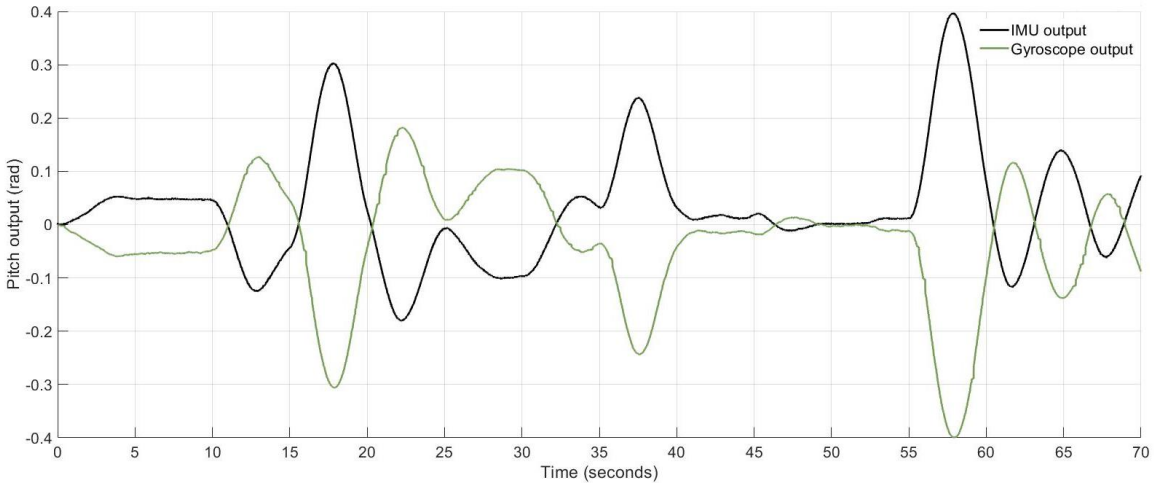

Figure 37: Pitch angle output. IMU versus encoders

For all the experiments, the drone was oriented in such a way that the Euler angles coming from both data sources were inverted (Figure 37). This mistake could easily be mended by a posttreatment of data by inverting the signals. It was left at that position throughout the experimentation because this inversion would not pose any impediment.

The longer the experiments lasted, the more informative the data collected from them is [54]. Because of the sudden communication failure, the experiments could only last up to 150 seconds without presenting problems, but usually they had a much shorter duration.

Although the first iterations seem informative and an identification algorithm could find a model that explains it in linear terms, the inputs lacked contents in the frequency domain that would excite the system enough to call the output *informative enough* [54]. The inputs so far served the purpose of initial understanding of the inner workings of the system. They also gave enough information to expect a second order system resulting from the identification process for the pitch angle θ. The next few iterations were modelled as white noise. As it has equal amount of energy across all the frequencies, it was the perfect input signal. An example can be seen in Figure 38 with a period of 5 seconds between jumps. It is displayed alongside the output signal collected from the encoders.

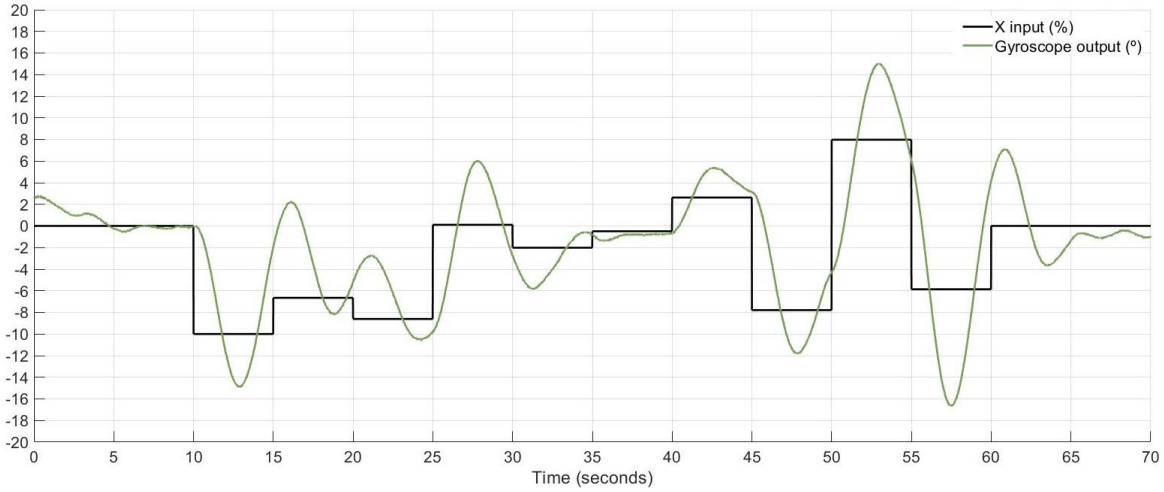

Figure 38: Input and output signals for the next iterations

Several tests were carried out using this type of input, ranging between ±10% of the total power of the system. They were used to identify the model for pitch θ, as it will be shown on the following section.

6.1.2 ROLL ANGLE EXPERIMENTS

After gathering enough data for pitch θ, it was expected that the roll ϕ would behave similarly. To test it, the same procedure and tests was applied. One of the initial experiments is displayed in Figure 39. Varying the Y input (power reference in usage percentage) between ±5%, visual comparison between pitch θ and roll ϕ gave out an important initial conclusion. It was clear that the roll response was much faster than the pitch response. This dynamic characteristic was expected to be identified by the algorithm while performing the system identification in Matlab's toolbox. The drone-gyroscope duo responded objectively faster to the same input range for this direction and had no delay. It was also an important finding in chapter 5, because inertia in this direction was smaller. The system oscillated much faster around the roll axis than around pitch axis. Another element that was clear is that roll angle ϕ did not have a dead zone around its equilibrium point.

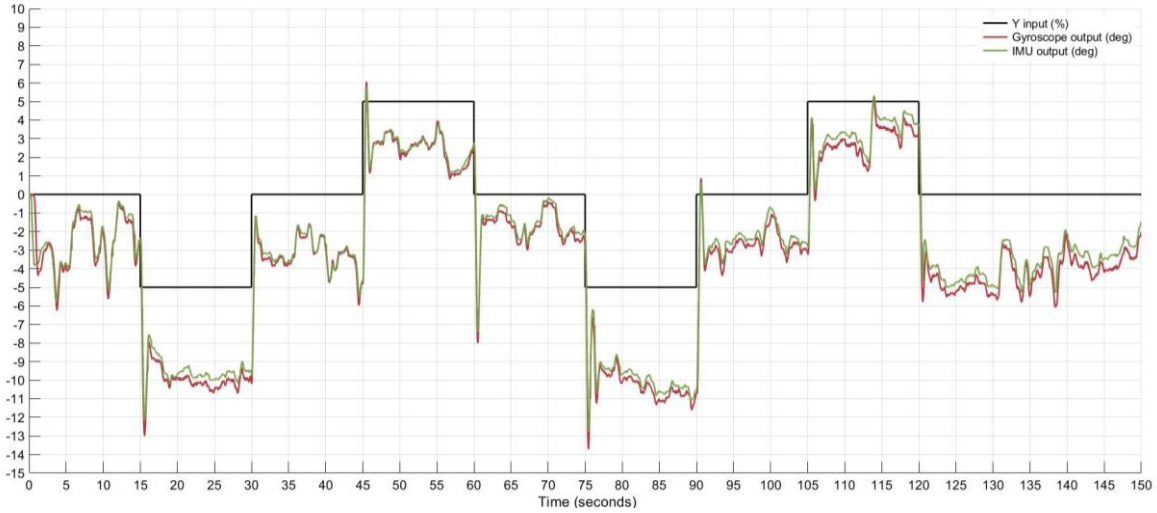

Figure 39: Initial input and output signals for roll

Visual comparison between the data quality of the inner gimbal (pitch) and the plate (Figure 36 and Figure 39 respectively), the hypothesis arose that either: a) the noise came internally from the boards and were not due to external disturbances, b) this type of rotation was affected by some unknown source of noise which moved through the physical system and affected the motors' performance, or c) the roll angle had a lower inertia than the pitch angle and therefore moved much faster, ergo the error was amplified.

The first hypothesis was tested by comparing both the gyroscope and the encoders outputs, as seen in Figure 39. Between both signals, a difference can be seen. That difference could be imputed to the IMU readings. The problem may also be due to the drone not being entirely tightly fastened to the platform, which was intentionally done to avoid damaging its components. Because of it, the gyroscope's readings will be favoured against the IMU's in the identification phase. The other two hypotheses were tested in section 6.2 during the identification and validation processes.

More dynamic inputs were experimented, with 5 seconds step changes in the input signal, one of which is displayed in Figure 40. The higher frequency of the input signal will allow the algorithm to better identify the dynamics around the roll axis, as done for the pitch axis. The results resemble the ones from a second order system, perhaps with a zero in the transfer function. They will be further analysed and utilized in subsection 6.2.2.

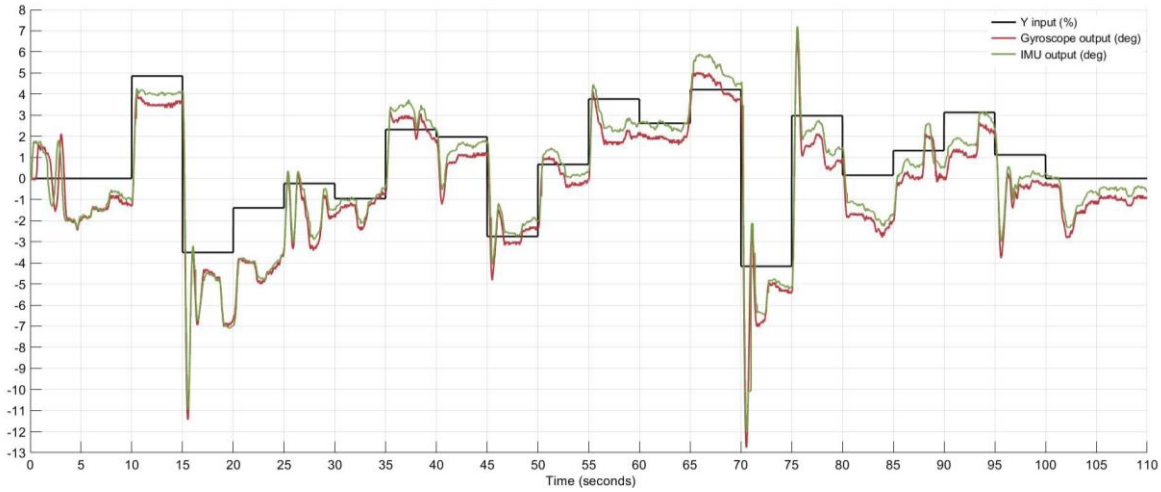

Figure 40: Second iteration of roll experiments

6.1.3 YAW ANGLE EXPERIMENTS

Third and last came the yaw angle or outer gimbal. As a general comment and unfortunately, the yaw output shares the namesake of its input 'Yaw', as displayed in Figure 12 in the Bus Assignment block, which is explained on the paragraph below it. It can also be seen in Figure 33. Physically, it was clearly different from the rest. The first difference was that it was not directly affected by gravity as the other two. Therefore, the gravity component of Equation 2 ($\vec{F} \times \vec{r}$) is equal to zero. That means that the system had infinity amount of equilibrium points, since it was stable at any given angle of the 360º degrees of rotation for an input equal to zero. In addition, the intention behind the identification is to control the angular position of the outer gimbal. When an input signal was applied to it, the angular position constantly changed. The reason behind it was that the system has a natural integrator. Summarizing, for a given constant input, the constant output response of the system was the angular velocity of the gimbal $\dot{\psi}$, not its angular position ψ.

For the reason explained previously, the angular velocity readings were initially intended to be utilized for the system identification. Then, the integrator would be forcibly added to the resulted model, and the complete mathematical expression that related the input with the angular position ψ would be obtained. To that end, three sources of angular velocity were available: a) the IMU's angular velocity published on the '/fcu/imu' ROS topic, b) the angular position coming from the same ROS topic, or c) the angular position coming from the gyroscope's encoders. The last two options could then be differentiated to obtain the angular velocity. Unfortunately, all of them are indirect measurements of the variable,

which is highly discouraged [62], [85] because of its high sensitivity to noise. Ideally, the measure should be done with a tachometer, which is more robust against noises. The differentiation of the angular position coming from the IMU was therefore discarded, because its noise would only be amplified by the mathematical operation.

In the first iteration of experiments, the velocity datasets obtained from the IMU were recorded, clearly exhibiting a signal with a high noise content. The input and output signals are displayed in Figure 41 and Figure 42 respectively. Those experiments were hardly usable to identify the underlying dynamics for obvious reasons, but it did show an initial sign of a dead zone around ±5% of power usage.

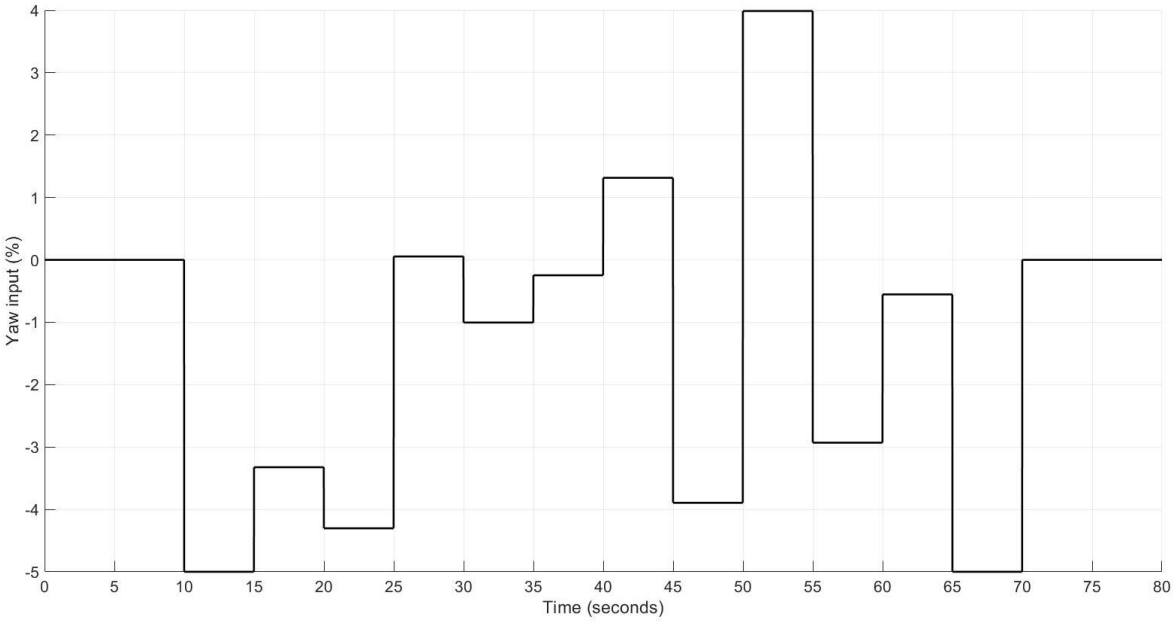

Figure 41: Input signal. First iteration of outer gimbal's experiments

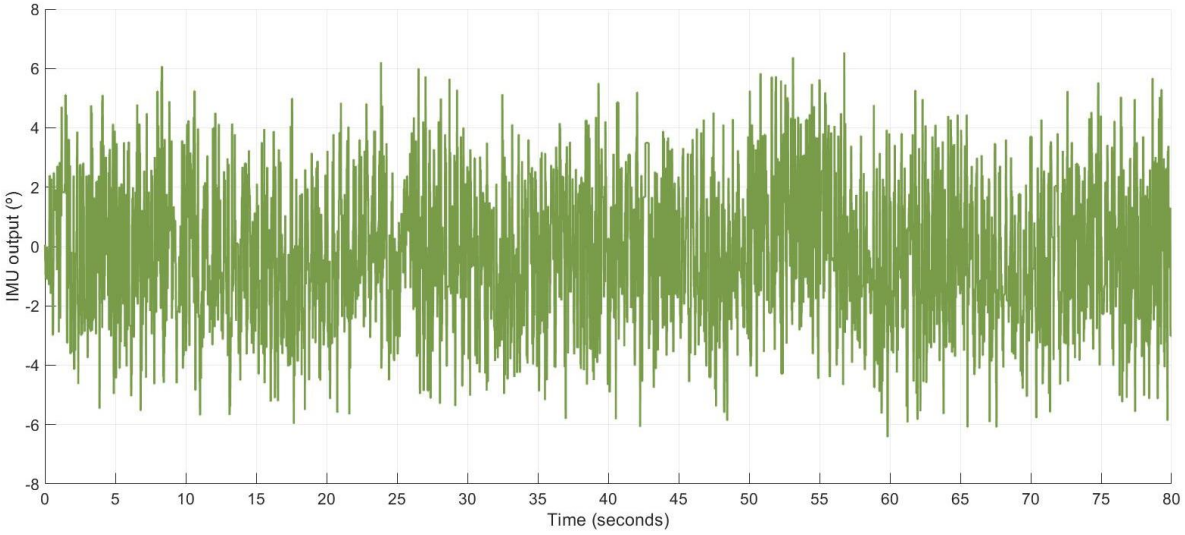

Figure 42: Output signal. First iteration of outer gimbal's experiments

The second iteration followed with an extended range in the input power applied to the system, ranging between ±40%, far exceeding the previous ranges of the other two variables. This can be observed on Figure 43. The outer gimbal (yaw) was expected to be the slowest because of its high inertia with respect to the plate (roll) and inner gimbal (pitch), as commented in subsection 5.2, which also demanded the increase in the power range. It could also be hypothesized that yaw is linear in an extended range of its input, because the nonlinear angle-dependent gravity component of Equation 2 did not affect it.

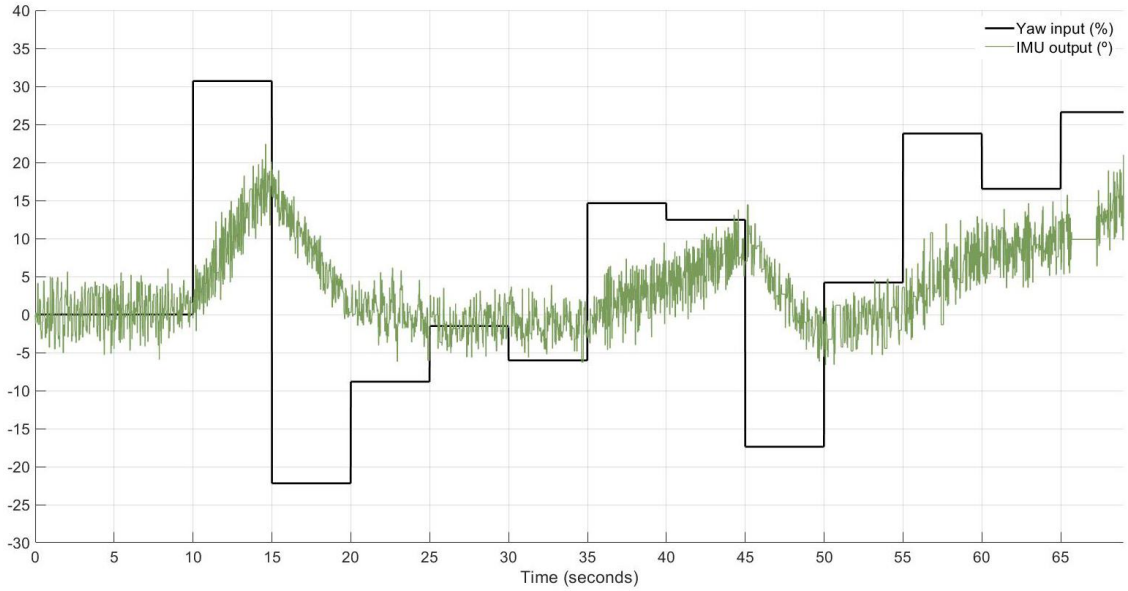

Figure 43: Second iteration of outer gimbal's experiments

Due to the high inertia, whose influence stood out during the second iteration, it was decided that the ‘Yaw' input signal’s period needed to be increased from 5 seconds to 15 seconds. This was done to better capture the dynamics throughout the tests. Nevertheless, this impacted negatively on the information contained on the signals recorded from the experiments. This is because the increase in period meant a decrease in frequential content of the input signal.

Therefore, the system could not be as energetically excited as previously expected. One way to counteract it was to extend the duration of the experiments [10]. The third iteration included this improvement. For this row of tests, an extended range of inactivity was seen beyond the ±5% dead zone discussed earlier, which will be tested further and validated in the third iteration. Note that the signal from the gyroscope (Figure 44) has the opposite direction as the signal from the IMU (Figure 45) for yaw.

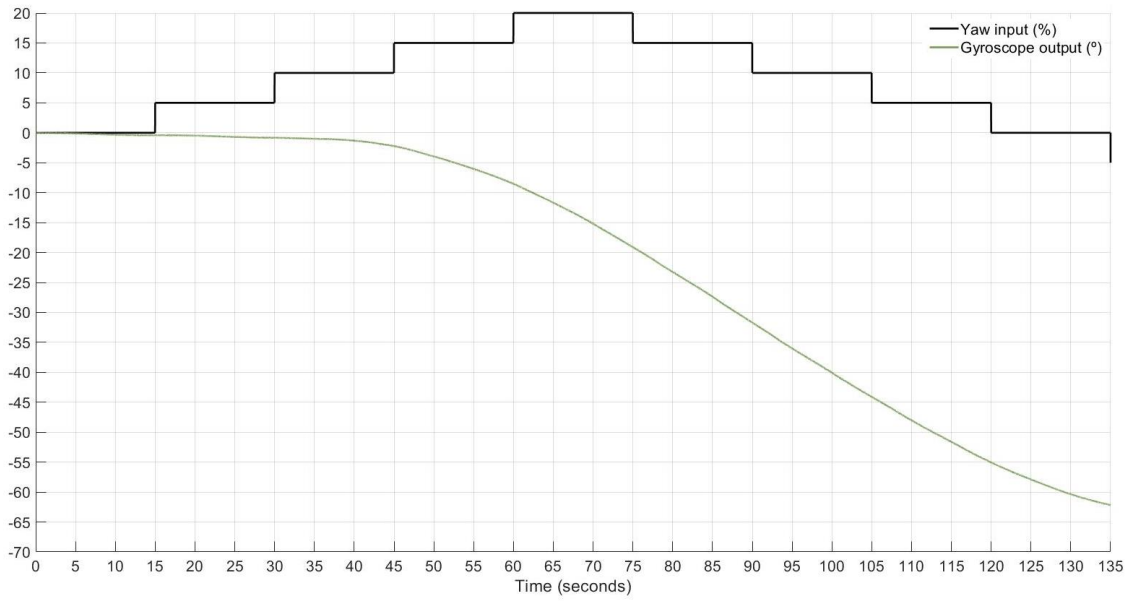

Figure 44: Third row of experiments

The third row of experiments relied on the output data from the gyroscope instead of the IMU’s angular velocity. Studying Figure 44, the dead zone was validated, as the angle was barely sensitive around ±15%. It was to be avoided intentionally in the fourth iteration. To stay away of this nonlinearity, the lower limits were set to ±20%. In the future, this property would hinder the controller’s actions over the motors to reach and maintain the setpoint (SP). For that reason, it was expected that yaw would a hard angle to control in the future.

After analysing all the previous rows of experiments, the fourth iteration was improved. This provided the identification algorithms with enough information to find a suitable model that described the dynamics of yaw, as explained on subsection 6.2.3. Previously, the output datasets were expressed in degrees as in Figure 44, unlike Figure 45, which was expressed in radians. This choice was made to accommodate the significant range of movement resulting from the extended power applied to the system.

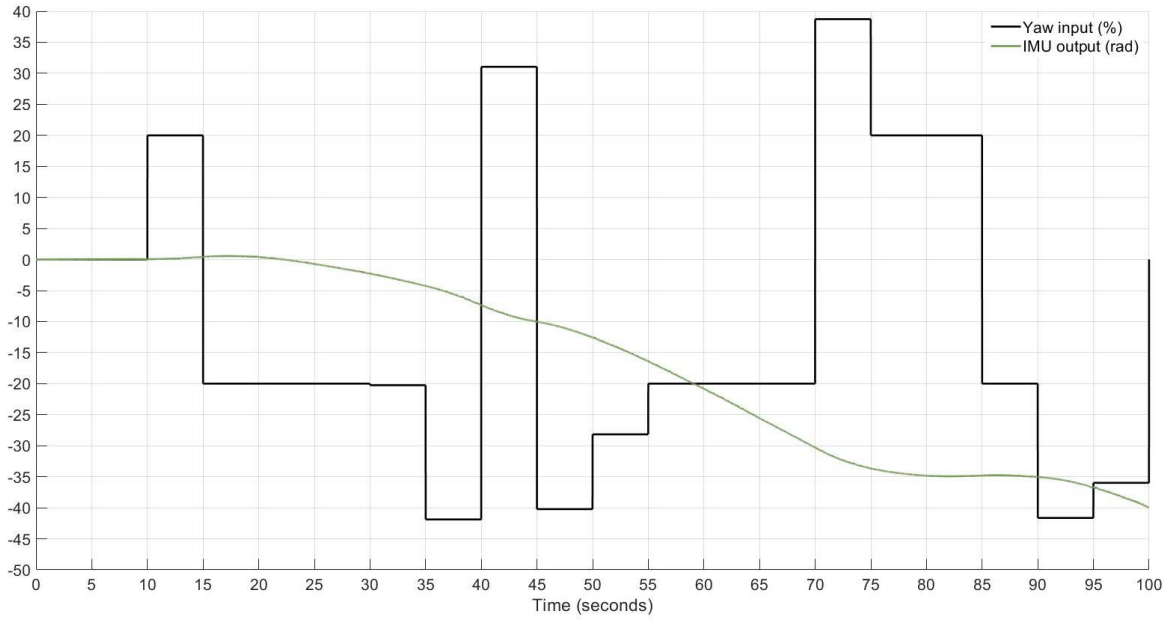

Figure 45: Fourth iteration of experiments

The last iteration concluded the experimentation phase of the model identification process. In the next section, data posttreatment will be needed to properly post-treat the data to avoid any unwanted periods, such as the stabilization time, which was usually set to be 10 seconds.

6.2 MODELS

In this section, the resulted models are calculated and analysed through the algorithms available in the System Identification Toolbox (SITB) of Matlab. The correct workflow of the identification process in the software is presented in [86], [87]. The section is divided into respective subsections for each angle to improve readability. The validation process presented in this section will be an initial phase of it, which is possible within the toolbox. The final validation will be presented in chapter 7, which are the results of the flight controller's implementation. Note that the book author of [10], [54] and the programmer of

the SITB are the same person, which lightens the weight of discrepancies in the calculations and notations between the main bibliography and the software.

Unfortunately, and as previously mentioned, the experiments were not entirely repetitive because it was not possible to maintain the UAV and its components in the same position. Especially when a battery change was needed, the system would be displaced and its centre of mass would change position. The initial validation played an important part of the identification process because it showed whether the obtained models explained the dynamics displayed by the system in different conditions. Therefore, overestimation or underestimation of the dynamics is possible.

The system identification is an iterative process that is summarized in the workflow on Figure 46. The model structure selection is based on the previous practical knowledge of the system. Given that no information was available before the start of the dissertation regarding the motion of the quadrotor-gyroscope combination, the experimentation phase had to be extended. If there were any information, the user could help the algorithm by reducing the number of poles and zeros it has to adjust. This is part of the model structure selection.

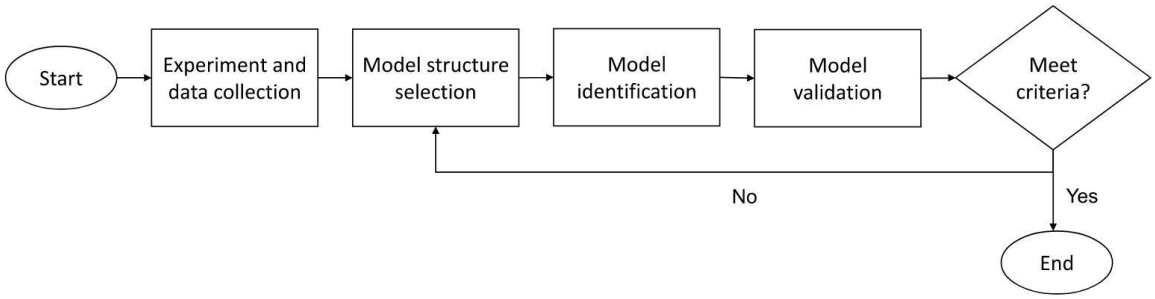

Figure 46: System Identification workflow

Adapted from source: [88]

Mathematically, each pole and zero that is selected by the user is a degree of freedom (DOF) granted to the algorithm. Ideally, this number exactly coincides with the real system. For example, overestimation could become apparent if the algorithm tries to explain noise, which is something that needs to be actively avoided to ensure the quality of the model. These errors are regularly avoided by validating the obtained model with several measurements of varying input signals.

Note that, although all the Euler angles would fit a nonlinear model better than a linear one, doing so would preclude the possibility of the mathematical calculations necessary to tune the PID parameters, as seen in chapter 7.

Statistical methods are employed to determine the numerical values of parameters that represent the absence of knowledge regarding specific details or general subsystems within a system [10]. There are three kinds of parametrized models [10], [11], [89]:

a) Tailor-made: constructed based on known physical phenomena or properties that affect a given system. Then, the parameters explain the extent to which these phenomena manifest themselves.
b) Ready-made: family of parametrised models that do not necessarily have a physical explanation, in comparison with tailored models. Their structure is used as vehicles to describe the general dynamics of the real system and work on an input-output structure. Algorithms try to find the best suited constant values for their parameters. The number of them is imposed by the user.
c) Process models: are common in many industries for modelling system dynamics and are applicable to different production contexts. These models' benefits include simplicity, help for estimating transport delays, and straightforward interpretations of the model coefficients as poles and zeros.

There were several structures available to choose from in SITB. It was stated that the dissertation objective aims to find ‘ready-made’ and process models for each angle, instead of mathematical representations of the quadrotor based on ‘tailor-made’ models. The first type of model is widely used, which, by experience, are known to be able to properly explain the dynamics of a wide range of systems if data is collected correctly.

Although there are at least four architectures (Figure 47 and Figure 48) available for this type of modelling, the experiments had a relative short duration and the Output-Error architecture worked better under these conditions. Additionally, the experience gathered in the previous phase (sections 6.1) and trial-error iterations done for this section showed that the Output-Error architectures appeared to suit the data the best, because the noise appeared mostly to be random and gaussian.

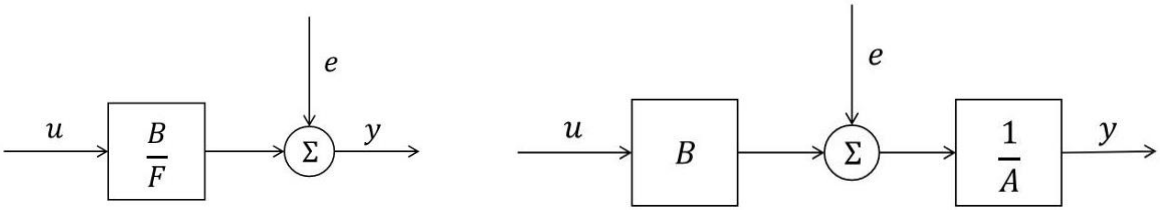

Figure 47: Output-Error (left) and ARX (right) 'ready-made' model architectures

Adapted from source: [10], [54]

All presented architectures based on the following mathematical expression, which is expressed as a discrete-time model:

$$y(t) = \eta(t) + \omega(t) = G(q,\theta)u(t) + H(q,\theta)e(t)$$

Equation 4: Linear discrete-time 'ready-made' model

Where:

$$\eta(t) = G(q,\theta)u(t)$$

$$G(q,\theta) = \frac{B(q)}{F(q)} = \frac{b_1 q^{-nk} + b_2 q^{-nk-1} + \cdots + b_{nb} q^{-nk-nb+1}}{1 + f_1 q^{-1} + \cdots + f_{nf} q^{-nf}}$$

$$\omega(t) = H(q,\theta)e(t)$$

$$H(q,\theta) = \frac{C(q)}{D(q)} = \frac{1 + c_1 q^{-1} + \cdots + c_{nc} q^{-nc}}{1 + d_1 q^{-1} + \cdots + f_{nd} q^{-nd}}$$

$\eta(t)$ is the noise-free output and $\omega(t)$ is the disturbance term. $G(q,\theta)$ is a rational transfer function of the shift operator q that explains the relationship between the input and the noise-free output of a system. $H(q,\theta)$ is the noise model present in the system, i.e. the transfer function that represents the relationship between the noise and the output. The parameter vector θ comprises the transfer function coefficients b_i, c_i, d_i and f_i, which are the subject of the identification. They are described by five structural parameters: nb, nc, nd, nf and nk. The discrete transfer function differs from a continuous one, since the discrete model is expressed in negative terms, because the system relies on past states or measurements for its future states. Thus, the negative terms appear. In the Output-Error (OE) structure, the disturbance signals are not modelled, thus, the noise model is chosen as $H(q,\theta) = 1$, which renders $\omega(t) = e(t)$. This means that the noise signal is the error $e(t)$, which is the difference between the actual output and the noise-free output [10], [54]. The

initial hypothesis to support the choice of this model is: the noise might originate in the boards and processors of the drone and not the actuators (i.e. motors), and thus, the noise would not affect the physical system.

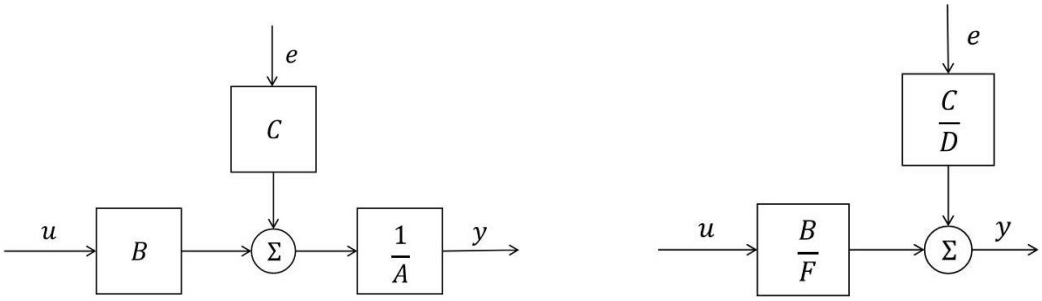

Figure 48: ARMAX (left) and BJ (right) 'ready-made' model architectures

Adapted from source: [10], [54]

A basic understanding of the algorithms implemented by Matlab to calculate the parameters in Equation 4 can be found on page 8 of [54]. Furthermore, page 363 of the book explains the computational implementation of the regressive linear Least Square Method (LSM or LSE) algorithm used by the software to minimize the cost function as to obtain the best suited parameter values of the model. The well-known cost function of the LSM method is the Mean Square Error (MSE) and is written as:

$$V_N(\theta, Z^N) = \frac{1}{N}\sum_{t=1}^{N} e^2(t,\theta) = \frac{1}{N}\sum_{t=1}^{N}(y(t) - \hat{y}(t|\theta))^2 = MSE$$

Equation 5: Mean Square Error minimized by the algorithm

Where Z^N is the row vector of concatenated pairs of inputs and outputs for a given experiment, $V_N(\theta, Z^N)$ is the cost function to be minimized and $\hat{y}(t|\theta)$ is the predicted output at the discrete time t for a given row vector of parameters θ using a model structure with given poles and zeros. To minimize the expression, its derivative is set to zero and the minimization calculations can be carried out to find the estimated values of the row vector θ:

$$\hat{\theta}_N = \arg\min_{\theta} V_N(\theta, Z^N)$$

$$0 = \frac{d}{d\theta} V_N\left(\theta, Z^N\right) \rightarrow \hat{\theta}_N = \left[\sum_{t=1}^{N} \varphi(t)\varphi^T(t)\right] \sum_{t=1}^{N} \varphi(t) y(t)$$

Note that the vector $\varphi(t)$ is the regression vector, whose components are called regressors. It contains the values of the recorded outputs and inputs in the experiment introduced into the algorithm.

Although basic concepts were presented here, no insight will be provided for their implementation nor will be discussed to justify them, as is it not within the scope of this work. Additionally, the SITB automatically calculates a parameter called 'Best Fit', which is the Normalized Root Mean Square Error (NRMSE) fitness value for a given estimated initial state expressed in percentage. It serves as an indicator of the degree to which the simulated or predicted model response aligns with the measurement data [86], [90], [91], as shown:

$$fit = \left(1 - \frac{\|y(t) - \hat{y}(t)\|}{\|y(t) - mean(y(t))\|}\right) \cdot 100\% = (1 - \text{NRMSE}) * 100\%$$

Equation 6: Fit value for the NRMSE parameter

Where $y(t)$ is the validation data output and $\hat{y}(t)$ is the model's output. Analysing the formula, the noisier the output signal is, the worse the fit will be, which was especially important for the roll angle.

Residuals are, therefore, the difference between $y(t)$ and $\hat{y}(t)$, calculated as:

$$Residuals = y(t) - \hat{y}(t)$$

As a general rule, at the start of every identification iteration for every Euler angle, a post-data treatment was needed to: a) eliminate the stabilization periods at the start of each experiment, b) detrend the input and output to rearrange its range (explained in section 4.4), c) change the units of the output dataset when using data from gyroscope (degrees to radians), and, d) in case a voltage drop was present for a given recorded output, eliminate the contaminated data.

To avoid filtering the data and ergo, introducing unwanted delay in it, the model identification process relied on the gyroscope's encoders, which were of good quality after all its problems were fixed, as previously mentioned. A comparison between encoders and IMU measurements can be observed in Figure 40.

6.2.1 PITCH ANGLE MODEL

After gathering enough data for pitch angle θ and its response to several different inputs, the model identification could begin. Alongside each test iteration, there was an adjacent identification to better understand the system. Many improvements to the tests' design came, not only from the analysis of the data itself, but from the analysis of the identification models for each iteration. Only the most relevant findings will be shown in the following subsections.

The dataset in Figure 38 was utilized to obtain a model. The raw measurements from the last row of test are like the one displayed in the mentioned figure and were used to validate the results inside the software. Adding to the previous knowledge of the system gathered in previous sections and chapters, the delay had to be calculated. After several calculations done in Matlab through its 'delayest' function [92], the raw data was estimated to have a delay which revolted around ±4, which means 0,04 seconds' delay, which was negligible in practical terms. This result could mean that the command could be misinterpreting noise for delay. Even though, a second order system is expected to suit the data the best, a comparison between the models of various poles and zeros visually helped understand the system better.

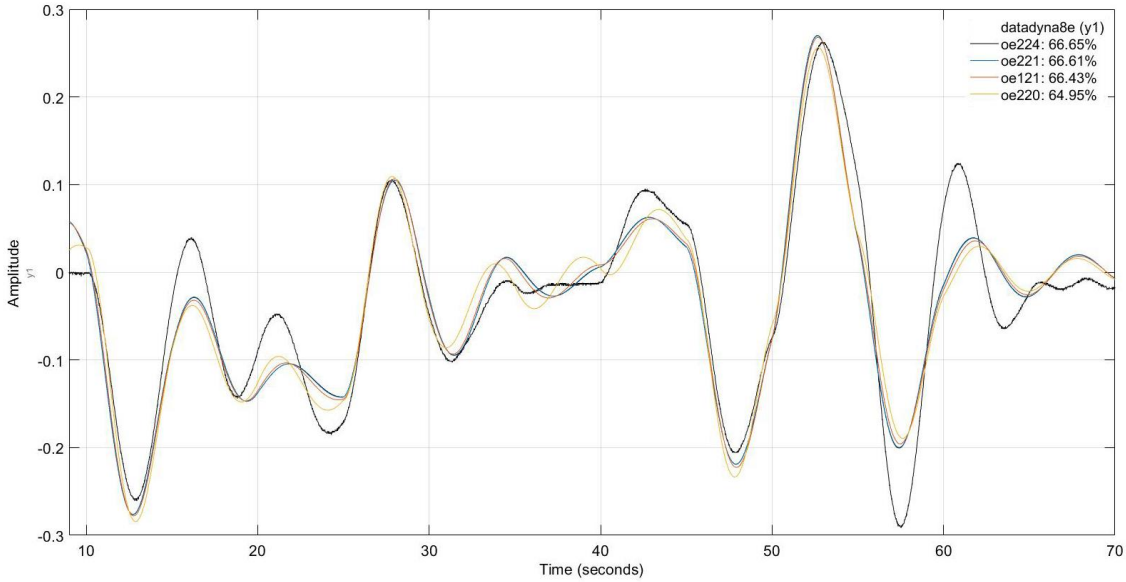

Figure 49: Pitch model identification. Comparison between lower-level models.

The workflow (Figure 46) allowed the comparison of models with different number of poles and zeros, i.e. degrees of freedom, as seen in Figure 49. Only the most significant

were left in the figure to simplify its understanding. The name of the model informs about its structure: the number of zeros (+1), poles and delay it has. For example, the ‘oe221’ has one zero, two poles and one sample delay. As explained earlier, the NRMSE fit is displayed in the figure. An early conclusion could be formulated from this result: models with one or no zero deliver similar results. This could mean that the zero on ‘oe221’ could be neglected, which was the case later, as shown in Equation 8. This is explained at the end of the subsection.

Higher degrees of freedom (DOF) given to the algorithm did not improve the fit percentage by more than 1%, as it can be seen in Figure 50. It is also clear that some of these structures have a strange behaviour, such as sudden jumps and spikes, which did not correspond with the raw data. For that reason, model ‘oe321’ was discarded, even though it had a higher fit.

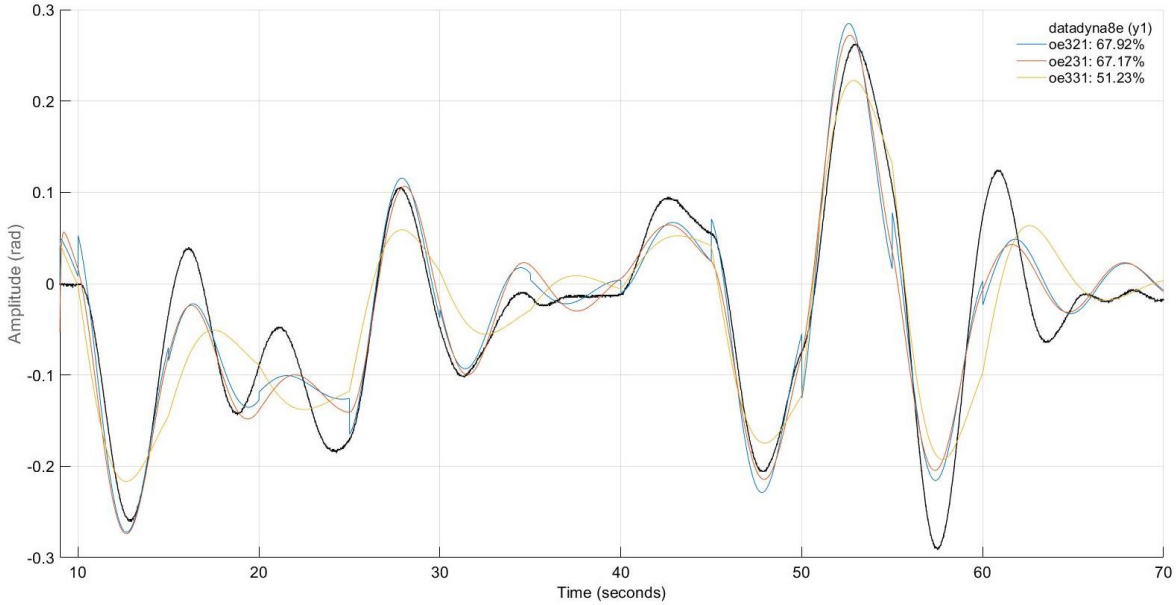

Figure 50: Pitch model identification. Comparison between higher-level models.

Four other experiments besides Figure 38 were utilized for the initial validation. One of them is displayed in Figure 51. The system with two poles and one zero performed the best versus the validation data throughout the process. As a delay equal to zero is not physically possible for a discrete model, the ‘oe220’ model was discarded and the ‘oe221’ was chosen as the pitch model moving forward. The chosen model is the following:

$$\hat{y}(t) = \left[\frac{B(z)}{F(z)}\right] u(t) + e(t)$$

$$B(z) = 1.115 \cdot 10^{-5}\, z^{-1} \; - \; 9.589 \cdot 10^{-6} z^{-2}$$

$$F(z) = 1 - 1.998\, z^{-1} + 0.9978\, z^{-2}$$

Equation 7: Chosen discrete-time pitch angle model

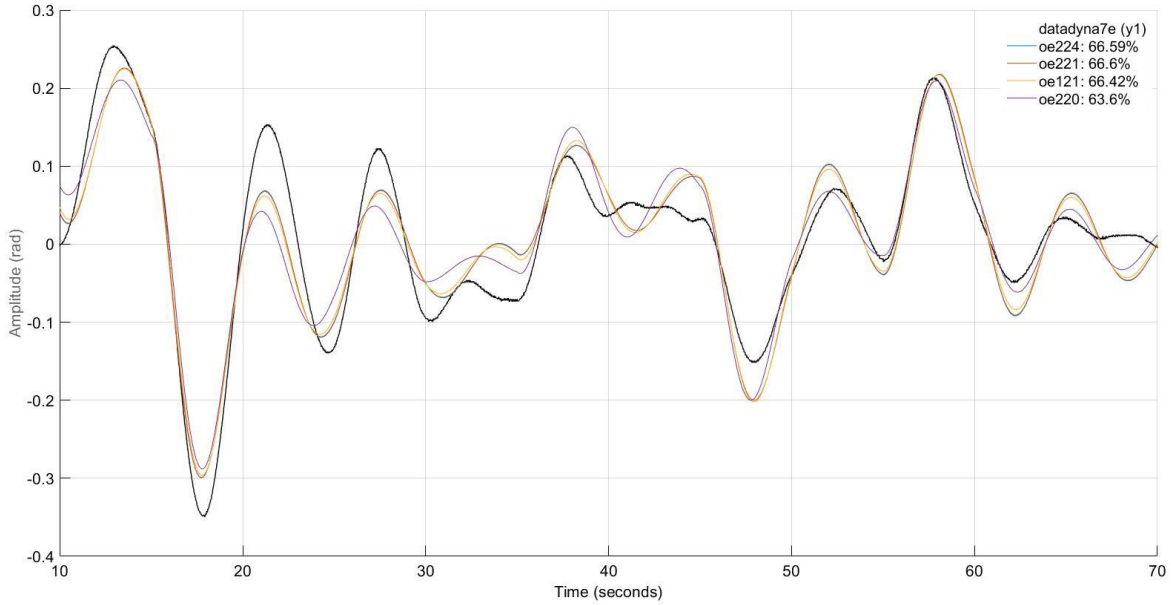

Figure 51: Pitch model validation experiment

Given that the model is an OE type structure, the noise is not modelled (Figure 47). The model may seem to have low fit values for the cases shown in Figure 49 and Figure 51. An analysis of the residual errors of the model in a histogram was due to further understand it. As it can be seen in Figure 52 and Figure 53, the noise does not appear to be completely of gaussian distribution. According to [10], [54], [88], this could be explained as follows:

1- The model (Equation 7) could be missing some of the dynamics within the real output $y(t)$ that are not explained by the estimated output $\hat{y}(t)$. Since a linear system was forced upon a nonlinear one, this outcome was inevitable.
2- Given that the cross correlation of residuals is low, the model does capture the main dynamics, that relate the input and output of the system.
3- Given that the autocorrelation of the residuals and inputs is high, it means that the disturbances are influenced by the dominant dynamics (input-output effect). This also means that the noise was not modelled correctly.

That is implicit in the OE model structure selection. The noise dynamics was not modelled because the experiments were not long enough for the dynamics of both the system and the noise to be discerned from each other and correctly estimated. Further studies could prove that the noise is, in fact, white noise (filtered or otherwise).

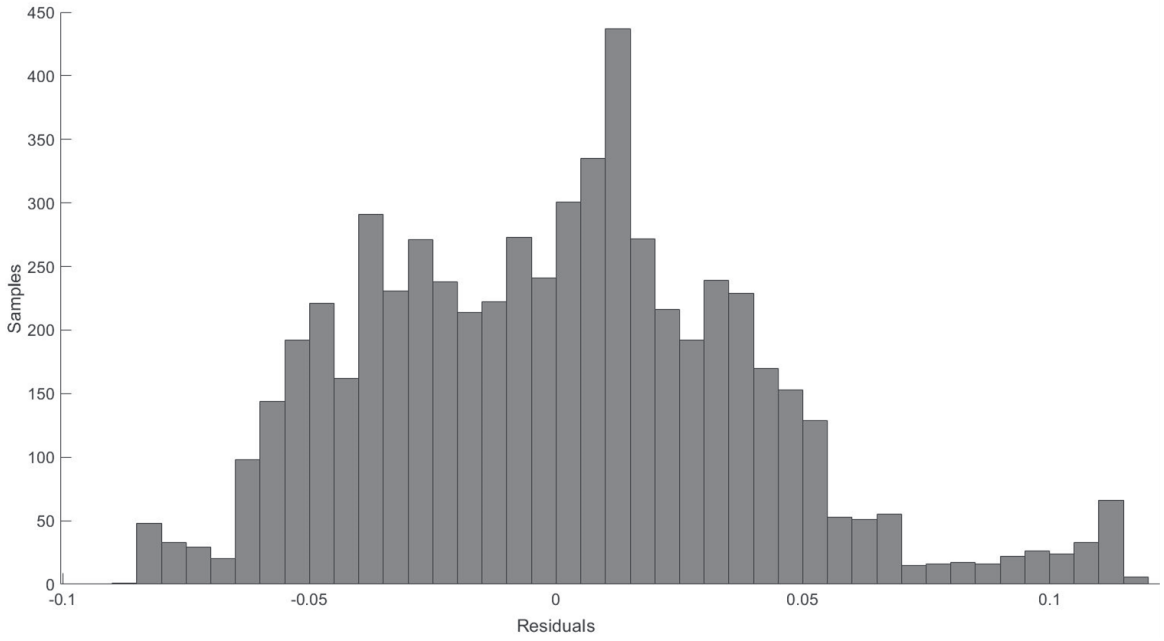

Figure 52: Residuals of the validation experiment

Based on the validation experiment (Figure 51)

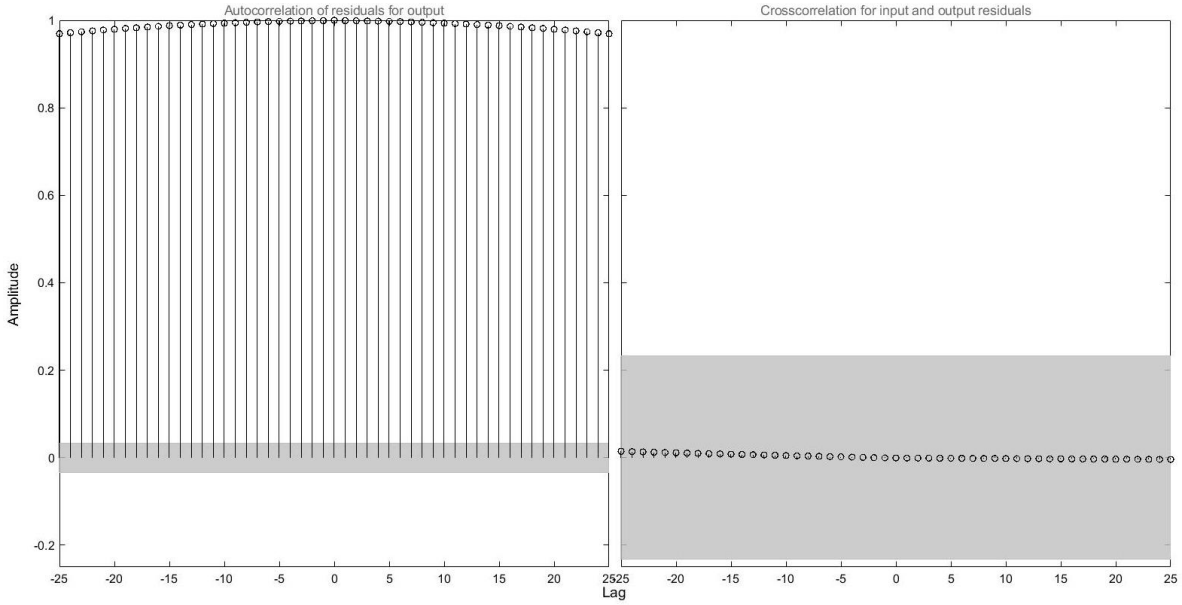

Figure 53: Autocorrelation (left) and cross correlation (right) graphs

Based on the validation experiment (Figure 51)

Additionally, the definition of a 'good model' (i.e. model quality), is dependent on its final goal or purpose, which for this dissertation is related to the model's use in control design [54]. A thorough statistical analysis and fine tuning of the obtained model would have not shown further improvements, given the conditions this study.

Continuing with the system identification, and with the information offered by Equation 7, a continuous model could also be found. In control theory, the position of poles and zeros of the system explains its dynamics. In discrete time, their position is dependent on the sample time utilized in practice to gather the data, which does not happen in continuous time. For that reason, they are displayed in the latter, which renders them independent of the time base. The continuous time model in the Laplace domain can therefore be written as:

$$G(s) = \frac{0.01485}{0.92 \cdot s^2 + 0.2012 \cdot s + 1}$$

Equation 8: Continuous model for the pitch angle

This model had the exact same fit value as the discrete-time model, which proves no information was lost in the transformation. In the continuous model, the zero from the discrete-time model was neglected because it did not affect the dynamics, as it was far from the origin. This could also be done for the discrete-time mathematical expression. The transfer function has the following complex conjugate pair as poles, which explains the oscillation of the system:

$$s_{1,2} = -0.1094 \pm 1.0368i$$

This concludes the identification and initial validation process of this angle. The continuous model (Equation 8) will be used to tune the PID controllers in chapter 7.

6.2.2 ROLL ANGLE MODEL

Moving onto the roll angle ϕ, the identification phase was done through the experiment presented on the last iteration of experiments. Particularly, the experiment shown in Figure 40 was the one used for the identification phase (Figure 54). The figure shows the posttreatment of the data, as explained for the previous angle. Without it, the fit values would drop considerably.

Note that, for this angle, the output datasets are noisier than the rest. Additionally, and as stated before, the roll angle ϕ is expected to resemble the motion of the quadrotor decoupled from the gyroscope, because the machine did not affect (add to) its inertia, in comparison to the other Euler angles.

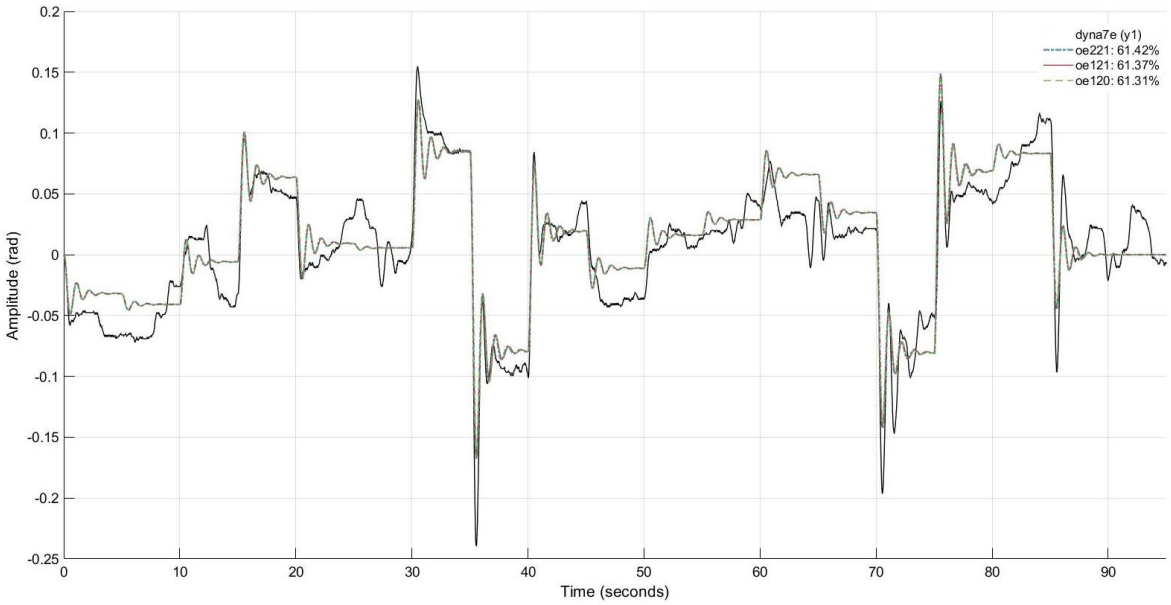

Figure 54: Identification process for the roll angle. Low-level order systems

The same workflow as for the previous angle was implemented. As seen in the identification process on Figure 54, the roll angle ϕ responded without delay, which was the results of the calculations done with the 'delayest' command of Matlab for every dataset. Second and third order systems were tried out based on previous knowledge gathered in section 6.1. No suggestion of the presence of zeros or a third pole were found during the iteration process of identification. The system is, therefore, very similar to the pitch angle, but much faster in open loop, caused by its lower inertia. The models were validated and 'oe121' was chosen based on the residuals (Figure 56), their autocorrelation and cross correlation (Figure 57). The analysis of this figures for the roll angle is analogous to the analysis for the pitch angle in the previous subsection. For readability, it will not be repeated.

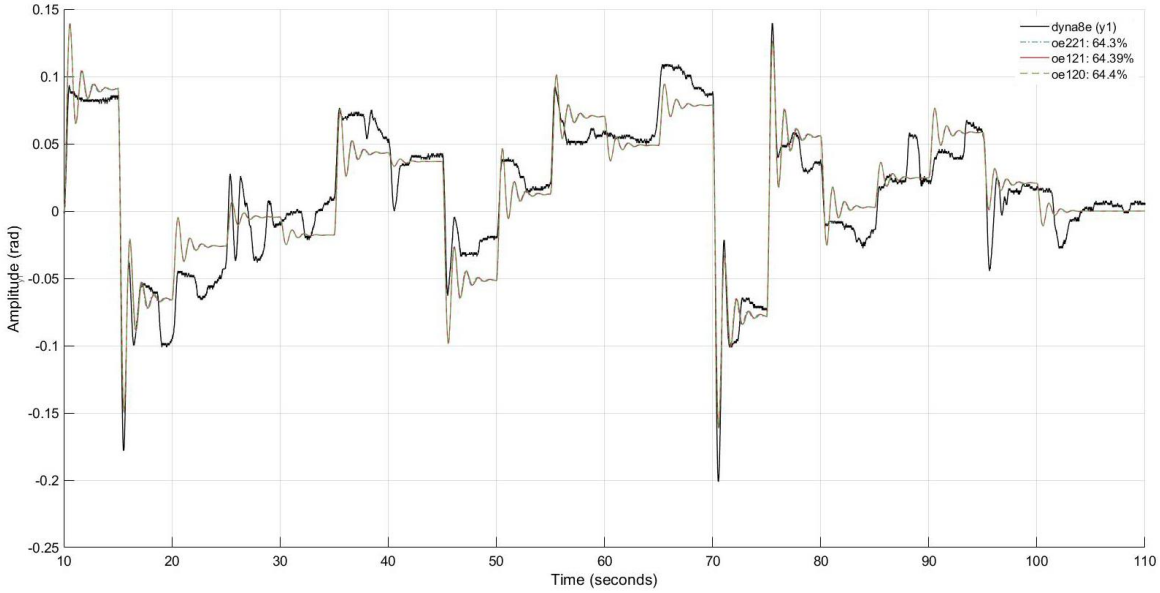

Figure 55: Initial validation process for the roll angle

Using the recorded data, and using the OE model structure, the following model was found:

$$\hat{y}(t) = \left[\frac{B(z)}{F(z)}\right] u(t) + e(t)$$

$$B(z) = 6{,}144 \cdot 10^{-5} \cdot z^{-1}$$

$$F(z) = 1 - 1{,}974 \cdot z^{-1} + 0{,}9774 \cdot z^{-2}$$

Equation 9: Discrete-time roll angle ϕ model

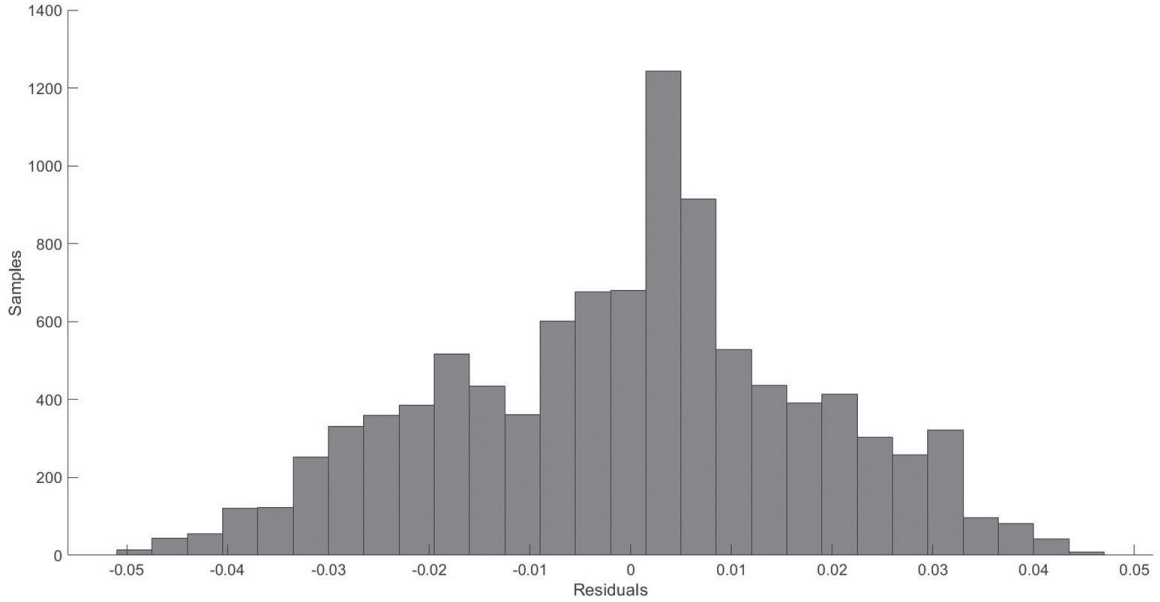

Figure 56: Analysis of residuals

Based on validation experiment (Figure 55)

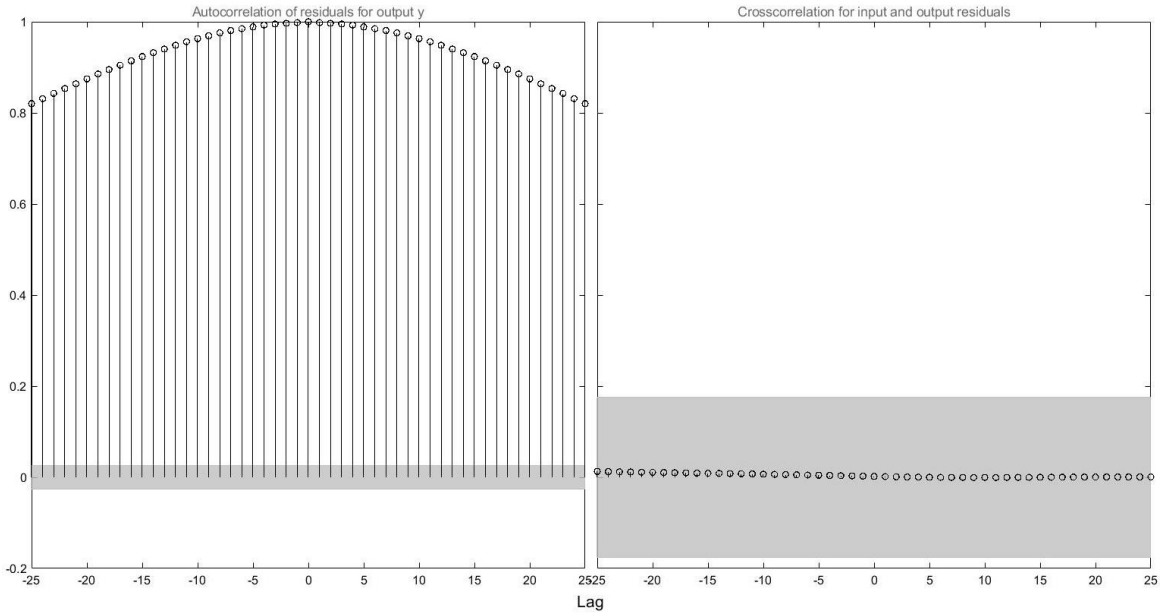

Figure 57: Autocorrelation and cross correlation graphs

Based on validation experiment (Figure 55)

In continuous time, the model can be expressed as:

$$G(s) = \frac{0.01868}{0.02961 \cdot s^2 + 0.06778 \cdot s + 1}$$

Equation 10: Continuous model for the roll angle ϕ

The poles of the transfer function are the following, which clearly show the oscillatory nature of the roll angle ϕ by its positioning in the Laplace domain:

$$s_{1,2} = -1{,}1448 \pm 5{,}6980i$$

6.2.3 YAW ANGLE MODEL

The last angle, yaw ψ, presented different dynamics compared to the other two Euler angles, which were described in sections 5.2 and 6.1. There were three sources of angular velocity available: a) the IMU's angular velocity published on the '/fcu/imu' ROS topic, b) the angular position coming from the same ROS topic, or c) the angular position coming from the gyroscope's encoders. The last two options could then be differentiated to obtain the angular velocity. Unfortunately, they are indirect measurements of the variable, which amplified the noise within the signals. In addition, the system had an apparent dead zone around its origin. For that reason, the position measurements coming from the gyroscope (option 'c') were used directly (and without differentiation) to find a suitable model.

Figure 45 shows the specific input and output measurements utilized to identify the model within Matlab's toolbox SITB. A 'ready-made' model was not possible to find, because the software could not work under the conditions imposed by the natural integrator, as commented earlier in subsection 6.1.3. The continuous model features of the software [93], [94] did allow the implementation of the integrator. An underdamped second order system was proposed alongside it. The results can be seen in Figure 58.

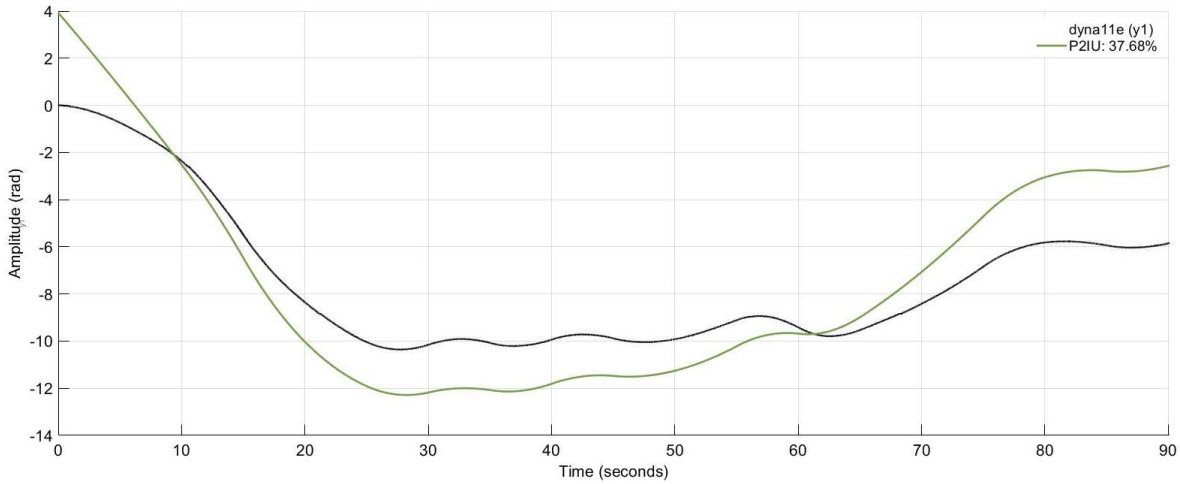

Figure 58: Identification process for the roll angle

Although the fit value is objectively low for the identification experiment (~38%), there seemed to be an error in the graph and its value in Figure 58. The result of the algorithm's optimization process is shown in Figure 59. The 'Fit to estimation data' value is the calculated with Equation 6, as explained by author Ljung [54], [86]. For some unknown reason, the displayed value and shape of the graph does not comply with the calculated data. Therefore, the correct value is ~94%. The validation experiments rendered the best results out of all the angles, reaching as low as ~82% and as high as ~94%. One of them can be seen in Figure 60.

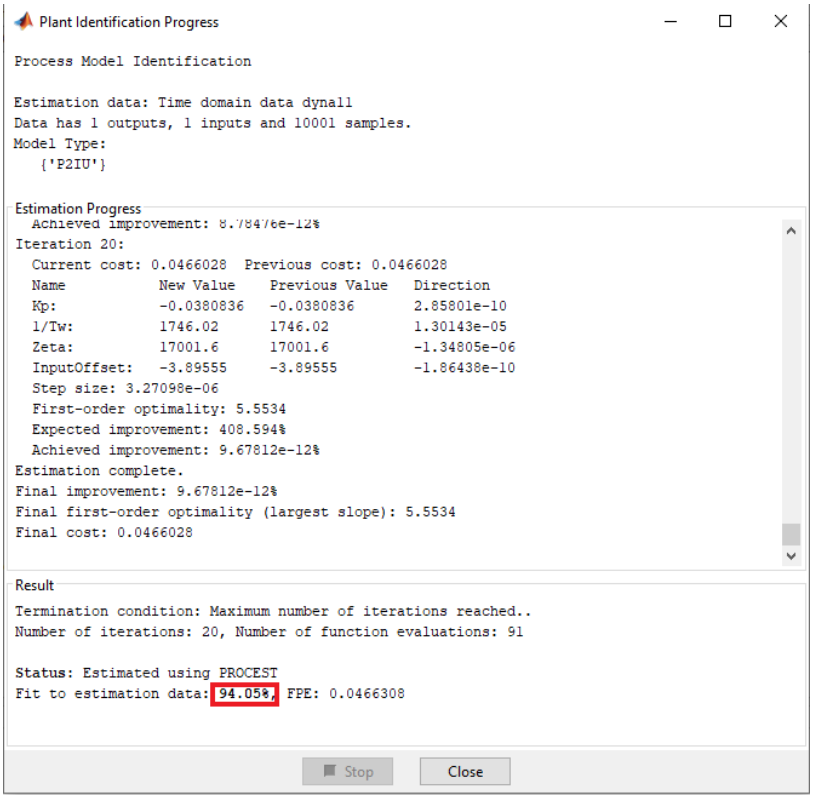

Figure 59: Results of the algorithm's optimization process for the identification experiment in Figure 58

Residual analysis can be seen in Figure 61. It cannot be considered a gaussian distribution. The noise was also interpreted through the autocorrelation and cross correlation of residuals can be seen in Figure 62. The figure resembles the findings for the pitch and roll angles (Figure 53 and Figure 57, respectively). Although the datasets for the yaw angle ψ might seem noise-free, the velocity measurements certainly had noise, as seen on Figure 42 and Figure 43. Some of it could be caused by the (apparent) indirect measurement of the velocity done inside the quadrotor's boards, which would not surface in this identification process, since those output signals are not utilized for it. Another source

of noise could be of external origin and unknown. The hypothesis for their origin were already mentioned for the other Euler angles and could be applied for the yaw angle ψ too.

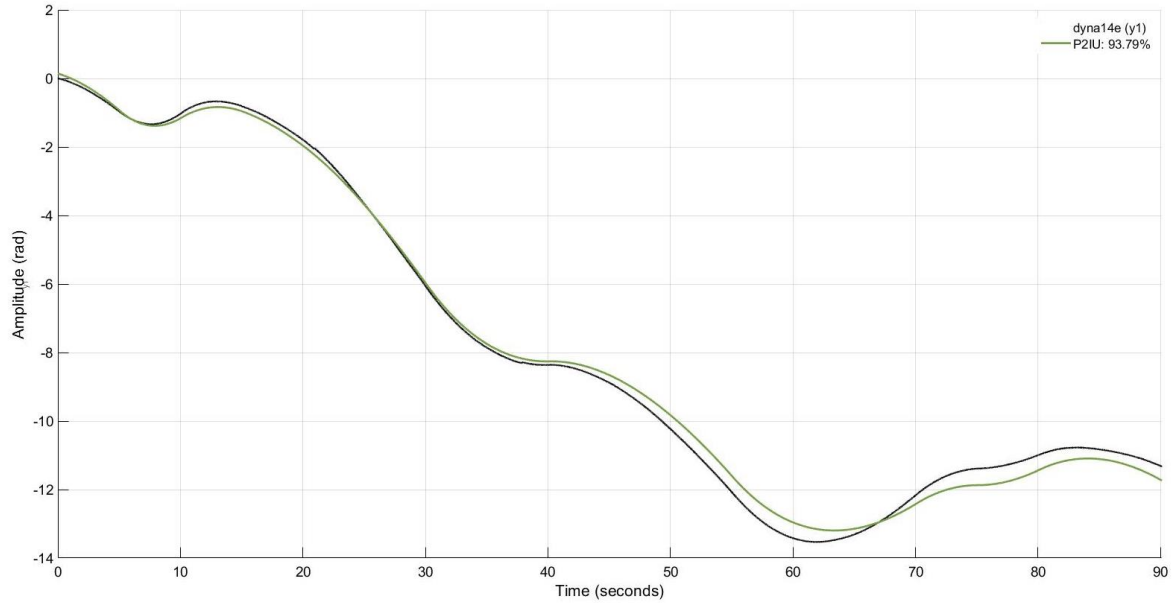

Figure 60: Validation experiment for yaw angle

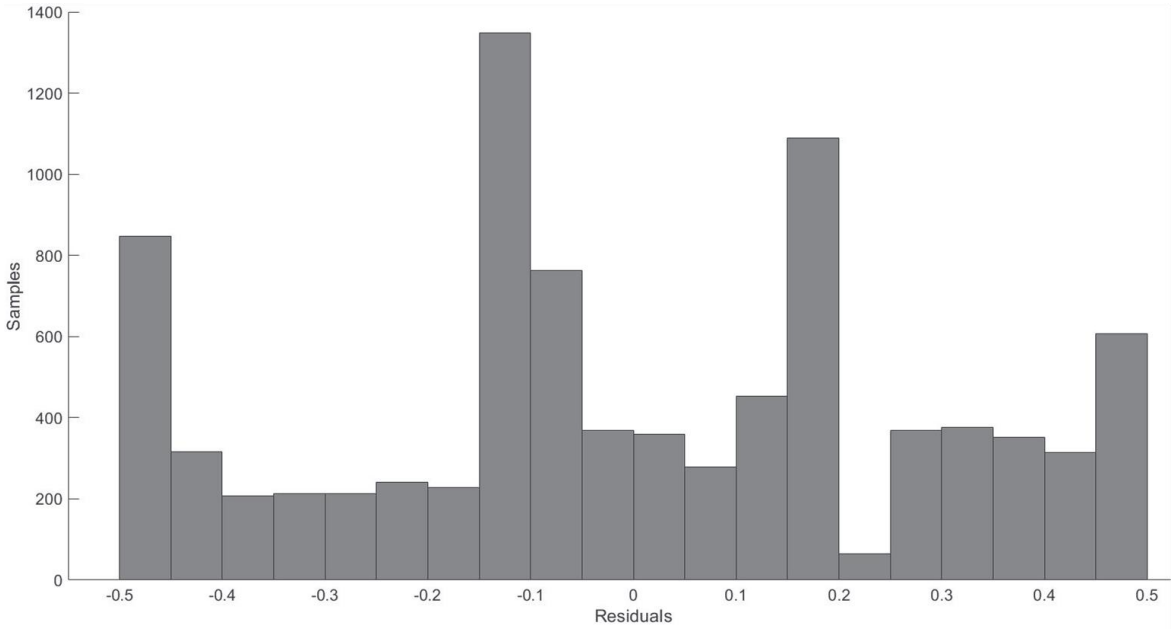

Figure 61: Residuals

Based on validation experiment (Figure 60)

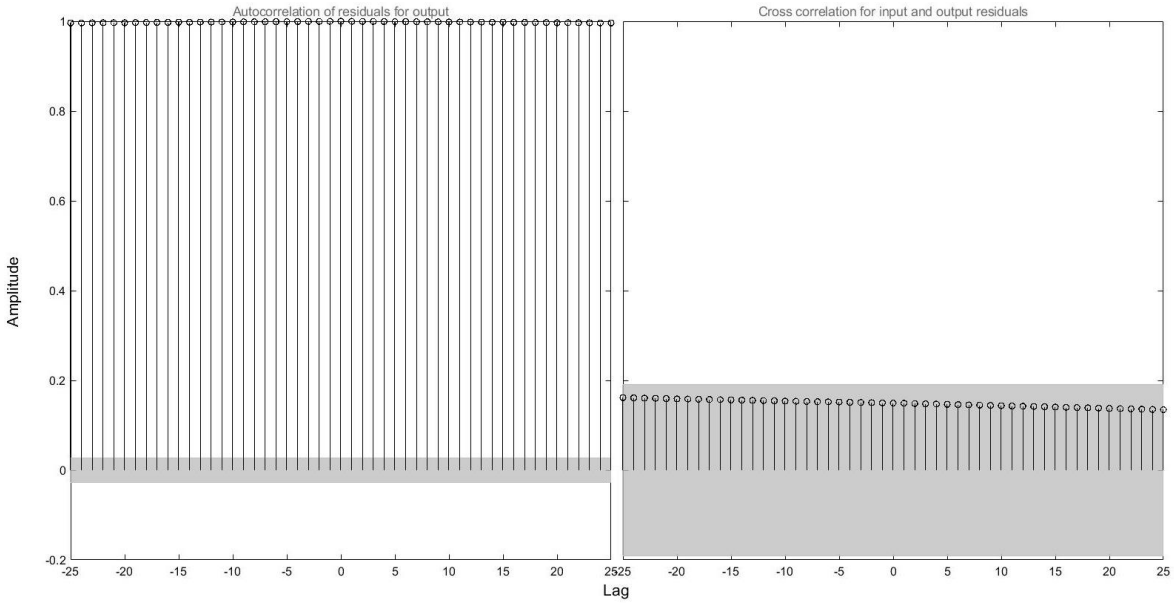

Figure 62: Autocorrelation and cross correlation of residuals

Based on validation experiment (Figure 60)

The obtained model had the following transfer function, which shows the natural integrator adjusted to its structure:

$$G(s) = \frac{0{,}03808}{s \cdot (3{,}28 \cdot 10^{-7} \cdot s^2 + 19{,}47 \cdot s + 1)}$$

Equation 11: Yaw model transfer function

Note that the yaw's direction was inverted to correspond the direction of the movement according to the drone's IMU readings. No zeros fitted into the model. The poles of the system showed that one of them (s_3) presented a very fast pole dynamics. They can be seen as:

$$s_1 = 0$$

$$s_2 = -0{,}51$$

$$s_3 = -5{,}93 \cdot 10^7$$

Therefore, the third pole s_3 could be neglected with no loss of information, and the controller's tuning could become easier. This will be done in chapter 7.

7. SYSTEM CONTROL

This chapter is dedicated to the selection and tuning of suitable flight controllers for each of the quadrotors' Euler angles, their implementation, and the results. Their deployment will be carried out separately and simultaneously. The controllers' tests were the final validation phase in the system identification process started in the previous chapter. The basic control theory concepts can be found on [60]. The necessary advanced concepts will be explained throughout the current chapter.

As explained in section 2.4, a PID structure was chosen for the dissertation. In addition, prior drone documentation asserted that it had already coded PID controllers, which could be configured easily. The results (models) from section 6.2 were used to tune the controllers.

For the new validation tests, a new interface was developed to work with the flight controllers through the MathWorks' framework. Note that the term 'system' was referenced as the drone-gyroscope duo for the other chapters. Nevertheless, in this chapter, the term will be used to describe the model's transfer function referred to each angle and will depend on the subsection. The term 'plant' is also widely used as a synonym.

7.1 CALCULATIONS AND TUNING

7.1.1 BACKGROUND

Several methods are available to calculate and tune PID controllers. A popular and (sometimes) effective method is the pole placement, which has its advantages and disadvantages [65]. To explain them and understand the method's utilisation in the calculations, a second and third order system will be firstly explained [62].

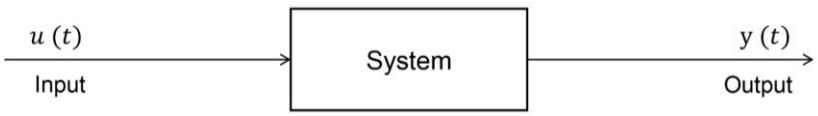

Figure 63: System representation in time domain

Adapted from source: [95]

An open loop can be presented as in Figure 63. The pitch and roll angle models (Equation 8 and Equation 10, respectively) were identified as second order models, which have the following standard form in Laplace domain:

$$G(s) = \frac{K \cdot \omega_n^2}{s^2 + 2 \cdot \xi \cdot \omega_n \cdot s + \omega_n^2}$$

Equation 12: Second order system. Standard form

The yaw angle model was identified as a second order model with a natural integrator, which renders a third order transfer function. In section 6.2.3 , it was stated that s_3 negligible. That renders a first order model with a natural integer, which effectively turns into a second order model. Naturally, this means that it has a different standard form as the other angles (Equation 13).

$$G(s) = \frac{K}{s \cdot (T \cdot s + 1)}$$

Equation 13: First order model with a natural integrator. Standard form

A third order system's standard form in closed loop can be seen in Equation 14. If the loop of the pitch and roll angle models (second order model in standard form) were to be closed and controlled with a PI, the block diagram of the closed loop system would look as Figure 64.

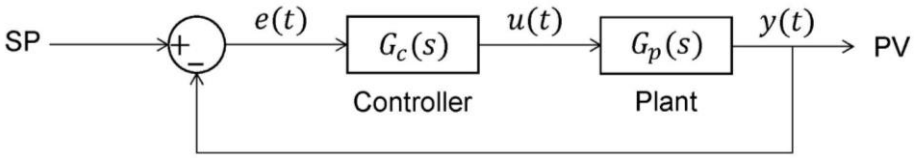

Figure 64: Closed loop system representation

$$\frac{PV}{SP} = \frac{\omega_{nd}^2 \cdot p}{(s^2 + 2 \cdot \xi_d \cdot \omega_{nd} \cdot s + \omega_{nd}^2)(s + p)}$$

Equation 14: Closed loop third order system transfer function. Standard form
Adapted from source: [62]

Where ξ_d = desired damping factor ω_{nd} = desired natural frequency

Where p is the position of the real pole of the system in closed loop. As its name mentions, the system has three poles.

$$s_{1,2} = -\xi_d \cdot \omega_{nd} \pm j \cdot \omega_{nd} \sqrt{1 - \xi_d^2}$$

$$s_3 = -p$$

The term p is unknown and, therefore, calculated based on specifications. According to [62], it can be expressed in terms of the damping factor ξ and natural frequency of the system ω_n. The proportional factor β is a distance factor between the poles of the third order system. If β is close to or smaller the unit, then it will affect the dynamics of the closed loop system.

$$\beta = \frac{p}{\xi_d \cdot \omega_{nd}}$$

Equation 15: Proportional factor of poles for a third order system

In terms of control tuning, the pole placement is a design method based on previous knowledge of the plant model. The intended outcome is to impose a desired dynamic onto the closed loop system. Regarding the input, it needs specifications, usually expressed in the time domain, such as: a) settling time and b) overshoot.

One of the disadvantages of the method is the amount of DOF the controller's structure has to offer, which is also present in other popular tuning techniques (i.e., direct tuning). For example, the PI or PD structures only offer two DOF. When the loop of a second order system is closed with a PI controller, the resulting system is a third order system with an added zero in its numerator (Equation 18).

$$G_p(s) = \frac{K \cdot \omega_{nm}^2}{s^2 + 2 \cdot \xi_m \cdot \omega_{nm} \cdot s + \omega_{nm}^2} = \frac{k}{s^2 + a \cdot s + b}$$

Equation 16: Simplified second order plant model. Known parameters

$$G_c(s) = \frac{K_p \cdot s + K_i}{s}$$

Equation 17: PI controller transfer function

$$\frac{PV}{SP} = \frac{G_c G_p}{1 + G_c G_p} = \frac{k \cdot (K_p \cdot s + K_i)}{s^3 + a \cdot s^2 + \left(b + k \cdot K_p\right) \cdot s + k \cdot K_i}$$

Equation 18: Closed loop transfer function for a second order system with a PI controller

Where $G_p(s) =$ Plant's transfer function $G_c(s) =$ plant controller

K_p = proportional gain K_i = integrative gain

PV = closed loop system output SP = setpoint for the plant

ξ_m = model's damping factor ω_{nm} = model's natural frequency

K = model's proportional gain

The mentioned system has four parameters. For that reason, the pole placement method does not always render acceptable results (as it will be seen on section 7.1.2). Unfortunately, turning the PI or PD controllers into PID does not improve the outcome, as the added DOF also increases the overall system complexity. This addition renders a third order system with two zeros. The technique purposely neglects the influence of the poles and zeros (e.g., dynamics) it cannot place (e.g., control). This is because it lacks the necessary DOF to impose the desired dynamics.

$$(s^2 + 2 \cdot \xi_d \cdot \omega_{nd} \cdot s + \omega_{nd}^2)(s + p) = s^3 + a \cdot s^2 + \left(b + k \cdot K_p\right) \cdot s + k \cdot K_i$$

Equation 19: Pole placement solution

The desired damping factor ξ_d and the natural frequency ω_{nd} are typically calculated based on the specifications of the user with the following equations:

$$\xi_d = \frac{-ln\left(\frac{M_p}{100}\right)}{\sqrt{\pi^2 + ln^2\left(\frac{M_p}{100}\right)}} \qquad \omega_{nd} = \frac{4}{T_s \xi_d}$$

Where M_p is the desired percentage of overshoot of the system and T_s is the desired settling time. Both are the specifications that were mentioned in the above paragraph. The desired frequency ω_{nd} is based on the two percent criteria for the settling time. In the following section, the problem will be tackled by different methods in order to correctly select the gains for the necessary controllers.

When a set of possible controller gains is calculated for a given purpose, a design performance index is needed. A performance index, which emphasizes the important system parameters, is a quantitative measure of a system's performance [61]. One of the widely utilized criteria for reducing system error and providing the appropriate PID gain values for desired system response requirements is integral time absolute error (ITAE) [96]. The ITAE index can both highlight faults that occur later in the response as well as minimize

the effect of any significant initial errors. The minimum value of the integral may be easily seen as the system parameters are changed, making it the performance metric with the best selectivity. However, it does not provide information about the controller's output values, since it only works with the transient response. It can be written as follows in both continuous and discrete time form (trapezoidal form):

$$ITAE = \int_0^T t|e(t)|dt \qquad \text{ITAE} = \sum_{k=0}^{M} k \left|\frac{e(k)+e(k-1)}{2} \cdot T_s\right|$$

Where e is the error between the setpoint and the output, T_s is the sampling time, k is a given sample and M is the total amount of samples. The discrete time formula is required to implement it in the calculation software. The characteristic of a system designed according to this criterion is that it has a slight overshoot in its transitory response and that its oscillations are properly dampened. This criterion has good selectivity and is an improved version of the integral absolute error (IAE) criterion. Although it may be easily measured experimentally, it is a difficult criterion to evaluate analytically. The ITAE values should be evaluated between the possible combinations of the same model, and not between two consecutive models.

7.1.2 YAW

The yaw angle model was written in Equation 11, which had a negligible pole, given its far off position in the Laplace plane. This rendered the following model and poles:

$$G_{pY}(s) = \frac{0{,}03808}{s \cdot (19{,}47 \cdot s + 1)} = \frac{K}{s \cdot (s - s_2)} = \frac{0{,}0020}{s \cdot (s + 0{,}51)}$$

$$s_1 = 0 \qquad s_2 = -\frac{1}{T} = -0{,}51 \qquad K = \frac{0{,}03808}{19{,}47} = 0{,}0020$$

Equation 20: Simplified yaw model and its poles

After that mathematical operation, the suitable controller structure was selected. Because the system had a natural integrator, it was not necessary to add it in the controller [59]. For that reason, a PD was chosen. After the analysis carried out in last chapter regarding the output signals' quality (for the yaw angle in particular), adding the derivative term could bring instability and poor control to the loop. On the other hand, if it were not present, the system would need a large proportional term, which could also destabilize the

loop. Therefore, the term was to be kept at a minimum. Using the representation found on Figure 64, the equations can be written as:

$$G_c(s) = K_p + K_d \cdot s$$

Equation 21: PD controller transfer function

$$\frac{PV}{SP} = \frac{G_c G_{pY}}{1 + G_c G_{pY}} = \frac{\left(K_p + K_d \cdot s\right)\left(\frac{K}{s \cdot (s + s_2)}\right)}{1 + \left(K_p + K_d \cdot s\right)\left(\frac{K}{s \cdot (s + s_2)}\right)}$$

$$\frac{PV}{SP} = \frac{K \cdot K_p + K \cdot K_d \cdot s}{s^2 + (s_2 + K \cdot K_d)s + K \cdot K_p}$$

Equation 22: Closed loop yaw transfer function

The transient-response specifications were set as to induce a fast response and avoid the studied dead zone (section 6.2.3). The drawback of setting it this way was that the closed loop would have a moderate overshoot. The calculations rendered the following results:

$$M_p = 10\% \qquad T_s = 40s$$

$$\xi_d = \frac{-ln\left(\frac{M_p}{100}\right)}{\sqrt{\pi^2 + ln^2\left(\frac{M_p}{100}\right)}} = 0{,}59 \qquad \omega_{nd} = \frac{4}{T_s \xi_d} = 0{,}17$$

Equalizing both characteristic polynomials (denominators) of Equation 12 and Equation 22 renders the following equation:

$$s^2 + 2 \cdot \xi_d \cdot \omega_{\mathrm{nd}} \cdot s + \omega_{\mathrm{nd}}^2 = s^2 + (s_2 + K \cdot K_d)s + K \cdot K_p$$

Equation 23: Yaw's pole placement solution

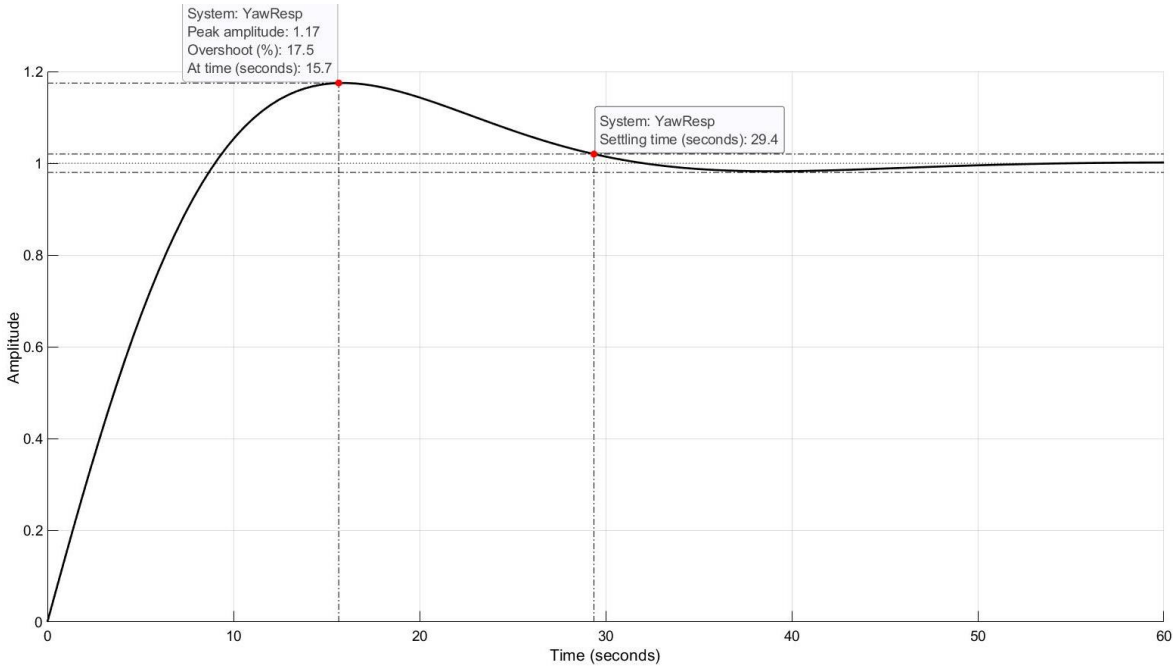

Figure 65: Closed loop yaw angle response

Therefore, the solution to the polynomial and its characteristics were:

$$K_p = 14{,}63 \qquad K_d = 76{,}01$$

$$s_{1,2} = -0{,}1 \pm j \cdot 0{,}1364 \qquad z_1 = -0{,}1925$$

$$M_{pr} = 17{,}5\% \qquad T_{sr} = 29{,}4s$$

$$\frac{PV}{SP} = \frac{0{,}1487 \cdot s + 0{,}02862}{s^2 + 0{,}20 \cdot s + 0{,}02862}$$

Equation 24: Yaw's tuned closed loop and its characteristics. First iteration.

The step response, settling time and overshoot of the tuned loop can be seen in Figure 65. Its overshoot and settling time were fairly affected by the zero. This was expected, given that the PD controller has only two DOF and the system (Equation 22) has three. The high value of its gains was surprising at first, since the system already had a natural integrator [62]. The results were also against the previous desire for the derivative term's minimization.

Nevertheless, the system was simulated in Simulink using the block diagram seen in Figure 66 to observe whether the characteristics were enough to avoid the dead zone (at least for the first part). The diagram had a saturation block to limit the PID's output (±100).

Additionally, the loop also had an anti-windup technique algorithm built in, as explained in [65]. As it will be observed in the results of section 7.3, the technique did not active. Therefore, it was omitted in the simulation phase for simplicity.

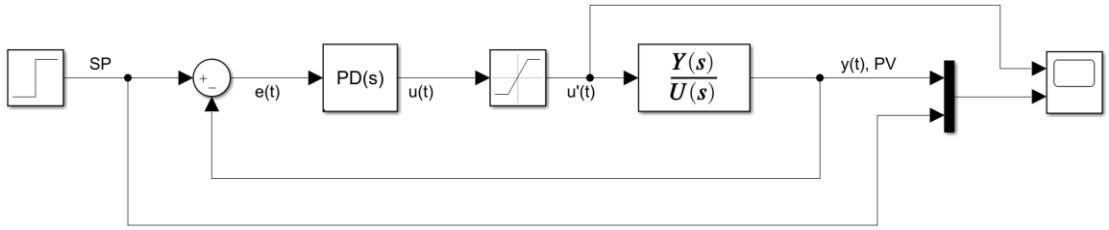

Figure 66: Diagram block for closed loop yaw angle

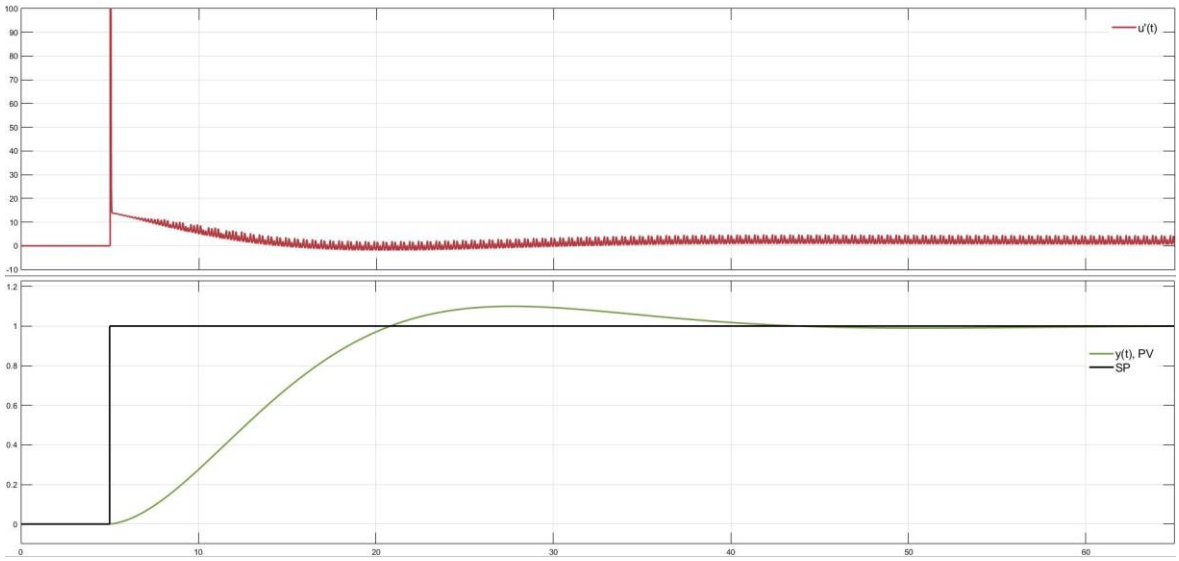

Figure 67: Tuned closed loop yaw angle response. First iteration. Controller's saturated output u'(t) (top) and response PV (bottom)

The ripple at the controller's output is clearly unwanted, as its presence destabilizes the time-discrete computational implementation. It was caused due to the high derivative term. As this response did not comply with any of the desired control signal characteristics, the PD controller was tuned again, but with more intense specifications.

A hypothesis was constructed for setting the new values, which was the following: the dead zone could be used to the control's advantage if the settling time T_s was set sufficiently high. As the second necessary condition for this tune iteration, the overshoot M_p needed to far exceed the first iteration given the previously explained reasons above. If the hypothesis is correct, then the system will settle faster, aided by the dead zone. Therefore, the following desired values were set:

$$M_p = 70\% \qquad T_s = 100s$$

Rounding up the numbers of the results, the closed loop and its characteristics were:

$$K_p = 60 \qquad K_d = 15$$

$$s_{1,2} = -0{,}0403 \pm j \cdot 0{,}3402 \qquad z_1 = -4$$

$$M_{pr} = 69\% \qquad T_{sr} = 93{,}9s$$

$$\frac{PV}{SP} = \frac{0{,}02933 \cdot s + 0{,}1173}{s^2 + 0{,}08068 \cdot s + 0{,}1173}$$

Equation 25: Yaw's tuned closed loop and its characteristics. Second iteration.

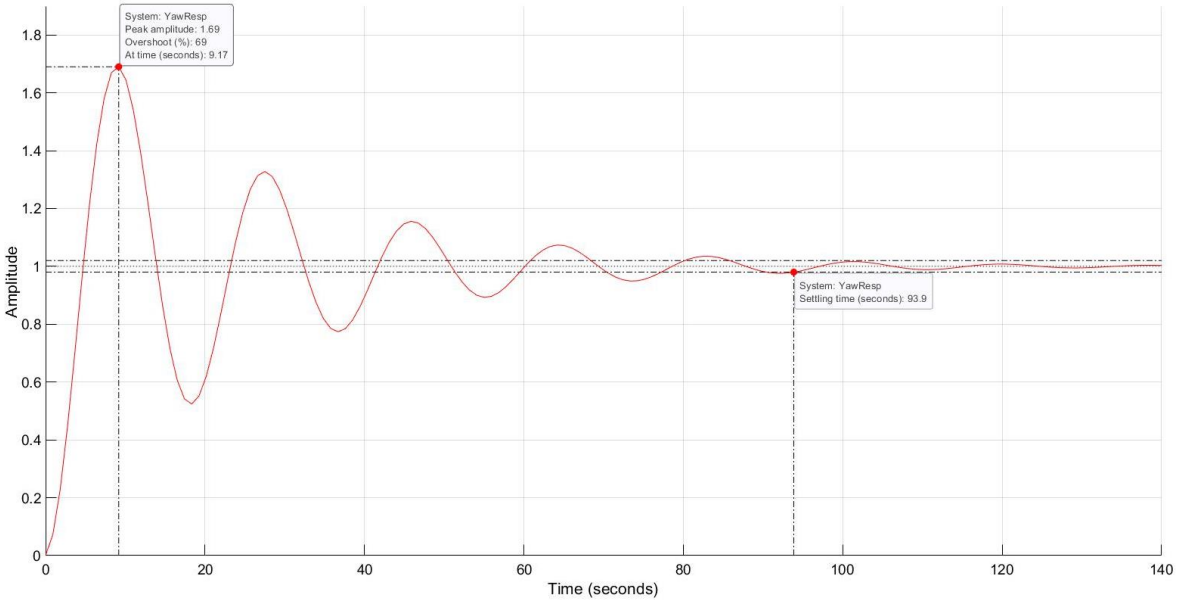

Figure 68: Closed loop yaw transfer function. Step response.

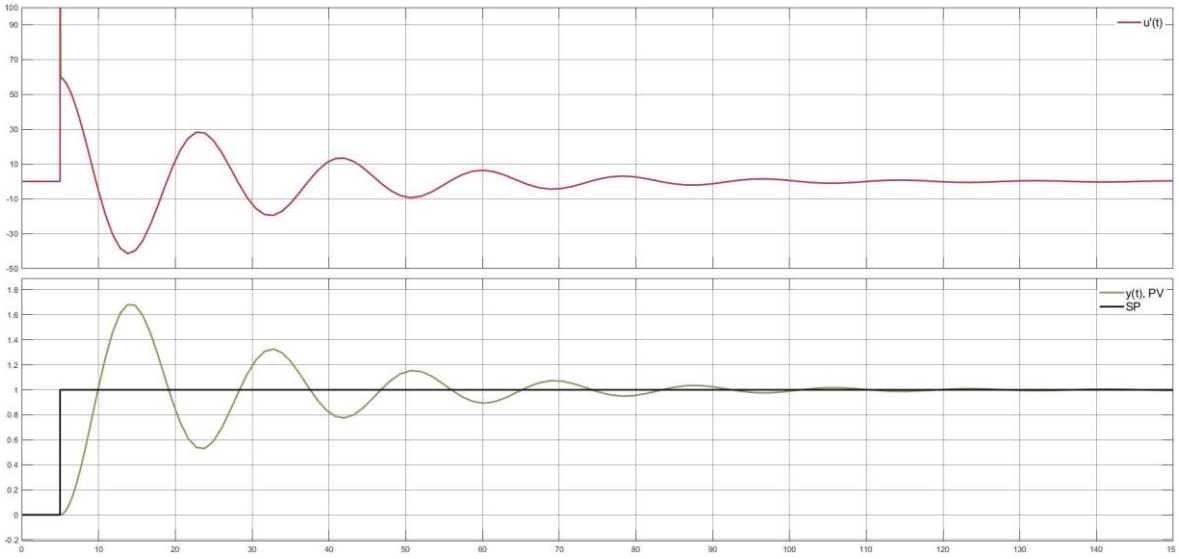

Figure 69: Tuned closed loop yaw angle response. Second iteration. Controller's saturated output u'(t) (top) and response PV (bottom)

Figure 69 displays the simulated response of the closed loop utilizing the same block diagram, as in Figure 66. Based on the hypothesis, in reality, the overshoot would be high, but the settling time would be far lower. Both the intended characteristics were accomplished during simulation: a) control actions far exceeding the dead zone (±20%) and b) a relatively low derivative term.

Before moving onto the test phase (section 7.3), the other two angles' controllers will be tuned.

7.1.3 PITCH

The parametric model for the pitch angle was written in Equation 8. It did not have a negligible pole. Rewriting it in standardized form (Equation 26) would render:

$$s_{1,2} = -0.1094 \pm 1.0368i$$

$$G(s) = \frac{k}{s^2 + a \cdot s + b} = \frac{0{,}01614}{s^2 + 0{,}2187 \cdot s + 1{,}087}$$

Equation 26: Continuous model for the pitch angle and its poles

A suitable controller structure for a second order model with no integrator (unlike yaw) is a PI, since it nullifies the static error ($e_{ss} = 0$) present with a proportional (P) controller [62]. The derivative term was avoided because of:

a) *Stability*: the term has the potential to enhance high-frequency noise and disturbances, which could cause the system to become unstable.
b) *Robustness*: the term renders PIDs more sensitive to parameter variations and model uncertainties, especially the derivative term.
c) *Data quality*: the quality of the signals was not reliable.
d) *Simplicity*: a PI controller is easier to tune than a PID.

The same logic was applied to the roll angle transfer function (explained in the next subsection). Note that the loop was always closed within the quadrotor (e.g., it made use of the IMU's readings) and not inside Simulink. That meant that the gyroscope's readings were not available for the control loop.

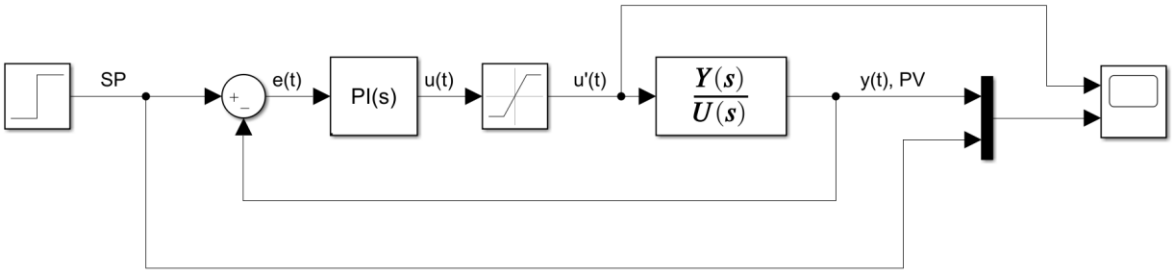

Figure 70: Closed loop for pitch and roll angles

As previously explained, the closed loop system for this case is found on Equation 18. Therefore, the pole placement design can be carried out by solving Equation 19, written as follows:

$$(s^2 + 2 \cdot \xi_d \cdot \omega_{nd} \cdot s + \omega_{nd}^2)(s + p) = s^3 + a \cdot s^2 + \left(b + k \cdot K_p\right) \cdot s + k \cdot K_i$$

$$s^2 \rightarrow \qquad p + 2 \cdot \xi_d \cdot \omega_{nd} = a$$

$$s^1 \rightarrow \qquad \omega_{nd}^2 + 2 \cdot \xi_d \cdot \omega_{nd} \cdot p = b + k \cdot K_p$$

$$s^0 \rightarrow \qquad \omega_{nd}^2 \cdot p = k \cdot K_i$$

Setting the following reasonable specifications (desired dynamics):

$$M_p = 2\% \qquad T_s = 40s$$

$$\xi_d = \frac{-ln\left(\frac{M_p}{100}\right)}{\sqrt{\pi^2 + ln^2\left(\frac{M_p}{100}\right)}} = 0{,}78 \qquad \omega_{nd} = \frac{4}{T_s \xi_d} = 0{,}13$$

The closed loop plant would not comply with any of the two desired values. The solution to the pole placement equation (Equation 19) does have negative poles, as it should have. However, the zero is positive. Therefore, the system does not comply with the transient-response specifications (Figure 71). This is also clearly observed with the negative term of the proportional gain K_p.

$$K_p = -66{,}1 \qquad K_i = 0{,}0191$$

$$s_{1,2} = -0{,}10 \pm j \cdot 0{,}0803 \qquad s_3 = -0{,}0187 \qquad z_1 = 2{,}8847 \cdot 10^{-4}$$

$$\frac{PV}{SP} = \frac{-1{,}067 \cdot s + 0{,}00031}{s^3 + 0{,}2187 \cdot s^2 + 0{,}02019 \cdot s + 0{,}00031}$$

Equation 27: Closed loop for the pitch angle transfer function. Together with its gains, poles and zero

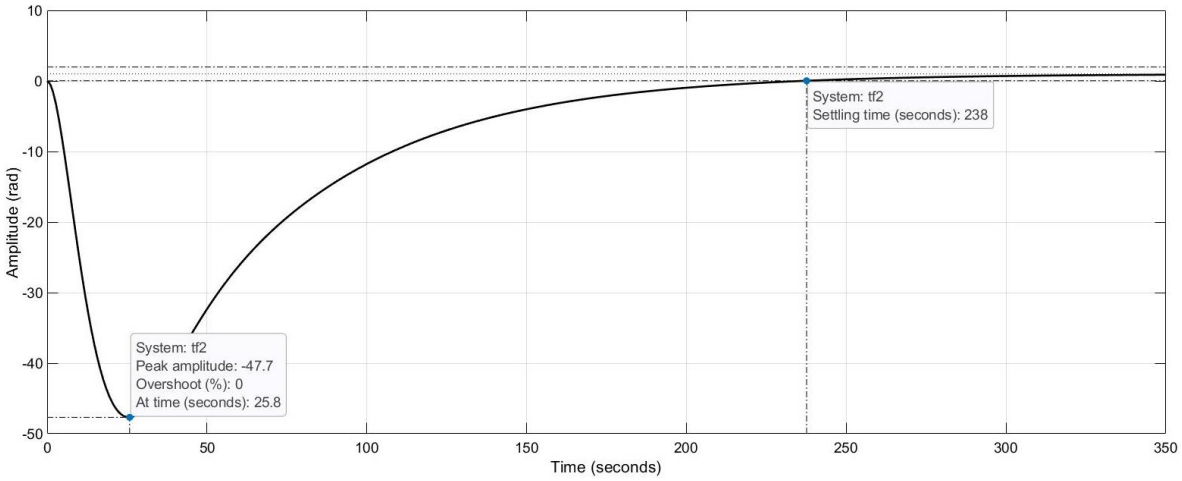

Figure 71: System response for Equation 27

Even if the specifications changed (within reasonable limits), the calculations would still produce unreasonable gain values and solutions. Nevertheless, it is still considered stable because the poles are negative. As commented in section 7.1.1, the PI controller only has two DOF and Equation 18 has four parameters (third order system with a zero). Therefore, the pole placement technique renders bad results for these types of systems. The same reasoning can be applied to the roll angle since it had the same structure as Equation 18 when the loop was closed.

To tackle this problem, an iterative algorithm was written for the parametrized models. It also worked with specifications such as the overshoot M_p and the settling time T_s. The

new algorithm rendered a vast number of possible combinations for K_p and K_i for the desired dynamics of the system in a closed loop.

To better analyse the combinations, the ITAE performance index was introduced. Note that the index was not the only parameter analysed to choose the gains. The behaviour of the plant in closed loop was analysed, which consisted of the settling time, oscillations, overshoot, ITAE index and the gains themselves.

Although the algorithm's tuning is done separately for each model, they are intended to be tested simultaneously as part of the final validation phase. Additionally, the drone's code had an anti-windup control technique built into it, but it is preferable to avoid reaching limits where its use is necessary. This is one of the advantages of the applied methodology. Additionally, the controllers' main purpose is to validate the models. As explained in the scope of the dissertation (section 1.2), they are not intended for field usage.

The desired dynamics were set as follows:

$$M_p \leq 2\% \qquad\qquad T_s \leq 40s$$

The iterative algorithm's step size was set to 1 to reduce both the computation elapsed time and the size of the solution matrix. The exact code to reproduce this outcome can be found in Appendix D, item IV. The algorithm found four solutions and the best was chosen according to the previous explanations. The settling time and overshoot of the selected loop (Equation 28) can be seen in Figure 72.

$$K_p = 3 \qquad K_i = 8 \qquad ITAE = 79{,}34$$

$$s_{1,2} = -0{,}052 \pm j \cdot 1{,}06 \qquad s_3 = -0{,}11 \qquad z_1 = -2{,}67$$

$$\frac{PV}{SP} = \frac{0{,}04842 \cdot s + 0{,}1291}{s^3 + 0{,}2187 \cdot s^2 + 1{,}135 \cdot s + 0{,}1291}$$

Equation 28: Tuned closed loop pitch angle transfer function. Together with its gains, poles and zero

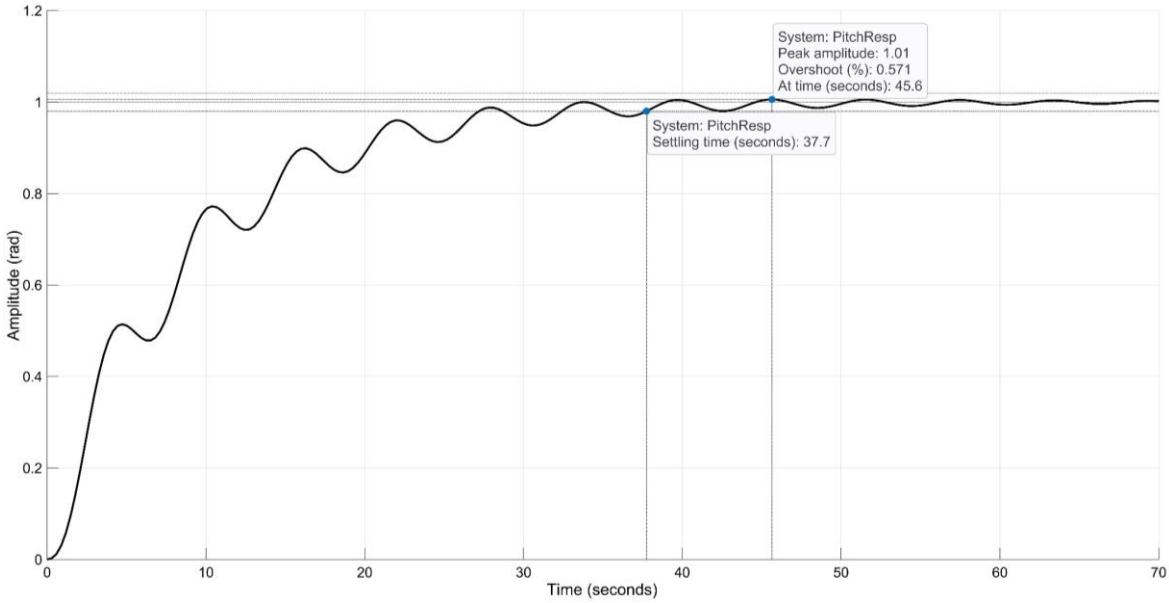

Figure 72: Pitch response for Equation 28

The system was simulated using Simulink based on the same block diagram seen in Figure 70. The results are seen in Figure 73. The tuned gains managed to produce a progressive increase in the output of the controller. For the simulation phase, the system's integrator did not windup.

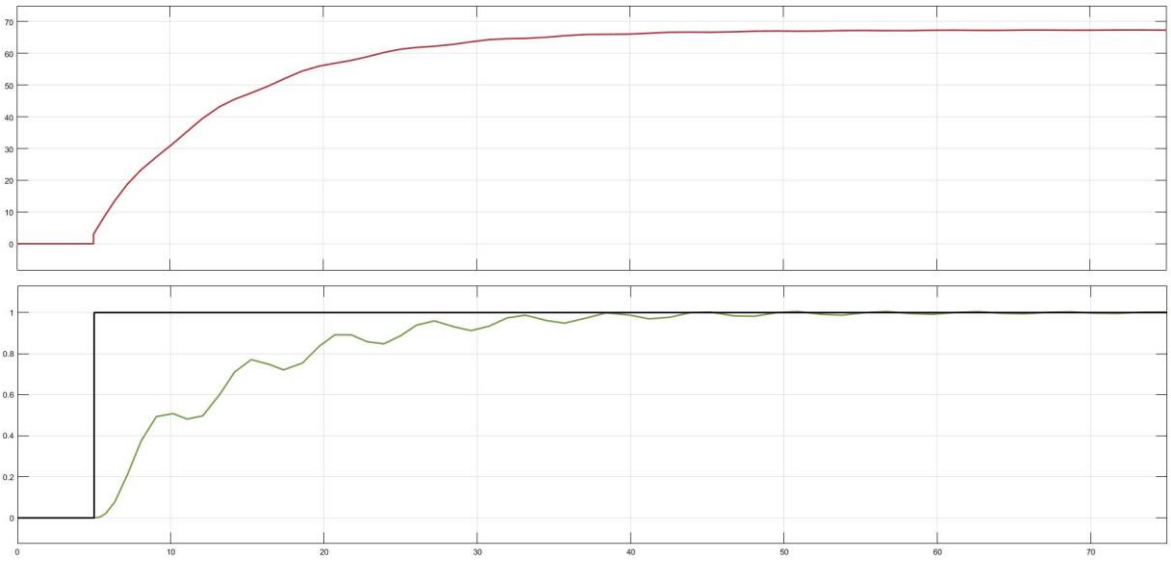

Figure 73: Tuned closed loop pitch angle response.
Controller's saturated output u'(t) (top) and response PV (bottom)

7.1.4 ROLL

The parametrized model for this angle in continuous time (Equation 10) expressed in the standardized form of a second order system was:

$$G(s) = \frac{k}{s^2 + a \cdot s + b} = \frac{0{,}725}{s^2 + 2{,}581 \cdot s + 34{,}38}$$

The model as it was, did not have any negligible poles. The chosen controller structure is a proportional-integral (PI). The same reasoning applies for both the pitch and the roll angles in terms of the control algorithm, the utilization of the iterative algorithm of Appendix D, item IV. Therefore, the control loop was stablished as in Figure 70.

The plate (roll angle) was objectively faster than the rest, as previously explained in the last chapter. In open loop, the angle has a settling time of ~3 seconds but an overshoot of ~49%. It was evident from its transfer function that it had an oscillatory nature because the plant was underdamped (poles had large imaginary parts with respect to its real parts). This occurs because the gyroscope did not add a large inertia to it, as it did happen with the pitch angle. To impose a critically damped dynamics onto the system would slow it down considerably. In order to maintain its fast dynamics but reduce its overshoot the following parameters were imposed:

$$M_p \leq 3\% \qquad\qquad T_s \leq 5s$$

Setting the algorithm with the same step size for the gains as before (equal to 1), the algorithm found 1677 combinations that complied with the specifications. The first choice was to minimize the ITAE performance index. The resulting controller, the closed loop transfer function, its poles and zero were:

$$K_p = 36 \qquad K_i = 84 \qquad ITAE = 0{,}59$$

$$s_{1,2} = -0{,}77 \pm j \cdot 7{,}63 \qquad s_3 = -1{,}03 \qquad z_1 = -2{,}33$$

$$\frac{PV}{SP} = \frac{26{,}1 \cdot s + 60{,}9}{s^3 + 2{,}581 \cdot s^2 + 60{,}48 \cdot s + 60{,}9}$$

Equation 29: Tuned closed loop roll angle model. First iteration. Together with its gains, poles and zero

As it can be observed in the tuned loop, the complex poles have a strong imaginary part, which produces strong, unwanted oscillations. A step response is observed in Figure 74.

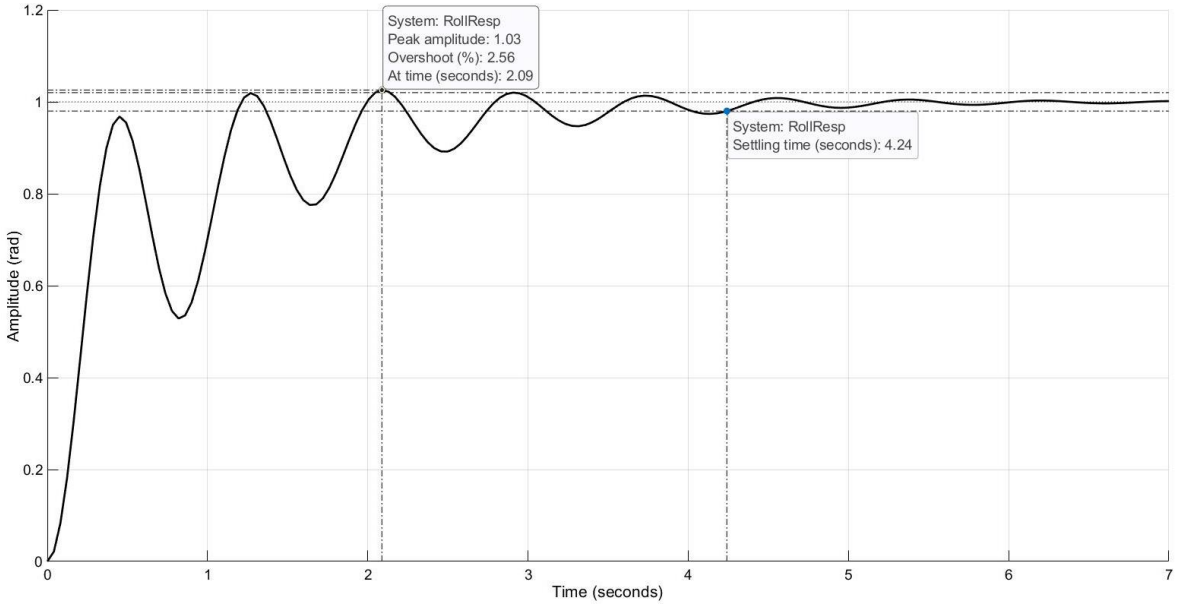

Figure 74: Roll angle response for Equation 29

The system was simulated utilizing Simulink based on the same block diagram seen in Figure 70. The results are observed in Figure 75. Not only the transient response was oscillatory, but the controller's saturated output $u'(t)$ also had a similar behaviour.

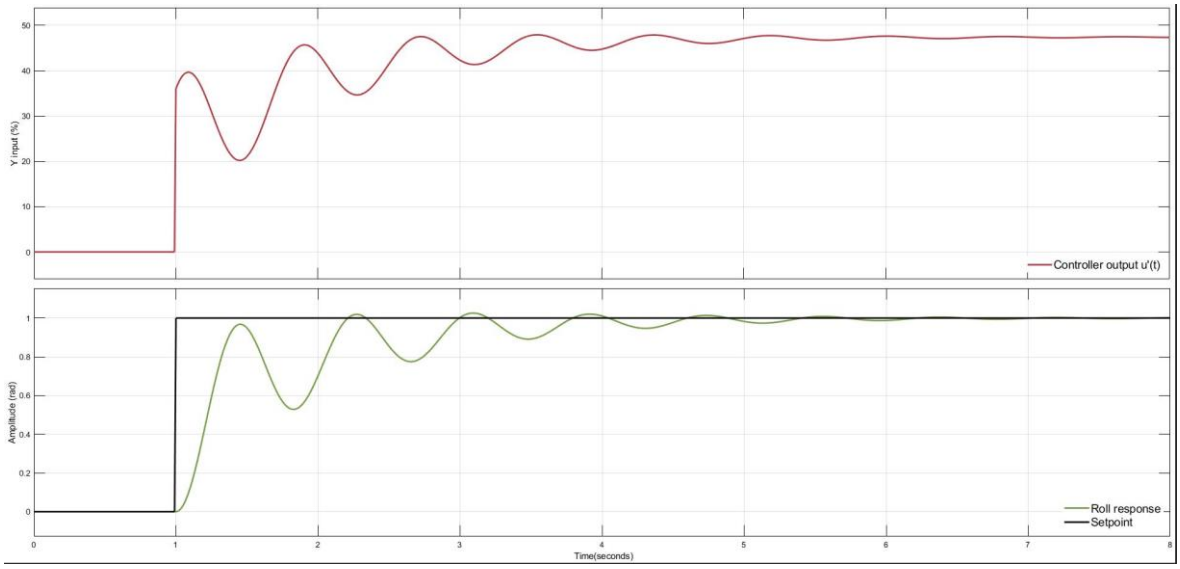

Figure 75: Tuned closed loop roll angle response. First iteration. Controller's saturated output u'(t) (top) and response PV (bottom)

To minimize the oscillations in both signals, new gains were chosen from the calculated combinations. The integrative term was then minimized, instead of the ITAE index. The resulting controller, the closed loop transfer function, its poles and zero were:

$$K_p = 3 \qquad K_i = 37 \qquad ITAE = 1{,}65$$

$$s_{1,2} = -0{,}91 \pm j \cdot 5{,}86 \qquad s_3 = -0{,}76 \qquad z_1 = -12{,}33$$

$$\frac{PV}{SP} = \frac{2{,}175 \cdot s + 26{,}82}{s^3 + 2{,}581 \cdot s^2 + 36.56 \cdot s + 26{,}82}$$

Equation 30: Tuned closed loop roll angle model. Second iteration. Together with its gains, poles and zero.

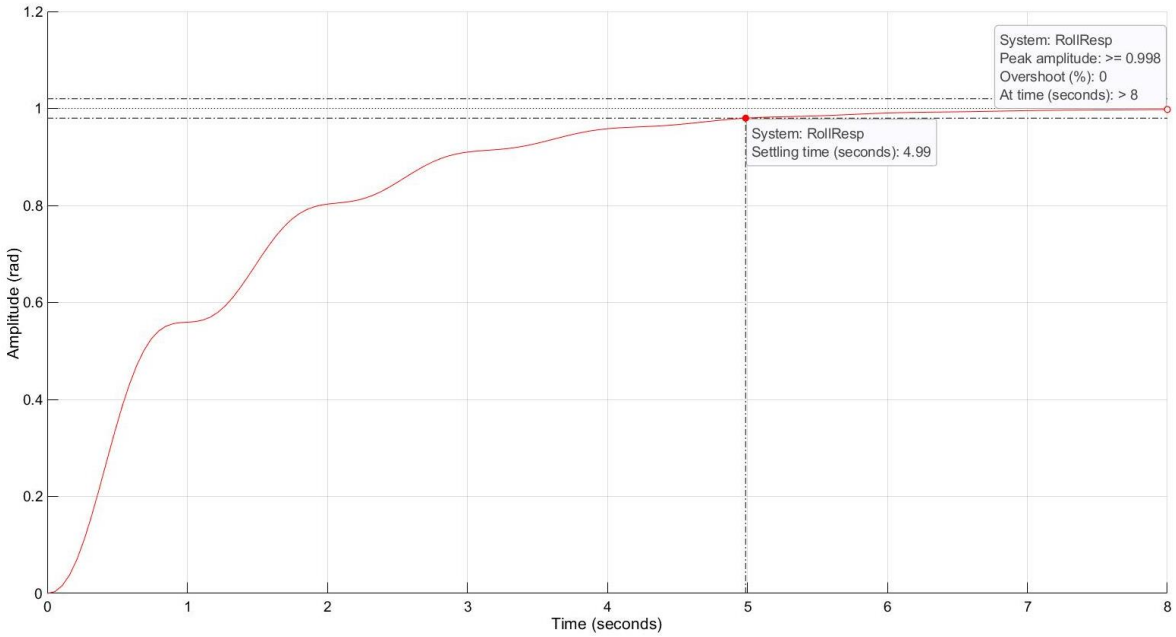

Figure 76: Roll angle response for Equation 30

The system was simulated utilizing Simulink based on the same block diagram seen in Figure 70. The results are observed in Figure 77. The oscillatory nature was heavily reduced, and the plant presented no overshoot, in comparison to the first iteration.

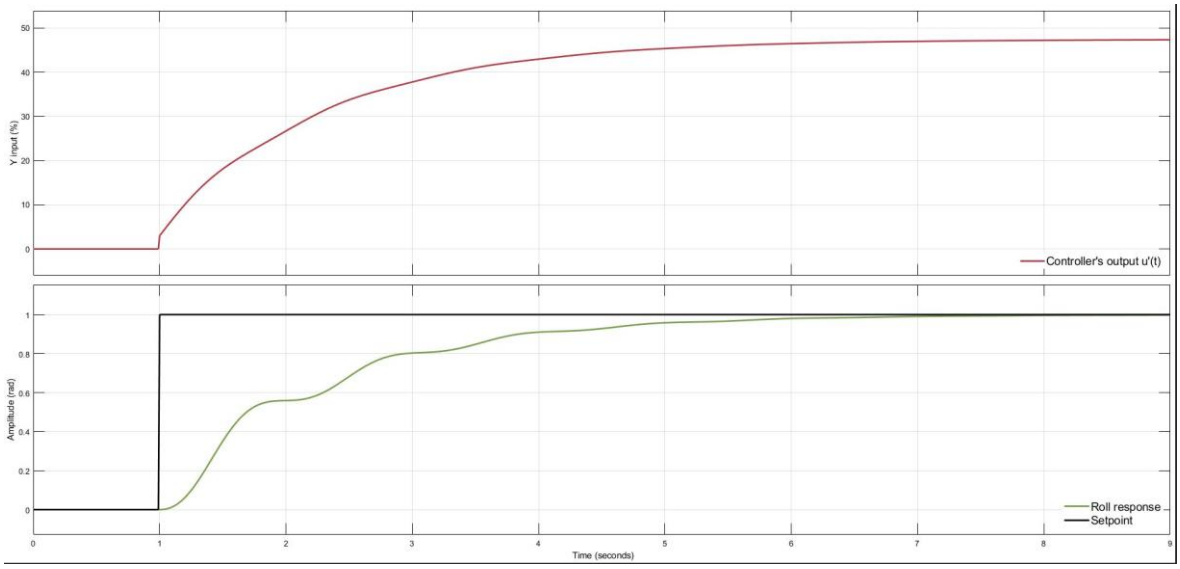

Figure 77: Tuned closed loop roll angle response. Second iteration. Controller's saturated output u'(t) (top) and response PV (bottom)

The experimentation phase in section 7.3 will present real responses of the drone-gyroscope duo for smaller inputs for all the angles, but the graphs should present the same dynamics.

7.2 CONTROLLER'S DIGITAL IMPLEMENTATION

After the calculations, it was necessary to develop a new interface for the test of the flight controllers based on the current code of the quadrotor. To accomplished this, the drone's control algorithm scripts were analysed.

According to the AscTec Hummingbird manual written in the university, there were two main coded flight controllers, which are based on section 3.6 of [65] ("Digital implementation"). These were: a) altitude controller, b) attitude controller, c) combination of them, and d) others. The possible structures for the first two are presented in Table 4. Since the dissertation's scope only included the development and implementation of attitude control, the altitude controller will not be considered in the current chapter.

Type (cfg)	PID configuration
0	Proportional controller (P).
1	Proportional and Integral controller (PI).

2	Proportional, Integral and Derivative controller (PID) with derivative calc.
3	Proportional + Derivative controller (P+D) with derivative calc.
4	Proportional and Integrator + Derivative controller (PI+D) with derivative calc.
5	Proportional + Derivative controller (P+D) with direct derivative output.
6	Proportional and Integral + Derivative (PI+D) controller with direct derivative output.

Table 4: Controller structures available in the quadrotor

Adapted from source: AscTec Hummingbird documentation written at UPC

The controllers' parameters were stored in a '.yaml' extension file called 'attitude_pid_data'. It can be found on the following folder inside the drone: 'ROS_Asctec/src/path_following_control/config'. This depends on the chosen structure or type. A section of the code can be seen as follows:

```
# PITCH pid_yaw:
    gains: {P: 0.95, I: 0.32, D: 0.10}
    param: {Ts: 0.01, N: 1000, b: 1.0}
    sat: {lower: -1.0, upper: 1.0}
    cfg: 6 # Controller: PI+D
# ROS PARAMETERS
    ros_attitude_cfg:
    freq: 100
```

The structure contained the following information:

- *gains*: controller gains Kp, Ki, and Kd
- *param*: sample time (Ts), filter coefficient to calculate the derivative (N) and the constant to reshape the input signal
- *sat*: saturation limits for the controller
- *cfg*: chosen structure for the control algorithm
- *freq*: refresh frequency for the ROS node for the controller

Note that the saturation levels are between -1 and 1. In practical terms, that meant that the output of the controller had a gain of 100. For example, if the calculations rendered a value of 5 for the proportional gain of a controller, the correct value for the code was 0,05. This fact was crucial when dealing with the algorithm. Note that the refresh frequency of the ROS node coincided intentionally with the one used for the experimentation phase in chapters 4, 5 and 6 ($freq = 100Hz \leftrightarrow T_s = 0{,}01s$).

After the analysis of the documentation, the interface could be developed. This can be observed in Figure 78. The model was divided into 5 segments:

a) *Attitude command*: delivered the setpoints to the drone. Units are in radians.

b) *Thrust command*: delivered the thrust power, which is set to ~40%, as in section 6.1.

c) *Controller output*: delivered the output of the PID to the processors, which are then internally converted into voltages applied to the motors. Units displayed in percentage.

d) *Euler angles output:* delivered readings of the Euler angles as converted by the drone. Units are set in radians.

e) *Gyroscope output:* delivered readings of Euler angles. Units are set in radians.

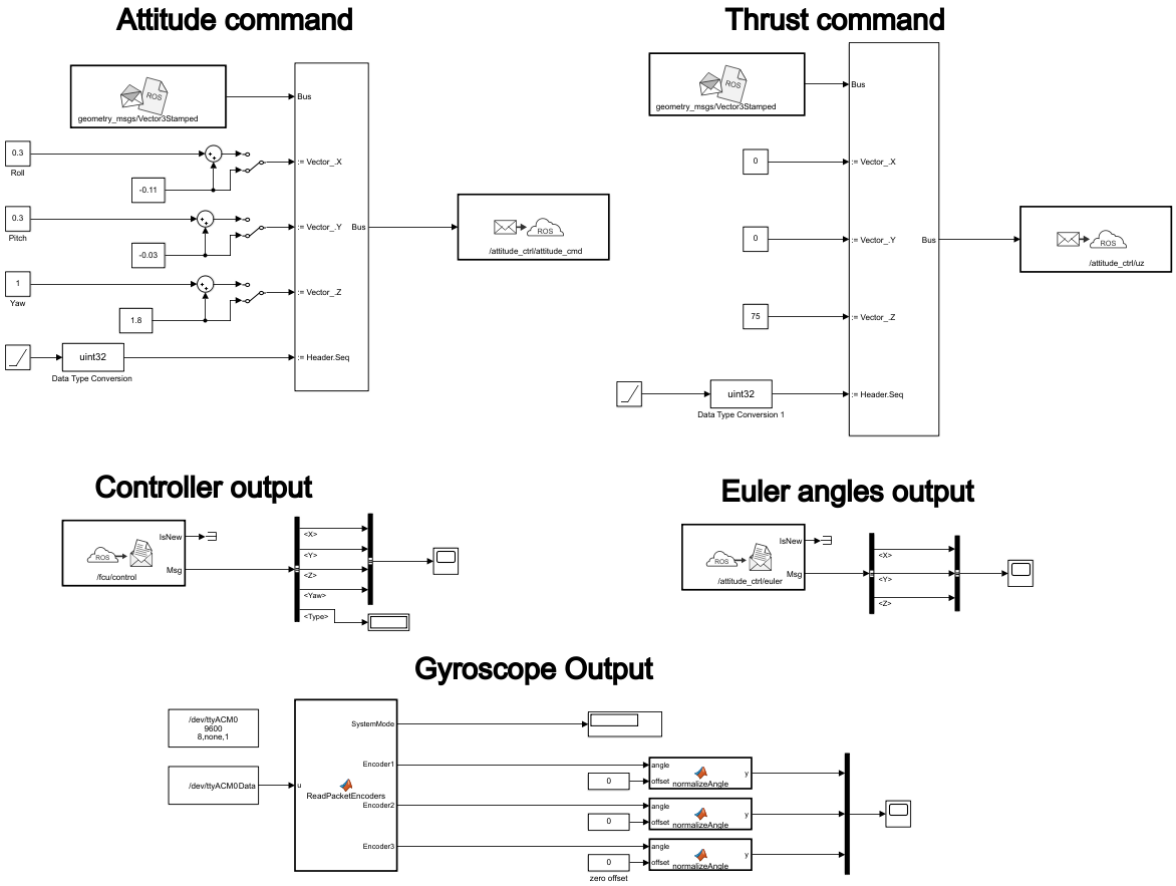

Figure 78: Control model interface 'SetPointsPID.slx'

A step-by-step guide to the execution of the ROS node for the chosen algorithm is presented in Appendix C, item VIII. This guide was used to test the algorithms in the next

section. It is recommended that only one Euler angle is controlled at a time. This is to prevent damage to the drone and the user. To accomplish this, set the PID gains for the uncontrolled Euler angles to zero. It will open their control loop.

7.3 RESULTS OF THE IMPLEMENTATION

After the analysis of the quadrotor's documentation and the development of the new controller test interface, the final validation could be completed. The experiments will be presented chronologically; that is: pitch, roll and yaw. The last subsection will present the simultaneous implementation of all the calculated controllers to analyse how they respond. Note that it was not possible to keep all the angles at a fixed position before the input signal stepped in, given the freedom the quadrotor had. The gimbals and plate were completely free during the controllers' tests, as they were during the identification and initial validation phases.

All the tests were carried out with the thrust variable (Z input, Figure 33) at ~40% to replicate the drone's (virtual) hovering condition. All the controllers' outputs (and therefore, the system's input) are shown in their rightful form. That meant that the graphs are not trended (0%→100). For further details, please refer to the paragraph below Figure 12. The controllers were not fined tuned, because it is not within the scope. Overall, the tests and their results are considered the final validation phase of the identification process.

To improve readability, Table 5 is presented before each result, to compare the simulated and real responses of the system. The real values are in the expected range, except yaw's settling time T_s, which diverges significantly as hypothesized.

Angle	Pitch		Roll		Yaw	
Type	Sim	Real	Sim	Real	Sim	Real
Kp	3	3	3	3	60	60
Ki	8	8	37	37	-	-
Kd	-	-	-	-	15	15
Mp (%)	0,571	0,856	0	2	69	60
Ts (sec)	37,7	26,79	5	8,1	94	19,7

Table 5: Comparison between simulated and real responses

7.3.1 PITCH LOOP

To summarize the road up to this point, the model found for the pitch angle had a ~66,6% fit value for both the identification and validation experiments (Figure 49 and Figure 51). The discrete model was then approximated in continuous time through a transfer function with the same fit values. Thereafter, a PI controller was tuned for it and the closed loop was tested with a step input signal and the resulting settling time and overshoot were 37,7 seconds and 0,6%, respectively (Figure 72).

To avoid stressing the quadrotor, the setpoint was established at 0,6 radians (difference between the steps shown in the figure). The pitch angle's setpoint and response are observed in Figure 79. Comparing it with the simulated (Figure 73), both graphs have a similar shape. The drone does respond faster than the transfer function and settles in around 28 seconds approximately.

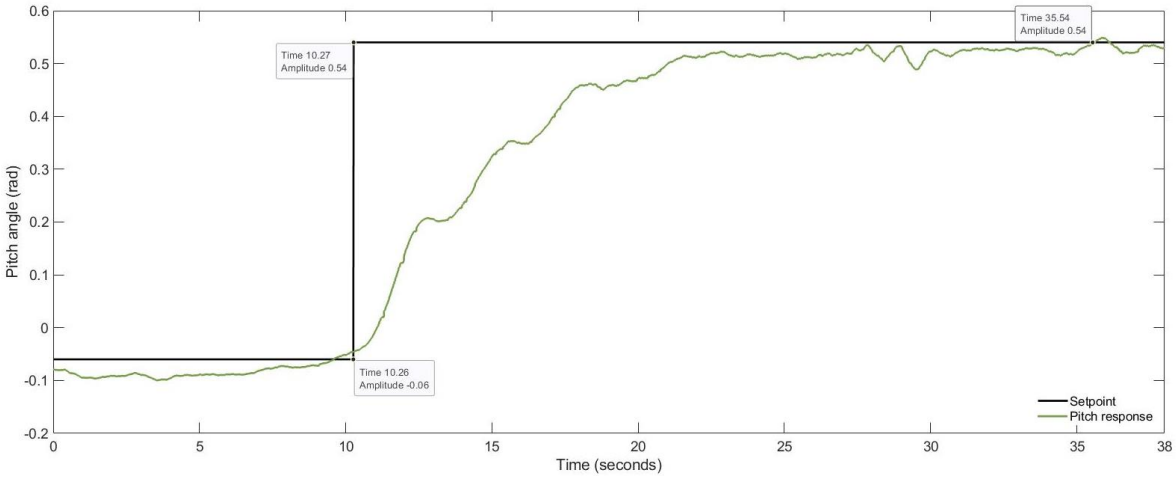

Figure 79: Pitch setpoint and response.
Kp = 3, Ki = 8

The controller's output can be observed in Figure 80. Comparing it with the modelled system in Figure 73, the controller's output had the expected behaviour, and the reasoning above applies again. The final validation phase for this angle concluded successfully for the models found on subsection 6.1.1 under the specified conditions of the experiments and system identification process overall.

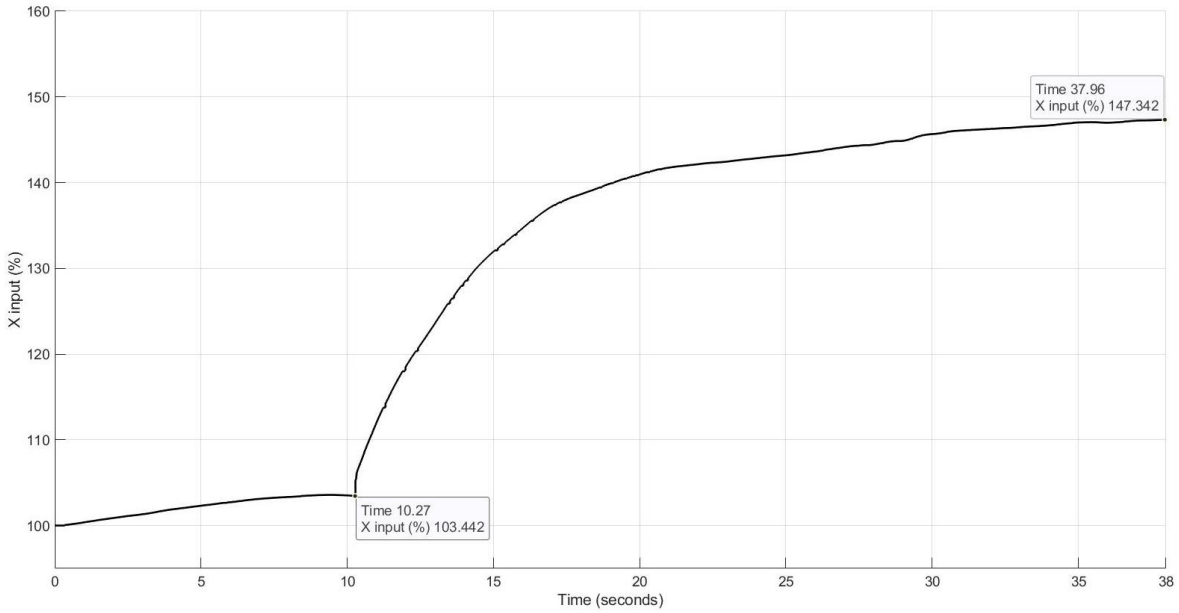

Figure 80: Pitch control.
Kp = 3, Ki = 8

The next subsection deals with the results of the implementation of the roll angle controller.

7.3.2 ROLL LOOP

Moving onto the roll angle, the model had a ~61% and ~64% fit values for the identification and validation experiments respectively (Figure 54 and Figure 55). The discrete model was then approximated in continuous time through a transfer function with the same fit values. Thereafter, a PI controller was tuned for it and the closed loop was tested with a step input signal and the resulting settling time was ~5 seconds, with no overshoot (Figure 76).

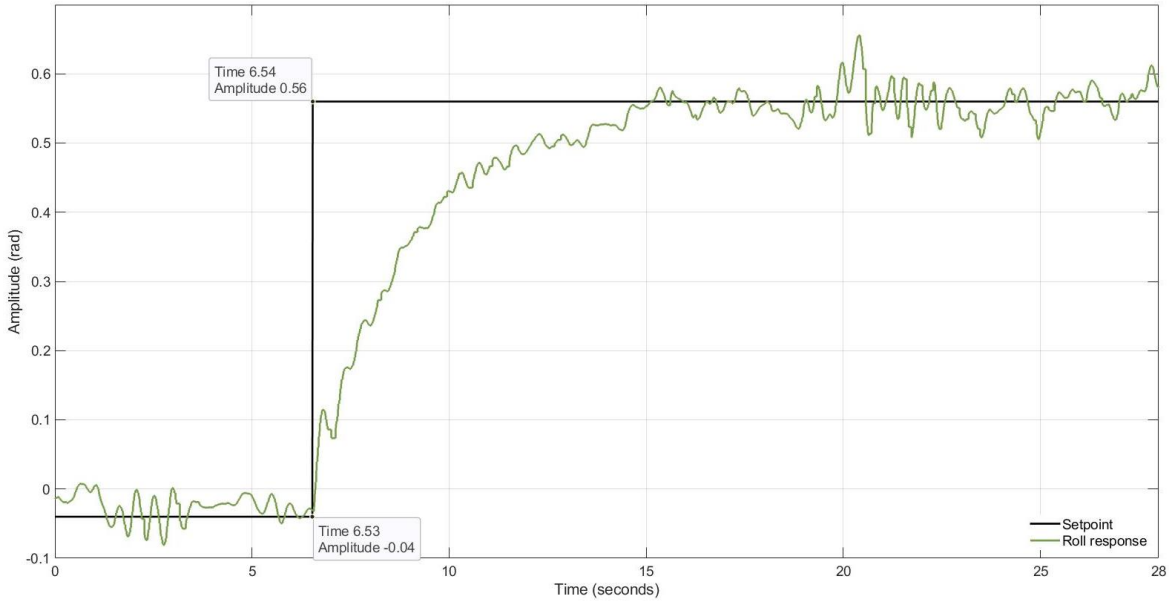

Figure 81: Roll setpoint and response.
Kp = 3, Ki = 37

According to Figure 81, the real response of the system did take considerably longer to settle than the simulated output, rendering a ~3,5 seconds difference between them. The real and simulated responses did maintain a strong similarity between them and did not overshoot, as specified to the iterative tuning algorithm. This could be due to the fit values. A proportional effect was observed by comparing the controller's real output and the simulated signal (Figure 82 and Figure 77). As the system responded faster, its controller's output was proportionally faster. The final validation phase for this angle concluded successfully for the models found on subsection 6.2.2 under the specified conditions of the experiments and system identification process overall.

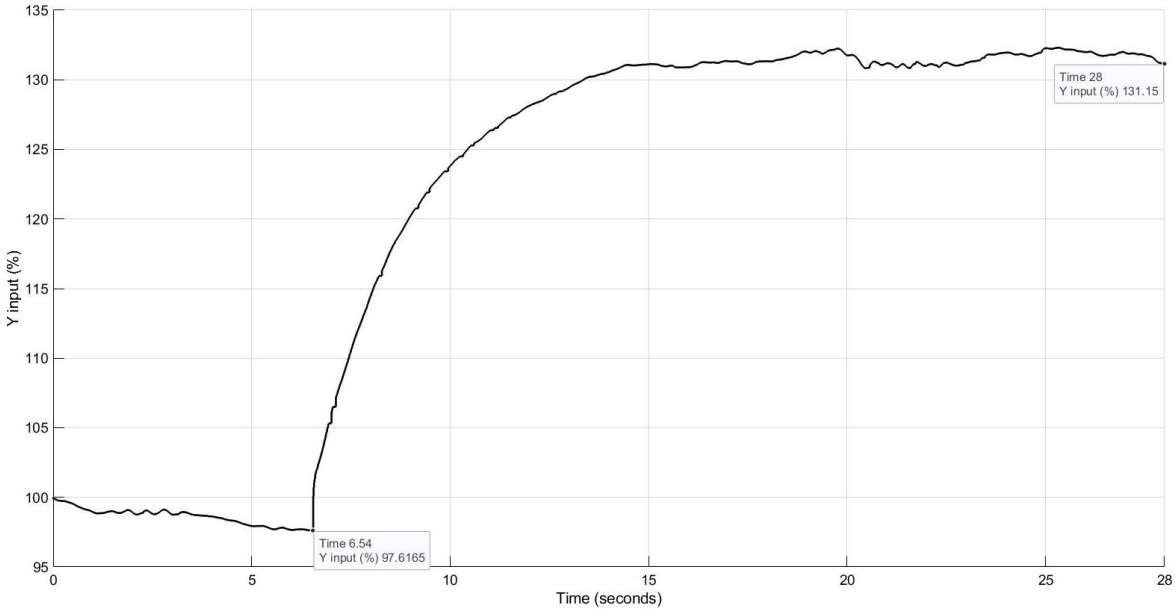

Figure 82: Roll controller output.
Kp = 3, Ki = 37

An additional conclusion could be added to the case. As observed in the identification and validation experiments utilized for this angle (Figure 54 and Figure 55), the controlled signal also had a strong influence of noise. This effect could not be damped even by closing the loop with a control algorithm. This further proves the existence of unmodelled noise, which could also be catalogued as uncontrollable. As they are set, the controller's gains could be fined-tuned to get the desired specifications stablished in section 7.1.4.

7.3.3 YAW LOOP

The final angle was yaw. Backtracking through the process, no discrete parametrized model was found for yaw, because of the SITB algorithm's limitations with the natural integrator. A continuous time model (a transfer function) was found, which had a ~94% MSE index value for both the identification and validation experiments respectively (paragraph below Figure 58 and Figure 60). Thereafter, a PD controller was selected and tuned, and the closed loop was tested with a step input signal. The resulting settling time and overshoot were ~94 seconds and 69% overshoot (Figure 68).

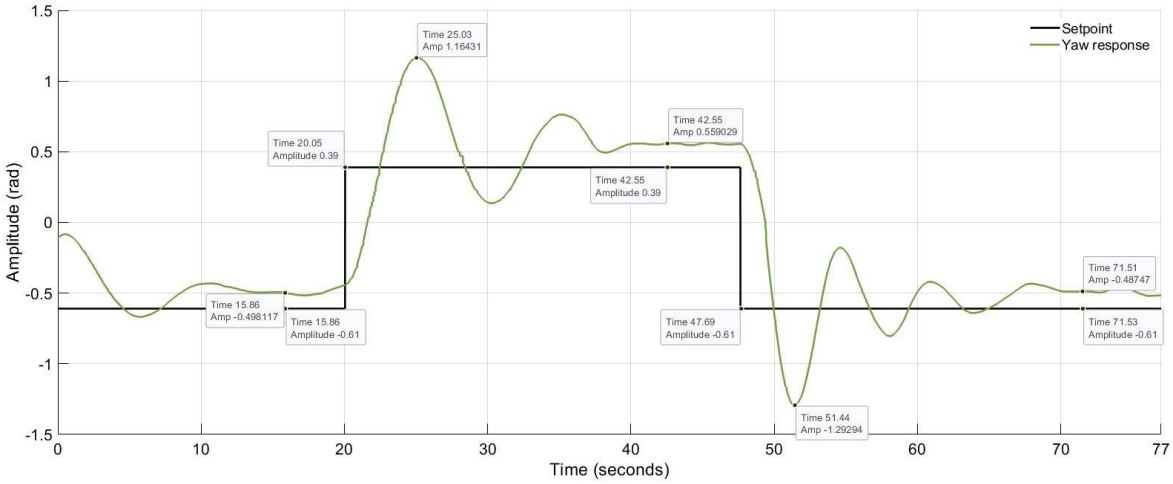

Figure 83: Yaw setpoint and response.
Kp = 60, Ki = 15

Analysing both the yaw response and the controller's output (Figure 83 and Figure 84), it was observed that the angle was not able to reach its setpoint, even though the controller still output a value different than 100 (0%). Apparently, the setpoint and the response had an offset between -0,11 and -0,17 radians. Furthermore, the controller's values at which the continuous offset happened were at the range of ~90 (-10%). Considering again the fact that this angle had a dead zone, the control offset is unavoidable as the loop is set (Figure 70).

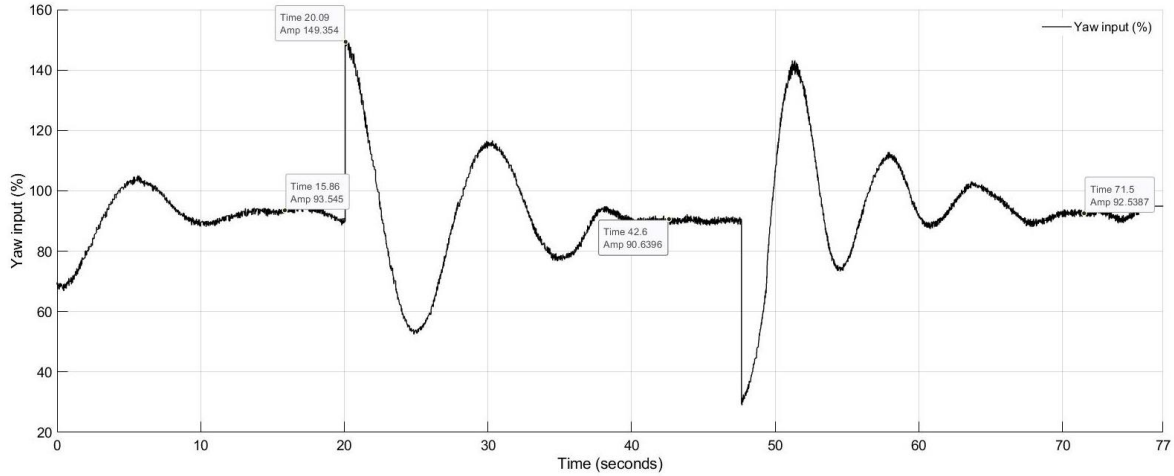

Figure 84: Yaw controller output.
Kp = 60, Ki = 15

7.3.4 SIMULTANEOUS CONTROL

Backtracking to the firsts paragraphs of chapter 6, the quadrotor was treated as a decoupled MIMO system equivalent to four SISO system during the identification process (identification, initial validation, and final validation phases). In reality, quadrotor-gyroscope duo was still coupled in a relative low degree. The pitch-yaw and roll-yaw pair were still coupled. This information was omitted to simplify the model identification. Further dissertations could study the system in this direction.

The effects of the simplifications were observed clearly through the simultaneous implementation of all the drone's controllers, although a further analysis will be provided at the end of the chapter. Figure 85, Figure 86 and Figure 87 show the pitch, roll and yaw setpoints and responses. The pitch angle showed an oscillatory nature, even though it was tuned to have relatively little.

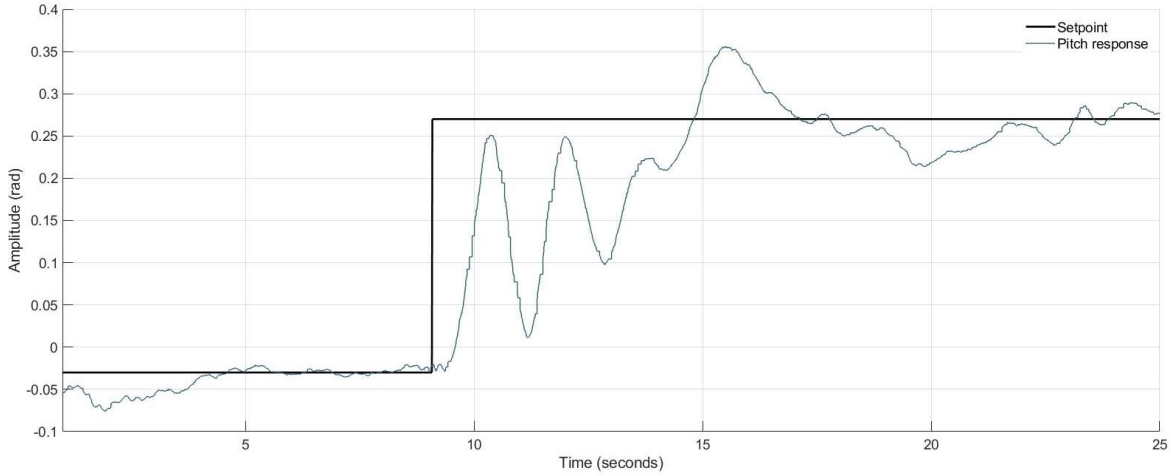

Figure 85: Pitch response

Additionally, it was expected that the noise component of the roll angle was intensified due to the conditions it was working on (simultaneity), together with its nature (Figure 40). Note that the noise and the pitch's oscillatory nature diminished their amplitude progressively when the yaw's controller reduced its output (Figure 88). This spoke about the influence of the yaw movement to the other angles.

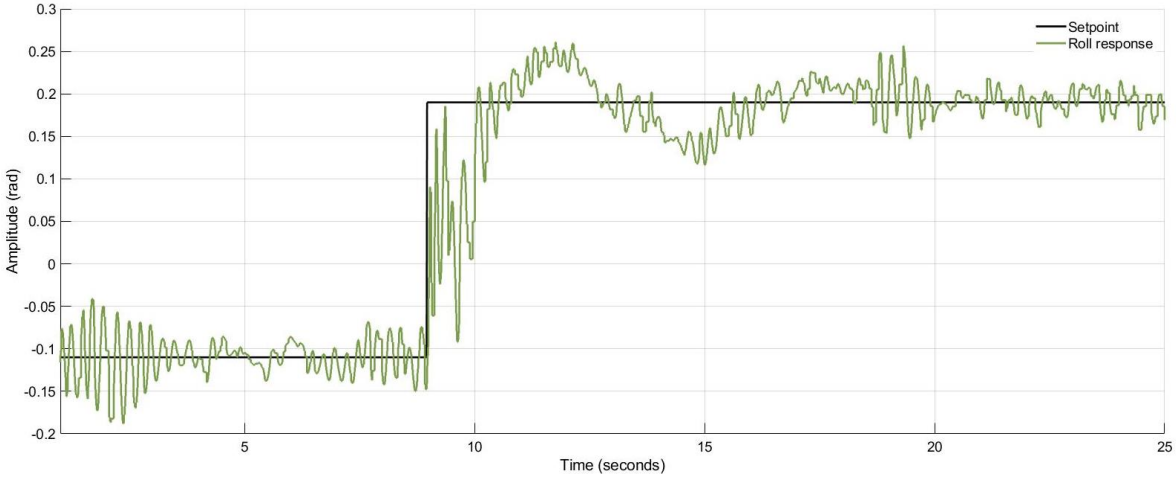

Figure 86: Roll response

Because the yaw reached its setpoint around the 25 second mark, it might be tempting to say that yaw's control was successful in reaching the desired values within specifications. The hypothesis then emerged that the other controllers were contributing to the movement (not control) of this last angle. This was due to the parity of pitch-yaw and roll-yaw variables, which were coupled to a certain degree. If the experiment could have continued further, more information might have been revealed. Unfortunately, this was fairly difficult, since the actuation on all DOFs at the same time drains the battery of the drone incredibly fast.

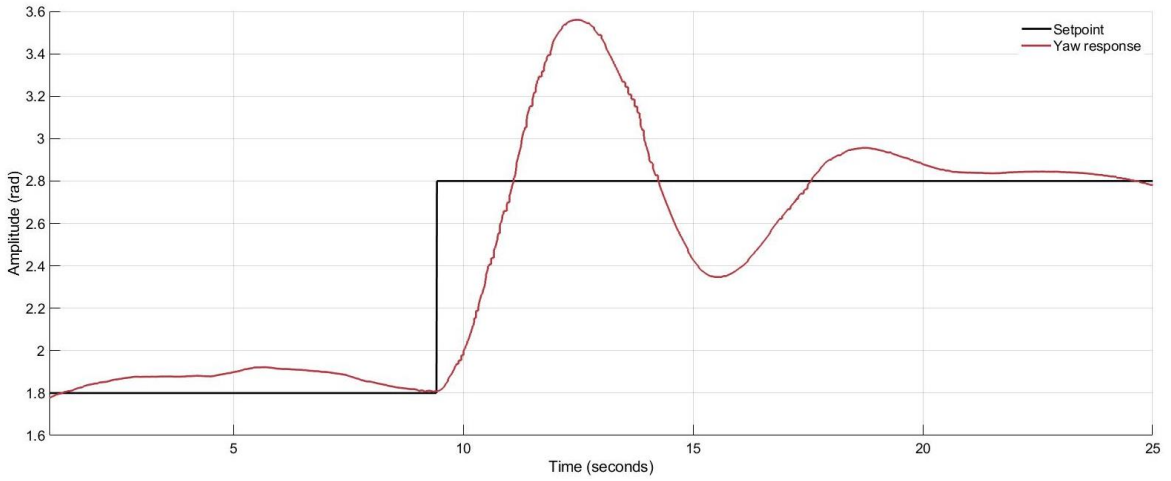

Figure 87: Yaw response

All the controllers manage to reach the stablished setpoints, but only the yaw angle managed to do it within the desired specifications (if the offset is neglected). Unfortunately, the identification process as stablished in the methodology (chapter 3) proved to be unsuccessful when the simultaneity was present.

Some final comments on the simultaneous implementation of the algorithms. There might have been other physical phenomena present during the final experiment of the dissertation. Equation 1 reduced the drone-gyroscope system to an equation explained through Newton's 2nd law applied to rotation. This was done because only one DOF was studied, while the other were left static. In reality, a gyroscope is objectively more complex than this simplification. Unfortunately, the study of this phenomena is not within the scope of the dissertation but might provide further venues of study.

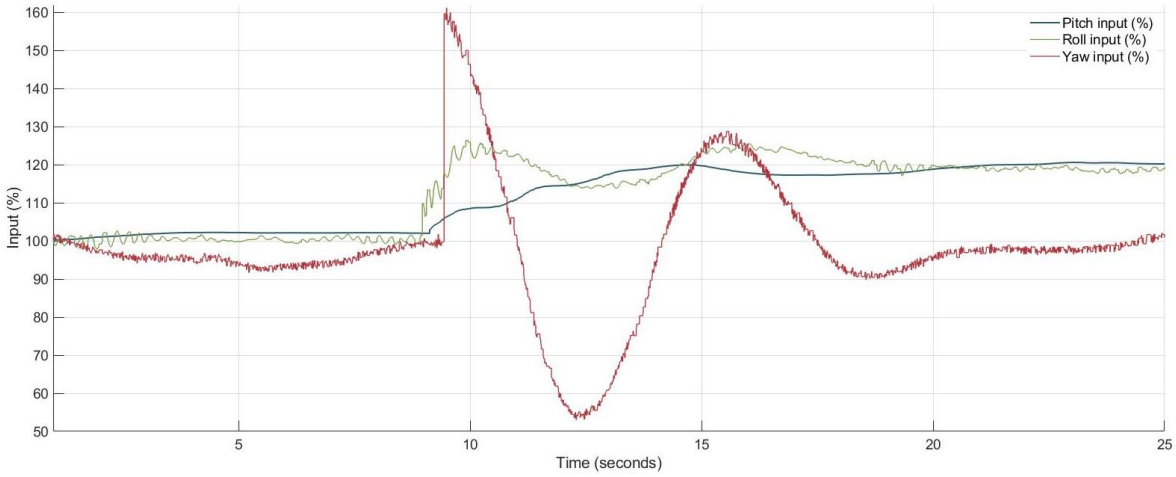

Figure 88: Controllers' output

8. CONCLUSIONS AND FURTHER VENUES OF STUDY

This chapter is dedicated to the conclusion and results of the dissertation, its impact on the university and its laboratory, and further research paths to be considered for the additional development of tools and knowledge within the university.

8.1 CONCLUSIONS

The dissertation had numerous objectives, which were all accomplished. Its most important one was the development and implementation of a drone-independent methodology on the system identification of a quadrotor mounted on a gyroscopic test bench, with the necessary development of frameworks and interfaces for the current and future studies.

After the literature review done on the subject, it was concluded that interest in the development and utilization of test benches (gyroscopic or of other type) for UAV research spiked in recent years alongside the interest in UAV themselves. Nevertheless, no record was found on the utilization of the system identification techniques used for this dissertation. The research done by the scientific community utilises complex models based on physical relationships between the variables and unknown parameters of the system. The dissertation diverged from the norm to study whether a parametric model could be estimated without imposing a physical interpretation of the system.

Before the identification, communication tools and user-friendly interfaces were developed successfully as a versatile package solution. They consisted of scripts and models for the MathWorks' products Matlab and Simulink. They are accompanied by descriptive guides and plagued by good practices, to avoid their incorrect harnessing. In the present, it has already contributed to Universitat Politècnica de Catalunya, whose teaching staff already employ the interfaces and frameworks as a learning tool for students on a regular basis. In the future, the framework could be further exploited to test other types of identification and control techniques in research and dissertations alike.

An initial equation using Newton's 2nd law of motion (Equation 1) for rotation was successfully developed for the parameter estimation phase (initial experimentation, chapter 5). This equation explained the rotation of the quadrotor along its axes (three degrees of freedom). It helped to estimate the parameters such as the inertia and friction coefficient of the quadrotor-gyroscope pair alongside each axis. It was found that the friction coefficient had a nonlinear nature, which affected the yaw angle particularly. This also affected the experiments that were conducted afterwards. As a standalone study, this contributes to the available information on the physical system for further studies, independently of which type of work it comprises.

The collected data gathered in chapter 5, in conjunction with Equation 1 spoke about the similarities of pitch θ and roll ϕ angles responses to a given input. On the other hand, the yaw was confirmed to be different from the first two variables, because gravity did not affect it directly.

Experiments were carried out for system identification (chapter 6) and information was collected, which in turn allowed the successful identification and initial validation of parametric models and transfer functions for each Euler angle in consideration. An extensive iterative process was necessary due to the complexity of the system. It also added a layer of complexity the fact that this study was the first to investigate the drone-gyroscope pair at the university. This meant that there was a lot of trial and error involved in the process.

The identification process consisted of three phases: identification, initial validation, and final validation phases. The first two phases were conducted in chapter 6 and the last was done through the controllers' tests in chapter 7.

The pitch was the first Euler angle that was investigated. It proved to have a negligible delay in its response. Fortunately, it also proved to have no dead zone around its equilibrium point. The experiments showed a nonlinear response to inputs, but was neglected, as nonlinear identification was not within the scope of this work. The identified parametric model stood out its second order dynamics with a fit value around ~66,6% for both the identification and validation phases. The final phase of the controller's test validated the model since it proved to have a similar transient response as the one specified in the controller tuning process.

The roll ϕ was the second Euler angle to be investigated. During the identification and validation experiments, its fast dynamics compared to the pitch θ was validated, although it did share similarities in their behaviour, according to Equation 1. Additionally, it had no discernible delay or dead zone. This angle proved to be the less affected by the coupled gyroscope in terms of inertia, the reason why it was faster. Unfortunately, the readings had a high content of noise, which was partially computed to the IMU's internal noise. There's some part of the noise that the study could not explain, and its source remain unknown. This could be furthered studied in future works.

The identified parametric model for roll ϕ stood out in its second order dynamics with a fit value around ~61% and ~64% for the identification and initial validation phases respectively. The final phase of controller's test validated the model since it proved to have a similar transient response as the one specified in the controller tuning process.

The yaw angle was the last to be investigated. It had a dead zone around ±10%, which was initially discovered during the identification and validation experiments, and later confirmed during the controller's test (subsection 7.3.3). Additionally, it had no apparent delay and had infinite equilibrium points, which coincided with the nullified gravity component of Equation 1. Moreover, it had a natural integrator and was slower than the rest because of higher inertia caused by the gyroscope.

Due to the natural integrator's restrictions in the SITB method, no discrete parametrized model for yaw was identified. Instead, a continuous time model was identified during the identification and initial validation phases, which rendered an MSE index value of ~94% for both cases. The transfer function was utilized for controller tuning and the final validation phase rendered results within the specifications. An offset was observed between the setpoint and the real response of the system, which was a consequence of the dead zone.

Finally, a simultaneous implementation of the three controllers was carried out. It can be concluded that the decoupled MIMO system was more entangled than expected. For that reason, it was confirmed that the SISO methodology of system identification applied during the dissertation rendered acceptable results, although further studies with a more complete methodology is advised for the future. Nevertheless, the controllers were able to control the system, but did not comply with the transient-response specifications imposed during the controller's tuning phase.

On another note, the contributions accomplished through this study are expected to have far-reaching potential positive effects, especially within the institution. Tools and interfaces were developed as part of the work, which will help further studies and teaching of wide range of subjects taught at Universitat Politècnica de Catalunya - ESEIAAT.

The UAV industry is expected to grow and expand massively in the next few years. This work has provided an initial assessment on the study of drones and the techniques used to model them. The more the UAVs are understood, the greater the benefits that can be harnessed. These benefits can be computed to CO2 emissions, social aspects (Appendix B) and economic terms (section 2.2). If the right regulations are enforced and the appropriate parameters are employed, this technology could provide suitable solutions for both the industry and the society.

8.2 POSSIBLE FURTHER STUDIES

The following components of the study were noted as promising topics for future studies and research:

1. The source of noise for the roll angle, whose source was not entirely unravelled.
2. Application of this work methodology to other UAVs under the same conditions, using both linear and nonlinear model structures.
3. Study of the implications of external disturbances such as wind, temperature, humidity, and electromagnetic fields.
   Note: The gyroscope's manufacturer provided motors for each DOF, which can be used to simulate wind disturbances under controlled, safe conditions.
4. Application of adaptive controllers such as RST to the models identified in this dissertation.
5. Further study to reduce the effect of the yaw's dead zone.
6. Application of system identification techniques, treating the system as a complex MIMO system.
   This should render a transfer function matrix on how each input cross correlate to each output.
7. Decoupling of the quadrotor-gyroscope pair and tuning of new controllers for field usage.

REFERENCES AND BIBLIOGRAFY

[1] Eureka Dynamics, ‘Product catalog’, 2023. https://eurekadynamics.com/product-catalog/ (accessed Apr. 22, 2023).

[2] B. Morcego Seix, ‘Morcego Seix, Bernardo | FUTUR’, 2023. https://futur.upc.edu/BernardoMorcegoSeix (accessed Apr. 22, 2023).

[3] R. Bartomeu, ‘Bartomeu Rubí’, 2023. https://orcid.org/0000-0002-8822-2681 (accessed Apr. 22, 2023).

[4] R. Ruiz, ‘Ruiz Royo, Adrian | FUTUR’, 2023. https://futur.upc.edu/AdrianRuizRoyo (accessed Apr. 22, 2023).

[5] Odroid, ‘Odroid-XU4 board’, 2023. https://wiki.odroid.com/odroid-xu4/odroid-xu4 (accessed Feb. 18, 2023).

[6] PhoenixNAP, ‘How Does SSH Work?’, 2020. https://phoenixnap.com/kb/how-does-ssh-work (accessed Feb. 18, 2023).

[7] B. Rubí Perelló, ‘Guidance, navigation and control of multirotors’, *TDX (Tesis Doctorals en Xarxa)*, Dec. 2020, Accessed: Jun. 10, 2023. [Online]. Available: https://upcommons.upc.edu/handle/2117/359685

[8] A. Ruiz Royo, ‘Estudio del control de actitud de un quadrotor Hummingbird de AscTec utilizando ROS’, *UPCommons*, Nov. 2018, Accessed: Jun. 10, 2023. [Online]. Available: https://upcommons.upc.edu/handle/2117/126407

[9] M. Marín De Yzaguirre, ‘Study of a UAV with autonomous LIDAR navigation’, Jun. 2021, Accessed: Jun. 19, 2023. [Online]. Available: https://upcommons.upc.edu/handle/2117/361456

[10] L. Ljung, *Modeling of dynamic systems*. in Prentice-Hall information and system sciences series. Upper Saddle River, New Jersey: PTR Prentice Hall, 1994.

[11] L. Ljung, *System Identification and Simple Process Models*. Linköping University Electronic Press, 2003. Accessed: Jun. 17, 2023. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A316777&dswid=5488

[12] S. Jatsun, O. Emelyanova, and A. S. Martinez Leon, 'Design of an Experimental Test Bench for a UAV Type Convertiplane', *IOP Conf Ser Mater Sci Eng*, vol. 714, no. 1, p. 012009, 2020, doi: 10.1088/1757-899X/714/1/012009.

[13] K. Putsep, A. Rassolkin, and T. Vaimann, 'Conceptual Test Bench for Small Class Unmanned Autonomous Vehicle Performance Estimation', in *19th IEEE International Power Electronics and Motion Control Conference, PEMC 2021, April 25, 2021 - April 29, 2021*, in Proceedings - 2021 IEEE 19th International Power Electronics and Motion Control Conference, PEMC 2021. Gliwice, Poland: Institute of Electrical and Electronics Engineers Inc., 2021, pp. 695–698. doi: 10.1109/PEMC48073.2021.9432509.

[14] S. Alvarado, N. Certad, G. Fernández-López, and M. González, 'Construction, identification and instrumentation of a low cost quadcopter research platform.', in *LARS/SBR*, 2017, pp. 1–6.

[15] P. Daponte *et al.*, 'DronesBench: an innovative bench to test drones', *IEEE Instrum Meas Mag*, vol. 20, no. 6, pp. 8–15, 2017, doi: 10.1109/MIM.2017.8121945.

[16] M. F. Santos *et al.*, 'Experimental Validation of Quadrotors Angular Stability in a Gyroscopic Test Bench', in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC), 10-12 Oct. 2018*, in 2018 22nd International Conference on System Theory, Control and Computing (ICSTCC). Piscataway, NJ, USA: IEEE, 2018, pp. 783–788. doi: 10.1109/ICSTCC.2018.8540660.

[17] U. Veyna, S. Garcia-Nieto, R. Simarro, and J. V. Salcedo, 'Quadcopters testing platform for educational environments', *Sensors*, vol. 21, no. 12, 2021, doi: 10.3390/s21124134.

[18] S.-E.-I. Hasseni and L. Abdou, 'Decentralized PID Control by Using GA Optimization Applied to a Quadrotor', *Journal of Automation Mobile Robotics & Intelligent Systems*, vol. 12, pp. 33–44, Jun. 2018, doi: 10.14313/JAMRIS_2-2018/9.

[19] H. M. N. Elkholy, *Dynamic Modeling and Control of a Quadrotor Using Linear and Nonlinear Approaches*. in Thesis (American University in Cairo. School of Engineering Interdisciplinary Program). American University in Cairo, 2014. [Online]. Available: https://books.google.es/books?id=nZr0rQEACAAJ

[20] J. S. Chiou, H. K. Tran, and S. T. Peng, 'Attitude Control of a Single Tilt Tri-Rotor UAV System: Dynamic Modeling and Each Channel's Nonlinear Controllers Design', *Math Probl Eng*, vol. 2013, 2013, doi: 10.1155/2013/275905.

[21] K. F. Aljanaideh, D. Bhattacharjee, R. Singh, and L. Ljung, 'New features in the system identification toolbox - Rapprochements with machine learning', in *19th IFAC Symposium on System Identification, SYSID 2021, July 13, 2021 - July 16, 2021*, in IFAC-PapersOnLine, vol. 54. Padova, Italy: Elsevier B.V., 2021, pp. 369–373. doi: 10.1016/j.ifacol.2021.08.387.

[22] W. Lacerda, L. da Andrade, S. Oliveira, and S. Martins, 'SysIdentPy: A Python package for System Identification using NARMAX models', *J Open Source Softw*, vol. 5, no. 54, p. 2384, Oct. 2020, doi: 10.21105/JOSS.02384.

[23] M. Guazzone, 'dcsxx-sysid: C++ libraries for system identification', 2014. https://github.com/sguazt/dcsxx-sysid (accessed May 07, 2023).

[24] GitHub, 'Ascending Technologies Helicopters Framework', 2017. https://github.com/ethz-asl/asctec_mav_framework (accessed Feb. 10, 2023).

[25] D. Lee, H. Jin Kim, and S. Sastry, 'Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter', *Int J Control Autom Syst*, vol. 7, no. 3, pp. 419–428, 2009, doi: 10.1007/s12555-009-0311-8.

[26] A. M. E. Ramírez-Mendoza, J. R. Covarrubias-Fabela, L. A. Amezquita-Brooks, O. García-Salazar, and W. Yu, 'Fuzzy Adaptive Neurons Applied to the Identification of Parameters and Trajectory Tracking Control of a Multi-Rotor Unmanned Aerial Vehicle Based on Experimental Aerodynamic Data', *J Intell Robot Syst*, vol. 100, no. 2, pp. 647–665, 2020, doi: 10.1007/s10846-020-01198-w.

[27] Andrew Tarantola, 'Dendra System's seed-spitting drones rebuild forests from the air', 2020. https://www.engadget.com/dendra-systems-seedspitting-drones-rebuild-

forests-from-the-air-170048595.html?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAADj1wWSqIUJztEHkrc5SGJnhCMDct9Ll7Dw-91qAIgMH0eoESHIrVNOQ20NKg1ZMzf96PBOJyKA0kX9D-5qPz5N04KzyGJ7nyt9ASYPbpF529n79Xc5iOU2WmDcMk0QGBkzKmlkFdwLX1VJBVw5dymH7Du4JS96QP1oTAdlTWEny (accessed Mar. 08, 2023).

[28] L. R. do Amaral, C. Zerbato, R. G. de Freitas, M. R. B. Júnior, and I. O. P. da Silva Simões, 'UAV applications in Agriculture 4.0', *Revista Ciencia Agronomica*, vol. 51, no. 5, 2020, doi: 10.5935/1806-6690.20200091.

[29] Business Standard, 'Drones from Garuda Aerospace to be deployed in earthquake-hit Turkey | Business Standard News', 2023. https://www.business-standard.com/article/current-affairs/drones-from-garuda-aerospace-to-be-deployed-in-earthquake-hit-turkey-123020801051_1.html (accessed Mar. 08, 2023).

[30] Y. Sun and O. Ma, 'Automating Aircraft Scanning for Inspection or 3D Model Creation with a UAV and Optimal Path Planning', *Drones*, vol. 6, no. 4, pp. 87–87, 2022, doi: 10.3390/drones6040087.

[31] European Commission, 'Unmanned aircrafts in the EU', 2023. https://defence-industry-space.ec.europa.eu/eu-aeronautics-industry/unmanned-aircraft_en (accessed Apr. 18, 2023).

[32] S. I. Abdelmaksoud, M. Mailah, and A. M. Abdallah, 'Control strategies and novel techniques for autonomous rotorcraft unmanned aerial vehicles: a review', *IEEE Access*, vol. 8, pp. 195142–195169, 2020, doi: 10.1109/ACCESS.2020.3031326.

[33] Dassault Systèmes, 'Mass and Section Properties', 2023. https://help.solidworks.com/2010/english/solidworks/sldworks/legacyhelp/sldworks/parts/HIDD_MASSPROPERTY_TEXT_DLG.htm (accessed Feb. 12, 2023).

[34] D. Kim and H. S. Oh, 'Black-box Optimization of PID Controllers for Aircraft Maneuvering Control', *Int J Control Autom Syst*, vol. 20, no. 3, pp. 703–714, Mar. 2022, doi: 10.1007/s12555-020-0915-6.

[35] T. Elmokadem and A. V. Savkin, ‘Towards fully autonomous UAVs: A survey’, *Sensors*, vol. 21, no. 18, Sep. 2021, doi: 10.3390/s21186223.

[36] S. Jha, M. Abhishek Manjrekar, M. S. Khapare, M. P. Jagtap, and M. V. Yadav, ‘Warehouse Inventory Management with Cycle Counting Using Drones’, 2021. [Online]. Available: https://ssrn.com/abstract=3869512

[37] A. Rejeb, K. Rejeb, S. J. Simske, and H. Treiblmaier, ‘Drones for supply chain management and logistics: a review and research agenda’, *International Journal of Logistics Research and Applications*, 2021, doi: 10.1080/13675567.2021.1981273.

[38] European Commission, ‘Drone Strategy: Creating a large-scale European drone market’, Nov. 29, 2022. https://transport.ec.europa.eu/news/drone-strategy-creating-large-scale-european-drone-market-2022-11-29_en (accessed Apr. 18, 2023).

[39] E. Cameron, ‘Skies Without Limits v2.0 - PwC UK’, 2022. https://www.pwc.co.uk/issues/emerging-technologies/drones/the-impact-of-drones-on-the-uk-economy.html (accessed Apr. 18, 2023).

[40] AUVSI, ‘Economic Report’, 2013. https://www.auvsi.org/our-impact/economic-report (accessed Apr. 18, 2023).

[41] Insider Intelligence, ‘Drone market outlook in 2023: industry growth trends and forecast’, Jan. 23, 2023. https://www.insiderintelligence.com/insights/drone-industry-analysis-market-trends-growth-forecasts/ (accessed Apr. 18, 2023).

[42] S. Jatsun, O. Emelyanova, and A. S. Martinez Leon, ‘Design of an Experimental Test Bench for a UAV Type Convertiplane’, in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Jan. 2020. doi: 10.1088/1757-899X/714/1/012009.

[43] S. Jatsun *et al.*, ‘Hovering control algorithm validation for a mobile platform using an experimental test bench’, *IOP Conf Ser Mater Sci Eng*, vol. 1027, no. 1, p. 012008, 2021, doi: 10.1088/1757-899X/1027/1/012008.

[44] F. Cazaurang, K. Cohen, and M. Kumar, *Multi-rotor Platform Based UAV Systems*. Elsevier, 2020.

[45] I. Ökten *et al.*, ‘Test Platform and Graphical User Interface Design for Vertical Take-Off and Landing Drones Test Platform and Graphical User Interface Design for Vertical Take-Off and Landing Drones Test Platform and Graphical User Interface Design for Vertical Take-Off and Landing’, 2022. [Online]. Available: https://www.researchgate.net/publication/363923815

[46] K. Cho, M. Cho, and J. Jeon, ‘Fly a Drone Safely: Evaluation of an Embodied Egocentric Drone Controller Interface’, *Interact Comput*, vol. 29, no. 3, pp. 345–354, May 2017, doi: 10.1093/iwc/iww027.

[47] Z. S. Hou and Z. Wang, ‘From model-based control to data-driven control: Survey, classification and perspective’, *Inf Sci (N Y)*, vol. 235, pp. 3–35, Jun. 2013, doi: 10.1016/J.INS.2012.07.014.

[48] D. Kim, H.-S. Oh, and I.-C. Moon, ‘Black-box Modeling for Aircraft Maneuver Control with Bayesian Optimization’, *Int J Control Autom Syst*, vol. 17, no. 6, pp. 1558–1568, 2019, doi: 10.1007/s12555-018-0401-6.

[49] I.-H. Choi and H.-C. Bang, ‘Adaptive command filtered backstepping tracking controller design for quadrotor unmanned aerial vehicle’, *Proc Inst Mech Eng G J Aerosp Eng*, vol. 226, no. 5, pp. 483–497, Oct. 2011, doi: 10.1177/0954410011415001.

[50] L. Derafa, A. Ouldali, T. Madani, and A. Benallegue, ‘Non-linear control algorithm for the four rotors UAV attitude tracking problem’, *The Aeronautical Journal*, vol. 115, no. 1165, pp. 175–185, 2011, doi: 10.1017/S0001924000005571.

[51] R. Cao, Y. Lu, and Z. He, ‘System identification method based on interpretable machine learning for unknown aircraft dynamics’, *Aerosp Sci Technol*, vol. 126, 2022, doi: 10.1016/j.ast.2022.107593.

[52] B. C. Min, C. H. Cho, K. M. Choi, and D. H. Kim, ‘Development of a Micro Quad-Rotor UAV for Monitoring an Indoor Environment’, in *Advances in Robotics*, J.-H. Kim, S. S. Ge, P. Vadakkepat, N. Jesse, A. Al Manum, S. Puthusserypady K, U. Rückert, J. Sitte, U. Witkowski, R. Nakatsu, T. Braunl, J. Baltes, J. Anderson, C.-C. Wong, I. Verner, and D. Ahlgren, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 262–271.

[53] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, ‘A survey of Open-Source UAV flight controllers and flight simulators’, *Microprocess Microsyst*, vol. 61, 2018, doi: 10.1016/j.micpro.2018.05.002.

[54] L. Ljung, *System Identification: Theory for the user*, 2nd ed. in Prentice-Hall information and system sciences series. Upper Saddle River: Prentice-Hall, 1999.

[55] K. Åström, *Computer-controlled systems : theory and design*, 3rd ed. in Prentice-Hall information and system sciences series. Englewood Cliffs, NJ: Prentice-Hall International, 1997.

[56] K. J. Keesman, ‘Time-varying Dynamic Systems Identification’, *System Identification: An Introduction*, pp. 195–221, 2011.

[57] J. Awrejcewicz, *Modeling, Simulation and Control of Nonlinear Engineering Dynamical Systems*. Springer, 2009.

[58] P. Olejnik, J. Awrejcewicz, and M. Fečkan, *Modeling, Analysis and Control of Dynamical Systems with Friction and Impacts*. World Scientific, 2018.

[59] B. C. Kuo, *Sistemas de control automático*. Pearson Educación, 1996.

[60] Katsuhiko. Ogata, *Modern control engineering*, 5th ed. Upper Saddle River, New Jersey [etc: Pearson, 2010. Accessed: Jun. 14, 2023. [Online]. Available: https://discovery.upc.edu/permalink/34CSUC_UPC/11q3oqt/alma991003687999706711

[61] R. C. Dorf, *Modern control systems*, Thirteenth edition. Harlow, England: Pearson, 2017. Accessed: Jun. 14, 2023. [Online]. Available: https://ebookcentral-proquest-com.recursos.biblioteca.upc.edu/lib/upcatalunya-ebooks/reader.action?docID=5185883

[62] Katsuhiko. Ogata, *Ingeniería de control moderna*, 2ª ed. México D.F. [etc: Prentice-Hall Hispanoamericana, 1993.

[63] Ó. Reinoso García, *Control de sistemas discretos*. in Serie de compendios Schaum. Madrid: McGraw-Hill, 2004.

[64] Ascending Technologies GmbH, ‘AscTec Hummingbird with AutoPilot User’s Manual’, 2013. https://www.yumpu.com/en/document/view/8667713/asctec-hummingbird-with-autopilot-users-manual-getting-started (accessed May 13, 2023).

[65] K. Åström, *PID controllers*, 2nd ed. Research Triangle Park: The International Society for Measurement and Control, 1995.

[66] Interdrone, ‘Intel acquires German drone company Ascending Technologies’, 2016. https://interdrone.com/news/intel-acquires-german-drone-company-ascending-technologies/ (accessed Apr. 04, 2023).

[67] GitLab, ‘GitLab - FFT GYRO’, 2023. https://gitlab.com/eurekadynamics/fft-gyro (accessed Feb. 10, 2023).

[68] B. Rubí, B. Morcego, and R. Pérez, ‘Quadrotor Path Following and Reactive Obstacle Avoidance with Deep Reinforcement Learning’, *J Intell Robot Syst*, vol. 103, no. 4, p. 62, 2021, doi: 10.1007/s10846-021-01491-2.

[69] B. Rubí, B. Morcego, and R. Pérez, ‘Deep reinforcement learning for quadrotor path following with adaptive velocity’, *Auton Robots*, vol. 45, no. 1, pp. 119–134, 2021, doi: 10.1007/s10514-020-09951-8.

[70] UPC, ‘Course guide 3200332-MASD2’, 2022. Accessed: Apr. 22, 2023. [Online]. Available: https://www.upc.edu/grau/guiadocent/pdf/ing/3200332/modelling-and-analysis-of-dinamic-systems-ii.pdf

[71] Open Robotics and A. Dattalo, ‘Introduction to ROS’, 2018. http://wiki.ros.org/ROS/Introduction (accessed Jun. 20, 2023).

[72] Open Robotics, ‘Master - ROS Wiki’, 2018. http://wiki.ros.org/Master (accessed May 05, 2023).

[73] J. M. O’Kane, ‘A gentle introduction to ROS’, 2014, Accessed: May 09, 2023. [Online]. Available: https://jokane.net/agitr/

[74] Anderson, ‘Hands-On Introduction to Robot Operating System (ROS)’, 2020. https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system%28ros%29/ (accessed May 06, 2023).

[75] MathWorks, '"system" command', 2023. https://www.mathworks.com/help/matlab/ref/system.html (accessed May 02, 2023).

[76] MathWorks, 'ROS Toolbox - Matlab', 2023. https://www.mathworks.com/products/ros.html (accessed May 02, 2023).

[77] MathWorks, 'Connect to ROS network', 2023. https://www.mathworks.com/help/ros/ref/rosinit.html?s_tid=doc_ta (accessed May 02, 2023).

[78] M. Achtelik, 'asctec_mav_framework - ROS Wiki', 2011. http://wiki.ros.org/asctec_mav_framework?distro=indigo (accessed May 02, 2023).

[79] MathWorks, 'rosRegisterMessages - MathWorks', 2023. https://es.mathworks.com/help/ros/ref/rosregistermessages.html (accessed Mar. 19, 2023).

[80] MathWorks, 'ROS Toolbox System Requirements', 2023. https://www.mathworks.com/help/ros/gs/ros-system-requirements.html (accessed Feb. 10, 2023).

[81] MathWorks, '"pythonPkgSrc" - MATLAB2020b/ROS interface', 2020. https://es.mathworks.com/matlabcentral/answers/612056-matlab2020b-ros-interface-unrecognized-function-or-variable-pythonpkgsrc (accessed Mar. 19, 2023).

[82] MathWorks, '"rosgenmsg" bug report', 2021. https://www.mathworks.com/support/bugreports/2343419 (accessed May 02, 2023).

[83] MathWorks, 'Parameter Estimation in Simulink', 2023. https://www.mathworks.com/help/sldo/parameter-estimation.html (accessed Feb. 10, 2023).

[84] L. Rosa, 'Elective in Robotics - University of Rome', 2010. http://www.dis.uniroma1.it/~venditt/didattica/eir/02_HummingBird.pdf (accessed May 13, 2023).

[85] K. Ogata, *Discrete-time control systems*, 2nd ed. Englewood Cliffs (New Jersey): Prentice-Hall, 1995.

[86] L. Ljung and MathWorks, ‘System Identification Toolbox™ User’s Guide - Matlab R2014b’, 2014. Accessed: Jun. 21, 2023. [Online]. Available: https://scholar.google.es/citations?view_op=view_citation&hl=es&user=2lo28DgAAAAJ&citation_for_view=2lo28DgAAAAJ:Se3iqnhoufwC

[87] L. Ljung, ‘Introduction to System Identification’, 2012. https://www.mathworks.com/support/search.html/videos/introduction-to-system-identification-81796.html?fq%5B%5D=asset_type_name:video&fq%5B%5D=category:ident/index&page=1 (accessed May 27, 2023).

[88] Matlab, ‘Linear System Identification’, 2022. https://www.youtube.com/watch?v=qC_C04SEV1E&ab_channel=MATLAB (accessed Jun. 01, 2023).

[89] L. Ljung, ‘Identification for control: simple process models’, in *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 2002, pp. 4652–4657 vol.4. doi: 10.1109/CDC.2002.1185112.

[90] MathWorks, ‘Compare identified model output with measured output - MATLAB compare’, 2023. https://www.mathworks.com/help/ident/ref/compare.html?s_tid=doc_ta (accessed May 28, 2023).

[91] MathWorks, ‘Goodness of fit between test and reference data for analysis and validation of identified models - MATLAB goodnessOfFit’, 2023. https://www.mathworks.com/help/ident/ref/goodnessoffit.html (accessed May 28, 2023).

[92] MathWorks, ‘Estimate time delay (dead time) from data - MATLAB delayest’, 2023. https://www.mathworks.com/help/ident/ref/delayest.html?s_tid=doc_ta (accessed May 28, 2023).

[93] MathWorks, ‘Process Models’, 2023. https://www.mathworks.com/help/ident/process-models.html?searchHighlight=process%20models&s_tid=srchtitle_process%2520models_1 (accessed Jun. 03, 2023).

[94] MathWorks, ‘Estimate Process Models using the SITB’, 2023. https://www.mathworks.com/help/ident/ug/estimate-process-models-using-the-app.html (accessed Jun. 03, 2023).

[95] G. James, *Advanced modern engineering mathematics*. Wokingham, England [etc: Addison-Wesley, 1993.

[96] A. O. Shuaib and M. M. Ahmed, ‘Robust PID Control System Design Using ITAE Performance Index’, *Int J Innov Res Sci Eng Technol*, vol. 3, pp. 15060–15067, 2014.

[97] UPC, ‘Taules retributives del personal docent i investigador Any 2022’, 2022.

[98] MathWorks, ‘MATLAB and Simulink - License’, 2023. https://www.mathworks.com/store/ (accessed Jun. 10, 2023).

[99] Microsoft, ‘Microsoft Office 365 in Microsoft Store’, 2023. https://www.microsoft.com/es-es/microsoft-365/buy/compare-all-microsoft-365-products?tab=1 (accessed Jun. 10, 2023).

[100] ECOportatil, ‘HP Probook 450 G5 15,6" i5 8250U, 8GB’, 2023. https://ecoportatil.es/producto/hp-probook-450-g5-156-i5-8250u-8gb-ssd-256gb-full-hd-a?gclid=CjwKCAjwvpCkBhB4EiwAujULMtywQWPUp-fwkPORqichXpbQzc_xZ_UJYzogAlGjoHUiWLZ4m6A85RoCD6YQAvD_BwE (accessed Jun. 10, 2023).

[101] RodriguezDiego, ‘Remote Controller Futaba FF7 System 2,4GHz’, 2023. https://rodriguezdiego.com/producto/emisora-futaba-ff7-sistem-24ghz-mi41000902/ (accessed Jun. 11, 2023).

[102] Amazon, ‘Tattu Batería LiPo 2300mAh 11.1V 45C 3S’, 2023. https://www.amazon.es/Tattu-Bater%C3%ADa-Multicopteros-Helic%C3%B3pteros-diversos/dp/B017GRB6B6/ref=mp_s_a_1_1?crid=1WP03F9UL1YMH&keywords=tattu+3s+2300mah&qid=1680499983&sprefix=tattu+3s+2300mah%2Caps%2C204&sr=8-1 (accessed Jun. 10, 2023).

[103] HardKernel, ‘ODROID-XU4 board’, 2023. https://www.hardkernel.com/shop/odroid-xu4-special-price/ (accessed Jun. 11, 2023).

[104] N. Belmonte, S. Staulo, S. Fiorot, C. Luetto, P. Rizzi, and M. Baricco, ‘Fuel cell powered octocopter for inspection of mobile cranes: Design, cost analysis and environmental impacts’, *Appl Energy*, vol. 215, pp. 556–565, Apr. 2018, doi: 10.1016/J.APENERGY.2018.02.072.

[105] L. Cozzi, O. Chen, and K. Hyeji, ‘The world’s top 1% of emitters produce over 1000 times more CO2 than the bottom 1% – Analysis - IEA’, 2021. https://www.iea.org/commentaries/the-world-s-top-1-of-emitters-produce-over-1000-times-more-co2-than-the-bottom-1 (accessed Jun. 06, 2023).

[106] M. Mulero-Pázmány, S. Jenni-Eiermann, N. Strebel, T. Sattler, J. J. Negro, and Z. Tablado, ‘Unmanned aircraft systems as a new source of disturbance for wildlife: A systematic review’, *PLoS One*, vol. 12, no. 6, Jun. 2017, doi: 10.1371/journal.pone.0178448.

[107] S. Borrelle and A. Fletcher, ‘Will drones reduce investigator disturbance to surface-nesting birds?’, 2017. Accessed: Jun. 06, 2023. [Online]. Available: https://www.researchgate.net/publication/316093269_Will_drones_reduce_investigator_disturbance_to_surface-nesting_birds

[108] J. Perry, ‘Hope to Achieve Net-Zero Emissions? Here’s How Drones Can Help - Consortiq’, 2023. https://consortiq.com/uas-resources/hope-to-achieve-net-zero-emissions-heres-how-drones-can-help (accessed Jun. 06, 2023).

[109] A. A. Nyaaba and M. Ayamga, ‘Intricacies of medical drones in healthcare delivery: Implications for Africa’, *Technol Soc*, vol. 66, Aug. 2021, doi: 10.1016/j.techsoc.2021.101624.

[110] I. ; Zubin, B. ; Van Arem, B. ; Wiegmans, and R. Van Duin, ‘Using drones in the last-mile logistics processes of medical product delivery A feasibility case study in Rotterdam Using drones in the last-mile logistics processes of medical product delivery: A feasibility case study in’, Transportation Research Board, 2020.

[111] S. M. H. M. R. S. A. H. Ehsan Rashidzadeh, ‘Assessing the sustainability of using drone technology for last-mile delivery in a blood supply chain’, *Journal of Modelling in Management*, 2021.

[112] M. A. Figliozzi, ‘Lifecycle modeling and assessment of unmanned aerial vehicles (Drones) CO2e emissions’, *Transp Res D Transp Environ*, vol. 57, pp. 251–261, Dec. 2017, doi: 10.1016/J.TRD.2017.09.011.

[113] EnviroTec, ‘Commercial drones have the potential to half CO2 emissions for freight’, 2021. https://envirotecmagazine.com/2021/10/26/commercial-drones-have-the-potential-to-half-co2-emissions-for-freight/ (accessed Jun. 07, 2023).

[114] IMechE, ‘Delivery drones “could halve freight emissions”’, 2021. https://www.imeche.org/news/news-article/delivery-drones-could-halve-freight-emissions (accessed Jun. 07, 2023).

[115] D. Hrovatin and A. Žemva, ‘Exploiting solar energy during an aerial mapping mission on a lightweight uav’, *Electronics (Switzerland)*, vol. 10, no. 22, Nov. 2021, doi: 10.3390/electronics10222876.

[116] J. Wivou, L. Udawatta, A. Alshehhi, E. Alzaabi, A. Albeloshi, and S. Alfalasi, ‘Air quality monitoring for sustainable systems via drone based technology’, *2016 IEEE International Conference on Information and Automation for Sustainability: Interoperable Sustainable Smart Systems for Next Generation, ICIAfS 2016*, Jul. 2016, doi: 10.1109/ICIAFS.2016.7946542.

[117] J. Burgués and S. Marco, ‘Environmental chemical sensing using small drones: A review’, *Science of The Total Environment*, vol. 748, p. 141172, Dec. 2020, doi: 10.1016/J.SCITOTENV.2020.141172.

[118] M. Maclas, C. Barrado, E. Pastor, and P. Royo, ‘The Future of Drones and their Public Acceptance’, *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, vol. 2019-September, Sep. 2019, doi: 10.1109/DASC43569.2019.9081623.

[119] A. Vacca, H. Onishi, and F. Cuccu, ‘Drones: military weapons, surveillance or mapping tools for environmental monitoring? The need for legal framework is required’, *Transportation Research Procedia*, vol. 25, pp. 51–62, Jan. 2017, doi: 10.1016/J.TRPRO.2017.05.209.

[120] C. Sandbrook, ‘The social implications of using drones for biodiversity conservation’, *Ambio*, vol. 44, pp. 636–647, Nov. 2015, doi: 10.1007/s13280-015-0714-0.

[121] Y. Jiang and C. Lyu, ‘Sky-high concerns: examining the influence of drones on destination experience’, *https://doi.org/10.1080/02508281.2022.2094582*, 2022, doi: 10.1080/02508281.2022.2094582.

[122] S. Anim-Yeboah, R. Apau, and M. Preko, ‘Drones in the Digital Transformation of Healthcare Delivery in Africa’, pp. 31–56, 2022, doi: 10.1007/978-3-030-77987-0_2.

[123] R. Luppicini and A. So, ‘A technoethical review of commercial drone use in the context of governance, ethics, and privacy’, *Technol Soc*, vol. 46, pp. 109–119, Aug. 2016, doi: 10.1016/J.TECHSOC.2016.03.003.

[124] R. Clarke and L. Bennett Moses, ‘The regulation of civilian drones’ impacts on public safety’, *Computer Law & Security Review*, vol. 30, no. 3, pp. 263–285, Jun. 2014, doi: 10.1016/J.CLSR.2014.03.007.

[125] Alexandre Albore, ‘asctec_hl_interface - ROS Wiki’, 2014. http://wiki.ros.org/asctec_hl_interface (accessed Mar. 18, 2023).

[126] MathWorks, ‘Waveform Generator block’, 2023. https://es.mathworks.com/help/simulink/slref/waveformgenerator.html?s_tid=doc_ta (accessed Mar. 18, 2023).

[127] MathWorks, ‘Band-Limited White Noise block’, 2023. https://es.mathworks.com/help/simulink/slref/bandlimitedwhitenoise.html?s_tid=doc_ta (accessed Mar. 18, 2023).

APPENDIX

A. BUDGET

This Appendix is dedicated to the budget of the execution of the dissertation. It is presented as a sum of both, the equipment and human resources' costs. It takes into account the labour, licences and equipment utilised.

Resource type	Resource	Unit	Cost per unit (€/unit)	Hours (h)	Cost per hour (€/h)	Total cost (€)	Comments
Labour	Researcher	1	N/A	650	5,5	€ 3.575,00	Cost of student intern at the UPC. Internal information
Labour	Tutor (1)	1	N/A	25	28,5	€ 713,45	Cost of professor based on annual data [97]
Labour	Tutor (2)	1	N/A	25	28,5	€ 713,45	Cost of professor based on annual data [97]
License	MathWorks products	1	139,0	N/A	N/A	€ 139,00	Toolboxes included [98]
License	Microsoft Office	1	69,0	N/A	N/A	€ 69,00	[99]
License	Ubuntu 16.04	1	N/A	N/A	N/A	€ -	Free license
License	ROS Indigo and libraries	1	N/A	N/A	N/A	€ -	Free license
Equipment	Laptop	1	455	N/A	N/A	€ 455,00	[100]
Equipment	Test bench	1	6200,0	N/A	N/A	€ 6.200,00	[1]
Equipment	Quadrotor	1	3515,0	N/A	N/A	€ 3.515,00	UPC internal information
Equipment	Remote control and channel receiver	1	289,0	N/A	N/A	€ 249,90	[101]
Equipment	LiPo Battery 2100 mAh	6	31,2	N/A	N/A	€ 187,14	[102]
Equipment	Odroid XU-4 board	1	59,0	N/A	N/A	€ 59,00	[103]
Equipment	Miscellaneous	10 %		N/A	N/A	€ 1.587,59	Assumed as 10% of the total cost to include any equipment breakage and services such as lighting, internet, etc.
Total						**€ 17.463,54**	

Table 6: Budget

B. ANALYSIS AND ASSESSMENT OF THE ENVIRONMENTAL AND SOCIAL IMPACT

This Appendix focuses on the environmental and social impact of Unmanned Aerial Vehicles (UAVs). The dissertation's main aim was to find a way to effectively identify and control drones with the expectation that their use will be vastly expanded into the commercial, industrial and consumer sectors. As explained in the literature review (section 1.5), the drone industry is set to have an annual revenue of €10 billion in the year 2030 and employ 100,000 people in Europe [41]. Therefore, it is important to assess the environmental and social implications of this technology.

I. ENVIRONMENTAL IMPACT

Alongside any new technology, it is important not only to see the potential benefits they could bring into the industry in terms of efficiency and reduction of waste, but also what it requires to manufacture, distribute, and deploy them (embodied carbon and life cycle analysis). Therefore, the risks and benefits of UAVs were assessed.

A. RISKS

Harmful batteries to the environment

In the world of drones, LiPo (Lithium Polymer) batteries are widely used. There are many benefits to using these compact, high-energy-density batteries to power unmanned aerial vehicles. However, because LiPo batteries are sensitive to improper handling and have a risk of thermal runaway, it is crucial to treat them carefully. UAVs powered by LiPo batteries need to be stored, charged, and handled in the right way to operate safely and reliably.

Figure 89 and Figure 90 below show the emissions assessment of drones with different battery types (battery and fuel cell). Figure 89 shows the total emissions required to manufacture the two types of batteries. However, the lifespan of the device needs to be taken into account when assessing the environmental impact, as it can decrease in severity from high to acceptable. In Figure 90, for instance, the findings show the cumulative energy demand and global warming potential (GWP) divided by the lifetime hours of the fuel cell and battery [104].

It is forthwith apparent that the production impact of the fuel cell-powered system is now comparable to that of the battery-powered one. The two devices' varied lifetimes, which are longer for the fuel cell than batteries, are what led to this shift. When using a full life cycle assessment, it is concluded that fuel cells are the preferred option.

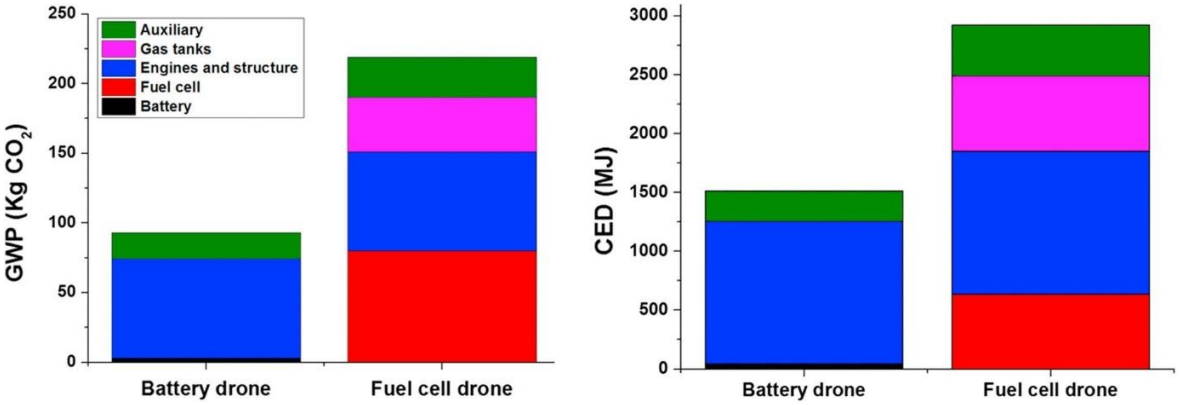

Figure 89: Impact distribution of Global Warming Potential (GWP) and Cumulative Energy Demand (CED) for the production stage of fuel cell and battery-powered drones

Source: [104]

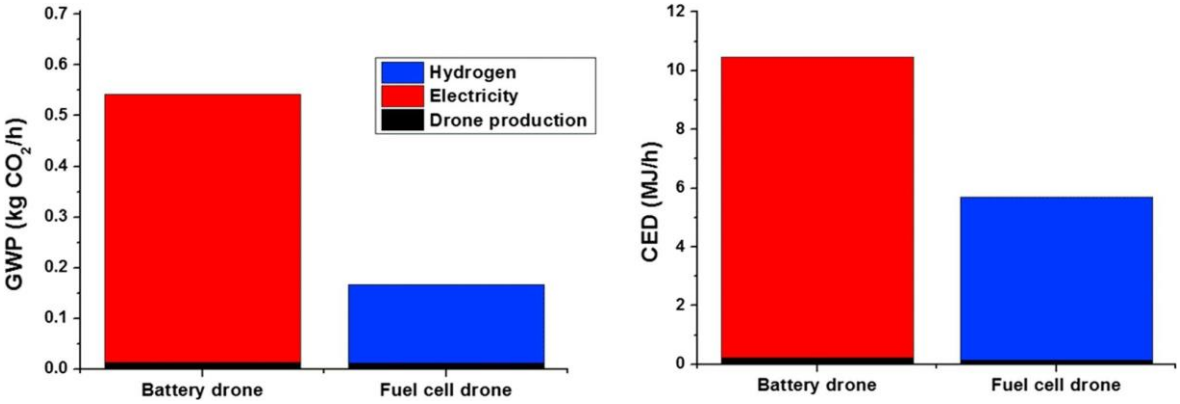

Figure 90: Impact distribution of the considered life cycle stages for GWP and CED

Source: [104]

The average emissions per person globally in 2021 was 4.7 tonnes of CO2 a year [105]. A rough estimation would be that the average person emits 0.55 kg CO2 per hour, which is approximately the same as a battery powered drone emits per hour (Figure 90). Although this number is quite high, further studies could be conducted where other types of drones are utilised, such as fuel cell ones, which have a much lower impact on the environment.

Noise issues

Drone noise is reported to be more unpleasant than any other transportation noise, as Figure 91 demonstrates (blue triangles are automobiles and the black dots are UAVs). The hums and whistles produced by the drone's moving blades and motors give forth a very tonal sound that is very distinctive and is perceived as more disruptive [106].

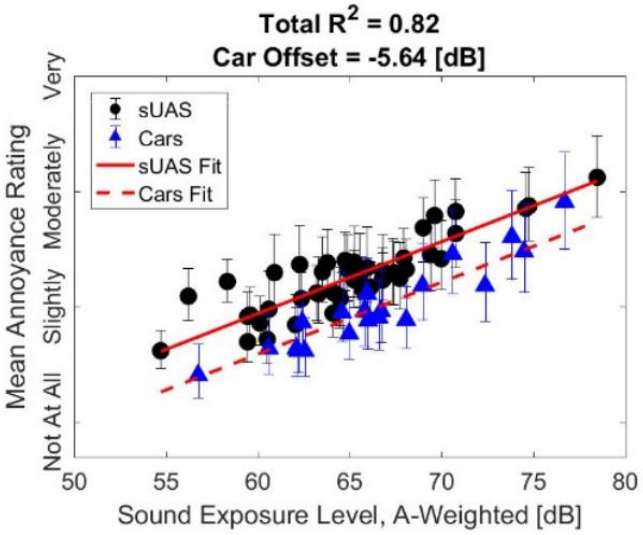

Figure 91: Drone vs. Cars sound exposure levels

Source: [106]

Some potential measures for mitigating drone noise are [106]:

a) Setting the maximum sound levels that drones can produce while in flight is a crucial noise management strategy for preventing noise at the source.
b) Controlling drone flightpaths would probably be a key factor in the future since the technology is set to advance. This is important to reduce noise nuisance, mitigate security and privacy issues and protect wildlife. Studies have revealed that drones can be hazardous by being disruptive and stressful to a variety of animals, thus additional thought must be given to the drone's potential influence on wildlife.
c) Controlling the number of flights in sensitive areas, for instance, in densely populated areas and protected wildlife zones.

The impact of noise on wildlife is explained in the following paragraph.

Biodiversity disturbance impact

Drones are relatively inexpensive nowadays, which has democratized access to high-resolution remote sensing data. Drones can monitor more colonies at higher spatial and temporal resolutions than conventional field approaches. However, there are drawbacks and risks associated with their use for conservation and data collection. For example, many surface-nesting birds, particularly vulnerable to disturbances. might decrease reproductive output and raise stress reactions. Consequently, the creation of drone-operation standards is a crucial first step in limiting disturbance to surface-nesting birds [107].

B. BENEFITS

Drones might cut carbon emissions down by up to 4.5 billion tons in just the transportation industry, according to the World Economic Forum [108]. They can be used for many purposes. For example, for planting trees, monitoring solar panels, delivering products, and many more.

Medical support at higher efficiencies

UAVs have several advantages, including quicker response times in medical emergencies, which can help save more lives, and environmental friendliness due to their lower CO2 emissions than traditional delivery methods using trucks and cars [109]

The phrase "last-mile delivery" describes the last leg of a business-to-customer service, during which, goods are transported by land from a depot to a destination point using vehicles like vans and small trucks [110]. The last-mile delivery is considered as one of the most significant, expensive, and polluting segments in the entire medical supply chain, especially when dealing with perishable goods, such as blood supply [111]. Therefore, incorporating a sustainable strategy into the last-mile logistic plan will be key as the UAV industry expands.

A study revealed lower emissions, which include a reduced service time and delivery costs when UAVs were added to the delivery fleet, with a reduction of 9% in CO2 emissions, 12% in service time, and 5.6% in cost per item [110]. The development of drone technology has ushered in a new era for the health sector. In the next paragraph, the benefits of delivery by drones are further investigated.

Lowering the CO2 emissions significantly

In most industrialized nations, a significant portion of overall green-house gas (GHG) emissions are related to transportation. The use of unmanned aerial vehicles (UAVs) could speed up deliveries and cut expenses [112].

The logistics company DHL has highlighted speedier delivery, lowered accident rates, and increased last-mile efficiency as three of the most important possible UAV benefits [112]. Commercial UAVs could reduce urban freight transport's CO2 emissions by almost half when compared to light commercial vehicles (LCVs) (Table 7), offering a more sustainably option for the logistics sector to lessen its environmental effect [113], [114].

Mode of transport	CO2 emissions produced over 24 hrs	Emissions reduction
LCV	3,394 g CO_2 (3 shifts)	
Large-sized UAV	1,800 g CO_2	47% reduction
Medium-sized UAV	2,160 g CO_2	36% reduction

Table 7: Comparison between different modes of transport in terms of CO2 emissions

Source: [113]

Using drones for surveying and monitoring: a way to reduce emissions

Pipelines, solar panels, and forests are just a few of the enormous regions that drones can examine rapidly and safely. This allows businesses to identify problems more promptly than ever before. In addition, higher maintenance can provide more efficient systems, that produce significantly fewer emissions than they would otherwise. For instance, a study found that using drones to examine solar panels can reduce emissions by up to 96% when compared to conventional techniques. They can also contribute to a 15% boost in the energy production of solar panels [115].

UAVs could also be utilised to gather data from the air sampling field for a specific area in 3D space. The necessary equipment for testing air quality is installed on a drone, then, the storing and monitoring devices will receive data collected by the system. Studies have shown that this system is much more efficient than the traditional method [116].

The usage of such platforms for environmental chemical sensing applications is expanding exponentially because of recent improvements in the downsizing of chemical instrumentation and in low-cost small drones. The quick uptake of chemically sensitive drones in scientific, industrial, and regulatory fields, including atmospheric research investigations, industrial pollution monitoring, and the enforcement of environmental rules, has optimized this industry [117].

II. SOCIAL IMPACT

A portion of the society has always initially rejected emerging technologies. The lack of prior knowledge regarding adverse effects and peoples' psychological aversion to the unknown, in addition to other issues that the immature technology may have, have a significant impact on how well it is received. According to experts, a balance between advantageous uses and inconveniences associated with this technology governs social acceptance [118]. This section of the report focuses on the positive and negative impacts that this technology could have on the community.

A. RISKS

Privacy concerns

The ethical ramifications and potential invasions of privacy and civil liberties of using drones for both military and civilian purposes have received a lot of attention [119]. Drones have been referred to as a "new surveillance" tool in the context of civil uses, raising the questions of being morally appropriate and crossing the line of civil rights. The ability of drones to now be slight and undetectable enough to enter areas that might otherwise be considered private is a cause for concern [120].

In addition to general privacy issues, the use of drones in touristic areas is also a cause of concern. Despite a small number of drone enthusiasts utilising them to capture their travel experiences, most of the visitors do not fly drones and might not want to be filmed. Understanding the responses of non-users would shed light on the part that drones play in influencing the destination experience. Drone annoyance is influenced by privacy worry and safety danger, which in turn has a negative impact on the traveller’s experience [121].

Potential fear generators

Drones have the potential to cause considerable fear, confusion, and hostility among those on the ground. In some cases, this might happen as an accidental consequence of drones being introduced. If people on the ground do not understand why drones are being introduced, they may generate worries and suspicion, particularly when they are used in remote areas of developing countries, because they might have had little or no prior exposure to this technology [120].

Potential for unethical data use

The public might be worried about how and why data gathered by conservation drones is used, especially when some of it could be sold without the people's consent for private uses, such as advertising and product marketing. Additionally, the data could be easily stolen, as it can be shot down, acquired, and destroyed by those seeking access to data [120].

B. BENEFITS

Safety

Since there is no pilot to suffer injuries in a crash, drones are typically thought to be safer for users than piloted aircrafts. They may also be less dangerous for anyone on the ground in an accident, because they typically crash with less damage and destruction than bigger piloted aircrafts. In addition, some drones have safety features that enable them to immediately land at a designated location after terminating a scheduled mission or if any issues arise [120].

More accessible data

According to studies, using modern technology might give local organizations more power, as it gives them the tools to gather their own data, enforce norms, and refute assertions. If drones could be utilized for community-based measurements and other important data, they may offer these advantages to the local population [120].

Improved healthcare quality

Alongside the economic benefits mentioned in the previous section, drones also have a positive social impact in the healthcare sector. For instance, studies have shown increased information, system, and service quality, along with higher user satisfaction from the adoption of drone technology for the medical sector in developing countries. In addition, quick delivery time for drones and accessing healthcare facilities in remote areas were identified as contributing factors to improve well-being of remote communities [122].

III. SUMMARY

As it was discussed in the previous sections, there are several possible uses for drones, both, in the public and commercial sectors, as well as in the industries of agriculture, business, environment, and energy.

In terms of environmental impacts, it was shown by several studies that the benefits and emission reductions that drones could bring highly outweigh the environmental risks that this technology could produce. However, in terms of social impact, they can represent major security and safety threats. Therefore, it is vital to adjust existing law in addition to adopting and enforcing new legislation [119], [123]. Hence, to protect the people's safety and property, consistent rules and regulations must be issued and adopted, and insurance responsibility is key in this regard [124].

Once this technology is more widely spread and local communities are more accustomed to it, drones could bring numerous benefits to many industries, such as healthcare and agriculture. To summarise, UAVs would bring sustainable solutions and provide further support to the whole society, from developing nations and marginalised communities.

C. EXPERIMENT SETUP

This Appendix is dedicated to the setup and operation of the university's AscTec Hummingbird drones in the workbench located at the university's TR11 building. A basic knowledge of Linux is needed to operate the system, although this guide will try to detail every step.

I. REQUIREMENTS FOR EXPERIMENTATION

The basic requirements to run experiments with the quadrotor are:

- Charged 11.1V batteries.
- Fully functioning, strapped drone.
- Testing workbench.
- Drone's scripts to launch its ROS master node (here detailed).
- Drone's and gyroscope's scripts and Simulink documents (here detailed).
- Laboratory laptop properly set up to work with the drone.
- Drone's remote control with charger.

Due to limited space within the drone, not all batteries may be suitable for mounting. The LiPo batteries at [102] are recommended for the drone's compartment.

II. MATLAB/SIMULINK ENVIRONMENT:

Various documents are available for communicating with the UAVs, including Matlab scripts and Simulink interfaces:

1. Scripts:
    - 'DroneSetup.m': basic script to establish a communication's channel through data structures that the drone can understand. Available at Appendix D, item I.
    - 'MsgRegister.m': script that enables the registration of custom messages of the ROS environment that are not available in Matlab by default. The script is essential for users who wish to modify the 'DroneClassEdition.slx' model on their personal computer. Available at Appendix D, item II and explained in Appendix C, item IX.

- ‘SampleReadEncoder.m’: script capable of reading the machine’s encoders. It can work under any operating system (Windows, Linux, or Mac). It can be found on Gitlab [67].

2. Simulink interfaces:
   - ‘NewtonSystem.slx’: this model refers to the work done in section 5.2, for the Newtonian system to explain the pitch and roll gimbal motion with the drone turned off. No connection to the drone is needed.
   - ‘DroneClassEdition.slx’: model capable of connecting with the drone for safe testing. It can handle any kind of test signal, such as steps, ramps and more. It can also connect simultaneously to the gyroscope. It can only be modified with the laboratory’s computer under Linux Ubuntu 16.04.
   - ‘SimulinkSampleEncoder.slx’: model capable of connecting to the new equipment, the gyroscope, at the laboratory. The model was retrieved from manufacturer's GitLab repository [67]. It is capable of reading the workbench’s encoders. It can work on top of any OS (Windows, Linux or Mac).
   - ‘SetPointsPID.slx’: model capable of sending PID setpoints (SP) to the quadrotor. Its segments are explained at section 7.2. A step-by-step for the implementation of the PID controllers is available in this Appendix, item VIII.

Some of the C++ scripts were used as a guidance to understand the previous form of communication with the drone and develop a more user-friendly interface in Matlab/Simulink here explained. The original files were developed by the ETH Zürich [24], [125], and, in a later stage, edited by the university’s professor and former students to open the control loop of the built-in controllers, and additionally, implement a PID controller algorithm coded at the university. As a result, this dissertation had a foundation for the development of the communication’s channel with the drone.

For the following sections, the scripts and models that are going to be used are:

- ‘DroneClassEdition.slx’ model.
- ‘DroneSetup.m’ script.
- ‘SetPointsPID.slx’ model.

Due to the nature of the Linux and ROS systems, the drone can only work inside an Ubuntu 16.04 OS in a modified Matlab R2020b, with certain conditions to be met regarding

Python and CMake versions. For this reason, only the laboratory's laptop is currently capable of communicating with the quadrotor.

III. GENERAL COMMENTS AND GYROSCOPE OUTPUT

To set up experiments with the drone, follow these steps:

1. Turn on the computer. It will automatically start up Ubuntu 16.04.
2. Open a Linux Terminal and enter 'sudo matlab' and type the computer's password to run it as admin. This should open the default Matlab version on it, which should be the R2020b. It is necessary to do it for the correct function of the communication's channel between the UAV and the laptop.
3. Open the 'DroneClassEdition.slx' model (Figure 92); it should open displaying three different segments: Drone Input, Drone Output and Gyroscope Output.

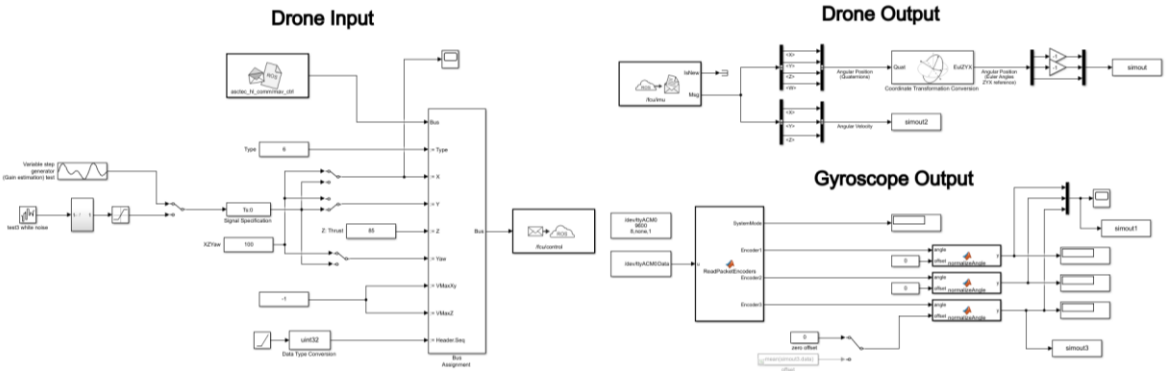

Figure 92: 'DroneClassEdition.slx' model view

Unlike Figure 92, the model has additional comments that were hidden to take the screenshot seen here. The comments will be visible in the file. Beware that they are not a replacement for this appendix, but an additional guidance for the user.

4. If the USB cable is connected to the computer, disconnect it. First connect the workbench to the power supply and turn it on. If done properly, the workbench's three orange lights should start turning on and off in a straight order. If this is not the case, try again.
5. Connect the USB cable to any USB port of the computer. By default, it should be named '/dev/ttyACM0'. All the files (scripts and models) are set up to work with this USB port.

6. When done correctly, all the gyroscope's orange lights turn on at the same time. If this is not the case, go back to step 4, disconnecting the USB cable from the laptop's port.
7. At this point, the encoders' data logging begins (Figure 93). Please, comment the Drone Input and Drone Output segments for safety reasons. This can be done by selecting the blocks and pressing the buttons Ctrl + Caps Lock + X. It can be undone (uncommented) in the same way.

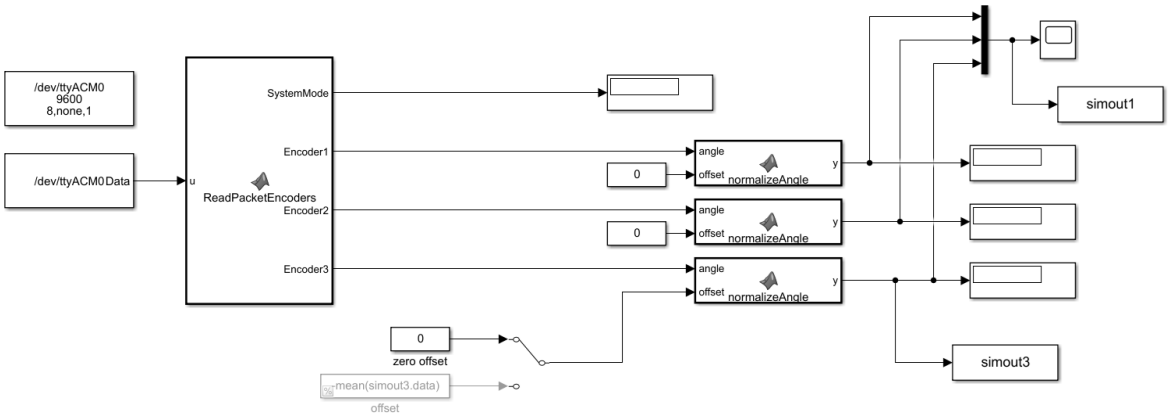

Figure 93: Gyroscope output segment

IV. DRONE POWERUP

After this, the drone's setup can begin. For that, a fully charged battery is needed and the system must be horizontally balanced. This is because the UAV's IMU sensor detects unbalances, and an alarm goes off.

8. Place and connect the drone's battery. After connecting it, the drone is partially powered on. If the drone's switch was previously left on the ON position, the drone's alarm will go off.
9. If that happens, disconnect the battery, and turn the switch to the OFF position.

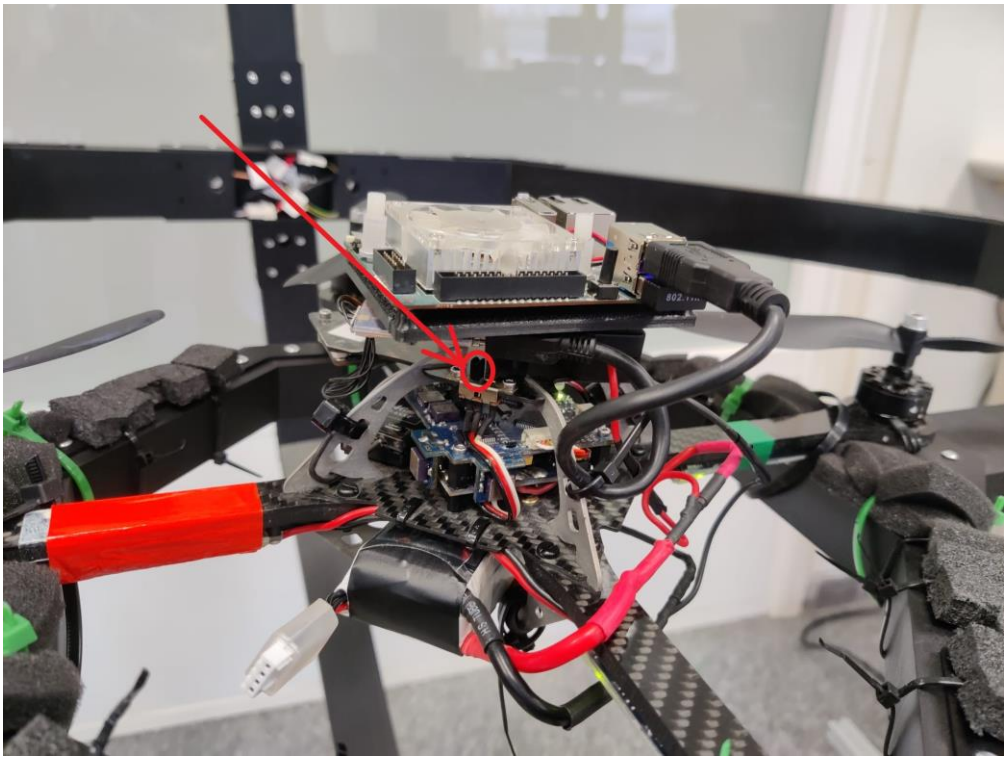

Figure 94: Drone's switch location

10. Hold the drone and the gyroscope to prevent it from moving from the horizontal position. Turn on the drone's power switch. The drone's blades should slightly move. This is an indication that the Electronic Speed Controller (ESC) for the motors work.
11. Two scenarios can occur in this point:
    a. First scenario: When switching the power on, the drone was perfectly horizontally balanced and motionless. This means that it will beep once and implies that the drone is correctly powered on.
    b. Second scenario: When switching the power on, the drone was not balanced and will start beeping nonstop. Then, switch the power off and try again.

    At this step, the user can turn on the drone using the remote control.
12. Plug the remote control to the grid to have enough battery for the experiments.
13. Turn on the remote control. Its voltage level will show on the screen. If it is below 8,4V, it will start beeping. The user should then change the battery or wait until it recharges.
14. At this point, the quadrotor can be turned on.

Warning: Check to see if the cables are in the sweep area of the blades and reposition them. This is critical to avoid damage to the quadrotor.

15. To turn on the drone, push the left stick down and to the right for about two seconds. The blades should start up. If done properly, jump to step 18.
16. If it did not work, the remote control could have been physically misconfigured.
17. There should be two sticks or switches popping up of the controller like ears. To reconfigure the controller, push back the left top stick and push forward the right stick, relative to the 'G' position.
18. Turn off the blades by repeating step 15.
19. Following step 11, the Wi-Fi connector on the Odroid-XU4 board (electronic board at the top of the drone) should light up after a few seconds.
20. Connect the USB cable hanging from the drone to the Odroid-XU4 board.
21. Open the 'Red' (network) window on the left task bar of the Ubuntu OS.
22. Deactivate the '*Hotspot inalámbrico*' switch. A pop-up window will show. Click the 'Detener hotspot' button.
23. At this point, in most cases, the computer will automatically detect and connect to the Wi-Fi network named 'Odroid AP' created by the drone. If this is not the case, try connecting to the network manually by searching for it in the list of available networks.
24. If that did not work, try the next step. Otherwise, jump to step 26.
25. Sometimes, the problem can be bypassed by turning on the airplane mode on the laptop and waiting 10-30 seconds to restart the Wi-Fi board. However, if it still does not work, try disconnecting the drone and starting again from step 8.
26. Open a new terminal and execute 'ssh odroid@10.0.0.200'.
27. Write the drone's password. Wait to establish connection with the drone. If done properly, the terminal should be named 'odroid@10.0.0.200', which means the user connected to the drone.
28. On the same terminal, execute 'roslaunch asctec_hl_interface fcu.launch'. This should launch the ROS master node that allows the drone to publish and subscribe to different topics. An example of an important topic being published by this node is 'fcu/imu', which is the one that will deliver the data from the IMU to Matlab through a subscription in the Simulink environment.
29. When done properly, there should not be any errors printed on the terminal (displayed in red), aside from the battery voltage. This first voltage reading is always 0 volts and should be ignored. That being the case, skip steps 30 and 31.

30. At this point, the UAV could present several errors while trying to execute the code on step 28. If this happens, press Ctrl + C in the terminal and try running the command again several times.
31. If that was ineffective, try going backtracking to step 8, disconnecting the battery and following the steps again, starting from the connection of the battery.
32. However, if that again did not help, it could mean that the battery is not fully charged and makes it difficult for the drone to run the code.

This terminal cannot be used anymore and should remain untouched from this point forward. If done properly, the drone and the gyroscope are correctly powered up and the drone is ready for a connection to the MathWorks products.

V. EXPLANATION OF THE SIMULINK FOR DRONE COMMUNICATIONS

Before the connection between Matlab and the UAV is established, a thorough explanation of the drone segments of 'DroneClassEdition.slx' model is needed for its correct, safe, and effective use.

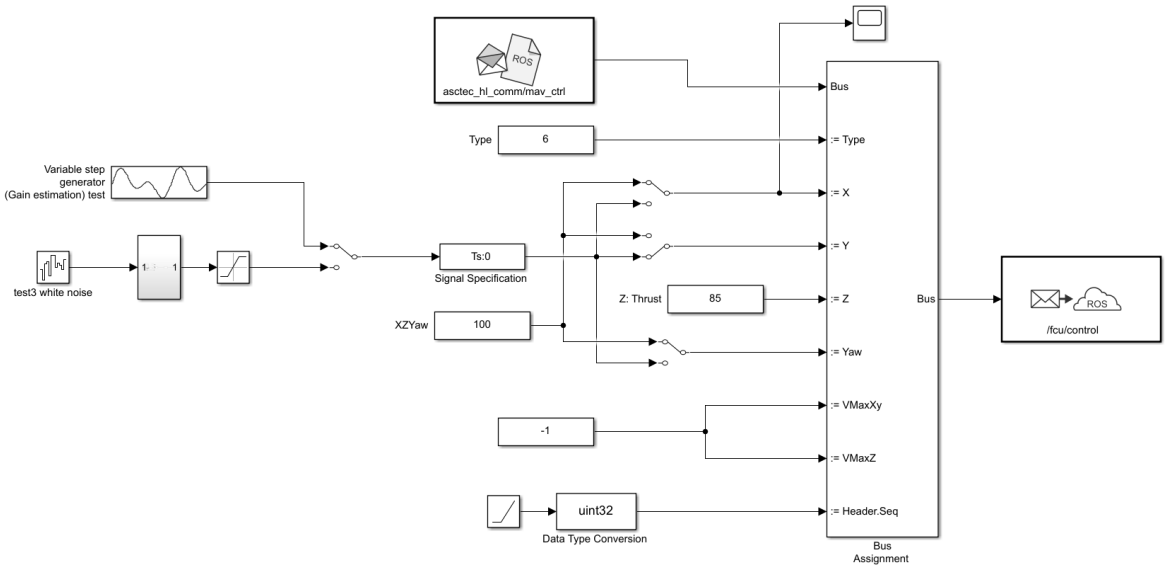

Figure 95: Drone Input segment

The mentioned segments can be seen in Figure 92. These are: Drone Input, Drone Output and Gyroscope Output. The first part was already explained from step 1 to step 7 and Figure 93.

This segment is a vital part of the model itself. It allows a real-time simulation of the system inside Simulink. The Serial Receptor block is the one that keeps the simulation in real time. If commented, this would render the system's time frame unusable, and the dataset would lose the sample time inside them. In short words, the model relies only on the *presence* of the Gyroscope Output segment, but not on its *readings*.

In Figure 95, there is a clean view of the second segment of the model: the Drone Input. It uses the ROS environment, particularly the '/fcu/control' topic with its dedicated custom message called 'asctec_hl_comm/mav_ctrl'. Simulink creates an empty message, which is then filled with the commands the user wants to test, such as step signals.

Quick explanation of the most important parts of the bus signals:

- **Type**: it represents the type of command the user is sending to the drone. Number 6 is intended for direct motor control. Data type is 'single'. Should not be changed.
- **X input**: it represents the **pitch** angle, but it is in fact the action of the motors to produce such a movement. Data type is 'single'. Its range goes from 0 (-100%) to 200 (100%), with 100 meaning 0%.
- **Y input**: it represents the **roll** angle, but it is in fact the action of the motors to produce such a movement. Data type is 'single'. Its range goes from 0 (-100%) to 200 (100%), with 100 meaning 0%.
- **Z input**: it represents the **thrust**, which is the simultaneous action on the four motors of the quadrotor. Data type is 'single'. Its range goes from 0 (0%) to 100 (100%). It is currently set to 85 (42,5%), which is the approximate thrust needed to make the drone hover in the air in an idle mode.
- **Yaw input**: it represents the **yaw** angle but is in fact, the action of the motors to produce such a movement. Data type is 'single'. Its range goes from 0 (-100%) to 200 (100%), with 100 meaning 0%.

To extend the battery's usage, the user should lower the thrust (Z input) to around 30 (15%). Based on practical use, almost double the number of experiments can be done with this reduction. This is recommended to professors because it prevents the necessity of exchanging batteries in the middle of a laboratory class.

It is imperative to understand that the quadrotor does not have a flight controller (PID or otherwise) on the 'DirectMotorControl' mode (message type 6, as explained earlier). That

means, for example, that given a pitch (X) input signal, pitch tends to change, but so do the other angles, although in a much smaller scale and therefore, negligible in a practical sense. This is because the manufacturer built in a decouplers for all the inputs-outputs relations, so that only when one input changes, one output does the same at the time.

In Figure 95 there is a manual switch at the input of the variable X (pitch). The Waveform Generator allows the generation of a complex input signal. This was used to generate a gradual step signal [126]. The other block, called White Noise, was used to generate a random step signal based on a white gaussian noise [127]. These blocks generate continuous signals of 'single' data type. Both must be transformed into a discrete signal with the Signal Specification block. This last block is not to be meddled with. Otherwise, the model will not work.

The last segment of the model is the Drone Output. The drone publishes the IMU's data collection in the '/fcu/imu' topic as quaternions XYZW. This can be observed in Figure 96. This is then transformed into Euler angles through an additional block. They are then rotated to match the dataset published on the 'attitude_ctrl/euler' topic by the drone. Angular velocities are available for the user, although the data contains a lot of noise. The data can then be stored in the Matlab workspace for further use.

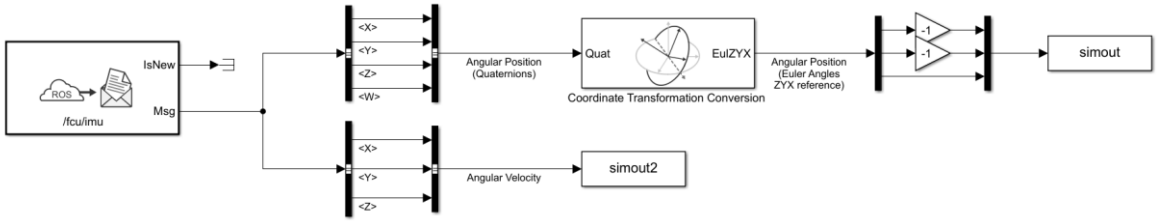

Figure 96: Drone Output segment

VI. TURNING ON AND TESTING

Following with the guide for experimentation, the script must now be explained. Without this, the Simulink interface will not be functional.

33. Open 'DroneSetup.m' script in Matlab. The script is also commented to ease its comprehension.
34. Run the 'Drone IP connection' segment to establish a communication's channel with the quadrotor.

35. Run the ‘Check connection’ segment. Read its comments.

At this point, please be *very careful* while running the model when the connection is established. At the Bus Assignment block of the ‘Drone Input’ segment (Figure 95) neither X, Y nor Yaw are to be left without an input. The block understands this as a zero (0), which is -100% in terms of its range, as explained earlier. If this happens, the motors will start spinning violently and could damage components or even injure the user. As a safety measure, keep the remote controller close and if this happens, please push the left stick down and to the right as explained in step 18.

36. Before turning on the motors, the cables should be checked to see if any of them land on the sweep area of the blades. If this is the case, please carefully move them to a safe position.
37. Run the ‘Motors ON’ segment of the ‘DroneSetup.m’ script. Beware that this turns on the blades. Safe distance is advised.
38. The next segment, called ‘Motors OFF’ turns off the blades. Turning off the blades is recommended through the remote controller, because it is faster and therefore safer. To accomplish this, please push the left stick down and to the right.
39. The final segment is a replica of the Simulink interface. It allows the publication of ‘asctec_hl_comm/mav_ctrl’ custom ROS messages to the ‘/fcu/control’ like the one explained from the end of step 31 on. This drone is not intended to work exclusively with this script. Test signals should not be delivered through this mean.

At this point, the user should have test signals for the experiment based on the blocks Waveform Generator and the White Noise blocks of Simulink. The subsystem found after the White Noise block adds a constant (100) to the generated signal to guarantee the 0% output, as explained earlier. There are also an initial and final stabilization times that the quadrotor needs, which is recommended to be around 5 seconds.

VII. SHUTTING DOWN THE SYSTEM

The correct procedure to shut down the drone and the gyroscope will be explained.

40. Close Matlab and Simulink.
41. Disconnect the USB cable from the laptop.
42. Switch off the gyroscope and unplug its power source.

43. Go to the terminal where the ‘fcu.launch’ file is running and press Ctrl + C.
44. Write ‘sudo shutdown now -h’ and run the command.
45. Write the quadrotor’s password if prompted.
46. Disconnect the USB cable in the quadrotor.
47. Switch off the drone and disconnect the battery from the drone.
48. Remove the battery from the frame and shut down the computer.

VIII. DRONE CONTROLLER’S IMPLEMENTATION GUIDE

This section will present a step-by-step guide to the execution of the ROS node for the chosen PID controller according to the Table 4. It is recommended to read section 7.2 to understand the implementation of the algorithm. Only one Euler angle is controlled at the time, but all of them can be tested simultaneously. Proceed with caution, as the closed loop might be unstable. If only one angle is to be controlled, set the PID gains for the uncontrolled Euler angles to zero. It will open the control loop.

1- Follow the step-by-step guide on this Appendix to power up the drone and execute the ‘fcu.launch’ file. Do **not** turn on the blades yet.
2- Open Simulink and the ‘SetPointsPID.slx’ model (Figure 78), instead of the ‘DroneClassEdition.slx’ model.
3- Change the controller’s parameters in the ‘attitude_pid_data.yaml’.
4- In a new Linux terminal, write and execute:
   `roslaunch path_following_control attitude_command.launch`
5- Check if the algorithm parameters are correct, especially the **gains** and structure (‘cfg’ parameter).
6- On the same terminal kill the node (Ctrl+C).
7- Write and execute:
   `rosrun path_following_control attitude_controller_node`
8- Run the Simulink interface and record the initial Euler angle published on the ‘attitude_ctrl/euler’ topic while keeping the blades in OFF state.
9- Stop the model and set the initial setpoint of the angle to the recorded value.
10- Turn on the blades.
11- Execute the model and wait for the system to stabilize.
12- Set a new setpoint to test the controller.

13- If the drone is unstable, turn off the blades and check whether the PID parameters are set between 1 and -1.

IX. MESSAGE REGISTRATION

This segment is dedicated to the explanation of how and why the user's computer needs to be set up before trying to modify the 'DroneClassEdition.slx' model. For the message registration, the following software is needed:

- Linux Ubuntu 16.04
- Python 2.7 and CMake 3.10.0
- Matlab R2020b (modified) with ROS Toolbox

The requirements for the previously mentioned toolbox must be checked [80]. This guide may work on Windows 10 or 11, but it has not been tested yet.

The modification of Matlab is very simple. A change in an internal script of Matlab is needed for the custom message registration. A complete guide follows. Further information can be found on the Matlab website [79], [81].

1. Open Matlab as a super user: 'sudo matlab' inside a Linux terminal.
2. Go to the following address inside the installation folder of Matlab: toolbox/ros/mlroscpp/+ros/+internal
3. Open the 'CatkinBuilder.m' script.
4. Edit this line so it has the correct version of CMake:
   CMAKEMINVERSION = '3.10.0'
5. Copy the following directory in a Linux file explorer to access the framework: sftp://odroid@10.0.0.200/home/odroid/ROS_AscTec/src/asctec_mav_framework/asctec_hl_comm
6. Copy the whole folder (not only the contents) and paste it to location inside the computer. Do not alter the files. They need to be in a safe location and are not to be moved.
7. Go to the 'MsgRegister.m' script and run it.
8. Follow the additional instructions from Matlab.

D. CODE

This Appendix is dedicated to scripts developed for this dissertation. A close examination of them will be carried out to ensure the correct use for the future.

I. 'DRONESETUP.M'

This script is used in conjunction with the 'DroneClassEdition.slx' model explained in Appendix C. It serves the purpose of connecting the MathWorks' framework to the ROS master node running on the quadrotor.

```
clc
clear all
return

%% General comments

% This script should NOT be ran all at once, but in segments, one after
% the other as here defined. This ensures the correct setup and safety of
% the equipment and its user.

% Only the last part can and should be substituted by the adjacent
% Simulink, which is set to send and record input and output signals from
% the drone and gyroscope (if needed).

%% Drone IP connection

% It will only work if the master node is running in the drone's system.
% Follow the guide to execute the [fcu.launch] file.

 ipaddress = '10.0.0.200'; % drone's IP address to which we connect

 try
     rosinit(ipaddress)
 catch ME
     disp("It is already connected or there is a problem with the master
node running on the drone.")
     disp("If it's the latter, please follow the guide to fix this is-
sue.")
     disp(ME.message)
 end

 % if the rosinit command does not execute correctly, run Matlab as a
 % super user. Inside a Linux's terminal --> sudo matlab

 % This last line should be copied and ran without spaces before or after
 % it.

 %% Check connection
```

```
 % This is not completely necessary, since Matlab prints a message on
% the command window if the previous step worked.
try
    rosnode list;
catch exception
     ipaddress = '10.0.0.200';
     rosinit(ipaddress)
end

% Output should be something like this:
% /fcu
% /matlab_global_node
% /rosout

%% Motors ON

% This allows the MATLAB user to turn on the drone's blades. It can also
% be used to turn off the drone, but it is not recommended because of
% safety reasons. Using the drone's radio controller transmitter, which
% is independent on the connection established between the UAV and the
% computer

% This structure allows the user to send a message and receive a response
% from the drone through a Client-Server comms architecture.
% This next code line creates a 'Service Client' (ROS structure) inside
% Matlab.

client = rossvcclient("/fcu/motor_control","DataFormat","struct");

% Creates a 'Request' message with the neccesary message structure for
% this message inside the ROS environment.

request = rosmessage(client);

% The request is then filled with the necessary information (ON-OFF).

request.startMotors = true

% request.startMotors = false % usa valores booleanos

% Comment the first line and uncomment the last if the drone is to be
% turned off.

% A service is called to execute the action and a 5 second timer is
% determined to establish a connection and receive a response. The timer
% could be set to 10 seconds if the drone is responding slowly. That
% usually means the battery is low. This may not always be the case.

response = call(client,request,"Timeout",5);

if response.motorsRunning == true
```

```
    disp("The motors are ON")

elseif response.motorsRunning == false

    disp("The motors were turned OFF")

else

    disp("The motors have not been able to be turned ON")

end

%% Motors OFF

client = rossvcclient("/fcu/motor_control","DataFormat","struct");
request = rosmessage(client);
request.startMotors = false;
response = call(client,request,"Timeout",5);

if response.motorsRunning == false

    disp("The motors are OFF")

elseif response.motorsRunning == true

    disp("The motors are still turned ON. Please try using the remote
controller to turn them OFF")

else

    disp("The motors have not been able to be turned OFF. Please try us-
ing the remote controller.")

end

%% Signals

% Allows the transmition of basic input signals to the drone, such as
% steps, ramps, sine or cosine functions.

% This segment emulates the C++ scripts found inside the drone, but
through
% MATLAB interface. There is also a Simulink file which allows the
% simultaneous collection of input and output signals from the drone.

% A ROS Publisher is created for this segment, from which the user can
send
% a 'asctec_hl_comm/mav_ctrl' type of message, which will be posted on
the
% 'fcu/control' topic, which the drone takes as an input.
% The 'mav_ctrl' message has been added to the laboratory's ROS message
% list through a registration process detailed on the 'MsgRegister.m'
file
```

```
% developed for this master's dissertation.

chatterpub = rospublisher("fcu/control","asctec_hl_comm/mav_ctrl","Data-
Format","struct");

% The next line creates an empty message, which is going to be filled and
% sent to the corresponding topic of the drone.

msg= rosmessage(chatterpub); % empty message creation
msg.type= int8(6); % DirectMotorControl type of control, according to the
% drone's documentation

msg.x= single(100); % pitch input, 0 (-100%) to 200 (100%).
msg.y= single(100); % roll input, 0 (-100%) to 200 (100%).
msg.z= single(85); % thrust input, 0(0%) to 200 (100%)
msg.yaw= single(100); % yaw input, 0 (-100%) to 200 (100%).

% This last for lines are the starting point (hovering) for every input
% signal to the drone. The 100th mark represent a hovering state. Thrust
% can be set up to be any value. As a practical mark, the drone hovers at
% 85.

% The input Euler angles (XYZ of the message) are not actually Euler
% angles, but it represents a tendency of the drone to move around a
given
% axis. The input goes from 0 (-100%) to 200 (100%).

msg.v_max_xy = single(-1);
msg.v_max_z = single(-1);
step = single(30);
msg;
rate = rosrate(15);
xx=msg.x;
duracion= 10;
time= 30+duration+5;

for i=1:300
    msg.Header.Seq=int8(i+1); % Do NOT touch this. The header needs is an
    % esential part of the input signal.
    send(chatterpub,msg); % This is the codeline responsable of publish-
ing
    % the input data from 'msg' to 'fcu/control' topic, which will be
    % picked up by the drone.
    waitfor(rate);

    % The script does NOT work on a real-time basis. For that, the user
    % needs to try the Simulink file. More information about this on the
    % drone's documentation from the dissertation.

    % There should be around a 5 second stabilitation mechanism to
    % establish hovering status.
    if i<30
        msg.x=xx; % Stabilization mechanism. Hovering.
```

```
    elseif i>30 && i<300-30
        msg.x= xx+step;
    else
        msg.x=xx; % Back to stability to finish the experiment. It is NOT
        % esential because if the drone is strapped to the gyroscope.
    end
end
```

II. ‘MSGREGISTER.M’

This code is meant for the registration of ROS custom messages inside the MathWorks environment, as explained in section IX.

```
%% Register custom messages in Matlab

 % Guide from MathWorks for the custom message registration:
 %  https://es.mathworks.com/help/ros/ref/rosregistermessages.html
 clc
 clear all

 % Copy the asctec_hl_comm folder (not only the contents but also the
 % folder itself) and place it inside a folder inside the computer. The
 % folderpath to the custom messages, services and actions is:

 % sftp://odroid@10.0.0.200/home/odroid/ROS_AscTec/src/asctec_mav_frame-
work/asctec_hl_comm

 %  The folderpath inside the computer is arbitrary:

 folderpath= "C:\Program Files\MATLAB\R2020b\ROSCustomMessages"

 % IMPORTANT: The files must be exactly the same without any changes.
Only
 % copy the folders named above, NOTHING else, otherwise the message
 % registry will fail.

 rosgenmsg(folderpath)

 % From this point forward, Matlab will give some instructions to follow.
 % After executing them, the messages, services, and actions will be
ready
 % to be used.

 % To check whether the messages were correctly registered by running the
 % following command inside the command window of Matlab:

 % rosmsg list
```

III. MODIFIED ‘CTRL_TEST.CPP’

The first version of this script is available on Github [78]. The entire framework was developed by the Swiss Federal Institute of Technology (Eidgenössische Technische Hochschule) and upgraded at the UPC. It is the basis of the communications channel developed for the dissertation. For the completely upgraded version, contact professor Morcego Seix. The code found on section I is based on this script.

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <ros/ros.h>
#include <asctec_hl_comm/mav_ctrl.h>
#include <asctec_hl_comm/mav_ctrl_motors.h>
void usage()
{
  std::string text("usage: \n\n");
  text = "ctrl_test_Souf_pitch motors argv[2]\n";
  text += "argv[2] = [1 | 0]\n\n";
  text += "ctrl_test_Souf_pitch test step argv[3] argv[4]\n";
  text += "argv[3] = amplitude of the step\n";
  text += "argv[4] = length of the step\n\n";
  text += "ctrl_test_Souf_pitch test ramp argv[3] argv[4]\n";
  text += "argv[3] = maximum value of both ramp\n";
  text += "argv[4] = length of each ramps\n\n";
  text += "ctrl_test_Souf_pitch test sine argv[3] argv[4] argv[5]\n";
  text += "argv[3] = amplitude of the sine\n";
  text += "argv[4] = frequency of the sine\n";
  text += "argv[5] = length of the sine\n";
  std::cout << text << std::endl;
}
int main(int argc, char ** argv) {
  ros::init(argc, argv, "interfacetest");
  ros::NodeHandle nh;
  ros::Publisher pub;
  if (argc == 1) {
    ROS_ERROR("Wrong number of arguments!!!");
    usage();
    return -1;
  }
  std::string command = std::string(argv[1]);
  std::string signal = std::string(argv[2]);

  if (command == "motors") {
    asctec_hl_comm::mav_ctrl_motors::Request req;
    asctec_hl_comm::mav_ctrl_motors::Response res;
    req.startMotors = atoi(argv[2]);
```

```
  ros::service::call("fcu/motor_control", req, res);
  std::cout << "motors running: " << (int)res.motorsRunning << std::endl;
 }
 else if (command == "test") {  // Steps Input Signal
 if (signal == "step") {
    asctec_hl_comm::mav_ctrl msg;
    msg.type = asctec_hl_comm::mav_ctrl::directMotorControl;
    msg.v_max_xy = -1;   // use max velocity from config
    msg.v_max_z = -1;
    msg.x = 100;         // pitch  0..200 -> -100%..100%
    msg.y = 100;         // roll   0..200 -> -100%..100%
    msg.z = 30;         // thrust 0..200 ->    0%..100%
    msg.yaw = 100;       // yaw    0..200 -> -100%..100%
    pub = nh.advertise<asctec_hl_comm::mav_ctrl> ("fcu/control", 1);
    ros::Rate r(15);     // ~15 Hz
    int xx=msg.x;        //pitch
    int time1 = 75+atoi(argv[4])*15;
    int time = time1+75;          //5s stabilisation + atoi(argv[4])s step + 5 s of stabilisation
    for (int i = 0; i < time; i++) {
      pub.publish(msg);
      if (i < 75) {                    //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else if ((i>=75) && (i<time1)) {    //length of the range
        msg.x=xx+atoi(argv[3]);           //atoi(argv[4])s step of atoi(argv[3]) amplitude
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else {                           //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
    }
    ros::spinOnce();
  }                                  //end Steps Input Signal
// Ramps Input Signal
  else if (signal == "ramp") {
    asctec_hl_comm::mav_ctrl msg;
    msg.type = asctec_hl_comm::mav_ctrl::directMotorControl;
    msg.v_max_xy = -1;   // use max velocity from config
    msg.v_max_z = -1;

    msg.x = 100;         // pitch  0..200 -> -100%..100%
    msg.y = 100;         // roll   0..200 -> -100%..100%
    msg.z = 30;         // thrust 0..200 ->    0%..100%
```

```
    msg.yaw = 100;       // yaw    0..200 -> -100%..100%

    pub = nh.advertise<asctec_hl_comm::mav_ctrl> ("fcu/control", 1);
    ros::Rate r(15);     // ~15 Hz
    int xx=msg.x;        //pitch
    int time1 = atoi(argv[4])*15;              //lenght of the ranges
    int time2 = 75+time1;
    int time3 = 75+time1*2;
 int time = time3+75;                          //5s stabilisation + atoi(argv[4])s growth ramp +
atoi(argv[4])s degrowth ramp + 5 s of stabilisation
    for (int i = 0; i < time; i++) {
      pub.publish(msg);
      if (i < 75) {                          //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else if ((i>=75) && (i<time2)) {          //length of the first range
        msg.x=xx+(i-75)*atoi(argv[3])/time1;      //atoi(argv[4])s ramp grow to atoi(argv[3])
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else if ((i>=time2) && (i<time3)) {        //length of the second range
       msg.x=xx-(i-time3)*atoi(argv[3])/time1;   //atoi(argv[4])s ramp grow to atoi(argv[3])
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else {                                 //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
    }
    ros::spinOnce();
  }                                          //end Ramp Input Signal
// Sine Input Signal
  else if (signal == "sine") {
    asctec_hl_comm::mav_ctrl msg;
    msg.type = asctec_hl_comm::mav_ctrl::directMotorControl;
    msg.v_max_xy = -1;   // use max velocity from config
    msg.v_max_z = -1;
    msg.x = 100;         // pitch  0..200 -> -100%..100%
    msg.y = 100;         // roll   0..200 -> -100%..100%
    msg.z = 30;         // thrust 0..200 ->    0%..100%
    msg.yaw = 100;       // yaw    0..200 -> -100%..100%
    pub = nh.advertise<asctec_hl_comm::mav_ctrl> ("fcu/control", 1);
    ros::Rate r(15);     // ~15 Hz
```

```
    int xx=msg.x;       //pitch
    int time1 = 75+atoi(argv[5])*15;
    int time = time1+75;            //5s stabilisation + atoi(argv[5])s sine + 5 s of stabilisation
    int PI = 4*atan(1);
    for (int i = 0; i < time; i++) {
      pub.publish(msg);
      if (i < 75) {                                   //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else if ((i>=75) && (i<time1)) {                //length of the range
msg.x=xx+atoi(argv[3])*sin(2*PI*atof(argv[4])*(i-75)/15.0);          //atoi(argv[5])s    sine    of
atoi(argv[3]) amplitude and atoi(atoi[4]) frequence
        if (!ros::ok())
          return 0;
        r.sleep();
      }
      else {                                          //5s stabilisation
        msg.x=xx;
        if (!ros::ok())
          return 0;
        r.sleep();
      }
    }
    ros::spinOnce();
  }                                                   //end Sine Input Signal
  }
 return 0;
}
```

IV. ITERATIVE TUNING ALGORITHM

This section presents the recursive tuning algorithm utilized in section 7.1. It has an ITAE performance index built in. The algorithm presented here is applicable directly to the pitch and roll angles. The pitch case is analysed as an example. The code was written in Matlab and commented as follows:

```
%% MATLAB code: Recursive tuning algorithm for pitch

clc
clear all
close all

num = [0.01614];
den = [1 0.2187 1.087];
t = 0:0.2:150;
```

```
k = 0;
s = tf('s');
tf1 = tf(num,den)

StepSize = 1;

for Kp = StepSize:StepSize:100;
    for Ki = StepSize:StepSize:100;
        gc = Kp+Ki/s; % PI controller
        G = feedback(gc*tf1,1); % closed loop TF
        y = step(G,t);
        infostep = stepinfo(G);
        ts = infostep.SettlingTime; % Settling time Ts
        m = max(y);
            if m<=1.02 % minimal overshoot (2%)
                if ts<=40.00
                    k = k+1;
                    % Compute the error signal
                    [y,tOut] = step(G,t);
                    e = ones(size(t)) - y';
                    % Compute the ITAE performance index
                    A = abs(e).*t;
                    ITAE = trapz(t,A,2);
                    solution(k,:) = [Kp Ki m ts ITAE];
                end
            end
    end
end
sortsolution = sortrows(solution,1,'descend');

Kp = sortsolution(1,1);
Ki = sortsolution(1,2);
gc = Kp+Ki/s;
PitchResp = feedback(gc*tf1,1);
step(PitchResp, 'r')
stepinfo(PitchResp)
grid on
box off
title('')
ylim([0 1.2])
```