Comparison of Controllers for a Quadcopter Model with Implemented Path Planning Algorithm

Author:	ALEJANDRO FAJARDO PIA
ID:	201800011
Supervisor:	SAIKAT DUTTA
Academic year:	2023/2024

SCHOOL OF MECHANICAL ENGINEERING

UNIVERSITY OF LEED

MECH3890 – Individual Engineering Project

PROJECT TITLE: Comparison of controllers for a Quadcopter Model with Implemented Path Planning Algorithm.

PRESENTED BY: Alejandro Fajardo Pia

SUPERVISED BY: Saikat Dutta

If the project is industrially linked, tick this box and provide details below ☐

COMPANY NAME AND ADDRESS:

STUDENT DECLARATION (from the "LU Declaration of Academic Integrity")

I am aware that the University defines plagiarism as presenting someone else's work, in whole or in part, as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance. I promise that in the attached submission I have not presented anyone else's work, in whole or in part, as my own and I have not colluded with others in the preparation of this work. Where I have taken advantage of the work of others, I have given full acknowledgement. I have not resubmitted my own work or part thereof without specific written permission to do so from the University staff concerned when any of this work has been or is being submitted for marks or credits even if in a different module or for a different qualification or completed prior to entry to the University. I have read and understood the University's published rules on plagiarism and also any more detailed rules specified at School or module level. I know that if I commit plagiarism I can be expelled from the University and that it is my responsibility to be aware of the University's regulations on plagiarism and their importance. I re-confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes. I confirm that I have declared all mitigating circumstances that may be relevant to the assessment of this piece of work and that I wish to have taken into account. I am aware of the University's policy on mitigation and the School's procedures for the submission of statements and evidence of mitigation. I am aware of the penalties imposed for the late submission of coursework.

Date 01/05/2024

Signed

Firmado por ALEJANDRO FAJARDO PIA - NIF:***6248** el día 02/05/2024 con un

Table of Contents

Table of figures ..... **V**

Table of tables ..... **VI**

Nomenclature ..... **VII**

Abstract ..... **VIII**

Chapter.1 ..... **1**

1.1.- Introduction 1

1.2.- Aim 1

1.3.- Objectives 2

1.4.- Project Report Layout 2

Chapter.2 – UAV dynamical equation modelling **3**

2.1.- Introduction: How does a quadcopter work 3

2.2.- Dynamic equations methodology 4

2.2.1.- Dynamic equations mathematical obtention 4

2.2.3.- Max torque available 5

2.2.4.- Simulink implementation: Dynamic equations block 6

2.3.- Dynamic equations results 6

2.4.- State space model methodology 7

2.3.1.- State space model mathematical obtention 7

2.5.- State space model results 8

2.6.- Chapter discussion 8

Chapter.3 – Virtual scenario creation and navigation **9**

3.1.- Introduction: Type of scenario 9

3.2.- Methodology 9

3.2.1- Scenario creation in MATLAB 9

3.3.2- How to navigate the virtual scenario 10

3.2.3.- Mission profile 10

3.2.4.- Path planning algorithms 11

3.2.5.- A* algorithm MATLAB implementation 11

3.3.- Path planning algorithm results 11
3.4.- Simulink implementation: Flight path block 12
3.5.- Chapter discussion 12
Chapter.4 – Controllers modelling 13
4.1.- Introduction: Different types of controllers available 13
4.2.- PID Methodology 13
4.2.1.- PID functioning 13
4.2.2.- Simulink Implementation: PID block 14
4.2.3.- PID tunning 15
4.3.- LQR Methodology 17
4.3.1.- LQR functioning 17
4.3.2.- Simulink implementation: LQR block 18
4.3.3.- LQR tunning 18
4.4.- PID results 19
4.5.- LQR results 20
4.6.- Chapter discussion 21
Chapter.5 – Other Simulink blocks 22
5.1.- Introduction: Other Simulink blocks 22
5.2.- Visualization block 22
5.3.- Energy consumption block 22
5.3.1.- Energy consumption block results 22
5.4.- Chapter discussion 23
Chapter.6 – Conclusion 24
6.1.- Achievements 24
6.2.- Discussion 24
6.3.- Conclusions 25
6.4.- Future work 25
References 26
Appendices 30

Appendix.1 – Extra calculations.....30
Appendix.1.1 - State space model obtention:.....30
Appendix 1.2.- Body frame to inertial frame transformation.....30
Appendix.2 – MATLAB code.....31
Appendix.3 – Simulink blocks.....38

Table of Figures

FIGURE 1: 4 movements of a quadcopter. 3
FIGURE 2: Quadcopter schematics. 4
FIGURE 3: 3D representation of the 2D matrix. 9
FIGURE 4: trajectory created by A* path planning algorithm. 11
FIGURE 5: Simulink controller PID block. 14
FIGURE 6: Step signal and PID controllers' response. 16
FIGURE 7: LQR problem 17
FIGURE 8: Simulink controller LQR block. 18
FIGURE 9: Trajectory followed by the quadcopter - PID. 19
FIGURE 10: Trajectory followed by the quadcopter - LQR. 20
FIGURE 11: Energy consumption of PID (top) and LQR (bottom). 23

Table of Tables

TABLE 1: Maximum response available..... 6
TABLE 2: Dynamical model required parameters. 6
TABLE 3: Requirements and performance comparison of PID.....16
TABLE 4: Mean deviation, Mission time and Runtime - PID.....19
TABLE 5: Mean deviation, Mission time and Runtime - LQR 20

Nomenclature

Acronyms:

UAV:	Unmanned aerial vehicles
LQR:	Linear quadratic regulator
MPC:	Model prediction control
PID:	Proportional-Integral-Derivative
rpm:	Revolutions per minute
ARE:	Algebraic Riccati equation

Notation:

ω_i:	Rotational velocity [rad/s]
F_i:	Lift forces produced by motors [N]
T_i:	Torque produced by motors [N*m]
ϕ:	Roll angle [º]
θ:	Pitch angle [º]
ψ:	Yaw angle [º]
l:	Arm length [m]
K_F:	Force coefficient
K_M:	Moment coefficient
U_i:	Movement response of quadcopter [N]
m:	Quadcopter's mass [m]
g:	Gravitational acceleration [m/s^2]
Ix, Iy, Iz:	Principal moments of inertia [kg/m^2]
P:	Weight [N]
F_{up}:	Quadcopters' lift force [N]
sc:	Scale value use for LQR tunning
P_{Mech}:	Mechanical power [Watts]
P_{Elec}:	Electrical power [Watts]

Abstract

UAVs are recognized to be at the vanguard of autonomous technology. These aerial vehicles are capable of performing a wide variety of tasks without the need of human intervention, however deep understanding of control techniques is required for completing the mission successfully. Hence, this paper aims to compare 2 different control systems −Proportional-Integral-Derivative (PID) and Linear-Quadratic-Regulator (LQR)− to determine which one develops a trajectory following mission through a urban scenario most effectively and efficiently.

Initially, the dynamic model of the quadcopter was developed using Newton-Euler's equations to represent the physics of the drone accurately. Next, a path planning algorithm was implemented to dictate the desired trajectory. Finally, both controllers were implemented and evaluated, and various results to compare were obtained such as trajectory deviation, mission time and energy consumption.

From the results obtained, the LQR showed a more optimal an accurate performance compared to the PID, while a similar degree of complexity in its implementation was exhibited. Consequently, the LQR is the preferred control system for the specified mission.

Chapter.1

1.1.- Introduction

In the actual environment of unmanned aerial vehicles (UAVs), autonomous navigation is a key concept for improving and optimizing their performance. Quadcopters in specific, offer agility and manoeuvrability and can be scaled to low dimensions, being the perfect selection for a dense urban environment. Nevertheless, time and dedication are needed to deeply comprehend the intricacy of control systems, to later implement them in the quadcopter platform.

In this project, various control systems were tested in a prepared digital scenario to study which one completes a specified mission in the most effective and efficient manner. By examining controllers ranging from the more than known PID (Proportional-Integral-Derivative) controller to other more optimised and complex systems, their limitations and strengths are shown and a conclusion of which one is the most suitable selection, was made. In addition, a path planning algorithm system was implemented to ensure that the UAV can complete its task in an autonomous way.

The whole project was based on a computational approach, where strong software as MASTLAB and Simulink provided a suitable strategy to develop the required dynamic models, to implement the path planning algorithm in the system and to develop multiple tests and obtain various results when comparing the controllers selected. This approach offers some advantages in comparison with an experimental approach: real life hazards to the UAV and people are avoided, only computational resources are required instead of expensive specialized equipment and safety measures, data collection is easier in software than in a real experiment and the process can be repeated multiple times without external and uncontrollable factors.

This paper contributed firstly to the progress in UAV technology, but also studied the implementation of this autonomous systems in other applications related to surveillance, emergency response, reconnaissance, and beyond urban environments.

1.2.- Aim

The aim of the project is to implement, comprehend, compare, and analyse the performance of various controllers tasked with following a designated path of waypoints provided by a path planning algorithm. This involves navigating through a virtual representation of an urban environment to evaluate the effectiveness of each controller.

1.3.- Objectives

- To find suitable literature information on how to describe the dynamic equations of the model, implement suitable path planning algorithms and lastly, comprehend the functioning and limitations different types of controllers have.
- To establish the dynamic equations of the model, making the necessary assumptions to obtain a simple yet accurate representation of a real quadcopter UAV, and implementing them in a Simulink case.
- To implement via MATLAB and Simulink a path planning algorithm which enables the UAV to reach its destination in a relatively efficient manner.
- To construct in Simulink multiple controllers which are able to follow the path planning algorithm route.
- To compare and study the results provided by each of the controllers and finally establish which one follows the path in the most accurate way, also considering the simplicity or complexity needed to implement the given algorithm.

1.4.- Project Report Layout

In this section, the structure of the project is presented including the information each chapter provides. Each chapter's follows a similar layout as the following one: a short introduction is provided, the methodology followed to obtain the results is explained, the results of the chapter are shown, and a final discussion of the conclusions obtained from the chapter is presented.

In first chapter, the dynamical model of the quadcopter was studied, and the state space model was also computed, followed up by its implementation in Simulink. Then, in chapter 2, the virtual scenario was created, and multiple ways of navigating it were analysed. This led to the selection of a determinate path planning algorithm, which implementation was then showed, and the resultant trajectory was also presented.

Chapter 4 and chapter 5 represent the main sources of results from the project, as they presented the performance of the controllers on the environment created. First, in chapter 4, different controllers were studied and 2 were chosen for comparison; its functioning, implementation, tunning, and results from each controller were obtained and a discussion on the most appropriate option was done. Finally, in chapter 5 the energy consumption of the quadcopter was analysed together with the visualization block of the Simulink script.

In the final chapter, the achievements done during the project were presented, together with an overall discussion of the report and a conclusion where the more optimal controller was chosen. Future work aspects were also included for further improvement of the project.

Chapter.2 – UAV dynamical equation modelling

2.1.- Introduction: How does a quadcopter work

A typical quadcopter has 4 different types of movements each produced by the forces and torques generated by the rotational velocity of its motors. Generally, UAV's rotors spin in pairs in the same direction, counteracting rotational moments and facilitating manoeuvrability. These 4 movements are shown in FIGURE 1.

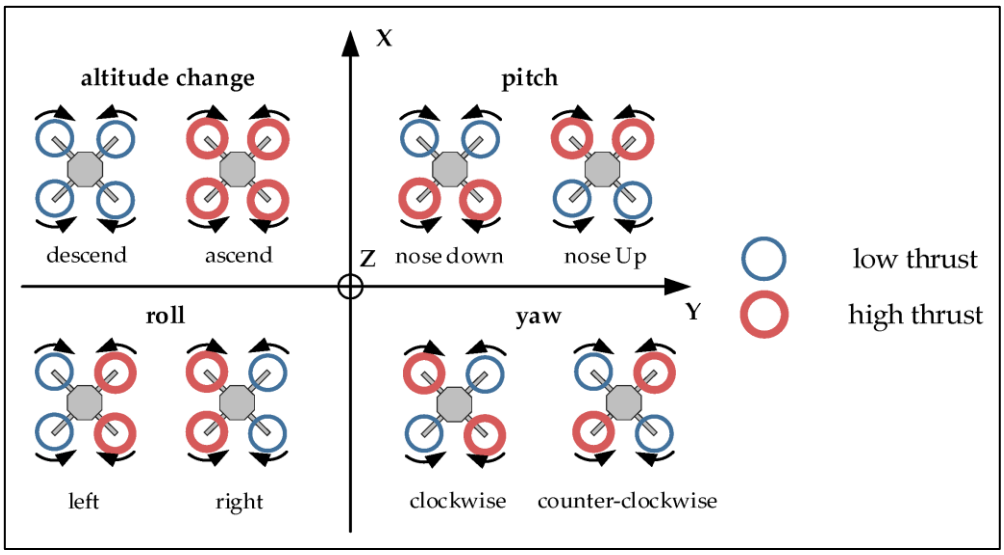


FIGURE 1: 4 movements of a quadcopter. [1]

The quadcopter's altitude adjustment is produced by the force generated by all four motors. Increasing this force results in upward flight. On the other hand, pitch and roll motions come from the increased thrust of two adjacent motors, multiplied by their distance from the centre of gravity. This produces a pitching or rolling moment depending on which motors increase in thrust. In addition, yaw motion is generated by opposing motors spinning in the same direction increasing its angular velocity, hence inducing rotational moment to the UAV. Understanding these principles is essential for later discussions on dynamic equations.

Nevertheless, this explanation only approximates the real behaviour of a quadcopter. As it can be seen in [2], there are plenty of other disturbances affecting this system. Some examples of these are: Aerodynamic forces acting on the blades of the rotors (drag), increase in lift do to ground effect, or even meteorological factors affecting the quadcopter.

However, all these disturbances fall out the scope of the project and were not considered, as the aim is to construct an accurate and usable model of a quadcopter for controller comparison, not to build and exact representation of how a quadcopter would function in all aspects of reality.

2.2.- Dynamic equations methodology

2.2.1.- Dynamic equations mathematical obtention

In this section, the dynamic equations of the quadcopter were obtained. To do this, first the representation of the quadcopter with the inertial frame is presented in FIGURE 2:

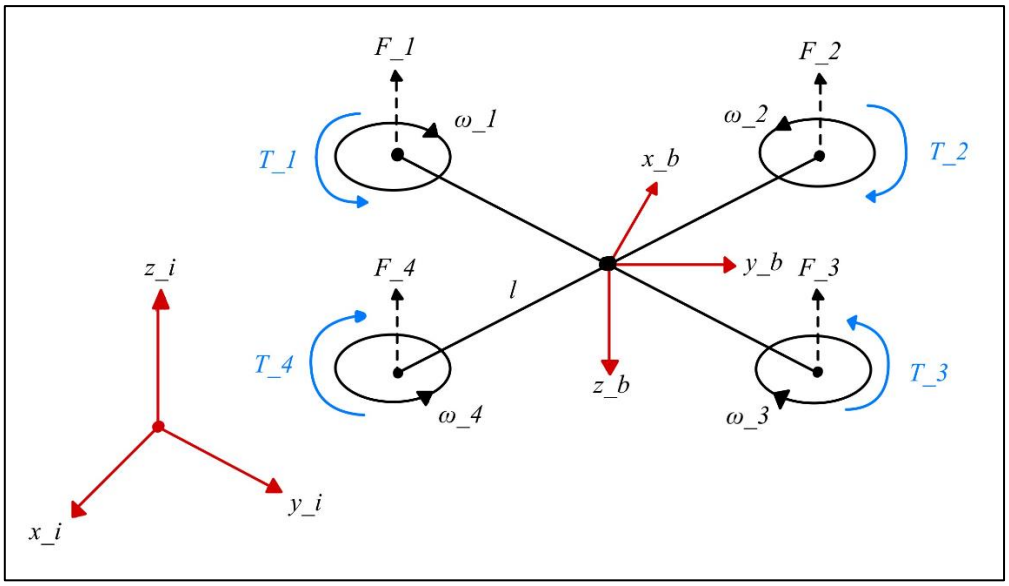


FIGURE 2: Quadcopter schematics.

Note that the body frame of the quadcopter positive z-axis is pointing downwards, while the inertial frame z-axis is opposite All four arms of the UAV are of equal length, each hosting identical motors at their ends. Rotational velocities are denoted as ω_i, the forces produced by each rotor are F_i and the torques T_i; roll, pitch and yaw are ϕ, θ and ψ respectively and finally the length of each arm is l.

Then, kinematics of the model were studied. First, the transformation matrix required for transforming from the body frame (x_b) to the inertial frame (x_i), also known as matrix R, was obtained from [3] and is shown in (1):

$$[R] = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi - c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\psi & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad where \quad \begin{matrix} c\alpha = \cos(\alpha) \\ s\alpha = \sin(\alpha) \end{matrix} \tag{1}$$

With the rotation matrix obtained, the dynamics of the model were studied. First, it was established that ω_1 and ω_3 would rotate clockwise, opposed to rotors 2 and 4. Next, the expressions for force and torque produced by each motor were defined, shown in expressions (2) and (3):

$$F_i = \omega_i^2 * K_F \tag{2}$$

$$T_i = \omega_i^2 * K_M \tag{3}$$

Where: K_F and K_M are force and moment coefficients depending on propeller properties.

Next, the upward force and roll, pitch and yaw moments were defined as U_1, U_2, U_3 and U_4 respectively, which are the responses of the quadcopter. These expressions were obtained by combining (2) and (3) produced by the motors, and referencing FIGURE 1 to check the quadcopter reaction to each force. This is seen in the expressions (4) to (7):

$$U_1 = F_1 + F_2 + F_3 + F_4 = (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) * K_F \quad (4)$$

$$U_2 = [(F_1 + F_3) - (F_2 + F_4)] * l = [(\omega_1^2 + \omega_3^2) - (\omega_2^2 + \omega_4^2)] * K_F \quad (5)$$

$$U_3 = [(F_1 + F_2) - (F_3 + F_4)] * l = [(\omega_1^2 + \omega_2^2) - (\omega_3^2 + \omega_4^2)] * K_F \quad (6)$$

$$U_4 = (T_1 + T_3) - (T_2 + T_4) = [(\omega_1^2 + \omega_3^2) - (\omega_2^2 + \omega_4^2)] * K_M \quad (7)$$

Finally, the dynamic equations of the quadcopter were computed using the 2 equations presented below: Newton's equation (8) and Euler's equation (9), both obtained from [3].

$$m * \begin{bmatrix} \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \end{bmatrix} = [R] \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} + W = [R] \begin{bmatrix} 0 \\ 0 \\ U_1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} I_x * \ddot{\phi} \\ I_y * \ddot{\theta} \\ I_z * \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \times \begin{bmatrix} I_x * \dot{\phi} \\ I_y * \dot{\theta} \\ I_z * \dot{\psi} \end{bmatrix} \quad (9)$$

2.2.3.- Max torque available

As seen before, U_i serves as the responses of the controllers. If no constraints were added, controllers could demand unlimited power, which is not feasible in reality. Thus, saturation limits were imposed at the controller output, limiting the maximum available response.

Given the relationship between forces/torques and angular velocity (seen in expressions (2) and (3)), limiting the angular velocity directly limits the available forces and moments. For recreational quadcopters, rotors range between $10\,000$ and $20\,000\ rpm$ [4], [5], this is why the maximum available velocity of each motor was chosen as $15\,000\ rpm$ as safety measure.

Now, looking at the expressions of the response U_i previously calculated, the maximum response for each case was easily computed as expression (10) to (13):

$$U_{1_{max}} = \left(\omega_{1_{max}}^2 + \omega_{2_{max}}^2 + \omega_{3_{max}}^2 + \omega_{4_{max}}^2\right) * K_F = 4 * \omega_{max}^2 * K_F \quad (10)$$

$$U_{2_{max}} = \left[\left(\omega_{1_{max}}^2 + \omega_{3_{max}}^2\right) - \left(\omega_{2_{min}}^2 + \omega_{4_{min}}^2\right)\right] * K_F = 2 * \omega_{max}^2 * K_F \quad (11)$$

$$U_{3_{max}} = \left[\left(\omega_{1_{max}}^2 + \omega_{2_{max}}^2\right) - \left(\omega_{3_{min}}^2 + \omega_{4_{min}}^2\right)\right] * K_F = 2 * \omega_{max}^2 * K_F \quad (12)$$

$$U_{4_{max}} = \left[\left(\omega_{1_{max}}^2 + \omega_{3_{max}}^2\right) - \left(\omega_{2_{min}}^2 + \omega_{4_{min}}^2\right)\right] * K_M = 2 * \omega_{max}^2 * K_M \quad (13)$$

Where: $\omega_{max} = 15\,000 * 0.1057\ rad/s$ and $\omega_{min} = 0\ rad/s$

The maximum available values obtained for the responses are shown in TABLE 1.

TABLE 1: Maximum response available

Maximum responses	
$U_{1\,max}$ [N]	29.6090
$U_{2\,max}$ [N m]	2.9609
$U_{3\,max}$ [N m]	2.9609
$U_{4\,max}$ [N m]	0.0197

2.2.4.- Simulink implementation: Dynamic equations block

Now, the Simulink implementation is explained. For the Dynamic equation block, 6 different sub-blocks were created inside of the *Dynamic equations* block, each of this simulating one of the equations obtained from the non-linear dynamics model (presented later in the results).

From each of these, the second derivative variable of each equation was obtained, which were then integrated to obtain position, orientation, velocity, and angular velocity. Additionally, saturation blocks, utilizing values from TABLE 1, were included at the controller output. The Simulink blocks are found in APPENDIX.3 .

2.3.- Dynamic equations results

Expanding expressions (8) and (9), the nonlinear dynamic model of the UAV was obtained, presented in expressions (14) to (19):

$$m * \ddot{X} = U_1 * (c\phi s\theta c\psi + s\phi s\psi) \quad (14)$$

$$m * \ddot{Y} = U_1 * (c\phi s\theta s\psi - s\phi s\psi) \quad (15)$$

$$m * \ddot{Z} = U_1 * (c\theta c\phi) - m * g \quad (16)$$

$$I_x * \ddot{\phi} = l * U_2 + \dot{\theta} * \dot{\psi} * (I_y - I_z) \quad (17)$$

$$I_y * \ddot{\theta} = l * U_3 + \dot{\phi} * \dot{\psi} * (I_z - I_x) \quad (18)$$

$$I_z * \ddot{\psi} = U_4 + \dot{\phi} * \dot{\theta} * (I_x - I_y) \quad (19)$$

Where: m is the mass, g is the gravitational acceleration and I_x, I_y and I_z are the principal moments of inertia. All parameter values are found in TABLE 2, obtained from [3].

TABLE 2: Dynamical model required parameters.

Parameters					
Length (l) [m]	0.2	Ix [kg/m^2]	0.11	Moment coeff (K_M)	4.00E-09
Mass (m) [Kg]	2.5	Iy [kg/m^2]	0.11	Force coeff (K_F)	3.00E-06
Gravity cte (g) [m/s^2]	9.81	Iz [kg/m^2]	0.04		

2.4.- State space model methodology

2.3.1.- State space model mathematical obtention

This section establishes the state-space model derived from the non-linear dynamic model, which is essential for the LQR section explained in Chapter.4. Constructing this model involves the non-linear model to be linearized around the equilibrium hovering point and assuming certain approximations. The approximations made, obtained from [3], were:

- Equilibrium hovering point: $P = F_{up} \rightarrow m * g = U_1$

- Small angles were considered: $c\alpha = 1$ and $s\alpha = \alpha$, hence derivatives of angles are 0.

- The drone was assumed to maintain same yaw orientation all time: $\psi = \dot{\psi} = 0$

Hence, the dynamic model presented before were converted to expressions (20) to (25):

$$Expression\ (10) \rightarrow \ddot{X} = g * \theta \tag{20}$$

$$Expression\ (11) \rightarrow \ddot{Y} = -g * \phi \tag{21}$$

$$Expression\ (12) \rightarrow \ddot{Z} = U_1/m - g \tag{22}$$

$$Expression\ (13) \rightarrow \ddot{\phi} = l/I_x * U_2 \tag{23}$$

$$Expression\ (14) \rightarrow \ddot{\theta} = l/I_y * U_3 \tag{24}$$

$$Expression\ (15) \rightarrow \ddot{\psi} = 1/I_z * U_4 \tag{25}$$

The procedure to obtain these expressions is seen in APPENDIX.1.1 It is necessary to note that for expression (22) the gravity constant, obtained from the weight force, cannot be directly included in the standard state-space model as it is a disturbance. This disturbance could be treated as a separated matrix inside the model given equation: $\dot{x} = Ax + Bu + Gw$ where w represents the disturbance found in the model. However, it could also be added to the controller as an external factor, easier to implement in Simulink. This means that expression (22) without the disturbance was converted to equation (26).

$$\ddot{Z} = U_1/m \tag{26}$$

Next, the state space form was computed, found in equation (27).

$$\dot{x} = A * x + B * u \tag{27}$$

Where x and u were defined as:

$$x = [X \quad Y \quad Z \quad \dot{X} \quad \dot{Y} \quad \dot{Z} \quad \phi \quad \theta \quad \psi \quad \dot{\phi} \quad \dot{\theta} \quad \dot{\psi}]^T \tag{28}$$

$$u = [U_1 \quad U_2 \quad U_3 \quad U_4]^T \tag{29}$$

2.5.- State space model results

Hence, combining expressions (27), (28) and (29), the state space model in matrix form was computed obtaining the matrix expression (30). This model implementation is explained later when the LQR is introduced in Chapter.4.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \ddot{X} \\ \ddot{Y} \\ \ddot{Z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ \dot{X} \\ \dot{Y} \\ \dot{Z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{l}{I_x} & 0 & 0 \\ 0 & 0 & \frac{l}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \quad (30)$$

2.6.- Chapter discussion

In this chapter, both the dynamic models model equation and the state space model of the quadcopter obtention were explained. This is a crucial part of the project, as the real representation of the virtual quadcopter fully depends on the *Dynamic equations* block.

Initially, assumptions of the physical phenomena actuating on the UAV were constructed. These assumptions ensured an accurate representation of the drone's movement without making the project too complex. Moreover, maximum values for thrust and torque were included to ensure that the motor propellers behave correctly. Finally, Euler's and Newton's equation were used to derive 6 equations that represent the non-linear model of the quadcopter.

On the other hand, accurate approximations were made to construct the state space model. This is a necessary model that will be used later in the LQR controller block.

In summary, the equations that define the model of the UAV were obtained without too complex processes, ensuring however that a reliable representation of the real movement of a quadcopter is obtained.

Chapter.3 – Virtual scenario creation and navigation

3.1.- Introduction: Type of scenario

The variety of scenarios where a quadcopter can be used is immense [6], spanning from open fields to crowded cities and the interior of buildings or even scenarios where natural catastrophes as earthquakes, fires or floodings have occurred. But even bigger are different tasks and missions that UAVs can perform, including surveillance or reconnaissance duties, rescue missions, goods delivery, and urban security tasks.

In this project, an urban-type scenario was chosen, where the quadcopter was required to navigate from a point A to a point B avoiding obstacles. This is mission profile, although applied to urban settings, can be extrapolated to multiple other duties mentioned above.

3.2.- Methodology

3.2.1- Scenario creation in MATLAB

The scenario that was virtualised in MATLAB is the representation of a city, hence buildings need to be modelled in 3D. Other objects like moving cars, traffic lights and people could be included, however the model restricted a high-fly zone where the drone actuated, where these assets were unnecessary.

The scenario was first constructed as a 2D binary map, formed by a $50x50$ matrix, where "1" denotes buildings and "0" represented free zone. Then, each "1" was converted into an obstacle in the 3D representation of the map with a height of $20\ m$. See FIGURE 3.

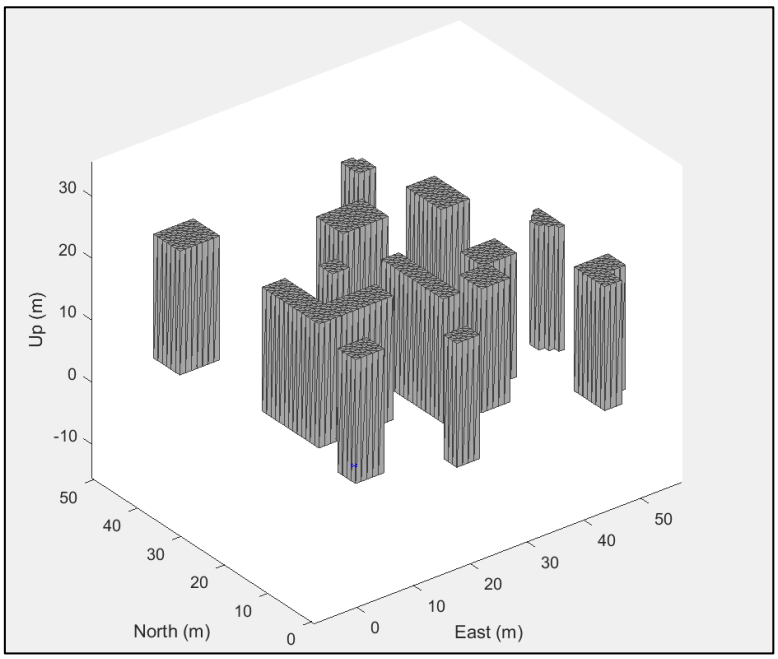


FIGURE 3: 3D representation of the 2D matrix.

Now, it is important to mention that two different 2D matrices were employed to construct the map. APPENDIX.2 provides the MATLAB code involved. The initial matrix was used for the Path planning algorithm, while the 3D map was based on a reduced version of this matrix, where the first and the last rows and columns of each 'building' were eliminated.

This was a necessary adjustment because, while the path planning algorithm planed the optimal trajectory by flying close to obstacles, the controllers did not follow exactly the path dictated due to their imperfections. Hence, in order to allow more manoeuvring space for the quadcopter, the dimensions of the 3D buildings were reduced by 1 unit.

3.3.2- How to navigate the virtual scenario

Once the scenario was created, it was necessary to understand the different ways a UAV could navigate it. Generally, path planning or obstacle avoidance algorithms are used to travel the scenario. Depending on their approach, 2 different types of algorithms exist [7]:

- *Global method algorithms*: algorithms that have overall information of the scenario, enabling them to plan a trajectory that guarantees the convergence to the target. This information is usually the position of the obstacles and disturbances of the map.

- *Local method algorithms*: algorithms that are provided with limited information from the environment and hence do not ensure convergence. These algorithms are based on data obtained from various instruments such as altimeters, ultrasonic or infrared sensors.

This paper opted for a global algorithm method, as no sensors or instruments needed to be implemented and ensuring that the UAV reached its final destination. Realistically, it can be assumed city or metropolis was 3D scanned using satellite scanning technology or similar methods [8], which enabled the drone to plan the desired trajectory.

3.2.3.- Mission profile

The mission profile was then defined: The main objective of the quadcopter was to navigate from an initial point to a final point without crashing into an obstacle, and following the path defined as accurately as possible, optimizing speed and energy consumption.

First, the drone ascended from ground level to 10 m. After this, the UAV followed the path dictated by the algorithm at a constant altitude until reaching the destination and descending to ground level, simulating a landing.

Although the mission is inherently 3D planned, the challenging part of avoiding the buildings occurred at constant height, hence this section could be reduced to 2D. This was a key aspect when developing and implementing the path planning algorithm, as it could be simplified to a more manageable 2D version instead of a more complex 3D one.

3.2.4.- Path planning algorithms

When selecting global path planning algorithms, a bast number of options exist. The fundamental one is *Dijkstra's algorithm* [9], which process is the following one: various points are created between the initial and the final point, a cost is assigned to travel between each point and then the algorithm selects the path to follow based on the least cost possible.

Although *Dijkstra's algorithm* always provides the shortest path, other more effective alternatives exist that also ensure reaching the desired destination. 2 of the most used path planning algorithms are the *A** [10] and the *RRT** [11] *algorithms,* while the first one uses a heuristic approach and works well with low dimensional grids, *RRT** functions as a sampling-based path planning algorithm and works better in high dimensional grids.

For the report, *A** was the algorithm chosen due to its simplicity and to the fact that the grid had small dimensions, although *RRT** could still be a suitable solution if preferred.

3.2.5.- A* algorithm MATLAB implementation

The MATLAB implementation of the *A** algorithm was rather simple and is shown in APPENDIX.2 Initially, the function "plannerAstargrid" [12] took the map matrix and created a map object. Then, the initial and the final points were defined and lastly, using the function "plan" [12] the *A** algorithm planned the most optimum trajectory to reach the destination.

3.3.- Path planning algorithm results

The trajectory planned is seen in FIGURE 4, with $(0,0)$ and $(35,50)$ as start and end.

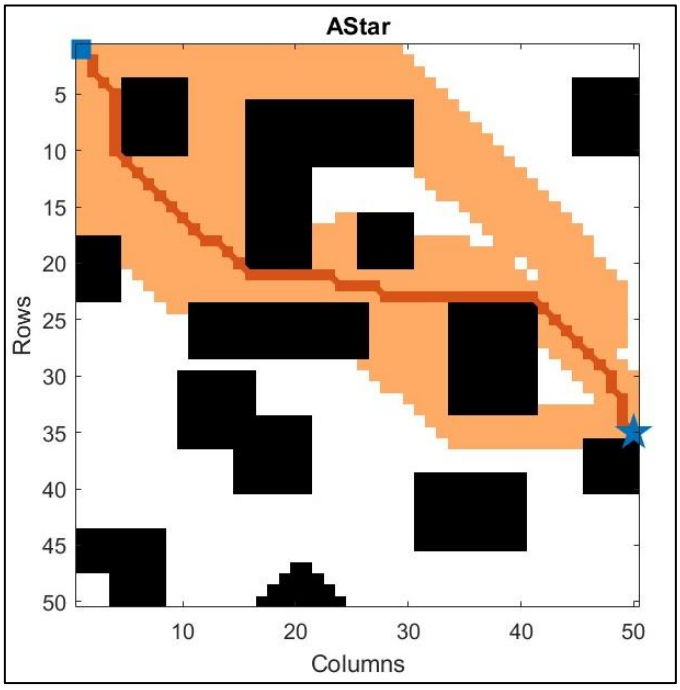


FIGURE 4: trajectory created by A path planning algorithm.*

The final output of these functions was a $[Nx2]$ vector of the $x-y$ coordinates of all the waypoints in order from start to finish, where N was the total number of waypoints.

3.4.- Simulink implementation: Flight path block

Once the 3D map and the vector of waypoints were created, the next step involved integrating this information in Simulink. This implementation was done in the *Flight path block,* which can be further analysed in APPENDIX.3 .

The main part of this section is the *Waypoint follower* block [13] which operates as follows: the inputs of the block are the real-time position of the quadcopter, the vector of waypoints and the *lookahead* distance, which is a variable that defines how closely the UAV follows the path, while the block gives a variety of outputs, where the *lookahead point*, which can be understood as the desired $x-y-z$ position, was the main one used in the project.

As a summary, this block sets the desired position as the following waypoint defined by the path planning algorithm once the UAV has reached the previous waypoint, until it reaches its destination.

3.5.- Chapter discussion

In this chapter, a urban scenario was created in 3D and a suitable trajectory that reached the destination avoiding the presented obstacles was obtained from the *A** path planning algorithm.

Initially, the scenario was created with sufficient obstacles to ensure a rigorous testing of the path planning algorithm was performed. The path planning algorithm on the other hand, provided an accurate solution for the mission profile specified. If the mission had been a 3D trajectory, a more complex algorithm and a different Simulink implementation would have been required, however approximating it to a 2D profile allowed for an easier implementation and less computational effort required.

Overall, the map and the trajectory provided by the *A** algorithm presented a challenging but interesting environment for the controllers to be evaluated, ensuring that the UAV completed its mission effectively.

Chapter.4 – Controllers modelling

4.1.- Introduction: Different types of controllers available

In this chapter, the controllers that were used in the project are presented. First, it is important to understand the multitude of available controllers, where the PID defines an industry standard due to its simplicity and facility of implementation [14]. Although the PID is popular option, there are multiple other considerations that present more efficient performance, although being more complex and computational demanding than PID. These include Model Prediction Control (MPC) [15], Linear Quadratic Regulator (LQR) [16], [17], Hybrid controllers [18] and even PID could be implemented in other forms, as for example a cascade PI [19], where 2 controllers are used in series; or an Integrated PID [2], used to consider angle deviations.

However, the objective of the paper was to compare 2 different controllers. Hence, the initial option was the PID, given its popularity. Then, multiple options were considered for the second controller. At the end, as it offered a good balance between complexity and optimisation, LQR was selected as the second controller, as it provided a more optimal response in exchange for having the equations implemented in state space form.

4.2.- PID Methodology

4.2.1.- PID functioning

First, the functioning of the PID is explained. This controller is divided into 3 different parts briefly explained below, as a detailed explanation would need more sophisticated concepts:

- *Proportional part*: deals with the current values of the error.

- *Integral part*: deals with the past values of the error.

- *Derivative part*: deals with the future values of the error.

Its operation is simple: 2 values of a parameter are introduced to the system, the desired value and the actual value, which are subtracted obtaining the error. Then, the PID minimises this error to 0 by introducing a value to the plant or system studied. This adjustment is influenced by the tunning of each part, which will change the error until it reaches 0.

This simplicity makes the PID a very affordable and easy option for indrustry. However, this controllers has its drawbacks: tunning process can be laborious and time-consuming without proper consideration (tools like apps or *Bode diagram* [20] can be used to make the process easier) and filters or safety mesures may be necessary to ensure controller convergence.

4.2.2.- Simulink Implementation: PID block

The implementation of the PID is shown in FIGURE 5. It shows that 4 different PID blocks were required, one for each movement of the quadcopter: altitude and roll-pitch-yaw control. Hence, the outputs of each block are the variables controlling each movement: U_1, U_2, U_3, U_4.

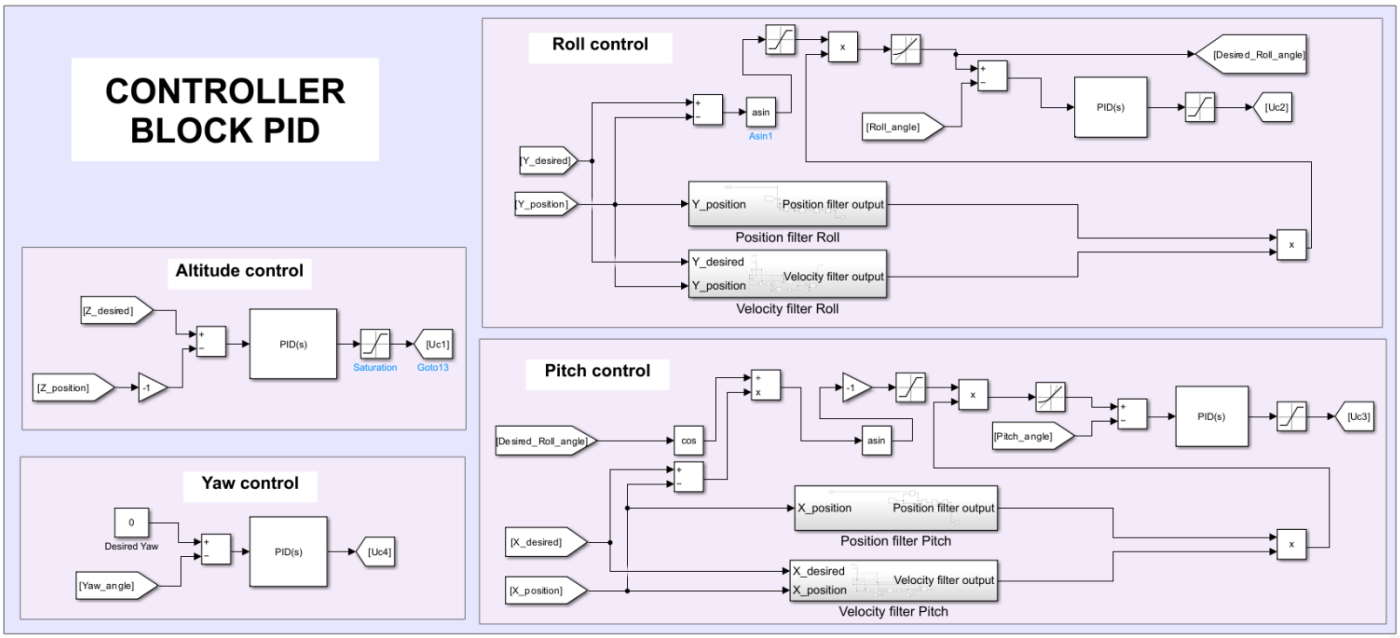


FIGURE 5: Simulink controller PID block.

In order for the PID to work, a conversion between Euler angles and $x - y - z$ coordinates was needed, as it can be seen in expressions (14) to (19), where the movements were defined by orientation not position. To do this conversion, the following needed to be assumed.

- The drone needed to be pointing in the direction of the final point to reach it.

- The quadcopter positive z-axis pointed downwards.

- The drone maintained constant yaw throughout.

Given the rotation matrix between the inertial frame and the drone body frame (expression (1)), the conversion between the global and the body frame was computed as expression (31):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{Inertial_{frame}} = [R] \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}_{UAVbody_{frame}} \tag{31}$$

Computing the matrix multiplication of (31) and assuming $\psi = 0$, expressions (32) and (33) were obtained. Further understanding of the calculation is seen in APPENDIX 1.2.

$$\phi = \arcsin(y) \tag{32}$$

$$\theta = -\arcsin\left(\frac{x}{\cos(\arcsin(y))}\right) \tag{33}$$

Once obtained, the desired $x - y$ coordinates were introduced to the controllers, which then are converted to the desired $\phi - \theta$ angles. These desired angles were compared with the actual roll and pitch angles, and the error was introduced to the PID block.

Returning to FIGURE 5, the altitude controller and the yaw controller were simpler to implement. The first one did not require any conversion, so the desired altitude was compared to the real one, and the error was introduced into the block. The second one, typically remained inactive due to the absence of external disturbances affecting the yaw angle. However, the roll and pitch controllers, apart from the conversion mentioned before, were introduced with 2 filters or safety features that ensure a correct functioning of the controller:

- *Position filter*: This filter reduced the output of the PID smoothly when approaching the desired position. This ensured that transitions between the desired angle were more gradual and did not suffer from abrupt changes.

- *Velocity filter*: This filter analysed the quadcopter velocity when being close to the desired position. If this velocity was excessive, the filter adjusted the angle direction to minimize overshoot.

4.2.3.- PID tunning

The following step after implementation was tunning the controllers. Multiple ways of tunning the PIDs exist: manual methods as Ziegler-Nichols [21] and Cohen-Coon [22] methods, Bode diagrams, trial and error, which is unprecise and time-consuming, and apps and software that automatically tune the PID. Hence, as Simulink was able to tune the PID whether aggressive or solid behaviour was needed [23], this last method was chosen.

During PID tunning, multiple things needed to be considered. The most important was if the controller reached the final destination. However other requirements [24] were defined to optimize the functioning and the results of the PID:

- *Rise time*: time required by the controller to reach 80% of the desired value.

- *Settling time*: time required by the controller to reach minimum error value of 5%.

- *Maximum overshot*: maximum peak value of the response above the desired value.

Tunning the PID controllers in this project was a complex task, due to multiple controllers sharing the same propellers and the indirect control of $x - y$ position with roll and pitch angles. Hence, the requirements were defined in a permissive setting, prioritizing mission competition.

For altitude control the 3 parameters were studied with more restrictive settings due to its simpler functioning. However, roll and pitch controllers had the inconveniences mentioned before, hence some oscillations were expected around the desired position value.

Nevertheless, an error of no more than $1\ m$ was permitted for these controllers to prevent the UAV from colliding with adjacent buildings. Lastly, yaw controller tunning requirements were not necessary, as it was tunned with a standard behaviour and lacked any disturbance.

Then, step signals of 5 and 2 values were introduced to altitude and roll-pitch angles respectively to evaluate their performance. This is seen in FIGURE 6.

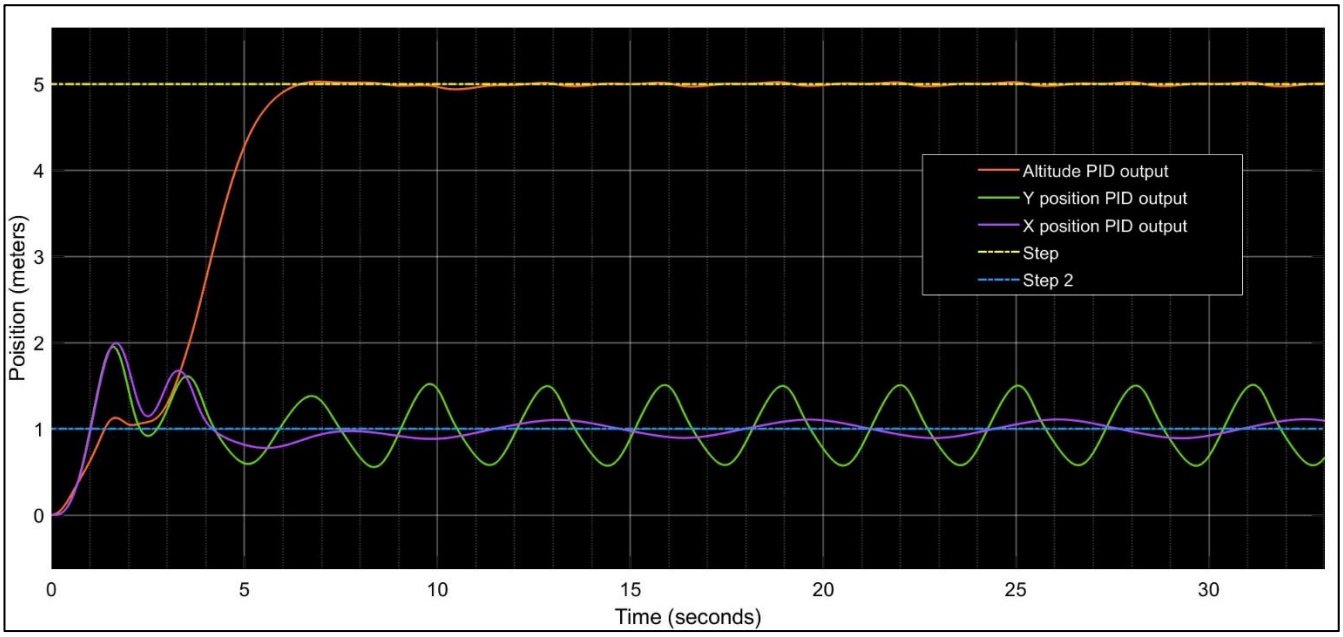


FIGURE 6: Step signal and PID controllers' response.

Then, the response of each controller was compared with the requirements and its performance was evaluated. The values are found in TABLE 3.

TABLE 3: Requirements and performance comparison of PID

	Rise Time [s]	Settling time [s]	Max Overshot [m]
Z-position requirement	5	7.5	1
Z-position output	4.734	6.004	0.026
X,Y-position requirement	2	X	1
X-position output	0.927	X	0.992
Y-position output	0.914	X	0.943

The responses of the controllers fell within the established parameters, and the controllers reached their desired positions, hence indicating successful tunning.

4.3.- LQR Methodology

4.3.1.- LQR functioning

This section studies the functioning of the LQR controller. First, the LQR control problem is shown in FIGURE 7:

LQR control problem is given as follows:

$$\begin{aligned} \underset{K}{\text{minimize}} \quad & J = \int_0^\infty \left[\mathbf{x}^T(t)Q\mathbf{x}(t) + \mathbf{u}(t)R\mathbf{u}(t)\right] dt \\ \text{subject to} \quad & \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \\ & \mathbf{x}(0) = \mathbf{x}_0 \\ & \mathbf{u}(t) = -K\mathbf{x}(t) \end{aligned}$$

where $Q \geq 0$ and $R > 0$, which are relative weights on the state and the control, respectively. The solution for the LQR control problem is given by

$$K = R^{-1}B^T P$$

where P is the solution of the following algebraic Riccati equation (ARE)

$$A^T P + PA - PBR^{-1}B^T P + Q = 0$$

FIGURE 7: LQR problem [25].

In order to solve the problem, 2 matrices – Q and R – apart from the state space model matrices A and B needed to be included, which follow the rules mentioned in FIGURE 7. These 2 matrices are diagonal matrices of dimensions $12x12$ and $4x4$ respectively, and they dictate the LQR tunning process.

Solving the presented problem required the solution of the matrix K presented in the previous figure, in addition to the resolution of the Algebraic Riccati Equation (ARE) [26] to derive matrix P. This process could be done step by step mathematically; however, MATLAB provides function 'lqr' [27] which simplified the process by directly computing K. The inputs of this function are matrices A, B, Q and R.

Matrix K then needed to be multiplied by vector x defined in equation (28), which could then be compared with the $x - y - z$ coordinates, and so the responses for U_i were obtained. Additionally, the disturbance $m * g$ mentioned in the state space model is integrated into the z coordinate.

4.3.2.- Simulink implementation: LQR block

The implementation of the LQR in Simulink is presented in FIGURE 8:

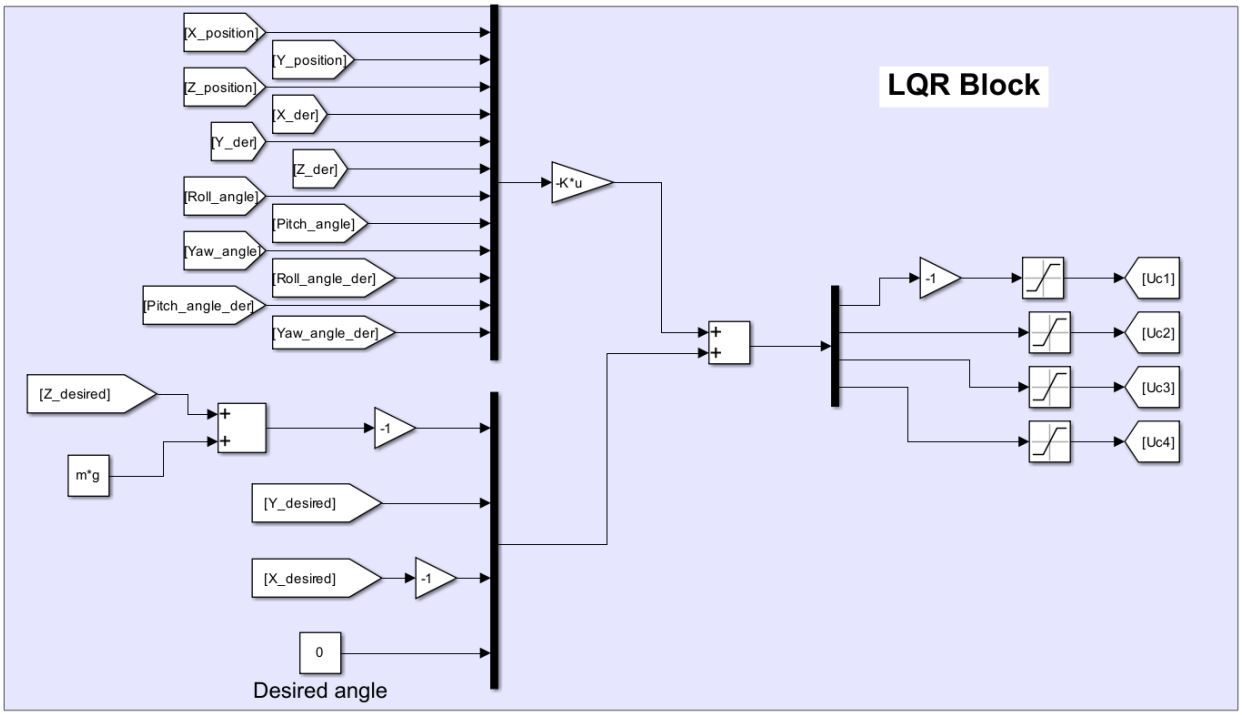


FIGURE 8: Simulink controller LQR block.

It can be seen the process explained in the previous section, where the product of $K * x$ was compared with the desired coordinates and yaw angle, and the system responses were obtained. Additionally, saturation limits were introduced as done in the PID case.

4.3.3.- LQR tunning

The process of LQR tunning is different from the PID controller [25]. Here, the controller was tuned by adjusting the values of the Q and R matrices. These 2 matrices are closely related, as increasing Q increases the convergence consuming more energy, whereas increasing R, which saves energy by slowing down the controller. To balance this, a value (sc) was assigned to Q, and $(1 - sc)$ was assigned to R. This means that increasing sc speeded up the convergence but more energy was used during the process.

Furthermore, different values of the Q and R diagonal were changed depending on the result needed. For example, in this case the third value of Q corresponding to z position and the first value of R corresponding on the elevation movement U_1 were reduced to avoid unnecessary energy consumption.

This tunning process involved trial and error to determine the most adequate values of Q and R for the mission's profile specified. In this case, 0.5 was the value selected for the general diagonal values and 0.4 was selected for the reduced values.

4.4.- PID results

Once the controller was correctly tunned, the Simulink script could be executed to check how it performed in the modelled environment and study the trajectory followed. The visualization is presented from an overhead view so the comparison with the path planning algorithm trajectory can be checked easier. The final trajectory is seen in FIGURE 9.

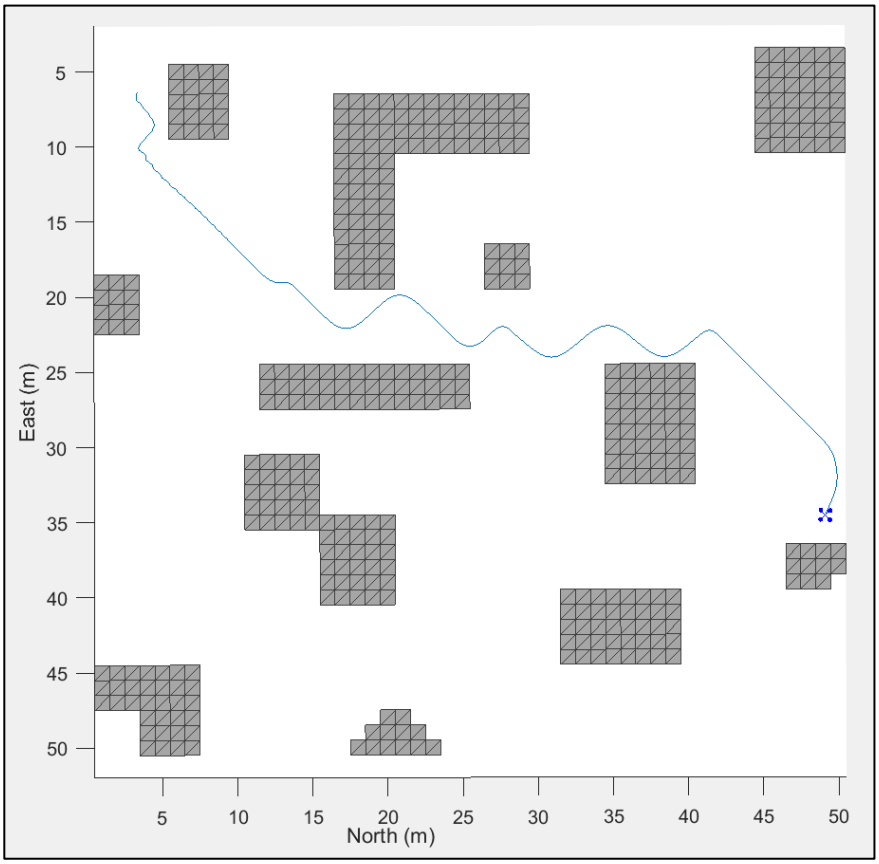


FIGURE 9: Trajectory followed by the quadcopter - PID.

The trajectory could be obtained in $x - y$ coordinates. Hence, 50 values equally spaced in time were obtained and compared with the waypoints vector obtained previously, and the mean deviation of the controller was calculated. Moreover, the mission time and the runtime of the Simulink script were derived. These results assessed both the controller efficiency and the computational resources needed to run it. Results are presented in TABLE 4.

TABLE 4: Mean deviation, Mission time and Runtime - PID

	Mean deviation			Mission time [s]	Runtime [s]
Controller	X position	Y position	Z position		
PID	10.75%	10.88%	2.76%	102.8	49.814

4.5.- LQR results

After completing the tuning process, the results from the LQR controller were obtained. Similar to the PID case, the trajectory followed by the quadcopter is presented from a top view for better comparison. Check FIGURE 10.

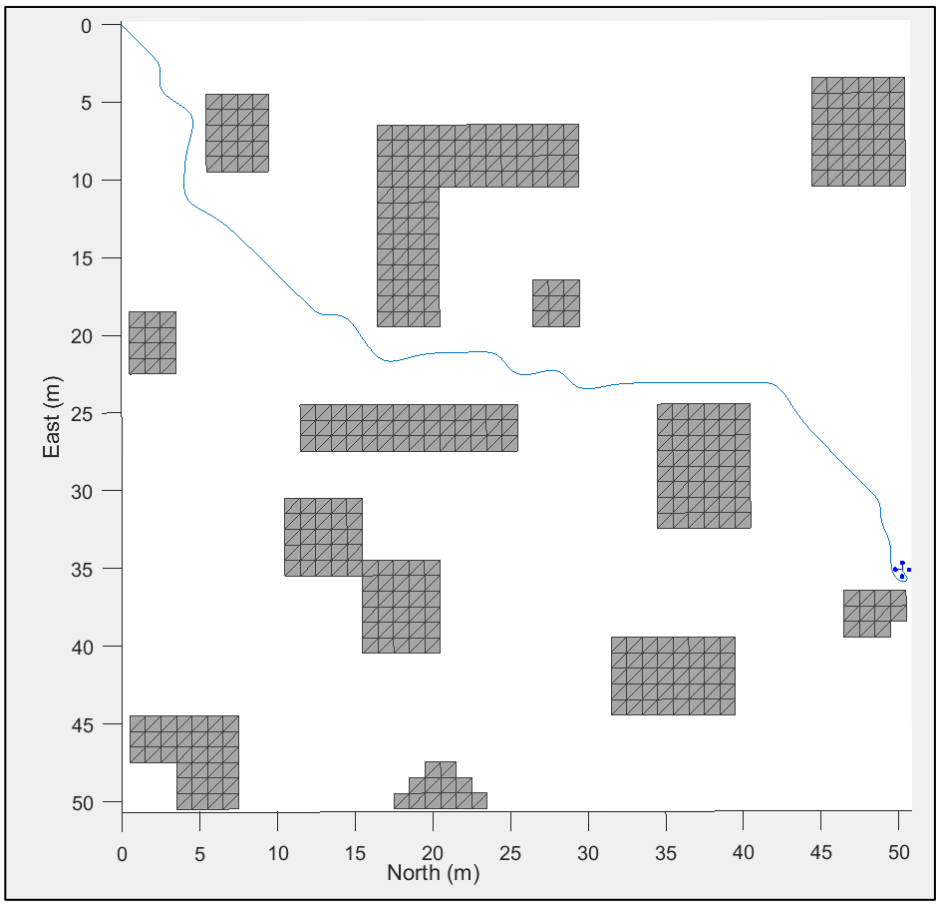


FIGURE 10: Trajectory followed by the quadcopter - LQR.

The same trajectory was obtained in $x - y$ coordinates as it was done previously, and the same thing occurs with the mission time and the script runtime. Results shown in TABLE 5.

TABLE 5: Mean deviation, Mission time and Runtime - LQR

	Mean deviation			Mission time [s]	Runtime [s]
Controller	X position	Y position	Z position		
LQR	1.69%	1.61%	3.10%	49.8	30.625

4.6.- Chapter discussion

Finally, the results obtained from both controllers were studied. First, their tunning and implementation were analysed.

Starting with the LQR, the state space model obtention involved an additional step compared with the PID, but its Simulink implementation and tunning processes were simpler, as it did not require any filter or safety measure and could be tunned easier. On the other hand, while PID equations were easier to obtain, its implementation was more complex due to the safety measures and tunning. However, alternative tuning methods may simplify the process, yet including additional understanding and effort.

Analysing the trajectories from both controllers, FIGURE 9 and FIGURE 10, it was seen that both controllers were not perfect and presented some errors when following the trajectory dictated. This was expected, as the controllers required to manage multiple variables and movements at the same time, and further tunning could be done to improve the result. However, both controllers successfully reached the final point without any collision.

Comparing both trajectories, the PID showed less accuracy than the LQR when reaching the desired coordinates, evident from visual inspecting the oscillations shown during the flight in FIGURE 9. Mathematically, TABLE 4 and TABLE 5 show that PID presented a mean deviation of around 10% in $x - y$ coordinates, while LQR managed to deviate only around 1% from the desired values.

Times required to finish the mission and to run the script were also analysed and presented in TABLE 4 and TABLE 5. Once again, the LQR outperformed the PID, needing half the time to complete the mission and requiring less computational effort than the PID to run. This last result was unexpected, as it would be predictable that the PID, being simpler than the LQR, took less computing time than its competitor; however, this suggested that the PID might need more effort in its tunning so less computational effort was required.

In summary, it can be stated that the LQR outperformed the PID in every operational aspect, demonstrating enhanced accuracy and efficiency than its competitor. In addition, both implementations could be assumed to be equally demanding, as the LQR required the state space model, but the PID tuning process was more laborious and time-consuming than expected.

Chapter.5 – Other Simulink blocks

5.1.- Introduction: Other Simulink blocks

Although the most important blocks were explained in previous chapters, there are still a few missing to be commented, as they were needed for the correct functioning of the program or offered interesting results to analyse. Hence, 2 more blocks are explained in this chapter.

5.2.- Visualization block

This block can be seen in APPENDIX.3 , and it controlled the visualization of the 3D map, quadcopter model and animations. The *UAV scenario scope* [28] and the *UAV scenario configuration* blocks [29] were needed for the correct importation and visualization of the 3D scenario and the quadcopter model, and required no inputs. On the other hand, the *UAV scenario motion write* block [30] changed the position and orientation of the quadcopter and showed the trajectory followed by it. Position, velocity, acceleration, and angular acceleration were the inputs, where orientation needed to be given in quaternion form [31].

5.3.- Energy consumption block

The *Energy consumption* block seen in APPENDIX.3 measured the power consumption in Watts during the mission. The torque produced by each motor and its rpm needed to be computed first, which was done using expression (3). Then, the rpm was converted to angular speed ω by multiplying it by $\pi/180$, hence the mechanical power of each rotor was computed knowing expression (34) obtained from [32].

$$P_{Mech} = Torque * \omega \tag{34}$$

Assuming a rotor efficiency of 70% [33], the electrical power was obtained knowing expression (35):

$$P_{Elec} = \frac{P_{Mech}}{Efficiency} \tag{35}$$

Finally, summing the electrical power consumption of the 4 motors, the final power consumption of the UAV was obtained.

5.3.1.- Energy consumption block results

Following the previous process, the following graphs were obtained for the PID and the LQR respectively during a constant level flight section of the mission (second 10 to 40), where the power consumption was measured in Watts. Check FIGURE 11.

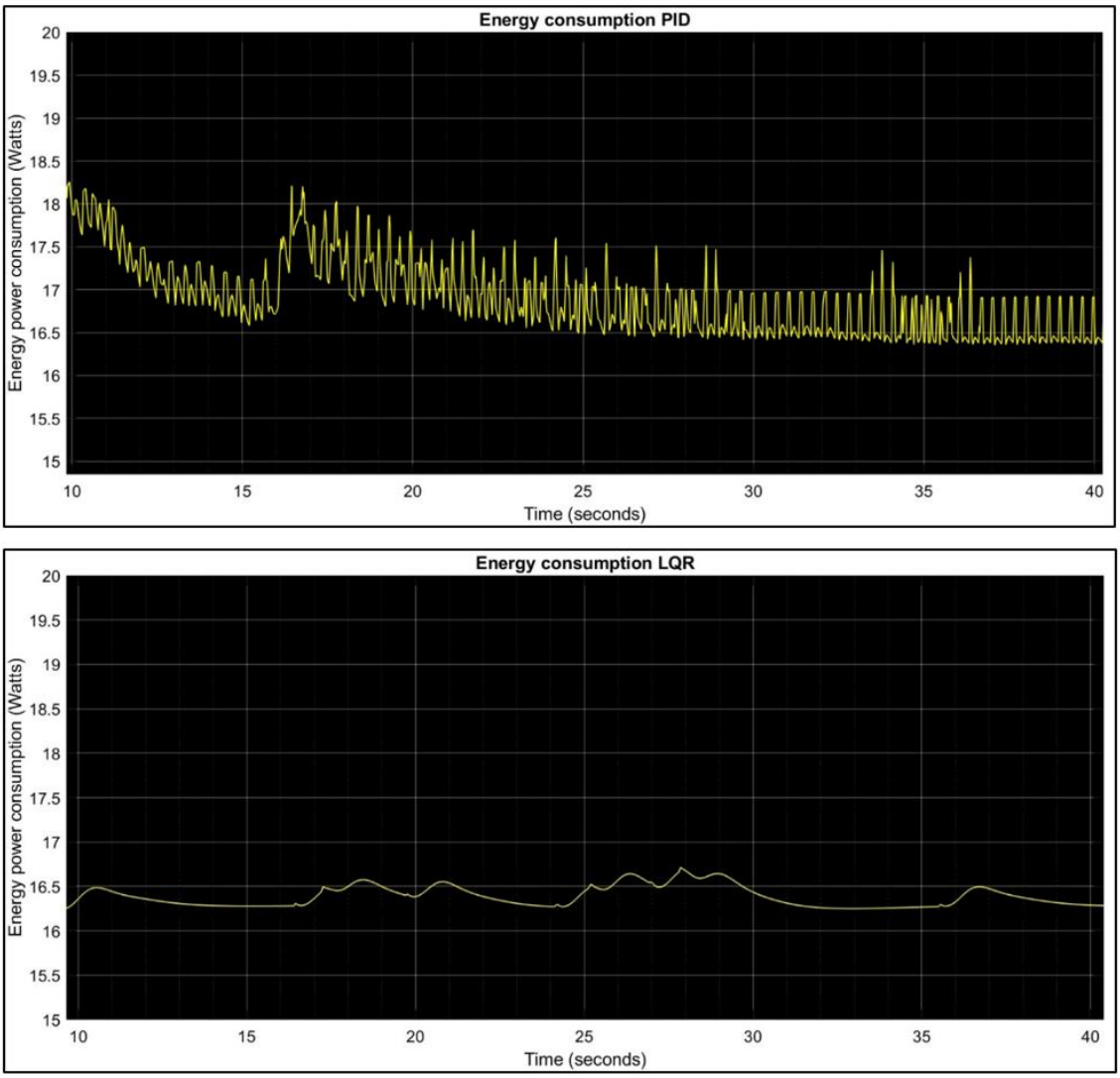


FIGURE 11: Energy consumption of PID (top) and LQR (bottom).

5.4.- Chapter discussion

In this chapter, 2 important blocks: the *Visualization* and the *Energy consumption* blocks were presented. The first one, was essential for visually comparing quadcopter trajectories. On the other hand, the *Energy consumption* block presented a valuable result to analyse. It needs to be said that the electrical power consumption of the quadcopter ranged inside the order of magnitude expected from a UAV of this characteristics [34].

Comparing both controllers both exhibited similar consumption per second, approx. 16 Watts. However, the LQR controller demonstrated more stable energy consumption with less oscillations than the PID. This was expected, as the behaviour of the PID presented before was more erratic than the LQR. However, considering total mission time, the LQR presented an overall lower energy consumption, as it finished the mission faster.

Therefore, although both controllers have similar energy consumption per second, the LQR was more stable and less noisy, and consumed less total energy than the PID.

Chapter.6 – Conclusion

6.1.- Achievements

- Valuable literature was obtained for the understanding and completion of the different chapters of the project.
- The non-linear dynamic model and the state space model of the quadcopter platform were mathematically obtained, and later implemented in Simulink.
- Different ways of creating and navigating the scenario were analysed, selecting, and implementing the A* path planning algorithm in MATLAB and Simulink as the system to create the desired trajectory.
- Multiple controllers were analysed and PID and LQR were selected as the more adequate ones given the paper context. These controllers were implemented and tested, and valuable results were obtained from them.
- Results from the previous controllers were analysed and compared and the LQR controller was selected as the most optimal and effective choice for completing the mission.

6.2.- Discussion

In this section, a general and concluding view of the work done during the project and the results obtained is presented. Moreover, critical thinking about the performance and industry application of the controllers is assessed.

Initially, a MATLAB and Simulink scripts which construct a digital urban scenario were created. Moreover, the script was able to make a physical representation of a quadcopter and direct the model safely through the map thanks to the path planning algorithm. This script provided a base for further development on control systems. What is more, it can serve as a template to create multiple other environments and perform other types of missions and tasks.

Then, the complexity or facility to implement the controllers in Simulink was studied. As it was commented before, the PID is simpler in its dynamic model, however LQR problem was easier to solve and implement thanks to the tools and functions provided by MATLAB and Simulink. Then, the implementation process resulted in a draw between both controllers in terms of complexity.

Reviewing the results obtained from Chapters 4 and 5, a clear winner emerged in this competition. The LQR outperformed the PID in every aspect: completing the mission faster, requiring less computational effort, consuming the same energy per second than the PID, but less considering the total consumption, and showing improved path-following capabilities.

Overall, the LQR demonstrated greater stability and accuracy than the PID. Nevertheless, it is important to highlight that both controllers, albeit having different performance, completed the mission successfully. In addition, PID could be highly benefited from more effort spent during the tunning process and studying if additional filters or safety measures were required. However, this tunning has been proved to be more laborious and complex than expected. Despite the less performance shown in this paper, PID's popularity in industry can be attributed to its versatility and suitability for many industrial applications.

In summary, LQR is the preferred choice for optimal results, but PID's versatility and cost-effectiveness make it a suitable selection for general duties. However, more exploration in tunning methods would improve the results obtained for both controllers, especially for PID.

6.3.- Conclusions

Based on the previous discussion, there are multiple evidence that make the LQR the preferred choice for the mission specified, thanks to its superior performance compared to the PID. However, both controllers completed the mission successfully, despite its gap in performance. This analysis highlights the importance of selecting the appropriate controller, depending on the missions' requirements. If accuracy and efficiency is needed, LQR is the choice. However, if adequate results are required for a wide range of industrial duties, although LQR is still a suitable election, PID offers great versatility and simplicity.

Furthermore, the discussion remarked the importance of tunning processes, especially for the PID, which would be highly benefited from it, reducing the gap between controllers.

In conclusion, this paper has provided valuable insights into control systems and their implementation. Thanks to this, other engineers can understand the functioning of multiple control systems and they can make decisions whether which system is more optimal to implement depending on the requirements of the task.

6.4.- Future work

- Implement 3D planning algorithms [35], so an optimised trajectory could be computed, facilitating navigation through varying heights and enhanced obstacle avoidance.
- Introduce moving obstacles to evaluate the quadcopter's response. However, it would require constant updates of the 3D map provided, or that inflight sensors and instruments were implemented to the UAV for obstacle detection.
- Further tunning of the PID and the LQR to analyse the limitations of each controller.
- Evaluate other types of controllers: MPC, hybrid controllers or different PID configurations.

References

[1] Waliszkiewicz, M., Wojtowicz, K., Rochala, Z. and Balestrieri, E. (2020). The Design and Implementation of a Custom Platform for the Experimental Tuning of a Quadcopter Controller. *Sensors*, 20(7), p.1940. Doi: HTTPs://doi.org/10.3390/s20071940.

[2] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, USA, 2007, pp. 153-158, Doi: 10.1109/IROS.2007.4399042.

[3] Ahmad, F., Kumar, P., Bhandari, A. and P. Patil, P. (2018). *Simulation of the Quadcopter Dynamics with LQR based Control*. [online] ScienceDirect. Available at: https://acortar.link/q4Hauk [Accessed 18 Nov. 2023]. Department of Mechanical Engineering. pp 327 - 329.

[4] JOUAV. (2023). *The Ultimate Guide to Heavy Lift Drone Motors*. [online] Available at: https://www.jouav.com/blog/heavy-lift-drone-motors.html#:~:text=For%20small%20toy%20drones%2C%20the [Accessed 30 Apr. 2024].

[5] www.dronefromchina.com. (n.d.). *What is RPM means of drone motor?* [online] Available at: https://www.dronefromchina.com/new/how-to-understand-drone-motor-rpm.html#:~:text=Generally%2C%20the%20RPM%20of%20a [Accessed 30 Apr. 2024].

[6] Drone Safe Register News and Blog. (n.d.). *What are The Positive Uses of Drones*. [online] Available at: https://dronesaferegister.org.uk/blog/amazing-drone-uav-uses.

[7] Rashid, A.T., Ali, A.A., Frasca, M. and Fortuna, L. (2013). Path planning with obstacle avoidance based on visibility binary tree algorithm. *Robotics and Autonomous Systems*, 61(12), pp.1440–1449. Doi: HTTPs://doi.org/10.1016/j.robot.2013.07.010.

[8] Anon, (n.d.). *3DCityDB Database – Homepage*. [online] Available at: https://www.3dcitydb.org/3dcitydb/.

[9] Javaid, A. (2013). *Understanding Dijkstra's Algorithm*. [online] papers.ssrn.com. Available at: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2340905

[10] Lin, M., Yuan, K., Shi, C. and Wang, Y. (2017). *Path planning of mobile robot based on improved A* algorithm*. ResearchGate.

[11] Noreen, I., Khan, A., & Habib, Z. (2016). A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*(10), 20.

[12] uk.mathworks.com. (n.d.). *A* path planner for grid map - MATLAB - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/nav/ref/plannerastargrid.html [Accessed 30 Apr. 2024].

[13] uk.mathworks.com. (n.d.). *Follow waypoints for UAV - Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/uav/ref/waypointfollower.html [Accessed 30 Apr. 2024].

[14] Borase, R.P., Maghade, D.K., Sondkar, S.Y. *et al.* A review of PID control, tuning methods and applications. *Int. J. Dynam. Control* **9**, 818–827 (2021). https://doi.org/10.1007/s40435-020-00665-4

[15] Afram, A. and Janabi-Sharifi, F. (2014). Theory and applications of HVAC control systems – A review of model predictive control (MPC). *Building and Environment*, 72, pp.343–355. Doi: HTTPs://doi.org/10.1016/j.buildenv.2013.11.016.

[16] Utem.edu.my. (2024). *View of Model of Linear Quadratic Regulator (LQR) Control System in Waypoint Flight Mission of Flying Wing UAV*. [online] Available at: https://jtec.utem.edu.my/jtec/article/view/5696/4011 [Accessed 1 May 2024].

[17] Ahmad, F., Kumar, P., Bhandari, A. and P. Patil, P. (2018). *Simulation of the Quadcopter Dynamics with LQR based Control*. [online] ScienceDirect. Available at: https://acortar.link/q4Hauk [Accessed 18 Nov. 2023]. Department of Mechanical Engineering. pp 327 - 329.

[18] 4] R. Kumar, M. Dechering, A. Pai, A. Ottaway, M. Radmanesh and M. Kumar, "Differential flatness based hybrid PID/LQR flight controller for complex trajectory tracking in quadcopter UAVs," 2017 IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 2017, pp. 113-118, Doi: 10.1109/NAECON.2017.8268755.

[19] P. Wang, Z. Man, Z. Cao, J. Zheng and Y. Zhao, "Dynamics modelling and linear control of quadcopter," 2016 International Conference on Advanced Mechatronic Systems (ICAMechS), Melbourne, VIC, Australia, 2016, pp. 498-503, Doi: 10.1109/ICAMechS.2016.7813499.

[20] Angélico, B.A., Campanhol, L.B.G. and Oliveira da Silva, S.A. (2014). Proportional–integral/proportional–integral-derivative tuning procedure of a single-phase shunt active power filter using Bode diagram. *IET Power Electronics*, 7(10), pp.2647–2659. Doi: HTTPs://doi.org/10.1049/iet-pel.2013.0789.

[21] Meshram, P.M. and Kanojiya, R.G. (2012). *Tuning of PID controller using Ziegler-Nichols method for speed control of DC motor*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6216102.

[22] Joseph, E. (2017). Cohen-coon PID Tuning Method; A Better Option to Ziegler Nichols-PID Tuning Method. *International Journal of Recent Engineering Research and Development (IJRERD)* ||, [online] 02(11), pp.141–145. Available at: https://eprints.federalpolyilaro.edu.ng/1247/1/Cohen-Coon.pdf [Accessed 1 May 2024].

[23] uk.mathworks.com. (n.d.). *Continuous-time or discrete-time PID controller - Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/simulink/slref/pidcontroller.html.

[24] Engineering LibreTexts. (2020). *4.2: Transient Response Improvement*. [online] Available at: https://goo.su/on0KDfi [Accessed 1 May 2024].

[25] Jongrae, Kim (n.d.). *Spacecraft Dynamics & Control MECH 5900M. Lecture slides week 5*. [online] University of Leeds. Available at: https://minerva.leeds.ac.uk/ultra/institution-page [Accessed 1 May 2024].

[26] Luther, W. and Otten, W. (1999). Verified calculation of the solution of algebraic Riccati equation. *Springer eBooks*, pp.105–118. Doi: HTTPs://doi.org/10.1007/978-94-017-1247-7_8.

[27] uk.mathworks.com. (n.d.). *Linear-Quadratic Regulator (LQR) design - MATLAB lqr - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/control/ref/lti.lqr.html.

[28] uk.mathworks.com. (n.d.). *Visualize UAV scenario and lidar point clouds - Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/uav/ref/uavscenarioscope.html [Accessed 1 May 2024].

[29] uk.mathworks.com. (n.d.). *Configure and simulate UAV scenarios - Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/uav/ref/uavscenarioconfiguration.html [Accessed 1 May 2024].

[30] uk.mathworks.com. (n.d.). *Update platform motion in UAV scenario simulation - Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/uav/ref/uavscenariomotionwrite.html [Accessed 1 May 2024].

[31] Euclideanspace.com. (2017). *Maths -Quaternion Transforms - Martin Baker*. [online] Available at: https://www.euclideanspace.com/maths/algebra/realNormedAlgebra/quaternions/transforms/index.htm.

[32] www.futek.com. (n.d.). *Electric Motor Output Power | How to measure the power output of an electric motor*. [online] Available at: https://www.futek.com/Electric-Motor-Output-Power.

[33] The energy savings network DETERMINING ELECTRIC MOTOR LOAD AND EFFICIENCY Figure 1 Motor Part-Load Efficiency (as a Function of % Full-Load Efficiency). (n.d.). Available at: https://www.energy.gov/eere/amo/articles/determining-electric-motor-load-and-efficiency#:~:text=Most%20electric%20motors%20are%20designed.

[34] Hasini, V., Abeywickrama, Beeshanga, A., Jayawickrama, Y., He, E. and Dutkiewicz (n.d.). *Empirical Power Consumption Model for UAVs*. [online] Available at: https://opus.lib.uts.edu.au/rest/bitstreams/30e4d4a8-2d55-4be5-b262-a21b0a8f6972/retrieve [Accessed 1 May 2024].

[35] Yuanhao, H., Shi, H., Hao, W. and Ruifeng, M. (n.d.). *XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE 3D Path Planning and Obstacle Avoidance Algorithms for Obstacle-Overcoming Robots*. [online] Available at: https://arxiv.org/pdf/2209.00871#:~:text=The%20path%20planning%20algorithm%20provides [Accessed 1 May 2024].

Appendices

Appendix.1 – Extra calculations

Appendix.1.1 - State space model obtention

$$\{1\}\quad \ddot{X} = \frac{1}{m} * (c\phi c\psi s\theta + s\phi s\psi) * U_1 \rightarrow small\ angle\ approx \begin{Bmatrix} c\alpha = 1 \\ s\alpha = alpha \end{Bmatrix} \rightarrow$$

$$\rightarrow \ddot{X} = \frac{1}{m} * (\theta + \phi\psi) * U_1 \rightarrow \psi = 0 \rightarrow \ddot{X} = \frac{U_1}{m} * \theta \rightarrow U_1 = m * g \rightarrow \ddot{X} = \frac{m * g}{m} * \theta = g * \theta$$

$$\{2\}\quad \ddot{Y} = \frac{1}{m} * (c\phi s\psi s\theta + s\phi c\psi) * U_1 \rightarrow small\ angle\ approx \begin{Bmatrix} c\alpha = 1 \\ s\alpha = alpha \end{Bmatrix} \rightarrow$$

$$\rightarrow \ddot{X} = \frac{1}{m} * (\psi\theta + \phi) * U_1 \rightarrow \psi = 0 \rightarrow \ddot{X} = \frac{U_1}{m} * \phi \rightarrow U_1 = m * g \rightarrow \ddot{X} = \frac{m * g}{m} * \phi = g * \phi$$

$$\{3\}\quad \ddot{Z} = \frac{1}{m} * (c\phi c\theta) * U_1 - g \rightarrow small\ angle\ approx \begin{Bmatrix} c\alpha = 1 \\ s\alpha = alpha \end{Bmatrix} \rightarrow$$

$$\rightarrow \ddot{Z} = \frac{1}{m} * U_1 - g \rightarrow g\ taken\ as\ disturbance \rightarrow \ddot{Z} = \frac{U_1}{m}$$

$$\{4\}\quad \ddot{\phi} = \frac{l}{I_x} * U_2 + \dot{\theta} * \dot{\psi} * \frac{(I_y - I_z)}{I_x} \rightarrow \dot{\phi} = \dot{\theta} = \dot{\psi} = 0 \rightarrow \ddot{\phi} = \frac{l}{I_x} * U_2$$

$$\{5\}\quad \ddot{\theta} = \frac{l}{I_y} * U_3 + \dot{\phi} * \dot{\psi} * \frac{(I_z - I_x)}{I_y} \rightarrow \dot{\phi} = \dot{\theta} = \dot{\psi} = 0 \rightarrow \ddot{\theta} = \frac{l}{I_y} * U_3$$

$$\{6\}\quad \ddot{\psi} = \frac{1}{I_z} * U_4 + \dot{\phi} * \dot{\theta} * \frac{(I_x - I_y)}{I_z} \rightarrow \dot{\phi} = \dot{\theta} = \dot{\psi} = 0 \rightarrow \ddot{\phi} = \frac{4}{I_z} * U_4$$

Appendix 1.2.- Body frame to inertial frame transformation

Knowing that:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [R] \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}$$

From the matrix calculation, the following equations were obtained:

$$\begin{matrix} x = -c\psi s\theta c\phi - s\psi s\phi \\ y = -s\psi s\theta c\phi + c\psi s\phi \end{matrix} \rightarrow \psi = 0 \rightarrow \begin{matrix} \{1\} \\ \{2\} \end{matrix}$$

$$\{1\}\quad x = -s\theta c\phi \rightarrow \theta = -\arcsin\left(\frac{x}{\cos\arcsin y}\right) = -\arcsin\left(\frac{x}{\cos\phi}\right)$$

$$\{2\}\quad y = s\phi \rightarrow \phi = \arcsin y$$

Appendix.2 – MATLAB code

```
%% Variable definition
% This section defines the necessary parameters for the functioning of the
% Simulink model

g = 9.81;                   % Gravity cte
m = 2.5;                    % Quadcopter mass
l = 0.2;                    % Length of each arm
Ix = 0.11;                  % Moments of inertia
Iy = 0.11;                  %
Iz = 0.04;                  %
b = 3*10^-6;                % Force coefficient
d = 4*10^-9;                % Moment coefficient

%% Max Force and Torque calculation
% Calculation of the maximum available force and torque given the maximum
% velocity of an standard motor

rpm = 15000*0.10472;   % max rpm
max_u1 = b*(4*rpm^2);
max_u2 = b*(rpm^2);
max_u3 = b*(rpm^2);
max_u4 = d*(2*rpm^2);

eff = 0.7;             % motor efficiency

q0 = [0 0 0 1];        % quaternion initial pos

%% Map creation
% This section creates a binary grid which functions as the map the path
% planning algorithm is trying to solve. Obstacles are denoted by a 1 and
% free obstacles paths are denoted by a 0

map = [0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
       0 0 0 0 1 1 1 1 1 1  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;

       0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
       0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
```

```
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  1 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;

    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  1 1 1 1 1 1 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;

    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 1 1 1
1 1 1 1  1 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
    0 0 0 0 0 0 0 0 0 1  1 1 1 1 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 1 1 1 1 1;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 1 1 1 1 1;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 1 1 1 1 1;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 1 1 1 1 1;
    0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1 1 1 1 1  1 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 1 1 1 1 1;
```

```
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 1 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 1 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 1 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 1 0 0  0 0 0 0 0 0 0 0 0 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 1 0 0  0 0 0 0 0 0 0 0 1 1  1 1 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 1 0 0  0 0 0 0 0 0 0 1 1 1  1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 1 0 0  0 0 0 0 0 0 1 1 1 1  1 1 1 1 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;];

% The variable mapLQR represents the same map as the one which is used
% by the path planning algorithm but with a decrease in 1 side in the
% first and last row and columns of each obstacle.
% This is done so the controllers have more room to manoeuvre as the
% planning algorithm path is too closed to the walls.


mapLQR = [0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0
0 0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 1 1 1 1 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 1 1 1 1 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 1 1 1 1 0  0 0 0 0 0 0 1 1 1 1  1 1 1 1 1 1 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 1 1 1 1 0  0 0 0 0 0 0 1 1 1 1  1 1 1 1 1 1 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 1 1 1 1 0  0 0 0 0 0 0 1 1 1 1  1 1 1 1 1 1 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  1 1 1 1 1 1 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 1 1 1 1 1 1;

      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
```

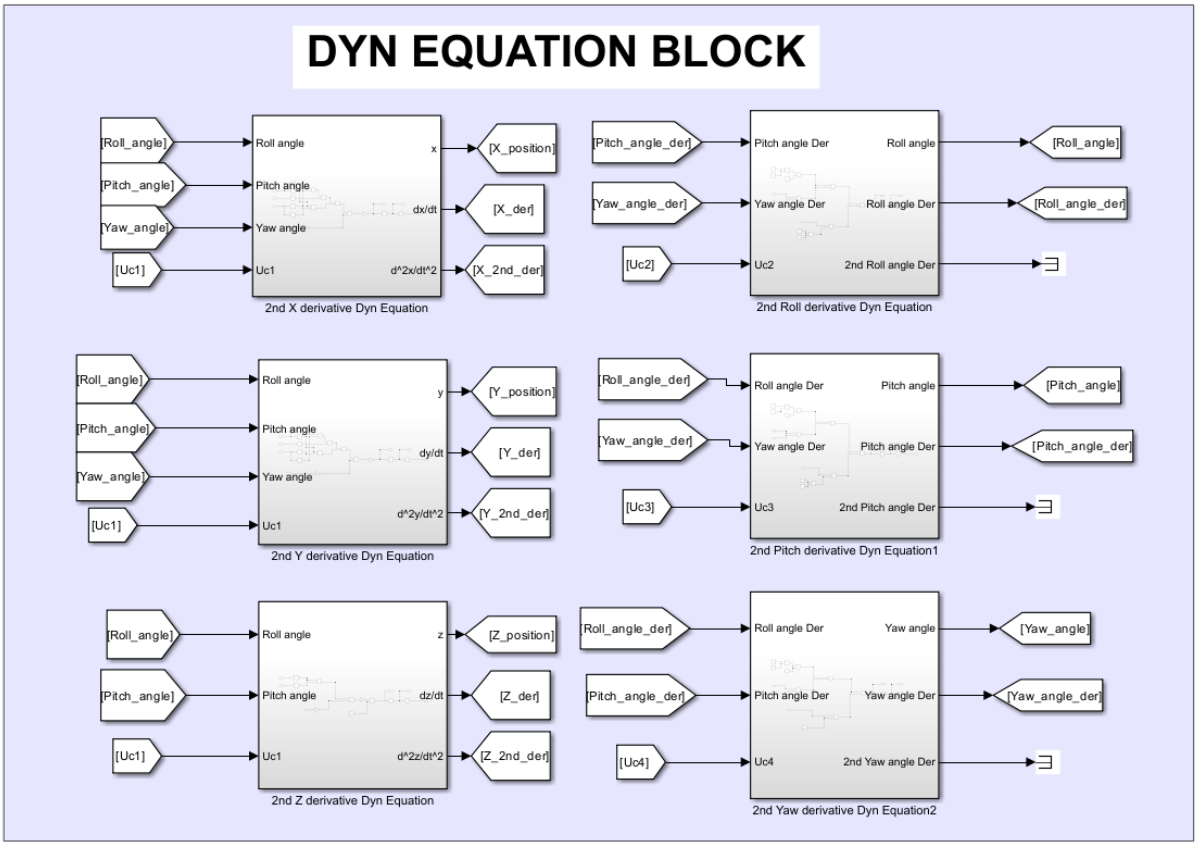
```
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1  0 0 0 0 0 0 1 1 1 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;

      1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1 1 1 1 1  1 1 1 1 1 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1 1 1 1 1  1 1 1 1 1 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1 1 1 1 1  1 1 1 1 1 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;

      0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 1 1
1 1 1 1  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  1 1 1 1 1 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 1 1 1 1;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 1 1 1 1;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 1 1 1 1;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 1 1 1 1 1  0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1
1 1 1 0  0 0 0 0 0 0 0 0 0 0;

      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1
1 1 1 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1
1 1 1 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1
1 1 1 0  0 0 0 0 0 0 0 0 0 0;
```

```
      0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 1 1 1 1 1
1 1 1 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      1 1 1 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 0 1  1 0 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 1 1  1 1 0 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;
      0 0 0 1 1 1 1 0 0 0  0 0 0 0 0 0 0 0 1 1 1  1 1 1 0 0 0 0 0 0 0  0 0 0 0 0 0
0 0 0 0  0 0 0 0 0 0 0 0 0 0;];

%% Path planning
% This section uses a A* path planning algorithm to find the correct
% waypoints to reach a designated goal

planner = plannerAStarGrid(binaryOccupancyMap(map)); % A plannerASatrGrid object
is created so
% later a trajectory can be plotted

start = [1 1];              % Define starting point
goal = [35 50];             % Define goal

path = plan(planner, start, goal); % Plan the path from the start and goal
% points assigned

dimPath = size(path);

%% Define scenario %%
% This section creates the virtual scenario where the drone will flight
% through the obstacles to reach its final destination.

Scenario = uavScenario("UpdateRate",100,"ReferenceLocation",[0 0 0]);

[row, col] = find(mapLQR == 1);

i = 1;
dim = size(row);

% First define the position of the obstacles in the variable
% Obstaclepositions

while i < dim(1)
    ObstaclePositions(i,1) = [row(i)];
    ObstaclePositions(i,2) = [col(i)];
    i = i+1;
end

ObstaclesWidth = 1;                         % Width of the obstacles
i = 1;

% Contruct the obstacles as rectangles with a given height and a width
% equal to 1.

for i = 1:size(ObstaclePositions,1)
    ObstacleHeight = 20;
```

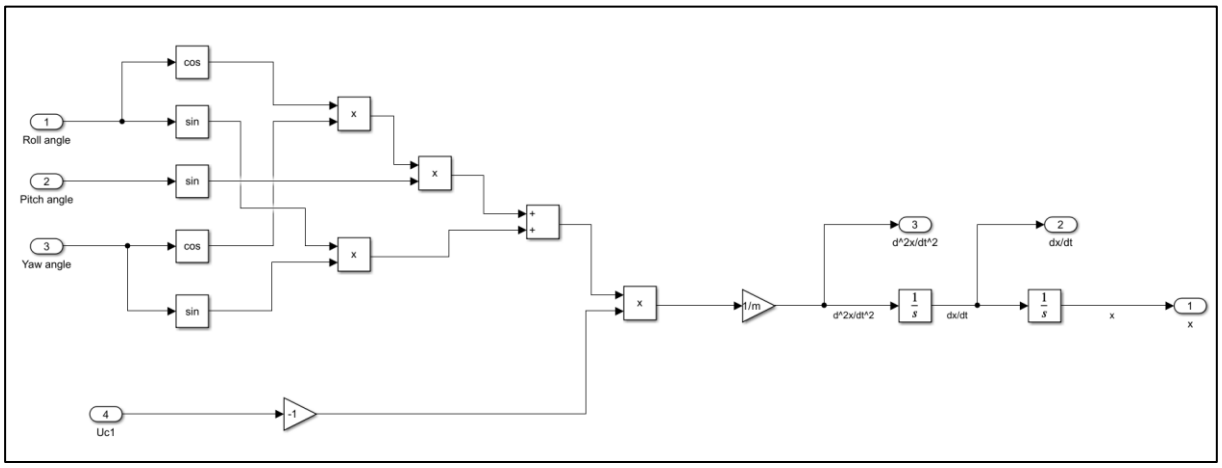
```
    addMesh(Scenario,"polygon", ...
        {[ObstaclePositions(i,1)-ObstaclesWidth/2 ObstaclePositions(i,2)-
ObstaclesWidth/2; ...
        ObstaclePositions(i,1)+ObstaclesWidth/2 ObstaclePositions(i,2)-
ObstaclesWidth/2; ...
        ObstaclePositions(i,1)+ObstaclesWidth/2
ObstaclePositions(i,2)+ObstaclesWidth/2; ...
        ObstaclePositions(i,1)-ObstaclesWidth/2
ObstaclePositions(i,2)+ObstaclesWidth/2], ...
        [0 ObstacleHeight]},0.651*ones(1,3));
end

%% Define UAV platform and initial positions %%
% This section constructs the UAV platform and the UAV initial position.

InitialPosition = [0 0 0];
InitialOrientation = [0 0 0];

% Define the reference frame, initial position and initial orientation.

platUAV =
uavPlatform("UAV",Scenario,"ReferenceFrame","NED","InitialPosition",InitialPositio
n,"InitialOrientation",eul2quat(InitialOrientation));

% Define platform of the UAV.

updateMesh(platUAV,"quadrotor",{1.2},[0 0 1],eul2tform([0 0 pi]));

%% Define a set of waypoints %%
% This section defines the waypoints the UAV will need to follow as a
% matrix of x y z

vector = linspace(10,10,dimPath(1));
vector(end) = 0;

Waypoints = [path transpose(vector)];

show3D(Scenario);
legend("Start Position","Obstacles")

%% LQR control gain
% This section solves the LQR problems for minimizing K, where matrices A B
% Q and R are obtained

A = [0 0 0  1 0 0  0 0 0  0 0 0;
     0 0 0  0 1 0  0 0 0  0 0 0;
     0 0 0  0 0 1  0 0 0  0 0 0;

     0 0 0  0 0 0  0 -g 0  0 0 0;
     0 0 0  0 0 0  g 0 0  0 0 0;
     0 0 0  0 0 0  0 0 0  0 0 0;

     0 0 0  0 0 0  0 0 0  1 0 0;
     0 0 0  0 0 0  0 0 0  0 1 0;
     0 0 0  0 0 0  0 0 0  0 0 1;

     0 0 0  0 0 0  0 0 0  0 0 0;
     0 0 0  0 0 0  0 0 0  0 0 0;
     0 0 0  0 0 0  0 0 0  0 0 0;];
```

```
B = [0 0 0 0;
     0 0 0 0;
     0 0 0 0;

     0 0 0 0;
     0 0 0 0;
     1/m 0 0 0;

     0 0 0 0;
     0 0 0 0;
     0 0 0 0;

     0 l/Ix 0 0;
     0 0 l/Iy 0;
     0 0 0 1/Iz;];

% If we want the simulation to converge faster, increment sc means more
% energy needed at the controller

sc = 0.5;
q_val = sc;
r_val = 1-sc;

q_val_z = 0.4;
r_val_z = 0.4;

Q = [q_val 0 0  0 0 0  0 0 0  0 0 0;
     0 q_val 0  0 0 0  0 0 0  0 0 0;
     0 0 q_val_z  0 0 0  0 0 0  0 0 0;

     0 0 0  q_val 0 0  0 0 0  0 0 0;
     0 0 0  0 q_val 0  0 0 0  0 0 0;
     0 0 0  0 0 q_val  0 0 0  0 0 0;

     0 0 0  0 0 0  q_val 0 0  0 0 0;
     0 0 0  0 0 0  0 q_val 0  0 0 0;
     0 0 0  0 0 0  0 0 q_val  0 0 0;

     0 0 0  0 0 0  0 0 0  q_val 0 0;
     0 0 0  0 0 0  0 0 0  0 q_val 0;
     0 0 0  0 0 0  0 0 0  0 0 q_val;];

R = [r_val_z 0 0 0;
     0 r_val 0 0;
     0 0 r_val 0;
     0 0 0 r_val;];

K = lqr(A,B,Q,R);
```

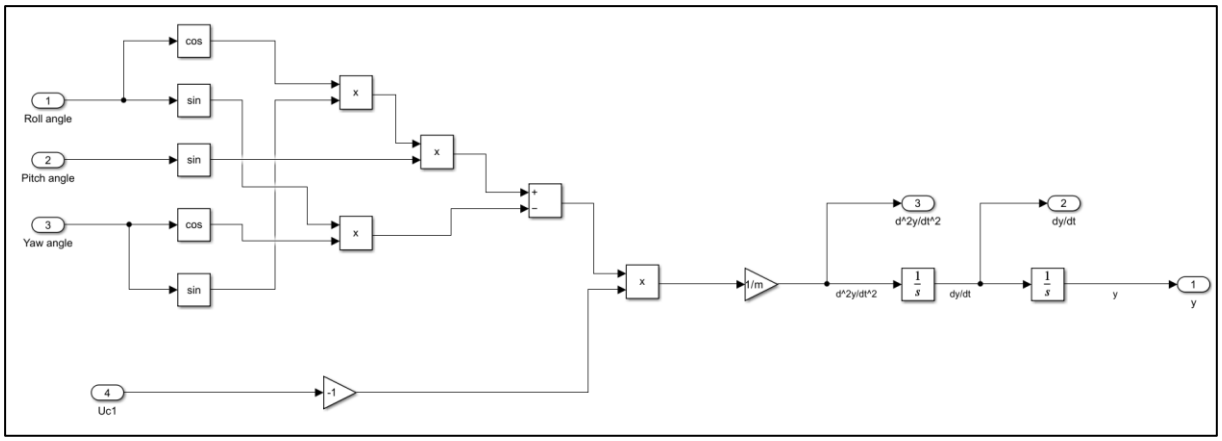
Appendix.3 – Simulink blocks



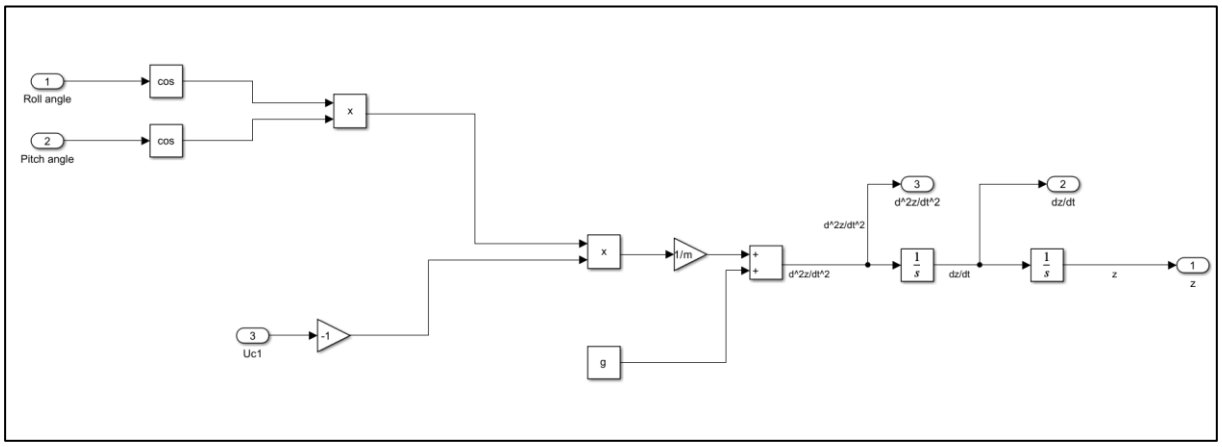
APPENDIX.3 FIGURE 1: Dynamic equations block



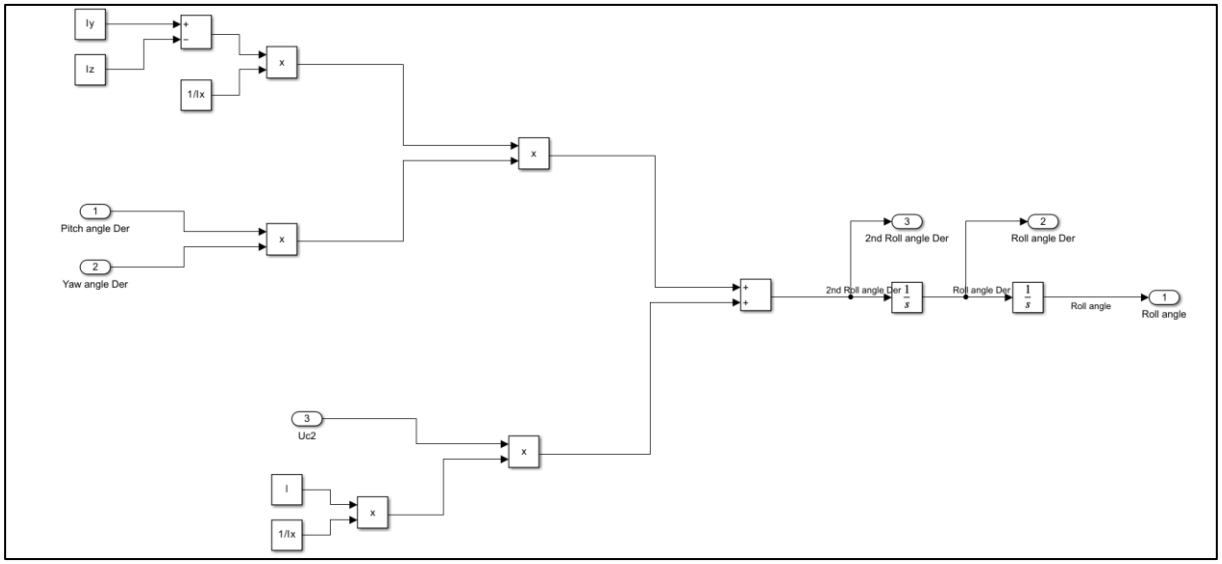
APPENDIX.3 FIGURE 2: 2nd X derivative subblock



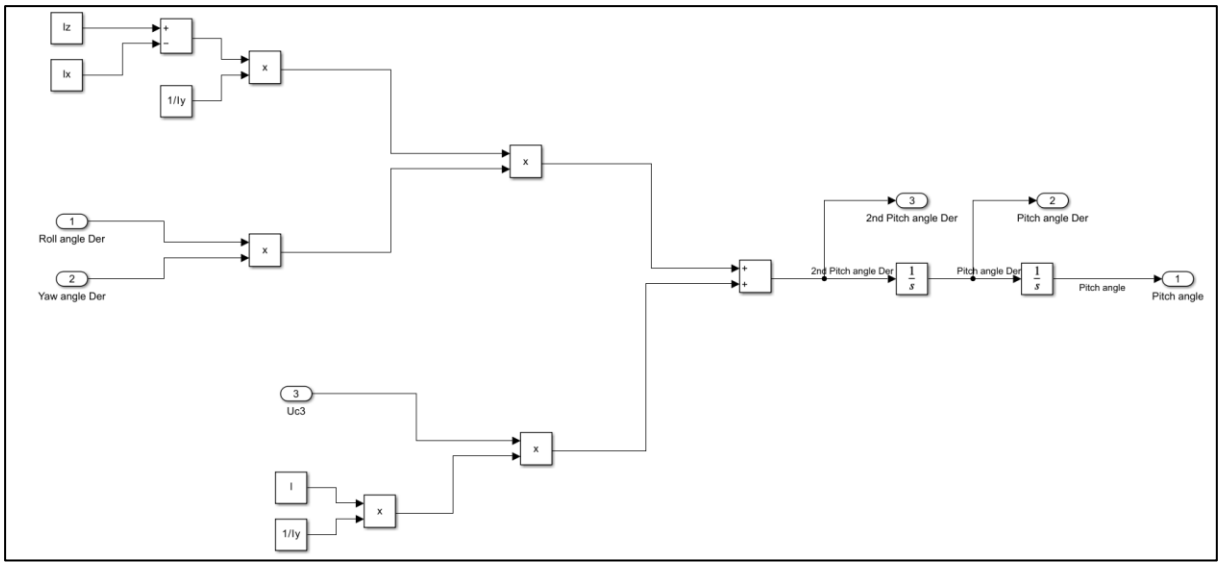
APPENDIX.3 FIGURE 3: 2nd Y derivative subblock



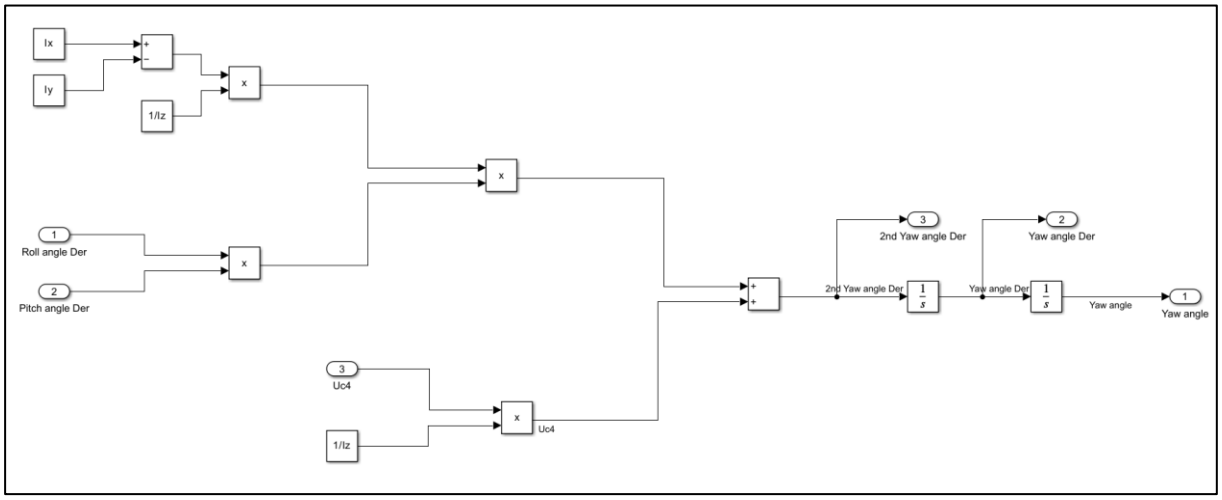
APPENDIX.3 FIGURE 4: 2nd Z derivative subblock



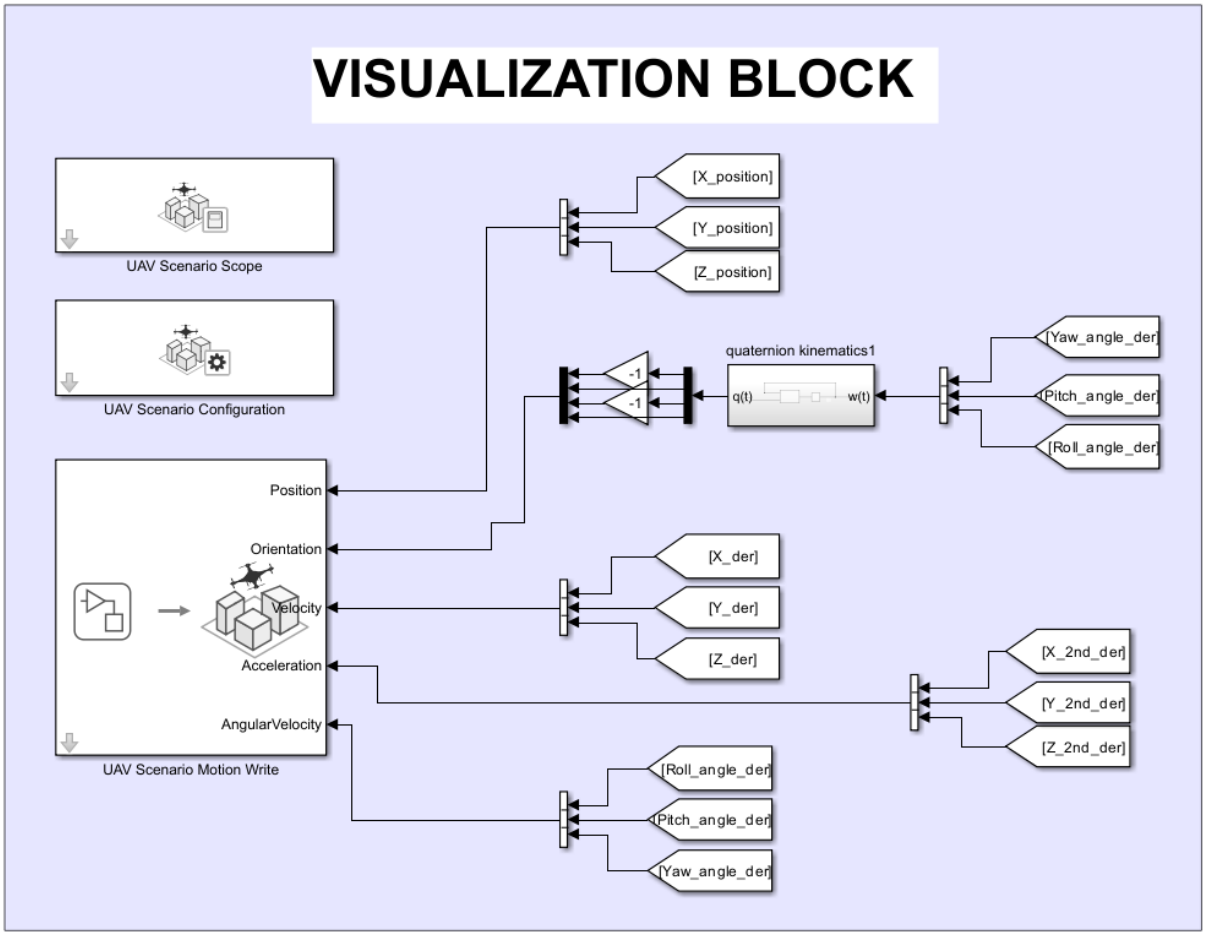
APPENDIX.3 FIGURE 5: 2nd Roll derivative subblock



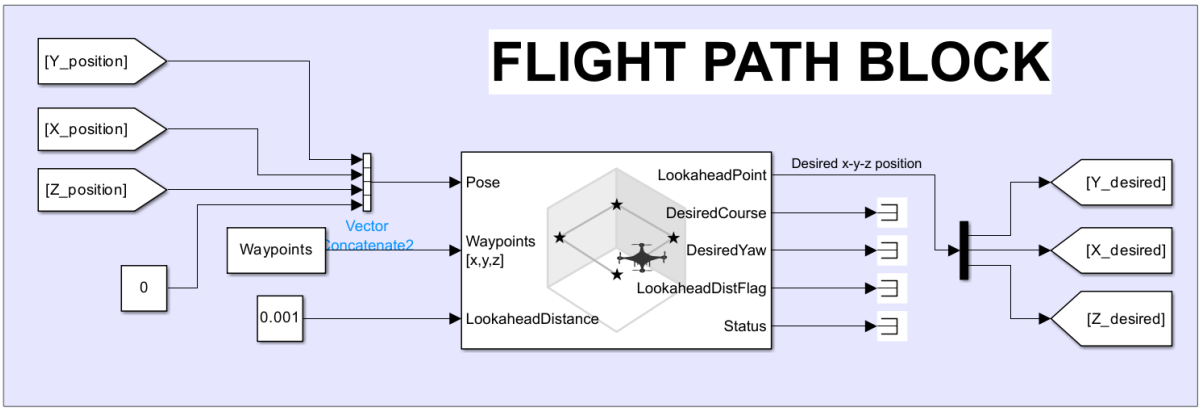
APPENDIX.3 FIGURE 6: 2nd Pitch derivative subblock



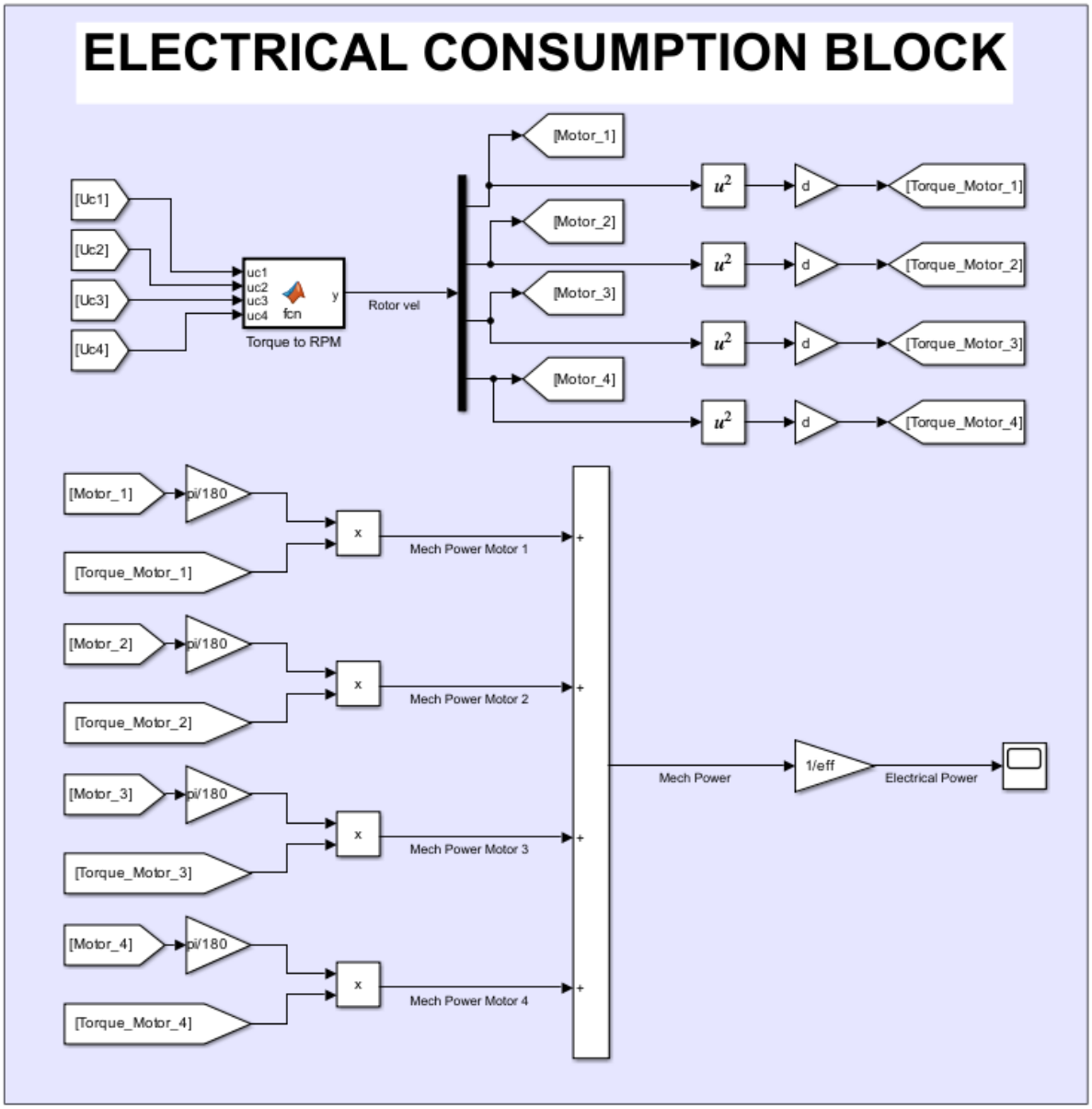
APPENDIX.3 FIGURE 7: 2nd Yaw derivative subblock



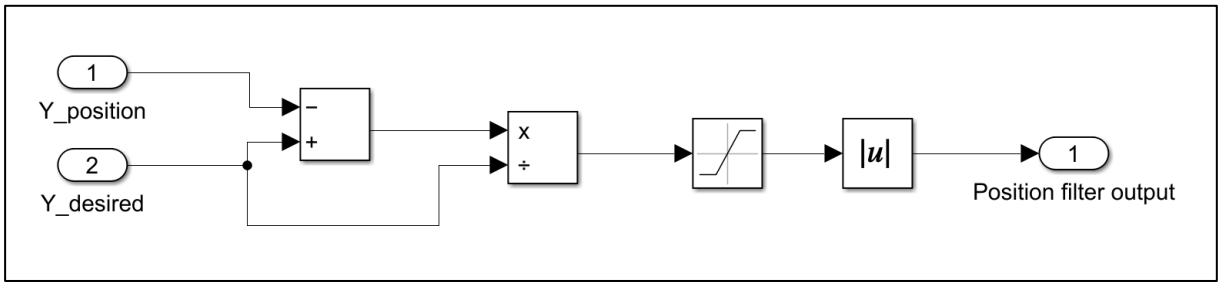
APPENDIX.3 FIGURE 8: Visualization block



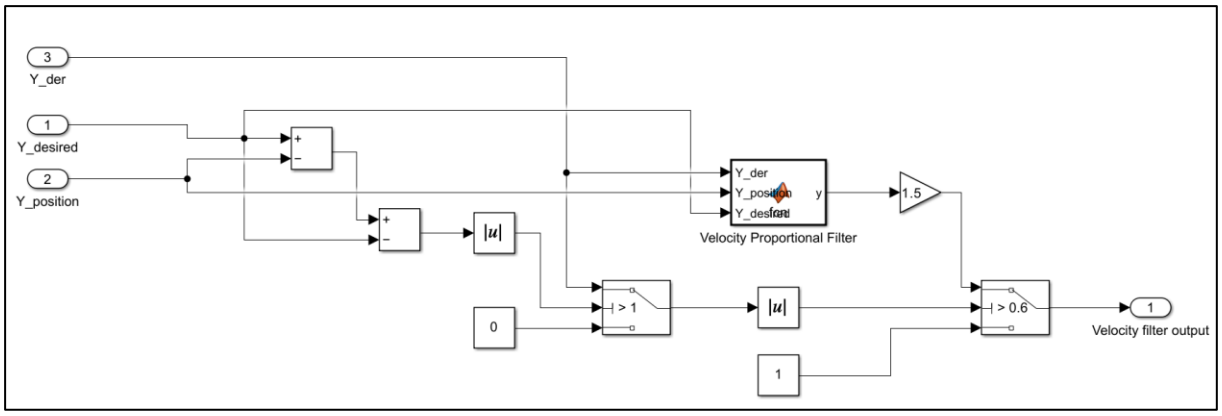
APPENDIX.3 FIGURE 9: Flight path block



APPENDIX.3 FIGURE 10: Energy consumption block



APPENDIX.3 FIGURE 11: PID position filter subblock



APPENDIX.3 FIGURE 12: PID velocity filter subblock